Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2015-022 June 4, 2015

Value-Deviation-Bounded Serial Data Encoding for Energy-Efficient Approximate Communication

Phillip Stanley-Marbell and Martin Rinard

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Value-Deviation-Bounded Serial Data Encoding for Energy-Efficient Approximate Communication

Phillip Stanley-Marbell Martin Rinard
MIT
psm@mit.edu, rinard@csail.mit.edu

Abstract

Transferring data between ICs accounts for a growing proportion of system power in wearable and mobile systems. Reducing signal transitions reduces the dynamic power dissipated in this data transfer, but traditional approaches cannot be applied when the transfer interfaces are serial buses. To address this challenge, we present a family of optimal *value-deviation-bounded approximate serial encoders* (VDBS encoders) that significantly reduce signal transitions (and hence, dynamic power) for bit-serial communication interfaces. When the data in transfer are from sensors, VDBS encoding enables a tradeoff between power efficiency and application fidelity, by exploiting the tolerance of many of the typical algorithms consuming sensor data to deviations in values.

We derive analytic formulations for the family of VDBS encoders and introduce an efficient algorithm that performs close to the Pareto-optimal encoders. We evaluate the algorithm in two applications: Encoding data between a camera and processor in a text-recognition system, and between an accelerometer and processor in a pedometer system. For the text recognizer, the algorithm reduces signal transitions by 55 % on average, while maintaining OCR accuracy at over 90 % for previously-correctly-recognized text. For the pedometer, the algorithm reduces signal transitions by an average of 54 % in exchange for step count errors of under 5 %.

1. Introduction

Computation is not the dominant source of *instantaneous power dissipation* in many wearable and mobile systems. These systems are often organized around sensors, whose power dissipation when active is often larger than that of many of the embedded processors (microcontrollers) with which they are typically paired. The sensors are typically sampled whenever computation is active and as a result the fraction of overall *energy usage over time* attributable to computation, relative to sensors, is also often small.

Figure 1 shows the average power dissipation when active, for a collection of components. The components include an implementation of the lowest-power variant of the ARM architecture currently available (Cortex-M0+[1] [6]), several state-of-the-art sensors [1, 15, 16, 20, 21] and a Bluetooth Low-Energy (Bluetooth LE) radio[2] [19]. In addition to computation and sensors, many wearable systems have organic light-emitting diode (OLED) displays, whose power dissipation is proportional to the number of pixels which are lit and to their color (there is no backlight). We have therefore included, for reference, the power dissipation of a 20×20 pixel subset of an OLED display, based on measurements we performed on one such display panel [10].

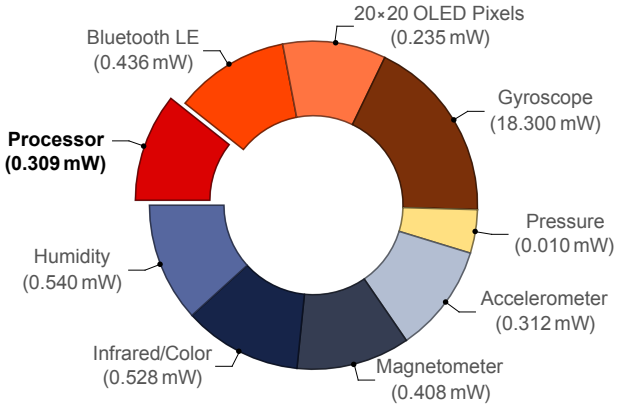


Figure 1. Sector plot of the power dissipation for several state-of-the-art system components typical of wearable and sensor-driven systems. The sectors are shown scaled logarithmically to simplify visualization of the large range of values: Clearly, not all systems will contain a gyroscope (which dominates the power breakdown in this collection of system components).

From Figure 1, it is clear that the processor dissipates less power when active than almost all the other components. Since most wearable systems sample their sensors periodically, the energy usage over time is also likely to still be dominated by components other than the processor.

[1] Running a `while(1)` loop from its on-chip SRAM at 2 MHz and 3.0 V.

[2] In *advertising/discoverable* mode.

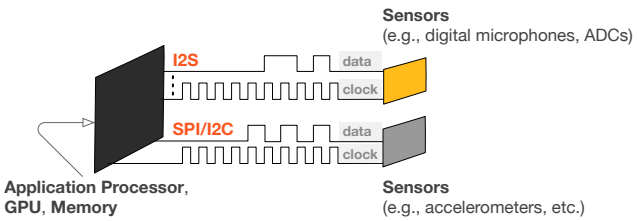


Figure 2. Single-ended serial interfaces such as SPI, I2C, and I2S are the dominant method for interconnecting processors with sensors in low-power wearable platforms.

1.1 Data transfer is a growing fraction of system power

Processors in wearable, embedded, and mobile platforms are often connected to many sensor integrated circuits (ICs). Despite packaging advances such as 3D-stacked dies and package-on-package bonding, the power dissipated in inter-IC data transfer has not scaled with semiconductor process technology advances as it has for the individual system components.

The cost for data movement is estimated to range from 18 fJ/bit-mm on-die, to 2 pJ/bit for printed circuit board (PCB) traces [11]. At data rates of just 10 Mb/s, and aggregated over several sensor interfaces, data transfer power becomes a significant fraction of system power. Since package and circuit board capacitances do not improve with semiconductor process advances, the relative proportion of system power attributed to data transfer will only grow relative to components such as processors as semiconductor technologies improve.

Figure 2 illustrates how, in a typical sensor-driven system, the processor interfaces with the system's sensors through bit-serial interfaces, rather than parallel buses. Serial interfaces enable packages with low pin counts and low PCB area. Given the speeds at which they operate, they typically do not employ any of the channel modulation techniques used in high-speed serial links.

By taking a holistic view of how the transferred data will be used by the system, we can formulate encodings that significantly reduce signal transitions on serial links. Since dynamic power dissipation is directly proportional to the number of signal transitions, such encodings, if efficient, reduce overall power dissipation.

1.2 VDBS encoders and Rake

Most of the data transferred on the serial interconnects of energy-constrained wearable platforms are from sensors. But, since the data are often generated by a process with some innate noise, the algorithms that consume them, even when requiring high-resolution data, are usually robust to small or occasional aberrations.

We exploit this observation to design a family of *value-deviation-bounded serial encoders* (VDBS encoders), for reducing signal transitions on serial interfaces. VDBS encoders permit a selectable amount of deviation of the transmitted data from their original values. Applied at only one end of a serial communication channel, they need no decoding and are not reversible.

For small bit widths (e.g., 8-bit values) and single values of tolerable deviation, VDBS encoders can be implemented using lookup tables. This is however not practical for word sizes of 10 to 24 bits which are common for the outputs of many sensors and ADCs; it is even more impractical when a system must support several different levels of encoding aggressiveness (amounts of tolerable deviation). To address this challenge, we present a practical algorithm for VDBS encoding, Rake, that is linear in the width of words to be encoded.

1.3 Contributions and outline

We introduce a new class of techniques for reducing the power dissipation in energy-constrained systems such as wearable and head-mounted systems. The techniques exploit the tolerance of many algorithms deployed in these contexts to deviations in their input values. We make the following contributions:

- **Formal definition of properties** of the family of possible value-deviation-bounded serial (VDBS) encoders (Section 2).
- **A practical algorithm, Rake**, for VDBS encoding (Section 3). For a maximum value deviation of 10 % in 8-bit values (i.e., a deviation of absolute value 25 on values that range from 0 to 255), Rake reduces signal transitions by 67 % on average. For maximum value deviations of 0.12 % of the full-scale range for 16-bit values, Rake reduces signal transitions by 41 %.
- **Numerical evaluation** of properties of optimal VDBS encoders and of Rake (Section 4). We show, empirically, that Rake reduces signal transitions almost as much as the optimal transition-reducing VDBS encoder, and induces almost as little deviation in values as the optimal deviation-minimizing VDBS encoder. We also show that VDBS encoders reduce transitions more than simply representing values with shorter words of equivalent effective number of bits.
- **End-to-end evaluation** of Rake deployed within two applications: We evaluate Rake in encoding data between a camera and processor in a text-recognition application, and between an accelerometer and processor in a pedometer application (Section 5). For the text recognizer, Rake reduces signal transitions by an average of 55 % while maintaining an OCR accuracy of over 90 % for previously-correctly-recognized text. For the pedometer system, Rake reduces signal transitions by 54 % on average, while leading to errors of less than 5 % in the number of reported steps.

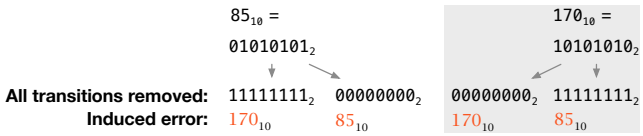
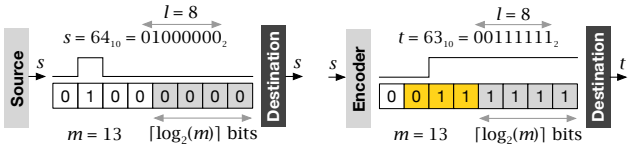


Figure 3. The maximum serial transition counts occur when words have alternating 0s and 1s in their binary representations.



(a) Without VDBS encoding. (b) With VDBS: fewer transitions.

Figure 4. In this example, the tolerable deviation, m, is 13 (i.e., 5 % of 255). VDBS encoding halves the number of transitions while incurring a value deviation, $|s - t|$, of just 0.39 % of the full-scale range. All bits except the most-significant bit are modified, not just the lower $\lceil \log_2(m) \rceil$.

- **An overview of analytic properties** of VDBS encoders. These provide insight into how VDBS encoding could be used to explore further directions beyond this work (Appendix A).
- **Formulation of bounds** on best-case effectiveness of VDBS encoders (Appendix B).
- **Detailed numerical studies** of the behavior of VDBS encoders across various word sizes and input distributions (Appendix C).

2. Value-Deviation-Bounded Serial Encoding

We consider *serial* communication interfaces, where the bits of a word are transmitted one at a time. For reasons of design size and cost, such communication interfaces are the norm in space-constrained embedded systems. For reasons related to crosstalk, they are also the norm in most state-of-the-art high-performance communication interfaces.

Because the dynamic power dissipation of electrical communication interfaces is proportional to the number of signal transitions (and to the square of the interface voltage), our goal is to reduce these transitions. For serial interfaces, these transitions occur between subsequent transmitted bits of the same word. In what follows, we will therefore refer to the number of such transitions as the *serial transition count* (STC). The maximum serial transition counts for l-bit values occur when they have alternating 0s and 1s in their binary representations (Figure 3). Figure 4 illustrates how, by permitting the transmitted values to differ from their original values, value-deviation-bounded serial encodings reduce the number of signal transitions when a value is transmitted over a serial interface.

There are three essential ingredients in the formulation of VDBS encoders[3]: ❶ The number of serial transitions when a value s is transmitted over a serial link (two transitions in Figure 4(a)), ❷ the difference in serial transition counts between two words (a difference of one between s and t in Figure 4), and ❸ the possible constraints under which serial transitions in a word may be minimized.

Definition 1 (*Serial transition count function, $\#_\delta(s)$*)**.** Let s be an l-bit unsigned integer with bits $s_0, s_1, \ldots, s_{l-1}$, from least- to most-significant bit. Then, we define $\#_\delta(s)$, the number of signal transitions in the serialization of s, as

$$\#_\delta(s) = \sum_{i=0}^{l-2} s_i \oplus s_{i+1}. \qquad \blacksquare$$

Definition 2 (*Serial transition count difference, $\Delta_{s,t}$*)**.** Let s and t be two l-bit words. Then, we define $\Delta_{s,t}$, as the absolute value of their difference in serial transition counts:

$$\Delta_{s,t} = |\#_\delta(s) - \#_\delta(t)|. \qquad \blacksquare$$

With the definitions of $\#_\delta(s)$ and $\Delta_{s,t}$, we now define the family of possible VDBS encoder functions. These functions solve the following problem: For an unsigned integer s, find a *proximal value*, t, such that $\#_\delta(t) \leq \#_\delta(s)$ and $|s-t| \leq m$, for some *tolerable deviation*, m.

Definition 3 (*Family of optimal VDBS encoders*)**.** Let s and t be two l-bit integers, and let m be an integer representing the difference in numeric value between s and t. We define a Boolean predicate $\mathrm{P}_{s,t,m}$ such that

$$\mathrm{P}_{s,t,m} = (|s - t| \leq m) \wedge ((\#_\delta(s) - \#_\delta(t)) \geq 0)\,.$$

There are four possible functions that reduce or maintain the serial transition count while ensuring that their output is within m of their input:

$$\mathrm{e}_1(s,m) = \left(\tau \text{ s.t. } \mathrm{P}_{s,\tau,m} \wedge \left(|s-\tau| = \min_{0<i<2^l-1} |s-i|\right)\right),$$
$$\mathrm{e}_2(s,m) = \left(\tau \text{ s.t. } \mathrm{P}_{s,\tau,m} \wedge \left(|s-\tau| = \max_{0<i<2^l-1} |s-i|\right)\right),$$
$$\mathrm{e}_3(s,m) = \left(\tau \text{ s.t. } \mathrm{P}_{s,\tau,m} \wedge \left(\Delta_{s,\tau} = \min_{0<i<2^l-1} \Delta_{s,i}\right)\right),$$
$$\mathrm{e}_4(s,m) = \left(\tau \text{ s.t. } \mathrm{P}_{s,\tau,m} \wedge \left(\Delta_{s,\tau} = \max_{0<i<2^l-1} \Delta_{s,i}\right)\right). \qquad \blacksquare$$

2.1 Properties of the optimal VDBS encoders

The properties of the four functions are important because they bound the behavior of all possible VDBS encoders:

[3] In what follows, we restrict our treatment to unsigned integers for clarity of exposition; many real-world use cases are of this type. The analysis easily extends to two's-complement, fixed-, and floating-point representations.

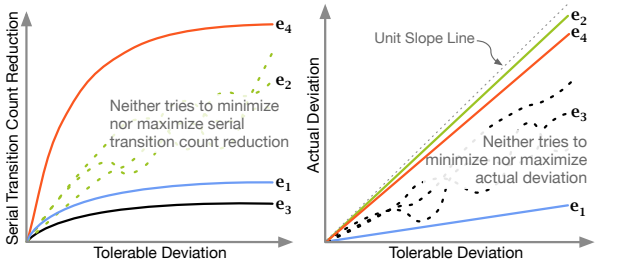


Figure 5. Illustration of relationships between the four classes of encoders, e_1, e_2, e_3, and e_4, across values of m.

① $\mathrm{e_1}(\mathbf{s}, \mathbf{m})$: smallest value difference within m of s that has same or smaller transition count.

② $\mathrm{e_2}(\mathbf{s}, \mathbf{m})$: largest value difference within m of s that has same or smaller transition count.

③ $\mathrm{e_3}(\mathbf{s}, \mathbf{m})$: smallest reduction in transition count from s, among all values within m of s.

④ $\mathrm{e_4}(\mathbf{s}, \mathbf{m})$: largest reduction in transition count from s, among all values within m of s.

Our objective is to obtain a method for VDBS encoding whose behavior encompasses the best of the properties of all the ideal encoders: Value deviations close to those of $\mathrm{e}_1(s, m)$, and transition reduction close to that of $\mathrm{e}_4(s, m)$.

Figure 5 pictorially illustrates the relationships between e_1, e_2, e_3, and e_4 across values of the tolerable deviation m. The function $\mathrm{e}_2(s, m)$ *maximizes the deviation from* s of the value it returns, with no constraint on maximally reducing transitions. Similarly, $\mathrm{e}_3(s, m)$ *minimizes the reduction in transition count*, but does not attempt to minimize the deviation from s.

The subset of three encoder types e_1, e_3, and e_4 are Pareto-optimal when considering both reduction in serial transitions and actual deviations for a given tolerable deviation. Because it is strictly dominated by e_4, the function e_2 is not in the Pareto set. As we shall see in Section 4, the behavior of the basic VDBS encoder that removes transitions from only the lower $\lceil \log_2(m) \rceil$ bits is similar to $\mathrm{e}_2(s, m)$ for many practical word sizes and tolerable deviations.

2.2 Practical implications of tolerable deviation, m

Because different applications will tolerate differing amounts of deviation, it is useful to have the tolerable deviation m as a parameter of the encoding. For example, sensor values used to guide a control system may tolerate only small m, whereas values representing pixels of a captured image in a text-recognition application can tolerate modest deviation. As we will show in Section 4, even small values of tolerable deviation can lead to significant transition reductions.

The need to dynamically control m makes the use of lookup tables (LUTs) impractical because a set of LUTs would be needed for each m. It is however possible to construct an efficient encoder that approaches the Pareto-optima in transition reduction and deviation minimization.

3. Rake: Practical VDBS Encoding

Given a value s and tolerable deviation m, the family of encoders described in the preceding section identify the possible optimum ways in which serial transitions can be reduced. Exact algorithms for these optimal encoders must select a value out of a set whose size is exponential in the length of words; they are therefore impractical in practice. On the other hand, the basic heuristic of simply removing transitions from the lower-order $\lceil \log_2(m) \rceil$ bits misses opportunities to reduce transitions at minimal cost. In the example of Figure 4, it cannot halve transitions at minimal value deviation of 1, like optimal VDBS encoders can.

3.1 The Rake algorithm

To address these shortcomings, we present an efficient algorithm that is linear in word size. The algorithm reduces transitions more than the basic approach and almost as much as the optimum VDBS encoder, e_4, that prioritizes transition reduction. At the same time, on average, it incurs deviations smaller than all optimum encoders except e_1 (the optimum VDBS encoder that prioritizes minimizing deviation). We call the algorithm *Rake*, because it operates in two sweeps of a word, accumulating values in the first sweep and leveling out transitions in the second. The algorithm (Figure 6) operates as follows.

First, moving across the l-bit input word s from least-significant bit (LSB) to most-significant bit (MSB), the *transition count register*, nt, stores the number of transitions seen to-date. The indices of these transitions are stored in the *transition indices array*, tr. For each transition, the length of the run of 0s or 1s leading to the transition is stored in the *run length temporary register*, rl. Each such run of 0s or 1s could be bit-wise negated to either increase or decrease the value of s; the change in value that such a negation would contribute is stored in the *cumulative run contribution arrays*, $cr0c$, for runs of 0s, and $cr1c$ for runs of ones.

Second, the algorithm moves across the input in the opposite direction, from MSB to LSB, inspecting only the nt bit positions that have transitions; these locations were previously stored in tr. For each of the nt transition locations in tr, it checks whether the deviation incurred by negating the bits that constitute a transition could be offset by the runs of lower-order bits of opposite polarity, as captured by the contents of $cr0c$ and $cr1c$. It removes the first transition that passes this check, and completes. The algorithm takes l steps as it traverses from the LSB to the MSB, followed by at most $nt - 2$ steps in the opposite direction. The maximum value of nt is $l - 1$, thus Rake takes a maximum of $2l - 3$ steps.

3.2 Efficient implementation and use of Rake

The linear-time behavior of the Rake algorithm means that, for example, for 24-bit values it requires only $2l - 3$ steps

```
RakeVdbsEncoder(s, m)
▷ First phase, from LSB to MSB (l steps), the transition
▷ count register, nt, stores the number of transitions seen.
for nt ← 0; i ← 0, i < l, i ← i + 1
▷ If two adjacent bits differ, store transition location in tr[].
    do if ((i < l − 1) ∧ (s_i ≠ s_{i+1}))
        then tr[nt] ← i
            if ((nt > 0))
            ▷ Determine the length of the run of 0s or
            ▷ 1s that the transition demarcates, storing
            ▷ the length in rl, and contribution in rc.
              then rl ← tr[nt] − tr[nt − 1]
                   rc ← (2^rl − 1) << tr[nt − 1]
            if ((nt > 0) ∧ (s_i = 0))
            ▷ For run of 0s, store contribution in cr0c.
              then cr0c[i] ← rc
            if ((nt > 0) ∧ (s_i = 1))
            ▷ For run of 1s, store contribution in cr1c.
              then cr1c[i] ← rc
            nt ← nt + 1
      elseif (i > 0)
      ▷ Pad the cumulative count arrays when there is
      ▷ no transition.
        then cr0c[i] ← cr0c[i − 1]
             cr1c[i] ← cr1c[i − 1]
▷ Second phase, from MSB to LSB, taking nt (less than l − 1)
▷ steps, inspect only the nt bit positions that have transitions.
while nt > 0
▷ rl is the run length and rc is its contribution if all bits in the
▷ run were flipped to remove the corresponding transition:
    do rl ← tr[nt] − tr[nt − 1]
      rc ← (2^rl − 1) << tr[nt − 1]
      ▷ Check whether deviation incurred by negating the
      ▷ bits that constitute a transition could be offset by
      ▷ the runs of lower-order bits of opposite polarity,
      ▷ as captured by the contents of cr0c and cr1c:
      if ((s_{tr[nt−1]} = 0) ∧ ((rc − cr1c[nt − 1]) ≤ m))
        then return (s + rc − cr1c[nt − 1])
      if ((s_{tr[nt−1]} = 1) ∧ ((rc − cr0c[nt − 1]) ≤ m))
        then return (s − rc + cr0c[nt − 1])
      nt ← nt − 1
return s
```

Figure 6. Rake algorithm for VDBS encoding. Most of the operations in the software pseudocode above can be implemented efficiently (and reused across the two phases) in a hardware implementation, without a need for the branching that the illustrative pseudocode implies.

(i.e., 45 steps), compared to having to explore a space of 16 million values for the exact optimal solution.

In practice, the Rake algorithm will be invoked once whenever a system is configured for a new tolerable deviation. The Rake value mappings may then either all be precomputed (at very low cost) and stored in a lookup table, computed as needed and cached, or obtained by a hardware implementation of Rake.

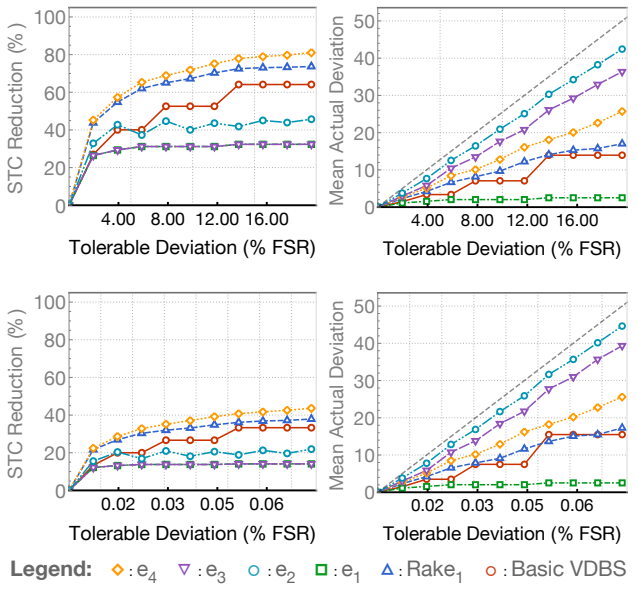


Figure 7. Mean serial transition count (STC) reduction and actual deviation, across all basis values, versus tolerable deviation expressed as a fraction of the full-scale range (FSR), for 8-bit values (top row), and 16-bit values (bottom row).

We are currently exploring efficient hardware implementations of Rake. The insight that enables efficient hardware implementations is that most of the operations in the Rake algorithm (Figure 6) can be implemented with comparators and multiplexers (for the **if()** statements), without the branching that the pseudocode implies. Furthermore, these circuits may be reused between the two phases of Rake. Given the fact that the power dissipation for many sensors far exceeds the power dissipation of a state-of-the-art ARM Cortex-M0+ microcontroller implementation (Figure 1), we expect the power overheads for a hardware Rake implementation to be yet lower than the already low overhead for a configuration-time software implementation of Rake.

4. Numerical Evaluation

We are interested in two objective measures of the efficacy of VDBS encoders: ❶ the *serial transition count reduction*, across all possible values for a given word length l and maximum tolerable deviation, m, and ❷ the *average actual deviation* that is incurred by encoders. The behavior of both of these properties is bounded by the family of encoders previously illustrated in Figure 5.

We evaluate both the ideal encoders of Section 2 as well as our Rake encoder of Section 3 under these two measures, by applying them to unsigned words with sizes of 8 and 16 bits. We pick these sizes because, as we shall see in Section 5, they are representative of the range of word widths for sensor and ADC values used in real-world systems.

4.1 Behavior across all possible basis values

Rake reduces transitions almost as much as the optimal encoder that prioritizes transition reduction, while inducing deviations smaller than all the optimal encoders except the one that prioritizes minimizing deviation. Even at moderate tolerable deviations of 5 %, the savings are almost twice those reported in the context of low-power deviation-free parallel and serial buses [2, 5].

The top row of Figure 7 shows the results of exhaustive numeric evaluation for all possible 8-bit values. For a maximum deviation of 10 % (i.e., a deviation of absolute value 25 on values that range from 0 to 255), Rake reduces signal transitions by 67 %. For this *maximum tolerable deviation*, the *mean actual deviation* is 10 (i.e., 4 %). Rake's reduction of serial transitions is greater than two of the three Pareto-optimal encoders and it is only 5 percentage points smaller than that of the optimal encoder that targets maximizing serial transition reduction (e_4). The observed mean deviation is also better than all but one of the Pareto-optimal encoders: It is less than 4 percentage points worse than the optimal encoder (e_1) that prioritizes minimizing deviation.

The results across all possible 16-bit values follow a similar trend (bottom row of Figure 7). For a maximum tolerable deviation of 0.12 % (i.e., a deviation of absolute value 79 on values that range from 0 to 65535), Rake reduces signal transitions by 41 %, with an accompanying mean deviation of 30 (0.05 % of the full-scale range).

4.2 Effective number of bits of encoded values

The effective number of bits (ENOB) captures the number of unique levels representable by encoded values, and is computed as $\log_2(|\{\text{unique encoder output values}\}|)$. Representing values with fewer bits reduces the number of signal transitions within transmitted words and in the clock signal. However, for the same ENOB (controlled by the tolerable deviation, m), VDBS encoders are more effective than simply employing shorter word sizes.

Figure 8 shows how, for the same ENOB, Rake applied to baseline 8-bit values reduces transitions 60 % more than simply employing shorter words. Furthermore, when considering the amount of control that can be exerted over transition reduction, Rake provides 7.4 times finer-grained control, as it enables fractional steps in the ENOB (controlled through m).

5. Application-Based Evaluation

In practice, the amount by which VDBS encoders can reduce transitions (and hence power dissipation), will depend on the distribution of values encountered. Likewise, the usefulness of VDBS encoding will depend on whether large reductions in signal transitions can be obtained for small deviations in encoded values, and whether these deviations do not adversely affect the application within which the encoding is deployed. To study the effect of VDBS encoding in

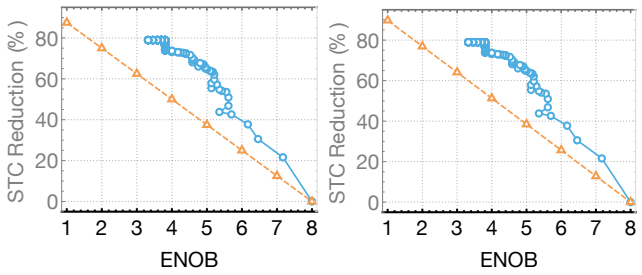


Figure 8. VDBS encoding provides finer-grained control of ENOB than the use of shorter word sizes (7.4 times more distinct values of effective ENOB in plots). The reduction in transitions achieved by VDBS encoding does not require changes to the datapath of applications consuming the modified data (e.g., changing algorithms to use 5-bit data instead of 8-bit data), as word sizes remain the same. Furthermore, the Rake VDBS encoder (○) reduces transitions by up to 24 *percentage points* (i.e., a 60% improvement in the percentage of transitions removed) more than using shorter words of equivalent ENOB (△), both considering only intra-word transitions (left), as well as total intra-word and clock transitions (right).

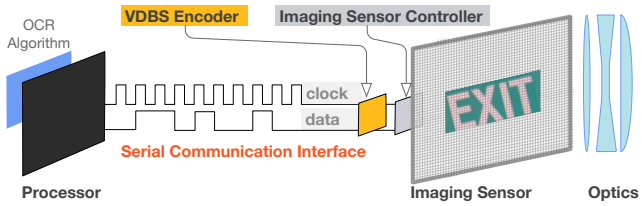


Figure 9. Illustration of an VDBS encoder in an optical character recognition application, as might be incorporated into an augmented-reality wearable system.

end-to-end application settings, we evaluate Rake in two application implementations.

5.1 Encoding camera data in a text-recognition system

We apply Rake to images in transfer between an image sensor and processor in a text-recognition system, such as that illustrated in Figure 9, and evaluate both the reduction in data transfer signal transitions as well as the effect on optical character recognition (OCR) errors.

We use version 3.02 of the Tesseract OCR system [14], widely regarded to be the most accurate open-source OCR package. For input, we use the test set from the ICDAR text image dataset [24], and select as our baseline the 392 images for which Tesseract returns the same recognition text as the benchmark's ground truth. We then apply Rake to each of these 392 text images, with degrees of tolerable deviation ranging from 0 % to 20 % of the full-scale range of the 8-bit per-color-channel pixel values. We quantify the errors in text recognition using the standard edit-distance-based

Tolerable Deviation	Image A	OCR Text	Transition Reduction	Image B	OCR Text	Transition Reduction
0%		"centre"	0%↓		"EXIT"	0%↓
2%		"centre"	43%↓		"EXIT"	39%↓
4%		"centre"	51%↓		"EXIT"	49%↓
6%		"centre"	59%↓		"EXIT"	55%↓
8%		"centrg"	63%↓		"EXIT"	58%↓
10%		"centre"	66%↓		"LTXIT"	61%↓
12%		"centre"	70%↓		""	70%↓
14%		"centre"	71%↓		""	72%↓
16%		"centre"	72%↓		""	72%↓
18%		"centre"	72%↓		""	73%↓
20%		"centre"	73%↓		""	73%↓

Figure 10. Effect on OCR accuracy of serial-transition-reducing encoding as a function of tolerable deviation (first column), for two different images (second and third columns). At higher tolerable deviation, there is greater transition reduction but this comes at the cost of OCR errors.

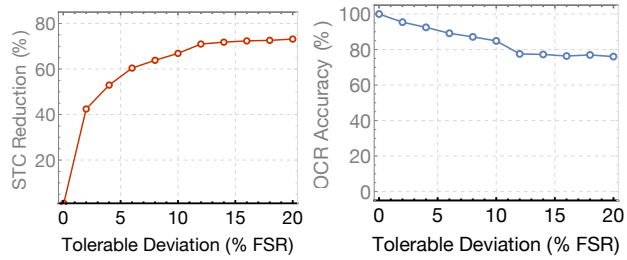


Figure 11. Averaged across the 392 text image inputs, Rake reduces transitions by 55 %, maintaining OCR accuracy of previously-correctly-recognized text at over 90 %.

metric used in the text-recognition literature [12]. Figure 10 presents an example of the effect of Rake on two input text images, as well as the effect on OCR accuracy and on transitions in the serialized image data.

Rake reduces transitions significantly with minimal effect on OCR error. As shown in Figure 11, with a target tolerable deviation of 5 %, Rake reduces serial transitions by over 55 %, while maintaining an OCR accuracy of over 90 % for previously-correctly-recognized text.

5.2 Encoding accelerometer data in a pedometer system

As a second evaluation of the effect of VDBS encoding in an end-to-end system, we apply Rake to accelerometer data in a pedometer (step counting) system (Figure 12).

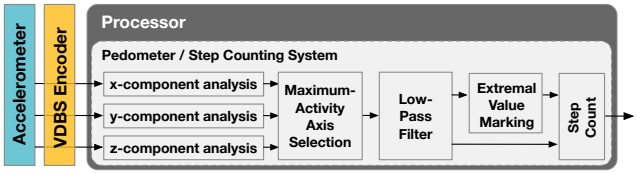


Figure 12. A VDBS encoder within a pedometer system.

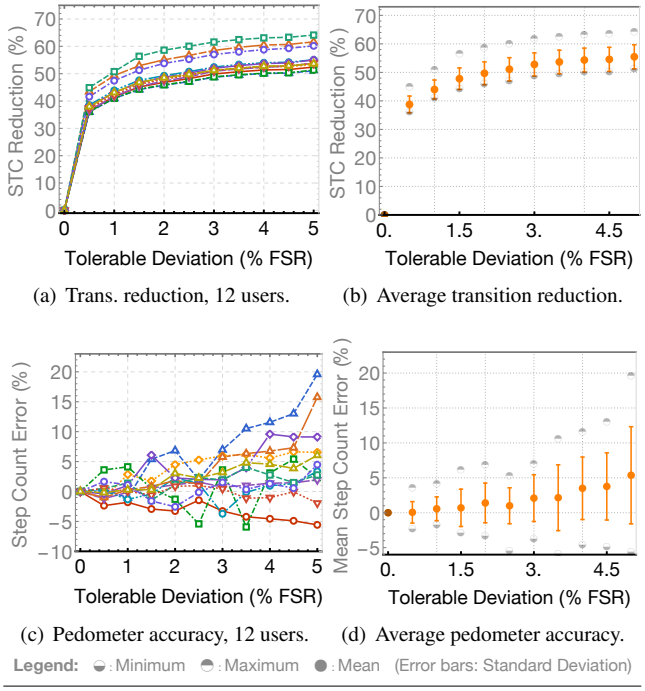


Figure 13. Rake reduces transitions by 54 % on average, while inducing step count errors of less than 5 % on average.

We use 3-axis accelerometer data sampled at 20 Hz, a total of 334377 samples or 4.6 hours worth of walking. The samples are taken from 12 different users in the publicly-available WISDM activity recognition dataset [9]. The WISDM dataset provides real-valued samples. In practice, however, actual accelerometer sensors provide a fixed number of bits of resolution, either directly or through the use of an ADC. We therefore convert the samples to 13-bit values to match the resolution of a state-of-the-art accelerometer [25]. We then apply Rake to the 13-bit data, with degrees of tolerable deviation ranging from 0 % to 5 % of the full-scale range of values, before passing the encoded data to a step counting algorithm [25]. As shown in Figure 13, at target tolerable deviations of 4 %, Rake reduces transitions by up to 63 % with a mean of 54 %, inducing step counting errors of less than 5 % on average.

6. Related Research

Power dissipation on communication buses has been a concern for many years, but the buses studied were parallel

buses, typically between processors and memories [2, 17]. On these buses, the data must be transferred correctly; substitution codes which exploit the ordering-distribution of values seen on the bus were thus used. Modern systems, whether high-performance, or energy-sensitive wearable platforms, predominantly use serial communication interfaces [8]. But, since serial interfaces transmit or receive only one bit at a time, they can't benefit from prior work on low-power encodings for parallel buses.

Encodings for serial video data [3, 13] exploit tonal locality in images to reduce transitions in exchange for representation overheads. Unlike VDBS encoders, they are specific to image data, and are not generally-applicable. General-purpose approaches to transition reduction include representing values with fewer bits [18], or using transition encoding [5]; as we show in this work, these are not as effective as our proposal. When the data transferred are from sensors however, many signal processing [7] and *recognition, mining, and synthesis* [4, 22] applications can tolerate errors in the data. VDBS encoders exploit this tolerance.

7. Conclusion

Wearable and health-tracking devices dissipate ever-larger fractions of their energy on sensor activation and data transfer. Since package and circuit board capacitances do not improve with semiconductor process advances, the fraction will continue to grow relative to components such as processors. For reasons of space and cost however, the data transfer happens over serial interfaces, not over parallel buses. This precludes encodings such as Gray codes.

VDBS encoders, introduced in this paper, reduce the dynamic power dissipation of serial buses when deviations in the values being transmitted are tolerable. We derived optimal VDBS encoders and presented an efficient VDBS encoder, Rake, that is close to optimal. We evaluated Rake through numerical studies as well as in two real-world end-to-end systems: OCR and a pedometer.

For the OCR system, Rake reduces signal transitions (and hence dynamic power for data transfer) by 55 % on average; it does so while maintaining OCR accuracy at over 90 % for previously-correctly-recognized text. For the pedometer system, Rake reduces signal transitions by 54 % on average, while leading to errors in the number of reported steps that are on average less than 5 %.

VDBS encoding is an exciting new direction for improving the energy-efficiency of communication and computing systems. There are many research avenues yet to be explored, ranging from hardware implementations of encoders to compute architectures that employ similar principles.

Appendix A Properties of Function $\#_\delta(n)$

Properties of the serial transition count (STC) function $\#_\delta(n)$, which we explore next, give insights into efficiency limits of VDBS encoders.

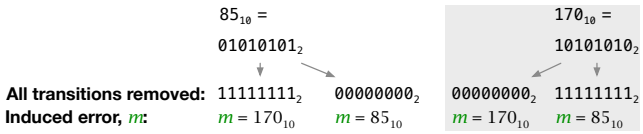


Figure 14. The maximum serial transition counts for l-bit values occur when they have alternating 0s and 1s in their binary representations.

Proposition 1 (Maximum serial transition count pattern)**.** When l is even, the maximum serial transition count occurs when the l-bit word has $\frac{l}{2}$ 0s and the same number of 1s. □

Proof (Maximum serial transition count pattern)**.** To maximize the serial transition count, there should be a transition in moving between every neighboring pair of bit positions. Thus, words with maximum serial transition count have $\frac{l}{2}$ 0s and the same number of 1s. ■

Corollary 1 (Maximum serial transition count basis values)**.** There are two values with maximum serial transition count. When l is even, these values are

$$\hat{b}_1 = \sum_{i=0}^{\frac{l}{2}-1} 2^{2i} = \tfrac{1}{3}(2^l - 1) \qquad (1)$$

and

$$\hat{b}_2 = 2^l - 1 - \sum_{i=0}^{\frac{l}{2}-1} 2^{2i} = \tfrac{2}{3}(2^l - 1).$$

□

This follows directly from the two cases in the proof of Proposition 1 (either 0 or 1 in least-significant bit). For example, Figure 14 illustrates how, for $l = 8$, the maximal-serial-transition-count words are 85 and 170.

Lemma 1 (Maximum serial transition count)**.** For every l-bit word n, $\#_\delta(n) \leq l - 1$. □

Proof (Maximum serial transition count)**.** The number of bits l in a word is a natural number. When l is 1, there are no transitions in the word, by definition of the serial transition count. For all other l, the maximum serial transition count occurs when all adjacent bits of a word differ. There are four cases in which this could happen, corresponding to whether l is even or odd, and whether the least-significant bit (LSB) contains a 1 or a 0.

First, consider the cases when l is even. When l is even, there are $\frac{l}{2}$ ones and $\frac{l}{2}$ zeros. If the LSB is 0, there will be one transition in moving from the LSB towards the most-significant bit (MSB), and each of the remaining $\frac{l}{2} - 1$ bits which are 0 will have two associated transitions. There will therefore be a total of $l - 1$ transitions. The same argument applies if the LSB were 1.

Next, consider the cases when l is odd. When l is odd, there are either $\lfloor \frac{l}{2} \rfloor$ bits which are 1 and $\lceil \frac{l}{2} \rceil$ bits which are 0, or vice versa, and the bit polarity appearing in the LSB

will occur $\frac{l-1}{2}+1$ times, and the opposite polarity to the LSB will occur $\frac{l-1}{2}$ times.

There will be one transition moving out of the LSB towards the MSB, followed by transitions in the remaining $l-1$ bits. Since l is odd, it follows that $l-1$ is even. But we showed above that such an even number of bits could contain at most $(l-1)-1$ transitions. Thus, when l is odd, the maximum number of transitions is also $1+(l-1)-1$. That is, the maximum number of transitions is $l-1$. ■

Theorem 1 (Serial transitions and Gray code)**.**
Let s be an l-bit integer, let $\mathrm{GrayCode}(s)$ denote the sth value in Gray code order for l-bit values, and let $\#_1(s)$ denote the count of 1s in s. Then, $\#_\delta(s) = \#_1(\mathrm{GrayCode}(s))$. □

We will use the following, the Gray code theorem [23], in the proof of Theorem 1. We include a self-contained adaptation of Wilf's original proof here so that our discussion stands on its own.

Theorem 2 (Wilf's Gray Code Theorem)**.**
Let s be an l-bit integer, and let g be the sth l-bit integer in Gray code order for l-bit values, where

$$s = \sum_{i=0}^{l-1} s_i 2^i \quad \text{and} \quad g = \sum_{i=0}^{l-1} g_i 2^i$$

Then,

$$g_i \equiv s_i + s_{i+1} \pmod 2 \quad (i = 0, \ldots, l-2). \qquad \square$$

For example, consider the 8-bit value 63. The string of rank 63 in the 8-bit Gray code, that is, the 63rd Gray code value, can be constructed as follows: for the ith bit, simply take the ith and $i+1$th bits of 63, and add them modulo 2.

Proof (Wilf's Gray Code Theorem)**.**
Let $\mathcal{L}_l$ be the list of l-bit strings in Gray code order. $\mathcal{L}_0$ is the empty list. $\mathcal{L}_l$ can be constructed recursively by:

- Let $\mathcal{L}_{l-1}^0$ be the list obtained by prefixing every element of $\mathcal{L}_{l-1}$ with an additional 0.
- Let $\overline{\mathcal{L}_{l-1}}^1$ be the list obtained by prefixing every element of the list $\mathcal{L}_{l-1}$, in reverse order, with an additional 1.
- $\mathcal{L}_l$ is the concatenation of $\mathcal{L}_{l-1}^0$ and $\overline{\mathcal{L}_{l-1}}^1$.

By construction therefore, the 2^l entries for an l-bit Gray code will be identical to the first 2^l entries for an $(l+1)$-bit Gray code; we use this property below.

We prove by induction on l that the property of Theorem 2 holds for all l-bit integers s. When $l = 0$, $\mathcal{L}_l$ is the empty list, and the property we seek to prove is vacuously true. Suppose the property of Theorem 2 holds for all strings on the list $\mathcal{L}_{l-1}$. By construction of $\mathcal{L}_l$, we know the property must also hold for the first 2^{l-1} items on $\mathcal{L}_l$. Suppose then, that $s \geq 2^{l-1}$. Let $s' = 2^l - 1 - s$. Then the property of Theorem 2 holds for the string that has Gray code rank s', since it is by its definition less than 2^{l-1}.

But, again by construction of the Gray code lists $\mathcal{L}_l$ from $\mathcal{L}_{l-1}$, the first $l-1$ bits of the strings with ranks s and s' are identical, while the most-significant bits, s_l and s'_l, of these corresponding strings, have the relation

$$s_l = 1 + s'_l.$$

At the same time, the binary representations of the integers s and s' have the relation

$$s_i \equiv 1 + s'_i \pmod 2 \quad (i = 0, \ldots, l-1),$$

and the property of Theorem 2 continues to hold for all strings on the list $\mathcal{L}_l$. ■

Proof (Serial transitions and Gray code)**.**
The proof is a direct result of Theorem 2. Let g be the Gray code representation for l-bit integer s. That is, g is the rank-s l-bit Gray code. The number of 1s in g, $\#_1(g)$, is

$$\begin{aligned}
\#_1(g) &= \sum_{i=0}^{l-1} g_i \\
&= \sum_{i=0}^{l-2} (s_i + s_{i+1} \pmod 2), \quad \text{from Theorem 2} \\
&= \sum_{i=0}^{l-2} (s_i \oplus s_{i+1}).
\end{aligned}$$

But this is exactly the $\#_\delta(s)$ from Definition 1. ■

Property 1 (Bound on serial transition count difference)**.**
For any two l-bit words s and t, the serial transition count difference, $\Delta_{s,t}$ is less than or equal to $l-1$. □

Proof (Bound on serial transition count difference)**.**
By construction, the serial transition count, $\#_\delta(s)$ for a non-negative integer s, is a natural number. Therefore, the largest serial transition count difference, will occur when either $\#_\delta(s)$ is zero and $\#_\delta(t)$ takes on the maximum value in the codomain of $\#_\delta(t)$, or vice versa. From Lemma 1, this maximum value is $l-1$. Thus the maximum serial transition count difference, $\Delta_{s,t}$ is $l-1$. ■

Property 2 (Minimum and maximum deviation at maximum serial transition count difference)**.**
Let s and t be two l-bit words with l even, and let these words have the maximal serial transition count difference, $\Delta_{s,t}$, of $l-1$. That is,

$$\min_{\Delta_{s,t}=l-1} \{|s-t|\} = \tfrac{1}{3}\left(2^{\Delta_{s,t}+1} - 1\right), \tag{2}$$

and

$$\max_{\Delta_{s,t}=l-1} \{|s-t|\} = \tfrac{2}{3}\left(2^{\Delta_{s,t}+1} - 1\right). \tag{3}$$

□

Proof (Minimum and maximum deviation at maximum serial transition count difference)**.**
Follows directly from Corollary 1. ■

For example, for $l = 8$, we have from Lemma 1 that the maximal serial transition count difference is $l - 1 = 7$. The minimum deviation between two words which have this maximum serial transition count difference, from Property 2, is 85. Therefore, to reduce the serial transition count of an 8-bit word by 7 transitions, one cannot do so with a replacement word that deviates from it by less than 85.

Appendix B Bounds on Serial Transition Count Reduction

The bounds of Property 2 are only specified for the case of maximal changes in serial transition count, not for any arbitrary reduction in serial transition count. General bounds across all possible values of serial transition count reduction are desirable, because they would enable us to answer questions such as:

- **By how much can serial transition counts differ** for a given value deviation? This will be captured by Definition 4 and Theorem 3 below.
- **By how much can values differ** for a given difference in serial transition count? This will be captured by Definition 5 below.

Definition 4 (Serial transition difference bound function)**.**
Given an l-bit integer m, let $\mathrm{f}(m)$ be a function yielding the amount by which the serial transition counts of two words s and t can differ if $|s - t| = m$. That is,

$$\mathrm{f}(m) = \max_{|s-t|=m} \{\Delta_{s,t}\} . \quad ■$$

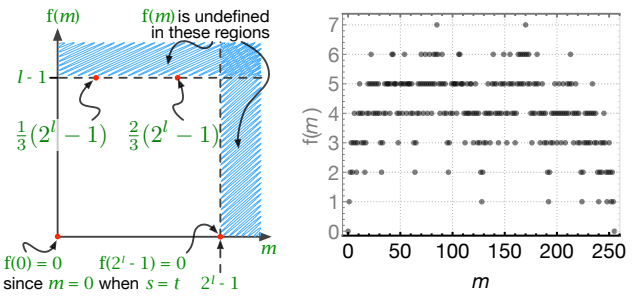
Why $\mathrm{f}(m)$ is important: The function $\mathrm{f}(m)$ is interesting because, if one had an exact expression or tight bounds for $\mathrm{f}(m)$, then an algorithm that searched for the serial-transition-reducing encoding for a value s could terminate as soon as it found a value t such that $\Delta_{s,t} = \mathrm{f}(m)$, since no better value than t is possible.

Theorem 3 (Bound on $\mathrm{f}(m)$)**.**
The function $\mathrm{f}(m)$ of Definition 4, for any l-bit value, m (with l even), is not monotone. The best linear monotone bound on $\mathrm{f}(m)$ is $\mathrm{f}(m) \leq l - 1$. □

Proof (Bound on $\mathrm{f}(m)$)**.**
Let s and t be two l-bit words, and let m be $|s - t|$, a value in the domain of f. If m is 0, then s is identical to t, and must have identical serial transition count, thus $\#_\delta(s) = \#_\delta(t)$ and therefore $\mathrm{f}(0) = 0$. If m is $2^l - 1$, then either s is $2^l - 1$ and t is zero, or vice versa. In both cases, their serial transition counts are 0 by definition, that is $\#_\delta(s) = \#_\delta(t) = 0$. Thus, when m is $2^l - 1$, $\mathrm{f}(m) = 0$.

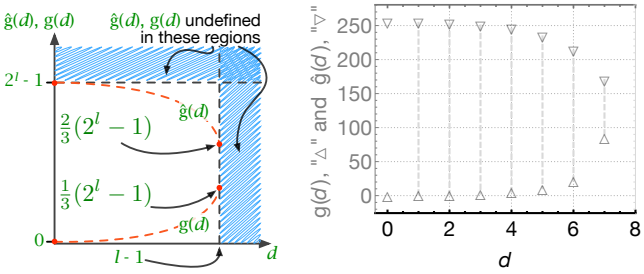
From Corollary 1 and Lemma 1, the maximum value of $\mathrm{f}(m)$ is $l - 1$, and it occurs at two values, $\hat{b}_1$ and $\hat{b}_2$ from Equation 1. Both $\hat{b}_1$ and $\hat{b}_2$ are greater than 0 and less than $2^l - 1$. Since $\mathrm{f}(0)$ is 0, $\mathrm{f}(\hat{b}_1)$ is $l - 1$, $\mathrm{f}(\hat{b}_2)$ is $l - 1$, and $\mathrm{f}(2^l - 1)$ is 0, it follows that $\mathrm{f}(m)$ is not monotone.

From Corollary 1 and Lemma 1, since there are two values of m for which $\mathrm{f}(m)$ takes on its maximum value of $l - 1$, it follows that the tightest linear bound on $\mathrm{f}(m)$ must pass through these points. Thus the tightest linear bound on $\mathrm{f}(m)$ is $l - 1$. ■



(a) Illustration of $\mathrm{f}(m)$. (b) Numerical evaluation of $\mathrm{f}(m)$.

Figure 15. The function $\mathrm{f}(m)$ yielding the amount by which the serial transition counts of two words s and t can differ if $s - t = m$, is not monotone.



(a) Illustration of $\mathrm{g}(d)$ and $\hat{\mathrm{g}}(d)$. (b) Numerical evaluation: $\mathrm{g}(d)$, $\hat{\mathrm{g}}(d)$.

Figure 16. At minimum serial transition count (STC) difference, $d = 0$, either s and t are identical, or they are different but take on values $s = 0$ and $t = 2^l - 1$ or vice versa.

Definition 5 (Value deviation bound functions)**.**
Let $\mathrm{g}(d)$ be the minimum amount by which two integers s and t can differ if their difference in serial transition count, $\Delta_{s,t}$, is d. Similarly, let $\hat{\mathrm{g}}(d)$ be the maximum amount by which two integers s and t can differ if their difference in serial transition count, $\Delta_{s,t}$, is d. That is

$$\mathrm{g}(d) = \min_{\Delta_{s,t}=d} \{|s - t|\} , \quad \text{and} \quad \hat{\mathrm{g}}(d) = \max_{\Delta_{s,t}=d} \{|s - t|\} . ■$$

Figure 15(a) illustrates several properties of $\mathrm{f}(m)$, and Figure 15(b) shows an empirical exact enumeration of $\mathrm{f}(m)$ across all possible 8-bit values.

Figure 16(a) illustrates several properties of $\mathrm{g}(d)$ and $\hat{\mathrm{g}}(d)$, and Figure 16(b) shows an empirical exact enumeration of $\mathrm{g}(d)$ and $\hat{\mathrm{g}}(d)$ for 8-bit values.

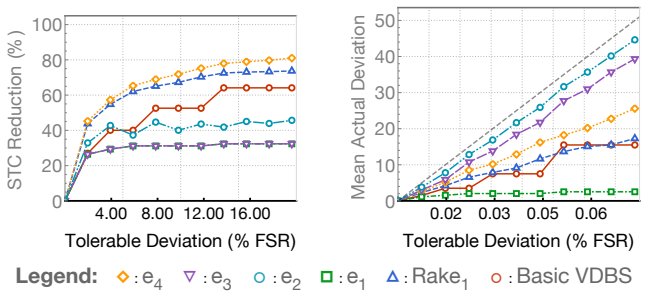


Figure 17. Mean reduction in serial transition count and increase in average observed value deviation versus tolerable deviation, for 8-bit words and tolerable deviation from 0–51.

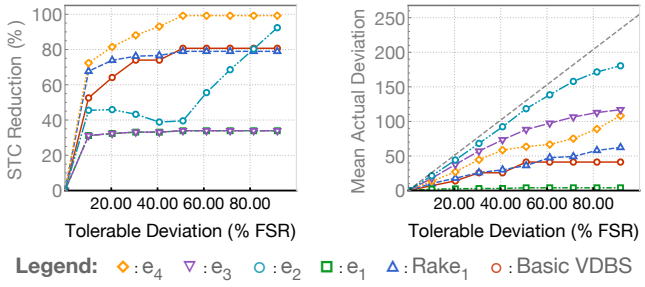


Figure 18. Mean reduction in serial transition count and increase in average observed value deviation versus tolerable deviation, for 8-bit words and tolerable deviation of 0–255.

Appendix C Details of Numerical Study

Section 4 presented a numerical evaluation of the behavior of VDBS encoders for 8- and 16-bit values. The behavior for other word sizes, presented next, follows a similar trend across the different Pareto-optimal encoders and the Rake encoder.

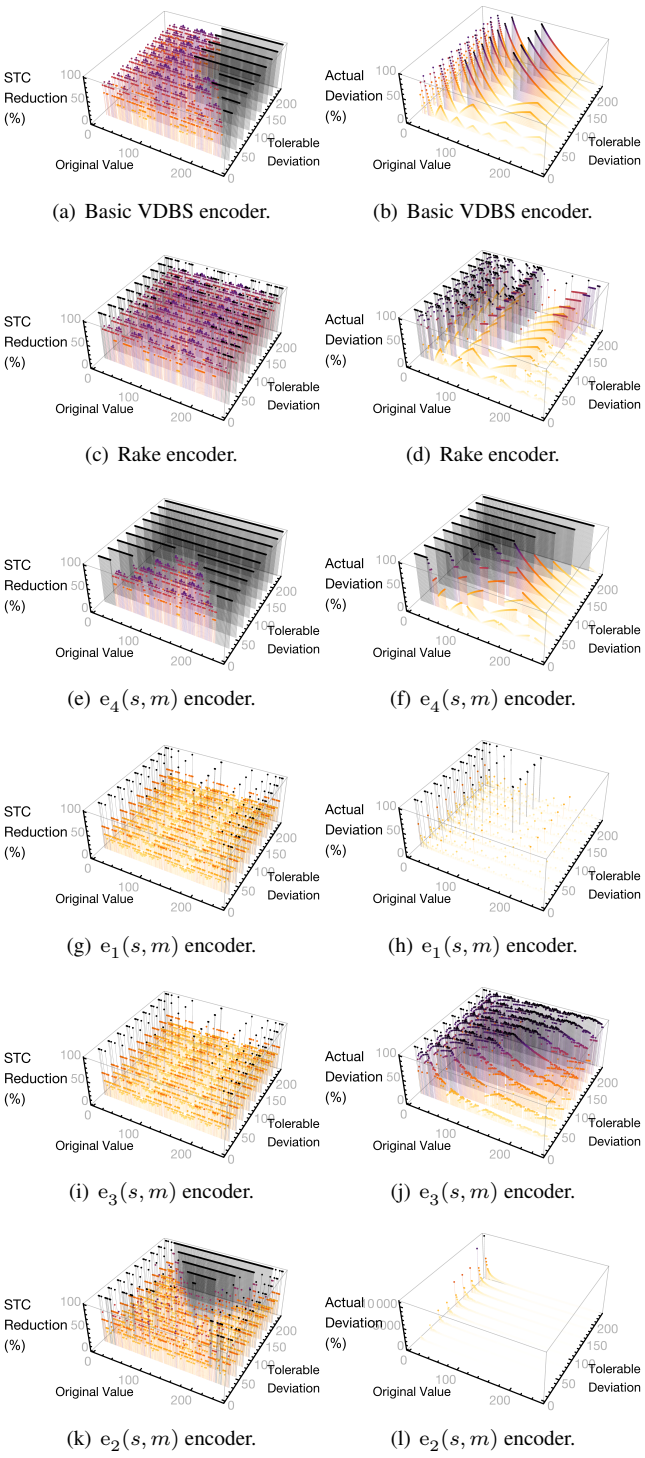
C.1 Behavior across all possible 8-bit values

Figure 17 and Figure 18 show the reduction in serial transition count and increase in average observed value deviation, as a function of the tolerable deviation for both the ideal and practical VDBS encoders.

In both figures, we observe that even at low values of tolerable deviation (e.g., tolerable deviations of less than 10% of the full-scale range of values), the reduction in serial transitions is significant, and is a reduction of over 60% for the Rake encoder. The Rake encoder also performs close to the optimal transition-reducing encoder, and far outperforms the basic VDBS encoder. Figure 19, Figure 20, and Figure 21 provide more detailed insight into the distribution of transition reduction and incurred deviation as functions of tolerable deviation.

C.2 Behavior across all possible 13-bit values

For 13-bit words, Figure 22, Figure 23, Figure 24, Figure 25, and Figure 26 show the reduction in serial transition count and increase in average observed value deviation, as a func-



(a) Basic VDBS encoder. (b) Basic VDBS encoder.

(c) Rake encoder. (d) Rake encoder.

(e) $e_4(s, m)$ encoder. (f) $e_4(s, m)$ encoder.

(g) $e_1(s, m)$ encoder. (h) $e_1(s, m)$ encoder.

(i) $e_3(s, m)$ encoder. (j) $e_3(s, m)$ encoder.

(k) $e_2(s, m)$ encoder. (l) $e_2(s, m)$ encoder.

Figure 19. Distribution of actual deviation and serial transition count (STC) reduction percentage, across all basis values, over the range of tolerable deviations, for 8-bit words and tolerable deviation from 0–255. The percentage actual deviation is a measure of *relative error*, as opposed to *absolute error*. The definition of the $e_2(s, m)$ encoder targets the largest actual value deviation smaller or equal to the tolerable deviation. For a given original value therefore, the percentage actual deviation may be greater than 100%.

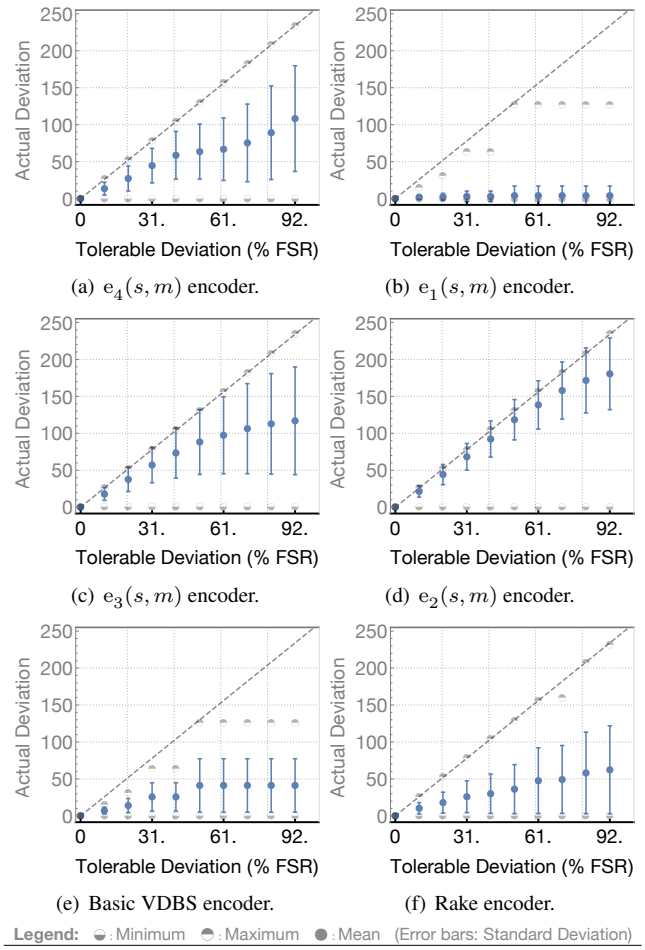


Figure 20. Actual value deviation versus tolerable deviation for 8-bit words and tolerable deviation from 0–255.

tion of the tolerable deviation for both the ideal and practical VDBS encoders.

The absolute value of tolerable deviation is the same as shown previously for 8-bit values, and is a maximum absolute value of tolerable deviation of 51 in Figure 22, and a maximum absolute value of tolerable deviation of 255 in Figure 23 to Figure 24. As a percentage of the full-scale range of 13-bit values however, these tolerable deviations correspond to 0.62 and 3.1% respectively.

For larger word sizes, one could in principle explore tolerating larger absolute values of deviations. Larger word sizes are however employed specifically to provide higher precision; we therefore only explore maximum absolute values of tolerable deviation in the ranges of 0–51 and 0–255.

C.3 Behavior across all possible 16-bit values

For 16-bit words, Figure 27, Figure 28, Figure 29, and Figure 30 show the reduction in serial transition count and increase in average observed value deviation, as a function of the tolerable deviation for both the ideal and practical VDBS encoders.

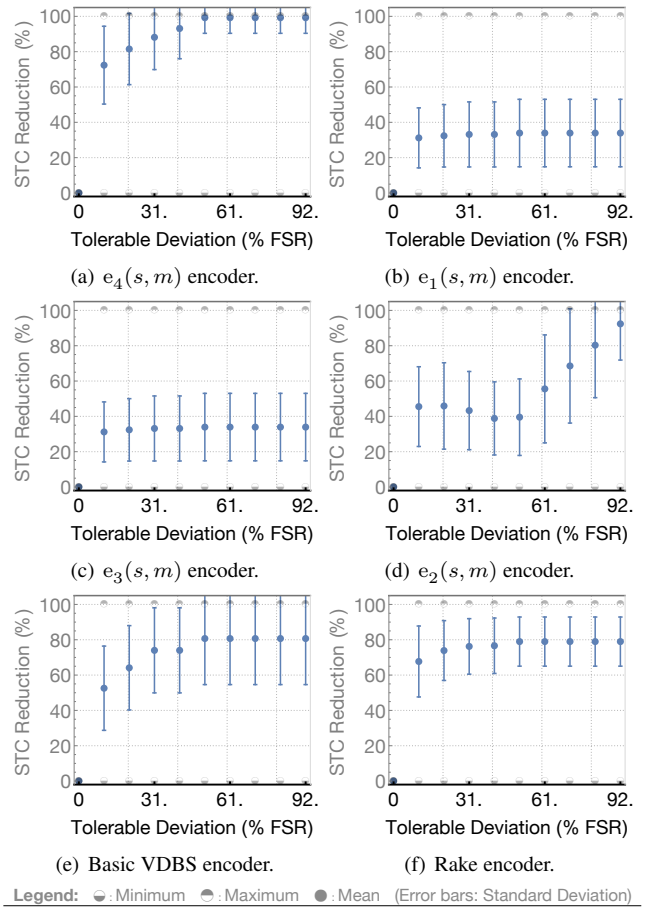


Figure 21. Serial transition count versus tolerable deviation for 8-bit words and tolerable deviation from 0–255.

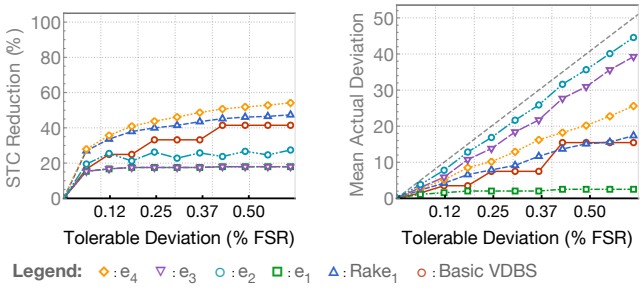


Figure 22. Mean reduction in serial transition count and increase in actual deviation versus tolerable deviation for 13-bit words and tolerable deviation from 0–51.

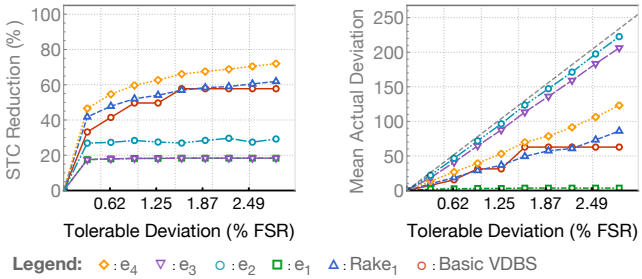
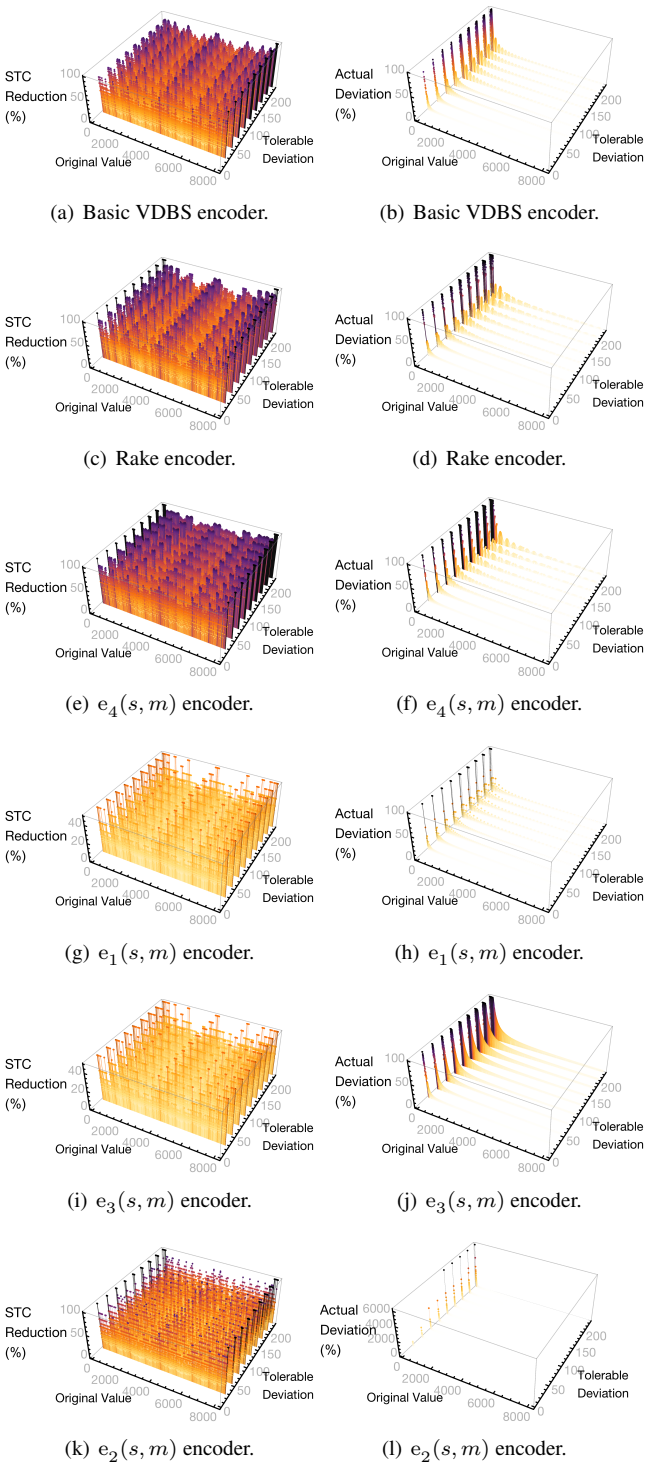


Figure 23. Mean reduction in serial transition count and increase in actual deviation versus tolerable deviation for 13-bit words and tolerable deviation from 0–255.



(a) Basic VDBS encoder. (b) Basic VDBS encoder.

(c) Rake encoder. (d) Rake encoder.

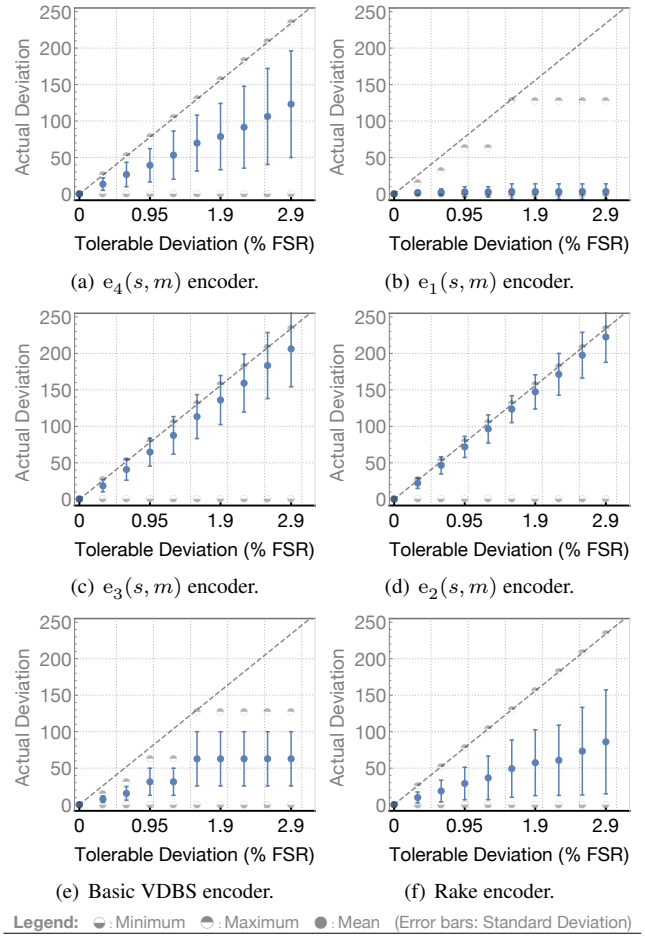
(e) $e_4(s, m)$ encoder. (f) $e_4(s, m)$ encoder.

(g) $e_1(s, m)$ encoder. (h) $e_1(s, m)$ encoder.

(i) $e_3(s, m)$ encoder. (j) $e_3(s, m)$ encoder.

(k) $e_2(s, m)$ encoder. (l) $e_2(s, m)$ encoder.

Figure 24. Distribution of actual deviation and serial transition count (STC) reduction percentage, across all basis values, over the range of tolerable deviations, for 13-bit words and tolerable deviation from 0–255. The percentage actual deviation is a measure of *relative error*, as opposed to *absolute error*. The definition of the $e_2(s, m)$ encoder targets the largest actual value deviation smaller or equal to the tolerable deviation. For a given original value therefore, the percentage actual deviation may be greater than 100%.



(a) $e_4(s, m)$ encoder. (b) $e_1(s, m)$ encoder.

(c) $e_3(s, m)$ encoder. (d) $e_2(s, m)$ encoder.

(e) Basic VDBS encoder. (f) Rake encoder.

Figure 25. Actual value deviation versus tolerable deviation for 13-bit words and tolerable deviation from 0–255.

C.4 Behavior across word sizes

For the same absolute value of tolerable deviation (say, 51), we observe from the preceding figures for 8-, 13-, and 16-bit values, that the reduction in serial transition count decreases with increasing word size, although it still remains substantial, at reductions of almost 40% for the practical Rake encoder given tolerable deviation of 51 (0.07% of full-scale range for 16-bit values).

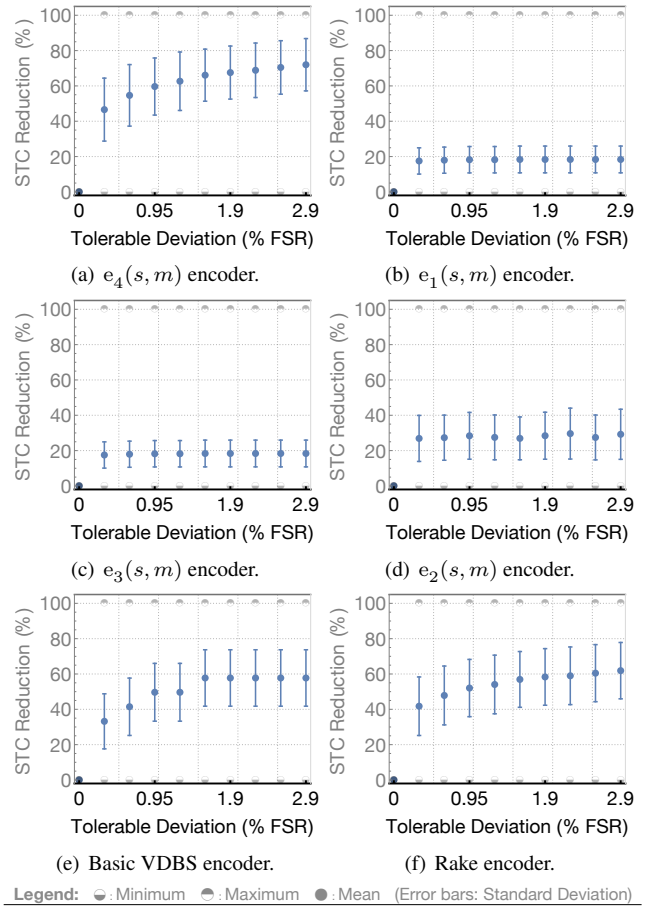


Figure 26. Serial transition count reduction versus tolerable deviation for 13-bit words, tolerable deviation from 0–255.

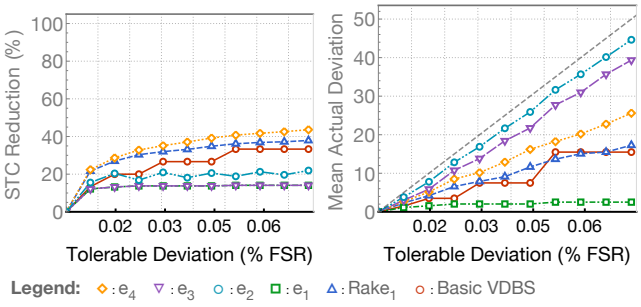


Figure 27. Mean serial transition count reduction and actual deviation, 16-bit words and tolerable deviation from 0–51.

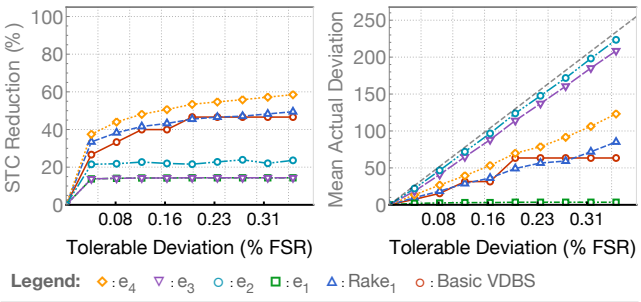


Figure 28. Mean serial transition count reduction and actual deviation, 16-bit words and tolerable deviation from 0–255.

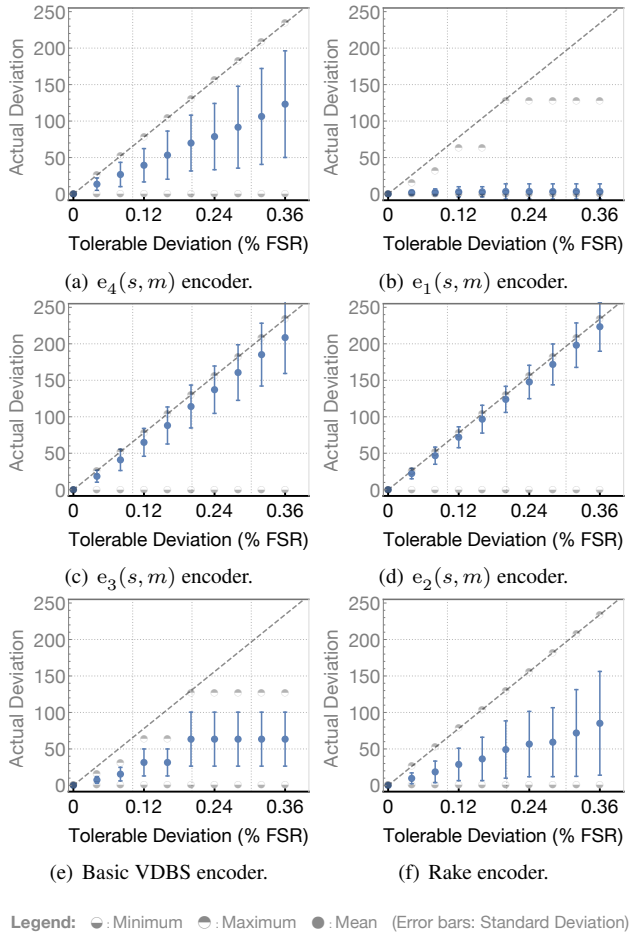
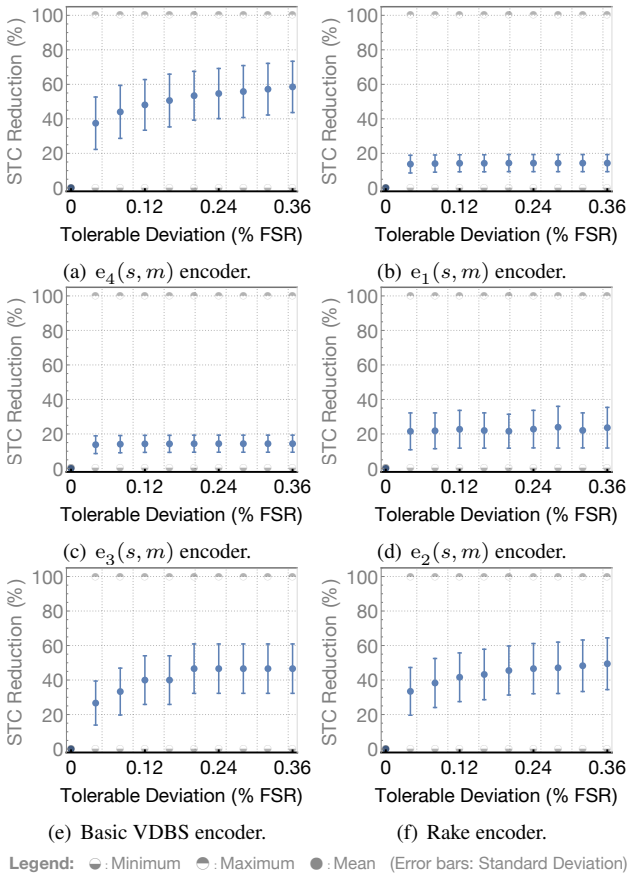


Figure 29. Actual deviation versus tolerable deviation for 16-bit words and tolerable deviation from 0–255.

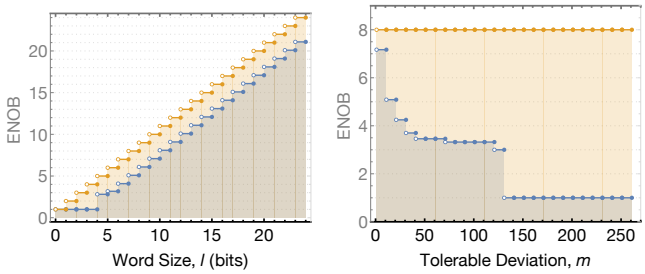


(a) $\mathrm{e}_4(s, m)$ encoder. (b) $\mathrm{e}_1(s, m)$ encoder.

(c) $\mathrm{e}_3(s, m)$ encoder. (d) $\mathrm{e}_2(s, m)$ encoder.

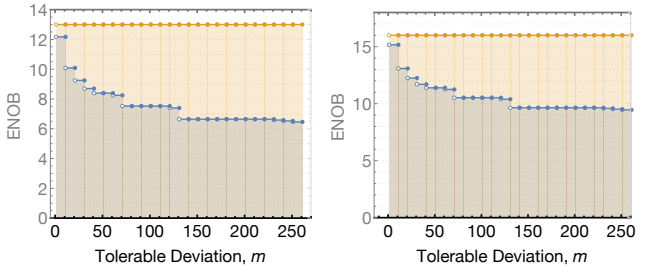
(e) Basic VDBS encoder. (f) Rake encoder.

Figure 30. Serial transition count reduction versus tolerable deviation for 16-bit words, tolerable deviation from 0–255.



(a) ENOB for word sizes up to 24 bits, for unencoded binary values (upper curve) and $\mathrm{e}_4(s,m)$-encoded values with $m = 10$.

(b) ENOB of 8-bit words, for unencoded binary values (upper curve) and $\mathrm{e}_4(s,m)$-encoded values with tolerable deviations of up to 255.



(c) ENOB of 13-bit words, for unencoded binary values (upper curve) and $\mathrm{e}_4(s,m)$-encoded values with tolerable deviations of up to 255.

(d) ENOB of 16-bit words, for unencoded binary values (upper curve) and $\mathrm{e}_4(s,m)$-encoded values with tolerable deviations of up to 255.

Figure 31. The optimum transition-reducing encoder ($\mathrm{e}_4(s,m)$) leads to a smaller effective number of bits (ENOB) for encoded words, than unencoded words of the same word size.

C.5 Effective number of bits for encoded values

Figure 31 shows the effect of word size and tolerable deviation on the effective number of bits (ENOB) for the optimal VDBS encoder that prioritizes transition reduction over deviation reduction ($\mathrm{e}_4(s,m)$). Particularly for larger word sizes, ENOB remains largely constant after an initial drop for the first few values of tolerable deviation.

Figure 32 shows the trend in mean number of serial transitions as a function of word size. Combined with the trend in transition reduction versus tolerable deviation seen previously for the ideal and practical VDBS encoders, there is more reduction in transition count of the output of VDBS encoders with increasing tolerable deviation (Figures 18, 23, 28), than there is reduction in transitions from reduced bit width alone for unencoded values.

Figure 33, Figure 34, and Figure 35 show the reduction in serial transition count as a function of the ENOB for unencoded words, as well as for values encoded by the ideal $\mathrm{e}_4(s,m)$ encoder that minimizes transitions. For unencoded words, the reduction in serial transition count is shown both when only considering the fact that shorter words lead to fewer transitions on any associated clock interface, as well as considering the effect of shorter words on within-word transitions (as previously seen in Figure 32).

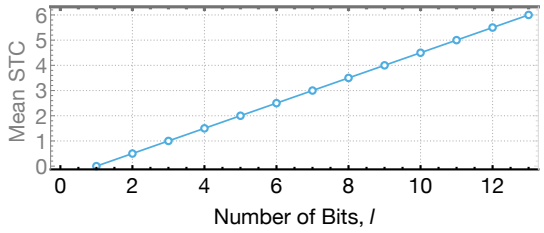
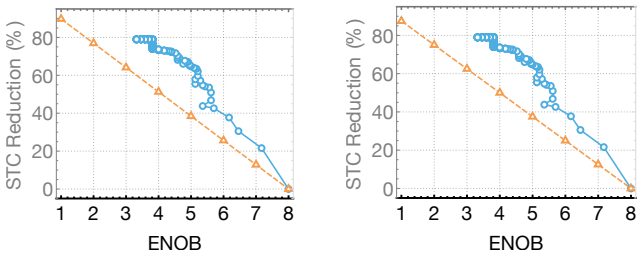


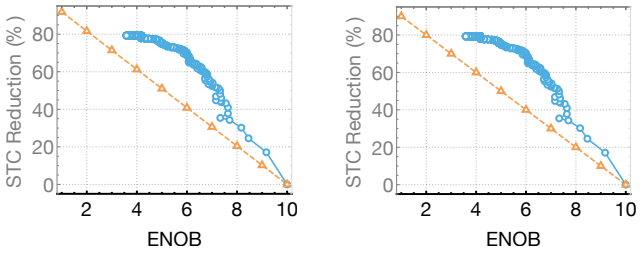
Figure 32. For l-bit words, the mean number of serial transitions is $(l-1)/2$.



(a) Transition reduction in both data and clock interfaces.

(b) Transition reduction in only clock interface.

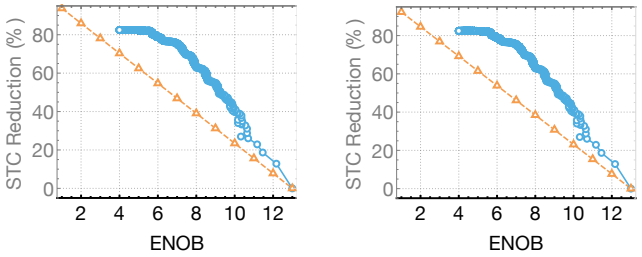
Figure 33. Reduction in serial transition count compared to unencoded 8-bit words, as a function of the ENOB for both unencoded shorter words (△) and values encoded by the ideal $\mathrm{e}_4(s,m)$ encoder that minimizes transitions (○).



(a) Transition reduction in both data and clock interfaces.

(b) Transition reduction in only clock interface.

Figure 34. Reduction in serial transition count compared to unencoded 10-bit words, as a function of the ENOB for both unencoded shorter words (△) and values encoded by the ideal $\mathrm{e}_4(s,m)$ encoder that minimizes transitions (○).

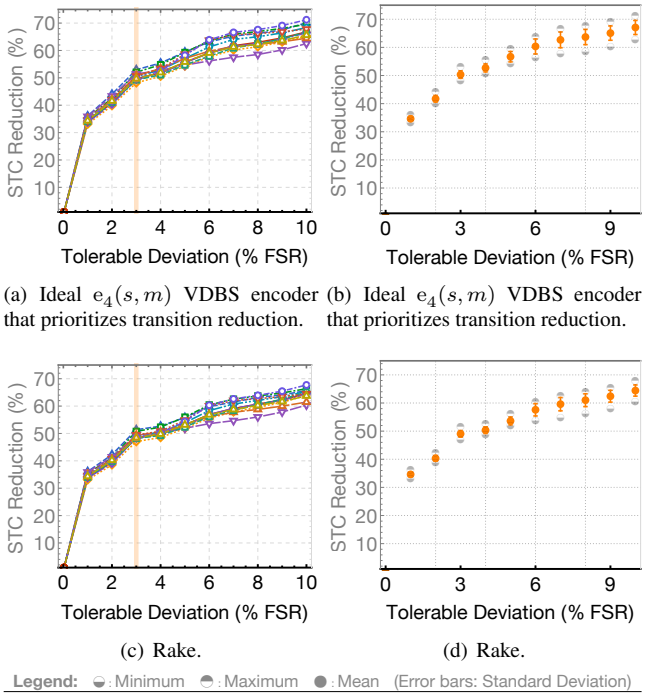


(a) Transition reduction in both data and clock interfaces.

(b) Transition reduction in only clock interface.

Figure 35. Reduction in serial transition count compared to unencoded 13-bit words, as a function of the ENOB for both unencoded shorter words (△) and values encoded by the ideal $\mathrm{e}_4(s,m)$ encoder that minimizes transitions (○).

Figure 36. Thumbnails of the 12 bitmap images used in the evaluation.



(a) Ideal $e_4(s,m)$ VDBS encoder that prioritizes transition reduction.

(b) Ideal $e_4(s,m)$ VDBS encoder that prioritizes transition reduction.

(c) Rake.

(d) Rake.

Figure 37. Reduction in serial transition count as a function of tolerable deviation, across the set of 12 test images from Figure 36.

C.6 VDBS encoding on image data

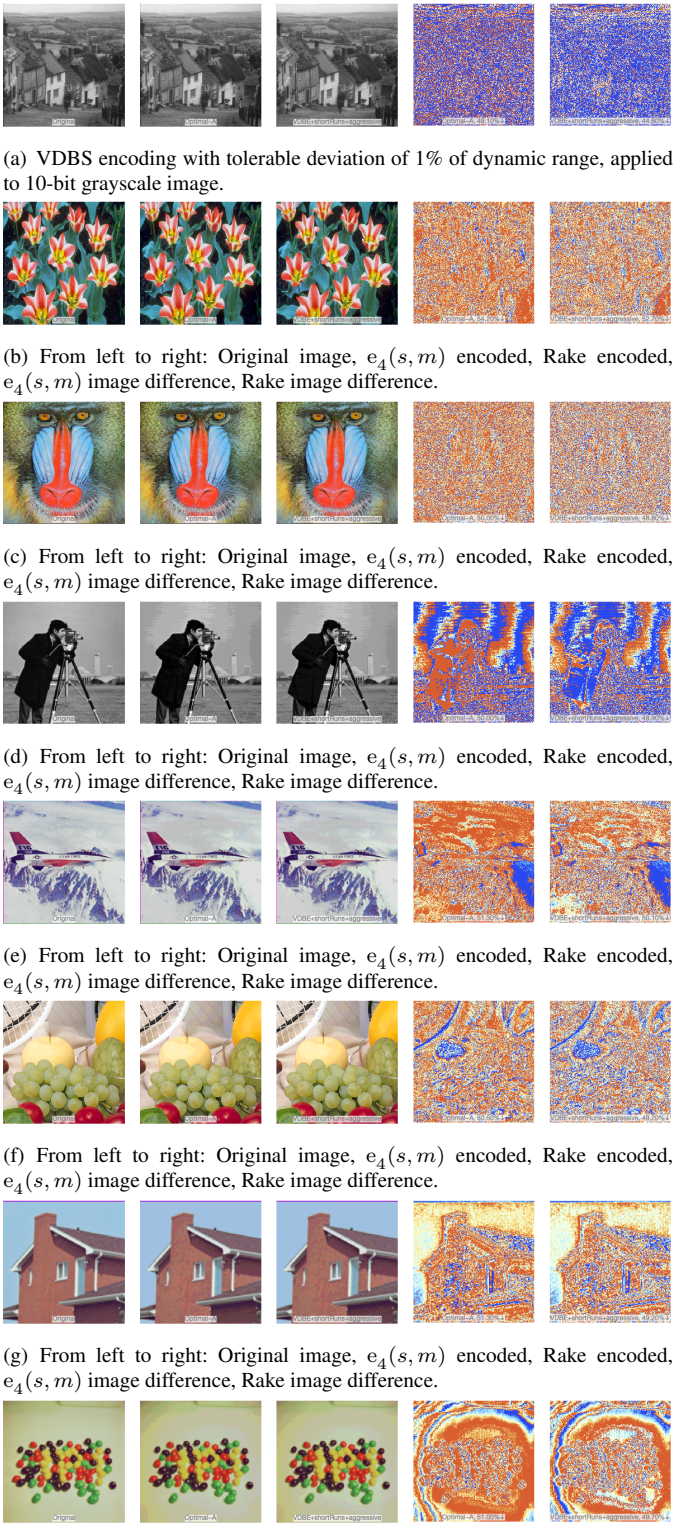
Figure 36 shows the set of 12 reference images used to evaluate the visual effect of VDBS encoding. Figure 37 shows the reduction in serial transition count across this set of images, as a function of the tolerable deviation. Figure 38 shows details of the effects of VDBS encoding for a representative subset of the images.

C.7 Encoding camera data in a text-recognition system

Figure 39 shows the set of text images used as input. Figure 40 shows the full data set for the effect of VDBS encoders on the end-to-end OCR application from Section 5.1.

C.8 Encoding accelerometer data in a pedometer system

Figure 41 and Figure 42 show details of the effect of VDBS encoders on the end-to-end pedometer application from Section 5.2. The figures illustrate the effect of encoding on the internal data representation in the pedometer application, as the data proceeds through signal processing stages.



(a) VDBS encoding with tolerable deviation of 1% of dynamic range, applied to 10-bit grayscale image.

(b) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

(c) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

(d) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

(e) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

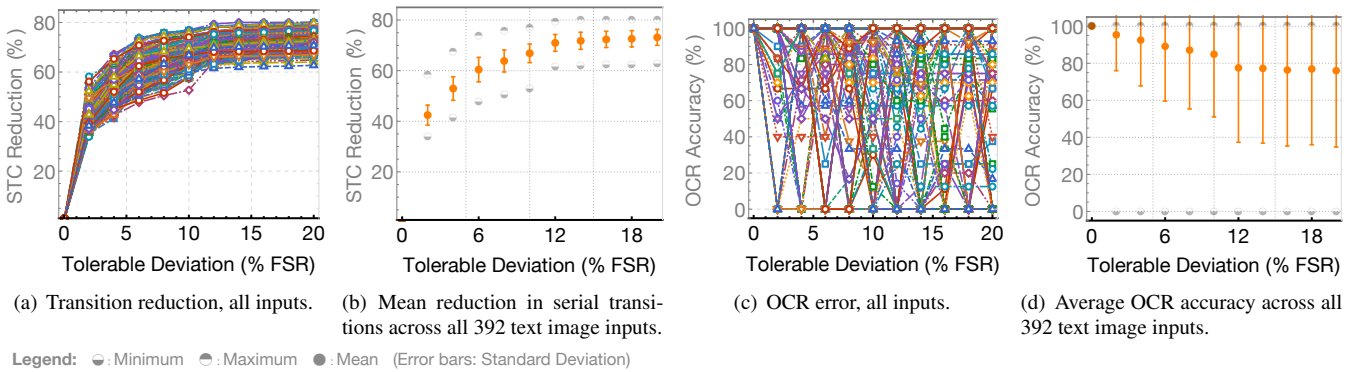
(f) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

(g) From left to right: Original image, $e_4(s,m)$ encoded, Rake encoded, $e_4(s,m)$ image difference, Rake image difference.

(h) VDBS encoding with tolerable deviation of 3 % of dynamic range, applied to 8-bit per channel RGB color image. Since VDBS encoding leads to a slight reduction in the effective number of bits used to encode values (Section 4.2), there is visible "staircase" distortion in regions with smooth gradients.

Figure 38. VDBS encoding with tolerable deviation of: (a) 1 % of dynamic range, applied to 10-bit grayscale; (b)–(h) : tolerable deviation of 3 % of dynamic range, applied to 8-bit per channel RGB color image.

Figure 39. 392 image subset from the ICDAR text recognition dataset [24] used in evaluation. This is the subset for which Tesseract [14] correctly reports OCR text identical to the benchmark-supplied ground truth.



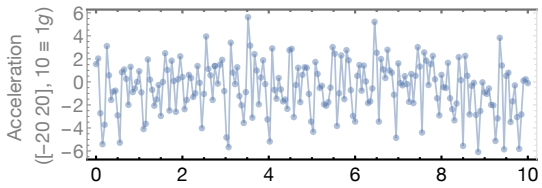
(a) Transition reduction, all inputs.

(b) Mean reduction in serial transitions across all 392 text image inputs.

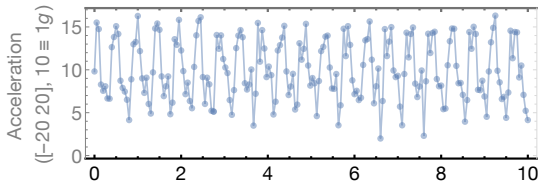
(c) OCR error, all inputs.

(d) Average OCR accuracy across all 392 text image inputs.

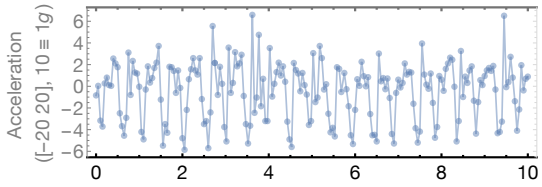
Figure 40. Rake reduces serial transitions by 55 % on average (a, b) while maintaining OCR accuracy of previously-correctly-recognized text at over 90 % (c, d).



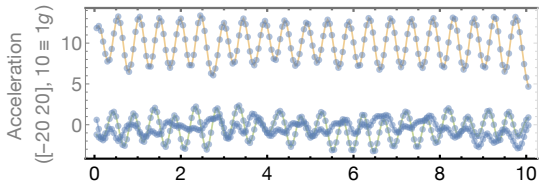
(a) X-axis accelerometer samples.



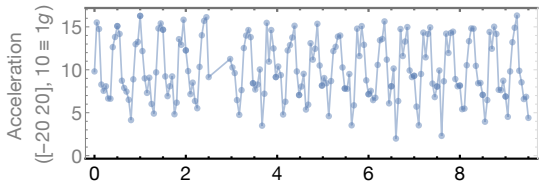
(b) Y-axis accelerometer samples.



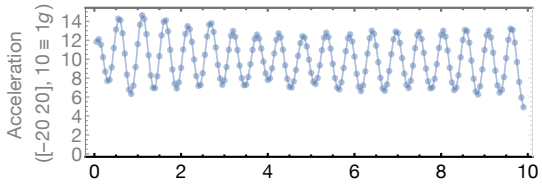
(c) Z-axis accelerometer samples.



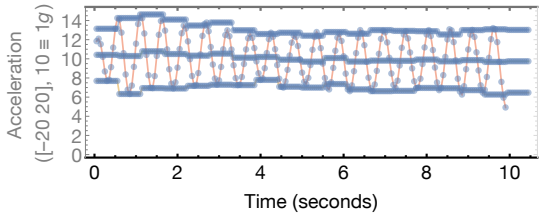
(d) All axes, low-pass filtered.



(e) Maximum-activity axis per 500 ms window.

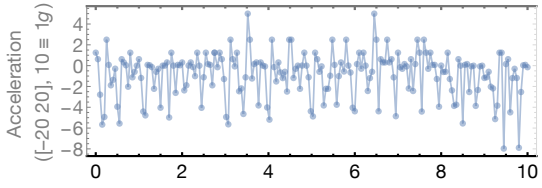


(f) Maximum-activity axis, low-pass filtered.

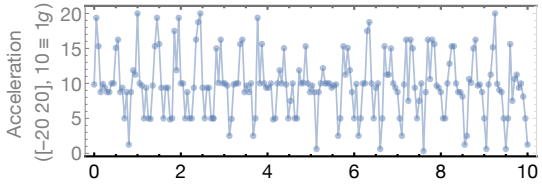


(g) Maximum-activity axis and thresholds.

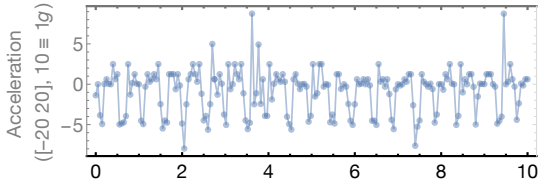
Figure 41. Unencoded accelerometer data as it progresses through pedometer algorithm stages for data from user 2 from the WISDM dataset [9]. The algorithm reports 19 steps.



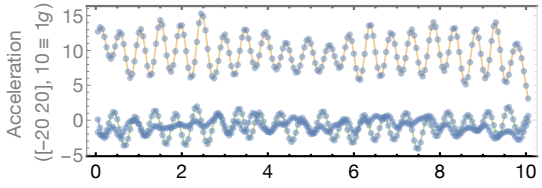
(a) X-axis accelerometer samples.



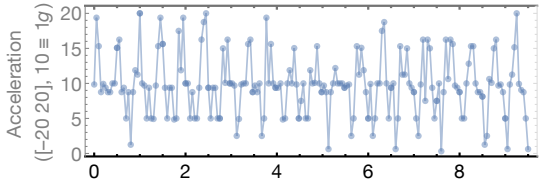
(b) Y-axis accelerometer samples.



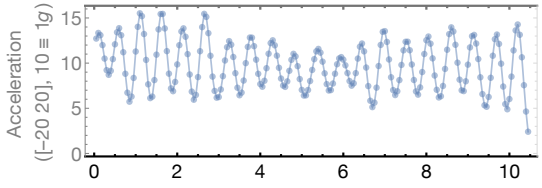
(c) Z-axis accelerometer samples.



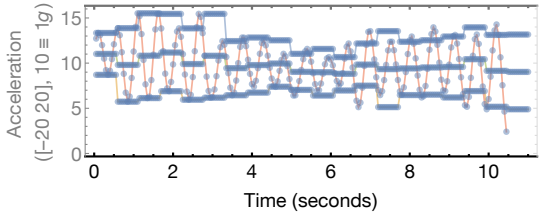
(d) All axes, low-pass filtered.



(e) Maximum-activity axis per 500 ms window.



(f) Maximum-activity axis, low-pass filtered.



(g) Maximum-activity axis and thresholds.

Figure 42. Rake encoded accelerometer data as it progresses through pedometer algorithm stages for user 2 of WISDM dataset [9]. The algorithm reports 20 steps (5.26% error). The reduction in serial transitions is however 54.38%.

References

[1] Bosch Sensortec. *BMX055 Small, Versatile 9-axis Sensor Module*, Data Sheet, November 2014.

[2] W.-C. Cheng and M. Pedram. Memory bus encoding for low power: a tutorial. ISQED '01, pages 199–204, 2001.

[3] W.-C. Cheng and M. Pedram. Chromatic encoding: A low power encoding technique for digital visual interface. DATE '03, 2003.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. DAC '13, pages 113:1–113:9, 2013.

[5] C.-T. Chiu, W.-C. Huang, C.-H. Lin, W.-C. Lai, and Y.-F. Tsao. Embedded transition inversion coding with low switching activity for serial links. *IEEE TVLSI*, 21(10):1797–1810, Oct. 2013.

[6] Freescale Semiconductor. *Kinetis KL03 32 KB Flash 48 MHz Cortex-M0+ Based Microcontroller*, Data Sheet, August 2014.

[7] R. Hegde and N. R. Shanbhag. Energy-efficient signal processing via algorithmic noise-tolerance. ISLPED '99, pages 30–35, 1999.

[8] R. Ho, K. Mai, and M. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr. 2001.

[9] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, Mar. 2011.

[10] One Stop Displays. *OSD2828GDEDF11 OEL Display Panel*, Data Sheet, 2006.

[11] Peter M. Kogge (editor). ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, Univ. of Notre Dame, CSE Dept. Tech. Report TR-2008-13, 2008.

[12] S. Rice, H. Bunke, and T. Nartker. Classes of cost functions for string edit distance. *Algorithmica*, 18(2):271–280, 1997.

[13] S. Salerno, A. Bocca, E. Macii, and M. Poncino. Limited intra-word transition codes: An energy-efficient bus encoding for lcd display interfaces. ISLPED '04, pages 206–211, 2004.

[14] R. Smith. An Overview of the Tesseract OCR Engine. *ICDAR*, 7(1):629–633, 2007.

[15] ST Microelectronics. *L3G4200D MEMS Motion Sensor: Ultra-stable Three-axis Digital Output Gyroscope*, Data Sheet, December 2010.

[16] ST Microelectronics. *LPS25H MEMS Pressure Sensor: 260–1260 hPa Absolute Digital Output Barometer*, Data Sheet, January 2014.

[17] M. R. Stan and W. P. Burleson. Bus-invert coding for low-power i/o. *IEEE TVLSI*, 3(1):49–58, Mar. 1995.

[18] M. Stephenson, J. Babb, and S. Amarasinghe. Bitwidth analysis with application to silicon compilation. PLDI '00, pages 108–120, 2000.

[19] Texas Instruments. *CC256x Bluetooth® and Dual-Mode Controller*, Data Sheet, January 2014.

[20] Texas Instruments. *HDC1000 Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor*, Data Sheet, Nov 2014.

[21] Texas Instruments. *TMP006/B Infrared Thermopile Sensor in Chip-Scale Package*, Data Sheet, November 2014.

[22] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: Energy-efficient neuromorphic systems using approximate computing. ISLPED '14, pages 27–32, 2014.

[23] H. S. Wilf and A. Nijenhuis. *Combinatorial algorithms: an update*. SIAM, 1989.

[24] S. Wong, S. Lucas, A. Panaretos, L. Velazquez, R. Young, and A. Tang. Robust word recognition dataset. In *ICDAR*, 2003.

[25] N. Zhao. Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer. *Analog Dialogue*, 44(06), June 2010.