Cooperative Cache Memory (CCM) based on the Performance Efficiency for 3D Stacked Memory System

Hongyeol Lim and Gi-Ho Park
Department of Computer Engineering, Sejong University
98, Gunja-Dong, Gwangjin-Gu, Seoul, 143-747, Republic of Korea
a) hylim@sju.ac.kr
b) ghpark@sejong.ac.kr, *corresponding author

Abstract: Three-dimensional (3D) memory stacking technology is one of the most promising technologies providing massive memory bandwidth, and thereby provides an opportunity to overcome performance bottlenecks in memory systems. In this paper, we analyze the performance of well-known efficient memory components such as stream buffer, L2 cache, and victim cache in order to exploit the massive memory bandwidth of the 3D stacked memory system. This paper also presents a method to determine the optimal memory capacity with the best performance efficiency (a.k.a. performance improvement efficiency (PIE)) and proposes a cooperative cache memory (CCM), which prefetches adaptively according to the memory access pattern in order to exploit the massive memory bandwidth of a 3D stacked memory system. The proposed CCM with a 512KB L2 cache and 256KB stream buffer with the proposed prefetching mechanism can deliver superior performance by about 13% over a conventional 8MB L2 cache.

Keywords: 3D stacked memory, data prefetching, performance efficiency, stream buffer, cache memory.

Classification: Integrated circuits (Logics)

References

[1] A. Jaleel et al., “High performance cache replacement using re-reference interval prediction (RRIP),” in Proceeding of the 37th annual Intl. Symposium on Computer Architecture, pp. 60-71, June, 2010.
[2] Carole-Jean Wu et al., “PACMan: prefetch-aware cache management for high performance caching,” Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 442-453, Dec., 2011.
[3] C. Liu et al., “Bridging the Processor-Memory Performance Gap with 3D IC Technology,” Design & Test Computers, pp. 556-564, 2005.
[4] T. Ono et al., “Adaptive Cache-Line Size Management on 3D Integrated Microprocessors,” Proceedings of the 24th International Symposium on Computer Architecture, 2009 International SoC Design Conference, pp. 472-475, 11, 2009.
[5] N.P. Jouppi, “Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffer,” Proceeding of
1 Introduction

Three-dimensional(3D) memory stacking technology is one of the most promising technologies capable of providing massive memory bandwidth between a last-level cache(LLC) and stacked memory using the through silicon vias(TSVs). This technology is expected to overcome the memory wall, a chronic issue in the traditional off-chip 2D memory system, by providing the high bandwidth from stacked memory.

On the other hand, the capacity of a last-level cache (LLC) integrated with the processor is increasing. Even though it has an important role to hide the long memory access latency in a memory system, it has been revealed in many studies that LLC becomes less efficient as capacity increases ([1], [2]). This issue in the 2D off-chip memory system will be still maintained even in the 3D-ICs based memory system.

To overcome this drawback, this paper investigates LLCs with large ca-
pacities from a new point of view in order to utilize the high memory bandwidth through redesigning the LLC, which is composed of a couple of efficient on-chip memory components. This paper considers a couple of well-known on-chip memory components such as an L2 cache, a stream buffer and a victim cache and presents a method to find out the optimal memory size to achieve the best performance efficiency by defining a simple analytical metric (a.k.a performance improvement efficiency (PIE)). In addition, this paper proposes a cooperative cache memory (CCM) constructed with an L2 cache and a stream buffer having the optimal size obtained from the proposed PIE model. An adaptive prefetching mechanism is applied to improve the performance of a stream buffer in the proposed CCM as well.

The rest of this paper is organized as follows. Section 2 explains several related works. The performance analysis of three candidate on-chip memories is discussed in Section 3. Section 4 proposes a cooperative cache memory (CCM) with an L2 cache and a stream buffer and presents a method to determine the optimal capacity of these memories. Section 5 summarizes this paper and presents potential future work.

2 Related Work

Various mechanisms have been proposed to exploit memory bandwidth and improve the cache performance([3] and [4]). The stream buffer proposed in [5] and [6] is one of the most efficient structures to exploit memory bandwidth and reduce memory access latency. In addition, a history-based prefetching mechanism using an indexing table and a correlation-based prefetching mechanism to predict prefetching blocks are proposed in [7] and [8], respectively.

Recently, [2] proposed an insertion policy of an incoming prefetching block in order to filter the useless prefetching blocks. Studies on the efficient cache structure and management mechanisms for 3D integration technology are also actively performed. This paper presents an efficient LLC memory structure and a prefetch mechanism that can utilize the advantage of 3D stacked memory system effectively.

3 Motivation: Performance analysis of various memory components

3.1 Simulation Methodology

A cycle-level x86-ISA based simulator for performance evaluation[9] with SPEC CPU 2000[10] and Mibench[11] benchmarks are used for performance analysis in this paper. The detailed system parameters for the simulation are presented in Table 1. The access latency of each cache memory is obtained from the CACTI 5.3[12] tool. The latency of the 2D off-chip memory is obtained from an actual x86 processor system by using LMbench tool[13]. The access latency of 3D stacked memory is obtained with the assumption for high bandwidth as used in [3] and [14]. The multi-way stream buffer is used and has 16-entries for each way as in [3].
3.2 Analysis of conventional approaches for L2 cache performance improvement in 3D stacked memory system

Increasing the cache block size and performing prefetching are the most straightforward approaches to improve the L2 cache performance based on the high bandwidth of 3D stacked memory. This paper analyzes the effect of these mechanisms on a 3D stacked memory system.

Figure 1 shows the performance of a 2MB L2 cache with various cache block sizes from 64B to 1KB. As shown in figure 1, increasing the cache block size cannot deliver meaningful performance improvements even in a 3D stacked system. When the cache block size becomes one KB, the performance of system decreases.

![Fig. 1. Performance of L2 cache memory according to the various cache block size on 3D stacked memory system.](image)

Prefetching is another straightforward mechanism to exploit the enormous memory bandwidth of 3D stacked technology to improve memory performance. Figure 2 presents the performance of a 2MB L2 cache memory on various fetch sizes that from 128B to 4KB. The 32 block fetch size means that 31 cache blocks are prefetched in addition to the demanded cache block upon a cache miss.

As shown in Figure 2, the aggressive prefetching of 32-blocks and 64-blocks degrades the system performance by about 10% and 18%, respectively.

---

**Table 1. Simulation parameters**

| Component                                    | L1 $ | L2 $         |
|----------------------------------------------|------|--------------|
| On-chip Cache                                |      |              |
| Capacity: 32KB+32KB, Block size: 64B,       |      |              |
| (I+D)Associativity: 4-way, Latency: 1 cycle  |      |              |
| Main Memory                                  |      |              |
| Capacity: 64KB ~ 8MB, Block size: 64B ~ 1KB,|      |              |
| (Unified)Associativity: 8-way, Latency: 5 ~ 71 cycle |      |
| Stream Buffer and Victim Cache               |      |              |
| Capacity: 256B ~ 8MB, Latency: 1 ~ 258 cycle, Block size: 64B, |      |
| Associativity: Fully-associativity (stream buffer constructed with multi-way) |      |              |

*The first block means the latency cycle which can be transferred of cache miss block from main memory to L2 cache, and the chunk block means the latency cycle which can be transferred of the next sequential blocks.*
This result means that while an appropriate amount of prefetching can be useful for performance improvement, excessive prefetching degrades the system performance due to the cache pollution even in a 3D stacked memory configuration.

3.3 Performance analysis of various memory structure in 3D stacked memory system.

Performance analysis for three well-known and efficient on-chip cache memory structures, i.e., a stream buffer, victim cache, and conventional L2 cache on the 2D and 3D memory system structure was performed. A stream buffer [3] was analyzed as a prefetch buffer because it is one of the most efficient memories as a dedicated prefetch memory to observe the performance impact of prefetching without the side effect such as a cache pollution.

Figure 3 illustrates the average CPI (cycle per instruction) of all benchmarks for these on-chip memory structures on the 2D memory and 3D stacked memory system.

In Figure 3, L2 cache memory can achieve a significant performance improvement by as much as 27% on average by increasing capacity on the 2D memory system from 64KB to 8MB. However, on a 3D stacked memory system, the conventional L2 cache structure cannot achieve performance improvements like a 2D memory system by increasing the L2 cache capacity. This is because a L2 cache miss penalty is significantly alleviated on
a 3D stacked memory system with the massive memory bandwidth. This implies that simply increasing the L2 cache capacity is not a cost-effective approach to achieve performance improvement of L2 cache memory in 3D stacked memory system.

On the other hand, a stream buffer can deliver meaningful performance improvements in both 2D and 3D memory systems when its capacity is increased up to 1MB. However, the performance of stream buffer is dramatically degraded when the capacity of it becomes larger than about 1MB due to the long access latency of a large fully associative stream buffer.

In the case of victim cache, it is expected as one of the most efficient memory structures that stores useful cache blocks that are evicted from the L2 cache. However, it cannot deliver meaningful performance improvements due to the long access latency of the fully-associative structure on both the 2D memory and 3D stacked memory system as shown in Figure 3. Based on this result, the victim cache will not be considered further as a candidate memory component in this paper.

Through this experiment, we found out that a conventional memory system with large capacity LLC could be a less promising approach to utilize the advantage of 3D stacked memory system. In addition, this paper has demonstrated that a stream buffer is one of the most efficient candidate memory components for exploiting the advantage of 3D stacked memory when has small size capacity. As a result, these memories revealed that the performance improvement does not linearly increased with the larger capacity due to the long access latency of the large memory components. Therefore, careful consideration for selecting a suitable capacity of a stream buffer and L2 cache is required to improve the performance with reasonable hardware cost.

4 Cooperative Cache Memory (CCM) based on the PIE

In this section, we introduce a simple analytical model (PIE) to find out the optimal capacity for an L2 cache and a stream buffer, as a candidate memory component. In addition, we propose a cooperative cache memory (CCM) constructed with an L2 cache and a stream buffer having the optimal capacity obtained from proposed PIE-based analysis. An adaptive prefetching mechanism that can improve the prefetching efficiency is introduced and applied to the proposed CCM as well in this section.

4.1 Configuration of the L2 cache and stream buffer of CCM based on PIE

As shown in Figure 3, the best performance for each memory component can be obtained with a 128KB stream buffer and 4MB L2 cache. However, it cannot deliver the best performance efficiency by just combining these memories. Alternatively, the performance improvement efficiency per additional capacity of memory component should be considered as an important metric. This paper defines a simple metric to obtain the best performance improvement efficiency (PIE) for each memory component as in equation (1).
\[
PIE = \frac{(CPI_\alpha - CPI_\beta)}{(Size_\beta - Size_\alpha)}
\] (1)

In equation (1), \(CPI_\alpha\) refers to the cycle per instruction(CPI) of a memory component with size of \(\alpha\) which is smaller than \(\beta\), and \(CPI_\beta\) means the CPI of a memory component with size of \(\beta\). As a result, this metric means that the performance improvement ratio per additional capacity of each memory component. From our PIE model, the memory configuration with a 512KB L2 cache and a 256KB stream buffer revealed that it has the best PIE.

4.2 Operation model of CCM having an L2 cache and stream buffer

Figure 4 illustrates the cooperative cache memory(CCM) constructed with an L2 cache and stream buffer.

![Image of cache architecture](image)

**Fig. 4.** Cache architecture of proposed CCM with an L2 cache and stream buffer.

Our proposed cache architecture has a conventional L1 cache and a lower-level cache system comprised of a 512KB L2 cache and a 256KB stream buffer, as shown in Figure 4. In the stream buffer, it has a counter bit to check whether the hit is the first hit or not, and it has an index table to maintain the reference of pair way in stream buffer. In addition, four ways in stream buffer are allocated in a dedicated way, for instruction and data accesses, i.e. two ways for instruction prefetching and two ways for data prefetching. Remaining ways are shared for both accesses.

A detailed operation model of our CCM is described in Algorithm 1. First, if a cache miss occurred in L1 cache then access\_CCM() searches the requested block in CCM(i.e. stream buffer and L2 cache). If a cache hit occurred in L2 cache, then the hit block is transferred to L1 cache memory. On the other hand, if a hit occurred in stream buffer and then a hit block transferred to both the L1 and L2 cache memory and stream buffer. It
Algorithm 1. Operation model of proposed mechanism

1. -If a L1 cache miss occurs then
2. -- hit_mem = access_CCM(L2 cache, stream buffer);
3. -- if (hit_mem == both memory) // a hit occurred in L2 cache and stream buffer
   ---- Transfer_cache_block(hit_block); // transfert a hit block into L1 cache.
   ---- Prefetching(LRU_way, miss_block, num_of_entry);
   -- endif
4. -- if (hit_mem == L2 cache)
   ---- Transfer_cache_block(hit_block); // transfer a hit block into L1 cache.
   -- endif
5. -- if (hit_mem == stream buffer)
   ---- Transfer_cache_block(hit_block); // transfer a hit block into L2 and L1 cache.
   ---- if (hit_count == 1) // hit is first time in the way.
      ------ pair_way = make_pair_way();
      ------ prefetching(pair_way, last_block_of_hit_way, num_of_entry);
   ---- elseif (hit_count >= 2)
      ------ prefetching(hit_way, last_block_of_hit_way, 1);
   ---- endif
   -- endif
6. -- if (hit_mem == NULL) //miss occurred in every memories.
   ---- Fetch (miss block); //fetch a miss block from the main memory.
   ---- Prefetching(LRU_way, miss_block, num_of_entry);
   -- endif
   - endif
7. - Prefetching(insert_way, prefetch_start_block, prefetch_entry)
   -- insert_way = LRU_way_in_stream_buffer;
   -- for (i = 0; i < prefetch_entry; i++) // in this paper, entry is set to 16.
      ---- prefetch_block_insert(insert_way, prefetching_start_block + next_sequential_block);
8. - make_pair_way()
   -- pair_way = LRU_way_in_stream_buffer;
   -- return insert_way = pair_way;

also initiates a prefetch request for the last entry in the hit way. Note that the basic operation model of adaptive prefetching mechanism with a stream buffer worked like [15].

Figure 5 illustrates the operation example in make_pair_way() of the proposed mechanism. As shown in Figure 5, the stream buffer checks whether the hit is the first hit or not. If the hit is identified as the first hit, then the pair way is selected from the shared ways to store additional prefetching blocks sequential to the last entry block of the hit way. Otherwise, if a hit is not the first hit, then a prefetching request is issued for the last entry without making a pair way. Note that an eviction way for making a pair way in the stream buffer is selected by the LRU (Least Recently Used) replacement policy. If an additional hit occurs in addition to the first hit in the pair way, then the prefetching blocks from the last entry of the hit way are inserted into the corresponding pair way to prevent the excessive expansion of the prefetch way. If a miss occurs in all memory components, then the requested block is fetched from the main memory into the L2 cache, and the sequential sixteen blocks from the requested block are fetched into the stream buffer. In this case, an eviction way in the stream buffer is also selected by the LRU policy.
Based on this mechanism, the proposed prefetching mechanism can alleviate the negative effect due to aggressively generated prefetch blocks by adjusting the amount of prefetching traffic according to the hit pattern in stream buffer, in addition it can utilize the massive memory bandwidth of the 3D stacked memory system.

1. First time hit occurred at "Way-1".
2. Make a pair way. (i.e. "Way-4")
3. Prefetch the sequential blocks at "Way-4" from "Way-1".

Ex 1) First time hit occurrence.

1. First time hit occurred at "Way-4".
2. Prefetch the sequential blocks at "Way-1" from "Way-4".

Ex 2) First time in paired way.

Fig. 5. Operation example of the paired way scheme of proposed mechanism in stream buffer.

4.3 Performance evaluation of CCM

Figure 6 presents the speedup of the CCM compared with other various memory configurations that are constructed with an L2 cache and a stream buffer without consideration of the performance efficiency (i.e. PIE).

Fig. 6. Performance of CCM compared with the various memory configurations. (The performance is normalized to a conventional 8MB L2 cache memory without a stream buffer.)
As shown in Figure 6, our CCM with a 512KB L2 cache and 256KB stream buffer can improve the performance by about 13% on average over a conventional 8MB L2 cache. The proposed CCM can also achieve a performance improvement up to about a 29% over a combined on-chip memory that was constructed with a 512KB L2 cache and 256KB stream buffer and can achieve superior performance by about 3% over a combined on-chip memory with a 4MB L2 cache and 128KB stream buffer that is constructed with the capacity delivering the best performance improvement for each memory component, as shown in Figure 3.

This result shows that allocating the capacity based on the PIE analysis is a good approach to select the optimal capacity for each memory component and the proposed adaptive prefetching scheme for the CCM can achieve a significant performance improvement as expected.

5 Conclusion

This paper analyzed the performance of various on-chip memory components that can utilize the massive memory bandwidth of a 3D stacked memory system. In addition, a new metric, i.e., performance improvement efficiency (PIE) is defined to discover the optimal capacity based on the performance efficiency analysis for each memory component. A cooperative cache memory (CCM) with an L2 cache and a stream buffer configured based on the PIE and an adaptive prefetching scheme are proposed to maximize the performance improvement for the stream buffer in this paper. Performance evaluation results show that the proposed CCM with the proposed prefetching mechanism with a 512KB L2 cache and a 265KB stream buffer can deliver a significant performance improvement by about a 13% than a conventional 8MB L2 cache. This paper focused on the performance efficiency of proposed memory on the 3D stacked memory. The investigation on the power efficient structure for our CCM on the 3D stacked memory system would be one of our future studies.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (NRF-2011-220-D00098). A preliminary version of this paper was published in IEEE International 3D Systems Integration Conference 2011.