Received March 21, 2020, accepted April 27, 2020, date of publication May 6, 2020, date of current version May 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992643

A Dynamic Branch Predictor Based on Parallel Structure of SRNN

LEI ZHANG, NING WU, FEN GE, FANG ZHOU, AND MAHAMMAD REHAN YAHYA
College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

Corresponding author: Ning Wu (wunee@nuaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61774086, and in part by the Fundamental Research Funds for Central Universities under Grant NP2019102 and Grant NS2017023.

ABSTRACT Branch predictor is a key component of processor, which can improve the efficiency of instruction execution. The branch predictor based on machine learning algorithm can achieve high branch prediction accuracy, but it has the disadvantages of long training time and high access delay. As a neural network algorithm, Recurrent Neural Network (RNN) is good at processing data related to time series, and can learn the correlation between data faster. Sliced Recurrent Neural Network (SRNN) parallelizes the RNN algorithm, effectively reducing the access delay of the RNN algorithm. In this paper, a dynamic branch predictor based on parallel structure of SRNN is proposed to accelerate the training time and reduces the computing delay. The optimal design parameters of predictor, which has prediction accuracy with lower source cost, are selected through a serial simulations. The experimental results show that the branch predictor proposed in this paper has higher prediction accuracy than the traditional Bimod and Gshare branch predictors under the same hardware consumption, and its branch prediction rate is 2.34% higher than the traditional Perceptron neural predictor in the short learning period.

INDEX TERMS Branch predictor, machine learning, recurrent neural network (RNN), sliced recurrent neural network (SRNN).

I. INTRODUCTION

The branch predictor is an important part of a modern processor. A high precision branch predictor can improve performance and reduce power consumption by reducing the number of instructions executed on the wrong path [1], [2]. In view of the importance of branch prediction, branch prediction has been widely studied. Among the proposed branch predictors in the last five Championship Branch Prediction (CBP) competitions, most are variants of the TAGE and Perceptron branch predictors, However, only a few researches have explored more advanced branch prediction machine learning methods [9]. The essence of branch prediction is a classification problem, and various machine learning algorithms have shown excellent performance in other fields [3], [4]. Applying machine learning algorithm to branch predictor can significantly improve the accuracy of branch predictor [1]. Therefore, this paper studies the application of RNN algorithm in branch predictor.

The associate editor coordinating the review of this manuscript and approving it for publication was Joanna Kołodziej.

Various branch predictors based on machine learning algorithm have been proposed. Jimenez *et al* propose a neural predictor based on perceptron in [5], which uses a layer of perceptron model to represent the forward inference process of branch prediction. In [6], the algorithm is improved and a path-based prediction algorithm with higher prediction accuracy is proposed. In [7], the piecewise-linear prediction algorithm is proposed, which optimizes the prediction accuracy of the linear non-separable function. Because most programs themselves tend not to jump when they encounter branch instructions, the prediction accuracy is improved by modifying the thresholds of taken and nottaken [8]. Mao *et al* put forward a branch predictor based on deep belief net in [9], and the misprediction rate is reduced by $3\% \sim 4\%$ on average. Jeremy *et al* propose a predictor based on the Naive Bayesian algorithm in [10], which transforms multiplication operation into addition operation, reducing hardware consumption and path delay caused by multiplication operation. Tarsa *et al* proposed a branch predictor based on a convolutional neural network in [11], which improves the prediction accuracy of branch instructions that are difficult to predict. The branch predictor based on machine learning algorithm obtains high branch prediction accuracy, but it has two defects: first, the branch predictor based on machine learning algorithm can obtain high prediction accuracy only in the ''stable period'' after a long time of training, and its prediction accuracy in the ''learning period'' with a short training time is low. Although the hybrid predictor is used to improve the prediction accuracy of the ''learning period'', it consumes more hardware area [12], [13]. Second, the traditional machine learning algorithm is complex in structure, it is not easy to be implemented in parallel structure, and it has high access delay in hardware implementation.

As a neural network algorithm which is good at processing sequence data, RNN can effectively learn the correlation between sequences [14]–[16]. Applying RNN algorithm to branch predictor can speed up the learning process. SRNN algorithm can parallelize RNN algorithm structure, realize pipeline design of forward inference process, and reduce access delay [31]. Therefore, this paper constructs a dynamic branch predictor based on SRNN algorithm by using two-level prediction model [17], [18] to improve the prediction accuracy of ''learning period'' and optimizes the hardware access delay. Finally, based on the SimpleScalar [19], [20] simulation environment, the influence of different variables on the prediction effect of the predictor is studied, and the advantages of the SRNN branch predictor are analyzed. The experimental results show that the dynamic branch predictor proposed in this paper based on SRNN algorithm has a higher prediction accuracy than the traditional Bimod [21], [22] and Gshare [23] dynamic branch predictor under the same hardware consumption, and its branch prediction rate in the ''learning period'' with shorter training time is 2.34% higher than the traditional Perceptron [5] neural predictor.

The rest of this paper is organized as follows: Section 2 introduces RNN algorithm and SRNN algorithm. In the section 3, the architecture of proposed SRNN branch predictor is introduced, and the prediction process and training process of predictor are introduced in detail. Section 4 uses the control variable method to explore the optimal design parameters of the predictor. In Section 5, the prediction performance of different predictors in different training time is compared, and the advantages of this predictor are analyzed. Section 6 summarizes the conclusions of this work.

II. BACKGROUND

A. RNN

RNN is one of the neural network algorithms. Unlike convolution, full connection and other kinds of neural network algorithms, RNN is good at processing data related to time series. Its algorithm model describes the correlation between different inputs [16]. Taking text processing as an example, it simulates the process of human processing text data. When people read articles, they will form long-term memory, remember the useful information they read before, and filter out the useless information, so that they can better understand the later text [15]. The RNN is similar to this, its network structure will remember the previous information, and this information will participate in the operation of the later nodes. A typical RNN structure is shown in Figure 1.

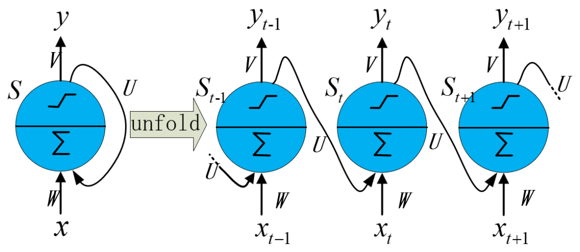


FIGURE 1. RNN algorithm structure.

```
while(true)
{
    A=7;
    B=7;
    if(A==1)    // Branch 1
      A=4;
    if(B==2)    // Branch 2
      B=4;
    if(C==3)    // Branch 3
      C=4;
    if(A==B)    // Branch 4
      D=1;
    .........   // Other code
}
```

FIGURE 2. Branch codes in program.

It can be seen from Figure 1 that the current state S_t of RNN not only depends on the input layer, but also the input from the previous state S_{t-1}, and will also transfer the current state to the next state S_{t+1}.

The calculation method of current state S_t and output y_t of RNN algorithm is shown in Equation (1).

$$\begin{aligned} S_t &= f(Wx_t + US_{t-1}) \\ y_t &= g(VS_t) \end{aligned} \tag{1}$$

Among them, f and g are activation functions [24], U represents the weight of the previous state, W represents the weight of the current input, and V represents the output weight of the current state.

In the process of program execution, each branch instruction is related to each other, and the branch direction of the branch instruction will be affected by other branch instructions. There are usually branch codes as shown in Figure 2.

There are 4 branch statements in the above code, branch 4 depends on branch 1 and branch 2 and does not depend on branch 3. If branch 1 and branch 2 are not executed, branch 4 will not be executed, and the execution result of branch 3 does not affect the execution of branch 4. The RNN algorithm can quickly learn the correlation between different branch instructions before and after. Therefore, the dynamic branch predictor based on RNN algorithm can obtain high prediction accuracy in the ''learning period''.

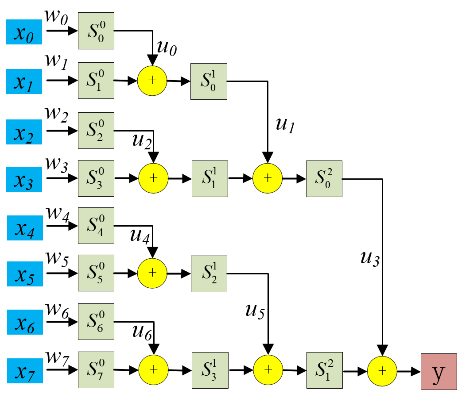


FIGURE 3. An 8-Input SRNN calculation model.

B. SRNN

Each state of RNN algorithm depends on the input of the previous state. This serial structure makes it difficult to parallelize. In the hardware implementation, the advantage of hardware parallelization can't be used to improve the calculation speed of RNN algorithm. Forward inference requires multiple cycles of calculation to complete, which is not conducive to reducing the delay of branch predictor.

In 2018, Zeping Yu *et al.* Proposed SRNN algorithm, which divides the input sequence of RNN algorithm into n subsequences, runs RNN on each subsequence in parallel, merges the output of each subsequence as a new input sequence, and repeats the above process until the calculation result is obtained [31]. The Figure 3 shows an 8-Input SRNN calculation model. The slice size of the calculation model is 2.

The calculation process of SRNN is as follows:

Multiply the 8 inputs with the input weight W to obtain the 8 intermediate states of the first layer, as shown in Equation (2).

$$\begin{aligned} S_0^0 &= f(x_0w_0) \quad & S_1^0 &= f(x_1w_1) \\ S_2^0 &= f(x_2w_2) \quad & S_3^0 &= f(x_3w_3) \\ S_4^0 &= f(x_4w_4) \quad & S_5^0 &= f(x_5w_5) \\ S_6^0 &= f(x_6w_6) \quad & S_7^0 &= f(x_7w_7) \end{aligned} \tag{2}$$

The 8 intermediate states are divided into 4 groups according to two as a group to calculate RNN, and 4 intermediate states of the second layer are obtained, as shown in Equation (3).

$$\begin{aligned} S_0^1 &= f(S_1^0 + u_0S_0^0) \quad & S_1^1 &= f(S_3^0 + u_2S_2^0) \\ S_2^1 &= f(S_5^0 + u_4S_4^0) \quad & S_3^1 &= f(S_7^0 + u_6S_6^0) \end{aligned} \tag{3}$$

The 4 intermediate states are also divided into 2 groups according to two as a group, and RNN calculation is carried out respectively to obtain 2 intermediate states of the third layer, as shown in Equation (4).

$$S_0^2 = f(S_1^1 + u_1S_0^1) \quad S_1^2 = f(S_3^1 + u_5S_2^1) \tag{4}$$

Finally, RNN calculation is carried out for the 2 intermediate states output by the third layer, and the final output result is obtained, as shown in Equation (5).

$$y = g(S_1^2 + u_3S_0^2) \tag{5}$$

Like RNN, f and g are activation functions. SRNN sliced and layered the RNN algorithm. The slicing structure is very suitable for parallel computing, and the layered structure is also conducive to pipeline design.

III. THE ARCHITECTURE OF PROPOSED SRNN BRANCH PREDICTOR

The dynamic branch predictor based on the parallel structure of SRNN proposed in this paper adopts a two-level predictor model [17], [18], which consists of a Global History Register (GHR) and a Pattern History Table (PHT). PHT table stores U weight $u_0, \ldots, u_{n-1}$ and W weight $w_0, \ldots, w_n$ required by SRNN algorithm, and global branch history is stored in GHR register. Its structure is shown in Figure 4.

When calculating branch prediction value y, a parallel RNN computing circuit based on SRNN is designed. The SRNN computing circuit consists of multiplier, accumulator and pipeline register. The circuit divides the pipeline stage according to the number of layers of SRNN, and each layer of SRNN can carry out parallel computing.

A. PREDICTION PROCESS

The prediction process of the branch predictor is mainly divided into two steps. First, the weight parameters $u_0, \ldots, u_{n-1}$ and $w_0, \ldots, w_n$ are obtained from the PHT table by hashing the PC values of branch instructions. Then, the SRNN algorithm was used to calculate the predicted value y by combining the obtained weight parameters with the branch history $x_0, \ldots, x_n$ stored in the GHR register. If $y > 0$, the result of branch prediction will jump. If $y < 0$, the result of branch prediction will not jump.

The calculation model of y simplifies some parameters in the SRNN algorithm. Take the activation function f as a linear function $f(x) = x$, take the output activation function g as a linear function $g(x) = x$. Assuming that the value in GHR is (1,−1,1,1), the weight parameter U is (1,2,3), and W is (−2,10,1,5), the calculation process of y is as shown in Equation (6).

$$\begin{aligned} S_0^0 &= x_0w_0 = 1 \times -1 = -2 \\ S_1^0 &= x_1w_1 = -1 \times 10 = -10 \\ S_2^0 &= x_2w_2 = 1 \times 1 = 1 \\ S_3^0 &= x_3w_3 = 1 \times 5 = 5 \\ S_0^1 &= S_1^0 + u_0S_0^0 = -10 + 1 \times -2 = -12 \\ S_1^1 &= S_3^0 + u_2S_2^0 = 5 + 3 \times 1 = 8 \\ y &= S_1^1 + u_1S_0^1 = 8 + 2 \times -12 = -16 \end{aligned} \tag{6}$$

So $y = -16 < 0$, the result of branch prediction is no jump.

B. TRAINING PROCESS

After the branch instruction is executed, the predictor updates the parameters in the PHT table according to the branch result

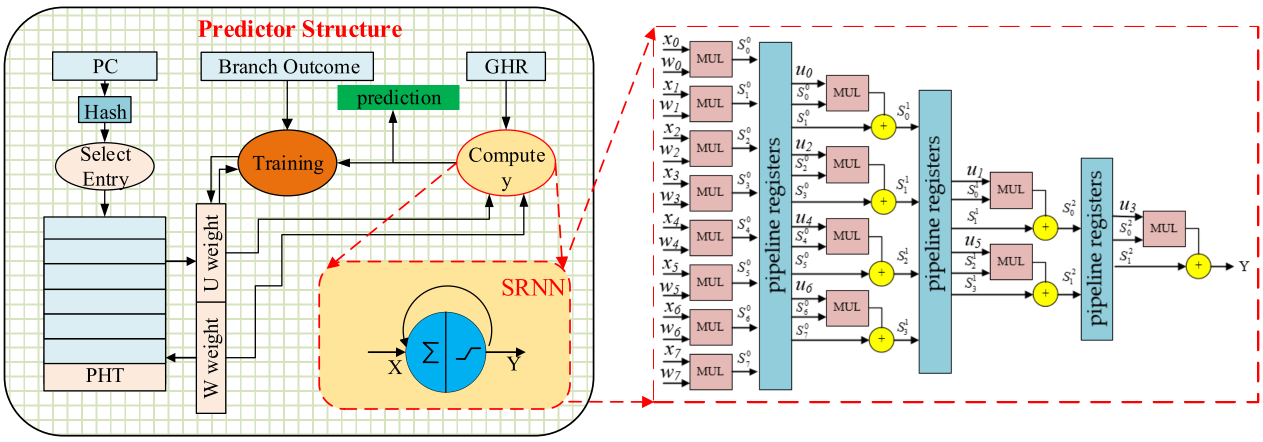


FIGURE 4. Structure of proposed SRNN branch predictor.

of the instruction. The predictor update algorithm needs to be executed, and the update algorithm is as follows:

```
Procedure train($y$:integer; $t$:boolean)
Begin
    if $|y| < \theta$ or $predict \neq t$ then
        for $i = 0$ to $n$ do
            $w_i = w_i + tx_i$
        end for
        for $i = 0$ to $n - 1$ do
            if $x_i \neq t$ then
                $u_i = 1$
            else
                $u_i = u_i + 1$
        end for
    end if
    $GHR = (GHR \ll 1)$ $or$ $t$
end
```

When the prediction result is not equal to the real result or the absolute value of the prediction result is less than the set threshold value, the update algorithm is executed. If x_i is the same as branch result t, its corresponding weight u_i and w_i are increased by 1. If x_i is different from branch result t, its corresponding w_i is decreased by 1, and u_i is set to 1. Then, the GHR register moves left to store the current branch result t. The core idea of the algorithm is to increase the value of the component with a strong correlation with the current branch instruction in the U and W vectors, and decrease the value of the component with a weak correlation with the current branch instruction.

IV. PARAMETER OPTIMIZATION DESIGN

In order to explore the optimal design parameters of predictor, the control variable method is used to study the influence of GHR register length, PHT table size, data representation precision and slice size on the prediction accuracy of SRNN branch predictor. The test data set is SPEC2000 [25], [26].

A. GHR LENGTH

The global branch history is stored in the GHR register. Larger GHR register length can record more history information of branch instructions. In order to study the influence of GHR register length on branch prediction accuracy, the experimental control PHT table size is 512, the data precision is an 8-bit signed integer, and gradually increases the length of the GHR register. The comparison of branch prediction accuracy of different test sets under different lengths is shown in Figure 5.

Figure 5 shows that when the length of the GHR register increases from 4 to 40, the branch prediction accuracy of the branch predictor will increase. When the length increases to 32, the branch prediction accuracy of each test set will reach saturation, and 32 can be selected as the length of the GHR register.

B. PHT TABLE SIZE

PHT table keeps weight parameters corresponding to each branch state. Larger PHT table can record more weight values corresponding to the branch state, which can reduce the impact of duplicate name problem on branch prediction. Similarly, using the control variable method, the length of the experimental control GHR register is 32, and the data precision is an 8-bit signed integer. Gradually increase the size of the PHT table, and obtain the comparison of branch prediction accuracy of different test sets under different sizes of PHT tables, as shown in Figure 6.

Figure 6 shows that when the PHT table size increases from 16 to 8192, the branch prediction accuracy of the branch predictor will increase. When the length increases to 1024, it will reach saturation. 1024 can be selected as the PHT table size.

C. DATA REPRESENTATION PRECISION

Benefiting from the good fault tolerance of neural networks [27], [28], the data format of the predictor is represented by signed integers, not by floating-point numbers. In order to study the effect of signed integers with different digits on the branch prediction accuracy, the control variable method is used to control the size of the PHT table to 512 and the length of the GHR register to 32,

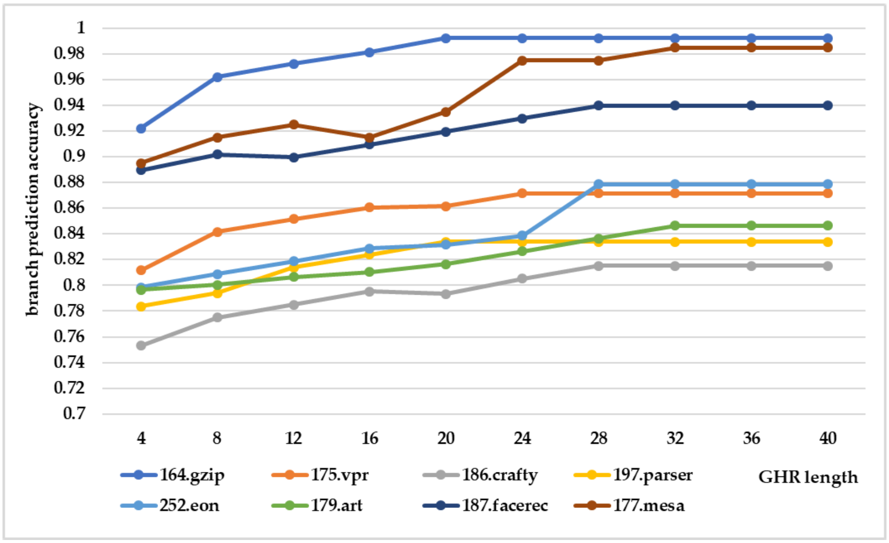


FIGURE 5. Influence of GHR length on prediction accuracy.

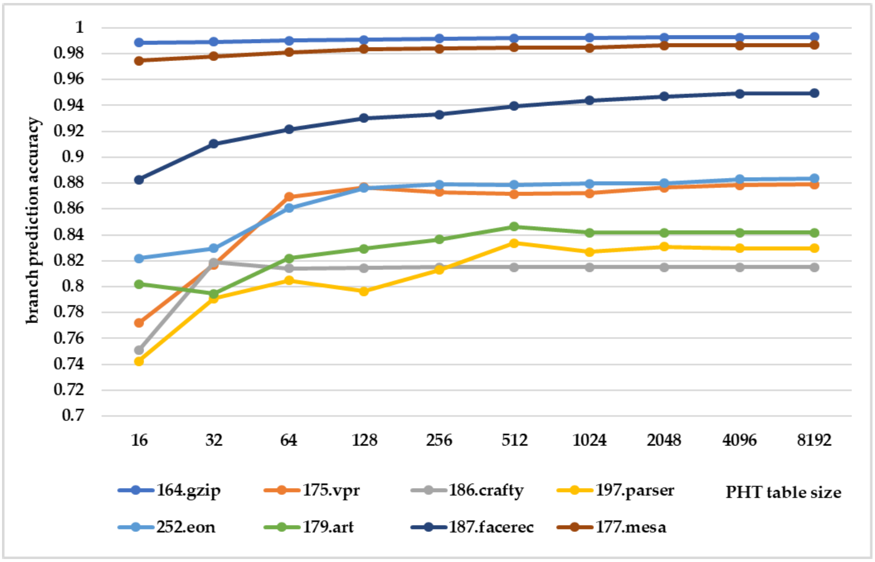


FIGURE 6. Influence of PHT size on prediction accuracy.

gradually increasing the data representation precision of signed integers, and the comparison figure shown in Figure 7 is obtained.

As the data representation precision increases from 3 to 10, the branch prediction accuracy of the branch predictor will increase. When the number of data bits increases to 8, the branch prediction accuracy of each test set reaches saturation, Therefore, 8-bit signed integer numbers can be selected to represent the data.

D. SLICE SIZE

SRNN algorithm needs to slice the input data of the RNN algorithm. If the slice is small, the accuracy of branch. prediction will be reduced, if the slice is large, the accuracy of branch prediction will be increased, but the parallelism of calculation will be reduced, and the calculation delay will be increased. In order to find the balance between prediction accuracy and calculation parallelism, control the size of PHT table to 512, the length of the GHR register to 32, and the data precision to an 8-bit signed integer, and gradually increase the size of slices. The comparison of branch prediction rates of different test sets under different slices is shown in Figure 8.

As can be seen from Figure 8, the larger the slice, the higher the branch prediction accuracy of the branch predictor. For SRNN predictor with GHR register length of 32, when the slice size increases to 6, the branch prediction accuracy of each test set reaches saturation. Therefore, when designing a specific SRNN predictor, it is necessary to explore the optimal slice size under a specific GHR register length, so as to achieve a better balance between prediction accuracy and computational parallelism.

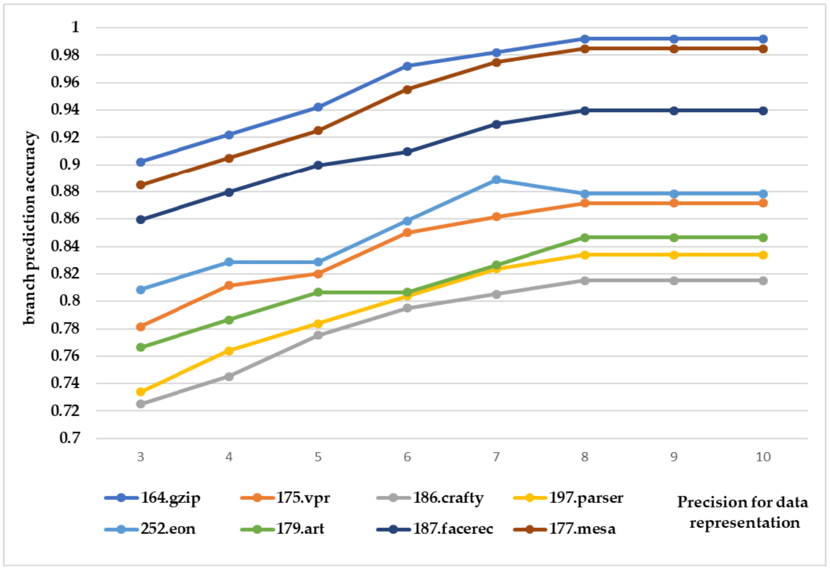


FIGURE 7. Influence of data representation precision on prediction accuracy.

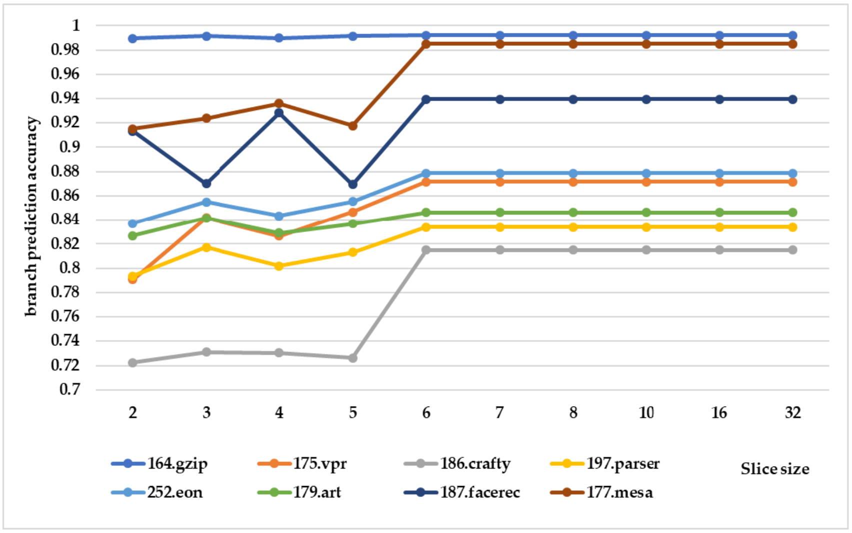


FIGURE 8. Influence of slice size on prediction accuracy.

V. COMPARISONS

In order to explore the advantages of the SRNN predictor, the prediction accuracy of different branch predictors is compared under the same hardware consumption. At the same time, the prediction performance of different predictors in different training time is analyzed. The test data sets are SPEC2000 [25], [26] and Dhrystone [29], [30].

In this paper, Bimod [21], [22], Gshare [23] and Perceptron neural network predictor [5] are selected to compare the branch prediction accuracy with the SRNN predictor proposed in this paper under the same hardware consumption. The PHT table size of all predictors is 512, and the GHR register length of Gshare, Perceptron and SRNN predictors is 32. Because these four branch predictors use PHT tables of the same size, and PHT tables are the main resource consumption of branch predictors, the hardware resources consumed by these four branch predictors are almost the same. The four branch predictors perform the same test set to get the branch direction prediction accuracy comparison chart as shown in Figure 9.

The prediction accuracy of Gshare is better than Bimod, but lower than the Perceptron predictor and SRNN predictor. In the test program with long execution time, the prediction effect of Perceptron predictor and SRNN predictor is equivalent, but in the test program with short execution time, the SRNN predictor proposed is better than that of Perceptron predictor.

Experiments in Figure 9 show that the execution time of the program will affect the prediction effect of the branch predictor. Different predictors have different prediction

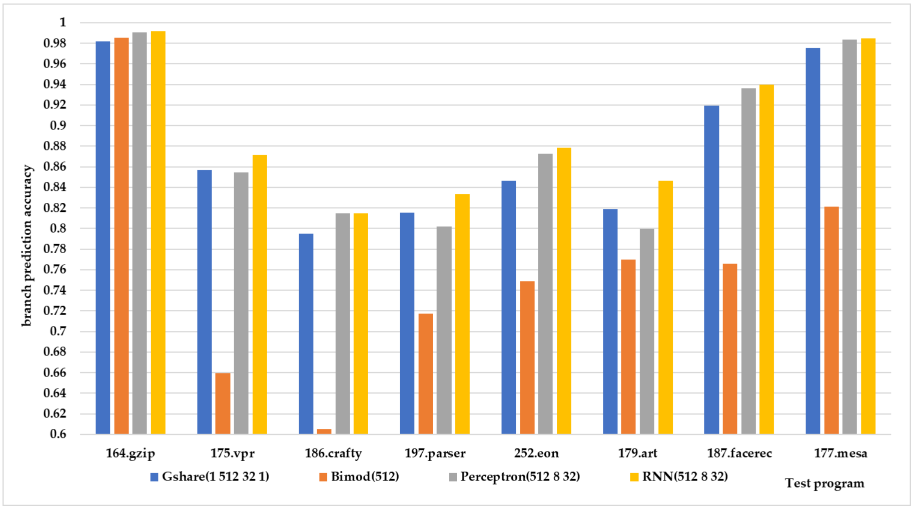


FIGURE 9. Comparison of different branch predictors.

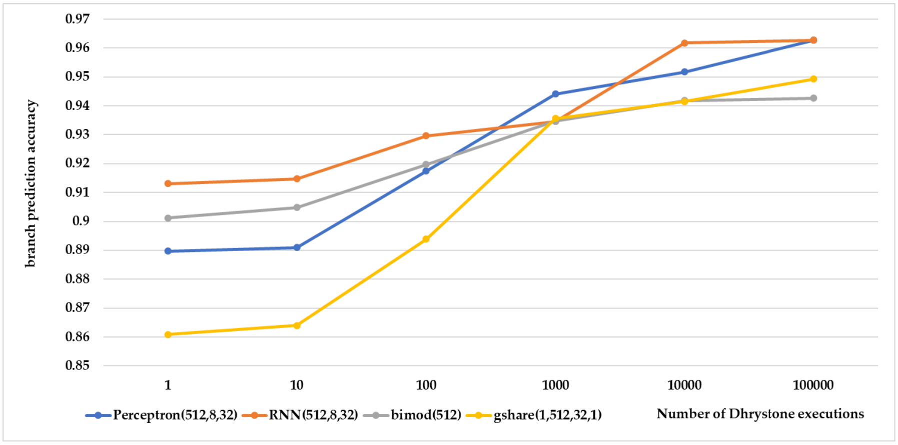


FIGURE 10. Change of prediction accuracy with a different training time of different branch predictors.

performances under different execution times. In order to study this problem, we use the Dhrystone processor test program to compare the prediction effect of different branch predictors under different training time. The training duration is achieved by controlling the number of cycles of the Dhrystone program. Figure 10 shows the branch prediction performance of different branch predictors under different Dhrystone execution times.

As can be seen from Figure 10, when the number of Dhrytone cycles is between 1-10, the execution time of the program is short, and the branch predictors have not been fully trained. At this time, the SRNN predictor is significantly better than other predictors, and its prediction accuracy is 2.34% higher than the traditional Perceptron neural network predictor. With the increase in the number of Dhrystone cycles, the branch prediction rate of each branch predictor gradually increases. When the cycle is executed 100000 times, the prediction rate reaches saturation. At this time, the prediction rate of Perceptron and SRNN predictors is equal, and better than that of Gshare and Bimod.

VI. CONCLUSION

In this paper, a branch predictor based on the parallel structure of SRNN is proposed, which has a higher prediction accuracy in the ''learning period'' and parallel RNN computing structure. Using SimpleScalar, the optimal parameters of GHR register length, PHT table size, data representation precision and slice size are studied, and the prediction accuracy of different predictors under different training time is compared. The experimental results show that the branch predictor proposed in this paper has higher prediction accuracy than the traditional Bimod and Gshare branch predictors under the same hardware consumption, and its branch prediction rate is 2.34% higher than the traditional Perceptron neural predictor in the short learning period.

REFERENCES

[1] S. Mittal, ''A survey of techniques for dynamic branch prediction,'' *Concurrency Comput., Pract. Exper.*, vol. 31, no. 1, Jan. 2019, Art. no. e4666.

[2] J. E. Smith, ''A study of branch prediction strategies,'' in *Proc. 25 Years Int. Symp. Comput. Archit. (ISCA)*, 1998, pp. 202–215.

[3] T. O. Ayodele, *Types of Machine Learning Algorithms*. Portsmouth, U.K.: InTech, 2010. [Online]. Available: https://www.researchgate.net/publication/221907660_Types_of_Machine_Learning_Algorithms

[4] A. Dey, "Machine learning algorithms: A review," *Int. J. Comput. Sci. Inf. Technol.*, vol. 7, no. 3, pp. 1174–1179, 2016.

[5] D. A. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proc. HPCA 7th Int. Symp. High-Perform. Comput. Archit.*, 2001, pp. 197–206.

[6] D. A. Jimenez, "Fast path-based neural branch prediction," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2003, pp. 243–252.

[7] D. A. Jiménez, "Piecewise linear branch prediction," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, Jun. 2005, pp. 382–393.

[8] D. A. Jiménez and C. Lin, "Neural methods for dynamic branch prediction," *ACM Trans. Comput. Syst. (TOCS)*, vol. 20, no. 4, pp. 369–397, Nov. 2002.

[9] Y. Mao, J. Shen, and X. Gui, "A study on deep belief net for branch prediction," *IEEE Access*, vol. 6, pp. 10779–10786, 2018.

[10] I. Hida, M. Ikebe, T. Asai, and M. Motomura, "A 2-clock-cycle Naïve Bayes classifier for dynamic branch prediction in pipelined RISC microprocessors," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Oct. 2016, pp. 297–300.

[11] S. J Tarsa, C.-K. Lin, G. Keskin, G. Chinya, and H. Wang, "Improving branch prediction by modeling global history with convolutional neural networks," 2019, *arXiv:1906.09889*. [Online]. Available: http://arxiv.org/abs/1906.09889

[12] L. Yang-Xu-Rui, C. Shu-Ming, and L. I. Yong, "Low-cost composite neural network branch predictor," *Comput. Eng.*, vol. 37, no. s1, pp. 174–178, Dec. 2011.

[13] P. Trivedi and S. Shah, "Reduced-hardware hybrid branch predictor design, simulation & analysis," in *Proc. 7th Int. Conf. Smart Comput. Commun. (ICSCC)*, Jun. 2019, pp. 1–6.

[14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, *arXiv:1409.2329*. [Online]. Available: http://arxiv.org/abs/1409.2329

[15] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1–24.

[16] L. Jain, *Recurrent Neural Networks*. Boca Raton, FL, USA: CRC Press, 2000. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-1-4615-1613-2_4

[17] M. Evers, S. J. Patel, R. S. Chappell, and Y. N. Patt, "An analysis of correlation and predictability: What makes two-level branch predictors work," in *Proc. 25th Annu. Int. Symp. Comput. Archit.*, 1998, pp. 52–61.

[18] T.-Y. Yeh and Y. N. Patt, "Two-level adaptive training branch prediction," in *Proc. 24th Annu. Int. Symp. Microarchitecture (MICRO)*, 1991, pp. 51–61.

[19] D. Burger and T. M. Austin, "The SimpleScalar tool set, version 2.0," *ACM SIGARCH Comput. Archit. News*, vol. 25, no. 3, pp. 13–25, 2002.

[20] T. Austin, E. Larson, and D. Ernst, "SimpleScalar: An infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[21] C. C. Lee, I. C. K. Chen, and T. N. Mudge, "The bi-mode branch predictor," in *Proc. MICRO*, vol. 30, 1998, p. 4.

[22] S. McFarling, "Combining branch predictors," Digit. Western Res. Lab., Palo Alto, CA, USA, Tech. Rep. TN-36, 1993.

[23] I. Kim, J. Jun, Y. Na, and S. W. Kim, "Design of a G-Share branch predictor for EISC processor," *IEIE Trans. Smart Process. Comput.*, vol. 4, no. 5, pp. 366–370, 2015.

[24] B. Karlik and A. Vehbi, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *Int. J. Artif. Intell. Expert Syst.*, vol. 1, no. 4, pp. 111–122, 2011.

[25] J. L. Henning, "SPEC CPU2000: Measuring CPU performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, Jul. 2000.

[26] A. KleinOsowski, J. Flynn, N. Meares, and D. J. Lilja, "Adapting the SPEC 2000 benchmark suite for simulation-based computer architecture research," in *Workload Characterization of Emerging Computer Applications*. Boston, MA, USA: Springer, 2001, pp. 83–100.

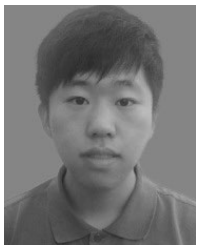
[27] T. Na, J. H. Ko, J. Kung, and S. Mukhopadhyay, "On-chip training of recurrent neural networks with limited numerical precision," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3716–3723.

[28] J. Ott, Z. Lin, Y. Zhang, S.-C. Liu, and Y. Bengio, "Recurrent neural networks with limited numerical precision," 2016, *arXiv:1608.06902*. [Online]. Available: http://arxiv.org/abs/1608.06902

[29] A. R. Weiss, "Dhrystone benchmark, history, analysis, scores and recommendations," ECL, LLC, Austin, TX, USA, White Paper, Oct. 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.392.7607

[30] R. P. Weicker, "Dhrystone: A synthetic systems programming benchmark," *Commun. ACM*, vol. 27, no. 10, pp. 1013–1030, Oct. 1984.

[31] Z. Yu and G. Liu, "Sliced recurrent neural networks," 2018, *arXiv:1807.02291*. [Online]. Available: http://arxiv.org/abs/1807.02291



LEI ZHANG was born in Hubei, China, in 1995. He received the B.E. degree in electronic information engineering from Yangzhou University, Jiangsu, China, in 2017. He is currently pursuing the master's degree with the Nanjing University of Aeronautics and Astronautics. His current research interests include artificial intelligence, RISC-V, and ASIC design.

NING WU was born in Anhui, China, in 1956. She received the B.S. and M.S. degrees from the University of Science and Technology of China, in 1982 and 1985, respectively. She is currently a Professor and a Ph.D. Supervisor with the Department of Electronic Engineering, Nanjing University of Aeronautics and Astronautics. Her research interests include digital system theory and technology, electronic system integration, and ASIC design.

FEN GE was born in Jiangsu, China, in 1981. She received the B.E., M.E., and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, in 2003, 2006, and 2010, respectively. She is currently an Associate Professor with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. Her research interests include many-core systems and network-on-chip design.

FANG ZHOU was born in Jiangsu, China, in 1979. She received the B.S., M.S., and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, in 2001, 2004, and 2015, respectively. She is currently a Lecturer with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. Her research interests include ASIC design and energy-efficient digital VLSI design.

MAHAMMAD REHAN YAHYA was born in 1983. He received the B.S. degree from the COMSATS Institute of Information Technology, Islamabad, in 2004, and the M.S. degree from UET Taxila, Pakistan, in 2006. He is currently pursuing the Ph.D. degree with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, China. His research interests include optical and hybrid network on chip architectures and FPGA based implementations.