Partial Address Field Architectures For Energy Efficient Caches in Embedded Systems – A Review

Inderjit Singh
Electronics and Communication Engineering Department
DAV University
Jalandhar, India
singh.inderjit@ymail.com

Tajinder Kaur
Computer Science and Engineering Department
Jaypee Institute of Information and Technology
Noida,UP
er.tajinderkaur@gmail.com

***Abstract*—** Most of the embedded processors utilize cache memory in order to minimize the performance gap between memory systems and processor. In embedded systems caches are normally implemented along with processors in one IC. The power consumed by the cache system constitutes the major fraction of the power dissipated by the embedded processors.
With increasing computational demands on embedded processors, set-associative caches are being used. In larger caches the major portion of power consumption occurs in address decoding including tag comparisons. Set-associative caches consume larger energy as compared to the direct mapped caches as i) set-associative caches have greater tag bits, ii) they have parallel organization of tag arrays, and hence parallel tag comparison dissipates more energy. It is further analyzed that not all the tag bits are necessary for a cache configuration to achieve a normal performance in terms of hit rate. Hence, architecture with reduced but
optimum number of tag bits is possible, which would consume lesser energy.
Currently lot of research is going on to find newer architectures for Cache designs so as to give better energy savings. In this paper it has been shown that there are various considerations while designing the cache architecture and a review work done is shown on existing power reduction and performance enhancement schemes in cache architecture design.

Keywords- *Low Power Cache, Set-associative caches, Cache architecture design, Partial tag comparison.*

*****

I. INTRODUCTION

Cache is the first level of memory hierarchy encountered once the address leaves the CPU. Comparatively larger data and instructions than requested by the processor are stored from main memory to a smaller but high speed cache memory. Processor preferably accesses this high speed cache memory, resulting in overall system performance improvement [1].

The cache memories in embedded architectures are implemented on-chip. Being comparatively larger than simple cores they dissipate a major fraction of power in embedded SoC architectures. Hence power dissipation in battery operated embedded devices and wireless mobile gadgets is a serious issue, making power management in cache design a concern in embedded architectures [2].

Earlier, performance and area were the two main factors involved in cache design, but recently power has become an increasingly more important design consideration. Largely, power consumption in cache occurs in data bit-lines, sense amplifiers and comparators. As the associativity is increased, the address decoding including tag comparison consume larger fraction of power [3]. Because larger associativity increases the number of parallel sets and hence increases the parallel tag comparisons in associative caches. We in this paper present here various architectural advances in modern cache design enabling lower power operations without affecting the functional performance.

Conventional caches use full tag array holding tag value of each cache line in the cache. In associative caches, when degree of associativity increases the numbers of sets increase and hence the numbers of parallel tag arrays. In a typical case of 16KB 8Way Qword cache, there are 21 tag bits in one set, and a single memory access triggers comparison of 168bits (21x8=168) of 8 parallel 21-bit tag values. Such an operation consumes power and introduces delay [4].

Although a lot of research is also going on for better cell structures, but many different successful architectural proposals has been given providing better performances.

II. PARTIAL ADDRESS DIRECTORY FOR CACHE ACCESS

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

A. Operation

This technique utilizes approach to partially compare the tag array (address bits) to speed up the cache operation. It uses a separate partial tag array called Partial Address Directory (PAD), as partial address. The PAD can speed up most cache array access by accurately predicting cache locations without having to wait for results from conventional cache directory (Full Tag) lookups. In each memory access operation, the partial tag (PAD) comparison enables the selection bits to select data from data array using data multiplexer. The figure 1 shows 4 way set associative design [5].

Using partial address comparison makes out a possibility of more than one match along the sets, which may trigger a fallacy situation where more than one selection bits of multiplexer are high. This situation (called ghost hit) has been rescued by incorporating the conventional tag-array called Cache Directory too, into the architecture. When such a situation occurs, the conventional cache directory comes into

 ISSN: 2321-8169
2814 – 2818

play, giving correct select signals and updates PAD. As this operation is a read, altogether once again; it consumes additional machine cycle.

In a normal operation, comparison in PAD late-selects one of the data values from different sets, without waiting Cache Directory lookups; thereby making the operation faster and power efficient.

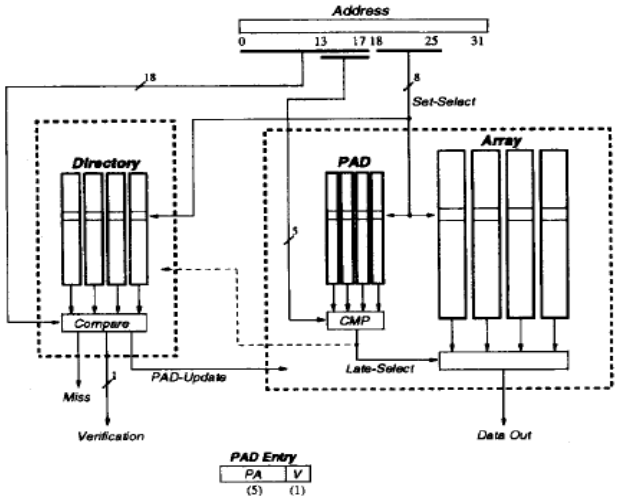

Figure 1. Partial Address Directory Architecture [5]

B. Gaps

- Design uses Cache directory (Full Tag) array and PAD (Partial Tag) array, i.e. two separate directory lookups. This will lead to consumption of larger area as compared to conventional design.
- Two tag arrays will use two set of comparators upon *ghost hit*, consuming larger power.
- Also author proposes the Cache Directory to be implemented separately, only PAD to be embedded into cache structure. It may result into larger access times when accessing a distant Full Tag Directory upon *ghost hit*.

C. Motivation

- The design achieves greater speeds for greater address bits, hence for the designs with greater associativity.
- Also as the number of sets increase, the number of distinct partial addresses (distinct PAD entries) per set increase.
- Although author proposes possibility of reducing Full Tag array thereby modifying the design with two arrays containing two portions of the full tag address i.e. two arrays having separate 16 and 5 bits of full 21 bit tag

D. Summary

- The design uses PAD as faster portion of Cache, and it achieves better speeds of operation consuming larger area and lesser flexibility.
- The author hasn't clearly reported comparison of power consumption.

III. Partial Tag Resolution in Data Value Predictors

This technique uses partial tag for data value prediction, resolving data dependences thereby increasing the Instruction Level Parallelism (ILP) for processing at considerable low hardware costs.

As clear in the figure 2, the conventional Full Resolution tag implementation stores full tag value into the Data/Tag array. The partial resolution figure 3, implementation stores only a part of tag bits. Owing to address sequences being linear in nature, only small portion of tag is able to uniquely predict the correct data values.

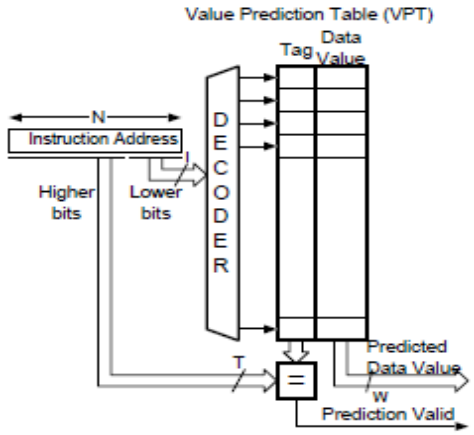

Figure 2. Full Resolution VPT [6]

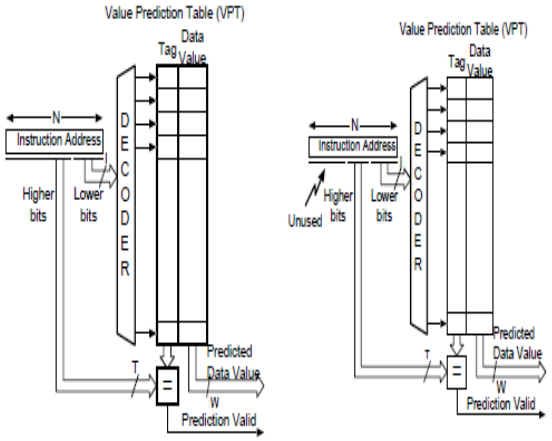

Figure 3. Partial Resolution VPT [6]

A. Operation

Lower address bits select the decoded line, and a portion of higher order address bits (tag) is then matched with the stored tag bits from the decoded line [6].

For an instruction address of N bits and VPT indexed by I bits, the tag address is $T = N - I - 2$ bits. But using partial resolution tag address is $T < N - I - 2$.

B. Gaps

This work introduces various partial tag resolution policies to reduce hardware costs, not the reduction of dynamic power.

 ISSN: 2321-8169
2814 – 2818

C. *Motivation*

- The Value predictors for Data Speculation don't require full resolution tag values.
- Destructive aliasing if introduced by partial resolution can be avoided by deciding enough bit length of tag address.
- The contribution of this work is the introduction of various partial tag resolution policies, such as stride, last-value, two-level, etc. However, the main focus of this work is reducing hardware costs, not the reduction of dynamic power; thus, the results of power management are not provided in their work.

IV. COST-EFFECTIVE VALUE PREDICTION USING PARTIAL TAG

This used above technique of reduced tag comparison in the caches. It also introduces value prediction policies. The partial tag address is generated by hash function in this technique [7].

A. *Motivation*

- It also shows the possibility of partial address generation and provided the hit ratios for each partial address generation method in Direct-Mapped caches as suggested by previous technique.
- The technique proposes the policies for Direct Mapped Caches not set associative caches.

V. PARTIAL TAG STRUCTURE: EARLY SWITCHING OF SENSE AMPLIFIERS

This technique adds a very small tag dedicated for the amplifiers. This small tag array is only several bits wide. It is organized and accessed as a regular tag.

The architecture of this partial comparison cache is given Figure 4(a), with the corresponding cache access paths shown in Figure 4(b). The shadowed path is the new SE path.

A. *Operation*

The small tag array keeps a copy of the least significant bits of the original tag. For each way, a few significant bits are compared. Only when there is a match, the sense amplifiers attach to the data array bit-lines will be enabled. Because of its small size, the new hardware can complete the comparison operation using only 60-80% of the original comparison time [8].

B. *Gaps*

The technique uses additional small tag array in addition to the regular tag array. Not area effective.

False-hits are possible in partial tag comparisons, in which the results of comparisons seem to be cache hits but are actually false hits.

Technique only emphasizes the power reduction in the multiple sense amplifiers on the data-array side, neglecting power consumption in comparators, which compare a large number of tag bits in the set associative caches.

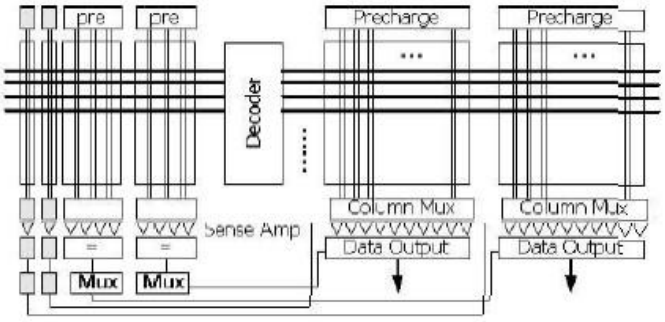

Figure 4(a) Early Switching of Sense Amplifiers [8]

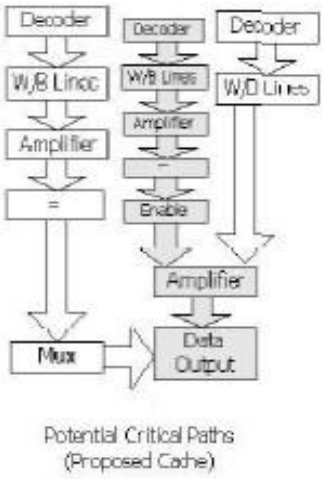

Figure 4(b) Critical Path [8]

C. *Motivation*

- Technique successfully uses partial tag comparison to enable the sense amplifiers well before the data reaches them, thereby enabling only one sense amplifier depending on the way selected.
- The technique is well designed for the set-associative caches, and reports 25-60% power efficiency over conventional technique

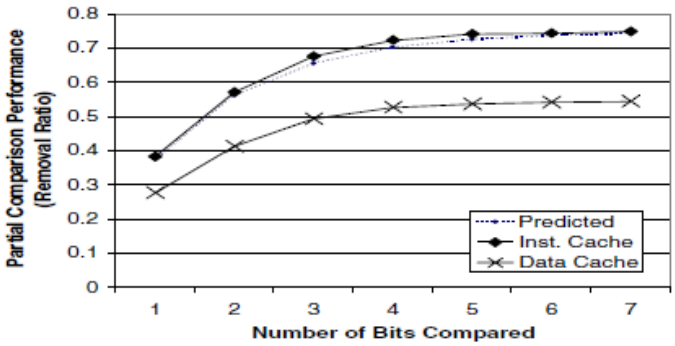

Figure 5. Partial Comparison vs. Number of Bits Compared [8]

VI. TAG OVERFLOW BUFFERING

D. *Operation*

Tag is broken into two parts. The LSB-side (*called TagL*) is stored normally in the cache as regular structure. The MSB-side (*called TagH*) of tag is stored in a separately into an

 ISSN: 2321-8169
2814 – 2818

external register called Tag Overflow Buffer. Similar to conventional cache operation, the TagL corresponding to the decoded wordline is compared with TagL address field generating TagL Hit/Miss. For the TagH side, the input TagH is compared with the existing contents of TOB, if it is also Hit cache operation is completed successfully. Upon miss, the Locality Change Detection mechanism predicts for the degree of change of locality, if requested address corresponds to a different locality plus the number of requests exceeds threshold, then the TOB is updated with the candidate TagH address [9].

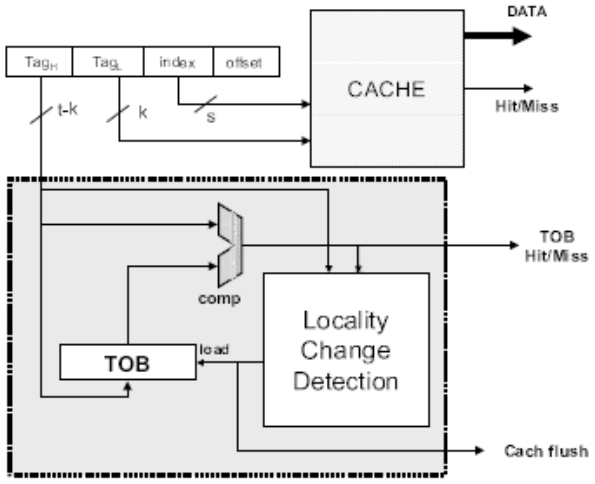

Figure 6. Dynamic TOB-Based Architecture [9]

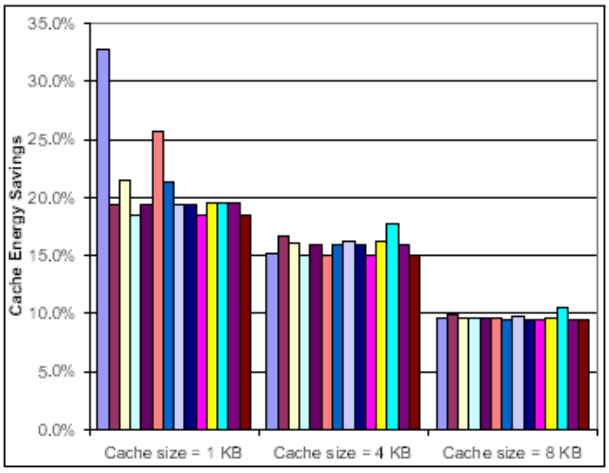

Figure 7. Percentage Energy Savings [9]

E. Gaps

- The architecture becomes very fragile when application program frequently change memory access localities i.e. the hit ratio can dip.
- Change of associativity has no impact on the power savings, the power reduction in these cases is no better than Direct Mapped Caches.
- The design works better for smaller caches.

F. Motivation

- The architecture is helpful for reducing false hits.
- In this policy, the average energy savings on the tag array is about 48%, corresponding to a savings of about 20% on the total cache energy.

VII. Compressed Tag Architecture

This technique uses a Locality Buffer called *LoB* like *TOB* but has more than one entry in it. Like in the previous techniques it also utilizes partial tag comparison in the cache structure, enable lower power consumption. Similar to techniques discussed earlier LSB-side tag called *TagL* is stored normally as a part of cache structure. The *TagH* is stored in accordance with the policies in the locality buffer containing a few entries. Also to select between the *LoB* entries *LCB* bits are also allocated along with the *TagL* bits.

A. Operation

Similar to conventional cache operation, lower tag bits i.e. *TagL* are compared resulting in Low Tag Hit/Miss. Also *LCB* bits corresponding to the selected wordline are used to select the *LoB* entry, which is compared with the *TagH* to generate *LoB* Hit/Miss.

Low Tag Hit along with *LoB* Hit convey an overall Hit operation. On the other hand, a *LoB* miss will trigger the other design policies [10].

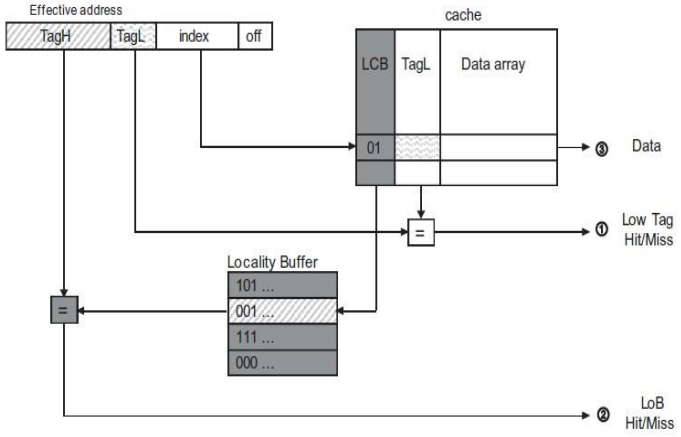

Figure 8. Compressed Tag Architecture of Cache [10]

B. Gaps

- The design policies limit it for Direct Mapped Caches only i.e. a similar design when applied to set-associative caches give no better results.
- As the cache size increases the energy consumption improvement is degraded.

C. Motivation

- The proposed architecture is free from false hits.
- The design establishes quite lesser degree of fragility and can handle frequent program locality changes.
- The design provides a flexible sensitivity to the frequency of change of program localities.
- Design works better for smaller caches.

 ISSN: 2321-8169
2814 – 2818

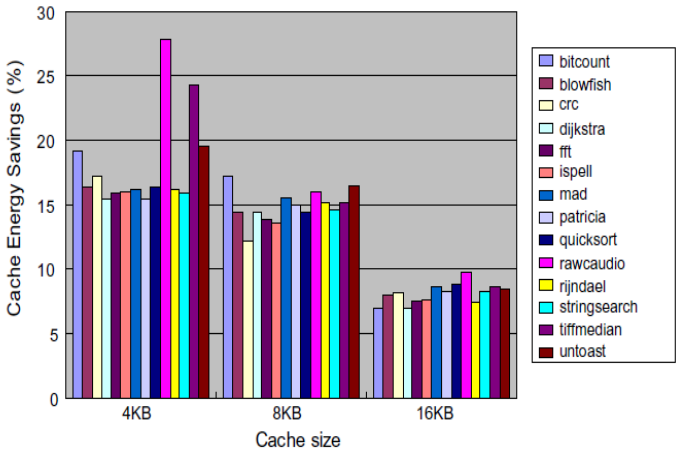

Figure 9. Percentage Energy Savings [10]

- Proves to be the best design utilizing existing principles to achieve average energy savings improvement of 20%, but for smaller and direct-mapped caches.

CONCLUSION

The partial address based techniques optimize the cache architecture for lower power by eliminating long tag bits and utilizing partial tag comparisons for data validation. Partial tag comparison also reduces the access times thus making the architecture faster yet energy efficient. *Kwak* and *Jeon* present a beautiful consolidated architecture that picks the cream of all techniques and saves about 20% of energy in small direct mapped caches. Common fundamental limitations to all techniques are additional hardware, false hits, false hit penalties and reframing of cache read/write policies.

Reviewed Partial tag architectures work better for direct mapped small sized caches. In larger caches the major portion of power consumption occurs in address decoding including tag comparisons. Set-associative caches consume even larger energy as compared to the direct mapped caches as i) set-associative caches have greater tag bits, ii) they have parallel organization of tag arrays, and hence parallel tag comparison dissipates more energy. It is further analyzed that not all the tag bits are necessary for a cache configuration to achieve a normal performance in terms of hit rate. Hence, architecture with reduced but optimum number of tag bits can be developed for set-associative caches using the reviewed principles presented here, which would consume much lesser energy compared to full tag set-associative architectures.

ACKNOWLEDGMENT

Inderjit Singh thanks all the anonymous reviewers who have given their valuable feedback which helped him successfully complete this paper.

REFERENCES

[1] Intel Corporation, “An Overview of Cache”, article online at http://www.intel.com/design/intarch/papers/cache6.htm

[2] Alipour, M.; Salehi, M.E.; Moshari, K., "Cache power and performance tradeoffs for embedded applications," Computer Applications and Industrial Electronics (ICCAIE), 2011 IEEE International Conference on, vol., no., pp.26,31, 4-7 Dec. 2011.

[3] Glen Reinman and Norman P. Jouppi; “CACTI 2.0: An Integrated Cache Timing and Power Model”, HP Labs, WRL Research Report 2000/7

[4] John L. Hennessy & David A. Patterson, “Computer Architecture: A Quantitative Approach”, Prentice Hall, third edition, Morgan Kaufmann Publishers, 2003.

[5] L. Liu, Partial address directory for cache access, in: IEEE Transactions on VLSI Systems, vol. 2, No. 2, pp. 226–240, June 1994.

[6] T. Sato, I. Artia, Partial resolution in data value predictors, in: Proceedings of International Conference of Parallel Processing, pp. 69–76, 2000.

[7] B-S. Choi, D-I. Lee, Cost-effective value prediction micro-operation using partial tag and narrow-width operands, in: PACRIM’01: IEEE Pacific Rim Conference on.

[8] R. Min, Z. Xu, Y. Hu, W.-B. Jone, Partial tag comparison: a new technology for power- efficient set-associative cache designs, in: VLSID’04: 17th International Conference on VLSI Design, pp. 183–188, Jan. 2004.

[9] M. Loghi, P. Azzoni, M. Poncino, Tag overflow buffering: an energy-efficient cache architecture, in: Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, pp. 520–525, March 2005.

[10] Jong Wook Kwak, Young Tae Jeon, Compressed tag architecture for low-power embedded cache systems, ELSEVIER, Journal of Systems Architecture, Volume 56, Issue 9, pp. 419–428, September 2010.