SAPIENZA
UNIVERSITÀ DI ROMA

FACOLTÀ DI INGEGNERIA DELL' INFORMAZIONE INFORMATICA E STATISTICA
Corso di Laurea Magistrale in Ingegneria Elettronica

Digital Integrated System Architecture II

Hardware Accelerator For Matrix Addition

Students:

Pisani Marco, Marcelli Andrea, Frontera Federico

GitHub page: matrix_add_accelerator

Date of delivery: 28/07/2024

Indice

1 Introduction

This report outlines the project and technical specifications of a matrix addition acceleration unit. This unit is designed to implement matrix operations in hardware, utilizing a local memory subsystem. The unit interfaces with external main memory and a host processor, which sends commands to the unit, allowing direct data access from the main memory by the unit itself.
The acceleration unit interfaces with the main memory and the host processor through defined signals and protocols. These interfaces enable command transmission and data transfer between the unit, the main memory and the CPU.
The project target is to enhance the efficiency of matrix calculations by exploiting parallelization and direct memory access capabilities, therefore reducing the load on the cpu and optimizing the use of main memory.

2 Architecture and Parallelization

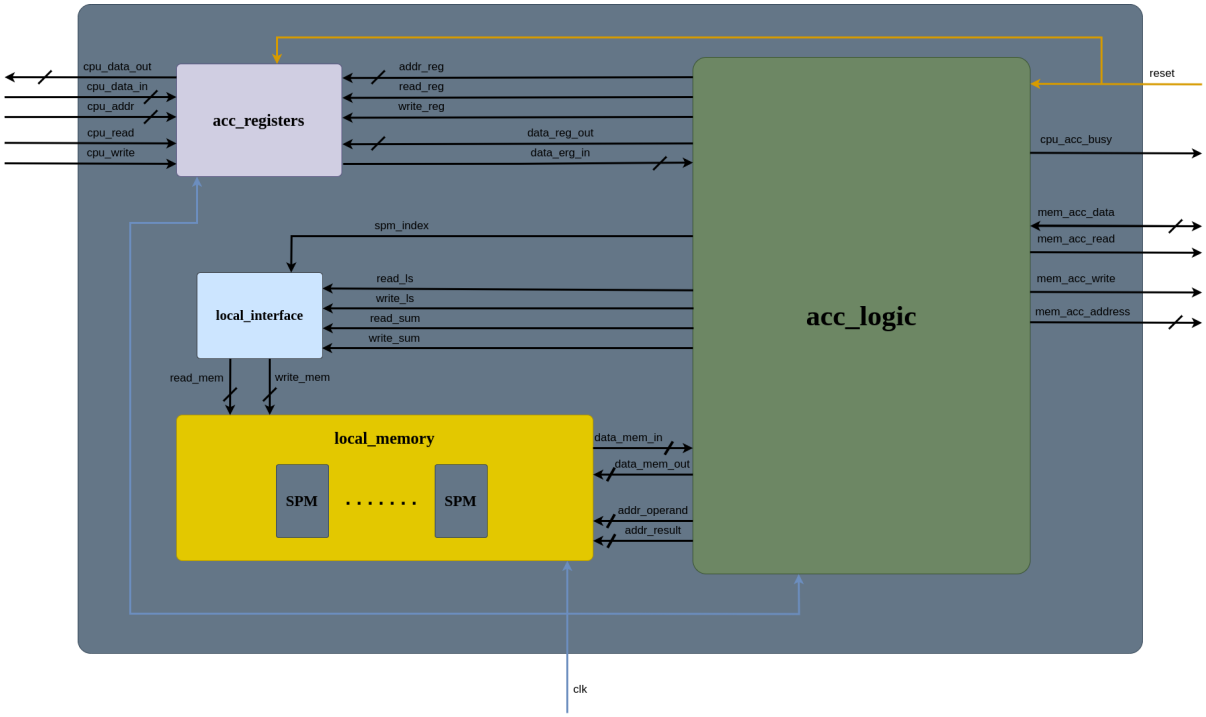


Figura 1: Block diagram of the accelerator (Accelerator_top.vhd)

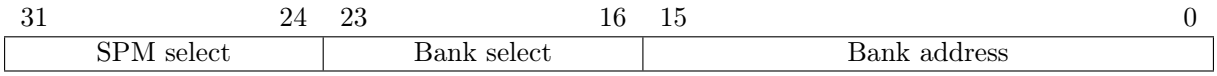
2.1 Functional description

The accelerator register file contains:

- **Current Status Register**
  Contains information about errors during the execution of instructions, such as invalid addresses
- **Instruction register**
  Instruction to be execute
- **Local memory address registers**
  Registers that contain addresses of the local memory.
- **Main memory address registers**
  Registers that contain addresses of the main memory.

The accelerator checks the instruction register at each clock cycle. Whenever the cpu writes an instruction, the accelerator clears the instruction register and starts decoding. Before writing the instruction the CPU should write the addresses of the matrices.

Addresses format:



When loading a matrix into the local memory, the matrix must start from the first SPM, so any address that has bits from 24 to 31 different from 0 will produce an error during the load/store instructions. Same principle applies during the add instruction for the result address.

The block *local_ interface* selects the SPM for the current instruction (one at a time for load/store and all together for add).

The block *local_ memory* is composed by a user-defined number of Scratch Pad Memories (SPM). There is 1 SPM for each adder and each adder can interact only with one SPM. The size of the SPMs can be changed by setting the number of banks it is composed of and the size of each bank. Each SPM has the same number of banks and each bank has the same size as the others.

2.2 Parameters

The accelerator can be customized by choosing the value of the following *GENERIC* in the top module:

- **SPM_NUM**
  Numbers of adders and SPM of the accelerator. Each adder is connected to an SPM and can only read and write from that SPM.
- **BANK_ADDR_WIDTH**
  Bits used for the address space of a single bank
- **SIMD**
  Number of banks for each SPM
- **N_RAM_ADDR**
  Number of registers that can contain an address of the local memory.
- **N_LOCAL_ADDR**
  Number of registers that can contain an address of the main memory

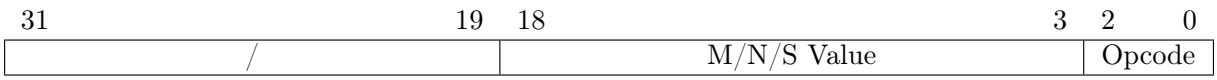
The total memory of the accelerator is:

$$\text{Local\_memory\_size} = 2^{\text{BANK\_ADDR\_WIDTH}} \cdot \text{SIMD} \cdot \text{SPM\_NUM} \cdot 4 \quad \text{[Bytes]}$$

3 Instruction Format

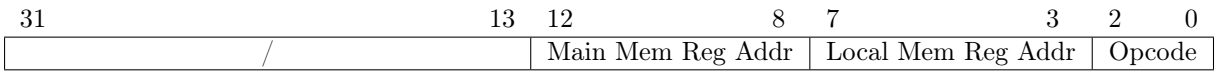
The supported commands are mainly six:

1. **Set matrix size M (rows):** The unit must be able to configure the dimensions M (number of rows) of the two matrices to be processed. The resulting matrix will also have M rows.
2. **Set matrix size N (columns):** The unit must be able to configure the dimensions N (number of columns) of the two matrices to be processed. The resulting matrix will also have N columns.
3. **Set stride value (S):** The stride value is used in the load and store matrix operations. It represents the space (number of memory locations) between the matrix elements in the main memory.

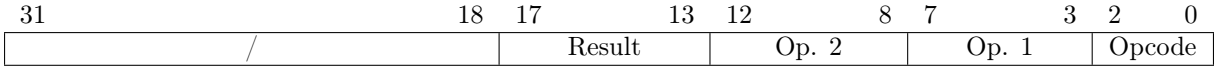


4. **Load a matrix from main memory to local memories:** This command uses addresses to indicate the matrix position in the memory and the stride value S to determine the intervals of access in the main memory.

5. **Write the result matrix from local memories to the main memory:** Similar to loading, this command uses addresses to indicate the matrix position in the main memory and the stride value S to determine the interval of access.



6. **Matrix addition in the local memory:** The unit performs the addition directly on the matrices loaded into the local memories. Addresses specify the position of the matrices in the local memories.



In the following table we present the Opcodes associated to each command:

Opcode	**Instruction Type**
100	SET M
101	SET N
110	SET S
001	LOAD
010	STORE
011	ADD

Tabella 1: Used Opcodes

4 Algorithmic State Machine Diagrams

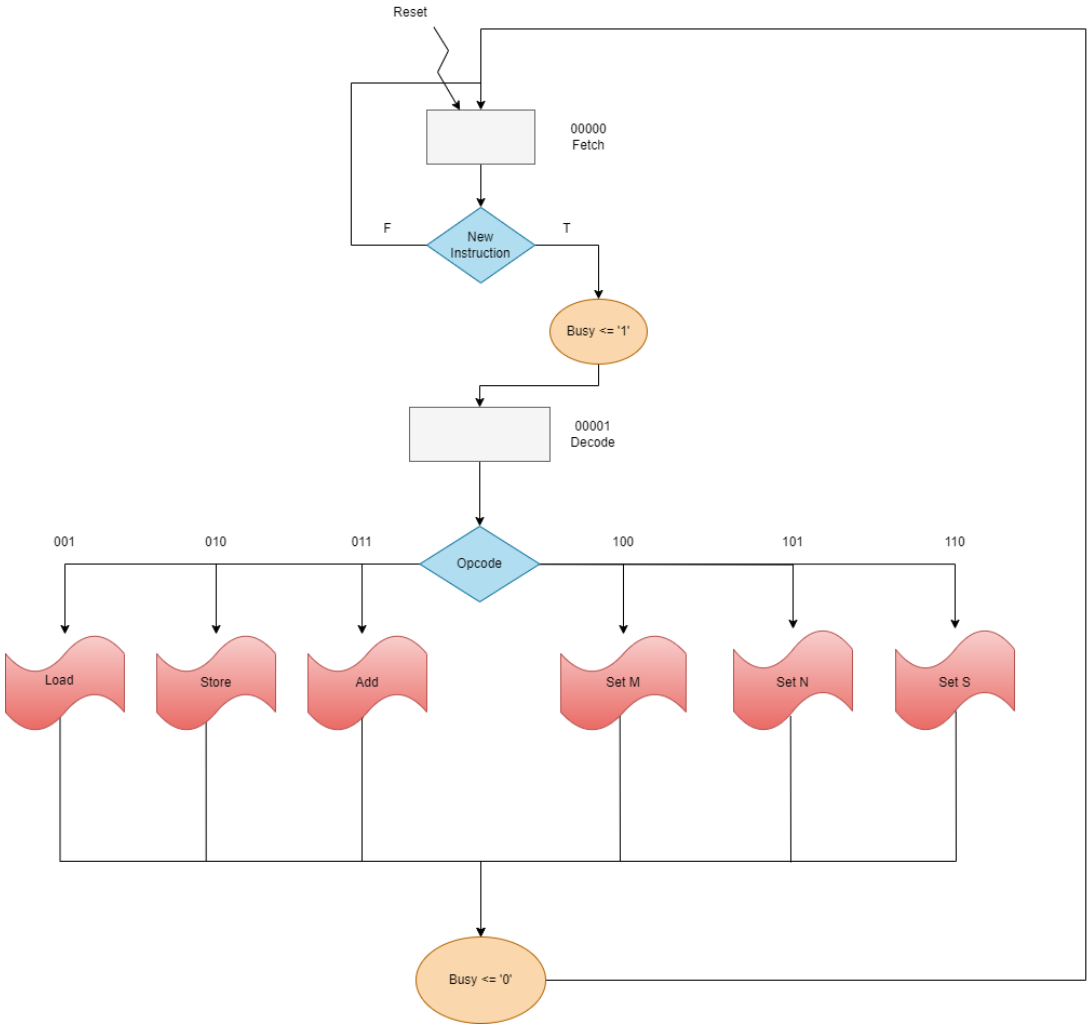


Figura 2: ASM diagram

4.1 Instructions ASM

SET_M asm diagram
SET_N asm diagram
SET_S asm diagram
LOAD asm diagram
STORE asm diagram
ADD asm diagram

5 Simulations

The file *testbench.vhd* contains the testbench to simulate each instruction. To load the main memory 2 files are needed containing the two matrices ("in1.txt" and "in2.txt"), one element each row; the output file ("out.txt") will contain the sum matrix. To properly launch the simulation the user needs to set the values of the constants of the *testbench.vhd* file:

Constant name	Meaning
c_spm_num	SPM_NUM
c_BANK_ADDR_WIDTH	BANK_ADDR_WIDTH
c_SIMD	SIMD
c_N_RAM_ADDR	N_RAM_ADDR
c_N_LOCAL_ADDR	N_LOCAL_ADDR
c_Dimensione	RAM size (each cells is 32 bits)
c_starting_addr_op1_ram	first matrix starting address in RAM
c_starting_addr_op2_ram	second matrix starting address in RAM
c_starting_addr_res_ram	result matrix starting address in RAM
c_M_dim	number of rows
c_N_dim	number of columns
c_S_val	Stride value
c_starting_addr_op1_local_mem	first matrix starting address in local memory
c_starting_addr_op2_local_mem	second matrix starting address in local memory
c_starting_addr_res_local_mem	result matrix starting address in local memory

5.1 Simulation results

Here are presented the waveform of a 4x4 matrix addition simulation:
SET_M SET_N SET_S simulation results
LOAD simulation results
STORE simulation results
ADD simulation results
The accelerator has been tested with very large matrices (400x300 elements) and has produced the correct results. Larger matrices were not tested due to hardware limitations.

6 Synthesis

The accelerator works properly in post-synthesis simulation with a clock up to 150 MHz without any timing violation. The file **testbench/tb_wrapper_verilog.v** contains a post-synthesis testbench that uses the constraint file **testbench/timing_constraints.xdc**. For this synthesis, the chosen parameters were: SPM_NUM = 2, SIMD = 2, BANK_ADDR_WIDTH = 6 due to hardware limitations.