FPGA-Based Hardware-Accelerated Matrix Multiplication on RISC-V Using Systolic Array Integration

Anirudh Mittal [1] Arjun A. Mallya [1] Shah Tirth [1] Joycee M. Mekie [1]

[1]Indian Institute of Technology Gandhinagar

Abstract

Matrix multiplication, particularly the naïve $O(n^3)$ algorithm, is computationally intensive and a bottleneck in many applications. This work integrates a custom `matmul` instruction into the **RV32IM RISC-V core**, offloading matrix operations to a hardware systolic array. Prototyped on a **Nexys4 DDR FPGA**, the system significantly accelerates computation by exploiting parallelism and pipelined data flow, demonstrating significant speedups for matrix-heavy workloads.

Introduction

RISC-V is an **open-source, modular ISA** known for its **simplicity** and **extensibility**. Its open-source instruction set architecture (ISA) allows designers to adapt and optimize the processor for a variety of domain-specific workloads. The standard RISC-V ISA defines a minimal set of aligned and compact instructions and provides extensive flexibility to introduce user-defined instruction extensions. This capability makes RISC-V especially well-suited for tailored instruction integration and experimentation with hardware accelerators.

Thus, we extend the **RISC-V ISA** with a **custom matmul instruction**. Offloaded to a **systolic array accelerator** and integrated on **FPGA**, this approach cuts execution time while still aligning with the **RISC-V pipeline**.

Design

The baseline architecture is an **open-source**, **5-stage pipelined RV32IM** core. To enable **hardware acceleration**, the design was modfied to incorporate a **Matmul Wrapper**, a **Systolic Array** and a **Memory Arbiter**. When the **matmul instruction's opcode** is detected in the **Decode stage**, the core pipeline is **stalled**. Control Signals are passed to the **Matmul Controller**, which orchestrates the fetching of matrix operands from **Data Memory**. These operands are then passed into the **Systolic Array** for **parallel computation**, with the final results **written back to memory**.

funct7	rs2	rs1	funct3	rd	opcode
0000000	Matrix B address	Matrix A address	000	Matrix C address	0001011

Figure 1. Instruction format of `matmul rd, rs1, rs2`

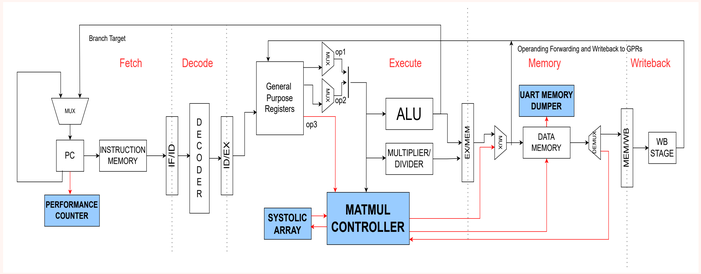
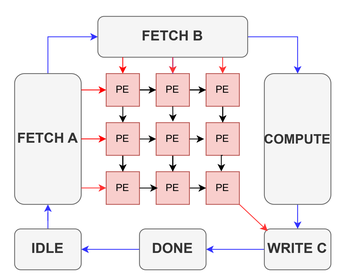


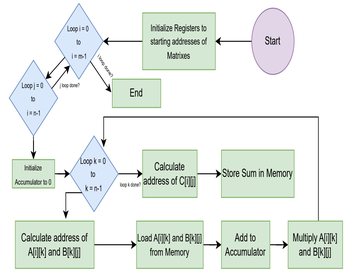
Figure 2. Implemented System Architecture and Systolic Array with RV32IM Core

Acknowledgement

We sincerely thank our mentor, **Prof. Joycee M. Mekie**, and the **IIT Gandhinagar Summer Research Internship Program** for their invaluable support and guidance.



(a) Systolic Array Integration with Matmul Controller



(b) Flow Diagram of mul-loop based multiplication on RISCV Core

Figure 3. Matrix Multiplication Comparison

Methodology and Implementation

The design process initiated with software-level modeling, where modifications were implemented in the **RISC-V GNU Toolchain** to incorporate the opcode and format for the `matmul` instruction. Next, the **Spike ISA Simulator** was enhanced by adding a **C functional model** of the instruction for verification purposes. Test programs, written in **C with inline assembly** for the `matmul` instruction, were compiled using the modified toolchain, resulting in a final `.hex` instruction file for **FPGA deployment**.

The processor was implemented on a **Xilinx Nexys4 DDR FPGA** utilizing **Vivado** software. A **performance counter** was used to profile execution cycles. **Functional correctness** was verified by transmitting results from the FPGA to a host PC via **UART**. **Timing violations** were addressed by employing the **Vivado Clock Wizard IP** to generate a stable **67 MHz system clock**, ensuring compliance with all timing constraints.

Table 1. Performance comparison of FPGA implementation with and without **matmul** for a 3×3 Matrix Multiplication over frequencies **67 MHz and 77 MHz** respectively

Sl. No.	Parameter	Value
1	Number of Clock Cycles	123
2	Time Period	15 ns
3	Total Clock Time	1,845 ns

(a) With MATMUL [Systolic Array]

Sl. No.	Parameter	Value
1	Number of Clock Cycles	785
2	Time Period	13 ns
3	Total Clock Time	10,205 ns

(b) Without MATMUL [Mul-Loops]

Results and Discussion

Following the successful **3x3 functional verification**, system performance was evaluated across varying matrix dimensions (*m*,*n*,*k*). Two key metrics were analyzed: **Execution Time (Speedup)** based on cycle counts and **On-Chip Power Consumption** estimated using the Vivado.

The table below compares **mul-loop based** and **hardware-accelerated** `matmul` implementations. The accelerator yields a substantial reduction in clock cycles, with **greater speedup at larger dimensions**, due to the **parallelism** of the systolic array architecture.

Table 2. Wall Clock Time (ns) vs. Matrix Dimensions

m,n,k	Cycles (Matmul)	Time (ns)	Cycles (Mul-Loop)	Time (ns)	Speedup
3,3,3	123	1845	785	10215	5.54
3,3,2	95	1330	564	7332	5.51
3,3,1	67	938	342	4104	4.38
2,2,3	85	1190	344	4472	3.76
2,2,2	71	923	266	3458	3.75
2,2,1	55	715	164	1968	2.75

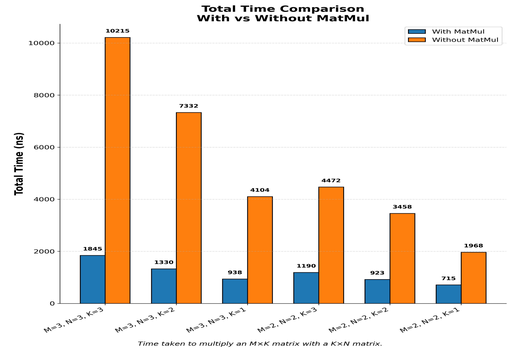


Figure 4. Bar Graph depicting the time consumption across different matrices

For power comparison, a common frequency of 67 MHz was taken, recording on-chip power of **0.27 W and 0.25 W** respectively, with and without matmul.

Conclusion

By extending the RV32IM core with a custom matmul instruction and offloading matrix operations to a **pipelined systolic array** on an FPGA, this work achieves substantial speedups and improved energy efficiency for **matrix heavy workloads**. The results validate the flexibility of the RISC-V ISA for domain specific acceleration and lay the groundwork for future enhancements in **compiler support** and experimentation with **multiple hardware accelerators**.

Future Scope

Future work includes integrating the accelerator into a full neural network pipeline and extending the RISC-V toolchain for automatic `matmul` substitution. We also aim to co-simulate with Spike for hardware-software testing.

References

[1] RISC-V International, "The official risc-v github organization." GitHub, 2024. Accessed on: 2024-07-02.

[2] G. Erfan, V. Bouac, A. Abdennebi, J.-M. Portal, and L. Torres, "A risc-v-based accelerator for general-purpose in-memory-computing with spintronic memristors," in *2024 IEEE 25th Latin American Test Symposium (LATS)*, pp. 1–6, IEEE, 2024.

[3] H. T. Kung and C. E. Leiserson, "Systolic arrays (for VLSI)," *Sparse Matrix Proceedings 1978*, pp. 256–282, 1979.