ACME: An energy-efficient approximate bus encoding for I2C

ACME: An Energy-Efficient Approximate Bus Encoding for I^2C

Chen Xie, Daniele Jahier Pagliari, Andrea Calimera, Enrico Macii, Massimo Poncino
Politecnico di Torino, Turin, Italy
name.surname@polito.it

***Abstract*—In ultra low power systems with many peripherals, off-chip serial interconnects contribute significantly to the total energy budget. Leveraging the error-resilience characteristics of many embedded applications, the approximate computing paradigm has been applied to serial bus encodings to reduce interconnect consumption. However, the power model considered in previous works was purely capacitive. Accordingly, the objective of these approximate encodings was simply to reduce the transition count. While this works well for most bus standards, one notable exception is represented by I^2C, whose open-drain physical connection makes the static energy consumed by logic-0 values on the bus extremely relevant.**

In this work, we propose ACME, the first approximate serial bus encoding targeting specifically I^2C connections. With a simple encoding/decoding scheme, ACME concurrently reduces both the static and dynamic energy on the bus by maximizing the number of logic-1 values in codewords, while simultaneously reducing transitions. Using an accurate bus model and realistic capacitance and resistance values selected according to the I^2C standard, we show that our encoding outperforms state-of-the-art solutions and reduces the total energy consumption on the bus by 57% on average, with an error smaller than 0.1%.

***Index Terms*—Bus Encoding, Approximate Computing, Energy Efficiency**

I. Introduction

Energy efficiency is a major challenge in the design of modern, battery-operated computing platforms, such as smartphones, wearables and Internet of Things (IoT) nodes. Most of these systems are equipped with tens of off-chip sensors, connected to processing elements (e.g., microcontrollers) via serial buses based on standard protocols such as Serial Peripheral Interface (SPI), Controller Area Network (CAN), Inter-Integrated Circuit (I^2C), etc. In these sensor-rich systems, data transfers become significant contributors to the total energy budget; previous research has shown that, in an ultra low-power IoT node, off-chip bus transmission energy can be comparable or superior to processing energy [1].

In the last decade, Approximate Computing (AxC) has emerged as a promising paradigm for the design of energy-efficient digital systems [2], which relies on the ability of many emerging applications (multimedia processing, machine learning, etc.) to tolerate errors at different levels during their execution, with insensible quality losses on final results [2]. In several previous works, AxC has been applied to the serial bus interfaces of sensor-rich devices [3], [4]: by encoding data in a lossy manner, these works achieved savings by reducing the number of electrical level transitions on the bus [3], [4].

The main motivation for focusing only on the number of transitions, in these previous works, was the usage of a purely capacitive bus power consumption model, where the total energy consumption was considered approximately equal to the dynamic energy. While this is a good approximation for most serial interconnections such as SPI and CAN, the case of the I^2C protocol is a notable exception. In fact, I^2C peripherals use an open-drain physical connection for both the Serial Clock Line (SCL) and the Serial Data Line (SDA), which can only drive the bus line to a low level. A pull-up resistor is then used to bring the line to a high voltage when the bus is idle [22]. Thus, besides the dynamic consumption due to level changes, a significant amount of *static* energy will be dissipated by the pull-up resistor at all times when the line voltage is low, i.e., in correspondence of logic-0 values. In our results, we will show that this static component is actually the dominant one, proving that existing approximate bus encodings are sub-optimal when applied to I^2C.

In this work, we propose the AxC Maximum-one Encoding (ACME), the first approximate serial bus encoding targeting specifically I^2C connections. ACME concurrently tackles both static and dynamic energy, by reducing both the number of level transitions and the number of logic-0s transmitted on the bus. With an accurate bus model, realistic pull-up resistance values selected according to the I^2C standard, and datasets relative to six different types of sensors, we show that our new encoding outperforms state-of-art solutions by reducing the total energy consumption on the bus by 57% on average, with less than 0.1% error on the decoded words.

II. Background and Related Works

A. I^2C Power Model

Differently from other off-chip serial bus standards, for which dynamic energy dominates, and which can be simply modeled as a capacitive channels [5]–[7], I^2C imposes the open-drain architecture for both SDA and SCL, including an NMOS transistor and a pull-up resistor *Rp*. This is shown in Figure 1, where, additionally, C_b represents the cumulative bus capacitance, including the contribution of the PCB and of the drivers of all devices connected to the bus [22].

The NMOS transistor acts as a switch driven by the data to be transmitted. In order to transmit a logic-1, the NMOS is turned off, and V_{out} is pulled up to the supply voltage by R_p. On the other hand, when transmitting a logic-0, the switch is turned on, so that V_{out} is connected to ground.

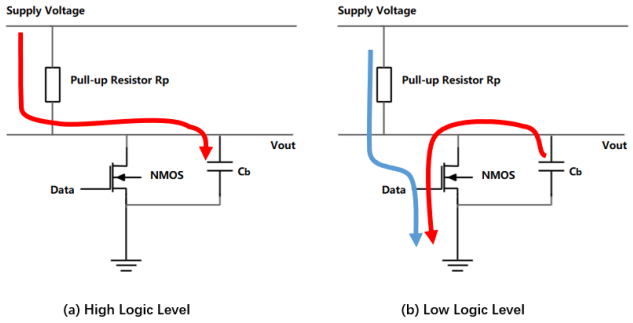

Fig. 1. I^{2}C open-drain architecture and current flow at High/Low logic levels.

Therefore, whenever V_{out} is logic low, there is a considerable *static* energy consumption due to the current flowing through the pull-up resistor *Rp* [8]. In addition to this static consumption, during a 0-to-1 transition, a dynamic dissipation occurs due to the charging of the bus capacitance C_b. Symmetrically, the discharge of C_b consumes dynamic energy during 1-to-0 transitions. The static and dynamic current flows are shown in blue and red respectively in Figure 1a (1b) for a high (low) logic level.

Previous work [8] has shown that static energy is dominant in the total energy consumption of an I^{2}C channel. Nonetheless, in this work we consider a complete model including both static and dynamic energy to evaluate our approximate encoding, applied to the SDA line. Specifically, we use (1) and (2) to compute the static and dynamic energy respectively:

$$E_{st} = \frac{{V_{dd}}^2}{R_p f} \tag{1}$$

$$E_{dy} = {V_{dd}}^2 \cdot C_b \tag{2}$$

where V_{dd} is the supply voltage, R_p the pull-up resistance, C_b the bus capacitance and f the transmission frequency. (1) holds only when the line is logic-low, while (2) holds for each level transition. The total energy is then simply computed summing the static and dynamic components. In order to accurately estimate all quantities in (1) and (2), our experiments are based on a Spice model of the channel and drivers.

B. Related Works

A large number of low-power data encodings for serial buses have been proposed in recent years. The earliest approaches proposed are *lossless* (or accurate), meaning that the transmitted data can be decoded exactly at receiver end without any information loss [5], [6], [9], [10]. Following the purely capacitive model mentioned above, these encodings focus on reducing the number of logic-level changes on the bus, often called Transition Count (TC). Many do it with variations of so-called Differential Encoding (DE) [11], a scheme that achieves TC reductions by transmitting the bitwise difference between subsequent words, instead of original data. Popular encodings such as SILENT [6] and Differential Bar Encoding (DBE) [9] are based on DE. Others try to overcome its limitation, i.e., the negative energy savings obtained when consecutive data transmitted are uncorrelated with each other [12], [10], [5]. However, these advanced lossless encodings lead to costly hardware implementations for the encoder and decoder (in terms of silicon area), which also reduce the potential savings due to their power overheads. At the same time, the TC reductions that they obtain are limited by the constraint of strict output correctness.

Leveraging the error tolerance properties of many sensor-based applications [2], researchers have thus started proposing lossy serial encodings for data buses. These solutions are based on AxC principles, and trade-off output data quality for additional energy reductions. In [7], the authors proposed one of the first lossy encodings, targeting image data. Their solution is based on saturating the Least Significant Bits (LSBs) of codewords under error limits to reduce the TC. More recently, an encoding called Rake [3] has been proposed, which reduces the TC by selectively inverting codewords bits. Serial T0 [4] uses a 0-TC constant code in place of words that differ negligibly from their direct predecessors (in terms of absolute value), while transmitting exactly only when the difference is large. This approach exploits the fact that errors are typically more tolerable for long sequences of similar data (called "idle" phases), which correspond to low-information-content transmissions. AXSERBUS [1] adopts similar principles but supports *three* modes of operation depending on the magnitude difference between consecutive words. By doing so, it achieves a better trade-off between energy saving and data quality. Finally, Approximate Differential Encoding (ADE) [11] uses DE as a baseline, but combines it with bounded approximations on LSBs to overcome its limitations on uncorrelated data. The latter is the closest prior work to our encoding, which is also based on bitwise operations combined with LSB-approximations. However, differently from this work, ADE, as well as all other previous lossy encodings, focus only on TC reduction. To the best of our knowledge, ours is the first approximate bus encoding to consider a non-purely-capacitve channel, therefore explicitly targeting devices based on I^{2}C.

III. Proposed Method

A. Approximate Computing Maximum-one Encoding

We propose a new lossy serial encoding that explicitly targets error tolerant peripherals (e.g., sensors) using I^{2}C connections. Our approach is called *Approximate Computing Maximum-one Encoding* (ACME), and is inspired by differential encoding (DE) and its approximate variant ADE [11]. The goal of ACME is to reduce TC on the bus, hence the dynamic energy, while simultaneously increasing the number of transmitted logic-1s, to reduce static energy, following the I^{2}C power model of (1) and (2). The encoding algorithm of ACME is only applied to *data* words, excluding parts of the I^{2}C protocol which cannot be approximated (e.g. *addresses* bits), and can be summarized by the following two equations:

$$b_{ax,i}[t] = \begin{cases} 0 \text{ if } i < l \\ b_i[t] \text{ otherwise} \end{cases} \qquad \forall i \in [0, n-1] \tag{3}$$

$$B_i[t] = b_{ax,i}[t] \odot b_{ax,i}[t-1], \ \forall i \in [0, n-1] \tag{4}$$

where b is the input word, b_{ax} an intermediate, approximate version of b, B the final codeword, and n the word length. Moreover, $\odot$ indicates the binary XNOR operator, i indicates the i-th bit of a word and t the t-th word in the temporal sequence transmitted on the bus. Lastly, l represents the number of approximated LSBs, and is the main parameter used in ACME to trade-off the approximation error and the energy reduction on the bus.

In the DE algorithm [6], codewords are computed as the bitwise difference of consecutive words. This is motivated by the fact that, when two consecutive inputs have similar binary values, their bitwise difference produces a sequence of 0s in the MSBs, which does not incur intra-word transitions once serialized. DE is particularly effective for error tolerant data traces, such as those produced by sensors, which are often constituted by long sections of highly correlated data (e.g., slowly increasing/decreasing or nearly constant) [4], [6], [11].

ACME takes inspiration from this simple yet effective idea, but replaces the bitwise difference (XOR) with an equality (XNOR), as shown in (4). The obvious result is that identical MSBs in consecutive words now yield codewords with long sequences of 1s (instead of 0s), reducing the static energy on an I^2C connection. Furthermore, as shown in (3), before generating codewords, ACME approximates each word by flooring to 0 its l-LSBs (this is the lossy part of the encoding). As opposed to rounding LSBs, as done in ADE [11], flooring introduces a bias in the approximation, but has the positive effect of making the LSBs of all words identical. After XNOR-ing and serialization, this will generate *another* sequence of constant 1s, further reducing both static and dynamic energy.

At the decoding end, ACME implements the following equation to reconstruct the (approximated) input:

$$b_{ax,i}[t] = B_i[t] \odot b_{ax,i}[t-1],\ \forall i \in [0, n-1] \qquad (5)$$

This encoding/decoding scheme has $O(n)$ complexity, and lends itself to a low-overhead implementation in either SW or HW. Furthermore, the number of approximated LSBs l is configurable, allowing a *runtime* tuning of the trade-off between approximation and energy saving. With l approximated bits, the maximum encoding error of ACME is $E_{max} = 2^l - 1$, and under the common assumption of a uniform distribution of LSBs [13], the expected error is $E_{max}/2$.

B. Examples of Operations

A visual representation of the operations of the conventional DE and ACME alogrithms is illustrated in Figure 2. Here, we use 8-bit data as an example, we set the number of approximated LSBs to $l = 2$, and we assume a MSB-first serialization. The bits that cause a transition are highlighted in red (note that the idle value of the SDA line is logic-high). Note that, to obtain the first encoded word, the XOR (XNOR) operation is performed with an initialized all-0s code.

An extended example of DE and ACME encoding is shown in Table I for 8-bit unsigned data. The table reports the TC generated by raw data and by the two encodings, considering

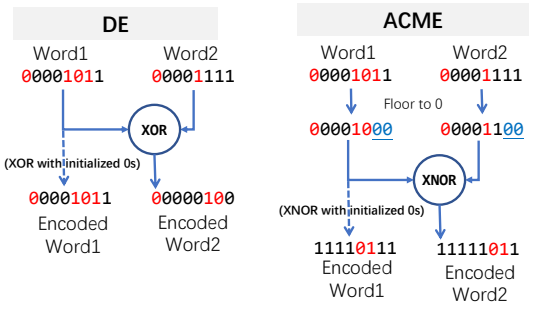

Fig. 2. Graphical explanation of ACME encoding (and DE for comparison).

both inter-word and intra-word transitions, and MSB-first serialization; moreover, the number of logic-1 bits in each word is also counted (column TC/1s). For ACME, we also report the absolute error ($\|E\|$ column) introduced on each word by flooring $l = 2$ LSBs to 0.

TABLE I
EXAMPLE OF DE AND ACME ENCODING FOR 8-BIT UNSIGNED DATA

Input		**DE**		**ACME (l=2)**		
Word	TC/1s	Word	TC/1s	Word	TC/1s	$\|E\|$
01001011	6/4	01001011	6/4	10110111	4/6	3
01001101	6/4	00000110	3/2	11111011	2/7	1
01001100	5/3	00000001	1/1	11111111	0/8	0
01001101	5/4	00000001	2/1	11111111	0/8	1
01010110	7/4	00011011	4/4	11100111	2/6	2
01100001	3/3	00110111	4/5	11001011	4/5	1
01000100	5/2	00100101	6/3	11011011	4/6	0
01001110	4/4	00001010	5/2	11110111	2/7	2
01001101	5/4	00000011	1/2	11111111	0/8	1
01001011	6/4	00000110	3/2	11111011	2/7	3
Totals	52/36		35/26		20/68	14

The table shows that, while DE is effective in reducing the TC, it also reduces the number of logic-1s transmitted, negatively affecting static energy. The same is true for the approximate DE variant ADE [11], not reported for sake of space. In contrast, ACME simultaneously achieves larger TC reductions than DE, thanks to LSB approximation, while also significantly increasing the number of logic-1s. In this example, the TC is reduced by 62%, while the 1s count is increased by 89%. This comes at the cost of a small approximation error for each transmitted word. On average, the error is 14/10 = 1.4, i.e., 0.5% of the full-scale value (255).

C. Hardware Implementation

ACME encoding and decoding can be easily implemented with simple and low-cost digital hardware circuits. The schematics of a N-bit implementation are shown in Figure 3.

The encoder (Figure 3a), implements the floor operation through an array of multiplexers. The selection signal for the multipliexers is l, i.e., the number of approximated LSBs, encoded on $log_2(n)$ wires, where n is the word length. For each approximated bit, the multiplexers forward a constant "0" to the next encoder stage, in place of the corresponding input word's bit b_i. For example, when $l_i = 0,\ \forall i \in [0, log_2(n) - 1],\ l_0 = 1$, which corresponds to $l = 1$ approximated LSBs,

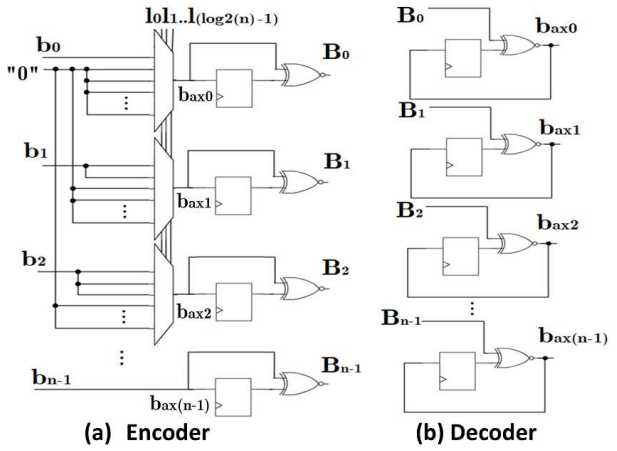

Fig. 3. Hardware Implementation of N-bit ACME.

b_0 is replaced by 0, while b_i, $\forall i \in [0, n-1]$, are selected. After saturation, an array of n XNOR gates is used to implement the bitwise equality operation among the current word and the previous one, memorized in a corresponding set of flip-flops.

The decoder (Figure 3b) just includes one flip-flop and one XNOR gate for each bit. The latter perform a bit-wise equality between corresponding bits of the current codeword and previous decoded word, to implement (5). For example, if $B_i[t] = 0$, the negated value of $b_{ax,i}[t-1]$ is produced in output as $b_{ax,i}[t]$, otherwise the affirmed $b_{ax,i}[t-1]$ is copied.

It is easy to see that these hardware implementations of the encoder and decoder achieve a throughput of 1 encoded/decoded word per cycle, and have a single clock cycle latency. At the same time, their simple circuital structures also lead to low area and power overheads, as shown in Section IV.

IV. Results

A. Setup

To evaluate the performance of ACME, we use a number of datasets to represent the data transmitted from/to various error resilient devices involving I^{2}C connections. Specifically:

- **Images** [15]: 100 24-bit RGB (8-bit per channel) images from the Linnaeus 5 dataset, to assess the transmission of pixel data from a camera or to a display.
- **ECG-ID** [16]: A dataset including 310 filtered Electrocardiogram (ECG) recordings from 90 patients, to represent biomedical sensors. Data are digitized with 12-bit precision as specified in the dataset.
- **Activity Recognition** [17]: Accelerometer, gyroscope and magnetometer samples collected for six physical activities, and with sensors in four positions (arm, belt, pocket and wrist). Accelerometer data are represented in 12-bit two's complement, while gyroscope and magnetometer data are in 16-bit two's complement, based on the precision of commercial sensor products [18]–[20].
- **Audio** [21]: Voice recordings from the Open Speech Repository, represented in 16-bit two's complement.

To prove the effectiveness of ACME, we compare it against three state-of-the-art approximate serial bus encodings: ADE [13], ST0 [14] and Axserbus [1]. Note that, in particular, even if Serial T0 and Axserbus are designed for dynamic power reduction only, they also tend to increase the number of logic-1s on the bus, by approximating small differences with the all-1s special pattern (see the original papers for details).

We use a SPICE model of the I^{2}C bus to compute the energy consumption of the different encodings. We initially select one of the I^{2}C operating modes for each input, based on the required throughput: for Accelerometer, Magnetometer, Gyroscope and ECG data, we use I^{2}C Standard-Mode, whereas for Image and Audio, we select Fast-Mode. We then set the bus frequency f to the maximum allowed by each operating mode, i.e., 100kHz and 400kHz respectively. Note that, while a lower bus frequency could suffice for some of the considered inputs, using the maximum f clearly increases the dynamic/static energy consumption ratio. Since the main advantage of ACME with respect to its comparison baselines is a greater static energy reduction, this is a worst-case scenario for our algorithm, and savings could only increase at lower frequencies. We set the bus voltage to $V_{CC} = 3.3V$ and the bus capacitance to $C_b = 150pF$ according to [23]; we then derive R_p from C_b using the plots available in the I^{2}C specification [22]; for Standard- and Fast-Mode, we obtain $R_p = 7.8k\Omega$ and $R_p = 2.3k\Omega$ respectively. All encodings are initially implemented and simulated in Python. Then, to evaluate the area and power of an hardware implementation, we synthesize the ACME encoder and decoder from an RTL description in VHDL, targeting a 45nm CMOS library from STMicroelectronics. Synthesis is performed with Synopsys Design Compiler and verified with Mentor Questa Simulator.

B. Bus Energy Reduction

Figure 4 shows the Pareto curves generated by ACME and the three comparison baselines in the error vs energy space for one Standard-mode ($f = 100kHz$, $R_p = 7.8k\Omega$) and one Fast-mode ($f = 400kHz$, $R_p = 2.3k\Omega$) dataset, i.e., Accelerometer and Audio data respectively. The x axes report the average absolute error obtained by each method after encoding and decoding, normalized to a % of the full-scale value of each dataset[1]. Different points are obtained varying the approximation parameters of the encodings. For ACME, we vary l, i.e., the number of floored LSBs. All other encodings also have a single approximation parameter, described in the respective papers, except for Axserbus [1], which has 2 (d_0 and d_m). For the latter, we perform a grid search over d_0 and d_m values with a step of powers of 2 (and keeping d_m always larger than d_0) to extract the Pareto fronts.

The y axes report the *dynamic*, *static* and *total* energy saving with respect to the transmission of raw (unencoded) data over the same I2C bus, for each encoding and approximation parameters setting. Although ACME is competitive also in terms of dynamic energy reduction, the results show that it is slightly outperformed by ADE on both datasets. This is due

[1]Note that the normalized error for audio data is lower due to the larger bit-width (16bit vs 12bit of the accelerometer samples).

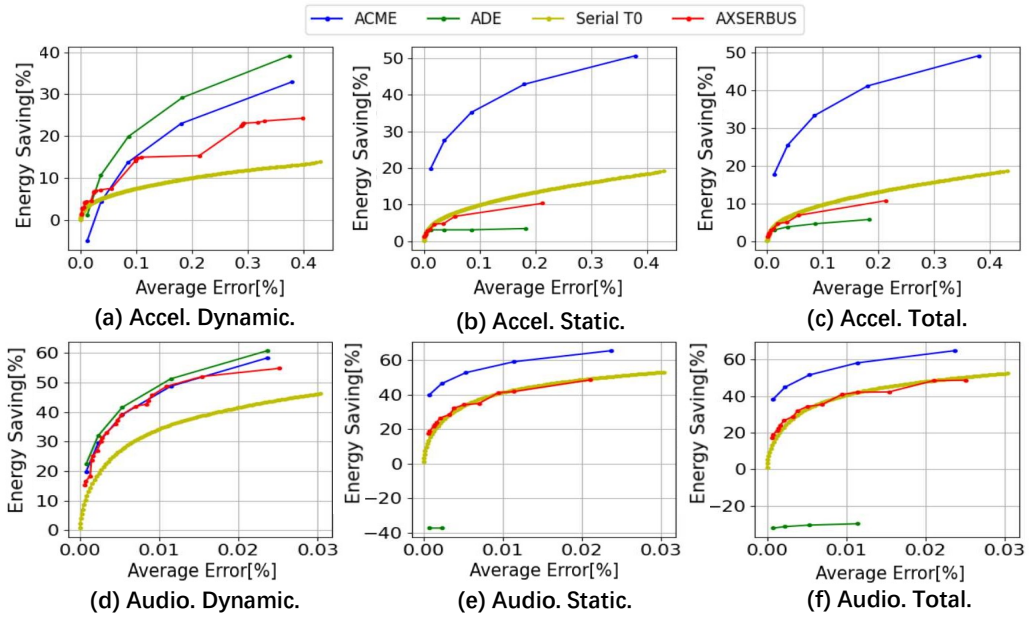

Fig. 4. Average error vs. energy saving Pareto curves obtained by ACME and 3 comparison baselines on accelerometer and audio samples.

TABLE II
ENERGY SAVINGS OF ACME AND OTHER APPROXIMATE BUS ENCODINGS UNDER TWO ERROR CONSTRAINTS.

Constraint	Encoding	Energy Saving [%]																	
		ECG			Accel			Magnet			Gyro			Audio			Image		
		Dyn.	Stat.	Tot.	Dyn.	Stat.	Tot.	Dyn.	Stat.	Tot.	Dyn.	Stat.	Tot.	Dyn.	Stat.	Tot.	Dyn.	Stat.	Tot.
C1	Serial T0	79.6	77.0	77.1	5.7	7.4	7.2	18.7	18.4	18.5	26.6	33.7	33.2	52.1	58.2	57.7	40.8	**45.4**	**44.8**
	Axserbus	**83.6**	81.4	**81.6**	7.2	4.8	5.0	32.7	6.2	7.5	33.5	27.7	27.6	68.9	48.5	48.7	**51.8**	38.8	40.4
	ADE	76.6	0.0	0.0	**10.6**	3.1	3.8	**46.7**	0.0	0.0	**53.9**	0.0	3.4	**69.5**	0.0	0.0	40.5	0.0	0.0
	ACME	76.1	**81.9**	81.5	4.4	**27.5**	**25.5**	39.6	**33.7**	**34.2**	43.7	**39.2**	**39.5**	67.2	**71.2**	**70.9**	38.3	45.0	44.2
C2	Serial T0	87.3	84.6	84.8	7.5	9.9	9.7	26.2	25.2	25.3	33.9	39.7	39.3	60.8	65.6	65.2	51.9	56.4	**55.9**
	Axserbus	**89.9**	86.1	86.4	14.1	6.8	6.8	40.2	6.2	7.6	41.4	33.8	33.9	**77.2**	51.8	53.5	**62.7**	49.4	50.3
	ADE	86.4	0.0	0.0	**19.9**	3.1	4.6	**55.6**	0.0	0.0	**61.8**	0.0	3.7	76.7	0.0	0.0	52.5	0.0	0.0
	ACME	86.2	**88.6**	**88.4**	13.7	**35.2**	**33.3**	48.2	**40.8**	**41.4**	51.7	**45.3**	**45.7**	74.4	**76.2**	**76.0**	50.3	**56.5**	55.8

to the lower average error and number of *inter-word* transitions achieved by ADE, thanks to LSB rounding instead of flooring.

In contrast, ACME dominates all other encodings (and ADE in particular) in terms of static energy reduction, achieving up to 30% (accel.) and 10% (audio) higher saving for the same error level. This is because it explicitly tries to increase the number of transmitted logic-1 when more error is allowed. In contrast, Serial-T0 and Axserbus achieve a similar increase, but only due to a "secondary effect" of their more frequent usage of the all-1 special pattern [1], [4]. Lastly, ADE actually transmits *less* logic-1s for larger errors; thus, increasing the amount of approximation generates Pareto-dominated points (hence not shown in the plots) with *higher* static energy. The total energy saving graphs are almost identical to the static ones, showing that the static component is indeed the dominant one in I^2C buses, and that, consequently, ACME is superior to its comparison baselines for these connections.

These results are extended to all 5 datasets in Table II. Since we could not show the complete Pareto fronts for all datasets for space reasons, we report the savings of all encodings under two different average error constraints (C1 and C2). For 12bit and 16bit datasets, we set C1=0.05% and C2=0.1%, whereas for 8bit images, we set C1=0.5% and C2=1%, since none of the encodings could reach <0.1% average error for such a small bit-width, except using a fully-accurate setting (e.g., $l = 0$ in ACME). The results show that ACME achieves the largest total savings under both constraints for all datasets except images, where it is slightly outpeformed by Serial T0, and ECG (under constraint C1), where it achieves 0.1% less saving than Axserbus. At the same time, ACME dynamic energy savings are comparable to other encodings, meaning that, although specifically designed for I^2C, ACME could also be effective on other buses. On average, when considering all 12/16bit datasets, ACME achieves a 57% total energy reduction at the cost of a 0.1% average error on the transmitted data. Note that several previous works have shown that such a small error is negligible if serially-transmitted data are then used as inputs for error tolerant applications (e.g., machine learning classification) [1], [24].

C. Dependence on I2C Bus Parameters

Figure 5 shows the variation in the total energy savings obtained by the different encodings for different values of R_p

and f supported by the I^{2}C standard. For sake of space, only accelerometer data are reported; in both graphs, the savings are obtained with an error constraint equal to C2 (i.e., 0.1%). In Figure 5a, f is fixed to 100kHz, whereas in Figure 5b, R_p is fixed to 7.8kΩ.

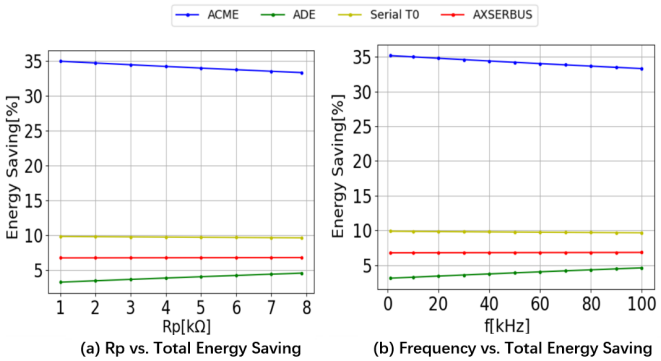

Fig. 5. Total Energy Saving versus R_p and f

As expected, since both a larger pull-up resistance and a higher clock frequency increase the dynamic/static energy ratio, decreasing any of the two values has a positive effect on the total savings of ACME. Axserbus and Serial-T0 also show similar trends (although with a smaller slope), and the only encoding that benefits from larger f and R_p is ADE, which is not effective in reducing static energy. Overall, ACME remains vastly superior to its competitors across a wide range of frequencies and resistances. Furthermore, note that R_p and f cannot be increased *simultaneously*, since (for a given C_b), a smaller pull-up resistor is needed to support higher frequencies. Therefore, these graphs show that ACME is effective in all realistic electrical-level I^{2}C bus settings.

D. Hardware Implementation

Table III reports the synthesis results of the ACME encoder and decoder of Figure 3 on 45nm CMOS. The two circuits are synthesized targeting the same clock frequency of the ADE encoder/decoder of [11] for direct comparison, even if this is not an operating frequency for I^{2}C, and the bit-width is set to 12-bit. Power results include both leakage and dynamic consumption, and are reported both at f, to compare with ADE, and at 100kHz (i.e., the maximum frequency of I^{2}C Standard-mode).

TABLE III
COMPARISON OF SYNTHESIS RESULTS FOR 12-BIT ACME AND ADE ENCODER AND DECODER

Circuit	f [GHz]	Enc.	Area [μm^2]	Pow.@f [mW]	Pow.@100kHz [mW]
Encoder	1.67	ACME	130.18	0.27	0.001045
		ADE	170.40	0.33	-
Decoder	2.50	ACME	105.84	0.61	0.00084
		ADE	101.61	0.60	-

The results show that, besides being much more effective on reducing static energy on the bus, ACME yields a similar-cost hardware implementation in terms of area and power. Furthermore, for realistic I^{2}C frequencies, the power overheads of the encoding/decoding circuitry are completely negligible with respect to the consumption of the bus, as shown in [14].

V. CONCLUSIONS

We have proposed ACME, the first approximate serial bus encoding targeting I^{2}C buses. ACME offers a runtime-controllable energy saving vs error trade-off, and simultaneously reduces both dynamic and static energy by eliminating value-transitions and increasing the number of transmitted logic-1s. Despite its simplicity, this mechanism achieves superior total energy savings compared to those of state-of-the-art approximate encodings, with a negligible hardware implementation overhead.

REFERENCES

[1] Y. Kim et al, "AXSERBUS: A quality-configurable approximate serial bus for energy-efficient sensing," in *ACM/IEEE ISLPED*, 2017, pp. 1–6.
[2] J. Han et al, "Approximate computing: An emerging paradigm for energy-efficient design," in *IEEE ETS*, 2013, pp. 1–6.
[3] P. Stanley-Marbell et al, "Reducing Serial I/O Power in Error-tolerant Applications by Efficient Lossy Encoding," in *ACM/EDAC/IEEE DAC*, 2016, pp. 62:1—-62:6.
[4] D. Jahier Pagliari et al, "Serial T0: Approximate Bus Encoding for Energy-efficient Transmission of Sensor Signals," in *ACM/EDAC/IEEE DAC*, 2016, pp. 14:1—-14:6.
[5] S. Ghosh et al, "Data Correlation Aware Serial Encoding for Low Switching Power On-Chip Communication," in *IEEE ISVLSI*, 2014, pp. 124–129.
[6] K. Lee et al, "SILENT: serialized low energy transmission coding for on-chip interconnection networks," in *IEEE ICCAD*, 2004, pp. 448–451.
[7] M. Poncino et al, "Low-energy RGB color approximation for digital LCD interfaces," *IEEE TCE*, vol. 52, no. 3, pp. 1004–1012, 2006.
[8] D. Friesel et al, "I2C considered wasteful: saving energy with host-controlled pull-up resistors," in *ACM IPSN*, 2019, pp. 315–316.
[9] S. Salerno et al, "Limited Intra-Word Transition Codes: An Energy-Efficient Bus Encoding for LCD Display Interfaces," in *ACM/IEEE ISLPED*, 2004, pp. 206–211.
[10] J. Zeng et al, "Transition inversion coding with parity check for off-chip serial transmission," in *IEEE ICECS*, 2014, pp. 634–637.
[11] D. Jahier Pagliari et al, "Approximate Differential Encoding for Energy-Efficient Serial Communication," in *ACM/IEEE GLSVLSI*, 2016, pp. 421–426.
[12] X. Ren et al, "Adaptive Low-Power Transmission Coding for Serial Links in Network-on-Chip," *Proc. Engin.*, vol. 29, pp. 1618–1624, 2012.
[13] D. Jahier Pagliari et al, "Approximate Energy-Efficient Encoding for Serial Interfaces," *ACM TODAES*, vol. 22, no. 4, pp. 1–25, 2017.
[14] D. Jahier Pagliari et al, "Zero-Transition Serial Encoding for Image Sensors," *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2563–2571, 2017.
[15] G. Chaladze et al, "Linnaeus 5 dataset for machine learning," 2017.
[16] A. L. Goldberger et al, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.
[17] M. Shoaib et al, "Towards physical activity recognition using smartphone sensors," in *IEEE UIC/ATC*, 2013, pp. 80–87.
[18] NXP, "MMA8451Q 3-axis, 14-bit/8-bit digital accelerometer," *Data sheet*, 2014.
[19] Freescale, "Xtrinsic MAG3110 three-axis, digital magnetometer," *Data sheet*, 2013.
[20] STMicroelectronics, "L3G4200D. MEMS motion sensor: three-axis digital output gyroscope," *Data Sheet*, 2010.
[21] OSR, http://www.voiptroubleshooter.com/open_speech/index.html
[22] NXP, "I2C-bus specification and user manual," 2014.
[23] Patrascoiu, Nicolae. "A low cost solution to monitor environmental parameters in industrial area perimeters." MATEC Web of Conferences. Vol. 305. EDP Sciences, 2020.
[24] D. Jahier Pagliari et al, "On the impact of smart sensor approximations on the accuracy of machine learning tasks," *Heliyon*, vol. 6, no. 12, p. e05750, 2020.