International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-830-833 ISSN: 2249-6645

A Study of Energy Efficient Embedded Processor and its Reuse

R. KATHIRESH[1] P. KALIDASS[2] M. SENTHIL KUMAR[3]

1. PG scholar, Department of ECE (PG), Ranganathan Engineering College, Coimbatore, India.
2. PG scholar, Department of ECE (PG), Ranganathan Engineering College, Coimbatore. India.
3. Assistant Professor, Department of ECE (PG), Ranganathan Engineering College, Coimbatore. India.

Abstract:
In Electronics Industry, designing of a processor must meet all requirements, but fails to meet one or two ones. In multi-applications like microprocessors, signal generations and testing of processors. An embedded processor must compute the necessary result when performed through instructions. The efficiency of instruction has attracted much attention since the instruction cache accesses consume a great portion of the whole processor power dissipation and finally leads to inefficient nature. We propose a memory architecture/structure cache to utilize the instructions delivery as an alternative way. The main theme is to reuse the retired instructions from the pipeline back-end of a processor and performs well and efficiency in power.

Key words: **-** Cache memories, Computer architecture, Energy Management, Microprocessors

I. Introduction

Improving the efficiency of instruction delivery has been an important strategy in boosting processor performance. In addition to employ cache memories, schemes for control flow also being proposed. Some of the well known research topics include branch prediction, instruction prediction and trace caches. On the other hand simply allocating more hardware to increase the size of instruction cache has become a viable option for embedded processors. To achieve better efficiency for the cache system, the filter cache scheme being developed to trade performance for better energy. However this leads to degradation in performance of increased access latency. Essentially the front end of processor improves the energy efficiency of instruction delivery. The efforts to aim to speculate the right program traces prior to the branch instructions are resolved, reduces the program execution latency or the energy consumption via the speculated trace information. Since the speculated traces given that they are correctly predicted, will ultimately be retired from pipeline and become history traces. These executed traces are potentially very useful in case of an embedded processor. For more number of instructions to execute the embedded processor will take more cycles to complete the loop due to branch prediction. Branching can be avoided, if embedded processor fetches instructions in the history trace.

II. Existing survey

To investigate the feasibility of delivering instructions from the pipeline back-end, we perform simulations using the processor model as shown in fig., The architecture consists of an embedded processor with additional D-Flip flops at each stage of the pipeline and HTB at the back end. The HTB is managed as a FIFO (first input first output) buffer to capture a fixed length of most recently retired instruction sequence. For each instruction fetched from the front-end, the HTB is searched to see if the same instruction also hits in the buffered history trace. The total instructions fetched are summed throughout the simulation to calculate the raw HTB hit rate

Due to the reduced complexity and size, the HTB is far less than power hungry than the instruction cache. If an instruction can be delivered from the HTB whenever a hit occurs, an energy saving proportional to the hit rate can be achieved. In this paper we propose a novel scheme called Trace Reuse (TR) cache to improve the energy efficiency of instruction delivery for embedded processors. These instructions are useless for program execution but are inevitable for perfect predictors.

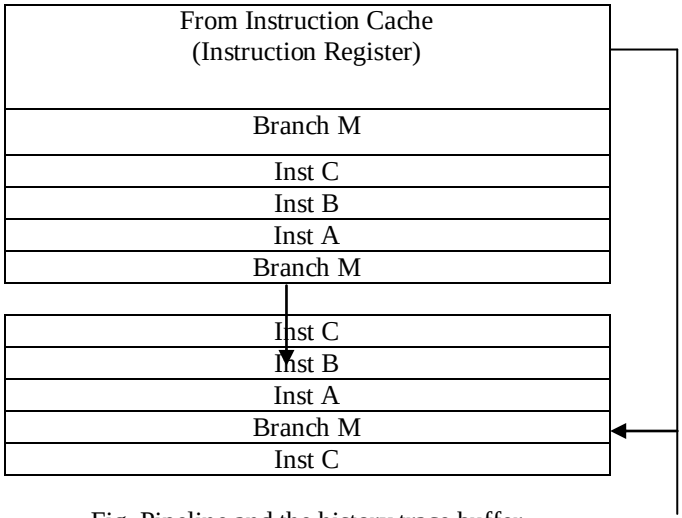


Fig. Pipeline and the history trace buffer

III. TR CACHE & design

The TR-CACHE is composed of a History Trace-Buffer (HTB), which collects the instructions retrieved from the pipeline back end of the processor, and a trace entry table. We present a TR Cache architecture that is capable of delivering instructions from HTB for embedded processors. The architecture of the TR Cache is shown as follows:

A simple multiplexer and mode switching logic are integrated into the fetch stage to select the proper instructions source. We present a TET design to index the HTB buffer for instruction source. The TET is a small memory structure used to group the instructions in the HTB buffer. The TET stores the trace-entry records each of which consists of the PC value of a control transfer instruction and the corresponding HTB entry index. The contents of HTB and TET are updated as follows: whenever an instruction is retired from the pipeline backend, it is buffered in the HTB along with its PC.Since the HTB is managed in a FIFO fashion. The oldest instruction will be discarded to make room for new one. If the discarded instruction is a marked trace entry, the corresponding record in TET will also be invalidated.

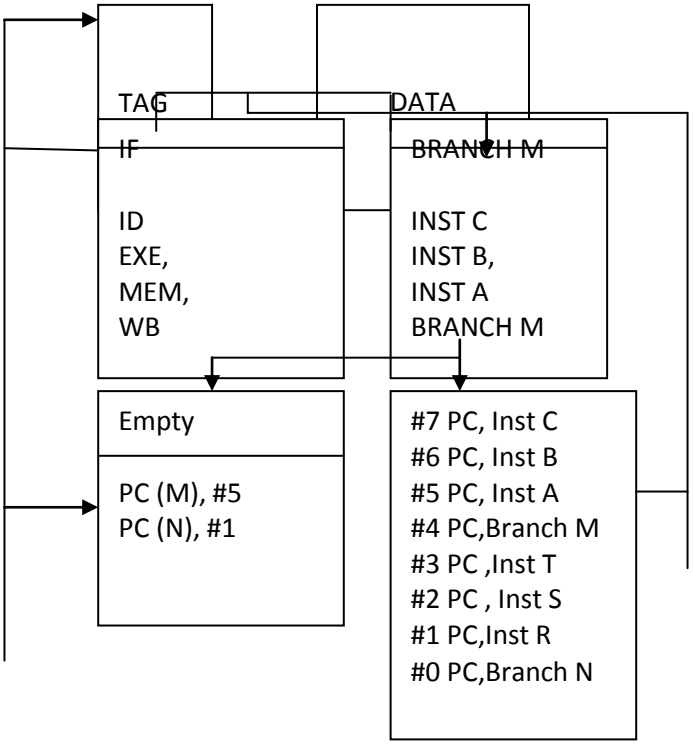


Fig: TR Cache Architecture for Embedded processor

To improve instruction buffering efficiency, a more complex basic block threading mechanism such as the design in may be used. On the other hand, the HTB is a simple FIFO without the complex logic for associative lookups. The HTB uses a simple index-based access mechanism, to be presented which brings advantages in power and area usage as compared with an associative look-up based design.

Flow method of TR Cache

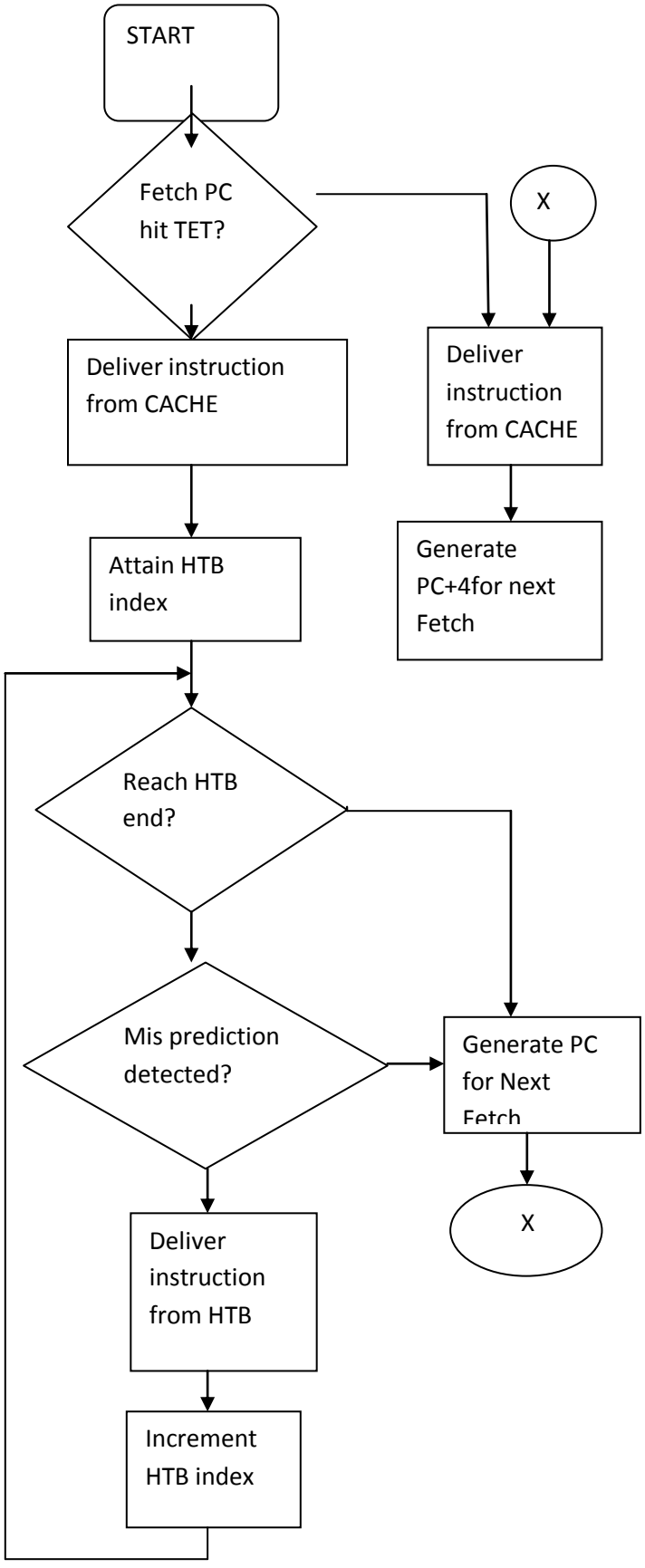
The TR Cache an alternative source for instruction delivery, a new access mode is integrated to the fetch logic. We name the original access mode as the cache mode and the new ones as the TRC Mode. By Default the processor is in cache mode cycle, the TET is searched in parallel with the cache. The processor remains in cache mode until the TET search returns a HIT.

When Hit occurs, the HTB index returned by the TET will be latched and the processor will switch to the TRC mode at the next cycle. The availability of the incoming instruction is conformed by checking the current HTB index against the HTB boundary pointers. If the index has reached the end of the HTB, the TRC-mode operation will be aborted and the next PC will be generated for the cache-mode operation.

HTB-Size (Instructions)	32	64	128	256
Max.TET record count	15	22	41	72

Table1.different HTB sizes Vs TET record count

The TET Size to be used is actually dependent on both the size of the HTB and the program behavior.The instruction delivery of TRC is as follows:



IV. RESULTS

The new RTL code of the modified processor core,which includes the register file and the completed pipelined datapath is synthesized using UMC 90-nm technology library.

A. Access time and area estimations

It is clear that the access time of the direct-mapped TETs is far les than that of the 16KB/32-Way Instruction Cache. This shows that TET search in parallel with the cache mode operation has no impact on the processor cycle time.

B.IPC Performance analysis

The average IPC is due to the large variations in the elapsed cycle counts of the programs.

$IPC_{average} = \Sigma I_K / \Sigma C_K$

The TR Cache provides significant performance improvement over the baseline processor.

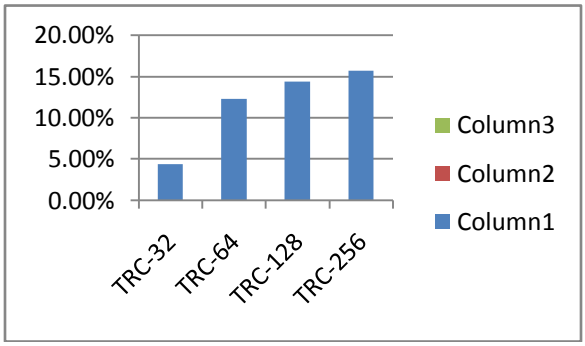


Fig. Average IPC Improvement Rate.

The basic idea behind branch predicition is to extract useful branch targets from the history trace.

C. Energy Efficiency

We present the evaluation of energy consumption and normalized energy-delay product of using the TR Cache. The Energy consumption of the fetch logic mainly comes from the power dissipation of the instruction cache and the augmented memory structures such as TET and HTB.

V. CONCLUSION

In this paper, the TR Cache architecture is proposed as an alternative source for instruction delivery of embedded processor. The processor can switch to the TR cache when a reusable trace is identified. The main difference of having TR Cache is that it can compute post–execution program information in a sequence over the conventional instruction cache.

REFERENCES

[1] yi-yting tsai,chung-ho chen,"*Efficient Embedded processors*"Tech.Rep.Sep 2011
[2] S.McFarling,"*Combining Branch predictors*",Digital WRL,Jun.1993.
[3] J.L.Hennessy and D.A.Patterson ,*Computer Architecture:A Quantitative* a*pproach*,4thed. San Francisco,CA:kaufman 2006.
[4] E.Rotenberg,S.Benett and J.E.Smith ,"*A trace cache microarchitecture and evaluation*",IEEETrans.Compt.,Vol.48,no.2,pp.111-120,Feb.1999
[5] A.Hossain,D.J.Pease,J.S.Burns and N.Parveen,"*Traces cache performance parameters*,"IEEETrans.Comput.Des.,Feb.2002,pp.348-255.
[6] J.Kin,M.,Gupta and W.H.Magione-Simith,"*Filter cache:AN energy efficient memory structure*",in proc.30th int.Symp./Microarch.,Dec 1997,pp.184-193.
[7] T.Austin,E.larson and D.Ernst,"*Simple* Scalar,"*An infrastructure for computer system modeling*",IEEE Trans.Comput., vol.35,no.5,pp.505-517,May 2007.
[8] A.Soldani and G.S.Sohi,"Dynamic Instruction Reuse",in Proc.24th ANNU.Int.Symp.Comput.Arch.,Jun.1997,pp.335-340.