RETROSPECTIVE: Design Tradeoffs for the Alpha EV8 Conditional Branch Predictor

André Seznec[1] Stephen Felix[2] Venkata Krishnan[1] Yiannakis Sazeides[3]
[1]*Intel* [2]*Graphcore* [3]*University of Cyprus*

I. Context

This paper was published at ISCA 2002 a year after the cancellation of the Compaq Alpha EV8 microprocessor project in its late phase of development. It is very likely that without this cancellation, the details of the branch predictor would not have been made public.

The main design decisions that lead to the detailed EV8 branch predictor were made during the summer of 1999. During that period, André Seznec was spending a one-year mid-career sabbatical year at Compaq, Stephen Felix was a senior design engineer at Compaq, Venkata Krishnan had recently joined Compaq after completing his PhD and Yiannakis Sazeides was spending a few months in industry before joining academy. Prior to joining Compaq, André Seznec had just published an Inria research report [O19][1] which served as the base of the EV8 branch predictor.

The ambition of the EV8 project was the design of the most performant processor of its generation. The EV8 processor was an 8-issue out-of-order simultaneous-multithreaded processor featuring a very deep pipeline. Therefore, the EV8's performance was highly dependent on the accuracy and the throughput of the overall instruction fetch mechanism. As a result, a significant silicon area was invested in the processor's front-end (referred to internally in Compaq as *IBox*). The EV8's front-end featured an indirect jump predictor, to the best of our knowledge EV8 was the first processor featuring such a type of predictor. Very importantly, the conditional branch predictor had being allocated a very large area budget to allow state-of-the art prediction accuracy and prediction throughput.

II. The Alpha EV8 front-end

The Alpha EV8 instruction fetch was very aggressive, even when considering the state-of-the-art 20 years later. EV8 was capable of fetching up to two consecutive 8-instruction blocks per cycle: an instruction block ended either at the end of an aligned 8-instruction block or on a taken branch. Not taken conditional branches were not terminating the instruction blocks. Consequently, up to 16 conditional branches have to be predicted per cycle, with a maximum of two being taken.

Predicting 16 conditional branches with a high accuracy is quite challenging, even when only a maximum of two of them can be taken. One possible solution to this problem, is to use a line predictor [O1] consisting of three direct mapped tables answering in a single cycle and predicting the addresses of the next two fetch blocks. The accuracy of such a line predictor is relatively poor. But this first prediction can be overridden by a complex PC-address-generation unit that provides in the next cycles: conditional branch prediction, return and jump prediction and branch target computation. In the Alpha EV8, the overriding prediction was completed in two cycles. In some current generation processors, overriding may happen 3 cycles or even later in the pipeline.

Therefore, the EV8 conditional branch predictor is delivering its predictions in two cycles. This enabled to use quite large (direct mapped) tables containing 2-bit counters and indexed with complex hashed index, but could not afford other complex logic (e.g., for updating local branch history).

III. The "academic" prediction scheme

Using local branch history to predict up to 16 branch outcomes per cycle is more than challenging. Therefore, the EV8 conditional branch predictor only used global branch/path history components. The EV8 conditional branch predictor was directly derived from the "academic" *2bcgskew* [O19], but was adapted to the various constraints of the real hardware design.

2bcgskew is a global history branch predictor featuring four direct-mapped tables of 2-bit counters (see Fig. 2 in the original paper). It is a hybrid predictor featuring a meta-predictor that chooses between two predictions: one from an e-gskew predictor [O15] and another from a bimodal predictor [O21]. The bimodal table serves as one of the predictor sources by itself as well as one of the components used for e–gskew prediction. With the help of a well-engineered partial update policy (i.e., not updating always but only when specific conditions are met), 2bcgskew was achieving state-of-the-art prediction at the time of its publication.

IV. Engineering for real hardware constraints

a) Branch history: Academic studies generally use global branch history, i.e., each conditional branch inserts a bit in the global history, or path history, i.e., the history is updated by each conditional branch. On the Alpha EV8, 0 to 16 branches can be predicted in a single cycle. Such global branch history scheme is not realistically implementable, since it requires too complex circuitry for inserting a variable number (0 to 16 bits) on each history vector update. To overcome this complexity, we used a combination of branch and path history,

[1][O..] will refer to the bibliography in the original paper

referred to as *lghist*. More specifically, we XORed only the direction of the last branch in the instruction block (first or second half) with a bit of its PC. Thus, at most a single bit is inserted per instruction block in lghist. Using lghist instead of full branch history was not found to result in any significant accuracy loss. It also resulted in the use of shorter length history registers than when using the conventional global branch history.

b) Three fetch block old history : The conventional way for predicting a conditional branch is to use the address and direction of the branch in the current block A to compute the address of the next block B. However, with a branch predictor answering in a 2-cycle interval, and fetching two blocks per cycle, the tables of the branch predictor are indexed with three-block ahead information, i.e. address A and history up to A are used to index the tables of the predictor to predict branches in block D. Using a 3–block lghist sometimes results in a slight accuracy loss. To recover this accuracy loss, we exploit the fact that each instruction block features up to 8 branches, so the predictor reads 8 contiguous entries from each table. At the end of the prediction a 8-to-8 permutation is performed on these entries, the permutation is controlled by 3 bits; one address bits for each of the 3 intermediate blocks (B, C and D). This allows recovering the accuracy loss due to the use of 3–block old lghist.

c) Conflict-free banked structure: The branch predictor is build with single-ported SRAM. Parallel access to contiguous predictions for a block is quite straightforward. Predicting two blocks in parallel would normally require two-ported SRAM (inducing much higher silicon area, longer access time, higher energy dissipation, ..) or would result in a loss of bandwidth. We have banked the predictor and we have developed a conflict-free access scheme guaranteeing by construction that any two successive fetch blocks will access two different banks in the branch predictor (see Section 6 of original paper).

d) Optimizing storage budgets: Although the EV8 predictor is using 2 bit-counters, on a correct prediction only the low order bit is updated. Consequently, each predictor table consisted of two physical tables, one for the high-order bit (prediction) and one for the low-order bit (hysteresis). As these tables are physically distinct, we optimized the design for different sizes for hysteresis and prediction tables. Finally, the accuracy obtained with a storage optimized 352 Kbits EV8 predictor stands the comparison with an "academic" 512Kbits 2bcgskew.

V. What is the heritage in 2023

An unfortunate development, the EV8 project cancellation enabled the publication of the EV8's predictor organization in great detail. This design remains the most accomplished design for the first generation branch predictors that are based on global branch history and 2-bit counters for prediction (prediction+hysteresis).

The paper illustrated the successful transfer from an academic concept to a practical design in industry. The starting point of the effort was an academic design that we were able to adapt to constraints that had not been originally anticipated. Moreover, the EV8 predictor was the first one to demonstrate that medium long branch history in the 30-bit range had to be considered to achieve high conditional branch prediction accuracy. This set the stage for the next generation predictors that rely on longer global history. Perceptron-based predictors [1], [2] and TAGE-like predictors [3] have been introduced a few years later.

Academic predictors, including 2bcgkew, based on meta-predictions and 2-bit counters, did not remain competitive in terms of prediction accuracy: at equal storage budgets, the EV8 predictor exhibits nearly twice the misprediction rate of a state of the art TAGE predictor.

Predictors of both new families are able to exploit very long histories, in the 100's of branches range, are much more resilient to aliasing than the EV8 predictor. However, for nearly a decade, these predictors were considered as unrealistic by the academic community because of their very long prediction latency. At the same time the industry was working at their adaptation to practical constraints. Nowadays, high–end cores implement TAGE or perceptron inspired predictors.

This clearly supports that microarchitectural research, under review for publication, should be judged (may be even with more weight) for its conceptual contribution rather on how well it is engineered.

References

[1] D. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings of the Seventh International Symposium on High Perform ance Computer Architecture*, 2001.

[2] A. Seznec, "Analysis of the o-geometric history length branch predictor," vol. 33, 07 2005, pp. 394– 405.

[3] A. Seznec and P. Michaud, "A case for (partially) tagged geometric history length branch prediction," *Journal of Instruction Level Parallelism*, vol. 8, pp. 1–23, 2006.