Embedded Systems Project Report

Group Members

Aaditya Baranwal (B20EE001)
Haardik Ravat (B20EE021)

Implementation: https://github.com/Haardik-Ravat/Embedded_ISA

Abstract

In this project, we developed a pipelined ARM-like processor in Verilog to execute a subset of instructions that included `ADD`, `SUB`, `MUL`, `MAC`, `LUDR`, `STUR`, `VADD`, `VSUB`, `VMUL`, `MOV`, `VMOV`, `VLD1`, `VST1`, `ADDS`, `SUBS`, `B`, `CBZ`, and `CBNZ`.

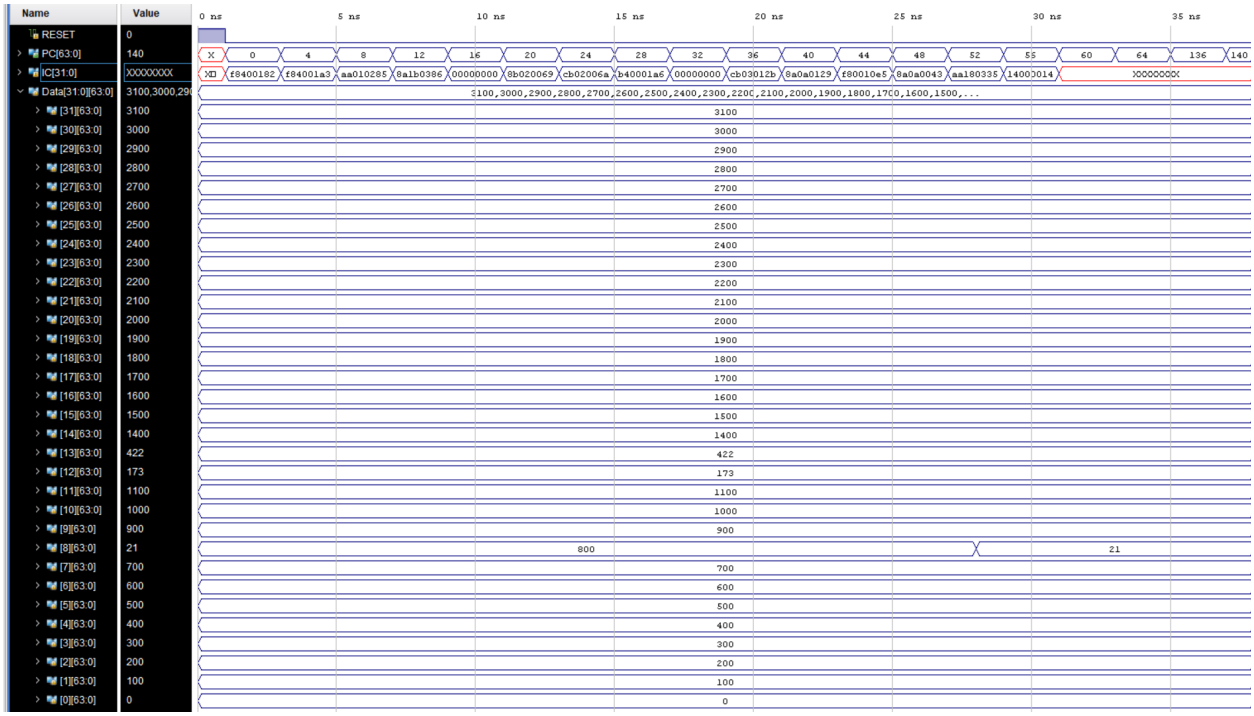
This project aimed to design and implement an ARM CPU for a custom(subset) instruction set architecture (ISA) based on the ARMv7 instruction set. The pipeline architecture included buffers and caches to optimize the execution of multiple instructions simultaneously. In addition to basic arithmetic instructions, the ISA included more complex instructions such as MAC (multiply and accumulate), LDUR (load unsigned register), VADD, VSUB, and VMUL (vector operations). The project also considered issues such as pipeline hazards and data memory management. Potential future scopes for the project include further optimization of pipeline stages and buffers/caches, incorporation of additional instructions and support for various data types, and potential optimization for machine learning applications.

Design

The processor was pipelined, allowing multiple instructions to run simultaneously. Buffers/caches were added between each stage of the processor's operation to facilitate this. This optimization reduced the amount of time it took to run each instruction.

We implemented an instruction memory consisting of 64 8-bit values for each index and a data memory consisting of 31 64-bit values to show that values could be accessed and stored via the CPU. The registers were initialized with values from 0-30, with

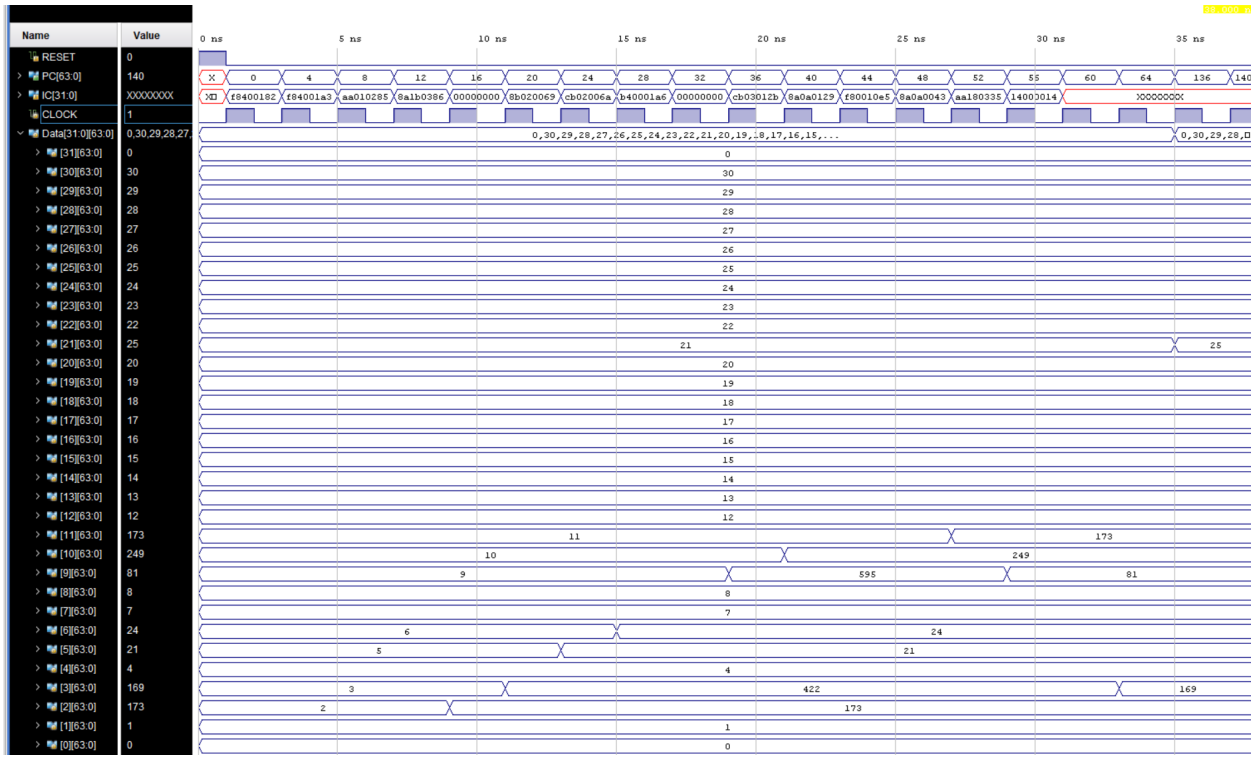
register 31 defined and set to 0. The data memory was initialized with values starting from 0-3100, with each memory content being 100 more than the previous index, except index 12 and 13, which had contents 173 and 422, respectively.



Data memory for all 32 computing registers

Implementation

We implemented the subset of instructions in Verilog. The `ADD`, `SUB`, and `MUL` instructions added, subtracted, and multiplied two operands, respectively, while the MAC instruction multiplied two operands and added the result to an accumulator. The `LUDR` instruction loaded a value from memory into a register, while the `STUR` instruction stored a value from a register in memory. The `VADD`, `VSUB`, and `VMUL` instructions added, subtracted, and multiplied two vectors. The ADDS and SUBS instructions were the same as the `ADD` and `SUB` instructions, respectively, but they also set flags to indicate whether the result was negative or zero. The `B` instruction unconditionally branched to an immediate value in memory. In contrast, the `CBZ` and `CBNZ` instructions branched to an immediate value in memory only if the specified register was zero or non-zero, respectively.



Register arrangement with clock sequencing.

CODEBASE

```
├── rtl/
│   ├── src/
│   │   ├── alu/
│   │   │   ├── alu.v
│   │   │   ├── control.v
│   │   │   └── mux.v
│   │   ├── control.v
│   │   ├── ic.v
│   │   ├── memory.v
│   │   ├── register.v
│   │   ├── helpers.v
│   │   ├── cpu.v
│   │   └── branch.v
│   └── sim/
│       └── testbench.v
├── doc/
```

```
│   ├── Rep.pdf
└── README
```

ADD

The ADD instruction in a processor's Instruction Set Architecture (ISA) allows two operands to be added together and the result stored in a destination register. In machine learning, the ADD instruction can be used in various ways. One example is in matrix addition, where the weights and inputs of a neural network layer are added together to produce an output.

SUB

The SUB instruction is used in the ISA of a processor to subtract two operands and store the result in a destination register. This operation is commonly used in mathematical computations that involve subtraction, such as calculating differences between values or computing gradients in optimization algorithms.

MUL

The MUL instruction is used in the ISA of a processor to multiply two operands and store the result in a destination register. This operation is commonly used in mathematical computations that involve multiplication, such as computing dot products, scaling matrices or tensors, and performing convolutions.

MUL instruction is a fundamental operation that is used extensively in neural networks. For example, in the feedforward computation of a neural network, the MUL instruction is used to compute the element-wise product between the weights and the inputs.

MAC

The MAC (Multiply and Accumulate) instruction is used in the ISA of a processor to perform multiplication and addition in a single instruction. Specifically, the MAC instruction multiplies two operands, adds the result to the value stored in a destination register, and stores the final result in the same destination register. This operation is commonly used in mathematical computations that involve dot products or matrix-vector multiplications.

In machine learning, MAC instruction is a crucial operation used extensively in various neural network architectures, such as fully connected and convolutional neural

networks. Specifically, the MAC instruction is used to compute the dot product between the weights and inputs and accumulate the result in a destination register. This operation is a performance bottleneck in many neural networks, and optimizing it is a critical challenge in hardware design.

LDUR

The LDUR (Load Register) instruction is used in the ISA of a processor to load a 64-bit value from memory into a destination register.

In machine learning, the LDUR instruction loads data into registers that are subsequently used in mathematical computations, such as matrix multiplications or activation functions. For example, the LDUR instruction can load the input feature maps into registers in a convolutional neural network before performing the convolution operation.

STUR

The STUR (Store Register) instruction is used in the ISA of a processor to store a 64-bit value from a source register in memory. This operation is commonly used in software programs that involve writing data to memory, such as file output, image rendering, or database management.

In machine learning, the STUR instruction stores the results of mathematical computations, such as the outputs of neural network layers or the gradients computed during backpropagation. For example, in a fully connected neural network, the STUR instruction can store the final layer's output in memory, which is subsequently used for prediction or evaluation.

VADD

The VADD (Vector ADD) instruction is used in the ISA of a processor to perform vector addition. Specifically, the VADD instruction takes two vectors of the same length as operands and adds the corresponding elements to produce a new vector of the same length.

In machine learning, the VADD instruction performs vectorized operations on large datasets, such as batch processing or parallel computing. For example, in a convolutional neural network, the VADD instruction can add the biases to the convolutional outputs in a vectorized manner, improving the network's performance.

VSUB

The VSUB (Vector SUBtract) instruction is used in the ISA of a processor to perform vector subtraction. Specifically, the VSUB instruction takes two vectors of the same length as operands and subtracts the corresponding elements to produce a new vector of the same length.

In machine learning, the VSUB instruction performs vectorized operations on large datasets, such as batch processing or parallel computing. For example, in a convolutional neural network, the VSUB instruction can subtract the means from the input feature maps in a vectorized manner, improving the network's performance.

VMUL

The VMUL (Vector MULtiply) instruction is used in the ISA of a processor to perform vector multiplication. Specifically, the VMUL instruction takes two vectors of the same length as operands and multiplies the corresponding elements to produce a new vector of the same length.

In machine learning, the VMUL instruction performs vectorized operations on large datasets, such as batch processing or parallel computing. For example, in a fully connected neural network, the VMUL instruction can compute the dot product between the input feature vector and the weight vector in a vectorized manner, improving the network's performance.

MOV

The MOV (Move) instruction is used in the ISA of a processor to move data from one register to another register. This instruction is often used in conjunction with arithmetic or logical operations to manipulate data within a program.

The MOV instruction can move data between different memory locations or registers, which is helpful for operations such as data preprocessing or data augmentation.

VMOV

The VMOV (Vector Move) instruction is used in the ISA of a processor to move data from one vector register to another vector register. This instruction moves entire data vectors in a single instruction, making it useful for performing vector operations such as vector addition, subtraction, and multiplication.

The VMOV instruction can move entire batches of data between different memory locations or registers, which is helpful for operations such as batch normalization or convolution.

VLD1

VLD1 (Vector Load 1) is an instruction that loads multiple consecutive elements from memory into a single vector register. The instruction takes a memory address as an argument, along with a vector register and the number of elements to load. The instruction assumes that the elements are stored consecutively in memory, with a fixed distance between them, and loads them in order into the vector register.

VST1

VST1 (Vector Store 1) is the inverse of VLD1, storing multiple consecutive elements from a vector register in memory. It also takes a memory address as an argument, a vector register and the number of elements to store. The instruction assumes that the elements will be stored consecutively in memory with a fixed distance between them.

ADDS

The ADDS (ADD with Set flags) instruction is used in the ISA of a processor to add two operands, similar to the ADD instruction. However, unlike the ADD instruction, the ADDS instruction updates the condition flags (negative, zero, carry, overflow) based on the addition result. This allows subsequent instructions to use these flags to make decisions or take different paths depending on the result of the addition.

In machine learning, the ADDS instruction is used in various numerical computations, such as linear algebra or statistical modelling. The condition flags can be used to make decisions or control the computation flow.

SUBS

The SUBS (SUBtract with Set flags) instruction is used in the ISA of a processor to perform the subtraction of two operands, similar to the SUB instruction. However, unlike the SUB instruction, the SUBS instruction updates the condition flags (negative, zero, carry, overflow) based on the result of the subtraction. This allows subsequent instructions to use these flags to make decisions or take different paths depending on the result of the subtraction.

In machine learning, the SUBS instruction is used in various numerical computations, such as linear algebra or statistical modelling. The condition flags can be used to make decisions or control the computation flow.

B

The B (Branch) instruction is used in the ISA of a processor to perform unconditional branching, where the program counter is set to a new memory location specified in the instruction. This allows the program to jump to a new location in memory and execute instructions from there.

The B instruction can be used in control flow constructs, such as loops or conditional statements, to change the flow of the program and perform different computations or tasks based on certain conditions or inputs.

CBZ

The CBZ (Compare and Branch if Zero) instruction is used in the ISA of a processor to perform conditional branching, where the program counter is set to a new memory location specified in the instruction only if a register contains a zero value. This allows the program to branch to a new location in memory if a particular condition is met and continue executing instructions from there.

It can be used to check for specific conditions or values in data and perform different computations or tasks based on them.

CBNZ

The CBNZ (Compare and Branch if Non-Zero) instruction is used in the ISA of a processor to perform conditional branching, where the program counter is set to a new memory location specified in the instruction only if a register contains a non-zero value. This allows the program to branch to a new location in memory if a specific condition is met and continue executing instructions from there.

It can be used to check for specific conditions or values in data and perform different computations or tasks based on them.

Future scope

- The scope for building efficient ISAs using subsets of the existing ARM ISA is enormous. ARM provides instruction set extensions, such as the ARMv8.2-A architecture, which includes the dot product extension that efficiently supports matrix multiplication. Other ARM extensions, like the integer dot product extension and the half-precision floating-point extension, can optimize the architecture for specific ML workloads.
- ARM processors can also be customized using hardware accelerators for convolution operations, which can significantly speed up the execution of convolutional neural networks. The ARM RISC ISA is an excellent choice for running ML algorithms on low-resource devices. Its efficiency, scalability, and specialization make it possible to perform complex calculations on devices with limited computational resources.

Conclusion

In this project, we developed a pipelined ARM-like processor in Verilog that executed a subset of instructions, including `ADD`, `SUB`, `MUL`, `MAC`, `LUDR`, `STUR`, `VADD`, `VSUB`, `VMUL`, `MOV`, `VMOV`, `VLD1`, `VST1`, `ADDS`, `SUBS`, `B`, `CBZ`, and `CBNZ`. We implemented instruction and data memories and initialized them with values to show that they could be accessed and stored via the CPU. We also implemented registers and initialized them with values from 0-30. The processor was pipelined to allow multiple instructions to run simultaneously, and we added buffers/caches between each stage.

Citations

- *Computer Organization and Design: The Hardware/Software Interface ARM Edition by D. Patterson and J. Hennessy, Morgan Kaufmann, 2016.*
- S. Liu et al., "Cambricon: An Instruction Set Architecture for Neural Networks," 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), Seoul, Korea (South), 2016, pp. 393-405, doi: 10.1109/ISCA.2016.42. https://ieeexplore.ieee.org/document/7551409
- https://cmosedu.com/jbaker/students/theses/Design and Implementation of an ISA and ISE Unit for the RZ9 Coprocessor System.pdf