A Programmable Analog-to-Information Converter for Agile Biosensing

Aosen Wang[1], Zhanpeng Jin[2] and Wenyao Xu[1]
[1]CSE Department, SUNY at Buffalo, NY, USA
[2] ECE Department, SUNY at Binghamton, NY, USA
{aosenwan, wenyaoxu}@buffalo.edu, zjin@binghamton.edu

ABSTRACT

In recent years, the analog-to-information converter (AIC), based on compressed sensing (CS) paradigm, is a promising solution to overcome the performance and energy-efficiency limitations of traditional analog-to-digital converters (ADC). Especially, AIC can enable sub-Nyquist signal sampling proportional to the intrinsic information in biomedical applications. However, the legacy AIC structure is tailored toward specific applications, which lacks of flexibility and prevents its universality. In this paper, we introduce a novel programmable AIC architecture, Pro-AIC, to enable effective configurability and reduce its energy overhead by integrating efficient multiplexing hardware design. To improve the quality and time-efficiency of Pro-AIC configuration, we also develop a rapid configuration algorithm, called RapSpiral, to quickly find the near-optimal parameter configuration in Pro-AIC architecture. Specifically, we present a design metric, trade-off penalty, to quantitatively evaluate the performance-energy trade-off. The RapSpiral controls a penalty-driven shrinking triangle to progressively approximate to the optimal trade-off. Our proposed RapSpiral is with $log(n)$ complexity yet high accuracy, without pre-training and complex parameter tuning procedure. RapSpiral is also probable to avoid the local minimum pitfalls. Experimental results indicate that our RapSpiral algorithm can achieve more than 30× speedup compared with the brute force algorithm, with only about 3% trade-off compromise to the optimum in Pro-AIC. Furthermore, the scalability is also verified on larger size benchmarks.

1. INTRODUCTION

Analog-to-information converter (AIC) [1] revolutionizes the design of low-power sensing micro-systems. The traditional analog-to-digital converter (ADC) is based on the Shannon-Nyquist sampling rule, while AIC is built on the basis of the compressed sensing (CS) theory [2]. CS enables sampling signals under a sub-Nyquist rate without information loss. Specifically, the CS sampling rate is proportional to the intrinsic information of signals and can significantly reduce the data volume in acquisition, transmission and analysis. Therefore, AIC can disruptively improve performance-energy trade-offs in size, weight and power (SWaP) constrained applications, such as wearable sensing [3].

There are several research work on AIC architecture design and implementation in different applications. Duarte et al. [4] fused a digital micromirror device and mathematical compressive sampling into a single-pixel camera. Chen et al. [5] applied the analog-to-information sampling into wireless electroencephalography (EEG) signal acquisition and analyzed the power and performance of the circuit model. Moreover, Baransky et al. [6] proposed a radar prototype employing AIC from both hardware and algorithm perspectives. However, these AIC structures are only designed for a specific task, which cannot be conveniently adapted in different applications. This inspires us to investigate a novel programmable AIC architecture to be universally applicable for a wide range of application scenarios. Moreover, configurability can improve the flexibility and performance-energy trade-off of AIC architecture.

In this paper, we present a novel programmable AIC architecture, Pro-AIC, to improve its flexibility in biosensing applications. The multiplexing design structure is employed to reduce the hardware and energy overhead. Furthermore, we propose a rapid configuration algorithm, RapSpiral, to quickly locate the near-optimal parameter configuration for Pro-AIC. It promotes the time efficiency, an urgent issue to evaluate the trade-off, due to the time-consuming ℓ_1 solver in CS reconstruction. Specifically, we identify a metric, trade-off penalty, to quantitatively evaluate the trade-off between performance and energy. The RapSpiral intelligently deploys the strategy of parameter configuration by a penalty-driven shrinking triangle. This technique has a rapid speed with *logarithmic* time complexity and is possible to resist the local minimum without parameter tuning. The experimental results indicate that the RapSpiral can obtain more than 30× speedup compared with the brute force algorithm, while only compromising 3% trade-off to the optimal solution. It can also keep the scalability on larger size problem.

2. COMPRESSED SENSING THEORY

Compressed sensing is a new emerging sampling scheme for the signals that are known to be sparse or compressible under a certain basis. We assume x is an N-dimension vector space and sampled using M-measurement vector y:

$$y = \Phi x, \tag{1}$$

where $\Phi \in R^{M \times N}$ is the sensing array, which models the linear encoding, and M is defined as the sampling rate in N-dimensional CS. The elements in Φ are either Gaussian random variables or Bernoulli random variables. Because of $M << N$, the formulation in Eq. (1) is undetermined, and the signal x can not be uniquely retrieved from sensing array Φ and measurements y. However, under certain sparsity-

ISLPED '16, August 08-10, 2016, San Francisco Airport, CA, USA

DOI: http://dx.doi.org/10.1145/2934583.2934596

inducing basis $\Psi \in R^{N\times N}$, the signal x can be represented by a set of sparse coefficients $u \in R^N$:

$$x = \Psi u, \tag{2}$$

that is, the coefficient u, under the transformation Ψ, only has few non-zero elements. Therefore, based on Eq. (1) and (2), the sparse vector, u, can be represented as follows:

$$y = \Phi\Psi u = \Theta_{M\times N} u, \tag{3}$$

where $\Theta_{M\times N} = \Phi\Psi$ is an $M \times N$ measuring matrix. In practical applications, original signals are analog in nature and needs to be quantized before transmitting and digital processing. Therefore, the compressed signal, y, should be processed by a quantization model formulated as follows:

$$\widehat{y} = Q_b(y), \tag{4}$$

where $Q_b(.)$ is the quantization function , and $\widehat{y}$ is the quantized representation of y with b bits. Due to the prior knowledge that the unknown vector u, is sparse, it can estimate the value, u, based on $\widehat{y}$ using the ℓ_0 minimization formulation as follows:

$$\widehat{u} = min\ \|u\|_0 \quad s.t. \quad \|\widehat{y} - \Theta u\| < \epsilon, \tag{5}$$

where ϵ is the reconstruction error margin. The formulation in Eq. (5) is a determined system with unique solutions. However, ℓ_0 minimization is an NP-hard problem. One of the methods to solve (5) is to approximate ℓ_0 minimization formulation to ℓ_1 minimization formulation:

$$\widehat{u} = min\ \|u\|_1 \quad s.t. \quad \|\widehat{y} - \Theta u\| < \epsilon. \tag{6}$$

Under the condition of Restricted Isometry Property (RIP) [7], the ℓ_1 problem is provably equivalent to minimizing ℓ_0 problem. The ℓ_1 minimization is convex and can be solved within the polynomial time. Therefore, the reconstructed signal, $\widehat{x}$, is retrieved by:

$$\widehat{x} = \Psi\widehat{u}. \tag{7}$$

3. PROGRAMMABLE ANALOG-TO- INFORMATION CONVERTER

In this section, we present the architecture of analog-to-information converter (Pro-AIC). We first introduce the circuit-level details of the Pro-AIC hardware implementation, and models of performance and energy. Then we analyze the computation issue of the entire framework to validate the importance of the rapid speed for design space exploration.

3.1 Architecture Overview

The analog-to-information architecture is shown as the dashed rectangles in Figure 1. It consists of two key components, i.e., a multiplexing randomized encoding module and a quantization module.

In Figure 1, as the circuit-level architecture shows, analog N-dimension raw sensor signal x is compressed into M-dimension y in the random encoding module. The random encoding module is designed adhering to the idea of time multiplexing [8] for power reduction. An analog multiplier is placed to calculate the product between input signal and random variable at specific time slot, which is coordinated by the multiplexing control logic. The time multiplexed integrator has M capacitors to store the intermediate result of summation. The time logic is also controlled by multiplexing control logic module. In the quantization module, there are a successive approximation register (SAR) [9] logic, a b-bit register, a digital-to-analog converter (DAC) and a comparator. When the compressed measurements come, the SAR logic sets the valid bit for the b-bit register. Then the digital register is transformed to analog value by DAC and

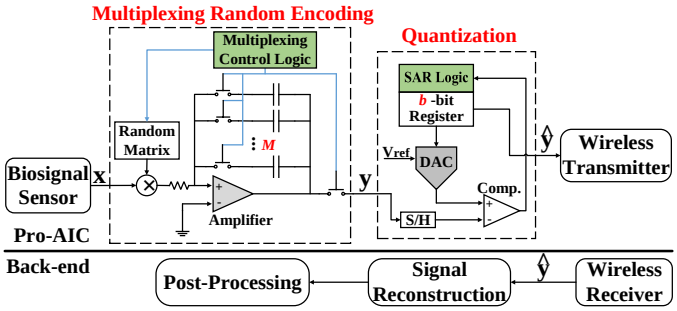


Figure 1: The block diagram of programmable analog-to-information converter architecture.

compared with the analog input signal by the comparator. The comparing result provides feedbacks for SAR logic for next-round bit setting. This procedure iteratively continues from the MSB to LSB of the b-bit register to approximate the analog input signal. A wireless transmitter streams these measurement data to the back-end, which will reconstruct the signal and execute post-processing.

3.2 Performance and Energy Models

We would like to introduce the models of energy and performance to evaluate the analog-to-information converter architecture. In wireless sensing applications, the wireless data exchanging becomes the bottleneck of power consumption in the sensor node, due to the low-power design of Pro-AIC. Therefore, the *energy consumption* is proportional to the volume of data stream in wireless communication formulated as follows:

$$E = C \times M \times b, \tag{8}$$

where M is the sampling rate, b is the bit resolution, and C is the energy per bit in wireless communication, which is determined by the specific protocol and usually a constant. In addition, we adopt the normalized root-mean-square difference between reconstructed signal and original signal as the *performance* metric of the entire architecture:

$$P = \frac{\|x - \widehat{x}\|_2}{\|x\|_2} \times 100\%, \tag{9}$$

where $\widehat{x}$ denotes the recovered signal, and x is the original input signal.

3.3 Challenges in Pro-AIC Configuration

The main challenge of Pro-AIC design is how to efficiently find the optimal parameter configuration, i.e., sampling rate M and bit resolution b. The power consumption in the Pro-AIC sensor node is proportional to the total transmitting bits, product of M and b. However, the performance of the Pro-AIC framework is based on ℓ_1 minimization solver, which is an extremely time-consuming procedure. We take a 256-sample EEG segment reconstruction as an example. The sizes of M and b are 256 and 16, respectively. We use CVX toolbox [10] to solve the minimization problem on the computer with 3.3GHz Intel i7 and 8GB RAM. The average time to compute the reconstruction under a specific parameter setup (M, b) is about 1.5 seconds. Under the brute force algorithm, the total time is the average time multiplying the design space size, about 1.7 hour in our example. In CVX, the ℓ_1-norm minimization problem is solved by second-order cone programming (SOCP) [11] method with time complexity of $O(N^3)$. If the signal length increases to 1024 samples, the brute-force needs about one and a half days. Therefore, a method to quickly detect the optimal parameter configuration is in an urgent need.

4. RAPID NEAR-OPTIMAL CONFIGURATION ALGORITHM IN PRO-AIC

In this section, we first identify a design metric, trade-off penalty to quantitatively evaluate the P-E trade-off. Then we discuss some insights from the distribution of the trade-off penalty in M-b space. Finally, we elaborate our rapid spiral scheme in details.

4.1 Design Metric: Trade-off Penalty

Our ultimate goal is to evaluate the P-E trade-off with respect to two parameters, sampling rate M and bit resolution b. To remove the scale effect, we first normalize the reconstruction error, P, and energy consumption, E, with their maximal values. To this end, we define a trade-off penalty Ct to indicate the endeavour to achieve a certain level of the P-E trade-off, as the following:

$$Ct(M,b) = \|(P_n(M,b), \alpha E_n(M,b)) - (0,0)\|_2, \qquad (10)$$

where P_n is the normalized reconstruction error and E_n is the normalized energy under the specific parameter configuration (M,b). The α is a parameter to tune the relative weight between P_n and E_n, which is always decided by specific application. The Ct is the Euclidian distance between current P-E point and the origin point.

Furthermore, if we take a close look at Eq. (10), we can find that the origin point is the ideal case, which holds no reconstruction error with zero energy consuming. The design space of Pro-AIC comprises all the possible P-E points. So we can infer that a good parameter configuration towards the optimal P-E trade-off in the design space is the P-E point who holds the minimal distance to the origin. The related problem can be formulated as the following:

$$(M,b)_{opt} = arg\min_{M,b} Ct(M,b). \qquad (11)$$

The smaller trade-off penalty means better trade-off achieved by our proposed architecture. Therefore, our objective in Pro-AIC design is to rapidly determine the optimal parameter configuration who holds the smallest trade-off penalty.

4.2 Trade-off Penalty Distribution in M-b Space

We have a look at the distribution of the trade-off penalty $Ct(M,b)$ to investigate its trend in the 2-D parameter space. An example of a real EEG signal segment to show the relationship between penalty Ct and the parameters, M and b, is depicted in Figure 2. Our objective is to seek for the global minimum trade-off penalty in M-b design space.

From the heat map in Figure 2, we can observe that the largest penalty regions appear at the left bottom corner and right top corner. The left bottom is with almost zero energy, but the configuration of small M and b will cause large reconstruction error. The right top is with the best reconstruction quality, but consuming the largest energy. So these two extremes are marked as the largest penalty. The minimum penalty, the optimal, locates at the middle part in this 2-D space, marked by the pink five-pointed star. We can also see that the area around the right top corner has a significant decreasing trend. Although the left bottom area also has the penalty-decreasing trend, there are too many fluctuations.

An intuitive heuristic method for trade-off penalty minimization is to use gradient descent (GD) algorithms. To keep a comprehensive study, besides taking the right top as the starting point, numbered as 1, we also randomly choose other eight starting points, numbered from 2 to 9. We show all the traces of GD by the white triangle marker from each starting point, holding locally largest trade-off penalty. A

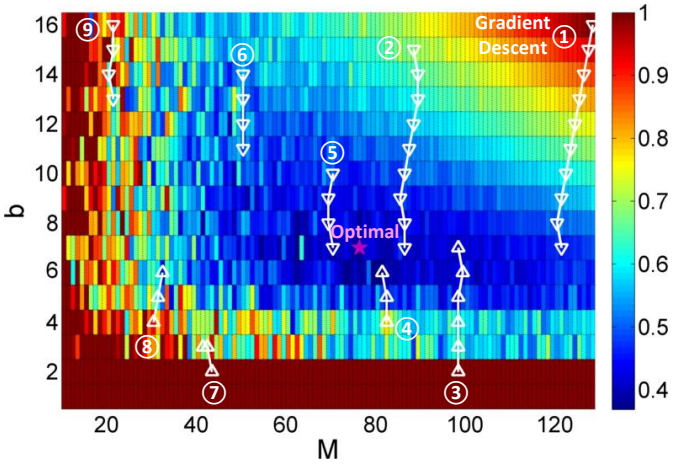


Figure 2: The trade-off penalty distribution in M-b space. The x axis is the sampling rate and the y axis is the bit resolution. Different colors indicate different penalty levels.

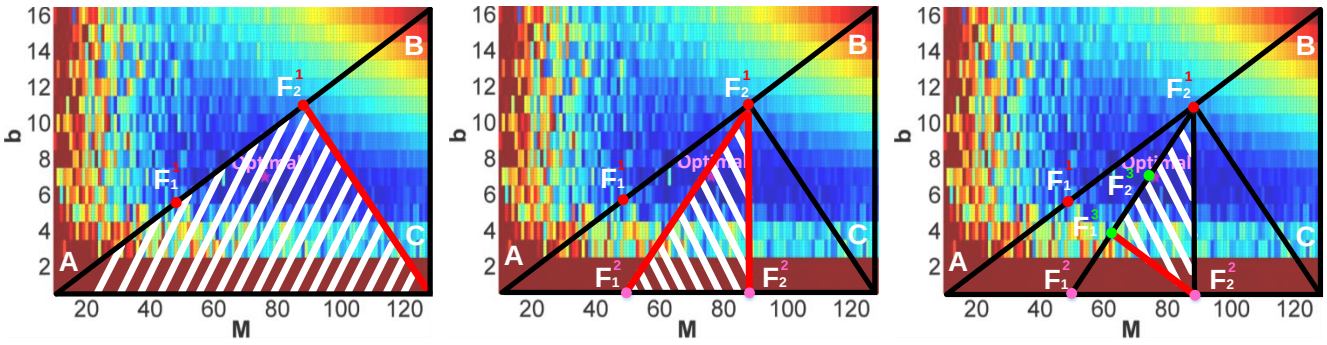
common trend is that the penalty decreases fast by declining of both M and b, but it is trapped into the local minimum after just a few steps. We can observe that there are numerous local minimum distributed in the design space, due to the lack of dominator between M and b. Especially, the starting point of case 4 and 5 are just near the optimal solution, but they don't have a chance to converge to the optimum. Therefore, the GD has a poor result affected by so many local minimum confusions, which make the rapid parameter configuration become an extremely difficult problem.

The neighbor by neighbor searching scheme makes GD easy to stop at a point with large penalty. The key to improving the result quality is to explore the possibility to resist the local minimum, yet keeping the rapid speed as GD. Note that a prior knowledge is we already know the size of the M-b space. It will be feasible to push penalty-driven triangle to converge to the small penalty area.

4.3 Rapid Configuration Algorithm

Our rapid configuration algorithm, RapSpiral, maintains a penalty-driven triangle to move towards the minimum penalty area, by drifting and shrinkage alternately. *We have two rules through the proposed scheme: 1), We always choose the longest triangle edge to evolute; 2), We adopt penalty comparison for edge trim to generate the new triangle.* This method can jointly optimize the trade-off penalty and time-efficiency. We provide a typical example to illustrate our idea in Figure 3.

Initially, we place a triangle on the design space, just like the triangle ABC in Figure 3 (a). *To avoid the drawback from neighborhood searching, we use the penalties of the three-equal-partition points on the edge to evolute the triangle.* This penalty-driven triangle can efficiently peel off the large penalty area and shrink to the area with small penalty. So we first partition edge AB into three-equals and compare the penalties on the two partition points, F_1^1 and F_2^1. Note that we use superscript to indicate the iteration number and adopt subscript as the partition number for partition points. If the penalty $Ct(F_1^1)$ is less than $Ct(F_2^1)$, we replace the edge endpoint B with F_2^1. Then if the case is that $Ct(F_1^1)$ is greater than $Ct(F_2^1)$, we replace the endpoint A with F_1^1. If the two penalties are equal, we update two endpoints simultaneously. In this example, we have $Ct(F_1^1)$ is less than $Ct(F_2^1)$, so we delete the area BCF_2^1. We can observe that the small penalty area is just between CF_2^1 and CF_1^1. There is no need to search the parameter config-



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Figure 3: A typical example to illustrate our RapSpiral algorithm to search for the optimal parameter configuration.

uration in BCF_2^1. Then, the new triangle becomes CAF_2^1. We drastically reduce the searching burden by this penalty-driven model. *Another significant heuristic technique is we always choose the longest edge to start the new evolution.* This is also the reason why we choose edge AB for the first triangle updating.

Accordingly, we move to iteration 2 in Figure 3 (b). In this iteration, we choose edge AC to evolute because it is the longest edge in CAF_2^1. The penalties are equal from the both equal-partition points F_1^2 and F_2^2. Thus, we update the two endpoints A and C with partition points F_1^2 and F_2^2, respectively. Then the new evoluted triangle is $F_1^2F_2^2F_2^1$. Note that we record the minimum penalty and corresponding parameter configuration for all our visited partition points:

$$Ct_{min} = \min_z\{Ct(F_1^z), Ct(F_2^z)\}. \quad (12)$$

This step is an easily ignored yet very significant measure to empower our spiral scheme possible to avoid the local minimum. *Because our stopping criterion is the area of the penalty-driven triangle less than a pre-defined threshold ϵ. The optimal penalty may occur at anytime in the evolution procedure. It is this technique that enlightens our method a much higher probability to hit the optimum than GD.* We will demonstrate this in the following experiment. In iteration 3 of Figure 3 (c), we continue to execute our rapid spiral scheme. We can observe that the penalty-driven triangle cuts off the large penalty area continuously and converges to the optimal penalty area. The pseudo code of the entire algorithm is listed in Algorithm 1.

In this algorithm, line 4 chooses the longest edge of the triangle. The line 6 calculates the two equal-partition points. The two if structures from line 7 to 12 judge the penalty comparison and make the decision of the updating for next iteration. Finally, from line 14 to 16, we record the configuration information if any one of the two points has smaller penalty. When the iteration terminates, we find the near-optimal parameter configuration (M, b) and corresponding trade-off penalty.

LEMMA 1. *The time complexity of the rapid spiral algorithm is $O(log(|M| \times |b|))$.*

Proof: In this algorithm, the area of one single triangle will be reduced at least one third at each iteration. This procedure will continue until the area of the current triangle is no more than the threshold ϵ. In the worst case, we only need to consider the evolved triangle with two thirds area remaining in each iteration. The recursive formula is $T(area)$=$T(2/3area)$+$O(1)$, and by the Master Theorem [12], the rapid spiral algorithm will converge by the time complexity of $log(area) = log(|M| \times |b|)$.

Algorithm 1 RapSpiral: Rapid Configuration Algorithm

Input: The axis of initial triangle A, B and C. Ct is the trade-off penalty.
Output: The optimal parameter (M_{opt}, b_{opt}) and the minimum penalty Ct_{min}

```
1: Ct_min ← INF;
2: while Area(ΔABC)> ε do
3:     //Choose the longest edge
4:     Line ← LongestEdge(ΔABC);
5:     //Spiral evolution
6:     (f1, f2) ← PartitionPointFind(Line);
7:     if Ct(f1)≥Ct(f2) then
8:         Line.startPoint ← f1;
9:     end if
10:    if Ct(f1)≤Ct(f2) then
11:        Line.endPoint ← f2;
12:    end if
13:    //Update parameters and minimum cost
14:    Ct_min ← min(Ct(f1), Ct(f2), Ct_min);
15:    (M_opt, b_opt) ← min_{M,b}{Ct_min};
16:    UpdateTopology(ΔABC);
17: end while
18: return (M_opt, b_opt), Ct_min;
```

5. EXPERIMENTS AND DISCUSSION

In this section, we perform extensive experiments on biosignal sensing to quantitatively demonstrate the performance of RapSpiral algorithm on configurability exploration. We compare parameter estimation accuracy and runtime among RapSpiral, gradient descent and brute force algorithms.

5.1 Experimental Setup

In biosignal family, EEG is a challenging representative, so we use the public available EEG benchmarks from the PhysioNet [13] as our testing data. We choose three lengths of EEG signal, N=128, 256, and 512. The 128-sample EEG has 120 segments and the other two lengths have 10 segments of each. The sampling rate of all the EEG segments is 256 Hz. All the experiments use Bernoulli random array as sensing array and uniform quantization strategy. The inverse discrete wavelet transform is taken as the orthogonal transformation basis Ψ. The weight parameter α is set as 1. All our experiments are carried out by Matlab, with Intel Core i7 3.4 GHz and 8 GB memory. We adopt IPv6-based communication [14] to model energy consumption of wireless data transmission. By its experimental result, we can take $C = 0.4$ uJ/bit in the energy model. We use sensitivity analysis [15] as the gradient descent method in discrete domain.

5.2 Trade-off Comparison

In this part, we examine the trade-off penalty among the RapSpiral, gradient descent and brute force algorithms. We use 120 continuous 128-sample EEG segments as the testing data. The maximal quantization bit resolution is 16 and the maximal measurement number is 128. For our RapSpiral method, we divide the entire design space into two equal triangles by the diagonal from the left bottom to the right top. The gradient descent starts from the right top corner, where b is 16 and M is 128. The brute force algorithm directly visits each possible parameter configuration. Due to the limited space, we illustrate accuracy results of 10 segments in Figure 4.

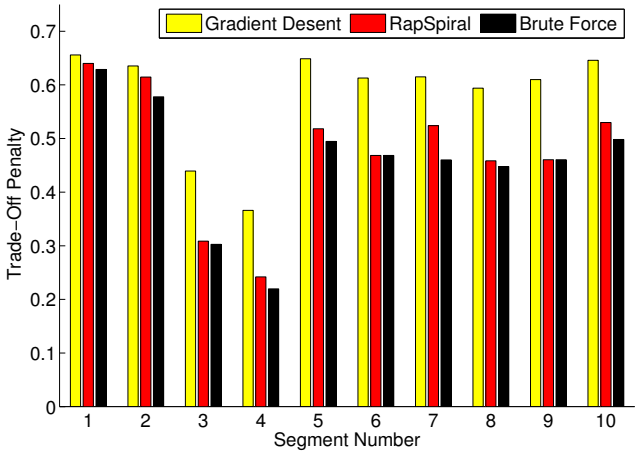
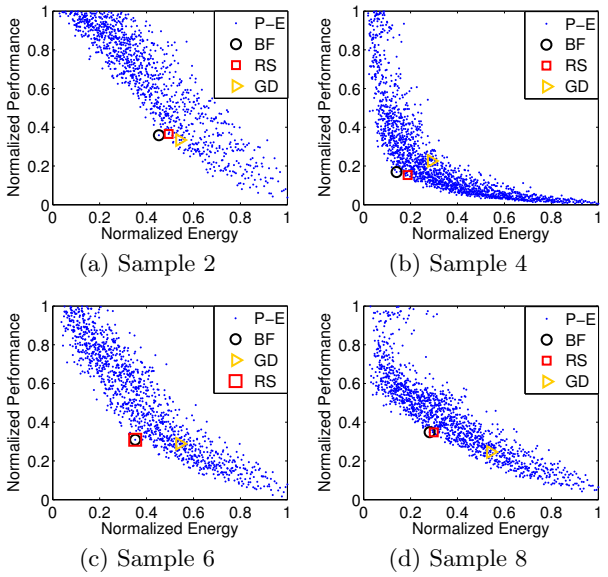


Figure 4: The minimum penalty of RapSpiral, gradient descent and brute force.

In Figure 4, the yellow bar indicates the minimal penalty found by gradient descent, the red bar is the result of our RapSpiral, and the black bar corresponds to the brute force algorithm. Intuitively, the brute force algorithm obtains the optimal solution. The penalty of our RapSpiral is much closer to the brute force case than gradient descent method. Specifically, the segment 6 and 9 both find their optimal solution. But the gradient descent is with poor accuracy without optimum hit experience. This is because gradient descent method is easily trapped into the local minimum.

More specifically, we choose four segments, number 2, 4, 6 and 8, to show their final parameter finding in the normalized P-E space in Figure 5. We can find that the three parameter configurations are very close for segment 2. For the other three segments, the configuration of gradient descent is far behind the brute force and our RapSpiral. Especially, our RapSpiral hits the optimal solution in Segment 6. If we have a closer look at the results from the four segments, a common trend is that our RapSpiral can always be near the inner envelop curve, while the gradient descent stops in the middle of the P-E space. The gradient descent is sensitive to the local minimum, which is fully distributed in the Pro-AIC design space. This firmly demonstrates the robustness of our RapSpiral algorithm resisting to the local minimum towards the near-optimal P-E trade-off.

For a comprehensive comparison, we define the average trade-off penalty ATP to the optimal configuration in the brute algorithm for specific algorithm as, $ATP = \frac{1}{L}\sum_i (Ct^i_{spec} - Ct^i_{brute})$, where the Ct^i_{spec} is the minimum penalty of the specific algorithm, either RapSpiral or gradient descent, for segment i. The Ct^i_{brute} is the global minimum penalty. The average penalties are 11.41% and 3.48% for the gradient descent and our RapSpiral. We use the maximal value to normalize each attribute in P-E space, so 3.48% makes a great difference compared with 11.41%. This demonstrates the high accuracy of our proposed RapSpiral algorithm.



(a) Sample 2 (b) Sample 4

(c) Sample 6 (d) Sample 8

Figure 5: The searching result of different strategies, gradient descent (GD), RapSpiral (RS) and brute force (BF).

5.3 Runtime Comparison

We design this experiment to compare the runtime of the three algorithms. The setup is the same as the accuracy comparison section, and the results of run time are shown in Figure 6. Note that due to the large runtime of the brute force case, we adopt the logarithmic scale of the runtime axis for a clearer comparison.

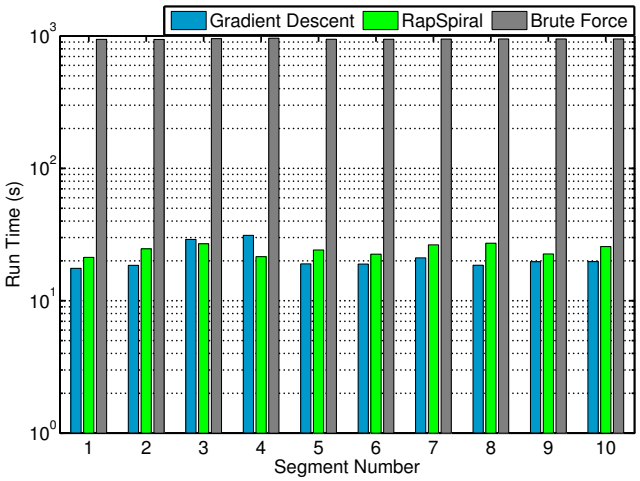


Figure 6: The runtime comparison of RapSpiral, gradient descent and brute force.

In Figure 6, the gradient descent and RapSpiral have the similar runtime, which are both much less than that of the brute force case. RapSpiral has a simple yet efficient searching strategy. The $O(log(|M| \times |b|))$ time complexity enables its rapid speed. For gradient descent in this experiment, it is trapped into the local minimum after several steps forwarding. Specifically, the gradient descent method goes down along the decreasing direction of bit resolution b. This is because large sampling rate M makes the bit resolution b dominate the signal reconstruction. When the gradient descent comes near the area with small penalty, the numerous

local minimum terminates its searching.

More specifically, we define SP_{avg} to to check the average runtime speedup compared with the brute force method over all the 120 EEG segments, formulated as, $SP_{avg} = \frac{1}{L}\sum_i (t^i_{brute}/t^i_{spec})$, where the t_{brute} is the entire runtime of the brute force algorithm and the t_{spec} represents the runtime of a specific method, gradient descent or RapSpiral. By this formula, we can calculate the average runtime speedups for gradient descent and RapSpiral as 42.94× and 38.81×, respectively. The two speed-ups are similar. This demonstrates the time efficiency of our RapSpiral algorithm.

5.4 Scalability

This experiment is designed to investigate the scalability of the RapSpiral algorithm. We choose ten 256-sample EEG segments and ten 512-sample EEG segments. We still keep the upper bound of bit resolution b as 16. The RapSpiral also divides the parameter M-b space into two triangles by the diagonal from left bottom to right top. The gradient descent begins its searching from the right top corner. The trade-off penalty of the brute force algorithm is the ground truth. The related trade-off penalty and runtime speedup are illustrated in Figure 7.

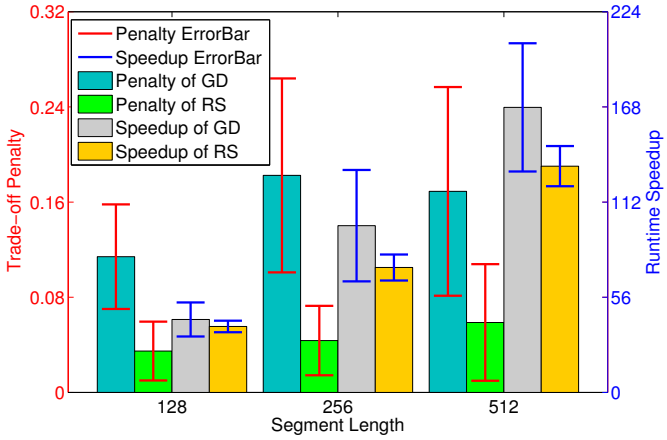


Figure 7: The scalability results of different sizes of design space.

The average penalties of GD and RapSpiral are represented by light-blue and green bars. The average speedup of GD and RapSpiral are grey and yellow bars. We also add two types of errorbars, i.e., the red one is the standard deviation of the trade-off penalty and the blue one indicates the standard deviation of runtime speedup. For the penalty, our RapSpiral is much less than the gradient decent method in all cases. As the segment length increases from 128 to 512, the average trade-off penalty of RapSpiral increases by a small amount, about 1%. However, the gradient descent method takes on fluctuations, which is related to the location of the local minimum.

For our RapSpiral framework, its speedup shows a linear increasing trend as the segment length double and double again. The time complexity of RapSpiral algorithm is $log(|M| \times |b|))$. As M increases by polynomial order k, we may have the complexity as $log(M^k) = klog(M)$. The complexity is transformed into linear increasing. This demonstrates the excellent scalability of our RapSpiral algorithm.

6. CONCLUSION AND FUTURE WORK

In this work, we investigated a programmable analog-to-information converter architecture and RapSpiral, an algorithm to rapidly optimize the configurability of Pro-AIC sensor node design. We introduced the basics of compressed sensing theory and Pro-AIC architecture. We presented a design metric, trade-off penalty, to quantitatively evaluate P-E Trade-off. Moreover, we proposed RapSpiral algorithm for the near-optimal parameter configuration. This method had no pre-training requirement and parameter tuning procedure. It guaranteed that the chosen triangle moves towards the optimal area. We also carried out extensive experiments on challenging EEG signals to verify the performance of our proposed algorithm. Experimental results demonstrated the high accuracy and fast speed of our RapSpiral.

In future work, we plan to design new efficient algorithms to take more parameters into account in Pro-AIC design to pursue better trade-off improvements. On the other hand, we will consider the micro-architecture optimization of the Pro-AIC sensor node.

7. ACKNOWLEDGMENTS

This work is in part supported by NSF grants CNS-1423061/ 1422417, ECCS-1462498/146247 and CNS-1547167.

8. REFERENCES

[1] Sami Kirolos, Jason Laska, Michael Wakin, Marco Duarte, Dror Baron, Tamer Ragheb, Yehia Massoud, and Richard Baraniuk. Analog-to-information conversion via random demodulation. In *Design, Applications, Integration and Software, 2006 IEEE Dallas/CAS Workshop on*, pages 71–74. IEEE, 2006.

[2] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[3] Aosen Wang, Zhanpeng Jin, Chen Song, and Wenyao Xu. Adaptive compressed sensing architecture in wireless brain-computer interface. In *Proceedings of the 52nd Annual Design Automation Conference*, page 173. ACM, 2015.

[4] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin E Kelly, Richard G Baraniuk, et al. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83, 2008.

[5] Fred Chen, Anantha P Chandrakasan, and Vladimir M Stojanovic. Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors. *Solid-State Circuits, IEEE Journal of*, 47(3):744–756, 2012.

[6] Eliahu Baransky, Gal Itzhak, Noam Wagner, Idan Shmuel, Eli Shoshan, and Yonina Eldar. Sub-nyquist radar prototype: Hardware and algorithm. *Aerospace and Electronic Systems, IEEE Transactions on*, 50(2):809–822, 2014.

[7] Argyrios Zymnis, Stephen Boyd, and Emmanuel Candes. Compressed sensing with quantized measurements. *Signal Processing Letters, IEEE*, 17(2):149–152, 2010.

[8] Zainul Charbiwala, Paul Martin, and Mani B Srivastava. Capmux: A scalable analog front end for low power compressed sensing. In *Green Computing Conference (IGCC), 2012 International*, pages 1–10. IEEE, 2012.

[9] Shitong Yuan, Hai Huang, Qilian Liang, and Qiang Li. Energy efficient comparator for successive approximation register adcs with application to wireless sensor networks. *International Journal of Sensor Networks*, 17(2):122–129, 2015.

[10] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[11] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1):193–228, 1998.

[12] Salvador Roura. An improved master theorem for divide-and-conquer recurrences. In *Automata, Languages and Programming*, pages 449–459. Springer, 1997.

[13] George B Moody, Roger G Mark, and Ary L Goldberger. Physionet: a web-based resource for the study of physiologic signals. *IEEE Eng Med Biol Mag*, 20(3):70–75, 2001.

[14] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 232–237. IEEE, 2012.

[15] Aosen Wang, Feng Lin, Zhanpeng Jin, and Wenyao Xu. A configurable energy-efficient compressed sensing architecture with its application on body sensor networks. *Industrial Informatics, IEEE Transactions on*, 12(1):15–27, 2015.