Exploring High Bandwidth Pipelined Cache Architecture for Scaled Technology

Amit Agarwal, Kaushik Roy, and T. N. Vijaykumar
Electrical & Computer Engineering
Purdue University, West Lafayette, IN 47907, USA

Abstract

In this paper we propose a design technique to pipeline cache memories for high bandwidth applications. With the scaling of technology cache access latencies are multiple clock cycles. The proposed pipelined cache architecture can be accessed every clock cycle and thereby, enhances bandwidth and overall processor performance. The proposed architecture utilizes the idea of banking to reduce bit-line and word-line delay, making word-line to sense amplifier delay to fit into a single clock cycle. Experimental results show that optimal banking allows the cache to be split into multiple stages whose delays are equal to clock cycle time. The proposed design is fully scalable and can be applied to future technology generations. Power, delay and area estimates show that on average, the proposed pipelined cache improves MOPS (millions of operations per unit time per unit area per unit energy) by 40-50% compared to current cache architectures.

1: Introduction

The phenomenal increase in microprocessor performance places significant demands on the memory system. Computer architects are now exploring thread-level parallelism to exploit the continuing improvements in CMOS technology for higher performance. Simultaneous Multithreading (SMT) [7] has been proposed to improve system throughput by overlapping multiple (either multi-programmed or explicitly parallel) threads in a wide-issue processor. SMT enables several threads to be executed simultaneously on a single processor, placing a substantial bandwidth demand on the cache hierarchy, especially on L1 caches. SMT's performance is limited by the rate at which the L1 caches can supply data.

One way to build a high-bandwidth cache is to reduce the cache access time, which is determined by cache size and set-associativity [3]. To reduce the access time, the cache needs to be smaller in size and less set associative. However, decreasing the size or lowering the associativity has negative impact on the cache miss rate. The cache miss rate determines the amount of time the CPU is stalled waiting for data to be fetched from the main memory. Hence, there is a need for L1 cache design that is large and provides high bandwidth.

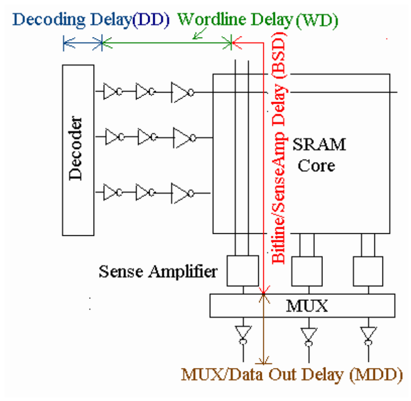
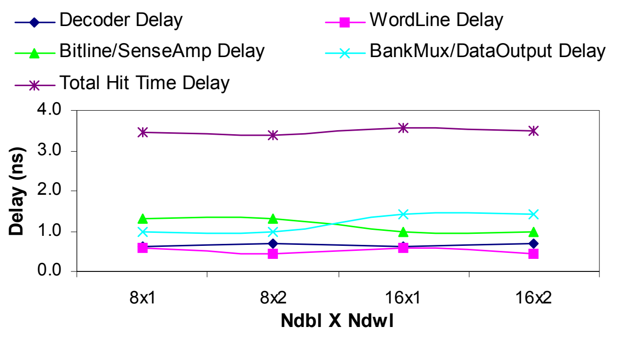


Fig. 1. Cache access delay.

The main issue with large cache is that the bit-line delay does not scale well with technology as compared to the clock cycle time [5]. Clock speed is getting doubled every technology generation making cache access latency to be more than one cycle. More than one cycle for cache access keeps the cache busy for those many cycles and no other memory operation can proceed without completing the current access. The instructions dependent on these memory operations also get stalled. This multi-cycle-access latency reduces the bandwidth of cache and hurts processor performance.

One of the techniques to increase the bandwidth is pipelining. Pipelining divides the cache latency into multiple stages so that multiple accesses can be made simultaneously. The problem in pipelining the cache is that data voltage level in the bit-lines remains at a fraction of supply voltage making it difficult to latch the bit-line data. We explore the idea of banking to make pipelining feasible in caches. We propose a scheme that can be utilized to divide the cache access latency into several stages as required by the clock cycle time and bandwidth requirement (as technology scales). The technique is also helpful in designing large-sized caches to decrease the miss rate while providing high bandwidth.



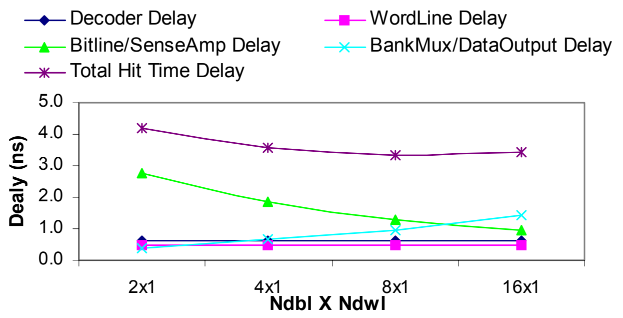


Fig. 2. Delay analysis for different Ndbl x Ndwl. a) 64K Cache, b) 128K Cache.

Decoder contributes around 15-20% to overall cache hit time delay. Nogami et.al [1] proposed a technique to hide the full or partial decoder delay by putting the decoder in a pipeline stage. However, even in the best case such a decoder-pipelined cache is limited by the word-line driver to data-out delay. Because this delay is 75-80% of the total cache delay, the decoder-pipelined cache has an imbalanced pipeline stage, which degrades the bandwidth.

2: Banking of cache

The cache access time can be divided into four parts (Fig. 1): (i) Decoding Delay (DD), (ii) Word-line Delay (WD), (iii) Bit-line to Sense Amplifier Delay (BSD), (iv) Mux to Data Out Delay (MDD). One of the main hit time reduction techniques in large caches is to bank the cache. The memory is partitioned into M smaller banks. An extra address word called bank address selects one of the M banks to be read or written. The technique reduces the word-line and bit-line (capacitance) and in turn reduces the WD and BSD. The bank address can be used to disable sense amplifiers and row and column decoders of un-accessed memory banks to reduce power dissipation.

The memory can be broken into smaller banks. Based on such banking, two parameters can be defined: N_{dwl} and N_{dbl} [2]. The parameter Ndwl indicates the number of times the array has been split with vertical cut lines (creating more, but shorter word-lines), while Ndbl indicates the number of times the array is split with horizontal cut lines (creating shorter bit-lines). The total number of banks is Ndwl x Ndbl. However, reduced hit time by increasing these parameters comes with extra area, energy and delay overhead. Increasing Ndbl increases the number of sense amplifiers, while increasing Ndwl translates into more word-line drivers and bigger decoder due to increase in the number of word-lines [4]. Most importantly, except in the case of a direct mapped cache (where Ndbl and Ndwl are both equal to one), a multiplexer is required to select the data from the appropriate bank. Increasing Ndbl increases the size of multiplexer, which in turn increases the critical hit time delay and energy consumption.

Fig. 2 shows the variation of different components of hit time with respect to Ndbl and Ndwl. A 64K and a 128K, 2-way caches were simulated using TSMC 0.25u technology. Increasing Ndbl decreases BSD due to shorter bit-lines but it also increases MDD due to larger number of banks. More banking requires large column multiplexer to select proper data. There is an optimum partitioning beyond which the increase in delay due to column multiplexer supercedes the delay gain due to reduced bit-lines. Increasing Ndwl does not make any difference in delay for 64K cache for current technology generations. In a 64k cache, a single word-line needs to have 2 cache blocks (64 byte) to maintain a good aspect ratio. Because the cache under consideration in Fig. 1 is a 2-way associative cache, which requires reading of two cache blocks simultaneously, changing Ndwl does not change the capacitance driven by word-line driver. For 128K cache, 4 cache blocks can be put in a single word-line. Dividing word-line into two parts decreases WDD, but increases DD due to the increased number of word-lines to be decoded. Several variations of the given architecture are possible. Variation includes positioning of sense amplifiers, the partitioning of the word and bit-lines, and the logic styles of the decoder used.

3: Proposed pipelined cache architecture

Pipelining is a popular design technique often used to increase the bandwidth of the datapaths. It reduces the cycle time of a block that may otherwise be governed by critical delay of the block. Unfortunately, unlike functional units, caches do not lend themselves to be pipelined to arbitrary depth. The key impediment to pipelining the cache into more stages is the bit-line delay cannot be pipelined because the signals on the bit-lines are weak, and not digital; latching can be done only after the sense amplifiers convert the bit-line signals from analog to digital. A bit-line is loaded by both the multiple word-line cells' capacitances, and the bit-line's wire capacitance and resistance. Consequently, the bit-line delay depends on the bank size, which is optimized for latency. In a latency-optimized cache, further dividing the cache (other than the decoder) into the two natural stages of bit-line + senseamps and multiplexor + data driver creates a substantially uneven split; the WD+BSD is considerably-a

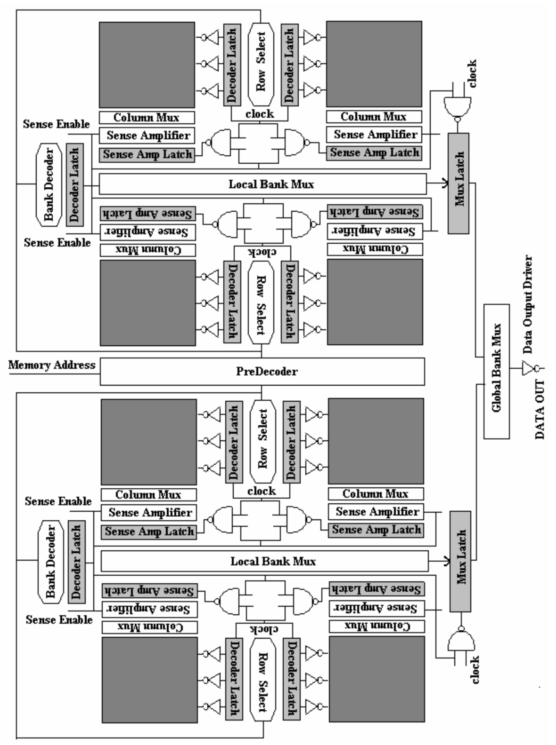


Fig. 3. Block diagram of a fully pipelined cache having bank configuration 4x2.

factor of two to four longer than the MDD step (Fig. 2). Thus, the uneven split renders ineffective pipelining a latency-optimized cache along such natural stages.

Partitioning the cache into multiple banks decreases BSD but increases MDD. Beyond a certain point more partitioning increases the multiplexer delay and it dominates the decrease in bit-line delay. It is interesting to observe that banking makes BDD delay to catch up the WD + BSD delay. Placing a latch in between divides the word-line to data-out delay into two approximately comparable sections. This technique has advantage of having more banks (low bit-line delay) than a conventional design permits. The increase in multiplexer delay due to aggressive banking is hidden in a pipeline stage. The clock cycle is governed by the word-line to sense amplifier delay (WD+BSD) and can be made smaller by aggressive banking. The technique can be used to aggressively bank the cache to get minimum possible WD+BSD which is limited by WD. For scaled technologies it might be necessary to bank the cache more so that BSD fits in a single clock cycle Aggressive banking makes MDD to go beyond the point where it can no longer fit into a single cycle. For dealing with this problem, multiplexer can be designed as a tree and can itself be pipelined into multiple stages as required.

Fig. 3 shows a block diagram of a fully pipelined cache. The cache is divided into three parts, a) Decoder, b) Word-line Driver to Sense Amplifier via Bit-lines, c) Bank Multiplexer to Data Output Drivers. Cache is banked optimally so that word-line driver to sense amplifier delay is comparable to a clock cycle time. Bank multiplexer is further divided into more pipeline stages as required by clock cycle time requirement.

3.1: Sense amplifier latches

Sense amplifiers are duplicated based on the number of horizontal cuts, Ndbl. A latch is placed at both input and output path of every sense amplifier to pipeline both read and write delay. Both input and output data is routed through bank multiplexer and data output drivers. The output of the bank decoder is also used to drive the sense enable signal of sense amplifiers. Only the accessed bank's sense amplifiers are activated and take part in sensing the data while the rest of them remain idle and do not increase dynamic energy. Increasing the number of banks increases the number of sense amplifier required, and in turn increases the number of latches. However, only one set of latches dissipates dynamic energy since switching occurs only on the datalines associated with accessed bank. Clock has to be routed to all the latches to synchronize the pipeline operation. To reduce the redundant clock power, clock gating was used. No extra control logic is required for clock gating. The outputs of bank decoder themselves act as control signals to gate the clock to un-accessed latches.

3.2: Decoder latches

All bank, row, and column decoders' outputs are also latched. Increasing the number of vertical cuts (Ndwl) increases the number of word-lines to decode. This requires a bigger row decoder and more number of latches. At the same time increasing the number of banks also increases the size of bank decoder and hence, the number of latches. However, at most only two lines switch in a single read/write operation. These two lines are: the one that got selected in previous cycle and the one that is going to be selected in current cycle. Hence, not all latches contribute to dynamic energy dissipation.

3.3: Multiplexer latches

To pipeline the multiplexer, sets of latches are placed at appropriate positions so that the multiplexer can be divided into multiple stages. The required number of latches depends on the aggressiveness of banking and how deep the latches are placed in the multiplexer tree. Similar to sense amplifier larches bank decoder output can be used to gate the clock in unused multilexer latches.

4: Results

In this section we evaluate the performance of conventional unpipelined cache with respect to technology

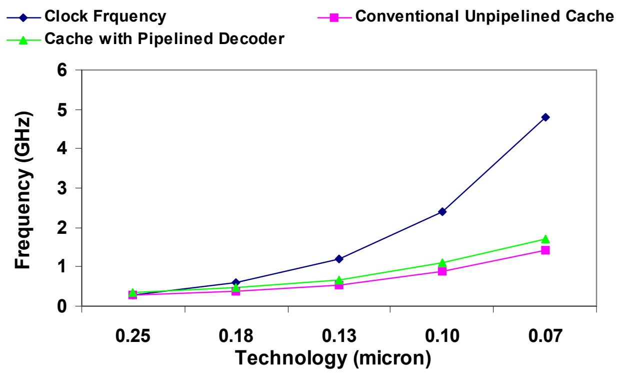


Fig. 4. Cache access freq vs. clock cycle freq.

scaling. We show the effectiveness of the proposed design in maintaining the pipeline stage delay within the clock cycle time bound. We extracted HSPICE netlist from the layout of a 64K cache using TSMC 0.25µ technology. We scaled down the netlist for 0.18, 0.13, 0.10 and 0.07µ technology and used BPTM [6] (Berkeley Predictive Technology Model) for simulating the cache for scaled technology.

4.1: Cache access latency vs CPU cycle

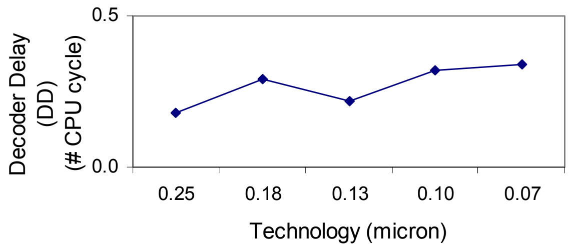
Current technology scaling trend shows that CPU clock frequency is getting doubled each generation. However, cache access latency does not scale that well because of long bit-lines. Fig. 4 shows the difference between clock frequency and conventional unpiplined cache access frequency for different technology generations. Here we assume that for 0.25µ technology the cache is accessed in one clock cycle. As technology is scaled clock cycle frequency is doubled. The cache is banked optimally to get minimum possible cache delay. Fig. 4 shows that as technology scales the gap between clock frequency and cache access frequency widens. Multiple clock cycles are required to access the cache (Table 1). Hence, cache cannot be accessed in every clock cycle. This reduces the bandwidth of cache and leads to processor stalls and loss in overall performance.

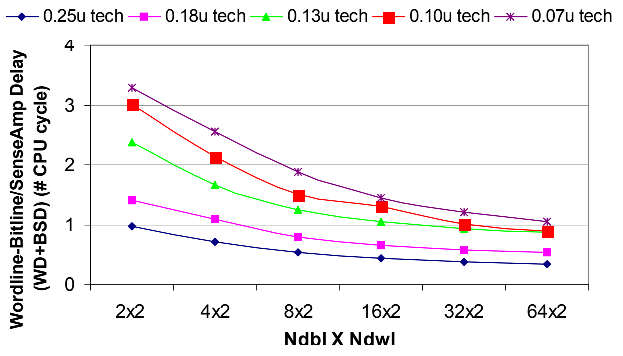
The third column of Table 1 shows the number of clock cycles required to access the cache with pipelined decoder [1]. For the design under consideration, the cache has two pipeline stages. First stage (decoder) is accessed in one clock cycle while second stage is accessed in the number of cycles given in Table 1. Putting decoder in pipeline reduces the clock cycle time requirement and enhances the bandwidth. However, the cache is still not capable of limiting the overall cache delay in pipeline stages to single clock cycle (Fig. 4).

The proposed pipeline technique is implemented in 64K cache for different technology generations. To scale the bit-lines, the cache is divided into multiple banks. The overall

Table 1. The Number of Clock Cycle Vs Technology

Technology (µ)	Unpipelined Conventional Cache	Cache with Pipelined Decoder
0.25	1	1
0.18	2	2
0.13	3	2
0.10	3	3
0.07	4	3





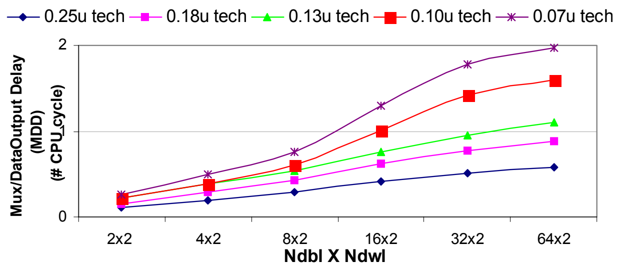


Fig. 5. Delay analysis. a) Decoder delay, b) Word-line to Sense amplifier delay via bit-line, c) Mux to Data output delay.

cache delay is divided into three parts: decoder delay (DD), word-line to sense amplifier delay (WD+BSD) and multiplexer to data output delay (MDD). Fig. 5 shows the variation of these delays with banking for different technology generations. The delay for a particular technology is normalized with respect to the clock cycle time for that technology as given in Fig. 4. To account for latch overhead, latch setup time and propagation delay are added to the WD+BSD and MDD. This overhead is around 10% of the clock cycle time for a given technology. DD remains within the bound of one clock cycle irrespective of technology scaling. Since Ndwl is kept constant, banking the cache does not affect DD. WD+BSD does not scale well

Table 2. Number of cache pipeline stages

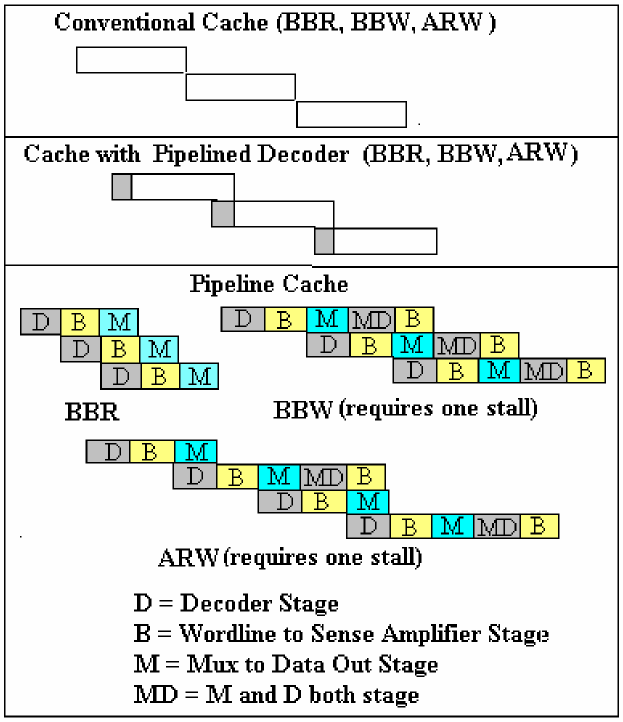
Tech (µ)	Banks (Ndbl X Ndwl)	Decoder Stage	WL - Sense Amp Stage	Mux to DataOut Stages	Total Cache Pipeline Stages
0.18	8x2	1	1	1	3
0.13	16x2	1	1	1	3
0.10	32x2	1	1	2	4
0.07	64x2	1	1	2	4

with technology. Scaling down the technology requires more clock cycles for a given bank configuration. For example, the number of cycles required for 4x2 bank configuration is one for 0.25µ and 0.18µ technology, whereas for 0.13µ and 0.10µ it is two cycles, and for 0.07µ technology it is three cycles (Fig. 4).

More banking makes the bit-line delay to go down. The goal behind banking is to get an optimal configuration for which the word-line driver to sense amplifier delay (with latch overhead) is equal a single CPU cycle. Over-banking will increase energy and area overhead without any performance improvement. Fig 5(b) shows that for a 0.25µ technology, the bit-line delay remains within the limit of one cycle for all bank configurations. Scaling the technology requires more banking to achieve one cycle bit-line delay, e.g. for 0.18µ, optimal banking is 8x2. Similarly for 0.13µ, 0.10µ, and 0.07µ technologies, optimal banking configurations are 16x2, 32x2, and 64x2, respectively.

For optimal banking, WD+BSD can be confined to a single clock cycle; however, it increases MDD. Since MDD consists of wire delay due to routing of the banks, it also does not scale effectively with technology. In the proposed scheme, the MDD is determined by the optimal banking configuration for a bit-line delay. Analyzing MDD (Fig. 5(c)) for the optimal configurations shows that for 0.25µ, 0.18µ, and 0.13µ technology, MDD remains within one clock cycle. For 0.10µ and 0.07µ technologies, optimal bank configurations, decided by bit-line delay, are 32x2 and 64x2 respectively, for which required MDD goes beyond one clock cycle. To make the cache fully pipelined with access frequency equivalent to clock frequency, the multiplexer can be divided into multiple pipeline stages. Table 2 shows the optimal banking requirement and the number of pipeline stages required to make the cache fully pipelined with access frequency of one clock cycle.

The design technique has its own disadvantages of having extra energy and area overhead. Table 3 shows the area and energy overhead associated with pipelining for different technology generations. The energy estimate considers the energy overhead due to extra latches, multiplexers, clock and decoders. It also accounts for the decrease in precharge energy as the bit-line capacitance is reduced. Increase in area due to extra precharge circuitry, latches, decoders, sense amplifiers, and multiplexer is considered in this analysis. To account for area and energy overhead together with performance gain we use MOPS



Conventional Cache (BBR, BBW, ARW)
Cache with Pipelined Decoder (BBR, BBW, ARW)
Pipeline Cache
D B M
D B M MD B
BBR
BBW (requires one stall)
ARW (requires one stall)
D = Decoder Stage
B = Wordline to Sense Amplifier Stage
M = Mux to Data Out Stage
MD = M and D both stage

Fig. 6. Stalls require for BBR, BBW, and ARW.

(million of operation per unit time per unit area per unit energy) as a metric for comparison. Three scenarios have been considered to calculate MOPS: i) Back to Back Read (BBR) operations, ii) Back to Back Write (BBW) operations, and iii) Alternate Read and Write (ARW). Fig. 6 shows the pipeline stalls required due to resource conflict for all three scenarios. In the case of conventional unpipelined cache, both read and write have to wait until previous read or write is finished. In a cache with pipelined decoder, DD is hidden into previous read or write access.

In the proposed pipeline design read access, is divided into three stages (Fig. 6). In the case of BBR, a read comes out of the cache in every clock cycle. A write is a read followed by a write and that requires five stages. Issuing write back-to-back encounters a structural hazard due to multiple access of the multiplexer stage in single write. Also the fourth stage requires the multiplexer and the decoder to be accessed simultaneously causing resource conflict in the pipeline. Issuing another write after two cycles of previous write resolves this hazard. Similar hazard is there in the case of ARW. Again issuing next read or write after two cycles of previous read or write resolves this hazard. Fortunately, similar stalls are required for the case when multiplexer is pipelined and accessed in two clock cycles (0.10 and 0.07µ technology). MOPS achieved by the pipelined cache is compared with the conventional and the cache with pipelined decoder.

Table 3 shows the percentage improvement in MOPS achieved by pipelining cache. For 0.18µ technology pipeline cache achieves 41% and 15.9% improvement in

Table 3. Performance, energy and area estimates.

Tech (μ)	% increase area	% increase Read Energy	% increase Write Energy	% increase in MOPS with respect to Conventional Cache			% increase in MOPS with respect to Cache with Pipelined Decoder		
				BBR	BBW	ARW	BBR	BBW	ARW
0.18	3.3	3.1	3.7	41.1	16.5	-6.5	15.9	3.86	-19.1
0.13	7.5	11.2	12.4	85.0	48.9	20.7	50.5	32.0	3.62
0.10	15.8	23.1	25.4	85.4	49.3	21.0	51.8	32.4	4.15
0.07	31.8	54.4	60.6	68.1	42.9	13.4	47.7	32.2	3.05

MOPS in the case of BBR and 16.5% and 3.86% improvement in MOPS in the case of BBW with respect to conventional cache and the cache with pipelined decoder, respectively. In the case of ARW, MOPS is lower for the proposed cache. For other technologies pipeline cache achieves significant improvement in MOPS ranging from 68.1 - 85.4% and 47.7 - 51.8% in the case of BBR, 42.9 - 49.3% and 32.0 - 32.4% in the case of BBW, and 13.4 - 21.0% in the case of ARW with respect to conventional unpipelined cache and the cache with pipelined decoder, respectively. The results show the effectiveness of the proposed methodology in designing a scalable pipelined cache that can be accessed every clock cycle.

4.2: Increasing the size of the cache

With growing need for higher performance, processors need larger cache to deal with large and independent workloads. Small caches cannot hold sufficient data that is required frequently by independent programs and leads to capacity and conflict misses. Increasing the size of the cache increases the access time. The proposed pipelined technique can be applied to design a large cache whose pipeline stage delay is equal to clock cycle time.

Fig. 7 shows the optimal banking required for pipelining large caches for different technology generations. Increasing the cache size increases either the number of rows or the number of columns. Increase in delay due to extra rows and extra columns can be reduced by increasing Ndbl and Ndwl, respectively. Increasing Ndwl divides the number of columns into multiple sections, creating more but shorter word-lines. It requires a bigger decoder. However, decoder stage delay for 64K cache is much smaller than clock cycle time (Fig. 5). Hence, there is sufficient slack available to merge the extra delay due to bigger decoder. Ndbl divides the bit-line capacitance and reduces the bit-line to sense amplifier delay.

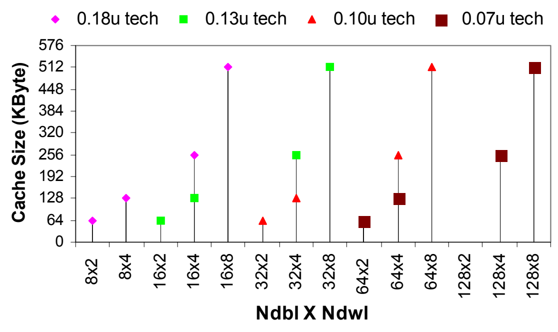


Fig. 7. Optimal banking required for pipelining the large size cache (64, 128, 256, and 512KB) for different technology.

5: Conclusions

In this paper we explored a design technique to effectively pipeline caches for higher bandwidth. Pipelining is made possible by aggressively banking the cache which makes word-line to sense amplifier delay to fit into single clock cycle. The technique is capable of splitting the cache into three, four or more pipeline stages as required by clock cycle time requirement. A large cache, having cache access frequency equal to clock frequency, can be designed by carefully banking the cache. The proposed pipeline cache dominates other designs in terms of the MOPS measure. The technique is fully scalable and very effective in pipelining future cache designs.

Acknowledgement: The research was funded in part by SRC (98-HJ-638) and by DARPA.

Reference

[1] K. Naogami, T. Sakurai et. al. A 9-ns hit-delay 32-kbyte Cache Macro for high speed RISC. IEEE Journal of Solid State Circuits, vol. 25, no. 1, February 1990.

[2] T. Wada and S. Rajan. An Analytical Access Time Model for On-Chip cache Memories. IEEE Journal of Solid State Circuits, Vol. 27, No 8, pages 1147-1156,August 1992.

[3] J.L. Hennessy and D.A. Patterson. Computer Architecture A Quantitative Approach. Morgan KaufMann, 2nd Edition.

[4] S. J. E. Wilson and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 93/5, Digital Equipment Corporation, Western Research Laboratory, July 1994.

[5] J. M. Rabaey. Digital Integrated Circuit. Prentice Hall, 1996.

[6] http://www-device.eecs.berkeley.edu/~ptm/

[7] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: maximizing on-chip parallelism. In Proceedings of the 22th Annual International Symposium on Computer Architecture, pages 392-403, June 1995.