Refresh Enabled Video Analytics (REVA): Implications on Power and Performance of DRAM Supported Embedded Visual Systems

Siddharth Advani*, Nandhini Chandramoorthy*, Karthik Swaminathan*,
Kevin Irick[†], Yong Cheol Peter Cho[‡], Jack Sampson* and Vijaykrishnan Narayanan*
*Pennsylvania State University [†]SiliconScapes LLC. [‡]ETRI, South Korea

Abstract— **Video applications are becoming ubiquitous in mobile and embedded systems. Wearable video systems such as Google Glasses require capabilities for real-time video analytics and prolonged battery lifetimes. Further, the increasing resolution of image sensors in these mobile systems places an increasing demand on both the memory storage as well as the computational power. In this work, we present the Refresh Enabled Video Analytics (REVA) system, an embedded architecture for multi-object scene understanding and tackle the unique opportunities provided by real-time embedded video analytics applications for reducing the DRAM memory refresh energy. We compare our design with the existing design space and show savings of 88% in refresh power and 15% in total power, as compared to a standard DRAM refresh scheme.**

I. Introduction

Due to the capacity of human vision systems for highly complex processing at very low power, many brain-inspired algorithms and architectures have been proposed to emulate the human visual cortex. [1, 2, 3]. Most works in this domain have focused mainly on enhancing the performance and energy efficiency of the computational fabrics and do not address the inefficiencies of the main memory system. The memory system contributes between 10-30% of the overall power of embedded video systems and mobile phones [4]. The increasing memory size in new generations of embedded systems and the use of stacked 3D architectures that increase on-chip temperatures have drawn increasing attention on reducing the memory refresh energy. Consequently, there have been sustained efforts to introduce new power-efficient techniques such as Low Power Auto Self Refresh, Temperature Controlled Refresh, Refresh Pausing, Fine Granularity Refresh and Data Bus Inversion in new memory standards such as DDR4 [5]. Tuning DRAM refresh based on the data characteristics has been proposed as early as 1998 [6]. Recently, a software approach, termed as *Flikker* was proposed that relies on the user to annotate critical and non-critical parts [7]. It also allows refresh rates to be different for critical and non-critical sections of the memory and conserves the refresh energy.

In this work, we focus on the unique opportunities provided by real-time embedded video analytics applications for reducing memory refresh energy. The analysis is based on a real-time image detection and recognition system that has been emulated on an FPGA based platform which detects objects of interest using an attention algorithm. These objects can then be picked up from subsequent frames and classified. This system is useful in a range of end applications such as unmanned air-vehicles, security cameras and aids for the visually-impaired. In this paper, we make the following contributions with regard to reducing the memory refresh energy in such a system.

- We identify that in streaming data, the lifetime of some parts of the data are significantly less than the refresh periods of a DRAM and disable refresh in these parts of the memory.
- We automatically recognize portions of an image as critical based on the saliency-recognition algorithms employed in vision algorithms and selectively refresh those portions. In contrast, to [7] which statically annotates portions of the code and partitions them into critical and non-critical regions, our automated system can exploit both data dependent and task dependent information to identify salient regions within a single image frame.
- In embedded systems, the lifetimes of streaming data are significantly less than the refresh periods of a DRAM. This enables us to disable refresh in these parts of the memory and eliminate refresh for portions that are guaranteed to be accessed within a specific time period. We propose the underlying architecture to support such a visual pipeline with an optimized memory sub-system.

II. System Overview

In this section, we provide a brief overview of the accelerators used in our system, as well as a brief overview on memory design and its role in our vision-based system.

A. Accelerators

Various attention-based computational models [8, 9] have used low-level features to build information maps which are then fused together to form what is called a *saliency map*. Given an image, this saliency map provides a compact representation in terms of what is most important in the image. We use Itti's model [10] of saliency which serves as a backend to the entire cognitive pipeline. We adopt a computational model called HMAX [11] that emphasizes hierarchical processing of objects, like in the visual cortex. Our goal here is to identify objects in a scene and the HMAX accelerator processes the Regions of Interest (RoI) generated by Saliency to identify the class or label of that object. The Action Recognition accelerator is extended from [12] by integrating spatio-temporal detectors with template matching and pooling layers to the object detection engine. This enables it to track features over time and classify temporal features such as human actions. Figure 1 shows a video frame extracted from a fixed camera setup on a tower and the resulting RoIs generated. HMAX can then be used for object and action recognition.

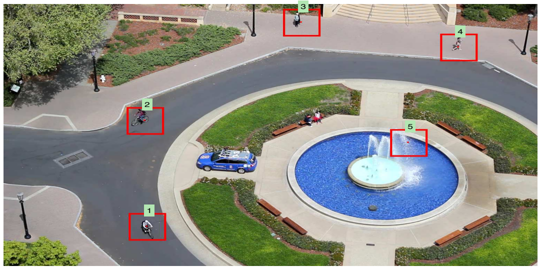


Fig. 1. RoIs generated by Saliency. The RoIs are subsequently classified with object labels by HMAX. Objects belonging to 'person' class can be further processed to classify the action.

B. Memory Design for Accelerator Systems

In a DRAM cell, data is retained for only as long as the capacitor can hold the charge injected into it during a write operation. Hence, the data in each cell needs to be refreshed periodically. A refresh operation involves reading a row of data and writing it back from the sense amplifier. The memory controller periodically issues refresh commands at rank-level granularity. A refresh controller keeps track of the address of the last row to have been refreshed so that the subsequent refresh command can be issued to the next group of rows. Each refresh command issued by the controller is appended to the command queue along with other memory read and write commands. According to JEDEC SDRAM standards [5], a DRAM device can issue a maximum of 8192 refresh commands within a 64 ms refresh window, leading to a refresh command being issued every 7.8μs (t_{REFI}). The major drawback of DRAM is the additional overhead in performance and power on account of the refresh operation. A single refresh command to a row causes the entire bank of the DRAM to stall. Hence several techniques have been incorporated into the DDR protocols to minimize overlaps between read/write and refresh commands. Figure 2a) shows the increasing nature of the parameter t_{RFC} - the amount of time that each refresh locks up the DRAM - across different DRAM densities. As an example of the impact, Figure 2b) shows energy consumption during various kernel-based computations across various DRAM generations. The refresh component in the DRAM is thus becoming a major source of concern with increasing DRAM densities. Considerable work has gone into the area of reducing refresh power consumptions. Some have used scheduling policies and retention profiling [13], while others have used software techniques [14]. In [15], the authors use a refresh pausing mechanism to exploit idle memory cycles. In [16], Fine Grained Refresh (FGR) for DDR4 DRAM systems is analyzed. However, most of these schemes are for a general purpose multicore system and are not geared to exploit the data-specific aspects prevalent in our system.

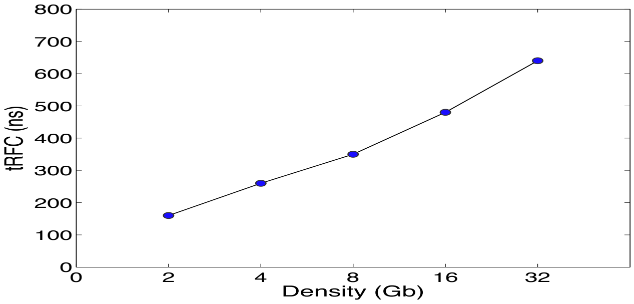


Fig. 2. a) The REF command time (tRFC) is increasing with each generation [5]. Values for 16 and 32 Gb devices are based on projections.

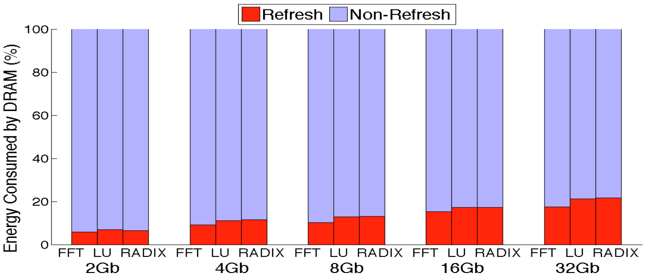


Fig. 2. b) Refresh energy distribution in Kernel computations

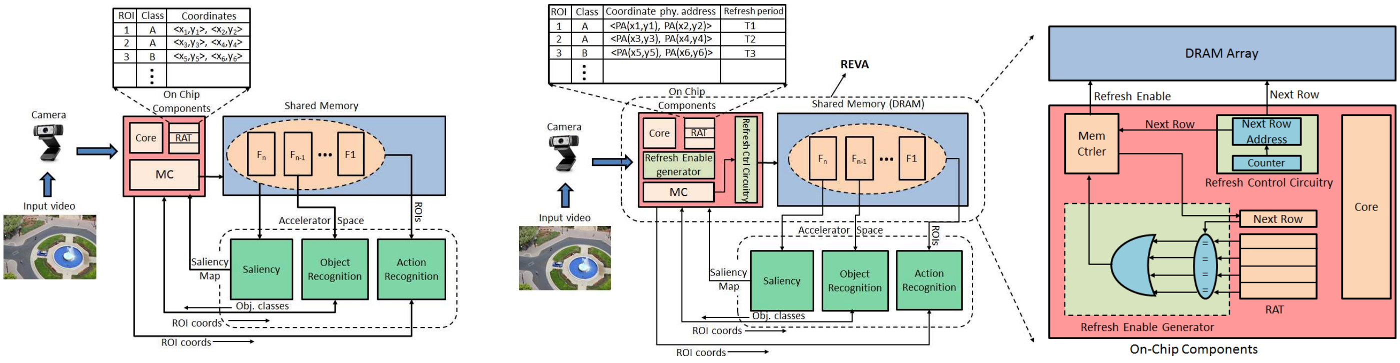
III. Architecture of Proposed System

Figure 3a) illustrates the baseline architecture for the vision analytics-based system. The input video is captured by the HD camera at a rate of 30 frames/s. These frames are stored in the off-chip DRAM, as shown in the figure. These frames are then read by the saliency accelerator and a *Saliency Map* is computed. It is then used to compute the RoIs. The Object Recognition accelerator classifies the objects in each RoI. The Action Recognition Engine requires a series of consecutive frames for action classification. Since each of these frames over the entire length of the video is stored in the main memory module, there is substantial overhead in terms of refresh energy. In our proposed design, shown in Figure 3b), the DRAM primarily serves the purpose of a stream buffer. Consequently the frames are rarely reused after processing, which may eliminate the need for refresh. However, some accelerator operations like action recognition, require a sequence of previously stored frames to be recalled. Therefore we only need to *selectively* refresh the RoIs involved in these functions. We thus propose *Differential Refresh* rates based on the RoI and object class with which it has been associated by the object recognition engine.

A. Differential Refresh Architecture

As explained above, in the time duration of Refresh Cycle Time (t_{RFC}), a group of rows is refreshed sequentially when a refresh pulse is scheduled. When a refresh pulse is sent every refresh interval (t_{ref}), we keep track of the next row to be refreshed in a bank in the *Next Row Address* register. At the end of every refresh iteration, the contents of this register is sent to the memory controller. We maintain an on-chip CAM structure called the *RoI Address Table (RAT)* to list RoIs in each frame and their corresponding starting addresses and sizes. The RAT is analogous to the processor register file with each new RoI resulting in a new entry being written into it. Since the action recognition accelerator requires around 20 frames for a reasonably accurate classification, assuming a maximum of 4 RoIs per frame, the RAT can have a maximum of 80 entries. When the address of the next row to be refreshed does not match an RoI address on associative search, Refresh Enable signal is set to 0. This ensures that rows of an image in a bank that do not belong to RoIs are refreshed less frequently than the rows that contain RoIs.

Each frame is interleaved across banks at page granularity. The RAT is updated once the saliency accelerator computes the RoIs as shown in Figure 3c). After a refresh iteration, when the



ROI	Class	Coordinates
1	A	<x_1,y_1>, <x_2,y_2>
2	A	<x_3,y_3>, <x_4,y_4>
3	B	<x_5,y_5>, <x_6,y_6>
	⋮	

ROI	Class	Coordinate phy. address	Refresh period
1	A	<PA(x1,y1), PA(x2,y2)>	T1
2	A	<PA(x3,y3), PA(x4,y4)>	T2
3	B	<PA(x5,y5), PA(x6,y6)>	T3
	⋮		

Fig. 3. a) Baseline architecture

Fig. 3. b) Architecture of Proposed System

Fig. 3. c) Design of REVA block

memory controller gets the *Next Row Address* and a match for this is found in the RAT, a refresh pulse is sent to the command queue; otherwise Refresh Enable signal is set to low. An RoI is a 2-d rectangular tile and might correspond to different rows in different banks; the group of rows containing a portion of the RoI might also have non-RoI regions striped across multiple banks. The energy overhead of refreshing non-RoI regions in rows that contain RoI is less than that incurred on applying a uniform auto-refresh policy to the entire image. Since the non-RoI regions of the image are not subsequently read/written to, we allow the non-critical sections of the image to degrade with infrequent refreshes. On the other hand, non-image data such as code are considered to be critical and assigned a standard auto-refresh policy. Note that for completely stream-based processing, a no-refresh policy can be implemented since each image row is read no more than once.

B. Comparison with existing schemes

Selectively refreshing the DRAM depending on data criticality has been demonstrated earlier in [7]. However, their scheme proposed mapping physical pages of an RoI to a specific partition in the bank, incurring additional OS overhead. In contrast, REVA does not involve page-level mapping of Virtual Addresses of RoIs to physical pages of a partition. Also, when the saliency accelerator generates the saliency map, RoIs are not re-written to dedicated variable refresh frequency partitions. Instead, the saliency map is used to derive RoIs in the input image already stored in memory and refresh frequencies are dynamically varied based on the RoI locations. Figure 4 contrasts the two schemes described above. It can be observed that REVA makes dynamic refresh allocations, which lends more flexibility and reduces the need for redundant RoI writes. REVA involves additional communication overhead between memory controller and refresh controller since the address of the next row to be refreshed is sent after every refresh iteration. To compute these overheads, we use McPAT 1.0 [17] to estimate the power of the CAM-based RAT. Our experiments resulted in an additional power of around 2.4 mW, which is less than 1% of the overall system power.

IV. Experimental Setup and Results

A. Experimental Infrastructure

Figure 5 shows the infrastructure setup we used to carry out our evaluations. We used DRAMSim2 [18], which we customized to incorporate our refresh mechanisms. The input traces to DRAMSim2 were generated using Xilinx Vivado [19] in the following manner. The different accelerators, namely saliency, object and action recognition were modelled in Verilog and their output was used to generate parameters which were fed to a traffic generator. This traffic generator outputs the memory trace comprising of read and write memory accesses. Table I shows the parameters used in our DRAM model.

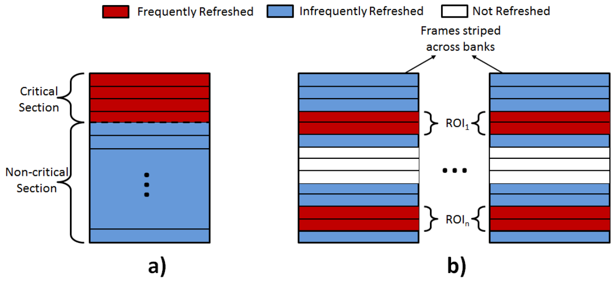


Fig. 4. Refresh mechanisms in a) Flikker b) REVA

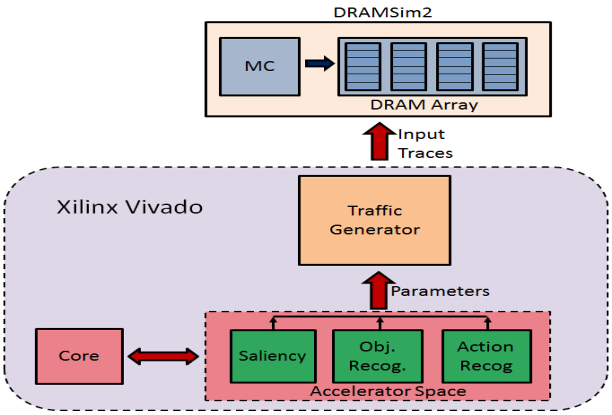


Fig. 5. Infrastructure setup

DRAM row buffer policy	Open page Closed after 4 accesses
DRAM configuration	DDR3-1600, 8 Gb, 1 channel, 1 rank, 8 banks/rank
Timing parameters (ns)	$t_{RP} = 11, t_{RCD} = 11$ $t_{RFC} = 350, t_{REFI} = 7800$
Current parameters (mA)	$I_{DD0} = 110, I_{DD1} = 135, I_{DD2P} = 40 = I_{DD2Q}$ $I_{DD2N} = 42, I_{DD3N} = 45, I_{DD4W} = 280$ $I_{DD4N} = 270, I_{DD5} = 215, I_{DD6} = 12$

TABLE I. DRAM PARAMETERS

B. Results

Figure 6 shows the total power consumption evaluated by feeding the traces generated from Vivado into DRAMSim2 for a 8 Gb DRAM. We evaluate REVA versus different baseline and state-of-the-art schemes. These include a completely refreshless system, *Off*, a state-of-the-art refresh-aware scheme like *Flikker* and a default mobile DRAM-based auto-refresh scheme (*Baseline*). REVA is dynamically capable of changing the refresh value of an RoI, by simply modifying the refresh period in the corresponding RAT entry, thus avoiding any additional writes that would be needed to re-write the RoI into a frequently refreshed memory. In contrast, Flikker places all the frames in a low refresh window. While the accuracy of tasks like object recognition are relatively unaffected by this, tasks like action recognition require a high degree of accuracy while recording frame-to-frame differences. This limits the range of tasks supported by Flikker. As a result, the expected gains from REVA's finer grained approach are feasible for a larger set of tasks, with the power savings always at least as much as Flikker. The maximum possible improvement would occur when we turn refresh completely off. On account of 88% of the refresh power being eliminated, REVA is able to achieve within 94% of this maximum power saving, without compromising on the accuracy that would result in a refreshless architecture. With the fraction of total power due to refresh continuing to increase with subsequent generations, the savings from REVA are expected to grow as well.

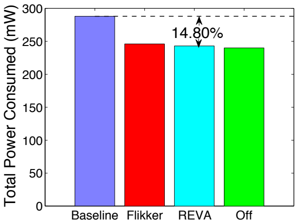


Fig. 6. Power comparisons for different schemes.

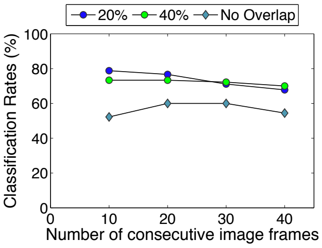


Fig. 7. Accuracy results for different video segment lengths and fractions of overlap between segments.

C. Sensitivity Analysis

While it can be argued that in a purely streaming embedded system one can turn off refresh, we provide a compelling reason to have a variable dynamic refresh mechanism. There can be instances when a burst of RoIs are generated and the HMAX accelerator cannot sustain a high throughput of generating class labels. RoIs need to be buffered in the DRAM for processing at a later stage. Further, in a multi-object scenario, if a person's action is being recognized, another person's actions need to be buffered so as to be processed at a later stage. Figure 7, shows different configurations of action recognition on the Weizmann dataset [20]. The results indicate that a purely streaming (no overlap of video segments) configuration affects accuracy considerably (by around 10%).

V. Conclusion

In this paper, we demonstrated the Refresh Enabled Video Analytics system, which is tuned to optimize energy utilization of the memory by exploiting data characteristics in vision-based applications. We showed that depending on the RoI stored in memory, the need for refresh varies from row to row across each memory bank. By adjusting refresh rates selectively depending on the need for re-use of different RoIs, we demonstrate 88% improvement in refresh power and 15% improvement in overall power over existing DRAM schemes.

Acknowledgments

This work was supported in part by NSF Expeditions in Computing Award 1317560, and NSF award 1213052. The work was also supported by infrastructure provided by NSF Award 1205618.

References

[1] A. Nere, A. Hashmi, and M. Lipasti, "Profiling Heterogeneous Multi-GPU Systems to Accelerate Cortically Inspired Learning Algorithms," in *IPDPS*, 2011.

[2] T. Chen, J. Wang, Y. Chen, and O. Temam, "DianNao : A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," in *ASPLOS*, 2014.

[3] S. Kestur *et al.*, "Emulating Mammalian Vision on Reconfigurable Hardware," in *FCCM*, 2012.

[4] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *Usenix Annual Technical Conference*, 2010.

[5] "JEDEC DDR3 and DDR4 SDRAM Standard," 2012.

[6] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the dram refresh count for merged DRAM/logic LSIs," in *ISLPED*, 1998.

[7] S. Liu, Pattabiraman K., T. Moscibroda, and B. Zorn, "Flikker: Saving DRAM Refresh-power through Critical Data Partitioning," in *ASLPOS*, 2011.

[8] L. Itti and C. Koch, "Computational Modelling of Visual Attention," *Nature Reviews Neuroscience*, 2001.

[9] N. Bruce and J. Tsotsos, "Saliency, Attention, and Visual Search: An Information Theoretic Approach," *Journal of Vision*, 2009.

[10] R. Peters and L. Itti, "Applying computational tools to predict gaze direction in interactive visual environments," *ACM Trans. on Applied Perception*, 2007.

[11] J. Mutch and D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *IJCV*, 2008.

[12] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *ICCV*, 2007.

[13] J. Stuecheli, D. Kaseridis, H. Hunter, and L. John, "Elastic Refresh: Techniques to Mitigate Refresh Penalties in High Density Memory," in *MICRO*, 2010.

[14] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh," in *ISCA*, 2012.

[15] P. Nair, C. Chou, and M. Qureshi, "A case for Refresh Pausing in DRAM memory systems," in *HPCA*, 2013.

[16] J. Mukundan, H. Hunter, K.-H. Kim, and J. Stuecheli, "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems," in *ISCA*, 2013.

[17] S. Li *et al.*, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[18] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *Computer Architecture Letters*, 2011.

[19] "Vivado Design Suite," 2014.

[20] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *ICCV*, 2005.