# Analysis and Optimization of the Branch Prediction Unit of SweRV EH1

Changbiao Yao[1*], Ziqin Meng[2], Wen Guo[2], Jianyang Zhou[2], Zichao Guo[2]
[1] Ningbo Fotile Kitchenware Co., LTD., Ningbo, China
[2]School of Electronic Science and Engineering, Xiamen University, Xiamen, China
[*]yaochangbiao@163.com

2022 IEEE 16th International Conference on Anti-counterfeiting, Security, and Identification (ASID) | 978-1-6654-9067-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASID56930.2022.9996038

***Abstract*—With the continuous improvement of processor performance requirements, technologies such as superscalar, deep pipeline, and multi-core which can improve instruction parallelism are frequently used. Under this technical background, branch prediction errors will increase the delay used to flush the pipeline and greatly reduce the performance of the processor. Therefore, for high-performance processors, branch predictors with high prediction accuracy are particularly important. Based on the open source RISC-V processor core SweRV EH1, this paper adopts two prediction predictors, the hybrid predictor, and the TAGE predictor to improve the prediction performance of the original processor. This paper uses the riscv-tests self-checking test scheme to verify the instruction set of the optimized processor and completes the prototype verification on the Kintex-7 KC705 FPGA. Based on PowerStone and CoreMark test programs, this paper separately evaluates the branch prediction performance and processor performance of the processor core with two kinds of branch predictors. Experiments show that the implementation of the hybrid predictor and the TAGE predictor respectively improves the branch prediction accuracy of PowerStone programs by 3.65% and 3.39%; the average branch prediction rate respectively reaches 85.98% and 90.06%. The performance of SweRV EH1 is respectively improved by 2.56% and 5.43%.**

***Keywords—Gshare Branch Predictor; Hybrid Branch Predictor; TAGE Branch Predictor; Branch Prediction***

## I. INTRODUCTION

In modern times, in order to achieve better performance, more and more processor designs tend to increase the parallelism of instructions [1]. High-performance processors usually use superscalar and deep pipeline architectures [2]. These two architectures are important ways to improve the degree of parallelism of instructions, and greatly increase the throughput of instructions [3]. Superscalar technology makes different pipelines execute different instructions in the same cycle and Deep pipeline technology makes different instructions executed in different pipeline stages of the same cycle. As the frequency of superscalar and deep pipeline usage increases, especially as speculative execution becomes popular [4], branch instructions will become increasingly important in determining processor performance. Since superscalar usually has multiple instruction fetch bandwidths, each prediction is made for one instruction fetch group, which greatly increases the number of instructions entering the pipeline at the same time, and also increases the penalty time for branch prediction errors. A processor with a deep pipeline architecture needs to flush too many instructions when the prediction is wrong, and the processor performance will also be greatly lost. This performance penalty even offsets the performance gains from deep pipelining and high fetch bandwidth. Therefore, for high-performance processors, the accuracy of the branch prediction module is very important, and some scholars even believe that the prediction of branch instructions may become a bottleneck restricting processor performance in the future [5].

The branch prediction technology is mainly used to reduce the impact of conflicts related to branch jump instructions [6]. It is to predict the jump direction and the jump target address of the branch instruction, but its implementation will be very different for different instruction set architectures and processor architectures. Therefore, during the development period of processors, many feasible prediction schemes have also been produced for different processor architectures. When adopting these prediction schemes, it is necessary to make adaptive changes and optimizations according to their processor architecture, so that the prediction performance of the prediction schemes can reach the best.

As the research target of this paper, SweRV EH1 is a RISC-V processor core open-sourced by Western Digital Corp. Since SweRV EH1 is a superscalar processor with a nine-stage pipeline, it has higher requirements for the accuracy of branch prediction. The SweRV EH1 branch prediction unit adopts the Gshare prediction scheme, which can not dynamically select an appropriate predictor according to the behavior of branch instructions, resulting in an unsatisfactory branch prediction accuracy. Based on the hardware platform of SweRV EH1, we used the hybrid predictor and TAGE predictor to optimize the branch prediction unit of SweRV EH1.

## II. BRANCH PREDICTION THEORY OF SWERV EH1

SweRV EH1 is based on the open source RISC-V instruction set architecture. The jump instructions in the RISC-V instruction set include unconditional jump instructions jal and jalr, and conditional jump instructions beq, bne, blt, bge, bltu and bgeu. This paper is mainly aimed at the research of jump direction prediction of the conditional jump instruction.

SweRV EH1 branch prediction unit adopts Gshare predictor for the prediction of conditional branch instruction direction. This method is designed based on 2bit BHT (Branch Prediction

978-1-6654-7735-2/22/$31.00 ©2022 IEEE

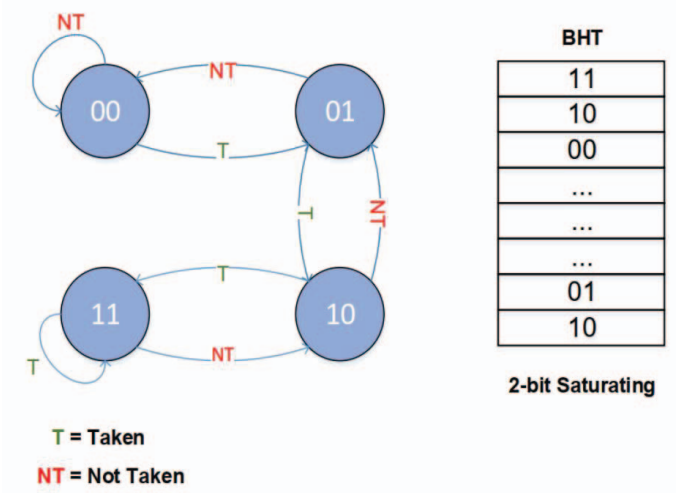


Figure 1. The structure diagram of 2bit BHT

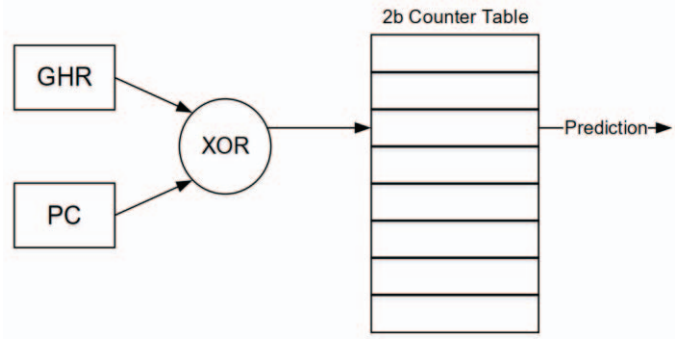


Figure 2. The structure diagram of Gshare predictor

Table). As shown in Fig. 1, 2bit states can be stored in each entry of 2bit BHT. There are 4 possible states in total, 00 (strong no jump), 01 (weak no jump), 10 (weak jump), 11 (strong jump). When predicting, the 2bit BHT is indexed by the low bits of the PC (Program Counter).The structure of the Gshare predictor is shown in Fig. 2. This prediction technique can take into account the jump history of all recent jump instructions, thereby improving the accuracy of branch predict-ion. The Gshare prediction scheme indexes the 2bit BHT through the XOR result of the PC value and the GHR (Global History Register), and finally obtains the prediction result. Compared with 2bit BHT, the Gshare predictor largely resolves the conflict of branch aliases. However, this prediction scheme needs a long time for training before reaching a stable state. For programs with frequent context switching, this prediction scheme cannot achieve ideal branch prediction performance.

## III. Design and Functional Verification of Hybrid Predictor Based on SweRV EH1

In order to improve the prediction ability of the original platform Gshare prediction scheme, we use a hybrid predictor that can select different predictors according to the branch instruction behavior to improve the overall performance of the branch prediction unit [7]. This design uses an effective dynamic selection mechanism for different branch instructions.

### A. Design Implementation

The hybrid predictor in this paper uses two predictors, Bimodal predictor and Gshare predictor. Bimodal predictor has a better prediction accuracy for loop branches with strong jump bias, while Gshare predictor has a better prediction accuracy for dependent branch instructions. By adding the Bimodal predictor to the original prediction unit of SweRV EH1, the prediction accuracy of the processor during the training phase of the Gshare predictor can be improved.

The design of the hybrid predictor in this paper is shown in Fig. 3. The Gshare prediction table is indexed by the XOR result of the PC and the GHR, the Bimodal prediction table is indexed by the PC, and the Select Table prediction table has the same structure as the Bimodal prediction table. When the instruction fetch unit issues an instruction fetch request, the PC will also be generated at the same time. Finally, the prediction results of the Bimodal predictor and the Gshare predictor are selected through the prediction results of the Select table.

In addition to designing the basic branch prediction architecture, the design of the branch prediction unit also needs to be implemented according to the characteristics of the instruction fetch unit. As a dual-issue superscalar processor, SweRV EH1 is different from ordinary processors that only fetch one instruction per cycle. Each instruction fetch cycle will fetch 128bit instructions from the I-Cache. Because SweRV EH1 supports RV32IMC instruction set architecture, the 128bit instructions may contain 32bit instructions or 16bit compressed instructions. In the worst case, one prediction needs to predict 8 instructions, so the hybrid predictor provides 8 corresponding prediction tables for these 8 instructions.

### B. Dynamic Simulation Verification

In order to ensure the correct function of the optimized processor, we did a dynamic simulation verification of the processor. The test set used in the simulation is riscv-tests [8], which is an instruction set self-checking test library officially released by RISC-V. The programs for these tests consist of assembly language. The program uses the instruction to be tested to calculate the pre-defined data and compares the obtained result with the pre-determined standard answer. If it is inconsistent, the instruction test fails. When each test set passes, the characters "test pass" will be printed to the terminal. Fig. 4 is a partial record of the optimized processor running the riscv-tests program. After simulation, the programs of riscv-tests are all passed.

### C. FPGA Prototype Verification

Prototype verification is a verification method used to verify the function and performance of Application-Specific Integrated Circuits (ASICs), Application-Specific Standard Products (ASSPs), and Systems-On-Chips (SoC). We integrate SweRV EH1 into a microcontroller MCU shown in Fig. 5 and connect the processor to the URAT serial port through the AXI bus. The FPGA model used for prototyping is Kintex-7 KC705. We use the Vavido tool to simulate, synthesize, and lay out the MCU in sequence to generate a bit file and finally download the bit file to the FPGA. The result entered into the terminal is shown in Fig. 6. It can be seen that after the MCU executes the

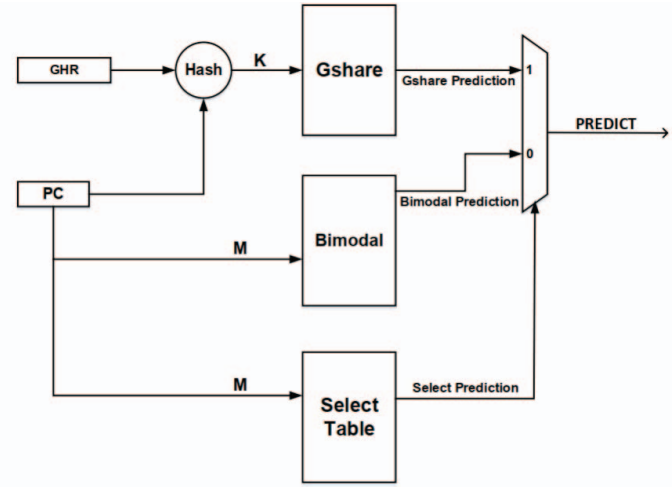


Figure 3. The structure diagram of hybrid predictor

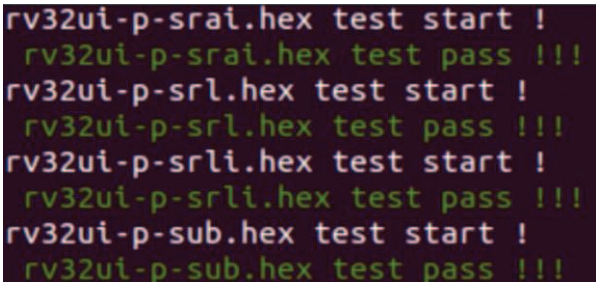


Figure 4. The result of riscv-test

HelloWorld program, it successfully outputs the running result to the terminal through UART.

## IV. Design and Functional Verification of TAGE Predictor Based on SweRV EH1

In addition to the implementation of the typical hybrid predictor, this paper also implements the more advanced TAGE predictor to explore the adaptation of the advanced branch predictor to the embedded processor. The full name of the TAGE predictor is Tagged GEometric History Length Branch Prediction, which uses a geometrically increasing history length to index the prediction component. The TAGE predictor uses the prediction component with the tag hit and the longest branch history to give the final prediction. If each component is not hit, the basic prediction component will give the prediction result. Typical processors that use this branch predictor are AMD Zen2 [9] and XiangShan [10].

### A. Design Implementation

A TAGE predictor consists of a base predictor T0 that provides default prediction results and a set of predictor components Ti with tag fields. Fig. 7 is a schematic of the TAGE predictor [11]. The base predictor T0 is usually a Bimodal predictor. The prediction component Ti with the tag field is indexed using a geometrically increasing global history length. On the TAGE predictor, most branch jump states are stored in prediction tables with shorter history, but the predictor is able to capture dependencies between very distant

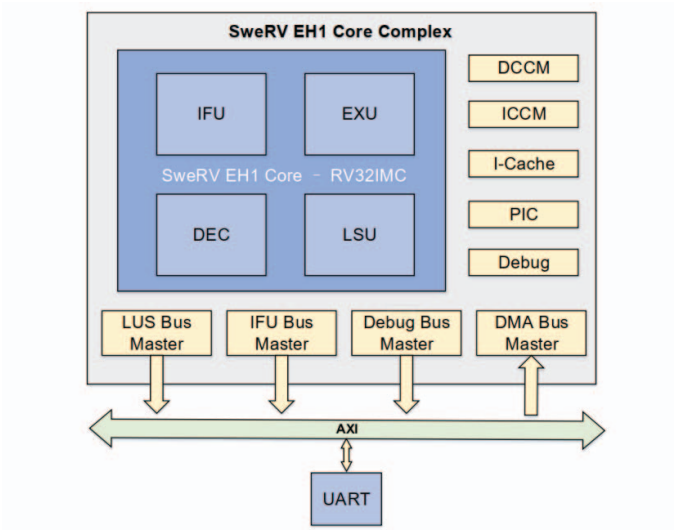


Figure 5. The system architecture of MCU

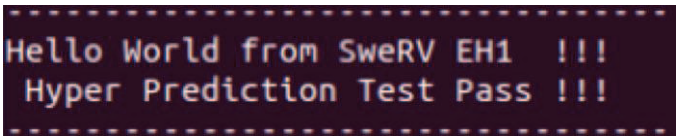


Figure 6. The result of FPGA prototype verification

branches and recent branches Through this compatibility, the prediction performance of the predictor can be improved. The final completed TAGE predictor on SweRV EH1 also predicts the instruction fetch group. In the worst case, the TAGE predictor will give 8 prediction results, corresponding to 8 16-bit instructions in the instruction fetch group.

### B. Dynamic Simulation Verification

The dynamic simulation method for this design is the same as in Chapter 3. After testing, the design has passed riscv-tests simulation.

### C. FPGA Prototype Verification

The FPGA prototyping method for this design is the same as Chapter 3. The difference is that all three prediction tables in the hybrid predictor are implemented through registers, which are implemented as LUTs during FPGA prototyping. In the branch unit based on the TAGE predictor, the capacity of the TAGE look-up table is very large, and when it is implemented with registers, there will be a problem of insufficient resources during prototype verification, so each prediction table of the TAGE predictor is implemented in SRAM. After testing, the program runs correctly.

## V. Performance Evaluation

We used the benchmark program PowerStone to test the processor and obtained the accuracy of branch prediction before and after optimization by summarizing the total number of branch instructions and the number of mispredicted branch instructions through the performance simulator. At the same time, in order to study the impact of branch prediction improv-

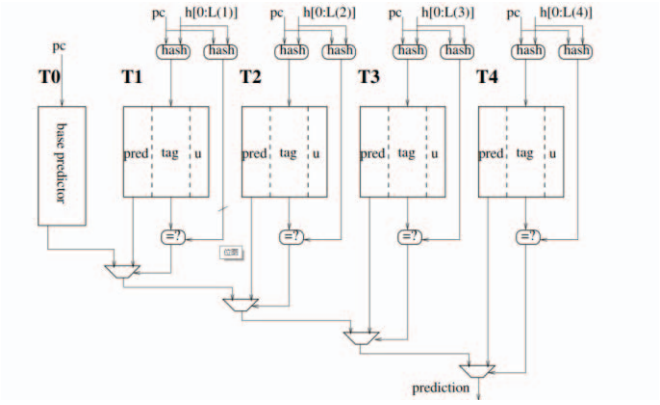


Figure 7. The structure diagram of TAGE predictor

ement on processor performance, we made the CoreMark benchmark test on the processors before and after optimization and measured the change of processor performance through the results of the CoreMark program.

### *A. The Result of Running PowerStone*

PowerStone is a benchmark program mainly used for embedded processors. After being compiled by the compiler GCC, the main types of branches that run in the processor are branch instructions, logical operation instructions, and load/store instructions. According to our statistics, the conditional branch instructions in the PowerStone test program account for 16.47% of all instructions. This high proportion makes this test program very suitable for testing the branch prediction function. The PowerStone test program is a series of programs, including programs with various functions such as data encryption, signal filtering, floating point, square root calculation, etc.

The results of all test sets of PowerStone are limited in space, so they are not listed here. Hybrid predictor was tested by running 100 rounds of PowerStone program and TAGE was tested by running 500 rounds of PowerStone program. It can be seen that the average prediction accuracy after using the hybrid predictor is 85.98%, the average branch prediction performance is 3.65% higher than before. The average prediction accuracy after using the TAGE predictor is 90.06%, the average branch prediction performance is 3.39% higher than before.

### *B. The Result of Running CoreMark*

Performance evaluation not only needs to test the branch prediction accuracy before and after optimization but also needs to evaluate the overall performance of the processor. Coremark is a typical test program, which is designed in C language and contains a variety of classic algorithm tests, such as matrix operations and linked list operations. Its test results are also very representative for processors. The results of running the CoreMark program for the two predictors are shown in Fig. 8 and Fig. 9.

It can be seen that in the case of iterating the CoreMark program once, the performance of the processor before branch unit optimization is 4.53 CoreMarks/MHz and the branch pred-

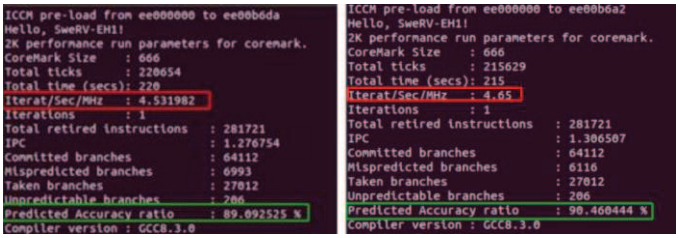


Figure 8. The CoreMark result of hybrid predictor

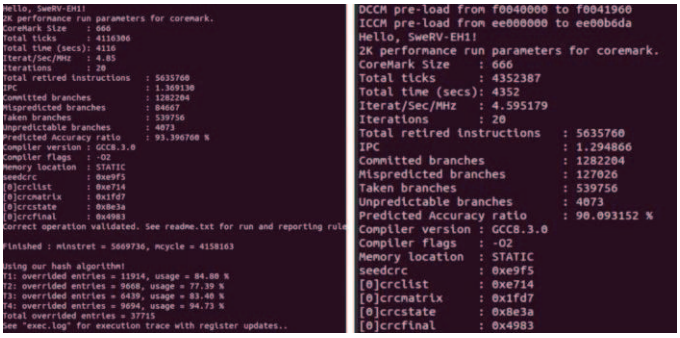


Figure 9. The CoreMark result of TAGE predictor

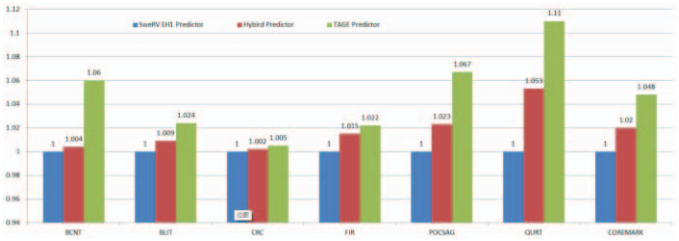


Figure 10. Comparison of the prediction rates of the three predictors

iction accuracy is 89.09%. The performance of the processor using the hybrid predictor is 4.65 CoreMarks/MHz and the branch prediction accuracy is 90.46%. The performance is improved by 2.65% and the branch prediction accuracy is improved by 1.37%. In the case of iterating the CoreMark program 20 times, the processor performance before branch unit optimization is 4.60 CoreMarks/MHz, the branch prediction accuracy rate is 90.09%. The performance of the processor using TAGE is 4.85 CoreMarks/MHz and the branch prediction accuracy rate is 93.40%. The performance is improved by 5.43% and the branch prediction accuracy rate is improved by 3.31%.

### *C. Comprehensive Analysis and Comparison*

The branch prediction performance of the SweRV EH1 predictor, the hybrid predictor, and the TAGE predictor when the training is mature are shown in Fig. 10. It can be seen from the figure that when the training is mature, the overall prediction performance of the TAGE predictor is higher than hybrid predictor and the original Gshare predictor, and the overall performance of the hybrid predictor is higher than that of the original Gshare predictor. In the previous comparison, both the Gshare predictor and the hybrid predictor are measured when the test program is executed 100 cycles. In this case, the branch prediction performance of the hybrid predictor

is 3.65% higher than that of the Gshare predictor, which means that the hybrid predictor has a faster response time than Gshare predictor. However, when the training is mature, the performance improvement of the hybrid predictor relative to the Gshare predictor is not significant.

Under the same compilation options, based on different prediction schemes, the performance results obtained by the SweRV EH1 processor when executing the benchmark program CoreMark are 4.53 CoreMark/MHz, 4.65 CoreMark /MHz, and 4.85 CoreMark/MHz respectively. It can be seen that the branch predictor can improve the overall performance of the processor to a certain extent, and the higher the prediction accuracy of the predictor is, the more the performance of the processor is improved.

## VI. CONCLUSION

This paper mainly focuses on the branch prediction unit of the SweRV EH1 processor and focuses on how to improve the prediction accuracy of the jump direction of conditional branches and the influence of the branch prediction accuracy on the performance of the processor. In this paper, we optimized the branch prediction unit of SweRV EH1 by using the method of hybrid predictor and TAGE predictor, carried out the dynamic simulation verification of the processor by using the method based on the riscv-tests test set, and completed the prototype verification of the processor on Kintex-7 KC705 FPGA.

After using the hybrid predictor, the average branch prediction accuracy of SweRV EH1 reached 85.98% when executing 100 rounds of PowerStone test program, which was 3.39% higher than before. When executing 1 round of CoreMark test program, the processor scored 4.65 CoreMarks /MHz, which is 2.65% higher than before optimization. In addition, after using the TAGE predictor, the average branch prediction accuracy of SweRV EH1 reached 90.06% when executing 500 rounds of PowerStone test program, which was 3.5% higher than before. When executing 20 rounds of CoreMark test program, the processor scored 4.85 CoreMarks /MHz, which is 5.43% higher than before optimization. It can be seen that both the hybrid predictor and the TAGE predictor can improve the branch prediction accuracy of the processor, and the more the prediction accuracy is improved, the more significant the improvement in processor performance will be.

## REFERENCES


[1] N. P. Jouppi, D. W. Wall, “Available instruction-level parallelism for superscalar and superpipelined machines,” ACM SIGARCH Computer Architecture News,1989.

[2] N. J. Wang, J. Quek, T. M. Rafacz, and S. J. Patel, “Characterizing the effects of transient faults on a high-performance processor pipeline,” 2004 International Conference on Dependable Systems and Networks (DSN 2004), Proceedings IEEE, 2004.

[3] Z. Qiang, “Design and optimization of branch prediction unit for Boom processor,” XiDian University, 2020.

[4] S. McFarling, “Combining branch predictors,” Digital Western Research Laboratory, Tech. Rep., 1993.

[5] P. Ranganathan, “The relative importance of memory latency, bandwidth, and branch limits to performance,” 1999.

[6] A. Pandey, “Study of data hazard and control hazard resolution techniques in a simulated five stage pipelined RISC processor.” 2016 International Conference on Inventive Computation Technologies (ICICT) IEEE, 2017.

[7] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeides, “Design tradeoffs for the Alpha EV8 conditional branch predictor,” ACM SIGARCH Computer Architecture News, vol. 30, no.2, pp. 295-306, 2002.

[8] T. Newsome, riscv-software-src/riscv-tests, RISC-V, May 25, 2019. Accessed on: May 18, 2022. [Online]. Available: https://github.com/riscv-software-src/riscv-tests.

[9] D. Suggs, M. Subramony, B. Dan, “The AMD “Zen 2” processor,” IEEE Micro, vol. 40, no.2, pp. 45-52, 2020.

[10] B. Yungang, “XiangShan: an open-source high-performance RISC-V processor,” Tech. rep., 2021.

[11] A. Seznec, and P. Michaud, “A case for (partially) TAgged GEometric history length branch prediction,” The Journal of Instruction-Level Parallelism, vol. 8, no.1, pp. 23, 2006.