

Omitting Refresh

A Case Study for Commodity and Wide I/O DRAMs

Matthias Jung, Éder Zulian, Deepak M. Mathew, Matthias Herrmann,
Christian Brugger, Christian Weis, Norbert Wehn
Microelectronic Systems Design Research Group
University of Kaiserslautern
67663 Kaiserslautern Germany
{jungma,zulian,deepak,herrmann,brugger,weis,wehn}@eit.uni-kl.de

ABSTRACT

Dynamic Random Access Memories (DRAM) have a big impact on performance and contribute significantly to the total power consumption in systems ranging from mobile devices to servers. Up to half of the power consumption of future high density DRAM devices will be caused by refresh commands. Moreover, not only the refresh rate does depend on the device capacity, but it strongly depends on the temperature as well. In case of 3D integration of MPSoCs with Wide I/O DRAMs the power density and thermal dissipation are increased dramatically. Hence, in 3D-DRAM even more DRAM refresh operations are required. To master these challenges, clever DRAM refresh strategies are mandatory either on hardware or on software level using new or already available infrastructures and implementations, such as *Partial Array Self Refresh* (PASR) or *Temperature Compensated Self Refresh* (TCSR).

In this paper, we show that for dedicated applications refresh can be disabled completely without or with negligible impact on the application performance. This is possible if it is assured that either the lifetime of the data is shorter than the currently required DRAM refresh period or if the application can tolerate bit errors to some degree in a given time window.

CCS Concepts

•Computer systems organization → Reliability; Processors and memory architectures;

Keywords

DRAM, Retention, Refresh, 3D-Integration, Reliability, Approximate Computing

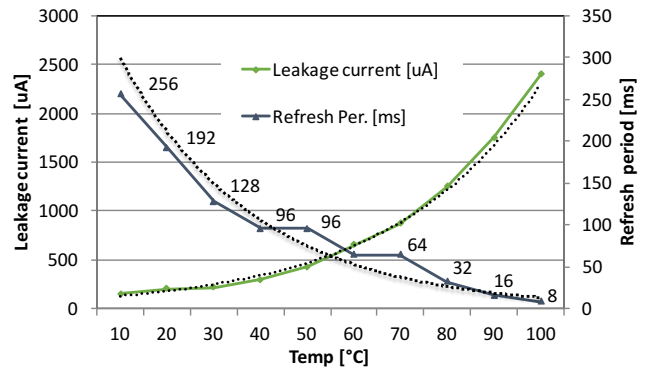


Figure 1: Leakage Current and Required Refresh Periods at Different Temperatures. [28, 31, 39]

1. INTRODUCTION

Memory energy consumption has become a significant concern in mobile computing, servers and high-performance computing platforms. There are applications, such as used in the GreenWave computing platform [25], in which 49% of the total power consumption has to be attributed to DRAMs.

DRAMs must be refreshed regularly due to their charge based bit storage property (capacitor). The retention time of a DRAM cell is defined as the amount of time that a DRAM cell can safely retain data without being refreshed [18]. This DRAM refresh operation must be issued periodically and causes both performance degradation and increased energy consumption. Liu et al. [29] predicted that 40% to 50% of the power consumption of future DRAM devices will be caused by refresh commands (compare Table 1).

Moreover, 3D integrated DRAMs like Wide I/O or HMC worsen the temperature behavior. Due to the much increased leakage at the cells the refresh frequency needs to be adjusted accordingly to avoid retention errors [39] as shown in Figure 1. Thus, the efficient utilization of the available DRAM bandwidth and the efficient usage of DRAM power-down modes and clever refresh techniques are major contributions to a high energy efficiency of DRAM subsystems and the computing system in which they are integrated [22].

Recently, a lot of research has been done (see Section 2) to minimize the number of refresh operations by means of different techniques. However, it was not considered that for some safety uncritical applications DRAM *Auto-Refresh* (AR) can be disabled completely without or with negligible

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MEMSYS '15, October 05-08, 2015, Washington DC, DC, USA

© 2015 ACM. ISBN 978-1-4503-3604-8/15/10...\$15.00

DOI: <http://dx.doi.org/10.1145/2818950.2818964>

Ref.	Impact on	DRAM Device Size (Gb)					
		2	4	8	16	32	64
[29]	Avg. Power	10%	15%	17%	24%	34%	47%
[5]	Energy	10%	14%	16%	18%	25%	?
[29]	CPU Perf.	-5%	-7%	-9%	-14%	-26%	-46%
[5]	CPU IPC	-5%	-8%	-11%	-18%	-36%	?
[5]	CPU Latency	4%	6%	8%	13%	24%	?

Table 1: Impact of Refresh on Future DRAM Devices

impact on the application performance, called *Omit Refresh* (OR) strategy. This strategy is feasible, if it is assured that either the lifetime of the data is shorter than the currently required DRAM refresh period or the application can tolerate bit errors to some degree in a given time window.

Similar to Flicker [30] we propose a reliable and an unreliable memory region. However, we go one step further: Instead of lowering the refresh rate in the unreliable region we apply the OR-Strategy.

As shown in Figure 3 the lowest 3D-DRAM layer has the highest average temperature. Hence, this layer requires a higher refresh rate than the rest of the DRAM stack [39]. Consequently, the lowest layer is a perfect candidate for applying the OR-Strategy by tolerating an unreliable memory layer in order to save refresh power. While the upper reliable part of the stack is refreshed the unreliable region can be accessed exclusively, which is managed by the DRAM controller. In combination with a compiler that can handle reliable and unreliable data types [40, 30] OR can be used even on CPU based applications.

A retention error aware DRAM model is key to analyze the impact of lower refresh rates or even disabled refresh (OR) on executed applications. Especially for error resilient applications, this can be exploited to save energy [16]. We measured [46] the retention times of WIDE I/O [14, 24] and DDR3 DRAM devices using different data pattern. We observed *data pattern dependencies* (DPD), shown in Figure 2 and *variable retention times* (VRT), similar to the study in [28] for commodity DRAMs. With this data we created a DRAM retention time error model [46] that can be used to study energy vs. reliability trade-offs. Obviously, the aspect ratio of the DRAM cell capacitor is increasing by technology scaling, while the dimensions of the select transistor are decreasing. Thus, the leakage power and the reported soft error rates are higher for newer DRAM device technologies [34]. We considered this effect in our executed experiments.

In this paper, we apply the OR-Strategy on different applications that range from image processing to big data accelerators. This highlights the feasibility of this approach and shows the impact on each selected application.

2. RELATED WORK

DRAM refresh has become one of the most important research topics in the DRAM community. This section gives a survey of the important state-of-the-art techniques to reduce the number of refreshes. A more detailed recent survey can be found in [5].

Reliability vs. Refresh.

The REVA [1] refresh scheme can be used in dedicated video applications. It refreshes only the important *region of interest* (ROI) in a video frame. Flicker [30] reduces the number of refreshes by partitioning a DRAM bank in a

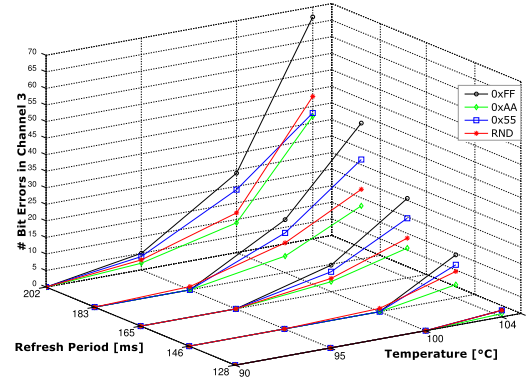


Figure 2: Measurement Results with Fixed Refresh Periods

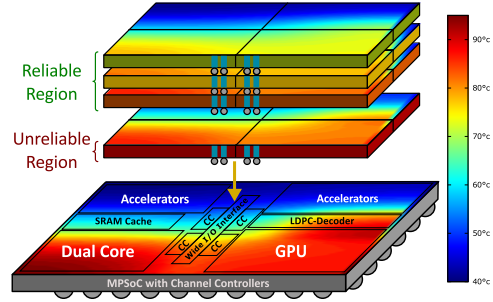


Figure 3: Wide I/O DRAM on an MPSoC, simulated with gem5 [8] DRAMSys [21], DRAM-Power [11] and 3D-ICE [41]

critical and non-critical region. The non-critical region will be refreshed with a lower refresh rate.

Selective Refresh.

PARIS [2], DTail [13] and ESKIMO [20] exclude rows that do not store useful data from being refreshed. Smart Refresh [15] refreshes only rows that have not been accessed recently. CREAM [47], [39] and [12] show per-bank and per-subarray refresh techniques. The authors of [37] issue refresh commands according to the stored data values. Flexible Auto-Refresh [7] shows a realistic implementation of a flexible and row selective refresh.

Thermal aware Refresh.

A temperature variation aware bank-wise refresh for 3D-DRAMs is presented in [39]. [22] presents how DRAM self-refresh can be approached in a staggered way.

Refresh Scheduling.

The idea of postponing DRAM refresh into self-refresh phases is presented in [6]. The authors of [35] show how to use JEDEC's DDR4 fine granularity refresh (FGR) efficiently, [43] presents an enhancement (EFGR). In [42], Elastic Refresh, an approach that adapts the refresh behavior according to the current workload is shown. Refresh Pausing [36] is a technique that allows to pause a current refresh operation to serve a DRAM access. To make refresh predictable for real-time application it can be triggered from software level, as shown in [4].

Retention Aware Refresh.

RAPID [44] is a software approach that allocates longer-retention pages before shorter-retention pages. The refresh rate is adjusted according to the shortest-retention page used. *RAIDR* [29] forms rows into retention time bins that are refreshed with different rates. *RIO* [2], a software approach, excludes weak rows from usage. *SECRET* [27] and [48] store the content of weak cells in a separate region in the DRAM by using an error correction pointer mechanism. The authors of [17] lower the overall refresh rate and refresh weak rows selectively by issuing single **ACT-PRE** commands. In *ProactiveDRAM* [45] a list of weak rows exists in the DRAM device and the DRAM itself decides at each refresh command if the row should be refreshed. *AVATAR* [38] tries to overcome VRT issues by combining an online ECC detection with row selective refresh.

Overall, it is very difficult to create a valid list of weak cells for DRAMs, as they experience VRTs and DPDs for their retention times. Moreover, in [46] it is shown that the temperature has a strong effect on VRT. Hence, it is infeasible during startup of a system to determine an exact list of weak cells that considers all parameters, such as temperature, retention time and DPD. Thus, it is preferable to omit refresh in the unreliable regions and use standard guard banding in the reliable regions for commodity devices (see Figure 1) and advanced margins for 3D stacked devices, as it was analyzed in [26]. Moreover, in the reliable part of the stack the previously mentioned state-of-the-art techniques e.g. temperature aware bank-wise refresh [39] can still be applied.

3. EXPERIMENTS

To prove our assumptions and to demonstrate the feasibility of the OR-Strategy we conduct three experiments. The first study investigates the similarity in large complex graphs while considering data that need no refresh. In the second experiment we evaluate the input data resilience of an LDPC decoder for wireless baseband processing. Finally, in the last application we execute an image processing task on a XILINX FPGA to obtain real measurement data of the applied OR-Strategy.

3.1 Big Data

Complex graphs are at the heart of today’s big data challenges like recommendation systems, customer behavior modeling, or incident detection systems. One reoccurring task in these fields is the extraction of network motifs, reoccurring and statistically significant subgraphs. In this example we show the influence of OR to a precisely tailored embedded architecture [10] for computing similarities based on one special network motif, the co-occurrence. It is based on efficient and scalable building blocks that exploit well-tuned algorithmic refinements and an optimized graph data representation approach.

In [10] the following method is considered for calculating similarities using the co-occurrence: In the example given in Figure 4, the similarity between *you* and *Liam* is based on the number of common friends, the so-called co-occurrence: $coocc(you, Liam) = 3$. The question arises whether the number three is significant. Assume *you* and *Liam* have thousands of friends, versus *you* have only three. For this random graphs are created based on the same degree sequence and the premise that nodes have no similarities.

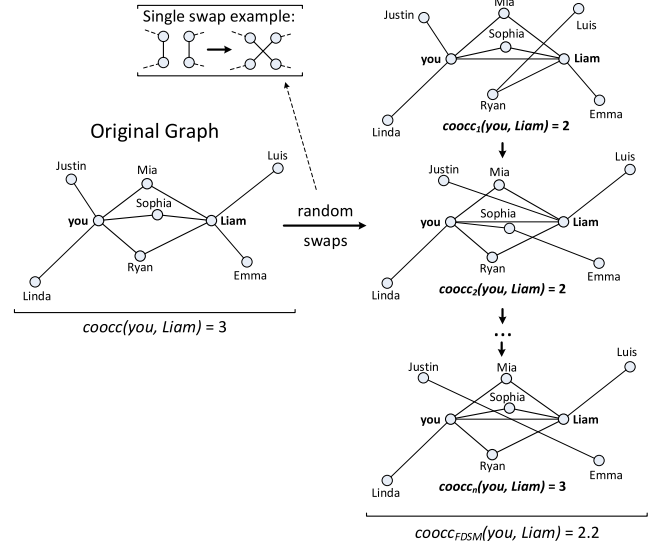


Figure 4: Co-Occurrence and Swapping in a Graph

To get the random graphs a sufficient number of pairs of edges are swapped, drawn uniformly at random, if and only if no multiple edges would arise due to the swap. This generates independent graphs with the same degree sequence, see Figure 4. Generating many of such graphs the expected co-occurrence can be calculated. That information can judge how significant the similarity in the original graph is.

However, for large graphs this is in general a very time and memory consuming job on standard computing clusters. Therefore, this application has been implemented as an ASIC [10], where the graph is stored in the DRAM as sparse adjacency matrix.

There is a reason to believe that a certain amount of errors in the large graph matrix will not influence the quality of the result. For instance, when this ASIC is used as accelerator in an MPSoC similar to Figure 3, the graph matrix can be stored in the unreliable region, whereas important control variables are stored in the reliable region of the memory.

To show the feasibility of this idea we simulated similarity measures for the Netflix dataset [19, 3] in combination with our retention time error model [46]. Netflix, a commercial video streaming service, has released 100,480,507 user ratings for all of their 17,700 movies from 480,189 users. For this simulation we use a data subset with 20,000 users.

The dataset is stored in both list and matrix form. The list allows to select edges uniformly at random, while the matrix allows to check for conflicting edges in constant time. The datasets for recommendation are typically sparse, as each user only rate a few movies. In our case, the matrix has a sparsity of 1% and a size of 337 Mbit. We store the edge in the matrix as zeros. In that case, retention errors (transitions from '1' to '0', as shown in [46] for a Wide I/O device) can only introduce new edges, but not delete existing edges. This is important for the algorithm to not introduce duplicate edges in the swap step. The list has a size of 99 Mbit and is stored in the reliable region.

In each co-occurrence calculation the complete dataset is touched at least one time and this is equivalent to a complete refresh. The required lifetime of the data in the DRAM

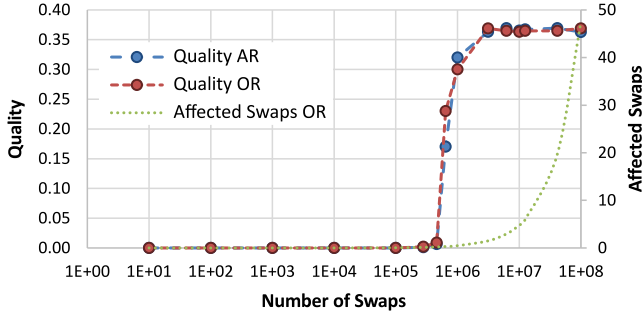


Figure 5: Netflix - 20k Users

between two co-occurrence calculations (swap-phase) is approximately $244ms$. In this time frame retention errors can occur. This is modeled with the model of [46] under a worst case temperature of $90^{\circ}C$.

To check the impact of omitting refresh, we compare the quality of the output of the Link Assessment algorithm with normal *Auto-Refresh* (AR) and OR. The quality is assessed based on a ground-truth, a set of human selected movie pairs of highly similar movies, as described in [9]. In our case, it is assembled based on all permutations of movie sequels. The higher these pairs are ranked in the output, the higher the quality. In [9] so called phase-transitions have been observed for key variables like the number of swaps. Figure 5 shows this phase transition with AR and OR. Important features are the position of the transition and the final level, the higher and more left the better. Both curves for AR and OR lie on each other, apart from statistical variations, no difference in quality can therefore be observed. In green the number of affected swaps are shown also in Figure 5. Of 10^8 swaps only 48 swaps are affected by retention errors on average. This explains the negligible impact of OR on the resulting quality.

3.2 Channel Decoding

In digital communication systems channel coding can be considered as key-technology to achieve reliable communication. Thus, it plays an integral role in all important modern communication standards.

One of the currently most sophisticated concepts of channel coding is *Low-Density Parity Check* (LDPC) coding. This circumstance is also reflected in its adoption to recent wireless communication standards like WiGig and WiFi. The strength of LDPC coding is its iterative decoding algorithm that allows the utilization of information on the reliability of each bit, so-called soft-information. As for most iterative decoders, the number of iterations is limited to a maximum value and can be adjusted on demand without deterioration of the result. As a consequence, the amount of iterations the decoder performs strongly depends on the degree of impairment of the currently received signal. This feature drastically reduces the energy consumption of the decoder.

In this experiment we determine the communications performance of an LDPC decoder that is used in an MPSoC as shown in Figure 3. The unreliable region of the DRAM is used for buffering the input soft-information. The communication system simulation consists of a source that generates random binary data, a WiGig LDPC channel encoder, a *Binary phase-shift keying* (BPSK) modulator that maps

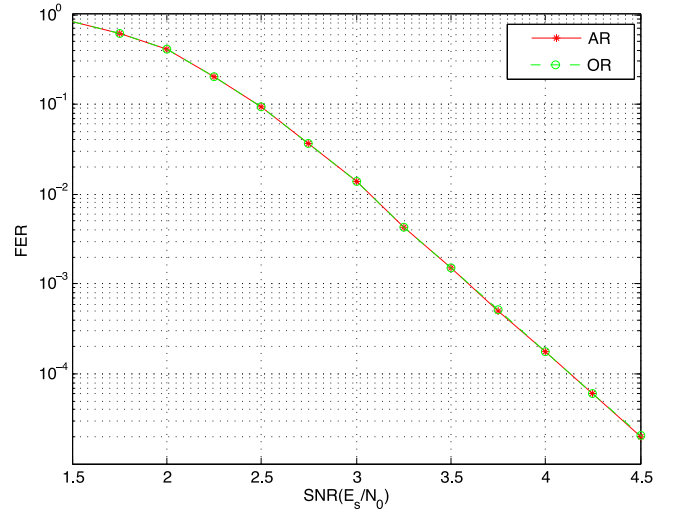


Figure 6: Communications performance of two WiGig decoders with and without OR

bits to the signal values $+1$ and -1 respectively, a channel that adds random gaussian-distributed noise to our signal, and the component's corresponding counterparts on receiver side. The channel code is a rate $13/16$ WiGig code of length $N = 672$. Communications performance is determined by Monte-Carlo simulation of 10^7 frames for a given *Signal-to-Noise-Ratio* (SNR) and keeping track of the corresponding *Frame-Error-Rate* (FER) after the channel decoding.

The results are shown in Figure 6. We can clearly see that although some of the bits in the input memory of the decoder have flipped (due to OR), the FER is the same as with enabled AR. This shows that the influence of the retention errors in the DRAM on the FER after decoding is much smaller than the influence of the errors induced by the channel. This is known as inherent error resilience of channel coding [16].

Figure 7 shows that for a worst case temperature of $100^{\circ}C$ and a refresh rate for AR of $t_{REF} = 8ms$ about 12% of the total energy can be saved by disabling the refresh (OR).

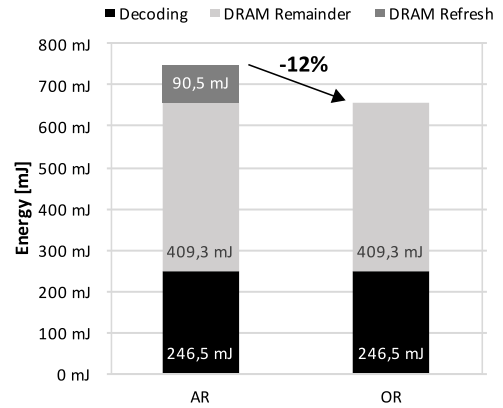


Figure 7: Energy Breakdown of WiGig Decoder and DRAM for 10^7 Frames at $SNR=3.5dB$

3.3 Image Processing

In this section we demonstrate that our OR-Strategy can be beneficial for even today's commodity DDR3 DRAMs by using an image processing FPGA application in which the lifetime of the image in the memory is less than the DDR3 refresh interval (t_{REF}).

In this application, an image stream from a camera is rotated by 90 degrees using the DRAM. Each image (frame) consists of 4,525,056 pixels with 48 bits per pixel. An image is stored into the DRAM at consecutive address locations and it is retrieved back from specific locations in a way such that the image is rotated. This application has a real-time constraint (deadline) of 8.6ms on the frame rate. As shown in [4] DRAM refresh is an unpredictable factor for worst-case execution (WCET) analysis. Therefore, it makes a lot of sense to disable refresh for applications with real-time deadlines smaller than t_{REF} .

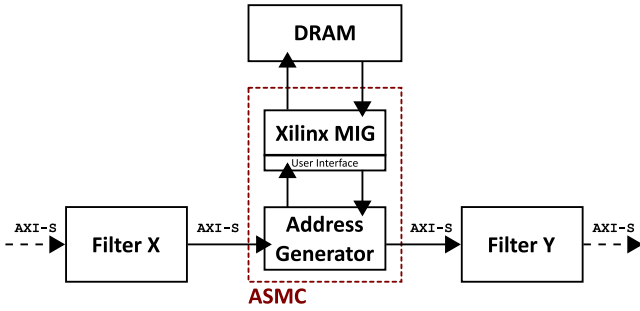


Figure 8: Image Processing Pipeline with Application Specific Memory Controller (ASMC)

Figure 8 shows a part of the image processing pipeline realized on a Xilinx Zynq ZC706 FPGA Evaluation Board [32]. The board consists of a 1 GB, x64 DDR3 SO-DIMM connected to the *Programmable Logic* (PL) side of the FPGA. We use a customized *Application Specific Memory Controller* (ASMC) based on the Xilinx *Memory Interface Generator* (MIG) [33]. This controller has a lean front-end, which consists of an AXI-Stream interface and an address generator. The address generator calculates the addresses to store and retrieve images, since AXI-Stream has no address information. It internally consists of a scrambler, which transforms [23] the bits of the generated addresses to achieve an optimized access pattern for this particular application.

The FPGA evaluation board has four different voltage rails controlled by Texas Instruments UCD90120A power supply sequencer and monitor [32]. We use the TI USB Interface adapter along with the TI Fusion Digital Power graphical user interface to record the voltage and current levels of the rails. The measurements are made only on the voltage rail 3, since this rail solely powers the DDR3 SO-DIMM and the PHY I/Os of the memory controller.

We conducted two different experiments: auto refresh enabled (AR) and with refresh disabled completely (OR). In both cases we sent a fixed number of frames (611,620) to the DRAM and monitored the execution time as well as the rail voltage and currents. With refresh enabled, the MIG issues an *Auto-Refresh* (AR) command to the DRAM every 7.8μs. These refreshes interfere with the normal *READ*/*WRITE* command execution of the MIG memory controller and therefore increases the time to process the complete set of frames.

However, in this application each frame is stored in the memory only for less than 9ms. Since this is much less than the 64ms refresh period of the DDR3 DRAM at normal temperature and the DRAM is exclusively used for the rotation task, we disable the refresh mechanism inside the MIG controller (by modifying the Verilog source code) and conducted the second experiment.

In our test setup all frames are checked for data corruption and no bit error occurred. As shown in Figure 9 the normal runtime to process 611,620 frames is 43.7min. This time is reduced by applying OR by 62s. The required energy can be reduced by 1.6%. Even though the savings in time and energy are not significant, this experiment shows the feasibility of the OR approach for a real device. As soon as more refresh operations are required for higher capacity commodity DRAM devices OR will be a perfect option for many applications.

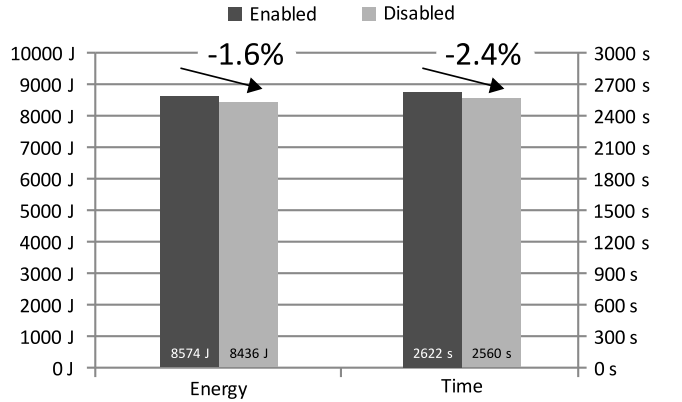


Figure 9: Results of enabled (AR) and disabled refresh (OR) for an image processing task

4. CONCLUSION

In this paper, we investigated the feasibility of the OR-Strategy for three different applications. We demonstrated that switching-off the refresh during co-occurrence calculation in graph processing has a negligible impact on the resulting quality of the prediction. We presented the influence on the QoS of an LDPC decoder for wireless baseband processing, while applying OR on its input data buffer. We found a very small increase in iterations with no impact on the FER. Additionally, we executed an image processing task on a XILINX FPGA with OR applied on the attached DDR3 DRAMs. This application was running error free for a huge number of frames and decreased the execution time and energy consumption. Overall, we have shown that the OR-Strategy is sensible to use in these three applications and in others that are able to tolerate errors to some degree. Especially, for future high-density DRAMs the OR-Strategy will be very beneficial as it will recover the available bandwidth and reduce the average power consumption.

5. ACKNOWLEDGMENTS

The authors thank the company Synopsys for their great support. This work was partially funded by the German Research Foundation (DFG) as part of the priority program Dependable Embedded Systems SPP 1500 (<http://spp1500.itec.kit.edu>) and the DFG grant no. WE2442/10-1.

6. REFERENCES

- [1] S. Advani, N. Chandramoorthy, K. Swaminathan, K. Irick, Y. Cho, J. Sampson, and V. Narayanan. Refresh Enabled Video Analytics (REVA): Implications on power and performance of DRAM supported embedded visual systems. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, pages 501–504, Oct 2014.
- [2] S. Baek, S. Cho, and R. Melhem. Refresh now and then. *Computers, IEEE Transactions on*, 63(12):3114–3126, 2014.
- [3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.
- [4] B. Bhat and F. Mueller. Making DRAM Refresh Predictable. In *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on*, pages 145–154, July 2010.
- [5] I. Bhati, M.-T. Chang, Z. Chishti, S.-L. Lu, and B. Jacob. DRAM Refresh Mechanisms, Trade-offs, and Penalties. *Computers, IEEE Transactions on*, PP(99):1–1, 2015.
- [6] I. Bhati, Z. Chishti, and B. Jacob. Coordinated Refresh: Energy Efficient Techniques for DRAM Refresh Scheduling. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design, ISLPED '13*, pages 205–210, Piscataway, NJ, USA, 2013. IEEE Press.
- [7] I. Bhati, Z. Chishti, S.-L. Lu, and B. Jacob. Flexible auto-refresh: enabling scalable and energy-efficient DRAM refresh reductions. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 235–246. ACM, 2015.
- [8] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.
- [9] C. Brugger, A. L. Chinazzo, A. F. John, C. De Schryver, N. Wehn, A. Spitz, and K. A. Zweig. Exploiting Phase Transitions for the Efficient Sampling of the Fixed Degree Sequence Model. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 2015.
- [10] C. Brugger, V. Grigorovici, M. Jung, C. Weis, C. D. Schryver, K. A. Zweig, and N. Wehn. A Custom Computing System for Finding Similarities in Complex Networks. In *Proceedings of the 2015 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 262–267, Montpellier, France, July 2015.
- [11] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, O. Naji, M. Jung, N. Wehn, and K. Goossens. DRAMPower: Open-source DRAM power & energy estimation tool.
- [12] K. K.-W. Chang, D. Lee, Z. Chishti, A. R. Alameldeen, C. Wilkerson, Y. Kim, and O. Mutlu. Improving DRAM performance by parallelizing refreshes with accesses. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, pages 356–367. IEEE, 2014.
- [13] Z. Cui, S. A. McKee, Z. Zha, Y. Bao, and M. Chen. DTail: A Flexible Approach to DRAM Refresh Management. In *Proceedings of the 28th ACM International Conference on Supercomputing, ICS '14*, pages 43–52, New York, NY, USA, 2014. ACM.
- [14] D. Dutoit, C. Bernard, S. Cheramy, F. Clermidy, Y. Thonnart, P. Vivet, C. Freund, V. Guerin, S. Guilhot, S. Lecomte, et al. A 0.9 pJ/bit, 12.8 GByte/s WideIO memory interface in a 3D-IC NoC-based MPSoC. In *VLSI Technology (VLSIT), 2013 Symposium on*, pages C22–C23. IEEE, 2013.
- [15] M. Ghosh and H.-H. Lee. Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs. In *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, pages 134–145, Dec 2007.
- [16] C. Gimmler-Dumont and N. Wehn. A Cross-Layer Reliability Design Methodology for Efficient, Dependable Wireless Receivers. *submitted to ACM Transactions on Embedded Computing Systems*, 2013.
- [17] Y.-H. Gong and S. Chung. Exploiting Refresh Effect of DRAM Read Operations: A Practical Approach to Low-power Refresh. *Computers, IEEE Transactions on*, PP(99):1–1, 2015.
- [18] T. Hamamoto, S. Sugiura, and S. Sawada. On the retention time distribution of dynamic random access memory (DRAM). *Electron Devices, IEEE Transactions on*, 45(6):1300–1309, Jun 1998.
- [19] <http://www.netflixprize.com/>, last access: 2014-12-01.
- [20] C. Isen and L. John. ESKIMO - energy savings using semantic knowledge of inconsequential memory occupancy for DRAM subsystem. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 337–346, Dec 2009.
- [21] M. Jung, C. Weis, N. Wehn, and K. Chandrasekar. TLM modelling of 3D stacked wide I/O DRAM subsystems: a virtual platform for memory controller design space exploration. In *Proceedings of the 2013 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO '13*, pages 5:1–5:6, New York, NY, USA, 2013. ACM.
- [22] M. Jung, C. Weis, N. Wehn, M. Sadri, and L. Benini. Optimized active and power-down mode refresh control in 3D-DRAMs. In *Very Large Scale Integration (VLSI-SoC), 2014 22nd International Conference on*, pages 1–6, Oct 2014.
- [23] M. Jung, C. Weis, N. Wehn, and N. Wehn. DRAMSys: A flexible DRAM Subsystem Design Space Exploration Framework. *IPSSJ Transactions on System LSI Design Methodology (T-SLDM)*, August 2015.
- [24] J.-S. Kim, C. S. Oh, H. Lee, D. Lee, H.-R. Hwang, S. Hwang, B. Na, J. Moon, J.-G. Kim, H. Park, J.-W. Ryu, K. Park, S.-K. Kang, S.-Y. Kim, H. Kim, J.-M. Bang, H. Cho, M. Jang, C. Han, J.-B. Lee, K. Kyung, J.-S. Choi, and Y.-H. Jun. A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4x128 I/Os using TSV-based stacking. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 496–498, Feb 2011.
- [25] J. Krueger, D. Donofrio, J. Shalf, M. Mohiyuddin,

- S. Williams, L. Oliker, and F.-J. Pfreundt. Hardware/software co-design for energy-efficient seismic modeling. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2011.
- [26] J. Lim, H. Lim, and S. Kang. 3D Stacked DRAM Refresh Management with Guaranteed Data Reliability. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, PP(99):1–1, 2015.
- [27] C.-H. Lin, D.-Y. Shen, Y.-J. Chen, C.-L. Yang, and M. Wang. SECRET: Selective error correction for refresh energy reduction in DRAMs. In *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, pages 67–74, Sept 2012.
- [28] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu. An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms. *SIGARCH Comput. Archit. News*, 41(3):60–71, June 2013.
- [29] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *Proceedings of the 39th Annual International Symposium on Computer Architecture, ISCA '12*, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.
- [30] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn. Flickr: Saving DRAM Refresh-power Through Critical Data Partitioning. *SIGPLAN Not.*, 46(3):213–224, Mar. 2011.
- [31] Micron Technology Inc. 4Gb: x16, x32 Mobile LPDDR3 SDRAM. July 2013.
- [32] Xilinx. zC706 Evaluation Board for the Zynq-7000 XC7Z045 All Programmable Soc User Guide. April 2015.
- [33] Xilinx, Inc. Memory Interface Generator (MIG). <http://www.xilinx.com/products/intellectual-property/mig.html>, 2015, Last Access: 18.02.2015.
- [34] J. Meza, Q. Wu, S. Kumar, and O. Mutlu. Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2015.
- [35] J. Mukundan, H. Hunter, K.-h. Kim, J. Stuecheli, and J. F. Martínez. Understanding and Mitigating Refresh Overheads in High-density DDR4 DRAM Systems. In *Proceedings of the 40th Annual International Symposium on Computer Architecture, ISCA '13*, pages 48–59, New York, NY, USA, 2013. ACM.
- [36] P. J. Nair, C.-C. Chou, and M. K. Qureshi. Refresh Pausing in DRAM Memory Systems. *ACM Trans. Archit. Code Optim.*, 11(1):10:1–10:26, Feb. 2014.
- [37] K. Patel, L. Benini, E. Macii, and M. Poncino. Energy-Efficient Value-Based Selective Refresh for Embedded DRAMs. In V. Paliouras, J. Vounckx, and D. Verkest, editors, *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, volume 3728 of *Lecture Notes in Computer Science*, pages 466–476. Springer Berlin Heidelberg, 2005.
- [38] M. K. Qureshi, D.-H. Kim, S. Khan, P. J. Nair, and O. Mutlu. AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems. *Memory*, 2(4Gb):20, 2015.
- [39] M. Sadri, M. Jung, C. Weis, N. Wehn, and L. Benini. Energy Optimization in 3D MPSoCs with Wide-I/O DRAM Using Temperature Variation Aware Bank-Wise Refresh. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–4, March 2014.
- [40] F. Schmoll, A. Heinig, P. Marwedel, and M. Engel. Improving the Fault Resilience of an H.264 Decoder Using Static Analysis Methods. *ACM Trans. Embed. Comput. Syst.*, 13(1s):31:1–31:27, Dec. 2013.
- [41] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschweiler, and D. Atienza. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *Proc. of ICCAD 2010*, 2010.
- [42] J. Stuecheli, D. Kaseridis, H. Hunter, and L. John. Elastic Refresh: Techniques to Mitigate Refresh Penalties in High Density Memory. In *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, pages 375–384, Dec 2010.
- [43] V. K. Tavva, R. Kasha, and M. Mutyam. EFGR: An Enhanced Fine Granularity Refresh Feature for High-Performance DDR4 DRAM Devices. *ACM Trans. Archit. Code Optim.*, 11(3):31:1–31:26, Oct. 2014.
- [44] R. Venkatesan, S. Herr, and E. Rotenberg. Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM. In *Proc. of HPCA*, 2006.
- [45] J. Wang, X. Dong, and Y. Xie. ProactiveDRAM: A DRAM-initiated retention management scheme. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on*, pages 22–27, Oct 2014.
- [46] C. Weis, M. Jung, P. Ehses, C. Santos, P. Vivet, S. Goossens, M. Koedam, and N. Wehn. Retention Time Measurements and Modelling of Bit Error Rates of WIDE I/O DRAM in MPSoCs. In *Proceedings of the conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2015.
- [47] T. Zhang, M. Poremba, C. Xu, G. Sun, and Y. Xie. CREAM: a Concurrent-Refresh-Aware DRAM Memory Architecture. In *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*, pages 368–379. IEEE, 2014.
- [48] D. Zhu, R. Wang, Y. Wei, and D. Qian. Reducing DRAM refreshing in an error correction manner. *Science China Information Sciences*, pages 1–14, 2015.