A new organization for a perceptron-based branch predictor and its FPGA implementation

Oswaldo Cadenas,Graham Megson and Daniel Jones
University of Reading, School of Systems Engineering, Reading RG6 6AY, UK
{o.cadenas, g.megson, d.l.jones}@rdg.ac.uk

Abstract

An unaltered rearrangement of the original computation of a neural based predictor at the algorithmic level is introduced as a new organization. Its FPGA implementation generates circuits that are 1.7 faster than a direct implementation of the original algorithm. This faster clock rate allows to implement predictors with longer history lengths using the nearly the same hardware budget.

1. Introduction

Dynamic branch predictors are key to high performance in pipelined processors. Recently, evaluation of branch predictors, inspired in neural methods learning mechanisms, seems to attract attention in research [1, 2]. It can deliver improvements in prediction accuracy of over 20% to the best existing predictors for the same amount of hardware [1]. However, reports are based on simulation analysis rather in real implementation. In fact, area estimates normally do not include the cost of the logic required to do the predictor computation but only on the BHT table size [3]. The extra accuracy of perceptron-based predictors will be nullified in practice due to long combinational delays to compute the prediction function. This paper introduces a new organization of computation to the basic perceptron predictor that is 1.7 faster in FPGA circuits providing the same prediction accuracy at the same amount of resources.

2. The perceptron-based branch predictor

A perceptron computes an output y as the dot product of a weigh vector $w_{1\ldots n}$ and an input vector $x_{1\ldots n}$. A perceptron predictor (PP) is a two-level predictor that stores perceptrons weights in a BHT table instead of two-bit saturating counters. The second level is a standard shift register of length h that keeps a record of the outcomes of the last h recent branches in a register G of h-bits. $G_{1\ldots h}$ is used as an input vector for the perceptron. For a perceptron of history h a BHT of 2^n entries stores in each entry weighs $w_{0\ldots h}$. A design space for the perceptron predictor has been studied in [1, 3]. The PP operates in two phases: a *prediction* and and and *update*. Prediction has to compute, $y = w_0 + \sum_{j=1}^{h} G_j w_j$. A bias input w_0 is introduced by a constant $G_0 = 1$. G_i holds *taken* or *not taken* past branch outcomes t represented as 1 and -1 respectively. Weighs are represented as sign integers. A negative output for y is interpreted as a *predict not taken* while a non-negative y is interpreted as a *predict taken*. The training phase is performed in parallel for $j = 1 \ldots h$ weights according to $w_j = w_j + tG_j$. The new outcome value t is shifted to position 1 of G. This training happens for either a wrong prediction or when $y \leq \theta$, a design parameter. On the left of Fig. 1 shows a direct organization to implement the PP for the case of a history length $h = 4$. Note a chain of adders/subtracters of length $h + 1$ generates the perceptron output y, this is is the critical path for making a prediction.

3. Rearranging the perceptron predictor

The original h weighs w_j for $j = 1 \ldots h$ are replaced by new $\tilde{w}_j$ weighs: $\tilde{w}_j = -w_j + w_{j+1}$;, $\tilde{w}_{j+1} = -w_j - w_{j+1}$ for $j = 1, 3, \ldots, n-1$. The perceptron output can now be computed by $y = w_0 + \sum_{j=1}^{h/2} (-G_{2j-1}) \tilde{w}_{2j+G_{2j-1}G_{2j}}$. Note the new perceptron output y only requires a chain of $h/2$ adders instead of h. This, theoretically, is a reduction in time by half to make a prediction. For each pair of bits of G, a selection is made to whether to take the difference of weighs ($\tilde{w}_j$ or the addition of weighs $\tilde{w}_{j+1}$. At the same time the selected weigh is added or subtracted towards the perceptron output y. This new organization to compute the prediction is shown on the right of Fig. 1. Note only two adders ($h/2 = 2$ in this case) is required to compute the perceptron predictor output y (the adder to accept w_0 can be eliminated). it is straightforward to obtain the new update computation for the modified weights. Increment by two or decrement by two operations are required to either $\tilde{w}_j$ or

Proceedings of the IEEE Computer Society Annual Symposium on VLSI
New Frontiers in VLSI Design
0-7695-2365-X/05 $20.00 © 2005 **IEEE**

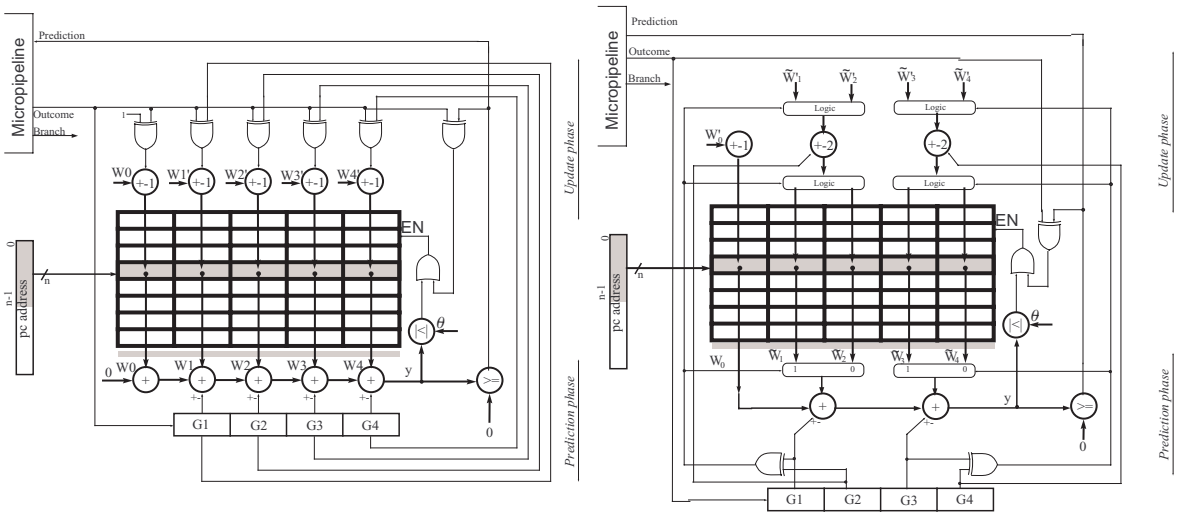

Figure 1. Left: Perceptron predictor and Right: Proposed predictor for $h = 4$.

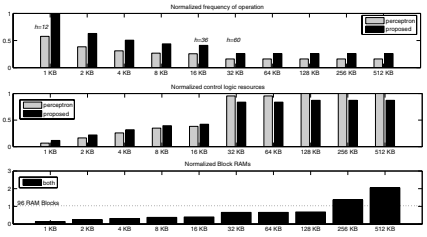

Figure 2. Comparing frequency, logic and RAM for PP (gray) and this proposal (black).

to $\tilde{w}_{j+1}$ but never to both new weights. This can be also advantageously exploited in hardware; we can implement only $h/2$ new incrementers plus some selection circuitry.

4. Results of FPGA implementation

We target our designs to a Xilinx Virtex2-3000 device using Xilinx ISE 6.3. We varied the BTB table size from 4KB to 256KB. This comprises implementations of history length from $h = 12$ to $h = 62$. Fig. 2 (top) shows the frequency of operation for BTB sizes from 1 KB to 512 KB for both designs normalized to the maximum frequency achieved in the target FPGA for any design. Maximum frequency was 73 MHz for $h = 12$ for a BTB of 1 KB. Clearly our proposed organization is always faster for any size implementation. Fig. 2b (middle) shows normalized area in FPGA resources for the control logic for both designs. This area is small of less than 8% of the total device resources. Comparing the size to implement the BHT tables, both designs take the same RAM (Fig. 2 bottom). For long history lengths any design seems to be impractical for FPGA implementation. Area-wise, both designs are roughly the same.

5. Discussion and conclusions

Time results indicate that our proposed organization to the perceptron predictor is capable of extending implementations for history lengths of $h = 4$ for the original predictor to $h = 10$ at 80 MHz for practical BHT sizes of 4-8 KB. This work confirms that for very long history lengths the perceptron predictor seems unsuitable for practical FPGA implementation in present technology. Nevertheless, should this simple perceptron-based predictor become suitable for FPGA or ASIC technology, our proposed organization for the perceptron predictor, for an implementation viewpoint, is clearly faster than the original at roughly the same total hardware.

References

[1] D. A. Jiménez and C. Lin, *Neural methods for dynamic branch predictors*, ACM Trans. on Computer Systems, Vol. 20, No. 4, Nov. 2002, pp. 369-397.

[2] K. Sunghoon, *Branch prediction using advanced neural methods*, TR, UC, Berkeley, 2003.

[3] D. A. Jiménez and C. Lin, *Dynamic branch prediction with perceptrons*. 7^{th} ISHPA, 2001.

Proceedings of the IEEE Computer Society Annual Symposium on VLSI
New Frontiers in VLSI Design
0-7695-2365-X/05 $20.00 © 2005 **IEEE**