A Novel Fixed-Point Conversion Methodology For Digital Signal Processing Systems

Phuong T.K. DINH† ‡, Linh T.T. DINH† ‡, Tung T. TRAN† ‡, Lam S. PHAM† ‡, Han Le DUC† ‡, Chi P. HOANG† ‡, and Minh D. NGUYEN† ‡, Nonmembers

SUMMARY Recently, most signal processing algorithms have been developed with floating-point arithmetic. While the fixed-point arithmetic is more popular with most commercial devices and low-power real-time applications which are implemented on embedded/ASIC/FPGA systems. Therefore, the optimal Floating-point to Fixed-point Conversion (FFC) methodology is promising solution. In this paper, we propose the FFC consisting of signal grouping technique and simulation-based word length optimization. In order to evaluate the performance of the proposed technique, simulations are carried out and hardware co-simulation on Field Programmable Gate Arrays (FPGAs) platform have been applied to complex Digital Signal Processing (DSP) algorithms: Linear Time Invariant (LTI) systems, multi-mode Fast Fourier Transform (FFT) circuit for IEEE 802.11 ax WLAN Devices and the calibration algorithm of gain and clock skew in Time-Interleaved ADC (TI-ADC) using Adaptive Noise Canceller (ANC). The results show that the proposed technique can reduce the hardware cost about 30% while being able to maintain its speed and reliability.

key words: FFC algorithm, 802.11 ax, TIADCs, ANC, FPGA, WL optimization, signal grouping

1. Introduction

Nowadays, DSP algorithms are used in almost all electronic systems. In multimedia applications, DSP has replaced all analog processing components in audio, video encoders/decoders, speech, and image processing. In telecommunications, DSP has been applied, not only in the baseband frequency domain, but also in intermediate and even in radio frequency domains. DSP algorithms can be implemented in General Purpose Processors (GPPs), Digital Signal Processors (DSPs), FPGAs, and Application-Specific Integrated Circuits (ASICs). Regarding GPPs and DSPs, the implementation can be fast and simple by using signal processing specific Application Programming Interfaces (APIs) and libraries. The APIs, libraries in GPPs and DSPs provide digital signal processing instructions/functions such as multiplier - accumulations, floating-point number operations and etc. However, when DSP algorithms are implemented in hardware architecture such as FPGAs and ASICs, they are often require more effort due to the lack of floating point supported libraries. Moreover, the hardware implementation of the floating point DSP algorithms usually leads to a huge chip with considerably energy consumption. In order to save hardware resources as well as energy consumption, the fixed-point architecture needs to be constructed from the floating-point algorithm because the fixed-point hardware is much cheaper and less complex than a floating-point one. Several methods for FFC in several environments have been proposed to help shorten implementation time. In [1], a method which converted floating-point Matlab application into fixed-point version for hardware implementation was proposed. The proposed method is associated with the Accel FPGA behavioral synthesis tool [2]. By using this tool, a synthesizable Hardware Description Languages (HDL) source code can be generated automatically from Matlab descriptions of DSP algorithms. In [3], a FFC methodology for digital VLSI signal processing systems is proposed. In [4], the same author of [3] presents a tool that automates the FFC process for the same kind of system. According to [4], this tool automatically optimizes fixed-point data types of arithmetic operators, including overflow modes, integer word lengths, fractional word lengths, and the number systems. In [5], the FRIDGE design environment is introduced. In this environment, there is a tool that supports the floating-point to fixed-point transformation in which signal processing algorithms coded in floating-point ANSI C are converted to a fixed-point representation in System C. In order to improve the conversion speed, the signals are usually considered in groups such that the fixed-point representation of a signal group is simultaneously determined [6]. In [7], the fixed-point conversion methodology is applied to a multi-channel Wiener filter-based noise canceling algorithm. In [7], they used informal, semi-automatically methodology to convert the floating-point algorithm into fixed-point Simulink model that can be implemented and validated in FPGA.

In this paper which is extended version of [7], we propose a robust and formal FFC methodology which can handle fixed-point conversion for complex DSP systems. Our approach eliminates the limitations of the individual block in system during conversion, which can reduce the complexity of hardware implementation and improve the conversion time. The proposed methodology consists of two main steps: signal grouping algorithm and word-length determination algorithm. For these two important steps we propose: 1) A new heuristic method to group signals which based on transfer functions, signal ranges and expectation of input power;
2) The simulation-based word length optimization algorithm is to detect the saturation point of the error functions of the word length. In order to evaluate the performance of the proposed method, it is applied to LTI systems, the multi-mode FFT for IEEE 802.11 ax WLAN Devices and the calibration algorithm for Time Interleaved Analog Digital Converters (TI-ADCs).

The rest of the paper is organized as follows. Section 2 presents the FFC methodology and the importance role of signal grouping. Section 2.1 presents signal grouping theory as well as the word-length optimization technique. In order to show the efficiency of the proposed calibration, Sect. 3 analyzes simulation results including field- programmable gate array (FPGA) validation. Finally, conclusions are given in Sect. 4.

2. Floating-Point to Fixed-Point Optimization Methodology

2.1 Signal Grouping Technique

As you known, fixed-point formats of all blocks is a time-consuming task because of the continuously assessing the precision of system based on simulation. Reducing number of blocks helps decreasing timing simulation so block grouping can reduce the number of simulation of word-length optimization. [6] indicates that there are some rules to group signal. All of these rules base on basic functions such as adders, subtracts, delays, registers and multiplies. Complicated functions such as accumulation adders and etc have not been mentioned. However, it is challenging to automatically signal group feed back blocks or loop back blocks in real systems as suggested in [6]. Therefore, in this case manual signal grouping would work more effectively. The main problem is that the accuracy of this process depends on the experience of designers.

2.1.1 Signal Grouping Theory

In this section, a new heuristic method to group which are based on transfer functions, signal ranges and expectation of input power will be proposed. Let us model DSP systems in terms of signal blocks and I/O signals. The signal blocks are referred to as filters, adders, multipliers, etc. The I/O signals are the input and output of the signal block. It is worth noting that if each I/O signal is assigned the different and particular signal-word length (WL) in fixed-point presentation, modelling, programming as well as optimizing techniques for the signal WLs of the DSP system become more complicated. Thus, we propose the grouping algorithm that is to categories the I/O signals into the group. The I/O signals in one group have the same WL. This reduces the complexity of WL optimization techniques.

Firstly, to understand about word-length (WL),... according to IEEE Standard for Floating-Point Arithmetic (IEEE 754), the floating-point format has three components: a sign bit S, an exponent exp and a mantissa m. A floating-point number’s value is computed equal \([-1]^s \times 2^{exp} \times \text{mantissa} \]. The bit-width of the mantissa defines the precision of the floating-point representation. A precision of p bits implies a mantissa with (p − 1) bits. In digital signal processing circuits, the blocks that have the same transfer function and the same expectation of input power are put into one group. The group’s word-length of fixed-point data format affects the quantization noise of signal, which has impact on the system performance. Besides, in [8], an expectation of the power of the floating-point quantization error, when a real signal x is converted to floating-point with p’ bits of precision as in Eq. (1):

\[
E(e_f^2) = 2.16 \times E(e_q^2) E(x^2) = 0.18 \times 2^{−2(p'−1)}E(x^2)
\]

(1)

where \( p' \) is the number of fraction bits im-

pemented as:

\[
e^q: \text{quantization noise}, \quad e^q = E(e^q) = E(e^q e^q·)
\]

\[
e^p: \text{quantization expectation of x in the form of floating-point data with p bits of precision.}
\]

\[
e^2: \text{expectation power of x.}
\]

In addition, as in [8], the expectation of the power of fixed-point quantization error, when a floating-point is converted to fixed-point can be determine by Eq. (2):

\[
E(e_f^2) = 0.18 \times 2^{−2(p−1)+1}(1+2^{−2})E(x^2)
\]

(2)

where \( \Delta = p - p' \)

Quantization noise exists in the conversion from real signal to the floating point as well as in the floating-point to fixed-point. From Eqs. (1), (2), when the signals with the same power of expectation are converted to fixed-point type with accuracy \( p' \), they will have the same power of quantization noise expectation. In addition, quantization noise is one of the parameters used to assess quality of fixed-point system. The fixed-point type of data will base on quantization noise to estimate the number of bits for fraction word-length of its. From Eqs. (1), (2) the word-length of the signal can be determined based on power of signal expectation and quantization noise. Moreover, in analog to digital conversion (ADC), the word-length of the signal is also depended on the number of quantum steps in the quantization phase.

In addition, the relationship between input, output signal and transfer functions of block is shown as in Eq. (3):

\[
y(t) = \int_{-\infty}^{\infty} h(\tau) x(t - \tau) d\tau = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau
\]

(3)

\[
y(t) = h(t) * x(t) = x(t) * h(t)
\]

where the input \( x(t) \) is Wide-Sense Stationary (WSS).

\[
E[y(t)] \text{ implies averages of output y(t) and it is ex pressed as:}
\]

\[
E[y(t)] = E[x(t)] \int_{-\infty}^{\infty} h(\tau) d\tau
\]

(4)
The power spectral density of $x(t)$ base on Fourier transform of autocorrelation can be expressed as:

$$S_x(f) = F\{R_x(\tau)\} = \int_{-\infty}^{\infty} R_x(\tau)e^{-2\pi j f \tau} d\tau$$  \hspace{1cm} (5)

Consequently, the expectation power of $x(t)$ can be calculated by spectral power density of $x(t)$:

$$E[(x(t))^2] = R_x(0) = \int_{-\infty}^{\infty} S_x(f)e^{-2\pi j f 0} df = \int_{-\infty}^{\infty} S_x(f) df$$  \hspace{1cm} (6)

The power spectral density of $y(t)$ can be calculated with $H(f)$ which is the transfer function of the constant frequency system:

$$S_y(f) = S_x(f)|H(f)|^2$$  \hspace{1cm} (7)

$$E[(y(t))^2] = \int_{-\infty}^{\infty} S_y(f) df = \int_{-\infty}^{\infty} S_x(f)|H(f)|^2 df$$  \hspace{1cm} (8)

Take derivation of formula (6) we have:

$$S_x(f) df = dE[(x(t))^2]$$  \hspace{1cm} (9)

Replace (9) into (8) we achieve:

$$E[(y(t))^2] = \int_{-\infty}^{\infty} |H(f)|^2 dE[(x(t))^2]$$  \hspace{1cm} (10)

From Eq. (10), if the same expectation of input power are fed into the same transfer function we can receive the same expectation of output power. The word-length of the input and output of all groups are set based on the input, output ranges of them. In addition, signal grouping has to obey the following rule: each group of blocks with the same transfer functions, the same range of inputs and output signals and the same expectation of the input power would be under one name. Signal grouping reduces complexity in fixed-point modelling and also to be easily programmable. When the number of groups is smaller than the number of blocks, the floating-point to fixed-point conversion is performed faster.

2.1.2 Signal Grouping Demonstration

According to the proofing in Sect.2.1, signal grouping is possible for complicated DSP System. Now we will investigate this process by operation of adder. From (1), the same expectation of signal power has the same word-length so we can investigate as follow: assume that $x_1(t), x_2(t), \ldots, x_n(t)$ satisfy $E[(x_1(t))^2] = E[(x_2(t))^2] = \ldots = E[(x_n(t))^2]$, which are fed into the same transfer function of $H(f)$, these outputs will be $E[(y_1(t))^2] = E[(y_2(t))^2] = \ldots = E[(y_n(t))^2]$. These outputs can apply to both the fixed-point data and floating-point data.

Besides, when $x_1(t), x_2(t), \ldots, x_n(t)$ is converted from floating-point to fixed-point, the power of quantization expectation error of $x$ in fixed-point with $p$ bits of precision is $E[e_1^2]$. From that, the received signals, $y_1(t), y_2(t), \ldots, y_n(t)$, have the same word length of fixed-point signal. Therefore, all of input signals which have the same expectation power will be grouped into one group named $G_{in}$. This group is fed into the same transfer function named $G_{trans}$. Resultantly, the output signals have the same expectation power and are grouped into the same group named $G_{out}$. Fig.1 shows that all the signals are in the same signal group of $G_{in}$, however, these signals are fed into two different transfer function groups of $G_{trans}$ and $G_{trans}$. Therefore, the output signals should be grouped into two separated groups named $G_{out}$ and $G_{out}$, respectively.

As mentioned above, all blocks can be grouped based on expectation power and transfer function. For simple systems which include adders and subtractions, we easily verify our introduced block grouping technique. Assume that the system has four inputs which are $x_1, x_2, x_3, x_4$ with the same expectation of signal power. These inputs are fed into the system as shown in Fig. 2.

These same input ranges put into the same transfer function as in Fig. 2 so they are grouped into one. Likewise, subtraction, multiplication blocks will be put into the same

---

**Fig. 1** Grouped blocks based on expectation power of signal and transfer function.

**Fig. 2** Group of adders.
Table 1  Expectation power of input and output.

<table>
<thead>
<tr>
<th>$x_1, x_3$</th>
<th>$x_2, x_4$</th>
<th>$PE_{add_{y_1}}$</th>
<th>$PE_{add_{y_2}}$</th>
<th>$PE_{sub_{y_1}}$</th>
<th>$PE_{sub_{y_2}}$</th>
<th>$PE_{mul_{y_1}}$</th>
<th>$PE_{mul_{y_2}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-1 1]</td>
<td>[−1 1]</td>
<td>0.0081</td>
<td>0.0081</td>
<td>0.0082</td>
<td>0.0082</td>
<td>1.1221 × 10^{-5}</td>
<td>1.1217 × 10^{-5}</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 0.0058$</td>
<td>$P_{e_3} = 0.0057$</td>
<td>$P_{e_4} = 0.0057$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1 1]</td>
<td>[−10 10]</td>
<td>0.0580</td>
<td>0.0578</td>
<td>0.0579</td>
<td>0.0577</td>
<td>1.068 × 10^{-4}</td>
<td>1.052 × 10^{-4}</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 0.0576$</td>
<td>$P_{e_3} = 0.0574$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1 1]</td>
<td>[−10, 000 10, 000]</td>
<td>[57.8892]</td>
<td>[57.4470]</td>
<td>[57.8892]</td>
<td>[57.4467]</td>
<td>0.1148</td>
<td>0.1152</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 57.8892$</td>
<td>$P_{e_3} = 57.4468$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1 1]</td>
<td>[0 1]</td>
<td>0.0082</td>
<td>0.0082</td>
<td>0.0081</td>
<td>0.0081</td>
<td>0.00083</td>
<td>0.00084</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 0.0058$</td>
<td>$P_{e_3} = 0.0057$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1 1]</td>
<td>[0 10]</td>
<td>0.0180</td>
<td>0.1800</td>
<td>0.0410</td>
<td>0.0410</td>
<td>1.0748 × 10^{-5}</td>
<td>1.0764 × 10^{-5}</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 0.058$</td>
<td>$P_{e_3} = 0.058$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-1 1]</td>
<td>[0 10]</td>
<td>0.0624</td>
<td>0.0622</td>
<td>0.0538</td>
<td>0.0535</td>
<td>1.0219 × 10^{-5}</td>
<td>1.0217 × 10^{-5}</td>
</tr>
<tr>
<td>$P_{e_1} = 0.0058$</td>
<td>$P_{e_2} = 57.5564$</td>
<td>$P_{e_3} = 57.9611$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[-10 000 10 000]</td>
<td>[57.7589]</td>
<td>57.5607</td>
<td>57.9654</td>
<td>57.5521</td>
<td>57.9567</td>
<td>0.11143</td>
<td>0.11144</td>
</tr>
<tr>
<td>$P_{e_1} = 57.7589$</td>
<td>$P_{e_2} = 57.7100$</td>
<td>$P_{e_3} = 57.8342$</td>
<td>$P_{e_4} = 57.6082$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Group if they have the same expectation of input power. Furthermore, to validate this signal grouping technique, some systems would be run with Monte Carlo simulation 1000-times. Our experiments apply to the adder, subtractor and multiplication system with four inputs $[x_1, x_3]$ which have the same expectation power of $[P_{e_1}, P_{e_3}]$ and $[x_2, x_4]$ which have the same expectation power of $[P_{e_2}, P_{e_4}]$, respectively. The outputs are $[add_{y_1}, add_{y_2}, sub_{y_1}, sub_{y_2}, mul_{y_1}, mul_{y_2}]$ and the expectation powers of them are $[PE_{add_{y_1}}, PE_{add_{y_2}}, PE_{sub_{y_1}}, PE_{sub_{y_2}}, PE_{mul_{y_1}}, PE_{mul_{y_2}}]$, respectively. These inputs will have the same length of fixed point. If the same expectation of input powers are symmetrical, the output of adder and subtractor will have the same expectation power. Also in Table 1 these adders, subtractors and multiplications with the same transfer function will be grouped into one. From the above: all blocks with the same transfer functions, the same range of inputs and output signals and the same expectation of input power would be grouped into one, so that reduce complexity in fixed-point modelling. Finally, the signals with the same input’s range and expectation power are delivered to the same transfer function blocks, which results in the same expectation power outputs. Hence, the word-length of inputs and outputs of these blocks is similar, which means that these blocks will be grouped into one. This technique helps to significantly decrease the number of groups whose word length need determining, especially with parallel systems such as the TI-ADC system and, thus, the amount of time to implement FFC can be strongly decreased.

2.2  The Word-Length Optimization Technique

This section focus on that it is a simulation based word-length optimization that automatically detects the saturation point of error function of the word-length. In order to implement fixed-point model on FPGA, we need to firstly design a Simulink model that data is floating-point form with infinity precision. Conventional FFC procedure has been established as in Fig. 3 and convert all blocks to fixed-point type one by one. There are two main phases in conventional FFC procedure: estimation of data ranges and word-length optimization. During the first phase, the data ranges of all blocks in Simulink model are estimated based on Monte Carlo simulation. Estimating signal ranges can determine the dynamic ranges of all blocks to avoid overflow and determine integer word-length (IW) of fixed-point type. During the second phase, word-length optimization will determine the suitable numbers of bits for all blocks of the fixed-point model to reduce the hardware resources and chip’s area while still maintain output system error or performance of system. In particularly, word-length optimization will repeat until the fraction word-length for every block and the target function are reached. As you know, the output error dependence on the word-length of each block is investigated.

In Fig. 3, the setting of initial the word-length for the first block is illustrated while the data of other unfixed blocks is
set with floating-point (64 or 32 bits). The system is checked if it reaches the target function (system error, performance system) or not. In case of failure, one unit is added to the initial word-length as long as the word-length is less than 32 or 64 bits. This process is repeated until the last one of the system. Finally, the whole system is run to check the suitability with the target functions and we have the fixed-point model. After all word-length of signal blocks are optimized based on the figure of metrics of digital communication. The target functions in conventional FFC algorithm is to get the difference between the performance of two systems using floating-point and fixed-point. This value should be as small as possible.

For example, for word-length optimization of the 25th block, the error dependence on the word length of this block is depicted in Fig. 4. As it can be seen, the output error is saturated as the word length is greater than 20, which means further increasing the word length does not result in better output accuracy. Therefore, the optimal word length for group 25 can be chosen as 20.

However, in case of complicated system, word-length optimization for all blocks is time consuming because simulations are repeated several times for all blocks. In order to reduce the simulation time of word-length optimization, all computational blocks are needed to be partitioned into groups (signal grouping) based on input signals and their transfer functions.

2.3 Pseudo Code for Proposed FFC Algorithm

Our proposal FFC technique as in Algorithm 1 includes both signal grouping and word-length optimization. The inputs of FFC algorithm are the floating-point architecture and some input patterns such as dynamic ranges, system performance or system error. The floating-point model is considered a DSP data-path named DG(V, E, F). DG(V, E, F) is a graph, where V is the nodes that include inputs, outputs, registers and operation functions such as MUX, DEMUX, addition, subtraction and multiply; E is the connecting edge between nodes; F is the label signaling that assign each edge to a function. The problem with Fixed-point is finding the signal labeling to assign each edge of DG with corresponding fixed point representation or expressed as function: Sfixdt(F), F is a vector containing the S, WL and FWL of every signal. Regarding fixed-point format, S is sign bit, FL is Fraction-Length and WL is Word-Length (WL includes FL and Integer Word-Length (IWL)). Determination of Sfixdt(F) functions is based on input patterns of system. The floating-point could be used as the reference to evaluate the implementation of floating point to fixed-point conversion. The output of FFC algorithm is the fixed-point model is assigned by fixed-point type Sfixdt(F), labelling F to all output blocks. C(F) is a target function that variations in word-length to design such as energy consumption, hardware resource, etc. D(F) calculate degradation in precision caused by a particular F and e implies the maximum precision such as error of system output or performance loss such as SNDR, SFDR loss tolerable by applications.

The procedure in Algorithm 1 is described as bellow:

1) Signal grouping: first, the blocks of the system are divided into the corresponding groups, then the established groups are named. A profiling model is created under the form of the floating point model that is called a DGinfo(V,E,F), where V is group of blocks, V= V1, V2, . . . , VN with N is number of groups of the system and Vi=V1i, V2i, . . . , VNi with Mi is the number of blocks of groupi. It is worth noting that the name of groups is i, i ∈ [1, 2, . . . , N], which is also the order to perform FFC for every group of the system.

2) Estimation of signal ranges: After the profiling model is put into Monte Carlo simulation to collect the input and output ranges of all groups, the signal values are calculated and stored in variables corresponding to input and output of each grouped block as mention in signal grouping phase. When the logging model is simulated by Monte Carlo method, data ranges of each group will be assigned to Workspace, which includes Outmax and Outmin variables. Hence, a suitable number of bits of each grouped block can be determined. Signal ranges estimation for integer word-length determination for all variables of all groups plays an important role in the conversion to avoid data overflowing which affects the accuracy of the system. So as a result, a broader range of signal would be set up as in [9].

3) Word-length optimization (after estimating the data ranges): this process starts setting of initial the word-length for the first groups (min,) while the data of other unfixed groups is set with floating-point (64 or 32 bits). D(F) of the system is checked if it satisfies or not. In case of failure, one unit is added to the initial word-length as long as the word-length is less than 32 bits. This process is repeated until the system satisfies the ϵ, when the word-length is optimized. After that, the simulation process starts again with the next
group until the last one of the system.

Finally, the whole system is run to check \(D(F)\) and we have the fixed-point model \(DG_{\text{out}}(V, E, F)\) with optimized \(Sfixdt(F)\). In case the system cannot satisfy \(D(F)\) or the word-length in the “word-length optimization stage” is more than 32 or 64 bits, signal grouping is carried out again.

**Algorithm 1** FFC algorithm

INPUT: \(DG_{\text{in}}(V, E, F) = (V: \text{nodes}; E: \text{edge between nodes}; F: \text{label signaling})\).
OUTPUT: \(DG_{\text{out}}(V, E, F)\) with \(Sfixdt(F)\); and \(F = \{S, WL_{\text{opt}}, FWL_{\text{opt}}\}\).

1: Workspace \(\leftarrow\) \{Outmin Outmax\}
2: \(Sfixdt(F) \leftarrow\) \{\}
3: \(IWL_i \leftarrow\) \{\(IW_{L_1}, IW_{L_2}, \ldots, IW_{L_N}\)\}
4: \(F \leftarrow\) \{\(F_1, F_2, F_3, \ldots, F_N\)\}
5: for \(i \in\) \{1, \ldots, \(N\)\} do
6: \(F_i \leftarrow\) \{\(S_i, WL_i, FWL_i\)\}
7: \(WL_i \leftarrow \min_i\)
8: while \(WL_i < 32\) do
9: \(WL_i = WL_i + 1\)
10: \(\text{Evaluated } D(F_i)\)
11: \(\text{Workspace } \leftarrow\) Workspace \(\cup\) \{\(WL_i, D(F_i)\)\}
12: if \(D(F_i)\) \(\leq \epsilon\) then
13: \(W_{L_{\text{opt}}}\)
14: jump to 17
15: end if
16: end while
17: \(\text{Update } F_i\) given \(Sfixdt(F)\)
18: \(\text{Update Workspace}\)
19: end for
20: \(\text{Evaluated } C(F)\) and \(D(F)\)

In common problems, the mentioned \(DG(V, E, F)\) will have feedback loop that often appear in LMS, adaptive LMS, ANC algorithm and etc. The issue with implementing fixed point in feedback loop systems is that it is difficult to perform completely automatically with available tools. Therefore, the implementation of semi-automatic FFC will be mentioned.

### 3. Experimental Results

#### 3.1 Set up for Experiments

In order to evaluate the utility of our FFC algorithm, simulations are carried out for finite impulse response filter (FIR filter)- one of the LTI systems, in which most of blocks are adders, subtractors, multipliers, delays and shifters as in [3]. Therefore, FFC need function grouping to reduce operation time. Our technique is then applied to FFT circuit for 802.11 ax WLAN devices which include many blocks with the same transfer functions. Finally, simulations are carried out for gain and clock skew mismatch calibration of TI ADC which includes a loop back inside. The Fig.5 shows the optimal fixed-point hardware architecture in real time signal data in sample-by-sample manner. \(\varepsilon^k\) blocks are delayed registers. The \(sfix18_{-E_{n16}}\) presents an 18-bit signed fixed-point data type with a Fraction Word Length (FWL) of 16. In our work, the hardware architecture is converted into Hardware Description Language (HDL) using MATLAB HDL Coder toolbox. The Vivado tool (Vivado 2015.4), the FPGA in the loop (FIL) and Matlab-simulink (2015.b) are required to validate the proposed calibration on a Xilinx FPGA board. Simulation demonstrates that the synthesized multi-mode FFT circuit operates well on the FPGA (Stratix IV- EP4SGX230KF40C2) and the synthesized gain and clock skew mismatch calibration of TI-ADC also function well on the Kintex 7-KC705.

The optimized word length is based on fixed-point performance of the system, which plays an important role in deciding the word length of each group in the system. Each system has some suitable target functions. For example, mean square error or SQNR are used to determine the word length of FIR, FFT circuit. Meanwhile, SFDR and SNDR are used to optimize the word-length of TI-ADC. It is noticeable that the optimized word-length of signals need to ensure the trade-off between the hardware cost and performance of system.

#### 3.2 The Proposed FFC Technique Application

##### 3.2.1 The Technique Application in LTI Systems

The proposed technique is first used in LTI system- FIR filter- the simplest system in the tree selected ones to check efficiency of our technique. From the impulse response of FIR filter in Eq. (11), the order of the groups in the 20th-order FIR filter designed in Matlab-Simulink is chosen. Then our FFC technique is used to convert the floating-point model into the fixed-point one.

\[
h[n] = \sum_{i=0}^{N} b_i \delta[n-i] \begin{cases} b_n & 0 \leq n \leq N \\ 0 & \text{other } n \end{cases} (11)\]

where, \(b_i\) is the \(i\)th coefficient of the filter.

The optimized word-length of group is the smallest one that satisfies the mean square error of the FIR filters, which is \(10^{-6}\) or \(-60\ dB\). When the word-length of the fixed group is being optimized, the floating-point format is set for non-fixed groups. During the optimizing time, if the mean square error of the FIR is less than \((10^{-6} + \epsilon)\) (\(\epsilon\) is an infinitesimal) for at least 5 simulation times, the smallest word-length will be chosen for the corresponding group. When the word-length of the formal group is already fixed, the process starts again with non-fixed groups. Figure 4 shows the dependence of word-length on mean square error in the 25th group which has a fixed word-length of 22 bits in the selected FIR filter. After finishing the whole process, we have the optimal fixed-point hardware architecture of the FIR Filter shown in Fig. 5.

Otherwise, the differences in the hardware resources for implementing FIR using our technique on FPGA and another such as FFC Tool Matlab, are shown in Table 2. In case of FFC Tool Matlab, it can only fix the word-length of individual blocks in the system but with our technique, both individual blocks and groups of blocks with the same transfer function are grouped. Therefore, results of the 31-group FIR filter and the 3-group FIR filter are shown in the second and
third column in Table 2. Owing to the fact that the number of groups affects mean square error of fixed systems, we have to consider the balance between accuracy and speed.

3.2.2 The Technique Application in Mutlimode FFT System

The efficiency of our technique is also validated for FFT circuit in 802.11ax WLAN devices. IEEE 802.11 ax is the latest amendment to the 802.11 wireless local area network (WLAN) standard, and it promises to create a faster connectivity at 2.4 GHz or 5 GHz band by utilizing OFDM with 1024 QAM, and multi-user MIMO.

\[ X(k) = \sum_{n_5=0}^{3} \sum_{n_4=0}^{5} \sum_{n_3=0}^{15} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} \sum_{n_0=0}^{1} x(n_5 + 4n_4 + 16n_3 + 256n_2 + 512n_1 + 1024n_0)W_{2048}^{nk} \]

(12)

where \( W_{2048}^{nk} = e^{-\frac{2\pi i}{2048} nk} \) is known as the twiddle factor (\( TWs \)) for DFT.

In 802.11ax, the FFT with multiple points is the same as that in 802.11ac, but the FFT size is increased by four times. 256, 512 and 1024-point FFTs are needed to support the bandwidths of 20 MHz, 40 MHz, 80 MHz and 80 + 80 MHz respectively, and 2048-point FFT is used to support the bandwidth of 160 MHz. Increasing the FFT size enables brings more efficient utilization of the data sub-carriers. Our design is based on a combination of mixed Radix and MDC/SDF architecture which was optimized to support the 802.11ax standard with low latency, area, power and high frequency.

In addition, the 'Twiddle factors (\( TWs \)) which are exploited under the form of sine/cosine functions with symmetry properties to reduce hardware resources of FPGA, are stored in the ROM. As shown in Fig. 11 the blocks are designed with Radix \(-2\) MDC for stages 1, 2, 3; Radix \(-2^4\) for stage 4 and Radix \(-2^5\) SDF for the last two stages. Design of FFT architecture using Matlab/ Simulink is based on the formula of FFT as in Eq. (12), so five stages can be created as in [10]. The combination of MDC and SDF architecture is based on expanding formula (12) that improves efficiency and throughput of the FFT processor. MDC architecture includes adders, multipliers and registers as shown in Fig. 6(a). Figure 6(b) shows SDF architecture in which every stage includes butterflies based on Radix and shift registers. Each output will be multiplied with corresponding TWs then fed into the next stage. Additionally, the first three stages are MDC architecture so the input signal is divided into four parts and then fed into four butterflies for every stage, so we can group all adders of butterflies of each stage into the same group. The output of these butterflies are fed into the multipliers to multiply with Tws of the max magnitude which depends on the order of \( x(k) \), so these multipliers are

<table>
<thead>
<tr>
<th>Parameters</th>
<th>FFC Tool Matlab</th>
<th>Proposed FFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of blocks</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>Error</td>
<td>( \approx 4e^{-7} )</td>
<td>( \approx 6e^{-7} )</td>
</tr>
<tr>
<td>Output bits</td>
<td>26</td>
<td>25</td>
</tr>
<tr>
<td>LUTs</td>
<td>1591</td>
<td>503</td>
</tr>
<tr>
<td>FF</td>
<td>1177</td>
<td>725</td>
</tr>
<tr>
<td>DSP</td>
<td>33</td>
<td>22</td>
</tr>
<tr>
<td>IO Pins</td>
<td>56</td>
<td>48</td>
</tr>
</tbody>
</table>

Fig. 4 The mean square error vs WLs of the 20th-order FIR filter.

Fig. 5 The optimal fixed-point hardware architecture of the FIR Filter.
grouped into the same group, and so on. In the same way, for another stage, we can group the blocks with the same transfer function, the same expectation power of signal and the same range of signals. Therefore, in our FFT circuit, 33 groups are created and they still satisfy the requirement of system error. Twos of stages are stored in ROM so we do not need to convert floating-point to fixed-point for them. FFT word-length optimization circuit is executed similarly to what have been done in the FIR filter above. Choosing mean square error of architecture determines word-length of each group as in Fig. 7. For example, choosing of WL’s Group 15 this 25 bits because of unchanging of mean square error of the system when the number of bits increasing from 25 bits. Figure 11 shows that in the first clock, all index data are fed into the first stage. The output of the first stage can be fed directly into the second stage and the output of the second stage can be fed directly into the third stage. Thus, there is no need to use RAM for buffering the output data from three stages above. The memory efficiency of this architecture is high. In the fourth stage, eight parallel radix 2^4 SDF architecture is used. The radix 2^4 butterfly is applied with shift registers and adders as in [10]. In the last stage, eight parallel radix 2^2 SDF architecture is used as in [11].

After performing our FFC technique, we have the fixed architecture as being shown Fig. 11. In the same way as in the first experiment, the hardware resources comparison results for implementing FFT circuit using our technique on FPGA and another technique using FFC Tool Matlab and another technique in [10] are shown in Table 3. We found that the number of fixed groups performed with the former is more than 1000 blocks, while with our technique that is only 33 groups. Thus, the amount of time for FFC is significantly decreased as shown in Table 12.

In addition, hardware resources of FPGA decrease by about 20% while their mean square error is ensured. Table 4 gives more information on implementing this designed FFT architecture on FPGA using the proposed technique compared to results of multi-mode FFT using FFC tool of Matlab and another technique in [10] to converse floating point to fixed point. As can be seen, signal grouping is not included in those two techniques.

Synthesis results of Cadence with Genus 162, XFAB SOI CMOS 180nm for multi-mode FFT for WLAN 802.11ax devices using Genus with Genus 162, XFAB SOI CMOS 180nm. Table 5 and occupies 12,254 mm^2 as shown in Table 6. The synthesized multi-mode FFT uses 249696 logic gates in total as illustrated in Table 7.

From the above, floating point to fixed point conversion proves to be applied in complicated systems and the reduction

---

Table 3  Comparison of number of blocks/groups.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of blocks/groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>[18]</td>
<td>2256</td>
</tr>
<tr>
<td>[10]</td>
<td>1248</td>
</tr>
<tr>
<td>FFC tool Matlab</td>
<td>1958</td>
</tr>
<tr>
<td>Proposed FFC</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 4  FPGA synthesis results of FFT circuit for 802.11 ax WLAN devices.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Proposed FFC</th>
<th>FFC Tool Matlab</th>
<th>[10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>$\approx 1e^{-6}$</td>
<td>$\approx 1.2e^{-6}$</td>
<td>$\approx 1e^{-6}$</td>
</tr>
<tr>
<td>Output bits</td>
<td>24</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>LUTs</td>
<td>55867</td>
<td>81610</td>
<td>80088</td>
</tr>
<tr>
<td>Memory bits</td>
<td>107472</td>
<td>137718</td>
<td>116992</td>
</tr>
<tr>
<td>Total Registers</td>
<td>20893</td>
<td>23016</td>
<td>47129</td>
</tr>
<tr>
<td>IO Pins</td>
<td>710</td>
<td>742</td>
<td>616</td>
</tr>
<tr>
<td>Latency (clock)</td>
<td>285</td>
<td>285</td>
<td>299</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>150</td>
<td>148</td>
<td>111</td>
</tr>
</tbody>
</table>

Table 5  Synthesis of power consumption of multi-mode FFT for WLAN 802.11ax devices using Cadence with Genus 162, XFAB SOI CMOS 180nm.

<table>
<thead>
<tr>
<th>Number of cell</th>
<th>Leakage Power (nW)</th>
<th>Internal Power (mW)</th>
<th>Net Power (mW)</th>
<th>Switching Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>249696</td>
<td>201</td>
<td>26.634</td>
<td>8.98</td>
<td>35.6</td>
</tr>
</tbody>
</table>
Table 6 Synthesis of area of multi-mode FFT for WLAN 802.11ax devices using Cadence with Genus 162, XFab SOI CMOS 180 nm.

<table>
<thead>
<tr>
<th>Number of Cell</th>
<th>Cell area (mm²)</th>
<th>Net area (mm²)</th>
<th>Total area (mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>249696</td>
<td>11.814</td>
<td>0.44</td>
<td>12.254</td>
</tr>
</tbody>
</table>

Table 7 Synthesis of logic gates of multi-mode FFT for WLAN 802.11ax devices using Cadence with Genus 162, XFab SOI CMOS 180 nm.

<table>
<thead>
<tr>
<th>Type</th>
<th>Instances</th>
<th>Area (mm²)</th>
<th>Area (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>113900</td>
<td>8.56</td>
<td>72.456</td>
</tr>
<tr>
<td>Logic</td>
<td>116170</td>
<td>3.07</td>
<td>25.986</td>
</tr>
<tr>
<td>Inverter</td>
<td>16743</td>
<td>0.134</td>
<td>1.135</td>
</tr>
<tr>
<td>Buffer</td>
<td>2906</td>
<td>0.05</td>
<td>0.423</td>
</tr>
<tr>
<td>Total</td>
<td>249696</td>
<td>11.814</td>
<td>100</td>
</tr>
</tbody>
</table>

Fig. 8 SNDR/SFDR vs. WLs of the compensated TIADC output: (a) SNDR. (b) SFDR.

of group number helps decrease time conversion of floating point to fixed point conversion.

3.2.3 The Technique Application in TI-ADC System

A further experiment was carried out to evaluate the efficiency of the proposed method for gain and clock skew mismatch calibration of TI-ADC. A TI-ADC exploits several individual ADCs operating in parallel. In such a time-interleaved structure, sub-ADCs sequentially carry out time shifted analog-to-digital conversion of an analog signal following a round-robin fashion. However, TI-ADC’s performance suffers from channel mismatches consisting of offset, gain and timing mismatches among sub-ADC. The severe effect of these channel mismatches is to produce frequency aliases on the output signal and hence, significantly degrades the Signal-to-Noise and Distortion Ratio (SNDR) and Spurious-Free Dynamic Range (SFDR) performance of TI-ADCs. In practice, the mismatch among the individual ADCs is mostly unknown since the imperfections of each sub-ADC are introduced during process, voltage, and temperature (PVT) variations. Therefore, in order to improve the TI-ADC’s performance, the mismatch errors in TI-ADCs must be continuously tracked and mitigated with calibration techniques. An adaptive noise canceller based on all-digital blind background calibration technique of the gain and timing mismatches in TI-ADC has been designed. By producing the linear presentation of the distortion signals, the gain and

Fig. 9 PSD of the compensated TIADC output before fixed-point and fixed-point: (a) PSD of signal before calibration. (b) PSD of signal after calibration with floating-point model. (c) PSD of signal after calibration with fixed-point model.

Table 8 FPGA synthesis results of gain and clock skew calibration of TI ADC.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Proposal FFC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic utilization</td>
<td>4%</td>
</tr>
<tr>
<td>Combination ALUTs</td>
<td>6409/1824 (4%)</td>
</tr>
<tr>
<td>Memory ALUTs</td>
<td>68/91200 (&lt; 1%)</td>
</tr>
<tr>
<td>Delicate logic registers</td>
<td>5920/182400 (3%)</td>
</tr>
<tr>
<td>DSP blocks</td>
<td>36/1288 (3%)</td>
</tr>
<tr>
<td>Total registers</td>
<td>5920</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>218.39</td>
</tr>
</tbody>
</table>
clock skew mismatch coefficients are estimated when the output signal-to-noise ratio of ANC systems is maximized. In correction procedure, the error signals are reconstructed and then subtracted from the distorted TIADC output to achieve the compensated signals. Our architecture eliminates the limitations of the aforementioned second estimation class such as relaxing the preexistence of an out-of-band region. The digital correction is performed by eliminating timing mismatch of the TIADC output. Assuming that the input signal \( x(t) \) is slightly over-sampled and bandlimited as \( X(e^{j\omega}) = 0 \) for \( |\omega T_s| \geq \pi \), the discrete-time Fourier transform (DFT) of the output \( y[n] \) can be expressed as [12], [13]

\[
Y(e^{j\omega}) = \sum_{k=0}^{M-1} X(e^{j\omega - \frac{2\pi k}{M}}) \tilde{H}_k(e^{j\omega - \frac{2\pi k}{M}}),
\]

where

\[
\tilde{H}_k(e^{j\omega}) = \frac{1}{M} \sum_{m=0}^{M-1} g_m e^{j\omega m} H_d(e^{j\omega}) e^{-j\frac{2\pi k}{M} m},
\]

\[
H_d(e^{j\omega}) = j\omega, \text{ for } -\pi < \omega < \pi,
\]

\( H_d(e^{j\omega}) \) is the frequency response of an ideal derivative filter [14].

\( X(e^{j\omega}) \) is the discrete-time spectrum of the sampled input \( x[n] = x(t)|_{t=nT_s} \).

By expanding, we analyze the output signal which includes input signals and mismatch errors as Eq. (15)

\[
y[n] = x[n] + e[n]
\]

\[
e[n] = c_g^T x_{g,n} + c_r^T x_{r,n}
\]

From (15), to receive the correct signal we need to estimate \( e[n] \) by \( c_g^T, c_r^T \) coefficients’s estimation by adaptive calibration using ANC with feed-back loop at the instant time of \( n \). From that, the calibration architecture is designed and then converted from floating point to fixed point. We then have optimal fixed-point hardware architecture as shown in Fig. 10. All the procedure uses Matlab/Simulink as in [15], [16], [17] and then is implemented on FPGA. The optimizing process of fixed-point conversion is defined by signal ranges and signal Word-Length. The signal range is the span of number that a fixed-point data type and the scale is used to convert signal values into binary representation. The signal WLs influence the SNDR and SFDR performance metrics of the calibration system using our proposed technique. In the optimal fixed-point (OFxp) hardware model, its WLs need to be optimized. Figure 8 presents SNDR and SFDR performance and number of bits (or WL) assigned to the compensated TIADC output. As shown in Fig. 8, although WLs of this group increase to more than 12 bits but the performance of system stabilizes at around 90 dB of SFDR and 60 dB of SNDR. For this reason, we can choose WLs of 12 bits for this group, which is optimized WL to trade-off between performance and hardware cost. The WL optimization of each group in the system is operated similarly. When word-length of all groups in the system are optimized, FFC of the system is completed and optimized WLs is fixed for all signal groups. After that, the PFD of signal before and after calibration with/without fixed-point model is shown Fig. 9. It is obvious that magnitude of SNDR and SFDR in dB satisfy specifications. As can be observed, the optimal values of WL of the compensated TIADC output is 12 bits in order to get the best trade-off between the best performance and hardware cost. This can be seen in Table 8 which shows the FPGA synthesis results of gain and clock skew calibration of TI-ADC that uses optimized WL of all groups fixed point model. The hardware resource accounts for \( \approx 7.36\% \) of resource of FPGA Kintex 7, XCKT325T.

Synthesis results of Cadence with Genus 162, XFAB
Fig. 11 The optimal fixed-point hardware architecture of the FFT circuit for IEEE 802.11 ax WLAN devices.
Table 9  Synthesis of power consumption of gain and timing mismatch calibration of TI-ADC using Cadence with Genus 162, XFab SOI CMOS 180nm.

<table>
<thead>
<tr>
<th>Number of cell</th>
<th>Leakage Power (nW)</th>
<th>Internal Power (mW)</th>
<th>Net Power (mW)</th>
<th>Switching Power (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20953</td>
<td>9.795</td>
<td>10.928</td>
<td>7.25</td>
<td>18.178</td>
</tr>
</tbody>
</table>

Table 10  Synthesis of area of gain and timing mismatch calibration of TI-ADC using Cadence with Genus 162, XFab SOI CMOS 180nm.

<table>
<thead>
<tr>
<th>Number of Cell</th>
<th>Cell area (mm²)</th>
<th>Net area (mm²)</th>
<th>Total area (mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20953</td>
<td>0.8764</td>
<td>0.0324</td>
<td>0.9088</td>
</tr>
</tbody>
</table>

Table 11  Synthesis of logic gates of gain and timing mismatch calibration of TI-ADC using Cadence with Genus 162, XFab SOI CMOS 180nm.

<table>
<thead>
<tr>
<th>Type</th>
<th>Instances</th>
<th>Area (mm²)</th>
<th>Area (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>61116</td>
<td>0.4590</td>
<td>52.381</td>
</tr>
<tr>
<td>Logic</td>
<td>12122</td>
<td>0.3971</td>
<td>45.319</td>
</tr>
<tr>
<td>Inverter</td>
<td>2715</td>
<td>0.0216</td>
<td>2.3</td>
</tr>
<tr>
<td>Total</td>
<td>20953</td>
<td>0.8764</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 12  Comparison of the floating point to fixed point conversion time of the FFT circuit, the TI-ADC system with and without our proposed FFC algorithm

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Time operation with our proposal FFC algorithm</th>
<th>Time operation without FFC algorithm with our proposal FFC algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT 2048</td>
<td>21,780x</td>
<td>293,700x</td>
</tr>
<tr>
<td>TI-ADC</td>
<td>27,060x</td>
<td>432,960x</td>
</tr>
</tbody>
</table>

SOI CMOS 180nm for gain and clock skew calibration of TI-ADC are shown in Tables 9, 10, and 11. Gain and clock skew calibration of TI-ADC consumes a total power of 18.178 mW as in Table 10 and occupies 0.9088 mm² as shown in Table 9. The synthesized calibration of TI-ADC uses 20593 logic gates in total as shown in Table 11.

In the TI-ADC system there is a feedback loop to estimate gain and timing mismatch coefficients using the Monte Carlo simulation to dedicate range of each block. In addition, accuracy of the system depends on the number of simulations. Difference from using the FFC tool of Matlab, our proposed technique can change the number of simulations to satisfy larger dynamic ranges and our work can be applied to systems with feedback loops.

Moreover, as can be seen from Table 12, the performance speed in both FFT circuit and TI-ADC system is much higher with the proposed algorithm. This table shows the conversion time without data range estimation of time for these systems. It is noticeable that the time for range estimation with and without signal grouping technique is similar. The conversion time from floating point to fixed point in these two systems has been shortened to only under 30,000s compared to 293,700s and 432,960s respectively before applying our proposal FFC algorithm. Therefore, it is useful with a system which has feed back loops, several blocks with the same transfer function and the same expectation power of signal. Especially, FFC technique can also be applied to these systems to improve the speed and, consequently, be less time-consuming. The amount of time saved depends on the number of groups reduced. In these systems, conversion time with signal grouping takes up only $\approx \frac{1}{10}$ conversion time without signal grouping.

4. Conclusion

In this paper, a Novel Fixed-Point Conversion Methodology For Digital Signal Processing Systems has been proposed. By introducing the algorithm of FFC, signal grouping based on their transfer function is executed when the number of systems’ groups decreases, so the process of floating-point to fixed-point conversion is faster. In other words, it removes the limitation of the FFC based on an individual block. The simulation results, including FPGA validation, confirm that the FFC algorithm based on signal grouping can almost fully satisfy the system criteria such as mean square error, SNDR, SFDR and performance time. The application of ASIC synthesis of the proposed technique in a way helps to estimate the chip area, power consumption and execution time. Results show that the proposed technique reduces about 30% of the hardware cost while keeping the system fast and reliable.

Acknowledgments

This research is funded by Hanoi University of Industry, Vietnam and EDABK/BKIC Laboratory, Hanoi University of Science and Technology.

References


Minh D. Nguyen obtained a PhD in Electrical Engineering from University of Kaiserslautern in 2009. He worked as a scientific staff at University of Kaiserslautern, Germany. From 2009 to 2016, he worked as Researcher and Lecturer in the School of Electronics and Telecommunications at Hanoi University of Science and Technology. His research activities involve digital hardware design, embedded system design, formal verification of digital design and embedded systems.