SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I7P109

Volume 11 Issue 7, 113-123, July 2024

Original Article

Design and FPGA Implantation of SoC for Multimedia Applications Using Deep Learning Based RNN Architecture for Low Power and High Throughput

B.N. Mohankumar[1], M.S. Kusuma[2], H.V. Pallavi[3], U. Rajashekhar[4], Neelappa[5]

[1]*Department of ECE, R.R. Institute of Technology, VTU, Karnataka, India.*
[2]*Department of ECE, Govt. S.K.S.J. Technological Institute, Karnataka, India.*
[3,5]*Department of E&C, Government Engineering College, Hassan, Karnataka, India.*
[4]*Department of ECE, Government Engineering College, Huvinahadagali, Karnataka, India.*

[1]*Corresponding Author : mohankumar4bn@gmail.com*

Received: 04 May 2024 Revised: 07 June 2024 Accepted: 05 July 2024 Published: 26 July 2024

Abstract *- Machine Learning (ML) techniques have become pivotal in the realms of customized integrated circuits are essential components in modern electronics, with two prominent types being ASICs and FPGAs, particularly in applications such as driverless vehicles, automotive electronics and big data analysis, where speed, power efficiency, and accuracy are paramount. This paper acknowledges a novel reconstructed hardware architecture at the SOC system level, including M33 processor and security to transfer steaming packets between processor and peripherals to meet the demands of modern ASIC and FPGA designs, offering high accuracy, low power consumption and increased throughput. The proposed system integrates an ML-based Support Vector Machine (SVM) with a fast-moving Advanced High-performance Bus (AHB) protocol, Floating Point (FP) operations, and support for I2C and I2S communication protocols. To enhance throughput and minimize latency, an AHB protocol and AHB to Advanced Peripheral Bus (APB) bridge have been implemented along with security algorithms, including SHA-256 and AES that are integrated into the vigorously reconstructed multi-processor. For Deep Learning (DL) - Recurrent Neural Networks (RNNs) based utilities, the system incorporates "Double-Precision Floating-Point (DPFP)" arithmetic operations. Design is implemented in Verilog HDL, which undergoes quality checks using the LINT tool and "Clock Domain Crossing (CDC)" analysis using Spyglass. Synthesis is carried out using a DC compiler for ASIC and Vivado Design Suite 2018 for FPGA execution and examination. The design architecture is connected with the SDK tool and Zynq processor to analyse data transmitted between software along hardware. Experimental outputs demonstrate that the custom accelerator can efficiently compute difficult ML classifiers for large datasets. Compared to state-of-the-art results, the proposed architecture offers a 24% improvement in outturn, a 27% scaling down in power consumption, and a 32% decrease in latency.*

Keywords *- AHB, APB, SVM, I2C, I2S, RNN, DPFP, SoC.*

1. Introduction

CNNs are widely used in image and video processing tasks due to their ability to capture spatial hierarchies in data. They are efficient in terms of both computation and memory usage, making them suitable for deployment on resource-constrained platforms like SoCs. RNNs are used for sequential data processing tasks such as speech recognition, natural language processing, and time-series analysis. They have applications in SoCs for tasks like audio and sensor data processing. Neuromorphic computing architectures mimic the structure and function of the human brain, offering potential advantages in terms of energy efficiency and real-time processing. These architectures are being explored for SoC implementations of DL algorithms. When selecting a DL algorithm for SoC applications in VLSI, it is essential to consider factors such as computational efficiency, memory usage, power consumption, and the specific requirements of the target application. Additionally, algorithmic optimizations and hardware accelerators can be combined to achieve the desired balance between performance and resource utilization [1].

On-chip communication networks can greatly benefit from executing Multi-processor with "Network-on-Chip (NoC)" routing. This study proposes a method that demonstrates significant reductions in energy consumption by minimizing input/output buffers, all while maintaining performance levels comparable to algorithms employing buffered routing methods. This approach is particularly suitable for real-world applications with low traffic volumes.

Reducing the complexity of buffer allocation and management techniques in multi-processors with NoC routing can enhance router performance and decrease router latency. This method simplifies router design, making it an attractive option for interconnection networks aimed at consistently achieving reduced peak throughput. However, it is essential to note that reduced buffer network designs lack several functionalities found in buffered networks, such as "Quality of Service (QoS)", support for different traffic classes, and management of energy, congestion, and faults [2].

This study focuses on integrating support for buffer-less routing algorithms into the design. Currently, Artificial Intelligence (AI) is playing an increasingly significant role in various applications, with Machine Learning (ML) being a key sub-system. These technologies are particularly effective in handling big data sets, using predictive systems to identify patterns and make decisions based on data analysis. Analogized to systems using all-around CPUs or GPUs for big data analysis, reconfigurable hardware offers significant speed advantages. Reconfigurable hardware allows for adapting hardware architecture to specific attributes of real-world problems, resulting in performance levels that surpass software-based solutions while holding an increased degree of pliability compared to traditional hardware implementations [3].

"Field-Programmable Gate Arrays (FPGAs)" stand out as a leading type of reconfigurable hardware device due to their adaptability, high-speed hardware timing, reliability, and parallel processing capabilities. These integrated circuits, known as FPGAs, contain programmable logic blocks that offer a range of benefits. Some FPGA families even allow for the reconfiguration of part of the chip while other portions remain operational, a feature widely utilized in accelerated computing. Their programmable nature makes FPGAs a versatile solution across various markets, particularly in the field of human conduct recognition. One key lead of FPGAs is their ability to respond quickly and save bandwidth by completing computing tasks promptly.

Ensemble methods are often used to build a group of classifiers, which enhance the classification of new data by considering the predictions of individual classifiers. This approach is known to outperform single classifiers and is effective in combining less correct classifiers to create highly errorless ones.

Human task recognition, a growing research area, involves detecting current activities using sensor data from accelerometers or wearable sensors. This technology is cost-effective, energy-efficient, and widely accessible, thanks to advancements in sensors and machine learning. Applications of human activity recognition include healthcare, eldercare, safe driving behaviour recognition, and military activity detection [4].

The need for accurate classification of human behaviour, along with reduced execution time and power consumption, is crucial in these application domains. Designing feature extraction and learning algorithms carefully is essential for achieving suitable reaction times. However, the complexity of executing these algorithms, along with the need for receivable differentiation rates, poses a challenging device problem in embedded systems. In the context of Network-on-Chip (NoC) architecture, the performance and efficiency of routers play a critical role.

This study proposes the utility of "Reduced Deflection Chipper (ReDC)" for error less deflection routers, employing double Programmable Data Networks (PDNs) with identical inputs but different initialization. This design aims to maximize output port efficiency, resulting in reduced flit deflection rates and improved overall performance compared to state-of-the-art buffer-less deflection routers. To address complexity issues and improve classification speed while maintaining low power consumption, it proposed to execute and validate a hardware design [5]. This design will leverage an FPGA-based, run-time reconfigurable architecture to ease the vigorous deployment of multiple accelerator cores [6]. The pivotal objectives of this paper shall divided into three distinct goals:

- Develop an FPGA-based architecture capable of supporting dynamically reconfigurable accelerator cores.
- Evaluate the scalability of the FPGA-based architecture to accommodate a large number of vigorously reconstructible accelerator cores.
- Facilitates hardware reconfigurability with parallelism of an FPGA to enhance distinguished speed performance analysed to traditional classifiers executing on usual-purpose CPUs.

"Support Vector Machines (SVM)"are used in this context to quest for hyperplane, which effectively classifies data points in N-dimensional space, where N stands for a number of features. Data points situated on opposite sides of the hyperplane shall be classified into distinct categories by utilizing judgemental boundaries. Support vectors are data points situated nearer to the hyperplane, impact position and orientation of the hyperplane.

The target is to find a plane that increases the boundary, which is the distance between data points of different classes. This enhances the accuracy of classifying future data points. Classifiers evaluated are implemented based on prior published work. The KNN method was tested with various neighbor counts, with the best outcome observed.

When the number of neighbors was set to 18, with equal weight given to each neighbor, a random forest ensemble approach was also employed in addition to the "Decision Tree Classifier (DTC)". The random forest consisted of 70 DTCs,

with the class receiving the most votes becoming the prediction of the model. Notably, each tree in a random forest predicted a different class [7]. The Keras library was used to construct a Multilayer Perceptron (MLP) with three hidden layers. The Convolutional Neural Network (CNN) designed for a Network-on-Chip (NoC) application consists of three sets of maximum pooling layers, three convolutional layers, and a fully connected layer preceding the output layer.

According to a previous study, Moore's Law has largely slowed down. While this may be a topic of debate, it is undeniable that all technologies inevitably reach a point of saturation. The growth in CPU performance and efficiency follows Moore's Law. To further enhance computer hardware performance and efficiency, alternative architectures like domain-specific hardware accelerators have emerged.

These accelerators utilize data specialization, parallelism, optimized local memory, and reduced overhead to improve performance and efficiency. By applying specialized reasoning to domain-specific data, performance and efficiency can be significantly enhanced, particularly in high-parallelism scenarios [10].

2. Literature Survey

With the rapid advancements in cloud computing and the Internet of Things (IoT), there is a growing need for high-performance and power-efficient designs to handle enormous amounts of data. Recent developments have focused on the fundamentals, but there is a performance gap in processing speed and the volume of data. Multi-processors with Network on Chip (NoC) have been proposed to address these challenges, utilizing packet-switched fabric for on-chip communication. This approach is crucial for various applications requiring energy-efficient systems and significant congestion reduction [11].

In this proposed work, a hybrid MP-NoC framework is designed, combining bufferless and buffered NoCs. This framework is tailored to specific applications and their performance demands, using trace-driven simulators to generate data similar to big data applications. Compared to traditional buffered Multi-processors with NoC, the proposed hybrid approach shows a substantial performance enhancement of up to seventeen percent on average and twenty-four percent at most in mixed applications. It also improves fairness by over thirteen percent and reduces power consumption by thirty-eight percent [12].

The design also addresses power and area issues in on-chip networks, using a MaS-based buffer-less Multi-processor with NoC for delivery without huge buffering essentials. This approach, compared to BLESS-Worm, reduces buffering needs at the receiver end by up to eighty percent. Simulation results demonstrate that MaS outperforms BLESS-Worm in power consumption reduction by nine percent, average packet latency by ten percent, and requiring fewer buffer resources [13].

To improve on-chip communication, utility photonic waveguides are suggested, leveraging nanophotonic technology to combine high-radix MP-NOC waveguides for higher output. The proposed Bufferless Photonic ClOs Network (BLOCON) maximizes the composition of silicon photonics. Additionally, fault tolerance mechanisms like FTDR-H and FTDR routers are proposed, reducing chip area consumption by twenty-seven percent in an 8x8 network and achieving higher throughput compared to existing algorithms under synthetic workload conditions [14].

In conclusion, the proposed approaches offer significant advancements in on-chip communication, power efficiency, and fault tolerance for multi-processor systems with NoCs, addressing key challenges in handling big data and improving overall system performance. More complex DL models usually require large amounts of memory and processing power, especially when working with State-of-the-Art (SOTA) models. As a result, many apps that use these models turn to cloud computing services, including Google Colaboratory, which is a free service [15]. AWS DL AMIs and Microsoft's MLOps service are also well-liked substitutes that expedite the training, deployment, and management of DL projects on Azure-like platforms. Nevertheless, using a cloud computing solution has several significant disadvantages. Many computational processes are managed in the cloud under the standard cloud computing paradigm, which may cause network congestion and unacceptably long delays in real-time situations [16].

Edge computing-related DL applications are common in many areas, such as smart multimedia, smart transportation, smart cities, and smart industries. When creating a DL architecture for a particular activity, it is important to take into account the limitations and restrictions that come with moving from cloud computing to edge computing [17]. These restrictions result from the intrinsic computational limitations of edge computing, especially when using low-power embedded devices with limited resources.

Two popular strategies are usually used to tackle this problem:

1) Using various compression techniques on pre-existing DL models, and
2) Optimizing the architecture right from the design phase.

In the first method, one or more compression methods must be used in order to execute a DL model on embedded devices. Examples of frequently utilized compression [18]. The methods include binarization, network distillation, neural network pruning, and quantization of model parameters [19].

The second strategy, on the other hand, aims to create an optimal architecture that eliminates the need for compression techniques after training. Notably, designs such as SqueezeNet, which achieve a limited number of parameters with negligible influence on accuracy, have made noteworthy contributions in this direction [20]. It is important to note that inference tasks which require far less processing power than training are the main applications for embedded devices. General-purpose microcontrollers are among these embedded devices, and they have been shown to be effective in a variety of industries, including Internet of Things applications [21].

Over the past ten years, the industry has seen a tremendous influx of specialized hardware devices that have made dealing with computational limits easier. In the context of DL, these hardware components also known as hardware accelerators consist of specialized and optimized hardware architectures designed to minimize system costs and power consumption while optimizing performance. Tensor Processing Units (TPUs) from Google, FPGAs, GPUs, and ASICs are a few examples of this type of hardware accelerator [22].

DL applications demand embedded systems to have a high processing capability to accomplish tasks in real-time and enough memory to store model data and parameters [23]. SoC devices are a promising option that combines a multitude of peripheral devices that are required for the complex requirements of DL applications with powerful computing capabilities. Leading producers such as Texas Instruments, Qualcomm, STM, and Samsung are creating AI-integrated circuits for edge computing [24].

3. Materials and Methods Proposed Sub-System Level Re-Configurable Architecture with RNN and Security Algorithms

The outright suggested design, as illustrated in Figure 1, features a Zynq software core processor coupled with a co-processor serving as a required accelerator. To facilitate constant observation and connection with the processor's stream, an instruction cache is used to turn Zynq on/off. In the suggested architecture, the cache is internally located within the Zynq, making direct connection access impossible. As a result, the processor fetches data directly from cache memory through instruction, rendering the relocation technique non-applicable.

To address this, data is stored in external memory, allowing the Zynq processor to approach and disable it as needed. Additionally, the customized accelerator in this architecture can access external cache memory instead of the internally placed BRAM memory. To enhance accessibility between the accelerator and DDR memory, a high-speed, small-sized cache IP is added, establishing direct interfaces between them. This setup enables efficient data access for specific addresses between DDR and the accelerator. The architecture also features an extra AHB, incorporating a counter and timer, which interfaces with the Zynq and peripherals to facilitate data transfer.

The AHB connects to the host interface for direct data communication, providing benefits like burst length support, power efficiency, low latency, system-level cache access, and support for unaligned byte strobes and addresses. During initialization, the boot image code is stored in SRAM with the use of software and DMA. After completing the SHA execution, the software retrieves the SHA HASH output via the APB interface and saves the resulting SHA hash value public key into memory.

Encryption keys, which are stored in plaintext as a "One-Time Programmable (OTP)" feature, are managed by hardware. These OTP keys undergo encryption using AES, and the encrypted keys are then reserved in flash memory. When needed, reserved keys are fetched from flash memory using the APB protocol and made available to AES and SHA for key exchange and decryption purposes.

The article is organized as shown in Figure 1, beginning with an examination of the goals and difficulties associated with putting RNN models into practice on embedded systems. After that, an extensive model for RNN applications is outlined, and several changes for the recurrent layers in RNN models are discussed. Nevertheless, there are many obstacles in the way of effectively executing RNN models in their original form on embedded devices.

Thus, scientists have turned to optimizations that aim to improve both the underlying platform and the RNN model. Along with platform-specific optimizations targeted at improving the hardware platform's performance, the optimizations done to the RNN model, referred to as algorithmic optimizations, are analysed and discussed. After that, a review of RNN hardware implementations suggested in the literature is given. The performances attained by these implementations, together with the implemented optimizations, are examined closely.

3.1. RNN for SoC's Applications

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to handle sequential data. Unlike feedforward neural networks, which process each input independently, RNNs maintain a hidden state that captures information about previous inputs in the sequence. This recurrent nature enables RNNs to exhibit dynamic temporal behavior and process sequences of varying lengths.

The defining characteristic of RNNs is the presence of recurrent connections that allow information to persist over time. At each time step 't', the network receives an input x_t and produces an output y_t while also updating its hidden state

h_t. The hidden state h_t serves as a memory that encapsulates information from previous time steps and influences the current output. The hidden state h_t at time step t is computed based on the current input x_t and the previous hidden state h_t-1, along with model parameters (weights and biases). This update equation can be formulated in Equation (1) as,

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_n) \quad (1)$$

Where,

- W_{hx} and W_{hh} are weight matrices governing the transformations of the input and hidden state, respectively.
- b_h is a bias vector.
- f is an activation function, commonly a non-linear function like the hyperbolic tangent (tanh $f(x)$ tanh) or Rectified Linear Unit (ReLU).

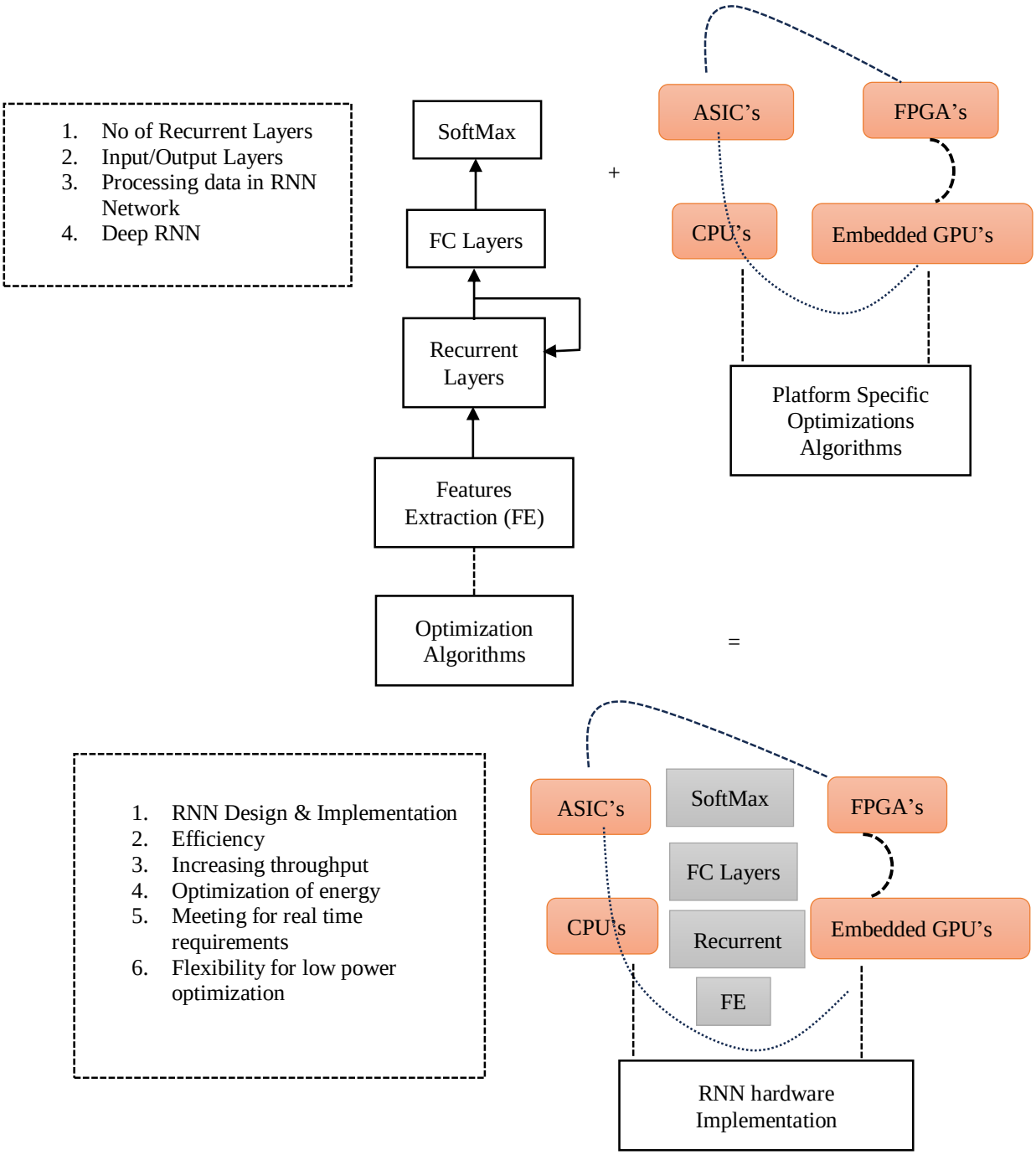

Fig. 1 Proposed top-level architecture of RNN models and SoC for ASIC & FPGA

The output y_t at time step t is typically computed based on the current hidden state h_t and model parameters. This can be expressed in Equation (2) as,

$$y_t = g(W_{yh}h_t + b_y) \quad (2)$$

Where,

- W_{yh} is a weight matrix connecting the hidden state to the output.
- The b_y is a bias vector.
- g is an activation function, which may vary depending on the task (e.g., softmax for classification, linear activation for regression).

RNNs are trained using backpropagation through time, an extension of the backpropagation algorithm adapted for sequences. BPTT calculates gradients with respect to the model parameters by unfolding the network through time and applying the chain rule recursively. RNNs are versatile and can be applied to various sequential data tasks, including tasks such as language modeling, sentiment analysis, machine translation, and named entity recognition benefit from RNNs' ability to model dependencies in text data.

RNNs are effective in modeling and predicting temporal patterns in time-series data, making them suitable for applications like weather forecasting, financial market analysis, and signal processing. Integrating a Recurrent Neural Network (RNN) into a System on Chip (SoC) using Verilog, a hardware description language, involves converting the computational logic of the RNN into hardware modules that can be synthesized and implemented on an FPGA or ASIC within the SoC. Here is an overview of how an RNN can be implemented in Verilog for SoC applications:

The basic building block of an RNN is the recurrent cell, which computes the hidden state update at each time step. In Verilog, you would design modules to represent these cells. Each cell module would include input ports for the current input x_t, the previous hidden state h_t-1, and output ports for the updated hidden state h_t and the current output y_t. The computation within the cell, including the activation functions and weight matrices, would be implemented using Verilog logic and arithmetic operators.

The weights and biases of the RNN are essential parameters that need to be stored in the SoC's memory or registers. In Verilog, you would define registers or memory blocks to hold these parameters, which would be accessible by the RNN cell modules during computation. Verilog inherently supports sequential execution, making it suitable for modeling the iterative nature of RNNs.

The proposed RNN-based DL would use Verilog constructs like always blocks to define the sequence of operations within each time step of the RNN. The always blocks would trigger the computation of the next hidden state based on the current input and the previous hidden state. While Verilog is primarily used for hardware description rather than training neural networks, you could potentially implement hardware accelerators or co-processors for Backpropagation Through Time (BPTT) within the SoC.

These hardware modules would compute the gradients and update the weights of the RNN based on training data. The Verilog code implementing the RNN would need to be integrated into the overall SoC architecture, which includes other components such as processors, memory interfaces, and peripheral devices. This integration involves connecting the RNN modules to the SoC's data and control buses, configuring memory interfaces to access parameter storage, and coordinating the execution of RNN computations with other tasks running on the SoC.

As with any hardware design, verification and testing are crucial steps in the development process. You would simulate the Verilog code using tools like ModelSim to ensure correct functionality and perform hardware-in-the-loop testing on FPGA prototypes to validate the RNN's performance in real-world scenarios. Overall, implementing an RNN in Verilog for SoC applications requires a solid understanding of both neural network theory and digital hardware design principles. It involves translating the mathematical operations of the RNN into hardware logic while considering factors such as resource utilization, timing constraints, and integration with the broader SoC architecture.

4. Functionality of Proposed RNN and other Peripherals Interface at Sysetm Level

The proposed dynamically reconfigurable multi-processor core enhances performance by efficiently handling data and address generation. It employs an AHB to APB bridge to connect with moderate peripherals I2S and I2C. This bridge is essential for efficiently transferring audio signals amongst processors and various audio processing algorithms, ensuring minimal latency. The PAB bridge optimizes clock cycles throughout 'setup' and 'access' circumstances in APB, aligning the speed of audio and connection elements to reduce data/packet losses. It achieves this by storing 32-bit packets in a "Transmit FIFO (Tx-FIFO)" with a minimum depth of 256, enabling later retrieval by serial peripherals.

This microprocessor, based on the Cortex-based ARM processor, integrates NoC routers and Security systems are utilized to connect with all peripherals. Every peripheral connects to the processor through AXI interconnects and bridges, which enhance output and reduce latency. The Master in serial protocols adheres to protocol standards, converting data into serial bits transferred to GPIO via FPGA PMOD connectors for interfacing with outer devices. It transmits serial bits alongside a serial clock, which the slave utilizes as an input clock to synchronize serial data bits. The slave

receives these bits, including start and stops bits, converts them into parallel data, and stores them in the Receiver FIFO (RX-FIFO) for access by the APB bridge as needed. The proposed top-level design of an ML-based SVM accelerator and audio signal transfer from processor to peripheral and its simulated results are illustrated in Figure 2, incorporates weights buffers and input/output buffers to buffer data for subsequent processing efficiently. To reduce off-chip memory traffic, a professional 3D MP-NOC redistributes output packets via a multi-banked input buffer rather than sending them to destination nodes in external memory. Operations of 3D-Multi-processor and SVM with NoC are carried out independently using various Processing Elements (PEs).

The control module generates overall control signals for the different modules, which are responsible for transmitting and controlling streaming data and delivering it to the weight buffers and multi-banked input for each PE. The I2C Slave controller core serves as a bridge between an I2C master device and a microprocessor. It features a customizable FIFO depth and a measured FIFO. Its standard APB interface allows seamless integration into any peripheral sub-system or SOC. Additionally, it provides several status bit flags to Simplify Software (SW) and IP bring-up processes.

These flags include FIFO Overrun/Underrun, Invalid register access to configuration/status register space, receiving a 19.2 MHz standard io_clock clock input, and a serial_clock (scl) of maximum 5MHz from the I2C master. The I2C slave supports 7/10-bit addressing and Clock Stretching only in an open-drain configuration. The I2C slave acts as a bridge between the microprocessor and the master device, receiving data from the microprocessor to be sent to the I2C master and vice versa.

5. Results and Discussion

The primary communication technique employed in this System on Chip (SoC) is the 3D-Multi-Processor with NoC, which facilitates all on-chip communications. The design aims to investigate and execute a prototype of asynchronous MP-NoCs on Field-Programmable Gate Arrays (FPGAs). The challenging task involves implementing a system of asynchronous MP-NoC on customary FPGAs, which necessitates the development of a readjusting portion in FPGAs to create design flow. This design represents a comprehensive and victorious way to start MP-NoC for an FPGA. During implementation, a 4-phased bundled data handshake protocol is utilized.

The design of MP-NoC includes two main components: network adapters along with router. The Open Core Protocol (OCP) interface joins the cores of the network, while a mesh topology is employed to create network connections in a small MP processor, which validates the MP-NoC through experimental results. Network on Chip (NoC) technology is gaining traction worldwide for communicating between SoCs.

Table 1. Comparison between proposed and existing in terms of slices and DSP of RNN

S. No.	Structure Details	No. of Slice LUT	No. of Slice Registers	No. of DSP Blocks
01	RNN [8]	1890	1899	16
02	RNN [10]	2100	2150	23
03	Proposed RNN	1350	1280	12

Furthermore, the replacement of the traditional RNN architecture with the proposed RNN not only enhances its performance but also optimizes hardware resources, power consumption, and cost. In the proposed design, a significant reduction of 50% in both area and power consumption has been attained compared to other existing systems, as shown in Table 1. Moreover, the proposed structure can be further enhanced by integrating hybrid adders along with shared LUTs, which find wide applications in cognitive radio networks and other domains.

This research paper is centered on the modeling of synchronous Multi-Processors with NoCs on Field-Programmable Gate Arrays. The main objective is to plan a synchronous circuit and later deploy it on standard FPGAs. The process includes creating a trial flow, conducting tests, and assessing its interpretation of FPGAs. A high-effort MP-NoC is developed to fulfill the specified requirements and is subsequently implemented on FPGAs. The use of an MP-NoC involves circuit connectors, switches, UART, and memory, which facilitate the primary handshake of MP-NoC in a 4-stage system.OCP Interface stands out as the best widely used protocol in this setup. To showcase the current plan in real-time, a functional topological model is necessary. Planned MP-NoC is validated on a minimum, versatile processor model. 3x3 MP-NoC, including its UART protocol, is architecture, and its Verilog HDL code replication is executed and verified on the Artix-7 FPGA kit using the Chipscope software tool.

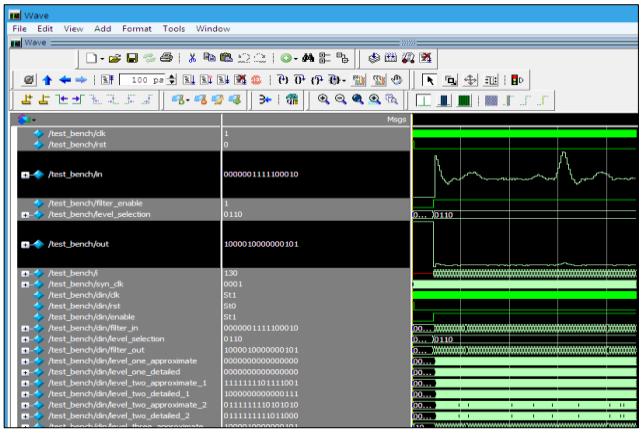

Fig. 2 Simulated results for audio samples transmission from processor to peripheral

The audio signal for the proposed System-on-Chip (SoC) with RNN applications using the offered technique, aiming to evaluate the effectiveness and utility of an extended top-level design of the SoC. The evaluation involves employing high-precision control signals with enhanced frequency response and precise filtering operations in the filtration process. To showcase transmit of data from an origin to a target, we employ an Integrated Controller (ICON) and Virtual Input/Output (VIO).

The efficiency of this data transmission is evaluated through metrics such as latency, Packet Delivery Ratio (PDR) and hardware resource utilization, including slices, Look-Up Tables (LUTs), flip flops, and area. Packet generation is achieved using a Traffic Pattern Generator. The outputs indicate significant improvements, including a 23% enhancement in LUTs, 14% in flip-flops, a 31% increase in throughput, and a 29% reduction in delay.

5.1. Traffic Generator Design MP with MP-DL-SoC

A dedicated traffic generator was developed and deployed to evaluate functionality, featuring sink signals and traffic supply. The traffic supply sends pre-defined data packets to fixed data storage, while the sink signal receives traffic data and can manage the correct input and output pairing. In a multi-processor with a NoC design, data packets traverse the network according to the diagram in Figure 3. Every entry in ROM data includes data information alongside a handshake signal that activates the simplest router using a single NOR gate.

A counter designed for asynchronous operation is clocked with an acknowledgement signal and features a fixed address with increased high values within the existing data size. The fixed data memory is triggered by the request signal without delay, ensuring proper low-level formatting of the fixed storage output by gating clock input with a reset signal. A demultiplexer is used for the output channel, with flit-type control signals, enabling the sink signal of traffic to obtain stored data packets for execution.

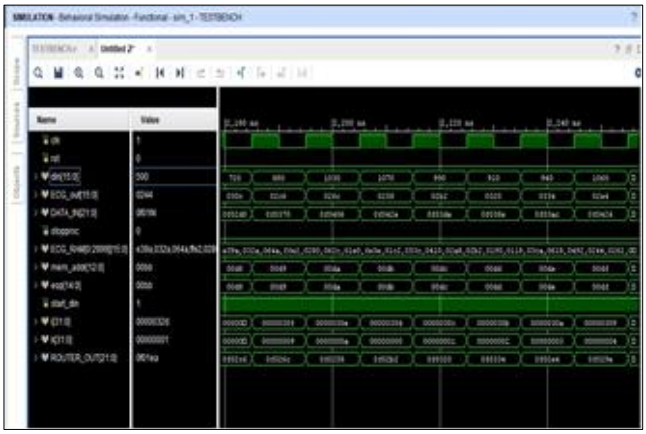

Fig. 3 Random data transfer from source to destination simulated results

In FPGA-based processing, incoming data packets are first stored in ROM. To extract and process this data, the JTAG cable interface is utilized. ChipScope Pro, a tool that interfaces via JTAG, is employed for verification. Its GUI-based, user-friendly interface simplifies debugging and design verification by allowing easy connection to any port for monitoring and analysis. The ChipScope tool is utilised to process and view data captured by cores. In sink design, the VIO ChipScope core collects data. Data signals are captured by the core on the falling edge of the request signal. When the internal storage of the VIO core is full, data is sent to the ChipScope program. Each signal requiring monitoring in the plan needs a VIO core. These cores track and archive data traces for any signal in FPGA during execution.

The design process for Chip Scope VIO core shall be approached in both ways, which involves gathering information from the programming environment. The first approach follows a conventional method, programming the core along a mechanically synthesized netlist. The second method involves programming the core using the typical Verilog HDL technique. However, the Verilog HDL technique may not be suitable for the suggested plan, as it is a finalised programming type and does not allow for dynamic modifications. Therefore, the first technique is preferable for gathering knowledge for vigorous programming.

In this approach, knowledge shall collected from the synthesizer using the existing handshake data path. It is important to note that the handshake signal is not a portion of the data traffic channel and, thus, does not impact the netlist synthesizer. In the suggested design of a multiprocessor with a NoC router, the VIO core and ICON control are integrated. ICON core oversees peripheral communication between JTAG, Chipscope software, and optimization planning tools, streamlining the design process. A clock signal synchronized with the falling edge of any signal is utilized with an appeal signal in the VIO core. The requested signal is channelled with a netlist of corresponding clock signals, as depicted in Figure 4.

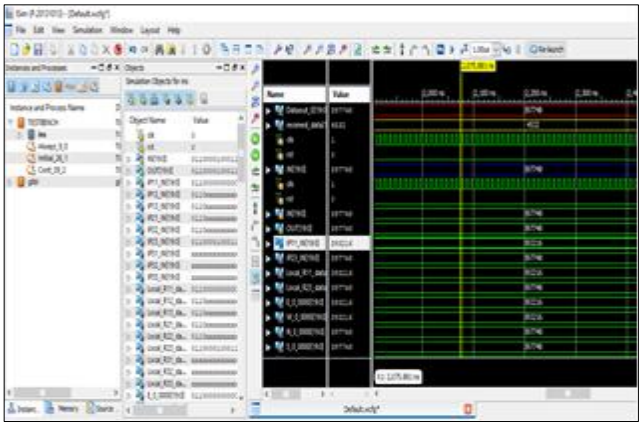

Fig. 4 FPGA validation of proposed work using Chipscope pro

The architectured MP-NOCs router is categorised into different Verilog entities, among 9 entities, following a structured approach with four different directions encoded in the header as “00”, “01”, “10”, and “11” for north, east, south, and west, respectively. An area chart illustrates various parameters, listed in Table.1, highlighting 590 latches and 2378 LUTs used to achieve a countering delay of less than 29%. To modify the data content of the ROM, different HDL data resources are used, achieved through Xilinx and Core generator from HDL-based techniques. ROM programming is accomplished using an accessible route lookup table in master NA. The primary target of the current execution is to illustrate MP-NoC in asynchronous mode on a customized FPGA.

In comparison to existing work on FPGA executed with asynchronous systems, typically operating in synchronous mode, this project is implemented with a general design flow for asynchronous FPGA executions. Designed Multi-processor with NoC operates in asynchronous mode, providing best-effort service. The system comprises two Network-on-Chip (NoC) Nodes executing in a master-slave configuration with the router. The routing mechanism is tailored to meet specific topology and routing needs, utilizing mesh routing and wormhole techniques. Deadlock resolution in multi-processor with NoC is achieved through a combination of two-dimensional XY routing and supply routing strategies.

In the intended design of a multiprocessor with a Network-on-Chip (NoC) operating in packet-based switching for data transfer, packets are uniquely identified using an infinite number to indicate the beginning with end of data packets. Additionally, data packets are encoded with extra handshake signals to facilitate their identification. The OCP interface is now part of the NAS, acting as the central hub for network connectivity. Synchronization tasks are managed through the bi-IP-op synchronizer. Operating at a frequency of 594.229MHz, the router uses 9% of the 3201 available LUTs and 542 latches for its components, potentially causing delays. The miniature multi-processor prototypes designed in asynchronous Multi-processor with NoC consist of 3 different CPUs with identical peripheral units arranged in a 3x2 mesh configuration. To prevent deadlocks depending on data messages, a dedicated way for request and response modes is implemented, ensuring infinite deadlock evasion.

While the approach may not completely address higher strategic delays of FPGA deadlocks, it efficiently utilizes FPGA’s logical resources. The goal is to utilize the existing FPGA resources effectively to avoid deadlocks successfully. Implementing asynchronous MP-NOCs to solve every issue, including delay matching criteria, is challenging due to FPGA prototyping techniques and a lack of supportive tools. To minimize delay in FPGA circuits, a different data way for data transmission is used.

This approach achieves relatively low detain to meet the criteria of the MP-NoC, measuring down data packet path with macros and design primitives. Despite challenges with deadlock avoidance tools, the design optimizes operations in real-time. For the complete design of Mp-NoC circuits, alternative LUT mapping should be considered to address all design issues effectively. Channel routing algorithms are utilized to determine the optimal path among interconnected sub-modules within an architectural framework, considering their spatial arrangement. Research has demonstrated that such algorithms significantly enhance Network-on-Chip (NoC) efficiency and stability, particularly in mitigating network congestion.

6. Conclusion

The machine learning-based Support Vector Machine (SVM) represents the cutting edge in classification accuracy for packets received from multiple processors to various peripherals. It achieves this high accuracy while minimizing computational complexity. Using MP and accelerator to implement the ML-based SVM, we planned and evolved a system through greater-level synthesis in Vivado Design Suite 2018. We evaluated its efficiency based on throughput, latency and power consumption, comparing it with the current state-of-the-art.

Our results demonstrate that the HLS-based implementation is superior in performance, being 23.4 times faster than a General Purpose Processor (GPP)-based design and 12.6 times faster than a Graphics Processing Unit (GPU). The suggested multiprocessor, integrated with various high-speed protocols in a block-level hardware configuration containing AXI interconnects and a Zynq processor was targeted to the Zynq-7000 evolvement FPGA board. It was then connected with the SDK environment, and the entire design and functionality were examined and verified through software programming.

The outcomes from our design show that the standard loop accelerator generated could efficiently compute complex machine learning classifiers with highly huge amounts of data. The implemented plan is capable of measuring the architecture of other PEs that is part of the 3D-NoC.

Our designed hybrid MP-NoC, depending on application-aware design for big data loads and transmission, is highly beneficial in enhancing energy efficiency and service quality across magnificent heterogeneous application loads. To aid in the selection of an efficient MP-NoC, we present a hybrid MP-NoC consisting of a specific MP-NoC, a buffered MP-NoC, and an application-aware technique in this research work.

This procedure significantly increases system efficiency. Additionally, we built a unique hybrid MP-NoC congestion optimization approach. By reallocating packets in congested

nodes and verifying the performance of different MP-NoCs' congestion, this approach can significantly enhance the overall system's energy efficiency. The area and power used by router buffers in NoC are major concerns in the submicron domain. The performance of synthetic traffic situations can be assessed utilising a flit-level, cycle-accurate network simulator. Our computational outputs illustrate that the designed routing algorithm maximizes power consumption by 19%, average latency by 24%, and area overhead by 47% compared to other conventional algorithms.

References

[1] Rakesh Pandey, and Aryabartta Sahu, "Performance and Area Trade-Off of 3D-Stacked DRAM Based Chip Multiprocessor with Hybrid Interconnect," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1945-1959, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Makoto Nagata, Takuji Miki, and Noriyuki Miura, "Physical Attack Protection Techniques for IC Chip Level Hardware Security," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 1, pp. 5-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Dipika Deb, Rohith M.K., and John Jose, "Flit Zip: Effective Packet Compression for NoC in MultiProcessor System-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 117-128, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[4] Hayate Okuhara et al., "A Fully Integrated 5-mW, 0.8-Gbps Energy-Efficient Chip-to-Chip Data Link for Ultralow-Power IoT End-Nodes in 65-nm CMOS," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 10, pp. 1800-1811, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Tom B. Herbert et al., "An Active Bandpass Filter for LTE/WLAN Applications Using Robust Active Inductors in Gallium Nitride," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 7, pp. 2252-2256, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Biswajit Bhowmik et al., "AI Technology for NoC Performance Evaluation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 12, pp. 3483-3487, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Joshua Mack et al., "Performant, Multi-Objective Scheduling of Highly Interleaved Task Graphs on Heterogeneous System on Chip Devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2148-2162, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Kewen Zhu et al., "An RF On-Chip Transformer with Fe3O4/GO Nanocomposite Film," *IEEE Transactions on Magnetics*, vol. 57, no. 2, pp. 1-5, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] R. Spiwoks et al., "CentOS Linux for the ATLAS MUCTPI Upgrade," *IEEE Transactions on Nuclear Science*, vol. 68, no. 8, pp. 2127-2131, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Richard Fenster, and Sébastien Le Beux, "RELAX: A Reconfigurable Approximate Network-on-Chip," *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, Singapore, pp. 381-387, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Arup Ray, Arijit De, and Tarun Kanti Bhattacharyya, "A Package-Cognizant CMOS On-Chip Antenna for 2.4 GHz Free-Space and Implantable Applications," *IEEE Transactions on Antennas and Propagation*, vol. 69, no. 11, pp. 7355-7363, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Hannes Radner et al., "Field-Programmable System-on-Chip-Based Control System for Real-Time Distortion Correction in Optical Imaging," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3370-3379, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Abhijit Das, John Jose, and Prabhat Mishra, "Data Criticality in Multithreaded Applications: An Insight for Many-Core Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 9, pp. 1675-1679, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Goeun Kim et al., "Impact of System-in-Package in Side-by-Side Discrete SoC-DRAM Configurations on SI, PI and Thermal Performance," *2021 IEEE 71st Electronic Components and Technology Conference (ECTC)*, San Diego, CA, USA, pp. 1805-1811, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] Zhe Jiang et al., "Brief Industry Paper: AXI-InterconnectRT: Towards a Real-Time AXI-Interconnect for System-on-Chips," *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Nashville, TN, USA, 2021, pp. 437-440. [CrossRef] [Google Scholar] [Publisher Link]

[16] Sergey A. Chusov et al., "Configurable Test Environment for RTL Simulation and Performance Evaluation of Network on Chip as Part of SoC," *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, St. Petersburg, Moscow, Russia, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[17] C. Sapna Kumari et al., "Performance Analysis of Cloud-based Health Care Data Privacy System Using Hybrid Techniques," International Journal of Biology and Biomedical Engineering, vol. 16, pp. 46-63, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] P. Loktongbam et al., "Design of a Wide Band Antenna with Defected Ground Structure for mm-Wave System on Chip Applications," *Microsystem Technologies*, vol. 28, pp. 2487-2497, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Le-Le Li et al., "SOCA-DOM: A Mobile System-on-Chip Array System for Analyzing Big Data on the Move," *Journal of Computer Science and Technology*, vol. 37, pp. 1271-1289, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] Sujin Cho, Sumi Lee, and Song Ih Ahn, "Design and Engineering of Organ-on-a-Chip," *Biomedical Engineering Letters*, vol. 13, pp. 97-109, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[21] Amitesh Singh Rajput, and Balasubramanian Raman, "Example Based Privacy-Preserving Video Color Grading," *Handbook of Multimedia Information Security: Techniques and Applications*, pp 63-87, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Nasir N. Hurrah, Shabir A. Parah, and Javaid A. Sheikh, "A Secure Medical Image Watermarking Technique for E-Healthcare Applications," *Handbook of Multimedia Information Security: Techniques and Applications*, pp. 119-141, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[23] Cong Ma et al., "Design and Evaluation of an FPGA-ADC Prototype for the PET Detector Based on LYSO Crystals and SiPM Arrays," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 6, no. 1, pp. 33-41, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[24] Asokan Bakhirathan, Raveendar Giridhar, and Gangadhara Kiran Kumar Lachireddi, "Heat Transfer Enhancement for On-Chip Cooling Application Using Novel Composite Heat Sink-Comparative Numerical Study," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 11, no. 8, pp. 1197-1205, 2021. [CrossRef] [Google Scholar] [Publisher Link]