POLITECNICO DI TORINO
Department of Electronic Engineering

Speed-up of RISC-V core using Logic-in-Memory operations

Master Thesis Summary

Supervisors:
Marco Vacca - Politecnico di Torino
Marco Ottavi - Università degli Studi di Roma Tor Vergata

Candidate:
Antonia Ieva - s253237

Problem statement

The *memory-wall* is a known issue in modern computing systems, that states the difference in terms of speed between memory and processor in a typical Von-Neumann architecture. The memory low speed masks the actual processor speed, becoming the bottleneck of the communication between these units. To overcome this problem, many research works started moving towards new memory architectures that allow to increase the communication speed or to distribute part of the arithmetic computations in the memory itself.

Proposed solution

The Logic-in-Memory (LiM) is a new memory architecture that offers the possibility to have a unit that merges storage and computational capabilities. The literature offers many applications of the Logic-in-Memory concept, but this work aims to integrate the Logic-in-Memory architecture in a real-world computing system such as RI5CY. The RI5CY processor, part of the RISC-V family, has been chosen because it offers the possibility to customise the available Instruction Set Architecture (ISA) and then to add new instructions that support the new memory potentials.
Therefore, this thesis goal is to integrate a Logic-in-Memory into the RI5CY computing system and demonstrate the benefits in terms of programs execution time.

Implementation

The implemented Logic-in-Memory architecture can perform the following operations:

- *Normal load and store operations.* Since this new memory still behaves as a traditional memory, each load/store instruction takes 1 clock cycle.
- *Bitwise operations.* Load and store can be performed together with a bitwise operation, using an input mask, in just 1 clock cycle. The available bitwise operations are AND, OR and XOR.

Only in case of a store, the bitwise operation is supported also on a range of memory locations, assuming that the input mask to use is the same for all the locations selected. Bitwise operations are supported only on 32-bit data.

- *Maximum and minimum.* A special load operation can give the maximum or minimum value on certain range of memory locations. The duration of this memory operation is 33 clock cycles for any range selected. Maximum and minimum are computed considering 32-bit data values.

To integrate this new memory into the RI5CY core system, two solutions have been explored:

1. *Same Interface project.* In this solution, the RI5CY introduces a new ISA extension with instructions that coordinate the new memory operations. It maintains the original interface with the memory to prioritise the usage of the core in already existing systems. However, to perform a logic memory operation, the LiM needs to be configured with a store instruction that writes the logic operation type to execute, in a specific memory location. After the configuration, the memory will interpret next loads and stores as logic loads and stores.

2. *New Interface project.* This solution tries to maximise the efficiency of the memory operations, so the interface processor-memory is customised. The new ISA extension of the RI5CY contains instructions that manage the memory logic operations according to the new interface requirements. This LiM version does not need to be previously configured. In fact, the LiM behaviour is determined directly by the load or store instruction sent to the memory with additional control signals, that specify the memory operation to perform.

Both implementations show an important reduction of the execution time in almost all the tested programs. In particular, custom programs simulations resulted in 60%-80% speed increase, which has been interpreted as an upper bound of the possible LiM improvement. While, standard programs simulations resulted in an execution time reduction of 5%-15%. As expected, the solution with the customised interface guarantees better results because the memory configuration step is not needed. However, in order to have much more flexibility, the Logic-in-Memory should support additional operations in order to balance the workload between memory and processor. In this way, a wider range of programs would benefit from the new memory.

For the above reason, the Logic-in-Memory approach seems to be a promising option to overcome the microprocessor-memory communication bottleneck in Von-Neumann architectures. In fact, it would guarantee a less number of accesses in memory. As a consequence, there would be a reduction of the number of operations performed with the memory frequency and therefore a speed-up in the program executions.