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

***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 for wide adoption. 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. Consequently, there is growing interest in energy-efficient algorithms and hardware for video analytics. 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 88.1% in refresh power and 14.8% in total power, as compared to a standard DRAM refresh scheme. Results indicate the benefits of utilizing ”smart” energy-efficient memories in next-generation real-time visual systems.**

I. Introduction

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 for wide adoption. 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. Consequently, there is growing interest in energy-efficient algorithms and hardware for video analytics. Deriving inspiration from the energy-efficiency of the visual cortex, many brain-inspired algorithms and architectures have been proposed to support energy-efficient video analytics applications [1, 2, 3, 4, 5].

In order to achieve fast and energy efficient implementations of these algorithms, it has been found general-purpose processors are inadequate, even with highly tuned software optimizations. Consequently, a host of application-specific accelerators are employed, allied with a general purpose core for other tasks such as synchronization and co-ordination between the individual specialized computation engines.

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 [6]. 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 [7]. However, software exploitation of these advanced hardware features has generally lagged [8, 9].

Tuning DRAM refresh based on the data characteristics has been proposed as early as 1998 [10]. Recently, a software approach, termed as *Flikker* was proposed that relies on the user to annotate critical and non-critical parts. This technique has been used to effectively exploit a modified version of the standard Partial Array Self Refresh (PASR) hardware mechanism [11]. It also allows refresh rates to be different for critical and non-critical sections of the memory and conserve the refresh energy. There have been several other works in the design of accelerator-based systems for vision analytics. For instance [12] proposes *EFFEX*, a specialized heterogeneous multicore design for vision applications in the embedded domain. In this paper, the authors have proposed customizations to the memory hierarchy, such as hardware-software co-optimized patch memory architectures. However, the memory optimizations are mainly restricted to the software level and there remains substantial scope for optimizing the memory architecture itself, especially from the point of view of improving energy and bandwidth utilization.

In this work, we focus on the unique opportunities provided by real-time embedded video analytics applications for reducing the memory refresh energy. The analysis is based on an real-time image detection and recognition system that has been emulated on a FPGA based platform. This system detects objects of interest using an attention algorithm and then the object can be picked up from a subsequent frame where it is recognized to be of a particular class. The recognized object can be further passed to a next level for more detailed recognition. This system represents generic components in a wide variety of end applications in unmanned air-vehicles, security cameras, visual-aids for visually-impaired and automatic weapon systems. We thus make the following contributions with regard to reducing the memory refresh energy based of this video analytics application.

- First, we recognize that in streaming data, the lifetime of some parts of the data are significantly less than the refresh periods of a DRAM. Therefore, we completely disable refresh in these parts of the memory. This is an enhancement to current techniques that reduce refresh times instead of completely shutting off the refresh in non-critical portions of the data. We are also able to eliminate refresh for portions of the memory that are guaranteed to be accessed within a specific time period due to the application specific nature of the embedded video system.

- Second, we are able to automatically recognize portions of an image as critical based on the saliency-recognition algorithms employed in brain-inspired vision algorithms and selectively refresh portions of data. This is contrast to prior efforts that have focused on manual annotations of critical and non-critical regions that limit such classification to be static. An example of such annotation in [11], annotates the code and certain data structures such as pointers to list of frames as critical while consider the image data itself as non-critical. This approach limits the granularity of such annotations to a coarse-grain, especially when such criticality is data-driven. In contrast, our automated system can exploit both data dependent and task dependent information to identify salient regions within a single image frame. The human visual cortex filters a significant amount of the raw visual stimuli for further processing by using attention mechanisms to identify the salient parts of the input. The salient features are determined by a combination of the low-level features of the stimuli as well as the feedback from the visual task being performed by the person. In this work, only the salient regions of an image frame can be refreshed while allowing the rest of the image to degrade without need for refresh.
- Third, we dynamically estimate the useful lifetime of buffered salient image data for further temporal analysis to predictively turn-off refresh for portions of the buffered data. When salient portions of the image are classified by the recognition engine based on the resulting class and the visual task to an accomplished by the vision system, the lifetime of the salient portion can be estimated. For example, if a salient region is determined to be a chair, instead of a luggage, and the task is to identify unattended luggage, that salient region can be marked as being less critical for the task and its refresh rate can be reduced (or refresh turned-off). Similarly, we can turn off the refresh for the buffered salient regions with luggage, if a person is identified next to that salient luggage later in the temporal sequence.
- Our scheme yields savings of 88.1% in refresh power and 14.8% in total power, as compared to a standard DRAM refresh scheme.

The rest of this paper is organized as follows. In Section II, we briefly describe the salient features of the vision-based system and the accelerators that are employed in it. Section III explains the considerations involved in designing the memory hierarchy for such a system. Section IV describes our proposed architecture design and highlights the additions over a baseline system. Section V enumerates our experimental results along with the performance and energy benefits that our design provides. Finally, we conclude with Section VI.

II. Background on Accelerators

In this section, we provide a brief overview on the vision-based accelerators used in our system.

A. Saliency Detection Accelerator

Visual attention has gained a lot of traction in computational neuroscience research over the past few years. Various computational models [13, 14, 15] have used low-level features to build information maps which are then fused together to form what is popularly called as a saliency map. Given an image to observe, this saliency map in essence provides a compact representation in terms of what is most important in the image. Our goal here is to localize objects in a scene and such a bottom-up saliency map proves to be very useful in a cognitive real-time system.

We use Itti's model [16] of saliency which serves as a backend to the entire cognitive pipeline. The model consists of a preprocessing Retina model, which takes the three color channels of the input image and produces luminance (I) and chrominance (RG, BY) components. These are then passed to a Visual Cortex model where the RG and BY channels are processed to produce the color (C) conspicuity map. The I channel is used to produce Intensity (I) conspicuity map and four Orientation (O) maps. The I channel from two consecutive image frames is used to produce the Flicker (F) conspicuity map and four Motion (M) maps. These 12 conspicuity maps are then combined to form what is called a saliency map.

B. Object Recognition Accelerator

Simple cells in the primary visual cortex are believed to extract local contour information from a visual scene. This information is important from the context of object recognition and detection and serves as the building block of early vision. The Scale Invariant Feature Transform (SIFT) is a well-known method used for object recognition and uses a Difference of Gaussian (DoG) approach to produce feature vectors that are invariant to translation, rotation, scale and other imaging parameters [17]. We focus our attention to a computational model called HMAX [18] 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.

We use the minimal HMAX implementation (hmin) which consists of two S (simple) layers and two C (complex) layers. The S layers (S1 and S2) perform template matching while the C layers (C1 and C2) perform max pooling. A feature vector is generate from this hierarchical model which is then passed through a classifier to classify the object. For a more detailed description of the model, we refer the reader to [18].

C. Action Recognition Accelerator

The visual cortex has two streams: the ventral pathway that is concerned with object identification and recognition and the dorsal pathway that is involved with understanding the object's motion information. HMAX was initially modeled to mimic the ventral stream. In [19], it is extended to include the dorsal stream as well, so as to classify temporal information such as human actions. Computationally, this is done by integrating spatio-temporal detectors into S1, while adding two additional layers, S3 (template matching) and C3 (pooling), which track features over time, providing time invariance to the model.

In terms of scene understanding and video analytics, an embedded visual platform can play a very critical role. For example, in video surveillance, knowing and predicting human behaviour can help avoid catastrophic events. Figure 1 shows a video frame extracted from a fixed camera setup on a tower and the resulting RoIs generated.

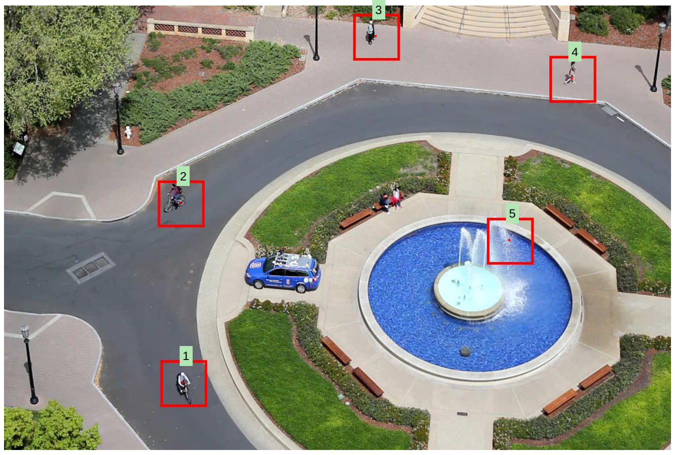

Fig. 1. Regions of Interest (RoI) generated by Saliency Model. These RoIs are classified with object labels by HMAX further in the pipeline. All objects belonging to 'person' class can be further processed to classify the action undertaken.

III. Memory Organization

Increase in computational power of most state-of-the-art systems has resulted in memory bottlenecks becoming a more and more important from the point of view of performance and energy constraints in the system. In this section, we briefly describe the role that memory design has to play in our vision-based accelerator system.

A. Motivation

Memory is an important component of almost all computer-based systems. Given the increase in cloud-computing and server farms, the scalability in terms of power and performance of memory devices has become an important design consideration. Due to its low-cost per bit, the dynamic random-access memory (DRAM) has become ubiquitous in most commercial applications. However as densities increase, the power wall is becoming an increasingly severe problem. Figure 2a) shows the increasing nature of the JDEC 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. More crucially, the refresh component in the DRAM is 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 [20], some have used retention profiling [21], while others have used software techniques [21]. In [22], the authors use a refresh pausing mechanism to exploit idle cycles in memory. In [8], Fine Grained Refresh (FGR) for DDR4 DRAM systems is analyzed. However, most of these schemes are for a general purpose multicore system. Consequently, they are not geared to exploit the data-specific aspects prevalent in our system, such as the knowledge of whether a particular region in memory needs to be refreshed or not.

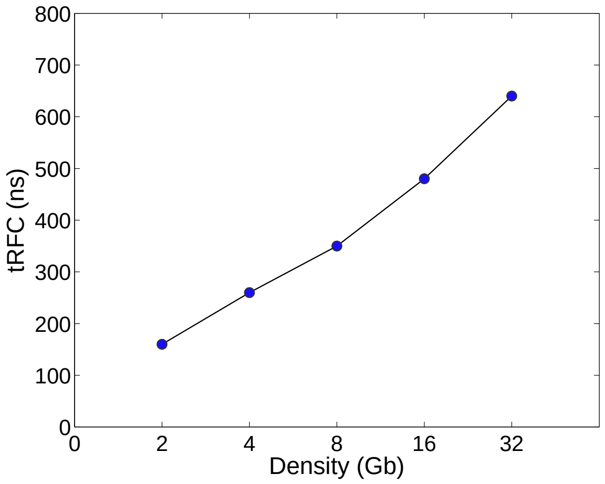

Fig. 2. a) The REF command time (tRFC) parameter is increasing with each generation [7]. The values for 16 Gb and 32 Gb devices are based on projections.

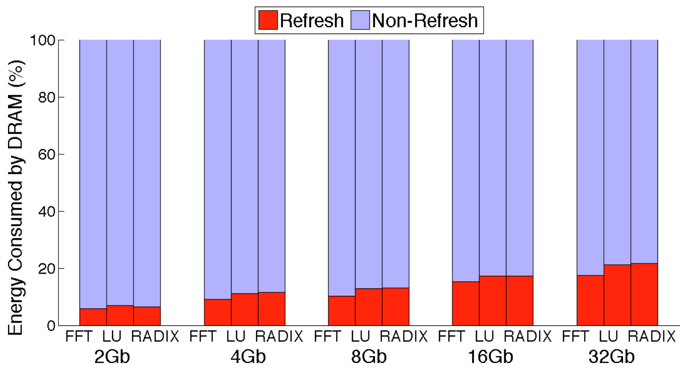

Fig. 2. b) Refresh energy distribution in Kernel computations

B. DRAM Memory Organization

DRAMs are organized into a series of hierarchical structures in order to exploit the potential for parallelism across accesses. The DRAM consists of one or more memory channels, each one of which has a dedicated memory controller. Each channel comprises of several ranks. Instructions can be issued to the DRAM at the rank level granularity. Each of these ranks are further subdivided into banks. A bank is a logical organization representing an independent memory array and consists of several rows. In general, each row is designed to hold a page of memory (2-4KB).

C. Refresh Operation in DRAMs

The defining attribute of DRAMs, in comparison to other memory technologies like Static RAMs and Hard Disks is the temporary nature of the data stored in it. Since a DRAM cell simply comprises of a single transistor-capacitor combination, the data is retained for only as long as the capacitor can hold the charge injected into it during a write operation. As a result it is essential to periodically refresh the data in each cell in order to maintain the integrity of the stored data. A refresh operation simply involves reading a row of data and writing it back from the sense amplifier. The write operation restores the charge in each DRAM cell capacitor to its original level. This refresh operation is handled by a built-in refresh controller. The memory controller periodically issues commands to carry out refresh to each rank.

The exact row to be refreshed and the corresponding bank are determined by the refresh controller. Most standard DRAM cells are architected with a 64 ms refresh window (t_{REFW}), that is each cell must be refreshed at least once every 64 ms. According to JEDEC SDRAM standards [7], a DRAM device can issue a maximum of 8192 refresh commands within this refresh window, leading to a refresh command being issued every 7.8μs (t_{REFI}). Consequently, increasing the number of rows per bank would necessitate multiple rows to be refreshed per command. The refresh controller keeps track of the address of the last row to have been refreshed so that the subsequent refresh command can enable it to issue refreshes to the next group of rows.

The major drawback of DRAM technology 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.

Each refresh command issued by the memory controller is appended to the command queue along with other read and write commands. Depending on the read and write traffic, it is possible to delay the refresh commands so as to reduce stalls due to the refresh operation preventing the read or write from being issued to the memory.

IV. System Architecture

In this section we describe in detail the system architecture, in particular the memory hierarchies along with the data mapping and refresh policies that we adopt.

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 is used to compute a *Saliency Map*. This saliency map is then used to compute the RoIs by the CPU, which are then stored on-chip in a table known as the *RoI Address Table (RAT)*. Each entry in the table corresponds to an RoI, and has a list of coordinates that specify its position in each frame. The Object Recognition accelerator classifies the objects in each ROI as belonging to a certain class. This information is also appended in the RAT. On the other hand, the Action Recognition Engine requires a series of consecutive frames to classify the exact action being carried out. Since each of these frames over the entire length of the video stored in the main memory module, there is a substantial overhead, especially in terms of refresh energy, in the baseline architecture.

In our proposed design, shown in Figure 3b), we recognize that 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 entirely. However, some accelerator operations, especially action recognition, which occurs on certain objects require a sequence of previously stored frames to be recalled. Hence, 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. We thus use a scheme called *Differential Refresh*, shown in Figure 3c), for which we describe the setup below.

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. A refresh pulse is sent every Refresh Interval (t_{ref}) time period. When a refresh is scheduled, 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 content of this register is sent to the memory controller. We maintain an on-chip CAM structure called the *RoI Address Table (RAT)* to list regions of interest (RoI) in each frame and their corresponding starting addresses. 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 Regions of Interest. We propose two schemes to support differential Refresh rates across different sections of an image.

Each image frame is interleaved across banks at page granularity utilizing Row Buffer Locality. The RAT is updated once the saliency accelerator computes the RoIs as shown in Figure 3c). After a refresh iteration, when the 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. Still, the energy overhead of refreshing non-RoI regions in rows that contain RoI is lesser than the energy overhead 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.

It needs to be noted that for completely stream-based processing, where there is a single-pass streaming access of the image, a no-refresh policy can be implemented for the entire image allowing corruption. Since a read operation inherently involves a refresh and each image row is read no more than once, this results in a no-refresh policy.

This scheme involves communication overhead between memory controller and the refresh controller since the address of the next row to be refreshed is sent after every refresh iteration. An alternative would be to assign fixed refresh rates for different sections of a bank. For example, the upper addresses of a bank can be configured for no refresh and the lower addresses for auto-refresh policy. However this would involve mapping the physical pages of a Region of Interest

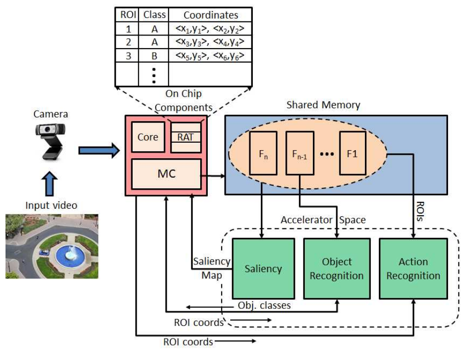
Fig. 3. a) Baseline architecture

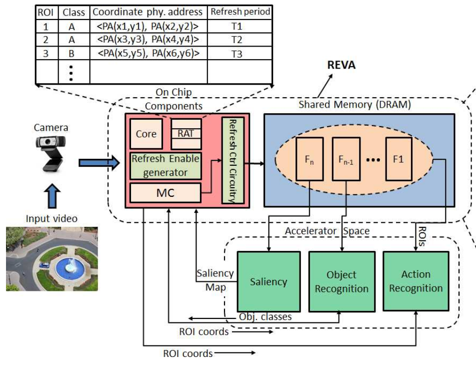
Fig. 3. b) Architecture of Proposed System

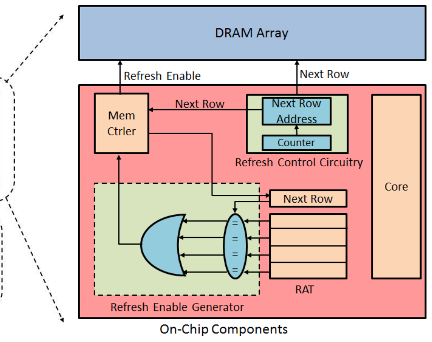
Fig. 3. c) Design of REVA block

to a specific partition in the bank incurring additional OS overhead and the scheme we propose 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 written to dedicated variable refresh frequency partitions again. Instead the saliency map computed is used to derive RoIs in input image already stored in memory and refresh frequencies are varied based on the location of RoIs.

In order to compute the overheads due to our scheme, we use McPAT 1.0 [23] 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.

B. Comparison with existing schemes

Selectively refreshing the DRAM depending on data criticality has been demonstrated earlier in [11]. However, unlike their scheme which consists of a static partitioning of the memory depending on the required refresh rate, our system is capable of dynamically allocating refresh periods to stripes of rows across banks. Consequently, by interleaving data across all banks, it is possible to control the refresh rates of each individual RoI. This also eliminates the need for the refresh regions to be re-written into a separate section of pre-allocated memory, as in the case of [11].

Figure 4 contrasts the two schemes described above. It can be observed that our scheme makes dynamic refresh allocation, which lends more flexibility and also reduces the need for redundant writes of the RoI.

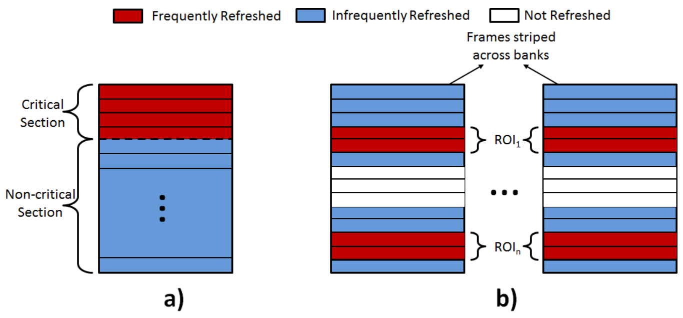

Fig. 4. a) Refresh mechanism in Flikker b) Refresh mechanism in our proposed scheme (REVA)

V. Experimental Setup and Results

A. Experimental Infrastructure

Figure 5 shows the infrastructure setup we used to carry out our evaluations. We used DRAMSim2 [24], a cycle accurate memory simulation tool, which we customized to incorporate our refresh mechanisms. The input traces to DRAMSim2 were generated using the Xilinx Vivado [25] environment in the following manner. The different accelerators, namely saliency, object recognitoin and action recognition, were implemented in Verilog and their output was used to generate parameters which was fed to a traffic generator. This traffic generator outputted the memory trace comprising of read and write memory accesses.

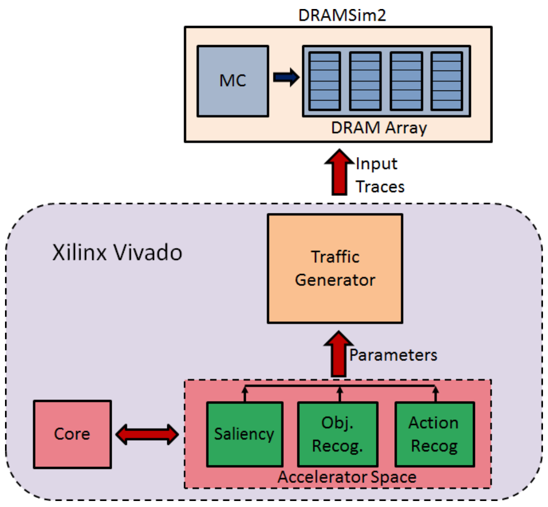

Fig. 5. Infrastructure setup

Table I shows the parameters used in our DRAM model, which we configured for simulation using DRAMSim2.

B. Results

Figure 6 illustrates the total power consumption evaluated by feeding the traces generated from Vivado into DRAMSim2 for a 8 Gb based DRAM. We evaluate our scheme 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*). The REVA architecture is

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

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, in accordance with its treatment of media-related data, the Flikker scheme 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, our scheme is able to achieve within 94% of this maximum power saving, without compromising on the accuracy that would result in a refresh-less 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.

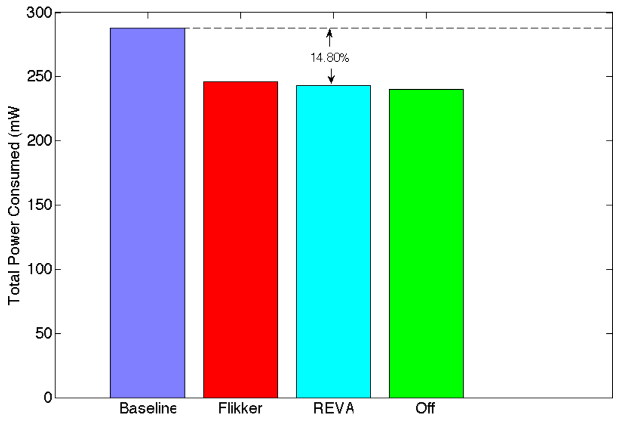

Fig. 6. Power comparisons for different schemes.

C. Sensitivity Analysis

While it can be argued that in a purely streaming embedded system, one can completely turn off refresh, we provide a compelling reason to have a variable dynamic refresh mechanism in place. There can be instances when a burst of RoIs are generated and the object recognition accelerator cannot sustain a high throughput of generating class labels. RoIs need to be buffered in the DRAM for processing at a later stage. Similarly, an object classified by HMAX may need to be evaluated further by SIFT for object matching or sub-class recognition. Then too the object needs to be stored for a period longer than the regular DRAM refresh interval. Lastly in a multi-object scenario, if a person's action is being recognized, another classified person's actions need to be buffered so as to process at a later stage. As shown in Figure 7, we evaluated different configurations of action recognition on the Weizmann dataset [26]. The results indicate that a purely streaming (no overlap of video segments) configuration affects accuracy considerably (by approximately 10%).

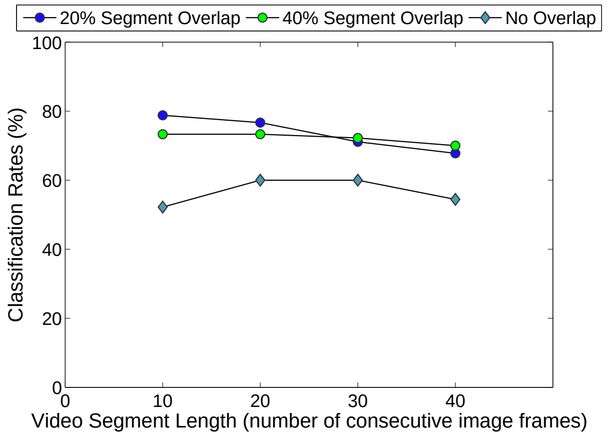

Fig. 7. Accuracy results for different configurations of video segment length and fraction of overlap between segments.

VI. Conclusion

In this paper, we demonstrated the Refresh Enabled Video Analytics (REVA) system, which is tuned to optimize energy utilization of the memory by exploiting data characteristics in vision-based applications. In this system we showed that depending on the region of interest (RoI) in an image stored in memory, the need for refresh varies from row to row across each memory bank. By adjusting refresh rates selectively depending the need for re-use of different regions of interest in video frames, we demonstrated an 88.1% improvement in refresh power and a 14.8% improvement in overall power over existing DRAM schemes.

References

[1] A. Nere, A. Hashmi, and M. Lipasti, "Profiling Heterogeneous Multi-GPU Systems to Accelerate Cortically Inspired Learning Algorithms," in *2011 IEEE International Parallel & Distributed Processing Symposium*. Ieee, May 2011, pp. 906–920.

[2] T. Chen, J. Wang, Y. Chen, and O. Temam, "DianNao : A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2014.

[3] S. Kestur, M. S. Park, J. Sabarad, D. Dantara, V. Narayanan, Y. Chen, and D. Khosla, "Emulating Mammalian Vision on Reconfigurable Hardware," in *IEEE International Symposium on Field-Programmable Custom Computing Machines*. Ieee, Apr. 2012, pp. 141–148.

[4] A. A. Maashri, M. Debole, M. Cotter, N. Chandramoorthy, Y. Xiao, V. Narayanan, and C. Chakrabarti, "Accelerating neuromorphic vision algorithms for recognition," in

Design Automation Conference. New York, New York, USA: ACM Press, 2012, p. 579.

[5] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, “Neuflow: A runtime reconfigurable dataflow processor for vision,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, June 2011, pp. 109–116.

[6] G. Carroll, Aaron Heiser, “An Analysis of Power Consumption in a Smartphone,” in *Usenix Annual Technical Conference (ATC)*, 2010.

[7] “JEDEC DDR3 and DDR4 SDRAM Standard,” 2012. [Online]. Available: http://www.jedec.org/category/technology-focus-area/main-memory-ddr3-ddr4-sdram

[8] J. Mukundan, H. Hunter, K.-h. Kim, and J. Stuecheli, “Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems,” in *International Symposium on Computer Architecture*, 2013.

[9] P. J. Nair, C.-C. Chou, and M. K. Qureshi, “Refresh pausing in dram memory systems,” *ACM Trans. Archit. Code Optim.*, vol. 11, no. 1, pp. 10:1–10:26, Feb. 2014. [Online]. Available: http://doi.acm.org/10.1145/2579669

[10] T. Ohsawa, K. Kai, and K. Murakami, “Optimizing the dram refresh count for merged dram/logic lsis,” in *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, Aug 1998, pp. 82–87.

[11] S. Liu, Pattabiraman Karthik, T. Moscibroda, and B. G. Zorn, “Flikker : Saving DRAM Refresh-power through Critical Data Partitioning,” in *Architectural Support for Programming Languages and Operating Systems*, 2011.

[12] J. Clemons, A. Jones, R. Perricone, S. Savarese, and T. Austin, “Effex: An embedded processor for computer vision based feature extraction,” in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, June 2011, pp. 1020–1025.

[13] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, Nov 1998.

[14] L. Itti and C. Koch, “Computational Modelling of Visual Attention,” *Nature Reviews Neuroscience*, vol. 2, no. 3, pp. 194–203, Mar. 2001.

[15] N. D. B. Bruce and J. K. Tsotsos, “Saliency, Attention, and Visual Search: An Information Theoretic Approach,” *Journal of Vision*, vol. 9, no. 3, pp. 5.1–24, Jan. 2009.

[16] R. J. Peters and L. Itti, “Applying computational tools to predict gaze direction in interactive visual environments,” *ACM Transactions on Applied Perception*, vol. i, no. May, pp. 1–21, 2007.

[17] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[18] J. Mutch and D. G. Lowe, “Object class recognition and localization using sparse features with limited receptive fields,” *International Journal of Computer Vision*, 2008.

[19] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, “A biologically inspired system for action recognition,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, Oct 2007, pp. 1–8.

[20] J. Stuecheli, D. Kaseridis, H. C.Hunter, and L. K. John, “Elastic Refresh: Techniques to Mitigate Refresh Penalties in High Density Memory,” in *43rd Annual IEEE/ACM International Symposium on Microarchitecture*. Ieee, Dec. 2010, pp. 375–384.

[21] J. Liu, “RAIDR : Retention-Aware Intelligent DRAM Refresh,” in *International Symposium on Computer Architecture*, vol. 00, no. c, 2012, pp. 1–12.

[22] P. Nair, C.-C. Chou, and M. K. Qureshi, “A case for Refresh Pausing in DRAM memory systems,” in *IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. Ieee, Feb. 2013, pp. 627–638.

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

[24] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, “Dramsim2: A cycle accurate memory system simulator,” *Computer Architecture Letters*, vol. 10, no. 1, pp. 16 –19, jan.-june 2011.

[25] “Vivado Design Suite,” 2014. [Online]. Available: http://www.xilinx.com/products/design-tools/vivado/

[26] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, “Actions as space-time shapes,” in *The Tenth IEEE International Conference on Computer Vision (ICCV’05)*, 2005, pp. 1395–1402.