ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

Adaptive Dead Time Predictor for SDRAM Memories System

S.Natarajan[1], V. Nishanth[2], E.Vijay Kumar[3], J.Niranjjan[4], S.Nitish Krishna[5]

Assistant Professor, Department of Electrical and Electronics Engineering, Vel Tech, Chennai, Tamil Nadu, India[1]

UG Students, Department of Electrical and Electronics Engineering, Vel Tech, Chennai, Tamil Nadu, India[2,3,4,5]

ABSTRACT*:* There is a huge gap between the speeds of the processor and the SDRAM memory. Therefore, lots of efforts are being made to decrease the SDRAM latency. One of the ways to achieve this goal is predicting whether to use the Open page or the Close page DRAM policy, i.e. dynamically switching from one policy to the other during program execution. In previous work the author of this paper considered such predictors and in this paper improvements of the dead time predictor are proposed. The dead time predictor is based on access interval time values. When a value that is equal to the last access interval time, multiplied by 2 or 4, elapses, it is predicted that the row has entered its dead time and a row precharge command is issued. Instead of multiplying only by 2 or 4 (as in previous work), in this paper several multiplication values, which are fixed throughout the whole program execution, are tried. Then an adaptive mechanism, which dynamically changes the multiplication value during program execution, is introduced. The gained adaptive dead time predictor yields the best results compared to all the variants with fixed multiplication values, with very little additional hardware requirements.

KEYWORDS*:* SDRAM, latency, adaptive, dead time, predictor.

I. INTRODUCTION

A desire for better potential utilization of processors, which are becoming faster and faster, demands a memory system with similar performance enhancements. A critical link in a hierarchically organized memory system is the main memory, implemented with chips of DRAM memory (Dynamic Random Access Memory). There is a huge gap between the speeds of the processor and the DRAM memory, called memory gap or memory wall [1], which is also increasing exponentially. C.Nagarajan et al [6,9,10] has developed the term wall points to the fact that if the trends continue, at one moment in the future we will hit that memory wall, and when this happens the computer system speed will no longer depend on the processor speed at all - it will be completely determined by the memory speed.

The processor-memory speed gap is generally being solved in two ways. First, cache memories are introduced. They are made of fast SRAM memories, with speeds similar to the processor's speed, so they are able to deliver all the needed data in time. But still, when cache misses occur and DRAM accesses are to be made, there are huge delays, long several tens of processor clock cycles. Additionally, these delays are also constantly increasing. This is the reason that various improvements of DRAM memories are also being introduced. These improvements include: synchronized DRAM memories (SDRAMs), larger numbers of independent banks, burst mode, various technological and scheme improvements.

The main consequence of these DRAM improvements is increased DRAM bandwidth (the quantum of data transferred per unit of time, but the latency (the first part of the delay, until the first datum arrives) is still large. Namely, DDR, DDR2, DDR3, etc. SDRAM memories all have practically the same DRAM core, with very little differences in the latency parameters. What changes is the bandwidth - DDR2 has twice the bandwidth of DDR, DDR3 has twice the bandwidth of DDR2 [2], etc. On the other hand, the latency changes much more slowly - only about 5% per year [3].

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

Therefore, in each new DRAM generation the latency part is becoming more and more dominant in the total delay time. It can be calculated that the latency part, when transferring a 128 B data block from the DDR3 SDRAM to the cache memory, comprises about 70-90% of the total delay time. Therefore, lots of efforts are being made to decrease the SDRAM latency.

A classic SDRAM controller uses two possible policies (strategies): Open Page (Row) policy and Close Page (Row) policy. When using the Open Page policy, the accessed row is kept open, which yields low latency if the next SDRAM access is directed to the same row, and high latency if the next SDRAM access is directed to some other row. In the first case (open row hit), the latency is equal to tCAS, and in the second case (open row miss), the latency is equal to the sum tRP+tRCD+ tCAS[2]. Here tRPis the row precharge (closing) time, tRCDisthe time needed for opening the row, and tCASis the column access time. When using the Close Page policy, each row is being closed after each access. This 'hides' tRP, so the latency is always the same - the sum tRCD+ tCAS[2].
A popular way to decrease the SDRAM latency is using a hybrid policy with help of predictors. In a perfect scenario, the Open Page policy should be used in cases of open row hits, and the Close Page policy should be used in cases of open row misses. If we correctly predict when to switch to Open Page policy we will have the latency of tCAS(instead of tRCD+ tCAS) and if we correctly predict when to switch to Close Page policy we will have the latency of tRCD+ tCAS(instead of tRP+ tRCD+ tCAS). The problem is that any wrong prediction actually increases the latency, instead of decreasing it. Therefore, we should minimize the number of wrong predictions.

In previous work the author of this paper redefined the metrics related to cache memories from [4] to be related to DRAM memories, as follows. Live time is a time interval that elapses from opening a row in a bank until the last access into that row before its closing. Dead time is a time which elapses from the last access to an open row until the moment of its closing. Access interval is a time interval which elapses between two consecutive accesses to an open row in a bank. Based on these redefinitions, three predictors were proposed [5], the simplest and cheapest of which is the dead time predictor (DTP). It keeps track of the last access interval time value for each of the banks. When a value that is equal to the last access interval time value, multiplied by 2 or 4 elapses, it is predicted that the row has entered its dead time and a proper

Row recharge command is issued. For some of the used benchmark programs multiplying by 2 was better option and for some of them it was multiplying by 4. I wanted to try with different values, like 8 or 16 and see the results. It is obvious that a greater multiplication value causes the system to have less number of misclosed rows (less number of wrong predictions that the row has entered its dead time), but it also causes less number of total predictions, i.e. greater number of omitted chances to close the row in advance. A less multiplication value, on the other hand, causes greater number of total predictions, which means a chance to have greater number of in-time row closings, as well as greater number or errors. It is logical that an ideal strategy would be to dynamically change the multiplication value, in order to adapt to the specific program, or specific part of the program, that is being executed.

II. RELATEDWORK

There are number of papers that deal with improving performances of the memory system. Some of them focus on improving performances of the cache memory subsystem and some on improving performances of the SDRAM memory. Some of the papers that consider improving performances of SDRAM memories try to improve the bandwidth, some optimize the SDRAM's power management, while some of them try to decrease latency of the SDRAM memory. This section will focus only on papers and patents that try to decrease the SDRAM's latency, mostly using predictors
M. Awasthi et al. [6] use a predictor that tracks the number of accesses to a given DRAM page to determine page closure decisions.
Y. Xu et al. [7] use solutions from the 2-level branch predictors to predict whether to close the open page or to keep it open, based on the analogy between the prediction of the branch as taken or not taken and the prediction of the next row as the same or different.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

M. Chiyuan and C. Shuming [8] collect statistical information of address distribution of all the instructions waiting for accessing memory and analyze regular changes of access addresses in order to make judgment of the appropriate moment to precharge each DRAM bank.
M. Chiyuan and N. Xiaoqiang [9] propose a new memory schedule policy oriented to stream architectures. Their policy can predict the access behavior of each DRAM bank and uses appropriate controller policy.
Y. H. Son et al. [10] propose asymmetric DRAM bank organizations to reduce the average access latency.
S.-I. Park and I.-C. Park [11] use 2b state machines to predict whether the successive memory access leads to a row hit or not and change the memory mode according to the prediction.
E. Krimer et al. [12] use saturation counters to maintain predictions of whether pending loads will conflict with older pending store operations.
B. Fanning [13] proposes a method and apparatus of dynamically adjusting a memory system's existing paging policy. The method generates a select signal according to at least one input signal and the existing paging policy to the
memory system and proceeds to modify the existing paging policy based on the generated select signal.
B. Emberling [14] proposes a predictive optimizing unit that is to be used with an interleaved memory, suitable in computer graphic systems. His unit maintains a queue of pending requests for data from the memory and prioritizes precharging and activating interleaves with pendingrequests.

III. SYSTEM SIMULATION MODEL

In this paper the same system simulation model and set of benchmark programs as in [5] was used. I integrated the program Sim-Outorder from Simplescalar 3.0 [15] with a DDR3 SDRAM simulator, written by myself. The simulated superscalar CPU issues 4 instructions on every clock cycle and supports out of order instruction execution, with a 2-level branch predictor. The CPU clock frequency was 3.2 GHz. There were 2 levels of caches - L1 contains separate instruction and data caches, both 16 kB large, with direct mapping and line sizes of 32 B, while L2 contains a unified cache, 2 MB large, with set-associative mapping, 4 lines per set, and line size of 128 B. All the cache memories use the write-back policy. The simulated DDR3 SDRAM has the following characteristics: 8 banks per chip, 8 k rows per bank, row capacity is 2 kB, tRP= tRCD= tCAS= 12.5ns.

TABLE I

OPEN ROW HIT PROBABILITIES OF THE USED BENCHMARKPROGRAMS

Benchmark program	*ORH prob.*
perl	0.07
compress	0.09
mcf	0.11
bzip2	0.26
ijpeg	0.33
gcc	0.36
cc1	0.37
m88ksim	0.47
li	0.53
anagram	0.70
go	0.72
go*	0.74
li*	0.80
m88ksim*	0.89
anagram*	0.90
compress*	0.91

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

Executions of 11 benchmark programs were simulated: cc1, compress, ijpeg, li, m88ksim, perl, go from SPEC95, bzip2, gcc, mcffrom SPEC2000 and anagram. The last one is a program that finds all the possible words that can be obtained by combinations of the letters of 3 names: Todd Austin, Scott Breach and GuriSohi. For 5 of the benchmark programs the simulated L2 cache size was large enough for the entire program's data to be placed in it. As a consequence the SDRAM had small number of accesses. In order to 'fix' this, versions with smaller L2 cache sizes were also tried for these programs, which yielded larger numbers of SDRAMaccesses.
The versions with larger L2 caches are signed with a star (*) after the benchmark name. The new versions give variety to the benchmark program set, since they obtain rather different open row hit probabilities, as can be seen in Table I - the probabilities range from only 0.07 for perl to even 0.91 for compress*, with gradual growth in between. This variety of open row hit probabilities really tests the used predictor and its prediction accuracies.

IV. DEAD TIME PREDICTOR WITHFIXED MULTIPLICATIONVALUES

The DTP (dead time predictor) is based on access interval time values. Simulation results show that the average dead time is much larger than the average access interval time, hence when a value that is equal to the last access interval time, multiplied by 2 or 4, elapses, it is predicted that the row has entered its dead time. To implement the DTP the DRAM controller needs several things:

- There has to be one counter for each bank to take care of the elapsed time since last access. In order to minimize the counters' length, they may be triggered with a signal derived by dividing the DRAM'sclock.
- A register for each bank is needed for storing the last access interval time. Every time there is an open page hit in any of the banks that means occurrence of a new access interval time, hence the counter's value for that bank should be stored into the proper register and the counter should be reset. The operation of multiplying by 2 or 4 could be implemented by fixing the least significant bit to zero (the two least significant bits if multiplying by 4) and then storing into this register from the nextposition(s).
- The DRAM controller also needs one comparator for each bank. In case the counter's value becomes greater or equal to the register's value the row is to beclosed.
- We also need a queue with orders for bank precharges from the DTP. The SDRAM controller performs a queue read operation and issues a row precharge of the proper bank(s), every time it isidle.

First I tried simulating 8 variants of the DTP: multiplying by 2, 4, 8, 16, 32, 64, 128 and 256. The gained average DRAM latencies, expressed in processor clock cycles, are shown in Fig. 1. We may notice 2 groups of programs. In the first group, which comprises the first 7 programs (perltocc1) the strategy x2 (multiplying by 2) is the best and the strategy x256 (multiplying by 256) is the worst. For some of the programs in this group there are significant differences between x2 and x256. For the rest of the programs the "middle" strategies (x8, x16 or x32) are mostly the best, while x2 and x256 are the worst. These results suggest trying an adaptive mechanism, which dynamically changes the multiplication value. The goal is to obtain performances similar to the performances of x2 in the first group of programs and similar to the performances of x8, x16 and x32 in the second group. Another advantage of such a mechanism might be the fact that the optimal multiplication value may change during program execution for the same program –in one part of the program it may be optimal to use x2 or x4, while in some other part it may be optimal to use x64 or x128, or even x256. A proper adaptive mechanism may optimally utilize all such situations.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

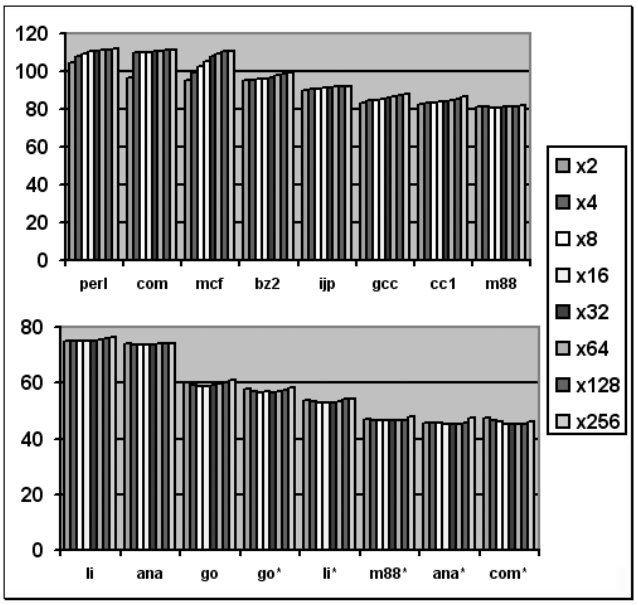

Fig. 1. Average DRAM latencies (expressed in processor clock cycles) for Dead Time Predictor with fixed multiplication values

V. ADAPTIVE DEAD TIMEPREDICTOR

In order to dynamically change the multiplication value, we may simply add a few bits to each of the banks, and use them as a saturated counter. The counter's value indicates the proper multiplication value. Every time the opened row is misclosed the counter is incremented (except its value is maximal).

Every time we omitted to close the opened row the counter is decremented (except its value is 0). The only needed change in the control of this adaptive DTP would be the multiplying itself, which can be simply accomplished by a shift operation (for proper number of positions). This means that the needed changes in the hardware, required to implement the adaptive mechanism would be rathersmall.

Three variants of the new adaptive DTP were tried - using only 1 bit (switching only between x2 and x4), using 2 bits (changing between x2, x4, x8 or x16), and using 3 bits (changing between x2, x4, x8, ..., x128 or x256). The results are shown in Fig. 2. The expressions ‘WFMV’ and ‘BFMV’ in this figure stand for Worst and Best Fixed Multiplication Value, respectively - the worst and the best of the 8 strategies from Fig. 1.

We can observe 2 things in Fig. 2. First, there are no huge differences between 1b,2band 3b - 1b is almost as good as 2b and 3b. Second, in practically all the cases, the 2b and 3b strategies yield identical (or in some cases slightly better) performances as (than) BFMV. This means that we accomplished the goal - the adaptive DTP yields the same (or slightly better) performances as the best possible basic DTP - x2 in the first group of programs and the "middle" strategies in the secondgroup.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

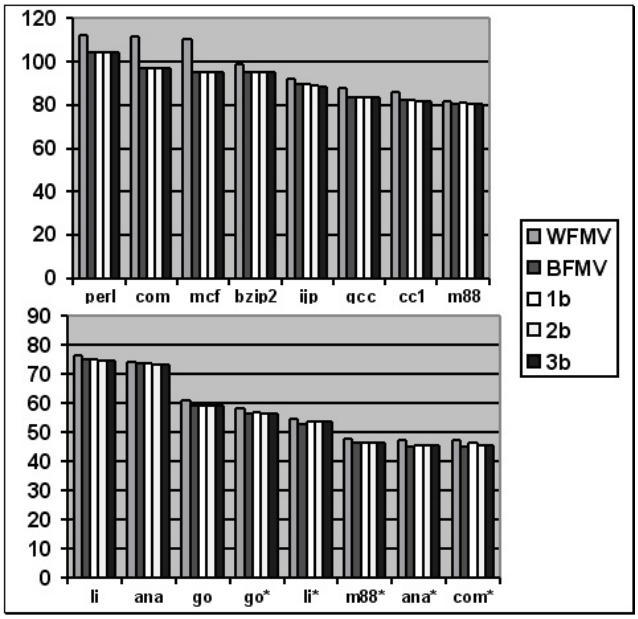

Fig. 2. Average DRAM latencies (expressed in processor clock cycles) for Adaptive Dead Time Predictor

It is obvious that the amount of needed additional hardware elements is very low. However, to confirm this, as well as to evaluate the actual costs, I implemented the DTP on an FPGA chip, Xilinx SpartanII family, model xc2v500-6fg256. The old x2 version demands 17,718 equivalent gates, with 1,404 used 4-input LUTs, and the new 2b version demands 18,454 equivalent gates, with 1,484 used 4-input LUTs. We may conclude that the adaptive DTP yields nice improvements with very little additional hardware investments. Future work will include studying possibilities of introducing similar adaptive/intelligent mechanisms to the other two proposed predictors.

VI. CONCLUSION

The optimal multiplication value that should be used in the dead time predictor varies from program to program. In some programs the smallest value (2) should be used, and in some programs the used value should be much greater (8, 16 or even 32). The optimal value may also change during program execution - various parts of the program may need different multiplication values to yield minimal SDRAM latency. In this paper an adaptive mechanism was proposed. It dynamically changes the multiplication value, and as such always chooses the optimal value for each program, or each part of the program, with very little additional hardware requirements. Computer simulations show that the gained adaptive dead time predictor yields the best results compared to all the variants with fixed multiplication values.

VII. ACKNOWLEDGEMENT

The research presented in this paper was supported in part by the Serbian Ministry of Education, Science and Technological Development [TR32012].

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 9, Issue 3, March 2020

REFERENCES

[1] W. A. Wulf, S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," ACM Computer Architecture News, Vol. 23, No. 1, pp. 20-24, March1995.
[2] B. Jacob, S. Ng, D. Wang, "Memory Systems: Cache, DRAM, Disk", Morgan Kaufman, 2007.
[3] J. Hennessy, D. Patterson, "Computer Architecture: A Quantitative Approach", 5th Edition, Morgan Kaufman,2011.
[4] Z. Hu, S. Kaxiras, M. Martonosi, "Timekeeping in the Memory System: Predicting and Optimizing Memory Behavior," Proc. 29th Ann. International Symp. Computer Architecture (ISCA '02), pp. 209-220, May2002.
[5] V. Stankovic, N. Milenkovic, "DDR3 SDRAM with a Complete Predictor," IEICE Transactions on Information and Systems, Vol. E93-D, No.9, September 2010, pp.2635-2638.
[6] C.Nagarajan and M.Madheswaran - 'Stability Analysis of Series Parallel Resonant Converter with Fuzzy Logic Controller Using State Space Techniques'- Electric Power Components and Systems, Vol.39 (8), pp.780-793, May 2011
[7] M. Awasthi, D. W. Nellans, R. Balasubramonian, A. Davis, "Prediction Based DRAM Row-Buffer Management in the Many-Core Era," Proc. International Conf. Parallel Architectures and Compilation Techniques, pp. 183-184,2011.
[8] M. Chiyuan, C. Shuming, "A DRAM Precharge Policy Based on Address Analysis," Proc. 10th Euromicro Conf. Digital System Design Architectures, Methods and Tools (DSD '07), pp. 244-248, Aug.2007.
[9] K Umadevi, C Nagarajan, "High Gain Ratio Boost-Fly Back DC-DC Converter using Capacitor Coupling", 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), 2nd and 3rd March 2018, organized by mahendra Engineering College, Mallasamudram, PP. 64-66,2018
[10] E Geetha, C Nagarajan, "Induction Motor Fault Detection and Classification Using Current Signature Analysis Technique", 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), 2nd and 3rd March 2018, organized by mahendra Engineering College, Mallasamudram, PP. 48-52,2018