# A NUCA Model for Embedded Systems Cache Design

Pierfrancesco Foglia, Daniele Mangano, Cosimo Antonio Prete
*Dipartimento di Ingegneria dell'Informazione, Via Diotisalvi 2, 56100 Pisa, Italy*
*{foglia,daniele.mangano,prete}@iet.unipi.it*
*Members of the Hipeac EU Network of Excellence*

## Abstract

*Future embedded applications will require high performance processors integrating fast and low-power cache. Dynamic Non-Uniform Cache Architectures (D-NUCA) have been proposed to overcome the performance limit introduced by wire delays when designing large cache. In this paper, we propose alternative designs of D-NUCA cache, namely Triangular D-Nuca Cache, to reduce power consumption and silicon area occupancy of D-Nuca cache. We compare the performances of Triangular D-NUCA cache with conventional rectangular organization. Results show that our approach is particular useful in the embedded applications domain, as it permits the utilization of half-sized NUCA cache with performance improvements.*

## 1. Introduction

Future mobile embedded environments need to support sophisticated applications such as speech recognition, visual feature recognition, secure wireless networking, and general media processing. These applications are computation intensive, and require more performance than the one current embedded processor can deliver [16], [17]. Traditional ways to increase performances of embedded processors include technology scaling (with the consequent performance increase due to the Moore's law) [17], and the tuning of cache hierarchy parameters [18]. Both the techniques will be no more applicable to the design of such family of processors, due to the wire delay problem and the power requirement of embedded hand-held applications. Traditional cache tuning and technology scaling techniques determine increased chip power consumption [17], [18], increased chip area devoted to the memory subsystem [14], [15], and an increased wire delay (also due to the increased memory area) , which, in turn, limits the overall processor performances [1]. As a consequence, new design techniques for processor architecture and memory hierarchy are required.

Focusing the attention on the memory subsystem, the NUCA (Non Uniform Cache Architecture) Caches [7, 8] have been recently proposed to overcome the bottleneck due to growing wire delays in general purpose systems. NUCA caches are large L2 caches, organized in sub-bank. Each sub-bank can be accessed independently, with an access time depending on its physical distance from the cache controller (it is the only delay that must be paid for accessing the particular bank), thus achieving non uniform access time.

In this paper we present an alternative design of D-NUCA cache, targeting the minimization of both silicon area and power consumption. Because of the geometric shape, we named our proposal Triangular Dynamic NUCA (TD-NUCA). Basically, TD-NUCA caches are D-NUCA caches with a variable number of banks when moving in the opposite direction of the controller. In particular, we deal with two organizations, increasing and decreasing TD-NUCA. The results of our investigation show that, in the general purpose case, TD-NUCA design enables to reduce the silicon area by approximately 50%, while paying some performance degradation. In the cases typical of embedded applications (where the application is known in advance), TD-NUCA cache permits a silicon area reduction of approximately 50%, while outperforming the solutions based on DNUCA cache. Our research highlights that more studies on the mapping policies and geometry are worthwhile, as we have explored only few directions (i.e. reducing the number of block and adopting fixed mapping policy) in a wide design space, nevertheless achieving promising results.

## 2. Rationale of TD-NUCA caches

In the NUCA architecture proposed in [7], [8], the L2 cache is organized in banks, which define a rectangular memory geometry. Each bank may be accessed independently, with an access time depending on the physical location of the bank. An interconnection infrastructure, based on switch (in the simpler design), on a wormhole routed 2-D mesh with point-to-point links, or, better, on a more general Network on Chip [23], [24], [25] is utilized to guarantee the bank-controller communications. The design space of NUCA cache is very large: a Mapping Policy defines the number of addressable banks and how memory lines are mapped to this bank; a Search Policy defines the set of possible locations for a line; a Movement Policy defines the way in which a line is moved, either



0-7803-9347-3/05/$20.00 

while resident in the cache or across different lifetimes in the cache. Many of these policies have been explored in [7,8]; results show that DNUCA cache achieves 1.5 times the IPC of a traditional Uniform Cache Architecture (UCA) of any size

A typical distribution of the bank accesses in a rectangular D-NUCA cache, when adopting the most performing policies ([7], [8]), is showed in Figure 1. Such a distribution is a consequence of the best migration mechanism [2], which implies that the most accessed data migrate near the controller, while less used data move toward the opposite side and possibly are evicted from the cache. The exact shape of accesses distribution depends on several issues (i.e. the application, the mapping policy, the migration policy, etc.), but it has a qualitative distribution as the one shown in Figure 1.

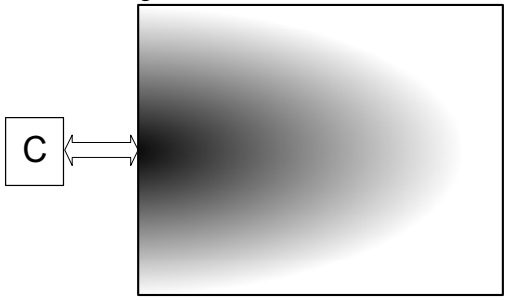


**Figure 1: Qualitative accesses distribution in a D-NUCA cache. Due to the migration policy, the most accessed banks are within the darker area.**

By analyzing the accesses distribution of Figure 1, our idea is to modify the original D-NUCA design, by eliminating from the cache the banks related to the less used data, i.e. by implementing a cache with a decreasing number of entries within a way, when moving in the opposite direction of the controller. In this way, we could be able to reduce the cache size, and consequently the cache area, with low performance degradation. By eliminating banks, we can also reduce the static power consumption. This is an important issue in the design of embedded system, as cache memory may consume up to 50% of total chip power, while static energy dissipation is going to account for an increasing portion of total energy in current and future technologies [19], [20].

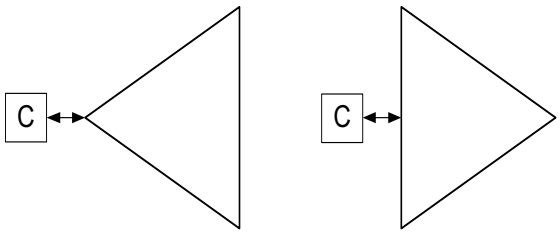


**Figure 2: Increasing and decreasing TD-NUCA organizations.**

The above idea leads to a triangular organization, i.e. a Triangular D-NUCA (TD-NUCA) memory and we consider the two organizations showed in Figure 2. According to the accesses distribution, the decreasing solution should meet lower miss rate than the increasing solution. On the other hand, by using specific technique [11], the increasing organization could allow a higher power consumption reduction. In fact, power consumption includes static and dynamic terms. The dynamic term could be reduced by using transistors with different threshold voltages: transistors located on most used banks are high-power and faster, whereas the transistors located in the further banks are low-power but slower [11].

## 3. D-NUCA caches and related works

Although significant prior work has evaluated large cache design [5, 4], the D-NUCA cache architecture has been proposed in [7, 8], showing that a dynamic NUCA design achieves 1.5 times the IPC of a traditional Uniform Cache Architecture (UCA) of any size. In order to evaluate the effects of the different organizations on system performance, the author developed an experimental methodology based on Cacti [12] to derive the physical model of cache memory, and an extended version of the sim-alpha simulator [3] to simulate the different organizations with parameters derived from Cacti. In a more recent work [9], a fully-associative approach for NUCA memory has been proposed. In this case, a Globally Asynchronous Locally Synchronous (GALS) Network on Chip (NoC) is employed as communication infrastructure, according to the switched-network paradigm. D-NUCA memories have been also analyzed in the contest of on chip multiprocessor as shared level-two caches [2]. In such a work commercial and scientific benchmarks have been employed, and the results show that migration mechanism are less effective for CMPs because 40-60% of L2 cache hits in commercial workloads are satisfied in the central banks, which are equally far from each processor. An approach for reducing the power consumption in NUCA cache memories has been proposed in [11]. In such a proposal the key idea is to allow the ways within a cache to be accessed at different speeds and to place infrequently accessed data into the slow ways.

A lot of works have been done on the design of low power cache in embedded systems. These studies are consequence of the importance of the cache on processor performances, and the increased power consumption (both static and dynamic) of the memory subsystem [19], [20].

An important trend in the design of low power, high performance hardware consists in the partitioning of hardware components in smaller and less energy-consuming units [26]. This trend has been utilized in both the design of processor internal architectures [27] and cache memories [12]. Following such approach, Kim et al. [26] proposed sub-cache to reduce power consumption of L1 instruction cache in embedded system. In sub-cache architecture, a cache is split into several smaller units, each of which is a

cache by itself. Another approach to reduce power consumption consists in adding a small instruction cache (tiny cache). It has been proposed by Jouppi [29], in the field of general purpose system, and it has been applied to the design of instruction cache for embedded systems [28]. Similarly, filter cache has been utilized to minimize energy consumption of instruction cache [30]: the idea is that if most of a program's time is spent in loop, then most hits occur in the filter cache.

All these papers address the problem of reducing cache memories power consumption in embedded systems. Like our work, most of these solutions utilize sub-banking or similar issues (little cache and/or sub-cache), but they not deal with the wire delay problem, so, differently from our work, they will be no more applicable to the design of future high performance, low power embedded processor.

Other techniques have been developed, but in most cases, they incur in additional latencies, so they are not useful for the design of high performance embedded system. With the delayed access [26], [31], the reduction of power consumption is achieved by activating only the cache bank that will be accessed: the access to data array is delayed until the access to the tag array indicates the right way. Other schemas try to predict the way which is accessed [32], or try to change the number of ways activated depending on the application behaviors [33].

Also configurable cache architectures have been studied. The first proposals deal with performance, while more recent works consider also power consumption. Ranganathan et al. [35] proposed configurable cache architecture for general purpose processors. When used in media applications, a large cache may not give benefits due to the data characteristics of media applications. In this case, the authors propose of dynamically reconfigure part of the cache, to be used for other processor activities, such as instruction reuse. Kim et al. [36] proposed a multifunction cache architecture, which partitions the cache into a dedicated cache and a configurable cache. The configurable part can be used to implement computations, for example, FIR and DCT/IDCT, which takes advantage of on-chip resources when an application does not need the whole cache.

Zhang et al., propose specific hardware on-chip implementing cache tuning heuristic, with the aims of reducing power consumptions in embedded processors [20]. They propose also [34] a reconfigurable cache architecture, in which cache size (by shutting up or down ways), line size and associativity (via way concatenations) may be tuned to the application needs. Way shutdown cache methods have been proposed independently by Albonesi [37] and by the designers of the Motorola M*CORE processor [38]. In those approaches, a designer would initially profile a program to determine how many ways could be shut down without causing too much performance degradation. Albonesi also discusses dynamic way shutdown and activation for different regions of a program. As NUCA and TDNUCA cache are highly reconfigurable, we plan to extend our work on TD-NUCA by evaluating reconfigurable techniques. In particular, we can consider a TD-NUCA cache as a NUCA cache obtained by applying way shutdown techniques. Besides, both the design introduces another dimension in the design space: due to the NOC infrastructure, also the mapping and searching algorithm can be dynamically changed. In our paper, we want to explore the potential benefits of applying such techniques. In another work we plan to explore the feasibility and the cost of implementing reconfigurable techniques in D-NUCA caches.

## 4. Design of TD-NUCA caches

In order to completely define the TD-NUCA model, besides the size and the numbers of blocks, the definition of the mapping, search, movement and replacement policies is needed. We evaluated different solutions, and we report in this paper only the meaningfully results. We mainly refer to the increasing TD-NUCA organization in order to present our proposal, but the whole work can easily be extended to the decreasing solution.

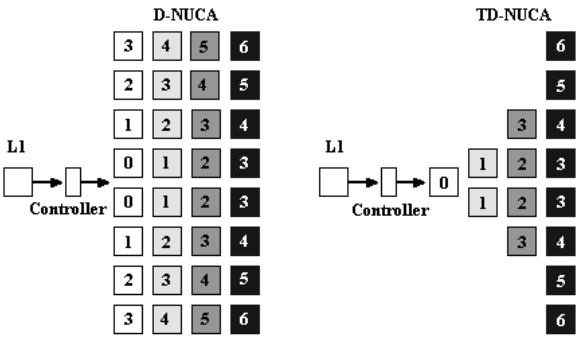


**Figure 3: Comparison between D-NUCA and TD-NUCA memories.**

With respect to the mapping policy, we adopt a spread-sets approach for the TD-NUCA organization. Differently from the original proposal [7], [8], in our case all of the bank sets share some banks in a similar way to the shared mapping proposed for rectangular D-NUCA caches. The figure 3 shows the comparison between four-way D-NUCA and four-way increasing TD-NUCA memories. The numbers superimposed on the cache banks show the hops (and the delays) needed to communicate with the controller.

Similarly to the mapping policies proposed in prior work, for TD-NUCA designs we adopt the simple mapping and fair mapping policies. The two different mapping policies are shown in the figure 4. In the simple mapping policy, a cache line of a generic bank can be mapped in the next way into two different banks1. The main drawback of this solution is that some memory addresses can be mapped only into the banks with high delay, whereas other memory addresses can be mapped only into banks with low delay. In the fair mapping policy, the two possible destinations for

[1]The sets are defined by all of the possible paths shown in figure, i.e. those that lead from the controller to the banks on the end column.

each cache line have the same delay, so that the average access times across all paths are equalized2.

We evaluated and adapted the incremental and multicast search policies. In the incremental search, the banks are searched in order starting from the closest bank until the requested line is found or a miss occurs in the last bank. According to this technique, a request is routed toward the first bank of the set, where, if no hit occurs, the request is routed through the shortest path toward the next bank of the set. This is repeated until a hit or a miss occurs. Unfortunately, this solution is effective only for the simple mapping. In fact, using the incremental search in conjunction with the fair mapping policy, in some cases the requests have to follow longer paths than the shortest ones, and the average access time to the banks is worse than the one in the rectangular D-NUCA. In order to overcome such a limitation, our final choice is to use the multicast search technique. In this case, all of the requests are routed in a middle channel from where they proceed in parallel across both the columns and the rows. Figure 5 shows an example of path followed by a request in both the search techniques, in the case of fair mapping policy.

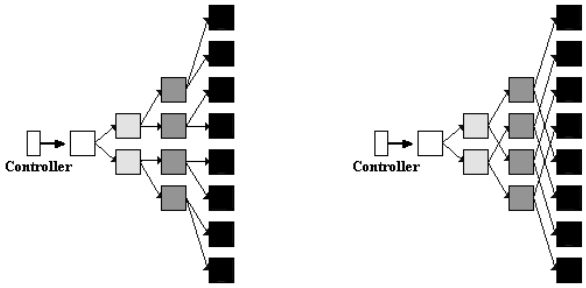


**Figure 4: The two different mapping policies.**

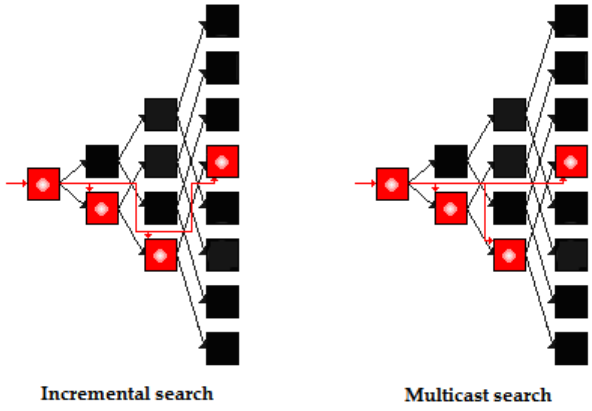


**Figure 5: Example of access with both incremental search and multicast search.**

With respect to movement policy, we employ the generation promotion technique proposed for rectangular D-NUCA caches. For the replacement policy, we consider two of the possible strategies proposed in the prior work, tail insertion in conjunction to a zero-copy policy and random insertion in conjunction to a one-copy policy.

[2] Also for this solution the sets are defined by all of the possible paths that lead from the controller to the banks on the end column.

## 5. Results

In order to evaluate our proposals, we use the same experimental methodology and some of the benchmarks proposed in [7, 8]. The SPECINT2000 benchmarks used for performance evaluation and the related parameters are listed in table 1. The parameters FFWD and RUN are respectively the number of instructions skipped to reach the phase start and the number of instructions simulated. Table 1 also lists the number of L2 accesses per 1 million instructions assuming 64KB level-1 instruction and data caches [6]. As done in the previous work, we assume a constant L2 cache area and vary the technology generation to scale cache capacity within that area, according to the SIA Roadmap [13] predictions. We analyze the same cases and use the same performance parameters (i.e. IPC and Miss Rate) of the work [7, 8], to get comparable results.

**Table 1: Benchmarks used for performance evaluation.**

| SPECINT2000 | Phase | | L2 load acc/ Million Instr. |
|---|---|---|---|
| | RUN | FFWD | |
| gcc | 2,367B | 300M | 25.900 |
| mcf | 5B | 200M | 260.620 |
| bzip2 | 744B | 1B | 9.300 |
| twolf | 511B | 200M | 22.500 |

We modified the extended sim-alpha simulator for supporting our TD-NUCA organization. In order to perform a meaningful comparison and validate the changes to the simulator, we repeated the simulations also for rectangular D-NUCA caches. The results that we obtained show that our TD-NUCA simulator is coherent to the previous D-NUCA simulator.

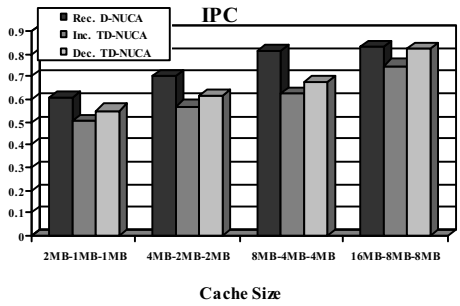


**Figure 6: IPC of rectangular D-NUCA and TD-NUCA caches.**

Our preliminary investigation uses a baseline configuration. Such a configuration uses fair mapping, multicast search, single bank promotion upon each hit and tail insertion. We explored different memory sizes, i.e. 1MB, 2MB, 4MB and 8MB comparing the results respectively with 16MB, 8MB, 4MB and 2MB rectangular D-NUCA caches. Figure 6 and Figure 7 summarize the simulation results obtained by averaging the different results

of all benchmarks. In particular, these figures show respectively the IPC and the miss rate for 2MB, 4MB, 8MB and 16MB D-NUCA caches and for the equivalent increasing and decreasing TD-NUCA caches (i.e. for 1MB, 2MB, 4MB and 8MB sizes).

The smallest gap between the IPCs (i.e. the best case) is achieved with 16MB D-NUCA and 8MB TD-NUCAs (i.e. 16-8-8 configuration). As in the previous work, the 16MB rectangular organization has 16x16 banks, whereas the 8MB TD-NUCAs have each four columns by two banks, four columns by four banks, four columns by eight banks and four columns by sixteen banks. First, we observe that the increasing solution has a lower IPC than the decreasing one. Since the miss rate is very low in both cases, such a result is due to the different average access times. The IPC of the decreasing cache is approximately that of the rectangular D-NUCA, whereas the increasing cache reduces the IPC by approximately 9%.

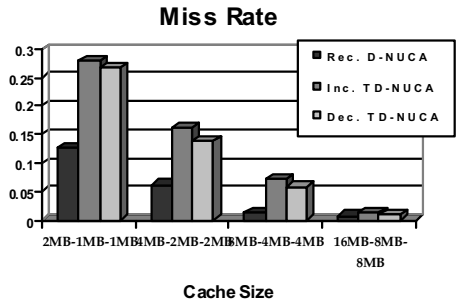


**Figure 7: Miss-rate of rectangular D-NUCA and TD-NUCA caches.**

The greatest gap between the IPCs (i.e. the worst case) is achieved with 8MB D-NUCA and 4MB TD-NUCAs (i.e. 8-4-4 configuration). In such case, the rectangular organization has 16x8 banks, whereas the TD-NUCAs have each two columns by two banks, two columns by four banks, two columns by eight banks and two columns by sixteen banks. Also the decreasing cache introduces performance degradation reducing the IPC by approximately 16% with respect to the rectangular cache. However, the performances of the decreasing cache are higher than increasing one.The figures show that increasing caches are better than decreasing ones. Moreover, TD-NUCA caches have approximately the same IPC of D-NUCA caches with the same sizes.

The figures 8, 9, 10 and 11, present the comparison between the IPC of rectangular D-NUCA and TD-NUCA caches respectively for gcc, mcf, bzip2 and twolf benchmarks. These figures show that, in some cases, the increasing TD-NUCA design is more performing than the equivalent rectangular design. These cases are gcc benchmark with 4-2-2 configuration, twolf benchmark with 4-2-2, 8-4-4 and 16-8-8 configurations, bzip2 with 16-8-8 configuration. In particular, the architectures based on TD-NUCA cache present the highest IPC. This means that, if the design issue is the realization of the most performing system running only the gcc or the bzip2 application, the optimal solution is based on the TD-NUCA cache, with a cache configuration whose size is half of the most performing D-NUCA cache. Such results indicate that TD-NUCA cache can be utilized in application specific domains, i.e. in embedded system, in order to minimize silicon area, power consumption and maximize performance. As the main difference between D-NUCA cache and TD-NUCA cache lies in the mapping policies, i.e. the correspondence of banks and memory address, the results indicate that further improvement can be achieved by exploring different and ad hoc mapping policies, with a design-simulate-analyze methodology, which is typical of the design of cache memory in embedded system [12], [21], [22].

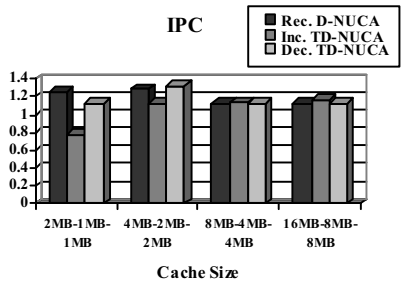


**Figure 8: IPC of rectangular D-NUCA and TD-NUCA caches when running the gcc benchmark.**

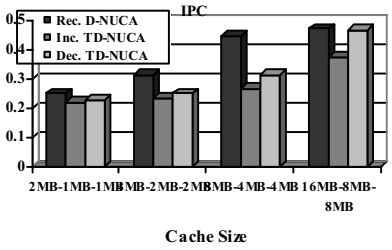


**Figure 9: IPC of rectangular D-NUCA and TD-NUCA caches when running the mcf benchmark.**

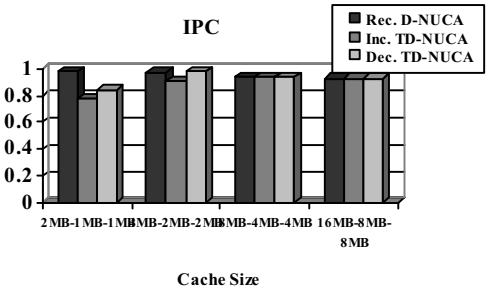


**Figure 10: IPC of rectangular D-NUCA and TD-NUCA caches when running the twolf benchmark.**

## 6. Conclusion

In this paper, we proposed a design targeting the optimization of NUCA cache memories, namely TD-NUCA caches. In particular, we deal with two different variants of TD-NUCA organization, increasing and decreasing TD-NUCA caches. The results of our investigation show that

decreasing TD-NUCA caches allow a reduction of the silicon area approximately by 50% without heavy performance degradation. We showed also that the triangular design outperforms the equivalent rectangular design in some specific domain applications. Such a result enables to use TD-NUCA caches in application specific and embedded domains. We believe that with efforts on the mapping and search policies, the TD-NUCA design can be further improved. The support of the emerging Network on Chip infrastructures will become the key technology to improve TD-NUCA caches. A TD-NUCA cache may be viewed as a subset of a D-NUCA cache, with bank interconnected via a NoC. Such infrastructure permits the reconfiguration of mapping policies, routing and number of active banks, to adapt the cache architecture to the specific application.

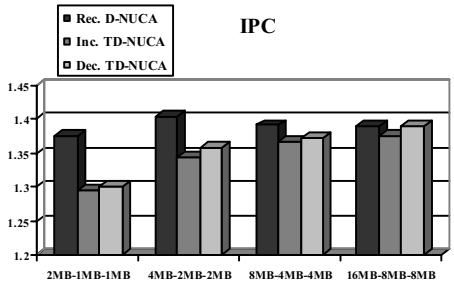


**Figure 11: IPC of rectangular D-NUCA and TD-NUCA caches when running the *bzip2* benchmark.**

## 7. Acknowledgement

This work has been supported by the Italian MIUR, under the FIRB Project "Methodology to Design High Performance Processor and Memory Architectures" and by the EC Hipeac Network of Excellence. Stephen Keckler furnishes us the Sim-Alpha and modified Sim-Alpha simulators.

## 8. References

[1] V. Agarwal, M. S. Hrishikesh, S.W. Keckler, and D. Burger. Clock rate vs. IPC: The end of the road for conventional microprocessors. Proc. 27th Annual Int. Symp. on Computer Architecture, pages 248–259, June 2000.

[2] B.M. Beckmann, and D.A. Wood. Managing Wire-Delay in Large Chip Multi-Processor Caches. Proc. 37th Int. Symp. on Microarchitecture, 2004.

[3] R. Desikan, D. Burger, S.W. Keckler, and T.M. Austin. Simalpha: A validated execution-driven alpha 21264 simulator. Technical Report TR-01-23, Dept. of Computer Sciences, University of Texas at Austin, 2001.

[4] E.G. Hallnor and S.K. Reinhardt. A fully associative software-managed cache design. 27th Symp. on Comp. Architecture, pages 107–116, June 2000.

[5] R.E. Kessler. Analysis of Multi-Megabyte Secondary CPU Cache Memories. PhD thesis, Univ. of Wisconsin-Madison, Dec. 1989.

[6] R.E. Kessler, M.D. Hill, and D.A. Wood. A comparison of trace-sampling techniques for multi-megabyte caches. IEEE Trans. on Computers, 43(6): 664–675, June 1994.

[7] C. Kim, D. Burger, S. W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. Int. Conf. on Arch. Sup. for Prog. Lang. and Oper. Sys., pp. 211-222, October, 2002.

[8] C. Kim, D. Burger, S. W. Keckler. Nonuniform cache architectures for wire-delay dominated on-chip caches. IEEE Micro, 23:6, pp. 99-107, November/December, 2003.

[9] A. Kodama, T. Sato. A Non-Uniform Cache Architecture on Networks-on-Chip: A Fully Associative Approach with Pre-Promotion. 10th Int. Symp. on Integrated Circuits, Devices and Systems, September 2004.

[10] D. Patterson and J. Hennessy. Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publishers, San Mateo, CA, 3rd edition, 2002.

[11] A. Sakanaka, T. Sato. A Leakage-Energy-Reduction Technique for High-Associativity Caches in Embedded Systems. MEDEA Workshop, pp.51-56, New Orleans (LU), September 2003.

[12] S. Wilton and N. Jouppi. Cacti: An enhanced cache access and cycle time model. IEEE Journal of Solid-State Circuits, 31(5):677–688, May 1996.

[13] The national technology roadmap for semiconductors. Semiconductor Industry Association, 1999.

[14] R. Otten, P. Stravers. Challenges in Phisical Chip Design. Proc. Int. Conf. on Computer Aided Design, San Jos´e, CA, pp. 84–91, November 2000.

[15] P. R. Groeneveld. Physical Design Challenges for Billion Transistor Chips. Int. Conf. on Computer Design: VLSI in Computers and Processors, Freiburg, Germany, pp. 78-83, Sept. 2002.

[16] M. Schlett. Trends in Embedded-Microprocessor Design. IEEE Computer, Vol. 31, N. 8, pp. 44-49, August 1998.

[17] B. Mathew, A. Davis, M. Parker. A Low Power Architecture for Embedded Perception. Proc. Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems, Washington D.C., pp. 46-56, Sept, 2004

[18] A. Gosh, T. Girvagis. Cache Optimization for Embedded Processor Cores: An Analytical Approach. ACM Trans. on Design Automation of Electronic Systems. Vol. 9, Issue 4, Pages: 419 – 440, October 2004.

[19] C. Zhang, F. Vahid, W. Naijar. A highly configurable cache architecture for embedded systems. Int. Symp. on Comp. Arch.. San Diego, CA, pp 136-146, June 2003.

[20] C. Zhang, F. Vahid, W. Naijar. A Highly Configurable Cache Architecture for Low Energy Embedded Systems. ACM Trans. on Emb. Computing Systems, Vol. 4, N. 2, pp. 363–387, May 2005.

[21] T. Sato. Evaluating Trace Cache on Moderate-Scale Processors. IEEE Computer, vol. 147, no. 6, 2000.

[22] A. Ghosh, T. Givargis. Cache Optimization for Embedded Processor Cores: An Analytical Approach. ACM Trans. on Design Automation of Electronic Systems, Vol. 9, N. 4, pp: 419 – 440, October 2004

[23] L. Benini and G. D. Micheli. Powering networks on chips. Proc. Int. System Synthesis Symp., pp 33–38, 2001.

[24] W. J. Dally and B. Towles. Route packets, not wires: On chip interconnection networks. Proc. D. A. C., pages 684–689, 2001.

[25] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In Proc. Design Automation Conference, pp. 667–672, 2001.

[26] S. Kim, N. Vijaykrishnan, M. Kandemir, A. Sivasubramaniam and M. J. Irwin. Partitioned instruction cache architecture for energy efficiency. ACM Trans. on Embedded Computing Systems, vol. 2, n. 2, 2003.

[27] J. Cruz, A. González, M. Valero, N. P. Topham. Multiple-Banked Register File Architectures. Proc. of 27th. Ann. Int. Symp. on Computer Architecture, Vancouver (Canada), June 12-14, 2000, pages 316-325.

[28] A. Gordon-Ross, S. Cotterell, F. Vahid. Tiny Instruction Caches for Low Power Embedded Systems. ACM Trans. on Embedded Computing System, vol. 2, n. 4, pp. 449-491, 2003.

[29] N. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. Proc. 17th Int. Symp. on Computer Architecture, pages 364--373, Seattle, Washington, May 1990.

[30] J. Kin, M. Gupta, W. MangioneSmith. The Filter Cache: An Energy Efficient Memory Structure. Symp. on Microarchitecture, pp. 184-193, 1997.

[31] A. P. Chandrakasan, W. J. Bowhill, F. Fox. Design of High-Performance Microprocessor Circuits. Wiley-IEEE Press, 2000.

[32] K. Inoue, T. Ishihara, K. Murakami. Way-predicting set-associative cache for high performance and low energy consumption. Proc. Int. Symp. on Low Power Electronics and Design, pp. 273-275, San Diego, CA, 1999.

[33] M. D. Powell, A. Agarwal, T. N. Vijaykumar, B. Falsafi, K. Roy. Reducing set-associative cache energy via way-prediction and selective direct-mapping. 34th Int. Symp. on Microarchitecture, pp. 54-65. Austin, TX, 2001.

[34] C. Zhang, F. Vahid, R. Lysecky. A self-tuning cache architecture for embedded systems. ACM Trans. on Emb. Comp. Systems, vol. 3, n. 2, 2004.

[35] P. Ranganathan, S. Adve, N. Jouppi. Reconfigurable caches and their application to media processing. Proc. 27th Annual Int. Symp. on Computer Architecture, pp. 214—224, Vancouver, Canada, 2000.

[36] H. Kim, A. K. Somani, A. Tyagi. A reconfigurable multi-function computing cache architecture. IEEE Trans. on VLSI Systems. Vol. 9, N. 4, PP. 509-523, August 2001.

[37] D. H. Albonesi. Selective cache ways: on-demand cache resource allocation. 32nd Int. Symp. on MicroA., pp. 248-259, Haifa, Israel, Nov. 1999.

[38] A. Malik, B. Moyer, D. Cermak. A low power unified cache architecture providing power and performance flexibility. Proc. 2000 Int. Symp. on Low Power Electronics and Design, pp-241-243 , Rapallo, Italy, July 2000.