Compressed-Sensing Sigma Delta ADC with SPG, OMP and CoSaMP

This is a MATLAB simulation program for compressed-sensing sigma-delta analog to digital converter (ADC).

Introduction to Basis Pursuit Denoising

The idea of compressed-sensing comes from basis pursuit denoising (BPDN) (https://en.wikipedia.org/wiki/Basis_pursuit_denoising) problem in machine learning. Basically, this problem has three equivalent forms:

1. Basis pursuit denoising: Minimize $\|x\|_1$ subject to $\left\|A\vec{x} - \vec{y}\right\|_2 \leq \sigma$
2. Basis pursuit: Minimize $\|x\|_1$ subject to $A\vec{x} = \vec{y}$
3. Lasso: Minimize $\left\|A\vec{x} - \vec{y}\right\|_2$ subject to $\|x\|_1 \leq \tau$

Machine learning researchers have developed various kinds of algorithms to solve BPDN problem. Some basic textbook materials can be found in Kevin P. Murphy's Machine Learning: A Probabilistic Perspective (https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_1?ie=UTF8&qid=1484504311&sr=8-1&keywords=machine+learning+a+probabilistic+perspective) Chapter 13.2.3 sparse linear models.

In this project, we are going to use three algorithms implmente in MATLAB:

1. SPGL1: Spectrial Projected Gradient for L1 minimization (https://github.com/mpf/spgl1)

   This algorithm is developed by Professor Michael P. Friedlander fron University of British Columbia. The details can be found in his paper Probing the Pareto frontier for basis pursuit solutions (https://www.cs.ubc.ca/~mpf/pubs/probing-the-pareto-frontier-for-basis-pursuit-solutions/). As a general comment on this implementation, it is an iterative algorithm that can solve large scale BPDN reconstruction problem. However, even for small scale problem it still runs for many iterations and takes a relative long time to find the root.
2. OMP: Orthogonal Matching Pursuit

   This algorithm is proposed by Stephane G. Mallat (https://www.di.ens.fr/~mallat/papiers/MallatPursuit93.pdf) in 1993. It is a greedy search algorithm on BPDN least square equation. It is suitable for small scale problem when dimension of X is small and computing requirement is affordable. In this scenario, it is much faster than SPGL1 but we do observe that it has some accuracy problem. Due to its greedy nature, OMP sometimes could not find all the features of X while SPGL1 could. In this project, we use Stephen Becker (https://www.mathworks.com/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery)'s implementation in MATLAB.
3. CoSaMP:Compressive Sampling Matched Pursuit

   This algorithm is from D. Needell, J. A. Tropp (https://arxiv.org/pdf/0803.2392v2.pdf). At hearts, this algorithm is greedy pursuits but it also incorporates ideas from the combinatorial algorithms to guarantee speed and to provide rigorous error bounds. The details can be found in the paper. In this project, we use Stephen Becker (https://www.mathworks.com/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery)'s implementation in MATLAB.

Introduction to Compressed Sampling

Here, I will present a highly simplified introduction to the compressed sampling. For more details and rigorous mathematical proof, please refer to the paper An Introduction To Compressive Sampling (http://authors.library.caltech.edu/10092/1/CANieeespm08.pdf).

What is signal sensing or sampling:

First, we will discuss what is sensing mechanism, in which information about a time domain signal $f(t)$ is obtained by a series of recording values $y_k = < f, \varphi_k >$. In another word, we correlate signal $f(t)$ with a series of pre-defined standard signals $\varphi_k(t)$, for which we have various kinds of choices. For example, if $\varphi_k(t)$ are Dirac delta functions (spikes), then we are sampling recording $f(t)$ with its discrete sampled values y. If $\varphi_k(t)$ are sine wave functions $sin(\omega_k t)$ in which $\omega_k = k \times \omega$, we are actually transforming $f(t)$ into frequency domain and this process is called discrete Fourier transformation. We give these pre-defined standard signals $\varphi_k(t)$ a new name: basis function. If they are orthonormal basis, we call them orthobasis. The sine wave functions described above is an orthobasis example.

Sparse signal and sparsity:

Mathematically speaking, in previous section, we have a signal f in t-domain (note this domain is not limited to time domain and can be extended to other domain, for example, two-dimension space domain for images) and we expand f in an orthonormal basis $\Phi = [\varphi_1, \varphi_2, \ldots, \varphi_3]$ as follows:

$$f(t) = \sum_{i=1}^{n} x_i \varphi_i(t)$$

By this expansion, we transform f from t-domain to a new domain and represent it with a series of coefficient $x_i = < f, \varphi_i >$. $f(t)$ may have vary complex form in t-domain, however, it could be sparse in the new domain. That is, many entry of x_i is actually 0 or very close to 0 and only S entries are nonzero. We denote these nonzero entries in a set X_S and call S as the sparsity of the original signal $f(t)$. To make it more clear, we consider $f_s(t)$ obtained by keeping only S nonzero terms. Here, we define

$$f_s := \Psi X_S$$

where from here, X_S is the vector of coefficients (X_i) with all but S entries set to zero. We say f_S is the sparse and approximate representation of the original signal $f(t)$ and we have

$$\|f - f_s\|_{l_2} = \| X - X_s \|_{l_2}$$

If coefficient X is sparse and compressible in the sense that the sorted magnitude of X_i decay quickly, then X should be well approximated by X_S and the error $\|f - f_s\|_{l_2}$ should be small. If you are farmilar with image processing, JPEG format is a very good example of this principle.

Orthobases coherence:

From previous section, we know that for a given signal f, it can have different representations in different orthobases. Let us consider two orthobases (Φ, Ψ). We define the coherence between these two orthobases as

$$\mu(\Phi, \Psi) = \sqrt{n} \cdot \max_{1 \le k,j \le n} \left| < \varphi_k, \psi_j > \right|$$

Basically, the coherence is checking how correlated those two orthobases are. You can image that if two orthobases are the same, the coherence is the maximum possible $\sqrt{n}$.

How about the least coherent orthobases? One example is the time domain spike basis $\varphi_k = \delta(t-k)$. This is the most common basis we use in real life. When you record a signal's waveform, you are actually using the spike basis. Its reciprocal basis is Fourier basis, $\psi_j(t) = \sqrt{n}e^{i2\pi jt/n}$, has coherence $\mu(\Phi, \Psi) = 1$. This is the smallest coherence we can have and thus we say spike basis and Fourier basis are maximally incoherent.

Finally, we would like point out that a random matrices are largely incoherent with any fixed basis Ψ. We will omit the rigorous proof here (if you are interested, you can refer to the paper I mentioned at the beginning) but just stated: with high probability, an orthobasis Φ uniformly at random is highly incoherent with any fixed basis Ψ. And the coherence is about $\mu(\Phi, \Psi) \approx \sqrt{2logn}$. As an exmaple, we will use a matrix

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$$

with entry a_{ij} equal to +1 or -1 randomly. This matrices is highly incoherent with spike basis or Fourier basis.

Sparse signal reconstruction:

Ideally, we would like to collect or measure all the n coefficient of f in our observation orthobasis. However, if we only collect part of the data

$$y_k = < f, \varphi_k >, k \in M$$

where M is a subset of cardinality $m < n$. Here, the only other information we know is in some domain Ψ the signal f has a sparse representation x. Maybe with the partial observation, we cannot reconstruct the exact signal f in general. But we can find the approximate reconstrution f^* given by $f^* = \Psi x^*$, where x^* is the solution to the optimization problem:

$$\min_{\widetilde{x} \in \mathbb{R}^n} \|\widetilde{x}\|_{l_0} \ subject\ to\ y_k = \ < \varphi_k, \Psi\widetilde{x} >, \forall k \in M$$

By minimize the L0 norm of $\widetilde{x}$, we actually mean to find as sparse solution as possible. However, we know that minimization of L0 norm is not a convex problem, instead, minimization of L1 norm is. Furthermore, we do know that minimization of L1 norm actually proposes a sparse solution and it is equivalent to minimization of L0 norm. Therefore, with a single change from L0 to L1, we have a solvable convex optimization problem:

$$\min_{\widetilde{x} \in \mathbb{R}^n} \|\widetilde{x}\|_{l_1} \ subject\ to\ y_k = \ < \varphi_k, \Psi\widetilde{x} >, \forall k \in M$$

Next, we will prove that if f is sufficiently sparse, the recovery via L1 norm minimization is actually exact rather than apporximate.

Theorem 1:

Suppose the coefficient vector x of f in the basis Ψ is S-sparse. Select m measurement in the observation domain ϕ and ϕ is uniformly random. Then if

$$m \ge C \times \mu^2(\Phi, \Psi) \times S \times \log n$$

for some positive constant C, the solution in the previous section gives the exact reconstrcution of f with overwhelming probability. The probability will exceeds $1-\delta$ if

$$m \geq C \times \mu^2(\Phi, \Psi) \times S \times \log(n/\delta)$$

We will skip the proof here. For details, please refer to the paper I mentioned. Here, I interpret the inequality in three points:

First, the coherence between two bases are very important. To find a relative incoherent basis with respect to any fixed basis, a random matrix ϕ should be selected. However, due to the fact that machine generated random number is not really random, the quality of randomness will determine the reconstruction quality later on.

Second, when the coherence $\mu(\Phi, \Psi)$ is close to 1, then on the order of $S \log n$ smaples suffice to reconstrut instead of n samples in normal sensing. The data storage space or transmitting bandwidth save is $S\frac{1}{n}\log n$.

Finally, the only priori we know is f is sparse in domain Ψ. We don't assume any other knowledge about the exact number of S, nor their locations, nor their amplitude. We simply run L1 minimization algorithm to find a sparse solution. And if the signal is truely sparse enough, the exact reconstruction occurs.

Denoising problem:

In the previous sections, we examined the reconstruction without noise. However, in real world, every kind of measurement or sensing introduces noise. Hence, will consider the previous problem in a more generous form. Consider $f = \Psi x$ and f is S-sparse in domain Ψ. We get the random sensing of the signal f in form of $y = R\Phi f$ where R is a m-by-n random matrix that extract data randomly. Then without noise, we can write $y = Ax$, where $A = R\Phi\Psi$. With noise, we can write $y = Ax + z$, where $A = R\Phi\Psi$ and z is noise. If R is identity matrix, namely, we take the exact measurment, then A is isometric and is not hard to prove that

$$\|x\|_{l_2} = \|Ax\|_{l_2}$$

However, if R is not identity, A is not rigorous isometric and we need to define restricted isometry property (RIP) of A.

For each integer S=1,2,..., define the isometry constant δ_s of matrix A as the smallest number such that:

$$(1-\delta_s)\|x\|_{l_2}^2 \leq \|Ax\|_{l_2}^2 \leq (1+\delta_s)\|x\|_{l_2}^2$$

holds for all the S-sparse vectors x. We will loosely say that a matrix A obeys RIP of order S if δ_s is not too close to one. With RIP, A can approximately preserves the Euclidean length of S-sparse vector x. In turn, this implies that x cannot be in the null space of A. Otherwise, our reconstruction of x is impossible.

Then, how does RIP relate to our reconstruction problem? Imagine that we get a measurment result y which in truth is equal to Ax_1. Then we find another root x_2 also satisfy $y = Ax_2$. Suppose δ_{2s} is also sufficiently less than one. Then we will have:

$$(1-\delta_{2s})\|x_1 - x_2\|_{l_2}^2 \leq \|Ax_1 - Ax_2\|_{l_2}^2 \leq (1+\delta_{2s})\|x_1 - x_2\|_{l_2}^2$$

This means, we can guarantee x_2 is very close to the underlying truth x_1. Even with noise presented, their different is only on the order of noise. We will formally write the theorem as below.

Theorem2:

If RIP of A holds, then the solution x_* to the follwing problem (basis pursuit):

$$Minimize \ \|\widetilde{x}\|_1, \ subject \ to \ A\widetilde{x} = \overrightarrow{y}$$
gives a very close approximation of the underlying truth x. If $\delta_{2s} < \sqrt{2} - 1$, then the solution x_* obeys
$$\|x^* - x\|_{l_2} \leq C_0 \ \| \ x - x_S \ \|_{l_1} / \sqrt{s} \ and$$

$$\|x^* - x\|_{l_1} \leq C_0 \ \| \ x - x_S \|_{l_1}$$
for some constant C_0. When x is actually S-sparse, x is equal to x_s and the solution x_* is equal to x. Hence, we get the exact construction. Please note that, compare to theorem 1, theorem 2 involves no probability. It is a deterministic theorem.

With noise presented, we introduce the third theorem:

Theorem 3:

If RIP of A holds, then the solution x_* to the follwing problem (basis pursuit denoising):
$$Minimize\|\widetilde{x}\|_1, \ subject \ \| \ A\widetilde{x} - \overrightarrow{y} \ \|_{l_2} \leq \epsilon$$
gives a very close approximation of the underlying truth x. Here, ϵ bounds the noise in the sensing of data. If
$$\delta_{2s} < \sqrt{2} - 1$$
then the solution x_* obeys
$$\|x^* - x\|_{l_2} \leq C_0 \ \| \ x - x_S \ \|_{l_1} / \sqrt{s} + C_1 \times \epsilon$$
for some constant C_0 and C_1. When x is actually S-sparse, x is equal to x_s and the solution x_* is equal to x. Hence, we get the exact reconstruction.

All the theorms involves long mathematical proof and we omit them for the sake of readability. If you are really interested in the provment of those inequality, please refer to the paper I mentioned above.

RIP Property:

You may feel strange that theorem 1 is probabilistic and theorem 2 and 3 is deterministic. So what happen from theorem 1 to theorem 2 and 3? The answer is RIP (restricted isometric property). Actually, we don't have an algorithm to generate an RIP matrix A. We only have some sensing processes to generate A obeys RIP with overwhelming probability. Some of them are mentioned in the paper An Introduction To Compressive Sampling (http://authors.library.caltech.edu/10092/1/CANieeespm08.pdf). Here, we will use the one that is most convenient to implment in real cicuits:

Form A by sampling i.i.d entries from a symmetric Bernoulli distribution $P(A_{i,j} = \pm 1/\sqrt{m}) = 1/2$

With overwhelming probability, A obeys the RIP (the condition on which theorem 2 and 3 are valid) provided that $m \geq C \times S \log(n/S)$. The probability of sampling a matrix A not obeying RIP when this inequality holds is exponentially small in m.

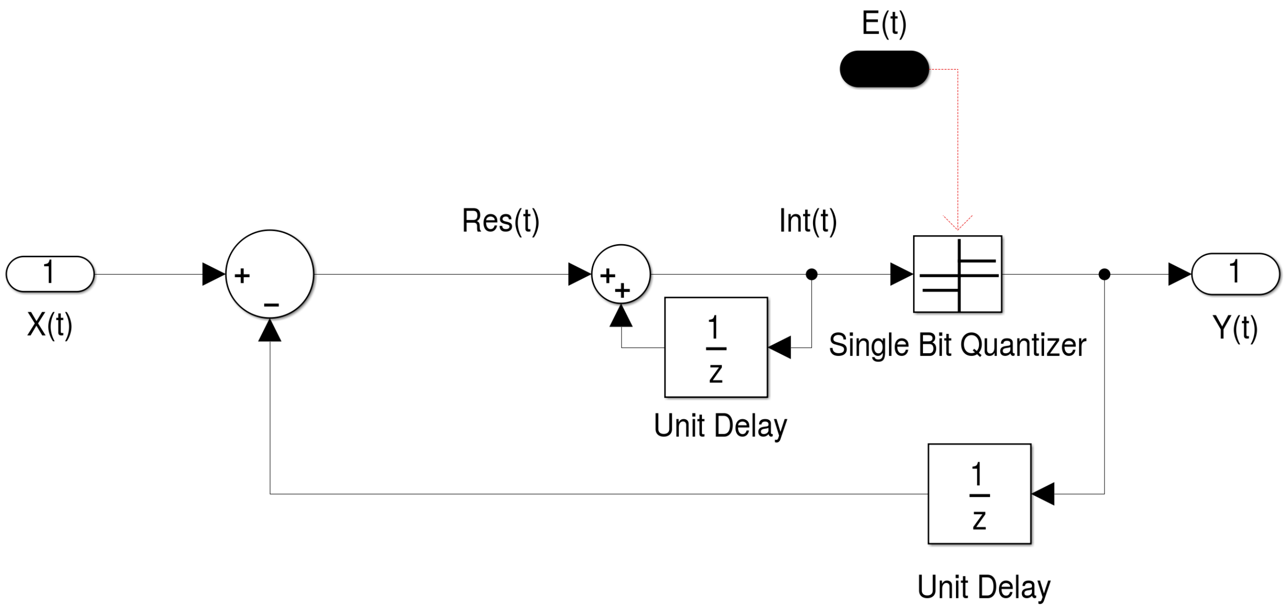
Finally,if Ψ is an arbitrary orthobasis and Φ is an m-by-n measurement matrix sampled randomly from a symmetric Bernoulli distribution, then $A = \Phi\Psi$ obeys RIP as well. The property is similar to the A directly sampled from a symmetric Bernoulli distribution. Note that here Φ is universal and does not any priori knowledge about the sparse orthobasis Ψ.

Compressed Sensing in Sigma Delta Analog to Digital Converter (ADC)

We have covered all the mathematical fundamentals to understand compressed sensing. We know that when A obeys RIP, compressed sensing is feasible and the reconstruction algorithm is available either through BP or BPDN problem. Now, let us take a look on how to integrate this process into Sigma Delta ADC.

The basic concepts of sigma delta ADC can be found in this book Analog Integrated Circuit Design (https://www.amazon.com/Analog-Integrated-Circuit-Design-David/dp/0471144487/). Now let us analyze circuits model and explain how it works.

Below is an traditional sigma delta ADC:



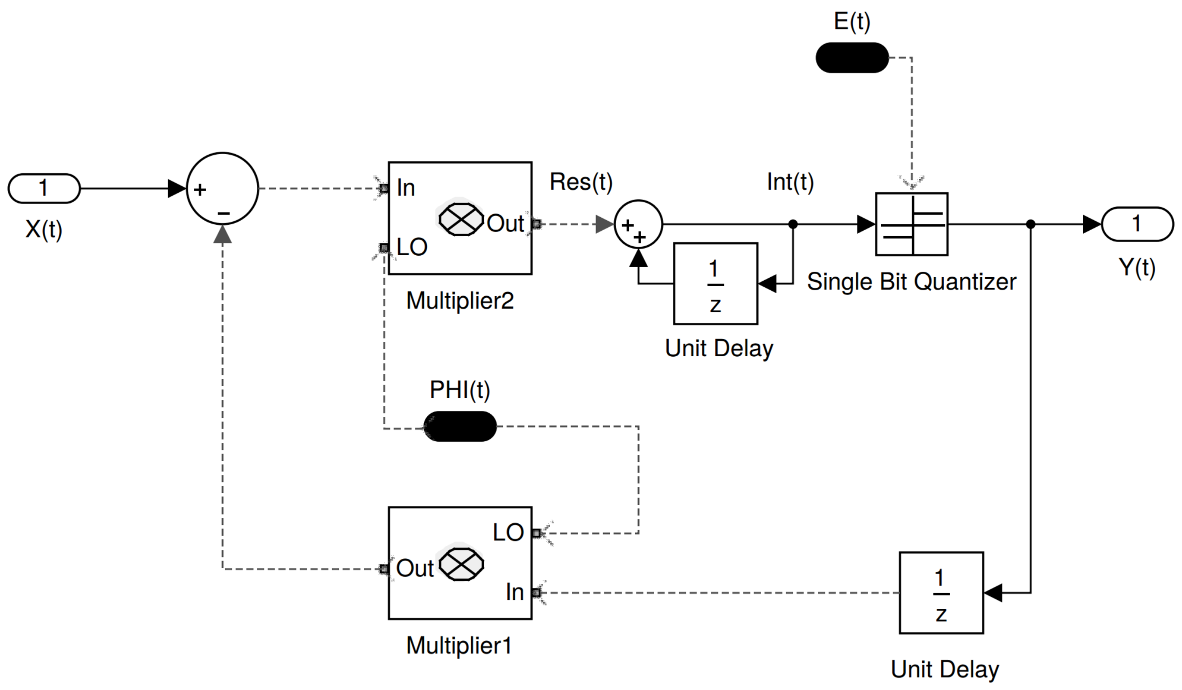
$X(t)$ stands for input signal, $Y(t)$ stands for ADC digital output, $Res(t)$ stands for the residual value after the first subtractor, $E(t)$ stands for the quantization noise introduced by the single bit quantizer and $Int(t)$ stands for integrator output. From this graph we can get three equations:

$$\begin{cases} Y(t) = Int(t) + E(t) \\ Int(t) = Res(t) + Int(t-1) \\ Res(t) = X(t) - Y(t-1) \end{cases}$$

It is not hard to solve it and get

$$Y(t) = X(t) + E(t) - E(t-1)$$

Here we get the behaviour of the traditional sigma delta ADC: since quantization noise will be deducted in the long run (more details will be shown later on), $Y(t)$ is approximately eqaul to input data $X(t)$, namely, convert analog input signal $X(t)$ to digital signal $Y(t)$. This is achieved by adding only shapped quantization noise $E(t) - E(t-1)$. Below is the compressed sensing sigma delta ADC structure used in this project:



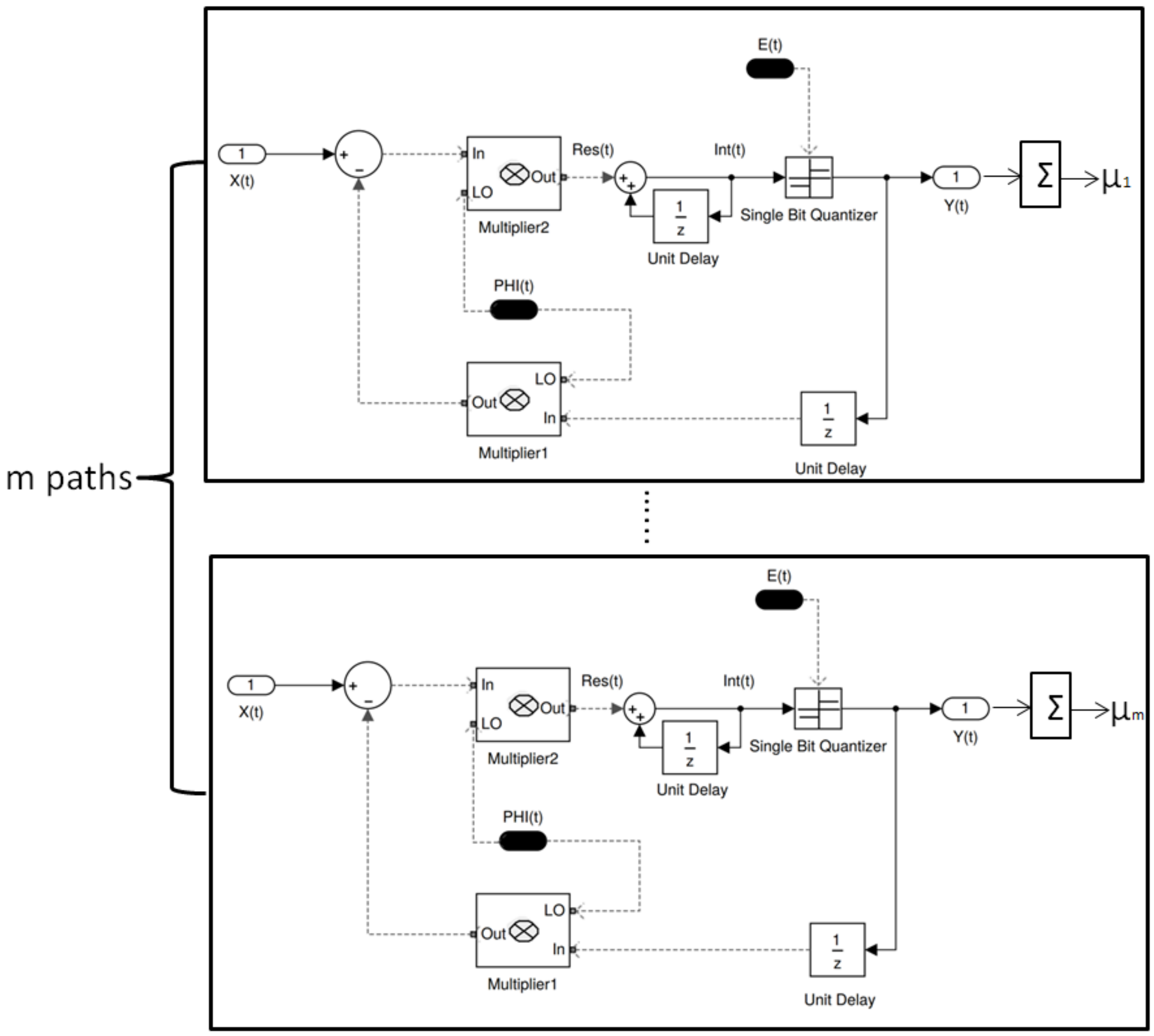
The symbology is the same as the traditional one above except for two multipliers and Φ. As we discussed in the theoretical part, Φ is +/- 1 sampled from Bernoulli distribution. In the real circuits, this part will be a common random number generator. With the multipliers and Φ, the equations for sigma delta ADC changes to:

$$\begin{cases} Y(t) = Int(t) + E(t) \\ Int(t) = Res(t) + Int(t-1) \\ Res(t) = \Phi(t)(X(t) - \Phi(t)Y(t-1)) \end{cases}$$

Please note that Φ is either +1 or -1 so Φ^2 is always 1. Then, it is not hard to solve the equations and get

$$Y(t) = \Phi(t)X(t) + E(t) - E(t-1)$$

Compare to the traditional sigma delta ADC, the new one still has the noise shaping capability. But now it is doing compressed sensing. Let us consider the setup below. For each of the sigma delta ADC, we put an adder after Y(t) to accumulate Y(t), namely $\mu = \sum_{t=1}^{n} Y(t)$, where n is the total sampling point of input signal $X(t)$. Then we deploy m such ADCs in parallel:



For each path of ADC, use the equatio we just get we can derive that:

$$\vec{\mu} = \begin{bmatrix} \mu_1 \\ \dots \\ \mu_m \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^{n} Y_1(t) \\ \dots \\ \sum_{t=1}^{n} Y_m(t) \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^{n} \Phi_1(t)X(t) \\ \dots \\ \sum_{t=1}^{n} \Phi_m(t)X(t) \end{bmatrix} + \begin{bmatrix} E_1(n) - E_1(1) \\ \dots \\ E_m(n) - E_m(1) \end{bmatrix} = \Phi X + \sigma$$

Here, we treat each path's accumulated output μ_i as one element of the output vector $\vec{\mu}$. Each path has its own random number generator so

$$\Phi = \begin{bmatrix} \Phi_1(1) & \dots & \Phi_1(n) \\ \dots & & \dots \\ \Phi_m(1) & \dots & \Phi_m(n) \end{bmatrix}$$

is a random matrix obeying RIP property. $X(t)$ is the input signal applied to all the m paths and is sampled at time from 1 to n.

$$\sigma = \begin{bmatrix} E_1(n) - E_1(1) \\ \dots \\ E_m(n) - E_m(1) \end{bmatrix}$$

is the accumulated shaped quantization noise. As stated before, shaping noise around DC is a important behavior of sigma delta ADC. When quantization noise is around DC, noise value should approach its average in the long run and therefore $E_i(n) \approx E_i(1)$. With a low path digital filter, we can almost eliminate this shaped quantization noise.

By grouping the m paths' accumulated output together, we get an interesting result:

$$\vec{\mu} = \Phi X + \sigma$$

Note that this is exactly the basis pursuit denoising (BPDN) problem as we stated at the beginning. With accumulated output $\vec{\mu}$ and random matrix Φ, we can use either SPGL1 or OMP or CoSaMP to reconstruct the sparse input signal $X(t)$. Note that $X(t)$ has a total length of n (sample points) and $\vec{\mu}$ has only m elements. Therefore, we compressed a sparse signal from length n to m. As stated in the theoretical part, as long as m is greater than the order of magnitude of $S \log n$, the reconstruction is feasible. By transmitting m symbols instead of n symbols, we could save a lot of wireless bandwidth resourses.

In the real world, most nature signal is not sparse in time domain. However, they are sparse in frequency domain. For example, a pure sine wave $X(t) = sin(2\pi f \cdot t)$ is infinitely long in time domain, but is only a spike at f in frequency domain. To really compressed sensing the nature signal, the time domain signal will be multiplied by an FFT matrix (https://en.wikipedia.org/wiki/Fast_Fourier_transform) to convert to frequency domain. Then the frequency-sparse signal will be compressed and reconstructed. Finally, multiply the reconstruction with the inverse of FFT matrix to get the original signal in time domain. Sigma Delta ADC naturally did the FFT conversion and we only need to do the inverse FFT conversion in MATLAB.