Energy Efficiency of Opportunistic Refreshing for Gain-Cell Embedded DRAM

Binyamin Frankel, Eyal Sarfati, Davide Rossi, *Member, IEEE*, and Shmuel Wimer, *Member, IEEE*

***Abstract*— On-die memories, which are traditionally implemented by SRAM, stop functioning properly when the supply voltage is scaled down aggressively; hence, embedded DRAMs (eDRAMs) are used instead. Opportunistic refreshing was proved to eliminate the performance loss incurred by the eDRAM refreshing must. We show here that Gain-Cell eDRAM (GCeDRAM) supplemented with opportunistic refreshing consumes significantly smaller power and energy than SRAM. Analysis supported by hardware simulations demonstrate that the same design point achieves maximum performance and minimum energy. Replacement of the data memory in the ultra-low power processor PULPino from SRAM to opportunistically refreshed GCeDRAM yielded 30% energy savings in the memory, which translated into 7% savings in the entire processor.**

***Index Terms*— Low-power processors, embedded memories, memory refreshing, gain-cell.**

I. Introduction

ULTRA-LOW power processors have become a crucial feature in the internet of things (IoT) era, since they operate at a very low power-supply voltage. On-die memories, an essential component of any processor, are usually implemented by SRAM technology that cannot function properly or reliably at a very low power-supply voltage. SRAM replacement by CMOS standard-cell latches is an option, but is very expensive in terms of area and power [1]. Certain commercial products have replaced SRAM with embedded dynamic random access memory (eDRAM) technologies [2], [3], which store their data in special capacitors, but these significantly increase the manufacturing cost of the processor. They also suffer from charge destruction during the read operation that requires immediate, power expensive refreshing for data restoration. In addition, eDRAMs require mandatory periodic refreshing caused by charge leakage over time.

A bit-cell type known as the *gain-cell* (GC) overcomes these drawbacks [4], [5]. The GC eDRAM (GCeDRAM) is CMOS compatible, has low manufacturing costs, and obviates the need for the read-refresh operation. GCs can have two to five transistors. Fig. 1 shows a 2T GC. A GC still requires periodic refreshing due to charge leakage. A key advantage of GC is its two separate read and write ports, which makes it possible to design memories supporting simultaneous read and write of; rows. The GCeDRAM *data retention time* (DRT) dictates the refreshing period, which can vary from a few to hundreds of microseconds [6] depending on the bit-cell structure and the technology used. An overview of the nature of the leakage of various GCs is found in [7]. As for any DRAM, GCeDRAM refreshing blocks the system access, thus causing performance loss.

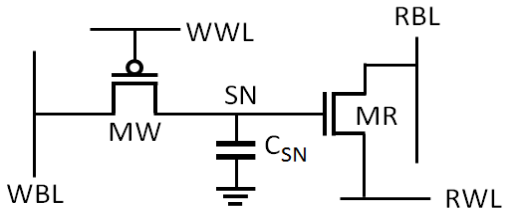


Fig. 1. Two-transistor gain-cell (2T GC). Write takes place through MW and read through MR. SN is the storage node.

In what follows we use the term row and line interchangeably. The main drawback of ordinary sequential *row-by-row* refreshing is the blockage for read/write (R/W) access of the central processing unit (CPU) during refreshing. An overview of the main refreshing algorithms for DRAM/eDRAM can be found in [8]. Refreshing uses the same ports as for CPU access. Due to GCeDRAM's separate read and write ports, the read and write of two different rows can take place simultaneously. Since refreshing requires a read and write operation, a sequence of contiguous refreshing can be partially overlapped to deliver an effective refreshing rate of one cycle per row.

One way to overcome the performance degradation problem is to use an *opportunistic* refreshing algorithm that has the advantage of not intervening in the normal memory access [9]. Rather, refreshing takes place concurrently with CPUR/W operations. Whereas performance usually conflicts with power and energy consumption, this work answers the question of whether this is the case for opportunistic refreshing. The main contribution of this study are:

1. Development of an accurate analytical energy model that agrees with real hardware power simulations.

Manuscript received 3 November 2022; revised 21 December 2022; accepted 22 December 2022. This work was supported in part by the Israel Chief Scientist under the GenPro Consortium of the MAGNET Program. This article was recommended by Associate Editor P. K. Meher. *(Corresponding author: Shmuel Wimer.)*

Binyamin Frankel and Shmuel Wimer are with the Engineering Faculty, Bar-Ilan University, Ramat-Gan 52900, Israel (e-mail: binyamin.frankel@gmail.com; wimers@biu.ac.il).

Eyal Sarfati is with the Electrical Engineering Department, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: eyal.sarfati@gmail.com).

Davide Rossi is with the Department of Electrical, Electronic and Information Engineering (DEI), University of Bologna, 40023 Bologna, Italy (e-mail: davide.rossi@unibo.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2022.3231866.

Digital Object Identifier 10.1109/TCSI.2022.3231866

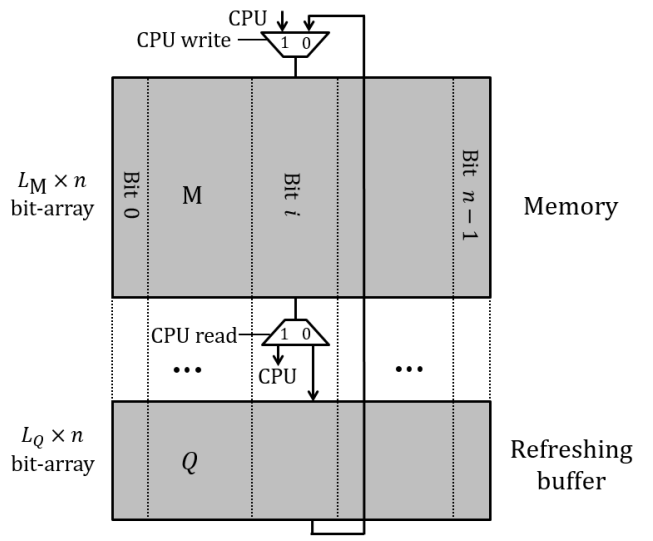


Fig. 2. A memory array and its associated refreshing buffer.

2. A proof that the same design point achieves both maximum performance and minimum energy.
3. Showing by a real processor design that for the same Vdd, opportunistically refreshed GCeDRAM consumes considerably smaller power and energy of 30% than SRAM does.

The rest of the paper is organized as follows. Section 2 presents briefly opportunistic refreshing background. Section 3 studies the relationship between optimal performance and energy efficiency. Section 4 analyzes energy consumption followed in Section 5 by optimization of the memory design parameters for minimum energy. Section 6 presents experimental results and Section 7 draws conclusions.

II. Opportunistic Refreshing

We use M to denote the memory, which for our purposes is the smallest unit maintaining its own refreshing. This can be a bank or a smaller unit in a memory, depending on the memory physical architecture. Opportunistic refreshing in the case where M access is either for read or for write (denoted as the R/W cycle) was discussed in [9]. This is the R/W regime in ultra-low power processor architectures [10] which we used for this energy study.

Fig. 2. Illustrates how opportunistic refreshing is working. The upper MUX and lower deMUX transfer the refreshed lines between the memory M and the refreshing buffer Q. Refreshing takes place sequentially, line-by-line, *simultaneously* and in coordination with the CPU access to M. 'Simultaneously' means that within the same clock cycle, while the CPU is reading from or writing into M, the refreshing performs a counter operation; i.e., writing into or reading from a refreshed line of M.

Proper refreshing must guarantee that the duration between two successive refreshes of any line does not exceed the DRT, denoted by N_{DRT}, and measured in clock cycles. It is obtained by dividing DRT by the clock cycle duration. The refreshing of most dynamic memories requires a buffer (FIFO queue). The content of a weakened line is first copied into the buffer and then written back to strengthen and restore the line's contents.

Opportunistic refreshing takes place sequentially row-by-row, but not necessarily contiguously in time. Refreshing stalls occur in CPU R-cycles if the refreshing buffer is empty or in W-cycles if the refreshing buffer is full.

Let N_{RR} denote the *refreshing period*, measured in clock cycles. It must ensure the timely refresh of all the L_{M} rows of M for any R/W access pattern. The larger the N_{RR}, the greater the likelihood that a higher portion of the refreshing will be concealed opportunistically and hence a smaller portion (if at all) will require enforced completion. Moreover, maximizing N_{RR} minimizes the ratio of the refreshing energy to the energy consumed by executing the program's code instructions. The longest N_{RR} that guarantees proper refreshing is [9]:

$$N_{\mathrm{RR}} = \left\lfloor \frac{N_{\mathrm{DRT}} + L_{\mathrm{M}}}{2} \right\rfloor. \tag{1}$$

Let the CPU R-cycle and W-cycle occur with respective probabilities $0 \le \mu \le 1$ and $\lambda = 1 - \mu$. This memory access model is compatible with the PULPino processor [10] used in this study, which supports either read or write memory access by the CPU. The $\mu + \lambda = 1$ assumption is stressed and pessimistic, since it means that the CPU accesses M on every cycle. In reality however, there are usually quite a lot of memory idle cycles where the CPU is executing other instructions, thus enabling refreshing read and write of two successive lines in the same cycle.

Let $\rho = \lambda/\mu$ be the CPU write-to-read probabilities ratio and denote by L_Q the number of Q entries (buffer lines). Depending on the CPU R/W patterns, it may happen that opportunistic refreshing is insufficient, a case when a refreshing completion is enforced, blocking M for CPU access. The performance loss $0 \le \gamma \le 1$ thus occurred is [9]:

$$\gamma = \begin{cases} 0, & \text{if} \dfrac{\rho\left(1-\rho^{L_Q}\right)}{(1+\rho)\left(1-\rho^{L_Q+1}\right)} N_{\mathrm{RR}} \ge L_{\mathrm{M}} \\ \dfrac{L_{\mathrm{M}}(1+\rho)\left(1-\rho^{L_Q+1}\right) - N_{\mathrm{RR}}\rho\left(1-\rho^{L_Q}\right)}{N_{\mathrm{RR}}\left(1-\rho^{L_Q+2}\right)}, & \text{otherwise} \end{cases} \tag{2}$$

Equation (2) captures the memory size L_{M}, the CPU write-to-read probabilities ratio ρ and the refreshing buffer capacity L_Q. The condition for $\gamma = 0$ implies that opportunistic refreshing suffices. Otherwise $0 < \gamma$ and refreshing completion enforcement must take place.

An opportunistic refreshing system was implemented in [9] and replaced the data SRAM of an ultra-low power RISC-V processor called PULPino [10] by GCeDRAM. Fig. 3 shows the resulting performance, and exhibits similarities of real hardware simulations to the analytical model (2). The symmetry of the surface around $\lambda = \mu = 0.5$ stems from $\gamma(\rho) = \gamma\left(1/\rho\right)$.

When $N_{\mathrm{RR}} = L_{\mathrm{M}}$ refreshing must take place at each cycle; hence, the CPU access to the memory is always blocked and $1 - \gamma = 0$. Smaller L_{M} leaves more cycles for useful CPU access and performance increases accordingly. Performance also depends on the R/W probabilities, $0 \le \mu \le 1$ and

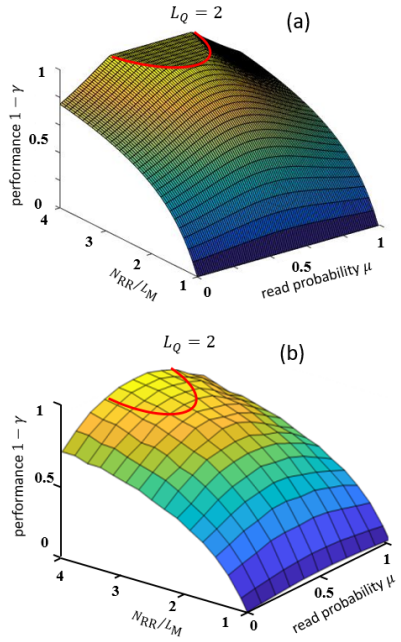


Fig. 3. Memory access performance: analytical model (a), hardware simulation (b) [9].

$\lambda = 1-\mu$, respectively. For a very small $\lambda$the refreshing buffer Q will often be empty, so it cannot supply rows to write back into M upon R-cycles. Oppositely and symmetrically, a very small μ will often saturate$Q$, avoiding reading M's lines into Q upon W-cycles. These characteristics are illustrated by the decline of the performance surface towards the margins at $\mu = 1$ and $\mu = 0$.

III. The Relationship Between Optimal Performance and Energy Efficiency

This work studies the division of an on-die, near CPU memory (e.g. L1 cache) into smaller physical units. Whereas the microarchitecture dictates its whole size L beforehand, its division into smaller refreshable units is subject to optimization. Since each unit M has its own refreshing hardware overhead, this sole consideration favors to decrease their total number $L/L_{\mathrm{M}}$by increasingL_{M}. Fig. 3a shows that for a sufficiently large $N_{\mathrm{RR}}/L_{\mathrm{M}}$ and for some symmetric interval around $\mu = \lambda = 1/2$ there is no performance loss, as shown by the red curves circumscribing the flat surfaces where$\gamma = 0$. This sole consideration favors to decreaseL_{M}. The following discussion optimizes these conflicting trends.

Since we are interested in keeping L_{M} as large as possible while maintaining maximal performance, the red curve in Fig. 3a defines optimal design points in the following sense. GivenN_{DRT}, L_Q, andμ, if maximum performance ($\gamma = 0$) is achievable, L_{M} should be selected on this curve. Other than the flat surface, any combination of L_{M} and μ uniquely defines the memory access performance.

The above L_{M} choice aims at reducing the refreshing control hardware, but only a small portion of the refreshing energy is associated with it. The dynamic R/W operations of the memory and refreshing buffer consume most of the energy. It is interesting to reveal how the optimal energy comply with the optimal performance curve. As shown subsequently, the projections on the (L_{M}, μ) plane of the maximum performance curve in Fig. 3a and the minimum refreshing energy curve in Fig. 7 are identical. In other words, the same design points yield maximum performance and minimum energy.

Let the nominal execution of a given program require Nclock cycles. Nominal means that N does not include any additional clock cycles that may be required for refreshing completion enforcement. The effective duration of the program is thus$N/(1-\gamma)$, where γ is the performance loss expression (2). The effective duration $N/(1-\gamma)$ is divided into refreshing periods of length N_{RR} each. Therefore, there are $\left[N/(1-\gamma)\right]/N_{\mathrm{RR}}$ refreshing periods, where each the L_{M} lines of a refreshable memory unit M are refreshed. Since a refresh requires a read and write operation from and into both M and$Q$, there are a total of $4L_{\mathrm{M}}$ dynamic refreshing operations. Recall that the refreshing occurs simultaneously in all L/L_{M} M units; thus, the total dynamic energy consumed during the execution of the program, as measured in terms of dynamic operations, is

$$E^{\mathrm{dyn}}_{\mathrm{refresh}} \propto \frac{N}{(1-\gamma)\,N_{\mathrm{RR}}} \times \frac{L}{L_{\mathrm{M}}} \times 4L_{\mathrm{M}} = \frac{4NL}{(1-\gamma)\,N_{\mathrm{RR}}}. \quad (3)$$

Substitution of (1) in (3) yields

$$E^{\mathrm{dyn}}_{\mathrm{refresh}} \propto \frac{8NL}{(1-\gamma)\,(N_{\mathrm{DRT}} + L_{\mathrm{M}})}. \quad (4)$$

As mentioned above, the performance loss expression (2) depicted in Fig. 3a has two regions: zero and non-zero loss, separated by the red curve. We first consider the flat surface where $\gamma = 0$, and substitute it into (4). There is

$$E^{\mathrm{dyn}}_{\mathrm{refresh}}\,(L_{\mathrm{M}}, \gamma = 0) \propto \frac{8NL}{(N_{\mathrm{DRT}} + L_{\mathrm{M}})} \quad (5)$$

Since L_{M} appears in the denominator, the dynamic refreshing energy minimization requires to increase L_{M} (decrease$N_{\mathrm{RR}}/L_{\mathrm{M}}$), hence approaching the red curve in Fig. 3a.

Other than $\gamma = 0$, for every value of N_{DRT}, ρ, and L_Q the performance loss $\gamma > 0$ is uniquely defined by L_{M}, and should be determined such that the dynamic refreshing energy in (4) is minimized. Substituting into (4) the expression for $\gamma > 0$ in (2) yields after some manipulations

$$E^{\mathrm{dyn}}_{\mathrm{refresh}}\,(L_{\mathrm{M}}, \gamma > 0) \propto \frac{8NL}{L_{\mathrm{M}}\left(1 + \frac{b-2a}{c}\right) + N_{\mathrm{DRT}}\left(1 + \frac{b}{c}\right)}, \quad (6)$$

where $a = (1+\rho)\left(1-\rho^{L_Q+1}\right), b = \rho\left(1-\rho^{L_Q}\right)$ and $c = \left(1-\rho^{L_Q+2}\right)$. Here again, L_{M} appears in the denominator. Hence the sign of the term $1 + (b-2a)/c$ determines the growth direction of$E^{\mathrm{dyn}}_{\mathrm{refresh}}\,(L_{\mathrm{M}}, \gamma > 0)$. Substituting the expressions of$a$, b, and c in $1 + (b-2a)/c$ yields a function$f\,(\rho)$. Recall that since$\rho = \lambda/\mu$, where $0 \le \mu \le 1$ and $\lambda = 1-\mu$, there is $0 \le \rho \le \infty$. It can easily be verified that $f\,(\rho) = f\,(1/\rho)$; hence, it is sufficient to find the sign of $f\,(\rho)$ for $0 \le \rho \le 1$. There is $f\,(0) = -1$ and some tedious derivation shows that$df/d\rho \le 0$, asserting that $f\,(\rho) < 0$ for $0 \le \rho \le 1$.

TABLE I
SRAM AND GC BIT-CELL COMPARISON

Bit-cell type		SRAM	2T GC	3T GC	4T GC
Area [μm^2]		0.352	0.152	0.186	0.23
Vdd [mV]		700	700	700	700
Leakage [pW/bit]	@27°C	9.07	3.27	3.29	3.33
	@85°C	166.5	56.3	56.9	58.1
Data retention time [µsec]	@27°C	∞	32	51	1691.4
	@85°C	∞	3.1	4.14	154.95

Consequently, the decrease of L_{M} in (6) reduces the dynamic refreshing energy. In terms of Fig. 3a,it means climbing along the surface towards the red curve, where the $\gamma > 0$ scenario stops and turns into the $\gamma = 0$ scenario discussed earlier. To conclude, for both $\gamma = 0$ and$\gamma > 0$, the zero performance loss curve in Fig. 3a exhibits minimum refreshing energy consumption. Section 5 shows this correlation also by hardware simulations.

IV. POWER ANALYSIS

Bit-cell dynamic power includes a portion proportional to L_{M}resulting from the R/W bit-lines. Section 3 showed by back-of-the-envelope arguments that L_{M} yielding maximum performance also minimizes the refreshing energy. Below we apply a more rigorous analysis to derive an expression of the energy dependence onL_{M}, based on extracted bit-cell energy parameters. We then present a back annotated gate-level power simulations of PULPino [11] to validate the model.

Table I compares the leakage power characteristics of 28nm SRAM to 2T, 3T and 4T GC bit-cells [5]. The GC choice tradeoff is clear in terms of area versus DRT. More transistors increase the DRT, thus decreasing the refreshing overhead, yielding higher CPU performance. Our design used a 4T GC bit-cell. Note that although its DRT is long, this is an average measured in silicon across a large sample, so a conservative DRT is used. The GCeDRAM design must therefore ensure wide DRT safety margins. Note as well that the leakage is independent of the number of transistors in the bit-cell. This follows since in all GCs the single pass transistor connecting the write bit-line to the storage node (MW in Fig. 1) produces most the leakage.

Fig. 2 shows that the refreshing buffer is also a GCeDRAM memory array comprising L_Q lines. Hence the energy consumption analysis should account for M and$Q$. In terms of leakage, there is no difference between the bits of M and$Q$, so we consider them as an array of $L_{\mathrm{M}} + L_Q$ lines. Their dynamic power differs though. This follows from the longer bit-lines of M compared to those of$Q$, in proportion with$L_{\mathrm{M}} \gg L_Q$, and the different numbers of R/W operations stemming from their different functions.

Let us amortize the CPU R/W operations occurred in M during N_{RR} period. The probability of an R/W-cycle is typically 0.3 [12], thus yielding $0.3N_{\mathrm{RR}}$R/Woperations, where the typical value of N_{RR} is a few thousand. In addition, each of M's L_{M} words is refreshed once during the N_{RR} period, involving the read and write of both M and$Q$, thus yielding a total of $4L_{\mathrm{M}}$R/W operations. The bit-array in Fig. 2 yields therefore $4L_{\mathrm{M}} + 0.3N_{\mathrm{RR}}$ word toggling.

A. Per-Bit Energy Consumption

The experimental results presented in Section 6 are based on gate-level dynamic power simulations using the parameters of Table. It provides the per-bit switching energies of 28nm SRAM and 4T GC bit-cells for reads and writes at 0.7Volts and 85°C [5]. It lists the GCeDRAM and SRAM per-bit energies for arrays of $8, 256, 512$ and 1024 lines. Note that though read and write operations access a single word, the power consumption is primarily affected by the length of the bit-lines extending along the entire array. Bit-line energy grows linearly with the increase in array size.

The SRAM per-bit read energy consists of its word-line charging energy from 0 to 1. In addition, the two bit-line energy grows linearly with increases in array size and is identical in all transitions. Identity follows from the two bit-lines which after being pre-charged, always transit oppositely at the end of a read cycle.

The per-bit write energy consists of the same word-line charging energy as in read, plus its internal storage node write energy. The latter depends on whether or not the bit-cell toggles its value. In addition, the two bit-lines either remain at the same value or toggle their value. We assume that on average one line is charged from 0 to 1, whereas the other stays at 0.

With respect to the bit-cell energies in GCeDRAM, recall that it has separate bit-lines for read and write operations. The read energy depends on whether 0 or 1 is read. For 0, the voltage of the pre-charged bit-line decreases by some amount, so the next read will have to re-charge it. In 1, there is no such energy consumption and next pre-charge has no cost. The remaining energy of the read follows from the word-line load charging, and is similar to 0 and 1.

The write energy has the same word-line load charging energy in all transitions. If the storage node of the bit-cell toggles its value, it consumes additional write energy. Here we are being conservative, by assuming that toggling in both directions consumes energy, although only charging from 0 to 1 consumes real energy. For the bit-line portion, we are also conservative, and assume that the bit-line consumes energy regardless of whether the written value is the same or the opposite of its previous value. The write energy growth with the array size is similar to that of SRAM.

Denote by $p_{\mathrm{R}}^{0\to0}$, $p_{\mathrm{R}}^{0\to1}$, $p_{\mathrm{R}}^{1\to0}$, and $p_{\mathrm{R}}^{1\to1}$ the dynamic read power of a bit-cell for the $0 \to 0$, $0 \to 1$, $1 \to 0$ and $1 \to 1$ transitions, respectively, as specified in Table II. $p_{\mathrm{W}}^{0\to0}$, $p_{\mathrm{W}}^{0\to1}$, $p_{\mathrm{W}}^{1\to0}$, and $p_{\mathrm{W}}^{1\to1}$ are defined similarly for write. Assuming random words for read and write, each bit has same probability of 0 or 1. The respective average bit read and write power is

$$p_{\mathrm{R}} = 0.25\left(p_{\mathrm{R}}^{0\to0} + p_{\mathrm{R}}^{0\to1} + p_{\mathrm{R}}^{1\to0} + p_{\mathrm{R}}^{1\to1}\right),$$
$$p_{\mathrm{W}} = 0.25\left(p_{\mathrm{W}}^{0\to0} + p_{\mathrm{W}}^{0\to1} + p_{\mathrm{W}}^{1\to0} + p_{\mathrm{W}}^{1\to1}\right). \quad (7)$$

TABLE II
SWITCHING ENERGY CONSUMPTION OF SRAM AND 4T GC BIT-CELLS

		SRAM [10^{-15} Joule]					4T GC [10^{-15} Joule]				
Lines	Bit energy	$0\to0$	$1\to1$	$0\to1$	$1\to0$	Ave.	$0\to0$	$1\to1$	$0\to1$	$1\to0$	Ave.
1 (bit-cell)	Read	0.140				0.140	0.073	0.051		0.073	0.062
	Write	0.174		0.311		0.243	0.269		0.394		0.332
8	Read	0.316				0.316	0.225	0.051		0.225	0.138
	Write	0.585		0.722		0.654	0.625		0.750		0.687
256	Read	6.56				6.56	5.62	0.051		5.62	2.83
	Write	15.1		15.3		15.2	13.2		13.3		13.3
512	Read	13.0				13.0	11.3	0.051		11.3	5.62
	Write	30.2		30.3		30.2	26.2		26.3		26.3
1024	Read	25.9				25.9	22.3	0.051		22.3	11.3
	Write	60.2		60.4		60.3	52.3		52.3		52.3

The gray columns in Table II show the per-bit power p_{R} and p_{W}.

Note also the bit-array peripheral circuits that comprise the decoders for word access, the input drivers for write and the output drivers for read). These are an integral part of the bit-array and may have a secondary effect on how to set L_{M} for minimum dynamic power. The analysis ignores them for the sake of its simplicity. Nevertheless, we will account them later in the experimental results.

It follows from Table II that the dependence of the per-bit average dynamic read power p_{R} on L_{M} satisfies

$$p_{\mathrm{R}} = p_{\mathrm{R}}^{\mathrm{cell}} + p_{\mathrm{R}}^{\mathrm{line}} = \alpha_{\mathrm{R}} + \beta_{\mathrm{R}} L_{\mathrm{M}}, \tag{8}$$

where the parameter α_{R} is the average dynamic power consumed by the bit-cell alone, and $\beta_{\mathrm{R}}$is a coefficient of the average bit-line power. Similarly, the bit-cell write power p_{W} satisfies

$$p_{\mathrm{W}} = p_{\mathrm{W}}^{\mathrm{cell}} + p_{\mathrm{W}}^{\mathrm{line}} = \alpha_{\mathrm{W}} + \beta_{\mathrm{W}} L_{\mathrm{M}}, \tag{9}$$

where the coefficients are defined analogously for a write operation.

We further characterize the average per-bit dynamic power in M, where averaging takes place across a complete refreshing period of N_{RR} cycles, thus reflecting the energy consumption. Let L be size of the entire memory, divided into L/L_{M}refreshable units of size L_{M} each. Let μ andλ be CPU R/WM-cycle probabilities defined before. Since the PULPino architecture supports only R/WM-cycles and there are also memory idle cycles, there is $\mu + \lambda \leq 1$. The analysis below, however, does not depend on this assumption and holds equally for processors supporting R + WM-cycles, a case where $\mu + \lambda > 1$ is possible. Specific values of μ and λ for a test bench of 11 known programs are presented in Section 6. The CPU access probability to M is therefore

$$\mu_{\mathrm{M}} = \mu \frac{L_{\mathrm{M}}}{L}, \quad \lambda_{\mathrm{M}} = \lambda \frac{L_{\mathrm{M}}}{L}. \tag{10}$$

The per-bit average power consumption of M due to the workload is

$$p_{\mathrm{CPU}}^{\mathrm{dyn\_M}} = \mu_{\mathrm{M}} p_{\mathrm{R}} + \lambda_{\mathrm{M}} p_{\mathrm{W}}. \tag{11}$$

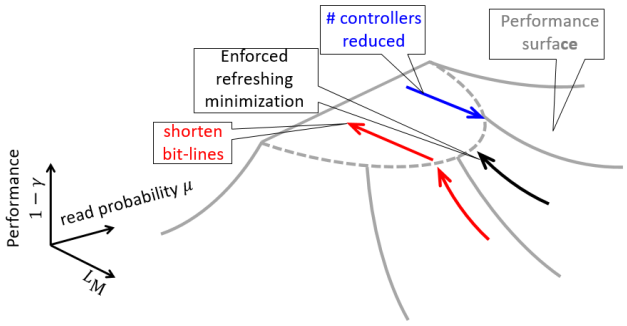


Fig. 4. Variations in memory unit size to minimize energy.

Substitution of (8), (9) and (10) into (11) yields the following per-bit dynamic power consumption due to the workload

$$p_{\mathrm{CPU}}^{\mathrm{dyn\_M}} = \mu \frac{L_{\mathrm{M}}}{L} (\alpha_{\mathrm{R}} + \beta_{\mathrm{R}} L_{\mathrm{M}}) + \lambda \frac{L_{\mathrm{M}}}{L} (\alpha_{\mathrm{W}} + \beta_{\mathrm{W}} L_{\mathrm{M}}). \tag{12}$$

All the parameters in (12) are technology and workload dependent, except forL_{M}, which is a design choice, and hence should be optimized.

A similar analysis applies to the Q array in Fig. 2. Whereas M's read and write probabilities are workload dependent, the number of Q reads from M and Q writes back into M during the N_{RR} period are L_{M} each. Hence, for the N_{RR} period there is

$$\mu_Q = \lambda_Q = \frac{L_{\mathrm{M}}}{N_{\mathrm{RR}}}. \tag{13}$$

The following is thus the per-bit dynamic power consumption of Q due to refreshing

$$p_{\mathrm{refresh}}^{\mathrm{dyn\_}Q} = \frac{L_{\mathrm{M}}}{N_{\mathrm{RR}}} (\alpha_{\mathrm{R}} + \beta_{\mathrm{R}} L_Q) + \frac{L_{\mathrm{M}}}{N_{\mathrm{RR}}} (\alpha_{\mathrm{W}} + \beta_{\mathrm{W}} L_Q). \tag{14}$$

The refreshing power is consumed not only by Q but by M as well, and stems from the L_{M} reads of Q from M and the L_{M} writes back from Q into M, yielding the following refreshing power consumption

$$p_{\mathrm{refresh}}^{\mathrm{dyn\_M}} = \frac{L_{\mathrm{M}}}{N_{\mathrm{RR}}} (\alpha_{\mathrm{R}} + \beta_{\mathrm{R}} L_{\mathrm{M}}) + \frac{L_{\mathrm{M}}}{N_{\mathrm{RR}}} (\alpha_{\mathrm{W}} + \beta_{\mathrm{W}} L_{\mathrm{M}}). \tag{15}$$

By summing the per-bit dynamic power components in (12), (14) and (15), we obtain the power consumed by a refreshable unit M:

$$p^{\mathrm{dyn}} = p_{\mathrm{CPU}}^{\mathrm{dyn\_M}} + p_{\mathrm{refresh}}^{\mathrm{dyn\_}Q} + p_{\mathrm{refresh}}^{\mathrm{dyn\_M}}. \tag{16}$$

V. REFRESHABLE UNIT SIZE OPTIMIZATION FOR ENERGY MINIMIZATION

Fig. 4 depicts the performance surface discussed in Section 2 and shown in Fig. 3. Let us consider the zero performance loss flat area enclosed by the dashed curve. Section 2 concluded that if the workload is such that refreshing is completely opportunistic with no performance loss it is worth setting L_{M} on the dashed curve. Fig. 4 illustrates it by the blue arrow direction. Section 3 concluded by a back-of-the-envelope energy calculation that the dashed curve also represents the optimal energetic design choices. Whenever

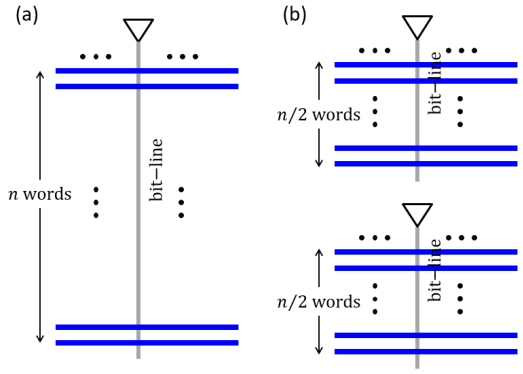


Fig. 5. Reducing dynamic load by shortening bit-lines.

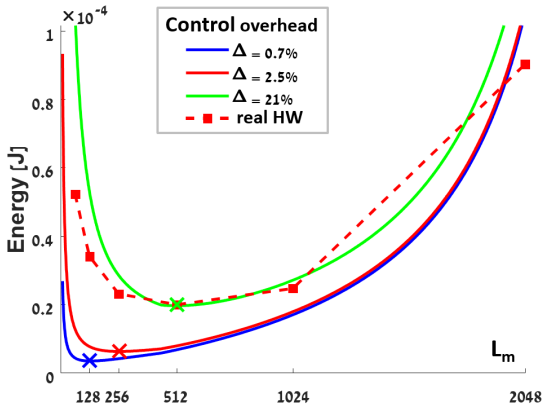


Fig. 6. The total energy of the entire memory compared to hardware simulations.

opportunistic refreshing is insufficient and there is performance loss, minimization of the extent of enforced refreshing completion favors smallL_{M}. Fig. 4 illustrates it by the climbing black arrow.

L_{M} affects the energy consumption not only by the implied number of refreshing operations, but also by the physical size of the bit-lines in the array, as expressed by the term $p_{\mathrm{R}}^{\mathrm{line}}$in (8) and $p_{\mathrm{W}}^{\mathrm{line}}$ in (9). To illustrate this impact, consider the example of the n-word memory depicted in Fig. 5. Assume that every word performs an R/W operation in turn. How much dynamic energy is consumed? This depends on the number of M units into which the entire memory is divided. Fig. 5(a) shows a one-unit implementation whereas Fig. 5(b) is a two-unit. In the one-unit, every R/W operation charges a line of twice the length as in a two-unit. Since the total number of R/W operations is independent of whether it is one or two units, whereas the bit-line energy is indeed dependent, two units cuts this energy by a factor of two. Table II shows that the bit-line energy is dominant; hence, the bit-line energy consumption favors smallL_{M}. Fig. 4 illustrates it by the red arrows.

The conflicting trends of L_{M} shown in Fig. 4 call for an energy analysis combining all these factors. Refreshing control logic (MUX, deMUX in Fig. 2 and counters not shown) also consume energy. This is added by a term Δ which is a percentage of the refreshing power, also including the static energy presented in Table I. Recall that the product of power by time duration (clock cycles) yields energy, so henceforth we turn p^{dyn} into energy. Given that a word of M comprises 32 bit-cells, the total dynamic energy of the entire memory comprising L/L_{M}M units, operating during the entire program that takes $N/(1-\gamma)$ clock cycles, is as in (17), shown at the bottom of the page. Expression (a) follows from the workload, and only involves memory accesses as dictated by the underlying running code. It is independent of the performance loss $0 \leq \gamma < 1$ incurred due to enforcement of refreshing completion. Expression (b) follows from the refreshing required during the entire running time$N/(1-\gamma)$.

The parameters in (17) are technology, workload and architecture dependent, except for the size of the bit-arrays L_{M} and L_Q, which are design choices. Since $L_Q \ll L_{\mathrm{M}}$ we can consider L_Q as a given which is not subject to energy optimization. We are simply left withL_{M}, and denote the total energy by $E(L_{\mathrm{M}})$ to emphasize its sole dependence on L_{M}.

Fig. 6 plots in solid lines $E(L_{\mathrm{M}})$ and the corresponding average power at 0.7Volts and 85°C. We use $T_{\mathrm{DRT}} = 25\mu Sec$, reflecting an average of various GCeDRAM implementations (see Table I in [5]). The operating frequency is 100MHz, yielding $N_{\mathrm{DRT}} = \left(25 \times 10^{-6}\right) \times \left(100 \times 10^{6}\right) = 2500$. The memory size in the PULPino [10] architecture is 32KB, which with 32 bits per line (4B) yields $L = 8192$ word lines. For the energy calculations, we assumed a memory stressed program comprising only R/W instructions. The program comprised 10^6 random R/W instructions with a CPU read probability of $\mu = 0.67$ and a write probability of $\lambda = 0.33$, where R/W ratio of 2:1 is commonly used [13]. Energy calculation used the per-bit power parameters of Table II.

The different curves in Fig. 6 show the impact of the control power overheads $\Delta = 0.7\%$, $\Delta = 2.5\%$ and $\Delta = 21\%$ of the refreshing power in (17) for a single refreshable unit M. It clearly shows that the division of the memory into too many M units with small L_{M} cancels out the energy savings obtained by the high likelihood of complete opportunistic refreshing and shortening the bit-lines. Hence, given the above refreshing control overhead there is an optimal L_{M} design point of minimum energy $L_{\mathrm{M}} = 128$, $L_{\mathrm{M}} = 256$, and $L_{\mathrm{M}} = 512$ marked on the corresponding curves.

VI. Experimental Results

Experimental results illustrate the validity of the analytical energy models presented above. They show that the energy consumed by the gate-level design behaves similarly to the model. It also compares the energies of opportunistically

$$E^{\mathrm{dyn}} = 32\left[\underbrace{N\frac{L}{L_{\mathrm{M}}}p_{\mathrm{CPU}}^{\mathrm{dyn\_M}}}_{(a)} + \underbrace{\frac{N}{1-\gamma}\frac{L}{L_{\mathrm{M}}}\left(p_{\mathrm{refresh}}^{\mathrm{dyn\_}Q} + p_{\mathrm{refresh}}^{\mathrm{dyn\_M}}\right)(1+\Delta)}_{(b)}\right]. \tag{17}$$

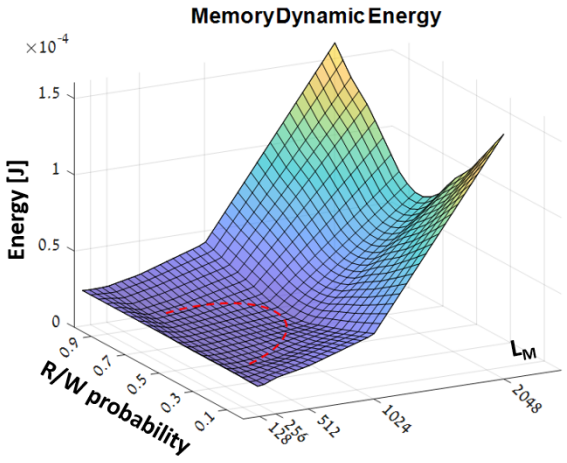


Fig. 7. Dynamic energy consumption of the data memory w/o refreshing controllers.

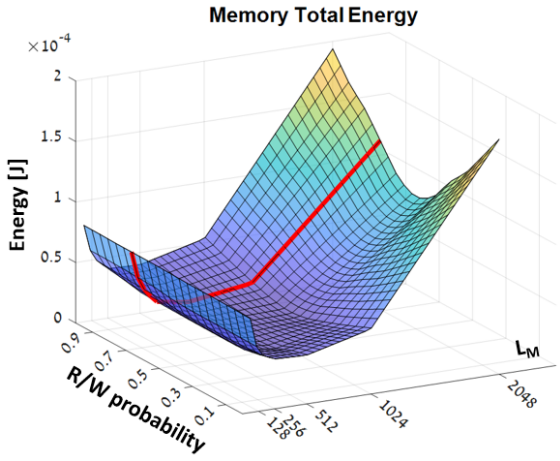


Fig. 8. Total energy consumption of the entire data memory.

refreshed GCeDRAM with a single-port SRAM implementation. For this purpose, the PULPino Verilog RTL was compiled with the Cadence design suite to gate-level with the 28nm TSMC library, supplemented with GCeDRAM bit-cell lib files based on [5].

Gate-level implementation targeted 100MHz clock frequency. The refreshing controller assumed DRT $= 25\mu$ sec, thus $N_{\mathrm{DRT}} = 2500$. The size of the refreshing buffer Q was $L_Q = 8$. The PULPino 32KB data memory was implemented with $L_{\mathrm{M}} = 2048, 1024, 512,\ 256, 128$ and 64, corresponding to 4, 8, 16, 32, 64 and 128 self-refreshable banks, respectively.

Each L_{M} was simulated with 11 programs of 10^6 pure, random R/Winstructions. Each program had a different R/W probability ratio, satisfying $\lambda + \mu = 1$. This yielded altogether 66 programs, 11 for each L_{M} size. The total runtime of every program was multiplied by the power obtained from the simulator [11], yielding the corresponding energy. Note that depending on the R/W random patterns, the execution of a program may take more than the nominal 10^6 cycles if opportunistic refreshing is insufficient and refreshing completion is enforced.

Fig. 7 shows the dynamic energy consumed by the arrays of the data memory, including their decoders and sense amplifiers. Equations (5) and (6) in Section 3 showed analytically that the optimal performance curve in Fig. 3a and Fig. 4 is also the minimum dynamic energy curve (controllers excluded). The corresponding red dashed line curve depicted in Fig. 7 is consistent with this analysis. Though the write energy is larger than the read energy as specified in Table II, the surface shows some symmetry around the R/W probabilities of $\lambda = \mu = 0.5$. This follows from the dominance of the clock energies consumed by both read and write but not accounted for in Table II.

Fig. 8 shows the total energy of the memory and the refreshing controllers, including all the underlying logic and the refreshing buffer in Fig. 2. It is shown that the minimum energy is achieved for $L_{\mathrm{M}} = 512$. The steep increase in energy at small L_{M} follows from the high number of controllers, in agreement with the blue arrow in Fig. 4. Another energy

TABLE III
POWER COMPARISON OF GCeDRAM OPPORTUNISTIC REFRESHING TO SRAM

Test Name	Total Instructions	CPU Read [%]	CPU Write [%]	Memory Power Reduction [%]	PULPino Power Reduction [%]
Bubblesort	55680	20.3	18.4	32.81	7.67
EDN	43111	26.7	2.5	35.44	7.90
FDCT	5926	25.3	4.4	31.15	7.40
Fibcall	4581	28.3	2.6	31.13	7.43
Fibonacci	24065	31.0	4.9	31.30	7.50
FIR	4222	29.3	2.8	29.99	7.17
Helloworld	4015	30.6	2.0	29.79	7.14
Insertsort	4721	28.8	4.9	31.20	7.43
Ludcmp	39884	16.4	10.7	33.31	7.81
MatrixMul8	21734	30.7	2.6	31.35	7.48
MatrixMul8_dotp	24411	29.8	2.4	32.53	7.72

increase incurred at small L_{M} is due to the shortening of the refreshing period $N_{\mathrm{RR}} \approx \lfloor (N_{\mathrm{DRT}} + L_{\mathrm{M}})/2 \rfloor$. Thus given a code of length$N$, the number of refreshing N/N_{RR} increases as well, where each refreshing refreshes the entire$L = 8192$ GCeDRAM lines.

The similarity of the dynamic energy model (17) to hardware simulations is demonstrated by the red section in Fig. 8 obtained for R/W probabilities $\mu = 0.67$ and $\lambda = 0.33$. Its projection depicted by the red dashed curve in Fig. 6 which was calculated for same probabilities, demonstrates good agreement of the model with real hardware behavior.

Since GCeDRAM is aimed at ultra-low power applications, it is important to show that its power consumption is competitive compared to SRAM. To this end, we compared the PULPino data cache implementations by opportunistically refreshed GCeDRAM to SRAM. Table III summarizes the power simulation results of 11 test programs, including all dynamic and static consumptions.

None of the tests incurred a performance loss. This is due to sufficient memory cycles allotted for opportunistic refreshing as shown by the percentage of CPU R/W instructions. The power reduction compared to SRAM is clear and consistent for all programs, achieving 30% reduction or more. Since no performance loss incurred; i.e., the run-time in both implementations was equal, the energy reduction is similar to the power

reduction. The rightmost column shows the impact on the PULPino total power, which was 7% or more on all tests. This confirms the attractiveness of replacing SRAM by GCeDRAM supplemented with opportunistic refreshing.

VII. Conclusion

Whereas performance typically requires trading off power and energy, we showed here that a GCeDRAM supplemented with opportunistic refreshing is not. Analysis supported by experimental results demonstrated its correlation between maximum performance and minimum energy. Not only is performance not degraded compared to SRAM, but the total energy is significantly smaller. Replacement of the SRAM data memory in the ultra-low power processor PULPino by opportunistically refreshed GCeDRAM yielded 30% energy savings in the memory, which translated into 7% energy savings in the entire processor.

Acknowledgment

The authors would like to thank Prof. Luca Benini of ETH Zürich and his group for availing the PULPino design environment and support of this research. They also grateful for the useful comments of the anonymous reviewers. Shmuel Wimer conducted this research while visiting Prof. Atsushi Takahashi Laboratory, Tokyo Institute of Technology.

References

[1] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Power, area, and performance optimization of standard cell memory arrays through controlled placement," *ACM Trans. Design Autom. Electron. Syst.*, vol. 21, no. 4, p. 59, May 2016.

[2] J. Barth et al., "A 500 MHz random cycle, 1.5 ns latency, SOI embedded DRAM macro featuring a three-transistor micro sense amplifier," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 86–95, Jan. 2008.

[3] K. C. Chun, P. Jain, T.-H. Kim, and C. H. Kim, "A 667 MHz logic-compatible embedded DRAM featuring an asymmetric 2T gain cell for high speed on-die caches," *IEEE J. Solid-State Circuits*, vol. 47, no. 2, pp. 547–559, Feb. 2012.

[4] A. Teman, P. Meinerzhagen, A. Burg, and A. Fish, "Review and classification of gain cell eDRAM implementations," in *Proc. IEEE 27th Conv. Electr. Electron. Eng. Isr.*, Nov. 2012, pp. 1–5.

[5] R. Giterman, A. Fish, A. Burg, and A. Teman, "A 4-transistor nMOS-only logic-compatible gain-cell embedded dram with over 1.6-ms retention time at 700 mV in 28-nm FD-SOI," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 4, pp. 1245–1256, Apr. 2018.

[6] P. A. Meinerzhagen, O. Andiç, J. Treichler, and A. P. Burg, "Design and failure analysis of logic-compatible multilevel gain-cell-based dram for fault-tolerant VLSI systems," in *Proc. 21st, Ed., Great Lakes Symp. Great lakes Symp. VLSI (GLSVLSI)*, 2011, pp. 343–346.

[7] B. Frankel and S. Wimer, "A self-refreshable bit-cell for single-cycle refreshing of embedded memories," *IEEE Trans. Comput.*, early access, Mar. 10, 2022, doi: 10.1109/TC.2022.3158481.

[8] A. Kazimirsky and S. Wimer, "Opportunistic refreshing algorithm for eDRAM memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 11, pp. 1921–1932, Nov. 2016.

[9] B. Frankel, R. Herman, and S. Wimer, "Queuing-based eDRAM refreshing for ultra-low power processors," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1131–1140, Sep. 2018.

[10] A. Traber et al. (2016). *PULPino: A Small Single-Core RISC-V SoC*. [Online]. Available: https://riscv.org/wp-content/uploads/2016/01/Wed1315-PULP-riscv3_noanim.pdf

[11] *Voltus IC Power Integrity Solution*. Cadence. [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/voltus-ic-power-integrity-solution.html#:~:text=Th%20Cadence%C2%AE%20Voltus%E2%84%A2,power%20grid%20of%20a%20chip

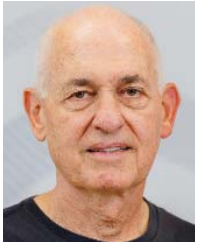
[12] M. Radulovic, R. S. Verdejo, P. Carpenter, P. Radojković, B. Jacob, and E. Ayguadé, "PROFET: Modeling system performance and energy without simulating the CPU," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 2, pp. 1–33, 2019.

[13] A. Shanbhag, S. Madden, and X. Yu, "A study of the fundamental performance characteristics of GPUs and CPUs for database analytics," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2020, pp. 1617–1632.

Binyamin Frankel received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Bar-Ilan University, Israel, in 2014, 2016, and 2022, respectively. Since 2021, he has been with Toga Networks. His research interests include VLSI circuits and systems design optimization and computer architecture.

Eyal Sarfati received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 2010 and 2018, respectively. Since 2010, he has been with Marvell, where he is currently a Physical Design Engineer. His research interests include VLSI circuits and system design optimization.

Davide Rossi (Member, IEEE) received the Ph.D. degree from the University of Bologna, Italy, in 2012. He is currently an Associate Professor with the University of Bologna. He has published more than 100 papers in international peer-reviewed conferences and journals in his research areas. His research interests focus on energy efficient digital architectures in the domain of heterogeneous and reconfigurable multi and many-core systems on a chip. This includes architectures, design implementation strategies, and runtime support to address performance, energy efficiency, and reliability issues of both high end embedded platforms and ultra-low-power computing platforms targeting the IoT domain. He was a recipient of the Donald O. Pederson Best Paper Award 2018, the 2020 IEEE Transactions on Circuits and Systems Darlington Best Paper Award, and the 2020 IEEE Transactions on Very Large Scale Integration (VLSI) Systems Prize Paper Award.



Shmuel Wimer (Member, IEEE) received the B.Sc. and M.Sc. degrees in mathematics from Tel Aviv University, Israel, in 1978 and 1981, respectively, and the D.Sc. degree in electrical engineering from the Technion—Israel Institute of Technology, Israel, in 1988. He held a research and development engineering and managerial positions in the industry. He was at Sagantec, IBM, National Semiconductor, and IAI-Israel Aerospace Industry. From 1999 to 2009, he was at Intel. Since 2009, he has been with the Engineering Faculty, Bar-Ilan University, Israel, where he is an Emeritus Professor. His research interests include VLSI circuits and systems design optimization and combinatorial optimization.