 doi:10.20944/preprints202204.0129.v1

DIGITAL DESIGN OF PROGRAMMABLE IMAGE PROCESSOR FOR FPGA

A PREPRINT

Abdullah Aman Khan
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
abdkhan@163.com

April 11, 2022

ABSTRACT

Many dedicated designs for real-time operations provide functionality on fixed-sized operators, but where speed, scalability, and flexibility are required, extensive research is demanded. Dedicated designs can provide real-time processing for many applications. This paper presents an FPGA-based design of a general image processor. The proposed design is based on a fixed-point representation of binary numbers. The proposed design provides a mechanism to manage matrices on-chip along with matrix arithmetic. The matrices are represented with simple identifiers and microinstruction that assist in the computation of many operations which are useful for solving complex problems. The design was successfully implemented and tested using VHDL language. The proposed design is an efficient architecture as a standalone processor with all embedding computational resources necessary for an embedded image processing application.

[1]

Keywords Digital Design · Digital Architecture · Image Processing · Machine learning · FPGA · Dedicated Design · Image Processor

1 Introduction

Over the past decades, the field of digital design has flourished as the human desire for luxury and comfort increased. The importance of man-made machines is known to everyone. Humans have started to rely on machines for decisions taken in many aspects of life. Human decision-making capabilities are limited as compared to machines in terms of speed, accuracy, and calculation. For speed and accuracy machines can be deployed to ease human life by saving time. Engineers and scientists have spent thousands of man-hours on the understanding and development of digital machines and various methods [1–24]. The mobile device sector has a great trend towards embedded systems, these devices provide all at one place functions like video/audio playback and recording, Video games, the Internet, Communication Facilities Taking Photos and so much more. To perform all these functions, a device must have a processing element that can perform in real-time while keeping the cost in view with an optimal trade-off between flexibility, performance, and cost. Computing techniques used in general processors (i.e., desktop computers) are not suitable for dedicated processors due to higher power and space requirements so alternate techniques are required. The cost of design and performance are directly proportional to each other. In some cases, the cost of the design also raises in terms of gates or transistors while providing flexibility.

The semiconductor industry has been helping to realize the digital designs presented by engineers over time. Moore's law predicted the doubling of the number of transistors used to build integrated circuits [25–27]. The semiconductor industry has been driven by this phenomenon. The future technology is moving towards the nanoscale, providing low

[1] codes and resources: https://github.com/abdkhanstd/image_processor

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

power consumption and smaller size. But the scaling factor of smart devices is still under great stress. Moving towards the atomic dimension makes scalability more complex.

The field of image processing and Machine learning is the most happening field of the era. Their applications are providing high comfort and accuracy zone in the market of electronics. The algorithms of machine learning and Image processing can be implemented on a general computer, but where speed is critical custom systems have to be developed for providing the same functionality as the general computer. Many flexible devices, such as FPGA [28], CPLD [29], ASIC [30], etc., were introduced into the market. These devices offer flexibility, and high performance followed by low power consumption. With the span of time, the trend of MPSoC [31] has increased. Modern systems are provided on-chip solutions for many well-known problems. These kinds of designs lay a tremendous burden on the designers as the requirements increase design complexity and challenges also increase. Handling matrices on the chip is a big challenge for designers, usually, designers provide a solution for a fixed size of operands, i.e., if an application-specific design is designed for $m \times n$ pixels it can process only fix the size of an image, whereas in some cases the size of the image needs to be changed.

In this paper, we present an educational purpose architecture and design of a simple processor which is capable of handling the two-dimensional matrices operations on-chip. Besides Matrix operations, the design is also capable of handling scalar operations, along with logical operations and branching. An Instruction Set Architecture is also presented which can perform numerous critical tasks related to image processing. This architecture provides flexibility. It doesn't restrict to fixed sizes of matrices. It also provides the mechanism for simple matrices handling of universal size as well as matrices size can be defined on runtime by using the provided simple instructions. Using these small microinstructions, the application of the processor can be enhanced to any extent depending on the technology available at the time.

2 fixed point representation

In real-life problems, mathematical numbers usually have a fractional part. For example, to represent a number half can be expressed by adding a fractional part after a decimal point i.e., (1/2=0.5). The fixed-point is a simple and easy way to represent fractional numbers in binary representation for a machine. Another way to represent fractional numbers is the floating point which is capable of representing fractional numbers with good precision. The floating-point representation will require separate hardware or module for conversion i.e., a floating-point unit (FPU). In many cases, speed is more critical than accuracy. fixed-point representation can be used in such cases to represent fractional numbers in binary. In legacy computers, the calculations were only done on integers, and programmers used a software-based method to deal with fractional numbers. A fractional part of a number falls between 0 and 1, in digital signal processing the use of the real number is essential. The fixed-point Binary number representation is a lightweight method in terms of speed and hardware requirements. This method can also be employed in the digital design of a certain application where a representation of real numbers is mandatory.

The binary point remains stationary at the same position where the number of binary digits in each word remains the same, whereas in floating-point the position of the decimal point is determined at the time of processing. As compared to the fixed-point, the floating-point has a much greater range of numbers that it can express. For fixed-point representations m bits can be used for the whole part and n bits can be used for the fractional part this is also referred to as $Qm.n$ format.

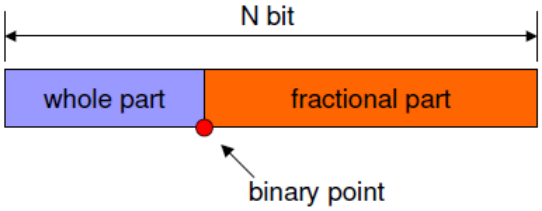

Figure 1: Representation of fixed-point numbers, with whole part (integer) and the fractional part.

The total number of bits required for expressing the real number is the number of bits of the whole part plus the number of bits used to represent the fractional part. For example, in a $Q3.3$ where m and n are equal to 3, the number of bits required to express the number will be 6. To represent a fractional number two parts are required, the integer part and the fractional part. The integral portion can be signed or unsigned. The following representation is shown for signed and unsigned numbers. An $N - bit(s)$ fixed-point Number in binary can be interpreted as an integer or a fractional

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

number. For example, an unsigned 4-bit number with Q2.2, 2bits integer part, and 2 bits for the fractional part. “0100” represents a fractional number 1.0, whereas if it’s viewed as an integer it represents 4.

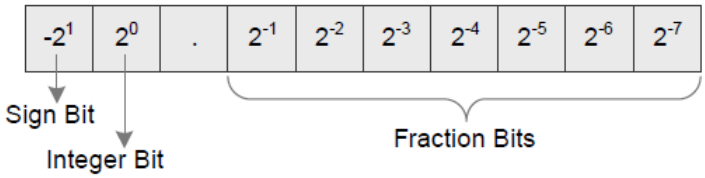

Figure 2: Radix 2 representation of binary fixed-point fractional numbers.

An $N - bit(s)$ fixed-point Number in binary can be interpreted as an integer or a fractional number. For example, an unsigned 4-bit number with Q2.2, 2bits integer part, and 2 bits for the fractional part. “0100” represents a fractional number 1.0, whereas if it’s viewed as an integer it represents 4.

3 Architecture design

Our design is based on a minute set of instructions. These instructions are the key components for controlling the functionality of the processor. Instructions are implemented at the behavior level for this design. Likewise, in a higher-level language such as Matlab, Mathematica the user only specifies the identifiers for the multiplication of a matrix, identifiers A and B represent matrices, and the user at the command prompt only specifies the function or command $A * B$ for the multiplication of these matrices. The rest of the process is transparent to the end-user. Similarly, this design provides microinstructions that are complex in nature but can benefit programmers to write large code with optimal flexibility. Besides this complex instruction, this architecture also provides on-chip memory management, a liner memory will be treated as a 2-Dimensional memory. The design is also capable of performing simple operations on scalar values. Usually, to provide flexibility and more performance, the cost of the hardware raises.

The design is inspired by a MIPS [32] Based processor, which was first designed by MIPS Computer Systems Inc. in 1980. MIPS is now one of the leading corporations in the embedded industry, their processors can be found in many modern devices like Digital cameras produced by canon, some windows mobile devices, Routers manufactured by Cisco, Gaming stations like Sony play station, etc.

Our design has several types of instruction, such as arithmetic of the matrix with matrix, matrix to scalar, scalar to scalar, Immediate Arithmetic of scalar and Matrices, Immediate loading and Branching operations. The instruction contains an op-code similar to. The Instruction set architecture for a max 256 x 256 matrix is as follows. Op-code is a five bits field that provides information about instruction type to the processor. The size is kept to 5-bits for future addition of new instructions. MS is the address of the Source Matrix, MT is the 4-bit address of the target matrix where the remaining identifier MD is the Address of the destination Matrix which is also 4-bits, some bits are kept at the end of the instruction for don’t care and can be used to specify offset.

RS, RT, and RD are the Sources, Target, and Destination addresses of the memory locations. The memory size can be kept according to need, for this the purpose of understanding and simulation the memory width of a cell is kept to 14-bits per cell, with 10-bits for the integer part and 4-bits for the fractional part, with the depth that can be kept according to requirements simply by changing parameters in the VHDL code. The load immediate can load values to a memory location in the memory. simulation memory width of a cell is kept to 14-bits per cell, with 10-bits for the integer part and 4-bits for the fractional part, with the depth that can be kept according to requirements simply by changing parameters in the VHDL code. The load immediate can load values to a memory location in the memory. The memory is kept likewise the register memory of a MIPS-based processor [32]. The memory is read line a combinatory circuit. There are two types of volatile memory commonly used SRAM and DRAM, memories are usually built using several chips. SRAM is built D-flip-flops whereas [33] a DRAM uses up a capacitor coupled with a transistor for 1-bit storage. The storage memory reads two locations from a common pool of memory, as the data is being held. It can be read at any time and can be transferred using wires, writing to the memory is done through a single port. The data will be available on the data output ports of the specified addresses. These addressees represent the operands on which computation is required. An index generating mechanism will generate the address of the operand for the matrices elements, whereas for scalars direct address will be used for the operands provided in the instruction set architecture. The memory will be written only when the signal write enable is set on the negative edge of the clock cycle.

The memory orientation is linear in nature, the addresses continue similarly as a vector. To provide a transparent functionality, another extra memory is employed to store the information of the matrix dimensions which stores the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

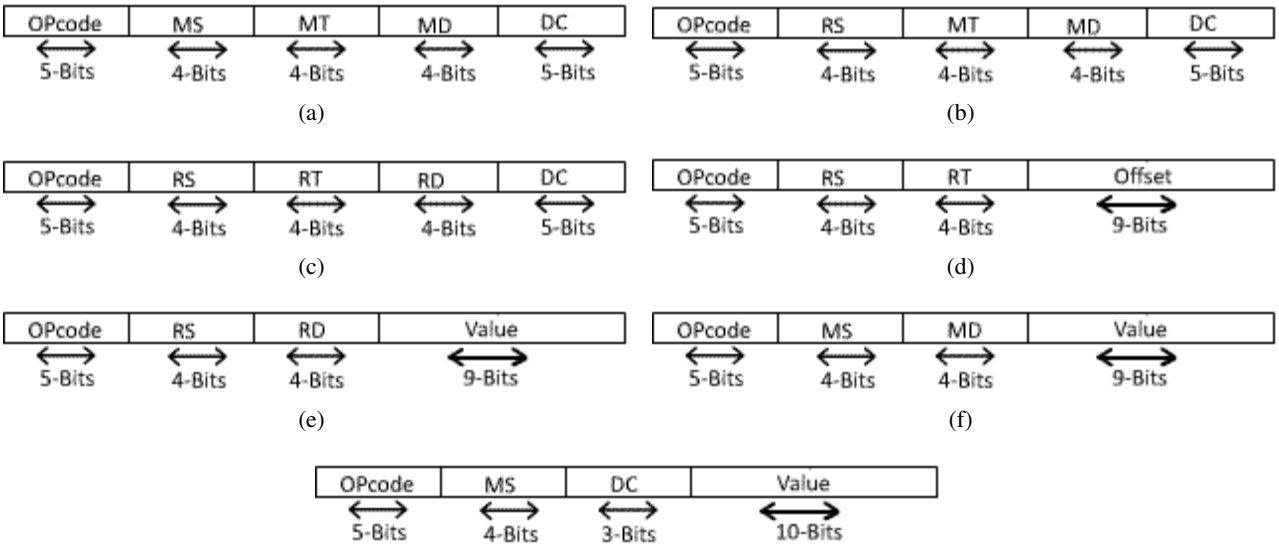

Figure 3: ISA and the op codes for: (a) Matrix with Matrix, (b) Matrix with Scalar, (c) Scalar with Scalar, (d) Branch Instructions, (e) Immediate Arithmetic Scalar, (f) Immediate Arithmetic Matrix, (g) Immediate Loading

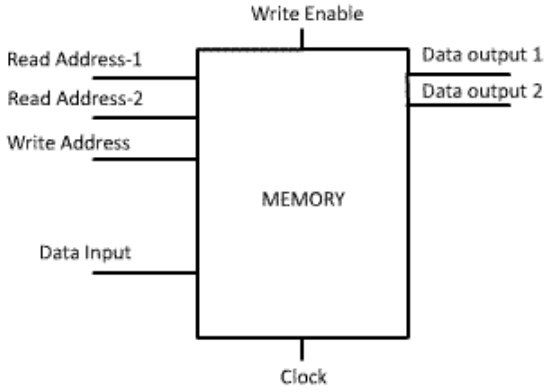

Figure 4: Radix 2 representation of binary fixed-point fractional numbers.

number of rows and number of columns for a specified matrix. Two matrices can be stored in the main memory in a continuous linear manner. For two matrices A and B that can be placed in the memory according to the following order:

$$A = \begin{bmatrix} A1 & A2 & A3 \\ A4 & A5 & A6 \\ A7 & A8 & A9 \end{bmatrix}, B = \begin{bmatrix} B1 & B2 & B3 \\ B4 & B5 & B6 \\ B7 & B8 & B9 \end{bmatrix} \quad (1)$$

In general computing the memory is usually managed by the operating system, this management is transparent to the user. In the case of this processor, a liner memory will be used, and the management of the memory will also be transparent to the end-user. The user will only have to write the microinstruction like “ADDm C, A, B”, The user will only specify whether it is required to add a scalar value or a matrix point to point. There are many orientations of memory that can be used, a dual-port read memory design like MIPS32 processor has a register file with output ports of data and three input ports for address, another port for data input plus some control pins like reading write control. In this case, a similar type of memory will be used, the major benefit of this memory is that it can provide two operands and a target location which can complete one instruction in a single cycle. A general code matrix handling mechanism could be written on the software side. But the addition of this functionality will cost more in terms of CPU cycles as the number of decisions and comparisons will be increased in general computing of the main memory that causes the major

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

Memory Address	Content
0	A1
1	A2
2	A3
3	A4
4	A5
5	A6
6	A7
7	⋮
17	B8
18	B9
	⋮

Figure 5: Linear storing of the two matrices A and B in the memory with their addresses.

bottleneck. For example, if the processor requires fetching data from the memory for arithmetic or a logical use can cost many machine cycles. Some general computing systems even use cache and virtual memory where the required memory location is not available on the cache which will cost even more cycles to fetch. But the beauty of the dedicated systems is that custom designs can overcome these limitations hence providing more performance. While in some cases the cost in terms of chi area may rise to a very large extent. A mechanism for the conversion of a matrix from its subscript index to a single index is adopted in the realization of this processor. The processor provides an on-chip mechanism for handling matrices, for this purpose a sparse memory is used. The layout of this small-sized memory is explained below, and the module is shown in the figure:

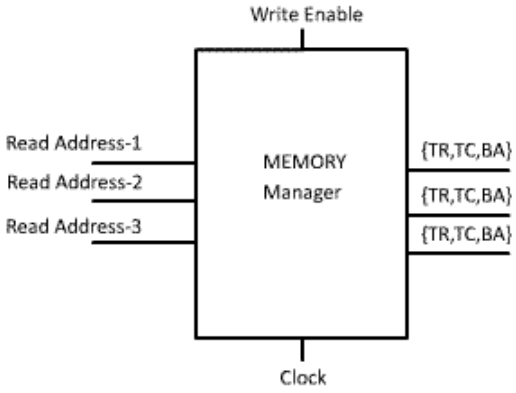

Figure 6: Memory manager View of a dual port memory MIPS32 Memory.

It is required to store 10 matrices in the memory of size m x n, the height of this memory will be 10, and the address lines must be 4-bits. The memory manager works as a writeable lookup table which can be written by setting the Write Enable to '1' on the negative edge of the clock. The data will be accessed from a common pool and will be available on the outputs. An input address is 2. The module will output the data available on the address location 2. The second location points to the attributes of Matrix Number 2, i.e. Total Rows (TR), Total Columns (TC), and Base Address (BA).

The total size (in bits) of the width of a single row in the memory manager is the sum of Total Bits (TR) + Total Bits (TC) +Total Bits (BA). The base Address is determined on-chip. The base address of the first size 3x3 matrix stored in the linear memory will be kept to zero. The base address of the second size 3 x 3 matrix stored in the memory will be kept 9, indicating the elements of the second matrix are starting from the ninth location in the memory.

$$A = \begin{bmatrix} A1 & A2 & A3 \\ A4 & A5 & A6 \\ A7 & A8 & A9 \end{bmatrix}, B = \begin{bmatrix} B1 & B2 & B3 \\ B4 & B5 & B6 \\ B7 & B8 & B9 \end{bmatrix}, C = \begin{bmatrix} C1 & C2 & C3 \\ C4 & C5 & C6 \\ C7 & C8 & C9 \end{bmatrix} \tag{2}$$

The memory manager module will hold the information about the specified matrix that is being addressed on the main memory. An element is a matrix that can be addressed by its location number. The following will explain the location index of a matrix. Trivially an element of a matrix can be expressed by A (x, y) where x is the row index and y are the column index. This is usually known as the subscript notation of an element; this unique element can also be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

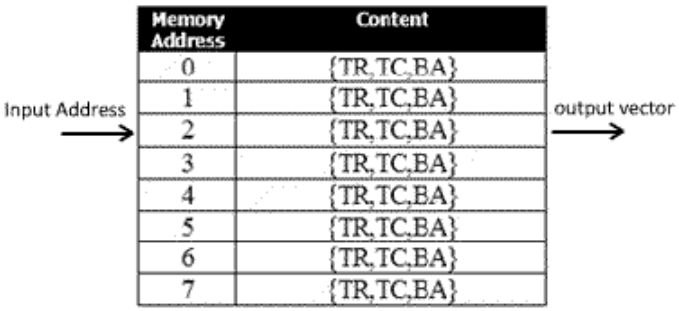

Figure 7: Memory manager View of a dual port memory MIPS32 Memory.

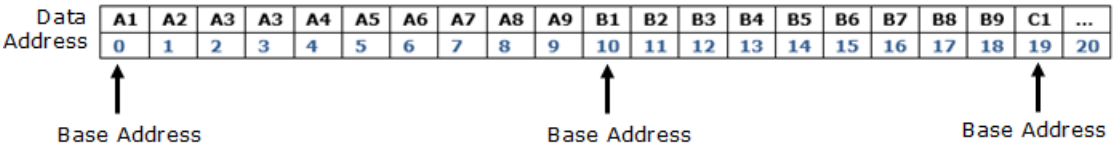

Figure 8: Memory manager View of a dual-port memory MIPS32 Memory.

represented by a unique index A(Z). In other words, it can be written as A (x, y) =A (Z), let us consider an example of a below 3 x 3 matrix.

$$A = \begin{bmatrix} A(1,1) & A(1,2) & A(1,3) \\ A(2,1) & A(2,2) & A(2,3) \\ A(3,1) & A(3,2) & A(3,3) \end{bmatrix} \tag{3}$$

$$A = \begin{bmatrix} A(1) & A(2) & A(3) \\ A(4) & A(5) & A(6) \\ A(7) & A(8) & A(9) \end{bmatrix} \tag{4}$$

$$A = \begin{bmatrix} A(1) & A(4) & A(7) \\ A(2) & A(5) & A(8) \\ A(3) & A(6) & A(9) \end{bmatrix} \tag{5}$$

Equation 3 Representation of matric locations in sub-index forms Equation 4 representation of elements of matrix in single index form in row-oriented manner Equation 5 Representation of matrix elements in column-oriented manner in single index. The benefit of using the index notation is that we can put all the items of a matrix in a continuous order even in a straight linear memory and can address the element with index representation. For more elaboration of this concept, let us consider the example of MATLAB function sub2ind (X, Y). This function converts the subscript i.e. the row-column value to a single index.

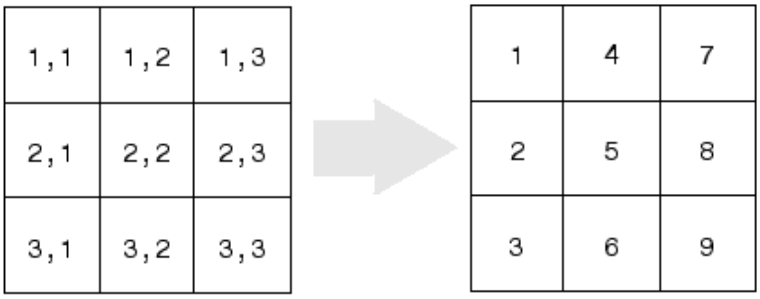

Figure 9: Conversion of subscript index to single index sub-index to single index for a 3 x 3 matrix.

To realize this conversion a methodology was adopted. The expression helps in converting the subscript values to single index values. Mathematically the methodology can be written as:

$$\begin{aligned} \mathrm{A(x,y)} = \mathrm{A(z)} = A\left(((x-1)*A_{TC})+y\right) \text{ Where} \\ \mathrm{Z} = ((x-1)*A_{TC})+y) \end{aligned} \tag{6}$$

$$\begin{aligned} \mathrm{A(x,y)} = \mathrm{A(Z)} = A\left(((y-1)*A_{TR})+x\right) \text{ Where} \\ \mathrm{Z}\left(((y-1)*A_{TR})+x\right) \end{aligned} \tag{7}$$

Equation 6 can be used to compute the single index in row orientation and the expression. Equation 7 can be employed to calculate a single index in column orientation like MATLAB. A simple MATLAB simulation code is presented here to justify the index conversion. The first provided code work in Row orientation and the second sample code works in a column-oriented manner.

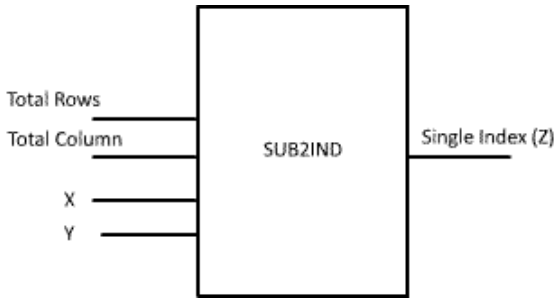

Figure 10: Sub to index conversion module.

Instruction memory contains the instruction or operation to be carried out. One slice or a row contains the Op-code, Source, Destination and the target Identifier of the matrix/scalar to be worked upon. The Op-code allows differentiating between the types of instructions according to which the control sequence will be generated. The nature of the instruction can be different from another. The flow control of a sequence is controlled by a special register Program Counter (PC). The size of the Program Counter Register depends upon the Height of the instruction memory. Suppose if the height of the Instruction memory is 128 Locations, then the PC register size will be $\log_2(128) = 7$ bits. The operating mechanism of this guide is similar to the memory manager. MD is the Destination Matrix. MS is the Source Matrix and MT is the Target matrix address.

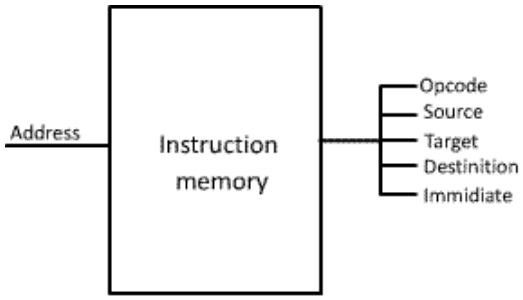

Figure 11: Instruction memory with outputs.

To provide a re-use mechanism of the functional components, a central unit is introduced which contains the main adder and is capable of performing logical decisions. The arithmetic and logic unit has two outputs, which is single elements. Arithmetic and Logic operations can be carried out on these operands. The ALU Op-code identifies what type of operations are to be performed on the operands [34]. It can be logical or arithmetic like Add, Subtract, Divide. The ALU performs arithmetic and logical operation on Fixed-point Binary and Simple Binary Notations. The Processor requires operating on Simple Binary Numbers and Fixed-point Binary Numbers together. The interpretation is different, but the arithmetic operations resemble in nature. Fast Adders can be implemented to minimize the delay. Similar components can be internally Implemented to maximize the throughput. This module is a combinatory circuit and It is not dependent on clock cycles.

The Control Unit dispatches the concerned control signals based on the nature of instruction in use [34]. This dispatch of instruction is based on the Op-code. The Control Unit is also a combinatory circuit. The Control Unit is also a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

combinatory circuit and is not dependent on clock cycles. The Control Unit is like a look-up table that will output the corresponding set of control vectors. The control signals then will be supplied to the corresponding fictional unit in the architecture. The control vector corresponding to the given Op-code will be available on the bus.

In general computing usually, loops are used to perform the matrix, addition, subtraction, etc. For example, to traverse a full Matrix the following sample code will be handy. This code will traverse all the elements of the matrix. To provide a control mechanism for the loop, a specially modified counter is built that gives a variable to increment over a clock cycle. The counter design has a filled to indicate the endpoint on which the counter will reset and raise a flag that shows the counter has completed one loop. This flag can be used to trigger other modules in the circuit. The initial value of the counter starts from one, on every clock cycle, the initial value is incremented by one.

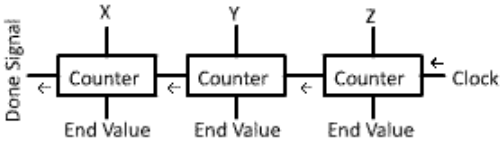

Figure 12: Instruction memory with outputs.

These counters use up an adder and some flip-flops internally. An internal view of the counter used in the design is presented below, the module just increments the value starting from one, until the last value. On the last value, the comparators compute the current and End value, if the current value is the last then an end flag (signal) is raised to inform that one iteration is completed.

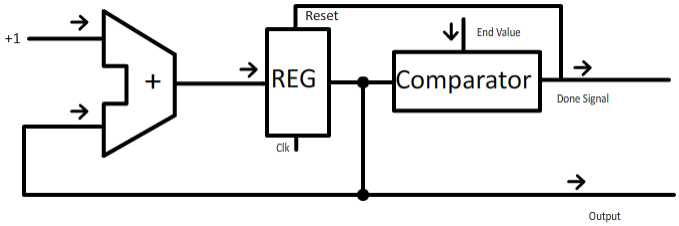

Figure 13: Internal view of the counter that uses an adder, register, and a comparator along with other combinational logic.

The multiplexer is one of the core components of digital design, this component can save a lot of resources in terms of hardware. Multiplexers can transfer or select data available on its inputs to the output port, based on the selection which can be selected from the selection switch. For N numbers of input lines, the required size of selection can be represented as $\log_2(N)$.

Every processor which has an instruction memory requires a mechanism to fetch the next instruction from the instruction memory. Usually, a special Register known as Program Counter [34] is employed to hold the address of the next instruction [34]. In a normal program flow the starting address '0' is loaded on the program counter register. After every clock cycles the register increment the instruction one by one. This scheme employs an adder to fulfill this increment. The current address is incremented by the adder and is available to the input of the program counter register which will be written on the next available clock cycle. In some cases, Branching might be required. The branch instruction directly holds the branch address, which is available on the Multiplexer available. In branching the logical operation ALU raises a flag that works as the selection line for the multiplexer placed in the instruction fetching mechanism. When the ALU Flag is set to high, the Multiplexer will select the forced input (branch Address) to the Micro Instruction memory, and the next address relative to the branch address will be written on the Program Counter Register. The dotted area shows the components and working of the next instruction fetching scheme, for simple processors this scheme is very commonly used, in the next sections the Instruction Fetching mechanism will be represented by a dotted line which indicates the same organization.

The proposed method comprises an on-chip memory manager the major benefit of using this unit is that it will provide a very large amount of flexibility as explained in the introduction section. The biggest challenge for an on-chip memory manager is implementing high-level programming techniques for the hardware. Designing fixed-size hardware restricts flexibility to a very large extent. The functionality of much bigger instruction can be implemented using small scalar instructions, But Complex functions like Matrix handling (size, Addition, Subtraction, Multiplication, etc.) are provided on-chip. To provide this functionality on-chip, a similar scheme is employed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

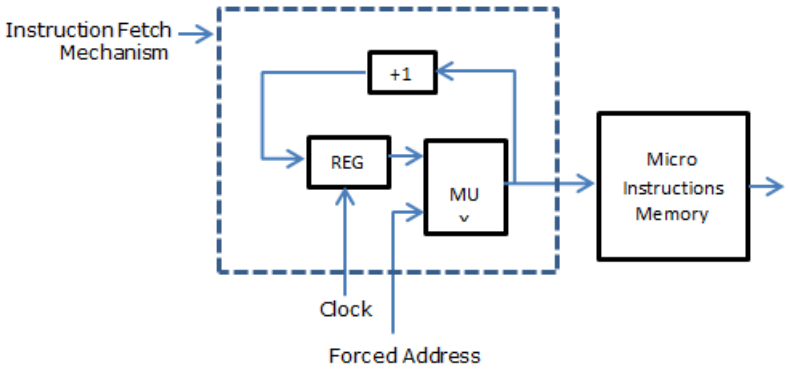

Figure 14: The mechanism used for fetching the new instruction and providing jumps represented by the forced address.

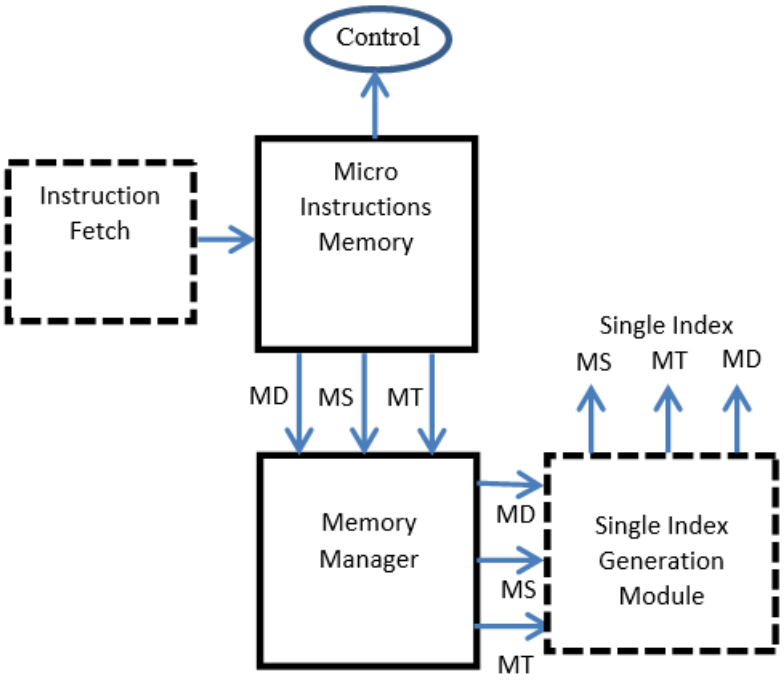

Figure 15: Data path of the single index generation mechanism MS is the source matrix address, MT is the target Matrix Address and MD is the Destination Matrix.

The instruction Fetching mechanism will fetch the next instruction. The instruction basically holds the address of the matrix. Let suppose; we can store 10 matrices on our Memory then the memory manager will hold the information for the size and base address of the specified matrix. This information about the total rows and columns of a matrix can also be manipulated after saving matrix locations. The information of the required corresponding Matrices is then transfer to the next section, which will compute the Sub index to a single index form.

The counters can generate (X, Y, Z) indices used in a loop of a high-level language. The counter will be provided with an end value, which will determine where to stop. The counter is initialized with a value one which is the default value of the program. This is done to match the index assignment like MATLAB, the matrix sub-indices start from 1 instead of 0, unlike other programming languages. The control unit sends control signals to the multiplexers, which helps in selecting the right end address for the counter. The major purpose of using these multiplexers is the nature of different operations, Addition and Subtraction may have different Index Generation as compared to matrix multiplication, thus multiplexer is used here to provide multi-functionality.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

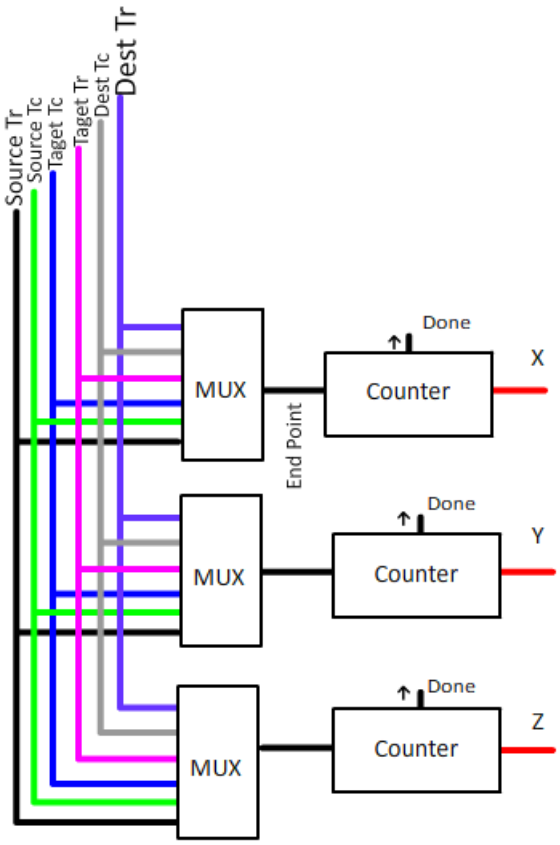

Figure 16: Switching network data path that provides flexibility to load a counter with desired end address.

For operations like matrix Addition, Transpose and Subtraction only two counters can play the trick, but to provide on-chip matrix multiplication a third counter has to be employed for the third index generation. The counters are provided with the end address, on every clock cycle, the counter starts moving towards its endpoint, each time the counter is incremented by one, as the identifier index of a loop is incremented on every successful completion. Another set of multiplexers is provided with the outputs of the counters, these multiplexers are then used to create a valid Sub Index Address if the matrix location is according to the operation in progress.

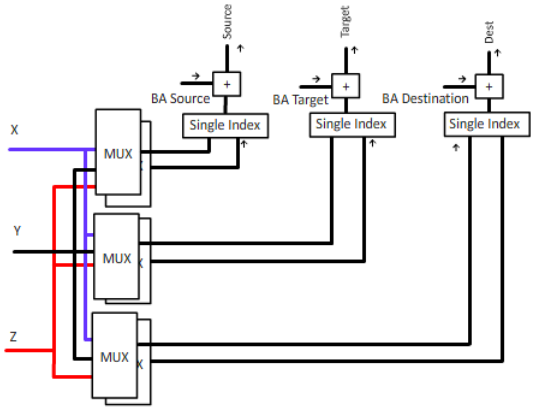

Figure 17: Sub part of the switching network, the control signals will select the appropriate sub-indices required for the larger complex operation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

The single index converts the Subindex to a single index as explained earlier in the Sub to the Index section in the above session. This module will convert the subscript to a single index, then for the specific matrix the single index is added to the base address of the matrix, and the resultant will be the exact address of the element in the main memory. The major benefit of providing an on-chip index conversion will provide flexibility and ease of use to the programmer, although this mechanism will cost more in terms of hardware. But will increase the functionality to a very large extent. Providing the on-chip looping scheme is quite challenging, each counter has to be triggered at the right time. The timing issues can be resolved by using an intelligent control methodology through which each instruction operation depends on the type of instruction that enables the control unit to decide which signal should be the clock of the counter that will trigger the next value. An intelligent switching network is embedded which determines the triggering pulse to a specific counter. Suppose in the case of Matrix Addition the Sub indexes are the same for the operands and the destination and can be done in two loops using a high-level programming language, the end signal of the Y index counter can trigger the next counter to go to the next state.

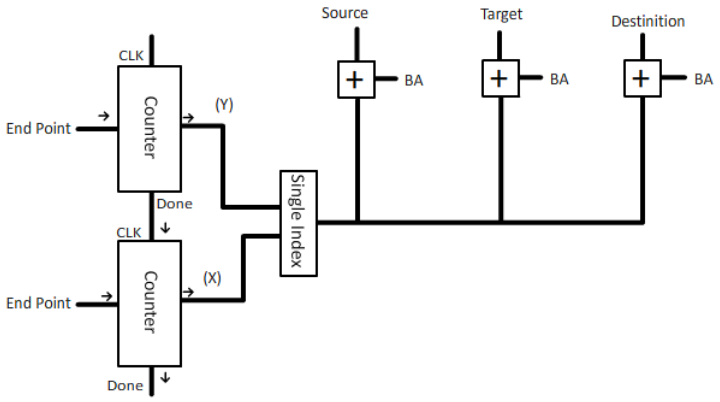

Figure 18: Single view of sub-index generation mechanism for operations like Matrix Addition, Subtraction, and Point to Point Multiplication, etc.

Matrix addition and subtraction is quite straightforward, only two indexes can represent the Source, target, and destination. The following figure shows that only two counters can be employed to generate matrix indices. Usually, Matrix addition and subtraction can be expressed as $Dest(X, Y) = \text{Source}(X, Y) \pm \text{Target}(X, Y)$. Only two identifiers can be used to traverse the whole matrices and luckily the increment is in the same order, whereas index generation for Matrix multiplication is quite complex as compared to matrix Addition and Subtraction.

Transpose of a matrix is simply computed by reversing the Sub-indices i.e., $\text{W(X, Y)} = \text{W(Y, X)}$ the methodology used is quite simple. For this purpose, a pair of counters can be used to generate the indices. The diagram below shows how a single pair of a counter can be used to generate indices.

The branching instruction transfers control to the specified line number of the instruction memory. The Jump address is stored in the instruction along with two operands addresses stored. The ALU will set the flag to high if the condition is true (for conditional Jump). For unconditional Jump, the ALU will set the flag.

The determination of length of one clock cycle can be determined by the following expression, where CCT is the Clock Cycle time and D is expressing some other latency such as data Arrival delay, clock skews adjustment, etc.

$$(\text{CCT}) = \sum \text{All Delays of components } + D \tag{8}$$

4 Results

The design was implemented using Verilog hardware descriptive language (VHDL) each instruction was tested individually along with a mixture of different instructions to verify the working of the instructions. The memory was populated with matrices, the memory is linear in nature, but with the help of on-chip memory management the liner memory will be handled as matrices.

For a testing purpose, the memory is initially populated with the variables shown in 9. The Values are placed linearly in the memory. The single index generation Module will separate the matrices from one another. For a series of test

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

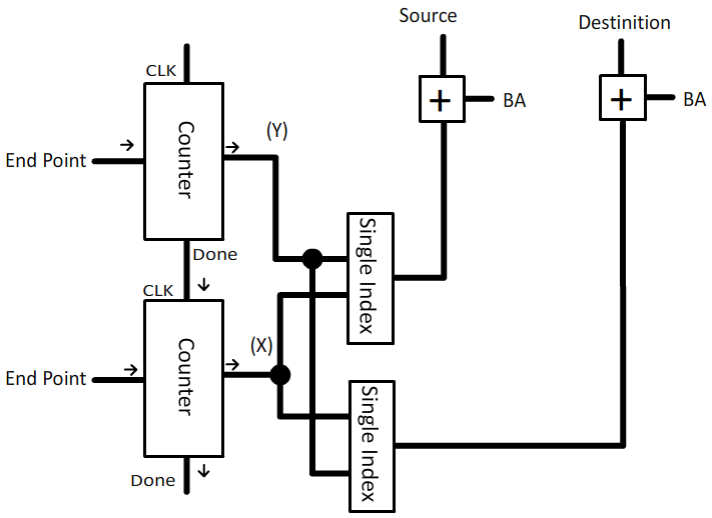

Figure 19: Single index generation mechanism for computing transpose of a matrix.

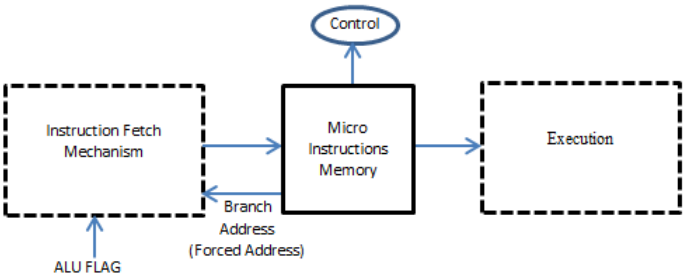

Figure 20: Top Level view of the final data path.

simulations, the following matrices are assumed to be preloaded into the memory of this processor, the matrices A and B are both of 3×3 dimensions. The proposed scheme can handle arbitrary dimensions of matrices as it holds the total row and column information of the matrices stored in the sparse memory manager.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{9}$$

To provide a better understanding, the instructions are named in textual format for easy recognition of the specific Op-code and the Arithmetic operation that is to be performed. The following piece of code first creates the matrix and registers the location on to the memory manager with the total number of rows and cols of the new matric, that are basically the dimensions of the new matrix stored in the memory. M0 represents the address 0 of the memory manager i.e. it is equal to (0000)2. $M3 = A$ and $M4 = B$ are the matrices that contain the operand matrices.

Listing 1: Test for adding two matrices.

```
% Our Assebmbly code for adding two matrices.
ADDm M0, M3, M4
```

The simulation results for the above piece of code are shown in the above wave timing diagram. In the literal list the pcin indicates the current instruction that is in progress, it indicates that processing a matrix of nine elements will require nine clock cycles to complete, after the ninth clock cycle the program counter is sent a signal to update its value and the next instruction is then fetched, The literal memory_data_in indicates the results of the computed value element by element on every clock cycle. Where memory_data_out_a, memory_data_out_b indicate the outputs of the memory which are the elements required for this operation. The input to the memory indicates the results of the element by

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

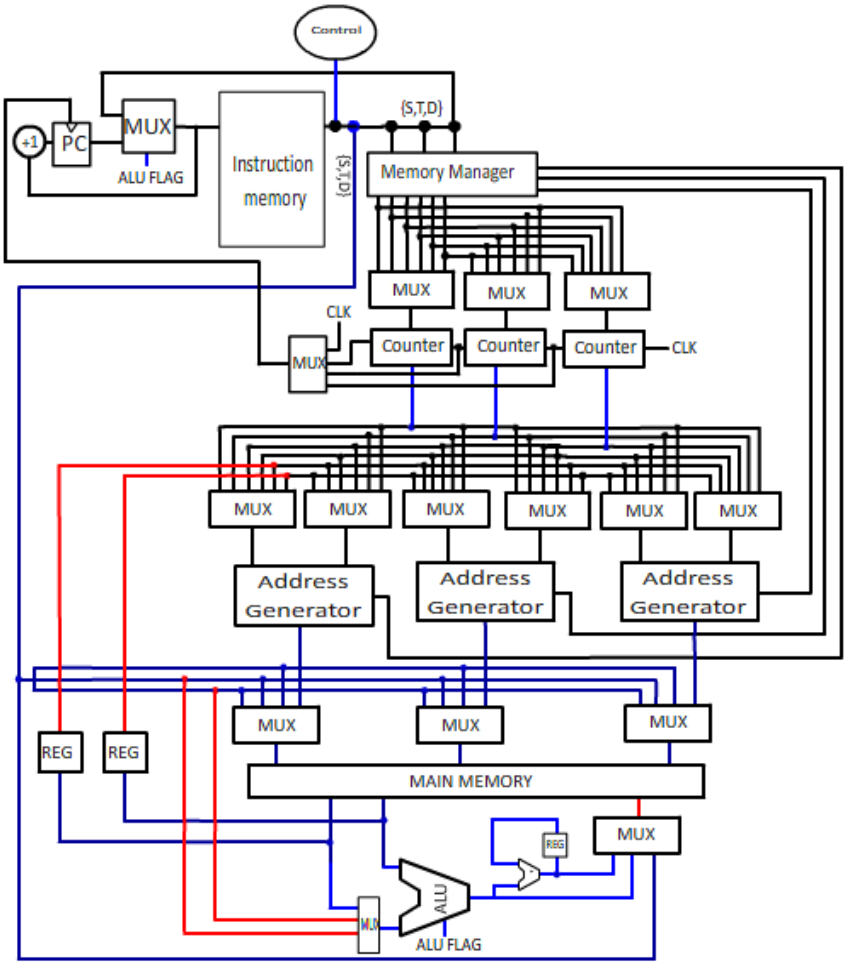

Figure 21: The all merged datapath i.e, the main architecture with the organization of components.

element addition of the matrices in a column-oriented manner. The matrix M3 elements were sored in the memory location from 50 to 58 as there were 9 locations, and M4 elements were stored from 41 to 49th memory locations and the write address ranges from 32 to 40. Some results from a non-synthesizable function that represents the fixed-point value in real number format used just for indication which was removed for synthesis later.

Listing 2: Test for adding two matrices.

```
% Our Assebmbly code for subtracting two matrices.
SUBm M0, M4, M3
```

The results of the subtraction of the two matrices are shown in the wave timing diagram. The resulting matrices contain elements with a negative sign, that are also indicated in the results of subtraction. The keyword SUBm indicates that two matrices are being subtracted. In the next wave timing diagram results of the dot, multiplication is shown, dot multiplication i.e. element to element multiplication. This functionality can be very useful while filtering images in the frequency domain. The simulation results are shown in the next figure, and the textual format is represented as:

Listing 3: Test point to point multiplication of two matrices.

```
% Our Assebmbly code for point to point multiplication of two matrices.
MULmd M0, M4, M3
```

The next wave timing diagram represents the results of the transpose of a matrix, write address of the memory memory_write_address indicates the new write location indicated by the literal indicated i.e., memory_write_address_b.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

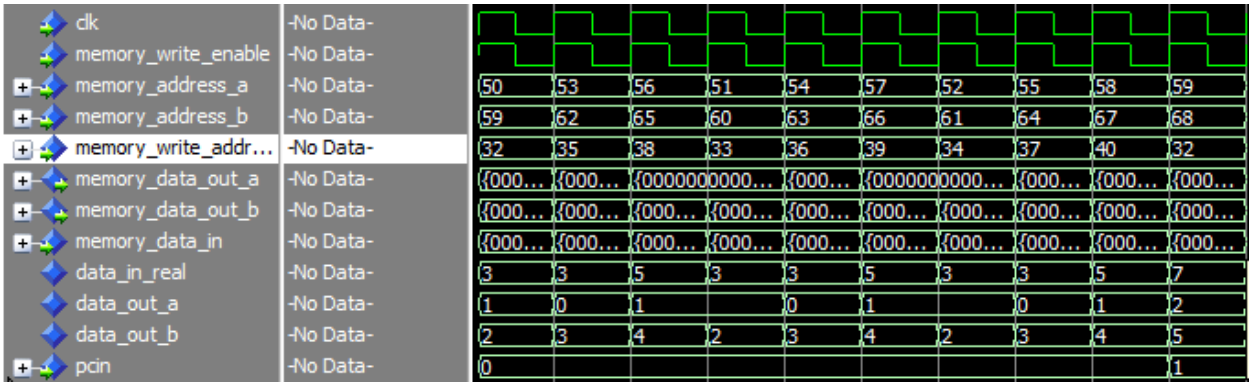

Figure 22: Simulation results of addition of matrix A and B for Listing 1.

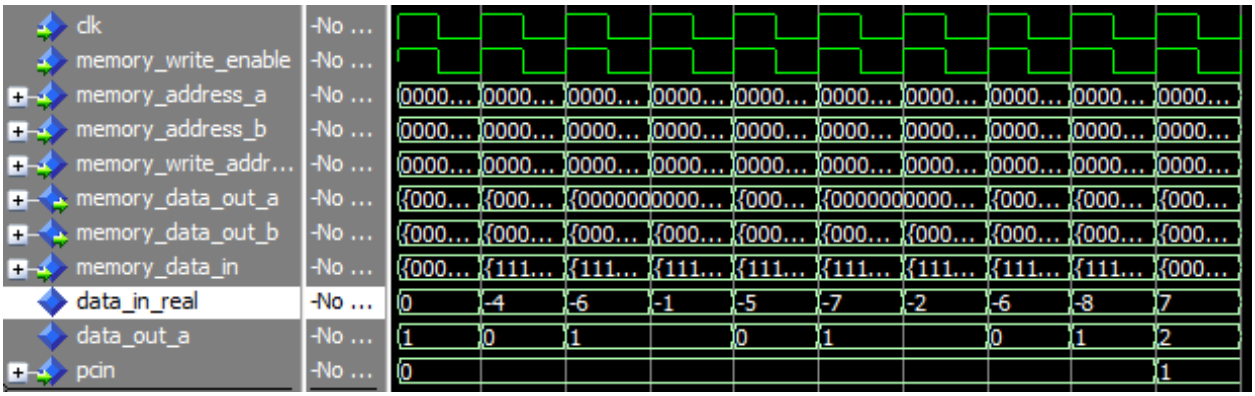

Figure 23: Wave timing diagram for subtraction of matrix B and A as shown in Listing 2.

The binary instruction can be represented as Dc indicates don't care term and has no effect at all as transpose requires one operand and a destination to write.

Listing 4: Testing transpose of a matrix.

```
% Our Assebmbly code for transpose of a matrix.
TP M0, DC, M4
```

The next wave timing diagram represents the multiplication results of two matrices, here whenever the sum of the multiplication of a row with a column is computed it is written onto the memory on the respective location, in the wave timing diagram the memory writes enable signal specifies when the resulting element is ready to write i.e., the sum of the multiplied row with the column.

Listing 5: Testing transpose of a matrix.

```
% Our Assebmbly code multiplication of matrices.
MULm M0, M3, M4
```

Simulation of the wave of handling scalar values is presented in the next diagram. The results are shown according to the instruction number. Scalar values require a single cycle to complete, the first 32 (less or more) locations were reserved for scalar values, the instruction LOD operates on immediate loading values to memory, the 0th instruction loads the prescribed value either integer or fractional, to the specified location, R1 represents the 1st location in the main memory i.e. the address one of the memory.

Listing 6: Testing transpose of a matrix.

```
% Our Assebmbly for handling scalar values.
0 LOD R1, 0;
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

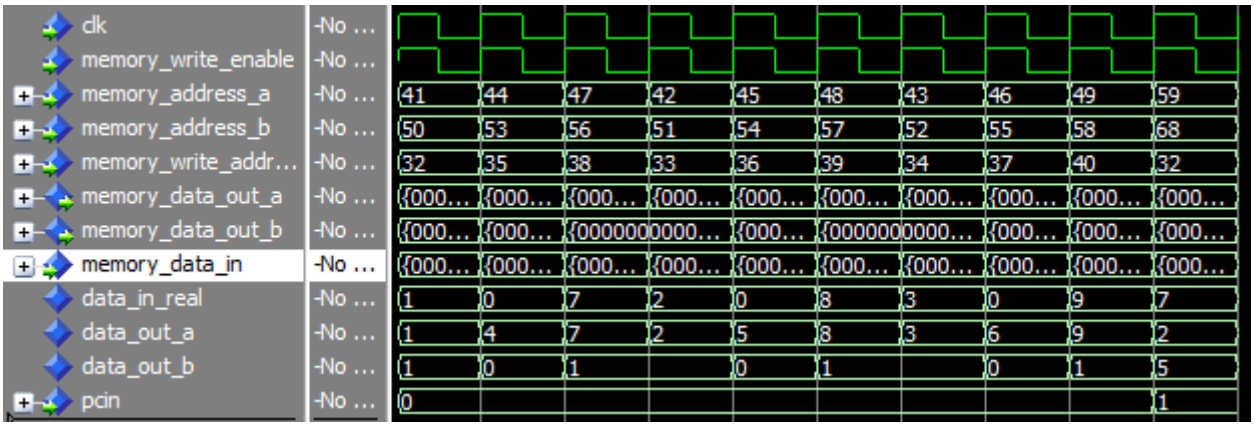

Figure 24: Wave timing diagram for point to point multiplication of two maatrices as shown in Listing 3.

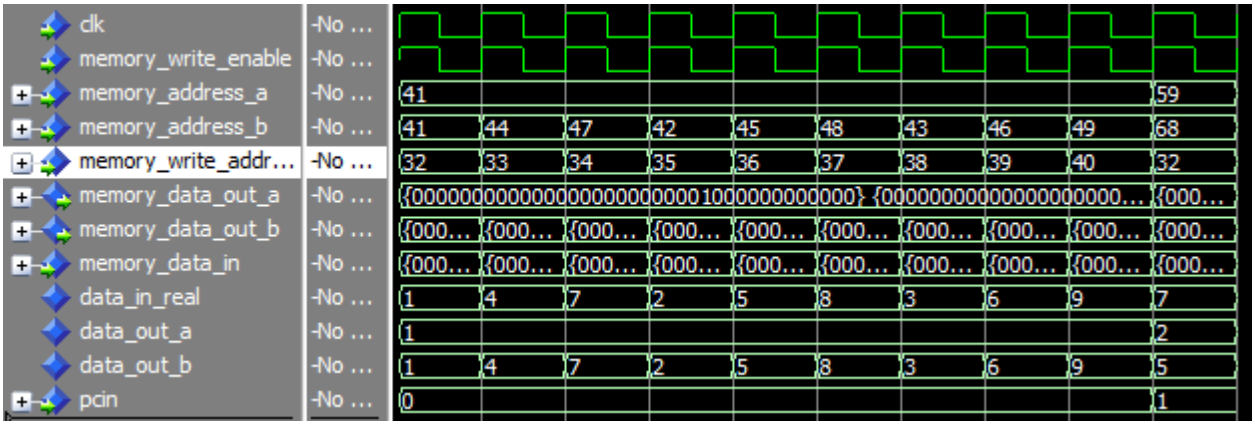

Figure 25: Wave timing diagram for transpose of a matrix as shown in Listing 4.

```
1 LOD R2, fixed (1);
2 ADD R0, R1, R2;
3 SUBB R0, R1, R2;
4 MUL R0, R1, R2;
```

Listing 7: Testing conditional branches.

```
% Our Assebmbly for handling conditional branches.
0 LOD R1, 0;
1 LOD R2, 0;
2 JEQ R0, R1, Line 10;
3 JEQ R2, R1, Line 10;
```

JEQ represents the Op-code for the jump when provided operands are equal. The literal pcin is indicating the flow of instruction, in the code snippet provided above the contents of locations R1 and R2 are equal and the results of the jump can be viewed in the simulation diagram. The provided values are non-fractional.

Listing 8: Testing writing and reading specific locations of a matrix

```
% Our Assebmbly writing and reading specific locations of a matrix.
0 LOD R10, 1
1 LOD R2, fixed (1)
2 LSR R2
3 WFSR M1, R10, R10 %Specifies the row and col
4 LRC R10, R10
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

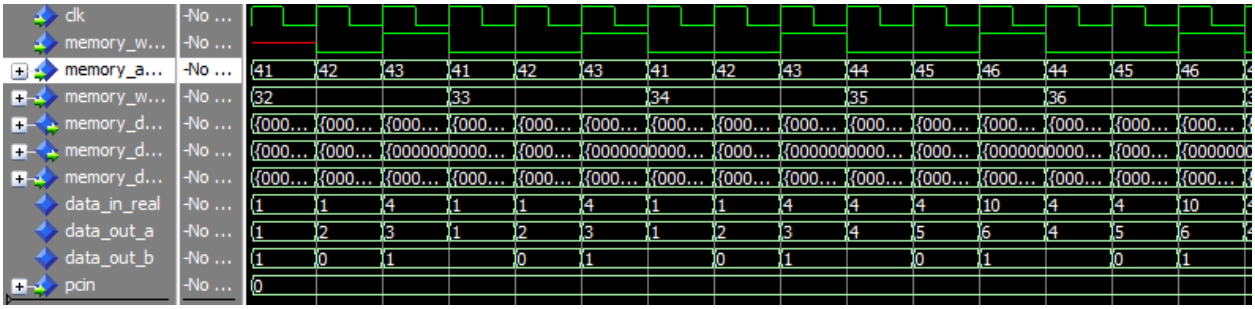

Figure 26: Wave timing diagram for multiplication of matrices as shown in Listing 5.

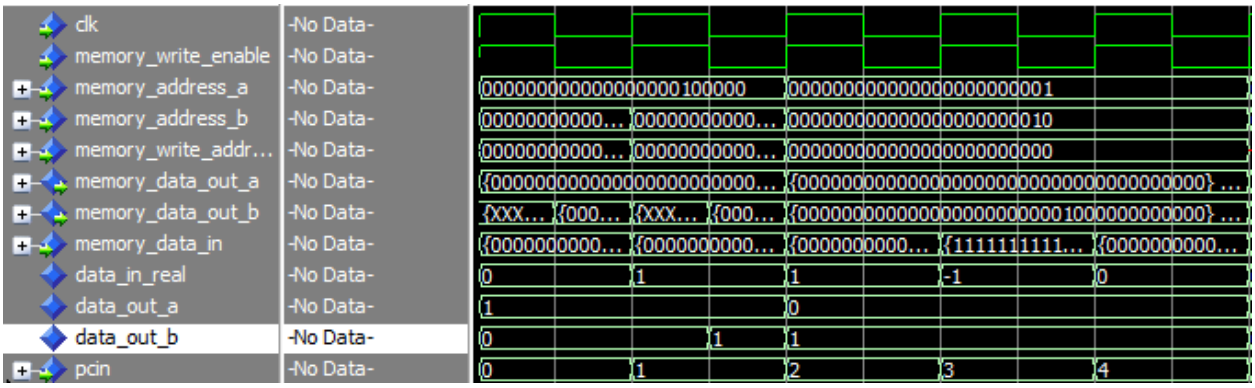

Figure 27: Wave timing diagram for code Listing 6.

```
5 WTSR M1
6 SFSR R12
```

This code segment loads a value onto a memory location which can be used as indices to represent elements of a matrix, the following results show that first the contents presented on the memory location R2 (i.e., 2) are loaded to the special register, then WFSR (write from Special register) instruction writes the contents of the first Special Register to the prescribed location in a matrix i.e. M1(R10,R10) where it represents M1(1,1) the first row and the first column, as indicated in the wave timing diagram, the first element of the M1 is stored on 41'st location of the main memory.

LRC loads the row and column index of the element from a prescribed memory location, this loads the data of the specified locations to two special registers used, this data themselves are the address of the row-column of a matrix. WTSR (write to special register) then loads the contents of the prescribed row and column of the matrix i.e. M1 (R10, R10) to the special register, afterward SFSR (save from special Register) copies the data available on the special register to the specified location.

5 Conclusion

The proposed processor design is capable of performing direct operations on matrices. It provides all the basic building blocks that are required for building larger computational programs. These basic building blocks include providing direct operations on the matrices. The proposed processor design provides an ease to microcode programmers with the flexibility for handling different dimensions of images. The matrices stored in the main memory can be subjected to matrices operations along with some scalar operations. The scalar operations can provide help in branching operations, besides this essential to write a value to a specific location of the memory. These scalars provide both functionalities. The presented design is a base model, and the clock cycle time (CCT) in this architecture can be seriously be improved with the help of implicit parallelism. The stages are organized in such a way that pipelined registers can be added between them. The simulation results indicate the successful implementation of the proposed design. In future, a fully convolutional model will be presented.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 April 2022 doi:10.20944/preprints202204.0129.v1

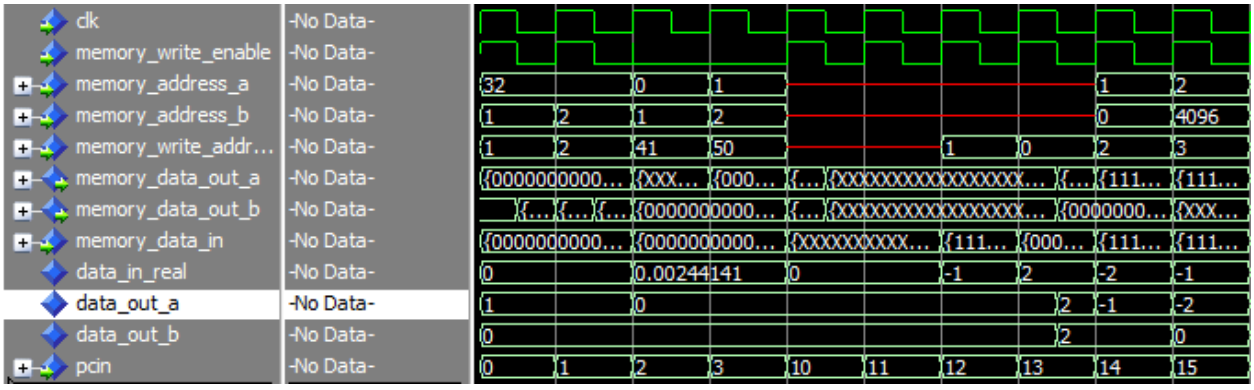

Figure 28: Wave Timing diagram for handling conditional branches i.e., Listing 7.

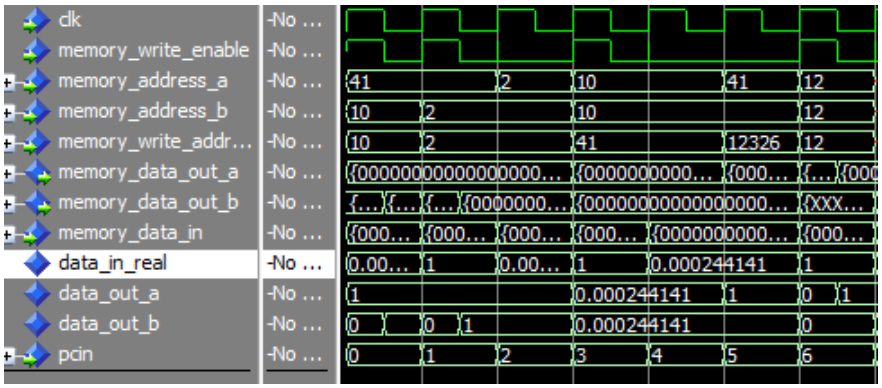

Figure 29: Writing and reading specific locations of a matrix as shown in Listing 8.

References

[1] Rajesh Kumar, Abdullah Aman Khan, Jay Kumar, Noorbakhsh Amiri Golilarz, Simin Zhang, Yang Ting, Chengyu Zheng, Wenyong Wang, et al. Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging. *IEEE Sensors Journal*, 21(14):16301–16314, 2021.

[2] Muhammad Usman Akram, Hassan Moatasam Awan, and Abdullah Amaan Khan. Dorsal hand veins based person identification. pages 289–294, 2014.

[3] Qi Huang, Waqas Amin, Khalid Umer, Hoay Beng Gooi, Foo Yi Shyh Eddy, Muhammad Afzal, Mahnoor Shahzadi, Abdullah Aman Khan, and Syed Adrees Ahmad. A review of transactive energy systems: Concept and implementation. *Energy Reports*, 7:7804–7824, 2021.

[4] Waqas Amin, Qi Huang, M Afzal, Abdullah Aman Khan, Zhenyuan Zhang, Khalid Umer, and Syed Adrees Ahmed. Consumers' preference based optimal price determination model for p2p energy trading. *Electric Power Systems Research*, 187:106488, 2020.

[5] Abdullah Aman Khan, Jie Shao, Waqar Ali, and Saifullah Tumrani. Content-aware summarization of broadcast sports videos: An audio-visual feature extraction approach. *Neural Process. Lett.*, 52(3):1945–1968, 2020.

[6] Rajesh Kumar, Wenyong Wang, Jay Kumar, Ting Yang, Abdullah Aman Khan, Wazir Ali, and Ikram Ali. An integration of blockchain and AI for secure data sharing and detection of CT images for the hospitals. *Comput. Medical Imaging Graph.*, 87:101812, 2021.

[7] Abdullah Aman Khan, Haoyang Lin, Saifullah Tumrani, Zheng Wang, and Jie Shao. Detection and localization of scorebox in long duration broadcast sports videos. In *International Symposium on Artificial Intelligence and Robotics 2020*, volume 11574, page 115740J. International Society for Optics and Photonics, 2020.

[8] Saifullah Tumrani, Zhiyi Deng, Abdullah Aman Khan, and Waqar Ali. Pevr: Pose estimation for vehicle re-identification. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 69–78. Springer, 2019.

[9] Ashir Javeed, Zhou Shijie, Abdullah Aman Khan, and Saifullah Tumrani. A robust method for averting attack scenarios in location based services. In *2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–6. IEEE, 2019.

[10] Abdullah Aman Khan, Saifullah Tumrani, Chunlin Jiang, and Jie Shao. RICAPS: residual inception and cascaded capsule network for broadcast sports video classification. In Tat-Seng Chua, Jingdong Wang, Qi Tian, Cathal Gurrin, Jia Jia, Hanwang Zhang, and Qianru Sun, editors, *MMAsia 2020: ACM Multimedia Asia, Virtual Event / Singapore, 7-9 March, 2021*, pages 43:1–43:7. ACM, 2020.

[11] Rehan Mehmood Yousaf, Saad Rehman, Hassan Dawood, Guo Ping, Zahid Mehmood, Shoaib Azam, and Abdullah Aman Khan. Saliency based object detection and enhancements in static images. In Kuinam Kim and Nikolai Joukov, editors, *Information Science and Applications 2017 - ICISA 2017, Macau, China, 20-23 March 2017*, volume 424 of *Lecture Notes in Electrical Engineering*, pages 114–123. Springer, 2017.

[12] Abdullah Aman Khan, Sidra Shafiq, Rajesh Kumar, Jay Kumar, and Amin Ul Haq. H3dnn: 3d deep learning based detection of covid-19 virus using lungs computed tomography. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 183–186. IEEE, 2020.

[13] Rajesh Kumar, Wenyong Wang, Jay Kumar, Ting Yang, Abdullah Aman Khan, Wazir Ali, and Ikram Ali. An integration of blockchain and AI for secure data sharing and detection of CT images for the hospitals. *Comput. Medical Imaging Graph.*, 87:101812, 2021.

[14] Waqar Ali, Wenhong Tian, Salah Ud Din, Desire Iradukunda, and Abdullah Aman Khan. Classical and modern face recognition approaches: a complete review. *Multim. Tools Appl.*, 80(3):4825–4880, 2021.

[15] Ali Waqar, Shao Jie, Abdullah Aman Khan, and Saifullah Tumrani. Context-aware recommender systems: challenges and opportunities. *Journal of Electronic Science and Technology*, 48(5):655–673, 2019.

[16] Waqas Amin, Qi Huang, M Afzal, Abdullah Aman Khan, Khalid Umer, and Syed Adrees Ahmed. A converging non-cooperative & cooperative game theory approach for stabilizing peer-to-peer electricity trading. *Electric Power Systems Research*, 183:106278, 2020.

[17] Waqas Amin, Qi Huang, Khalid Umer, Zhenyuan Zhang, M Afzal, Abdullah Aman Khan, and Syed Adrees Ahmed. A motivational game-theoretic approach for peer-to-peer energy trading in islanded and grid-connected microgrid. *International Journal of Electrical Power & Energy Systems*, 123:106307, 2020.

[18] Zahoor Ahmed, Hasan Zulfiqar, Abdullah Aman Khan, Ijaz Gul, Fu-Ying Dao, Zhao-Yue Zhang, Xiao-Long Yu, and Lixia Tang. ithermo: A sequence-based model for identifying thermophilic proteins using a multi-feature fusion strategy. *Frontiers in Microbiology*, page 82.

[19] Abdullah Aman Khan and Jie Shao. Spnet: A deep network for broadcast sports video highlight generation. *Comput. Electr. Eng.*, 99:107779, 2022.

[20] Saifullah Tumrani, Parivish Parivish, Abdullah Aman Khan, and Wazir Ali. Two stream pose guided network for vehicle re-identification. In *2021 3rd International Conference on Image Processing and Machine Vision (IPMV)*, pages 11–16, 2021.

[21] Ijaz Gul, Lizhu Aer, Min Zhang, Hanjia Jiang, Abdullah Aman Khan, Muhammad Bilal, Ruiqing Fang, Juan Feng, Hongjuan Zeng, and Lixia Tang. Multifunctional 3d-printed platform integrated with a smartphone ambient light sensor for halocarbon contaminants monitoring. *Environmental Technology & Innovation*, 24:101883, 2021.

[22] Abdullah Aman Khan. A generic scalable dft calculation method for vectors and matrices using matrix multiplications. 2022.

[23] Muhammad Husnain Haider, Hub Ali, Abdullah Aman Khan, Hao Zheng, M. Usman Maqbool Bhutta, Shaban Usman, Pengpeng Zhi, and Zhonglai Wang. Autonomous mobile robot navigation using adaptive neuro fuzzy inference system. In *2022 International Conference On Innovations And Development Of Information Technologies And Robotics (IDITR)*, 2022.

[24] Hui Xu, Pengpeng Zeng, and Abdullah Aman Khan. Multimodal interaction fusion network based on transformer for video captioning. *Comput. Electr. Eng.*, 2022.

[25] BY MICHAeL SIYANG LI. Keeping up with moore's law. *Dartmouth Undergraduate Journal of Science*, pages 20–23, 2013.

[26] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[27] David Burg and Jesse H Ausubel. Moore's law revisited through intel chip density. *PloS one*, 16(8):e0256245, 2021.

[28] Eric Monmasson and Marcian N Cirstea. Fpga design methodology for industrial control systems—a review. *IEEE transactions on industrial electronics*, 54(4):1824–1842, 2007.

[29] Stephen Brown and Jonathan Rose. Fpga and cpld architectures: A tutorial. *IEEE design & test of computers*, 13(2):42–57, 1996.

[30] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure dpa resistant asic or fpga implementation. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, volume 1, pages 246–251. IEEE, 2004.

[31] Wayne Wolf, Ahmed Amine Jerraya, and Grant Martin. Multiprocessor system-on-chip (mpsoc) technology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1701–1713, 2008.

[32] Varuna Eswer and Sanket Dessai. Processor performance metrics analysis and implementation for mips using an open source os. *International Journal of Reconfigurable and Embedded Systems*, 10(2):137, 2021.

[33] Sourov Roy. *Fault Diagnosis and Condition Monitoring of Power Electronic Components Using Spread Spectrum Time Domain Reflectometry (SSTDR) and the Concept of Dynamic Safe Operating Area (SOA)*. University of Missouri-Kansas City, 2021.

[34] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.