# Hybrid branch prediction for pipelined MIPS processor

**Ali S. Al-Khalid, Safaa S. Omran**
Middle Technical University, Electrical Engineering Technical College, Iraq

**Article Info**

***Article history:***

Received May 10, 2019
Revised Nov 15, 2019
Accepted Jan 11, 2020

***Keywords:***

Dynamic branch prediction
Hybrid branch predictor
Static branch prediction
The hazards

**ABSTRACT**

In the modern microprocessors that designed with pipeline stages, the performance of these types of processors will be affected when executing branch instructions, because in this case there will be stalls in the pipeline. In turn this causes in reducing the Cycle Per Instruction (CPI) of the processor. In the case of executing a branch instruction, the processor needs an extra clocks to know if that branch will happen (Taken) or not (Not Taken) and also it requires calculating the new address in the case of the branch is Taken. The prediction that the branch is T / NT is an important stage in enhancing the processor performance. In this research more than one method of branch prediction (hybrid) is used and the designed circuit will choose different types of prediction algoritms depending on the type of the branch. Some of these methods were used are static while the other are dynamic. All circuits were built practically and examined by applying different programs on the designed predictor algorithm to compute the performance of the processor.

***Corresponding Author:***

Safaa S. Omran,
Middle Technical University,
Electrical Engineering Technical College, Iraq.
Email: Omran_safaa@ymail.com

## 1. INTRODUCTION

Branch predictors now considered one of the basic units in the modern microprocessors that use pipeline stages in their design. This unit (BP) makes a prediction for the branch instructions that if the branch will be Taken or Not Taken. Previously when processors were designed without branch prediction unit, the processor requires more clock cycles by making a delay in the pipeline stages in each coming of branch instruction in order to know if that branch is Taken or Not Taken and also to calculate the target address in the case of Taken [1, 2].

In general 20% out of the instructions in a program is branch instructions; this means that is in every 5 instructions there is one branch instruction [3]. Hence, predicting the behaviour of the branch (which is Taken or Not Taken) is very important and affects the performance of the processor. The penalty associated with mispredicted branches in modern pipelined processors has a great effect on performance. The performance penalty is increased as the pipelines deepen and the number of mispredicted instructions increases. For example, the AMD Athlon processor has 10 stages in the integer pipeline [4], while the Intel NetBurst microarchitecture used in the Pentium 4 processor is hyper-pipelined with a 20-stage branch prediction penalty [5]. The rest of this work will be as follows; in the next section a theoretical review for some types of static and dynamic branch prediction methods. Then a section will presents the designed branch prediction circuit and the following sections will presents results and conclusions respectively.

## 2. THEORETICAL BACKROUN

There are two kinds of branch prediction, one called static while the other is dynamic, and here a brief explanation for some types of branch prediction methods.

### 2.1. Static branch prediction

If the technique of the used branch prediction circuit gives the same prediction for all types of branches is known as static branch prediction [6, 7]. While if the prediction changes with the running time, this is called a dynamic branch prediction. For example the processor i486 used static branch prediction algorithm, in which at each coming branch the prediction is always Not Taken [8]. But most of the branches are taken especially the branches of kind Loop, where the branch is taken for all the number of the loop except the last one the branch is Not Taken. As an example for a loop of 100 cycles, 99 of them are taken and only the last one is not taken. Hence, another technique of static branch prediction is used in Pentium 4 processor which is Backward Taken/Forward Not Taken (BTFNT). This is done by counting the value of the new address if it is less than the current address then the prediction is Backward Taken and if the address is greater than the current address then the prediction is Forward Not Taken [9, 10]. The advantage of using static branch prediction algorithms is that they are very easy to implement and needs simple hardware circuit to be added to the designed processor.

### 2.2. Dynamic branch prediction

This type of branch prediction technique will take the advantage of the available information through the run time of branch behavior. The main idea in dynamic branch prediction is to take into account the state of the branch as the time is run which gives better prediction from the static branch prediction [11, 12]. One of the earliest methods used as a dynamic branch prediction is the algorithm presented by [13] (also known as Bimodal Predictor), which is shown in Figure 1. In this algorithm the prediction consists from table recording each previous prediction where it was taken or not taken. It is shown in Figure 1 that the circuit of the branch prediction consists from a group of $2^m$ counters where each one of them recording the previous state of the branch. Since there are $2^m$ counters (entries) the address of the branch instruction (PC) should be hashed down to m bits.

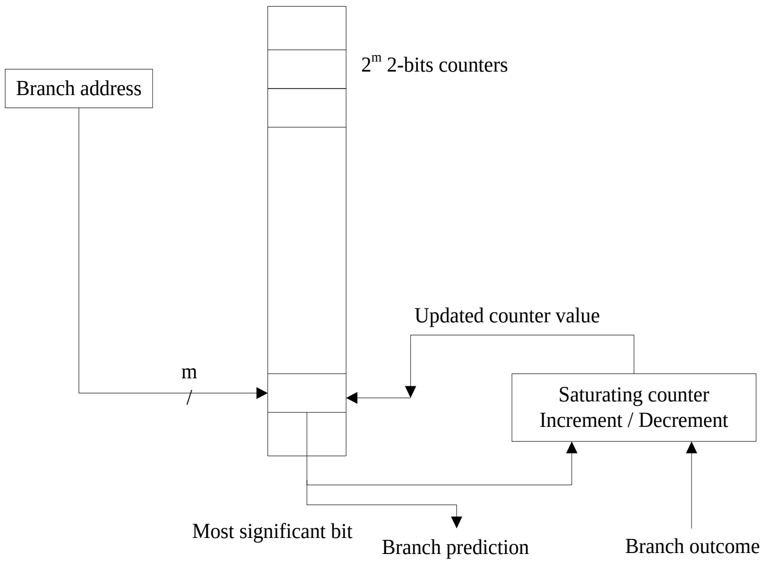

Figure 1. Architecture of basic dynamic branch predictor algorithm

The size of the used counter in this technique consists of 2-bits, which is best from using 1-bit. The MSB of the counter indicates the prediction state while the LSB of the counter indicates the past branch state. In the case of increasing the number of counter bits to 3, the improvement in the predictor algorithm is very small, so that it is always prefers to use 2-bits counter with less hardware from using a larger counter [1, 14].

Some of the researchers used the two-level predictor, which uses a history for the most branch outcomes. These outcomes are stored in a Branch History Register (BHR) which is a shift register where the outcome for each branch is shifted inside the shift register and the oldest outcome is shifted out and discarded [13, 15, 16]. Figure 2 shows the structure for the two-level branch prediction algorithms with global history.

It is clear from Figure 2 that level one consists from the circuit of the Global History Register (GHR) while level two is the circuit of the table which consists from the saturated counters. This table called the Pattern History Table (PHT). The prediction in this algorithm using the outcome of a 4 most recent branches with 2-bits from the branch address to compose a 6-bit which is the index to one of the 64 counters from the PHT. Also there exists another type of the two-level predictor, which is known as the Local History two-level predictor [17-19] shown in Figure 3.

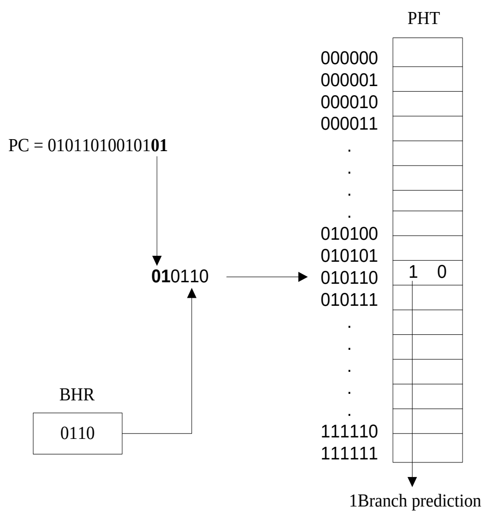

Figure 2. Two-level predictor with global history

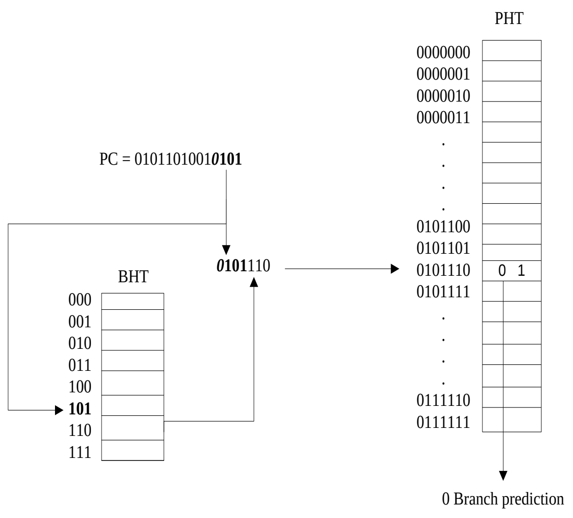

Figure 3. Two-level predictor with local history

This method is done by replacing the Branch History Register (BHR) by a set of counters which known as Branch History Table (BHT). The branch address is used to select one of the entries of the BHT and according to the selected number of these entries as shown in Figure 3. This address will select one of the existing entries (BHR) in the BHT, in which it will give the local history. The contents of the chosen BHR will be combined with the PC to index to one of the counters in the PHT.

### 2.3. Hybrid Branch Prediction

Because there are different types of branches exists in the programs, may be these types are correlated with different types of history. Hence some of the branches may be is better to use the global history algorithm while other branches are better to use with it the local history or any other algorithm correlated with local history algorithm. This difference in the type of prediction algorithms leads some of the researchers to use a Hybrid Branch Prediction (HBP) [20-25]. One of the earliest researchers who used the HBP is [17] who suggests what is called the Tournament Predictor as shown in Figure 4. It is clear from this figure that the Meta predictor (M) consists from a table of 2-bit counters which indexed to it by using the two lower order bits from the branch address. According to the content of these counters the multiplexer will select the predictor $P_0$ in the case of the MSB=0, and choosing the predictor $P_1$ in the case of MSB=1. The Meta predictor works to predict for which algorithm prediction method $P_0$ or $P_1$ is correct.

When the branch outcome is available, the predictors $P_0$ and $P_1$ are updated according to their respective update rules. While the Meta predictor is updated according to different rules. The 2-bit counters will be used in the predictors are finite state machines (FSMs), where the inputs are typically the branch outcome and the previous state of the FSM. For the Meta predictor, the inputs are $C_0$, $C_1$ and the previous FSM state, where $C_i$ is one if $P_i$ predicted correctly. Table 1 lists the state transitions.

When $P_1$'s prediction was correct and $P_0$ miss predicted, the corresponding counter in M is incremented, saturating at a maximum value of 3. While, when $P_1$ miss predicts and $P_0$ predicts correctly, the counter is decremented, saturating at zero. If both predictors are correct, or both miss predict, the counter in M is unmodified. The prediction lookups on $P_0$ and $P_1$ with the state for M are all performed in parallel. When the prediction operations for the three predictors are done, the Meta predictor is used to choose one of the multiplexer lines $P_0$ or $P_1$. The processor Compaq Alpha 21264 [26, 27] used the HBP algorithm as shown in Figure 5.

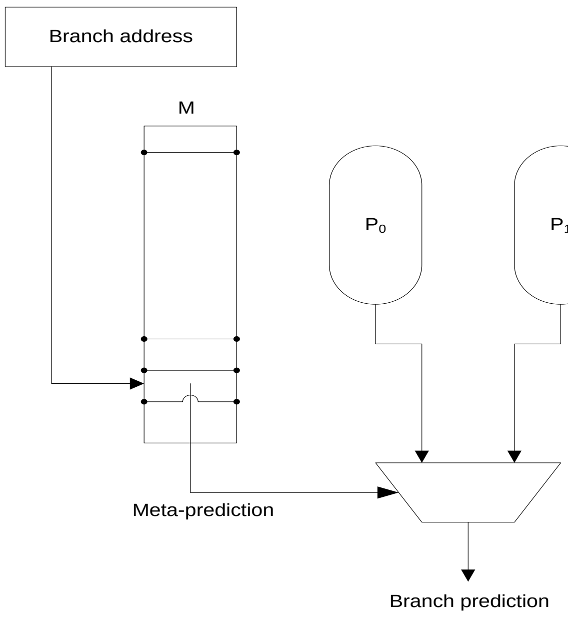

Figure 4. The tournament selection mechanism

Table 1. Tournament Meta predictor update rules

| $C_0$ ($P_0$ correct) | $C_1$ ($P_1$ correct) | modification toM |
|---|---|---|
| 0 | 0 | do nothing |
| 0 | 1 | saturating increment |
| 1 | 0 | saturating decrement |
| 1 | 1 | do nothing |

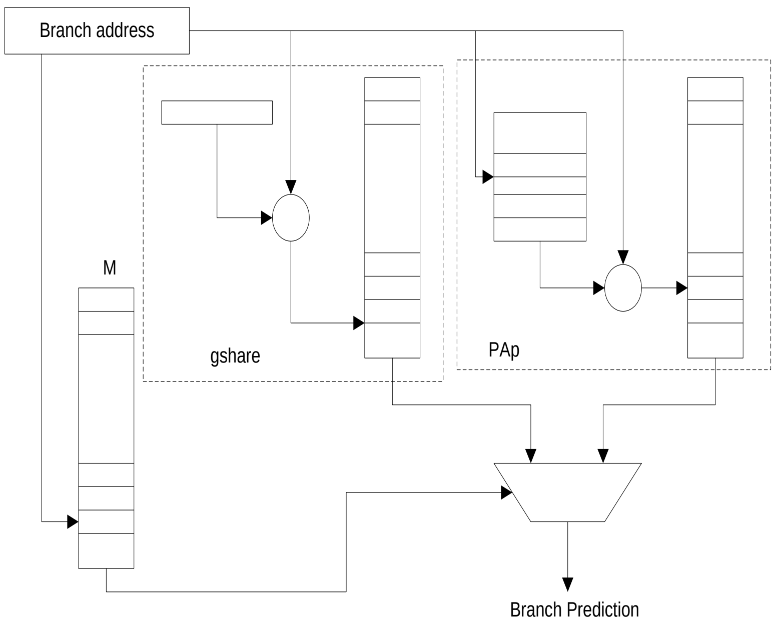

Figure 5. Tournament hybrid for compaq alpha 21264

## 3. SYSTEM DESIGN AND IMPLEMENTATION

In this research the hybrid prediction method is used in the design of the used processor. The processor is a MIPS (Microprocessor without Interlocked Pipelined Stages) pipelined processor with five pipeline stages. The design of this type of processor is a part from the work in the subject advanced computer technology for the MSc course study. This designed algorithm was synthesized using the Xilinx ISE (Integrated Software Environment) design suite 14.7, and using the Vertex-4 Kit with operating frequency of 50MHz. The branch prediction algorithm is designed for the MIPS Processor to be as follows:

- In the case of Unconditional branch and Call/Return, a static algorithm of Always Taken is used. This is because of its simple design and also for its less miss prediction penalty.
- In case of the branch is Conditional, a dynamic algorithm which is the Two-Level algorithm is used as shown in Figure 2.
- Finally in the case of branch of type Loop, the predictor will be dynamic branch predictor of type Bimodal as shown in Figure 1.
- For the two types (two level and bimodal) of dynamic branch prediction a 1024 2-bits counters were used, which is in this case approximately the effect of aliasing not exists. At start all the 2-bit counters will be saturated (its value is 11). For the two-level predictor a 32 Branch History Register is used in the Branch History Table.
- A selector is used to select the type of the prediction algorithm from the three designed algorithms $P_0$, $P_1$, and P2 according to Table 2.

Table 2. Prediction type selection

| Selector o/p | Predictor |
|---|---|
| 00 | $P_0$ |
| 01 | $P_1$ |
| 10 | $P_2$ |

The Prediction Type Circuit (PTC) shown in Figure 6 is used to decide the type of the prediction algorithm according to the executing branch instruction. Figure 6 shows the designed hybrid algorithm. As shown from Figure 6, the input to the Prediction Type Circuit is bits 0:5 and bits 26:31 from the branch address, this group of bits known as function and Op Code respectively. The Prediction Type Circuit examines these two sets of bits and decides the type of branch instruction, and hence the output depends on the type of the branch, then the selector selects one of the predictors according to Table 2.

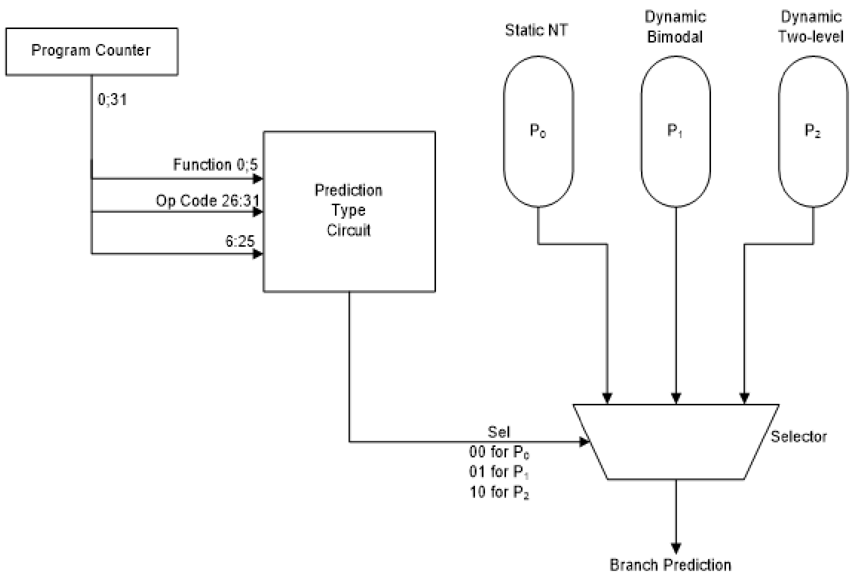

Figure 6. Structure of the designed branch prediction algorithm

## 4. RESULTS

In order to test the designed hybrid branch predictor algorithm and to compare it with different branch prediction algorithms, three different programs were written and executed using the MIPS pipelined processor with the following cases:

- Without using any BP algorithm.

- Using the designed HBP algorithm.
- Using static BP hardware algorithm that always NT.
- Using the Bimodal dynamic BP algorithm with 1024 2-bits counters.

Table 3 shows the different recorded results. It is clear from Table 3 that the hybrid Branch Prediction algorithm gives best results and this is because of using more than one algorithm, where each algorithm is suitable for certain types of branch instructions. Also it is clear that using BP algorithm of any type (static or dynamic) gives better results than not using branch prediction algorithm

Table 3. Execution time for different BP algorithms

| | | No Branch Prediction | Static Not Taken | Dynamic Bimodal | Hybrid BP |
|---|---|---|---|---|---|
| Program Execution Time | Test program 1 | 273 ns | 250 ns | 218 ns | 202 ns |
| | Test program 2 | 366 ns | 314 ns | 297 ns | 272 ns |
| | Test program 3 | 507 ns | 471 ns | 432 ns | 411 ns |

## 5. CONCLUSION

There are different kinds of branch instructions, so that, there is a certain algorithms were suitable for some types for branches while other kinds of branches are suitable for other types of Branch Prediction Algorithms. Hence, three different predictors were used in this work in the same structure which is known as a Hybrid Branch Predictor. This predictor is tested by using a designed 32-bits pipelined MIPS processor using the Xilinx vertex-4 kit. Different test programs were written to test the designed hybrid branch predictor algorithm with the MIPS processor and the results compared with other types of prediction algorithms and it is found that the HBP gave the best results.

## REFERENCES

[1] L. Hennessy and D. Patterson, “Computer Architecture: A Quantitative Approach,” *Cambridge, MA: Morgan Kaufmann Publishers*, 2019.

[2] S. Mittal, “A survey of value prediction techniques for leveraging value locality,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 21, September 2017.

[3] S. P. Dandamudi, “Fundamentals of computer organization and design,” *Springer-Verlag New York*, 2003.

[4] Meyer D.,"AMD-k7 technology presentation," Microprocessor Forum, October 1998.

[5] Hinton, Glenn, Dave S, Mike U, Darrel B, Doug K, Alan K, and Patrice R," The microarchitecture of the Pentium 4 processor," *Intel Technology journal Q1*, pp. 1-13, 2001.

[6] D. C. Burger and T. M. Austin, “The Simple Scalar tool set, version 2.0,” in *ACM SIGARCH Computer Architecture News*, vol. 25, no. 3, pp. 13-25, January 2002.

[7] M. Kampe, P. Stenstrom, and M. Dubois, “The FAB predictor: Using Fourier analysis to predict the outcome of conditional branches,” in *Proceedings Eighth International Symposium on High Performance Computer Architecture*, Cambridge, MA, USA, pp. 223–232, 2002.

[8] Corporation, I., “Embedded Intel486 Processor Hardware Reference Manual,” *Intel Corporation*, 1997.

[9] Corporation, I., “Intel® Architecture Optimization Reference Manual,” *Intel Corporation,* 2003.

[10] P.-Y. Chang, E. Hao, T.-Y. Yeh, and Y. Patt, “Branch classification: a new mechanism for improving branch predictor performance,” in *MICRO*, pp. 22–31, 1994.

[11] P.-Y. Chang, M. Evers, and Y. N. Patt., "Improving branch prediction accuracy by reducing pattern history table interference," *Proceedings of the 1996 Conference on Parallel Architectures and Compilation Technique*, Boston, MA, USA, pp. 48-57, 1996.

[12] J. Bonanno, A. Collura, D. Lipetz, U. Mayer, B. Prasky, and A. Saporito, “Two level bulk preload branch prediction,” *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA),* Shenzhen, pp. 71-82, 2013.

[13] E. S. James, “A study of branch prediction strategies, in 25 years of the international symposia on Computer architecture (selected papers),” *ISCA '98: 25 years of the international symposia on Computer architecture (selected papers)*, pp. 202-215, August 1998.

[14] M. Al-Otoom, E. Forbes, and E. Rotenberg, “EXACT: Explicit Dynamic-Branch Prediction with Active Updates,” *CF '10: Proceedings of the 7th ACM international conference on Computing frontiers,* pp. 165-176, May 2010.

[15] A. F. Joseph, and M. F. Stefan, “Predicting conditional branch directions from previous runs of a program,” *ASPLOS V: Proceedings of the fifth international conference on Architectural support for programming languages and operating systems,* pp. 85-95, September 1992.

[16] Y. Ma, H. Gao, and H. Zhou, “Using indexing functions to reduce conflict aliasing in branch prediction tables,” in *IEEE Transactions on Computers*, vol. 55, no. 8, pp. 1057-1061, Aug. 2006.

[17] S.McFarling, “Combining Branch Predictors,” *Western Research Laboratory*, 250 University Avenue, June 1993.

[18] G. H. Loh, “Simulation differences between academia and industry: A branch prediction case study,” *IEEE International Symposium on Performance Analysis of Systems and Software, 2005. ISPASS 2005,* Austin, TX, pp. 21-31, 2005.
[19] M.Srilatha, K. Artur, and G. Dirk, “Branch Prediction Using Selective Branch Inversion,” *1999 International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00425),* Newport Beach, CA, USA, pp. 48-56, 1999.
[20] Goyal and J. Singh, "Two-level alloyed branch predictor based on genetic algorithm for deep pipelining processor," in *International Journal of Modern Education and Computer Science*, vol. 9, no. 5, pp. 27-33, May 2017.
[21] P.-Y. Chang, E. Hao, and Y. N. Patt, “Alternative implementations of hybrid branch predictors,” *Proceedings of the 28th Annual International Symposium on Microarchitecture*, Ann Arbor, MI, USA, pp. 252-257, 1995.
[22] A. Baniasadi and A. Moshovos, “SEPAS: A highly accurate energy-efficient branch predictor,” *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (IEEE Cat. No.04TH8758),* Newport Beach, CA, USA, pp. 38-43, 2004.
[23] A. Falcon, J. Stark, A. Ramirez, K. Lai, and M. Valero, “Prophet/critic hybrid branch prediction,” *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004,* Munchen, Germany, pp. 250-261, 2004.
[24] M. Evers, P.-Y. Chang, and Y. N. Patt,“Using hybrid branch predictors to improve branch prediction accuracy in the presence ofcontext switches,” in *ISCA*, pp. 3–11,1996
[25] M. Evers, “Improving branch prediction by understanding branch behavior,” Ph.D. dissertation, The University of Michigan, 2000.
[26] R. E. Kessler, “The Alpha 21264 Microprocessor,” in *IEEE Micro*, vol. 19, no. 2, pp. 24-36, March-April 1999.
[27] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeides, “Design tradeoffs for the Alpha EV8 conditional branch predictor,” in *ISCA*, pp. 295–306, 2002.

## BIOGRAPHIES OF AUTHORS

**Ali S. Al-Khalid** was born in Baghdad, Iraq in 1958. He received his BSc and MSc in electrical engineering in 1980 and 1983 respectively from the University of Baghdad. He is now an assistant professor working at the College of Electrical Engineering Technical College, the middle technical university, Baghdad, Iraq. His main interest is in instrumentation and measurement in power line. He is also interested in the field of cryptanalysis.

**Safaa S. Omran** was born in Iraq. I graduated from University of Baghdad in 1978, and then I got the MSc from the same University in 1984. Now I am working in the Electrical Engineering College / Middle Technical University as an assistant prof. My interest working researches in the field of microprocessor design for embedded systems, Image processing and cryptography system design.