ELSEVIER

Microprocessors and Microsystems 23 (1999) 527–536

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.nl/locate/micpro

Path-based branch prediction using signature analysis

S. Lee*, I.-C. Park, C.-M. Kyung

Department of Electrical Engineering, KAIST, 305-701 Taejon, South Korea

Received 30 October 1998; received in revised form 9 July 1999; accepted 5 August 1999

Abstract

In high-performance processors, the accuracy of branch prediction plays a significant role in enhancing computer execution. A new hardware approach is presented in this paper to dynamically predict branch directions using path information. As an execution path contains large information, we compress the large information using a technique based on the linear feedback shift register (LFSR) that is widely used in testing and error correction coding. A modified version of LFSR, called windowed LFSR, is developed to calculate the signature of a path in one cycle. Since the windowed LFSR has a very regular structure, it can be easily implemented. For most of the benchmarks, the proposed prediction scheme shows better prediction accuracy than the pattern-based predictions.

Keywords: Branch prediction; Linear feedback shift register; Microprocessor design; Computer execution

1. Introduction

In modern times, processors that issue multiple instructions and have deep pipelining, reducing branch penalty are crucial in exploiting maximal performance of processors. As shown in Fig. 1, the pipeline has to stall for a number of cycles if a branch instruction is encountered because the branch is usually resolved at the later pipeline stage. To reduce the penalty, branch prediction is actively employed in high-performance processors. As the accuracy of branch prediction has a great influence on the overall performance, there have been a variety of efforts taken towards reducing the misprediction rate even at the expense of additional chip area.

Starting from a simple prediction scheme in which the direction of a branch is determined by the previous history of the branch [1], prediction is advanced to correlation schemes to enhance the prediction accuracy. The correlating branch prediction [8,10,11] is based on the fact that the direction of a branch is closely related to the associated path, i.e. the sequence of basic blocks executed just before the branch instruction. The correlation schemes can be classified into two categories. The first category called pattern-based correlation uses the directions of preceding branches to represent the associated path and the second called path-based correlation directly uses the path information.

Although pattern-based correlation schemes are prevalent due to their prediction accuracy and relatively simple hardware, they often make wrong decisions that could be correctly predicted if path information were provided [9,12]. A path-based correlation scheme was proposed [9], but failed to improve the prediction accuracy due to the inefficiency in the way of representing the path information.

In this paper, we analyse the factors responsible for the low prediction accuracy in the previous path-based scheme and propose a new approach to deal with the problems. To cope with the large information in a path, a compressing technique is employed in our scheme. The compression is based on the linear feedback shift register (LFSR) which is widely used in testing and error correction coding. An extended LFSR is proposed to compress the path information at the rate of one cycle. In most of the benchmarks, the prediction accuracy of the proposed scheme is equal to or better than that of the pattern-based predictions.

The remainder of this paper is organized as follows. Section 2 reviews previous work on the correlation-based scheme, and Section 3 introduces LFSR and the modification. Section 4 presents a new path-based prediction scheme and its properties. We discuss experimental results in Section 5 with focusing on the effect of parameters on performance. Section 6 summarizes our study and concludes with describing future work.

2. Related works

To enhance the prediction accuracy, Pan et. al. proposed a

* Corresponding author.
E-mail address: sjlee@vslab.kaist.ac.kr (S. Lee)

0141-9331/99/$ - see front matter
PII: S0141-9331(99)00056-3

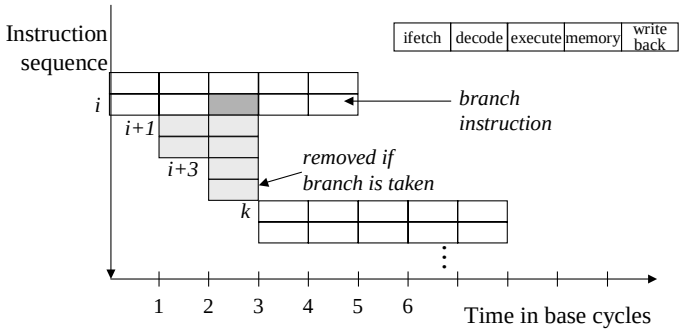


Fig. 1. In a 2-issue superscalar processor, the pipeline stalls until the target address of a branch instruction is resolved at the execution stage. The branch penalty is 8 instructions in this case.

correlation-based branch prediction scheme which is dynamically evaluated in the hardware [10]. Besides the branch history register that stores the directions of a set of preceding branch instructions, the address of a branch instruction is also used to select a predictor in the correlation-based prediction scheme, as shown in Fig. 2. The prediction table is accessed by the lower *m*-bit part of the branch address and the recent *n*-bit history of previous branches. Each entry of the prediction table contains a 2-bit up/down counter to remember previous directions of the corresponding branch. In general, this pattern-based correlation scheme, named *gselect* [8] or *GAs* [11], outperforms the simple schemes that consider only the branch address.

Though the pattern history register is relatively simple, it is difficult to represent the path information correctly. Due to the following two observations, the path-based scheme can give more accurate prediction than the pattern-based schemes:

1. Two different paths can yield the same pattern. Such a case is called *pattern aliasing* [12],and occurs frequently in call-return boundaries [9].
2. In compilers, simple *switch* statements are usually optimized by using indirect jump instructions whose target addresses are normally stored in a table. This makes it very difficult to determine the next basic block by using only the direction information.

As the second observation occurs more often in C++ programs that usually contain more functions and indirect jumps than C programs [3], the advantage of the path-based prediction scheme becomes more prominent with the wider acceptance of object-oriented programming.

To directly utilise the path information rather than the pattern history, two methods were proposed. First, the static correlating prediction scheme proposed by Young [12] determines the direction of a branch in compile time by considering the path to the branch. Second, proposed by Nair, a dynamic path-based scheme implemented in hardware [9]. Its organization is similar to that of the pattern-based scheme except that, in place of the pattern history, lower *k* bits of target addresses of preceding branches are concatenated as shown in Fig. 3. Thus, only a small portion of target addresses is collected as path information.

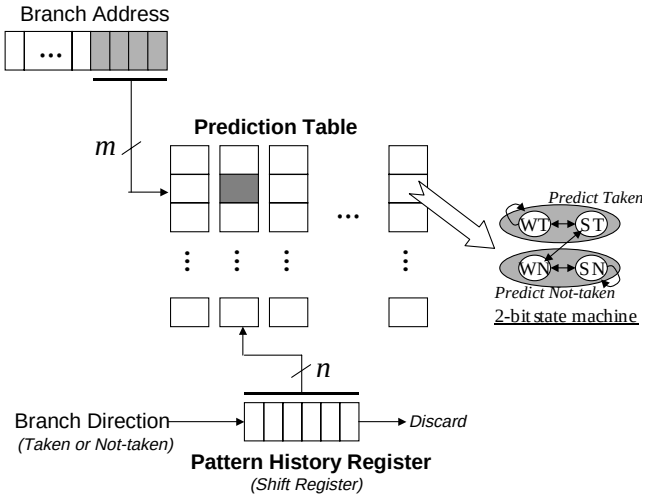


Fig. 2. Pattern-based correlation scheme. The prediction table is indexed with the branch address and/or the directions of the preceding branches.

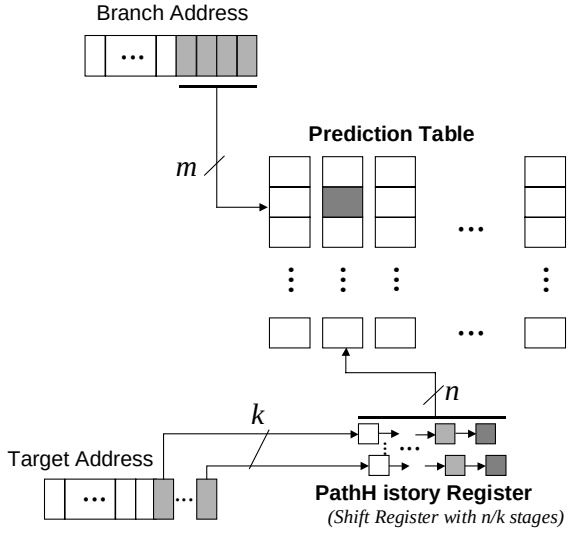


Fig. 3. Path-based correlation scheme proposed by Nair [9]. The prediction table is indexed with the branch address and/or the part of target address of the preceding branches.

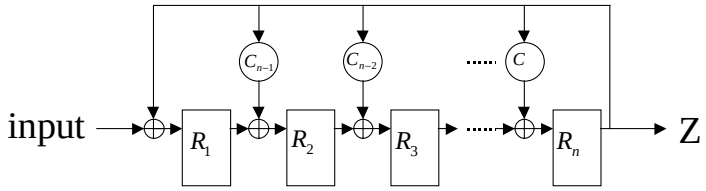


Fig. 4. Single-input LFSR where C_i represents the coefficient of a characteristic polynomial and R_i holds the signature.

When the number of bits extracted from a branch is not enough to distinguish it from others, prediction accuracy can be decreased due to the interference among different paths sharing the same entry of the prediction table. On the other hand, if the number of bits is too large, many entries of the prediction table can be allocated for cases that never occur in practice.

3. Signature analysis

Signature analysis is a compression technique based on the concept of cyclic redundancy checking and it is usually realized in hardware using the linear feedback shift register (LFSR).

3.1. Linear feedback shift register

The LFSR [1], shown in Fig. 4, is a shift register in which the shifted output is fed back to the register inputs through XOR gates. Characteristic polynomials denoted as $\{C_1–C_n\}$ determine the feedback path, where C_i takes either 0 or 1. The feedback value that comes from Z is applied to every stage in which C_i is 1. This means the feedback path is parameterized by C_i. A characteristic polynomial that can generate $2^n - 1$ periodic states is called a primitive polynomial. The primitive polynomial is commonly used to generate pseudo-random numbers as it can generate all patterns except the all zero pattern.

For a large number of test patterns, checking the results of a circuit requires a large sized ROM to store all expected result values to be compared with actual results. Instead of checking the result for each test pattern, we can compress the large number of results into one value called a signature,

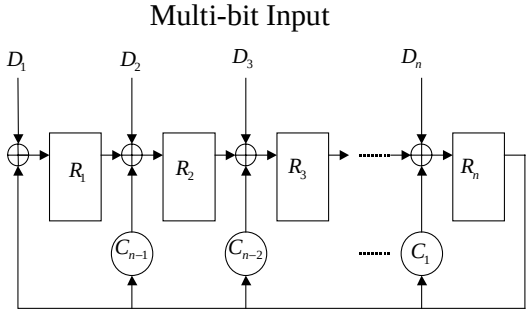


Fig. 5. Multiple-input LFSR. Each bit of the n-bit input is connected to a 3-input XOR gate.

and compare it with one reference value calculated by asserting the test patterns to a simulator. If the compression yields a quite different signature even for slightly dissimilar test patterns, we can assume that the final signature stands for the whole results.

The following properties of the LFSR are worthwhile to note;

1. The LFSR distributes all possible input bit-streams quite evenly over all possible signatures.
2. The probability, *P(n)*, that an n-bit signature generator results in the same value for two different input patterns approaches 2^{-n}.

The properties are obtained by assuming that all possible patterns are equally distributed. Although the assumption does not hold good; and *P(n)* is dependent on the characteristic polynomial, experimental results show that the primitive polynomials help reduce the probability, *P(n)*.

The signature analysis can be extended for multiple output circuits. A multi-bit input LFSR that can be used for this purpose is shown in Fig. 5, where multiple inputs, $\{D_1–D_n\}$, are simultaneously fed to the LFSR.

3.2. Windowed LFSR

The LFSR generates one signature for all inputs asserted. If such an LFSR is used to compress branch addresses in a path, it produces the signature using all the preceding addresses, thus resulting in a different signature for every branch met in run time. Since the signature identifies an entry of the prediction table, the same branch encountered in run time may point to a different location.

In general, the path to a branch instruction is defined by a fixed number of branch addresses executed before the branch instruction. Therefore, to utilise the previous history, we need to generate the same signature for the same path and calculate one signature per branch instruction by considering only a fixed number of preceding branch instructions. This is impossible in the standard LFSR that is developed for compressing all inputs asserted.

In this paper, we propose a new LFSR called windowed LFSR. As shown in Fig. 6, the windowed LFSR consists of several LFSR stages that operate in a pipelined manner. The first stage compresses only one address, the second stage compresses two addresses, and finally the lth stage can compress l addresses. Since all the stages work in a

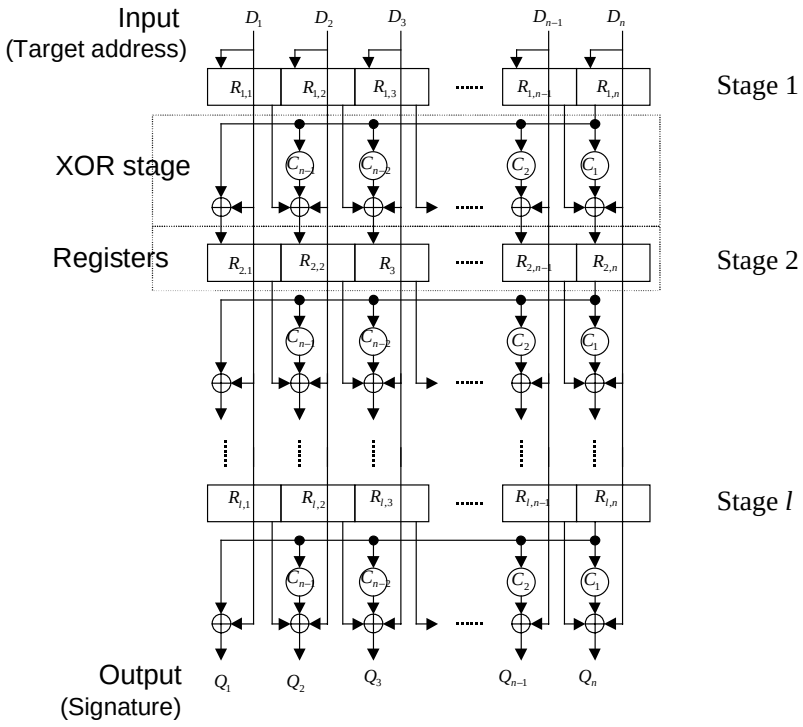


Fig. 6. Windowed LFSR. The output is an n-bit signature made from l inputs where l denotes the window size.

pipelined fashion, we can generate the signature of a path consisting of l addresses at the rate of one cycle.

4. Branch prediction using path-based correlation

4.1. Path-based correlation

The target address of a branch can be regarded as the start address of the following basic block. Thus, a sequence of target addresses is an adequate means to represent a path.

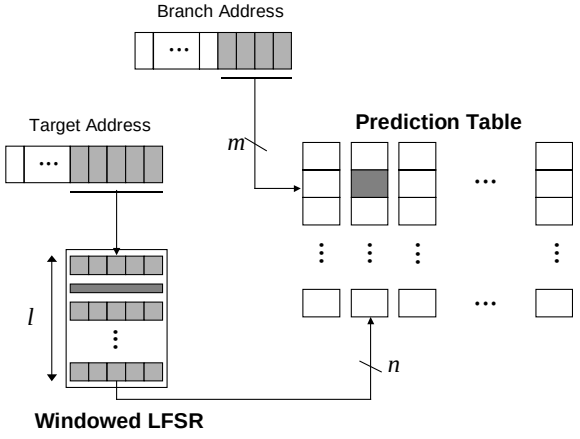


Fig. 7. Basic structure of the proposed path-based scheme. The prediction table is accessed with the lower part of a branch address and an n-bit signature generated from the windowed LFSR. The signature is calculated by using l target addresses of preceding branches.

Although target addresses consist of many bits, a relatively small number of bits are sufficient to index an entry of the prediction table. Therefore, a good hashing function is essential for path-based predictions.

The basic structure of our path-based scheme is shown in Fig. 7. The proposed windowed LFSR makes a signature for each fixed-length path by using target addresses of preceding control instructions.

Three parameters are defined in Fig. 7 to model the proposed path-based scheme.

- m: the number of bits in the branch addresses for indexing the prediction table
- n: the number of bits in the signature to be used for indexing the prediction table
- l: (correlation length) the number of stages in the windowed LFSR

We will use the (m, n, l) notation to represent the hardware configuration shown in Fig. 7. In contrast to the pattern-based scheme in which n and l have the same value, they are separated in our scheme. This separation makes it possible to adjust correlation without changing the prediction table size and to control the prediction rates.

In the previous path-based scheme two parameters, number of bits and bit locations, are statically determined [9]. As only a few bits are considered in the target address of a preceding branch [7], there is still much information loss. On the other hand, in windowed LFSR, large parts of target addresses are considered and there is no limitation in

Table 1
Statistics of benchmark programs

CINT95 benchmarks	Input	Number of dynamic		
		Branches	Indirect jumps	Unconditional branches
GO	train/2stone9 *	27,929,154	2,830,531	5,207,465
M88KSIM	train/ctl.in	18,751,361	3,192,401	4,558,039
GCC	train/jump.i	27,285,351	3,451,413	4,198,245
COMPRESS	ref/bigtest.in *	14,012,016	3,461,676	4,972,375
XLISP	train/test.in	24,787,835	7,522,965	10,515,395
IJPEG	train/vigo.ppm *	15,627,942	2,149,259	2,692,476
PERL	ref/scrabbl *	21,388,367	5,151,467	3,590,305
VORTEX	train/vortex.in *	19,461,436	2,111,798	2,700,339

increasing the correlation length. Therefore, our path-based scheme considers more information than the previous scheme in selecting an entry of the prediction table.

Now, let us consider the area overhead of windowed LFSR. If we count only storage elements and assume that the 2-bit up/down counter is used in the prediction table, the ratio of the area of windowed LFSR to that of the prediction table can be represented as follows.

$$\frac{\text{size(windowed LFSR)}}{\text{size(prediction table)}} = \frac{n \times l}{2^{m+n+1}}$$

If we assume $(m, n, l) = (2, 14, 10)$, for example, the area overhead is only 1%. Moreover, it becomes smaller as the prediction table size increases.

4.2. Experimental environment

We evaluated our branch scheme with a trace-driven simulation technique. The benchmarks consisting of eight SPEC CINT95 programs are tested on an UltraSPARC workstation using 'Shade' simulator [4]. The statistics of the benchmarks are summarised in Table 1.

Since some of the SPEC benchmark programs that are marked with * in Table 1 take too long a simulation time to complete, we need to reduce the size of the trace. As applied in numerous literature [6], some parameters are modified to balance the number of branches while preserving the properties of the benchmarks.

Primitive polynomials [2] are employed as characteristic polynomials of the windowed LFSR. The 2-bit up/down counter is adopted for branch prediction, and we take target addresses of conditional branches as inputs to the windowed LFSR. Target addresses of indirect jumps and unconditional branches are excluded to compare fairly with the pattern-based schemes.

4.3. Characterization

Fig. 8 shows prediction rates with regard to the change of two parameters, n and l, where the optimal correlation length for a given prediction table is denoted by a circle. The optimal correlation length increases as the prediction table size increases.

As we increase the correlation length to examine longer paths, the number of table entries occupied by a branch increases, and prediction accuracy for the branch improves. However, it also results in greater interference between different branches for a fixed-sized prediction table. For

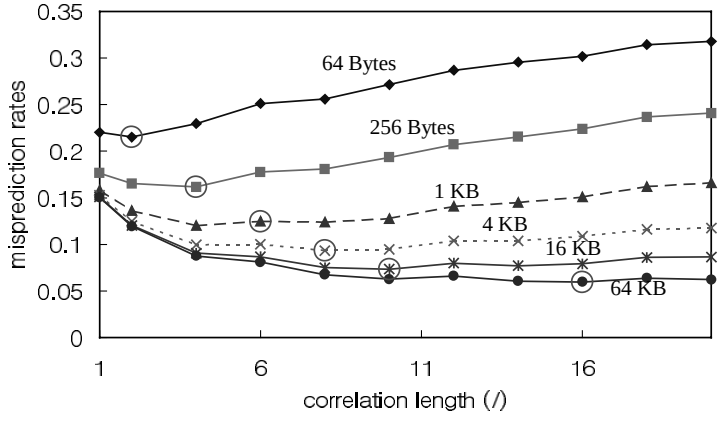


Fig. 8. Misprediction rates for GCC according to various prediction table sizes and correlation lengths. The branch address is not used ($m = 0$) in this experiment. The circled point for each prediction table size represents the optimal correlation length, in which the misprediction rate is minimal.

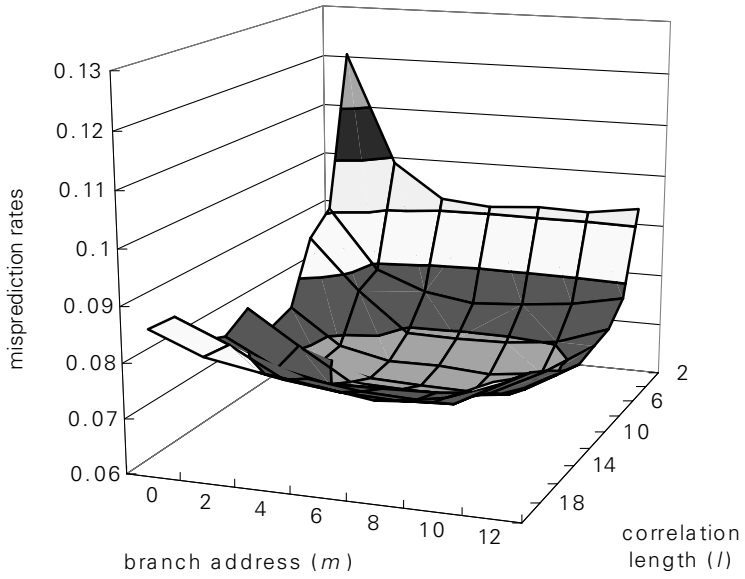


Fig. 9. Misprediction rates for GCC when the number of bits occupied by the branch address varies from 0 to 12. The graph is generated for a fixed-sized prediction table of 16 KB. This figure implies that the misprediction ratio is dependent on the number of bits in the branch address and the correlation length.

this reason, a long correlation length is inappropriate for a small-sized prediction table.

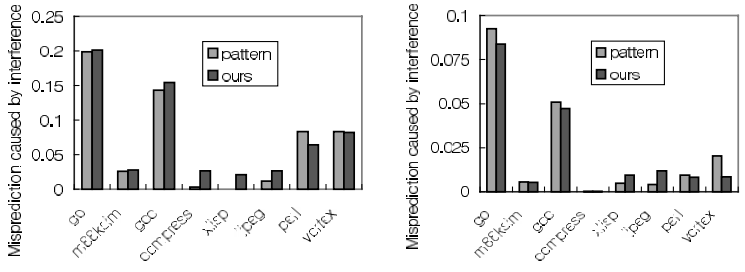
In Fig. 9, the change of prediction rate according to the variation of m and l is shown for a 16 KB prediction table. A sequence of target addresses is a perfect representation of a path in itself, but concatenating the branch address can improve the prediction rate. By including the branch address in indexing the prediction table, the interference is limited to the paths having the same branch. Until 6 bits of the branch address are used ($m \leq 6$), prediction accuracy increases gradually.

However, as more bits of branch addresses are considered, there are smaller bits remaining to represent a path. Thus, intensive interference between paths related to the same branch decreases the prediction rate. This is shown in Fig. 9, where the prediction rate degrades as m grows to greater than 8.

4.4. Interference between signatures

Contrary to the pattern-based scheme, in which the prediction rate is greatly affected by interference between different paths that have the same pattern, our path-based scheme does not have such intrinsic interference. However, there is another interference that is induced by paths having the same signature. Two graphs in Fig. 10 show the effects of interference introduced in the pattern-based scheme and the proposed path-based scheme, when the branch address is not considered ($m = 0$). For different values of m, the experiment shows similar results.

As shown in Fig. 10(a), the amount of interference in our scheme is slightly larger than that of the pattern-based one for a 1 KB prediction table. However, as the prediction table size increases, the interference becomes smaller than that of the pattern-based one as shown in Fig. 10(b). This fact



(a) 256 byte prediction table (m,n,l) =(0,10,12)

(b) 16K byte prediction table (m,n,l) =(0,16,20)

Fig. 10. Interference effects of pattern-based and the proposed path-based schemes. As the prediction table size increases, the proposed scheme reduces interference more rapidly than the pattern-based scheme.

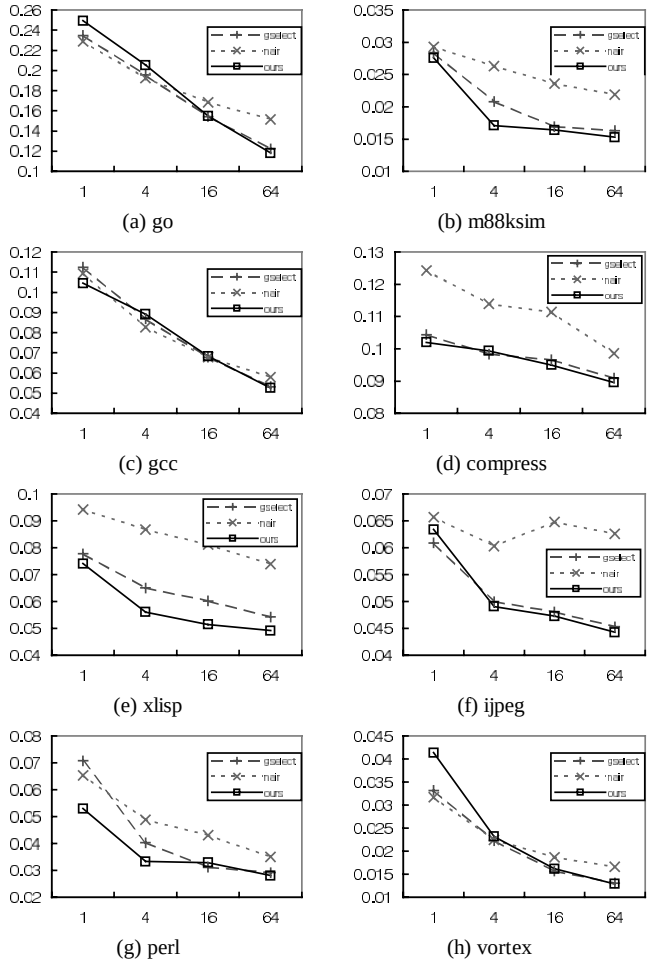


Fig. 11. Comparisons of misprediction rate between *gselect* and Nair's scheme (the X-axis denotes the prediction table size in KB). The same correlation is assumed for all schemes; 6 bits from the branch address and the remaining bits from the branch history.

implies that our scheme is more effective in reducing interference as the prediction table size increases. It is known that small interference usually leads to better prediction [12].

4.5. Comparison with other prediction schemes

Our scheme is compared to other prediction algorithms—*gselect*, and Nair's—in Fig. 11. For all schemes, 6 bits are assigned to the branch address and the remaining bits to correlation regardless of the prediction table size. In Nair's method, 2 bits of a target address are used, so its correlation length is a half of the others.

If the prediction table size is small, our scheme shows some improvement in several benchmarks, especially for XLISP and PERL, but not for the others. As the prediction table size grows, our scheme catches up and overcomes the others for all benchmarks. It becomes evident especially when the prediction table size exceeds 16 KB, which can be implemented reasonably with the current technology. A reason for this tendency is that the interference of our path-based scheme is less than that of pattern-based schemes for the large-sized prediction table, as mentioned in Section 4.4. The reduction of interference results in better prediction accuracy.

We compared our scheme with another widely used prediction scheme, *gshare*, where the branch address and the pattern are mixed by XORing [8]. In our scheme, the branch address can be fed into the windowed LFSR and mixed with the signature, as *gshare* does. In our scheme, since the correlation length is not dependent on the number of bits assigned to represent path information, a large number of bits in the signature can be involved in the XORing. It leads to reducing the aliasing between signatures and results in superior prediction accuracy even at the same correlation length, as shown in Fig. 12.

Our scheme outperforms *gshare* for all benchmark programs, especially for GO benchmark—known to be hard for achieving good prediction accuracy. Our misprediction rate falls more rapidly compared to *gshare* as the prediction table size grows.

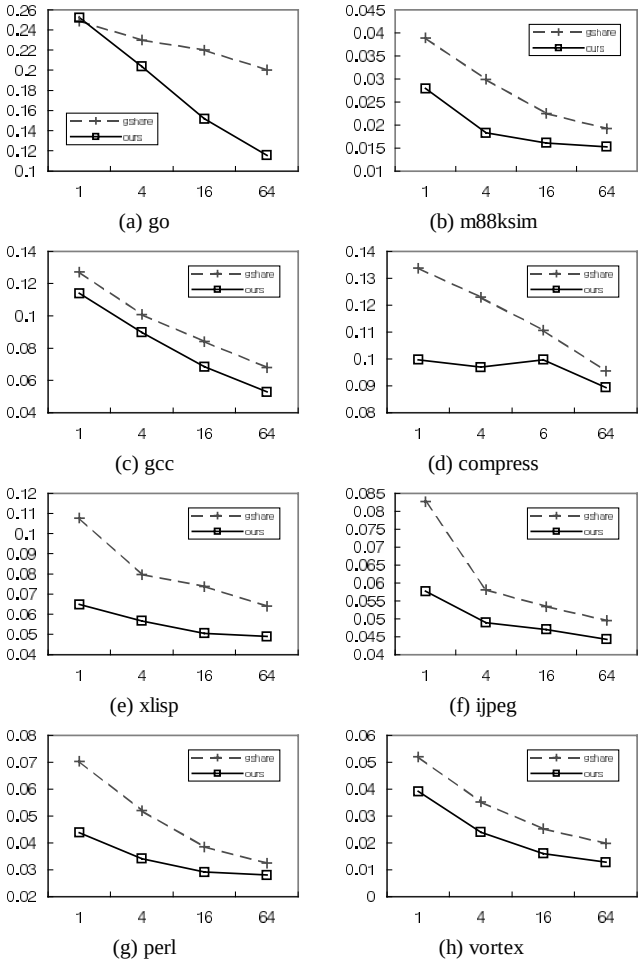


Fig. 12. Comparisons of misprediction rate betwen *gshare* and our path-based prediction scheme (the correlation: *X*-axis denotes the prediction table size in KB). Each scheme is experimented with the same parameters.

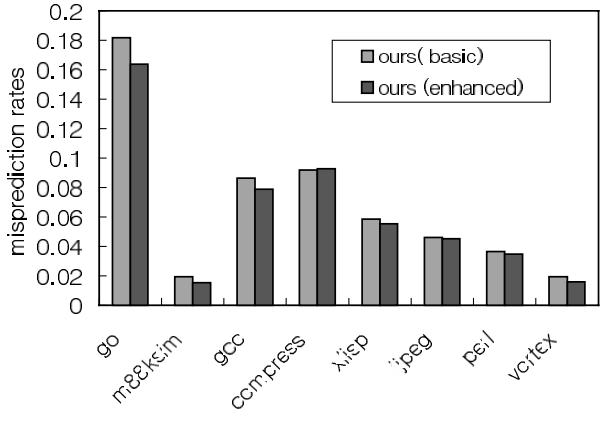


Fig. 13. The performance of the enhanced scheme for 16 KB prediction table. (0, 16, 20) is applied for the *basic* scheme and (0, 16, 12) for the *enhanced* scheme.

5. Properties related to a path

5.1. Branch address as an input to windowed LFSR

Since a target address points to the leader of a basic block, it can be used as an input to the windowed LFSR. Considering the pipelining, however, it is hard to determine the target address of a branch at the time of updating windowed LFSR, and thus a special hardware is required to resolve this problem. Calder et al. suggested an architecture where a lower part of target address is stored in the instruction cache [5], but it needs additional logic to detect the case of exceeding boundary.

```
var := 0;
   …
if (cond) then
   var := 1;
…
if (var1) then    {predict}
```

At present, most of the commercial processors use Branch Target Buffers (BTB) [1] to predict target addresses. Pattern-based schemes also have to predict the branch directions in order to update the pattern history register, but the prediction of branch directions is easier and more accurate than that of target addresses.

To alleviate the difficulty of predicting target addresses, we can use the branch addresses as inputs to the windowed LFSR. However, dealing with only branch addresses can miss the fact that the second branch is dependent on the assignment following the first branch.

To cope with the problem, the branch direction must be considered in conjunction with the branch address. We examine three cases in which the target address, the branch address or the branch address XORed with the direction is used as an input to the windowed LFSR. In experimental results, differences between the three cases are small, but those that use either the target address or the branch address with the direction are slightly superior to the one based on only the branch address.

5.2. Reducing correlation length

In contrast to the pattern-based scheme in which branch directions are used as a pattern, in the path-based scheme it is possible to exclude fall-through branches without loss of prediction accuracy. Treating a sequence of fall-through basic blocks as one basic block does not destroy path information at all. If we assume that the probability of taken branches is equal to that of fall-through branches, about half the correlation is enough to ensure the same performance.

A problem with this technique is that it is impossible to determine where the current basic block is. Hence, the prediction table is indexed together with the current branch

address and the signature. Although there is no profit in the number of occupied prediction table entries the correlation length, corresponding to the number of stages in the windowed LFSR, can be reduced by half without sacrificing prediction rate. This scheme is called *enhanced*.

This *enhanced* scheme is compared with the *basic* scheme in Fig. 13, where the correlation length of the enhanced scheme is 12 and that of the basic scheme is 20. The value of correlation length is determined to give the best prediction rate for each scheme.

Prediction rates of the *basic* scheme and the *enhanced* scheme are almost equal to each other, but there are several cases where the *enhanced* scheme outperforms the *basic* scheme even with the reduced area.

6. Conclusion

A new approach to dynamic branch prediction using path-based correlation is presented in this paper. To handle the large information of a path, the proposed scheme is based on a compression technique—signature analysis—that is actively being used in testing and coding. Furthermore, we have presented a novel circuit called windowed LFSR to generate the signature of a path in one cycle. The windowed LFSR has a very regular structure and requires negligible hardware resources.

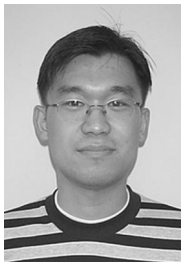
Our path-based scheme separates correlation from the prediction table size, and gives freedom to improve prediction accuracy. Simulation results for a number of benchmarks reveal that the proposed scheme compares favourably to the widely used pattern-based schemes, and to the previous path-based prediction proposed by Nair. In most of the benchmarks, the proposed scheme shows better prediction accuracy than the above schemes.

The target address has a merit in that it can handle indirect jumps. In SPEC benchmark programs, the portion occupied by indirect jumps is small and the benefit obtained by adopting target addresses of indirect jumps is negligible [6]. But in other benchmarks such as C++ programs, we expect that adopting target addresses is significant in improving prediction rates. We also proposed a technique to reduce correlation length and thus save area.

There is more work that needs to be done in the near future. One example is the need to investigate the primitive polynomials in the windowed LFSR. Although it is known that the primitive polynomial gives the best random property in general, there is a question on whether it is optimal for branch prediction because other polynomials may lead to better signatures.

References

[1] J. Hennessy, D.A. Patterson, Computer Architecture: a Quantitative Approach, 2nd ed, Morgan Kaufmann, Los Altos, CA, 1996.
[2] M. Abramovici, M.A. Breuer, A.D. Friedman, Digital Systems Testing and Testable Design, Computer Science Press, Rockville, MD, 1990.
[3] B. Calder, D. Grunwad, B. Zorn, Quantifying behavioral differences between C and C++ programs, Technical Report CU-CS-698-94, University of Colorado, January1994.
[4] R.F. Cmelik, D. Keppel, Shade: a fast instruction set simulator for execution profiling, Technical Report TR-93-12, Sun Micro Systems Lab, 1993.
[5] B. Calder, D. Grunwad, Fast and accurate instruction fetch and branch prediction, in the 21st Annual International Symposium on Computer Architecture, 1994, pp. 2–11.
[6] P.-Y. Chang, E. Hao, Y.N. Patt, Target prediction for indirect jumps, in the 24th Annual International Symposium on Computer Architecture, 1997, pp. 274–283.
[7] Q. Jacobson, S. Bennett, N. Sharma, J.E. Smith, Control flow speculation in multiscalar processors, in: the 3rd International Symposium on High-Performance Computer Architecture, 1997, pp. 218–229.
[8] S. McFarling, Combining branch predictors, WRL Technical Note TN-36, Digital Equipment Corporation, June 1993.
[9] R. Nair, Dynamic path-based branch correlation, in the 28th Annual International Symposium on Microarchitecture, 1995, pp. 15–23.
[10] S.-T. Pan, K. So, J.T. Rahmeh, Improving the accuracy of dynamic branch prediction using branch correlation, in the 5th Proceeding of the International Conference on Architectural Support for Programming Languages and Operating Systems, 1992, pp. 76–84.
[11] T.-Y. Ye, Y.N. Patt, Alternative implementation of two-level adaptive branch prediction, in the 19th Annual International Symposium on Computer, 1992, pp. 124–134.
[12] C. Young, N. Gloy, M.D. Smith, A comparative analysis of schemes for correlated branch prediction, in the 22nd Annual International Symposium on Computer Architecture, 1995.



Seunjeogn Lee was born in Seoul, Korea, in 1972. He received his BS and MS degrees in Electrical Engineering from KAIST in 1993 and 1995, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, KAIST. His research interests include VLSI architecture and system verification.

Chong-Min Kyung received his BS degree in Electronics Engineering from Seoul National University in 1975, and his MS and PhD degrees in Electrical Engineering from KAIST in 1977 and 1981, respectively. From 1981 to 1983, he worked at Bell Telephone Laboratories, Murray Hill, New Jersey as a postdoctoral member of the technical staff in the area of semiconductor device and processors.

In-Cheol Park received his BS degree in Electronics Engineering from Seoul National University in 1986, and his MS and PhD degrees in Electrical Engineering from KAIST in 1988 and 1992, respectively. From 1995 to 1996, he worked at IBM T.J. Watson Research Center, Yorktown, New York as a postdoctoral member of the technical staff in the area of circuit design. In 1996, he joined the faculty of the Department of Electrical Engineering, KAIST where he is now an Assistant Professor. His research interests include CAD algorithm for high-level synthesis and VLSI architectures for general-purpose microprocessors.