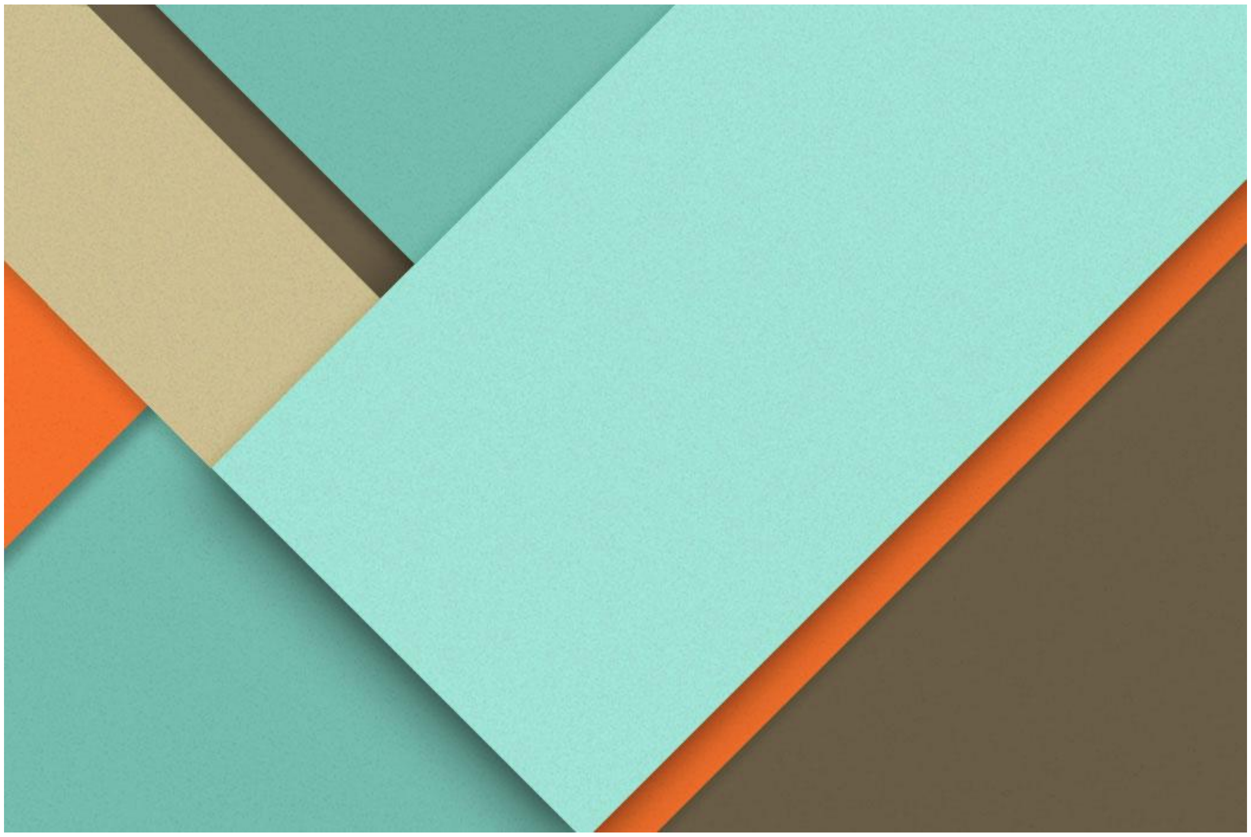


HPCA Assignment 1

28.03.2022

GROUP - 5

Indian Institute of Technology
Kharagpur

Table of Contents

Team Members

Sr. No	Name	Roll Number	Email ID
1	Debajyoti Dasgupta	18CS30051	debajyotidasgupta6@gmail.com
2	Archisman Pathak	18CS30050	archisman.pathak@gmail.com
3	Somnath Jena	18CS30047	somnathjena.2011@gmail.com
4	Rounak Patra	18CS30048	rounakpatra2000@gmail.com
5	Shubhi Shukla	21CD91R10	shuklashubhi6@gmail.com
6	Abhinav Bohra	18CS30049	abhinavbohra09@gmail.com
7	Ayan Chakraborty	18EC10075	ayan.ch1508@gmail.com
8	Rudrajyoti Roy	18EC10047	rudrajyotiroy@gmail.com
9	Tushar Gupta	18CS30044	gupta.tushar@iitkgp.ac.in
10	Vaishnavi Raghvendra Munghate	18CS30045	munghatevaishnavi@gmail.com
11	Vattikuti Rahul Naga Sri Bharadhwaj	18CS30046	rahul.bharadhwaj9@gmail.com
12	Chakka Venugopal	18CS30052	chakkavenu@gmail.com
13	Kothapalli Dileep	18CS30053	kothapallidileep10122000@gmail.com

Contributions

Sr. No	Name	Contributions
1	Debajyoti Dasgupta	Programming Simulations | Generating Statistics | Report
2	Archisman Pathak	Analyzing statistics
3	Somnath Jena	Programming Simulations | Generating Statistics
4	Rounak Patra	Analyzing statistics | Plotting Statistics
5	Shubhi Shukla	Analyzing statistics | Plotting Statistics
6	Abhinav Bohra	Analyzing statistics

7	Ayan Chakraborty	Discussing statistics | preparing report
8	Rudrajyoti Roy	Discussing statistics | preparing report
9	Tushar Gupta	Preparing Report
10	Vaishnavi Raghvendra Munghate	Analyzing statistics
11	Vattikuti Rahul Naga Sri Bharadhwaj	Analyzing statistics
12	Chakka Venugopal	Discussing statistics
13	Kothapalli Dileep	Discussing statistics

Goals

1. Configure an Out of Order CPU with custom parameters in Gem5
2. Simulate a Benchmark program on the CPU with different parameter combinations
3. Extract the top 10 configurations based on CPI values for the benchmark program
4. Plot graphs showing the variation of other performance measures versus each of the top 10 combinations
5. Analyze the results and provide adequate justification

Brief Introduction & Methodology:

Gem-5 is an architectural simulator supporting different ISAs such as MIPS and ARM. Custom Hardware components can be added and simulated as well. In this experiment, we do not design any novel hardware, and instead, configure the existing hardware components with custom parameters to observe their effects on performance. Certain parameters are kept fixed, and the others are varied as given in the assignment.

The different configurations are automated through a script, and the top 10 configurations wrt CPI are chosen. They are re-simulated, and the other performance measures are extracted from their stats files using a parser script. The stats file is automatically generated by Gem-5. The extracted performance parameters are then plotted to observe their variations. Finally, we analyze our results and provide relevant

Code Structure:

```
├── README.md
├── config.py
├── m5out
│    ├── 64kB_32kB_512kB_4_8_TournamentBP_128_64/stats.txt
│    ├── 64kB_32kB_512kB_4_8_TournamentBP_192_16/stats.txt
│    ├── 64kB_32kB_512kB_8_8_BiModeBP_128_16/stats.txt
│    ├── 64kB_32kB_512kB_8_8_BiModeBP_128_64/stats.txt
│    ├── 64kB_32kB_512kB_8_8_BiModeBP_192_16/stats.txt
│    ├── 64kB_32kB_512kB_8_8_BiModeBP_192_64/stats.txt
│    ├── 64kB_32kB_512kB_8_8_TournamentBP_128_16/stats.txt
│    ├── 64kB_32kB_512kB_8_8_TournamentBP_128_64/stats.txt
│    ├── 64kB_32kB_512kB_8_8_TournamentBP_192_16/stats.txt
│    └── 64kB_32kB_512kB_8_8_TournamentBP_192_64/stats.txt
├── options.py
├── plot.py
├── qsort4
└── run.py
```

Custom scripts

config.py

- This file contains all the code segments required for creating the architecture that the simulator needs to simulate. This file is configured for automatically taking in variable parameters (as described in the assignment) as arguments to the file
- Following are the important objects that are defined in this file which helps in the simulation modeling of the components of the architecture
    - **L1Cache** - This class inherits from the Cache class from m5.objects and is used to build the L1 caches (including the data and the instruction cache)
    - **L1DCache** - This class inherits from the L1Cache class and is used to implement the data cache
    - **L1ICache** - This class inherits from the L1Cache class and is used to implement the instruction cache

    - **L2Cache** - This class inherits from the Cache class from m5.objects and is used to implement the L2 cache.
    - **LXBar** - A coherent crossbar based memory bus used to connect L2 cache with other components
    - **SystemXBar** - A system bus that is used to connect the L2 cache with the system memory controller
    - Apart from this there are **DDR3 Ram**, **DerivO3CPU**, **Memory Controller** etc. which help in building the entire system.

options.py

- This file contains all the variable arguments that are required for the config file to build the cache objects required during the simulations
- This file is imported and used by the config file for argument parsing. Hence they need to be present in the same directory

run.py

- This file contains the code to spawn new subprocesses to run the simulations with all possible values for the variable parameters.
- Outputs of all simulations are generated inside runs directory with the name of the simulation as the sub-directory. For example if the parameters' values are l1d_size: 32kB, l1i_size: 32kB, l2_size: 128kB, l1_assoc: 2, l2_assoc: 4, bp_type: LocalBP, ROBEntries: 128, numIQEntries: 16 then all the outputs for this simulation along with stats.txt will be inside runs/32kB_32kB_128kB_2_4_LocalBP_128_16 directory. This makes the task of extracting stats corresponding to a particular simulation easier.
- All successfully executed simulations are also logged into a logs.txt file so that in case some simulation fails then we need not run all the previous simulations that have been successfully completed and only execute the failed ones.

- Using python's itertools module all possible combinations of parameter values are generated and using os module the command to run a simulation is passed inside the system method and all such simulations are executed one by one.

qsort4

- Compiled binary of the qsort4.c file that was given for benchmarking. This binary is compiled using GCC on X86 architecture.
- This binary is used in generating all the results mentioned in the report and the simulation stats file as well

m5out

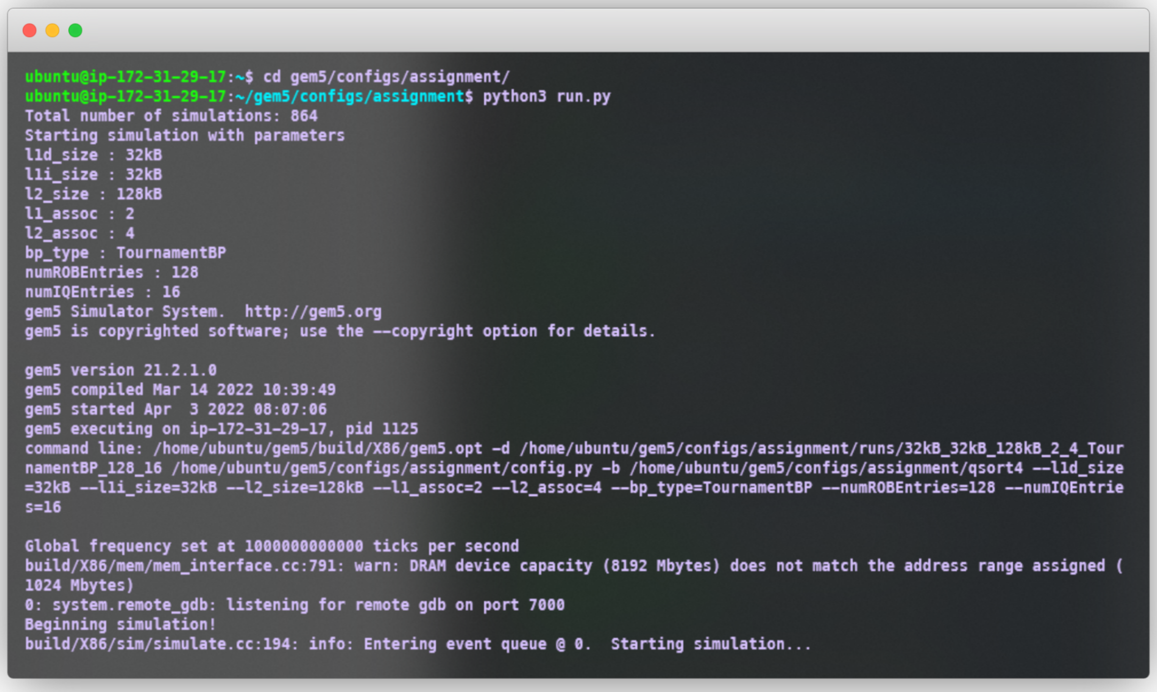
- This folder contains the runs folder. The runs folder contains the simulation results folder for different combinations of configurations as described above in runs.py. These simulation results are filtered only for the top 10 configurations.
- These are the actual statistics of the results of the simulation generated for the qsort4 binary on the different combinations (top 10 by CPI values) of the parameters mentioned in the question.

plot.py

- This file contains all the codes that are related for analyzing and generating plots requested in the problem statement. This file makes use of matplotlib for generating plot and regex for converting the statistics to machine readable format

How to run the code:

```
# Install gem5 (inside your home directory)

# Install dependencies
$ sudo apt install build-essential git m4 scons zlib1g zlib1g-dev \
         libprotobuf-dev protobuf-compiler libprotoc-dev\
         libgoogle-perftools-dev python-dev python

# Clone the repository of gem5
$ git clone https://gem5.googlesource.com/public/gem5

# Change directory to gem5
$ cd gem5

# Build the binary
$ python3 `which scons` build/X86/gem5.opt -j9

#---------------------------------------------------------------------------
# Case 1: Cloning github repository
# Clone github repository or use the submission
$ git clone https://github.com/debajyotidasgupta/HPCA-Assignment-1.git
# Change directory to assignment
$ cd HPCA-Assignment-1
# Copy files from the assignment directory to the benchmark programs
directory
$ cp -r assignment/ ~/gem5/configs/
#---------------------------------------------------------------------------
# Case 2 using assignment submission
# Create assignment directory in the gem5/config directory
$ mkdir ~/gem5/configs/assignment
# Copy the contents of the submission to the assignment directory
$ cp -r Group_5_HPCA_Assignment_1/* ~/gem5/configs/assignment/
#---------------------------------------------------------------------------
# Change directory to the gem5 root directory
$ cd ~/gem5

# Run the gem5 simulation
$ build/X86/gem5.opt -d configs/assignment/m5out
configs/assignment/config.py -b configs/assignment/qsort4

# To run simulations with all possible configurations
$ cd ~/gem5/configs/assignment
$ python3 run.py

# To run plots file
$ python3 plot.py

# PS: Running all simulations might take about 30 hrs!!
```



```
ubuntu@ip-172-31-29-17:~$ cd gem5/configs/assignment/
ubuntu@ip-172-31-29-17:~/gem5/configs/assignment$ python3 run.py
Total number of simulations: 864
Starting simulation with parameters
l1d_size : 32kB
l1i_size : 32kB
l2_size : 128kB
l1_assoc : 2
l2_assoc : 4
bp_type : TournamentBP
numROBEntries : 128
numIQEntries : 16
gem5 Simulator System.  http://gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 21.2.1.0
gem5 compiled Mar 14 2022 10:39:49
gem5 started Apr  3 2022 08:07:06
gem5 executing on ip-172-31-29-17, pid 1125
command line: /home/ubuntu/gem5/build/X86/gem5.opt -d /home/ubuntu/gem5/configs/assignment/runs/32kB_32kB_128kB_2_4_Tour
namentBP_128_16 /home/ubuntu/gem5/configs/assignment/config.py -b /home/ubuntu/gem5/configs/assignment/qsort4 --l1d_size
=32kB --l1i_size=32kB --l2_size=128kB --l1_assoc=2 --l2_assoc=4 --bp_type=TournamentBP --numROBEntries=128 --numIQEntrie
s=16

Global frequency set at 1000000000000 ticks per second
build/X86/mem/mem_interface.cc:791: warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (
1024 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
Beginning simulation!
build/X86/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
```

Analysis & Plots:

Total number of possible configurations = 2*2*3*3*2*3*2*2 = 864

- l1d size: 32kB, 64kB
- l1i size: 32kB, 64kB
- l2 size: 128kB, 256kB, 512kB
- l1 assoc: 2, 4, 8
- l2 assoc: 4, 8
- bp type: TournamentBP, BiModeBP, LocalBP
- ROBEntries: 128, 192
- numIQEntries: 16, 64

Out of these total 864 configurations, the top 10 configurations (with respect to CPI values) are described below:

Serial No.	l1d size (kB)	l1i Size (kB)	l2 size (kB)	l1 assoc	l2 assoc	bp type	ROB Entries	Num IQ Entries	CPI
1	64	32	512	8	8	BiModal	128	64	**0.531673**
2	64	32	512	8	8	BiModal	192	16	**0.531673**
3	64	32	512	8	8	BiModal	192	64	**0.531673**
4	64	32	512	8	8	Tournament	192	16	**0.531673**
5	64	32	512	8	8	Tournament	192	64	**0.531673**
6	64	32	512	8	8	Tournament	128	16	**0.531673**
7	64	32	512	8	8	BiModal	128	16	**0.531673**
8	64	32	512	8	8	Tournament	128	64	**0.531673**
9	64	32	512	4	8	Tournament	192	16	**0.531690**
10	64	32	512	4	8	Tournament	128	64	**0.531690**

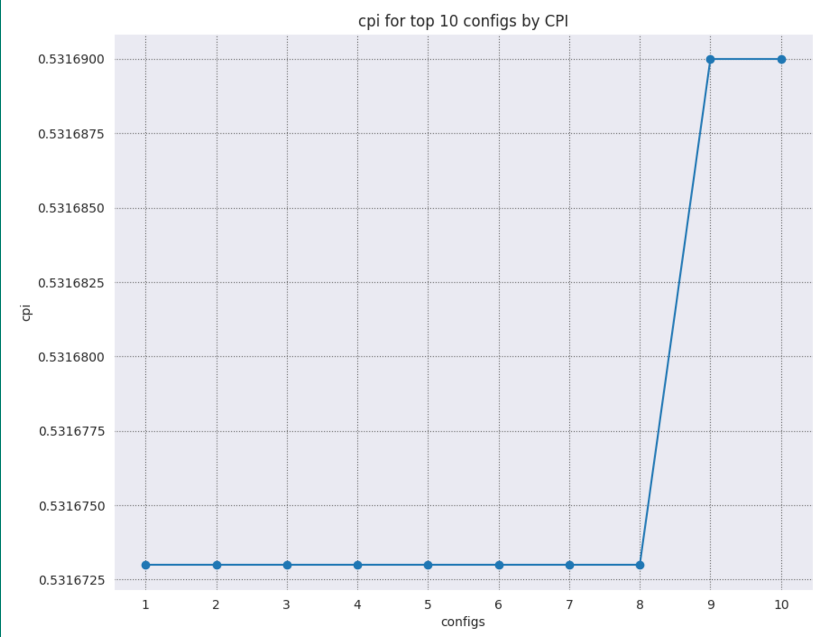


Fig 1: CPI for Top 10 configurations

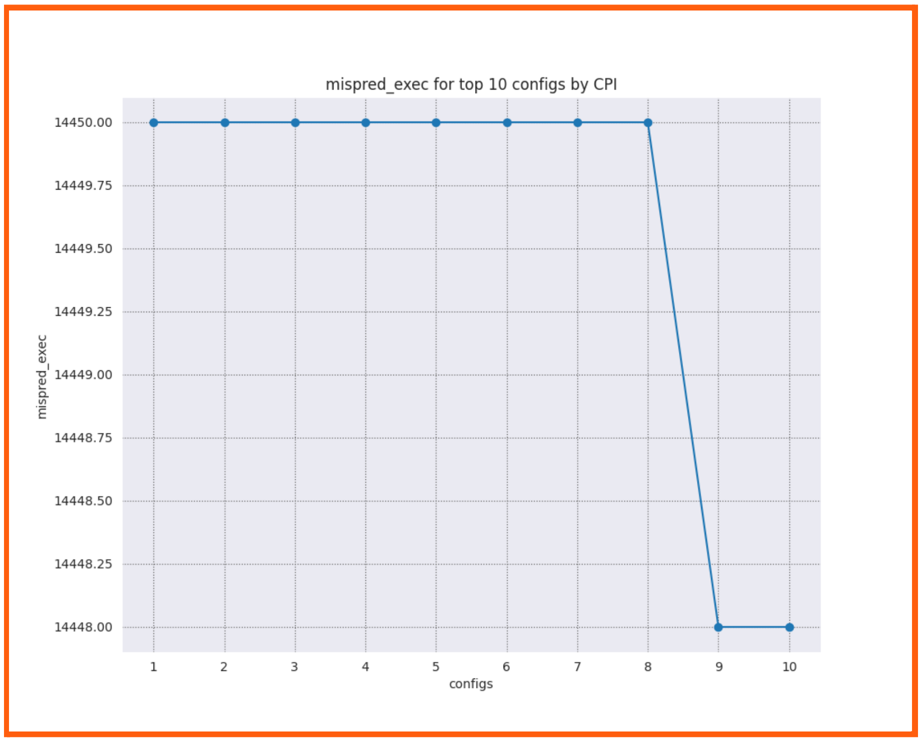


Fig 2: Mispredicted branches during execution

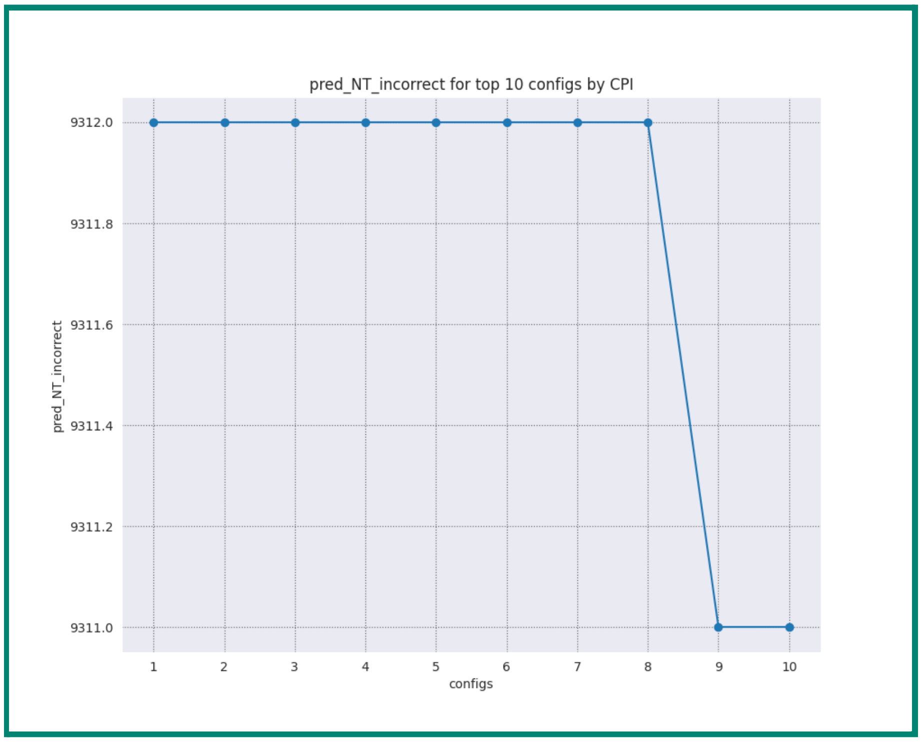


Fig 3: Number of branches that were predicted not taken incorrectly

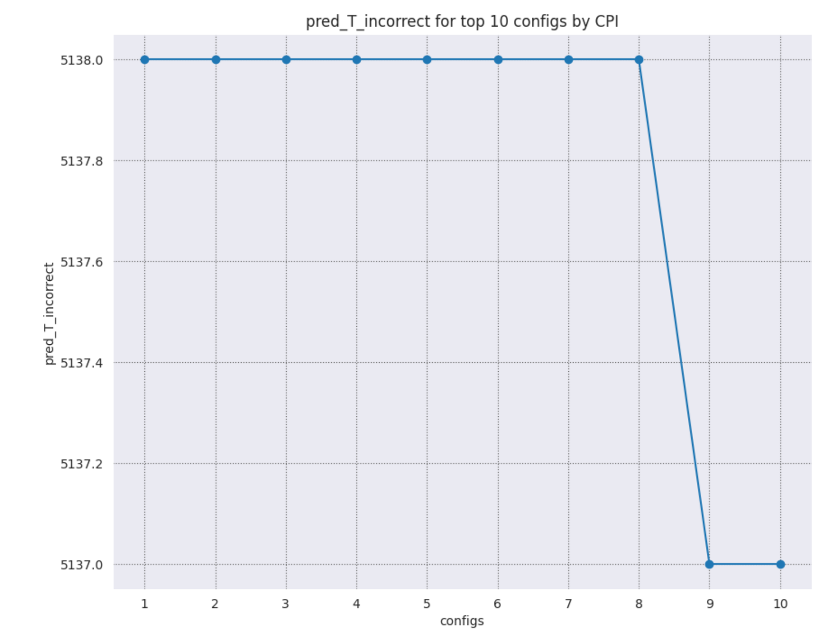


Fig 4: Number of branches that were predicted taken incorrectly

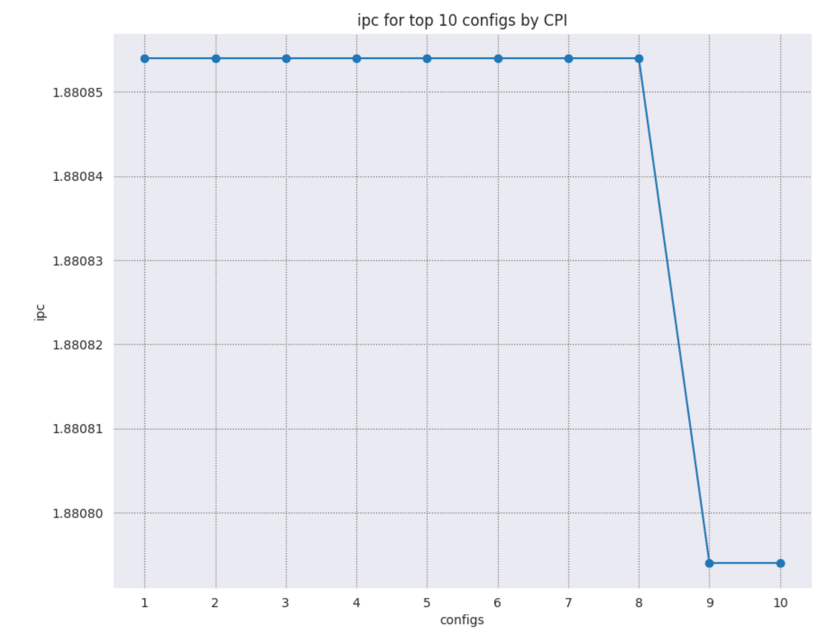


Fig 5: IPC for Top 10 configurations

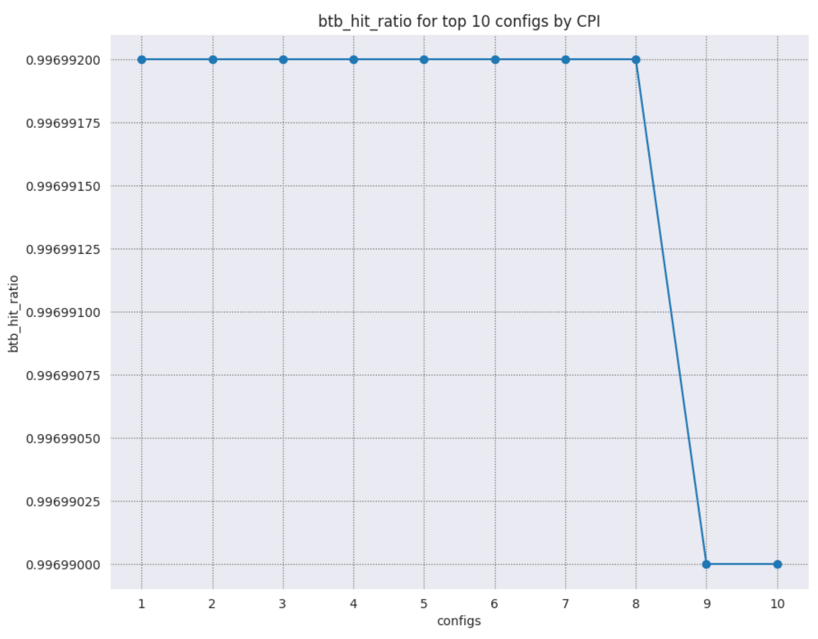


Fig 6: Number of BTB hit percentage for top 10 configurations

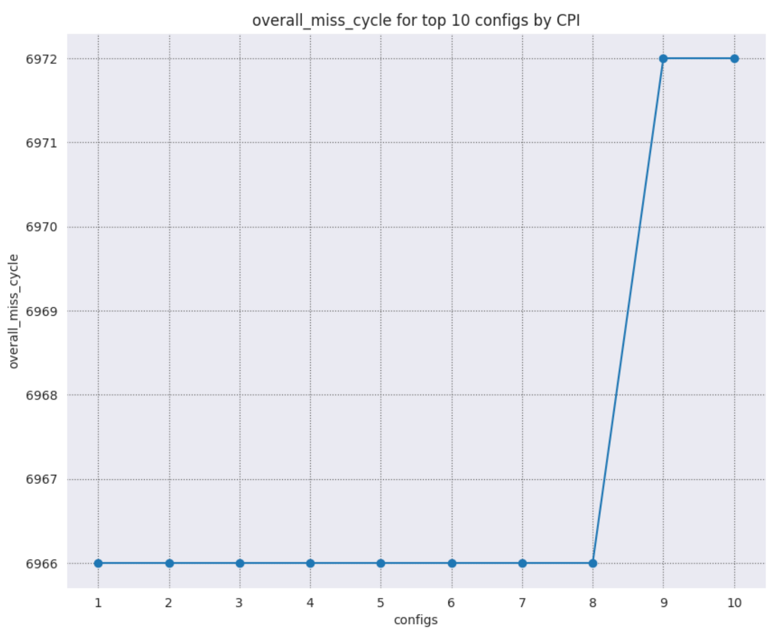


Fig 7: Number of overall miss cycles for top 10 configurations

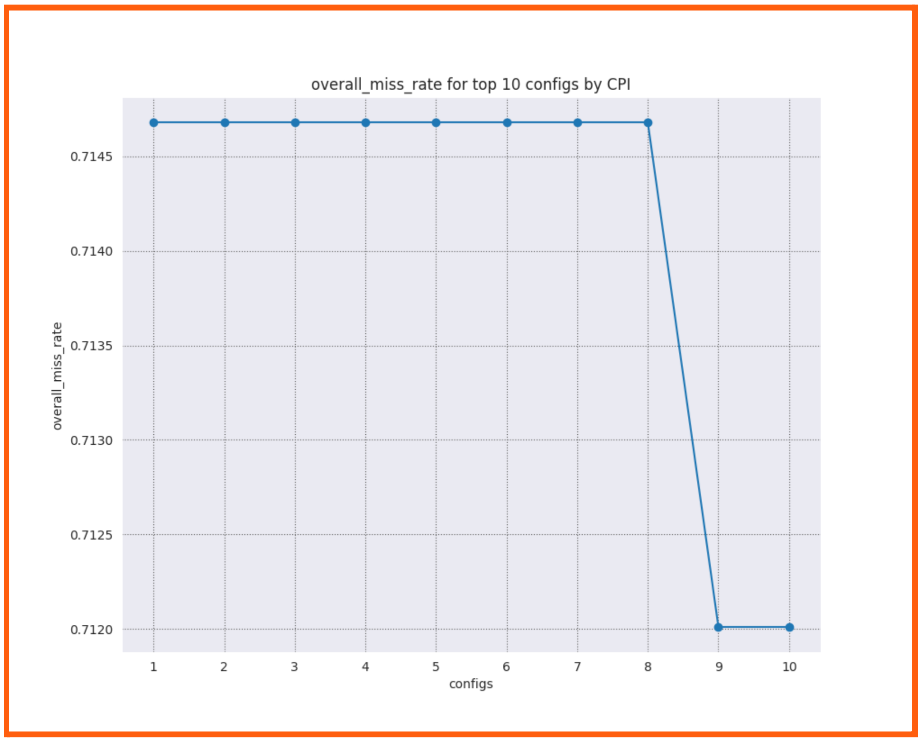


Fig 8: Miss rate for top 10 configurations

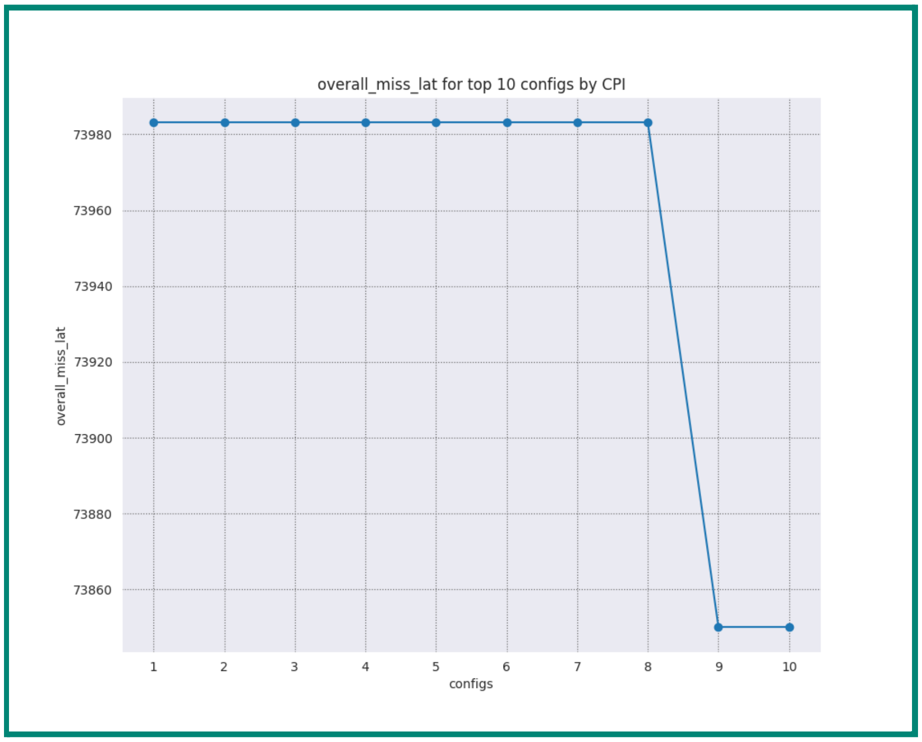


Fig 9: Overall Average Miss Latency for top 10 configurations

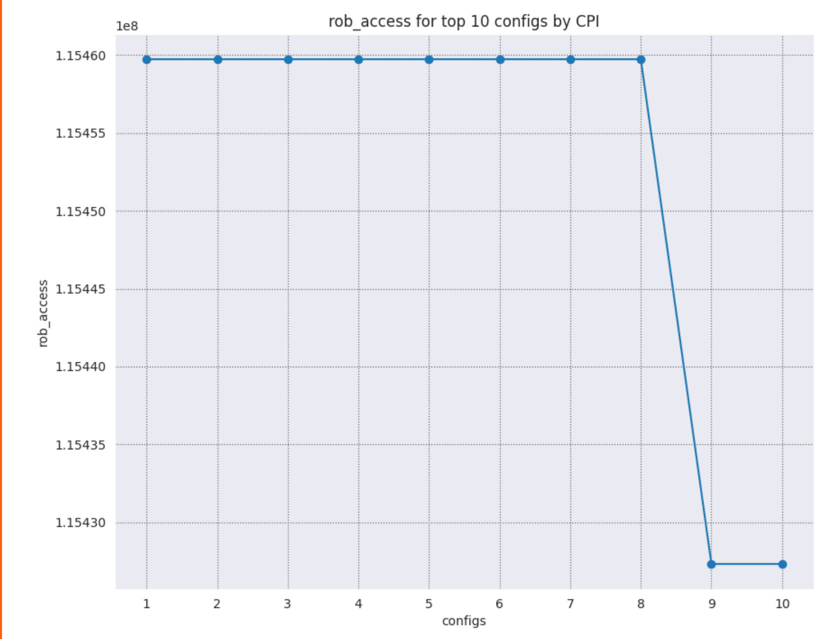


Fig 10: The number of ROB accesses (read and write both)

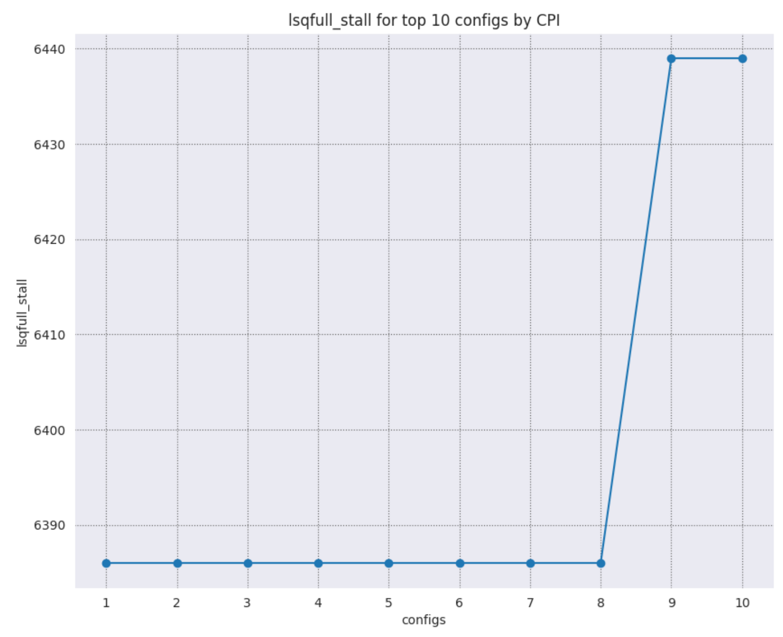


Fig 11: Number of times the LSQ has become full, causing a stall

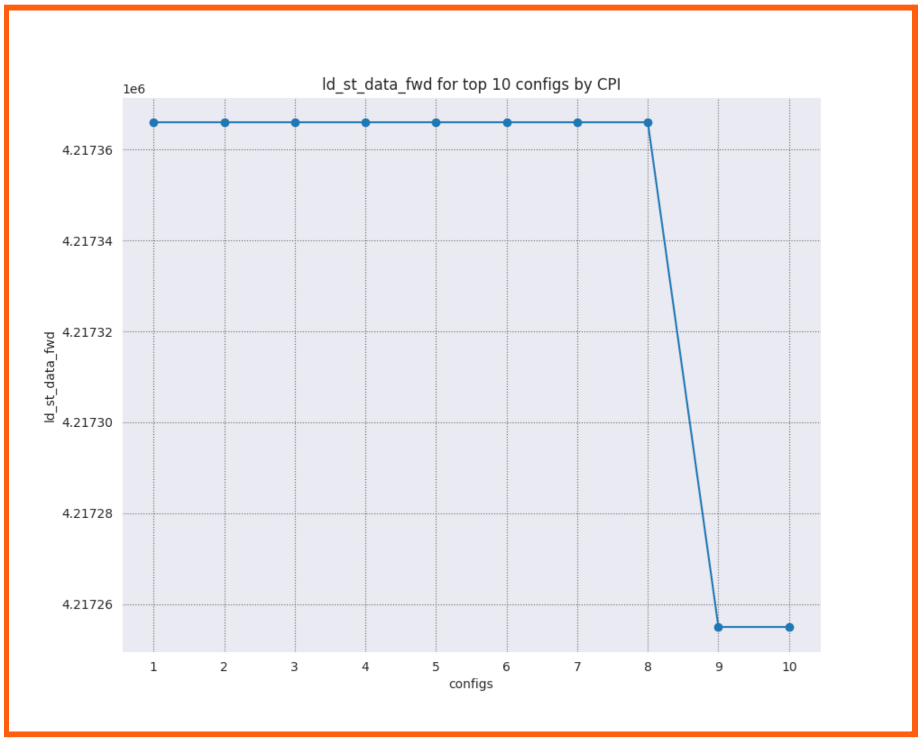


Fig 12: Number of loads that had data forwarded from stores

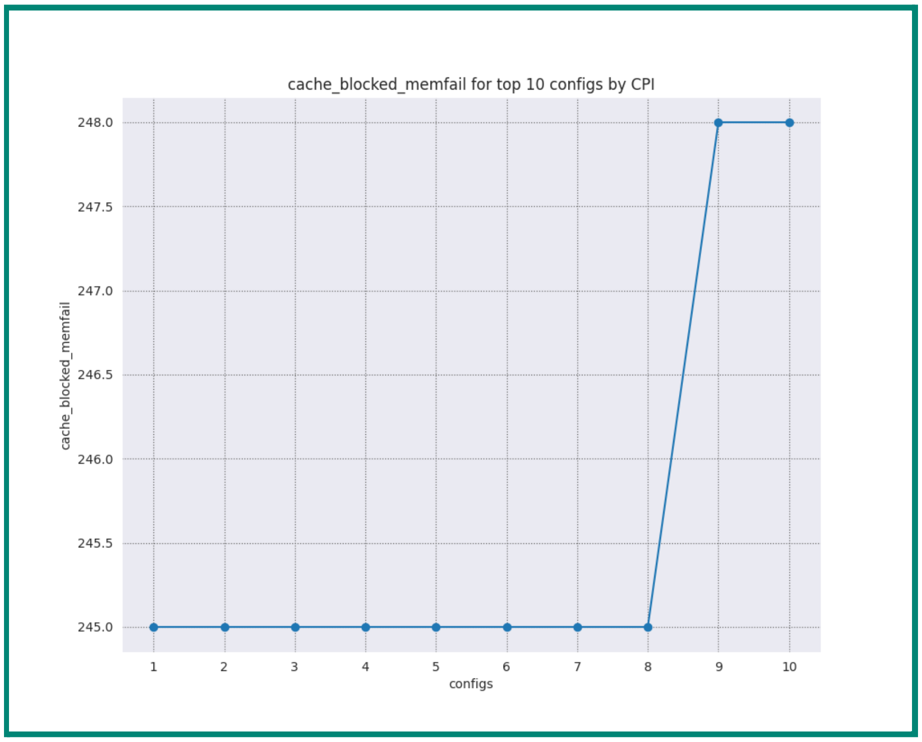


Fig 13: Number of times access to memory failed due to the cache being blocked

Results & Discussions:

- We observe that changing the ROB and Instruction Queue length does not significantly affect the CPI, given other optimal parameter choices. Hence, the best design should be chosen to have the least possible hardware usage, as performance is not affected.
- Higher L1/L2 Data cache size leads to more IPC due to more probability of L1/L2 cache hit, but L1/L2 cache hit time increases as well. While the increase in hit time is only architecture specific, the L1/L2 cache hit probability depends on the application as well. In this case we observe that the optimum sizes of cache for maximum hit rate are 32 KB (L1 Instruction Cache), 64 KB (L1 Data Cache) and 512 KB (L2 cache).
- For branch prediction, BiModal predictor and Tournament Predictor have significantly greater prediction accuracy compared to Local predictor. Therefore, these predictors result in greater IPC.
- We make another observation that decreasing the L1 Associativity increases the CPI slightly. This is to be expected because of higher miss rates. However for resource constrained devices, this reduction in associativity is worth it compared to the very small performance degradation.

References:

The following lines from stats.txt are used for observing the performance parameters for each configuration:

- 'system.cpu.cpi': 'cpi',
- 'system.cpu.iew.branchMispredicts': 'mispred_exec',
- 'system.cpu.iew.predictedNotTakenIncorrect': 'pred_NT_incorrect',
- 'system.cpu.iew.predictedTakenIncorrect': 'pred_T_incorrect',
- 'system.cpu.ipc': 'ipc',
- 'system.cpu.branchPred.BTBHitRatio': 'btb_hit_ratio',
- 'system.cpu.rob.reads': 'rob_reads',
- 'system.cpu.rob.writes': 'rob_writes',
- 'system.cpu.iew.lsqFullEvents': 'lsqfull_stall',
- 'system.cpu.lsq0.forwLoads': 'ld_st_data_fwd',
- 'system.cpu.lsq0.blockedByCache': 'cache_blocked_memfail'
- ‘system.cpu.l2cache.overallMisses::total': ‘overall_miss_cycle’
- ‘system.cpu.l2cache.overallAvgMissLatency::total': ‘overall_miss_latency’
- 'system.cpu.l2cache.overallMissRate::total': ‘overall_miss_rate’