A Survey of Dual Data Cache Systems

Zivojin Sustran Sasa Stojanovic Goran Rakocevic* V.M. Milutinovic Mateo Valero**

Department of Computer Engineering
School of Electrical Engineering
University of Belgrade
11000, Belgrade, Serbia
{zika, stojsasa, vm}@etf.bg.ac.rs

(*)Mathematical Institute of the Serbian Academy of Sciences and Arts
11000, Belgrade, Serbia
goran.rakocevic@gmail.com

(**)Departamento de Arquitectura de Computadores
Universitat Politècnica de Catalunya
08034, Barcelona, Spain
mateo@ac.upc.edu

***Abstract*-Dual data cache (DDC) systems have attracted considerable research effort in the past decade or so, with their Divide et Impera tactic. DDC systems divide data according to their access patterns and use different caching strategies on them. In the first part of this paper, one possible classification taxonomy, is proposed and described. The second part of this paper represents a survey of the existing solutions classified according to proposed criteria, presenting their organization, benefits, shortcomings, and intended use.**

INTRODUCTION

The disparity between processor and main memory performance continues to grow and as a result there have been proposed many techniques for hiding the latency of memory accesses.

A large group of proposed solutions is based upon the dual data cache (DDC) system [1][2][12], which tries to take advantage of different patterns in data accesses. A DDC system caches the data into two physically and/or logically separated cache subsystems. The data with similar access patterns are stored in the same cache subsystem and an organization of each cache subsystem is optimized for the corresponding type of a data access pattern. Partitioning the data can be done at run time and/or compile time, and the system can be implemented with or without possibility to detect changes in the data access patterns. The DDC system reduces the hit miss ratio of the classical cache system, while occupying less silicon die area and consuming less power. If the data access pattern is predictable, the DDC system can also reduce latency of the read hit by intelligently prefetching data.

The data pattern access can be determined based on the type of locality that the data exhibit, the utilized data structure, the segment of the main memory where the data is stored, etc. Two types of localities, spatial and temporal, are important for determining the data access pattern. If a particular memory location is referenced at a particular time and it is likely that a nearby memory location will be referenced in the near future, the data item in that memory location exhibits the spatial locality. If a particular memory location is referenced at a particular time and there is high probability to access the same memory location in near future, the data in that memory location exhibits temporal locality. In determining the data access pattern, it is useful to know the data structure: scalar, vector, and complex. Access to scalar data is similar to access to data that exhibit temporal locality (particularly if the accesses happen in a loop), access to vector data is similar to access to the data that express spatial locality, and access to complex data cannot be predicted easily. Several other terms, important for determining the data access pattern, are presented in Table I.

This survey is mainly focused on comparing performance evaluation of different solutions, because of indisputable importance of cache performance. In modern systems, cache memory occupies roughly one third of silicon die area, and we try to present a possibility for reduction of the transistor count in DDC. One viewpoint, which is especially important in embedded and mobile systems, is concentrated on DDC power consumption and energy dissipation.

The goal of this paper is to give a comprehensive insight into the research area of DDC. It covers several different approaches for splitting data cache in order to hide latency delays. A taxonomy that defines possible classification criteria is proposed. Existing solutions are presented in the context of these criteria.

TABLE I
DEFINITION OF TERMS USED

Term	Definition
Locality prediction table	Locality prediction table is a history table with information about the most recently executed load/store instructions used to predict the type of locality of referenced data.
2D spatial locality	Data that exhibit this type of locality if they are stored in matrix data-type and there is a high probability that in near future memory access to the neighboring element will happen.
Neighboring and OBL algorithms	Algorithms used for prefetching data that exhibit 2D and spatial locality.
Java processor	Implementation of Java Virtual Machine on FPGA chip.

A PROPOSAL FOR CLASSIFICATION OF DUAL DATA CACHE SYSTEMS

In order to provide a wide and extensive view in the field of DDC, possible criteria for classification are proposed. Our choice of the classification criteria relies on the possibility to classify all existing systems into the appropriate non-overlapping subsets of systems.

The first criterion for classification is based on use type of the processor, for which the cache system is being designed. Types of use can be general or special-purpose. Under general are classified those solutions used for arbitrary types of applications and general-purpose computing, while under special-purpose are classified those solutions that produce better results for one type of application and/or embedded computing. The general-purpose systems have a higher demand for performance. The special-purpose systems a have higher demand for reducing power consumption and a better usage of die space.

The second criterion for classification is based on whether the processor, for which the cache system is being designed, will be a part of uniprocessor or multiprocessor system. The reduction of cache size, through the use of the DDC system, opens a possibility for more cores on a die in multiprocessor systems.

The third criterion for classification is based on where is placed the mechanism for determining the type of locality that data exhibit. There are two possibilities: compiler-assisted, if the type of locality is determined by use of compiler and/or profiler, and compiler-not-assisted, if the type of locality is determined in hardware or in another system layer. This criterion shows whether the DDC system can adapt to changes of data access patterns and how can the system handle data that do not follow the expected access pattern.

The classification criteria were chosen to reflect the essence of the basic viewpoint of this research. The classification tree was obtained by successive application of the chosen criteria and it is presented in Figure 1. The leaves of the classification tree are the examples (research efforts) elaborated briefly later on, in the Existing Solutions section of this paper.

EXISTING DUAL DATA CACHE SYSTEMS

For each class of DDC systems we present the existing implementations, if any.

General Uniprocessor Compiler-Not-Assisted

The general uniprocessor compiler-not-assisted (GUN) is a class of proposed solutions where data localities and caching strategies are determined solely in hardware and where the help of a compiler is not necessary. We present two solutions only for this class, because they were published roughly at the same time and they had an impact on every other solution.

1. The Dual Data Cache

Gonzalez, Aliagas, and Valero proposed in [1][12] a novel data cache design for superscalar processors, named dual data cache (DDC), at the Universitat Politècnica de Catalunya, in order to resolve four main issues regarding data cache design: large working sets, pollution due to non-unit strides, interferences when the stride and the number of sets are not co-prime, and prefetching. The DDC is a data cache partitioned into two independent cache subsystems: one is designed to exploit both spatial and temporal locality (named the Spatial Cache) and the other one is designed to exploit temporal locality (named the Temporal Cache), as depicted in Figure 2. The sub-cache system where the missed data is cached or whether not to cache the missed data, for every cache miss, is determined using information contained in the locality prediction table (LPT); if the information for a particular instruction has not reached a "stable state" in the LPT, the data will be placed in the default sub-cache system. The authors have evaluated the DDC system with three different types of benchmarks and the results show better performance of the DDC system in comparison with the conventional cache system of the same size, for benchmarks that have memory references that exhibit both types of locality. The proposed solution does not reduce power consumption or transistor count of the conventional cache system and the use of cache memory space can be reduced by not allowing duplication of data in both temporal and spatial sub-cache systems.

2. The Split Temporal/Spatial Data Cache

Milutinovic, Markovic, Tomasevic, and Tremblay in [2] present the split temporal/spatial (STS) cache system for effective utilization of different types of locality in data

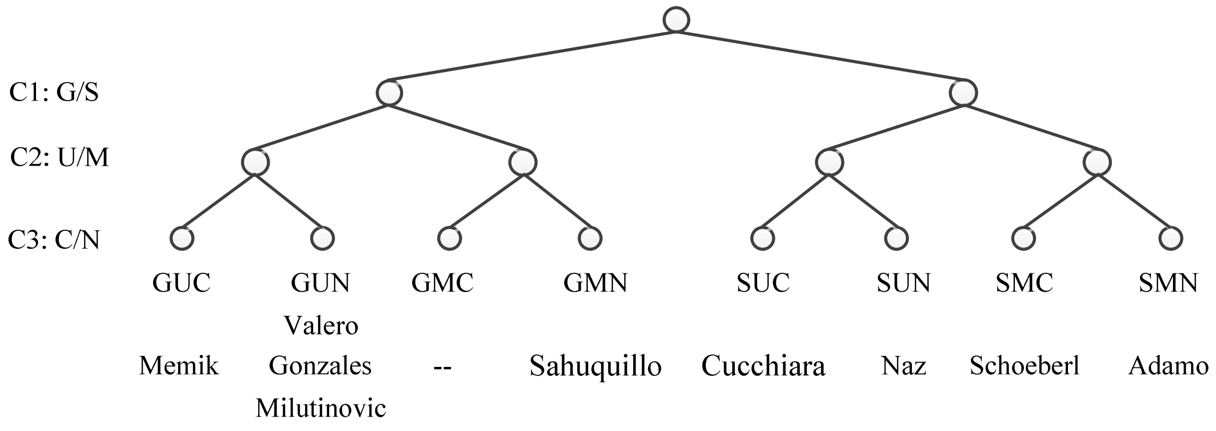

Figure 1: The classification three of Dual Data Cache systems. ***Legend:*** *G/S – general vs. special purpose; U/M – uniprocessor vs. multiprocessor; C/N - compiler assisted vs. hardware; GUC, GUN, GMC, GMN, SUC, SUN, SMC, SMN – abbreviation for eight classes of DDC.* ***Description:*** *The classification tree obtained by successive application of the chosen criteria.* ***Implication:*** *The class of general uniprocessor compiler assisted DDC system does not have known implementations.*

accesses; the research was done in the School of Electrical Engineering at the University of Belgrade with the intention of performance evaluation of the STS cache system, comparatively with the conventional cache system. Similar to the previously described cache system (DDC), the STS cache system, illustrated in Figure 3, is divided into spatial and temporal sub-cache systems, but instead of a large second level cache used for the both sub-cache systems, a two times smaller second level cache is used as the second level cache for the temporal sub-cache system, while the spatial sub-cache system has only one level. The sub-cache system where the missed data is cached, for every cache miss, is determined by a run-time algorithm, implemented in hardware, which dynamically tags/retags block of data against their locality; a compile time algorithm, which tags blocks of data based on the data structure, and a profile-time algorithm, similar to the run-time algorithm, are used to minimize the effect of a "cold start" period. The performance evaluation was done with ready-to-use traces, without the compile-time algorithm, and the results show considerable performance gain over the conventional cache system and a similar cache hit ratio; the same authors in [3] show that this approach is also able to reduce the complexity (die area occupied by the design) and consequently power consumption. The proposed cache system can be improved by adding a mechanism for detecting and bypassing data that do not express any type of locality and a mechanism for better utilization of the vector data that has non-unit strides.

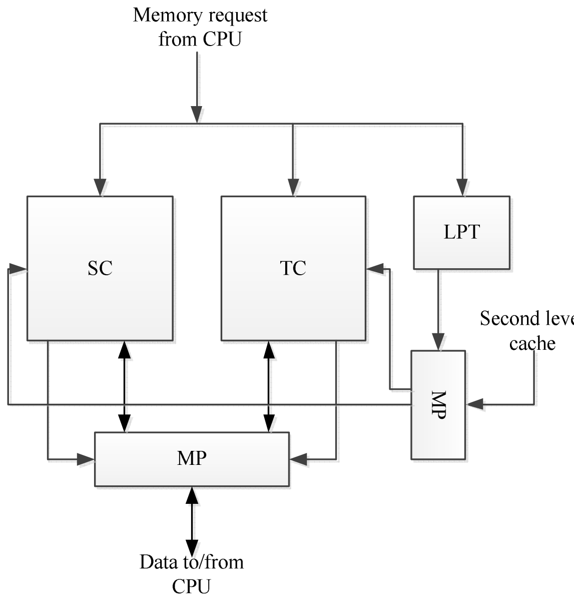

Figure 2: The Dual Data Cache system. ***Legend:*** *CPU – central processing unit; SC – spatial sub-cache; TC - temporal sub-cache; LPT – locality prediction table.* ***Description:*** *The cache organization is divided into two sub-cache systems: the temporal and the spatial sub-cache system.* ***Explanation:*** *The cache system is split into a "temporal" sub-cache system and "spatial" sub-cache system to allow use of different caching strategies for the different types of locality that data exhibit. The data block is cached in different sub-caches system based on an associated locality in the LPT. The locality for each data block is determined by previous accesses to the data block.* ***Implication:*** *The same data block can be cached in both sub-cache systems.*

General Uniprocessor Compiler-Assisted

The general uniprocessor compiler-assisted (GUC) is a class of proposed solutions where data localities and caching strategies are determined in software and hardware. Help of the compiler is necessary for solutions in this class.

3. The Northwestern Solution

Memik, Kandemir, Haldar, and Choudhary presented in [4] the Northwestern solution (NS) for improving cache performance using compiler and hardware techniques; the research was performed at the Northwestern University with the idea to investigate interaction between hardware and software techniques to optimizes data locality. The NS uses the compiler techniques for code areas that have regular data access pattern and uses dedicated instructions to selectively turn on/off the hardware base technique for code areas with irregular data access pattern. The cache system consists of conventional cache system, for caching data that do not

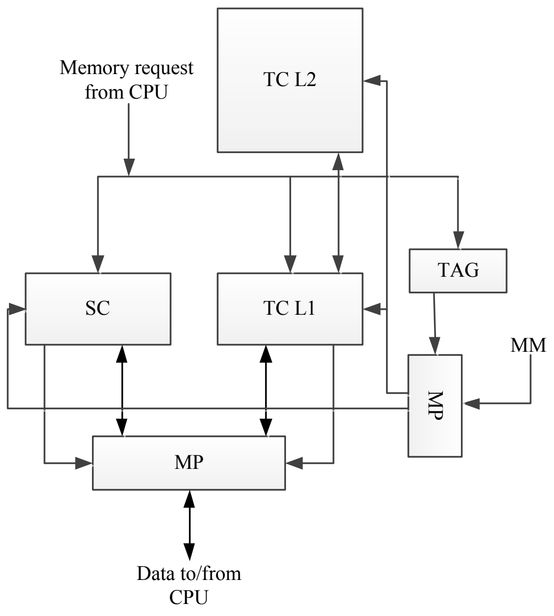

Figure 3: The Split Temporal Spatial cache system. ***Legend:*** *MM – main memory; CPU – central processing unit; SC – spatial sub-cache with prefetching mechanism; TC L1 and TC L2– the first and second level of the temporal sub-cache; TAG – unit for dynamic tagging/retagging data.* ***Description:*** *The cache organization is divided into two sub-cache systems: the temporal and the spatial sub-cache system. The temporal sub-cache system has two levels with one-word block size, while the spatial sub-cache system has only one level with the usual block size and a hardware implemented prefetching mechanism.* ***Explanation:*** *The cache system is split into a "temporal" sub-cache system and "spatial" sub-cache system to allow use of different caching strategies for the different types of locality which data exhibit. The data block is cached in different sub-caches system based on an associated tag. The tag for each data block is determined by previous accesses to the data block.* ***Implication:*** *The data block only can change sub-cache system if cache miss for that data block occurs.*

exhibit spatial locality and have high access frequency, and the small buffer, for caching data that exhibit spatial locality, as illustrated in Figure 5. Data that do not exhibit spatial locality and do not have high access frequency is not cached. Compiler techniques are affine loop and data transformation, used to optimize temporal and spatial locality aggressively. When compiler techniques are used on the data, the data is cached as in a conventional cache system. The simulation results confirm that the NS have better performance compared to systems with pure-hardware, pure-software, or combined hardware/software non-selective techniques for optimizing data locality, while using the same die area and power consumption.

General Multiprocessor Compiler-Not-Assisted

The general multiprocessor compiler-not-assisted (GMN) is a class where data localities and caching strategies are determined solely in hardware and where the help of a compiler is not necessary. The processor for which cache system is created will be used in multiprocessor environment.

4. The Split Data Cache in Multiprocessor System

Sahuquillo and Pont in [5] proposed extension of the STS cache system, presented in [2], in order to adapt it for the use in shared multiprocessor environment; the research was conducted in the School of Electrical Engineering at the University of Belgrade with the intention of studying the advantages over the conventional cache systems. The cache system is basically the same as the STS system with a couple of differences; a tag for the data is kept until the data eviction and when the tag for data is changed, the data is relocated to another sub-cache system. A snoop controller is added to the both sub-cache systems, as depicted in Figure 4. The extension of the Berkeley cache coherence protocol is used in conjunction to the STS cache system. When a hit occurs, the sub-cache system sends signal to another sub-cache system to stop it from accessing the bus and supplies the processor with the requested data. When a miss occurs, the spatial sub-cache system requests the data on the bus and the requested data is allocated in the spatial sub-cache system. The both sub-cache systems snoop the bus for invalidation signal. Simulation shows that the STS system in multiprocessor environment can achieve the same performance as the conventional cache, while occupying smaller space on the die and using less power. Enabling tag history for the data being evicted can avoid "cold start" period when the data is being accessed after eviction, especially when eviction of shared block happens as

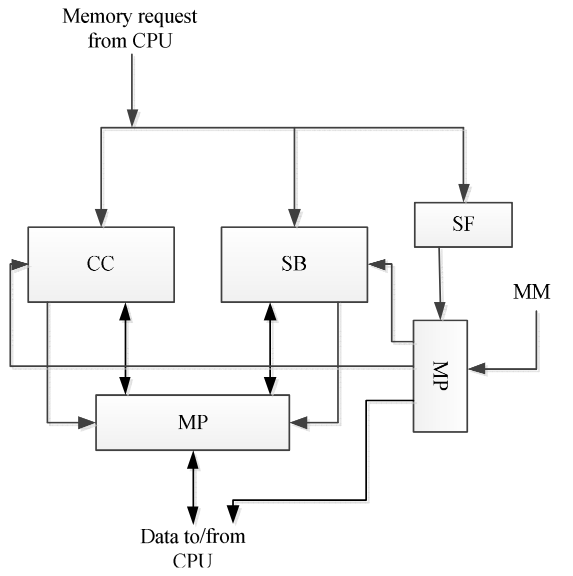

Figure 5: The Northwestern solution. Legend: CPU -- central processing unit, CC -- conventional cache, SB -- small FIFO buffer, SF -- unit for detection of data frequency access and if data exhibit spatial locality , MM -- main memory, MP -- multiplexer. ***Description:*** *The cache system consists of the conventional cache and small FIFO buffer. The SF unit controls where data is fetched.* ***Explanation:*** *The CPU can turn on/off the SF. If the SF is turned off software techniques for optimizing data locality are used and data are cached into the conventional cache. If the SF is turned on hardware technique for optimizing data locality is used and data are cached into the conventional cache or the small buffer if data has highly access frequency or they exhibit spatial locality.* ***Implication:*** *Dedicated instructions are necessary to turn on/off the SF.*

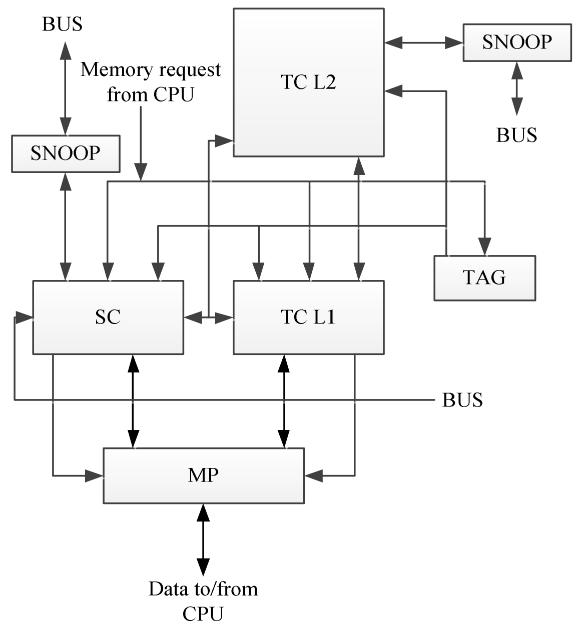

Figure 4: The Split Data Cache system in Multiprocessor system. ***Legend:*** *BUS – system bus; CPU – central processing unit; SC – spatial sub-cache with prefetching mechanism; TC L1 and TC L2– the first and second level of the temporal sub-cache; TAG – unit for dynamic tagging/retagging data; SNOOP – snoop controller for cache coherence protocol.* ***Description:*** *The cache organization is divided into two sub-cache systems: the temporal and the spatial sub-cache system. Each sub-cache system has the associated snoop controller.* ***Explanation:*** *The cache system is split into a "temporal" sub-cache system and "spatial" sub-cache system to allow use of different caching strategies for the different types of locality which data exhibit. The data block is cached in different sub-caches system based on an associated tag. The tag for each data block is determined by previous accesses to the data block that happened after last time the block is fetched.* ***Implication:*** *The data can change sub-cache systems in which are stored. The data is fetched only by the spatial sub-cache system.*

a result of write operation by another processor.

General Multiprocessor Compiler-Assisted

The general multiprocessor compiler-assisted (GMC) is a class that does not include any existing implementations to the best of our knowledge. The GMC class makes full sense, so the GMC research avenue represents a potentially fruitful research target [11].

Special Uniprocessor Compiler-Not-Assisted

The special uniprocessor compiler-not-assisted (SUN) is a class of proposed solutions where data localities and caching strategies are determined solely in hardware and where the help of a compiler is not necessary. The cache systems belonging to this group are optimized for special-purpose applications.

5. *The Reconfigurable Split Data Cache*

Naz, Kavi, Oh, and Foglia in [6] present the reconfigurable split data cache (RSDC) architecture for embedded systems, in order to accomplish a better utilization of die area; the research was performed at the University of North Texas. The RSDC system detects spatial or temporal locality that data exhibit and fine-tunes cache policies according to that type. The cache is divided into the array cache, for exploiting spatial locality, the scalar cache, for exploiting temporal locality, and the victim cache, for lowering the associativity of the scalar cache, as depicted in Figure 6. Every sub-cache system is divided into multiple partitions, proposed in [7], which can be used for purposes other than conventional caching (instruction reuse, as lookup tables, prefetching, etc.), or can be turned off to reduce power consumption. Turning off some parts of cache system can result in 50% reduction in power consumption. Reconfiguration of the cache partitions is used in applications that have lower memory requirements and can benefit from specialized hardware for non-standard processor activity. Paper [7] concludes that implementing reconfigurable partitions requires only a small amount of additional logic and additional wiring, while the cache access time is increased by a relatively small percentage. Inducing data locality based on data-type can generate misclassified data, because the array data can exhibit temporal locality.

Special Uniprocessor Compiler-Assisted

The special uniprocessor compiler-assisted (SUC) is a class of proposed solutions where data localities and caching strategies are determined in software and hardware. The cache systems belonging to this group are optimized for special-purpose applications.

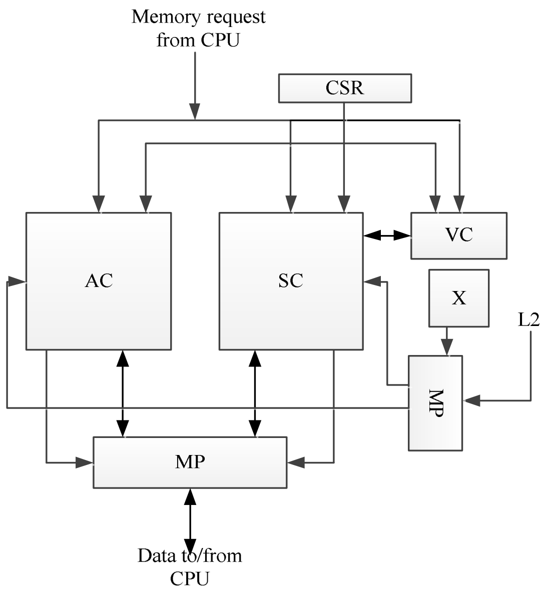

Figure 6: The Reconfigurable Split Data Cache. ***Legend:*** *AC – array cache, SC – scalar cache, VC – victim cache, CSR – cache status register, X – unit for determining data-type, L2 – second level cache, MP – multiplexer.* ***Description:*** *The cache is divided into three sub-caches systems: the array cache, for exploiting spatial locality, the scalar cache for exploring temporal locality, and the victim cache for lowering associativity of the scalar cache.* ***Explanation:*** *The unit X determines type of data that is being fetched and place data into a proper sub-cache system (scalar or array cache). Data is placed in the victim cache only when the block containing that data is evicted from the scalar cache. In the CSR is the information about which partition is used for conventional caching and which one is not.* ***Implication:*** *Checking which partition is used for conventional caching requires additional logic and wiring in sub-cache systems, and increases access time to sub-cache systems.*

6. *The Data-type Dependent Cache for MPEG Application*

Cucchiara, Prati, and Piccardi in [8] propose the data-type dependent cache for MPEG applications (DDM), for effective use of 2D spatial locality image data; the research was done at the Universita di Modena with the intention of achieving better performance for multimedia applications, than when using the conventional cache systems. The compiler classifies each memory reference as either addressing image or non-image data. Based on the class of data, different prefetching algorithms are used to cache data. Information about image data-type, address range, and row size, is stored in a dedicated memory table, using a special procedure call. This information is used every time, when memory reference occurs, for deciding which prefetching mechanism will be used. Authors proposed a new prefetching algorithm, called Neighbor prefetching, for exploiting 2D spatial locality and a standard One-Block-Lookahead (OBL) prefetching algorithm for exploiting spatial locality of the non-image data. In Figure 7 is illustrated the organization of the cache system. The DDM can be used for applications characterized by a substantial amount of image and video processing. The compiler classifies the data based on variable declarations generated by a programmer and this approach can miss-classify data, if the programmer uses a non-appropriate programming style. The compiler requires a specialized instruction set to access memory table. Inclusion of another sub-cache system for exploiting temporal locality of scalar data can potentially improve the overall performance of the DDM approach. The die space area and power consumption are not considered in [8].

Special Multiprocessor Compiler-Not-Assisted

The special multiprocessor compiler-not-assisted (SMN) is a class of proposed solutions where data localities and caching strategies are determined solely in hardware and where the help of a compiler is not necessary. The cache systems belonging to this group are optimized for special-purpose applications in multiprocessor environment.

7. The Texas Solution

Adamo et al. present in [9] similar solution to the RSDC for use in embedded multiprocessor systems, named Texas solution (TS) cache system; the research was performed at the University of North Texas. The data is divided based on type and placed in different sub-cache systems. The TS cache has two sub-cache systems: the array sub-cache system for storing array data and the scalar sub-cache system for storing scalar data, as illustrated in Figure 8. The small fully associative FIFO buffer is associated with the array sub-cache system to enable hardware based prefetching. When a miss occurs, the missed block is fetched into the array sub-cache system and also the next block is fetched into buffer to avoid cache pollution by displacing needed data in an untimely manner. Authors have showed that the TS cache can deliver same performance as a conventional cache while occupying less die area, making it good choice for first level cache in embedded multiprocessor systems on chip and leaving more space for processor cores. Inducing data locality based on data-type can generate misclassified data, because the array data can exhibit temporal locality. Determining the best cache coherence protocol is important, because the cache coherence protocol can greatly affect performance of cache system.

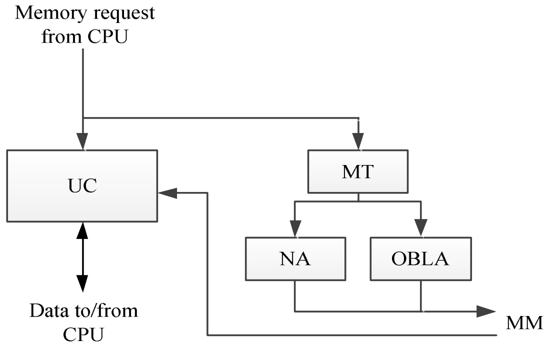

Figure 7: The data-type dependent cache for MPEG applications. ***Legend:*** *UC – unified data cache; MT – memory table for image information; NA – unit for prefetching data by the Neighbor algorithm; OBLA - unit for prefetching data by the OBL algorithm; MM – main memory.* ***Description:*** *The cache organization is not divided into sub-cache systems. Different types of locality exhibited by data are exploited by different prefetching units.* ***Explanation:*** *Image data is pre-fetch using NA, in order to exploit 2D spatial locality, and non-image data are pre-fetched using the OBLA unit, in order to exploit standard spatial locality. The MT unit contains information about the type of locality exhibited by data, stored by compiler.* ***Implication:*** *A specialized instruction set is required to change content of the MT.*

Special Multiprocessor Compiler-Assisted

The special multiprocessor compiler-assisted (SMC) is a class of proposed solutions where data localities and caching strategies are determined in software and hardware. The cache systems belonging to this group are optimized for special-purpose applications in multiprocessor environment.

8. The Time-Predictable Data Cache

Schoeberl, Puffitsch, and Huber proposed in [10] the time-predictable (TP) data cache for on-chip multiprocessor system, built from the Java processor (JOP) cores; the research was done at the Vienna University of Technology with intention of enabling the tight worst-case execution time analysis for real-time applications. The TP cache system stores scalar data in different sub-cache systems, based on data memory access type, while array data is being bypassed. The TP cache system is divided into two sub-cache systems: the fully associative sub-cache system with LRU replacement and a direct mapped cache, as depicted in Figure 9. The

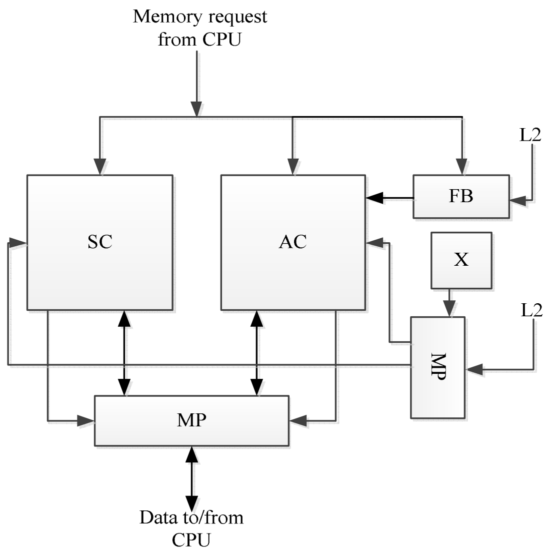

Figure 8: The Texas solution cache. ***Legend:*** *AC – array cache; SC – scalar cache; FB– FIFO buffer; X – unit for determining data-type; L2 – second level cache; MP – multiplexer.* ***Description:*** *The cache is divided into three sub-caches systems: the array cache, for exploiting spatial locality, the scalar cache for exploring temporal locality, and the FIFO buffer for array data prefetching.* ***Explanation:*** *The unit X determines type of data that is being fetched and place data into a proper sub-cache system (scalar or array cache). Data is placed in the FIFO buffer only when a miss occurs on the array data and the fetched data is in the next block.* ***Implication:*** *Because scalar data are not fetched in the array sub-cache system, the FIFO buffer is flushed less frequently and provides a decrease in the number of misses in the array sub-cache system.*

compiler divides data into several ways based on data memory access type. Constant and static data are cached in the direct mapped sub-cache system, while dynamic data (located on the heap) are cached in the associative cache. The authors argue that splitting data cache simplifies the cache coherence protocol and that reduces its limiting factor on multiprocessor system scalability, because it can detect shared data and enforce data invalidation only when it is truly necessary. Bypassing array data can have impact on performance, because spatial locality that array data exhibit is not exploited. The authors show that this approach is also able to reduce the complexity (die area occupied by the design) and consequently power consumption, compared to conventional approach.

CONCLUSION

Essentially, the purpose of this survey was to provide an extensive coverage of data access prediction patterns and utilization principles to hide memory access latency. A huge amount of research effort was spent to develop the cache systems based on the dual data cache approach. We tried to give a broad overview of the existing approaches, in term of applications for which the cache system is being designed. As processors continue to be used in an increasing number of application domains, it is important to ensure that cache systems that can use a significant fraction of the on-chip transistors are reduced in size as much as possible, while retaining a suitable performance. We tried to emphasize which solutions accomplish this goal. In embedded computing power consumption is also a limiting factor, and we specially pointed to designs with low power consumption.

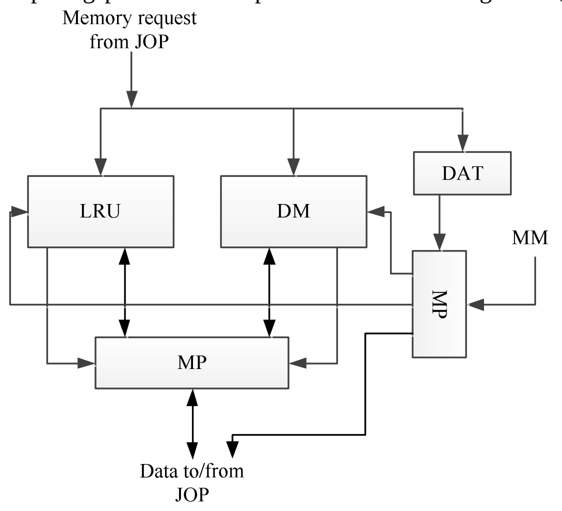

Figure 9: The Time-Predictable data cache. Legend: MM – main memory; JOP – Java processor; MP – multiplexer; LRU – fully associative sub-cache system with LRU replacement; DM – direct mapped sub-cache system; DAT – unit for determining data memory access type. ***Description:*** *Constant and static data are stored in the direct mapped sub-cache system. The LRU stores the object header and object fields. Array data is not cached.* ***Explanation:*** *The complier places different type of data into different memory areas. The DAT unit determines whether to cache data and where to cache data.* ***Implication:*** *This implementation allows a simplified cache coherence protocol that can invalidate data only when a write to shared data occurs.*

Proposed solutions for multiprocessor systems offer possibilities for reducing power consumption and the used die area compared to conventional cache systems. Some of them are uniprocessor solutions adapted for use in multiprocessor systems with a small change in cache coherence protocol used for conventional cache systems. The TA data cache is designed specifically for use in multiprocessor systems and even simplifies the cache coherence protocol and that reduces its limiting factor on multiprocessor system scalability

Large amounts of algorithms for determining data access pattern have been proposed. Some of them are simple, some are more complex, but all of them have more or less problems with dealing with some specific data access patterns. We feel that through selective combination, some of these algorithms, i.e., dual data cache systems, can achieve better results.

Beside the use of this survey for designing conventional computer systems, we believe that this survey can open new horizons for designing computer systems with a transactional memory. Especially the cost of abort procedures in systems with the transactional memory can be reduced using a dual data cache system.

REFERENCES

[1] A. Gonzalez, C. Aliagas, and M. Mateo, “Data cache with multiple caching strategies tuned to different types of locality,” *Proceedings International Conference on Supercomputing*, July 1995, pp. 338-347
[2] V. Milutinovic, M. Tomasevic, B. Markovic, M. Tremblay, “The split temporal/spatial cache: initial performance analysis,” *Proceedings of the SCIzzL-5*, Santa Clara, California, USA, March 1996, pp. 72-78.
[3] V. Milutinovic, M. Tomasevic, B. Markovic, M. Tremblay, “The split temporal/spatial cache: initial complexity analysis,” *Proceedings of the SCIzzL-6*, Santa Clara, California, USA, September 1996, pp. 89-96.
[4] G. Memik, M. Kandemir, M. Haldar, A. Choudhary, “A selective hardware/compiler approach for improving cache locality,” *Northwestern University Technical Report CPDC-TR-9909-016*, Evanston, Illinois, USA, 1999.
[5] J. Sahuquillo, A. Pont, “The split data cache in multiprocessor systems: an initial hit ratio analysis,” *Proceedings of the Seventh Euromicron Workshop on Parallel and Distributed Processing*, February 1999, pp. 27-34.
[6] A. Naz, K.M. Kavi, J. Oh, and P. Foglia, “Reconfigurable split data caches: a novel scheme for embedded systems,” *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*, March 2007, pp. 707-712.
[7] P. Ranganathan, S. Adve, N.P. Jouppi, “Reconfigurable caches and their application to media processing,” Proceedings of the 27th International Symposium on Computer Architecture, June 2000, pp. 214-224.
[8] R. Cucchiara, A. Prati, M. Piccardi, “Data-type dependent cache prefetching for MPEG applications,” *21st IEEE International Performance, Computing, and Communications Conference*, April 2002, pp. 115-122.
[9] O. Adamo, A. Naz, T. Janjusic, K.M. Kavi, and C. Chung, “Smaller split L-1 data caches for multi-core processing systems,” *Proceedings of ISPAN*, December 2009, pp.74-79.
[10] M. Schoeberl, W. Puffitsch, and B. Huber, “Towards time-predictable data caches for chip-multiprocessors,” *Proceedings of SEUS*, November 2009, pp. 180-191.
[11] Z. Sustran, “Comparing two multiprocessor oriented compiler-assisted approaches to general purpose processing: the case of abort in transactional memory system,” Technical Report of FP7 BalCon Project (Ph.D. Thesis in Preparation), Thessaloniki, Greece, September 2011.
[12] M. Valero, “The DDC cache,” Invited lecture at the UPC, Barcelona, Spain, December 1994.