Techniques for Modulating Error Resilience in Emerging Multi-Value Technologies

Magnus Själander
Department of Computer and Information Science
Norwegian University of Science and Technology
Department of Information Technology
Uppsala University, Sweden
magnus.sjalander@idi.ntnu.no

Gustaf Borgström
Department of Information Technology
Uppsala University, Sweden
gustaf.borgstrom@it.uu.se

Mykhailo V. Klymenko
Department of Chemistry
University of Liège, Belgium
mklymenko@ulg.ac.be

Françoise Remacle
Department of Chemistry
University of Liège, Belgium
fremacle@ulg.ac.be

Stefanos Kaxiras
Department of Information Technology
Uppsala University, Sweden
stefanos.kaxiars@it.uu.se

ABSTRACT

There exist extensive ongoing research efforts on emerging atomic scale technologies that have the potential to become an alternative to today's CMOS technologies. A common feature among the investigated technologies is that of multi-value devices, in particular, the possibility of implementing quaternary logic and memory. However, multi-value devices tend to be more sensitive to interferences and, thus, have reduced error resilience. We present an architecture based on multi-value devices where we can trade energy efficiency against error resilience. Important data are encoded in a more robust binary format while error tolerant data is encoded in a quaternary format. We show for eight benchmarks an average energy reduction of 14%, 20%, and 32% for the register file, level-one data cache, and main memory, respectively, and for three integer benchmarks, an energy reduction for arithmetic operations of up to 28%. We also show that for a quaternary technology to be viable a raw bit error rate of one error in 100 million or better is required.

Keywords

Approximate Computing, Energy Efficiency, Quaternary Logic, Emerging Technologies, ALU, Cache

CCS Concepts

•**Computer systems organization** → **Architectures;** *Processors and memory architectures;* Reliability; •**Hardware** → *Power and energy;*

CF'16, May 16 - 19, 2016, Como, Italy

ISBN 978-1-4503-4128-8/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2903150.2903154

1. INTRODUCTION

CMOS technology scaling, the current driver of Moore's law, is faced with practical and fundamental limits. The search for low-power alternatives to CMOS is intensifying with a large number of emerging technologies being investigated, e.g., single atom and single electron transistors [1,2]. Many such technologies present new features, which provide opportunities to develop new computer systems. Multi-level devices are one such common feature [3–5] that is already employed by several commercial memory technologies, e.g., in multi-level flash [6] and phase change memory [7]. Multi-level devices increase the on-chip information density, as they are capable of representing more than two distinct logical levels. Thus, multi-level devices have the potential of providing smaller, faster, and more energy-efficient circuit implementations as fewer wires, memory cells, and logic gates are required to transfer, store, and process a fixed amount of data. The gain in information density is commonly achieved at the cost of reduced error resilience of the multi-level devices. A multi-level device has more physical states, which by necessity reduce the margins between the individual states. Thus, making the device more susceptible to errors, see Figure 1.

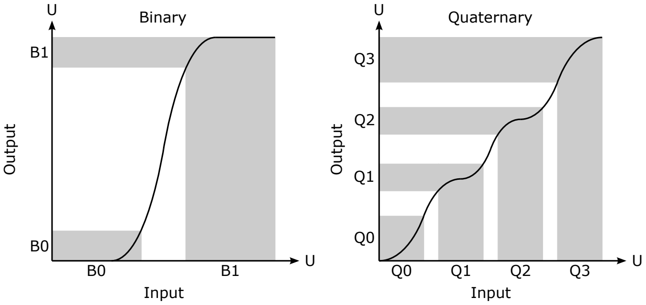

Figure 1: Illustration of input to output transfer functions of a fictitious binary and quaternary device, e.g., a memory cell or buffer. Note the much wider margins for the binary compared to the quaternary device.

There is an increasing research focus on approximate computing, i.e., on trading power, performance, and *reliability* against each other [8]. Previous work has shown that approximate computing can be used to improve performance and endurance of multi-level phase change memory (PCM) [9].

In this work, we present a technique where the resilience against errors for multi-value technologies can be modulated simply by the way data are encoded, Section 3. We leverage approximate computing to create an architecture where critical operations and storage of data are represented in a binary and more error resilient format, while less critical operations and data are represented in a quaternary and more compact format, Section 4. We evaluate our system by running eight different approximate benchmarks while varying the probability that an error manifests to estimate output quality and energy dissipation improvements of the system, Section 5. We show that significant energy reductions can be achieved while still producing adequate results, Section 6.

2. EMERGING NANOELECTRONICS

The transfer characteristic shown in Figure 1 is typical for so-called current quantizers [10], which are often encountered in nanoelectronics and are considered for applications in multi-valued logic systems. The current quantizers can be based on the charge states of a single electron transistor (SET) characterized by periodic Coulomb oscillations causing, in turn, the negative differential resistance. The Coulomb oscillations are transformed into a step-wise transfer function by combining the SET with a conventional MOSFET in a simple electrical circuit where the MOSFET serves to stabilize the electrical current through the SET [10]. Also, the quantization of electrical current naturally appears in a single-atom transistor (SAT) [11] where an electron is confined in such a small volume that a discrete energy spectrum within a given charge configuration can be observed. Continued improvements in nanotechnology, making it possible to fabricate smaller objects with better control of their shapes, leads to operating temperatures of SET and SAT that reach room temperature. A recent example is that of a silicon nanowire FET showing Coulomb blockade at room temperature [12].

The two examples, mentioned above, involve a control over the charge state by a static gate voltage. Another, more complicated but also more robust, way to generate a step-wise transfer characteristic is to apply a proper alternating signal to the gate electrode implementing a turnstile single-electron pump. For instance, J. P. Pekola et al. [13] propose a source of quantized electric current based on a SET consisting of two junctions between normal and superconducting metals driven by an alternating gate voltage. In their device, the current between source and drain is a step-wise function of the gate amplitude. K. Nishiguchi et al. [14] implement a step-wise transfer function defined by the electrical current through a charge sensor attached to a quantum box where the number of electrons is controlled by a turnstile single-electron pump.

Although all these examples are very different, most of them have a common feature that their step-wise transfer characteristics suffer from thermal noise, shot noise and 1/f noise caused by the background charge fluctuations. The statistics of these fluctuations can be controlled by temperature or by controlling the electrostatics through a gate electrode. However, the general trend is that better single-electron charge control, allowing to resolve the quantization of electrical current, leads to slower device operation and/or requires a low-temperature regime of operation.

In the SET current quantizer, the most probable bit error associated with those fluctuations is caused by random changes in the electron population of the quantum dot. At low temperatures, these jumps are dominated by the shot noise and 1/f noise involving changing of the population by a single electron. Therefore, the most probable bit error at reasonably small noise level at the gate electrode of SET is a random jump between adjacent quantization levels. Fluctuations between others levels are less probable. Consequently, operating in the mode when all quantization levels are involved is less robust comparing to the mode when the logic is based on the charge states separated by several units of the charging energy. However, the first case is characterized by higher information density implying less energy dissipation.

3. MODULATING ERROR RESILIENCE

We consider any device where the output can be described as a continuous transfer function of its inputs. For multi-level logic, this transfer function contains plateaus that presents a relatively stable output even if the input value varies slightly [15]. These plateaus appear as discrete ranges that can be used to represent logic states. Such devices have higher probability that the output of a device will shift, due to interference, to a nearby level than to a level far away from the expected level. Here we use the term interference for any physical property that affects the output such that it deviates from the ideal transfer function, e.g., input noise or manufacturing variances. We take advantage of this property to create two logic representations for representing approximate and precise data.

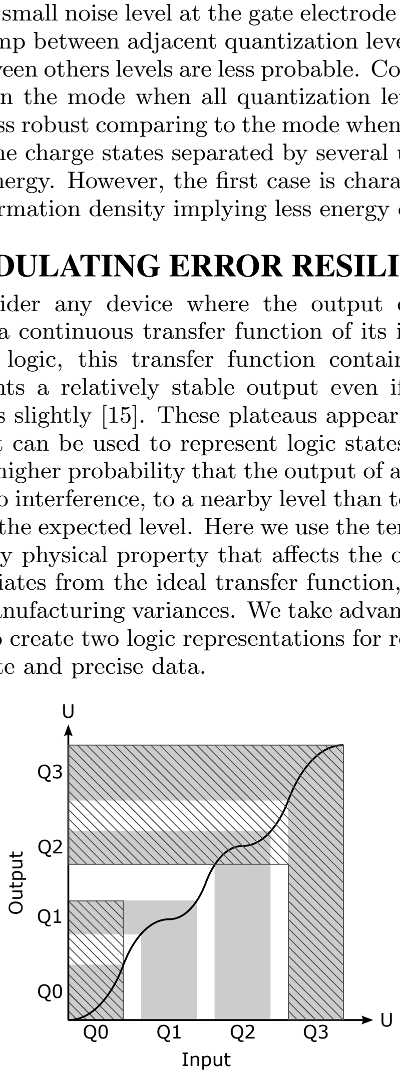

Figure 2: Illustration of a quaternary device that can operate on binary data. Gray regions represent the input and output ranges when working with quaternary data. Striped regions represent input and output ranges when working with binary data.

Figure 2 illustrates a transfer function of a fictitious quaternary device in some unit (U), e.g., voltage or current. The figure shows in light gray the four different quaternary states (Q0, Q1, Q2, and Q3) and how they are represented as ranges on the input and output of the device. When representing approximate data, we use all logic states of the device. This maximizes the amount of data that can be represented per device, but it also increases the probability that

OR

B\A	0	1	2	3
0	0	1	2	3
1	1	1	3	3
2	2	3	2	3
3	3	3	3	3

AND

B\A	0	1	2	3
0	0	0	0	0
1	0	1	0	1
2	0	0	2	2
3	0	1	2	3

XOR

B\A	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

MAX

B\A	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	2	3
3	3	3	3	3

MIN

B\A	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	2	2
3	0	1	2	3

INV

A	$\overline{A}$
0	3
1	2
2	1
3	0

Figure 3: Truth tables of quaternary gates with their binary operations highlighted in gray.

a state shifts to a nearby state (e.g., Q2 gets interpreted as Q1 or Q3). Precise data is represented in binary form by only using the maximum (Q3) and minimum (Q0) states as input to the device. When the output is read nearby states are also considered as part of the maximum (Q2 and Q3) and minimum (Q0 and Q1) states (see striped regions). This increases the reliability of the device as the probability is low that Q0 would shift to Q2 or that Q3 would shift to Q1. However, it also requires (at least) twice the number of devices compared to representing the data approximately.

4. SYSTEM ARCHITECTURE

We have designed a complete system based on the principal of improved error resilience through the use of performing binary operations on quaternary devices (see Section 3).

4.1 Logic and Arithmetic Operations

Logical operations on quaternary gates work without any special consideration when operating on binary data, as seen in Figure 3. A quaternary zero (Q0) also represents a binary zero (B0) while a quaternary three (Q3) represents a binary one (B1), highlighted in gray in the figure. From the figure, it is apparent that if the three in the gray areas would be replaced with one then the gray areas represent the equivalent binary operation.

Performing additions on binary data using quaternary devices are not as straightforward. Considering the basic operation of a half (HA) and full adder (FA) the sum is not correct for all inputs. When adding two binary ones (quaternary three), the resulting sum becomes quaternary two when we instead need a quaternary three for a correct result, see Figure 4. Thus, it is not possible to design a ripple-carry adder simply from full-adder gates.

Ripple-carry adders are relatively slow and in most cases a more performant adder implementation is desirable. A carry-lookahead adder (CLA) is commonly used when implementing high-speed adders. The carry-lookahead technique is based on the creation of a generate (G) and propagate (P) signal for each significance pair of the input operands, see Figure 5. The generate and propagate signals are then used to calculate the carry for each digit. The key insight is that the carry (and the generate and propagate) signal is never anything other than zero or one, regardless, of which base the input operands have. The generate and propagate structure constitutes the majority of all gates in a CLA and all of them perform binary operations. Only the periphery circuits, i.e., the initial generate and propagate circuit and final summation circuit, operate at a higher base.

HA

B\A	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

FA (carry=1)

B\A	0	1	2	3
0	1	2	3	0
1	2	3	0	1
2	3	0	1	2
3	0	1	2	3

Figure 4: Truth table showing the sum of a quaternary half and full adder. Gray regions highlight erroneous binary representation.

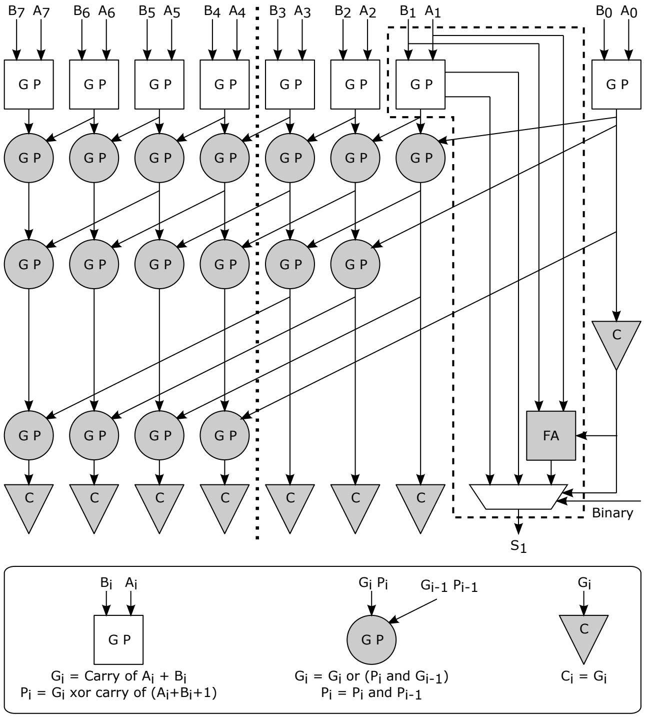

Figure 5: Illustration of a quaternary Kogge-Stone adder with the capability of operating on binary data. Gates in white perform quaternary operations, all other gates perform binary operations.

Figure 5 shows a Kogge-Stone [16] CLA that can perform both quaternary and binary additions. Gates in white are quaternary while all other gates perform binary operations. The initial generate signal is one when the input-pair of the operands would generate a carry and zero in all other cases, i.e., the carry of an HA. The initial propagate signal is one when the sum of the input-pair is exactly three, i.e., a carry is generated if a carry would come from the next lower significance level, hence the name propagate. The generate and propagate can hence be created as depicted in Figure 6 using the carries from a quaternary HA and FA (Figure 7).

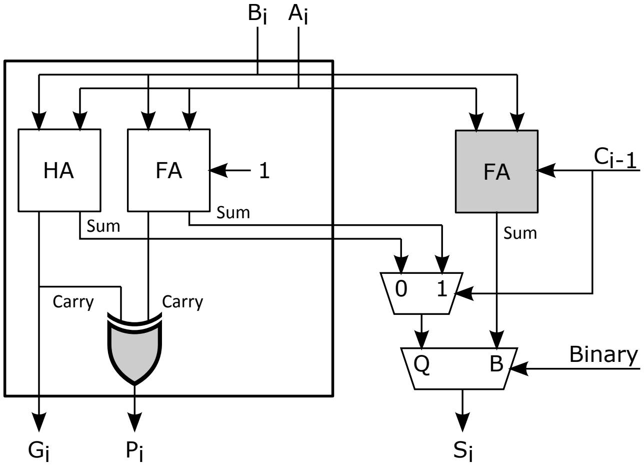

Figure 6: Circuits for the generate and propagate signals and the computation of the final sum.

HA

B \ A	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1

FA (carry=1)

B \ A	0	1	2	3
0	0	0	0	1
1	0	0	1	1
2	0	1	1	1
3	1	1	1	1

Figure 7: Truth table showing the carry of a quaternary half and full adder. Gray regions highlight binary representation. Note that the result is only zero or one since it is the carry that is shown.

As the quaternary HA and FA does not generate a correct binary sum (see Figure 4) a binary FA and a two-input multiplexor are required for each digit, see Figure 6. The delay through the binary FA can be made equal to that of the quaternary path. The additional multiplexor adds negligible delay to the critical path, since the *Binary* control signal of the multiplexor is available directly from the start of the addition. Thus, even though the control is a high-fanout signal it will have been applied to the multiplexor before the sums have been calculated.

Subtraction can be handled as in conventional binary implementations by creating the complement of the subtrahend and then perform an addition with the minuend. The complement is created by inverting the subtrahend. Multiplication and division are more challenging and are left as future work. One simple approach would be to convert approximate data to precise data, perform the multiplication or division, and then convert it back.

4.2 Memory Hierarchy

We base the cache on the idea by M. Själander et. al. [17]. The memories storing data consist of quaternary memory cells, e.g., the single-electron memory by H. Inokawa et. al. [18]. The cache lines are organized such that each line stores N quaternary bytes (q-byte). Binary bytes (b-byte) can be stored by combining pairs of q-bytes as shown by the striped region of a quaternary word (q-word) in Figure 8.

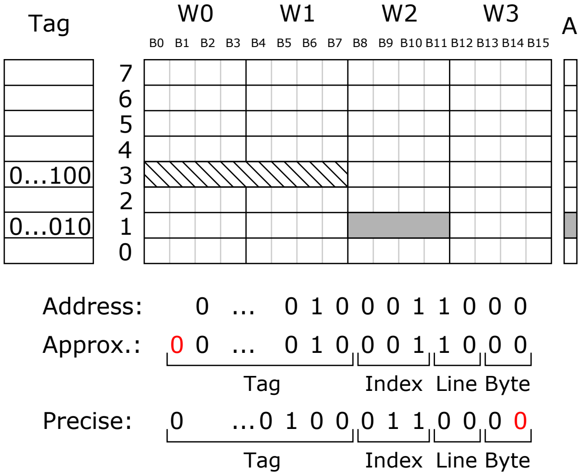
Tag
W0
W1
W2
W3
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15
A
7
6
5
4
3
2
1
0
0...100
0...010
Address: 0 ... 0 1 0 0 0 1 1 0 0 0
Approx.: 0 0 ... 0 1 0 0 0 1 1 0 0 0
Tag
Index
Line
Byte
Precise: 0 ...0 1 0 0 0 1 1 0 0 0
Tag
Index
Line
Byte

Figure 8: One way of an approximate cache, with eight sets, sixteen approximate q-bytes per cache line ($N = 16$), and four q-bytes per q-word.

Addressing memory becomes more complicated when bytes of different size have to coexist in the same memory system. The same address will have to access different physical locations depending on the type of the access (binary or quaternary). When accessing quaternary data, the address matches the granularity of the physical organization and, therefore, does not have to be modified. When accessing binary data, the physical organization does not match, since two q-bytes are needed for one b-byte. Thus, only half the number of b-bytes fit in a cache line, which needs to be reflected by how the data are addressed. This is achieved by shifting the address one step to the left, (see Precise in Figure 8). The shift of the address for binary accesses increases the address size. To create equally sized addresses, an approximate address is prepended with a most significant zero, (see Approximate in Figure 8). The resulting addressing scheme has the affect that two different addresses could map to the same physical location. To distinguish between the two addresses, a cache line is therefore only allowed to contain one type of data and a bit (A in Figure 8) is used to indicate the type of data that the line contains.

Binary and quaternary data should not be intermixed too finely in the address space to avoid fragmentation with unusable space. For efficient use of available cache space the smallest allocated space for binary or quaternary data should be a whole cache line that is cache line aligned. This is likely smaller than the smallest size that is desirable for main memory. For simplicity, the OS page size can be used as the smallest granularity.

Registers can be handled such that even registers can be used to store binary data while all registers can be used to store quaternary data. If an even register (i) holds binary data then the following odd register ($i + 1$) cannot be used for storing quaternary data. A field for each pair of registers can then be used to indicate if it contains binary data.

4.3 Modulating the Precision

A desirable property of an approximate computing system is the ability to control the tradeoff between precision and energy. Reducing the supply voltage causes a quadratic reduction in power and attempts have therefore been made to overscale the voltage without adjusting the frequency [19].

A significant problem with voltage overscaling is that the precision does not scale gracefully with the voltage. For memory, it is just as likely that the most significant digit will flip as the least significant digit. For arithmetic circuits, the critical path is generally the one for the most significant digit, i.e., when overscaling the voltage it is more likely that an error will manifest at the more significant digits than the less significant digits. Different circuit and implementation techniques have been attempted to more gracefully scale the errors with the voltage through either various error correcting techniques [20] or slack redistribution [21] to avoid affecting the more significant digits.

With the ability to control the error resilience simply by how the data are encoded it is possible to control the precision at the bit level. If the data have no strict constraints on precision then all digits can be encoded as quaternary to minimize the amount of active logic that is required to compute, transfer, and store it. On the other hand, if certain guaranties on precision are required then a suitable number of the more significant digits can be encoded as binary while the less significant digits can be encoded as quaternary. By increasing the number of more significant digits that are encoded as binary the precision can be improved at the cost of more active logic and thus a higher energy dissipation. For logic operations, this tradeoff is literally controlled by the encoding of the data without any required modifications of the circuits. To be able to control the precision of an adder (Figure 5) the binary-control signal has to be applied to the appropriate digits. One drawback with the mixed encoding (i.e., both binary and quaternary encoding in the same data word) is that the cache optimization described in Section 4.2 is not possible. Only fully approximate words for which the size becomes half the size of a precise word can benefit from this technique. Mixed-encoded words still benefit from lower energy dissipation as fewer memory cells have to be read and written when accessing the word in the cache.

4.4 System

An application has to be able to convey information about what type of operation or memory access that is to be performed. To avoid the need of a specialized instruction set architecture with both approximate and precise instructions we instead keep this information together with the data in the architecture. The load instructions are modified such that the specify the number of digits that are encoded quaternary and this information is kept together with the loaded data (e.g., the A field for each cache line and pair of registers in Figure 8)[1]. This additional metadata is used to determine the mode (precise or approximate) when accessing data from registers and the caches as well as when performing logic and arithmetic operations. A set of new instructions is introduced to convert between approximate and precise data types. We use the term Approx32 for this approximate mode to clearly indicate that all digits are quaternary.

The precise representation is not immune to interference and unintentional changes from Q0 to Q1 or from Q3 to Q2 might occur. Depending on a particular technologies probability for this type of change precise data could, e.g., be restored when reading data from the cache (low probability), from the register file (medium probability), or from pipeline registers (high probability).

5. EVALUATION

We have used EnerJ [22] to classify data as either approximate or precise and to simulate the proposed architecture. EnerJ is a type system for Java and enables programmers to specify which data that can be treated approximately. The type system assures that approximate data are guarded in such a way that it does not bleed into and affect precise operations and data.

In the original version of EnerJ data can either be specified as precise or as approximate (by using the @Approx annotation). For this evaluation, we have chosen to extend this annotation by introducing three new annotations @Approx8, @Approx16, and @Approx24, which enables a programmer to specify the amount of approximation that an application can tolerate. The new types specify how many of the least significant bits of an originally binary data type that are encoded as quaternary (i.e., the four least significant digits of an @Approx8 data type are quaternary and the remaining digits are binary). The original @Approx keyword signifies that the whole data type is encoded as quaternary.

The EnerJ classified data are only available within the java runtime environment (JRE) and are not exposed to the architecture itself. Thus, it is not possible to use a conventional architectural simulator in combination with EnerJ. Instead, the architectural modeling needs to be performed within the JRE itself. We have extended the EnerJ JRE with our own cache model (described in Section 4.2) and replaced existing error models with error models for our intended quaternary technology.

Performing architectural simulations within the EnerJ JRE instead of on an architectural simulator impose limits on the evaluations that can be performed. It is, e.g., not possible to accurately determine the timing, as the architectural modeling is not separated from the execution of the application. There is no way of determining how much time is used for executing the application versus the time used for modeling the architectural behavior. We are therefore only evaluating energy aspects even though the use of quaternary data improves the hit rate for caches and enables more data to be kept in registers. These improvements would have a positive impact on performance and, thus, improve energy even further than what is being reported.

5.1 Benchmarks

We use six benchmarks (scimark2 [lu, smm, sor], imagefill, jmeint, and raytrace) that been developed together with the EnerJ framework and two benchmarks (sobel and fft) that we have developed ourselves to evaluate the proposed architecture. All benchmarks have been annotated with EnerJ's approximate annotations to indicate which data and operations that can be treated approximately by the architecture.

5.2 Error Models

Error correcting codes (ECC) are commonly used to protect data stored in memories. This is especially true for multi-value memories (flash, PCM), which have high raw bit error rates (RBER). We assume that all data are protected by an ECC that can detect and correct three bits and use the equation by N. Mielke et al. to calculate the effective bit error rate given a RBER [23]. For logical operations, we apply the RBER directly on each bit and use the functional model to estimate the bit error rate for individual bits of additions and subtractions.

[1] A single A field for the whole cache line requires that different approximate data types are cache-line aligned.

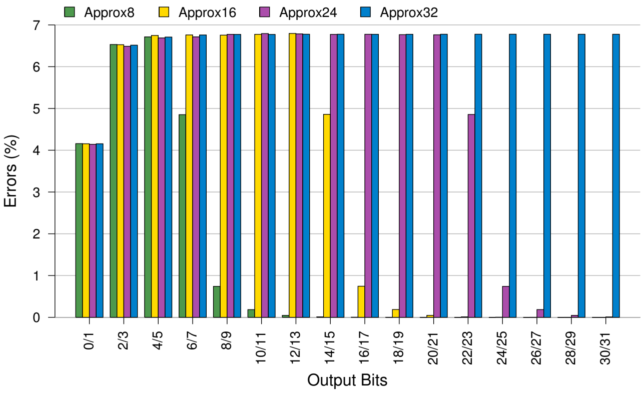

Figure 9: Error probabilities for each output bit pair when performing additions with different modes of approximation on a 32-bit Kogge-Stone adder under the assumption of an RBER of 1%.

We have implemented a functional model of the Kogge-Stone adder described in Section 4.1 and used it to create an error model. We inject errors at the input of the quaternary gates and observe how these propagate to the final result. The probability for error(s) at the output is then recorded and used for the architectural simulations. Figure 9 shows an example where the RBER was arbitrarily chosen to one in 100 (1% chance). Since the output both contains quaternary and binary digits the data are grouped into bit pairs, e.g., 0/1 and 2/3. The figure also shows the error probability for the four different modes of approximation (8, 16, 24, and 32 bits encoded quaternary). It is clearly shown that the probability for an error in the more significant digits rapidly decrease at the point where the digits are encoded as binary. Because of the carry chain, it is possible for an error that appears in the quaternary part to propagate into the binary, more significant part. But, the probability for an error in the more significant part is drastically reduced compared to only using the quaternary encoding.

5.3 Energy Estimation

We use switching gate equivalents to estimate the energy of the adder in precise and approximate modes. For the approximate mode where all digits are quaternary the gates that switch for creating the initial generate and propagate signals and the final sum are reduced by half. The carry-tree (the gray GP-dots in Figure 5) has a higher switching activity as the generate and propagate signals of the lower half (right side of the dotted line) enters the upper half (left side of the dotted line). This causes switching at the inputs of GP-dots in the upper half, but will not cause any switching of their outputs. For a 32-bit Kogge-Stone adder, 80 of the 129 GP-dots will have inputs that switch. We therefore conservatively estimate the energy of an approximate addition to being 62% (80/129) of a precise addition.

Table 1: Energy Estimate Comparison Between Kogge-Stone and Brent-Kung

	Approx8	Approx16	Approx24	Approx32
Kogge-Stone	124/129 (96%)	112/129 (87%)	96/129 (74%)	80/129 (62%)
Brent-Kung	52/57 (91%)	46/57 (81%)	38/57 (67%)	31/57 (54%)

Table 1 shows the number of switching gates in the carry-tree of a Kogge-Stone and Brent-Kung adder under different modes of approximation. The energy reduction for the Kogge-Stone is almost 40% when operating on Approx32 data while only a marginal improvement of 4% is achieved when operating on Approx8 data. Kogge-Stone is only one example of a carry-lookahead adder (CLA) and there exist many different organizations in terms of number of gates and connectivity that trade speed, power, and area against each other. For our approximate adder the connectivity between less significant parts to more significant parts also matters as it will determine how much of the carry tree is active during the different approximate modes. It is outside the scope of this work to evaluate the most suitable CLA configuration. Here, we instead list one alternative, Brent-Kung, that reduce the switching activity compared to the Kogge-Stone. For Approx32 operations, the energy almost reaches the ideal reduction of 50% and for Approx8, we see a more substantial reduction of almost 10%. Unless it is explicitly mentioned the presented data are with the Kogge-Stone results.

The cache has to be designed in such a way that data of variable width can be read from it to save energy. This can be achieved by designing the cache such that most modes of operations access two SRAM memories while during the Approx32 mode only one of the SRAMs is accessed. We estimate the energy savings by modeling a single way of a cache with a 4kB data array and 32-byte cache line size. The tag array is modeled as a 128 byte SRAM with a 32-bit wide port. According to CACTI [24] (version 6.5) the energy dissipation of the tag array is 32% of the data array (when using the 32nm ITRS-HP technology at 350K and optimizing for energy). The estimate of a 2kB 16-bit wide data array the energy dissipation is estimated to be 52.4% of the 4kB data array (100%). Designing the cache with two 16-bit SRAMs instead of a single 32-bit SRAM incurs a slight overhead of 4% $((32\% + 2 \times 52.4\%)/(32\% + 100\%))$ for precise accesses while Approx32 accesses only dissipate 64% $((32\% + 52.4\%)/(32\% + 100\%))$.

For the main memory we do not save in terms of access power as a whole cache line is read and written at a time. The benefit of having Approx32 data is that twice the number of data items are read/written than compared to precise data. This reduces the total number of required accesses to main memory, which saves energy.

We scale the energy gain with the number of bits that are accessed when accessing data for other approximate modes than Approx32, i.e., Approx8 access 28 of the 32 memory cells in a precise word.

6. RESULTS

We first present results without modulating the approximation (fixed approximation), i.e., only Approx32 is used. This identifies the best possible energy reduction that can be achieved when using the proposed system. Next we present results when trading energy against precision by modulating the approximation.

6.1 Fixed Approximation

Figure 10 shows the correctness of the output for the eight benchmarks when the REBR is changed from one error in 100 to one in 100 million. We ran each benchmark 100 times for each REBR and the graph shows the average, minimum,

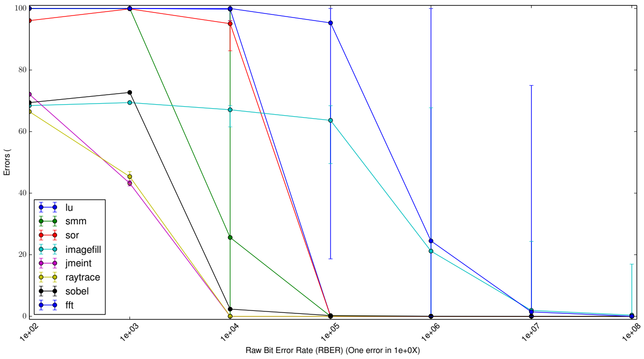

Figure 10: Output errors of eight benchmarks when the raw bit error rate (RBER) is changed from one in 100 to one in 100 million. The whiskers show the maximum and minimum error while the line shows the average for 100 runs.

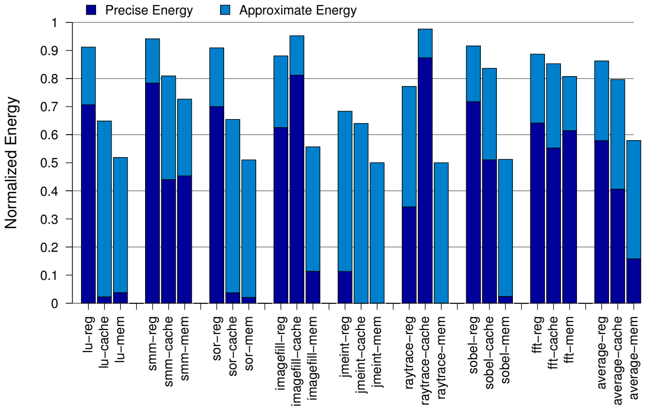

Figure 11: Relative register file (reg) and cache energy when executing approximately.

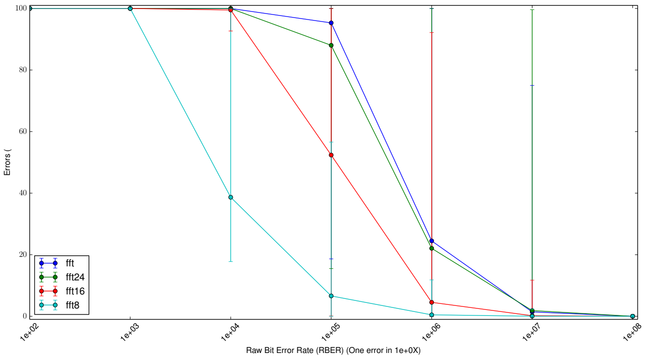

Figure 12: Output errors when modulating the approximation.

Table 2: L1 DC Miss Rate

Benchmark	Original (%)	Approximate (%)	Improvement (%)
lu	5.13	2.67	48%
smm	7.09	4.37	38%
sor	2.13	1.08	49%
imagefill	18.21	17.89	2%
jmeint	1.30	0.65	50%
raytrace	0.50	0.25	50%
sobel	0.10	0.05	50%
fft	1.65	0.02	99%

and maximum correctness. As seen in the figure, six of the benchmarks produce correct results for all 100 runs at an REBR of one in 100,000. However, for imagefill and fft an REBR of one in 100 million is needed to get close to reliable outputs. An REBR of one in 10-100 million has been reported for both PCM and Flash [23, 25–27]. Our results indicate that a quaternary technology would have to reach at least this level of correctness for it to be viable.

Figure 13 shows the output of a randomly selected run from imagefill under varying RBER. The result for an error rate of one in 100 or 1,000 is not usable, but already at an error rate of 1 in 10,000 the result could be of use in certain situations. So even though imagefill does present errors for an RBER of one in 10 million (the particular run in Figure 13 had a single error) the result can be useful.

Figure 11 shows the energy usage for the register file and cache normalized to a system which would only use precise operations. The average register file, cache energy, and main memory savings are 14%, 20%, and 32%, respectively. Jmeint shows the greatest improvement with 32% register-file, 36% cache, and 50% main-memory energy savings. This is due to the high fraction of data that can be treated approximately in jmeint. The reason for main memory showing the greatest improvement is due to locality of the different types of data. Precise data tend to be constants and consist of a smaller working set, which fits in the cache. While approximate data tends to be arrays that are strided with less temporal locality. The more compact representation of approximate data makes it possible to store more data in the cache. This improves the miss rate and, thus, reduces the number of accesses to main memory.

Table 2 shows the miss rate for the modeled level one (L1) data cache (DC), 4-way, 16KiB, with 32-byte line size. The column denoted original shows the miss rate for the eight benchmarks when approximate computing is not used. The last column shows the improvements in miss rate that are achieved when storing data approximately in the cache. For most of the benchmarks the improvements are close to 50% while for fft almost all cache misses can be avoided as the working set now fits in the L1 DC when storing it approximately. The reduced miss rate has the effect of fewer main memory accesses, which improves both energy efficiency (see Figure 11) and performance.

Only three of the benchmarks use approximate integer operations (imagefill, sobel, and fft). The energy for arithmetic operations is reduced by 11%, 23%, and 16% for imagefill, sobel, and fft, respectively.

6.2 Modulated Approximation

We use the fft application to evaluate the benefits of modulating the approximation. The reason for only using fft is that it is the only application that only use integer operations (most benchmarks use floating point) to include the effect of the adder and that operate on 32-bit wide data (both imagefill and sobel operate on pixels with 8-bit values).

Figure 12 shows how modulating the approximation affects the amount of errors in the output of the application. FFT is the same as shown in Figure 10 and require an REBR of 1 in 100 million to not provide any errors in our case. Using Approx24 instead of Approx32 does hardly provide any

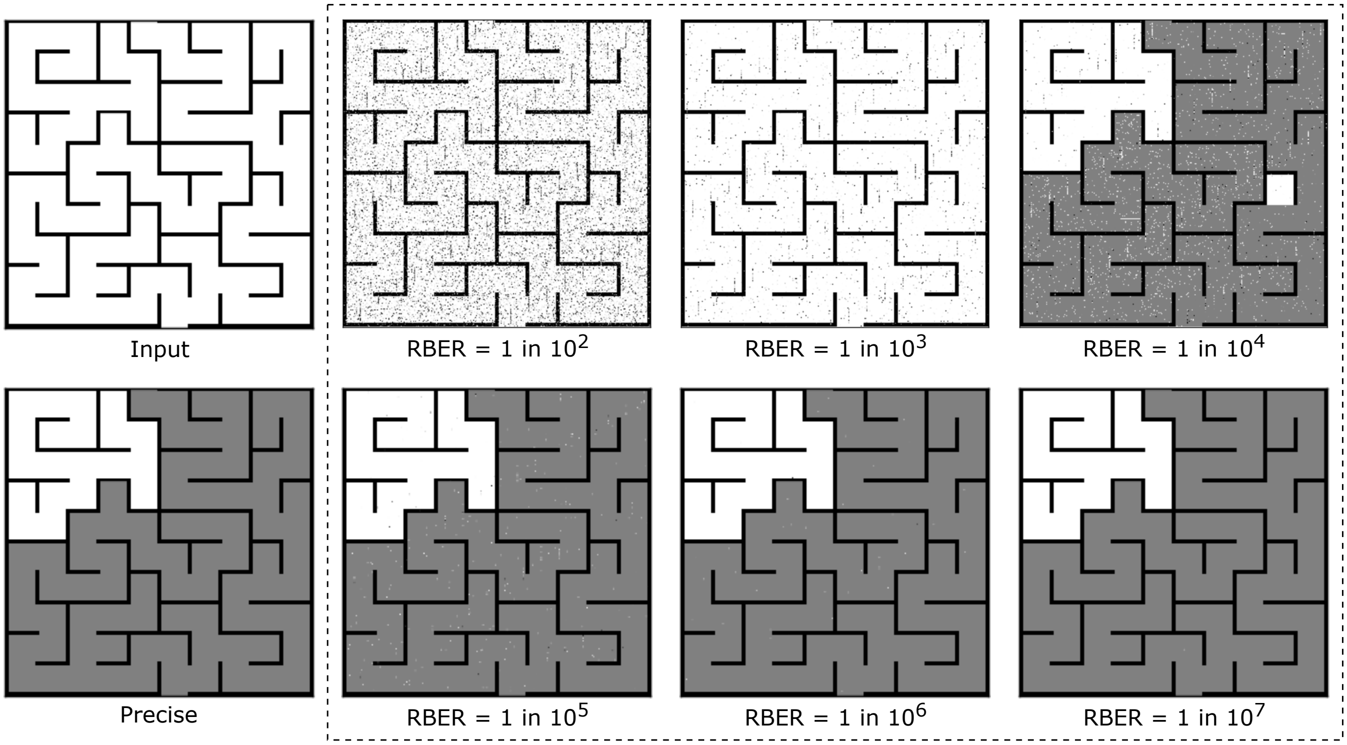

Figure 13: Example outputs from the imagefill benchmark under different RBER.

Table 3: Energy Estimate for fft when Modulating the Approximation.

	Approx8	Approx16	Approx24	Approx32
Arithmetic	94%	88%	80%	72%
Register	95%	90%	86%	81%
Cache	94%	88%	82%	77%
Memory	95%	90%	86%	81%

benefit at all as the two curves tracks each other closely (fft24 and fft, respectively). However, when reducing the approximation to Approx16 the amount of errors in the output is significantly reduced and an RBER of 1 in 10 million might be acceptable for some use cases. Reducing the approximation further to Approx8 another magnitude increase in the REBR could be acceptable.

Table 3 shows the energy reduction for arithmetic operations and the memory system when modulating the approximation. Here we use the energy estimates for the Brent-Kung adder instead of the Kogge-Stone as it shows better results when modulating the approximation. This is why the Approx32 results in a reduction of 28% instead of 16%, as reported in Section 6.1.

7. CONCLUSION

The digital age has come to rely on rapid improvements in performance, which has been driven by ever cheaper devices as predicted by Moore's law. However, contemporary CMOS technologies are reaching their practical and physical limits. An alternative could be one of the many researched and emerging nano technologies. Operating at the nano scale poses many challenges, one of which is reliable operations. We have presented a technique where reliability can be modulated for a multi-value technology. By encoding data either as binary or quaternary it is possible to trade energy efficiency against reliability. With this technique, we might be one step closer to building systems based on emerging multi-value technologies.

Acknowledgment

This work was supported by the proactive collaborative project TOLOP (318397) of the Seventh Framework Program of the European Commission.

8. REFERENCES

[1] J. Verduijn, G. C. Tettamanzi, and S. Rogge. Wave function control over a single donor atom. *Nano Letters*, 13(4):1476–1480, 2013.

[2] V. Deshpande, R. Wacquez, M. Vinet, X. Jehl, S. Barraud, R. Coquand, B. Roche, B. Voisin, C. Vizioz, B. Previtali, L. Tosti, P. Perreau, T. Poiroux, M. Sanquer, B. De Salvo, and O. Faynot. 300 K operating full-CMOS integrated single electron transistor (SET)-FET circuits. In *International IEEE Electron Devices Meeting*, pages 8.7.1–8.7.4, December 2012.

[3] MV Klymenko and F Remacle. Quantum dot ternary-valued full-adder: Logic synthesis by a multiobjective design optimization based on a genetic algorithm. *Journal of Applied Physics*, 116(16):164316, 2014.

[4] M Seo, C Hong, S-Y Lee, HK Choi, N Kim, Y Chung, V Umansky, and D Mahalu. Multi-valued logic gates based on ballistic transport in quantum point contacts. *Scientific reports*, 4, 2014.

[5] F Remacle, JR Heath, and RD Levine. Electrical addressing of confined quantum systems for quasiclassical computation and finite state logic machines. *Proceedings of the National Academy of Sciences of the United States of America*, 102(16):5653–5658, 2005.

[6] Dae-Seok Byeon, Sung-Soo Lee, Young-Ho Lim, Jin-Sung Park, Wook-Kee Han, Pan-Suk Kwak, Dong-Hwan Kim, Dong-Hyuk Chae, Seung-Hyun Moon, Seung-Jae Lee, Hyun-Chul Cho, Jung-Woo Lee, Moo-Sung Kim, Joon-Sung Yang, Young-Woo Park, Duk-Won Bae, Jung-Dal Choi, Sung-Hoi Hur, and Kang-Deog Suh. An 8 Gb multi-level NAND flash memory with 63 nm STI CMOS process technology. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 46–47 Vol. 1, February 2005.

[7] H.-S.P. Wong, S. Raoux, SangBum Kim, Jiale Liang, John P. Reifenberg, B. Rajendran, Mehdi Asheghi, and Kenneth E. Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, December 2010.

[8] Jie Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Proceedings of the IEEE European Test Symposium*, pages 1–6, May 2013.

[9] Adrian Sampson, Jacob Nelson, Karin Strauss, and Luis Ceze. Approximate storage in solid-state memories. In *Proceedings of the ACM/IEEE Annual International Symposium on Microarchitecture*, pages 25–36, December 2013.

[10] Hiroshi Inokawa, Akira Fujiwara, and Yasuo Takahashi. A multiple-valued logic and memory with combined single-electron and metal-oxide-semiconductor transistors. *IEEE Transactions on Electron Devices*, 50(2):462–470, 2003.

[11] Michael Klein, Gabriel P Lansbergen, Jan A Mol, Sven Rogge, Raphael D Levine, and Francoise Remacle. Reconfigurable logic devices on a single dopant atom-operation up to a full adder by using electrical spectroscopy. *ChemPhysChem*, 10(1):162–173, 2009.

[12] Romain Lavieville, François Triozon, Sylvain Barraud, Andrea Corna, Xavier Jehl, Marc Sanquer, Jing Li, Antoine Abisset, Ivan Duchemin, and Yann-Michel Niquet. Quantum dot made in metal oxide silicon-nanowire field effect transistor working at room temperature. *Nano letters*, 2015.

[13] Jukka P Pekola, Juha J Vartiainen, Mikko Möttönen, Olli-Pentti Saira, Matthias Meschke, and Dmitri V Averin. Hybrid single-electron transistor as a source of quantized electric current. *Nature Physics*, 4(2):120–124, 2008.

[14] K. Nishiguchi, Y. Ono, and A. Fujiwara. Operation of silicon single-electron devices at room temperature. *NTT Technical Reviews Letters*, 5(6):1–6, 2007.

[15] M. Seo, C. Hong, S.-Y. Lee, H. K. Choi, N. Kim, Y. Chung, V. Umansky, and D. Mahalu. Multi-valued logic gates based on ballistic transport in quantum point contacts. *Scientific Reports*, 4, January 2014.

[16] Peter M Kogge and Harold S Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, 100(8):786–793, 1973.

[17] M. Själander, N. S. Nilsson, and S. Kaxiras. A tunable cache for approximate computing. In *Proceedings of the IEEE International Symposium on Nanoscale Architecture*, pages 88–89, July 2014.

[18] H. Inokawa, A. Fujiwara, and Y. Takahashi. A multiple-valued SRAM with combined single-electron and MOS transistors. In *Proceedings of the Device Research Conference*, pages 129–130, June 2001.

[19] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger. Architecture support for disciplined approximate programming. In *Proceedings of the Architectural Support for Programming Languages and Operating Systems*, pages 301–312, 2012.

[20] Debabrata Mohapatra, Vinay K Chippa, Anand Raghunathan, and Kaushik Roy. Design of voltage-scalable meta-functions for approximate computing. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1–6. IEEE, 2011.

[21] Andrew B Kahng, Seokhyeong Kang, Rakesh Kumar, and John Sartori. Slack redistribution for graceful degradation under voltage overscaling. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 825–831. IEEE, 2010.

[22] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. EnerJ: Approximate data types for safe and general low-power computation. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 164–174, June 2011.

[23] Neal Mielke, Todd Marquart, Ning Wu, Jeff Kessenich, Hanmant Belgal, Eric Schares, Falgun Trivedi, Evan Goodness, and Leland R Nevill. Bit error rate in NAND flash memories. In *Proceedings of the IEEE International Reliability Physics Symposium*, pages 9–19. IEEE, 2008.

[24] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. CACTI 6.0: A tool to model large caches. Technical Report HPL-2009-85, HP Laboratories, April 2009.

[25] Daniele Ielmini, Andrea L Lacaita, and Davide Mantegazza. Recovery and drift dynamics of resistance and threshold voltages in phase-change memories. *IEEE Transactions on Electron Devices*, 54(2):308–315, 2007.

[26] S. Yeo, N. H. Seong, and H.-H. S. Lee. Can multi-level cell PCM be reliable and usable? Analyzing the impact of resistance drift. In *Proceedings of the Workshop on Duplicating, Deconstructing and Debunking*, June 2012.

[27] Yu Cai, Erich F Haratsch, Onur Mutlu, and Ken Mai. Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 521–526. IEEE, 2012.