A Mixed-signal Self-Calibration Technique for Baseband Filters in System-on-Chip Mobile Transceivers

A Thesis Presented
by
Yongsuk Choi
to
The Department of Computer Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical and Computer Engineering
Northeastern University
Boston, Massachusetts
Nov 2013
Abstract

Taking advantage of continuous advances in the integrated CMOS technology, systems-on-chip (SoC) designs that include analog mixed-signal (AMS) circuits are widely used. Their applications in communications, signal processing, and embedded systems are also increasing. Even though the technology and circuit performance are improved, the analog circuits still suffer from process variations. Furthermore, testing the integrated circuits is becoming increasingly complicated and costly. Therefore, low-cost performance-tuning is required after fabrication to increase yield and to maintain high performance.

In the past, the analog and mixed-signal circuits were tested in different ways to achieve satisfactory test performance. Most common solutions used to be incorporated with automated test equipment (ATE) and device interface boards (DIB). However, for high-performance testing, the cost and time to design the AMS ATE and DIB are increasing. Alternatively, digitally-assisted on-chip built-in calibration of analog integrated circuits is an effective solution to reduce test time and cost, to improve test accuracy, and to eliminate the necessity of external test equipment. As a new approach, an off-line self-calibration technique for analog filter’s cut-off frequency is proposed in this thesis. This can be done by comparing signal amplitude in the pass-band and at the cut-off frequency of the filter.

As a target application of the proposed technique, an analog low-pass filter used in a modern mobile transceiver system is selected, and this consists of RF front-end and DSP block. In the application, the cut-off frequency is accurately tuned in spite of process variations using the proposed self-calibration technique. A digital controller and a magnitude calculator are used to achieve high tuning accuracy and to minimize hardware complexity and silicon area. The magnitude calculator is required to estimate the FFT outputs that are represented with complex numbers.
To verify and demonstrate the proposed tuning technique, the circuits are implemented using a standard 130nm CMOS technology. The digital blocks consume only 0.027mm\(^2\) silicon area with 26-bit of word length of a digital data path in the control flow. The area is a significantly smaller portion compared to the main DSP block of a typical transceiver and the size of the FFT engine that consumes around 1mm\(^2\) of silicon area. The tuning error after calibration is less than 0.4% from its target.
Acknowledgements

This thesis could not have been written without Dr. Kim, who not only served as my supervisor, but also encouraged and challenged me throughout my academic program. He and other faculty members, Dr. Onabajo and Dr. Lombardi, guided me through the thesis process, never accepting less than my best efforts. I thank them all.
Contents

Abstract ........................................... iv
Acknowledgements ................................. vi
List of Tables .................................... ix
List of Figures ................................... x

1 Introduction ................................. 1
   1.1 Motivation .................................. 2
   1.2 Organization of Thesis .................... 4

2 Continuous-Time Baseband Tunable Filter .... 5
   2.1 Receiver Architectures ....................... 6
      2.1.1 Heterodyne Receiver ..................... 6
      2.1.2 Direct Conversion Receiver .............. 7
   2.2 Channel Filter Architectures ............... 9
      2.2.1 Transconductance-C Filters ............. 9
      2.2.2 Active-RC Filters ..................... 12
      2.2.3 Source-Follower-Based Filter .......... 13

3 Proposed Frequency Tuning Method for Continuous-Time Filters 20
   3.1 Existing Tuning Methods ................... 20
      3.1.1 Master-slave scheme .................. 20
3.1.1.1 Phase-Locked Loop Based Tuning Method using a Voltage-Controlled Filter ........................... 21

3.1.1.2 Phase-Locked Loop Based Tuning Method using a Voltage-Controlled Oscillator ...................... 22

3.1.2 Charge-Comparison Based Tuning Method .................................................. 23

3.2 Proposed Tuning Method .................................................................................. 25

3.2.1 Principle of Proposed Algorithm .................................................................... 25

3.2.2 Tuning Algorithm ......................................................................................... 26

3.2.3 Hardware Requirement and Limitation .......................................................... 31

3.2.3.1 Coherent Sampling of FFT ...................................................................... 31

3.2.3.2 Sine-Wave Signal Generation ................................................................. 32

3.2.4 Implementation of the Proposed Tuning Technique ....................................... 33

3.2.4.1 Magnitude Calculator ............................................................................. 33

3.2.4.2 FFT Implementation .............................................................................. 36

4 Simulation Results and Discussion ....................................................................... 38

4.1 Design and Simulation of a Source-Follower Based Biquad Filter ..................... 38

4.1.1 Source-Follower Based Biquad Filter Design .............................................. 38

4.1.2 Capacitor Array Design ............................................................................. 40

4.1.3 Simulation Results ...................................................................................... 42

4.2 Simulation Results for the Closed-Loop Tuning Algorithm ................................... 46

4.3 Synthesis and Layout ...................................................................................... 49

5 CONCLUSION ................................................................................................. 50

A Verilog-HDL source code for the digital controller .............................................. 51

REFERENCES ................................................................................................. 75

viii
List of Tables

1.1 General LTE configurations [13] ........................................ 3
4.1 Biquad filter parameters ........................................ 40
4.2 Summary of the filter parameter ............................. 46
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Heterodyne receiver</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Block-level partitioning of a modern wireless device [16]</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>A typical wireless radio using direct conversion architecture</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Single-ended OTA configuration</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>gm-C implementation of a biquadratic filter</td>
<td>11</td>
</tr>
<tr>
<td>2.6</td>
<td>Opamp-RC integrator</td>
<td>12</td>
</tr>
<tr>
<td>2.7</td>
<td>Source-follower</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>A voltage-mode source-follower based biquad filter [26]</td>
<td>15</td>
</tr>
<tr>
<td>2.9</td>
<td>Half small-signal equivalent circuit for the differential mode</td>
<td>16</td>
</tr>
<tr>
<td>2.10</td>
<td>Transfer function of a source-follower based filter</td>
<td>17</td>
</tr>
<tr>
<td>2.11</td>
<td>Folded version of the source-follower based biquad cell</td>
<td>18</td>
</tr>
<tr>
<td>2.12</td>
<td>Building even-order filter by cascading second-order stages</td>
<td>19</td>
</tr>
<tr>
<td>2.13</td>
<td>Building odd-order filter by cascading second-order stages and adding a single pole</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Master-Slave Tuning Scheme</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Phase-locked loop using a voltage-controlled filter tuning diagram</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Phase-locked loop using a voltage-controlled oscillator for tuning</td>
<td>23</td>
</tr>
<tr>
<td>3.4</td>
<td>Charge detection tuning diagram</td>
<td>24</td>
</tr>
<tr>
<td>3.5</td>
<td>Conceptual flowchart of cut-off frequency tuning</td>
<td>27</td>
</tr>
</tbody>
</table>
3.6 Block diagram of the proposed cut-off frequency tuning algorithm (bold lines - buses with multiple bits) .................................. 28
3.7 Tuning control mode and control signals ................................ 30
3.8 OTA-C quadrature voltage-controlled oscillator structure in [46] . . . 32
3.9 Estimated magnitude superimposed on the actual magnitude for 10K randomly generated sample data .................................... 35
3.10 Error (dB) in the approximated magnitude ............................ 36
3.11 Radix-2 64-point FFT processor (bold lines - buses with multiple bits) 37

4.1 Fourth-order filter schematic .................................................. 39
4.2 Bias circuit ................................................................. 40
4.3 7-bit capacitor array ....................................................... 41
4.4 Implementation of the capacitor array in Fig. 4.3 ....................... 42
4.5 Loop gain stability simulation .............................................. 43
4.6 Input-referred noise (IRN) ................................................... 44
4.7 Output power versus input power for a 1MHz input tone for two-tone test ................................................................. 44
4.8 Output power versus input power with 1MHz input tone .......... 45
4.9 Test-bench for the frequency tuning algorithm ......................... 47
4.10 Magnitude response of the filter for different capacitor array settings .......................................................... 47
4.11 Simulation results of magnitude response after tuning ............... 48
4.12 Digital controller outputs during a simulation of the tuning loop ... 48
4.13 Layout of tuning blocks .................................................... 49
Chapter 1

Introduction

Taking advantage of continuous advances in the integrated CMOS technology, systems-on-chip (SoC) designs that include analog mixed-signal (AMS) circuits are widely used. Their applications in communications, signal processing, and embedded systems are also increasing. As higher specifications are required for analog circuits due to new applications, the performance of the circuits has kept improving. Nevertheless, the analog circuits still suffer from process variations, and performance-tuning is required after fabrication to increase yield and to maintain the high performance. Therefore, testing the integrated circuits is becoming increasingly complicated and costly. [1]

Conventionally, one of the low-cost testing solutions is automated test equipment (ATE) and device interface boards (DIBs). The most common way is to use an AMS ATE tester with an analog DIB [2]. However, the tester needs to have much better performance than the circuits under test (CUTs) in terms of accuracy, speed and noise. In addition, DIBs should be designed carefully, considering the CUT. Therefore, it is a challenging and time consuming work. As another possible solution, digitally-assisted calibration of analog integrated circuits is gaining popularity. The on-chip built-in calibration (BIC) turns out to be an effective solution to reduce test
time and cost, to improve test accuracy, and to eliminate the necessity of the external test equipment. The analog BIC techniques used to be challenging because data converters (ADCs) and digital signal processing (DSP) resources such as fast Fourier transform (FFT) engines are required for accurate analysis in frequency domain and tuning of analog circuits. This makes the BIC approach unrealistic due to power and silicon area overhead.

This thesis proposes an efficient mixed-mode self calibration technique for baseband filters.

1.1 Motivation

Existing digitally-assisted analog circuit calibration and recovery mechanisms include gain and linearity tuning of low-noise amplifiers [3–5], second-order nonlinearity and mismatch correction for mixers [6–9] as well as linearity enhancements for baseband filters [10]. Alternatively, calibration methods have been proposed, which incorporate existing ADC and DSP resources to directly quantize the output signals of the analog circuits for computation of the FFT and automatic tuning with DAC [11,12].

This thesis focuses on the circuit parameter extraction and the performance tuning algorithm with minimally embedded hardwares. In the example of the 4G LTE standard transceivers, which adopts an orthogonal frequency-division multiple access (OFDMA) scheme, a 2048-points FFT with a corresponding sampling rate of 30.72 MHz are used in 20MHz channel bandwidth [13]. In Table 1.1, the general configurations of LTE is specified. Also, a 12-bit ADC with 65-MSPS sampling speed can be used. It also includes RF front-end blocks such as LNA, mixer and analog filters with digital resources for built-in calibration such as an FFT engine and ADC. This example is used as a reference in this thesis because it provides not only a solution
Table 1.1: General LTE configurations [13]

<table>
<thead>
<tr>
<th>Channel Bandwidth, MHz</th>
<th>1.4</th>
<th>3</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT</td>
<td>128</td>
<td>256</td>
<td>512</td>
<td>1024</td>
<td>1536</td>
<td>2048</td>
</tr>
<tr>
<td>Number of Occupied Subcarriers with DC</td>
<td>76</td>
<td>151</td>
<td>301</td>
<td>601</td>
<td>901</td>
<td>1201</td>
</tr>
<tr>
<td>Sampling Frequency, MHz</td>
<td>1.92</td>
<td>3.84</td>
<td>7.68</td>
<td>15.36</td>
<td>23.04</td>
<td>30.72</td>
</tr>
<tr>
<td>Number of RBs</td>
<td>6</td>
<td>15</td>
<td>25</td>
<td>50</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td>Frame Duration, ms</td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-carrier spacing, kHz</td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC-FDM symbols per subframe</td>
<td></td>
<td></td>
<td></td>
<td>6/5 (normal/extended CP)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

to minimize hardware complexity and silicon area but also to maintain analog-circuit performance by off-line tuning mechanisms.

In the LTE transceiver, the analog baseband channel filter is responsible for adjacent channel selectivity, anti-aliasing and dynamic range maximization [14]. A channel filter with low input referred noise (IRN) and high linearity is important for the performance of the whole RF front-end. In terms of the channel selectivity, adjusting the cut-off frequency is essential.

A robust, accurate, reliable and low-cost automatic tuning technique for fine tuning of the analog baseband filter’s cut-off frequency is proposed to overcome the problems with the existing tuning techniques. This thesis includes the control circuit design of the tuning algorithm and the FFT’s magnitude calculation block circuit implementation. The proposed digitally-assisted tuning technique is free from reference signal feedthrough since tuning does not take place during data transmission, which is a problem of master-slave based tuning methods.
1.2 Organization of Thesis

In the consecutive part of this thesis, followings will be discussed. In Chapter II, the existing biquadratic filter will be covered to give basic knowledge to understand the biquadratic filter and its tunability. The circuit structure of linearity tunable filters will be investigated and explored. The source-follower based continuous-time filter will be addressed as a reference circuit for the proposed tuning technique. In Chapter III, conventional filter tuning methods will be addressed. Those are classified into master-slave tuning methods and digitally-assisted tuning methods. The proposed tuning algorithm will be introduced and its implementation will be described. Simulation results will be presented in Chapter IV, followed by the conclusion in Chapter V.
Chapter 2

Continuous-Time Baseband

Tunable Filter

The analog baseband low-pass filter is the stage between a radio-frequency (RF) front-end and a variable gain amplifier. In section 2.1.2, the direct-conversion receiver is presented, and the baseband filter is a part of it. The analog baseband filter is responsible for performing channel-select filtering, either partially with the DSP block or completely in the analog domain. It can be used as a continuous-time or a discrete-time filter. The continuous-time filters have a speed advantage when compared to discrete-time filters, since no sampling is required [15]. Also, the analog-to-digital converter (ADC) followed by variable gain amplifier stage requires anti-aliasing filtering, which can be performed only by continuous-time filters.

The goal of the analog baseband section is to deliver the desired signal to the demodulator with tolerable impairments due to circuit non-idealities [16]. Wireless receivers should extract a weak desired signal from lots of random standardized signals. Successfully tolerating and rejecting undesired spectral content (termed as blockers) close to the desired signal demands requires attentive planning of the analog
signal processing chain that precedes the digital demodulator. In this chapter, several receiver architectures and baseband filter architectures will be presented.

2.1 Receiver Architectures

The RF front-end plays an important role in a mobile transceiver. It consists of all the components in the receiver that process signals at the original incoming radio frequency before it is converted to an intermediate frequency. In the RF front-end, there are several radio receiver architectures. Two of the most commonly used receiver architectures will be reviewed in this section.

2.1.1 Heterodyne Receiver

The heterodyne architecture, shown in Fig. 2.1, is based on a concept that the received RF signal is down-converted into one or multiple intermediate frequencies (IF) before the final translation into the baseband. It has been the most widely used architecture in radio receivers in the past. The heterodyne receiver architecture has high sensitivity and selectivity [17]. However, several frequency conversions are required for this architecture because it uses more than one intermediate frequency (IF) stages.

Figure 2.1: Heterodyne receiver
Thus, the output signal from the mixer has to be filtered and an additional stage is needed. Another drawback in this receiver design is a reduced level of integration because the high-Q RF and IF filters have to be implemented with off-chip passive components in practice. In addition, the incorporation of several receiver stages increases the power consumption and complexity of the system. In comparison, the more compact direct-conversion architecture is suitable for full integration, which is why it has become popular in wireless communication system applications.

2.1.2 Direct Conversion Receiver

Together with the heterodyne receiver, the direct-conversion receiver is a most frequently used receiver architecture. In contrast to the heterodyne receiver, direct-conversion receiver does not use an IF stage. Hence, it is also called homodyne or zero-IF receiver. Development of the integrated circuits and SoC technologies at high frequency allows the RF circuits and baseband signal processing circuits to be integrated in a CMOS chip. A conceptual block-level partitioning of an SoC integrated receiver is shown in Fig. 2.2.

![Block-level partitioning of a modern wireless device](image)

Figure 2.2: Block-level partitioning of a modern wireless device [16]
In the Fig. 2.3, a wireless receiver using the direct conversion architecture is shown. Similarly to the heterodyne receiver, the incoming RF signal is first filtered by an off-chip band-select filter. Then, a LNA amplifies the signal in the next stage.

The baseband low-pass filters can be integrated with the remaining sections of the receiver. Moreover, the integrated analog filters can be designed to be programmable, which makes the direct-conversion receiver suitable for multi-band, multi-standard operation [18,19].

There are many advantages of the direct conversion receiver such as high density low power, less components than heterodyne receiver, and no frequency imaging problem because no intermediate frequency is used. Therefore, there are wide application areas of the direct-conversion receiver.

Figure 2.3: A typical wireless radio using direct conversion architecture
However, the selectivity and linearity requirements of the analog baseband filter become more stringent compared to heterodyne because of the absence of preceding highly linear high-Q RF and IF filters. Also, there are still remaining design challenges. One of the drawbacks is a DC offset problem such that the offset could be large enough to overload the baseband amplifiers and overcome the desired signal reception. In addition, signal leakage paths can occur in the receiver, and flicker \((1/f)\) noise creates design challenges in the direct conversion architecture.

2.2 Channel Filter Architectures

In this section, we will review several types of common filter topologies for integrated continuous-time filters and will discuss the advantages and drawbacks of each of them. Several different circuit techniques to implement analog baseband filters exist. The most popular ones are based on operational amplifiers with resistors and capacitors (opamp-RC) technique, and transconductors with capacitors \((g_m\text{-C})\). In addition, a source-follower based biquadratic technique is presented.

2.2.1 Transconductance-C Filters

An operational transconductance amplifiers (OTA) can be defined as an amplifier where all nodes are low impedances except the input and output nodes. OTAs are used to implement transconductance-capacitor \((g_m\text{-C})\) integrators. In Fig. 2.4, a differential input and a single-ended output OTA is shown. NMOS transistors, \(M_1\) and \(M_2\), are matched with equal W/L ratios. PMOS transistors, \(M_{31}\) and \(M_{41}\), are matched with equal W/L ratios as well. The terminology “1:K” indicates that \(M_4\) and \(M_5\) can be sized K times wider (K:1) than \(M_{41}\) and \(M_{51}\). Assuming that \(\beta_1 = \beta_2, \beta_{31} = \beta_{41}\), it is observed that the current \(i_{d3}\) or \(i_{d4}\) is given by
\[ -i_{d31} = i_{d41} = \frac{g_{mn}}{2} \cdot (V_{IN2} - V_{IN1}) = i_d \]  

(2.1)

where, \( g_{mn} \) is transconductance parameter of \( M_1 \) and \( M_2 \). Furthermore, if \( \beta_4 = K \cdot \beta_{41} = K \cdot \beta_{31} = K \cdot \beta_3 \) and \( K \cdot \beta_{51} = \beta_5 \), then \( i_{d4} = -i_{d5} = K \cdot i_{d41} = -K \cdot i_{d31} \). If the impedance of the capacitor is small compared to the OTA output resistance (at higher frequencies),

\[ i_{out} = i_{d4} - i_{d5} = 2K i_d \]  

(2.2)

and the transconductance of the OTA is given by

\[ G_m = K \cdot \sqrt{2k_n I_B (W/L)_1} \]  

(2.3)

Using single-ended OTAs, a typical \( g_mC \) biquadratic filter implementation is shown in Fig. 2.5.
The transfer function from input to low-pass output can be written as

\[
\frac{V_{\text{OUT}}(s)}{V_{\text{IN}}(s)} = \frac{G_{m1}G_{m2}}{s^2 + s \cdot \frac{G_{mq}}{C_2} + \frac{G_{m2}G_{m3}}{C_1C_2}}
\]  

(2.4)

Assuming the transconductance (except for \(G_{mq}\)) and capacitor values are the same (\(G_{m1} = G_{m2} = G_{m3} = G_m\) and \(C_1 = C_2 = C\)), the cut-off frequency, quality factor (\(Q\)), and the DC-gain (\(K\)) can be written as

\[
\begin{align*}
\omega_0 &= \frac{G_m}{C} \\
Q &= \frac{G_m}{G_{mq}} \\
|K| &= \frac{G_m}{G_{mq}}
\end{align*}
\]  

(2.5)

Typically, gm-C filters are less complex compared to opamp-RC filters and they can be used at higher frequencies. The reason is that the gm-C integrator is an open-loop configuration, and the minimum gain-bandwidth-product (GBW) of the OTA is only required to be more than the cut-off frequency of the filter [20]. However, the
open-loop nature also results in poor linearity of gm-C filters compared to opamp-RC filters.

2.2.2 Active-RC Filters

The operational amplifier RC technique, also called active-RC technique, is based on the Miller integrator shown in Fig. 2.6 [21]. The transfer function of the inverting integrator can be expressed as

$$H(s) = -\frac{1}{sRC}$$  \hspace{1cm} (2.6)

![Opamp-RC integrator](image)

Figure 2.6: Opamp-RC integrator

If an ideal opamp with sufficient gain and GBW was available, the active-RC filter would perform with excellent linearity due to the feedback mechanism [22]. In addition, it also has low excess noise and large swing, resulting in large dynamic range [23]. Active-RC filters are insensitive to the parasitic capacitance, including the capacitance of opamp input and output nodes and interconnection, as far as the opamp can be considered to be an ideal voltage-controlled voltage source (VCVS) [20]. The parasitic capacitances arising at the output and the negative input of the opamp are not directly connected in parallel with the integrating capacitor placed in the feedback path.

However, active-RC filter performance strongly depends on the opamp’s performance, mainly GBW, gain, and output impedance. The GBW should be at least
eight times of $f_{\text{cutoff}} \cdot Q_{\text{filter}}$, where $f_{\text{cutoff}}$ is the cut-off frequency of the filter and $Q_{\text{filter}}$ is the maximum quality factor. This condition is typically difficult to meet under power consumption constraints. In addition, high-gain opamps require complicated multi-stage topologies, such as the Miller compensatetwo-stage architecture, or nested-compensation three-stage architecture [20]. These consume large amounts of power, especially for high GBW.

### 2.2.3 Source-Follower-Based Filter

As discussed in the previous section, the classical gm-C topology suffers from low linearity. Therefore, there are many efforts to improve the linearity of the transconductance [10,24,25]. However, these structures either consume high power (several mW) or require high supply voltage (over 3.3V). In addition, the fully differential active-RC designs have high power consumption and require common-mode feedback circuits for stability. For this reason, we discuss a circuit with low supply voltage, low power, a simple design with high linearity and tunability as adopted model of the proposed filter tuning algorithm in this thesis. The circuit in Fig. 2.8 is a voltage-mode biquad cell that is based on the source follower. First, the source-follower will be investigated in order to identify the benefits of this structure.

**Source-follower based first-order filter**

The source-follower circuit is shown in Fig. 2.7 and the transfer function of source-follower can be written as

$$H(s) = \frac{g_m}{g_m + g_{ds} + g_{do}} \cdot \frac{1}{1 + s \cdot \frac{C_L}{g_m + g_{ds} + g_{do}}} \quad (2.7)$$

where $g_{do}$ is the output conductance of the current source $I_O$, and $g_m$ and $g_{ds}$ are the transconductance and output conductance of the transistor. The source-follower has
a good linearity due to the presence of the local feedback. The closed-loop gain is given by

$$G_{\text{loop}} = \frac{g_m}{g_{ds} + g_{do}}, \quad (2.8)$$

which improves the linearity of the circuit. The larger transconductance value results in a larger loop gain and then in a better linearity. From the equation 2.9, the transconductance is reciprocally proportional to overdrive voltage ($V_{OV}$). The lower $V_{OV}$ can achieve the higher $g_m$ value:

$$g_m = \frac{2I_O}{V_{GS} - V_{TH}} = \frac{2I_O}{V_{OV}}. \quad (2.9)$$

This result is completely different from other active filters, such as $g_m$-C where the linearity is improved at the cost of larger $V_{OV}$, and then larger current (for a given $g_m$) and power consumption [26]. Therefore, high linearity and low power consumption is the advantage of this filter topology.
Source-follower based biquadratic filter

A. Filter parameter

A source-follower based biquad filter design is shown in Fig. 2.8. Assume that all transistors are designed with the same sizes and have the same DC bias currents. As a consequence, they all exhibit the same transconductance, which can be written as

\[ g_{m1} = g_{m2} = g_{m3} = g_{m4} = g_m \]  (2.10)

![Figure 2.8: A voltage-mode source-follower based biquad filter [26]](image)

Fig. 2.9 shows the half small-signal equivalent circuit, which is valid for the differential mode. From this circuit, the filter transfer function can be found as

\[ H(s) = \frac{1}{s^2 \cdot \frac{C_1 C_2}{g_m} + s \cdot \frac{C_1}{g_m} + 1} \]  (2.11)
Also, the cut-off frequency ($\omega_0$), quality (Q) factor and the DC-gain (K) are given by equation 2.12.

$$\begin{align*}
\omega_0 &= 2 \cdot \pi \cdot f_0 = \frac{g_m}{\sqrt{C_1 \cdot C_2}} \\
Q &= \sqrt{\frac{C_2}{C_1}} \\
|K| &= 1
\end{align*}$$

(2.12)

The output common-mode voltage is self-biased by the transistor gate-source voltage ($V_{GS}$) values, without adding any additional circuit such as a common-mode feedback circuit. In addition, the source-follower can drive a resistive load without substantially modifying its linearity performance and its pole frequency. Also, from equation 2.12, the cut-off frequency of the filter can be adjusted by changing the $C_1$ or the $C_2$ value.

**B. DC-Gain Loss**

The source-follower based cell exhibits a DC-gain sensitivity to the bulk transconductance, the effect of which is a DC-gain reduction. The DC-gain can be calculated as a function of the ratio between the transistors' bulk transconductance and their
transconductance ($\eta$) as follows

$$DC\text{-gain} = \frac{1 - \eta}{1 - \eta + \eta^2}$$  \hspace{1cm} (2.13)

This effect can be reduced with a sourcebulk connection. For the PMOS devices, the bulk connection is always available in standard CMOS processes and this allows this gain loss to be avoided. On the other hand, for the NMOS devices, this connection is only possible if the technology has the double-well option. In this case, however, the parasitic capacitance between well and substrate has to be taken into account. In the filter design proposed in the next section, the ratio between the NMOS transistors' bulk transconductance and their transconductance ($\eta$) is 0.4. The simulated transfer function of a design in 130nm technology using the topology in Fig. 2.8 is shown in Fig. 2.10. The filter transfer function exhibits a 20-MHz cut-off frequency, the gain is -5.8 dB in the passband.

Figure 2.10: Transfer function of a source-follower based filter
C. Minimum Supply Voltage

To set up operating points of the transistor, we have to consider the voltage swing of each transistor. Suppose that all transistors have the same overdrive voltages. The minimum supply voltage is given by

\[ V_{DD,\text{min}} = V_{sat} + V_{GS1} + V_{GS3} + V_{swing} + V_{sat} \]

\[ = 4 \cdot V_{sat} + 2 \cdot V_{th} + V_{swing} \] (2.14)

For a 0.13\(\mu\)m CMOS technology, assuming that \(V_{sat} = 140\) mV, \(V_{GS1} = V_{GS3} = 450\) mV, \(V_{swing} = V_{th} = 400\) mV, a \(V_{DD,\text{min}} = 1.5\) V is needed. A popular solution to reduce the \(V_{DD,\text{min}}\) of stacked structures is to use a folded structure. The folded version of Fig. 2.8 circuit is shown in Fig. 2.11. This structure requires a \(V_{DD,\text{min}}\) slightly lower than the stacked on, and it is given by

\[ V_{DD,\text{min}} = V_{sat} + V_{GS1} + V_{swing} + V_{sat} \]

\[ = 3 \cdot V_{sat} + V_{th} + V_{swing} \] (2.15)

![Figure 2.11: Folded version of the source-follower based biquad cell](image)

18
Using the above bias points for the devices, the $V_{DD,min}$ for the folded solution is 1.0 V. Therefore, the folded topology is a good candidate for low-voltage applications, such as for the 0.13μm CMOS technology that supports a supply voltage of 1.2 V. Due to the presented advantages, we chose this architecture as a reference filter design to verify proposed cut-off frequency tuning algorithm of filters. The design parameter and simulation results are presented in chapter 4.

High-Order Cascading

A biquad filter is a type of linear filter that implements a transfer function that is the ratio of two quadratic functions. It is very useful to realize higher-order filters by cascading the second-order filter stages. The concept of cascading biquad filter stages to realize higher-order filters is illustrated in Fig. 2.12 and Fig. 2.13. To implement an n-order filter, n/2 stages are required. Fig. 2.13 extends the concept to odd-order filters by adding a first-order section with a single pole.

Figure 2.12: Building even-order filter by cascading second-order stages

Figure 2.13: Building odd-order filter by cascading second-order stages and adding a single pole
Chapter 3

Proposed Frequency Tuning Method for Continuous-Time Filters

3.1 Existing Tuning Methods

In integrated analog filters, parameter deviations and performance degradations occur due to process and temperature variations and aging. First, conventional tuning approaches such as the master-slave method and charge-comparison based method will be discussed. Then the following sections address the proposed technique for center frequency tuning. It is difficult to make quantitative comparison of tuning techniques; nevertheless, each technique is presented by its weaknesses and advantages in terms of accuracy and practicality.

3.1.1 Master-slave scheme

The conceptual architecture of the master-slave tuning scheme is shown in Fig. 3.1. In this method [27–29], an external reference source is used and the generated signal
is applied to the master filter. The master filter is a replica of the slave. The master filter is used for tuning, and is not used for signal processing. The slave filter is available for signal processing all the time. The tuning block applies the same control signals to the master and the slave filters. With this approach, the matching between two filters is important because it affects the tuning accuracy. Therefore, component matching should be considered carefully to minimize errors. Matching techniques such as common-centroid or interdigitated layout are often used for such purpose. Also note that both master and slave circuits have different loads and different signal paths. Even though the master filter can be tuned precisely, the loading difference will degrade the tuning accuracy.

3.1.1.1 Phase-Locked Loop Based Tuning Method using a Voltage-Controlled Filter

This tuning technique [30] is based on the fact that a voltage-controlled filter (VCF) and a phase detector can be used to tune the cut-off frequency. The basic block diagram is shown in Fig. 3.2. The phase difference between the reference signal and the filter output is detected and lowpass-filtered to generate the control voltage for the cut-off frequency of the low-pass filter. If the VCF is a second-order low-pass filter, the
90° phase shift that occurs at the cut-off frequency is typically used. The PLL loop can track the phase difference of the two signals until 90° phase difference is attained. Either a digital multiplier or an XOR gate can be used as an implementation of the phase detector. The output of the phase detector will have zero time average if there is 90° phase difference between the two signals. There are few drawbacks with this technique: The accuracy of the technique is limited by the offsets of the phase detector and low-pass filter and the reference signal feed-through to the output of the filter can limit the filter’s dynamic range. In addition, both the reference signal and the filter output should pass through a comparator in order to obtain a digital waveform if an XOR gate is used as a phase detector. In this case, the comparator offset will affect the accuracy of the frequency tuning. Approximate calculation results [31] show that when the offset error is 2° in the phase detector, the tuning error will be around 1%.

3.1.1.2 Phase-Locked Loop Based Tuning Method using a Voltage-Controlled Oscillator

The block diagram of a frequency tuning technique with a voltage controlled oscillator (VCO) [32] is shown in Fig. 3.3. The master VCO is constructed with a replica of the integrator in the slave filter. The master VCO oscillates at the reference frequency, and the slave filter cut-off frequency will track the VCO. The phase detector compares the reference frequency with output signal frequency of the VCO. This eliminates the absolute phase accuracy required, such that the phase detector design is more relaxed.
and is not affected by phase offset. This method has an advantage over VCF based tuning, which is that the reference input signal can be a square wave instead of a sinusoidal.

![Phase-locked loop using a voltage-controlled oscillator for tuning](image)

Figure 3.3: Phase-locked loop using a voltage-controlled oscillator for tuning

However, this technique is dependent on the matching of the filter and the master VCO, which is the same characteristic as of the master-slave based tuning method. The VCO implementation causes matching problems with respect to the filter to be tuned [33]. The oscillation amplitude has to be limited to ensure the linear operation. Note that the filter is used in its linear region. The VCO should be in linear region as well, naturally for better matching. To achieve better matching, the filter and VCO should be physically as close as possible. Also, the performance of the filter is degraded in terms of noise due to the feed-through from the reference signal and VCO. Therefore, this method involves a trade-off between matching and isolating the filter from the noise.

### 3.1.2 Charge-Comparison Based Tuning Method

The conceptual schematic of this technique [34] is shown in Fig. 3.4. The phases $\phi_1$ to $\phi_3$ correspond to the switches $S_1$ to $S_3$, respectively. These are connected to three non-overlapping clocks that determine the charge transfer phase of the system. In phase one (when $\phi_1$ is high), the capacitor $C_1$ is discharged and $V_{C1} = 0$. On the other hand, $C_2$ retains its previous charge. In phase two (when $\phi_2$ is high), the capacitor $C_1$ charges with constant current up to $I/G_m$. And the capacitor $C_2$ also...
retains its previous charge. When the $\phi_2$ is high at phase three, the capacitor $C_1$ charge is shared with the capacitor $C_2$. The amplifier feedback forces $V_{C2}$ equal to $V_{ref}$. Therefore, the transferred charge will be balanced by extracting a current of $NI$.

![Figure 3.4: Charge detection tuning diagram](image)

\[
\frac{C_1}{G_m} = T_2 = \frac{N}{f_{clk}}
\]

(3.1)

where $T_2$ is the period that corresponds to phase $\phi_2$ and also matches with a time constant of an integrator, $N$ is current mirror ratio between reference current of $G_m$ and the amplifier. The time constant of integrator (eq. 3.1) locks to an accurate frequency and the tuning signal used to adjust the time constant of the main filter integrators.

The advantage over previous schemes is that the clock frequency can be chosen to be at much higher frequencies compared to filter bandwidth. Therefore, clock feedthrough falls out of band and is attenuated by filter. However, this scheme suffers from inaccuracy caused by mismatch between the master $G_m$-C block and the slave due to different loading and parasitics. Moreover, finite gain of the integrator and amplifier produces error in accuracy almost at the same level as the reciprocal of the individual gains.

24
3.2 Proposed Tuning Method

A digitally-assisted automatic frequency tuning method for continuous-time filters is proposed in this section. This technique uses the fact that the DC signal power is twice the AC signal power at the cut-off frequency. The basic concept, control flow, and hardware implementation are presented.

3.2.1 Principle of Proposed Algorithm

In general, the cut-off frequency in a low-pass filter is defined as a boundary between a passband and a stopband. It is taken to be the point in the filter response at the frequency at which the output power has dropped to half of the nominal passband value. This occurs when the output voltage level has dropped by

\[ \frac{1}{\sqrt{2}} \quad \text{or} \quad 20 \log\left(\frac{1}{\sqrt{2}}\right) \approx -3\text{dB} \quad (3.2) \]

or the output power level has dropped by

\[ \frac{1}{2} \quad \text{or} \quad 10 \log\left(\frac{1}{2}\right) \approx -3\text{dB} \quad (3.3) \]

from the passband value.

Therefore, if the voltage or power level of the filter output in the passband and the desired cut-off frequency can be compared, it is possible to tune the cut-off frequency of the low-pass filter.

To measure and compare the peak voltage amplitude of the filter output at specific frequencies, the analysis region has to be converted from time domain to frequency domain. There are several methods to measure the peak voltage amplitude [35–37]. However, they are limited by signal harmonics and ripples, which leads to less accuracy when measuring the voltage amplitude in the transient domain because of
noise and interference signals. Therefore, the desired signal should be separated from them in order to extract accurate information.

Transformation is used to convert a time domain function to a frequency domain function and vice versa. The most common time-to-frequency transformation for this purpose is the Fourier transformation. Since analyzing sinusoidal functions is easier than analyzing general-shaped functions, this method is very useful and widely used. Theoretically, the Fourier transformation is used to convert a signal of any shape into a sum of infinite number of sinusoidal waves. However, implementation and size of the fast Fourier transformation (FFT) engine is related to its resolution and the number of high-order harmonics of interest. Our research group developed a way to reduce the required size of the FFT for spectrum analysis with negligible accuracy degradation introduced [38].

Frequency spectrum analysis can be realized by using the existing resources on a transceiver chip, such as the ADC and FFT engine for quantization of the filter output signal and frequency analysis. By analyzing and comparing the frequency domain information, we can adjust the cut-off frequency of the filter.

The conceptual flow chart of cut-off frequency tuning is shown in Fig. 3.5, and the detailed algorithm will be discussed in the next section.

### 3.2.2 Tuning Algorithm

A block diagram of proposed frequency tuning algorithm is shown in Fig. 3.6. It includes external equipment such as the clock generator and the sinusoidal signal generator. The AC source is the sine wave signal generator as an input signal to low-pass filter. It generates the desired frequency to be tuned, cut-off frequency of the filter, and a low frequency sine wave signal which has negligible power loss from the DC power level.
Once the tuning mode is turned on, the FLAG\_TUNING signal in Fig. 3.7 is logically high to start the measurement of the filter output at the passband and the DC measurement mode. The filter input is disconnected from the previous stage, and connected to the tuning blocks. In addition, the clock signal passes to the clock divider which provides a divide-by-8 signal from the input reference clock. It is connected to the ADC and digital blocks through the multiplexer as shown in Fig. 3.6. Two different clock frequencies are needed for each DC and AC measurement mode. The clock frequency used in the DC measurement mode is chosen as an integer multiple to simplify the clock divider circuit.

The low-pass filter is used as object to be tuned. Its output signal is captured by a 10-bit ADC. The digitized output of the ADC is passed to the input nodes of the fast Fourier transform (FFT) block to analyze the filter output in the frequency domain.
Figure 3.6: Block diagram of the proposed cut-off frequency tuning algorithm (bold lines - buses with multiple bits)
The size of FFT in modern transceivers is generally 512 to 2048 points [13]. In this thesis, a 64 points FFT is selected using the coherent sampling technique from [38].

Because the FFT outputs are complex numbers, consisting of real and imaginary values, the magnitudes of the complex numbers have to be calculated to compare their voltage levels at certain discrete frequencies (called as FFT bins). The magnitude calculator block plays a key role. The calculated DC value is stored in the register and compared with the result from the AC measurement mode after the tuning mode is changed to it.

In the AC measurement mode, the decoded switch array controls the capacitor array of the low-pass filter circuit and changes the capacitor value that affects the cut-off frequency of the filter. After that, the magnitude in the AC measurement mode is calculated again, and those two values are compared using the comparator. The output signals of the comparator block indicate that the AC value is smaller or larger than the RMS of DC value. For example, if the AC value is smaller than the RMS of DC values, the cut-off frequency of the current setup is lower than expected. Therefore, the capacitor value is reduced to increase the cut-off frequency from the equation 2.12. If the comparator result is converges within a specified range, the comparator block sends a signal to the controller which informs that the filter is optimized and the tuning flow will stop. Otherwise, if all the capacitor array combinations cannot meet the accuracy requirement, the least difference value is stored in the register and it will be adopted at the end of the tuning mode.

**Digital Calibration Control**

The digital controller is designed to manage each digital blocks efficiently and systematically. In Fig 3.7, the control mode is changed from rest to DC measurement, AC measurement and again to rest mode, and this is initiated by the FLAG_TUNING signal. When the tuning is activated, the controller generates a trigger signal (TRIG)
of the FFT block which indicates the starting point of sampling. In addition, it controls a switch control signal for the capacitor array, and a wait signal to allow the filter to settle after switching the capacitor array. The signal named FLAG_OUT notifies that the frequency spectrum is coming out, and it is used by the following blocks to count the number of bins. The FLAG_OPT signal shows that the filter is optimized within a setup range and the END_TUNE signal without the FLAG_OPT signal means that it is tuned to the best possible center frequency, but not within a given error specification. Another purpose of the controller is operation of the digital blocks with low power consumption. In particular, the FFT engine has a high power consumption, which is why it is controlled by the ENABLE signal that activates the FFT engine to be operated when needed.

![Figure 3.7: Tuning control mode and control signals](image)

Figure 3.7: Tuning control mode and control signals
3.2.3 Hardware Requirement and Limitation

3.2.3.1 Coherent Sampling of FFT

Coherent sampling is a useful and efficient technique to evaluate the spectral performance of analog/mixed signal circuits [39–42] because it increases the FFT accuracy and eliminates the need for a window function if certain conditions are met. Coherent sampling of a single tone assures that its power in the spectrum is contained in exactly one frequency bin. The condition for coherent sampling is given as

\[
\frac{f_{1^{st}\text{bin}}}{f_{\text{sample}}} = \frac{N_{\text{cycle}}}{\text{NFFT}}
\]

(3.4)

where \(f_{1^{st}\text{bin}}\) is the fundamental frequency bin after the DC value, \(f_{\text{sample}}\) is the sampling frequency, \(N_{\text{cycle}}\) is the integer number of cycles of the signal to be sampled, and \(\text{NFFT}\) is the length of the FFT engine.

To ensure coherent sampling, one should first determine the number (usually a prime number) of integer cycles (\(N_{\text{cycle}}\)) that fits into the predefined sampling window, and use it to approximate the input frequency to the near optimal frequency that exactly matches with one of the discrete frequency bins in the spectrum for the given FFT length [43]. Under the condition in equation 3.4, there will not be any leakage because the coherent sampling guarantees an exact integer number of input signal cycles.

In the proposed algorithm, 64 is used as the \(\text{NFFT}\) and one cycle is taken for the sampling window. The sampling frequency (\(f_{\text{sample}}\)) is different in the DC and AC measurements. For the DC measurement mode, we can take 125 KHz or a near value (referring to Fig. 2.10) as the first bin (\(f_{1^{st}\text{bin}}\)), resulting in the \(f_{\text{sample}} = 16\text{MHz}\). In the AC measurement mode, if we chose 20 MHz as the \(f_{1^{st}\text{bin}}\), then the required sampling frequency for the ADC and FFT is 1.28 GHz and this specification requires a high-performance ADC and FFT, also resulting in high power consumption.
Therefore, by taking a higher number of bins in the frequency spectrum of FFT, the sampling frequency requirement can be lowered. Hence, the 10-th frequency bin is selected such that the \( f_{\text{sample}} = 128 \) MHz with \( f_{1^{\text{st}} \text{bin}} = 2 \) MHz can be used.

### 3.2.3.2 Sine-Wave Signal Generation

In a test environment, test signals with coherent input frequencies can be generated arbitrarily with standard automatic test equipment. Even though the proposed on-chip self-calibration technique could be utilized to reduce off-chip resource requirements and the number of required test outputs of the chip for test cost reduction, it is envisioned to be more valuable during in-field testing and self-calibrations. On-chip test signals can be generated by dedicated circuits for built-in self-test such as the 40-MHz generator in [44], or by sinusoidal oscillators with wide-frequency tuning range such as the 1 to 25-MHz oscillator in [45].

![OTA-C quadrature voltage-controlled oscillator structure in [46]](image)

Signal generation methods with a digital foundation are advantageous in ensuring coherence with the proposed approach. For example, the 41-MHz signal generator in [46] contains a block that creates a stepwise approximation of a sine wave using
a digital master clock \( (f_{clk}) \) that is 16 times higher than the output frequency. This synthesized sine wave is subsequently processed by an analog filter to generate a purer sinusoidal output with a 67-dB spurious-free dynamic range. Since the signal generator in [46] takes up only 0.1mm\(^2\) in 0.35-\(\mu\)m CMOS technology, it would be a good candidate for applications that require the generation of coherent input signals on the chip. With such a signal generator, the master clock that produces \( f_{1st_{bin}} \) in equation 3.4 can be directly derived from \( f_{sample} \) with a simple digital divider, or vice versa.

### 3.2.4 Implementation of the Proposed Tuning Technique

Additional technical challenges as well as significant increase in cost is incurred when estimating the voltage amplitude at radio frequencies. However a cost effective and computationally inexpensive technique is possible based on mathematical analysis, proper frequency selection, and formulation of appropriate algorithms. The key elements of the proposed tuning algorithm shown in Fig. 3.6 are the magnitude calculator and digital calibration control. All verilog hardware description language (verilog-HDL) codes to implement the digital blocks are presented in Appendix A.

#### 3.2.4.1 Magnitude Calculator

The Fast Fourier transform (FFT) is a standard mechanism that is widely used for spectral analysis. The FFT algorithm calculates the spectrum of the input signal at certain discrete frequencies called FFT bins, which are separated by the FFT fundamental frequency known as FFT resolution. The FFT engine produces a complex output, \( Z(Re, Im) \) consisting of \( N \)-bit real and imaginary parts represented by \( Re \) and \( Im \), respectively. The mathematical way of calculating the magnitude of a complex
number requires a square root operation as defined by equation 3.5:

\[
\text{Magnitude}\{Z\} = \sqrt{Re^2 + Im^2}
\]  

(3.5)

The real and the imaginary part at the output of the FFT engine are less meaningful when interpreting the power level of the spectral components. In order to determine the power level of the spectral components it is required to calculate the magnitude of the each spectral component. To achieve the desired measurement accuracy, the numbers are usually represented in fixed-point or in floating-point notation which poses a significant area and power overhead for on-chip estimation of the magnitudes. Therefore, the traditional way to extract the power spectrum from the FFT output is to transfer the numbers generated at the FFT output to the off-chip resources (such as a PC) and to exploit mathematical tools such as MATLAB etc. to calculate the power spectrum. However these approaches are very inefficient and put limits on on-chip built-in-calibration (BIC) and built-in-test (BIT) approaches where it is critical to obtain an estimation of the spectral characteristics for dynamic tuning of the circuit under test (CUT).

An alternative way to determine the magnitude of a complex number based on the ‘alpha max plus beta min’ algorithm is adopted and is defined by [47]:

\[
\text{Magnitude}\{Z\} \approx \alpha \cdot \max(|Re|, |Im|) + \beta \cdot \min(|Re|, |Im|),
\]  

(3.6)

where \(\max(|Re|, |Im|)\) and \(\min(|Re|, |Im|)\) represent the maximum and the minimum absolute values of the real and imaginary part respectively. The approach represented by equation 3.6 is a linear approximation of the magnitude of a complex number. The above approximation is simple and can be efficiently implemented for on-chip estimation of the magnitude. The absolute values (\(|\ |\)) are easily calculated by just dropping sign bits. Both the \(\max(|Re|, |Im|)\) and \(\min(|Re|, |Im|)\) calculation can be
done with one comparison. However, two new coefficients, $\alpha$ and $\beta$ are introduced in the approximation. The values of $\alpha$ and $\beta$ can be iteratively determined depending on the desired accuracy, and other performance parameters such as area, power and the available computational resources. Simulations were performed with MATLAB to determine the values of $\alpha$ and $\beta$ to achieve good accuracy. In the simulation, the values of $\alpha$ and $\beta$ are randomly generated from a Gaussian distribution with different mean and standard deviation, and the error introduced by the approximation is estimated. Fig. 3.9 shows the estimated magnitude for 10K samples superimposed on the actual magnitude with $\alpha = 1$ and $\beta = 1/4$. The error introduced by the equation 3.6 is around 1 dB, which is shown in Fig. 3.10.

The value of $\alpha = 1$ helps to reduce the count of the N-bit fixed point/floating-point multiplication, which is why $\alpha = 1$ and $\beta = 1/4$ was chosen, allowing the binary calculation (divide by $2^2$) to be easily and efficiently implemented by a bit-shift operation.

![Figure 3.9: Estimated magnitude superimposed on the actual magnitude for 10K randomly generated sample data](image-url)
3.2.4.2 FFT Implementation

A radix-2 64-point FFT engine is implemented to determine the spectral characteristics of the signal generated at the output of an ADC. The FFT engine is based on the standard decimation-in-time algorithm. It is designed as a serialized, streaming I/O FFT block that accepts streaming complex input and generates streaming complex output continuously with every clock cycle after an initial latency of 136 clock cycles, where each output corresponds to a frequency bin. The input and the output data streams are represented in two’s complement Q10.0 and Q14.0 format, respectively.

The output of the 10-bit ADC represents the integer portion of the input data, where four additional fractional bits are appended to achieve a resolution of $-84$ dBc. The integer portion of the output data is comprised of 14 bits to capture the overflow that is generated during the FFT computation.

Fig. 3.11 shows the block diagram of the FFT engine, where $I_{in}$, $I_{out}$, $Q_{in}$, and $Q_{out}$ represent the real and the imaginary parts of the input and the output data streams, respectively. The input of the FFT block consists only of real numbers, and...
the input imaginary values are all zeros. The input data is passed to the FFT logic unit and the processed data is carried to the Butterfly unit for further arithmetic operations or to the DualPort RAM unit for storage in the registers. The functions of the FFT logic unit are to reorder the output bins of the FFT engine, to calculate addresses for the butterfly unit, and to count the delay for feedback registers. Its outputs are the twiddle indexes, addresses for the DualPort RAM, calculated real and imaginary parts of data, and FFT outputs \((I_{out}, Q_{out})\). The twiddle indexes are passed to the twiddle table unit, where the sine and cosine values are selected for the calculations inside the butterfly unit. The outputs of the butterfly unit and the DualPort RAM are two pairs of real and imaginary numbers that are calculated in parallel. To reduce hardware complexity, the DualPort RAM serves as feedback delay resistors, and a minimized butterfly unit is used.
Chapter 4

Simulation Results and Discussion

In this chapter, the design and simulation results for a reference filter (introduced in chapter 2) are presented. This filter was used to demonstrate the proposed tuning algorithm through simulations.

4.1 Design and Simulation of a Source-Follower Based Biquad Filter

4.1.1 Source-Follower Based Biquad Filter Design

A reference fourth-order Bessel baseband filter satisfying typical LTE receiver specifications has been designed as a cascade of two single-branch cells. The fourth-order source-follower based biquad filter is displayed in Fig 4.1. A 130nm standard CMOS technology was used to design with a supply voltage of 1.2 V. All the transistors are designed with 0.5μm (non-minimum) channel length to reduce output impedance effects. The first stage is composed of PMOSFETs and the second stage is designed with NMOSFETs.

Equation 2.12 is re-written here in terms of the transconductance, where the output conductance of transistors are not ignored:
According to equation 4.1, a small \( g_m \) value indicates small capacitor area for a given Q value. On the other hand, \( g_m \) determines the input noise value:

\[
IRN^2 = \frac{64}{3} \cdot \frac{kT}{g_m}
\]  

The \( g_m \) value has to be selected from a trade-off between input noise and capacitor area. Therefore, \( g_m = 2mA/V \) was chosen and the capacitor values is around 10pF. Also, the channel bandwidth is designed as 20MHz.

All filter parameters are listed in Table 4.1. The parameters for the first cell are for the biquad cell on left side in Fig. 4.1, and the bottom half of the table presents the biquad cell design with NMOS transistors (on the right side in Fig. 4.1).
Table 4.1: Biquad filter parameters

<table>
<thead>
<tr>
<th></th>
<th>$g_m$ [mA/V]</th>
<th>$g_{mb}$ [mA/V]</th>
<th>$C_{n1/2}$</th>
<th>$C_{n2/2}$</th>
<th>Q</th>
<th>$f_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1$^{st}$ cell</td>
<td>$M_{P1}, M_{P3}$</td>
<td>2.0</td>
<td>0.35</td>
<td>5.5 pF</td>
<td>19.2 pF</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>$M_{P2}, M_{P4}$</td>
<td>2.0</td>
<td>0.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2$^{nd}$ cell</td>
<td>$M_{N1}, M_{N3}$</td>
<td>2.0</td>
<td>0.26</td>
<td>6.1 pF</td>
<td>11.2 pF</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>$M_{N2}, M_{N4}$</td>
<td>2.0</td>
<td>0.33</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The threshold voltage of NMOS transistor varies from 300 to 350mV, and the PMOS threshold voltage is from 360 to 420mV. To allow for enough output swing range with the low supply voltage (1.2V), the overdrive voltage of $M_1$-$M_4$ are biased at around 150mV. The bias circuit for the filter is shown in Fig. 4.2. It was designed to reduce the sensitivity to supply voltage variations.

![Bias circuit](image)

Figure 4.2: Bias circuit

4.1.2 Capacitor Array Design

The filter cut-off frequency is tuned by changing the capacitors $C_{12}$ and $C_{22}$ in Fig. 4.1. The fixed capacitor corresponds to 20MHz bandwidth of the filter [48]. To achieve 50% of tuning range, from 15MHz to 25MHz, a 7-bit capacitor array was
designed, which is visualized in Fig 4.3. The capacitance is increased and hence the filter bandwidth is reduced (2.12) by switching the binary-weighted unit capacitors ($C_k$s). $C_1$ is the minimum unit capacitor with 80 fF and the other capacitance are set as $C_k = 2^k \cdot C_1$.

![Figure 4.3: 7-bit capacitor array](image)

In principle, all of the capacitors in the filter could be implemented with four parallel capacitor matrices. However, this would lead to a large chip area. Therefore, the design was finalized to minimize the number of capacitors and the area by merging the separate capacitor arrays. In the final design, the frequency response is tuned with 7-bit binary-weighted switched-capacitor matrices and the default value is $C_X$ in parallel with $C_7$ ($C_k=0$ for $k=1$ to 6).

The schematic of the capacitor array is shown in Fig. 4.4. During the phase $\phi_k$, the capacitor $C_k$ is connected to the fixed capacitor $C_X$ in parallel. On the other hand, in the phase $\overline{\phi}_k$, the $C_k$ is disconnected from the $C_X$ and the transistor $M_{Nk}$ is used to avoid charge accumulation [49]. The implementation of the small unit capacitors demanded careful layout design in order to minimize the parasitic capacitances.
4.1.3 Simulation Results

In this section, simulation results are summarized for the filter performance parameters such as loop stability, noise and linearity. In addition, the automatic tuning algorithm is verified.

The stability of the positive feedback is guaranteed. The loop gain obtained by cutting the loop differentially at the gate of $M_{P1}$ and $M_{P3}$ in the PMOS biquad, and $M_{N2}$ and $M_{N4}$ in the NMOS biquad (Fig. 4.1) at low frequency is given by

$$G_{loop} = \frac{g_{m2} \cdot (g_{do} + g_{ds2})}{(g_{m2} + g_{do} + g_{ds2}) \cdot (g_{m1} + g_{ds1})}$$  (4.3)

Therefore, it is always less than 1. This analysis also agrees with the simulation result shown in Fig. 4.5. The maximum loop gain is 0.14 at 31.8MHz. Therefore, the loop gain is always less than one and the stability is guaranteed.

Fig 4.6 shows the noise performance of the biquad filter. The simulated input-referred noise of the biquad filter is presented, and the minimum input-referred noise
density is about $11.86 \text{nV/} \sqrt{\text{Hz}}$ at 20MHz, mainly from thermal noise. When the frequency approaches the cut-off frequency, the noise density rises up from the equation 2.12 and 4.2. Also, the integrated root-mean-square of the total input noise is $35.12 \mu V_{rms}$ over the bandwidth specified from 100kHz to 30MHz.

In-band linearity was simulated with a two-tone at 10MHz and 11MHz. This corresponds to an in-band input third-order intercept point (IIP3) of 23dBm, as shown in Fig. 4.7.

The input-referred 1-dB compression point ($P_{in,1dB}$) was simulated using a 1MHz input tone. In Fig. 4.8, the output power is plotted as a function of the input power when a 1MHz sine-wave signal is applied at the input. From these simulation setup, the $P_{in,1dB}$ is obtained as 4.46dBm.
IRN = 11.86 nV/\sqrt{\text{Hz}} @ 20 MHz

Figure 4.6: Input-referred noise (IRN)

IIP3 = 23 dBm

Figure 4.7: Output power versus input power for a 1 MHz input tone for two-tone test
The dynamic range is calculated as

\[ DR = \frac{2}{3} \cdot \left( P_{in,1dB} + 174 - NF - 10 \cdot \log(BW) \right) [dB], \quad (4.4) \]

where the \( P_{in,1dB} \) is the input-referred 1-dB compression point in dBm, and the factor of 174 dB normalizes the measurement to the theoretical noise floor of -174 dBm/Hz. From the above equations, the dynamic range of the filter is calculated as 69.3 dB at 20 MHz frequency. The NF is the noise figure that can be calculated from

\[ NF = 10 \cdot \log \left( \frac{V_{n,\text{out}}}{A} \right)^2 \frac{1}{4kTR_s}, \quad (4.5) \]

where \( A \) is the voltage gain. The key filter performance parameters are listed in Table 4.2.
Table 4.2: Summary of the filter parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>CMOS 130 nm</td>
</tr>
<tr>
<td>Power supply</td>
<td>1.2 V</td>
</tr>
<tr>
<td>Current consumption</td>
<td>0.427 mA</td>
</tr>
<tr>
<td>Power consumption</td>
<td>0.513 mW</td>
</tr>
<tr>
<td>DC gain</td>
<td>-5.8 dB</td>
</tr>
<tr>
<td>$f_{-3dB}$</td>
<td>19.6 MHz</td>
</tr>
<tr>
<td>$f_{-3dB}$ tuning range</td>
<td>16.2 - 24.4 MHz</td>
</tr>
<tr>
<td>Input Referred Noise</td>
<td>11.86 nV/$\sqrt{Hz}$</td>
</tr>
<tr>
<td>in-band IIP3 ($f_1 = 10$ MHz, $f_2 = 11$ MHz)</td>
<td>23 dBm</td>
</tr>
<tr>
<td>1dB compression point</td>
<td>4.46 dBm</td>
</tr>
</tbody>
</table>

4.2 Simulation Results for the Closed-Loop Tuning Algorithm

The test setup for the proposed filter frequency tuning algorithm (in Fig. 3.6) is presented in Fig. 4.9. The two sine-wave signals are connected to input of the filter through a multiplexer which is controlled by the MODE signal from the controller block. The output signal of the filter is amplified and transmitted to the FFT engine through the ADC. The control signal for the binary capacitor array is sent from the controller based on the output of the FFT engine and the consecutive digital calculations as described in section 3.2.2.

The simulated AC frequency responses with different capacitor array settings are shown in Fig. 4.10. The initial cut-off frequency of the filter is in a range of 16.2MHz to 24.4MHz. The result after the automatic tuning simulation is shown in Fig. 4.11. The cut-off frequency of the filter is tuned between 19.92MHz to 20.07MHz. This corresponds to 0.39% tuning accuracy. Also, the digital controller outputs corresponding to one of the simulation results are displayed in Fig. 4.12.
Figure 4.9: Test-bench for the frequency tuning algorithm

Figure 4.10: Magnitude response of the filter for different capacitor array settings

\[ f_{3dB} = 16.2 \text{ MHz} \sim 24.4 \text{MHz} \]
Figure 4.11: Simulation results of magnitude response after tuning

Figure 4.12: Digital controller outputs during a simulation of the tuning loop
4.3 Synthesis and Layout

The FFT engine and all the other digital blocks are implemented in verilog-HDL (Appendix A) and synthesized to the gate level netlist with a 0.13μm standard CMOS technology PDK. The generated gate level netlist was ported to the place and route tool to complete the physical layout of the digital blocks and to evaluate the overall area and power requirements. The FFT engine occupies a chip area of around 0.48mm$^2$ with 70% density. The estimated power dissipation for the 128MHz 64-point FFT computations at 1.2V supply voltage is 65.38mW.

![Figure 4.13: Layout of tuning blocks](image)

The other digital blocks, a controller, a magnitude calculator, and a comparator are also implemented. The layout of these blocks are displayed in Fig. 4.13. The estimated are consumption is 0.027mm$^2$ with 80% density, and power dissipation for the 128MHz clock speed (with 1.2V supply voltage) is 0.16mW.
Chapter 5

CONCLUSION

A digitally assisted on-chip built-in self-calibration algorithm for cut-off frequency tuning of baseband filters was proposed. It overcomes the signal feedthrough problem of conventional master-slave based frequency tuning techniques. This method uses the relationship between the power values at the DC and at cut-off frequency. It has the advantage of using the existing DSP engines to achieve hardware complexity minimization and the high tuning accuracy.

To verify the tuning algorithm, a 4-th order source-follower based biquad filter was used as a reference baseband low-pass filter. The frequency of the filter was changed with a 7-bit binary weighted capacitor array. The analog continuous filter with 20MHz cut-off frequency for a standard LTE analog front-end application has been designed using 130nm standard CMOS technology at 1.2V supply voltage. Its 3dB frequency tuning range is from 16.2MHz to 24.4MHz.

A digital controller and tuning blocks were synthesized and implemented. The estimated area is 0.027 $mm^2$ with 80% placement density, and the estimated total power consumption with a 128MHz clock speed is 0.16mW. The simulation results show that the errors of the automatic frequency tuning algorithm are within 0.4%.
Appendix A

Verilog-HDL source code for the digital controller

```verilog
module clock_divider(
    clk,
    RST,
    CE,
    ClkDiv8
);

input clk;
input RST;
input CE;
output ClkDiv8;

reg T1, T2, T3;

always @(posedge clk, posedge RST)
```

51
if (RST) begin
    T1 <= 1'b0;
end else if (CE) begin
    T1 <= ˜T1;
end

always @(posedge T1, posedge RST)
    if (RST) begin
        T2 <= 1'b0;
    end else if (CE) begin
        T2 <= ˜T2;
    end

always @(posedge T2, posedge RST)
    if (RST) begin
        T3 <= 1'b0;
    end else if (CE) begin
        T3 <= ˜T3;
    end

assign ClkDiv8 = T10;

endmodule

module counterNFFT(
    clk,
RST,
DVALID,
count
);

parameter CNT_Lnegth = 6; // log(2)NFFT

input clk, RST, DVALID;
output [CNT_Lnegth−1:0] count;

//parameter NFFT = 64;

reg [CNT_Lnegth−1:0] cnt_mag;

always @(negedge clk)begin
    if (RST)
        cnt_mag <= 0;
    else if (DVALID)
        cnt_mag <= cnt_mag + 1;
    else
        cnt_mag <= 0;
end

assign count = cnt_mag;
endmodule

module comparator
(
    clk ,
    DVALID,
    ready ,
    ...
RST,
Din,
Valid_DC,
Valid_f3dB,
mode,
stop,
flag_opt,
flag_calc_done,
LOOP_INC,
LOOP_DEC
);

parameter integer_width = 16;
parameter fraction_width = 10;

parameter MODE_DC = 2'b01;
parameter MODE_f3dB = 2'b10;
parameter MODE_REST = 2'b00;
parameter BIN_NUM_DC = 1;
parameter BIN_NUM_F3dB = 10;
parameter LOOP_CNT_BITS = 5;
parameter NUM_ITER = 2**LOOP_CNT_BITS;
//parameter ERROR_PERCENT = 1.56%
//ERROR_RATE = 1/2^6 *100 in code
parameter HIGH = 1'b1;
parameter LOW = 1'b0;
parameter DIFF_DEFAULT = 26'b01_1111_1111_1111_1111_1111_1111 ;

input clk;
input DVALID;
ninput ready;
input RST;
input signed [integer_width + fraction_width-1 : 0] Din;
input Valid_DC;
input Valid_f3dB;
input [1:0] mode;

output stop;
output flag_opt;
output flag_calc_done;
output LOOP_INC;
output LOOP_DEC;

reg [integer_width + fraction_width-1 : 0] reg_DC;
reg [integer_width + fraction_width-1 : 0] reg_f3dB,
    reg_DC_rms;
wire [integer_width + fraction_width-1 : 0] ERROR_RATE;
reg reg_stop;

always @(negedge clk, posedge RST) begin
    if (RST) begin
        reg_f3dB <= 0;

55
reg_stop <= 0;

end

else if (mode == MODE_DC) begin
    reg_f3dB <= 0;
end

else if (mode == MODE_f3dB) begin
    if (Valid_f3dB)
        reg_f3dB <= Din;
end

else if (mode == MODE_REST) begin
    reg_f3dB <= 0;
end

else if (!DVALID) // unnoticed condition =>
    operation stop
    begin
        reg_f3dB <= 0;
        reg_stop <= 1;
    end
end

assign stop = reg_stop;

reg [integer_width + fraction_width-1 : 0] Dout1, Dout3, Dout4, Dout6, Dout8;
always @(*) begin
    if (RST) begin
Dout1 <= 0;
Dout3 <= 0;
Dout4 <= 0;
Dout6 <= 0;
Dout8 <= 0;
end

else if (Valid_DC) begin

Dout1 <= (Din >> 1);
Dout3 <= (Din >> 3);
Dout4 <= (Din >> 4);
Dout6 <= (Din >> 6);
Dout8 <= (Din >> 8);
end
end

always @(negedge clk, posedge RST) begin

if (RST) begin

reg DC rms <= 0;

end

else if (Valid_DC)

reg DC rms <= (Dout1 + Dout3 + Dout4 + Dout6 + Dout8);

end

reg [integer_width + fraction_width −1 : 0] reg_DIFF_DC_3dB;

reg [integer_width + fraction_width −1 : 0] reg_temp;
always @(posedge clk, posedge RST) begin
    if (RST) begin
        reg_DIFF_DC_3dB = DIFF_DEFAULT;
    end
    else if (DVALID) begin
        if (ITER_CNT > 5'b0_0000)
            if (mode == MODE_f3dB) begin
                reg_temp = (reg_DC_rms - reg_f3dB);
                begin
                    if (reg_temp[integer_width + fraction_width - 1] == 1)
                        reg_unsigned = ~reg_temp + 1'b1;
                    else
                        reg_unsigned = reg_temp;
                end
                reg_DIFF_DC_3dB = reg_unsigned;
            end
    end
end

assign ERROR_RATE = (reg_DC_rms >> 6);
assign wire_DIFF_DC_3dB = reg_DIFF_DC_3dB;

reg reg_LOOP_INC;
reg reg_LOOP_DEC;
reg reg_flag_opt;
always @(posedge clk, posedge RST) begin
  // reset
  if (RST) begin
    reg.flag_opt <= 0;
    regLOOP_INC <= 0;
    regLOOP_DEC <= 0;
  end

  // optimized
  else if (wire_DIFF_DC_3dB < ERROR_RATE) begin
    reg.flag_opt <= HIGH; // tuning process end
    reg_LOOP_INC <= 0;
    reg_LOOP_DEC <= 0;
  end

  // not optimized
  else if (mode == MODE_f3dB) begin
    if (ITER_CNT > 5'b00000) begin
      if (reg_temp[integer_width + fraction_width - 1] == 0) begin
        // positive input
        reg_LOOP_INC <= 1;
        reg_LOOP_DEC <= 0;
        reg_flag_opt <= LOW; // keep going tuning
      end
      else if (reg_temp[integer_width + fraction_width - 1] == 1) begin
        // negative input
        reg_LOOP_DEC <= 1;
      end
    end
  end

end
reg LOOP_INC <= 0;
reg flag_opt <= LOW;  // keep going tuning
end
end

else if(mode != MODE_f3dB) begin
  reg LOOP_INC <= 0;
  reg LOOP_DEC <= 0;
end
  //end

assign LOOP_INC = reg_LOOP_INC;
assign LOOP_DEC = reg_LOOP_DEC;
assign flag_opt = reg_flag_opt;

reg reg_flag_calc_done;
reg [LOOP_CNT_BITS - 1:0] reg_ITER_CNT;
wire [LOOP_CNT_BITS - 1:0] ITER_CNT;
//count iteration number for unoptimized case.
always @(posedge DVALID, posedge RST, posedge
  reg_flag_calc_done) begin
  if(RST) begin
    reg_ITER_CNT <= 0;
    reg_flag_calc_done <= 0;
  end
  else if(DVALID)
if (reg_flag_calc_done)
    reg_ITER_CNT <= 0;
else if (mode==2'b10)
    if (!reg_flag_opt)
        if (reg_ITER_CNT != (NUM_ITER-1))
            reg_ITER_CNT <= reg_ITER_CNT + 1'b1;
else if (reg_ITER_CNT == (NUM_ITER-1))
    begin
        reg_ITER_CNT <= reg_ITER_CNT;
        reg_flag_calc_done <= 1;
    end
end

assign ITER_CNT = reg_ITER_CNT;
assign flag_calc_done = reg_flag_calc_done;

reg reg_AC_edge;
wire w_AC_edge;
always @(posedge ready, posedge RST) begin
    if (RST)
        reg_AC_edge <= 0;
    else if (ready)
        reg_AC_edge <= 1;
end
assign w_AC_edge = reg_AC_edge;
endmodule
module mag_est
(
  clk,
  DVALID,
  RST,
  re_fft,
  im_fft,
  mode,
  MagEstm,
  Valid_DC_Value,
  flag_AC_MODE,
  Valid_f3dB,
  ovf_flag,
  stop,
  calc_end
);

parameter integer_width = 16;
parameter fraction_width = 10;
parameter CNT_Lnegth = 6; // log(2)NFFT
parameter MODE_DC = 2’b01;
parameter MODE_f3dB = 2’b10;
parameter MODE_REST = 2’b00;
parameter BIN_NUM_DC = 1;
parameter FLAG_DC_BIN = BIN_NUM_DC + 1;
parameter BIN_NUM_F3dB = 10;
parameter FLAG_f3dB_BIN = BIN_NUM_F3dB + 1;
input clk;
input DVALID;
input RST;
input [1:0] mode;
input calc_end;
output stop;
output Valid_DC_Value;
output flag_AC_MODE;
output Valid_f3dB;
input signed [integer_width + fraction_width−1:0] re_fft;
input signed [integer_width + fraction_width−1:0] im_fft;
output signed [integer_width + fraction_width−1:0] MagEstm;
output ovf_flag; // ovf flag

// 2s complement signed to unsigned conversion
reg [integer_width + fraction_width−1:0] reg_re_fft, 
    reg_im_fft;
wire [integer_width + fraction_width−1:0] w_re_fft,
    w_im_fft;

always @(*) begin
    if (re_fft[integer_width + fraction_width−1] == 1) // MSB is 1
        reg_re_fft = (~re_fft + 1'b1); // make it positive number
    else if (re_fft[integer_width + fraction_width−1] == 0)
reg_re_fft = re_fft;

if (im_fft[integer_width + fraction_width −1] == 1)
  reg_im_fft = ¬im_fft + 1'b1;
else if (im_fft[integer_width + fraction_width −1] == 0)
  reg_im_fft = im_fft;
end
assign w_re_fft = reg_re_fft;
assign w_im_fft = reg_im_fft;

wire [CNT_Lnegth−1:0] CNT_TRIG;
counterNFFT counterNFF_top (.clk(clk), .RST(RST), .DVALID(DVALID), .count(CNT_TRIG));

// set synthesizable initial condition
reg [integer_width + fraction_width −1 : 0] reg_min_data,
    reg_max_data;
wire [integer_width + fraction_width −1 : 0] w_reg_min_data,
    w_reg_max_data;
reg reg_Valid_DC_Value, reg_Valid_f3dB;
reg reg_stop;

always @(posedge clk, posedge RST) begin
  if (RST) begin
    reg_min_data <= 0;
    reg_max_data <= 0;
    reg_Valid_DC_Value <= 0;
  end
reg_Valid_f3dB <= 0;
reg_stop <= 0;
end

// find min, max
else if(DVALID) begin
  // DC mode
  if (mode == MODE_DC) begin
    if (CNT_TRIG == FLAG_DC_BIN) begin
      reg_Valid_DC_Value <= 1;
      if (w_re_fft >= w_im_fft) begin
        reg_max_data <= w_re_fft;
        reg_min_data <= w_im_fft;
      end
      else if (w_re_fft < w_im_fft) begin
        reg_max_data <= w_im_fft;
        reg_min_data <= w_re_fft;
      end
    end
    else begin // CNT_TRIG != FLAG_DC_BIN
      reg_Valid_DC_Value <= 0;
      reg_max_data <= 0;
      reg_min_data <= 0;
    end
  end
end

// f3dB mode
else if (mode == MODE_f3dB) begin
    if (CNT_TRIG == FLAG_f3dB_BIN) begin
        if (w_re_fft >= w_im_fft) begin
            reg_max_data <= w_re_fft;
            reg_min_data <= w_im_fft;
            reg_Valid_f3dB <= 1;
        end
        else if (w_re_fft < w_im_fft) begin
            reg_max_data <= w_im_fft;
            reg_min_data <= w_re_fft;
            reg_Valid_f3dB <= 1;
        end
    end
else begin // CNT_TRIG != FLAG_f3dB_BIN
    reg_Valid_f3dB <= 0;
    reg_max_data <= 0;
    reg_min_data <= 0;
end
end

// rest mode
else if (mode == MODE_REST) begin
    reg_Valid_DC_Value <= 0;
    reg_Valid_f3dB <= 0;
    reg_max_data <= 0;
    reg_min_data <= 0;
end
// unnoticed condition => operation stop

else begin
    reg_Valid_DC_Value <= 0;
    reg_Valid_f3dB <= 0;
    reg_max_data <= 0;
    reg_min_data <= 0;
    reg_stop <= 1;
end

end

assign Valid_DC_Value = reg_Valid_DC_Value;
assign Valid_f3dB = reg_Valid_f3dB;

assign w_reg_max_data = reg_max_data;
assign w_reg_min_data = reg_min_data;
assign stop = reg_stop;

// Magnitude Calculation of complex number; max + 1/4*min
reg [integer_width + fraction_width−1 : 0] regMagEstm;

//always @(Valid_DC_Value, Valid_f3dB, RST) begin
always @(*)begin
    if(RST)
        regMagEstm <= 0;

    //else if(CNT_TRIG == 2)begin
else if((Valid_DC_Value || Valid_f3dB)
regMagEstm <= (w_reg_max_data + (w_reg_min_data >> 2));
else
    regMagEstm <= 0;
end
assign MagEstm = regMagEstm;
assign ovf_flag = regMagEstm[integer_width + fraction_width - 1];

reg reg_flag_AC_MODE;
always @(negedge Valid_DC_Value, posedge RST, posedge calc_end) begin
    if (RST)
        reg_flag_AC_MODE <= 0;
    else if (calc_end)
        reg_flag_AC_MODE <= 0;
    else if (!RST)
        reg_flag_AC_MODE <= 1;
end
assign flag_AC_MODE = reg_flag_AC_MODE;
endmodule

module controller(
    clk,
    Clk_Div8,
    DVALID,
    ready,
    RST,
    ...
    )
mode,
START,
flag_opt,
flag_calc_done,
Valid_TUNE,
END_TUNING,
SW_CODE_O,
RST_CODE,
LOOP_INC,
LOOP_DEC,
Valid_DC,
flag_DC,
calc_end,
stop
);

parameter integer_width = 16;
parameter fraction_width = 10;
parameter CNT_Lnegth = 10; // log(2)NFFT
parameter MODE_DC = 2’b01;
parameter MODE_f3dB = 2’b10;
parameter MODE_REST = 2’b00;
parameter BIN_NUM_DC = 1;
parameter BIN_NUM_F3dB = 10;

input clk;
input Clk_Div8;
input  DVALID;
input  ready;
input  RST;
input  flag_opt;
input  flag_calc_done;
input  Valid_TUNE;

input  LOOP_INC;
input  LOOP_DEC;
input  RST_CODE;
input  Valid_DC;
input  flag_DC;

output  reg [1:0] mode;
output  START;
input  stop;
output  reg  END_TUNING;
output  calc_end;
output  [LOOP_CNT_BITS−1:0] SW_CODE_O;

parameter  LOOP_CNT_BITS = 5;
parameter  INITIAL_CODE = 2**(LOOP_CNT_BITS−1);
parameter  MAX_VF = 2**LOOP_CNT_BITS−1;
parameter  MIN_VF = 0;

wire  calc_end;
reg  reg_calc_end;
wire w_flag_opt;
wire w_flag_calc_done;
assign w_flag_opt = flag_opt;
assign w_flag_calc_done = flag_calc_done;
always @(posedge clk, posedge RST) begin
  if (RST)
    reg_calc_end <= 0;
  else if (!RST)
    reg_calc_end <= (w_flag_opt || w_flag_calc_done);
end
assign calc_end = reg_calc_end;

// start/stop of tuning loop: Valid_TUNE
always @(posedge clk, posedge RST) begin
  //always @(posedge clk, posedge RST, posedge ready) begin
    if (RST) begin // change to reset mode
      END_TUNING <= 0;
      mode <= MODE_REST;
    end
    else if (calc_end) begin // calculation ended: optimized or not
      END_TUNING <= 1;
      mode <= MODE_REST;
    end
    else if (flag_DC) begin // change to AC mode
      END_TUNING <= 0;
      mode <= MODE_f3dB;
  end
end
else if (Valid_TUNE) begin  // change to DC mode
    END_TUNING <= 0;
    mode <= MODE_DC;
end

else if (!Valid_TUNE) begin  // stop tuning
    END_TUNING <= 1;
    mode <= MODE_REST;
end
end

// generate START signal to FFT

reg unsigned [LOOP_CNT_BITS−1:0] reg_VF_CODE;
always @(posedge ready, posedge RST_CODE)
    begin
        if (RST_CODE)
            reg_VF_CODE <= INITIAL_CODE;
        else if (mode == MODE_f3dB) begin
            if (LOOP_INC) begin
                if (reg_VF_CODE!=MAX_VF)
                    reg_VF_CODE <= (reg_VF_CODE + 1'b1);
            end
            else if (LOOP_DEC) begin
                if (reg_VF_CODE!=MIN_VF)
                    reg_VF_CODE <= (reg_VF_CODE − 1'b1);
            end
            else if (SW_CODE_O == 0)
                reg_VF_CODE <= INITIAL_CODE;
        end
    end
assign SW_CODE_O = reg_VF_CODE;

// generate flag_settled from ready
reg unsigned [2:0] CNT_SW;
reg flag_settled;
always @(posedge clk, posedge RST) begin
    if (RST || !ready) begin
        CNT_SW <= 0;
    end
    else if (ready) begin
        if (CNT_SW != 3'b111)
            CNT_SW <= CNT_SW + 1'b1;
        else if (CNT_SW == 3'b111)
            CNT_SW <= 3'b000;
    end
end

always @(posedge clk) begin
    if (CNT_SW == 3'b111)
        flag_settled <= 1'b1;
    else
        flag_settled <= 1'b0;
end

// generate START signal to FFT
reg reg_START;
reg unsigned [2:0] CNT_START;
always @(posedge clk, posedge RST) begin
  if (RST)
    CNT_START <= 3'b111;
  else if (flag_settled)
    CNT_START <= 3'b001;
  else if (CNT_START == 3'b111)
    CNT_START <= 3'b111;
  else
    CNT_START <= (CNT_START + 3'b001);
end

always @(posedge Clk_Div8, posedge RST, posedge calc_end) begin
  if (RST || calc_end || (mode==MODE_REST))
    reg_START <= 1'b0;
  else if (CNT_START == 3'b010)
    reg_START <= 1'b1;
  else if (CNT_START == 3'b111)
    reg_START <= 1'b0;
end
assign START = reg_START;
endmodule
Bibliography


