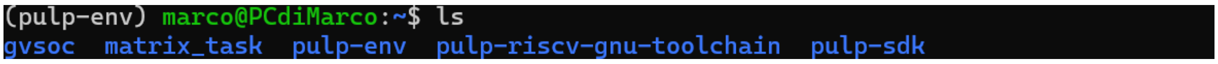
1 PULP Environment Setup and 2D Matrix Implementation

This report details the establishment of a development environment for the *Parallel Ultra-Low-Power* (PULP) platform on an Ubuntu 18.04 system, with a final execution of a 2D matrix multiplication ($C = A \times B$) targeting the `rv32imc` RISC-V architecture.

1.1 Environment Configuration and Conflict Resolution

The setup process was done by going through technical challenges. Some of them were identified and resolved to ensure a functioning integration:

- **Standalone GVSoC Module:** GVSoC is no longer bundled within the PULP-SDK as in previous versions, but exists as a *standalone* repository. It was cloned as a separate module.
- **Python Version Management:** GVSoC instructions were developed using "a fresh Ubuntu 22.04 (Jammy Jellyfish)", as stated in the documentation, and uses Python features (such as *type hinting* introduced in Python 3.9+) that are incompatible with the native Python 3.6 provided by Ubuntu 18.04. The solution was a standalone version of **Python 3.10** compiled from source and isolated within a virtual environment named `pulp-env`. This allowed the installation of modern dependencies (*psutil, rich, prettytable*) that were needed by GVSoC, withouth affecting any other installation such as pulp-toolchain and pulp-sdk.
- **SDK and PMSIS Integration:** The PULP-SDK was configured to link to rules identified in $PULP - SDK - HOME/tools/rules/pmsis_rules.mk$.



```
(pulp-env) marco@PCdiMarco:~$ ls
gvsoc  matrix_task  pulp-env  pulp-riscv-gnu-toolchain  pulp-sdk
```

Figure 1: Final downloaded environment

1.2 Kernel Implementation and Memory Limit Analysis

The primary objective was the implementation of a matrix multiplication kernel for the **512 KiB multi-banked L2 memory**. Data was explicitly allocated in the L2 section using the `PI_L2` attribute from the PMSIS API, and placement was verified by auditing physical addresses (mapped in the `0x1c000000` range).

1.2.1 L2 Maximum Memory Test

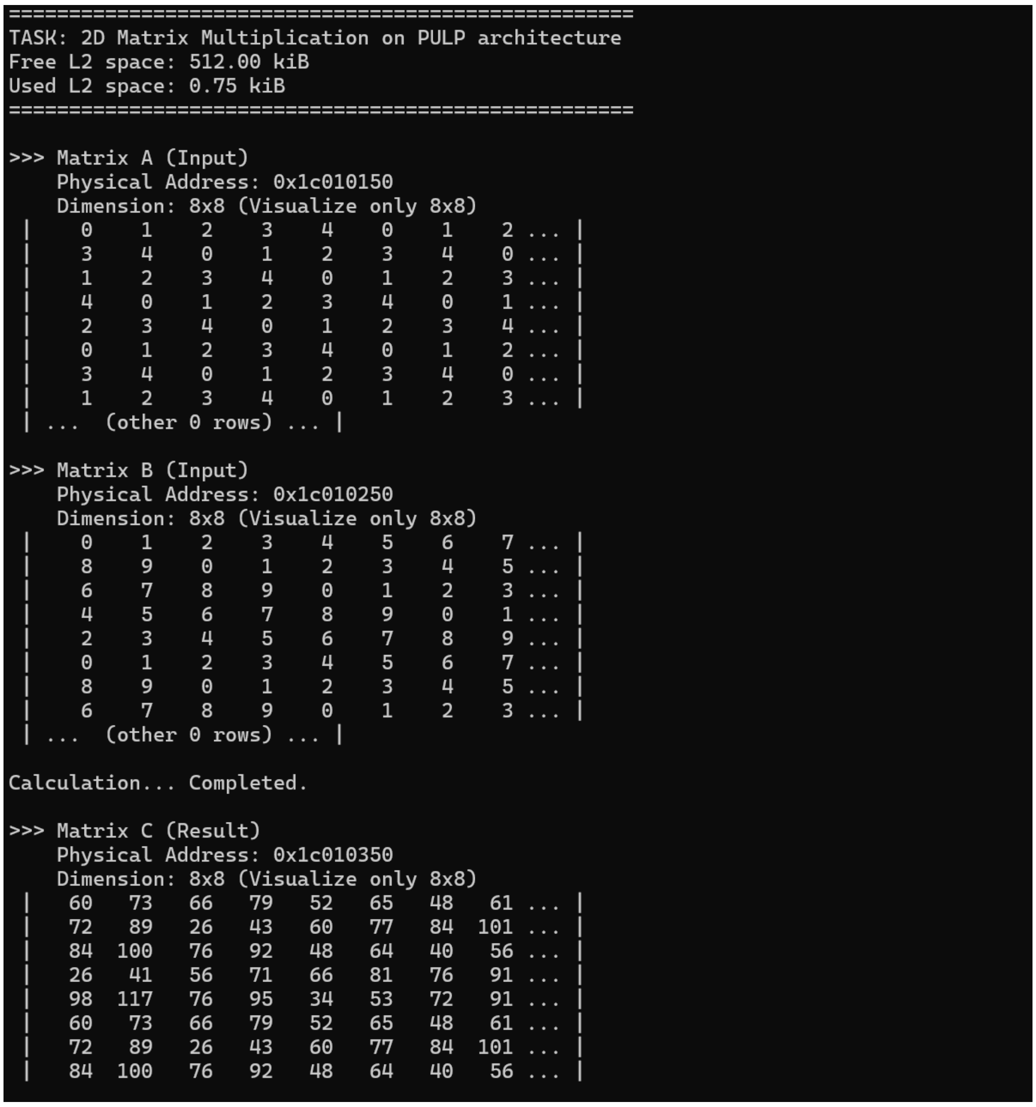
To fulfill the requirement of determining the architectural limits, the matrix dimension N for the square matrices (32-bit integers, 4 bytes per element) was incrementally increased. The total memory occupied by the three matrices (A, B, C) is defined by the formula: Memory_total $= 3 \times (N^2 \times 4)$ bytes. "3" is the number of matrices, N is the dimension and "4" are the bytes for each number.

```
PI_L2 int matA[N*N];
PI_L2 int matB[N*N];
PI_L2 int matC[N*N];
```

The maximum stable dimension achieved was 195×195.

At $\mathbf{N = 195}$, the data weight is approximately **445.60 KiB**. The remaining ≈ 66.4 KiB of the L2 memory successfully accommodated the executable code, runtime descriptors, and the system stack, preventing memory corruption or simulation crashes. The GVSoC simulator output successfully reported the expected values, validating both the algorithm and the environment stability.

A trial with a 8×8 matrices was done (figure 2), and the result is correct.



```
=====================================================
TASK: 2D Matrix Multiplication on PULP architecture
Free L2 space: 512.00 kiB
Used L2 space: 0.75 kiB
=====================================================

>>> Matrix A (Input)
    Physical Address: 0x1c010150
    Dimension: 8x8 (Visualize only 8x8)
 |    0    1    2    3    4    0    1    2 ... |
 |    3    4    0    1    2    3    4    0 ... |
 |    1    2    3    4    0    1    2    3 ... |
 |    4    0    1    2    3    4    0    1 ... |
 |    2    3    4    0    1    2    3    4 ... |
 |    0    1    2    3    4    0    1    2 ... |
 |    3    4    0    1    2    3    4    0 ... |
 |    1    2    3    4    0    1    2    3 ... |
 | ...  (other 0 rows) ... |

>>> Matrix B (Input)
    Physical Address: 0x1c010250
    Dimension: 8x8 (Visualize only 8x8)
 |    0    1    2    3    4    5    6    7 ... |
 |    8    9    0    1    2    3    4    5 ... |
 |    6    7    8    9    0    1    2    3 ... |
 |    4    5    6    7    8    9    0    1 ... |
 |    2    3    4    5    6    7    8    9 ... |
 |    0    1    2    3    4    5    6    7 ... |
 |    8    9    0    1    2    3    4    5 ... |
 |    6    7    8    9    0    1    2    3 ... |
 | ...  (other 0 rows) ... |

Calculation... Completed.

>>> Matrix C (Result)
    Physical Address: 0x1c010350
    Dimension: 8x8 (Visualize only 8x8)
 |   60   73   66   79   52   65   48   61 ... |
 |   72   89   26   43   60   77   84  101 ... |
 |   84  100   76   92   48   64   40   56 ... |
 |   26   41   56   71   66   81   76   91 ... |
 |   98  117   76   95   34   53   72   91 ... |
 |   60   73   66   79   52   65   48   61 ... |
 |   72   89   26   43   60   77   84  101 ... |
 |   84  100   76   92   48   64   40   56 ... |
```

Figure 2: 8×8 Matrix calculation output

The files included in this project are the `Makefile`, `matrix.c` and the executed log `exe_matirx.log` that contains a 195×195 multiplication.

1.3 Conclusion

The successful integration of the PULP toolchain and SDK was achieved by bypassing software incompatibilities through the use of isolated environments and modern host compilers. The "stress" test demonstrated that matrices up to 195×195 can be processed within the L2 memory constraints.