IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 11, 2018, accepted December 4, 2018, date of publication December 20, 2018,
date of current version January 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2888927

Frequency Hopping and Parallel Driving With Random Delay Especially Suitable for the Charger Noise Problem in Mutual-Capacitive Touch Applications

SHIH-LUN HUANG, (Member, IEEE), SHENG-YI HUNG, AND CHUNG-PING CHEN
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
Corresponding author: Shih-Lun Huang (slhuang123@gmail.com)

ABSTRACT Charger noise can cause inaccurate touch points and fake touch points to appear; this can cause a device to behave incorrectly. The intensity of charger noise could be much larger than the intensities of the original touch signals. Furthermore, the frequency of charger noise varies for each different charger. Therefore, industry experts have identified charger noise as the most difficult problem in capacitive touch applications. The demand for a solution to this problem has become crucial for the mobile market. In this paper, we prove that a particular combination of frequency hopping and repeated integration is an effective method to handle this problem. In addition, we propose an efficient discrete Fourier transform-based algorithm to select an effective sensing frequency. We propose parallel driving with random delay to enhance the signal-to-noise ratio (SNR). We show an efficient hardware–software co-design that facilitates the application of our methods in touch ICs. The experimental results show that our methods can increase SNR by over 45 dB and can find an effective sensing frequency fast and dynamically.

INDEX TERMS Charger noise, parallel driving, frequency hopping, repeated integration, mutual-capacitive touch.

I. INTRODUCTION

Capacitive sensors are extensively applied in many consumer electronics because they make user interfaces become more intuitive, more convenient, and more interactive. These types of sensors could be used as electrical buttons and switches to replace the traditional mechanical ones. Besides, they also provide the multi-touch functionality, which enables rich ways to control devices, so a large number of smartphones and tablets have appeared in the past ten years.

Capacitive touch ICs suffer from various noises, such as display noise, charger noise, and inherent noise. Inherent noise refers to random noise signals that can be significantly reduced through proper circuit design since this type of noise is due to the fundamental properties of circuits. However, charger noise and display noise are interference noises that physically couple into the sensor through the battery charger and the display components. They cannot be reduced just by circuit design skills because they are irrelevant to the essentials of the circuits. Fortunately, display noise is a predictable characteristic of the display, so it can be resolved during the touch solution development. That is to say, display noise does not happen again once it is reduced during the design phase of the product. However, charger noise is unpredictable, depending on the charger to be used.

Noise reduction plays an important role in the multi-touch algorithm because it affects the accuracy of touch points. For charger noise, inaccurate touch points and fake touch points may appear so that a device has a wrong behavior. The makers of chargers often simplify their designs to reduce their manufacturing costs, which results in large noises accordingly. The intensity of charger noises can exceed touch signals over 10 times [2]. Besides, the frequency of charger noise varies for each different charger. Moreover, charger noise only appears when a touch event happens. So it is more difficult to detect charger noise. Therefore, industry identifies charger noise as the most difficult problem in capacitive touch applications [2]. Mohamed *et al.* [3] mentioned that their noise analysis is regularly done by measuring noise spectrum at all frequencies using Fast Fourier Transform (FFT) while no driving signal is applied. Because their FFT method

is computationally expensive, it has a high hardware cost and a large power consumption. To find out charger noise efficiently, we propose a real-time charger noise analysis algorithm, which is a low-cost implementation and can be easily integrated into the current devices without a lot of modifications.

There are a few papers proposed to resolve the noise problem [4]–[10]. They all presented that repeated integration is an effective method for this problem. However, the aforementioned works focused on the analog front-end implementation. They need large capacitors for charge accumulation, which is a heavy burden on the competitive hardware cost. In addition, they cannot detect charger noise quickly and do a real-time manipulation. Moreover, they did not propose the formal charger noise model to prove whether or not the method is effective under any condition. Unfortunately, as the noise frequency is close to the sensing frequency, the method may become ineffective and cannot handle the noise immediately. We present the mathematical model of charger noise and prove that repeated integration is effective under certain conditions.

To make repeated integration effective, Klein [2] indicated that the frequency hopping is a method to find an effective sensing frequency. However, he did not explain when and how to execute frequency hopping. Hotelling *et al.* [11] used three sensing frequencies to get three map values, and performed median filtering on the three maps. There are two drawbacks of this method. On the one hand, it reduces its frame rate by up to 2/3. On the other hand, it is ineffective if it meets two or more noisy frequencies at the same time. To keep the original frame rate and to avoid noisy frequencies, we propose an efficient algorithm to detect the charger noise frequency and to find a good sensing frequency fast and dynamically.

There are two types of capacitance sensing methods: mutual capacitance sensing and self-capacitive sensing. We focus on mutual capacitance sensing in this paper. The common diagram of the mutual capacitance sensing is shown in Fig. 1. The X-Y crossings of the patterns form the mutual capacitors. When fingers touch the panel, the charges at the sensing nodes change. A capacitive touch IC converts the quantities of charges to digital values, which are also called sensing values.

In mutual capacitance sensing, a time-multiplexed or time-interleaved acquisition is a conventional method to capture a variation of one mutual capacitor at one time slot. Recently, the parallel sensing method is widely used to increase the frame rate and handle the larger touch screens. This method mainly repeats several front-end analog circuits to reduce the total capture time of a whole sensing map. Nevertheless, the number of front-end analog circuits needs to be carefully estimated to prevent an unnecessary waste. On the other hand, the parallel driving method [10], [12]–[15] is also extensively applied in industry to get a better signal-to-noise ratio (SNR). As the number of parallel driving channels is larger, the SNR is better. Unfortunately, the parallel driving method cannot

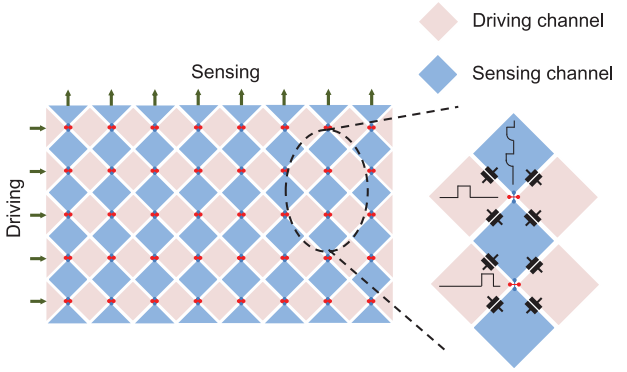

FIGURE 1. The common diagram of the mutual capacitance sensing.

basically improve the frame rate because the parallel driving sequence needs to be long enough to obtain the sufficient information for decoding the mixed signals. After that, we can get the touch signals of those nodes. Note that the parallel driving method cannot be applied to self-capacitive sensing.

The driving sequence for the parallel driving channels is usually generated based on orthogonal codes, such as Walsh-Hadamard (W-H) codes. As the number of parallel driving channels grows, the received signals at sensing channels may be larger than the limited supply voltage level. To deal with this problem and let the number of parallel driving channels grow as possible, Park *et al.* [16] presented the application of various weighing matrices to lower the received signals. Moreover, Ma *et al.* [17] presented the application of the differential Manchester code to detect the variation between the un-touch and touch condition without the problem of the large received signals. This parallel driving method is suitable for combating random noise because the samples of a random noise are regarded as uncorrelated random variables. However, it is weak for charger noise, which is a periodic noise. Hence, we propose the random delay technique to make the samples of a periodic noise also regarded as uncorrelated random variables.

A conventional multi-touch algorithm framework is shown in Fig. 2 [18]. There are four main steps to obtain coordinates of touch points and their identifications (IDs). The first two steps control the operations of hardware, and use the algorithms of software to analyze noise and to reduce noise. The two steps may be executed iteratively to obtain a noiseless data map. After getting a noiseless data map, accurate touch points could be calculated, presented in [19]–[23]. After getting the coordinates of the touch points, the tracking algorithm is used to identify and track the corresponding touch points in two sequential frames, presented in [24]–[29]. The third and fourth steps are pure software-related topics. However, the first and second steps need an efficient hardware-software co-design to achieve a performance and cost optimization. Although most works presented the advanced hardware solutions to achieve better performance, such as SNR and speed, than the previous ones, they did not consider the manufacturing cost and the other side effects.

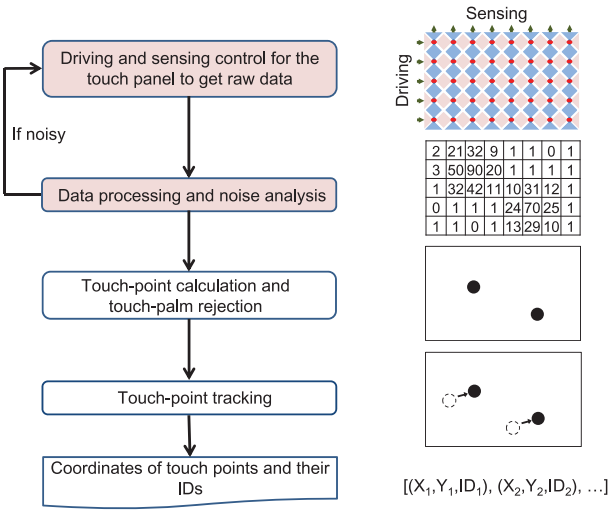

FIGURE 2. The multi-touch algorithm framework [18].

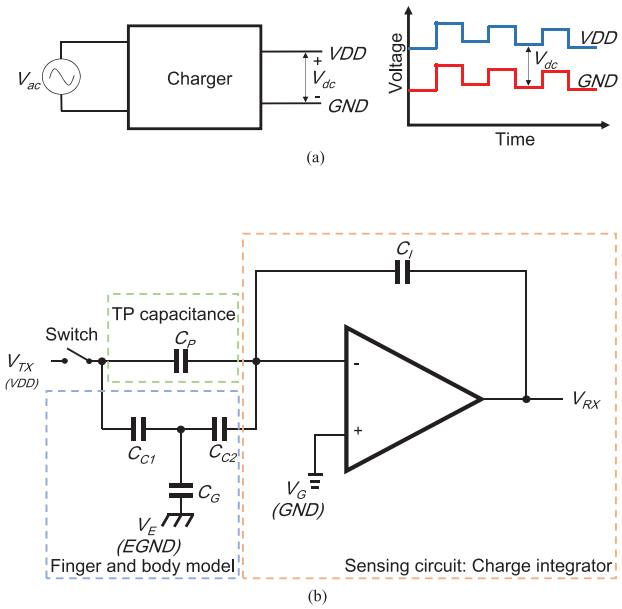

FIGURE 3. The model of a basic touch sensing architecture.

In this paper, we show that our methods assist in analyzing noise and selecting an appropriate sensing frequency dynamically to improve the SNR and keep the speed, while they do not need to change the original hardware architecture and merely slightly increase the hardware cost for analysis.

To sum up, our work has the following distinguished features and theoretical findings:

- The mathematical model of charger noise is presented and it is proved that repeated integration is effective under certain conditions.
- To keep the original frame rate and to avoid noisy frequencies, an efficient algorithm is proposed to detect the charger noise frequency and to find a good sensing frequency fast and dynamically.
- The random delay technique is proposed to make the samples of a periodic noise regarded as uncorrelated random variables, which makes the parallel driving method effective for charger noise.
- Our methods do not need to change the original hardware architecture and merely slightly increase the hardware cost for analysis.

Experimental results show that our algorithm takes acceptable run-time on a 50 MHz ARM-M0 processor based on the user experience in [30]. In addition, the results also show that our methods can increase SNR by over 45 dB, compared to that without our charger noise reduction algorithm.

The rest of this paper is organized as follows. Section II gives an introduction to the circuit model affected by charger noise, presents the mathematical model of charger noise, and formulates the charger noise reduction problem. Section III presents 1) that the combination of frequency hopping and repeated integration is an effective method, 2) an efficient DFT-based algorithm to select a good sensing frequency, 3) parallel driving with random delay to enhance SNR, and 4) an efficient hardware-software co-design about adopting these methods. Experimental results are reported in Section IV, and discussion is given in Section V. Conclusions are given in Section VI.

II. MODELING OF CHARGER NOISE

In this section, we introduce the cause of charger noise, give its mathematical model and formulate the charger noise reduction problem.

An AC/DC charger is a device that converts AC into DC and then supplies DC to a circuit via external power and ground pins as shown in Fig. 3(a). The voltage of DC, which is the difference in electric potential between a power pin and a ground pin, is generally constant and stable. Although the electric potentials of the power pin and the ground pin may fluctuate, the circuit can work properly as long as the potential difference is desired. However, for touch ICs, when fingers touch on a touch screen, the fluctuation phenomena would emerge because fingers have another reference point (the earth's ground). For a clear explanation, we illustrate the charger noise caused by the fluctuation with the model of a basic mutual-capacitive touch sensing architecture as shown in Fig. 3(b), where V_{TX} is the driving signal used to charge the capacitor C_P of the touch panel, V_{RX} is the sensing value used to detect the capacitance changes influenced by the finger-coupled capacitors C_{C1} and C_{C2}, C_G is the earth capacitor, V_E is the earth's ground, and V_G is the system's ground. When the system plugs in a charger of poor quality, V_G will fluctuate. V_{TX} has the same fluctuation with V_G because V_G is its reference ground, but V_E is constant. Therefore, charger noise is a kind of finger-coupled noise because it only emerges when fingers touch. Its intensity is in proportion to the touch signal because the larger touch signal means that C_{C1} and C_{C2} are larger. The noise is difficult to deal with because it has very high intensity. Besides, the form of charger noise is the combination of a fundamental frequency

and its harmonics [2]. So, its function is described as follows:

$$N(t) = \sum_{n=1}^{K} A_n \cos(2\pi f_d n t + \varphi_n), \quad K = \lfloor 1000/f_d \rfloor, \quad (1)$$

where t is time, φ_n is the phase of the n-th harmonic, A_n is the amplitude of the n-th harmonic, and f_d is the fundamental frequency (KHz). A_n is close to 0 when n is larger than $1000/f_d$, benefiting by the low-pass filter formed from the resistive and capacitive (RC) loading of a touch panel [3], [10].

The sampling function of a touch controller is derived by:

$$S[m] = T + N[m] = T + \sum_{n=1}^{K} A_n \cos(2\pi f_d n m/f_s + \varphi_n), \quad (2)$$

where m is the m-th sampling, f_s is the sensing frequency, and T is the original noiseless signal. Note that T is considered as a constant in a short period.

After our modeling of charger noise, our objective is to minimize the interference from the charger noise for extracting the original touch signal. Because repeated integration is a well-known method to suppress a periodic noise, the objective is further formulated as follows:

Charger Noise Reduction Problem: Determine the sensing frequency f_s and the repeating number r so that the charger noise is minimized and the final result I converges to the original touch signal as follows:

$$\frac{1}{r}\sum_{m=1}^{r} N[m] \approx 0. \quad (3)$$

$$\Longrightarrow I = \frac{1}{r}\sum_{m=1}^{r} S[m] \approx T. \quad (4)$$

III. THEOREMS AND AN EFFECTIVE AND EFFICIENT ALGORITHM

In this section, we 1) prove that the combination of frequency hopping and repeated integration is an effective method, 2) propose an efficient DFT-based algorithm to select a good sensing frequency, 3) propose parallel driving with random delay to enhance SNR, 4) summarize the whole algorithm flow, and 5) propose an efficient hardware-software co-design about adopting these methods.

A. THEOREMS FOR THE CHARGER NOISE REDUCTION PROBLEM

First, the charger noise reduction problem (3) can be transformed to (5) because of the commutative law for addition. The solution to (6) is also a solution to (5). Therefore, we can solve (6) to get a solution to the charger noise reduction problem.

$$\sum_{n=1}^{K}\left(\frac{1}{r}\sum_{m=1}^{r} A_n \cos(2\pi f_d n m/f_s + \varphi_n)\right) \approx 0 \quad (5)$$

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d k m/f_s + \varphi_k) \approx 0, \quad k \in \{1, 2, \ldots, K\} \quad (6)$$

The following theorems are used to get the solutions for (6):

Theorem 1: (Repeated Integration Theorem) If 1) $r/2k$ is a positive integer, and 2) $f_d \times r/f_s$ equals a positive odd integer, then

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d k m/f_s + \varphi_k) = 0, \quad k \in \{1, 2, \ldots, K\}. \quad (7)$$

Proof 1: Because $f_d/f_s = c/r$, where c is a positive odd integer,

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d k m/f_s + \varphi_k) = \sum_{m=1}^{r} A_k \cos(2\pi c k m/r + \varphi_k).$$

Then, because of the condition 1), $r/2k$ is a positive integer, we divide the summation into $r/2k$ groups, and the above expression equals

$$\begin{aligned}
&(A_k \cos(2\pi c k \frac{1}{r} + \varphi_k) + A_k \cos(2\pi c k \frac{r/2k+1}{r} + \varphi_k)) \\
&+ (A_k \cos(2\pi c k \frac{2}{r} + \varphi_k) + A_k \cos(2\pi c k \frac{r/2k+2}{r} + \varphi_k)) \\
&+ (A_k \cos(2\pi c k \frac{3}{r} + \varphi_k) + A_k \cos(2\pi c k \frac{r/2k+3}{r} + \varphi_k)) \\
&+ \ldots \\
&+ (A_k \cos(2\pi c k \frac{r/2k}{r} + \varphi_k) + A_k \cos(2\pi c k \frac{r/k}{r} + \varphi_k))
\end{aligned}$$

Further, we apply the sum-to-product formulas, and the above expression equals:

$$\begin{aligned}
&2A_k \cos(\frac{\pi c}{2}) \cos(\pi c k \frac{r/2k+2}{r} + \varphi_k) \\
&+ 2A_k \cos(\frac{\pi c}{2}) \cos(\pi c k \frac{r/2k+4}{r} + \varphi_k) \\
&+ 2A_k \cos(\frac{\pi c}{2}) \cos(\pi c k \frac{r/2k+6}{r} + \varphi_k) \\
&+ \ldots \\
&+ 2A_k \cos(\frac{\pi c}{2}) \cos(\pi c k \frac{3r/2k}{r} + \varphi_k)
\end{aligned}$$

Finally, since c is a positive odd integer, $\cos(\frac{\pi c}{2})$ is zero. Hence, the expression equals zero. ■

Theorem 1 states that if we can obtain the noise frequency f_d, we can find an appropriate set of f_s and r by this theorem.

Property 1: If 1) $r/2k$ is a positive integer, and 2) $f_d \times r/f_s$ equals a positive odd integer, then f_s is not equivalent to f_d.

Theorem 2: (Repeated Integration Theorem for Aliasing 1) If 1) $r/2k$ is a positive integer, 2) f_{mirror} is a mirrored frequency of the frequency f_d with respect to the sensing frequency f_s ($f_{mirror} = f_d - f_s$), and 3) $f_{mirror} \times r/f_s$ equals a positive odd integer, then (7) still holds.

Proof 2: Because f_{mirror} is a mirrored frequency of the frequency f_d with respect to the sensing frequency f_s, $f_{mirror} = f_s - f_d$.

Therefore,

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi (f_s - f_{mirror})km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi km - 2\pi f_{mirror} km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi f_{mirror} km/f_s - \varphi_k).$$

Because $f_{mirror}/f_s = c/r$, where c is a positive odd integer, the rest of this proof is similar to Proof 8. So, the above expression also equals zero. ■

Theorem 3: (Repeated Integration Theorem for Aliasing 2) If 1) $r/2k$ is a positive integer, 2) f_{folded} is a folded frequency of the frequency f_d with respect to the sensing frequency f_s ($f_{folded} = f_d - y \times f_s$, $y \in$ integer), and 3) $f_{folded} \times r/f_s$ equals a positive odd integer, then (7) still holds.

Proof 3: Because f_{folded} is a folded frequency of the frequency f_d with respect to the sensing frequency f_s, $f_{folded} = f_d - y \times f_s$, $y \in$ integer.

Therefore,

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi (f_{folded} + y \times f_s)km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi f_{folded} km/f_s + 2\pi ykm + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi f_{folded} km/f_s + \varphi_k).$$

Because $f_{folded}/f_s = c/r$, where c is a positive odd integer, the rest of this proof is similar to Proof 8. So, the above expression also equals zero. ■

Theorem 4: (Repeated Integration Theorem for Aliasing 3) If 1) $r/2k$ is a positive integer, 2) $f_{mirror,folded}$ is a mirrored and folded frequency of the frequency f_d with respect to the sensing frequency f_s ($f_{mirror,folded} = y \times f_s - f_d$, $y \in$ integer, $y \geq 2$), and 3) $f_{mirror,folded} \times r/f_s$ equals a positive odd integer, then (7) still holds.

Proof 4: Because $f_{mirror,folded}$ is a mirrored and folded frequency of the frequency f_d with respect to the sensing frequency f_s, $f_{mirror,folded} = y \times f_s - f_d$, $y \in$ integer, $y \geq 2$.

Therefore,

$$\sum_{m=1}^{r} A_k \cos(2\pi f_d km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi (y \times f_s - f_{mirror,folded})km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi ykm - 2\pi f_{mirror,folded} km/f_s + \varphi_k)$$
$$= \sum_{m=1}^{r} A_k \cos(2\pi f_{mirror,folded} km/f_s - \varphi_k).$$

Because $f_{mirror,folded}/f_s = c/r$, where c is a positive odd integer, the rest of this proof is similar to Proof 8. So, the above expression also equals zero. ■

B. DFT FOR FINDING WHERE THE NOISES ARE

As charger noise always changes with different chargers, we need to design a real-time monitor to observe the behavior of noise and to find the frequency of noise. Once we obtain the frequency of noise, we can choose an appropriate sensing frequency to eliminate noise. First, we choose a default sensing frequency f_s to capture a number r of signals. Then, based on DFT, we calculate intensities of frequency components with the following function:

$$X_k = |\sum_{n=0}^{r-1} S[n] \times e^{-j2\pi kn/r}|, \quad 0 \leq k \leq r/2, \ k \in \mathbb{Z}, \qquad (8)$$

where X_k is the intensity of the frequency bin k whose frequency range is between $f_s \times k/r$ and $f_s \times (k+1)/r$, and $\mathbb{Z}$ is the set of integers. When a signal, a noise, or its alias falls within a frequency range, the intensity of the corresponding frequency bin will show. The touch signal appears in the first bin because it is a DC signal. If the repeating number r is larger, the bandwidth of a frequency bin is smaller, and the frequency of noise can be identified more accurately. However, the large repeating number r sacrifices the frame rate of a touch system or increases the hardware cost. Therefore, we can only use a limited repeating number r to keep the frame rate. To find a good sensing frequency, we select a frequency that meets Theorems 1, 2, 3, or 4. We usually select a higher frequency than a lower frequency because a lower sensing frequency sacrifices the frame rate. Note that when the sensing frequency approaches the frequency of noise, (4) would not equal T and the noise signal appears at the first bin like the touch signal. As a result, it is difficult to extract touch signals from sensing values. To avoid colliding with the frequency of noise, we make the sensing frequency automatically have a slight change within a range. The change can help the frequency analyzer to separate the noise frequency from the first and last bins. Our DFT analyzer is implemented by the radix-2 Cooley-Tukey algorithm [31], which is not only fast but also easy to implement in a low-cost MCU because it has a small code size in a recursive form. On the contrary, the FFT method [3] is implemented by extra hardware, which is a hard burden to a low-cost design. Besides, the execution time of the radix-2 Cooley-Tukey algorithm only requires $O(r \log r)$.

C. PARALLEL DRIVING AND RANDOM DELAY

Parallel driving almost exists in every mutual-capacitive touch device because it is useful for combating

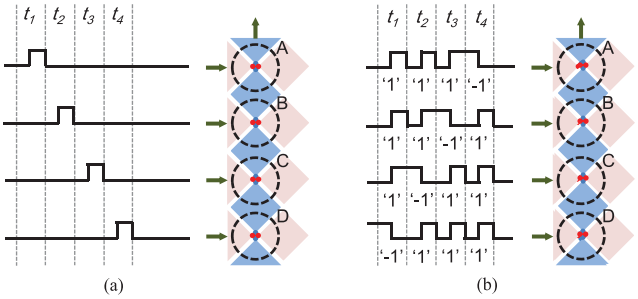

FIGURE 4. Driving methods for the mutual-capacitive touch sensors. (a) The driver sends one driving signal in one time slot, which is called a sequential driving. (b) The driver sends multiple driving signals in one time slot, which is called a parallel driving.

$$\begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} T_A \\ T_B \\ T_C \\ T_D \end{bmatrix} = \begin{bmatrix} T_A \\ T_B \\ T_C \\ T_D \end{bmatrix}$$

(a)

$$\begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} T_A \\ T_B \\ T_C \\ T_D \end{bmatrix} = \begin{bmatrix} T_A + T_B + T_C - T_D \\ T_A + T_B - T_C + T_D \\ T_A - T_B + T_C + T_D \\ -T_A + T_B + T_C + T_D \end{bmatrix}$$

(b)

FIGURE 5. An example of the values captured by a sensor. (a) In a case of sequential driving, the values of the node *A*, *B*, *C*, and *D* are captured at time t_1, t_2, t_3, and t_4, respectively. (b) In a case of parallel driving, the captured values are the combinations of the nodes being driven.

random noise. The conventional parallel driving method is suitable for combating random noise because random noise samples are regarded as uncorrelated random variables. However, the conventional parallel driving method is weak for charger noise because periodic noise may result in the correlated samples which have non-zero covariances. (Refer to the note after the proof of Theorem 5.) So, we apply the random delay technique to represent the samples of a periodic noise as uncorrelated random variables. Therefore, parallel driving with random delay can combat not only random noise but also periodic noise.

Fig. 4 shows the difference between the sequential driving and the parallel driving. See Fig. 4(a) for an example of the sequential driving. The driver sends one driving signal in one time slot, and then the sensor captures the coupling signal from the sensing channel. The coupling signal represents the touch signal of a certain node. The sensing values in Fig. 4(a) are shown in Fig. 5(a), where the values of the node A, B, C, and D are captured at time t_1, t_2, t_3, and t_4, respectively. It can also be regarded as the 4×4 identity matrix multiplies the vector $\mathbf{v}=[T_A\ T_B\ T_C\ T_D]^T$. So, the sequential driving directly obtains the touch signal of each node. See Fig. 4(b) for an example of the parallel driving. The driver sends multiple driving signals in one time slot, and then the sensor captures the coupling signal from the sensing channel. The coupling signal represents the combined touch signal of certain nodes. The driving signals in Fig. 4(b) are [1 1 1 −1], [1 1 −1 1], [1 −1 1 1], and [−1 1 1 1] at time t_1, t_2, t_3, and t_4, respectively, where '1' and '−1' represent positive driving and negative driving, respectively. Note that the sensing period of the conventional parallel driving method is fixed, which represents $t_2 - t_1 = t_3 - t_2 = t_4 - t_3$. Those driving signals form the

$$\begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} T_A \\ T_B \\ T_C \\ T_D \end{bmatrix} + \begin{bmatrix} n(t_1) \\ n(t_2) \\ n(t_3) \\ n(t_4) \end{bmatrix} = \begin{bmatrix} T_A + n(t_1) \\ T_B + n(t_2) \\ T_C + n(t_3) \\ T_D + n(t_4) \end{bmatrix}$$

(a)

$$\begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{matrix}\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} T_A \\ T_B \\ T_C \\ T_D \end{bmatrix} + \begin{bmatrix} n(t_1) \\ n(t_2) \\ n(t_3) \\ n(t_4) \end{bmatrix} = \begin{bmatrix} T_A + T_B + T_C - T_D + n(t_1) \\ T_A + T_B - T_C + T_D + n(t_2) \\ T_A - T_B + T_C + T_D + n(t_3) \\ -T_A + T_B + T_C + T_D + n(t_4) \end{bmatrix}$$

(b)

$$\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} T_A + T_B + T_C - T_D + n(t_1) \\ T_A + T_B - T_C + T_D + n(t_2) \\ T_A - T_B + T_C + T_D + n(t_3) \\ -T_A + T_B + T_C + T_D + n(t_4) \end{bmatrix} = \begin{bmatrix} T_A + \frac{n(t_1) + n(t_2) + n(t_3) - n(t_4)}{4} \\ T_B + \frac{n(t_1) + n(t_2) - n(t_3) + n(t_4)}{4} \\ T_C + \frac{n(t_1) - n(t_2) + n(t_3) + n(t_4)}{4} \\ T_D + \frac{-n(t_1) + n(t_2) + n(t_3) + n(t_4)}{4} \end{bmatrix}$$

(c)

FIGURE 6. An example of the values captured by a sensor with a noise. (a) In a case of sequential driving, the values of the node *A*, *B*, *C*, and *D* are captured with the noise at time t_1, t_2, t_3, and t_4, respectively. (b) In a case of parallel driving, the captured values are the combinations of the nodes being driven plus the noise. (c) The inverse of the matrix is used to decode the combinations, and then the individual values are obtained for the node *A*, *B*, *C*, and *D*.

driving matrix,

$$D_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}. \tag{9}$$

The vector $\mathbf{t}$ are multiplied by the driving matrix to get the sensing values as shown in Fig. 5(b), where $(T_A + T_B + T_C - T_D)$, $(T_A + T_B - T_C + T_D)$, $(T_A - T_B + T_C + T_D)$, and $(-T_A + T_B + T_C + T_D)$ are captured at time t_1, t_2, t_3, and t_4, respectively. Hence, the parallel driving needs a decoder to separate those combinations of touch signals. The decoder for Fig. 5(b) is $(D_{4\times4})^{-1}$ which is the inverse of the driving matrix.

Parallel driving has a better SNR than sequential driving when they have the same total driving and sensing time. If the noise is a random noise, the SNR of parallel driving has a $\sqrt{p}$ times improvement at most, where p is the number of parallel driving channels. For example, the number of the parallel driving channels is 4 in Fig. 4(b), so its SNR is enhanced by $\sqrt{4} = 2$ times compared to the sequential driving case in Fig. 4(a). As shown in Fig. 6(a) and (b), the noises are added for the sequential driving case and the parallel driving case during the sensing periods. As shown in Fig. 6(c), the sensing values are multiplied by the inverse of the driving matrix to obtain the touch signals of those nodes. The noise parts of the touch signals are like average after decoding. If the noise is a random noise, the standard deviation of the noise after decoding is one-half of the standard deviation of the original noise based on the following definition and theorem:

Definition 1: $\mathbf{D}$ is a p-by-p driving matrix. $D[i, j]$ is the driving signal of the driving channel j at time i. $\mathbf{D}$ has the properties:

1) the entries of $\mathbf{D}$ are all 1 or −1.
2) $(1/\sqrt{p}) \cdot \mathbf{D}$ is an orthogonal matrix.

Definition 2: $N_{t_1}, N_{t_2}, \ldots,$ and N_{t_p} are the random variables of a random noise captured at time $t_1, t_2, \ldots,$ and t_p. Those random variables are uncorrelated and equal.

Lemma 1: The standard deviation of N_{t_i} has the property: $\sigma(N_{t_i}) = \sigma(-N_{t_i})$.

Proof:

$$\sigma(N_{t_i}) = \sqrt{E[N_{t_i}^2] - (E[N_{t_i}])^2}.$$

$$\begin{aligned}\sigma(-N_{t_i}) &= \sqrt{E[(-N_{t_i})^2] - (E[-N_{t_i}])^2} \\ &= \sqrt{E[N_{t_i}^2] - (E[-N_{t_i}])^2} \\ &= \sqrt{E[N_{t_i}^2] - (-E[N_{t_i}])^2} \\ &= \sqrt{E[N_{t_i}^2] - (E[N_{t_i}])^2} = \sigma(N_{t_i}).\end{aligned}$$

■

Theorem 5: (Parallel Driving Theorem for Noise Reduction) If 1) $\mathbf{D}$ is used to be the driving matrix and 2) the noise $\mathbf{r}$ is a random noise, then the standard deviation of the noise after decoding, which is done via multiplication by $\mathbf{D}^{-1}$, is $1/\sqrt{p}$ of the standard deviation of the original noise.

Proof 5: Let $\mathbf{v}$ be the vector whose components are the original touch signals of p nodes, 1, 2, ..., and p. Let $\mathbf{n}$ be the vector whose components are the noises captured at time $t_1, t_2, \ldots,$ and t_p. Let $\mathbf{r}$ be the vector whose components are the sensing values at time $t_1, t_2, \ldots,$ and t_p. Then, $\mathbf{r}$ can be described by the following equation:

$$\mathbf{r} = \mathbf{D}\mathbf{v} + \mathbf{n}. \tag{10}$$

By multiplying $\mathbf{r}$ by $\mathbf{D}^{-1}$ to get $\mathbf{v}'$, which is the vector whose components are the decoded touch signals, the following equation is obtained:

$$\mathbf{v}' = \mathbf{D}^{-1}\mathbf{r} = \mathbf{v} + \mathbf{D}^{-1}\mathbf{n} = \mathbf{v} + \mathbf{n}', \tag{11}$$

where $\mathbf{n}'$ is the noise vector after decoding.

From (11), the decoded touch signals also have the decoded noises except the original touch signals. Because the original touch signals are constant, the variance of the decoded touch signals can be expressed by:

$$Var(N_i') = Var(\sum_{j=1}^{p} D^{-1}[i,j] \times N_{t_j}), \tag{12}$$

where N_i' is the random variable of $n'[i]$, and N_{t_j} is the random variable of $n[j]$. Note that $N_{t_1} = N_{t_2} = \ldots = N_{t_p}$ because they are uncorrelated and equal random variables.

Because $(1/\sqrt{p}) \cdot \mathbf{D}$ is an orthogonal matrix, the following equations are obtained:

$$\begin{aligned}((1/\sqrt{p}) \cdot \mathbf{D}^T)((1/\sqrt{p}) \cdot \mathbf{D}) = \mathbf{I} = \mathbf{D}^{-1}\mathbf{D} \\ \Longleftrightarrow (1/p) \cdot \mathbf{D}^T\mathbf{D} = \mathbf{D}^{-1}\mathbf{D} \\ \Longleftrightarrow \mathbf{D}^{-1} = (1/p) \cdot \mathbf{D}^T,\end{aligned} \tag{13}$$

where $\mathbf{I}$ is the identity matrix.

We then replace $\mathbf{D}^{-1}$ in (12) by (13). Thus, we have the following equation:

$$Var(N_i') = Var(\sum_{j=1}^{p}(1/p) \cdot D^{-T}[i,j] \times N_{t_j}) \tag{14}$$

$$= (1/p^2)Var(\sum_{j=1}^{p} \cdot D^{-T}[i,j] \times N_{t_j}). \tag{15}$$

Because of Definition 1, Definition 2, and Lemma 1, the entries of $\mathbf{D}$ are all 1 or -1, we further have the following equation:

$$Var(N_i') = (1/p^2)Var(\sum_{j=1}^{p} N_{t_j}) \tag{16}$$

$$= (1/p^2)(\sum_{j=1}^{p} Var(N_{t_j}) + 2 \sum_{j,k:1<=j<k<=p} Cov(N_{t_j}, N_{t_k})) \tag{17}$$

$$= (1/p^2) \sum_{j=1}^{p} Var(N_{t_j}) \tag{18}$$

$$= (1/p)Var(N_{t_1}) \tag{19}$$

$$= (1/p)Var(N_o), \tag{20}$$

where $Cov(N_{t_j}, N_{t_k})$ denotes the covariance of N_{t_j} and N_{t_k} which is zero because they are uncorrelated, N_o represents the random variable of the sampled noise at any time, and $N_o = N_{t_1} = N_{t_2} = \ldots = N_{t_p}$.

Therefore,

$$\sigma(N_i') = (1/\sqrt{p}) \cdot \sigma(N_o). \tag{21}$$

■

Note that if $N_{t_1}, N_{t_2}, \ldots,$ and N_{t_p} are correlated, implying that their covariances are not zero, (21) becomes:

$$\sigma(N_i') > (1/\sqrt{p}) \cdot \sigma(N_o). \tag{22}$$

Thus, we apply the random delay technique to make each sampling independent and identically distributed (i.i.d.) so that charger noise can be effectively suppressed like random noise. Between two driving signals, a random delay, which is a uniform distribution on [0, P], is inserted, where P is the period of the charger noise. This can guarantee that captured signals are uncorrelated. Therefore, if there are n driving signals, we are required to insert at least $n-1$ random delays into those driving signals to ensure that those captured signals are uncorrelated. When those captured signals are uncorrelated, they can achieve the effectiveness of noise reduction as illustrated in Theorem 5. See Fig. 7 for an example of the parallel driving method with random delay. We use the driving matrix $D_{4\times4}$ to produce the driving patterns, and insert three random delays into those driving signals. The four captured signals are uncorrelated because of these three random delays.

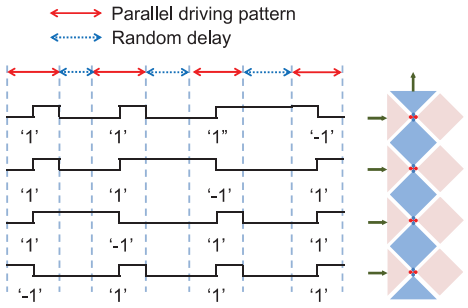

FIGURE 7. An example of the parallel driving method with random delay.

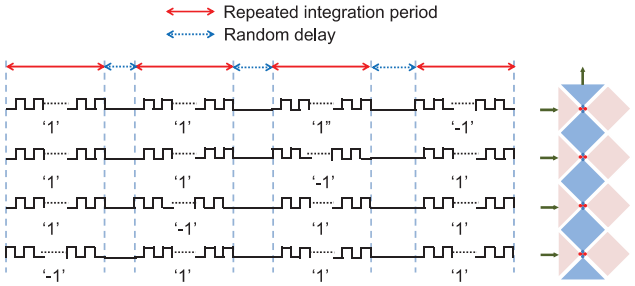

FIGURE 8. An example of the parallel driving method with random delay combined with the repeated integration method.

D. INTEGRATION OF PARALLEL DRIVING AND REPEATED INTEGRATION

Repeated integration is the most effective method to suppress charger noise. However, because of the limited samples, DFT can find the frequency region where noise falls, but cannot find its exact frequency. Therefore, the noise may not be totally suppressed and could be suppressed further. On the other hand, in mutual capacitance sensing, parallel driving is common, so we are required to integrate repeated integration into it. Besides, we can simultaneously utilize parallel driving with random delay to suppress charger noise further.

There are two steps for the integration of parallel driving and repeated integration (I-PDRI). The first step is that each driving pattern of parallel driving consists of a driving group of repeated integration. Each driving pattern obtains one value of repeated integration. The second step is that we insert a random delay after each driving group until the last driving group. Finally, the inverse matrix of the driving matrix is used to obtain those decoded touch signals. See Fig. 8 for an example. We use the driving matrix $D_{4\times4}$ to produce the driving patterns. Then, we execute the repeated integration for the first driving pattern and insert a random delay after the repeated integration. Next, we do the same steps until the last driving pattern.

The following theorem shows its SNR improvement:

Corollary 1: (Corollary of Theorem 5) The noise standard deviation of I-PDRI is $1/\sqrt{p}$ of the noise standard deviation of repeated integration.

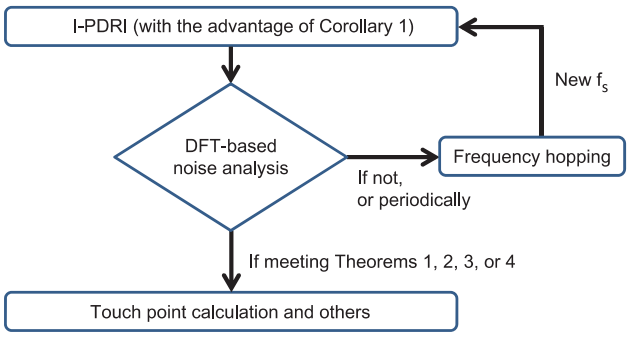

FIGURE 9. The algorithm flow for noise reduction.

Proof 6: Based on the proof of Theorem 5, (10) can be extended for I-PDRI to be:

$$\mathbf{r}_m = \mathbf{D}\mathbf{v} + \mathbf{n}_m, \tag{23}$$

where m is the m-th captured signal in one group of repeated integration.

After doing repeated integration, we can obtain:

$$\bar{\mathbf{r}} = \frac{1}{r}\sum_{m=1}^{r}\mathbf{r}_m = \mathbf{D}\mathbf{v} + \bar{\mathbf{n}}, \tag{24}$$

where $\bar{r}$ and $\bar{n}$ are averages for each group of repeated integration, r is the repeating number of repeated integration.

The p elements in $\bar{n}$ are uncorrelated because random delays are inserted between two adjacent groups of repeated integration.

The rest of this proof is the same with the proof of Theorem 5. Finally, we can obtain:

$$\sigma(\overline{N'}_i) = (1/\sqrt{p}) \cdot \sigma(\overline{N}_o), \tag{25}$$

where $\sigma(\overline{N'}_i)$ is the noise standard deviation of node i of I-PDRI, and $\sigma(\overline{N}_o)$ is the noise standard deviation of repeated integration. ■

The repeated integration method can rapidly suppress the charger noise as mentioned in Sec. III-A. In addition, the parallel driving method with random delay can further reduce the charger noise with a slightly increased time caused by random delay. The more parallel driving channels are used, the greater the noise reduction is.

E. THE WHOLE ALGORITHM FLOW FOR NOISE REDUCTION

Our previously discussed algorithm flow is shown in Fig. 9. The three main steps are summarized as follows:

1) **I-PDRI**: We control the driving circuit and obtain the sensing values using the sensing frequency from the previous analysis. If we have no previous analysis at the beginning, a default setting is used. The sensing value of each parallel driving group on the panel is calculated by repeated integration from a series of consecutive sampling values. After the completion of one parallel driving group, the next group starts. In this process,

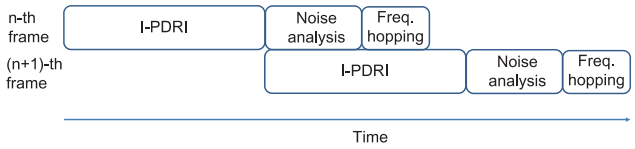

FIGURE 10. The pipeline architecture of our whole algorithm flow.

we record some sampling values for the next analysis. We are not required to record all sampling values for the next step, which would be unnecessary and wasteful of hardware resources. This is because charger noise is a type of finger-coupled noise. Therefore, we only record the sampling values of a certain suspicious touch point. A certain suspicious touch point can be picked from the noisy sensing channels. After picking the node, we drive the node and do sufficient samplings for noise analysis.

2) **DFT-based noise analysis**: After the previous step, we obtain the sensing values of a whole frame and a series of consecutive sampling values. We first calculate the intensities of frequency components by (8). Then, we search for the noisy frequency bin, and consider whether the noise frequency, the sensing frequency and the repeating number meet Theorems 1, 2, 3, or 4. If any of those theorems have been met, we pass the sensing values of a whole frame to the next stage and then calculate the accurate touch points as shown in Fig. 2.
3) **Frequency hopping**: If none of the aforementioned theorems have been met, we use the noise frequency and the repeating number to calculate an appropriate sensing frequency. When the current sensing frequency is close to the frequency of noise, we cannot find the noisy frequency bin. This is because the noise signal appears at the first bin like a DC signal. Therefore, we change the sensing frequency periodically. To separate the noise frequency from the first bin, the frequency change range must be larger than the number f_s/r.

Assume that m is the number of driving channels, n is the number of sensing channels, and r is the repeating number. Because the touch panel has $m \times n$ nodes and every node has r samplings, the repeated integration step requires $O(m \times n \times r)$ time. Calculating intensities of frequency by (8) requires $O(r \log r)$ time applying the radix-2 Cooley-Tukey algorithm and searching for the noisy frequency bin just requires $O(r)$ time because of r frequency bins, so the time complexity of the noise analysis step is $O(r \log r)$. The time complexity of the frequency hopping step is $O(1)$ because it uses the noise frequency and the repeating number to calculate an appropriate sensing frequency.

F. HARDWARE-SOFTWARE CO-DESIGN

To optimize the computing efficiency, our algorithm can be implemented with a pipeline architecture as shown in Fig. 10. When the DFT-based noise analysis step and the frequency hopping step are executed, the I-PDRI step for the next frame is executed simultaneously. The I-PDRI step is implemented by hardware because regular and continuous sampling requires hardware to accurately control the sampling duration. The DFT-based noise analysis step and the frequency hopping step are implemented by software because they have very low time complexities and do not need to be accelerated.

IV. EXPERIMENTAL RESULTS

To verify the correctness and effectiveness of our algorithm and theorems, we constructed a platform to simulate a mutual-capacitive touch system using C/C++. The touch system had 12 driving channels and 8 sensing channels. The sensing frequency was initially set to 300 KHz, the background noise was random noise with amplitude $[-100, 100]$, and the amplitude of each touch point was 1000. Charger noise was finger-coupled, so we directly added charger noise to touch points in our experiments, which was the same as the industry testing standard [2]. The form of charger noise is a combination of a fundamental frequency and its harmonics [2]. The fundamental frequency of our charger noise was set to 290 KHz, and the amplitudes for the fundamental frequency and the second harmonic were set to 3000, which was much larger than that of the touch signal. The intensity of every signal is a transformation of its sensing value (voltage).

Four scenarios were proposed to evaluate the performance of the algorithms:

- Scenario 1 was sequential driving without repeated integration.
- Scenario 2 was sequential driving with repeated integration, which is mainly used to discuss the effectiveness of Theorems 1, 2, 3, and 4. The repeating numbers were 16, 32, 64, and 128, respectively.
- Scenario 3 was parallel driving with random delay but without repeated integration, which is mainly used to discuss the effectiveness of Theorem 5. The driving matrices were 2-by-2, 4-by-4, and 8-by-8, respectively. The 2-by-2 matrix is

$$D_{2\times 2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

  The 4-by-4 matrix is $D_{4\times 4}$ as (9).
  The 8-by-8 matrix is

$$D_{8\times 8} = \begin{bmatrix} D_{4\times 4} & D_{4\times 4} \\ D_{4\times 4} & -D_{4\times 4} \end{bmatrix}.$$

- Scenario 4 was I-PDRI, which is mainly used to discuss the effectiveness of Theorems 1, 2, 3, and 4, and Corollary 1. The repeating number was 64. The driving matrices were the same as those in Scenario 3.

These scenarios were investigated under two cases:

- Case 1 was five fingers touching on the panel with the background noise and the charger noise.

TABLE 1. The SNRs among all the scenarios under the two cases before frequency hopping. The fundamental frequency of charger noise is 290 KHz. The sensing frequency is 300 KHz.

Case	Scenario 1	Scenario 2				Scenario 3			Scenario 4		
	S	S	S	S	S	M:2x2	M:4x4	M:8x8	M:2x2	M:4x4	M:8x8
	No R	R: 16	R:32	R:64	R:128	No R	No R	No R	R:64	R:64	R:64
Case 1	-9.96	-1.90	14.17	14.55	16.74	-6.93	-3.91	-0.93	17.56	20.62	23.58
Case 2	24.81	36.60	40.00	43.16	45.54	27.85	30.86	33.85	46.24	49.19	52.19

TABLE 2. The SNRs among all the scenarios under the two cases after frequency hopping. The fundamental frequency of charger noise is 290 KHz. The sensing frequency is 326 KHz.

Case	Scenario 1	Scenario 2				Scenario 3			Scenario 4		
	S	S	S	S	S	M:2x2	M:4x4	M:8x8	M:2x2	M:4x4	M:8x8
	No R	R: 16	R:32	R:64	R:128	No R	No R	No R	R:64	R:64	R:64
Case 1	-9.96	9.90	14.55	30.04	30.39	-6.93	-3.91	-0.93	33.04	36.10	39.08
Case 2	24.81	36.60	40.00	43.16	45.54	27.85	30.86	33.85	46.24	49.19	52.19

- Case 2 was five fingers touching on the panel only with the background noise and without the charger noise. This case was a comparison group to show the SNR differences with that of Case 1 influenced by charger noise.

The SNR in a touch system is calculated using the following equations [3]:

$$SNR = 20 \times \log_{10} \frac{AVG_{touch} - AVG_{un-touch}}{\sigma(n)}; \quad (26)$$

$$\sigma(n) = \sqrt{\frac{\sum_{n=1}^{100}(S_{touch}[n] - AVG_{touch})}{100}}, \quad (27)$$

where AVG_{touch} and $AVG_{un-touch}$ are the averages of 100 frame values for a certain touch point and a certain untouch point, respectively. $S_{touch}[n]$ is the n-th frame value for a certain touch point.

Table 1 shows the SNRs among all the scenarios under the two cases before frequency hopping, where the sensing frequency is 300 KHz. Fig. 11(a) shows Scenario 1 under Case 2, which only has the background noise. Fig. 11(b) shows Scenario 1 under Case 1. These two figures show that the charger noise severely interfered with the sensing channels, so we were not able to differentiate where the touch points were. Fig. 12(a) shows Scenario 2 with the repeating number 64 under Case 1. As shown in Fig. 12(a), the noise was not suppressed enough because the sensing frequency may have been in a little close proximity to the frequency of the charger noise. These results proved that it is necessary to get a well-tuned sensing frequency. Note that the SNRs of Fig. 11(b) and Fig. 12(a) are approximately -9.96 dB and 14.55 dB, respectively.

Next, we executed the noise analysis and frequency hopping steps to find a well-tuned sensing frequency. The algorithm first found a sensing channel where a suspicious touch point was and used the data of a certain node on the sensing channel for analysis. For the chosen point, we did 256 continuous samplings. Then, the algorithm analyzed the data using (8). Fig. 13(a) shows the intensities of all frequency

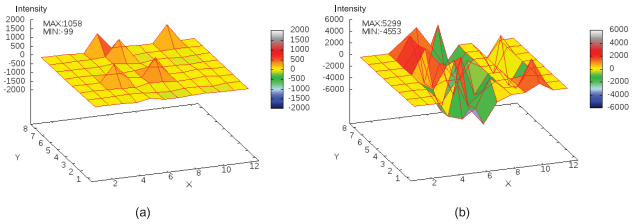

FIGURE 11. (a) The raw map only with the background noise. (b) The raw map with the background noise and the charger noise.

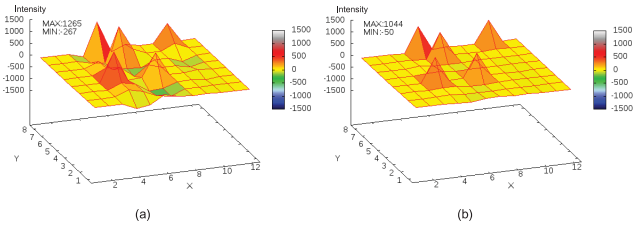

FIGURE 12. (a) The raw map with the repeated integration method for Fig. 11(b). (b) The raw map with the repeated integration method and the frequency hopping method for Fig. 11(b).

components. The noisy frequency bins are the 9-th bin and the 8-th bin. For the 9-th bin, $f = 300 \times (256 - 9)/256 = 289$ KHz. For the 8-th bin, $f = 300 \times (256 - 8)/256 = 291$ KHz. So, the frequency f of the noise or its aliases should be 290 KHz. The frequency f could be f_d, f_{mirror}, f_{folded}, or $f_{mirror,folded}$. (Please refer to Theorems 1, 2, 3, and 4.) 326 KHz is chosen as the new sensing frequency because $f \times r/f_s (= 290 \times 64/326)$ is close to a positive odd integer number based on Theorems 1, 2, 3, and 4. The execution time of the above two steps was 3.5 ms.

After frequency hopping, Table 2 shows the SNRs among all the scenarios under the two cases, where the sensing frequency is 326 KHz. Fig. 12(b) shows Scenario 2 with the repeating number 64 under Case 1, where the touch signals are distinguishable, and its SNR achieved 30.04 dB. Besides, Fig. 13(b) shows the intensities of all frequency components

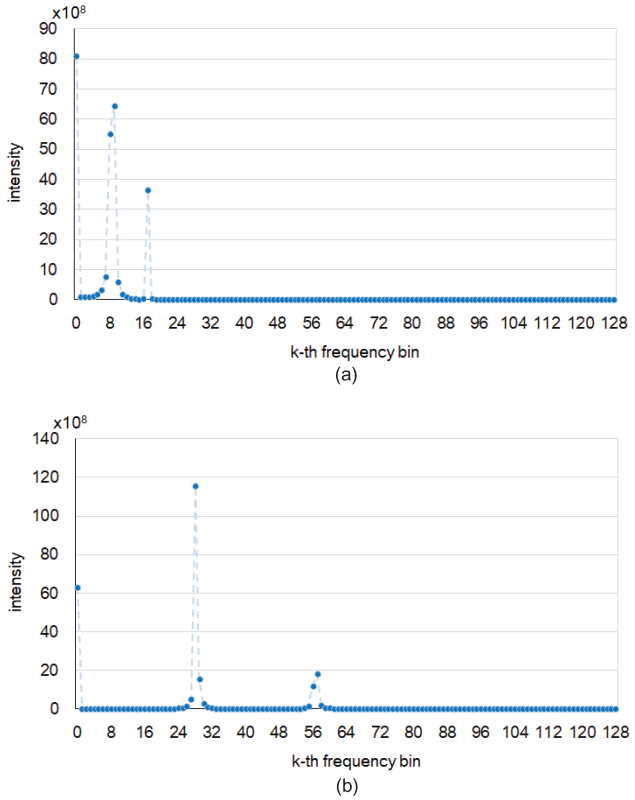

FIGURE 13. The intensities of all frequency components are calculated by (8). (a) When the sensing frequency is 300 KHz, the 9-th and 8-th frequency bins are noisy. (b) When the sensing frequency is 326 KHz, the 28-th frequency bin is noisy.

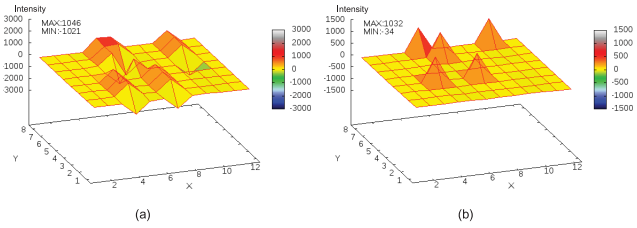

FIGURE 14. The maps with the parallel driving method with random delay for Fig. 12(b). (a) Before decoding. (b) After decoding.

with the new sensing frequency. The noisy frequency falls in the 28-th frequency bin, which is far away from the sensing frequency and meets one of Theorems 1, 2, 3, and 4.

Because our touch system is mutual-capacitive, we applied the parallel driving method with random delay to improve SNR further. Fig. 14 shows the maps of Scenario 4 under Case 1 with a 4-by-4 driving matrix before decoding and after decoding. Before decoding, the SNR was 30.04 dB, which was the same as that of the sequential driving. However, after decoding, the SNR achieved 36.10 dB with approximately 6 dB of improvement. When we used an 8-by-8 driving matrix, the SNR achieved 39.08 dB. Relative to Scenario 1, we obtained 49.04 dB of improvement at most.

In the original settings, the fundamental frequency of the charger noise is lower than the sensing frequency. To fully verify the effectiveness of our algorithm, we changed the fundamental frequency of charger noise to 310 KHz, which is the mirrored frequency with respect to the sensing frequency 300 KHz. We obtained the similar frequency spectrum as Fig. 13(a). Hence, we still identified the frequency f as 290 KHz. Therefore, after the next frequency analysis, the 12-th and 13-th frequency bins are noisy, and its SNR with parallel driving is 23.1 dB. The bins are still close to the sensing frequency, so we picked 279 KHz as the new sensing frequency. Table 3 and Table 4 show the SNRs among all the scenarios under the two cases before frequency hopping and after frequency hopping, respectively. Relative to Scenario 1, we obtained 45.02 dB of improvement using an 8-by-8 driving matrix in Scenario 4.

V. DISCUSSION

First, we summarize the SNR improvements among all scenarios in the experimental results. Then, we discuss the effectiveness of the repeating time based on Theorems 1, 2, 3, and 4 from the experimental results. Finally, we discuss the appropriate execution time for our algorithm.

A. THE IMPROVEMENTS OF SNRs

In Case 2, see all the SNRs in four tables, the SNRs are the same because the SNRs have no relationship with the sensing frequency for the random noise. Besides, Scenario 2 shows that the repeating time n has an improvement factor of $\sqrt{n}$ relative to Scenario 1 because of well-known Bienayme formula. Scenario 3 shows that a p-by-p driving matrix has an improvement factor of $\sqrt{p}$ relative to Scenario 1 because of Theorem 5. Scenario 4 shows that the repeating time n and a p-by-p driving matrix has an improvement factor of $\sqrt{np}$ relative to Scenario 1 because of Corollary 1. (An improvement factor of $\sqrt{np}$ represents $3\log_2 np$ dB of improvement)

In Case 1 of Table 2, we found that the SNRs of Scenario2 with the repeating time 64 have approximately 40 dB of improvement after frequency hopping relative to Scenario 1. This shows that the frequency hopping and the repeated integration is the most effective method to eliminate charger noise. Scenario 4 shows that a p-by-p driving matrix has an improvement factor of $\sqrt{p}$ relative to Scenario 2 with the repeating time 64 because of Corollary 1.

B. THE EFFECTIVENESS OF THE REPEATING TIME

In Scenario 2 under Case 1 of Table 2, we also changed the repeating time to 128 and 32 in separate trials. For the repeating time 128, we obtained an SNR of 30.39 dB which only represented less than 1 dB of improvement relative to the repeating time 64. This proved that when Theorem 1, 2, 3, or 4 is met, the periodic noise can be almost entirely suppressed. Regarding the repeating time 32, its SNR was 14.55 dB, which was similar to the SNR without frequency hopping. It's because 290x32/326 is 28.5, which is not close to a positive odd integer and thus does not satisfy Theorems 1, 2, 3, or 4.

TABLE 3. The SNRs among all the scenarios under the two cases before frequency hopping. The fundamental frequency of charger noise is 310 KHz. The sensing frequency is 300 KHz.

Case	Scenario 1	Scenario 2				Scenario 3			Scenario 4		
	S	S	S	S	S	M:2x2	M:4x4	M:8x8	M:2x2	M:4x4	M:8x8
	No R	R: 16	R:32	R:64	R:128	No R	No R	No R	R:64	R:64	R:64
Case 1	-9.96	-1.93	14.13	14.70	16.35	-6.93	-3.91	-0.93	17.72	20.76	23.77
Case 2	24.81	36.60	40.00	43.16	45.54	27.85	30.86	33.85	46.24	49.19	52.19

TABLE 4. The SNRs among all the scenarios under the two cases after frequency hopping. The fundamental frequency of charger noise is 310 KHz. The sensing frequency is 276 KHz.

Case	Scenario 1	Scenario 2				Scenario 3			Scenario 4		
	S	S	S	S	S	M:2x2	M:4x4	M:8x8	M:2x2	M:4x4	M:8x8
	No R	R: 16	R:32	R:64	R:128	No R	No R	No R	R:64	R:64	R:64
Case 1	-9.96	10.24	14.61	26.01	27.04	-6.93	-3.91	-0.93	29.01	32.08	35.06
Case 2	24.81	36.60	40.00	43.16	45.54	27.85	30.86	33.85	46.24	49.19	52.19

C. THE APPROPRIATE EXECUTION TIME

The acceptable latency is 10 ms based on the user experience in [30], so we must consider whether our algorithm can be run in 10 ms. In our algorithm flow, because the repeated integration step is implemented by hardware, we can increase the number of the receivers to reduce the total sensing time, but it will increase the hardware cost. For example, our touch system needs 3 receivers for the sensing frequency 326 KHz and the repeating number 64, or 8 receivers for the sensing frequency 326 KHz and the repeating number 128. Because our touch panel had only 8 sensing channels, we were only able to use 8 receivers. Regarding the hardware, the noise analysis and frequency hopping steps were implemented by software. To verify the computing efficiency, we ran the noise analysis and frequency hopping steps on an ARM-M0 processor operating at 50 MHz. The execution time was 3.5 ms for 256 sampling data, which was acceptable. If other parts of the multi-touch algorithm take much time, the execution times of the two steps can be reduced to 1.6 ms and 0.5 ms for 128 sampling data and 64 sampling data, respectively. The use of an excessively small number of sampling data would sacrifice the accuracy of noise analysis, but it may still achieve a sufficient SNR.

VI. CONCLUSIONS

Capacitive sensors suffer from various noises, such as display noise, charger noise, and inherent noise. Charger noise is unpredictable and depends on the charger to be used. Therefore, the industry consensus is that charger noise is the most difficult problem in capacitive touch applications. First, we have presented the mathematical model of charger noise, which is the form of charger noise is the combination of a fundamental frequency and its harmonics. Then, we have proven that the combination of frequency hopping and repeated integration is effective to suppress charger noise under certain conditions, as shown in Theorems 1, 2, 3, and 4. Based on those theorems, we could find an appropriate sensing frequency and a certain repeating number for charger noise. As charger noise always changes with different chargers, a DFT-based real-time monitor has been designed to observe the behavior of noise and to find the frequency of noise. After we obtain the frequency of noise, we can choose an appropriate sensing frequency to eliminate noise. To obtain a higher SNR, the parallel driving is a common technique for mutual capacitive sensors without increasing sensing time. However, it is suitable for a random noise, not for a periodic noise. Therefore, we have proposed the random delay technique to make a periodic noise become an uncorrelated noise. Integrating repeated integration into parallel driving with random delay, we can suppress charger noise further. In addition, to optimize the computing efficiency, we have introduced the methods to implement these ideas in hardware pipeline and software, which only increases little hardware cost.

Based on the results achieved, it was concluded that our algorithm can increase SNR a lot for the charger noise, which is also with theorems supporting. For charger noise, frequency hopping and repeated integration can have approximately 40 dB of improvement relative to sequential driving without repeated integration. I-PDRI can an improvement factor of $\sqrt{p}$, where p is the number of parallel driving channels. Therefore, I-PDRI can increase SNR by over 45 dB. Besides, our algorithm can be employed for the random noise, which also has a theoretical improvement of $\sqrt{np}$ relative to sequential driving without repeated integration.

Finally, our results are limited to the charger noise and the random noise in mutual-capacitive touch applications. The other types of applications, such as self-capacitive touch applications, optical touch applications, biometric capacitive fingerprint applications, and biometric optical fingerprint applications, could be considered based on our algorithm and theorems in future studies.

ACKNOWLEDGMENT

The authors thank Wallace Academic Editing for editing this manuscript. They also thank Dr. C. C. Chen for offering assistance in completing this revision. This paper was

presented at the 2017 IEEE International Symposium on Circuits and Systems [1].

REFERENCES

[1] S.-L. Huang, S.-Y. Hung, and C.-P. Chen, "An efficient DFT-based algorithm for the charger noise problem in capacitive touch applications," in *Proc. IEEE Symp. Circuits Syst.*, Baltimore, MD, USA, May 2017, pp. 2094–2097.
[2] H. W. Klein, "Noise immunity of touchscreen devices," Cypress Semicond. Corp., San Jose, CA, USA, White Paper, Feb. 2013.
[3] M. G. A. Mohamed, K. Cho, and H. Kim, "Frequency selection concurrent sensing technique for high-performance touch screens," *IEEE/OSA J. Display Technol.*, vol. 12, no. 11, pp. 1433–1443, Nov. 2016.
[4] T.-H. Hwang, W.-H. Cui, I.-S. Yang, and O.-K. Kwon, "A highly area-efficient controller for capacitive touch screen panel systems," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 1115–1122, May 2010.
[5] S. Ko *et al.*, "Low noise capacitive sensor for multi-touch mobile handset's applications," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Beijing, China, Nov. 2010, pp. 1–4.
[6] I. S. Yang and O. K. Kwon, "A touch controller using differential sensing method for on-cell capacitive touch screen panel systems," *IEEE Trans. Consum. Electron.*, vol. 57, no. 3, pp. 1027–1032, Aug. 2011.
[7] J.-H. Yang *et al.*, "P-134: A high-SNR area-efficient readout circuit using a delta-integration method for capacitive touch screen panels," in *Soc. Inf. Display Int. Symp. Tech. Dig.*, Boston, MA, USA, Jun. 2012, pp. 1570–1573.
[8] J.-H. Yang *et al.*, "A highly noise-immune touch controller using Filtered-Delta-Integration and a charge-interpolation technique for 10.1-inch capacitive touch-screen panels," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2013, pp. 390–391.
[9] H. K. Ouh, J. Lee, S. Han, H. Kim, I. Yoon, and S. Hong, "A programmable mutual capacitance sensing circuit for a large-sized touch panel," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2012, pp. 1395–1398.
[10] J.-E. Park, J. Park, Y.-H. Hwang, J. Oh, and D.-K. Jeong, "A 100-TRX-channel configurable 85-to-385 Hz-frame-rate analog front-end for touch controller with highly enhanced noise immunity of 20 V_{pp}," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2016, pp. 210–211.
[11] S. P. Hotelling, C. H. Krah, and B. Q. Huppi, "Multipoint touch surface controller," U.S. Patent 8 279 180, Oct. 2, 2012.
[12] H. Shin, S. Ko, H. Jang, I. Yun, and K. Lee, "A 55 dB SNR with 240 Hz frame scan rate mutual capacitor 30×24 touch-screen panel read-out IC using code-division multiple sensing technique," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2013, pp. 388–389.
[13] M. Hamaguchi, A. Nagao, and M. Miyamoto, "A 240 Hz-reporting-rate 143×81 mutual-capacitance touch-sensing analog front-end IC with 37 dB SNR for 1 mm-diameter stylus," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2014, pp. 214–215.
[14] M. Hamaguchi, M. Takeda, and M. Miyamoto, "A 240 Hz-reporting-rate mutual-capacitance touch-sensing analog front-end enabling multiple active/passive styluses with 41 dB/32 dB SNR for 0.5 mm diameter," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2015, pp. 120–121.
[15] C. Park *et al.*, "A pen-pressure-sensitive capacitive touch system using electrically coupled resonance pen," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2015, pp. 124–125.
[16] J. K. Park, C.-J. Lee, D.-Y. Kim, J.-H. Chun, and J. T. Kim, "Application of weighing matrices to simultaneous driving technique for capacitive touch sensors," *IEEE Trans. Consum. Electron.*, vol. 61, no. 2, pp. 261–269, May 2015.
[17] H. Ma, S. Heo, J. J. Kim, and F. Bien, "Algorithm for improving SNR using high voltage and differential Manchester code for capacitive touch screen panel," *Electron. Lett.*, vol. 50, no. 24, pp. 1813–1815, 2014.
[18] S.-L. Huang, S.-Y. Hung, and C.-P. Chen, "Clustering-based multi-touch algorithm framework for the tracking problem with a large number of points," in *Proc. ACM/IEEE Des., Automat. Test Eur.*, Grenoble, France, Mar. 2015, pp. 719–724.
[19] Z. Baharav and R. Kakarala, "Capacitive touch sensing: Signal and image processing algorithms," in *Proc. SPIE Comput. Imag.*, San Francisco, CA, USA, Feb. 2011, pp. 1–12.
[20] X. Wu *et al.*, "Touchware: A software–based technique for high–resolution multi–touch sensing devices," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 17, no. 1, pp. 17–30, Oct. 2014.
[21] M. G. Mohamed, T.-W. Cho, and H. Kim, "Efficient multi-touch detection algorithm for large touch screen panels," *IEIE Trans. Smart Process. Comput.*, vol. 3, no. 4, pp. 246–250, 2014.
[22] J.-S. An, S.-K. Hong, and O.-K. Kwon, "A highly linear and accurate touch data extraction algorithm based on polar coordinates for large-sized capacitive touch screen panels," *IEEE Trans. Consum. Electron.*, vol. 62, pp. 341–348, Nov. 2016.
[23] I. Guarneri, A. Capra, G. M. Farinella, F. Cristaldi, and S. Battiato, "Multi touch shape recognition for projected capacitive touch screen," in *Proc. IEEE Int. Conf. Comput. Vis. Theory Appl.*, Lisbon, Portugal, Jan. 2014, pp. 111–117.
[24] S.-L. Huang and C.-P. Chen, "A significant multi-touch algorithm for the tracking problem based on the Hungarian algorithm," in *Soc. Inf. Display Int. Symp. Tech. Dig.*, Vancouver, BC, Canada, Jun. 2013, pp. 1505–1508.
[25] M. Simmon and D. Pickett, "Multi-touch tracking," U.S. Patent 8 866 790 B2, Oct. 21, 2014.
[26] C.-L. Lin, Y.-M. Chang, U.-C. Lin, and C.-S. Li, "Kalman filter smooth tracking based on multi-touch for capacitive panel," in *Soc. Inf. Display Int. Symp. Tech. Dig.*, Los Angeles, CA, USA, May 2011, pp. 1845–1847.
[27] C.-L. Lin, C.-S. Li, Y.-M. Chang, T.-C. Lin, J.-F. Chen, and U.-C. Lin, "Pressure sensitive stylus and algorithm for touchscreen panel," *IEEE/OSA J. Display Technol.*, vol. 9, no. 1, pp. 17–23, Jan. 2013.
[28] C.-L. Lin, Y.-M. Chang, C.-C. Hung, C.-D. Tu, and C.-Y. Chuang, "Position estimation and smooth tracking with a fuzzy logic-based adaptive strong tracking Kalman filter for capacitive touch panels," *IEEE Trans. Ind. Electron.*, vol. 62, no. 8, pp. 5097–5108, Aug. 2015.
[29] C.-L. Lin *et al.*, "Tracking touched trajectory on capacitive touch panels using an adjustable weighted prediction covariance matrix," *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 4910–4916, Jun. 2017.
[30] A. Ng and P. H. Dietz, "The need for speed in touch systems," in *Soc. Inf. Display Int. Symp. Tech. Dig.*, Vancouver, BC, Canada, Jun. 2013, pp. 547–550.
[31] S. G. Johnson and M. Frigo, "Implementing FFTs in practice," in *Fast Fourier Transforms*, C. S. Burrus, Ed. Houston, TX, USA: Rice Univ., Sep. 2008, ch. 11.

SHIH-LUN HUANG received the double B.S. degree in computer science and electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2006, and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2008.

He is currently pursuing the Ph.D. degree in electronics engineering with National Taiwan University. He is also with Synaptics Incorporated as a Staff Engineer for IC design. His research interests include advanced touch screen control architectures, hardware-software co-design of embedded systems, and electronic design automation with an emphasis on placement and routing. He received the Best Paper Award from the IEEE International Conference on Computer Design, in 2010.

SHENG-YI HUNG received the B.S. degree in computer science from National Chiayi University, Chiayi, Taiwan, in 2003, and the M.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2009. He is currently pursuing the Ph.D. degree in electronics engineering with National Taiwan University. His research interests include hardware–software co-design of embedded systems, IC/package/PCB co-design, design automation for analog circuits, and electronic design automation with an emphasis on placement and routing.

CHUNG-PING CHEN received the B.S. degree in computer science and information engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1990, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin, in 1996 and 1998, respectively.

From 1996 to 1999, he was with Strategic CAD Labs, Intel Corporation, as a Senior CAD Engineer. Since 1999, he has been an Assistant Professor with the Electrical and Computer Engineering Department, University of Wisconsin–Madison. Since 2003, he has been an Associate Professor with the Department of Electrical Engineering (EE), National Taiwan University, Taipei. He is currently a Professor with the GIEE, BIO, and EE Department, National Taiwan University. His research interests include EDA and BIO topics, including computer-aided design and microprocessor circuit design with an emphasis on interconnect and circuit optimization, circuit simulation, statistical design, and signal/power/thermal integrity analysis and optimization.

Dr. Chen served as a Program Committee Member and/or Organizer of the DAC, ICCAD, DATE, ISPD, ASPDAC, ISQED, SASIMI, VLSI/CAD Symposium, and ITRS. He received the D2000 Award from Intel Corporation and the National Sciences Foundation Faculty Early Career Development Award (CAREER), from 1999 to 2001. He also received the 2002 SIGDA/ACM Outstanding Young Faculty Award and the 2002 IBM Peter Schneider Faculty Development Award.