Appendix
Time-Domain Modelling of an Electrothermal DAFLL

This appendix provides a time-domain modeling method for the digitally-assisted electrothermal frequency-locked loop (DAFLL). A set of basic Matlab codes have been developed, which can be used in further simulations of various time-domain effects in the behavior of a DAFLL. The electrothermal filter (ETF) of the loop realizes part of its signal chain in the thermal-domain. It will be shown how the analogy between the thermal and electrical domains can be used to develop a network of resistors and capacitors that reproduce the ETF characteristic. This network can be embedded in the time-domain model of the loop, which is in principle an analog mixed-mode system, with continuous-time and discrete-time functions. These can be modeled with the behavioral simulation techniques used for data converters.

A.1 Time-Domain Simulation of an ETF

Chapter 3 provided an analytical modeling of the ETF based on its thermal impedance (see Sects. 3.3 and 3.4). This model mainly describes the frequency-domain characteristics of an ETF by means of numerical calculations, which are rather difficult to use in time-domain simulations. It should be noted that the thermal-impedance model has been extensively used in determining the various sensitivity functions that relate the DAFLL performance to various error sources.

A simpler method for the time-domain modeling of an ETF is to match the frequency-domain response of a low-pass filter, e.g. a network of R and C elements (electrical-domain resistors and capacitors), to that of the ETF. This method was previously used in the development of a *wind sensor* that incorporated electrothermal filters. The step response measured for the wind sensor was fitted to that of a Foster network of 20 parallel RC segments (see Fig. A.1). Here, the wind sensor model has been adopted and its RC values have been modified to map its phase, amplitude

S.M. Kashmiri and K.A.A. Makinwa, *Electrothermal Frequency References in Standard CMOS*, Analog Circuits and Signal Processing,
DOI 10.1007/978-1-4614-6473-0, © Springer Science+Business Media New York 2013

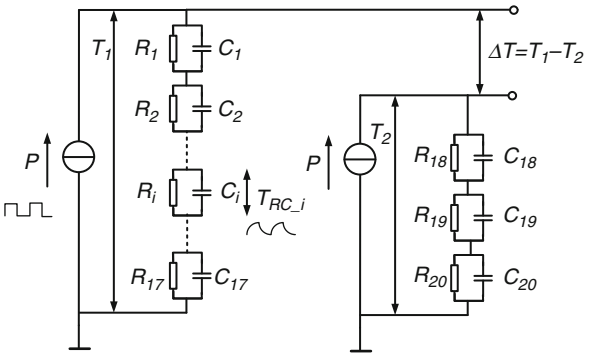

Fig. A.1 The Foster equivalent RC network of the ETF that is used in the time-domain simulations

and step response to that of the ETF, as predicted by the thermal-impedance model (see Fig. 3.16). The resulting R and C values are reported in Table 3.2.

The Foster network shown in Fig. A.1 is excited by the square-wave current sources with amplitude P (ETF heater power). This current (power dissipation) results in an AC voltage (AC temperature variations) at the output of the network (thermopile output). This output is denoted by ΔT, and is the difference between the voltages (temperatures) T_1 and T_2, which represent the *hot* and *cold* junction temperatures of the thermopile, respectively:

$$\Delta T = T_1 - T_2. \tag{A.1}$$

Using super positioning, ΔT can be written as a function of each RC segment's voltage (temperature):

$$\Delta T = \sum_{i=1}^{17} T_{RC\_i} - \sum_{j=18}^{20} T_{RC\_j}. \tag{A.2}$$

The goal is to calculate the ΔT (thermopile output) in the time-domain using the P, R_i and C_i values, and the heater excitation frequency. Simulations were carried out in Matlab (A.3), where numbers are mainly stored and processed in vectors with an integer number of elements. This means that the time-domain simulations can be done at a limited number of instances, which introduces quantization in time. As a result, a minimum time-step of Δt can be defined as the time space between every two calculation instances of ETF output.

In order to calculate T_{RC_i} for each segment in (A.2), the step response of a single RC element (shown in Fig. A.2) is considered. The voltage (temperature) across this parallel RC segment, denoted by $T_{RC}(t)$, when excited by a step current source $i(t) = P{\cdot}u(t)$ is:

$$T_{RC}(t) = P \cdot R \cdot (1 - e^{-t/\tau}). \tag{A.3}$$

where $\tau = R{\cdot}C$.

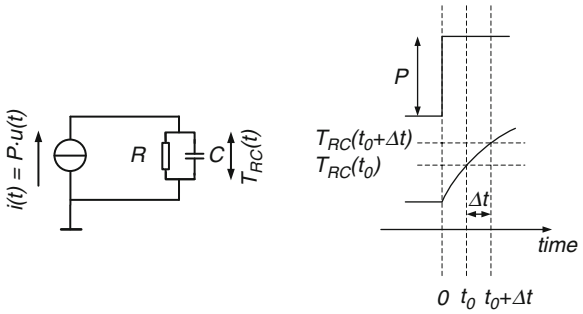

Fig. A.2 Discrete-time values of an RC segment step response

As shown in Fig. A.2, $T_{RC}(t)$ can be calculated at discrete time instances, each Δt seconds apart. The goal is to derive a *generic* equation for $T_{RC}(t)$ that can be plugged into a *generic* Matlab code for calculation of ETF output signal at discrete time instances. To do so, parameter t can be expanded into a discrete equation:

$$t = t_0 + n \cdot \Delta t. \tag{A.4}$$

with t_0 as an initial moment of time and n as an integer. In the next step, $T_{RC}(t)$ should be calculated at two consecutive instances: t_0 and $t_0 + \Delta t$ (see Fig. A.2). This allows $T_{RC}(t_0 + \Delta t)$ to be derived as a function of $T_{RC}(t_0)$, which enables the compilation of a *generic* discrete-time equation for T_{RC}(n). Using (A.3):

$$T_{RC}(t_0 + \Delta t) = P \cdot R \cdot (1 - e^{\frac{-(t_0+\Delta t)}{\tau}}). \tag{A.5}$$

which can be modified by adding and subtracting an extra term:

$$T_{RC}(t_0 + \Delta t) = P \cdot R \cdot (1 - e^{\frac{-(t_0+\Delta t)}{\tau}}) + P \cdot R \cdot e^{\frac{-\Delta t}{\tau}} - P \cdot R \cdot e^{\frac{-\Delta t}{\tau}}. \tag{A.6}$$

By re-arranging:

$$T_{RC}(t_0 + \Delta t) = P \cdot R \cdot (1 - e^{\frac{-t_0}{\tau}}) \cdot e^{\frac{-\Delta t}{\tau}} + P \cdot R \cdot (1 - e^{\frac{-\Delta t}{\tau}}). \tag{A.7}$$

The first term on the right hand side of (A.7) is equal to $T_{RC}(t_0)$ and thus:

$$T_{RC}(t_0 + \Delta t) = T_{RC}(t_0) \cdot e^{\frac{-\Delta t}{\tau}} + P \cdot R \cdot (1 - e^{\frac{-\Delta t}{\tau}}). \tag{A.8}$$

Using (A.4), (A.8) can be expanded into an equation for discrete samples $n = 1$, 2, 3, ... such that:

$$T_{RC}(n) = T_{RC}(n-1) \cdot exv + P \cdot R \cdot (1 - exv). \tag{A.9}$$

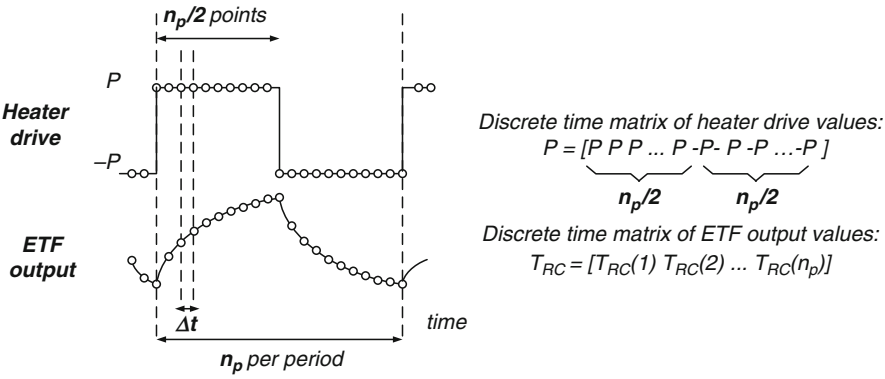

Fig. A.3 An illustration of the ETF continuous and discrete-time input and output waveforms, and matrixes that could be programmed in Matlab to hold the discrete-time values

where

$$exv = e^{\frac{-\Delta t}{\tau}}. \tag{A.10}$$

Within a complete period of the ETF heater drive defined by $T_{\text{drive}} = 1/f_{\text{drive}}$, a total number of n_p discrete time instances can be considered (see Fig. A.3). With this assumption, the value of Δt can be calculated as:

$$\Delta t = \frac{T_{drive}}{n_p}. \tag{A.11}$$

Furthermore, the heater drive signal $P(n)$ can be written as:

$$P(n) = \begin{cases} P & for \quad 1 \leq n < \dfrac{n_p}{2} \\ -P & for \quad \dfrac{n_p}{2} \leq n < n_p \end{cases}. \tag{A.12}$$

where + P and –P result in a differential thermopile output signal (see Figs. 3.7 and 3.8 for fully differential thermopile structures).

Using (A.2), (A.9), and (A.10), a *generic* equation can be developed, which calculates the ETF output signal within a period of the heater drive:

$$\begin{aligned} \underbrace{\Delta T(n)}_{n=1,2,..,n_p} &= \sum_{i=1}^{17} T_{RC\_i}(n-1) \cdot exv_i + P(n) \cdot R_i \cdot (1 - exv_i) \\ &\quad - \sum_{j=18}^{20} T_{RC\_j}(n-1) \cdot exv_j + P(n) \cdot R_j \cdot (1 - exv_j). \end{aligned} \tag{A.13}$$

where

$$exv_k = \frac{-\left(\frac{T_{drive}}{n_p}\right)}{R_k \cdot C_k}. \tag{A.14}$$

The RC values reported in Table 3.2 for the ETF model can be loaded into matrixes to be used in a *generic* Matlab code that calculates the ETF output for a period of the heater drive:

$$\begin{aligned} R &= [R_1\ R_2\ R_3\ \ldots\ R_{17}\ R_{18}\ R_{19}\ R_{20}] \\ C &= [C_1\ C_2\ C_3\ \ldots\ C_{17}\ C_{18}\ C_{19}\ C_{20}]. \end{aligned} \tag{A.15}$$

The result is the following Matlab code that can be extended to any number of periods by extending the vectors that define the heater drive input:

```
tau = R.*C;                 % time constant vector
np = 100;                   % number of points per heater drive period
fdrive = 100e3;             % heater drive frequency = 100kHz in this
                            case
Tdrive = 1/fdrive;          % heater drive period
delta_t = Tdrive/np;        % time step
power = 2.5e-3;             % heater power = 2.5mWin this case
P(1:np/2) = power;          % first half period of heater drive
P(np/2+1:np) = -power;      % second half period of heater drive
s_tp = 24*0.5e-3;           % number of thermocouples times Seebeck
                            coefficient of one
pol(1:17)=1;                % summation polarity for hot junction side
                            of network
pol(18:20)=-1;              % summation polarity for cold junction
                            side of network
exv = exp(-delta_t./tau); % realization of exv_k in equation A.14
dT((1:20),1) = 0;
for i=2:length(P);          % This loop runs per calculation point of
                            the ETF output
  dT(:,i) = dT(:,(i-1)).*exv +
   P(i)*R.*(1-exv);         % realization of sigma functions in A.13
  V_ETF(i) = pol*
   dT(:,i)*s_tp;            % translation to voltage and summation
end;                        % At the end, the ETF output signal is
                            stored in V_ETF matrix
```

Simulation results for the ETF output signal based on this code, and within a few periods of the heater drive at 100 kHz, and at two levels of the heater power are shown in Fig. 3.35. This code will be embedded into another code that builds the complete time-domain model of the DAFLL. This will be discussed in the following section.

A.2 Time-Domain Simulation of the DAFLL

The DAFLL system architecture was described in Chap. 4, as depicted by the block-diagram of the loop shown in Fig. 4.25. This loop includes a phase-domain ΔΣ modulator (PDΔΣM), a 12-bit digitally-controlled oscillator (DCO), and a digital integrator. The block-diagram that formed the basis on which the Matlab model of the loop is developed and shown in Fig. A.4.

The output of the RC network representing ETF is fed to the PDΔΣM, where it is multiplied by the output of a phase DAC. The result is integrated by the continuous-time integrator of the modulator, *V_integ_PDSD*, which drives the quantizer. The PDΔΣM, the DAFLL's digital integrator, and the DAC driving the VCO, are sampled at frequency f_s. The DAC has a total number of bits $= N_{\mathrm{DAC}}$ and a reference voltage $= V_{\mathrm{ref}}$. The VCO has a voltage-to-frequency gain $= K_{\mathrm{VCO}}$. The output frequency of the VCO updates a parameter in the loop: f_{VCO}, from which f_{drive} of ETF and f_s, are extracted.

Matlab is a discrete-time environment in which the values of various signals in the loop can be processed and stored in vectors. The loop is simulated for a limited number of sampling periods $= N_{\mathrm{sample}}$. Each sampling period includes an integer number of ETF heater drive periods $= 2 \times N_{\mathrm{fs}}$, where N_{fs} represents the number of half-periods. Per half-period of ETF drive signal, there are a limited number of time instances $= N_{\mathrm{points}}$. The continuous analog signals within the loop, e.g. the ETF output, *V_ETF*, and *V_integ_PDSD*, are only calculated within these time instances. Therefore, the time interval among each two calculation points, denoted by Δt, is the smallest time interval in the whole calculations. In every simulation run of DAFLL, the total number of calculation points is equal to:

$$N_{\mathrm{total}} = \underbrace{N_{sample}}_{\text{Total number of samples}} \times \underbrace{2 \times N_{fs}}_{\substack{\text{Total number of ETF drive periods} \\ \text{per loop sample}}} \times \underbrace{N_{\mathrm{points}}}_{\substack{\text{Total number of time instances in} \\ \text{one half period of ETF drive}}} \tag{A.16}$$

The minimum time interval Δt of the loop is updated every time the DCO output frequency is updated. The value of Δt remains the same (for the calculation of the intermediate points) until the next sampling moment of the loop, when the DCO frequency is updated. Considering a division by 16 between f_{VCO} and f_{drive} (see Fig. A.4), the value of Δt can be calculated from:

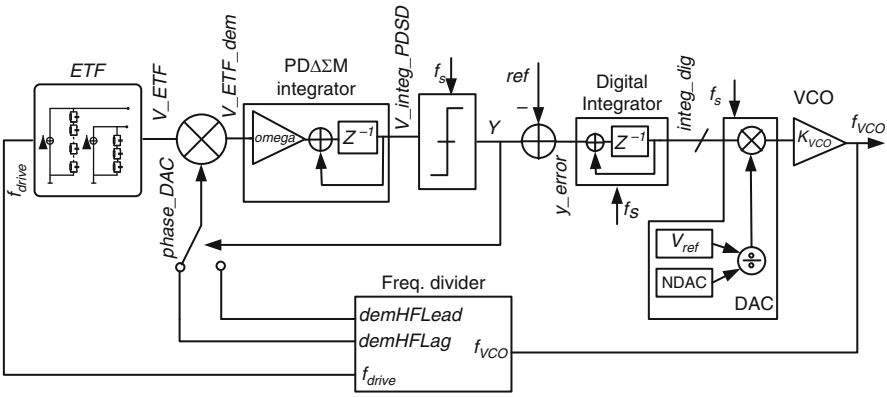

Fig. A.4 Block-diagram of DAFLL used for Matlab simulations

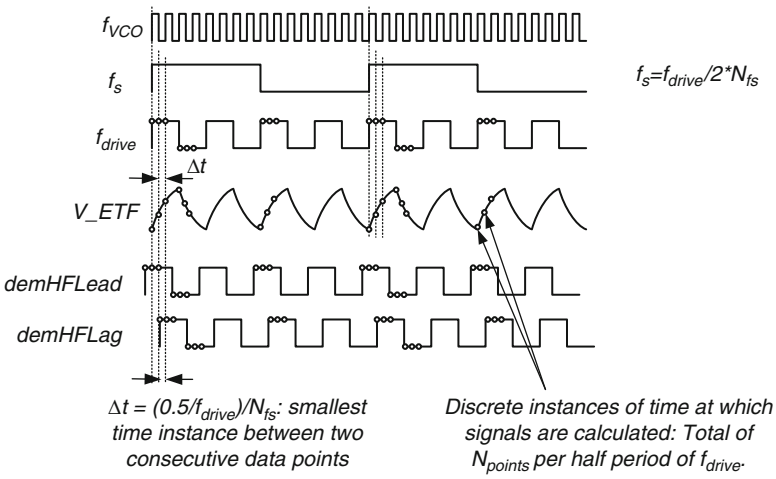

Fig. A.5 Some of discrete-time instances within DAFLL continuous-time signals, and the time interval Δt between them

$$\Delta t = 1/(16 \cdot f_{VCO} \cdot 2 \cdot N_{\text{points}}). \tag{A.17}$$

All the periodic signals within the loop can be defined in matrixes with an integer number of elements. For instance: f_{drive}, the signal that drives the ETF heater, can be defined by a vector with an N_{total} number of elements [see (A.16)]. In this vector, the value of elements toggle periodically between $+P$ and $-P$ (P is the heater power) at every N_{points} element. The phase DAC reference signals can also be defined in the same way. These are equivalent to the f_{drive} vector, with the difference being that their elements toggle between +1 and −1. The lead and lag phase shifts of the phase DAC references, in reference to f_{drive}, can be simply implemented by shifting the vector elements forwards and backwards (see Fig. A.5) by a total number of:

$$N_{shift} = 2 \cdot N_{\text{points}} \cdot \frac{phase\_ref}{360^{\circ}}. \tag{A.18}$$

where *phase_ref* can for instance be equal to ±45°.

Once the signals in the DAFLL system block diagram are pre-defined, an algorithm (see Fig. A.6) can be defined that runs for the total number of samples and calculates the intermediate points for the internal signals. This algorithm involves three loops. The *most inner loop* runs per half-period of the ETF heater drive signal. This is a *for loop* in Matlab that runs for N_{points}. Within this loop, the ETF output signal $V_{\text{ETF}}(n)$, and the result of its multiplication with the phase DAC output, $V_{\text{ETF\_dem}}(n)$, are calculated. Furthermore, this loop calculates the PDΔΣM's integrator output, *V_integ_PDSD*. Consequently, the integrator's continuous-time output signal is calculated every time one of the intermediate ETF output signal points is calculated. This reproduces the continuous-time integrator's signal in an over-sampled fashion in regard to the modulator's sampling frequency.

An ideal continuous-time integrator has a frequency-domain transfer:

$$H(s) = \frac{\omega_0}{s}. \tag{A.19}$$

in which ω_0 is its unity-gain frequency (Fig. A.7a), i.e. the frequency at which the amplitude response of the integrator reaches the 0 dB gain point. In the time-domain, the output of such integrator will be a ramp in response to a unit step input (Fig. A.7b). After Δt seconds the integrator output growth (from zero initial condition) is:

$$\Delta V_{out}(\Delta t) = \omega_0 \cdot \Delta t. \tag{A.20}$$

In the discrete time simulation environment of Matlab, a calculation of the continuous-time integrator output at the *ith* discrete sample can be written as:

$$V_{out}(i) = V_{out}(i-1) + v_{in}(i) \cdot \omega_0 \cdot \Delta t. \tag{A.21}$$

For the DAFLL, Δt can be calculated from (A.17).

The *intermediate loop* of the DAFLL algorithm (see Fig. A.6) runs for the f_{drive} half periods within one sampling period of the loop. Therefore, the loop runs for $2 \cdot N_{\text{fs}}$ times between every two samples. The *most outer loop* runs for the total number of samples N_{sample}. Within this loop, the PDΔΣM quantizer and the phase DAC outputs are updated. Also within the same loop, the bitstream of the modulator, *Y* (see Fig. A.4), is compared with the DAFLL phase reference, *ref*. This is a consecutive set of one's and zero's at a rate of $f_s/2$ and represents a 90° phase shift in the ETF (for *phase_ref* values of ±45). The result of this comparison is integrated by the digital integrator, whose output then updates the DAC input

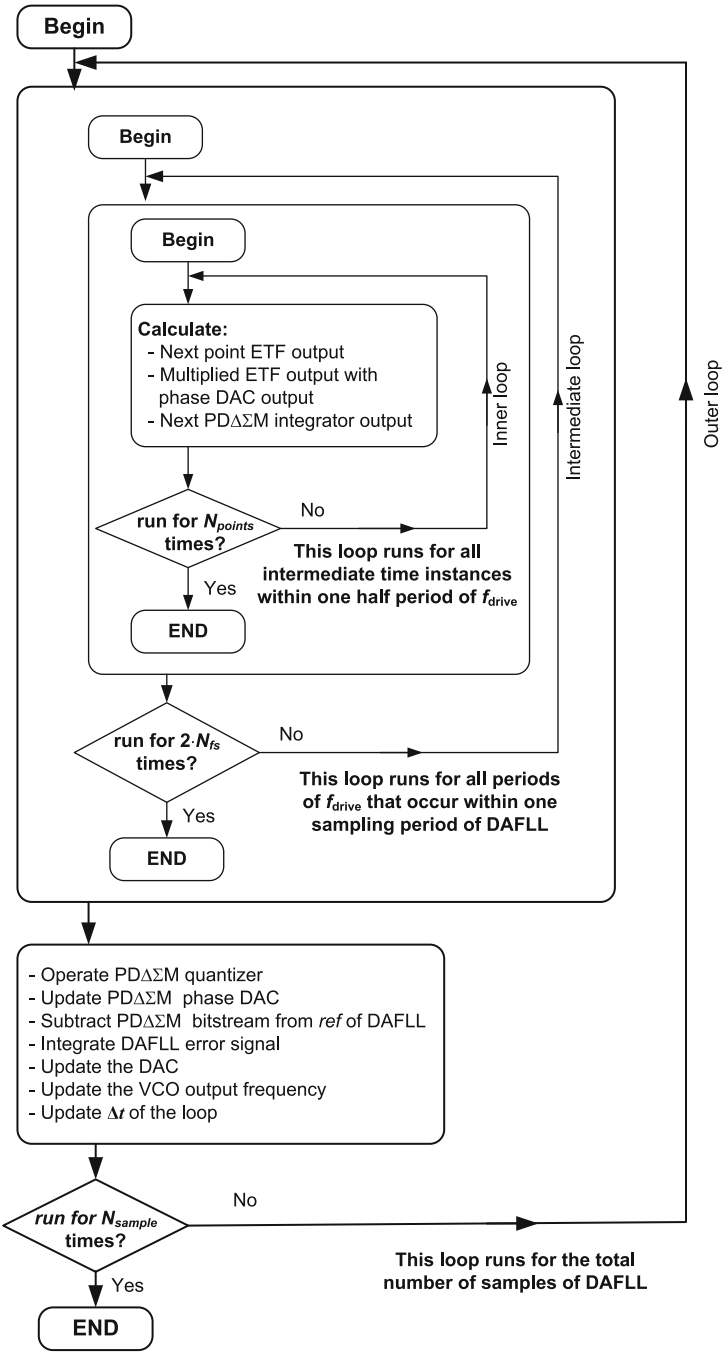

Fig. A.6 The DAFLL time-domain calculation algorithm

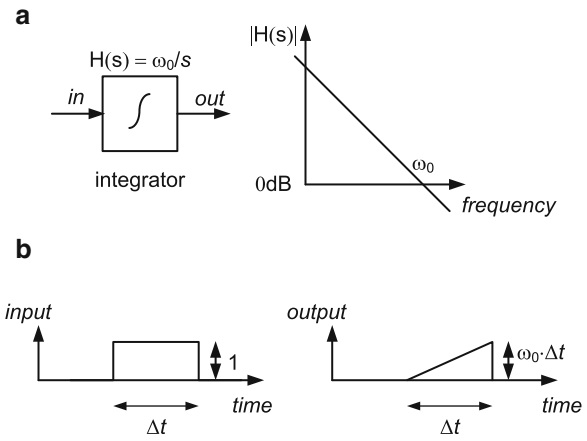

Fig. A.7 A continuous-time integrator: (**a**) frequency-domain response; (**b**) step-response

every time the outer loop runs. This then changes the VCO output frequency, f_{VCO}, which updates f_{drive} and Δt eventually.

Based on the proposed algorithm a Matlab code can be devised which forms the basis for the time-domain simulations of the DAFLL shown in Fig. A.4. This code can be further expanded for the addition of non-idealities or the study of loop dynamics in the time domain. A few simulation results, such as the loop startup and step response, as well as the output spectrum of the PDΔΣM and digital integrator, were shown in Chap. 4 and in Figs. 4.26 and 4.27. The generic Matlab code that implements the proposed algorithm is as follows:

```
% Initial definitions
s_tp = 24*0.5e-3;          % Thermopile sensitivity
kvco = 0.9e6;              % VCO sensitivity [Hz/V]
Nsample = 8192;            % Number of samples of the loop
Nfs = 2;                   % Number of ETF drive periods per sample
Nperiods = 2*Nfs*Nsample;% Number of half periods of ETF drive to be
                           simulated
Npoints = 24;              % Number of intermediate points in every
                           half period of fdrive
f = 100e3;T = 1/f;         % Initial condition for ETF drive frequency
fvco = f*ones(1,Nsample);% Define a vector that holds fVCO values
vco_in = zeros(1,Nsample); % Define a vector holding VCO input signal
fs = f/Nfs;                %Sampling frequency of DAFLL
fT = 300e3;                % 0-dB frequency of the continuous-time
                           integrator [Hz]
NDAC=12;Vref=4;            % DAC definition
```

```
demHFLead = zeros (1, Nperiods*Npoints); % phase DAC's Lead feedback
                                         signal
demHFLag = zeros (1, Nperiods*Npoints); % phase DAC's Lag feedback
                                         signal
V_integ_PDSD =
 zeros(1, Nperiods*Npoints);             % PDSD's integrator signal
y = ones(1,Nsample);                     % PDSD's bitstream

% DAFLL reference bit stream (90 degrees phase), and error signal of
the loop
yerror = zeros(1,Nsample);
diginteginit = 256;       % The initial condition at which the digi-
                          tal integrator starts
integ_dig = diginteginit*
 ones(1,Nsample);         % error signal pre-allocation
ref = zeros(1,Nsample);   % DAFLL reference signal
ref(3:2:Nsample)=1;       % DAFLL reference signal

% Definition of the ETF drive square-wave
range = (1:2*Npoints: Nperiods*Npoints); % initialization of
                                           vectors
fdrive = zeros (1, Nperiods*Npoints);  % initialization of vectors
fdrive(range) = 1;                     % initialization of vectors
cycle = [ones(1,Npoints),
 zeros(1,Npoints)];                    % initialization of vectors
fdrive = filter
 (cycle,1,fdrive);                     % initialization of vectors
fdrive = 2*fdrive - 1;
demHFLead = circshift (fdrive, [1 -Npoints/4]); % Definition of the
                                                  Lead feedback
demHFLag = circshift (fdrive, [1 Npoints/4]);  % Definition of the
                                                 lag feedback
phase_DAC = demHFLead;                          % Initial phase DAC
                                                  value
P = 25/(2*1.2e3);         % The heater drive power [W]
R = etf_model(:,2);       % The vector including ETF's Foster net-
                          work resistances
C = etf_model(:,3);       % The vector including ETF's Foster net-
                          work capacitances
pol(1:17)=1;              % polarity of summation for hot junction
                          side of network
pol(18:20)=-1;            % polarity of summation for cold junction
                          side of network
tau = R.*C;               % Time constant vector
V_ETF = zeros (1,Nperiods*Npoints);     % pre allocating thermopile
                                          output signal
V_ETF_dem = zeros (1,Nperiods*Npoints); % pre allocating demodulated
                                          ETF output

ix = 1;
% Assignment of startup values so that the loop can start working
delta_t = T/(2*Npoints);
```

```
exv = exp(-delta_t./tau);
dT((1:20),1) = 0;
% An initial condition for the continuous time integrator's cut off
frequency
omega = 2*pi*fT/fs/(2*Npoints*Nfs);
% The DAFLL loop simulation
% The outer loop runs per loop sample
for sdx = 2:Nsample
% The mid loop runs per half period of fdrive in one sampling period
of sigma-delta
  for jx = 2:2*Nfs+1
    % The inner loop runs per intermediate calculation steps
    for px = 1:Npoints
    % Calculating the ETF output value for the current intermediate
    point
    ix = ix + 1;
    ix = min(Npoints*2*Nfs*Nsample,ix);
    dT(:,ix) = dT(:,(ix-1)) .*exv + P*fdrive(ix)*R.*(1-exv);
    V_ETF(ix) = pol*dT(:,ix)*s_tp;
    % multiply ETF output with phase DAC feedback phase
    V_ETF_dem(ix) = V_ETF(ix)*phase_DAC(ix);
    % phase-domain delta sigma's integrator value evaluation
    V_integ_PDSD(ix) = V_integ_PDSD(ix-1) + omega*V_ETF_dem(ix);
    end
  end
   % Quantizer
   y(sdx) = sign(V_integ_PDSD(ix));
   % Phase DAC
   if (y(sdx)==1)
     phase_DAC = demHFLead    else
     phase_DAC = demHFLag   end;
  % DAFLL phase summation node: subtraction, error signal generation
  yerror(sdx) = ref(sdx)- (((y(sdx)*-1)+1)/2);
  %subtract PDSD bitstream & phase ref
  integ_dig(sdx) = integ_dig(sdx-1) + yerror(sdx-1);  % digital integrator

  % Update the DCO DAC value
  vco_in(sdx) = 4*(integ_dig(sdx))*(Vref/2^(NDAC-1));
  % Update the VCO frequency
  fvco(sdx) = max(kvco*vco_in(sdx),100e3);
  % Feedback the VCO output to the ETF, i.e. close the DAFLL loop
  feedback
  f = fvco(sdx)/16;
  T = 1/f;
  fs = f/Nfs;
```

```
    delta_t = T/(2*Npoints); % update the time stamp of the loop based
    on latest fVCO
    exv = exp(-delta_t./tau);
    % Update the PDSD's continus time integrator cut off frequency
    omega1 = 2*pi*fT/fs/(2*Npoints*Nfs);
  end
```

Summary

This book investigates the concept of generating accurate on-chip frequencies based on the thermal properties of silicon. The work presented here shows that electrothermal (thermal-diffusivity-based) frequency references are feasible in standard CMOS processes. Furthermore, the possibility of scaling such references into the more modern CMOS processes is studied. The improvements achieved in the performance of the scaled reference show that such frequency references strongly benefit from process scaling.

A frequency reference is an indispensable part of most electronic systems. In recent years, a lot of effort has been devoted to the replacement of quartz crystal oscillators with integrated solutions. The main motivations have been the reduction in size and cost as well as the increase in reliability of electronic circuits. Chapter 2 provides an overview of the recent developments in the implementation of silicon-based frequency references. In this survey, various types of oscillators including MEMS, RC, LC, ring, and electron-mobility-based are studied. The MEMS and LC oscillators have been commercialized, so far. These references achieve stabilities of < 1 ppm and 50 ppm, respectively, enabling them to compete with the quartz crystal oscillators.

Apart from some earlier thermal oscillators, the thermal properties of silicon have received much less attention in generation of on-chip frequencies. These concepts are discussed in Chap. 3. In principle, the heat generated in a silicon substrate diffuses at a defined rate which is determined by the thermal diffusivity of silicon, D. This forms the basis for the thermal delay lines, around which the thermal oscillators are made. A thermal delay line embeds a heater at a distance s from a temperature sensor within a silicon substrate. Its delay is then governed by D and s. The former is a stable property of IC-grade silicon, while the latter is defined by lithography. The early thermal oscillators had large levels of output jitter and high levels of power consumption. These were mainly due to heat losses in the substrate, which resulted in very small signals at the output of their temperature sensors.

Chapter 3 further describes how a thermal delay line can be extended into a more complex structure called an electrothermal filter (ETF). An ETF realizes a heater

S.M. Kashmiri and K.A.A. Makinwa, *Electrothermal Frequency References in Standard CMOS*, Analog Circuits and Signal Processing,
DOI 10.1007/978-1-4614-6473-0,

(e.g. a resistor) at a close distance s (less than 25 μm) from a relative temperature sensor (e.g. a thermopile), and thus can be realized in standard CMOS processes. This structure behaves like a low-pass filter, whose well-defined phase response is determined by D and its geometry. Measurements on ETFs have shown tolerances in the order of 0.1%. Chapter 3 describes how embedding an ETF into a feedback loop with a narrow noise-bandwidth solves the unacceptable jitter performance of the previous thermal oscillators. Such loop is called an electrothermal frequency-locked loop (FLL). An FLL locks the output frequency of a VCO to the accurate phase shift of the ETF. As a result, this frequency is determined only by the ETF and thus not affected by the VCO's tolerances and drifts. Earlier implementations of electrothermal FLLs are investigated in Chap. 3. Since the output frequency of an FLL follows the same $T^{-1.8}$ temperature dependence of D, these FLLs were aimed as accurate temperature-to-frequency converters.

The temperature-dependent output frequency of FLLs showed promising levels of untrimmed device-to-device frequency spread, which was in the order of ±0.25% over the industrial temperature range. Chapter 3 concludes that such levels of inaccuracy imply that a temperature-compensated electrothermal FLL should form a suitable basis for the realization of accurate electrothermal frequency references.

There were challenges associated with the integration of early FLLs. These required large off-chip capacitors in order to implement their narrow noise-bandwidths, making their CMOS integration rather difficult. Therefore, Chap. 4 proposes a digitally-assisted electrothermal FLL (DAFLL), in which the narrow noise-bandwidth of the loop is implemented by a digital filter. A DAFLL incorporates a digital phase detector (a phase-domain ΔΣ modulator) that digitizes the phase shift of an ETF, a digital phase reference, a phase summation node, and a digitally-controlled oscillator (DCO). An implementation in a standard 0.7μm CMOS achieved an output frequency with an untrimmed device-to-device spread of ±0.3% (3σ) from −55°C to 125°C. This level of inaccuracy is comparable to the early FLLs. Furthermore, the output frequency of the loop followed the same $T^{-1.8}$ temperature dependence, associated with the thermal diffusivity of silicon. The DAFLL provided a solution for the integration challenge of electrothermal FLLs and therefore was adopted as the foundation of the electrothermal frequency references proposed in this book.

Chapter 5 describes how an electrothermal frequency reference can be realized by temperature-compensating a DAFLL. This was done by measuring the temperature of the die using an on-chip band-gap temperature sensor and injecting the digital temperature information into the loop. The result was the first fully-integrated electrothermal (thermal-diffusivity-based) frequency reference, realized in a standard 0.7 μm CMOS process. The reference has an output frequency of 1.6 MHz, dissipates 7.8 mW, and achieves an absolute inaccuracy of ±0.1% over the military temperature range (−55°C to 125°C) with a single room-temperature trim. Its worst-case temperature coefficient of ±11.2 ppm/°C allows for trimming without temperature stabilization, which simplifies the trimming procedure and thus reduces trimming costs.

Chapter 6 describes the scaling of an electrothermal frequency reference into a more modern CMOS process. The performance of an electrothermal frequency reference is mainly determined by its ETF. The accuracy and phase-frequency characteristic of an ETF are functions of its geometry. Scaling the ETF enables trade-offs regarding the accuracy, power consumption, output frequency and jitter. By scaling the ETF and adopting a more modern CMOS technology, a scaled electrothermal frequency reference was developed in a standard 0.16 µm CMOS process. The ETF's critical dimension *s* (the distance between heater and temperature sensor) was scaled from 24 to 4.7 µm, which results in 5.5× more SNR and allows for a 10× higher frequency. The improved lithographic accuracy of the 0.16µm process (compared to the previous 0.7 µm process) maintains its accuracy. The resulting frequency reference has an output frequency of 16 MHz (10× higher compared to previous work), occupies an area of 0.5 mm^2 (12× smaller), dissipates 2.1 mW (3.7× less), and has an rms output jitter of 45 ps (7× less). Measurements on 24 devices show that the reference maintains the ±0.1% level of inaccuracy (from −55°C to 125°C) achieved by the previous work with a single room-temperature trim.

About the Authors

S. Mahdi Kashmiri was born on March 21st, 1980 in Tehran, Iran. He received his B.Sc. degree in electrical engineering from Tehran University, Tehran, Iran, in 2001, and his M.Sc. degree in microelectronics (*cum laude*) from Delft University of Technology, Delft, The Netherlands in 2006. In April 2012, he received his Ph.D. degree from the same university for his work on electrothermal frequency references in standard CMOS. From June 2001 to August 2004, he was a system design engineer at Parman Co. Tehran, Iran, where he worked on the development of a SDH fiber optics telecommunication system. From September 2005 to October 2006, he was an intern at the mixed-signal circuit and systems group of Philips Research Laboratories (currently NXP research), Eindhoven, The Netherlands, where he worked on a wide-band continuous-time sigma-delta modulator. Since October 2010 he has been with the precision systems group of Texas Instruments Incorporated, Delft, The Netherlands (formerly National Semiconductor Corporation). His research interests include the analog and mixed-signal integrated circuits, precision analog systems, and data converters.

Mr. Kashmiri received the 2009 Young Scientist Award of the European Solid-State Circuits Conference (ESSCIRC).

Kofi A.A. Makinwa received the B.Sc. and M.Sc. degrees from Obafemi Awolowo University, Nigeria in 1985 and 1988 respectively. In 1989, he received the M.E.E. degree from the Philips International Institute, The Netherlands and in 2004, the Ph.D. degree from Delft University of Technology, The Netherlands.

From 1989 to 1999, he was a Research Scientist with Philips Research Laboratories, Eindhoven, The Netherlands, where he worked on interactive displays and on front-ends for optical and magnetic recording systems. In 1999, he joined Delft University of Technology, where he is now an Antoni van Leuwenhoek Professor in the Faculty of Electrical Engineering, Computer Science and Mathematics. His main research interests are in the design of precision analog circuitry, sigma-delta modulators, smart sensors and sensor interfaces. This has resulted in 4 books, 18 patents and over 150 technical papers.

S.M. Kashmiri and K.A.A. Makinwa, *Electrothermal Frequency References in Standard CMOS*, Analog Circuits and Signal Processing,
DOI 10.1007/978-1-4614-6473-0,

Kofi Makinwa is on the program committees of the European Solid-State Circuits Conference (ESSCIRC) and the Advances in Analog Circuit Design (AACD) workshop. He has also served on the program committee of the International Solid-State Circuits Conference (ISSCC), as a guest editor of the Journal of Solid-State Circuits (JSSC) and as a distinguished lecturer of the IEEE Solid-State Circuits Society (2008–2011). He is a co-recipient of several best paper awards: from the JSSC, ISSCC, Transducers and ESSCIRC, among others. In 2005, he received a Veni Award from the Netherlands Organization for Scientific Research and the Simon Stevin Gezel Award from the Dutch Technology Foundation. He is an alumnus of the Young Academy of the Royal Netherlands Academy of Arts and Sciences and an elected member of the IEEE Solid-State Circuits Society AdCom, the society's governing board.

Index

S.M. Kashmiri and K.A.A. Makinwa, *Electrothermal Frequency References in Standard CMOS*, Analog Circuits and Signal Processing,
DOI 10.1007/978-1-4614-6473-0,