Quality aware selective ECC for approximate DRAM

Giulia Stazi, Antonio Mastrandrea, Mauro Olivieri, and Francesco Menichelli

Dept. of Information Engineering, Electronics and Telecommunications (DIET)
Sapienza University of Rome, Via Eudossiana, 18, 00184 Roma, Italy
{g.stazi,antonio.mastrandrea,mauro.olivieri,francesco.menichelli}@uniroma1.it

Abstract. Approximate DRAMs are DRAM memories where energy saving techniques have been implemented by trading off bit-cell error rate with power consumption. They are considered part of the building blocks in the larger area of approximate computing.
Relaxing refresh rate has been proposed as an interesting solution to achieve better efficiency at the expense of rising error rate. However, some works have demonstrated that much better results are achieved if at word-level some bits are retained without errors (i.e. their cells are refreshed at nominal rate), resulting in architectures using multiple refresh rates.
In this paper we present a technique that can be applied to approximate DRAMs under reduced refresh rate. It allows to trim error rate at word-level, while still performing the refresh operation at the same rate for all cells. The number of bits that are protected is configurable and depends on output quality degradation that can be accepted by the application.

Keywords: Approximate memory, Transprecision Computing, ECC memory

1 Introduction and previous works

Approximate computing is a design paradigm for low power systems that proposes to expand the degrees of freedom in digital system design by allowing inaccurate or approximate operations in circuits. The idea at the base of approximate computing is the fact that many real-world applications do not require exact mathematical computations, since their input and output data are inherently affected by noise and errors. Approximate memories are part of the building blocks of this approach and are intended as memory circuits that do not store data exactly and indefinitely, but are affected by errors during read/write operations or tend to spontaneously forget data with the passage of time [1, 2].

Depending on their technology, circuits for approximate memory have been proposed by scaling V_{dd} for SRAMs [3] and by reducing refresh rate under the nominal value for DRAMs [4]. These circuit-level proposals lay the groundwork for practical implementations that can be used in programmable architectures as

main approximate memory [5]. Applications that can tolerate a certain amount of errors can then allocate their data structures and buffers in these memories. These application, called ETAs (Error Tolerant Applications), will produce an output with degraded quality as the effect using approximate memories. The final assumption of approximate computing is that the amount of approximation (i.e. errors) can be tailored on the specific problem, trading off energy savings up to the limit of acceptable output quality.

2 Approximate memories in real applications

The first approach to approximate memories relies on allowing errors uniformly distributed on the array of bit cells (i.e. all cells are subject to the same voltage scaling or the same refresh rate). The validity of the choice is based mostly on the simplification that it involves at circuit level, since it does not require to modify the array internal circuit, but signals and power supply at the interfaces. However, considering uniform error distribution means to not take into account the exponential relation between different bit weights in a data word, which is instead an important characteristic that should be considered by approximate memory circuit, even at the expense of increasing circuit complexity.

2.1 Exact MSBs in an approximate data word

The first and intuitive approach is to design the memory array in order to save MSBs in exact bit-cells. Considering DRAMs, [6] proposes using two different refresh rates, one at nominal rate and one at reduced rate. Cell arrays are rearranged in a way that the nominal refresh rate is applied to bit cells for MSBs (exact MSBs), while the reduced refresh rate is applied to bit cells for LSBs (approximate LSBs). The number of exact MSBs and approximate LSBs depends on applications, for example, for 32bit words a number from 1 to 8 exact MSBs and, respectively, 31 to 24 approximate LSBs have been found to be of interest [7]. Requiring exact cells in an approximate data word has direct impact on the following characteristics:

- it raises output quality under the same error rate in LSB cells or, conversely, allows for higher error rate in LSBs while meeting the required output quality;
- it reduces overall energy saving, since a portion of the cells is working at nominal conditions;
- it increases circuit complexity requiring dual refresh rate in DRAMs.

2.2 Bit dropping for LSBs and bit reuse

The second approach results from exploration of the relation between output quality and BER on the LSBs [7]. LSBs in a data word can be dropped and set to a constant value (i.e. 0) with a marginal impact on output quality degradation.

Table 1. List of Hamming codes

#Check bits (n-k)	#Total bits (n)	#Data bits (k)	Name	Rate
2	3	1	Hamming(3,1)	1/3
3	7	4	Hamming(7,4)	4/7
4	15	11	Hamming(15,11)	11/15
5	31	26	Hamming(31,26)	26/31
6	63	57	Hamming(63,57)	57/63

It is a technique that is proposed since it achieves energy savings with a simple circuit implementation (bit cells are powered off or even omitted).

Previous works have proposed to use selective ECC in SRAM to reduce errors in MSB (1) by enlarging memory words as in classical ECC memory systems (i.e. 32bit memory word are expanded to 36bit, introducing 4bit ECC) [8] (2) by reusing LSB dropped bits [9]. The contribute of our work is (1) to design selective ECC specific for approximate DRAM memory systems (2) to allow tailoring selective ECC to the specific application, by first analyzing its output quality degradation related to bit error rate, looseness level and dropped bits.

3 Quality aware selective ECC

The idea of quality aware selective ECC consists in a two step process. First, an application is analyzed in order to find the desired tradeoff between output quality and approximate memory parameters (i.e. error rate, level of approximation [1], dropped bit); then an error correcting code is chosen in order to reduce error rate in a specific portion of data bits. In order to avoid increasing memory requirements with additional ECC bit, bit dropping and reuse is always considered for the additional check bits required by ECC.

3.1 ECC codes for approximate memories

In order to reduce hardware complexity, (n,k) SEC (single error correcting) Hamming codes were considered. In this notation, k indicates the number of protected bits (data bits), while n is the code length, including additional check bits. We note that SEC codes can provide also error detection (e.g. double error detection typically), but for our scope error detection is not used: in case of detected errors, program execution continues as for undected errors, in approximate memory. Table 1 summarizes the most common Hamming codes. We note that, as a general rule, increasing the number of data bits k produces more efficient codes, since the rate k/n increases. However, larger k are effective at very small error rates (as is common in exact memories). In approximate memories typical error rates are much larger (i.e. from 10^{-4} to $10^{-2} errors/(bit \times s)$ [1]) and, as consequence, shorter codes are desirable since enlarging n increases the probability of multiple errors within the same word, which cannot be corrected.

3.2 Looseness level

With Looseness Level we intend the concept, introduced in [1], of having a certain number of exact MSBs in an approximate data words. As an example, Table 2 (left) reports results obtained on a 32 bit integer FIR filter, showing how Looseness level (i.e. the number of exact MSBs) can impact output SNR.

Instead of using exact DRAM cells for MSBs, the idea is to use a single, and slower, refresh rate for all cells, while using SEC ECC in order to reduce error rate in MSBs. In this way, MSBs are still affected by errors, but their error rate is reduced with respect to LSB cells.

3.3 Impact of bit dropping and bit reuse

Table 2 (right) reports results obtained on the same 32 bit integer FIR filter, showing how bit dropping (i.e. powering them off and reading them as '0') impacts output SNR. As already confirmed in literature, output SNR is only slightly dependent on LSBs. Instead of powering them off, these LSBs can be effectively reused as checkbits for the MSBs, without requiring additional bits.

Table 2. FIR, output SNR [dB]

Looseness Level	**Fault rate** $[errors/(bit \times s)]$			
	10^{-1}	10^{-2}	10^{-3}	10^{-4}
12 MSBs	70.5	83.4	93.5	104.2
8 MSBs	46.5	59.6	69.3	80.3
4 MSBs	22.6	35.3	45.5	56.4
1 MSB	4.6	17.2	27.6	38.2

# of dropped bits			
4 LSBs	**8 LSBs**	**12 LSBs**	**16 LSBs**
134.7	122.4	106.1	82.2

4 Implementation and results

Given the list of Hamming codes in Table 1, it appears that the most suitable for our application are Hamming (3,1),(7,4) and (15,11). This choice depends on two factors, first we assume to protect single 32 bit words in memory, in order to not impact read/write speed; infact, protecting with a single code larger data size would require to multiple read/write on the entire data. Secondly, given the relatively high bit error rate of approximate memories, longer SEC codes tend to fail due to the rising probability of multiple errors.

Figure 1 shows the formats considered for 32 bit data, where k MSBs are protected by SEC ECC, $32 - n$ bits are left unprotected and $n - k$ dropped and reused as checkbits. Assuming a uniform error probability p_e for each bit, expressed as $errors/(bit \times s)$, the probability of having i errors in a set of n bits is:

$$P_e(n,i) = \binom{n}{i} p_e^i (1-p_e)^{n-i};$$

Considering the SEC ECC code, protected bits will contain errors for $i \geq 2$; hence:

$$Pecc_e(n) = \sum_{i=2}^{n} P_e(n,i) = \sum_{i=2}^{n} \binom{n}{i} p_e^i (1-p_e)^{n-i};$$

In order to get a measure of the improvement, we can find the equivalent error rate peq_e, considered as the error rate that n bits (without ECC) should have to produce the same $Pecc_e(n)$.

$$Peq_e(n) = \sum_{i=1}^{n} P_e(n,i) = 1 - \sum_{i=0}^{0} P_e(n,i) = 1 - (1 - peq_e)^n;$$

Equivalent bit error rate peq_e for ECC protected bits can be obtained with $Peq_e(n) = Pecc_e(n)$:

$$peq_e = 1 - \sqrt[n]{1 - Pecc_e(n)};$$

Assuming 32 bit data stored in approximate memory, Figure 1 resumes how selective ECC could be applied using Hamming (3,1), (7,4) and (15,11) codes. The most appropriate choice depends on the application; for example, according to Table 2, a range from 8 to 12 protected MSBs results in an output SNR between $60dB$ to $93dB$, while dropping 4 LSBs does not significantly impact SNR. In this case Hamming (15,11) seems the most suitable choice.

Table 3 reports the results that can be obtained on typical target application considering the previous Hamming codes. It shows that MSBs protected by SEC codes expose and equivalent BER significanlty lower than unprotected bits. Considering the previous example, Hamming (15,11) and a BER of 10^{-3} on cells produces an equivalent BER of 6.94×10^{-6} on MSBs.

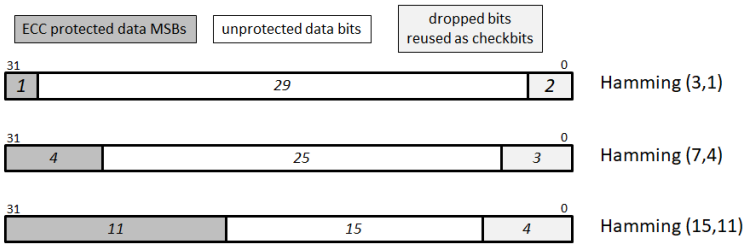

Fig. 1. 32 bit ECC data format in approximate memory

Table 3. BER for 32 bit data in approximate memory

Hamming (3,1) word		
ECC prot. 1 bit	**unprot.** 29 bit	**drop** 2 bit
9.42E-03	1.00E-01	-
9.93E-05	1.00E-02	-
9.99E-07	1.00E-03	-
1.00E-08	1.00E-04	-
1.00E-10	1.00E-05	-

Hamming (7,4) word		
ECC prot. 4 bit	**unprot. 25 bit**	**drop 3 bit**
2.29E-02	1.00E-01	-
2.90E-04	1.00E-02	-
2.99E-06	1.00E-03	-
3.00E-08	1.00E-04	-
3.00E-10	1.00E-05	-

Hamming (15,11) word		
ECC prot. 11 bit	**unprot. 15 bit**	**drop 4 bit**
3.92E-02	1.00E-01	-
6.45E-04	1.00E-02	-
6.94E-06	1.00E-03	-
6.99E-08	1.00E-04	-
7.00E-10	1.00E-05	-

5 Conclusion

In this paper we proposed the use of selective ECC in approximate DRAM memory tailored to quality requirements of applications. We started from the consideration that in many works and use cases it has been demonstrated the effectiveness of limiting approximate cells to LSBs while leaving a portion of MSBs exact. However, this approach requires higher complexity in memory circuits and circuits surrounding the cell array. For DRAMs, it requires to produce and distribute multiple refresh rates in the array.

Due to the relatively high error rates in approximate memories, SEC codes reduce but do not eliminate errors. This is completely acceptable and we demonstrated that for typical error rates in the order of 10^{-3} to 10^{-4}, Hamming codes (7,4) and (15,11) can reduce error rate on MSBs of factor between 1/100 to 1/1000. Future works will implement the technique in simulation models and apply it to error tolerant applications, allowing the characterization and the comparison with respect to previous techniques.

References

1. G. Stazi, A. Mastrandrea, M. Olivieri, and F. Menichelli, "Full system emulation of approximate memory platforms with appropinquo," *Journal of Low Power Electronics*, vol. 15, no. 1, pp. 30–39, 2019.
2. F. Menichelli, G. Stazi, A. Mastrandrea, and M. Olivieri, "An emulator for approximate memory platforms based on qemu," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2016, pp. 153–159.
3. F. Frustaci, D. Blaauw, D. Sylvester, and M. Alioto, "Approximate srams with dynamic energy-quality management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2128–2141, 2016.
4. A. Raha, S. Sutar, H. Jayakumar, and V. Raghunathan, "Quality configurable approximate dram," *IEEE Transactions on Computers*, vol. 66, no. 7, pp. 1172–1187, 2017.
5. G. Stazi, F. Menichelli, A. Mastrandrea, and M. Olivieri, "Introducing approximate memory support in linux kernel," in *Ph. D. Research in Microelectronics and Electronics (PRIME), 2017 13th Conference on*. IEEE, 2017, pp. 97–100.
6. J. Lucas, M. Alvarez-Mesa, M. Andersch, and B. Juurlink, "Sparkk: Quality-scalable approximate storage in dram," in *The memory forum*, 2014, pp. 1–9.
7. G. Stazi, L. Adani, A. Mastrandrea, M. Olivieri, and F. Menichelli, "Impact of approximate memory data allocation on a h.264 software video encoder," in *International workshop on Approximate and Transprecision Computing on Emerging Technologies (ATCET)*. Springer, 2018.
8. I. Lee, J. Kwon, J. Park, and J. Park, "Priority based error correction code (ecc) for the embedded sram memories in h. 264 system," *Journal of Signal Processing Systems*, vol. 73, no. 2, pp. 123–136, 2013.
9. F. Frustaci, D. Blaauw, D. Sylvester, and M. Alioto, "Better-than-voltage scaling energy reduction in approximate srams via bit dropping and bit reuse," in *2015 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 2015, pp. 132–139.