International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

# Performance Assessment of RISC-V Architecture

**Lakshamaiah Alluri, Bhaskar M, Hemant Jeevan Magadum**

***Abstract:*** ***This paper presents the performance evaluation of RISC – V architecture based processor using Gem5 simulator. The performance analysis metrics such as bandwidth, latency, throughput, branch prediction, pipeline stages and memory hierarchy of the processor architecture are studied using Gem5 simulator. Different simulation models are carried out to arrive the best reference model for RISC-V architecture design and development. In this reference model cache memory functionality feature is verified with the verification methodology called Universal Verification Methodology (UVM).***
***From simulations it is found that both the program and data cache provides optimum performance in terms of execution time, hit rates, miss rate and miss latencies.***

***Keywords : RISC-V, Gem5, UVM, Evaluation, SystemC, Openvera.***

## I. INTRODUCTION

In processor architecture design, designers generally use software simulation tools to verify the functionality of design and evaluate their ideas further. As the technology advances design and verification engineers has to face more challenges in their respective area, especially in the microprocessor architecture design. Emerging technology like multi-core architectures with deeper pipeline stages, and different levels of memory hierarchy will increase the complexity of design. At the same time designers have to explore simulation frame work to evaluate the design and its performance metrics. Most of the simulation tools are vender specific and licensed based, term based and too costly. Gem5 simulator is one of the choice to model the basic design and it is open source simulator having flexibility to model the design and also can explore the processor performance metrics for reference model of the design

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
**Lakshmaiah Alluri***, HDG, CDAC, Thiruvananthapuram, India. Email: laxman@cdac.in.
**Dr. M Bhaskar**, Professor, Department of Electronics and Communication, NIT, Trichy, India. Email: bhaskar@nitt.edu
**Hemant Jeevan Magadum**, ITNS, CDAC, Thiruvananthapuram, India. Email: hemant@cdac.in



## II. ARCHITECTURE OF CPU MODEL

### A. Micro- Architecture of CPU

The Micro-architecture model of the CPU is shown in figure 1. It has five stages of pipeline, in-order to have the processor execution, having configurable control path and data path in micro-architecture design. The five stage pipeline includes Fetch, Decode, Execute, Memory and Write back.

- Fetch stage: Fetching the instructions from the instruction memory by incrementing the value of program counter by four in sequential order

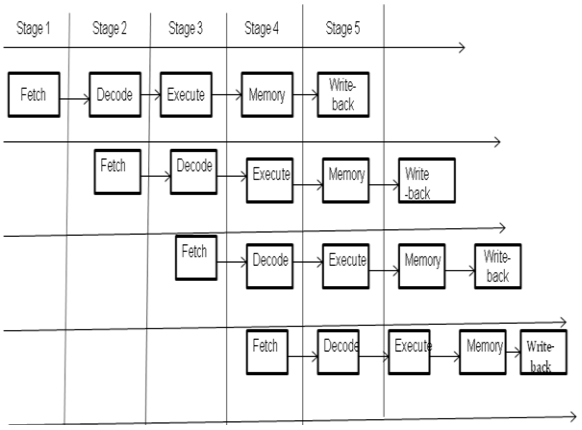


**Fig. 1.Micro-architecture of CPU**

- Decode stage: fetched instruction should be decoded as per the given instruction format and can be identified by the opcode and operands of the instruction, at this stage designer will come to know what operation should be performed and also operand addresses.
- Execute stage: Based on the op-code, the operation will be performed by reading the operands from the particular location of the memory. At this stage sometimes will get the results and some time will check the conditions based on the instruction
- Memory stage: This micro-architecture supports memory load and store operations. If instruction is memory load type, the content from the respective location of the memory must be loaded into register. Suppose, if the instruction is store type the content from the register must be stored into memory.
- Write-back: In the execution stage result of the instruction is obtained that can be written into register file at the write back stage.

### B. High Performance Micro-Architecture CPU Model

In order to get the high performance architecture model, modelling of different micro-architecture simulation models by running the Gem5 simulator loaded with the developed work function are carried out. Deeper pipeline stages can support branch prediction, different executions units are in parallel to achieve higher throughput and also integrating with separate multiplier for multiplication and division operations .Register renaming technique is used for to overcome data hazards in pipeline. By configuring the configuration parameters such as Cache memory sizes, Cache line size, Cache line replace policy, Main memory to Cache memory mapping techniques of L1&L2 Cache memory enables the performance enhancement features such as Memory Management unit, Branch prediction logic, Cache line pre-fetcher, Snoop control unit and main memory to cache memory mapping techniques, and memory controller. These are interfaced with high performance bus for internal operations. The high performance micro-architecture rely upon the above mentioned architecture features to improve the performance. The figure 2 shows the high performance micro-architecture details

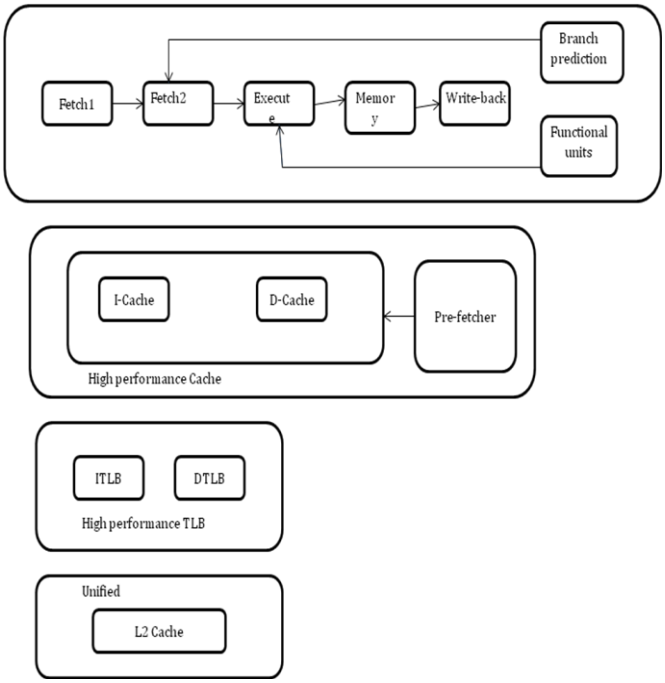


**Fig.2. High performance Micro-architecture**

- Branch Prediction:
- Memory management unit
- Cache line Prefetcher
- Instruction and Data TLB
- Snoop Control unit:
- Floating point unit
- L1 Caches
- L2 Caches

- Branch Prediction
  Branch prediction is one of the performance improvement technique it predicts whether the branch will be taken or not taken. Proper implementation of branch prediction technique will avoid stalls in the pipeline and improve the performance
- Memory management unit:
  Memory Management Unit is an address translation mechanism to translate virtual address to physical address. So that each virtual address can be mapped to the corresponding physical address and some of the virtual addresses and its corresponding physical addresses can be maintained in SRAM memory is called Translation Look-aside Buffer (TLB). In general most of the embedded systems have limited main memory in order to access that we have to enable Memory management unit. For each virtual address that is generated by the processor first search into the TLB of its corresponding physical address, if it is available in TLB, it can reduce the memory access time and improve the performance
- Cache line Pre-fetcher:
  It plays a vital role to pre-fetch the expected data based on locality of reference from main memory in advance by evicting the cache line from the cache memory based on evection policy. It increases the hit rate and improves the performance
- Instruction and Data TLB:
  Micro-architecture supports separate instruction and data memory for each Translation Look-aside Buffer to maintain the virtual address to physical address mapping. Whenever we have to access the same address reference again, it can access instruction or data very fast
- Snoop control Unit:
  Cache coherency is one of the important feature in multi-core processor architecture that means shared resource data should be same for each core local cache memory. Suppose any one of the core changes the data in that local memory L1 that should be informed to the remaining processors on the bus through any one of cache coherence protocol like MOSI, MOS, and MESI implementing cache coherence processor performance is being improved.
- Floating point unit:
  High performance micro-architecture integrating with floating point unit to perform the floating point operations. Most of the real time embedded systems which can support analog parameters must be processed to get the accurate values. For commercial and business applications it gives better accuracy so that better performance can be achieved.
- L1 Cache
  In memory hierarchy, L1 Cache memory is small size and has fast access time, the processor can access it very fast so that latency can be reduced. High performance micro-architecture supports Harvard architecture that is the reason it has separate instruction cache and data cache. Based on locality of reference principle instruction or data can be pre-fetched into cache memory. Temporal locality refers to same data can be reused in near future and spatial locality refers to next address of data can be used near future. Even though L1 cache size is small, pre-fetched data or instruction can be filled with the cache eviction line using cache line replacement policy. The internal organization of cache memory features such as cache line size, line replacement policy and mapping techniques are analyzed by running the Gem5 simulator with application work function.

*Retrieval Number: F9352038620/2020©BEIESP*
*DOI:10.35940/ijrte.F9352.038620*
*Journal Website: www.ijrte.org*



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*

Configuring the different sizes of the L1 cache sizes, line sizes, replacement policies arrived at the reference model of the design the size of the L1 is 256KB, line size of 64bytes, 8-way set associativity mapping and least recently used line replacement policy and observed the performance improvement

- L2 Caches

  L2 cache is unified cache in the high performance micro-architecture design. It is also functions based on the principle of locality of reference. Processor unable to find data in the L1 that is instruction or data that means L1 Cache miss occurs then has to search the instruction or data in L2 Cache memory. In L2 Cache hit occurs data can be loaded in L1 Cache or processor can read data directly. Whenever cache line is reallocated in L1 cache the evicted cache line written back to L2 cache for future purpose. Suppose miss occur in L2 cache than data bring back from main memory to the L2 Cache memory that can be taken care of by the cache controller. In high performance micro-architecture design L2 cache size, line size, mapping technique and replacement policies are fixed 512KB, 64bytes, 8-way set associativity and least recently used line replacement policy respectively and observed the performance improvement in the design

## III. RISC-V: ISA OVERVIEW

RISC-V is an open source architecture and was developed by the Barkly University of California. RISC-V ISA was initially designed to support academic institutions and computer architecture researches and slowly evolved to become a standard architecture for commercial applications. Currently academic institutions like IIT-Madras, IIT-Bombay and other universities focusing on RISC-V open source to develop their applications in different purposes. IIT-Madras is well advance in design aspect, they have already tape-out RISC-V architecture based 64-bit Shakti processor on silicon. Some of the multinational companies they are in design race to develop commercial applications. Private firms are in well advanced in their designs by using RISC-V ISA

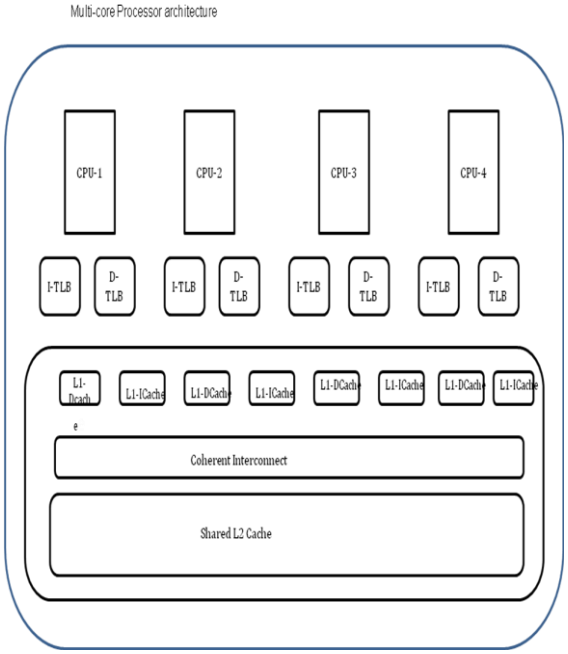


**Fig.3 RISC-V Architecture for mutli-core**

RISC-V ISA Supporting features

- Rich instruction set
- Load store architecture
- Supports IEEE 754 floating point
- Multi-core support
- Supports vendor tool set
- Real time applications
- support's industry and Academia

It supports 32-bit and 64-bit address space for different applications, OS support, and hardware implementations.

## IV. PERFORMANCE EVALUATION

### A. Proposed Methodology

Proposed methodology to evaluate the processor performance metrics using Gem5 simulator. Study of Gem5 simulator and its environment features is carried out and using which high performance micro-architecture model is simulated effectively by running different simulation models for micro-architecture details of RISC-V processor architecture. On observing the simulator parameters, to get the expected values, have to reconfigure with difference values for better performance by repeating the process with different configuration parameters and arrived at reference models of the design. The high performance micro-architecture features are shown in figure 3.

### B. Building Gem5

- Gem5 is an event driven computer system simulator
  - Components can be placed as per the requirement using configuration file parameters, can be initialized and reconfigured. Figure 4 shows the configuration

```
[root]
type=Root
children=system
eventq_index=0
full_system=false
sim_quantum=0
time_sync_enable=false
time_sync_period=100000000000
time_sync_spin_threshold=100000000

[system]
type=System
children=clk_domain cpu dvfs_handler mem_ctrl membus
boot_osflags=a
cache_line_size=64
clk_domain=system.clk_domain
eventq_index=0
init_param=0
kernel
kernel_addr_check=true
load_addr_mask=1099511627775
load_offset=0
mem_mode=timing
mem_ranges=0:536870911
memories=system.mem_ctrl

[system.clk_domain]
type=SrcClockDomain
children=voltage_domain
clock=1000
domain_id=-1
eventq_index=0
init_perf_level=0
voltage_domain=system.clk_domain.voltage_domain

...

[system.membus]
type=CoherentXBar
clk_domain=system.clk_domain
eventq_index=0
header_cycles=1
snoop_filter=Null
system=system
use_default_range=false
width=8
master=system.cpu.interrupts.pio system.cpu.interrupts.int_slave system.mem_ctrl.port
slave=system.cpu.icache_port system.cpu.dcache_port system.cpu.interrupts.int_master system.system_port
```

**Fig.4 Example for config.ini file.**

## V. ANALYZING SIMULATION OUTPUT

In this paper, architecture exploration has been carried out by configuring different cache and memory configurations. The simulator outputs has been analyzed with performance evaluation as goal to arrive at optimum configurations. The various results obtained are as follows

### A. Simulation time vs iCache Size

The simulation time for varying iCache sizes has been studied. Like expected, as iCache size increases, simulation time reduces. The simulation time remains constant beyond 64KB. This indicates that instruction cache size is now high enough to store all the required instructions for the given workload.

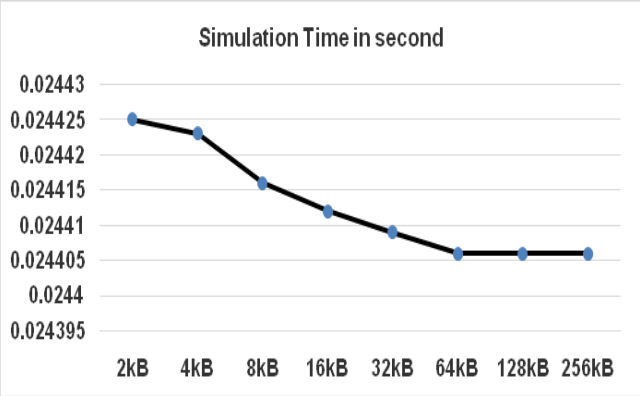


**Fig.5.Simulation time vs iCache for cacheline size of 64 bytes.**

For a cache line size of 32 bytes the simulation time is higher since more cache misses occur in comparison with a cache line of 64 bytes size. This is shown in Figure 5 and 6.

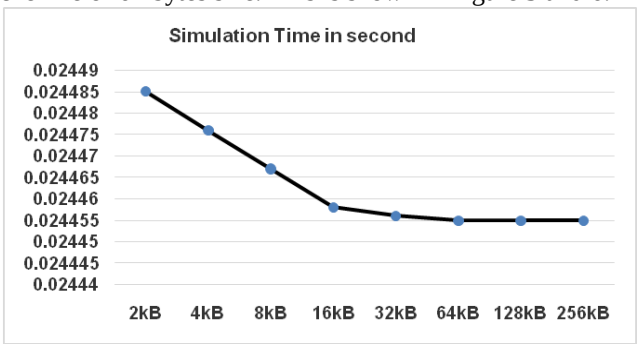


**Fig.6.Simulation time vs iCache for cacheline size of 32 bytes.**

### B. Tag replacements vs iCache size

The number of tag replacements for cachelines of 64 bytes and 32 bytes are shown in Figures 7 and 8 respectively.

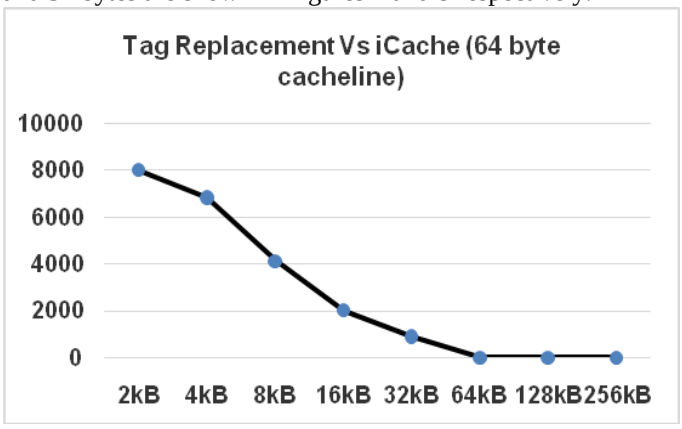


**Fig.7.Tag replacements vs iCache for cacheline size of 64 bytes.**

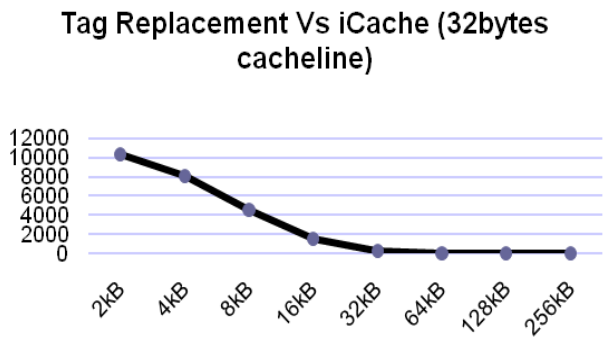


**Fig.8.Tag replacements vs iCache for cacheline size of 32 bytes.**

### C. Hit and Misses vs iCache size

As cache line size increases the miss rate reduces by taking advantage of spatial locality thereby reducing number of misses. For iCache in particular, this has more effiect due to sequentially of instructions. This is shown in Figures 9, 10 and 11.

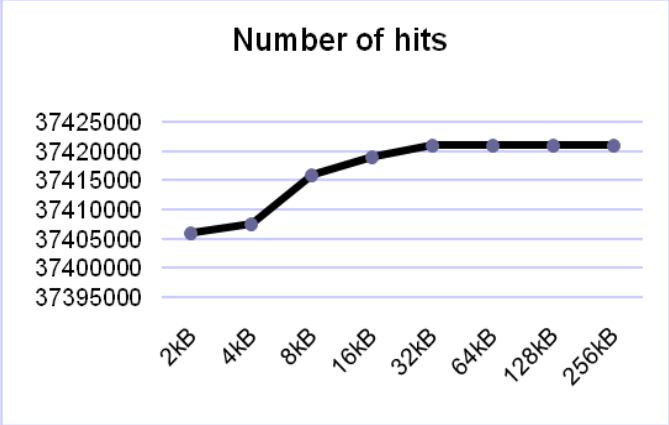


**Fig.9.Overall number of hits for iCache with 32 bytes cacheline.**

### D. Simulation time vs cacheline Size

The simulation time for varying cacheline sizes are shown. Here the iCache and dCache were fixed at 32 KB sizes.

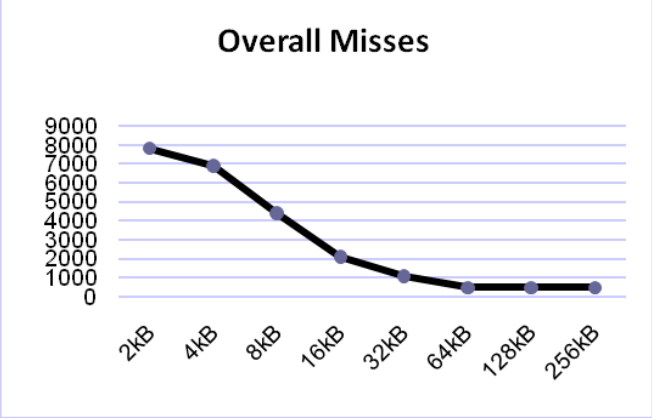


**Fig.10.Overall number of misses for iCache with 64 bytes cacheline.**

*Retrieval Number: F9352038620/2020©BEIESP*
*DOI:10.35940/ijrte.F9352.038620*
*Journal Website: www.ijrte.org*



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication*

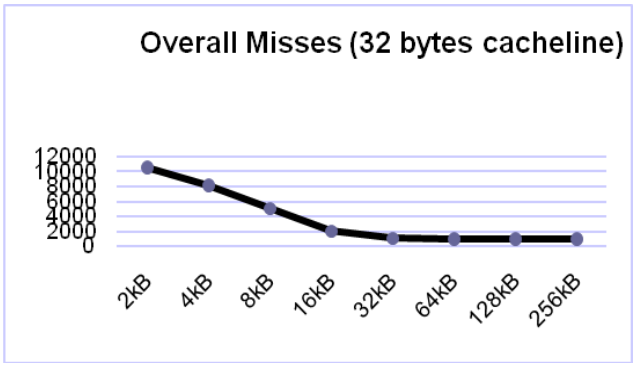


**Fig.11.Overall number of misses for iCache with 32 bytes cacheline.**

## VI. RESULT AND ANALYSIS

### A. Gem5 Results

The Gem5 simulator outputs were analysed to arrive at optimal L1 cache architecture. Based on graphical analysis shown in previous chapter L1 cache architecture with 32 bytes cache line an instruction cache of 16KB and data cache of 16KB is found to give optimum performance in terms of execution time, hit rates, miss rate and miss latencies.

### B. Functional Verification of instruction cache

The given direct mapped instruction cache has been verified in QuestaSim using UVM. The developed UVM test bench is shown in Figure 12. The verification for icache reset operation and cache write has been implemented and the obtained coverage is shown in Figure 13.

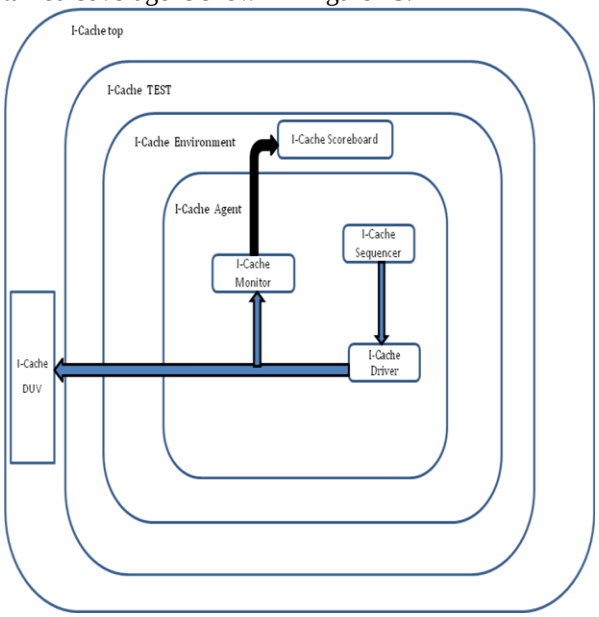


**Fig.12.UVM implemented for iCache verification.**

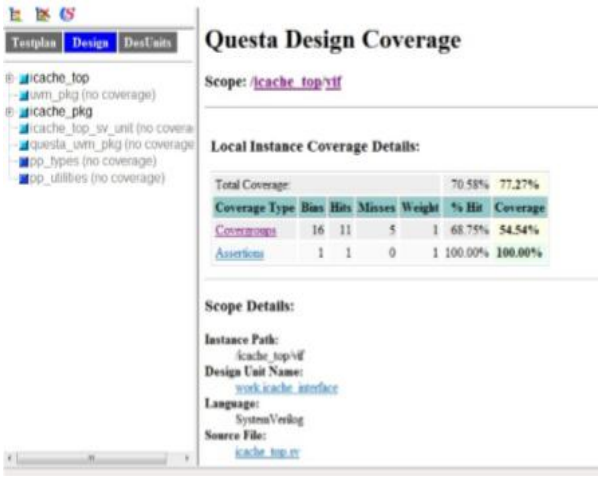


**Fig.13.Overall Functional Coverage obtained.**

## VII. CONCLUSION

Performance evaluation is a challenging task for design engineers. Gem5 simulator is a great tool which aids in design space exploration of processor architectures. In this paper, Performance evaluation has been carried out for various configurations in Gem5 simulator to find an optimal configuration. Focus of work has been on analysis of L1 Cache properties for a given RISC-V architecture. Functional verification of an instruction cache developed based on the proposed architecture has also been carried out using UVM methodology and the obtained coverage has been presented. The future work aims at developing an extensive UVM test bench from the implemented test bench for complete verification of a given processor architecture.

## REFERENCES

1. Nathan Binkert, Bradford Beckmann,Gabriel Black, Steven K. Reinhardt , Ali Said, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell,Muhammad Shoaib, Nilay , Vaish, Mark D. Hill, and David A. Wood. "The Gem5 Simulator" in ACM SIGARCH Computer Architecture News, Vol 39, 2011, pp. 1-6.
2. Anastasiia Butko, Rafael Garibotti, Luciano Ost ,and Gilles Sassatelli. "Accuracy Evaluation of GEM5 Simulator System" in 7nth International Workshop on reconfigurable communication –centric systems on chip, 2012.
3. Shagufta,Muhammad Aleem,Muhammad Arshad Islam and Muhammad Azhar Iqbal, "A Comparative Study of Heterogeneous Processor Simulators" in International Journal of Computer Application, (0975 8887), vol 148,2016.
4. Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu,Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill and David A. Wood, "Multifacets General Execution-driven Multiprocessor Simula tor (GEMS) Toolset" in ACM SIGARCH Computer Architecture News, vol. 33 No. 4, 2005.
5. Pedro Araujo, "Development of a reconfigurable multi-protocol verification environment using UVM methodology", 2014.
6. Alan Ma and Allan Zacharda, "Utilizing SystemC for Design and Verification" Mentor Graphics.
7. Andrew rew, Waterman and Krste Asanovi, "The RISC-V Instruction Set Manual Volume I: User-Level ISA", CS Division, EECS Department, University of California, Berkeley, May 7,2017.
8. The gem5 simulator. Available: http://www.gem5.org

## AUTHORS PROFILE

**Lakshmaiah Alluri** received the Btech, Mtech degree in Electronics and communication. He registered PHD in NIT Trichy. Currently working in CDAC, Thiruvananthapuram. His research interest includes VLSI, microprocessor design and Real time systems.

**M, Bhaskar** received the B.E. degree in Electronics and Communication Engineering from Bharathiar University, Coimbatore, India, in 1992, M.E. degree in Microwave and Optical Engineering from Madurai Kamaraj University, Madurai, India in 1995 and Ph.D. Degree in High speed on-chip Interconnect design from National Institute of Technology, Tiruchirappalli, India in 2015. He worked as project trainee in Antenna Division, Indian Space Research Organization (ISRO), Bangalore, India from July 1994 to February 1995. He worked as faculty in Shanmuga College of Engineering (now Sastra University), Thanjavur, India from June 1995 to April 1997. Since 1997, he has been with the faculty of the National Institute of Technology, Trichy (Formerly known as Regional Engineering College, Trichy). Currently he is the Professor of the Electronics and Communication Department. He has coauthored one book and has published numerous papers in Journals, international and national conferences out of which several of them have been selected for the best paper award. He has done many projects using Programmable DSPs. His research interests include Architecture and applications of DSPs, On-chip RF wireless transceivers, Low power system on a single chip (SOC) design, the design and performance analysis of high speed transceivers for on-chip interconnects and VLSI design for Biomedical.

**Hemant Jeevan Magadum** received the B.E. degree in Electronics. Currently working in CDAC, Thiruvananthapuram. His research interest includes embedded development, Real time application and Intelligent transportation system.