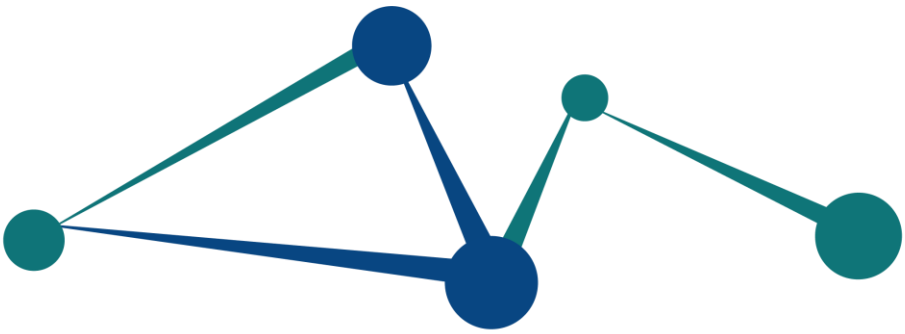

NEUROPULS

Deliverable 5.3

Memory hierarchy models and simulation components: iteration 1

Start date of the project: 1st January 2023

Duration 48 months

Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Document Classification

Document Title	D5.3 Memory hierarchy models and simulation components: iteration 1
Author(s)	P06 – BSC-CSN – Mikel Fernandez, P09 NKUA – George Papadimitriou, P01 CNRS - Fabio Pavanello – Ranjeet Dwivedi – Regis Orobtchouk
Work Package	WP5 – System Architecture Modelling and Simulation
Dissemination Level	PU = Public
Nature	R = Report
Doc ID Code	NEUROPULS_D5.3_BSC
Keywords	Simulation, gem5, NVMain, memory hierarchy, security

Document History

2024-06-30	Table of contents, structure defined, abstract and introduction	P06 BSC-CNS – Mikel Fernandez
2024-07-30	Background on gem5	P09 NKUA – George Papadimitriou
2024-09-01	Optical memory based on PCM devices	P01 CNRS– Fabio Pavanello – Ranjeet

	Document History
	Dwivedi – Regis Orobtchouk

Document Validation

Project Coordinator	P1 CNRS – Fabio Pavanello Fabio.pavanello@cnrs.fr
Date	2024-09-30

Document Abstract

This report details the development of a simulation infrastructure designed to enable a comprehensive evaluation of computing systems that integrate neuromorphic accelerators with primary storage subsystems, such as DDR (Double Data Rate) memory or cache. The infrastructure facilitates detailed system-level analysis, focusing on the computing platform's characteristics, particularly when incorporating photonic hardware and Phase-Change Memory (PCM), a type of non-volatile random-access memory.

This deliverable provides an overview of the methodology, tools, and key considerations involved in establishing the system architecture modeling and simulation infrastructure. It emphasizes the importance of this infrastructure in enhancing the understanding and optimization of advanced computing systems.

Table of contents

1. Introduction

1.1 Objectives

The primary objectives of this work package (WP5), and thus, of this deliverable, are the following:

a) **Development of Efficient Subsystem Simulation Tools for Main Memory:** The first objective is to create efficient simulation tools for evaluating the impact of neuromorphic technology when implemented as the Main Memory in the computer hierarchy. This involves simulating the Main Memory subsystem within the project's simulator and integrating advanced neuromorphic accelerators to model their behavior. The goal is to compare this technology with state-of-the-art and leading-edge technologies, thereby facilitating the evaluation of the innovative components being developed in this project.

b) **Development of Efficient Subsystem Simulation Tools for Cache Memory:** The second objective is similar to the first but focuses on the Cache memory subsystem. The goal is to efficiently simulate the Cache subsystem within the project's simulator, integrating advanced neuromorphic accelerators to model their behavior. By comparing this technology with existing leading-edge technologies, the project aims to evaluate the innovative components under development.

c) **Portable Implementation and Integration of Simulation Tools:** The third objective is to develop portable implementations of both simulation tools to allow the integration into the project's main simulator. This task aims to facilitate the integration ensuring compatibility with the project simulator.

1.2 Progress assessment

Task 2 at Work Package 5 (WP5_T2) is running from month 1 to month 39, with direct contribution to two deliverables: this one, “Memory hierarchy models and simulation components: iteration 1” (D5.3, due month 21) and “Memory hierarchy models and simulation components: iteration 2” (D5.4, due month 42). In this deliverable we report on the status of WP5_T2, which is ongoing at the time of writing. In this subsection we map the progress on the task to the objectives described in subsection 1.1.

Objective a) “*Development of Efficient Subsystem Simulation Tools for Main Memory*”: Progress at 90%. The development and testing of the memory simulator is almost ready for use.

Objective b) “*Development of Efficient Subsystem Simulation Tools for Cache Memory*”: Progress at 30%. The cache memory simulator is being updated in order to be usable in the context of the project.

Objective c) Portable Implementation and Integration of Simulation Tools: Progress at 25%. We have defined a common baseline version of the simulator to use for the project,

and we have shared the configuration and output parameters that will provide an interface to the memory and cache simulators to the project partners. Source code integration is still to be started until objectives a) and b) are finalized.

1.3 Deliverable Organization

The subsequent sections of this document are organized to provide a comprehensive understanding of the infrastructure development of WP5_T2.

2. Background & Related Work

The focus of task WP5_T2 is modeling the system memory hierarchy for systems that employ neuromorphic accelerators hardware (including those considered in the Use Cases of the project). The novel aspect of this task is the insertion of PCM memory models (with parameters coming from the design and characterization work in WP2-WP4) at different levels of the memory hierarchy and the evaluation of its contribution to the main aspects of the system: performance and power consumption which combined determine the energy efficiency of the system.

Conventional memory technologies, such as Dynamic Random-Access Memory (DRAM) and Static Random-Access Memory (SRAM), present some technological limitations regarding emerging requirements such as low density, high standby power, and reliability. As a response, Phase-Change Memory (PCM) has emerged as one of the most promising non-volatile memory (NVM) solutions. PCM cells utilize chalcogenide alloys, like Ge2–Sb2–Te5 (GST), to store information by modulating the material's resistance as it transitions between amorphous (high-resistance) and crystalline (low-resistance) states.

The PCM operation involves two key processes: the SET operation and the RESET operation. During the SET operation, representing a logical "1," the phase-change material undergoes crystallization. This process is triggered by heating the material through an electrical pulse or laser irradiation, causing the material to transition into a crystalline state once the temperature exceeds the crystallization threshold. Conversely, the RESET operation, coding a logical "0," involves applying a different electrical pulse followed by a rapid cessation or a brief exposure to lower laser heat, leading the material to revert to its amorphous state.

Adopting NVMs as a replacement for current DRAM-based main memory architectures offers substantial benefits, such as reduced total energy consumption while maintaining equivalent capacity. However, as feature sizes continue to shrink, supply voltages decrease, and on-chip density increases, computer systems are expected to become more vulnerable to both hard and transient errors.

Replacing SRAM-based on-chip caches with NVMs has the potential to significantly enhance system performance due to their larger capacity and reduced power consumption, attributed to the zero-standby leakage characteristic of NVMs.

State-of-the-art NVM simulation methods must consider the unique characteristics of NVMs, enabling user applications to tailor these simulations for customized interfaces. However, such customization imposes higher demands on developers. Architectural-level NVM simulators focus on hardware details, including memory cells and transistor sizes in physical designs.

2.1 gem5 Simulator

The gem5 simulator [1] is a widely adopted system-level computer architecture simulator, offering a flexible and modular platform for modeling and simulating various aspects of computer systems. Its primary purpose is to allow researchers and developers to investigate and assess new architectural concepts, system designs, and software optimizations within a simulated environment before deploying them on physical hardware. It supports cycle-level simulation for a diverse range of computer architectures, including x86, Arm, MIPS, RISC-V, and others. This makes it a versatile tool for researchers focused on different computer architectures at the system level, encompassing the microarchitecture, architecture, operating systems, and application layers of the computing stack. Gem5 is open-source and has become widely recognized in both academic and industrial research communities.

Gem5's architecture is modular and extensible, enabling users to easily customize and enhance the simulator to meet their specific requirements. It features models for various components, such as processors, memory hierarchies, caches, interconnects, and peripheral devices, offering a comprehensive environment for studying computer system behavior. Gem5 evolved from the M5 simulator [2] to overcome M5's limitations and create a more modular and extensible framework. Its design philosophy emphasizes modularity and extensibility, utilizing a component-based architecture that allows users to easily mix and match simulation components to accurately model different aspects of computer systems. The simulator includes detailed models of memory hierarchies, including caches, main memory, and storage, as well as support for on-chip and off-chip interconnects. This allows researchers to analyze the impact of different memory configurations and communication architectures on system performance. Gem5 is widely used in both academic and industrial research. It supports various studies, such as microarchitecture exploration, performance analysis, software development, and the validation of new architectural ideas before their implementation in actual hardware.

In the context of WP5 of the NEUROPULS project, we build a baseline gem5-based simulation infrastructure which will incorporate all simulation activities of WP5. Specifically, we build a microarchitecture-level simulation framework (as it is detailed presented in the deliverable D5.9) using the latest version of the gem5 simulator, which allows (i) deterministic, (ii) end-to-end, (iii) cycle-level execution of (iv) large workloads (v) on top of an operating system (i.e., full-system simulation); this combination is impossible at lower levels. This framework is built on top of the gem5 simulator and is extended to support the RISC-V ISA (instruction set architecture) along with accelerator models. More details can be found in the deliverables D5.9 and D5.1.

2.2 Memory hierarchy simulators

There are specific requirements for the simulation of PCM memory models that require specialized simulators. Unlike more widely used memory technologies such as DRAM or SRAM, NVM simulators are not as widely used and readily available and require some additional effort to integrate, analyze, and tune to the needs of the project. Moreover, the simulating cache memories and memory controllers require specific integration in different simulator components.

We integrate two specific simulation components into the gem5-based project simulator, NVMain for the cache memory and NVMInterface for the memory controller.

2.2.1 NVMain

NVMain is an external simulator tailored for Non-Volatile Memory (NVM) systems, which introduces two pivotal features that enhance its appeal for NVM memory modeling. Firstly, its modular design facilitates easy integration and toggling of various techniques aimed at addressing endurance and write-related challenges, as well as hybrid memory configurations, enabling thorough exploration of architectural design spaces. Secondly, NVMain stands out as the first main memory simulator to incorporate data values efficiently, which proves invaluable in assessing application-level implications of NVM designs, particularly those based on multi-level cell architectures.

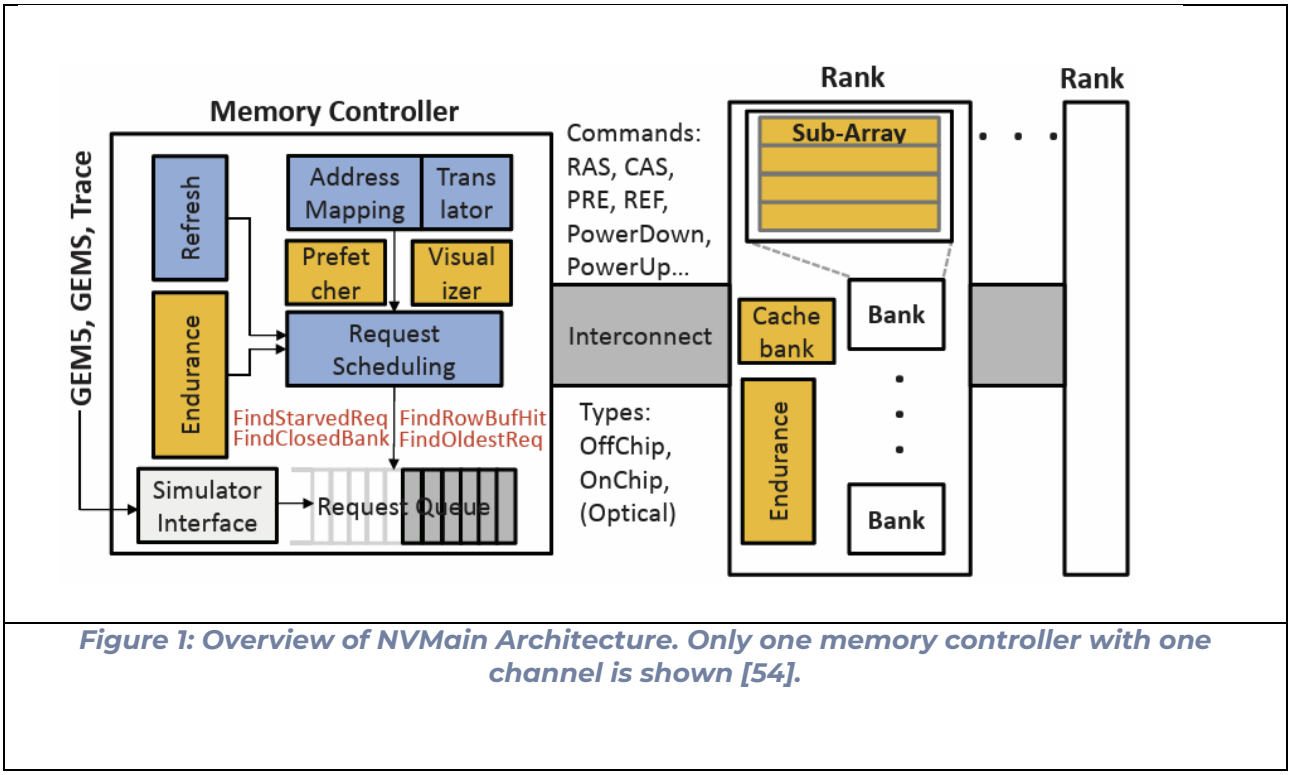

Figure 1: Overview of NVMain Architecture. Only one memory controller with one channel is shown [54].

One of the main reasons for the selection of NVMain is that it has been used for PCM cache memory simulation [3] and is one of the few works of this kind. However, NVMain is not readily available or integrated into the gem5-based project simulator and requires effort to port.

2.2.2 NVMInterface

NVMInterface is a GEM5 component which allows instantiation and accurate simulation of NVM memories. Its main difference with respect to NVMain is that it is readily available in the gem5-based project simulator, and it is designed to simulate the main system memory. To that end, it provides different configurable parameters and statistics to accurately model the device and collect metrics of its use. For example, some of these metrics include read or write bursts per bank, which only make sense in the context of DRAM but not for caches.

Although NVMInterface is already supported in GEM5 and can be configured as the memory type of the system, an effort has been made to test it with some configurations similar to those that may be required within the project, to ensure the component is ready to be used

3. Modeling oPCMs for memory hierarchy simulation

Task WP5_T2 focuses on identifying and configuring the simulation infrastructure required to validate and extrapolate the behavior of the optical chip. The first step involves selecting a suitable simulator from the available options that fit the project's requirements.

3.1 Alternatives considered

We build our memory hierarchy simulator with GEM5 as a base, and then add specific simulators capable of modeling the memory hierarchy with the required parameters. To select the best simulator for the caches and the main memory we first investigated the capabilities of the memory and cache simulators included in gem5, and then we investigated the different external simulators that were available.

Regarding the memory controller component Gem5 includes DRAMSim2 [4] and DRAMSim3 [5] simulators as components. Although DRAMSim3 supports STT-RAM it does not explicitly support PCMs.

Other memory simulators such as Ramulator [6] support PMCs and should be portable to gem5, however it is not integrated and is not part of the Gem5 package.

Since 2020 Gem5 supports NVM memories through the NVMInterface [7] component. This component, however, does not work for caches and is limited to simulating the

memory controller. NVMInterface has been selected as the memory controller simulator for task WP5_T2 in the project.

In order to simulate NVM caches we investigated the availability of external simulator and found NVMain [8] and NVMExplorer [9]. NVMain provides very extensive simulation metrics and, unlike NVMExplorer, presents some integration into older versions of Gem5, so it has been selected as the cache simulator for task WP5_T2.

3.2 NVMain

NVMain is the selected cache memory simulator, as it is a specialized framework tailored for Non-Volatile Memory (NVM) systems.

The NVMain configuration process involves two primary steps: integrating NVMain into the GEM5 simulator and identifying the parameters that need to be tailored to replicate the chip's behavior. Initially, NVMain was integrated into early versions of GEM5 to analyze the impact of Phase-Change Memory (PCM) technologies, such as Intel Optane, on systems where they were deployed. However, this integration was eventually abandoned in a fork of the NVMain project, leading to the obsolescence of the integration due to significant changes in the simulator's internal structure. Currently, the NVMain simulator integrated with GEM5 is only compatible with older versions of GEM5, which are incompatible with the newer architectures and improvements present in the GEM5-based project simulator. In this sense, the list of integrations that can found are:

https://github.com/SEAL-UCSB/NVmain (original)
https://github.com/cyjseagull/gem5-nvmain-hybrid-simulator (all in one repo)
https://github.com/SonodaKazuto/gem5-nvmain-docker (dockerized)

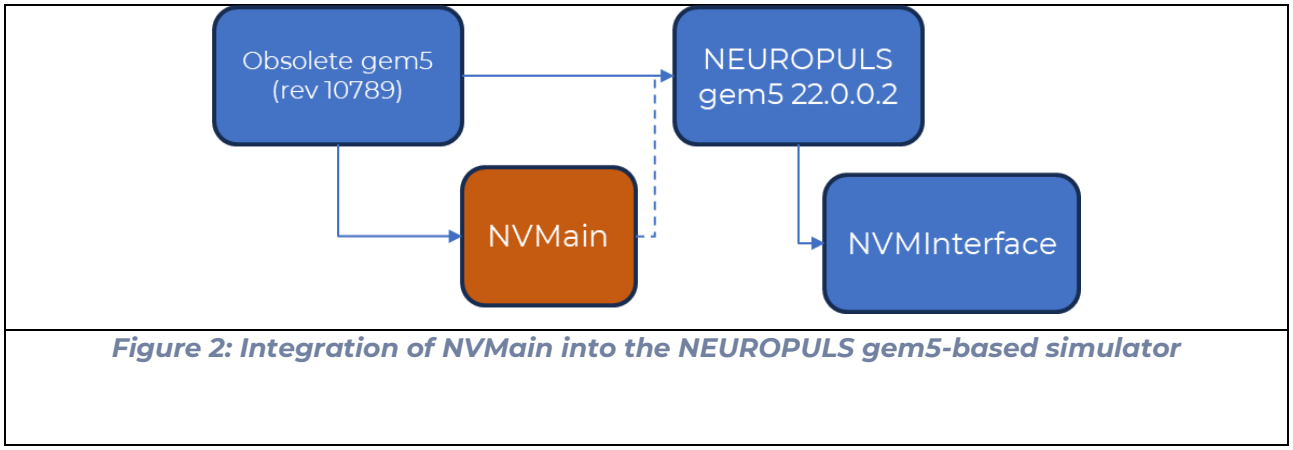

Figure 2: Integration of NVMain into the NEUROPULS gem5-based simulator

To address this issue, the Barcelona Supercomputing Center (BSC) undertook the task of integrating NVMain into the selected version of GEM5 (version 22.0.0.2). This process involved tracking the latest common commit between NVMain and the GEM5 main branch and applying patches to the source code to resolve integration issues related to that old commit. However, it was found that due to significant refactoring within the GEM5 code—particularly in the memory component, cache subsystem, and cache

protocol (previously based on Ruby, which has since been discontinued)—a clean merge was not possible. Consequently, the structural changes within GEM5 are carefully tracked to fully reintegrate NVMain.

Additionally, NVMain originally used Python 2, which is incompatible with GEM5's use of Python 3. BSC is making a substantial effort to update NVMain to be compatible with Python 3, thereby enabling its integration into GEM5.

In order to identify the parameters and statistics offered by the simulator, BSC integrated the obsolete version of NVMain and GEM5 allowing to identify the relevant configuration parameters that can imitate similar behaviors on NVM to those that may be required within the project. The following figures show an example of the X86 architecture where NVMain was integrated successfully on the simulator. The first figure shows a brief summary of the classes used on the example architecture simulation.

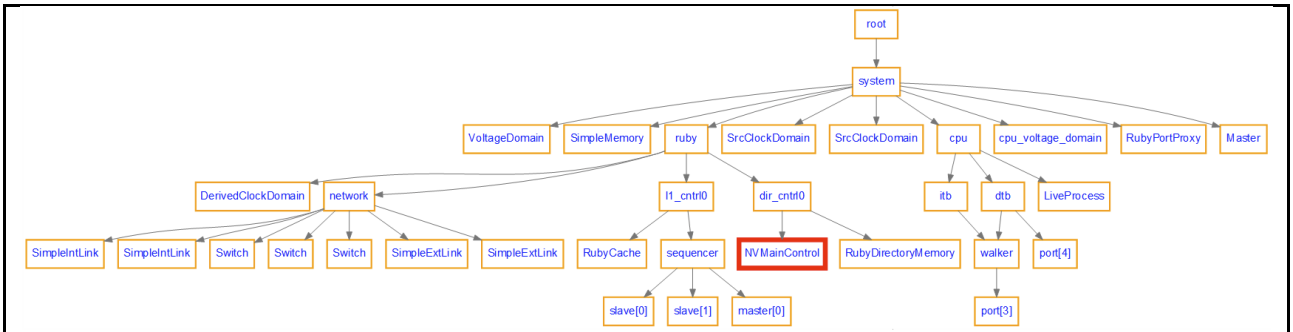

Figure 3: NVMain class diagram showing the architecture of the simulator

Notice that the integration happens on the cache subsystem based on the ruby subtree – dir_contrl0, where a memBuffer is instantiated. The system took the defaults values from the SE.py script file, while the ruby dir_contrl0 is tuned to use NVMainControl and takes its values as shown in the figure below.

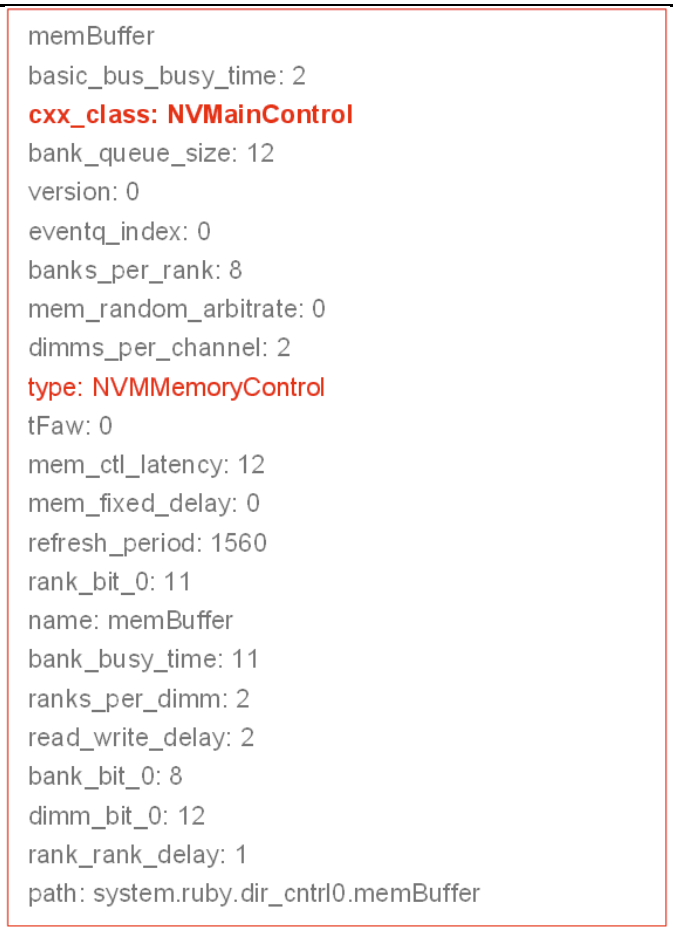

Figure 4: NVMain class showing its parameters

By using this approach, as from the integration of the NVMain onto the GEM5 simulator is expected to analyze the behavior of the optical non-volatile memories (oPCM) used in this project, due to NVMain simulator highly configurable interface, i.e., setting timings or energy needed to perform an operation. Also, it was obtained in this way, the list of statistics we will use compare the effect on performance or power consumption compared with other state of the art technologies. More information on parameter identification and analysis is presented in section 4.

3.3 NVMinterface

NVMInterface is a component within the GEM5 simulator that allows for the instantiation and accurate simulation of non-volatile memory (NVM) systems. Unlike NVMain, NVMInterface is readily available in GEM5 version 22.0.02 and is specifically designed to simulate the main system memory. It offers various configurable parameters and

provides statistics that accurately model the device and collect usage metrics. For example, some of these metrics include read or write bursts per bank, which are relevant in the context of DRAM but not for caches. A full list of available metrics is presented in section 4.2.

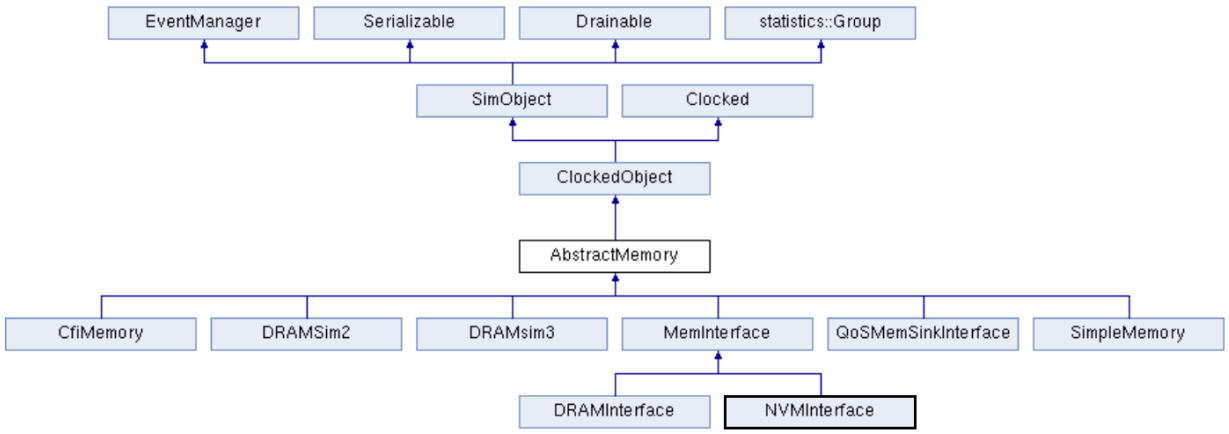

Figure 5: GEM5's AbstractMemory Class diagram, NVMInterface class highlighted [12]

Although NVMInterface is already supported in GEM5 and can be configured as the system's memory type, additional testing has been conducted using configurations similar to those required for the project. This ensures that the component is fully prepared for use in the project's specific context. To that end we simulated a RISCV system and used the example configuration 'configs/nvm/sweep.py' included in gem5.

```
scons build/RISCV/gem5.opt -j16
build/RISCV/gem5.opt configs/nvm/sweep.py
```

Figure 6: Code block showing how to compile and run a RISCV simulator using the nvm/sweep.py example configuration.

We then modified and tested different configurations to ensure the NVMInterface component could work with a baseline of possible combinations of parameters. The following ones were tested:

Parameter	Description	Parameters tested	Untested configurations

nvm_type	Type of memory to use	NVM_2400_1x64	CfiMemory,DDR3_1600_8x8,DDR3_2133_8x8,DDR4_2400_16x4,DDR4_2400_4x16,DDR4_2400_8x8,DRAMInterface,GDDR5_4000_2x32,HBM_1000_4H_1x128,HBM_1000_4H_1x64,HBM_2000_4H_1x64,HMC_2500_1x32,LPDDR2_S4_1066_1x32,LPDDR3_1600_1x32,LPDDR5_5500_1x16_8B_BL32,LPDDR5_5500_1x16_BG_BL16,LPDDR5_5500_1x16_BG_BL32,LPDDR5_6400_1x16_8B_BL32,LPDDR5_6400_1x16_BG_BL16,LPDDR5_6400_1x16_BG_BL32,NVMInterface,QoSMemSinkInterface,SimpleMemory,WideIO_200_1x128
nvm_ranks	Number of ranks to iterate across	1, 2, 4, 8	Higher powers of 2
rd_perc	Percentage of read commands	0, 50, 100	Other integers between 0 and 100
addr_map	NVM address map policy	RoRaBaCoCh, RoRaBaChCo, RoCoRaBaCh	-

With the purpose of enabling evaluation in the second iteration of the WP5_T2 task we also investigated different ways of connecting the NVMInterface component to the simulator. NVMInterface can be instantiated as a single memory controller, or as part of a hybrid memory controller where a DRAM interface is also created. The use of a hybrid memory controller requires the use of the HeteroMemCtrl and the use of both NVM and DRAM interfaces, which may present additional complexity.

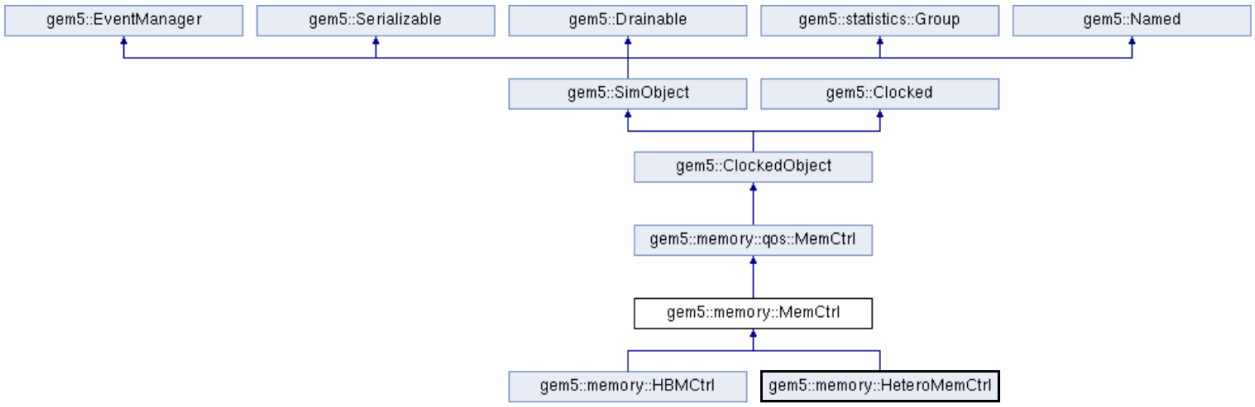

Figure 7: GEM5's MemoryCtrl Class diagram, HeteroMemCtrl class highlighted [12]

The use of HeteroMemCtrl has been tested and is an open possibility if the experimentation phase requires it, but preliminary it seems like configuring either a NVM or a DRAM interface is easier and reduces complexity, as well as removing interactions that may appear between the memory controllers if instantiated together. For that reason, we have made a small modification to the MemConfig component that allows a single NVM controller to be used.

```
--- a/configs/common/MemConfig.py
+++ b/configs/common/MemConfig.py
@@ -256,7 +256,7 @@ def config_mem(options, system):
                 # Create a controller if not sharing a channel with DRAM
                 # in which case the controller has already been created
                 if not opt_hybrid_channel:
-                    mem_ctrl = m5.objects.HeteroMemCtrl()
+                    mem_ctrl = nvm_intf.controller()
                     mem_ctrl.nvm = nvm_intf

                     mem_ctrls.append(mem_ctrl)
```

Figure 8: Diff patch showing the substitution of the HeteroMemCtrl for the NVMInterface controller.

4. Configurations and interfacing

A very big effort has been made to provide clear interfaces into the memory hierarchy simulator in order to ensure a smooth integration within the project. For each of the components involved, NVMain and NVMInterface, a deep analysis of their configurable parameters and their simulation metric outputs have been conducted. Providing a clear view of the available configurations and outputs provides a good view of what are the experimental setups that can be prepared, and what is the analysis that we can expect to conduct afterwards.

In this section we present a list of available configurations and metrics, and provide sample configuration values where possible.

4.1 NVMain

4.1.1 NVMain configuration

Four configuration files have been used as reference of what could be potentially useful parameters for a PCM cache. These NVMain configurations are based on a paper by Samsung [11] and comprise different possible variations for a PCM simulation.

Below we present the selected reference configuration and a summary of their configuration.

1. **Pcm.config**: A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth. Phase Change Memory has a very high resistance ratio, allowing for global sense amplifiers. This configuration assumes global sense amplifiers, which decrease area at the cost of increased tRCD.
2. **PCM_MLC_estimated.config**: Same as pcm.config with the following additional specifications: 256-bit row buffer based on up to 256-bit program width in each half, and each half is an independent bank. Each chip acts as a single x8 device with an LPDDR interface (not LPDDR-N).
3. **PCM_ISSCC_2012_4GB.config**: Same as PCM_MLC_estimated.config but also assumes each device has only 2 partitions to allow for banks in a 4GB memory system.
4. **PCM_MLC_example.config**: Similar to PCM_ISSCC_2012_4GB.config.

We have performed an in-depth analysis of the configurable parameters for NVMain, including source code analysis to identify any possible configurations. We have then matched these to the parameters used in the aforementioned reference configurations. We developed a taxonomy to classify the parameters and mark them as used or unused, and we note the reference values if any. Each parameter is classified as one of the following classes: General Memory, DRAM Energy, Energy parameters, General Geometry parameters, Endurance, Power Down, Simulation Control, Device Timing, Memory Controller, and MLC. In the table below we present all the parameters with a description and their value in each of the aforementioned configurations (1 to 4).

- General memory parameters

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
BPC	Number of bits per clock cycle	8	8	8	8
BusWidth	Bus width in bits. JEDEC standard is 64-bits	64	64	64	64
DeviceWidth	Number of bits provided by each device in a rank. Number of devices is calculated using BusWidth / DeviceWidth.	8	8	8	8
CLK	Clock (MHz). Example: 666 MHz clock (1333 MT/s DDR). Clock period = 1.5 ns	400	400	400	400
MULT	Multipler. 1 for DDR, 2 for DDR2, 4 for DDR3	4	8	8	8
RATE	Data Rate. 1 for SDR, 2 for DDR	2	2	2	2
CPUFreq	NVMain use CLK and CPUFreq to do the synchronization. The ratio CLK/CPUFreq is actually used. So a simple CLK=1 and CPUFreq=4 still works for simulation. However, it is straightforward to make it informative.	2000	2000	2000	2000
RefreshRows	RefreshRows is the number of rows to refresh per refresh operation. For example, RefreshRows = ROWS/8192 for DRAM	4	4	4	4
UseRefresh	Whether use refresh	false	false	false	false
StaggerRefresh					
UsePrecharge					
OffChipLatency					

- General geometry memory parameters

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
ROWS	Number of rows in one bank	32768	65536	16384	65536
COLS	Number of VISIBLE columns in one LOGIC bank	64	1024	1024	1024
CHANNELS	Number of channels in the system	4	1	1	1
RANKS	Number of ranks per channel	1	1	1	1
BANKS	Number of banks per rank	8	2	4	2
RAW	Row Activation Window (RAW), which specifies the maximum Activation in a rolling window. Options: * For 2D DRAM, RAW = 4, Four Activation Window (FAW). * For WideI/O DRAM, RAW = 2, Two Activation Window (TAW)	4	4	4	4
MATHeight	whether enable sub-array level parallelism (SALP). Options: No SALP: MATHeight = ROWS. SALP: number of subarrays = ROWS / MATHeight	32768	65536	16384	65536
RBSize					

- DRAM Energy

These values are unused in the context of the project.

Name	Description	Values
EIDD0	DRAM style power calculation. Values in mA, taken from datasheet.	-
EIDD1		
EIDD2P0		
EIDD2P1		
EIDD2N		
EIDD3P		
EIDD3N		
EIDD4R		
EIDD4W		
EIDD5B		
EIDD6		

- Energy Parameters

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
Eopenrd	Subarray write energy per bit	0.001616	0.001616	0.001616	0.001616
Erd	Memory device energy and power parameters. Read/write values are in nano Joules. NVSIM energy is per word. These values are the energy required to read a page into a row buffer. Erd is the read energy from a single mat. Other energy values are taken from CACTI	0.081200	0.081200	0.081200	0.081200
Eref	Subarray write energy per bit	0	0	0	0
Ewr	Memory device energy and power parameters. Read/write values are in nano Joules. Ewr is the write energy (SET or RESET, they are the same)	1.684811	1.684811	1.684811	1.684811
Ewrpb	Memory device energy and power parameters. Read/write values are in nano Joules. Ewrpb: Subarray write energy per bit.	0.000202	0.000202	0.000202	0.000202
Eactstdby	Subarray write energy per bit				
Eprestdby	Subarray write energy per bit				
Epda	Subarray write energy per bit	0	0	0	0
Epdpf	Subarray write energy per bit	0	0	0	0
Epdps	Subarray write energy per bit	0	0	0	0

Voltage	DRAM style power calculation.	1.5	1.5	1.5	1.5
Ereset	SLC energy		0.054331		0.054331
Eset	SLC energy		0.101581		0.101581
tWP0	SLC energy		50		40
tWP1	SLC energy		50		60
Eleak	Memory device energy and power parameters. Read/write values are in nano Joules. NVSIM energy is per word. These values are the energies required to read a page into a row buffer. Eleak: Energy leaked in 1 sec (or just the wattage) in milli Joules	3120.202	3120.202	3120.202	3120.202

- Endurance

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
EnduranceModel	Endurance model parameters. This is used for Non-volatile memory	NullModel	NullModel	NullModel	NullModel
DataEncoder					
EnergyModel	Options. current, energy. Defaults to current	energy	energy	energy	energy
EnduranceDist	EnduranceDistVariance	Normal	Normal	Normal	Normal
EnduranceDistMean		1000000	1000000	1000000	1000000
EnduranceDistVariance		100000	100000	100000	100000
EnduranceDistMean	Endurance model parameters. This is used for Non-volatile memory				
EnduranceDistVariance	Endurance model parameters. This is used for Non-volatile memory				
FlipNWriteGranularity	Endurance model parameters. This is used for Non-volatile memory				

- Power Down

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
UseLowPower	whether use low power mode?				

PowerDown Mode	PowerDown mode. Options: FASTEXIT: Precharge PowerDown with Fast Exit. SLOWEXIT: Precharge PowerDown with Slow Exit				
InitPD	whether set the DRAM in powerdown mode at the beginning?	false	false	false	false
tRDPDEN	Part of "powerdown mode enter and exit ". Interval between Read/ReadA and PowerDown	9	5	5	5
tWRPDEN	Part of "powerdown mode enter and exit ". Interval between Write and PowerDown	68	68	68	68
tWRAPDEN	Part of "powerdown mode enter and exit ". Interval between WriteA and PowerDown	68	68	68	68

- Simulation Control

These parameters configure the simulation behaviour, not the simulated memory.

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
PrintGraphs	Simulation control parameters	false	false	false	false
PrintAllDevices					
PrintConfig					
PeriodicStatsInterval	Simulation control parameters	100000000	100000000	100000000	100000000
PrintPreTrace	Simulation control parameters	false	false	false	false
EchoPreTrace	Simulation control parameters	false	false	false	false
TraceReader	Simulation control parameters				

- Device Timing parameters [13]

These configurations express values in number of cycles.

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
tAL	Additive Latency in memory clock cycles	0	0	0	0
tBURST	length of data burst	4	4	4	4
tCAS	tCAS is also known as tCL	5	1	1	1
tCCD	tCCD DRAM timing parameter. Number of cycles that must be allowed to elapse between successive column commands.	4	2	2	2
tCMD		1	1	1	1

tCWD	Delay between issuance of the column-write command and placement of data on the data bus by the memory controller	4	4	4	4
tRAW	tRAW is the corresponding window width (in cycle). See RAW.	20	20	20	20
tOST	ODT switching time	0	0	0	0
tPD	Part of "powerdown mode enter and exit "	1	1	1	1
tRAS	Defines the minimum time between ACT and PreCharge	17	0	0	0
tRCD	RAS# to CAS# Delay (tRCD). Controls the number of clocks inserted between a row activate command and a read or write command to that row.	22	48	48	48
tRDB					
tREFW	Refresh window between two refreshes to a cell (in cycle, e.g., 64ms/tCK). Options: * for DDR3, 64ms (normal) or 32ms (thermal extended). * for LPDDR3, 32ms (normal) or 16ms (thermal extended). * for 3D DRAM, 32ms/16ms/8ms are all possible	42666667	42666667	42666667	42666667
tRFC	Minimum time between ACT to Read	100	100	100	100
tRP	PreCharge to ACT same Bank	60	1	1	1
tRRDR	tRRDR is used for tRRD by default. TRRD defines the minimum ACT to ACT. Number of clocks between two row activate in different banks of the same rank (R)	2	4	4	4
tRRDW	Number of clocks between two row activate in different banks of the same rank (W)	11	4	4	4
tPPD					
tRTP	Read to Precharge delay. Number of clocks that are inserted between a read command to a row pre-charge command to the same rank.	3	3	3	3
tRTRS	for DDR-1, tRTRS can be 0	1	1	1	1
tWP		60	285	60	
tWR	This field determines the number of turn-around clocks on the data bus needs to be inserted between commands on different rank.	6	0	0	0
tWTR	Write to Read command Delay	3	3	3	3
tXP	Part of "powerdown mode enter and exit "	3	3	3	3
tXPDLL	Part of "powerdown mode enter and exit "	200000	200000	200000	200000
tXS	Part of "powerdown mode enter and exit "				
tXSDLL	Part of "powerdown mode enter and exit "				

- Memory Controller

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
ClosePage	Whether use close-page row buffer management policy? Options: 0--Open-Page, the row will be closed until a row buffer miss occurs. 1--Relaxed Close-Page, the row will be closed if no other row buffer hit exists. 2--Restricted Close-Page, the row will be closed immediately, no row buffer hit can be exploited	0	0	0	0
ScheduleScheme	Command scheduling policy. options: 0--fixed priority, 1--rank first round-robin, 2--bank first round-robin	2	2	2	2
HighWaterMark	Write drain high watermark	32	32	32	32
LowWaterMark	Write drain low watermark	16	16	16	16
BanksPerRefresh	the refresh granularity (the number of banks refreshed together in a rank). This must NOT be 0 when UseRefresh is true. [[the number of banks in a refresh (in lockstep)]]	2	2	2	2
DelayedRefreshThreshold	the number of refresh that can be delayed. Options: 1 -- 8 (DDR-3 defines the maximum 9*tREFI). When 1 is applied, immediate refresh is used, otherwise the refresh can be delayed. [[the threshhold tht indicates how many refresh can be delayed]]	1	1	1	1
AddressMappingScheme	address mapping scheme. options: SA:R:RK:BK:CH:C (SA-Subarray, R-row, C:column, BK:bank, RK:rank, CH:channel)	SA:R:RK:BK:C:CH	R:RK:BK:CH:C	R:RK:BK:CH:C	R:RK:BK:CH:C
MemoryPrefetcher					
PrefetchBufferSize					
MEM_CTL	Specify which memory controller to use. Options: PerfectMemory, FCFS, FRFCFS, FRFCFS-WQF, DRC (for 3D DRAM Cache)	FRFCFS	FRFCFS-WQF	FRFCFS	FRFCFS
CTL_DUMP	whether dump the memory request trace?				
INTERCONNECT	options: OffChipBus (for 2D), OnChipBus (for 3D)	OffChipBus	OffChipBus	OffChipBus	OffChipBus
ReadQueueSize	FRFCFS-WQF specific parameters. read queue size	32	32	32	32
WriteQueueSize	FRFCFS-WQF specific parameters. write queue size				

HighWaterMark	FRFCFS-WQF specific parameters. write drain high watermark. write drain is triggerred if it is reached				
LowWaterMark	FRFCFS-WQF specific parameters. write drain low watermark. write drain is stopped if it is reached				
WriteQueueSize	write queue size	32	32	32	32
RBsize			4	256	256

- Multi-level cell (MLC) configuration

		Configuration Values			
Name	**Description**	**1**	**2**	**3**	**4**
nWP00	2-level MLC average program pulse count			3	0
nWP01	2-level MLC average program pulse count			3	7
nWP10	2-level MLC average program pulse count			3	5
nWP11	2-level MLC average program pulse count			3	1
WPMaxVariance	2-level MLC variance (01 and 10 only).				

In the table below we present a summary showing only configured parameters, with the parameter value cell shaded in green if it is common to all configurations. We omit the description of each parameter; the reader may refer to the previous tables for more information.

Name	**Configuration values**
BPC	8
BusWidth	64
DeviceWidth	8
CLK	400
MULT	4 or 8
RATE	2
CPUFreq	2000
Eopenrd	0.001616
Erd	0.081200
Eref	0
Ewr	1.684811
Ewrpb	0.000202
Epda	0
Epdpf	0
Epdps	0

Voltage	1.5
EnduranceModel	NullModel
EnergyModel	energy
InitPD	false
PrintGraphs	false
PrintPreTrace	false
EchoPreTrace	false
RefreshRows	4
UseRefresh	false
PeriodicStatsInterval	100000000
ROWS	16384, 32768, or 65536
COLS	64 or 1024
CHANNELS	1 or 4
RANKS	1
BANKS	2, 4 or 8
RAW	4
MATHeight	16384, 32768 or 65536
tAL	0
tBURST	4
tCAS	1 or 5
tCCD	2 or 4
tCMD	1
tCWD	4
tRAW	20
tOST	0
tPD	1
tRAS	0 or 17
tRCD	22 or 48
tREFW	42666667
tRFC	100
tRP	1 or 60
tRRDR	2 or 4
tRRDW	4 or 11
tRTP	3
tRTRS	1
tWP	60, 285, or left unconfigured
tWR	0 or 6
tWTR	3
tXP	3
tXPDLL	200000
tRDPDEN	5 or 9
tWRPDEN	68

tWRAPDEN	68
ClosePage	0
ScheduleScheme	2
HighWaterMark	32
LowWaterMark	16
BanksPerRefresh	2
DelayedRefreshThreshold	1
AddressMappingScheme	SA:R:RK:BK:C:CH or :RK:BK:CH:C
MLCLevels	2 or left unconfigured
WPVariance	0, 1 or left unconfigured
UniformWrites	True, false or left unconfigured
Ereset	0.054331 or left unconfigured
Eset	0.101581 or left unconfigured
tWP0	40, 50 or left unconfigured
tWP1	50, 60 or left unconfigured
nWP00	0, 3 or left unconfigured
nWP01	3, 7 or left unconfigured
nWP10	3, 5 or left unconfigured
nWP11	1, 3 or left unconfigured
MEM_CTL	FRFCFS or FRFCFS-WQF
INTERCONNECT	OffChipBus
ReadQueueSize	32
EnduranceDist	Normal
EnduranceDistMean	1000000
EnduranceDistVariance	100000
Eleak	3120.202
WriteQueueSize	32
PreTraceFile	pcm.trace
IgnoreData	True or left unconfigured
RBsize	4, 256 or left unconfigured
TraceReader	NVMainTrace

4.1.2 NVMain output

After successful program execution, a statistics file is generated by the simulator with the relevant information. We provide 2 tables when use the classical cache architecture (non-ruby), the first one with the first level caches (L1I for instruction and L1D for data) and second level cache (L2), while the second table refers to the structures in charge of the address translation in the CPU for data and instructions (dtb and itb), the so called TLBs. For the sake of simplicity, the main stats found for each group have been summarized in each table, meaning that some of them can be applied to a specific type

of resource. For instance, ReadEx only applies to L2 cache level, while WriteReq only applies to L1-dcache level.

Cache	Description
tags.pwrStateResidencyTicks.UNDEFINED	Cumulative time (in ticks) in various power states
tags.tagsinuse	Cycle average of tags in use
tags.total_refs	Total number of references to valid blocks.
tags.sampled_refs	Sample count of references to valid blocks.
tags.avg_refs	Average number of references to valid blocks.
tags.warmup_cycle	Cycle when the warmup percentage was hit.
tags.occ_blocks.inst	Average occupied blocks per requestor
tags.occ_blocks.data	Average occupied blocks per requestor
tags.occ_percent.inst	Average percentage of cache occupancy
tags.occ_percent.data	Average percentage of cache occupancy
tags.occ_percent.total	Average percentage of cache occupancy
tags.occ_task_id_blocks.1024	Occupied blocks per task id
tags.age_task_id_blocks_1024.0	Occupied blocks per task id
tags.age_task_id_blocks_1024.1	Occupied blocks per task id
tags.occ_task_id_percent.1024	Percentage of cache occupancy per task id
tags.tag_accesses	Number of tag accesses
tags.data_accesses	Number of data accesses
pwrStateResidencyTicks.UNDEFINED	Cumulative time (in ticks) in various power states
WritebackClean_hits.writebacks	number of WritebackClean hits
WritebackClean_hits.total	number of WritebackClean hits
ReadCleanReq_hits.inst	number of ReadCleanReq hits
ReadCleanReq_hits.total	number of ReadCleanReq hits
.WriteReq_hits.data	number of WriteReq hits
.WriteReq_hits.total	number of WriteReq hits
demand_hits.inst	number of demand (read+write) hits
demand_hits.total	number of demand (read+write) hits
overall_hits.inst	number of overall hits
overall_hits.total	number of overall hits
ReadExReq_misses.data	number of ReadExReq misses
ReadExReq_misses.total	number of ReadExReq misses
.WriteReq_misses.data	number of WriteReq misses
.WriteReq_misses.total	number of WriteReq misses
ReadCleanReq_misses.inst	number of ReadCleanReq misses
ReadCleanReq_misses.total	number of ReadCleanReq misses
ReadSharedReq_misses.data	number of ReadSharedReq misses
ReadSharedReq_misses.total	number of ReadSharedReq misses
demand_misses.inst	number of demand (read+write) misses

demand_misses.data	number of demand (read+write) misses
demand_misses.total	number of demand (read+write) misses
overall_misses.inst	number of overall misses
overall_misses.data	number of overall misses
overall_misses.total	number of overall misses
ReadExReq_miss_latency.data	number of ReadExReq miss cycles
ReadExReq_miss_latency.total	number of ReadExReq miss cycles
.WriteReq_miss_latency.data	number of WriteReq miss cycles
.WriteReq_miss_latency.total	number of WriteReq miss cycles
ReadCleanReq_miss_latency.inst	number of ReadCleanReq miss cycles
ReadCleanReq_miss_latency.total	number of ReadCleanReq miss cycles
ReadSharedReq_miss_latency.data	number of ReadSharedReq miss cycles
ReadSharedReq_miss_latency.total	number of ReadSharedReq miss cycles
demand_miss_latency.inst	number of demand (read+write) miss cycles
demand_miss_latency.data	number of demand (read+write) miss cycles
demand_miss_latency.total	number of demand (read+write) miss cycles
overall_miss_latency.data	number of overall miss cycles
overall_miss_latency.total	number of overall miss cycles
WritebackClean_accesses.writebacks	number of WritebackClean accesses(hits+misses)
WritebackClean_accesses.total	number of WritebackClean accesses(hits+misses)
ReadExReq_accesses.data	number of ReadExReq accesses(hits+misses)
ReadExReq_accesses.total	number of ReadExReq accesses(hits+misses)
.WriteReq_accesses.data	number of WriteReq accesses(hits+misses)
.WriteReq_accesses.total	number of WriteReq accesses(hits+misses)
ReadCleanReq_accesses.inst	number of ReadCleanReq accesses(hits+misses)
ReadCleanReq_accesses.total	number of ReadCleanReq accesses(hits+misses)
ReadSharedReq_accesses.data	number of ReadSharedReq accesses(hits+misses)
ReadSharedReq_accesses.total	number of ReadSharedReq accesses(hits+misses)
demand_accesses.inst	number of demand (read+write) accesses
demand_accesses.data	number of demand (read+write) accesses
demand_accesses.total	number of demand (read+write) accesses
overall_accesses.inst	number of overall (read+write) accesses
overall_accesses.data	number of overall (read+write) accesses
overall_accesses.total	number of overall (read+write) accesses
ReadExReq_miss_rate.data	miss rate for ReadExReq accesses
ReadExReq_miss_rate.total	miss rate for ReadExReq accesses
.WriteReq_miss_rate.data	miss rate for WriteReq accesses
.WriteReq_miss_rate.total	miss rate for WriteReq accesses
ReadCleanReq_miss_rate.inst	miss rate for ReadCleanReq accesses
ReadCleanReq_miss_rate.total	miss rate for ReadCleanReq accesses

ReadSharedReq_miss_rate.data	miss rate for ReadSharedReq accesses
ReadSharedReq_miss_rate.total	miss rate for ReadSharedReq accesses
demand_miss_rate.inst	miss rate for demand accesses
demand_miss_rate.data	miss rate for demand accesses
demand_miss_rate.total	miss rate for demand accesses
overall_miss_rate.inst	miss rate for overall accesses
overall_miss_rate.data	miss rate for overall accesses
overall_miss_rate.total	miss rate for overall accesses
ReadExReq_avg_miss_latency.data	average ReadExReq miss latency
ReadExReq_avg_miss_latency.total	average ReadExReq miss latency
.WriteReq_avg_miss_latency.data	average WriteReq miss latency
.WriteReq_avg_miss_latency.total	average WriteReq miss latency
ReadCleanReq_avg_miss_latency.inst	average ReadCleanReq miss latency
ReadCleanReq_avg_miss_latency.total	average ReadCleanReq miss latency
ReadSharedReq_avg_miss_latency.data	average ReadSharedReq miss latency
ReadSharedReq_avg_miss_latency.total	average ReadSharedReq miss latency
demand_avg_miss_latency.inst	average overall miss latency
demand_avg_miss_latency.data	average overall miss latency
demand_avg_miss_latency.total	average overall miss latency
overall_avg_miss_latency.inst	average overall miss latency
overall_avg_miss_latency.data	average overall miss latency
overall_avg_miss_latency.total	average overall miss latency
blocked_cycles.no_mshrs	number of cycles access was blocked
blocked_cycles.no_targets	number of cycles access was blocked
blocked.no_mshrs	number of cycles access was blocked
blocked.no_targets	number of cycles access was blocked
avg_blocked_cycles.no_mshrs	average number of cycles each access was blocked
avg_blocked_cycles.no_targets	average number of cycles each access was blocked
writebacks	number of writebacks
ReadExReq_mshr_misses.data	number of ReadExReq MSHR misses
.WriteReq_mshr_misses.data	number of WriteReq MSHR misses
.WriteReq_mshr_misses.total	number of WriteReq MSHR misses
ReadExReq_mshr_misses.total	number of ReadExReq MSHR misses
ReadCleanReq_mshr_misses.inst	number of ReadCleanReq MSHR misses
ReadCleanReq_mshr_misses.total	number of ReadCleanReq MSHR misses
ReadSharedReq_mshr_misses.data	number of ReadSharedReq MSHR misses
ReadSharedReq_mshr_misses.total	number of ReadSharedReq MSHR misses
demand_mshr_misses.inst	number of demand (read+write) MSHR misses
demand_mshr_misses.data	number of demand (read+write) MSHR misses
demand_mshr_misses.total	number of demand (read+write) MSHR misses

overall_mshr_misses.inst	number of overall MSHR misses
overall_mshr_misses.data	number of overall MSHR misses
overall_mshr_misses.total	number of overall MSHR misses
.WriteReq_mshr_miss_latency.data	number of WriteReq MSHR miss cycles
.WriteReq_mshr_miss_latency.total	number of WriteReq MSHR miss cycles
ReadExReq_mshr_miss_latency.data	number of ReadExReq MSHR miss cycles
ReadExReq_mshr_miss_latency.total	number of ReadExReq MSHR miss cycles
ReadCleanReq_mshr_miss_latency.inst	number of ReadCleanReq MSHR miss cycles
ReadCleanReq_mshr_miss_latency.total	number of ReadCleanReq MSHR miss cycles
ReadSharedReq_mshr_miss_latency.data	number of ReadSharedReq MSHR miss cycles
ReadSharedReq_mshr_miss_latency.total	number of ReadSharedReq MSHR miss cycles
demand_mshr_miss_latency.inst	number of demand (read+write) MSHR miss cycles
demand_mshr_miss_latency.data	number of demand (read+write) MSHR miss cycles
demand_mshr_miss_latency.total	number of demand (read+write) MSHR miss cycles
overall_mshr_miss_latency.inst	number of overall MSHR miss cycles
overall_mshr_miss_latency.data	number of overall MSHR miss cycles
overall_mshr_miss_latency.total	number of overall MSHR miss cycles
ReadExReq_mshr_miss_rate.data	mshr miss rate for ReadExReq accesses
ReadExReq_mshr_miss_rate.total	mshr miss rate for ReadExReq accesses
ReadCleanReq_mshr_miss_rate.inst	mshr miss rate for ReadCleanReq accesses
ReadCleanReq_mshr_miss_rate.total	mshr miss rate for ReadCleanReq accesses
ReadSharedReq_mshr_miss_rate.data	mshr miss rate for ReadSharedReq accesses
ReadSharedReq_mshr_miss_rate.total	mshr miss rate for ReadSharedReq accesses
demand_mshr_miss_rate.inst	mshr miss rate for demand accesses
demand_mshr_miss_rate.data	mshr miss rate for demand accesses
demand_mshr_miss_rate.total	mshr miss rate for demand accesses
overall_mshr_miss_rate.inst	mshr miss rate for overall accesses
overall_mshr_miss_rate.data	mshr miss rate for overall accesses
overall_mshr_miss_rate.total	mshr miss rate for overall accesses
ReadExReq_avg_mshr_miss_latency.data	average ReadExReq mshr miss latency
ReadExReq_avg_mshr_miss_latency.total	average ReadExReq mshr miss latency
ReadCleanReq_avg_mshr_miss_latency.inst	average ReadCleanReq mshr miss latency
ReadCleanReq_avg_mshr_miss_latency.total	average ReadCleanReq mshr miss latency
ReadSharedReq_avg_mshr_miss_latency.data	average ReadSharedReq mshr miss latency
ReadSharedReq_avg_mshr_miss_latency.total	average ReadSharedReq mshr miss latency
demand_avg_mshr_miss_latency.inst	average overall mshr miss latency
demand_avg_mshr_miss_latency.data	average overall mshr miss latency
demand_avg_mshr_miss_latency.total	average overall mshr miss latency
overall_avg_mshr_miss_latency.inst	average overall mshr miss latency
overall_avg_mshr_miss_latency.data	average overall mshr miss latency

overall_avg_mshr_miss_latency.total	average overall mshr miss latency
replacements	number of replacements

TLB_walker_cache	**Description**
tags.pwrStateResidencyTicks.UNDEFINED	Cumulative time (in ticks) in various power states
tags.tagsinuse	Cycle average of tags in use
tags.total_refs	Total number of references to valid blocks.
tags.sampled_refs	Sample count of references to valid blocks.
tags.avg_refs	Average number of references to valid blocks.
tags.warmup_cycle	Cycle when the warmup percentage was hit.
tags.tag_accesses	Number of tag accesses
tags.data_accesses	Number of data accesses
pwrStateResidencyTicks.UNDEFINED	Cumulative time (in ticks) in various power states
blocked_cycles.no_mshrs	number of cycles access was blocked
blocked_cycles.no_targets	number of cycles access was blocked
blocked.no_mshrs	number of cycles access was blocked
blocked.no_targets	number of cycles access was blocked
avg_blocked_cycles.no_mshrs	average number of cycles each access was blocked
avg_blocked_cycles.no_targets	average number of cycles each access was blocked
replacements	number of replacements

It is also worth mentioning that NVMain provides information regarding the cache in the simulator default output (terminal). This information provides more details about the behavior of the cache, such as the power consumption. The full list is presented in the next table.

NVMain Stats
rankX.bankXsubarray0.subArrayEnergy
rankX.bankXsubarray0.activeEnergy
rankX.bankXsubarray0.burstEnergy
rankX.bankXsubarray0.writeEnergy
rankX.bankXsubarray0.refreshEnergy
rankX.bankXsubarray0.cancelledWrites
rankX.bankXsubarray0.cancelledWriteTime
rankX.bankXsubarray0.pausedWrites
rankX.bankXsubarray0.averagePausesPerRequest
rankX.bankXsubarray0.measuredPauses
rankX.bankXsubarray0.averagePausedRequestProgress
rankX.bankXsubarray0.measuredProgresses
rankX.bankXsubarray0.reads

rankX.bankXsubarray0.writes
rankX.bankXsubarray0.activates
rankX.bankXsubarray0.precharges
rankX.bankXsubarray0.refreshes
rankX.bankXsubarray0.worstCaseEndurance
rankX.bankXsubarray0.averageEndurance
rankX.bankXsubarray0.actWaits
rankX.bankXsubarray0.actWaitTotal
rankX.bankXsubarray0.actWaitAverage
rankX.bankXsubarray0.worstCaseWrite
rankX.bankXsubarray0.num00Writes
rankX.bankXsubarray0.num01Writes
rankX.bankXsubarray0.num10Writes
rankX.bankXsubarray0.num11Writes
rankX.bankXsubarray0.averageWriteTime
rankX.bankXsubarray0.measuredWriteTimes
rankX.bankXsubarray0.mlcTimingHisto
rankX.bankXsubarray0.cancelCountHisto
rankX.bankXsubarray0.wpPauseHisto
rankX.bankXsubarray0.wpCancelHisto
rankX.bankXbankEnergy
rankX.bankXactiveEnergy
rankX.bankXburstEnergy
rankX.bankXrefreshEnergy
rankX.bankXbankPower
rankX.bankXactivePower
rankX.bankXburstPower
rankX.bankXrefreshPower
rankX.bankXbandwidth
rankX.bankXdataCycles
rankX.bankXpowerCycles
rankX.bankXutilization
rankX.bankXreads
rankX.bankXwrites
rankX.bankXactivates
rankX.bankXprecharges
rankX.bankXrefreshes
rankX.bankXactiveCycles
rankX.bankXstandbyCycles
rankX.bankXfastExitActiveCycles

rankX.bankXfastExitPrechargeCycles
rankX.bankXslowExitPrechargeCycles
rankX.bankXactWaits
rankX.bankXactWaitTotal
rankX.bankXactWaitAverage
rankX.bankXaverageEndurance
rankX.bankXworstCaseEndurance
rankX.totalEnergy
rankX.backgroundEnergy
rankX.activateEnergy
rankX.burstEnergy
rankX.refreshEnergy
rankX.totalPower
rankX.backgroundPower
rankX.activatePower
rankX.burstPower
rankX.refreshPower
rankX.reads
rankX.writes
rankX.activeCycles
rankX.standbyCycles
rankX.fastExitActiveCycles
rankX.fastExitPrechargeCycles
rankX.slowExitCycles
rankX.actWaits
rankX.actWaitTotal
rankX.actWaitAverage
rankX.rrdWaits
rankX.rrdWaitTotal
rankX.rrdWaitAverage
rankX.fawWaits
rankX.fawWaitTotal
rankX.fawWaitAverage
.mem_reads
.mem_writes
.rq_rb_hits
.rq_rb_miss
.wq_rb_hits
.wq_rb_miss
.total_drains

.total_drain_writes
.average_writes_per_drain
.minimum_drain_writes
.maximum_drain_writes
.total_drain_cycles
.average_drain_cycles
.minimum_drain_cycles
.maximum_drain_cycles
.total_non_drain_cycles
.average_drain_spacing
.minimum_drain_spacing
.maximum_drain_spacing
.total_read_cycles
.average_read_spacing
.minimum_read_spacing
.maximum_read_spacing
.total_readqueue_size
.average_predrain_readqueue_size
.minimum_predrain_readqueue_size
.maximum_predrain_readqueue_size
.total_reads_during_drain
.average_reads_during_drain
.minimum_reads_during_drain
.maximum_reads_during_drain
.starvation_precharges
.averageLatency
.averageQueueLatency
.averageTotalLatency
.measuredLatencies
.measuredQueueLatencies
.measuredTotalLatencies
.simulation_cycles
.wakeupCount
totalReadRequests
totalWriteRequests
successfulPrefetches
unsuccessfulPrefetches

4.2 NVMInterface

4.2.1 NVMInterface configuration

NVMInterface is instantiated as a GEM5 DRAM object as part of a MemCtrl Memory Controller. In the following table the parameter names and descriptions are presented. Their default values for the "NVM_2400_1x64" configuration provided by gem5 are also shown as reference. This configuration provides NVM delays and architecture to mimic a PCM-like memory. It also mimics a 64-bit media-agnostic DIMM interface.

Parameter	Description	Reference Value
addr_mapping	The address mapping scheme used to map physical addresses to channels (Ch), ranks (Ra), banks (Ba), row (Ro), and column (Co), MSB to LSB.	RoRaBaCoCh
banks_per_rank	The number of banks per rank in the memory device	16
burst_length	The amount of data transferred in one burst operation	8
clk_domain	The clock domain that this memory component operates in	system.clk_domain
conf_table_reported	XXX	true
device_bus_width	The width of the memory device's data bus	64
device_rowbuffer_size	The size of the row buffer in the memory device	256
device_size	The total size of the individual memory device	549755813888
devices_per_rank	The number of memory devices per rank in the memory module	1
eventq_index	The index of the event queue that this component is assigned to	0
image_file	Path to an image file used for the memory's initial content	""
in_addr_map	Whether this memory should appear in the global address map.	true
kvm_map	Whether KVM should map this memory into the guest address space during acceleration.	true
max_pending_reads	NVM DIMM could have buffer to offload read commands; defines buffer depth, which will limit the number of pending reads	64
max_pending_writes	NVM DIMM could have buffer to offload read commands; defines buffer depth, which will limit the number of pending writes	128
power_state/clk_gate_bins	Power state parameters	20
power_state/clk_gate_max		1000000000000
power_state/clk_gate_min		1000
power_state/default_state		UNDEFINED
power_state/eventq_index		0

power_state/leaders		[]
power_state/possible_states		[]
range	The memory address range handled by this memory component, defining which addresses are mapped to it.	0:536870912
ranks_per_channel	The number of memory ranks per channel	1
read_buffer_size	The size of the buffer that holds read requests in the memory controller	64
tBURST	Timing parameters, in picoseconds	3332
tCK		833
tCS		1666
tREAD		150000
tRTW		1666
tSEND		14160
tWRITE		500000
tWTR		1666
two_cycle_rdwr	Two cycles required to send read and write commands, boolean	true
write_buffer_size	The size of the buffer that holds write requests in the memory controller	128

4.2.2 NVMInterface output

When the simulation finishes the following statistics related to NVMInterface are provided to the user

Name	Description
readBursts	Number of NVM read bursts
writeBursts	Number of NVM write bursts
perBankRdBursts	Per bank write bursts
perBankWrBursts	Per bank write bursts
totQLat	Total ticks spent queuing
totBusLat	Total ticks spent in databus transfers
totMemAccLat	Total ticks spent from burst creation until serviced by the NVM
avgQLat	Average queueing delay per NVM burst
avgBusLat	Average bus latency per NVM burst
avgMemAccLat	Average memory access latency per NVM burst
avgRdBW	Average DRAM read bandwidth in MiBytes/s
avgWrBW	Average DRAM write bandwidth in MiBytes/s
peakBW	Theoretical peak bandwidth in MiByte/s
busUtil	NVM Data bus utilization in percentage
busUtilRead	NVM Data bus read utilization in percentage

busUtilWrite	NVM Data bus write utilization in percentage
pendingReads	Reads issued to NVM for which data has not been transferred
pendingWrites	Number of outstanding writes to NVM
bytesPerBank	Bytes read within a bank before loading new bank

5. Expected future evaluation

The final objective of the task is to enable experimentation with relevant PCM configurations in order to easily conduct an evaluation with the selected configurations. This section describes the different types of PMC memories and their internal behaviour and features (section 5.1), and the strategy for integration and how to model them into the simulator using the provided configuration parameters and output metrics (section 5.2).

5.1 Approaches for Operating PCM Memories and related modeling

Two common strategies exist to write and read PCM memories: either optical or electrical. In the former case when they need to be programmed short pulses shall be used. For the latter, a small amplitude signal, either pulsed or continuous wave, can be used.

One of the key advantages of an optical read-out approach consists of being able to achieve extremely low latency, but also extremely low power optical signals (limited by SNR and photodetector sensitivities) which translates into overall low-power consumption. However, the writing phase poses a series of trade-offs from an architectural point of view to operate PCM memories.

In the next subsections, we will discuss two separate ways to operate PCM memories in terms of their writing phase, i.e., optically, and electrically, while we will focus solely on an optical read-out approach. A novel photonic device working as PCM memory will be discussed as well as related benchmarking strategies.

5.1.1 Optical writing approach for integrated thin-film PCM patches

In an optical writing approach, the PCM patch located above the photonic waveguide is stimulated by short optical pulses which gradually change its crystalline (and amorphous) fraction, thereby encoding data directly in its phase.

Depending on the signal-to-noise ratio (SNR) that can be achieved from a writing as well as a reading point of view, a certain bit resolution will be achieved. While this approach is the most promising for rapidly writing PCM memories (below ns for good heat

evacuation and suitable material properties), it comes with 2 major limitations: (i) the optical waveguides for read and write operations shall be separated – this is related to the fact that optical writing signals shall not modify multiple memory elements and (ii) to achieve low optical losses different waveguide materials might need to be used e.g., Si (for read-out signals) versus SiN (for writing signals) due to two-photon absorption effects which may take place for high-power pulses.

5.1.2 Electrical writing approach for integrated thin-film PCM patches

Another approach that is under investigation concerns the possibility of writing the PCM memories with short electrical pulses, similarly to what is done in D4.1 for the neuromorphic architectures, to set the weights of the accelerator. While this approach does not require to increase the number of waveguides, it does increase the number of routing wires. However, given the possibility of using multiple interconnect layers for wiring, this approach simplifies the routing schemes that are needed. However, it comes with a major warning that electrical contacts cannot come too close to an optical device. Therefore, the writing speed is limited by the thermal constant of the dielectric distribution around the Si waveguide, which is hundreds of ns from D3.4. Power consumption is also increased in this scenario due to the absence of local heating of the PCM patch as instead the case for the optical writing scenario.

5.1.3 Proposal for a PCM device as optical memory

In Neuropuls, we developed a novel photonic device that allows to be used as optical PCM memory by using mode hybridization in a multimode waveguide. The cross-section is identical to the one for the accelerator weights based on a thin GeSe patch on top of the Si waveguide. However, for best operation, the waveguide and patch widths have different dimensions compared to the ones implemented in the MZI mesh to set up the hybrid operation mode.

Initially, we have foreseen an electrical writing of this device where we expect to store at least 70 levels (~ 6 bits) based on its device length (7 um with 100 nm long sections) and the linear change of its characteristics (i.e., optical phase) as a function of its degree of crystallization, as shown in Figure 9. This is thanks to an approach which uses the ultra-low optical losses of GeSe and this new device concept which allows to achieve a much longer GeSe-based device compared to more classical GST-based devices.

Further investigations are ongoing from a modeling perspective to enhance the number of levels that can be encoded and more energy-efficient and low latency strategies for their writing.

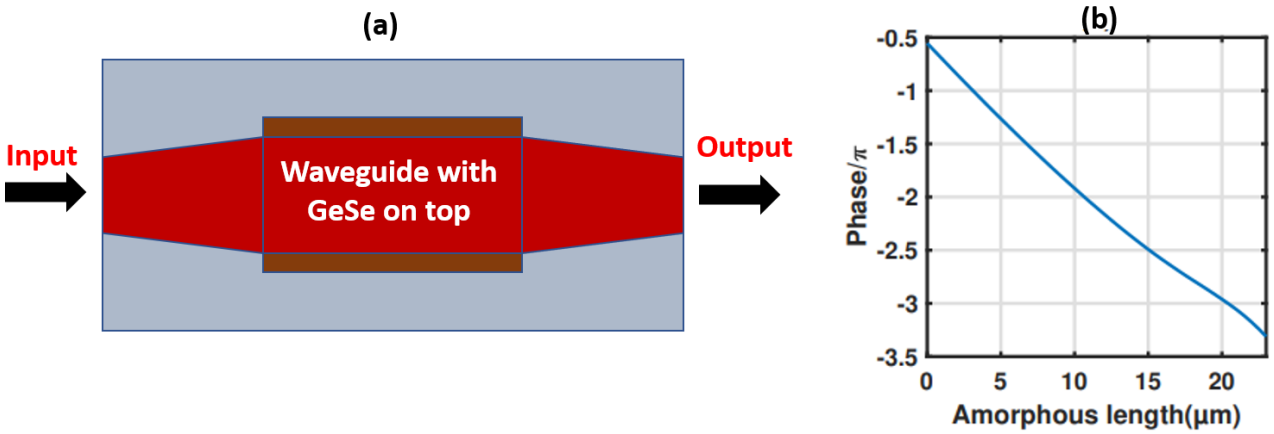

Figure 9: (a) top view of the proposed device and (b) variation of phase shift with the degree of crystallization.

5.1.4 Benchmarking strategies and modeling

To benchmark the PCM memories, we will be looking at how they perform under classical tasks such as matrix-matrix multiplication in the optical domain where, e.g., the PCM elements store the matrix elements for one of the matrices to be multiplied, while the other one is encoded in the incoming optical signals in a crossbar architecture.

We will also investigate typical read/write operations for memories under real computing flow scenarios. Parameters such as power consumption, latency, speed/throughput, and distributions will be obtained to provide input to the gem5 platform.

To extract these parameters, compact behavioral models such as the one presented in D3.4 based on a transfer matrix approach for PCM-based Mach-Zehnder interferometers (written by electrical pulses) are currently being explored.

5.2 Integration of Different Approaches in the Simulator

As described in the previous section, PCM memories have a very different nature. This presents a challenge in their integration in the simulator, which is required to accommodate these different types. PCM memories may have very different features, including differences in voltages, operation timing, among others.

To solve this challenge, we have worked on collecting a list of configurable parameters for the simulator components, as presented in section 4. This information aids prepare the experiments where relevant information can be provided, and providing experimental results comparing PMCs to other types of memories.

In order to do so we provide two different methods, one for each memory type (and simulator component):

- For the memory controller (NVMInterface) the different configuration parameters can be set and the metrics can be directly obtained from the gem5 simulation output. Experimentation will be limited to the configurable parameters, which should be enough as they already model a PMC-like memory as shown in section 4.2.1. The evaluation of the results is easily done using the 19 different output metrics provided by the simulation.
- For the caches (NVMain) a more complex integration is needed and we follow the ideas presented by Narayan et al. [10] and works like [14], where the required components are described. The optical cache in NVMain is implemented using a DRAM-like interface, mapping cache lines into an array. The very large number of configurable parameters present in NVMain can be narrowed down using the tables presented in section 4.1.1, where different configuration files based on real PCM memory parameter values.

6. Conclusions

The NEUROPULS memory hierarchy simulation infrastructure provides a highly configurable environment enabling experimentation with various cache and memory designs. By offering a simulation platform based on state-of-the-art microarchitecture-level simulators, such as gem5, the NEUROPULS infrastructure will become a playground for future innovation. NEUROPULS partners and researchers can experiment with novel architectures, algorithms, and configurations.

The NEUROPULS memory hierarchy simulation infrastructure allows for the exploration of intricate design spaces, facilitating a more in-depth understanding of how different configurations may impact the overall system performance.

During the first iteration of this task, we have analyzed a variety of options available for cache and memory simulation, we studied their relevance and their suitability to simulate PCMs, and studied the state of the art to understand their capabilities as research tools. The outcome of this analysis is the selection of two components, NVMain and NVMInterface, that will be integrated into the NEUROPULS simulation infrastructure. For each component, we analyzed their configuration parameters and their output metrics. This is done in order to ensure they can model and simulate relevant caches and memories, such that investigating different designs enables the researchers to compare different memory technologies and assess their performance. A plan for conducting the future evaluation during the second iteration of the task has also been prepared. A compact ultra-low-loss PCM-based device that can be used as an optical memory has also been proposed. Such a device is actuated electrically, while its read-out is carried out optically. Due to the very low losses, the read-out can be carried out with low power signals, thus benefiting the overall system energy consumption.

7. References

[1] Lowe-Power, et al., The gem5 Simulator: Version 20.0+, arXiv, 2020, https://arxiv.org/abs/2007.03152.
[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi and S. K. Reinhardt, "The M5 Simulator: Modeling Networked Systems," in IEEE Micro, vol. 26, no. 4, pp. 52-60, July-Aug. 2006, doi: 10.1109/MM.2006.82.
[3] A. Narayan, Y. Thonnart, P. Vivet, A. K. Coskun, A. Joshi "Architecting Optically Controlled Phase Change Memory" in ACM Transactions on Architecture and Code Optimization, Volume 19, Issue 4, Article No.: 48, Pages 1 – 26 doi: 10.1145/3533252
[4] P. Rosenfeld, E. Cooper-Balis, B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator" in IEEE Computer Architecture Letters (Volume: 10, Issue: 1, Jan.-June 2011), doi: 10.1109/L-CA.2011.4
[5] S. Li, Z. Yang, D. Reddy, A. Srivastava, B. Jacob "DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator" in IEEE Computer Architecture Letters (Volume: 19, Issue: 2, 01 July-Dec. 2020), doi: 10.1109/LCA.2020.2973991
[6] H. Luo, Y. C. Tuğrul, F. N. Bostancı, A. Olgun; A. G. Yağlıkçı, O. Mutlu, "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator" in IEEE Computer Architecture Letters (Volume: 23, Issue: 1, Jan.-June 2024), doi: 10.1109/LCA.2023.3333759
[7] GEM5 user documentation, NVMInterface, https://www.gem5.org/2020/05/27/memory-controller.html
[8] M. Poremba, T. Zhang, Y. Xie, "NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems", in IEEE Computer Architecture Letters (Volume: 14, Issue: 2, 01 July-Dec. 2015), doi: 10.1109/LCA.2015.2402435
[9] L. Pentecost; A. Hankin; M. Donato; M. Hempstead; G. Wei; D. Brooks, "NVMExplorer: A Framework for Cross-Stack Comparisons of Embedded Non-Volatile Memories" in 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), doi: 10.1109/HPCA53966.2022.00073
[10] A. Narayan, Y. Thonnart, P. Vivet, A. Coskun, and A. Joshi, "Architecting optically controlled phase change memory," ACM Transactions on Architecture and Code Optimization, vol. 19, no. 4, pp. 1–26, Dec. 2022, doi: 10.1145/3533252.
[11] Y. Choi, I. Song, M. Park, H. Chun, S. Chang, B. Cho, J. Kim, Y. Oh, et al., "A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth," IEEE Conference Publication | IEEE Xplore, Feb. 01, 2012. https://ieeexplore.ieee.org/document/6176872
[12] GEM5 developer documentation, AbstractMemory, http://doxygen.gem5.org/develop/classAbstractMemory.html
[13] JEDEC Solid State Technology Association. https://www.jedec.org
[14] Khan, A. A., Hameed, F., & Castrillon, J. (2018). NVMain Extension for Multi-Level Cache Systems. In Proceedings of the Rapido'18 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (Vol. 17, pp. 1–6). RAPIDO '18: Methods and Tools. ACM. https://doi.org/10.1145/3180665.3180672
[15] GEM5 developer documentation, MemCtrl, http://doxygen.gem5.org/release/v22-0-0-0/classgem5_1_1memory_1_1MemCtrl.html