Passive Switched-Capacitor based Filter Design, Optimization, and Calibration for Sensing Applications

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

MARTIN STURM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

PROF. RAMESH HARJANI

January, 2013

Acknowledgements

I would like to express my gratitude to my advisor, Prof. Ramesh Harjani, whose guidance and friendship has made this thesis possible and influenced my life over the last six years. I am grateful for his support and enthusiastic interaction throughout my PhD. The parties he and Savita have hosted continue to have a lasting impact on me.

Along with Prof. Harjani, the analog design lab has exposed me to many indispensable characters. I would like to thank the senior members who introduced me to the lab and guided me through my first two years: Narasimha Lanka, Kin-Joe Sham, and Satwik Patnaik. Their enthusiam for fine dining provided many hours of enjoyment. I am ever-thankful to have worked with Bodhisatwa Sadhu on the CRAFT project. He was always there to keep us focused and on-schedule while I continuously wandered into analysis and optimization. I am thankful to Jaehyup Kim for his layout tips and enjoyed discussing his project, as it was such a new topic for both of us. Taehyoun Oh was a wonderful friend who has shown me what can be accomplished with an indomitable work ethic. I am also grateful to Mohammad Elbadry, Sudhir Kudva, Sachin Kalia, and Ashutosh Mehra for their friendship and making the lab enjoyable. Rakesh Palani has often shown me completely new ways to look at analog design, and I thank him for that.

I am thankful for the independent work ethic and real-world design knowledge I

gained as part of the University of Minnesota Solar Vehicle Project as an undergraduate.

I would like to express my sincere thanks to Prof. Rhonda Franklin, Prof. James Leger, and Prof. Marvin Marshak for serving on my final defense committee.

I am grateful to MOSIS, TAPO and IBM for providing fabrication of prototype chips. I would also like to express my gratitude toward the funding agencies: Center for Circuits and Systems Solutions (C2S2), Semiconductor Research Corporation (SRC), the U.S. Army Research Laboratories (ARL), and Defense Advanced Research Projects Agency (DARPA) who made this research possible.

I would like to thank my brothers, Scott and Eugene, who put up with me as a roommate while I kept odd hours working toward deadlines. I would like to thank my fiancé, Katie, who has seen me through the last year and a half of my PhD. Finally, I am grateful to my parents for everything they have give me.

Dedication

Dedicated to my parents,
and my grandparents

Abstract

This thesis discusses the limitations of wideband software defined radios (SDR). Spectrum sensing is identified as an important aspect required of SDR based cognitive radios. Several architectures and implementations are reviewed. An RF sampler followed by analog signal processing is identified as a critical block enabling low-power wideband digitization.

Passive switched-capacitor charge-domain processing is introduced. Its implementation simplicity and lack of active power consumption are enticing. The effect of kT/C noise is analyzed in detail. Linear and nonlinear computation errors are modeled, and circuit techniques for their reduction are developed. Simulations are used to optimize power and computational dynamic range.

For spectrum sensing, the design of CRAFT (Charge Re-use Analog Fourier Transform): an RF front-end channelizer for wideband software defined radios based on a 16-point analog-domain FFT is described. The design relies on charge re-use to process a 5GS/s input with an average output SNDR of 47dB, and consumes only 12.2pJ/FFT conversion (3.8mW). These numbers represent orders of magnitude improvement on the work reported previously in literature. The thesis also discusses the system-level modeling and mitigation of circuit non-idealities in CRAFT. These design principles enable this implementation to achieve a large dynamic range even at high speeds. Additionally, these techniques can be easily extended to improve the performance of other passive switched-capacitor designs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Wireless Ubiquity

Wireless technology has permeated our world and the way we interact with it. In 2011, the number of cell-phones in use by the world's 6.9 billion people totaled 6.0 billion devices. With convenience in mind, more and more functions are being implemented wirelessly (Fig. 1.1). The expanding number and complexity of applications for wireless is obscured from the user by the increasing amounts of wireless functionality integrated into single devices. Smartphones are an example of this and currently contain up to ten independent radio designs.

1.2 Spectral Congestion

Since wireless signals propagate using electromagnetic waves, interference between devices on the same frequency can be a big concern. Due to this, the frequency spectrum has been divided and allocated to specific applications. However, the growth of wireless communication has exposed the flaws of the frequency allocation system. Spectral

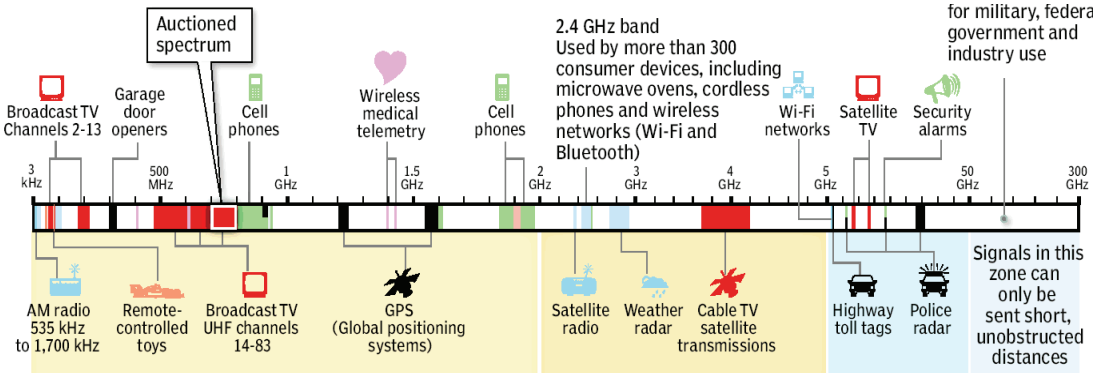


Figure 1.1: Frequency spectrum allocation in the United States (Source: New America Foundation, MCT, howstuffworks.com; Graphic: Nathaniel Levine, Sacramento Bee)

congestion has caused concern as to how it may stifle further growth of the industry [2].

Spectral congestion arises primarily due to less than ideal frequency planning as well as the rigid nature of the spectrum licensing process. An example spectrum licensing diagram by the Federal Communications Commission, USA, is shown in Fig. 1.3 [3]. Currently, all frequencies have been allocated for specific applications and users. When the licensed band for an application becomes full there is no more capacity for additional users. Consider what happens when excessing cell-phone communication (850MHz & 1.9GHz in the USA) traffic causes base stations to saturate and calls to be dropped. An example frequency-usage recording taken at Berkeley, California, USA [4] is shown in Fig. 1.2 [4]. Heavy usage and congestion at a few frequencies (e.g. 0–1, 1.9, 2.4 GHz) is seen, while minimal activity is present at others (e.g. 0.5% utilization in the 3–6 GHz band).

Inefficiency caused by the strict spectrum allocation can be solved by allowing use of licensed bands. Multiple users can share the same spectrum as licensed users as long as the do not cause harmful interference. Spectrum sharing approaches being pursued include: ultra-wideband (UWB), and cognitive radio (CR).

In the ultra-wideband scenario (IEEE 802.15.3a, IEEE 802.15.4a), a secondary user

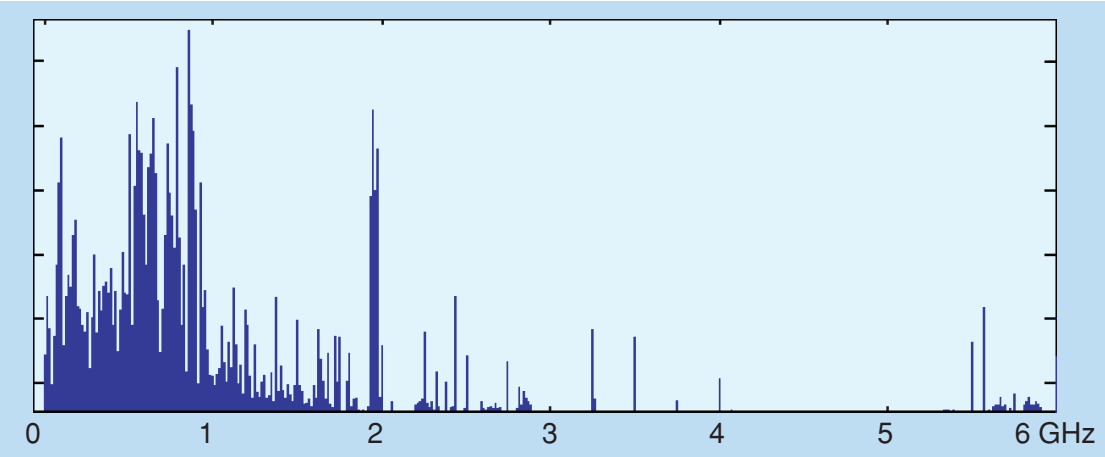


Figure 1.2: Spectrum utilization in downtown Berkeley showing 30% usage below 3GHz and 0.5% between 3–6 GHz [4]

is allowed if its transmitted power spectral density (PSD) is so low that it does not interfere with license users. A very wide bandwidth (>500MHz) is used to allow sufficient total output power for reliable communication. For cognitive radio (IEEE 802.22), dynamic spectrum access (DSA) is required. To accomplish this, high-speed spectrum sensing must be employed to detect which bands are not in use and when a primary user appears. This work focuses on the spectrum sensing aspect required by cognitive radios.

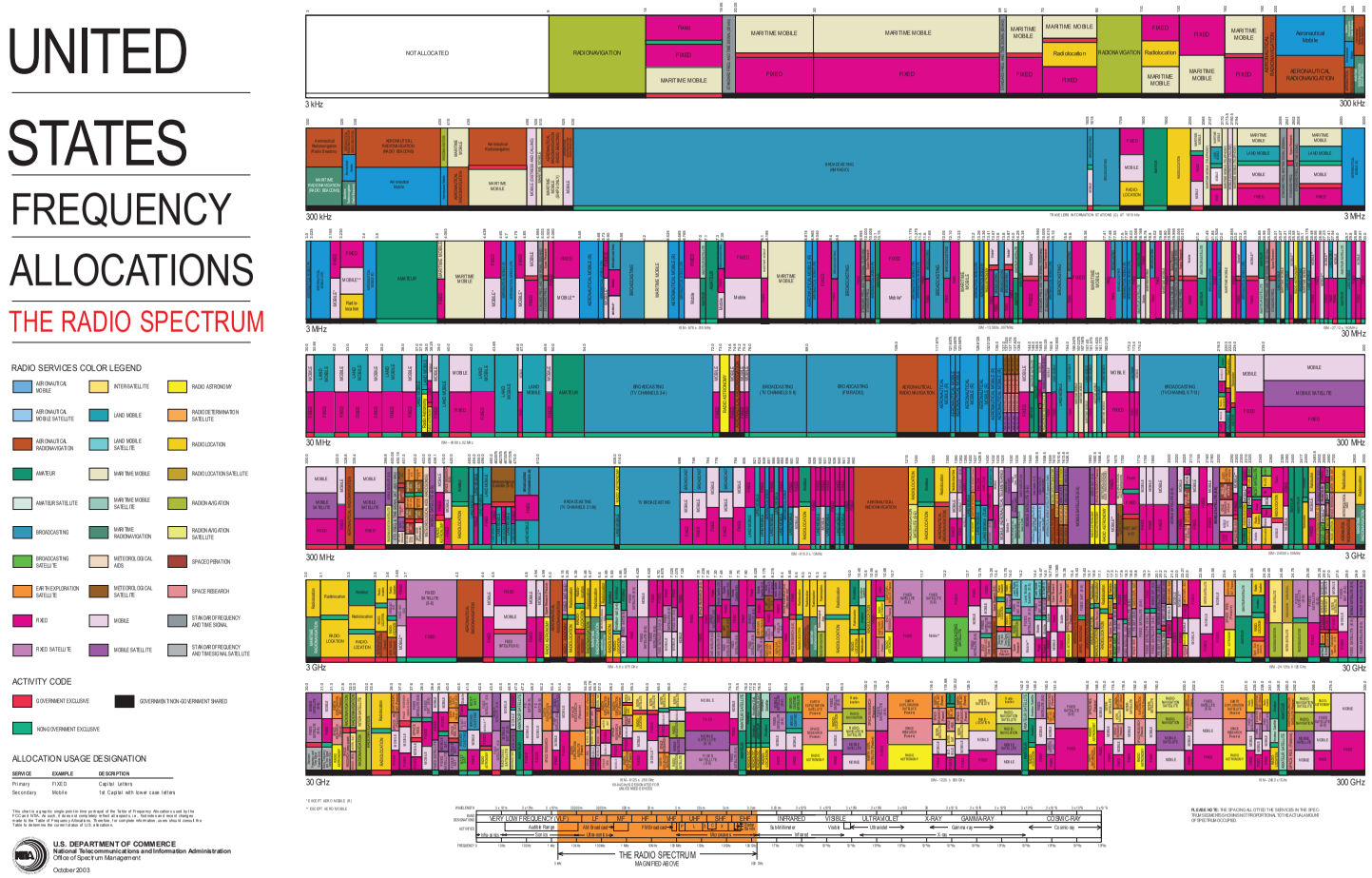


Figure 1.3: FCC spectrum allocation in the USA

1.3 Dynamic Spectrum Access

To solve the problem of spectral congestion, dynamic spectrum access allows a secondary user to temporarily use licensced, but unoccupied, frequency bands. To operate in this fashion, radio nodes must detect the incident RF power levels of different frequencies in real-time to find empty frequency bands. DSA attempts to identify these openings and communicate within them. This improves spectral usage efficiency and is a very useful tool in the fight against spectral congestion. New standards are already being explored to define the operating parameters of this kind of communication (IEEE 802.22 in the USA).

1.4 Cognitive Radio

Dynamic spectrum access is the primary intelligence within the concept of cognitive radio and differentiates it from a software-defined radio. The precise definition of a cognitive radio still remain unclear. An attempt to define it was made by the FCC [5]. Minimally, a cognitive radio is a dynamically adaptable device which can switch frequencies and protocols to maximize availability. When more than one secondary user is present, a basic negotiation capability is necessary. These requirements entail that the device must at least be able to: sense the available spectrum, change its operating frequency and bandwidth, vary its output power levels, and switch between communication protocols [6]. If cognitive radio designs were able to achieve such wide ranging general functionality, they would form a complete wireless solution.

A cognitive radio can be separated into two components, a software define radio (SDR), and a software based intelligence and control block. In this thesis, implementation of the wideband SDR digitization unit is emphasized. Real-time wideband digitization allows a wider bandwidth to be monitor, maximizing the probability of detecting

available spectrum. The cognitive radio then decides which unused frequency and protocol to use [7].

1.4.1 Spectrum Sensing

Dynamic spectrum access requires that the RF spectrum is monitored by a spectrum sensor. To perform real-time wideband monitoring, the cognitive radio's hardware must be rather unique. At the circuit level, implementing a spectrum sensor is challenging. Additionally, a cognitive radio requires continuous monitoring of all candidate frequency bands. Power consumption is often a limiting factor, even for narrowband observation.

1.5 Organization

This dissertation focuses on the architecture and circuit designs for wideband SDR receiver front-ends.

Chapter 2 explores different architectures for spectrum sensing in SDR. The need for analog signal processing prior to digitization to lower total power consumption, is emphasized. Frequency domain discrimination techniques are focused upon due to their reduced digitization power. An analog signal processor is identified as an interesting blocks for wideband SDR sensing realization.

Chapter 3 introduces the concept of passive switched-capacitor circuits. Fundamentals circuit blocks which perform discrete-time analog charge-domain computations are presented. Non-ideal behaviors of these processing elements are covered later.

Chapter 4 details the practical error sources of passive switched-capacitor operations. Their effects upon the accuracy and precision of computations is derived. Methods for their modeling, reduction, and correction are described. Simulation examples are used to show optimization trade-offs in the prescence of non-linear effects.

Chapter 5 analyzes the effects of noise upon the charge-domain computation results. The total noise is calculated by examining each contributing source and how it manifests on the outputs. Different charge-domain circuits are considered and the resulting noise correlation between outputs is determined.

Chapter 6 discusses the design of a passive switched-capacitor discrete Fourier transform based RF signal processor for use prior to digitization. The concept, non-idealities modeling and mitigation, implementation details, and optimization of the Charge Reuse Analog Fourier Transform (CRAFT) engine prototype is described. Measurement results are presented.

Chapter 7 summarizes the thesis, and draws conclusions regarding the applicability of passive switched-capacitor circuits to wideband SDR-based sensing.

Chapter 2

Spectrum Sensing Architectures

2.1 Introduction

Cognitive radios function on a spectrum sharing concept and allow secondary users to co-exist alongside primary users without causing harmful interference. This is predicated on the capability of the secondary users' ability to actively sense the RF spectrum and determine which frequency bands are unoccupied. The desire for versatile solutions has driven the focus towards operating over wide frequency bandwidths. Wideband spectrum sensing is a topic currrently limited by RF front-end and digitization architectures.

For a greater flexibility through increased software capability, all architectures for software defined radios strive to place the ADCs and DACs as close to the antenna as possible. A simple implementation of this concept was first proposed by Joseph Mitola [7] where the architecture incorporated an immediate conversion of the received RF signal to the digital domain as shown in Fig. 2.1.

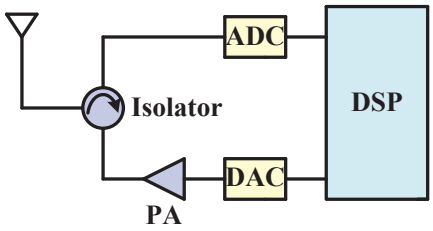


Figure 2.1: The conceptual Cognitive Radio architecture proposed by J. Mitola

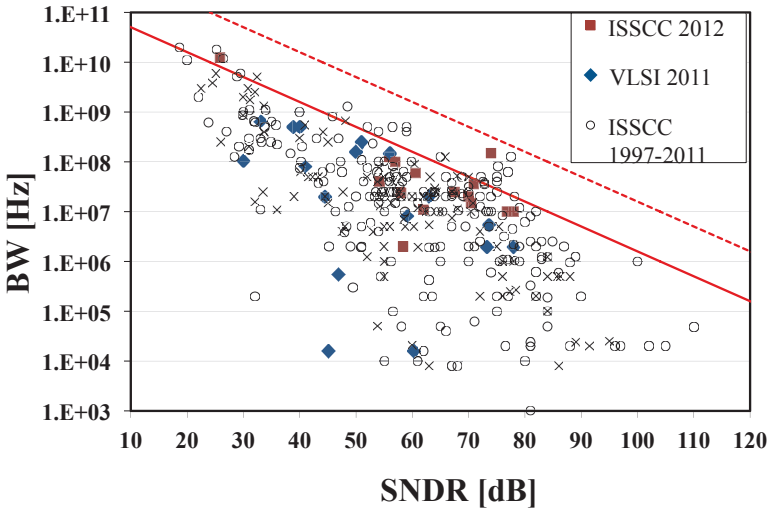


Figure 2.2: A survey of ADCs from 1997–2012 [1]

2.2 Wideband Digitizer Architecture

The ADC resolution and speed requirements that the architecture shown in Fig. 2.1 entails make it impractical. A tremendous amount of power is necessary by an ADC to meet the requirements of a direct-to-digital wideband radio. A plot of the dynamic range and bandwidth of published ADCs [1] is shown in Fig. 2.2.

2.2.1 Time-Domain

In order to ease the requirements on the ADCs, time-interleaving can be used to reduce the sampling rate each ADC sees [8]. All the ADCs still see the full bandwidth and

input swing causing their dynamic range requirements to not be relaxed. The high speed sampling circuitry can also be a performance limiter since it must still operate with input signals within the aggregate sampling bandwidth.

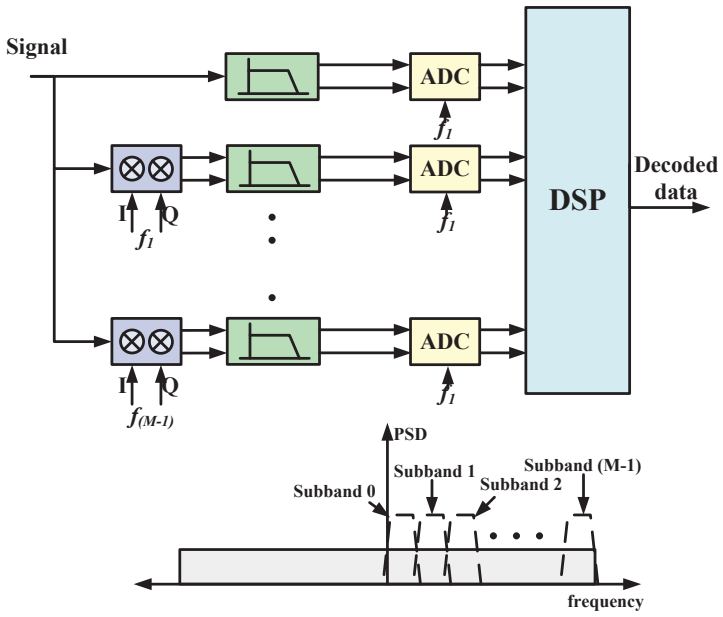


Figure 2.3: Low pass filterbank approach for channelization

2.2.2 Frequency-Domain

In order to reduce the dynamic range requirements on the ADCs, it is possible to transform the signal to a different domain prior to digitization [9]. Specifically, a frequency domain transform is particularly attractive [10]. A frequency domain transform can be approximated in practice using band-pass filters for channelization, as proposed in [11]. This reduces the dynamic range requirements of the ADCs but introduces the problem of designing impractically sharp band-pass filters. Replacing sharp band-pass filters by frequency down-converters followed by sharp low-pass filters eliminates this problem [12] as shown in Fig. 2.3. However, these are based on PLLs, mixers and low-pass

filters [13], or on injection locked oscillators [14][1], and can be power hungry. Moreover, harmonic mixing of signals within the SDR input bandwidth severely corrupts the channelized baseband signals. Additionally, signal reconstruction from the digitized filter-bank outputs is challenging.

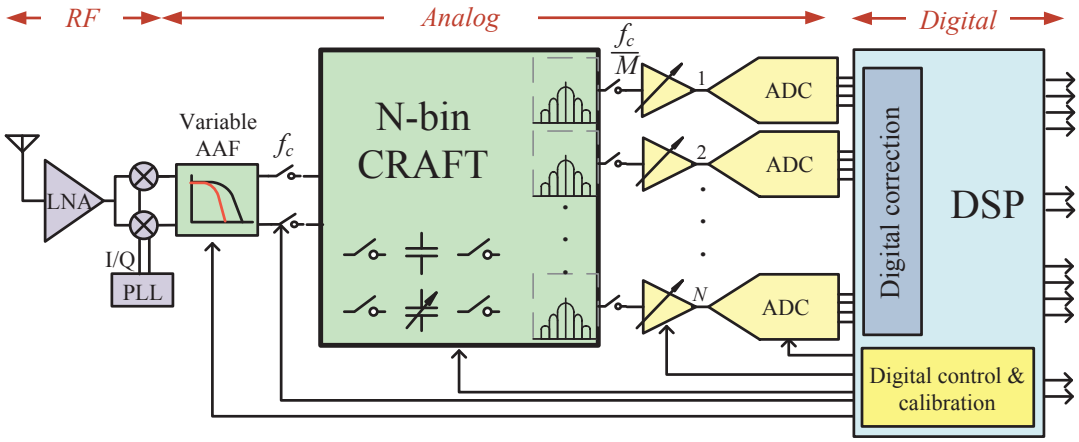


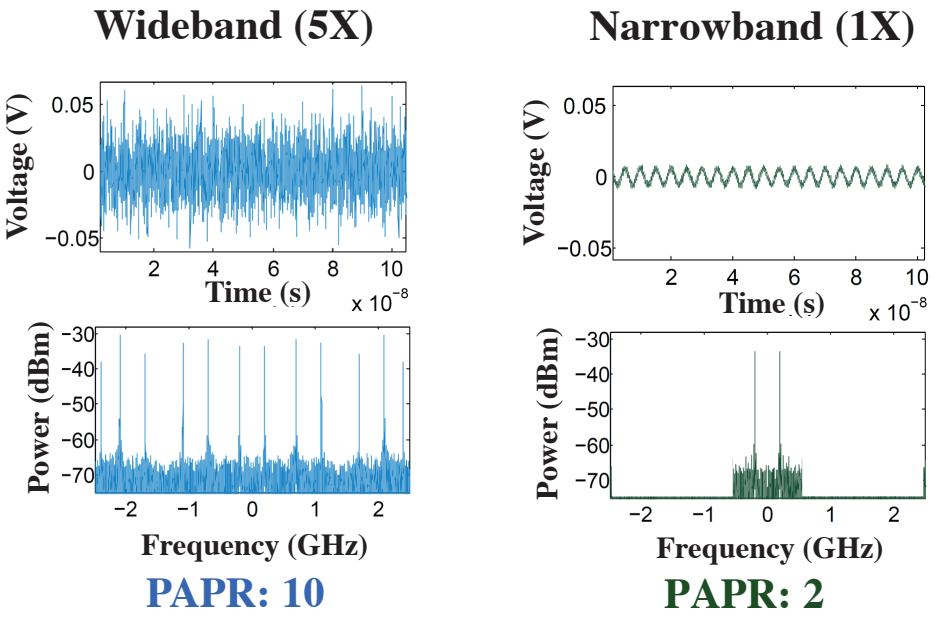
Figure 2.4: An envisioned SDR architecture enabled by CRAFT

In this thesis, we propose a frequency-domain digitizer using an analog domain DFT (CRAFT) before entering the ADCs (Fig. 2.4). This approach provides advantages through ADC dynamic range reduction due to the analog-domain channelization performed. In addition, the analog DFT processing provides signal spreading benefits, does not suffer from harmonic mixing, allows simple reconstruction in the digital domain, and can be implemented with ultra-low energy per DFT conversion. The architecture and implementation of this block is detailed in Chapter 6.

For ADC dynamic range calculations, signals are assumed to be distributed in frequency as shown in Fig. 2.5(a). The peak to average power ratio (PAPR) in this case may be as high as 10. In comparison, separating it into five equal frequency channels,

[1]Note that injection locked oscillators have the advantage of a larger noise suppression bandwidth ($\approx$ lock range) [15] and provide better reciprocal mixing robustness compared to PLLs (assuming the reference phase noise is better than the VCO phase noise).

one of which is shown in Fig. 2.5(b), reduces the PAPR to two (factor of five reduction). In general, an N path channelization of spread signals reduces the dynamic range by N times[2], causing an N times dynamic range reduction for the ADCs.



(a) wideband input

(b) channelized input

Figure 2.5: Wideband and channelized signals showing dynamic range reduction

2.3 Conclusions

In this chapter, several SDR architectures for cognitive radio applications were discussed. Additionally, new circuit topologies that are potential candidates for satisfying the required specifications were reviewed. Existing architectures for tackling the extremely challenging problem of wideband spectrum sensing was explored, and a new architecture for the same was proposed.

[2]Unlike the example shown with perfect channelization, when the signals distributed, but not directly on-bin, the DFT provides sufficient signal spreading for this approximation to roughly hold.

Chapter 3

Fundamental Charge-Domain Operations

3.1 Introduction

The fundamentals of passive switched-capacitor compuations are introduced in this chapter. Circuits for charge-domain addition and scalar multiplication are presented. Variants which can perform actions such as: weighted addition, and multiplication plus offset are mentioned.

In later chapters, the settling behavior and error sources in these circuits are considered. Their impact on design performance is modeled and optimized. These concepts are ultimately used in the design of CRAFT, a passive switched-capacitor 16-point FFT.

3.2 Share Operations

A share operation is used to implement addition and any number of terms can be summed (charge-averaged). Inputs can also be weighted differently if they are stored

on capacitors of different size.

The simplest share operation, having two operands and one switch, is introduced in the following section. Examples of 2 and 4-point share operations in a complete system can be seen in Fig. 6.8 of Chapter 6.

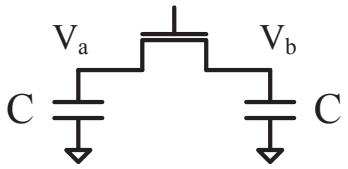


Figure 3.1: 2-point share operation

2-point Share

The simplest share operation is one which has two input operands on capacitors of equal size, as shown in Fig. 3.1. The charge-domain expression in Eq. (3.1) shows the result with complete settling in relation to the initial capacitor voltage V_{A0} and V_{B0}.

$$V_A = V_B = \frac{CV_{A0} + CV_{B0}}{2C} = \frac{V_{A0} + V_{B0}}{2} \tag{3.1}$$

With incomplete settling, the time-domain settling response of the circuit must be determined. This is done by examining the differential equations for the system. They are shown below for a generalized (unequal capacitors) 2-point share operation.

$$\begin{aligned} C_a \dot{V}_a &= (V_b - V_a)/R_{sw} \\ C_b \dot{V}_b &= (V_a - V_b)/R_{sw} \end{aligned} \tag{3.2}$$

When the operand capacitors are equally sized ($C_a = C_b = C$) an unweighted share operation occurs. The solution to this system of equations can be found in Sec. 4.5.

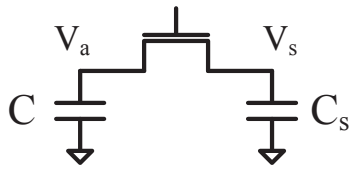


Figure 3.2: 1-point multiply operation

3.3 Multiply Operations

A multiply-only operation can only be performed on one operand[1] at a time. This is accomplished by connecting a second capacitor, C_s, to the operand capacitor, C, using a single switch as shown in Fig. 3.2. Depending on the stealing capacitance C_s and initial voltage on it, multiply or multiply plus offset operations can be realized.

1-point Multiply

The result of the charge-domain computation with complete settling is shown in Eq. (3.3), where V_{a0} and V_{s0} are the initial voltages on the input operand capacitor and stealing capacitors, C and C_s, repectively.

$$V_a = V_s = \frac{CV_{a0} + C_sV_{s0}}{C + C_s} = m \cdot V_{a0} + (1-m) \cdot V_{s0} \tag{3.3}$$

This yields a simple scalar multiplication of the input V_{a0} by m $(= C/(C + C_s))$ when the stealing capacitor holds no initial value $(V_{s0} = 0)$.

With incomplete settling, the time-domain settling response of the circuit must be determined. This is done by solving the differential equations for the system:

[1]Multiple capacitors, each representing copies of the same operand, can utilize a share and multiply switch structure to perform multi-point scalar multiplication. This does not reduce the total capacitance required but removes the need to accurately match the stealing capacitors between separate 1-point multiply operations for the operand copies independently.

$$\begin{aligned} C\,\dot{V}_a &= (V_s - V_a)/R_{sw} \\ C_s\,\dot{V}_s &= (V_a - V_s)/R_{sw} \end{aligned} \tag{3.4}$$

Note the similarity between a multiply (Eq. (3.4)) and a share operation with unequal operand capacitors (Eq. (3.2)). The only difference between the two solutions is that a multiply operation usually has zero initial value on one capacitor (C_s). The solution to this system of equations can be found in Sec. 4.5.

This operation creates two output copies of the computation result. There can be several uses for this additional output created by the stealing capacitor. Normally, the stealing capacitor will be reset ($V_{S0} = 0$) before the full computation sequence is repeated. It can be reset earlier so that it can be used again at a later time in the computation sequence.

It can also be used as an additional input operand for future operations. Note that, since it is usually a different size capacitor, similar succeeding operations to those performed on C will have to account for the different capacitor size. This may limit the matching between these two continuing computation paths.

If $C_s = C$, a multiply operation can be used as an operand duplicator at the expense of voltage attenuation by $\frac{1}{2}$. When $C_s \ll C$ it is a low-attenuation operand duplicator. One potential use for this is to perform additional side-computations that do not require as much dynamic range since their kT/C noise is higher.

Alternatively, the stealing capacitor's voltage can be retained as a record of the initial input operand's value at an earlier time in the computation. This may be useful because of the destructive nature of in-place charge-domain operations. If the value is kept until the next processing cycle, this opens the possibility of IIR filtering.

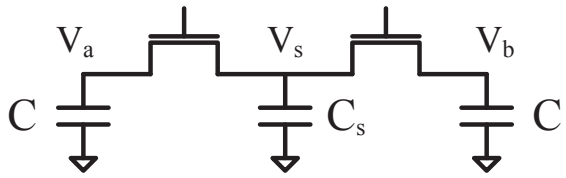


Figure 3.3: 2-point share and multiply operation

3.4 Share and Multiply Operations

A share and multiply operation is used to implement addition followed by multiplication. This is partcularly useful since it requires only one computaton phase rather than the two required for a share followed by a multiplication operation. Any number of input terms can be used and sub-unity multiplication factors can be realized. The inputs can also be weighted differently if they are stored on capacitors of different size.

The simplest share and multiply operation, having two operands and two switches, is introduced in the following section. Examples of 2 and 4-point share and multiply operations, with as many as 10 switches, can be seen in Fig. 6.8 of Chapter 6.

2-point Share and Multiply

The simplest share and multiply operation is one which has two input operands on capacitors of equal size, two switches, and one additional stealing capacitor C_s, as shown in Fig. 3.3.

The charge-domain expression in Eq. (3.5) shows the result with complete settling in relation to the initial capacitor voltages V_{a0}, V_{b0}, and V_{s0}.

$$V_a = V_b = V_s = \frac{CV_{a0} + CV_{b0} + C_sV_{s0}}{2C + C_s} = m \cdot \frac{V_{a0} + V_{b0}}{2} + (1-m) \cdot V_{s0} \qquad (3.5)$$

The multiplication factor, caused by the stealing capacitor being connected by the switches, is m $(=2C/(2C+C_s))$. If the stealing capacitor is cleared before the operation,

$V_{s0}=0$, the result of Eq. (3.5) is simiply a scaled average of the input voltages.

With incomplete settling, the time-domain settling response of the circuit must be found using the differential equations for the system. They are shown below for a generalized (unequal operand capacitors) 2-point share and multiply operation.

$$\begin{aligned} C_a\,\dot{V}_a &= (V_s - V_a)/R_{sw,1} \\ C_b\,\dot{V}_b &= (V_s - V_b)/R_{sw,2} \\ C_s\,\dot{V}_s &= (V_a - V_s)/R_{sw,1} + (V_b - V_s)/R_{sw,2} \end{aligned} \tag{3.6}$$

When the operand capacitors are equally sized ($C_a=C_b=C$) an unweighted share and multiply operation occurs. The solution of this case can be found in Sec. 4.5.

Chapter 4

Charge-Domain Computation Error Sources

4.1 Introduction

Analog circuits are vulnerable to many sources of non-ideality which are generally classified as either systematic or random, and input-independent or dependent. The combination of these these effects determine the system dynamic range.

Some non-ideal effects are mitigated in the analog domain using design and layout techniques such as: circuit linearization, negative feedback, offset cancellation circuits, and common-centriod layout [16]. Digital correction can also be used after digitization, allowing simpler circuits to be used or when no circuit techniques of high-enough performance exist. Input-dependent effects are usually more complicated than input-independent effects and must be modeled and corrected as they are a primary driver of the systems' overall dynamic range. While the complexity of their complete correction may be prohibitive, technology scaling and approximation techniques improve the benefit and advantages from digital back-ends.

Systematic effects can be caused by things such as: improper circuit design, device modeling errors, chip layout parasitic effects, or large-scale IC manufacturing variability. Random effects are primarily related to manufactured device variablility, but may still be correctable by the digital back-end with appropriate procedures (i.e. Sec. 6.6.1).

In passive switched-capacitor circuits, non-ideal effects are a critical consideration because results are stored as charges on capacitors, making all circuit nodes high-impedance. The passive nature means that there are no active devices performing computatations. Consequently, any advantages they have, as seen in active switched-capacitor (SC) designs [17], cannot be realized. For example, in charge-domain operations, switch channel charge must be handled without low-impedance or virtual-ground nodes of traditional SC circuits [18].

In the context of passive charge-domain computations, input-independent, systematic non-idealities arise primarily from the mismatch among the capacitors and the parasitic capacitances of the passive switched-capacitor circuitry. Input dependent errors, such as switch resistance variation, limit the dynamic range of computations without systematic error (or with effective digital correction).

The specific linear and nonlinear error sources most important to high-performance passive switched-capacitor design are summarized below.

Linear Error Sources

1. *Clock Feedthrough:* MOS switch parasitic capacitance allow computation clock signal to propagate to the sensitive operand nodes. It is analyzed, and the conditions under which it has a residual effect is covered in Sec. 4.2.

2. *Capacitor Mismatch:* Systematic and random capacitor mismatch modify the charge-domain computations in a input-independent, deterministic manner. The analysis of capacitor-size mismatch is covered in Sec. 4.4.

3. *Incomplete Settling:* Charge-domain circuits have limited time to perform computations. Excluding voltage-dependent effects (which are analyzed and modeled separately), settling error is covered in Sec. 4.5.

4. *Computation Noise:* Processing and reset noise generate linear computation errors. They are analyzed separately in Chapter 5.

Non-Linear Error Sources

1. *Charge Absorption and Injection:* MOS switches absorb and inject charge into computation nodes as their conduction channels are formed and dissipated. The effects of this and the reason for its nonlinear nature are detailed in Sec. 4.3.

2. *Settling Nonlinearity:* MOS switches have voltage-dependent on-resistance. This introduces nonlinear variation in the amount of settling acheived, depending on the input values of the computation. Modeling methods, simulation results, and optimization techniques can be found in Sec. 4.6.

Single-ended and pseudo-differential operand implementations are introducted in Sec. 4.5. Their utility in suppressing certain linear errors is covered in this chapter. Mitigation techniques for most error sources are covered in this chapter and in Chapter 6, where modeling and system-level optimization are also discussed. Additional error sources not analyzed here are enumerated below.

Other Error Sources

1. *Sampler Charge Injection* (Nonlinear): Since a sampler must precede any charge-domain computation, any corruption it causes will become part of the result. Aspects in relation to the design of the CRAFT are discussed in Sec. 6.3.1.

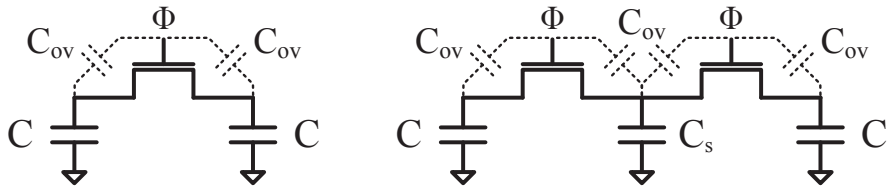
2. *Sampler Noise* (Linear): Preparing the operand capacitors with their initial values introduces a noise in their value.

3. *Leakage* (Nonlinear): Capacitors and computation switches will suffer from leakage, corrupting the passively-held values. It will be both voltage and temperature dependent, however, it is somewhat mitigated with a pseudo-differential design.

4. *Parasitics* (Nonlinear): Voltage dependence of parasitic capacitances, such as MOS contact junctions, cause input-dependence of the total operand capacitance.

4.2 Clock Feedthrough

The MOS switches used to perform computations have overlap capacitances, $C_o v$, which allow the clock signal to disturb the computation nodes voltages. This undesired parasitics are shown in the share and share and multiply diagrams of Fig. 4.1.



(a) 2-point share (b) 2-point share and multiply

Figure 4.1: 2-point charge-domain with switch overlap capacitance

Each charge-domain operation consists of one rising and one falling clock edge (Φ: from 0 to V_{dd}) applied to the switch gates. Analysis of the effect of clock feedthrough for both share and share and multiply operations follows in this section. General cases for 2-point operations are analyzed in addition to the specific implementations employed by CRAFT, having equal operand capacitor sizes. In the context of modeling and mitigation, clock feedthrough is also discussed in Sec. 6.4.5. Finally, situations under

which clock feedthrough effects can be neglected, as well as mitigation techniques, are discussed at the conclusion of this section.

4.2.1 Share Operations

A 2-point share operation with equal operand capacitors is shown in Fig. 4.1(a). It suffers equal and opposite disturbances (ΔV and $-\Delta V$) on the operands as the switch closes and opens.

$$\Delta V = V_{dd} \cdot \left(\frac{C_{ov}}{C + C_{ov}} \right) \tag{4.1}$$

This common-mode shift to both operands and subsequent restoration yields no net effect on the charge-domain operation.

In the case of a general share operation, where the operands capacitor are different sizes (say, C_1 and C_2), the effect of clock feedthrough is not self-cancelling. This is determined by considering the charge ($\Delta Q = C \cdot \Delta V$) injected and removed from each operand. In this case, clock feedthrough charge passing through the overlap capacitances will have different values and is shown below.

$$\Delta Q_1 = V_{dd} \cdot \left(\frac{C_{ov}}{1 + \frac{C_{ov}}{C_1}} \right) \approx V_{dd} C_{ov}$$
$$\Delta Q_2 = V_{dd} \cdot \left(\frac{C_{ov}}{1 + \frac{C_{ov}}{C_2}} \right) \approx V_{dd} C_{ov}$$

However, since any system should presumably have a low overlap-to-operand capacitance ratio ($C_{ov} \ll C$) ΔQ_1 and ΔQ_2 are approximately equal. Even under this assumption, the charge-domain equations in (4.2), which contains the effects of turn-on clock feedthrough, complete setting, and turn-off feedthrough, show that the clock feedthrough error cancels only if C_1 equals C_2.

$$
\begin{aligned}
V_{out,C_1} &= \frac{\overbrace{(Q_1+Q_2)}^{\text{inputs}}+\overbrace{(\Delta Q_1+\Delta Q_2)}^{\text{turn-on}}}{C_1+C_2}-\frac{\overbrace{\Delta Q_1}^{\text{turn-off}}}{C_1} \\
V_{out,C_2} &= \underbrace{\frac{(Q_1+Q_2)+(\Delta Q_1+\Delta Q_2)}{C_1+C_2}}_{\text{charge-averaging}}-\frac{\Delta Q_2}{C_2}
\end{aligned}
\tag{4.2}
$$

4.2.2 Multiply Operations

A multiply operation is equivalent to a weighted share, which was examined at the end of the previous section. Clock feedthrough cancels only when $C = C_s$ ($m = 0.5$).

4.2.3 Share and Multiply Operations

Unlike share operations, the effects of clock feedthrough are no longer self-cancelling for share and multiply operations when the operand capacitors are equally sized. This is because the stealing capacitor C_s, and all of its charge, are removed from the system when the switches turn off. While it is normally reset[1] ($Q_s = 0$) before the operation, after the charge-redistribution phase, the stealing capacitor contains a different amount of charge injected into the system by the switch turn-on clock feedthrough. Then, when the falling clock edge removes charge from the system, it is disconnected from the system without being able to redistribute and cancel the initial, averaged, turn-on contribution.

$$
\begin{aligned}
V_{out,C} &= \frac{\overbrace{(Q_1+Q_2)}^{\text{inputs}}+\overbrace{(2\Delta Q+2\Delta Q_s)}^{\text{turn-on}}}{2C+C_s}-\frac{\overbrace{\Delta Q}^{\text{turn-off}}}{C} \\
V_{out,C_s} &= \underbrace{\frac{(Q_1+Q_2)+(2\Delta Q+2\Delta Q_s)}{2C+C_s}}_{\text{charge-average and multiply}}-2\frac{\Delta Q_s}{C_s}
\end{aligned}
\tag{4.3}
$$

[1] Ground or a system-defined common-mode voltage can be used as a reference level. This allows no net charge to be introduced by this capacitor at the start of the operation. Any error in the initial value will just create an offset error.

Using the same notation as the share operation section above, Eq. (4.3) is written for a 2-point share and multiply operation with equal operand capacitors, as shown in Fig. 4.1(b). Note that there are two C_{ov} paths for to C_s, hence the $2\Delta Q_s$ term.

Q_1 and Q_2 are the initial charges on the two operand capacitors of capacitance C. It is clear from these equations that clock feedthrough will leave a residual error on the operand capacitors' final value ($V_{out,C}$) for any case other than $C = C_s$ (a share and multiply operation with $m = \frac{2}{3}$).

In the most general case, with unequal operand capacitors, and a non-zero initial charge, Q_s, on C_s, the charge-domain expressions is written below.

$$
\begin{aligned}
V_{out,C_1} &= \frac{\overbrace{(Q_1+Q_2+Q_s)}^{\text{inputs}} + \overbrace{(\Delta Q_1+\Delta Q_2+2\Delta Q_s)}^{\text{turn-on}}}{C_1+C_2+C_s} - \frac{\overbrace{\Delta Q_1}^{\text{turn-off}}}{C_1} \\
V_{out,C_2} &= \frac{(Q_1+Q_2+Q_s)+(\Delta Q_1+\Delta Q_2+2\Delta Q_s)}{C_1+C_2+C_s} - \frac{\Delta Q_2}{C_2} \\
V_{out,C_s} &= \underbrace{\frac{(Q_1+Q_2+Q_s)+(\Delta Q_1+\Delta Q_2+2\Delta Q_s)}{C_1+C_2+C_s}}_{\text{charge-average and multiply}} - 2\frac{\Delta Q_s}{C_s}
\end{aligned}
$$

It is evident above that the error on the operand capacitors will only be cancelled if $C_1 = C_2 = C_s$, and all are much greater than C_{ov} (to make $\Delta Q_1 \approx \Delta Q_2 \approx \Delta Q_s$).

4.2.4 Conclusions

In this section, it was shown how a 2-point unweighted share operation is not corrupted by clock feedthrough. This is also true for share operations (of equal operand capacitance) with any number of operands, as long as there are no extraneous (non-operand) nodes caused by the switching network. However, when the operand capacitors are not of equal size, there are residual voltage offset errors left on each of the output copies (operands).

Similary, for a 2-point share and multiply operation, it was shown how clock feedthrough

error will be present except when the operand and stealing capacitors are equal. This is true for a generalized share and multiply operation as well.

As clock feedthrough causes an offset in the charge equation, its effect is a linear computation error. Additionally, its magnitude depends on the supply voltage, operand capacitor sizes, the operation's multiplication factor m (determining C_s), the switch configuration, and the switch sizes. While the error is net-charge neutral in certain types of computations, it is always important to consider the common-mode voltage shift induced at the start of the operation as it may impact other aspects of the computation due to other voltage dependent effects, such as switch resistance.

As charge-domain processing speeds increase, clock feedthrough error increases in amplitude. This is because faster operations require a smaller $R_{sw}C$ product, where R_{sw} is the switch resistance. As switch resistance decreases (and switch width W increases correspondingly), overlap capacitance increases since $C_{ov} \propto W$. This negatively impacts the ratio C_{ov}/C, which determines the clock feedthrough voltage step magnitude, as seen in Eq. (4.1). Clock feedthrough is ideally cancelled by adding half-dummy switches [19] at the expense of doubling switching power and requiring differential clocks.

Interestingly, a pseudo-differential implementation (i.e. Fig. 4.6) also suppresses clock feedthrough error. This is due to the fact that the positive and negative operand representations will suffer equal effects, effectively transforming the error into a common-mode component.

4.3 Charge Absorption and Injection

During passive switched-capacitor operations, turning on nMOS switches causes charge absorption from the source and drain nodes as device channels are formed. Similarly, turning off the switches causes this channel charge to be injected back into the system. This causes perturbations of the voltages held at these high-impedance operand nodes.

This effect is similar to clock feedthrough in that it occurs at both edges of the clock transition. However, it is influenced by additional factors, such as: clock rise and fall-times, and the node impedances. Moreover, the amount of charge absorbed from, and injected into, the system depends on the MOS switches' drain and source terminal voltages, and the channel capacitance C_{ch}.

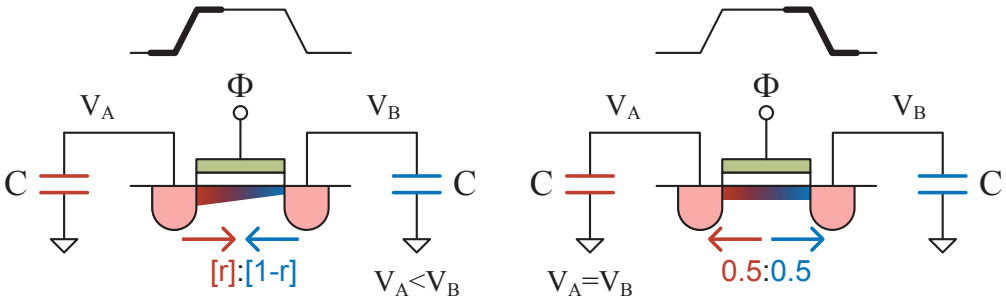
The magnitude of the charge injected by each switch, Q_{inj}, and the corresponding voltage step, is voltage-dependent as shown below,

$$Q_{inj} = (V_{dd} - V_{tn} - V_{out}) \cdot C_{ch} \tag{4.4}$$

where V_{out} is the result of the charge-domain computation, V_{dd} is the clock supply voltage, and V_{tn} is the nMOS switch threshold voltage. The channel capacitance is also voltage dependent: $C_{ch} = f(V_{dd}, V_{out})$; although, it is a weak dependence for inversion mode devices in deep triode. This voltage dependence causes a nonlinear effect.

For charge absorption, the initial operand node voltages play an important part as shown in Fig. 4.2(a). Depending on the initial node voltages, the switch spends some of its turn-on transition time forming a channel in the saturation region of operation and then continues into the triode operating regime. This influences how much charge is contributed by each terminal in forming for the channel. As a result of the sharing operation, the switch being on for a while, this unequal charge absorption error is distributed among the capacitors causing an offset error. Charge injection, as shown in

Fig. 4.2(b), occurs at the end of the operation and errors on the final values.



(a) Absorption (saturation & triode) (b) Injection (deep triode)

Figure 4.2: Charge absorption and charge injection in a share operation

The impact of charge arbsorption and injection effects on an actual design are covered in Sec. 6.4.6. Also, the assumptions and supporting references are discussed in relation to the simulation models developed and mitigation techniques employed.

4.3.1 Share Operations

For a share operation with complete settling and no floating (non-operand) nodes, the channel charge from each switch is injected equally into every operand node. If all operands have equal capacitance, or the clock fall-time is fast enough [20], this injected-charge splits equally, as shown in Fig. 4.2(b), and causes an equal-magnitude voltage step on all the outputs. However, while the effect of charge-injection on a share operation is equal among output copies (operands), the magnitude is output-voltage dependent as shown in Eq. (4.4).

For a weighted share operation with operands of unequal capacitances, the voltage offset errors induced by the injected charge will differ among the outputs. If both outputs are used for further calculations, care must be taken to ensure that their differences do not adversly impact the subsequent results.

4.3.2 Multiply Operations

A multiply operation suffers from charge injection just like the weighted share mentioned at the end of the previous section.

4.3.3 Share and Multiply Operations

Just as in a share operation, the channel charge of every switch in a share and multiply computation splits equally and is injected into the circuit's nodes. The charge splitting ratio can also be influenced by clock fall time and node impedances[2] [20]. The voltage-dependence of the injected charge, as shown in Eq. (4.4), will influence the magnitude of the effect as well.

In the more complex case of a weighted share and multiply operation, where operand capacitors are unequal, the injection-induced offsets among the outputs will manifest as different voltage steps. Just as for share operations, if both outputs are used for further calculations, care must be taken to ensure that their differences do not adversly impact the subsequent results.

4.3.4 Conclusions

In this section, the manners in which charge absorption and charge injection effect the results of charge-domain computations were outlined. Charge absorption effects can be treated as system-wide offsets under complete (or nearly) settling because charge in the system is redistributed during the computation. The nonlinear effect of charge injection at the end of share and share and multiply operations were discussed. To first-order, its magnitude depends on: supply voltage, operand capacitor sizes, the switch configuration, and the switch sizes.

[2]In particular, switches between operand capacitors, C, and the stealing capacitor, C_s, will see different impedances at their terminals if $C_s \neq 2C$ for a 2-point share and multiply operation.

Like clock feedthrough, the voltage-independent component of charge injection is ideally cancelled by adding half-dummy switches. This doubles switching power and requires differential clocks with good symmetry. However, the voltage-dependent component of the injected charge, which is dominant under high-swing situations (desired for large SNR) is not rejected by half-dummies [19]. Even a pseudo-differential implementation (i.e. Fig. 4.6) is not advantageous since the positive and negative operand representations will suffer effects of different magnitudes due to the voltage-dependence of the injected charge.

As charge-domain processing speeds increase, switch channel charge effects increases in amplitude. This is due to faster operations requiring a smaller $R_{sw}C$ product, where R_{sw} is the switch resistance. As switch resistance decreases (and switch width W increases correspondingly), channel capacitance increases since $C_{ch} \propto C_{ox}WL$, where L is the switch length, and C_{ox} is a capacitance per gate area parameter of the device. This negatively impacts the error-to-signal charge ratio shown below, which a measure of the computational nonlinearity (dynamic range) due to charge injection.

$$\frac{Q_{inj}}{Q_{sig}} \propto \frac{C_{ch}V_{dd}}{CV_{sig}}$$

With technology scaling, this charge ratio improves since $C_{ch} \propto L^2$ with constant switch resistance, while V_{sig} can scale proportionally with V_{dd} to maintain enough voltage headroom. Additionally, if the operand capacitance C scales by $1/\sqrt{(}V_{sig})$, to maintain constant SNR, the switch resistance would have to decrease (increasing switch width and C_{ch}) by the same factor to maintain a constant settling time constant $R_{sw}C$. Under all these conditions, the charge injection error to signal ratio improves with technology scaling.

4.4 Capacitor Mismatch

In the charge-domain, an operand's value is represented by the charge stored on a capacitor, Q, which is a product of the voltage and capacitance ($= C \cdot V$). With a voltage-domain sampler, only the voltage of the input is accurately captured. Therefore, the matching of capacitances is important and any deviation from the ideal values will cause an error in the weighting of variables entering an operation. Improper variable weighting will modify the operation from its desired intent.

The expression for a generalized charge-domain computation with any number of capacitors and complete settling is shown below.

$$V_{i,\text{final}} = \frac{\sum C_i V_{i,\text{init}}}{\sum C_i} \tag{4.5}$$

Charge-averaging performed across capacitors C_i, each with initial voltage $V_{i,\text{init}}$, leaves an equal final voltage $V_{i,\text{final}}$ on all the operands. If each capacitor differs from its ideal value ($C_i' = C_i + \Delta C_i$) the computational result is modified. With capacitor mismatch Eq. (4.5) can be reorganized as shown below.

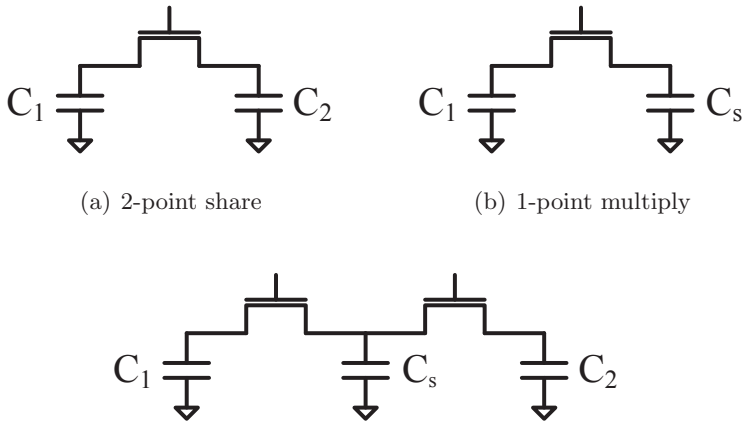
$$\begin{aligned} V'_{i,\text{final}} &= \frac{\sum (C_i + \Delta C_i) V_{i,\text{init}}}{\sum (C_i + \Delta C_i)} \\ &= \frac{\sum C_i V_{i,\text{init}}}{\sum (C_i + \Delta C_i)} + \frac{\sum \Delta C_i V_{i,\text{init}}}{\sum (C_i + \Delta C_i)} \\ &= \underbrace{\frac{\sum C_i V_{i,\text{init}}}{\sum C_i}}_{\text{ideal result}} \cdot \underbrace{\frac{\sum C_i}{\sum (C_i + \Delta C_i)}}_{\text{scaling error}} + \underbrace{\frac{\sum \Delta C_i V_{i,\text{init}}}{\sum (C_i + \Delta C_i)}}_{\text{offset error}} \end{aligned} \tag{4.6}$$

In this form, the effect of capacitor mismatch is shown to be an input-independent scaling factor combined with a linear input-dependent offset.

Fortunately, when there is no voltage dependence on the weighting (i.e. purely systematic capacitor mismatch), this causes a linear and correctable error. This fact is employed for the correction method used by CRAFT and described in Sec. 6.6.1.

Additionally, systematic linear errors are easily incorporated into LTI models, such as the one developed for CRAFT in Appendix B.

Two types of capacitor mismatch based errors are discussed below. When implemented in large systems, operand capacitors may be placed next to one another in the sampling array for improved matching and be of equal size. On the other hand, stealing capacitors are usually a different size and may be located next to the computational switches. This leads to different amounts of matching performance. Share operations only suffer from operand-capacitor mismatch, while share and multiply operations can also suffer from stealing-capacitor mismatch. They will naturally happen simultaneously but are analyzed separately here for clarity. Their combined effect for a share and multiply operation is summarized at the end of this section.



(a) 2-point share

(b) 1-point multiply

(c) 2-point share and multiply

Figure 4.3: Generic charge-domain operations used for mismatch analysis

4.4.1 Operand Capacitor Mismatch

The effects of operand capacitor mismatch in charge-domain operations are briefly covered here. Useful approximations of their impact are derived. At a system level, these

systematic errors can be corrected with the calibration method utilized by CRAFT and outlined in Sec. 6.6.1.

Share Operations

A 2-point share operation utilizing two capacitors, C_1 and C_2, is shown in Fig. 4.3(a). With initial values on these capacitors of V_1 and V_2, the result of the charge-domain operation (with complete settling) is based off of Eq. (4.5) and is shown below.

$$V_{out} = \frac{C_1 V_1 + C_2 V_2}{C_1 + C_2}$$

At the end of the operation, the two capacitors represent two result copies, each having a value of V_{out}. In the particular case of an equally-weighted share operation (addition), these capacitors are ideally of equal value, $C_1 = C_2 = C$. Under these conditions the addition operation (charge-averaging) gives the modified result below.

$$V_{out} = \frac{CV_1 + CV_2}{2C} = \frac{V_1 + V_2}{2}$$

However, when the sharing capacitors have a mismatch of ΔC ($=C_2 - C_1$), the charge-domain computation result becomes V'_{out}, as shown in Eq. (4.7) below.

$$V'_{out} = \frac{C \cdot (V_1 + V_2) + \Delta C \cdot V_2}{2C + \Delta C} \neq \frac{V_1 + V_2}{2} \tag{4.7}$$

This can be separated into ideal and error components and simplified further, under the assumption that capacitor matching is decent ($\Delta C \ll C$), using the binomial approximation: $(1-x) \approx (1+x)^{-1}$, for small x.

$$V'_{out} = \frac{V_1 + V_2}{2} \cdot \overbrace{\left(\frac{1}{1+\frac{\Delta C}{2C}}\right)}^{\text{scaling error}} + \overbrace{V_2 \cdot \frac{\Delta C}{2C}\left(\frac{1}{1+\frac{\Delta C}{2C}}\right)}^{\text{offset error}}$$
$$\approx \frac{V_1 + V_2}{2} \cdot \left(1 - \frac{\Delta C}{2C}\right) + V_2 \cdot \frac{\Delta C}{2C}\left(1 - \frac{\Delta C}{2C}\right)$$

Furthermore, since $\Delta C \ll C$, the squared term in the offset error can be dropped, giving

$$V'_{out} \approx \frac{V_1 + V_2}{2} + \frac{V_1 - V_2}{2} \cdot \frac{\Delta C}{2C} \tag{4.8}$$

This shows that the net effect of operand capacitor mismatch in a share operation is an input-value dependent offset. Also, the error magnitude is directly correlated with the amount of matching ($\frac{\Delta C}{C}$). Note that for systematic mismatch, the $\frac{\Delta C}{2C}$ factor between two particular capacitors will be constant, creating a completely linear error. The higher-order terms ignored in making the approximation (4.8) do not contain any input dependence, so linearity is maintainted even in the full solution (Eq. (4.7)).

Multiply Operations

A 1-point multiply operation utilizing two capacitors, C_1 and C_s, is shown in Fig. 4.3(b). With initial values on these capacitors of V_1 and V_s, the result of the charge-domain operation (with complete settling) is shown below.

$$V_{out} = \frac{C_1 V_1 + C_s V_s}{C_1 + C_s}$$

At the end of the operation, the two capacitors represent two result copies, each having a value of V_{out}. In the particular case of a pure scalar multiply operation (without any offset), the stealing capacitor has no initial charge ($V_s = 0$). Then, the operation (charge-stealing) has the result below, with multiplication factor $m = C_1/(C_1 + C_s)$.

$$V_{out} = \frac{C_1}{C_1 + C_s} V_1 = m \cdot V_1$$

However, when the operand capacitor has a mismatch of ΔC_1, the charge-domain computation result becomes V'_{out}, as shown in Eq. (4.9) below.

$$V'_{out} = \frac{C_1 + \Delta C_1}{C_1 + \Delta C_1 + C_s} V_1 \neq m \cdot V_1 \tag{4.9}$$

This can be separated into ideal and error components and simplified further, under the assumption that capacitor matching is decent ($\Delta C_1 \ll C_1$), using the binomial approximation: $(1-x) \approx (1+x)^{-1}$, for small x.

$$\begin{aligned} V'_{out} &= \frac{C_1}{C_1+C_s} V_1 \cdot \overbrace{\left(\frac{C_1+C_s}{C_1+\Delta C_1+C_s}\right)}^{\text{scaling error}} + \overbrace{V_1 \cdot \frac{\Delta C_1}{C_1+\Delta C_1+C_s}}^{\text{offset error}} \\ &= mV_1 \cdot \left(\frac{1}{1+m\frac{\Delta C_1}{C_1}}\right) + V_1 \cdot m\frac{\Delta C_1}{C_1}\left(\frac{1}{1+m\frac{\Delta C_1}{C_1}}\right) \\ &= mV_1 \cdot \left(\frac{1+\frac{\Delta C_1}{C_1}}{1+m\frac{\Delta C_1}{C_1}}\right) \\ &\approx mV_1 \cdot \left(1+\tfrac{\Delta C_1}{C_1}\right)\left(1-m\tfrac{\Delta C_1}{C_1}\right) \end{aligned}$$

Since $\Delta C_1 \ll C_1$, the squared term in the error factor can be dropped, giving

$$V'_{out} \approx m\left(1 + (1-m)\tfrac{\Delta C_1}{C_1}\right) \cdot V_1 = m' \cdot V_1 \tag{4.10}$$

This shows that the net effect of operand capacitor mismatch in a multiply operation is a scaling error. The error magnitude is correlated with the level of matching ($\frac{\Delta C_1}{C_1}$). Note that for systematic mismatch, this factor will be constant, creating a completely linear error. The terms ignored in making the approximation (4.10) do not contain any input dependence, so linearity is maintainted even in the full solution (Eq. (4.9)). Notably, the coefficient error percentage, $\frac{m'-m}{m} = (1-m)\frac{\Delta C_1}{C_1}$, is always less than the operand capacitor mismatch ratio ($\frac{\Delta C_1}{C_1}$).

Share and Multiply Operations

For a 2-point share and multiply operation, as shown in Fig. 4.3(c), three capacitors make up the operation. C_1 and C_2 form the input/output operands, and C_s is an extra capacitor used to steal charge away and perform sub-unity scaling of the final charge-averaged result. The result of the most general form (see Eq. (4.5)) of this charge-domain operation with complete settling is shown below.

$$V_{out} = \frac{C_1V_1 + C_2V_2 + C_sV_s}{C_1 + C_2 + C_s}$$

At the end of the operation, all three capacitors have the voltage V_{out}. The two operand capacitors, C_1 and C_2, represent two copies of the output result. The stealing capacitor, C_s, can also be used as an additional third output, if desired.

For the purposes of the designs and analysis here, the two operand capacitors have initial voltages of V_1 and V_2 while the initial value of C_s is zero[3] (or some of common-mode voltage reference V_{cm}). Furthermore, in the case of an equally-weighted share and multiply operation, the operand capacitors are ideally of equal value ($C_1 = C_2 = C$). Under these conditions, the addition (charge-averaging) occurs concurrently with the scaling operation and gives the modified result below.

$$V_{out} = \frac{CV_1 + CV_2}{2C + C_s} = \left(\frac{2C}{2C + C_s}\right)\frac{V_1 + V_2}{2} = m \cdot \frac{V_1 + V_2}{2} \tag{4.11}$$

When the operand capacitors are not identical due to mismatch, this causes a change from the ideal computation. By using Eq. (4.11) and substituting in $C_1 = C + \Delta C_1$ and $C_2 = C + \Delta C_2$, the modified charge-domain expression is written,

$$V'_{out} = \left(\frac{2C + \Delta C}{2C + \Delta C + C_s}\right)\frac{V_1 + V_2}{2} = m' \cdot \frac{V_1 + V_2}{2} \tag{4.12}$$

[3]Any error in this value will introduce a charge-error to the computation. An initial value error of ΔV_s will cause a final value offset error of $\Delta V_{out} = (1 - m) \cdot \Delta V_s$ in the share and multiply operation.

where, $\Delta C = \Delta C_1 + \Delta C_2$. The effect of operand capacitor mismatch is clearly visible as the multiplication factor m' now differs from the ideal value in Eq. (4.11). This modified scaling factor can be rewritten as shown below.

$$m' = \frac{2C + \Delta C}{2C + \Delta C + C_s} = \frac{1}{1 + \frac{C_s}{2C + \Delta C}} \tag{4.13}$$

To allow further simplification, the term in the denominator of Eq. (4.13) is examined and rewritten using the binomial expansion as follows.

$$\begin{aligned} \frac{C_s}{2C + \Delta C} &= \frac{C_s}{2C}\left(\frac{1}{1 + \frac{\Delta C}{2C}}\right) \\ &\approx \frac{C_s}{2C}\left(1 - \frac{\Delta C}{2C}\right) \end{aligned}$$

Substituting this result back into Eq. (4.13) gives the following result.

$$\begin{aligned} m' &\approx \frac{1}{1 + \frac{C_s}{2C}\left(1 - \frac{\Delta C}{2C}\right)} = \frac{1}{1 + \frac{C_s}{2C} - \frac{C_s}{2C}\frac{\Delta C}{2C}} \\ &\approx \left(\frac{1}{1 + \frac{C_s}{2C}}\right)\left(\frac{1}{1 - \left(\frac{C_s}{2C}\frac{\Delta C}{2C}\right) / \left(1 + \frac{C_s}{2C}\right)}\right) = m \cdot \left(\frac{1}{1 - m\left(\frac{C_s}{2C}\frac{\Delta C}{2C}\right)}\right) \end{aligned}$$

Once again, since $\Delta C \ll C$ and $0 < m < 1$, the binomial expansion can be used. Additionally, using the relationship $\frac{C_s}{2C} = \left(\frac{1}{m} - 1\right)$ gives the result in Eq. (4.14).

$$m' \approx m \cdot \left(1 + (1-m)\frac{\Delta C}{2C}\right) \tag{4.14}$$

This expression shows how the modified scaling factor m' relates to of the ideal value m in a less complex fashion than the exact result of Eq. (4.13). Note that this effect is independent of the initial operand voltages and is also similar to a multiply operation (Eq. (4.10)). Just as for share-only and multiply-only operations, systematic operand capacitor mismatch causes linear and correctable computation errors of magnitude proportional to the capacitor matching ratio. Also, the coefficient error percentage,

$\frac{m'-m}{m} = m\frac{\Delta C}{C}$, is always less than the operand capacitor mismatch ratio ($\frac{\Delta C}{C}$).

4.4.2 Stealing Capacitor Mismatch

A share operation does not suffer from stealing capacitor mismatch due to its lack of non-operand stealing capacitors. Due to the similarity between the analysis of multiply and share and multiply operations, only the case of a share and multiply is performed here.

For a 2-point share and multiply operation, as shown in Fig. 4.3(c), the ratio of the stealing capacitor, C_s, to the operand capacitors determines the scaling factor, m. The specific case under consideration is one of equal operand weights ($C_1 = C_2 = C$) and with initial voltages of V_1 and V_2 on the two operand capacitors. Also, as mentioned for operand capacitor mismatch, a non-zero initial value on the stealing capacitor C_s causes an offset error.

Any deviation (say, ΔC_s) of the stealing capacitor from its ideal value (C_s) changes the computation's effect from the ideal result in Eq. (4.11). The effect of stealing capacitor mismatch on this 2-point operation is shown in Eq. (4.15) below.

$$V_{out} = \left(\frac{2C}{2C + C_s + \Delta C_s}\right) \cdot \frac{V_1 + V_2}{2} = m' \cdot \frac{V_1 + V_2}{2} \tag{4.15}$$

The modified scaling factor for the operation, m', is expanded and simplified below.

$$\begin{aligned} m' &= \frac{1}{1 + \frac{C_s + \Delta C_s}{2C}} = \frac{1}{1 + \frac{C_s}{2C} + \frac{\Delta C_s}{2C}} \\ &= \left(\frac{1}{1 + \frac{C_s}{2C}}\right)\left(\frac{1}{1 + \left(\frac{\Delta C_s}{2C}\right)/\left(1 + \frac{C_s}{2C}\right)}\right) = m \cdot \left(\frac{1}{1 + m\frac{\Delta C_s}{2C}}\right) \end{aligned} \tag{4.16}$$

Assuming reasonable matching ($\Delta C_s \ll C_s$), the exact result for m' in Eq. (4.16) can be rewritten as shown below using the binomial approximation for the denominator.

$$m' \approx m \cdot \left(1 - m\frac{\Delta C_s}{2C}\right) \tag{4.17}$$

With stealing capacitor mismatch that is proportional to the operand capacitor mismatch ($\frac{\Delta C_s}{C_s} = \frac{\Delta C}{C}$), the coefficient error percentage, $\frac{m'-m}{m} = (1-m)\frac{\Delta C}{C}$, is always less than the capacitor mismatch ratio ($\frac{\Delta C}{C}$).

4.4.3 Conclusions

Assuming decent capacitor matching, the super-position principle can be used to combine the operand and stealing capacitor mismatch effects from Eqs. (4.14) and (4.17) to give the following result for a 2-point share and multiply operation.

$$m' = \left(\frac{2C+\Delta C}{2C+\Delta C + C_s+\Delta C_s}\right) \approx m \cdot \overbrace{\left(1 + (1-m)\frac{\Delta C}{2C}\right)}^{\text{operand mismatch}} \overbrace{\left(1 - m\frac{\Delta C_s}{2C}\right)}^{\text{stealing mismatch}} \tag{4.18}$$

This shows that multiplication ratio error due to capacitor mismatch is on the order of the capacitance matching ratio ($\frac{\Delta C}{C}$) and peaks for $m = 0.5$.

4.5 Incomplete Computation Settling

For a discrete-time signal processor in direct-RF or wideband-IF applications very high sampling and processing speeds are necessary. In a system with real-time processing, the speed of charge-domain operations must directly trade-off with the settling accuracy of the operations. Switch sizes can be increased for settling improvements at the expense of power consumption and increased charge-injection and clock-feedthrough errors.

To optimize the computation accuracy, analysis of the time-domain settling characteristics give useful insight. In this section, the settling equations for different types of charge-domain operations are determined. Settling speeds depend on the particular switch configuration and, consequently, different switch topologies can be contrasted by comparing their settling time-constants (τ_s and τ_d) and power (number of switches).

This section will consider two aspects of incomplete settling. First, the equations for the settling behavior of 2-point share and 2-point share and multiply single-ended operations are shown. They are written in a form that distinguishes the ideal result from the different types of error components. These are considered single-ended operations because the two input operands are each represented by a voltage on a single capacitor referenced to ground (or some common-mode voltage V_{cm}). Next, a psuedo-differential representation is introduced whereby operands are composed of a pair of capacitors holding postive and negative components of the value. The case of the 2-point share and multiply operation is used to show a similar settling equation form with pseudo-differential operands. Mitigation techniques, such as RC settling-error cancellation (RCX), are developed based on these models and the exploitable correlation between error sources. The impact of incomplete settling upon implementation considerations is covered in Chapter 6 with the design and optimization of CRAFT.

Intuition Incomplete settling causes the final values of charge-domain computations to be linearly related to the initial values by fixed settling factors. This is expressed in the following example relationship: $V(t_s) \propto V(0) \cdot e^{-t_s/\tau}$. Therefore, computation accuracy targets can be described as a required number-of-time-constants ($n = -t_s/\tau$) of settling. The exponential settling behavior gives an 8.68dB ($= -20 \log_{10} e$) reduction in error per settling time-constant. Since the error is relative the the initial voltage difference, the maximum error is bounded by the full scale operand swing V_{fs}.

$$\textit{Maximum Settling Error:} \qquad V_{fs} \cdot e^{-n} \;\Rightarrow\; n \times -8.68\text{dBFS}$$

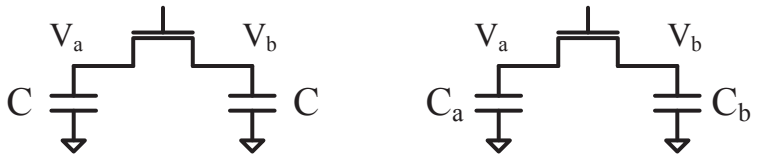
This means that while the dynamic range (computational accuracy) due to settling is input dependent, it can be modeled as an average value with specific variation and peak bound. The influence of voltage-dependent effects upon the actual amount of settling achieved is covered in Sec. 4.6.

4.5.1 Single-Ended Operand Settling

2-point Share Consider two operand capacitors, each with capacitance C and time-varying voltages of $V_a(t)$ and $V_b(t)$. A 2-point share operation, implemented as in Fig. 4.4(a), is defined by the system of differential equations (3.2) and has the settling response of Eq. (4.19). The input capacitors, holding initial values of v_{a0} and v_{b0}, are connected by a switch of resistance R_{sw} at $t = 0$. This yields a differential settling time-constant: $\tau_d = R_{sw}C/2$.

$$\begin{aligned} V_a(t) &= \tfrac{1}{2}\left(v_{a0} + v_{b0}\right) + \tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d} \\ V_b(t) &= \underbrace{\tfrac{1}{2}\left(v_{a0} + v_{b0}\right)}_{\text{ideal result}} - \underbrace{\tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d}}_{\text{settling error}} \end{aligned} \tag{4.19}$$

A weighted 2-point share operation, implemented as in Fig. 4.4(b), is also defined by the system of differential equations (3.2) and has the settling response of Eq. (4.20). The



(a) 2-point share (b) 2-point weighted share

Figure 4.4: Basic single-ended share operations used for settling analysis

weighting and settling factors are $w = C_a/(C_a{+}C_b)$ and $\tau = R_{sw}C_{eq}$; a switch of resistance R_{sw} is used and the series-connected equivalent capacitance is $C_{eq} = C_aC_b/(C_a{+}C_b)$.

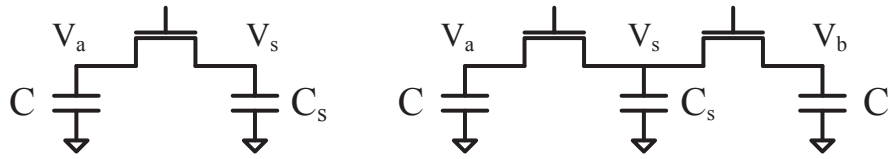
$$\begin{aligned} V_a(t) &= \overbrace{\big(w\,v_{a0} + (1{-}w)\,v_{b0}\big)}^{\text{weighted-sum term}} + \overbrace{\big(v_{a0} - v_{b0}\big)\cdot(1{-}w)\,e^{-t/\tau}}^{\text{difference term}} \\ V_b(t) &= \underbrace{\big(w\,v_{a0} + (1{-}w)\,v_{b0}\big)}_{\text{ideal result}} - \underbrace{\big(v_{a0} - v_{b0}\big)\cdot w\,e^{-t/\tau}}_{\text{settling error}} \end{aligned} \tag{4.20}$$

This result is consistent with Eq. (4.19) when $C_a = C_b = C$ for $w = 0.5$ and $C_{eq} = C/2$ in a equally-weighted share operation. Note that a weighted share is equivalent to a 1-point multiply operation where the stealing capacitor has some initial voltage.

1-point Multiply A 1-point multiply operation, as shown in Fig. 4.5(a), is defined by the system of differential equations (3.4) and has the settling response below.

$$\begin{aligned} V_a(t) &= \overbrace{v_{a0}\cdot m\big(1 + \big(\tfrac{1-m}{m}\big)e^{-t/\tau}\big)}^{\text{multiply term}} + \overbrace{v_{s0}\cdot(1{-}m)\,\big(1 - e^{-t/\tau}\big)}^{\text{offset term}} \\ V_s(t) &= \underbrace{v_{a0}\cdot m}_{\text{ideal}}\underbrace{\big(1 - e^{-t/\tau}\big)}_{\text{scale eror}} + \underbrace{v_{s0}\cdot(1{-}m)}_{\text{ideal}}\underbrace{\big(1 + \big(\tfrac{m}{1-m}\big)e^{-t/\tau}\big)}_{\text{scale error}} \end{aligned} \tag{4.21}$$

The multiplication and settling factors are $m = C/(C{+}C_s)$ and $\tau = R_{sw}C_{eq}$ for a switch of resistance R_{sw} and series-connected equivalent capacitance $C_{eq} = C\,C_s/(C{+}C_s)$. Usually, the offset term in not desired and is removed when $v_{s0} = 0$. The Eqs. (4.20) and (4.21) are effectively identical due to implementation similarities between Figs. 4.4(b) and 4.5(a).



(a) 1-point multiply (b) 2-point share and multiply

Figure 4.5: Basic single-ended multiply operations used for settling analysis

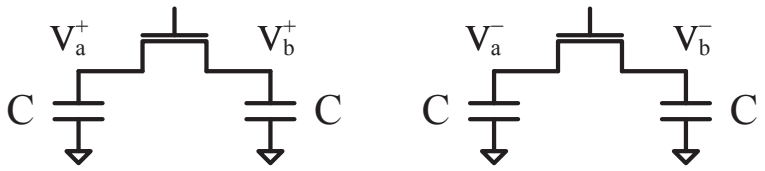
2-point Share and Multiply A 2-point share and multiply operation, as shown in Fig. 4.5(b), is defined by the system of differential equations (3.6) and has the settling response of Eq. (4.22). The input capacitors (C), with initial values v_{a0} and v_{b0}, are connected to the stealing capacitor (C_s, with no initial value[4]) by switches of resistance R_{sw}. The input-to-stealing-capacitor ratio defines the scaling factor of the operation: $m = 2C/(2C+C_s)$.

$$\begin{aligned} V_a(t) &= \overbrace{\tfrac{1}{2}\left(v_{a0}+v_{b0}\right) m \left[1+\left(\tfrac{1-m}{m}\right) e^{-t/\tau_s}\right]}^{\text{scaled-sum term}} + \overbrace{\tfrac{1}{2}\left(v_{a0}-v_{b0}\right) e^{-t/\tau_d}}^{\text{difference term}} \\ V_b(t) &= \underbrace{\tfrac{1}{2}\left(v_{a0}+v_{b0}\right) m}_{\text{ideal result}} \underbrace{\left[1+\left(\tfrac{1-m}{m}\right) e^{-t/\tau_s}\right]}_{\text{scaling error factor}} - \underbrace{\tfrac{1}{2}\left(v_{a0}-v_{b0}\right) e^{-t/\tau_d}}_{\text{difference settling error}} \end{aligned} \tag{4.22}$$

The difference and sum-settling time-constants for a 2-point share and multiply operation (2-switch variant), τ_d and τ_s respectively, are

$$\tau_d = R_{sw}C \qquad\qquad \begin{aligned} \tau_s &= R_{sw}\cdot\left(C \parallel C_s/2\right) \\ &= R_{sw}C\cdot(1-m) \end{aligned} \tag{4.23}$$

[4]Initial stealing capacitor value $v_{s0} = 0$, or $v_{s0} = V_{cm}$, where V_{cm} is the common mode voltage used as reference for all operations.



(a) Positive-half (b) Negative-half

Figure 4.6: Pair of 2-point operations for computing with psuedo-differential operands

Error Reduction

Settling error is reduced if switches are widened, capacitance is decreased, or switch on-times are increased. These are standard trade-offs without any being overly advantageous. Adding additional switches to the circuit network can provide settling improvement in a power-efficient manner. Depending on the nodes connected, different configurations can be formed. Moreover, the extra switches' resistance can be different than the base set of switches to provide a useful knob in the settling vs. power trade-off. This technique was to optimization the performance of CRAFT (Chapter 6); the beneficial impact is shown in Eq. (6.12) and its associated text.

4.5.2 Pseudo-Differential Operand Settling

Pseudo-differential operands are described by the relationships of Eq. (4.24), where V^+ and V^- are the voltages on separate capacitors for the positive and negative components of the operand (referenced to ground or a common-mode voltage V_{cm}).

$$\begin{aligned} V_A &= V_a^+ - V_a^- \\ V_B &= V_b^+ - V_b^- \end{aligned} \tag{4.24}$$

Consequently, charge-domain operations are performed separately on these components, as is shown in Figs. 4.6 and 4.8. An example time-domain settling response of these differential operands is shown in Fig. 4.7.

Since a share operation is the same as a share and multiply with $m=1$ ($C_s=0$), the settling behavior of a pseudo-differential 2-point share and multiply operation will be determined. As shown in Eq. (4.24), the pseudo-differential operand $V_A(t)$ is composed of two share and multiply settling operations. Using Eq. (4.22) with variables $m' = m\left[1+\left(\frac{1-m}{m}\right)e^{-t/\tau_s}\right]$ and $\epsilon = \frac{1}{2}\left(v_{a0}-v_{b0}\right)e^{-t/\tau_d}$, Eq. (4.24) becomes:

$$\begin{aligned} V_A(t) &= \overbrace{\left[\tfrac{1}{2}\left(v_{a0}^{+}+v_{b0}^{+}\right)m'^{+}+\epsilon^{+}\right]}^{V_a^{+}(t)} - \overbrace{\left[\tfrac{1}{2}\left(v_{a0}^{-}+v_{b0}^{-}\right)m'^{-}+\epsilon^{-}\right]}^{V_a^{-}(t)} \\ &= \left(v_{a0}^{+}+v_{b0}^{+}\right)\underbrace{\tfrac{1}{2}(m'^{+}+m'^{-})}_{m'_d} + \underbrace{(\epsilon^{+}-\epsilon^{-})}_{\epsilon_d} \end{aligned} \tag{4.25}$$

The differential nature ($v_{a0}^{+} = -v_{a0}^{-}$ and $v_{b0}^{+} = -v_{b0}^{-}$) allows this share and multiply expression to be written in a form similar to the single-ended expressions $V_a^{+}(t)$ and $V_a^{-}(t)$ within Eq. (4.25) ($m' = m'_d = 1$ for a share operations).

$$\begin{aligned} V_A(t) &= \left(v_{a0}^{+}+v_{b0}^{+}\right)m'_d + \epsilon_d \\ V_B(t) &= \left(v_{a0}^{+}+v_{b0}^{+}\right)m'_d - \epsilon_d \end{aligned} \tag{4.26}$$

For the differential operands, scaling error and inter-copy error, m'_d and ϵ_d shown below, now include the settling effects of the positive and negative components of the pseudo-differential representation.

$$\begin{aligned} m'_d &= \tfrac{1}{2}(m'^{+}+m'^{-}) = m\left[1+\left(\tfrac{1-m}{m}\right)\tfrac{1}{2}(e^{-t/\tau_s^{+}}+e^{-t/\tau_s^{-}})\right] \\ \epsilon_d &= (\epsilon^{+}-\epsilon^{-}) = \left(v_{a0}^{+}-v_{b0}^{+}\right)\tfrac{1}{2}(e^{-t/\tau_d^{+}}+e^{-t/\tau_d^{-}}) \end{aligned} \tag{4.27}$$

Under small operand swings $\tau_d^{+} \approx \tau_d^{-}$ and the differential error's relationship to the two single-ended operations (positive and negative) is clearest: $m'_d \approx m'$ and $\epsilon_d \approx 2\epsilon$.

Error Reduction

By recognizing that the differential settling error (ϵ) between pairs of operands components, V_a^+ and V_a^-, and V_b^+ and V_b^-, is differentially correlated a method for its cancellation is proposed. This technique is called RC settling error cancellation (RCX) and is introduced here; further detail and analysis can be found in the mitigation discussion of Sec. 6.4.3.

Since V_a^- and V_b^- settle to the same value with error ϵ of opposite sign, they can be interchanged to form the new differential operands (compare Eqs. (4.24) and (4.28)).

$$\begin{aligned} V_{A,\mathrm{RCX}}(t) &= V_a^+(t) - V_b^-(t) \\ V_{B,\mathrm{RCX}}(t) &= V_b^+(t) - V_a^-(t) \end{aligned} \tag{4.28}$$

The effect of the reassignment of these operand components is shown in Fig. 4.7 and can be implemented using only wire-swapping. As shown, swapping the output wires between the operands greatly reduces differential settling error. While the pseudo-differential "inputs" $(+,-)$ to the operation are (V_a^+, V_a^-) and (V_b^+, V_b^-), the capacitors holding the "output" results (two copies) are (V_a^+, V_b^-) and (V_b^+, V_a^-).

This rewiring causes only a slight change to Eq. (4.26), but it can still be written in the same form, as shown in Eq. (6.14). This technique provides perfect cancellation[5] when the settling constants of the positive and negative operations, τ^+ and τ^-, are equal. Due to mismatch, parasitics, and voltage-dependence effects, as described in Sec. 4.6, this is never achieved. Nevertheless, it can be shown that RCX always yields an improvement (Eq. (6.15)); this is also illustrated visually in the simulation results of the following section.

[5]The inter-copy error is not actually being canceled, it is just translated into a common-mode component. In case a stage with common-mode rejection follows, this settling error is rejected.

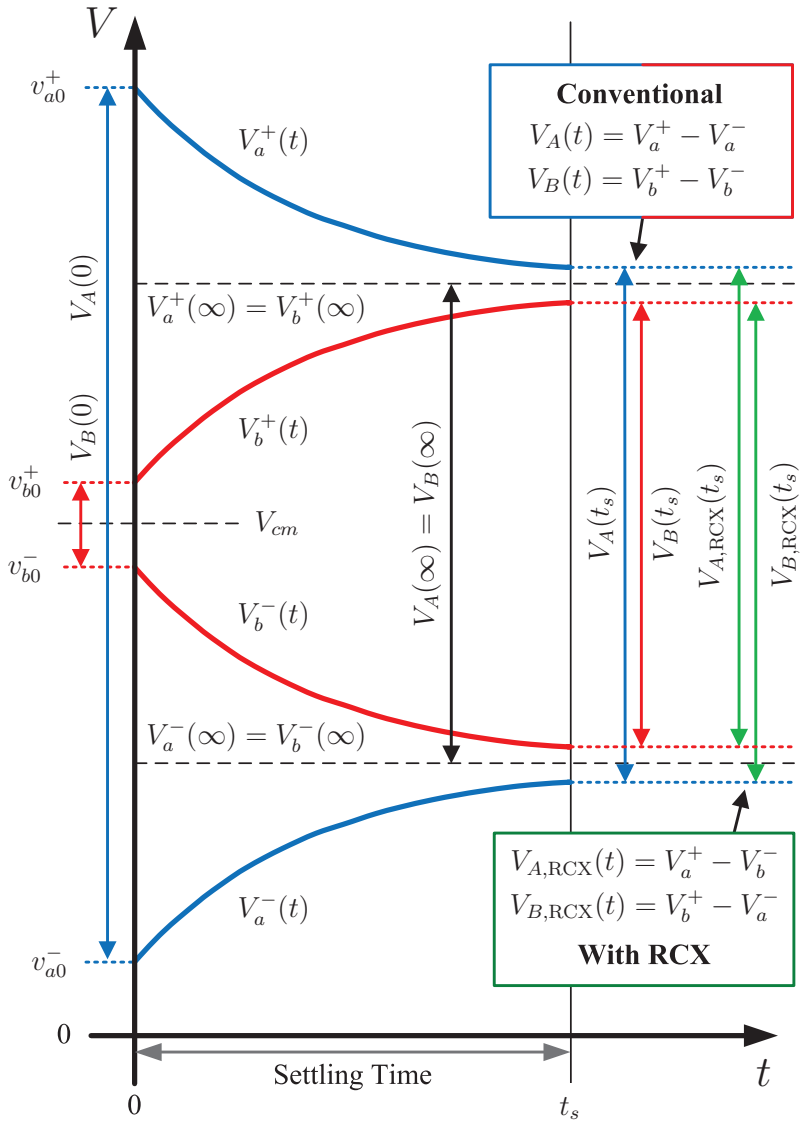


Figure 4.7: An example pseudo-differential charge-domain operation (share, m=1) settling

4.6 Voltage-dependent Settling

In Sec. 4.5 it was show how incomplete settling defines the final values of charge-domain computation in relatation to the initial values and fixed settling factors of the form $e^{-t_s/\tau}$. If the circuit were to behave ideally, as a simple network of matching switch-resistances and capacitors, this would be a linear and predictable error. However, the on-resistance of the nMOS processing switches are dependent on operand voltages. Deviation from the nominal switch resistance value, $R_{sw,\text{nom}}$ shown below, causes a range of settling time-constants to be realized depending on the operand value combinations.

$$R_{sw,\text{nom}} = R_{triode}(V_G, V_D, V_S) = R_{triode}(V_{dd}, V_{cm}, V_{cm})$$

Furthermore, the time-variation of node voltages cause each R_{sw} to vary during the charge-domain computations. For example, a 2-point share operation (Fig. 4.4(a)) has one switch with resistance shown below.

$$R_{sw}(t) = R_{triode}(V_G(t), V_D(t), V_S(t)) = R_{triode}(V_{dd}, V_a(t), V_b(t)) \tag{4.29}$$

Consequently, systems of differential equations for share, multiply, and share and multiply operations, Eqs. (3.2), (3.4), and (3.6) respectively, no longer have solutions as previously shown, which assume no time or node voltage dependence of τ. This also transforms incomplete settling from a purely linear error into a nonlinear error.

Intuition While the impact of resistance variation upon the realized settling factor is important, it is also critical to consider the magnitude of the error. For example, if two input operands are high voltages, this will increase nMOS switch resistances and negatively impact the settling-factor. However, if the difference between the input values is small, this will reduce the magnitude of the final error as it is a product of this initial difference and the settling factor. Therefore, accurate modeling of the impact

of voltage-dependent settling effects is necessary to aid simulation based verification of computation error at a more abstract level than device-level system simulations.

4.6.1 Modeling

To avoid the complexity and simulation time of circuit and device-level simulations, settling behavior can be modeled using iterative numerical evaluation of the differential equations for the circuit network. The use of a small time-step ($\Delta t \ll t_s$) followed by re-computation of switch resistance values provides faithful results. For additional speed improvement, the time-step size can be variable as switch resistances will not change drastically once node voltages have partially settled.

Alternatively, if small time-steps guarantee that time-constants are relatively constant ($\tau(t+\Delta t) \approx \tau(t)$), the differential equations' solution can be directly applied in an iterative manner. This method provides additional simulation speed improvements while still allowing control of simulation accuracy through control of the step size (Δt). It also provides a useful level of abstraction from the circuit details as only the value of τ changes depending on the implementation while the exponential form remains the same. This method becomes less accurate for more complex implementations with large numbers of operands and switches. This is due to the fact that $\tau_s(t)$ and $\tau_d(t)$ in the settling equations become complex functions of many instantaneous switch resistances. The evaluation of these values gives a network-wide settling factor rather than representing the settling factors between different pairs of nodes in the implemenation as a full iterative differential equation solving process would.

The following text describes how the simulated settling results are linked with the settling-constant based design methodology utilized. The cases of a 2-point share and a 2-point share and multiply operation are shown. This design methodology is fundamentally summarized by stating that each operation is intended to have $n = t_s/\tau_{\text{nominal}}$

time-constants of exponential settling. The evaluation of the actual distribution of $n' = t_s/\tau_{\text{effective}}$ helps validate that the desired computation accuracy bound is not exceeded. Also, the direct impact of trade-offs between signal swing and supply voltage can be characterized as shown in the simulation examples under different operating conditions like Figs. 4.9–4.14.

Iterative Exponential Solution

For a 2-point share operation (ie. Fig. 4.4(a)) the settling response of Eq. (4.19) is modified to (4.30) and uses the instantaneous settling factor $\tau_d(t) = R_{sw}(V_{dd}, V_a(t), V_b(t)) \cdot C/2$. Note that the constant $\tau_d(t_i)$ is used to compute the new voltages, V_a and V_b, at t_{i+1} assuming non-varying switch resistances.

$$\begin{aligned} V_a(t_{i+1}) &\approx \left(\tfrac{V_a(t_i)+V_b(t_i)}{2}\right) + \left(\tfrac{V_a(t_i)-V_b(t_i)}{2}\right) e^{-(t_{i+1}-t_i)/\tau_d(t_i)} \\ V_b(t_{i+1}) &\approx \left(\tfrac{V_a(t_i)+V_b(t_i)}{2}\right) - \left(\tfrac{V_a(t_i)-V_b(t_i)}{2}\right) e^{-(t_{i+1}-t_i)/\tau_d(t_i)} \end{aligned} \tag{4.30}$$

For a 2-point share and multiply operation (ie. Fig. 4.5(b)), the settling time constants τ_s and τ_d of Eq. (4.23) are valid only when $R_{sw,i} = R_{sw,\text{nom}}$. In actuality they are functions of the instantaneous switch resistances, as expanded below when implemented with two switches, which in turn depend on node voltages as in Eq. (4.29).

$$\begin{aligned} \tau_s(t) &= \tau_s(R_{sw,1}(t), R_{sw,2}(t)) \\ \tau_d(t) &= \tau_d(R_{sw,1}(t), R_{sw,2}(t)) \end{aligned} \tag{4.31}$$

Just as for the share operation, the exponential settling solution Eq. (4.22) can be utilized iteratively as shown below.

$$\begin{aligned} V_a(t_{i+1}) &\approx \left(\tfrac{V_a(t_i)+V_b(t_i)}{2}\right) m \left[1 + \left(\tfrac{1-m}{m}\right) e^{-(t_{i+1}-t_i)/\tau_s(t_i)}\right] + \left(\tfrac{V_a(t_i)-V_b(t_i)}{2}\right) e^{-(t_{i+1}-t_i)/\tau_d(t_i)} \\ V_b(t_{i+1}) &\approx \left(\tfrac{V_a(t_i)+V_b(t_i)}{2}\right) m \left[1 + \left(\tfrac{1-m}{m}\right) e^{-(t_{i+1}-t_i)/\tau_s(t_i)}\right] - \left(\tfrac{V_a(t_i)-V_b(t_i)}{2}\right) e^{-(t_{i+1}-t_i)/\tau_d(t_i)} \end{aligned} \tag{4.32}$$

Effective Settling-constants

From the simulated node voltages, effective settling time-constants, $\tau_{s,\text{eff}}$ and $\tau_{d,\text{eff}}$, can be calculated for any specific time instant. When substituted into the original settling equations, Eqs. (4.19)–(4.22) (example shown below for Eq. (4.22)), they provide the same answers at time t as the results with switch resistance variation.

$$\begin{aligned} V_a(t) &= \tfrac{1}{2}\left(v_{a0}+v_{b0}\right) m \left[1+\left(\tfrac{1-m}{m}\right)e^{-t/\tau_{s,\text{eff}}(t)}\right] + \tfrac{1}{2}\left(v_{a0}-v_{b0}\right) e^{-t/\tau_{d,\text{eff}}(t)} \\ V_b(t) &= \tfrac{1}{2}\left(v_{a0}+v_{b0}\right) m \left[1+\left(\tfrac{1-m}{m}\right)e^{-t/\tau_{s,\text{eff}}(t)}\right] - \tfrac{1}{2}\left(v_{a0}-v_{b0}\right) e^{-t/\tau_{d,\text{eff}}(t)} \end{aligned} \tag{4.33}$$

The value of τ_{eff} at time t is not the instantaneous time-constant (i.e. $\sim R_{sw}(t)\cdot C$), but rather the effective settling-factor ($e^{-t/\tau_{\text{eff}}(t)}$) that has occurred relative to $t=0$. They are calculated as follows for a 2-point share and multiply operation using Eq. (4.33) ($V_a(0)=v_{a0}$ and $V_b(0)=v_{b0}$),

$$\begin{aligned} \tau_{s,\text{eff}}(t) &= -t/\ln\left[\frac{(V_a(t)+V_b(t)) - m(V_a(0)+V_b(0))}{(1-m)(V_a(0)+V_b(0))}\right] \\ \tau_{d,\text{eff}}(t) &= -t/\ln\left[\frac{(V_a(t)-V_b(t))}{(V_a(0)-V_b(0))}\right] \end{aligned} \tag{4.34}$$

Heretofore referenced as simply τ_s ($=\tau_{s,\text{eff}}(t_s)$) and τ_d ($=\tau_{d,\text{eff}}(t_s)$), these effective time-constants lump the effects of voltage and time variation, as well as resistance differences in multi-switch configurations rather than referring to the instantaneous time-constant value (i.e. $f(R_{sw,i}(t))\cdot C$). This reinterpretation allows the analysis and notation of Eqs. (4.26), (6.14), and (6.15) to be retained while still keeping the familiar number-of-time-constants settling form as seen in Eq. (4.33).

4.6.2 Simulations

Using the modeling techniques described in the previous section, settling simulations including voltage-dependent switch resistance are shown here. From these visual results,

the impact of switch nonlinearity upon the realized setting values, as well as trade-offs of different settling amounts, supply voltage, and signal amplitudes are more easily appreciated and interpreted.

The particular case examined consists of a pair of 2-point share operations to perform calculations, as seen in Fig. 4.8. A pseudo-differential representation is used because it is beneficial for mitigating or cancelling other error sources described in this chapter, as well as allowing RCX error cancellation to be implemented by wire-swapping.

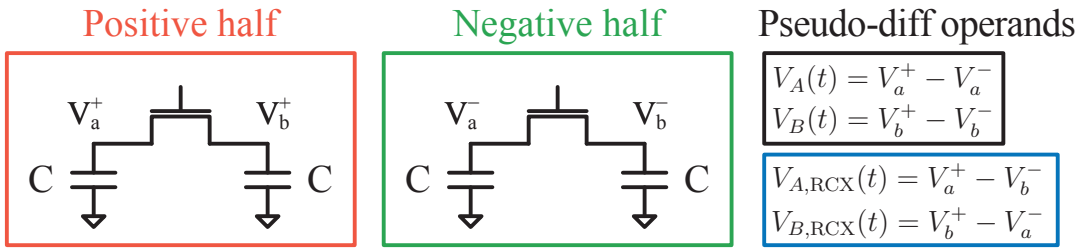
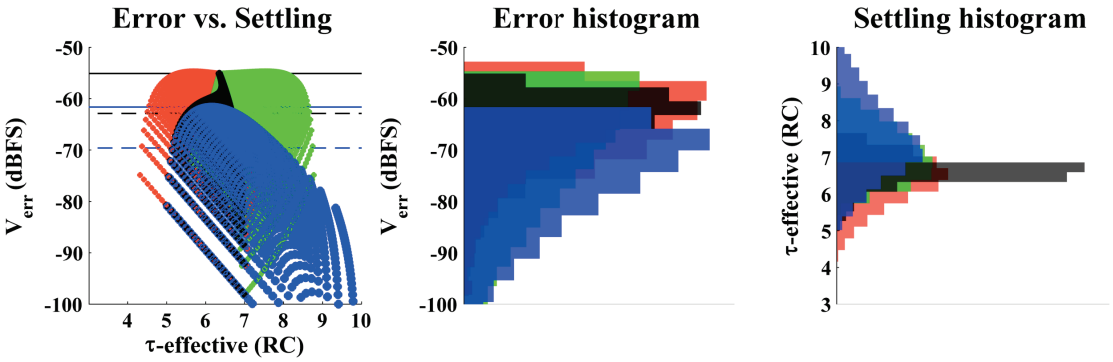


Figure 4.8: A pair of 2-pt share operations for psuedo-differential computations

The color coding in all simulation results of this section signify the following: red and green indicate the positive and negative halves of the differential representation; black is the psuedo-differential result constructed in Eq. (4.24) and repeated in Fig. 4.8; blue is the differential operand from Eq. (4.28) which performs the RCX error cancellation technique developed in Sec. 4.5.2.

For a 2-point share operation, the amount of settling on the outputs is determined by calculating the remaining differential error among the outputs (see Eq.(4.19)). The differential error is calculated as shown in Eq. (4.35) for the four cases plotted in the simulation result here. This error can be used to calculate the effective settling, $\tau_{d,\mathrm{eff}}$, in a manner similar to Eq. (4.34).

$$\begin{aligned} &\textbf{Positive:} && V_{\mathrm{err}} = V_a^+ - V_b^+ && \textbf{Pseudo-diff:} && V_{\mathrm{err}} = V_A - V_B \\ &\textbf{Negative.:} && V_{\mathrm{err}} = V_a^- - V_b^- && \textbf{RCX P-diff:} && V_{\mathrm{err}} = V_{A,\mathrm{RCX}} - V_{B,\mathrm{RCX}} \end{aligned} \tag{4.35}$$



Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $t_s/\tau_d = 7$, $V_{dd} = 1.35$V, $V_{cm} = 0.3$V, and $V_{in\text{-se}} = V_{FS\text{-se}} = \pm V_{cm}$, IBM 65nm CMOS

Figure 4.9: 2-point share settling spreading

Alternatively, the settling error can be reference to the maximum operand swing, V_{FS}, to calculate the settling in a manner that relates to the computational dynamic range. When referenced to full-scale (i.e. dBFS), the value of V_{FS} depends on whether it is a single-ended or differential result. For single-ended operands $V_{FS\text{-se}} = \pm V_{cm}$ (for nMOS-only designs), and for differential operations $V_{FS\text{-diff}} = 2V_{FS\text{-se}}$.

MATLAB® simulations of every input combination, with operands varying over their full swing range (excluding cases of input and differential symmetry), are shown in Fig. 4.9. This plots represents 2,000 independent pseudo-differential non-linear settling computations, where each one is a single point on the error vs. settling graph. The variation in setting constant (τ), due to switch resistance variation, spreads the results horizontally; the error magnitude is influence by both the settling amount and the initial difference between the inputs to the computations. Histograms of the error magnitude and effective settling are also shown. The simulation conditions are listed just below the figure. Solid lines in the error vs. settling diagram represent the peak error of differential (black) and differntial+RCX (blue) results. The dashed lines mark the average (rms) error levels. These lines correspond with the levels seen in the error histogram. The difference between the black and blue lines represents the net benefit of the RCX error

cancellation technique.

The complex nature of nonlinear computation settling, combined with the number of design parameters that influence it (t_s/τ, V_{dd}, V_{cm}, and V_{FS}), make simulations invaluable. The effects of non-linear settling on dynamic range is quantified by the peak and rms computation error. Additional simulations and their application in charge-domain computation dynamic range optimization is shown in the following section.

4.6.3 Optimization

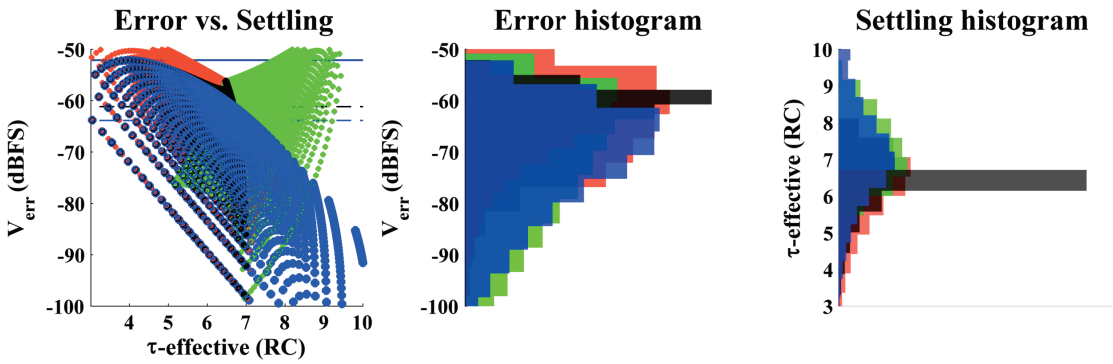
By examining the non-linear settling behavior of individual charge-domain operations, optimization of system-level design parameters can be performed. This is useful to ensure the maximum computational error goal is not exceeded under any condition. The following are several examples showing the impact of different tradeoffs

Setting vs. Design Parameter Trade-offs

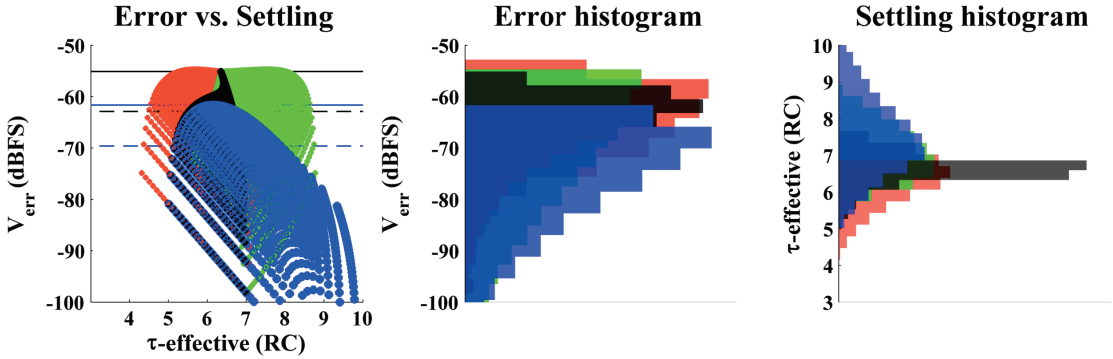
1. *Supply voltage* (V_{dd}): Fig. 4.10 shows improving settling and reduced variation with increasing supply voltage. As V_{dd} increases, so does the switch turn-on voltage headroom. This reduces switch resistance variation. Results are also shown side-by-side in Fig. 4.11 for more direct comparison. Energy per computation increases by $(V_{dd}-V_{tn}-V_{cm})^{0.5}$ for these 65nm switches[6].

2. *Settling-factor* (t_s/τ): Fig. 4.12 shows improving settling with increased settling factor. This is accomplished by: allowing more settling time (t_s), reducing switch resistance (increasing width), or reducing operand capacitance. Greater settling factors require a higher V_{dd} to reduce settling-factor spread to fully realize their potential. Energy per computation increases by $(V_{dd}-V_{tn}-V_{cm})^{0.5}$ and inversely to switch resistance for these 65nm switches.

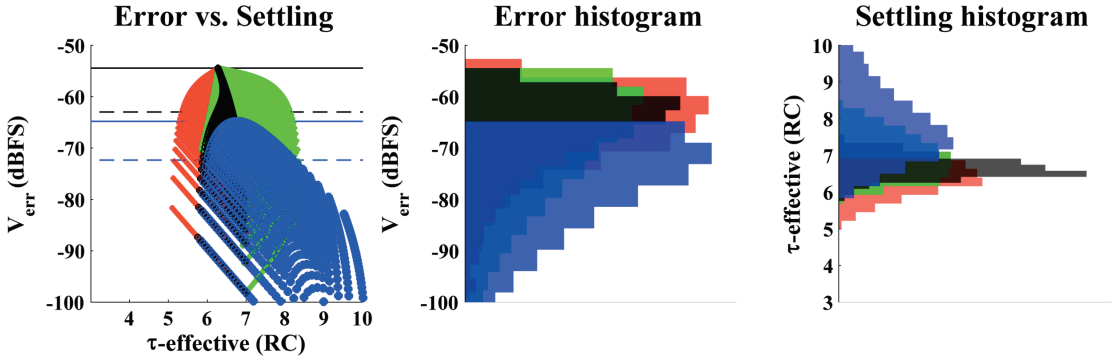
[6] An empirical model for triode switch resistance in this technology is discussed in Chapter 6



(a) $V_{dd} = 1.20$V



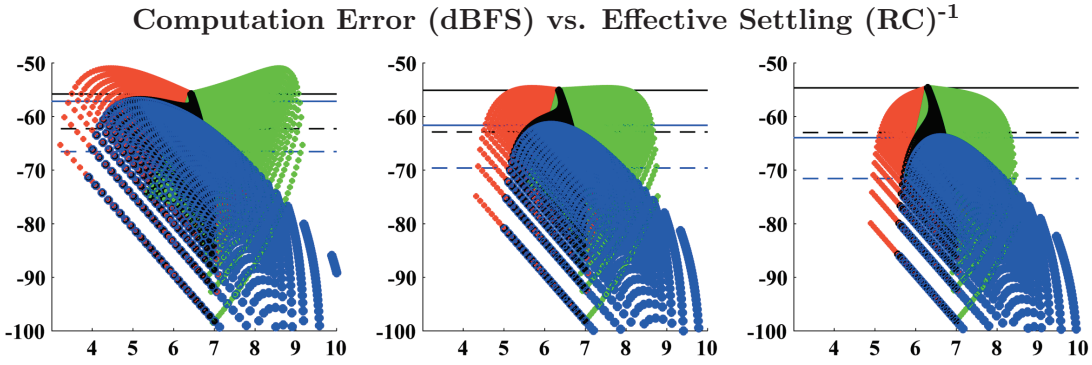
(b) $V_{dd} = 1.35$V



(c) $V_{dd} = 1.50$V

Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $t_s/\tau_d = 7$, $V_{cm} = 0.30$V, $V_{in\text{-se}} = V_{FS\text{-se}} = \pm V_{cm}$, IBM 65nm CMOS

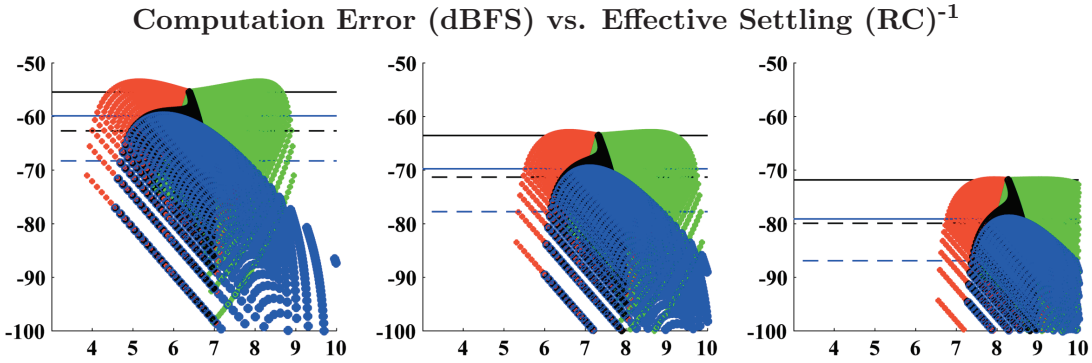
Figure 4.10: Settling error/spread vs. V_{dd} (with histograms)



(a) $V_{dd}=1.25$V (b) $V_{dd}=1.35$V (c) $V_{dd}=1.45$V

Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $t_s/\tau_d=7$, $V_{cm}=0.30$V, $V_{in\text{-se}}=V_{FS\text{-se}}=\pm V_{cm}$, IBM 65nm CMOS

Figure 4.11: Settling error/spread vs. V_{dd}



(a) $t_s/\tau_d=7$, $V_{dd}=1.30$V (b) $t_s/\tau_d=8$, $V_{dd}=1.40$V (c) $t_s/\tau_d=9$, $V_{dd}=1.50$V

Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $V_{cm}=0.30$V, $V_{in\text{-se}}=V_{FS\text{-se}}=\pm V_{cm}$, IBM 65nm CMOS

Figure 4.12: Settling error/spread vs. t_s/τ_d

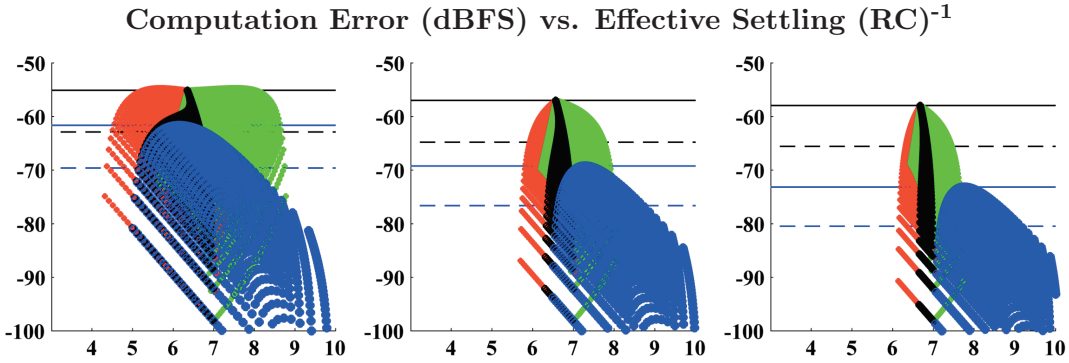
3. *Signal swing* ($V_{FS} \propto V_{cm}$): Fig. 4.13 shows improved settling and reduced variation with reducing full-scale signal swing. This increases switch voltage headroom and reduces resistance variation. It comes at the expense of SNR reduction as the maximum signal amplitude decreases. These results show a 12dB improvement in peak RCX error for a 6dB degradation in SNR (0.3V to 0.15V).

4. *Signal amplitude* (V_{in}): Fig. 4.14 shows improved settling and reduced variation with reduced signal amplitude (constant V_{FS} and V_{cm}). This represents how error is affected by input signal levels. Inputs are not always 0dBFS and previous share and multiply operations may have attenuated the input levels. A 9dB improvement in peak RCX error (dBFS) is seen for a 6dB degradation in SNR (0.3V to 0.15V). Relative to the maximum signal level, the computational nonlinearity is 15dB (=9dB+6dB) lower for a −6dBFS input (6dB SNR penalty).

4.6.4 Conclusions

The effect of the voltage dependence of switch resistance in charge-domain operations was examined. Modeling techniques and simulations were used to perform computational accuracy optimization. The effectiveness of the RCX error cancellation technique was verified, even in the precense of large variations in settling-factor.

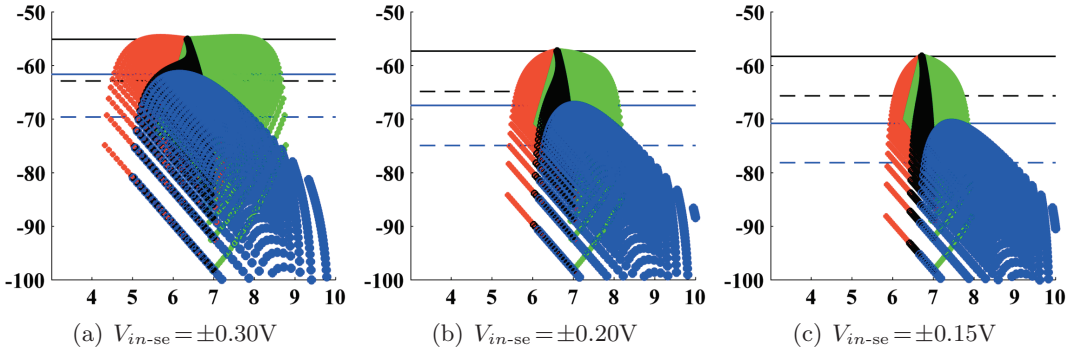
System-wide simulation, with each operation simulated as was done here, are used to optimize the nonlinearity of CRAFT in Chapter 6. RCX is used in all the CRAFT butterflies, as seen in Fig. 6.8, and simulations show a net improvement of about 10dB.



(a) $V_{in\text{-se}} = \pm 0.30$V (b) $V_{in\text{-se}} = \pm 0.20$V (c) $V_{in\text{-se}} = \pm 0.15$V

Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $t_s/\tau_d = 7$, $V_{dd} = 1.35$V, $V_{cm} = |V_{in\text{-se}}|$, $V_{FS\text{-se}} = V_{in\text{-se}}$, IBM 65nm CMOS

Figure 4.13: Settling error/spread vs. $V_{in\text{-se}} = \pm V_{cm}$



(a) $V_{in\text{-se}} = \pm 0.30$V (b) $V_{in\text{-se}} = \pm 0.20$V (c) $V_{in\text{-se}} = \pm 0.15$V

Legend: Pos. (red), Neg. (green), Pseudo-diff (black), RCX P-diff (blue) (see Eq. (4.35))
2-point share: $t_s/\tau_d = 7$, $V_{dd} = 1.35$V, $V_{cm} = 0.30$V, $V_{FS\text{-se}} = V_{in\text{-se}}$, IBM 65nm CMOS

Figure 4.14: Settling error/spread vs. $V_{in\text{-se}}$ (constant V_{cm})

4.7 Conclusion

The analysis and simulation of important error sources were covered in this chapter. Linear error sources can be corrected using digital means. However, non-linear error, such as voltage-dependent settling, must be analyzed, modeled, and optimized. For non-linear errors, finding the mean and variation of their effect upon the computations allows them to be incorporated into models of charge-domain system (such as Appendix B) using statistical approximations. The analysis and simulations of this chapter, in conjunction with the measurements of the CRAFT design in Chapter 6, are convincing of the fact that passive switch-capacitor processing can be optimized to a high-level of performance.

Chapter 5

Charge-Domain Computation Noise Analysis

5.1 Introduction

In any analog system, noise is a critical issue to consider as it sets the signal-to noise ratio (SNR) of the system. In discrete-time, passive, switched-capacitor processing, the integrated noise level is set by the size of the capacitors. After the switches open, noise manifests as a final-value distrubance with power (variance) of $\overline{q_n^2} = \mathrm{kT}C$ (charge) or $\overline{v_n^2} = \mathrm{kT}/C$ (voltage), where k is the Boltzmann constant, and T is the temperature in Kelvin.

Over many noise events, the sampled noise is a random variable related to an RMS charge or voltage disturbance in the following manner:

$$Q_{n,rms} = \sqrt{\overline{q_n^2}} = \sqrt{\mathrm{kT}C} \qquad \text{or} \qquad V_{n,rms} = \sqrt{\overline{v_n^2}} = \sqrt{\mathrm{kT}/C} \tag{5.1}$$

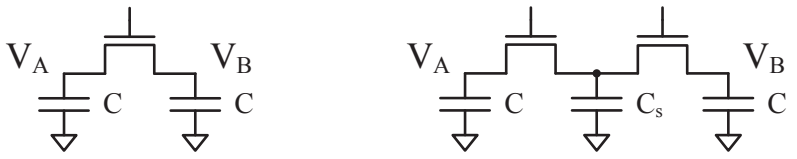
By considering the effects of many independent noise events in a system, a total RMS noise level can be calculated. Compared to the full-scale signal RMS level of a single

sinusoidal tone, this usually defines the maximum SNR. The noise calculations from this section were used to model the system noise throughout the four cascaded processing stages of CRAFT, as described in Sec. 6.

The analyses of both *processing* and *reset* noise are covered separately below. The magnitudes of the noise as well as any correlation between operand output copies is calculated. This is followed by an analysis of the total noise results for different types of operations and outputs.

5.2 Processing Noise

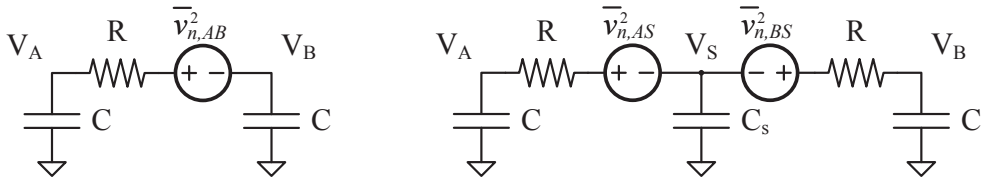
The output noise at the end of a charge-domain computation is called *processing* noise. It is treated separately from any noise in the intial voltages of any of the computation capacitances. Processing noise of share, multiply, and share and multiply operations are calculated in the following sections.



(a) Share (b) Share and multiply

Figure 5.1: 2-point operations used for processing noise analysis

A brief description of the analysis technique is as follows. First, the charge-domain operations shown in Fig. 5.1 are drawn with switches (of on-resistance R) and voltage noise sources in Fig. 5.2. Then, transfer functions are calcualted from each noise source, to each output node. These noise transfer functions (NTFs) are inherently low-pass. After integration over all frequencies, the noise power contribution of each source, to each node, is determined.



(a) Share (b) Share and multiply

Figure 5.2: 2-point operations showing switch on-resistance and noise sources

When the switches open ($R \to \infty$) the system bandwidth goes to zero (DC) and the instantaneous value of the noise-induced voltage fluctuation of each node is sampled. Therefore, the integrated noise power from each noise source, representing the variance of an independet variable, can be summed to find the total sampled noise power on each node. Additionally, there may be correlation between the total noise on output copies depending on the NTFs.

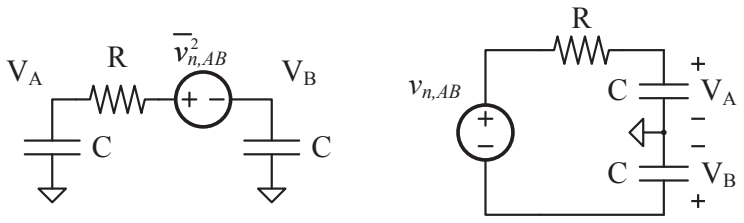
5.2.1 Share Operations

The processing noise of a share operation is easily calculated for a simple 2-point implementation, as seen in Fig. 5.1(a). The input/output operands are held on equivalently-sized capacitors C at the nodes V_A and V_B.

The circuit diagram for a 2-point share operation, including noise source, is drawn in Fig. 5.3(a). To find the noise contribution to each operand capacitor, the NTFs are easily determined from the redrawn circuit network of Fig. 5.3(b). The operand capacitors connect in series to define the system low-pass pole and a capacitive voltage divider separates the noise contribution among the two capacitors. The NTF responses are written in the Laplace domain ($s = j\omega$) below.

$$\frac{V_{n,A}}{v_{n,AB}}(s) = \frac{1}{2}\left(\frac{1}{1+sRC/2}\right) \qquad \text{and} \qquad \frac{V_{n,B}}{v_{n,AB}}(s) = -\frac{V_{n,A}}{v_{n,AB}}(s) \tag{5.2}$$

Next, the transfer function is integrated to find the noise power contributed to node A



(a) Network for NTF analysis (b) Redrawn

Figure 5.3: 2-point share with noise source (a) and redrawn (b) for easier NTF analysis

due to the the noise source (switch) between nodes A and B.

$$\overline{v_{n,AB\to A}^2} = 2\int_0^\infty \left|\frac{V_{n,A}}{v_{n,AB}}(j\omega)\right|^2 \mathrm{d}\omega.$$

Likewise, the integral of the NTF to node B is done in a similar fashion. This results in the following noise power quantities:

$$\overline{v_{n,AB\to A}^2} = \overline{v_{n,AB\to B}^2} = \frac{1}{2}\cdot\frac{\mathrm{kT}}{C} \tag{5.3}$$

Furthermore, the total noise at each output node is a sum of all contributing sources. A 2-point share has only one noise source, which gives the following relationships:

$$\overline{v_{n,A}^2} = \overline{v_{n,AB\to A}^2}$$
$$\overline{v_{n,B}^2} = \overline{v_{n,AB\to B}^2}$$

In summary, the total processing noise of a 2-point share operation with equal operand capacitors, as shown in Fig. 5.1(a), is written in Eq. (5.4) below.

$$\overline{v_{out,\mathrm{proc}}^2} = \overline{v_{n,A}^2} = \overline{v_{n,B}^2} = \frac{1}{2}\cdot\frac{\mathrm{kT}}{C} \qquad \text{and} \qquad \rho_{A,B} = -1 \tag{5.4}$$

Additionally, the output operands have perfectly differential correlation ($\rho_{A,B,\mathrm{proc}}$) since their NTFs (Eq. (5.2)) have opposite magnitude and filter the same noise source.

Weighted Share Operation

In a more general case, when the operand capacitor of a share operation are different sizes, say C_A and C_B, the noise power on each output is given below,

$$\overline{v_{n,A}^2} = \left(\frac{C_B}{C_A+C_B}\right)^2 \cdot \frac{\mathrm{kT}}{C'} \qquad \text{and} \qquad \overline{v_{n,B}^2} = \left(\frac{C_A}{C_A+C_B}\right)^2 \cdot \frac{\mathrm{kT}}{C'}$$

where C' $(= C_A C_B/(C_A+C_B))$ is the series combination of C_A and C_B. Note that capacitive voltage dividers, as seen in Fig. 5.3(b), divide the total kT/C' noise and generate negative correlation between the two outputs. These results can be rewritten with respect to the capacitance at each operand node.

$$\overline{v_{n,A}^2} = \left(\frac{C_B}{C_A+C_B}\right) \cdot \frac{\mathrm{kT}}{C_A} \qquad \text{and} \qquad \overline{v_{n,B}^2} = \left(\frac{C_A}{C_A+C_B}\right) \cdot \frac{\mathrm{kT}}{C_B} \tag{5.5}$$

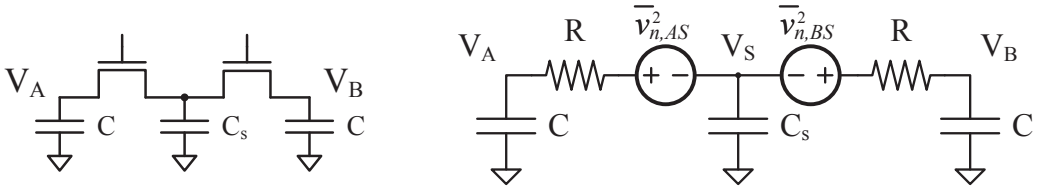
In this form it is clear that this is consistent with Eq. (5.4) when $C_A = C_B = C$.

5.2.2 Multiply Operations

The processing noise of a 1-point multiply operation, as shown in Fig. 3.2, is equivalent to that of a weighted 2-point share operation. The result in Eq. (5.5) applies, where node B refers to the stealing capacitor (substitute C_s for C_B).

5.2.3 Share and Multiply Operations

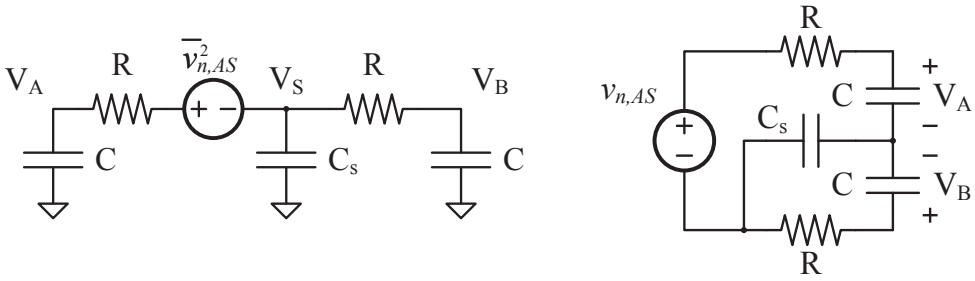
The processing noise of a share and multiply operation is more complex than that of a share operation. A 2-point implementation, as seen in Fig. 5.4(a), is analyzed. The input/output operands are held on equivalently-sized capacitors at the nodes V_A and V_B. The stealing capacitor, C_s at node V_S, provides a multiplication factor in the operation. During the computation, the switches have resistance R, and the noise is modeled using voltage sources as shown in Fig. 5.4(b).



(a) Share and multiply (b) Noise model

Figure 5.4: 2-point share and multiply implementation (a) and noise model (b)

To determine the noise at each capacitor, the noise transfer functions of each noise source to every node must be determined. Due to symmetry in the circuit, this can be accomplished by considering only one noise source, as shown in Fig. 5.5(a). Its NTFs are more easily determined when the circuit is redrawn, as shown in Fig. 5.5(b).



(a) Network for NTF analysis (b) Redrawn

Figure 5.5: 2-point share and multiply with noise source (a) and redrawn (b) for easier NTF analysis

The nework shown in Fig. 5.5(b), while complex, has a net low-pass response. The NTFs are determined by writing the expressions for the multi-stage voltage division factors to each node, V_A, V_B, and V_S, in the s-domain.

$$\frac{V_A}{v_{n,AS}} = \frac{\frac{1}{sC}}{R+\frac{1}{sC}+\left(\frac{1}{sC_s}\,||\,\left(R+\frac{1}{sC}\right)\right)} \tag{5.6}$$

$$\frac{V_B}{v_{n,AS}} = -\frac{\frac{1}{sC_s}\,||\,\left(R+\frac{1}{sC}\right)}{R+\frac{1}{sC}+\left(\frac{1}{sC_s}\,||\,\left(R+\frac{1}{sC}\right)\right)}\cdot\frac{\frac{1}{sC}}{R+\frac{1}{sC}} \tag{5.7}$$

$$\frac{V_S}{v_{n,AS}} = -\frac{\frac{1}{sC_s}\,||\,\left(R+\frac{1}{sC}\right)}{R+\frac{1}{sC}+\left(\frac{1}{sC_s}\,||\,\left(R+\frac{1}{sC}\right)\right)} \tag{5.8}$$

The $||$ operator indicates a parallel electrical combination ($Z_1\,||\,Z_2 = Z_1Z_2/(Z_1+Z_2)$). Simplification of Eqs. (5.6)–(5.8) is accomplished using the following expressions:

$$\left(R\,||\,\tfrac{1}{sC}\right) = \frac{R}{1+sRC}$$

$$\left(\tfrac{1}{sC_s}\,||\,\left(R+\tfrac{1}{sC}\right)\right) = \left(\frac{1}{1+n_s}\right)\frac{1}{sC}\,\frac{1+sRC}{1+sRC\left(\frac{n_s}{1+n_s}\right)}$$

where n_s is the stealing to operand capacitor ratio C_s/C. This results in the s-domain transfer functions shown below in a familiar pole-zero format.

$$\frac{V_A}{v_{n,AS}}(s) = \left(\frac{1+n_s}{2+n_s}\right)\frac{1+sRC\left(\frac{n_s}{1+n_s}\right)}{1+sRC\left(\frac{n_s}{2+n_s}\right)}\cdot\frac{1}{1+sRC} \tag{5.9}$$

$$\frac{V_B}{v_{n,AS}}(s) = -\left(\frac{1}{2+n_s}\right)\frac{1}{1+sRC\left(\frac{n_s}{2+n_s}\right)}\cdot\frac{1}{1+sRC} \tag{5.10}$$

$$\frac{V_S}{v_{n,AS}}(s) = -\left(\frac{1}{2+n_s}\right)\frac{1}{1+sRC\left(\frac{n_s}{2+n_s}\right)} \tag{5.11}$$

Next, the capacitance ratio parameters (n_s) in Eqs. (5.9)–(5.11) are transformed into multiplication factors ($m=2C/(2C+C_s)$) using the following four relationships:

$$\left(\frac{n_s}{1+n_s}\right) = \left(\frac{1-m}{1-m/2}\right) \qquad \left(\frac{n_s}{2+n_s}\right) = (1-m)$$

$$\left(\frac{2+n_s}{1+n_s}\right) = \left(\frac{1}{1-m/2}\right) \qquad \left(\frac{1}{2+n_s}\right) = \frac{m}{2}$$

Finally, the three NTFs are presented below in a most-useable form.

$$\frac{V_A}{v_{n,AS}}(s) = \left(1 - \frac{m}{2}\right) \cdot \frac{1 + sRC\left(\frac{1-m}{1-m/2}\right)}{1 + sRC(1-m)} \cdot \frac{1}{1 + sRC} \tag{5.12}$$

$$\frac{V_B}{v_{n,AS}}(s) = -\frac{m}{2} \cdot \frac{1}{1 + sRC(1-m)} \cdot \frac{1}{1 + sRC} \tag{5.13}$$

$$\frac{V_S}{v_{n,AS}}(s) = -\frac{m}{2} \cdot \frac{1}{1 + sRC(1-m)} \tag{5.14}$$

Due to symmetry, the NTFs of all noise sources in Fig. 5.4(b) can be determined, and have the equivalences shown in Eq. (5.15) below.

$$\frac{V_A}{v_{n,AS}} = \frac{V_B}{v_{n,BS}} \qquad \frac{V_B}{v_{n,AS}} = \frac{V_A}{v_{n,BS}} \qquad \frac{V_S}{v_{n,AS}} = \frac{V_S}{v_{n,BS}} \tag{5.15}$$

To find the noise power contributed to each node, these NTFs are integrated.

$$\overline{v^2_{n,AS\to A}} = 2\int_0^\infty \left|\frac{V_{n,A}}{v_{n,AS}}(j\omega)\right|^2 \mathrm{d}\omega$$

Likewise, the integral of the NTF to nodes B and S are done in a similar fashion. These results, combined with the symmetry relationships in Eq. (5.15), allow all of the noise power quantities to be expressed below.

$$\overline{v^2_{n,AS\to A}} = \overline{v^2_{n,BS\to B}} = \left(\frac{1 - m + m^2/8}{1 - m/2}\right) \cdot \frac{\mathrm{kT}}{C} \tag{5.16}$$

$$\overline{v^2_{n,AS\to B}} = \overline{v^2_{n,BS\to A}} = \frac{1}{8}\left(\frac{m^2}{1 - m/2}\right) \cdot \frac{\mathrm{kT}}{C} \tag{5.17}$$

$$\overline{v^2_{n,AS\to S}} = \overline{v^2_{n,BS\to S}} = \frac{1}{4}\left(\frac{m^2}{1 - m}\right) \cdot \frac{\mathrm{kT}}{C} = \frac{m}{2} \cdot \frac{\mathrm{kT}}{C_s} \tag{5.18}$$

Furthermore, the total noise at each output node is a sum of all contributing sources. A 2-point share and multiply with two switches has two independent noise sources, which gives the following relationships:

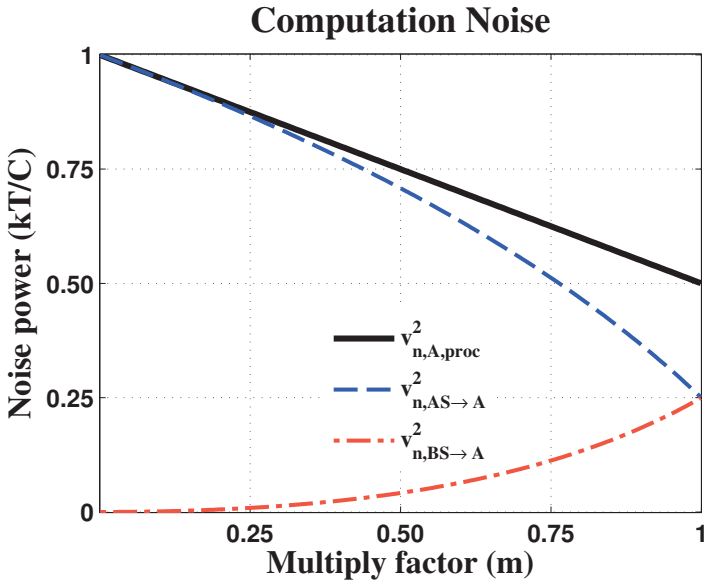


Figure 5.6: 2-point share and multiply operand capacitor processing noise

$$\overline{v_{n,A}^2} = \overline{v_{n,AS\to A}^2} + \overline{v_{n,BS\to A}^2}$$
$$\overline{v_{n,B}^2} = \overline{v_{n,AS\to B}^2} + \overline{v_{n,BS\to B}^2}$$
$$\overline{v_{n,S}^2} = \overline{v_{n,AS\to S}^2} + \overline{v_{n,BS\to S}^2}$$

In summary, for the operand outputs, capacitors C on nodes A and B (see Fig. 5.4(a)), the total processing noise is shown in Eq. (5.19) and plotted in Fig. 5.6.

$$\overline{v_{out,\mathrm{proc},C}^2} = \overline{v_{n,A}^2} = \overline{v_{n,B}^2} = \left(1-\frac{m}{2}\right)\cdot\frac{\mathrm{kT}}{C} \tag{5.19}$$

This matches the result for a share operation in Eq. (5.4) when $m=1$ ($C_s \to 0$). Also, this is consistent with the expected result in the extreme case of $m=0$ ($C_s \to$ inf). In this case, the large steaing capacitor provides an effective short between the circuit halves, causing each operand capacitor to receive kT/C noise independently. The processing noise onto the stealing capacitor (C_s) is determined below for completeness.

$$\overline{v^2_{out,\mathrm{proc},C_s}} = \overline{v^2_{n,S}} = \frac{1}{2}\left(\frac{m^2}{1-m}\right)\cdot\frac{\mathrm{kT}}{C} = m\cdot\frac{\mathrm{kT}}{C_s} \tag{5.20}$$

5.3 Input and Reset Noise

Assumptions are often made in charge-domain operations. One of these is that the initial voltage of all capacitors in the network are the ideal desired value. However, any operation to set their voltage or previous computation will introduce a sampled noise term. Some of these noise disturbances are overwritten or modified, depending on the type of sampling method used[1]. The remaining noise components will influence the computational result and is called *input* or *reset* noise. In particular, *input* noise occurs is realted to the operand capacitors and *reset* noise refers to the effect of noise on stealing capacitors. These noise contributions are discussed in the section below.

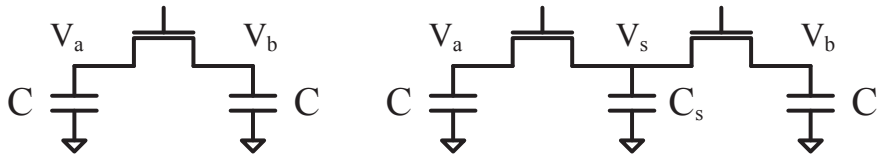
5.3.1 Share Operations

Share operations, like Fig. 5.7(a), suffer from *input* noise on their operand capacitors. Input operands will always be corrupted by noise, whether it is from a sampling operation or a previous computation. The charge-domain description of this operation can be used to determine the output noise contributed by the initial value noise of the inputs.

$$\begin{aligned} V_{out} &= \tfrac{1}{2}(V_A+V_B) \\ \Rightarrow\quad \overline{v^2_{n,out,\mathrm{init}}} &= \left(\tfrac{1}{2}\right)^2\left(\overline{v^2_{n,A,\mathrm{init}}}+\overline{v^2_{n,B,\mathrm{init}}}\right) \end{aligned} \tag{5.21}$$

Whether the inputs come from a voltage-sampling operation (noise kT/C) or from a previous share operation (processing noise $\frac{1}{2}\,\mathrm{kT}/C$ and input noise $\frac{1}{2}\,\mathrm{kT}/C$), the result is $\overline{v^2_{n,out,\mathrm{init}}} = \frac{1}{2}\,\mathrm{kT}/C$. If the two inputs have correlation this sum may be reduced.

[1] Voltage-sampling will overwrite the initial value noise but will contribute kT/C upon its completion, while charge domain (Gm) sampling will integrate on top of it



(a) 2-point share (b) 2-point share and multiply

Figure 5.7: 2-point operations used for input and reset noise analysis

5.3.2 Share and Multiply Operations

Unlike a share operation, share and multiply operations also have a capacitor that is not part of the sampling system. This stealing capacitor, C_s, is nominally reset before the charge-domain computation occurrs. Resetting the stealing capacitor is accomplished by turning on the processing switches while the operand capacitors are also being reset by the sampler or by adding a reset switch to node V_S for switching power consumption if a slower reset operation is allowable (not requiring the low switch resistance of the processing switches). When the switches open at the end of the reset operation, noise is sampled onto C_s.

For a 2-point share and multiply operation with equal-sized operand capacitors (Fig. 5.7(b)) the charge-domain expression is shown below for a multiplication factor of $m = 2C/(2C+C_s)$. From this, the computation output noise due to the initial value noise of the operand and stealing capacitors can be found, as shown in Eq. (5.22).

$$\begin{aligned} V_{out} &= m \cdot \frac{V_A+V_B}{2} + (1-m) \cdot V_S \\ \Rightarrow \quad \overline{v^2_{n,out,\text{init}}} &= \underbrace{\left(\frac{m}{2}\right)^2 \left(\overline{v^2_{n,A,\text{init}}} + \overline{v^2_{n,B,\text{init}}}\right)}_{\textit{input}\text{ noise}} + \underbrace{(1-m)^2\, \overline{v^2_{n,S,\text{init}}}}_{\textit{reset}\text{ noise}} \end{aligned} \tag{5.22}$$

The noise power sampled onto the stealing capacitor during the reset operation is $\overline{v^2_{n,S,\text{init}}} = \text{kT}/C_s$. Since the reset-value error is uncorrelated with the initial-value noise on the operand capacitors it can be considered separately, as shown below.

$$\begin{aligned}\overline{v_{n,out,\text{reset}}^2} &= (1-m)^2\,\overline{v_{n,S,\text{init}}^2} \\ &= (1-m)^2 \cdot \frac{\text{kT}}{C_s} = \frac{m(1-m)}{2} \cdot \frac{\text{kT}}{C}\end{aligned} \tag{5.23}$$

Note that the result is also written normalized to kT/C for later convenience.

The noise in the initial value of the operands propagates to the output by the following relationship:

$$\begin{aligned}\overline{v_{n,out,\text{input}}^2} &= \left(\frac{m}{2}\right)^2 \left(\overline{v_{n,A,\text{init}}^2} + \overline{v_{n,B,\text{init}}^2}\right) \\ &= \frac{m^2}{2} \cdot \frac{\text{kT}}{C}\end{aligned} \tag{5.24}$$

Just like for a share operation, any previous sampling or computation operation will have noise kT/C. This generates the total *input* noise shown in Eq. (5.24). If the two input operands have correlation this sum may be reduced.

5.4 Noise Correlation

As dictated by the NTFs, each independent noise source contributes to each output node with different magnitude and sign. Therefore, the operands and stealing capacitor have some correlation, ρ, between them. Between the two operand capacitor outputs (nodes A and B in Fig. 5.4(a)), the processing noise of a share and multiply operation has the correlation coefficient value shown in Eq. (5.26) and is plotted in Fig. 5.8. The details of its calculation follow.

The total noise voltage sampled on each output can be considered to be a linear combination of the instantaneous random voltage of each noise source. This is represented by the equation below; the left-hand side corresponds to the sampled noise voltage on the outputs and the right-hand side is a linear combination of $v_{n,AS}$ and $v_{n,BS}$, denoting

the noise sources.

$$\begin{bmatrix} v_{n,A} \\ v_{n,B} \\ v_{n,S} \end{bmatrix} = \begin{bmatrix} H_{n,AS\to A} & H_{n,BS\to A} \\ H_{n,AS\to B} & H_{n,BS\to B} \\ H_{n,AS\to S} & H_{n,BS\to S} \end{bmatrix} \begin{bmatrix} v_{n,AS} \\ v_{n,BS} \end{bmatrix} \tag{5.25}$$

The H_n entries are the integrated noise voltage transfer functions and are similar to the total integrated noise powers derived from the system NTFs. They are in the voltage domain rather than a noise power and preserve the polarity of the NTF as shown below.

$$H_{n,AS\to A} = \text{sign}\left(\frac{V_A}{v_{n,AS}}(0)\right) \cdot \sqrt{\overline{v^2_{n,AS\to A}}}$$

Note that $H^2_{n,AS\to A} = \overline{v^2_{n,AS\to A}}$. Due to the symmetry equivalences in Eq. (5.15), all entries in Eq. (5.25) can determined using Eqs. (5.16)–(5.18).

The correlation coefficient of output operands in a share and multiply operation is

$$\begin{aligned} \rho_{A,B,\text{proc}} &= \frac{\text{Cov}(v_{n,A}, v_{n,B})}{\sqrt{\text{Var}(v_{n,A})}\sqrt{\text{Var}(v_{n,B})}} = \frac{2\,H_{n,AS\to A}\cdot H_{n,BS\to A}}{\sqrt{\overline{v^2_{n,A}}}\sqrt{\overline{v^2_{n,B}}}} \\ &= -\frac{m\sqrt{m^2+8(1-m)}}{(2-m)^2} \end{aligned} \tag{5.26}$$

Its value is always less than zero and reaches -1 at $m = 1$, which is consistent with the differential correlation of a share operation. This means that there is always some differential noise correlation that can be used to reduce processing noise. The stealing capacitor output also correlates with the operand outputs and is calculated below.

$$\begin{aligned} \rho_{A,S,\text{proc}} &= \frac{\text{Cov}(v_{n,A}, v_{n,S})}{\sqrt{\text{Var}(v_{n,A})}\sqrt{\text{Var}(v_{n,S})}} = \frac{\left(H_{n,AS\to A} + H_{n,BS\to A}\right) H_{n,AS\to S}}{\sqrt{\overline{v^2_{n,A}}}\sqrt{\overline{v^2_{n,S}}}} \\ &= -\frac{m}{4}\,\frac{\sqrt{m^2+8(1-m)} - m}{\sqrt{1-m}\,\sqrt{2-m}} \end{aligned} \tag{5.27}$$

When reset noise (see Eq. (5.23) in Sec. 5.3) is included, the correlation becomes

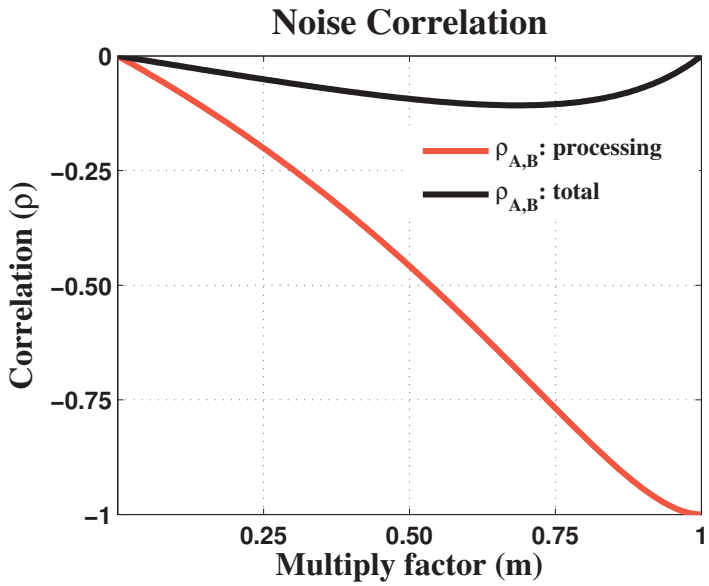


Figure 5.8: 2-point share and multiply operand correlation coefficient

$$\rho_{A,B,\text{reset}} = -\frac{2\,H_{n,AS\to A}\cdot H_{n,BS\to A}+\overline{v^2_{n,\text{reset}}}}{\sqrt{\overline{v^2_{n,A}}+\overline{v^2_{n,\text{reset}}}}\sqrt{\overline{v^2_{n,B}}+\overline{v^2_{n,\text{reset}}}}} \tag{5.28}$$

$$= -m\frac{\sqrt{m^2+8(1-m)}-m^2+3m-2}{(2-m)(2-m^2)} \tag{5.29}$$

Finally, when the input noise (Eq. (5.24)) is also considered, the correlation in the total noise (Eq. (5.32)) between operand outputs is calculated below.

$$\begin{aligned}\rho_{A,B,\text{total}} &= -\frac{2\,H_{n,AS\to A}\cdot H_{n,BS\to A}+\overline{v^2_{n,\text{input}}}+\overline{v^2_{n,\text{reset}}}}{\overline{v^2_{n,out,C}}} \\ &= -\frac{m}{2}\left(1+\frac{\sqrt{m^2+8(1-m)}}{2-m}\right)\end{aligned} \tag{5.30}$$

The correlation of the processing and total noise in a 2-point share and multiply operation are plotted in Fig. 5.8. Since input and reset noise generate a correlated noise on the output operands, they reduce the correlation in the total noise output.

5.5 Total Noise

As charge-domain computations are linear, superposition can be used to find the total noise on the output operands due to input, reset, and processing noise. Since they occurr at different times, these noise events are uncorrelated and add as shown below.

$$\overline{v_{out}^2} = \overline{v_{out,\text{input}}^2} + \overline{v_{out,\text{reset}}^2} + \overline{v_{out,\text{proc}}^2} \tag{5.31}$$

As described previously, only share and multiply operations suffer from reset noise. Therefore, the total noise of a share operation is simply the operand input and processing noise, from Eqs. (5.21) and (5.4). For equal operand capacitor sizes, this totals

$$\begin{aligned}\overline{v_{out}^2} &= \overline{v_{out,\text{input}}^2} + \overline{v_{out,\text{proc,C}}^2} \\ &= \frac{1}{2}\cdot\frac{\text{kT}}{C} + \frac{1}{2}\cdot\frac{\text{kT}}{C} = \frac{\text{kT}}{C}\end{aligned}$$

Share and mulitply operations suffer from all three forms of noise. When the outputs are taken from equally-sized operand capacitors, the total noise is a sum of the noise powers in Eqs. (5.19), (5.23), and (5.24) and is shown below.

$$\begin{aligned}\overline{v_{out,C}^2} &= \overline{v_{out,\text{input}}^2} + \overline{v_{out,\text{proc,C}}^2} + \overline{v_{out,\text{reset}}^2} \\ &= \frac{m^2}{2}\cdot\frac{\text{kT}}{C} + \left(1-\frac{m}{2}\right)\cdot\frac{\text{kT}}{C} + \frac{m(1-m)}{2}\cdot\frac{\text{kT}}{C} \\ &= \frac{\text{kT}}{C}\end{aligned} \tag{5.32}$$

If the stealing capacitor is not reset before processing, introducing memory of the prevous operation, the computation still has input and processing noise. Like the operands of previous computations, the stealing capacitor result from the previous operation will have its own processing noise (Eq. (5.20)). However, this is considered a reset noise term since it is independent and not generated by the subsequent computation.

When the result on a stealing capacitor (C_s) is used as an additional output, it is

useful to look at its noise. Just like the operand capacitors, if a reset is performed, there will be a reset noise component in addition to the input and processing noise.

$$\begin{aligned}\overline{v_{out,C_s}^2} &= \overline{v_{out,\text{input}}^2} + \overline{v_{out,\text{proc,C}_\text{s}}^2} + \overline{v_{out,\text{reset}}^2} \\ &= \frac{m^2}{2}\cdot\frac{\text{kT}}{C} + m\cdot\frac{\text{kT}}{C_s} + (1-m)^2\cdot\frac{\text{kT}}{C_s} \\ &= \frac{\text{kT}}{C_s}\end{aligned} \tag{5.33}$$

A summary of noise results discussed above are shown in Table 5.1. The output node column denotes which node, operand C or stealing capacitor C_s, the noise is calculated for. For comparison with the operand capacitor results, the C_s-output share and multiply operation result is shown referenced to kT/C in addition to kT/C_s. When input, reset, and processing noise are considered the total noise becomes kT/C on and operand capacitors and kT/C_s on stealing capacitors.

Table 5.1: Summary of total noise in charge-domain operations

Operation	**O/p**	**Noise power**					
type	**node**	processing		proc+reset		proc+res+in	
Share (Sec. 5.2.1)	C	$\frac{1}{2}$	$\times\frac{\text{kT}}{C}$	$\frac{1}{2}$	$\times\frac{\text{kT}}{C}$	1	$\times\frac{\text{kT}}{C}$
Multiply (Sec. 5.2.2)	C	$1-m$	$\times\frac{\text{kT}}{C}$	$1-m^2$	$\times\frac{\text{kT}}{C}$	1	$\times\frac{\text{kT}}{C}$
Multiply	C_s	m	$\times\frac{\text{kT}}{C_s}$	$1-m^2$	$\times\frac{\text{kT}}{C_s}$	1	$\times\frac{\text{kT}}{C_s}$
		$\frac{m^2}{1-m}$	$\times\frac{\text{kT}}{C}$	$m(1+m)$	$\times\frac{\text{kT}}{C}$	$\frac{1-m}{m}$	$\times\frac{\text{kT}}{C}$
Share/Mult. (Sec. 5.2.3)	C	$1-m/2$	$\times\frac{\text{kT}}{C}$	$1-m^2/2$	$\times\frac{\text{kT}}{C}$	1	$\times\frac{\text{kT}}{C}$
Share and Multiply	C_s	m	$\times\frac{\text{kT}}{C_s}$	$1-m+m^2$	$\times\frac{\text{kT}}{C_s}$	1	$\times\frac{\text{kT}}{C_s}$
		$\frac{m^2}{2(1-m)}$	$\times\frac{\text{kT}}{C}$	$\frac{m(1-m+m^2)}{2(1-m)}$	$\times\frac{\text{kT}}{C}$	$\frac{2(1-m)}{m}$	$\times\frac{\text{kT}}{C}$

The total charge-domain computation noise that manifests on the operand capacitors is plotted in Fig. 5.9. The noise components of Eqs. (5.19), (5.23), and (5.24) are plotted as dashed lines. Similarly, the total noise on the stealing capacitor is kT/C_s (Eq. (5.33)).

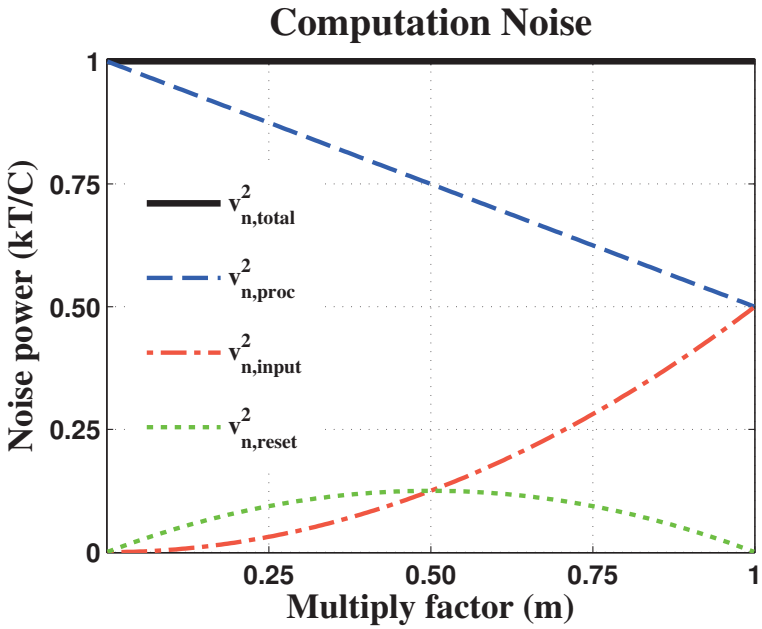


Figure 5.9: 2-point share and multiply operand capacitor noise

The computation noise that appears on the stealing capacitor is plotted in Fig. 5.10. The individual noise components of Eqs. (5.18), and (5.23) are plotted as dashed lines. Their combination forms the processing and processing-plus-reset noise denoted by: $v_{n,S}^2 = \overline{v_{out,\mathrm{proc},C_s}^2}$, and $v_{n,S+\mathrm{reset}}^2 = \overline{v_{out,\mathrm{proc},C_s}^2} + \overline{v_{out,\mathrm{reset}}^2}$ (Eqs. (5.20) and (5.33)).

Operand and stealing capacitor noise are plotted on the same axis in Fig. 5.11, excluding input noise, for comparison. When input noise is included, the results are kT/C and $\mathrm{kT/C_s} = \frac{m}{2(1-m)} \cdot \mathrm{kT}/C$, respectively. As expected, the stealing capacitor output has less noise than each operand capacitor when $m < 2/3$ since $C_s > C$. The operand capacitors can be combined in a future stage to provide noise averaging and their differential correlation provides further noise reduction (see Fig. 5.12); this is not possible from the single stealing capacitor output. Therefore, the stealing capacitor output only has superior noise performance compared to averaged operand outputs when $m \lesssim 0.5$ ($C_s = 2C$).

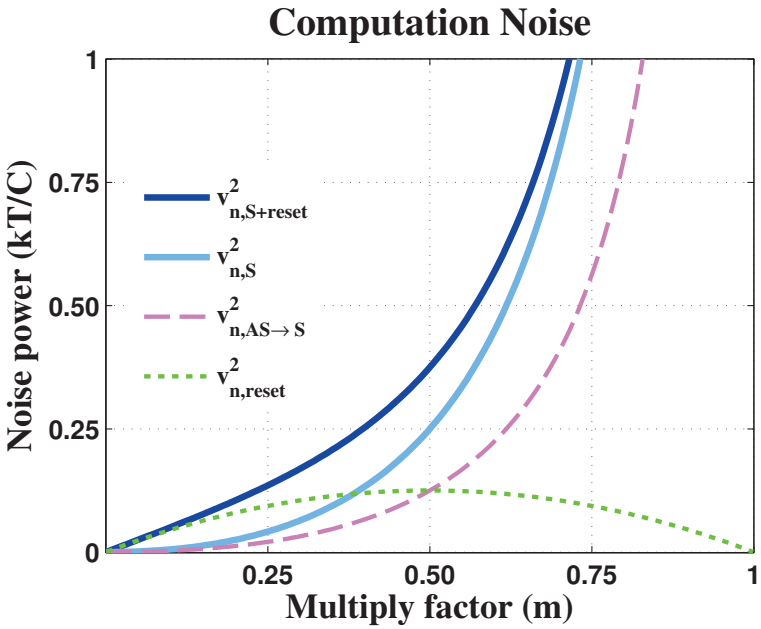


Figure 5.10: 2-point share and multiply stealing capacitor noise

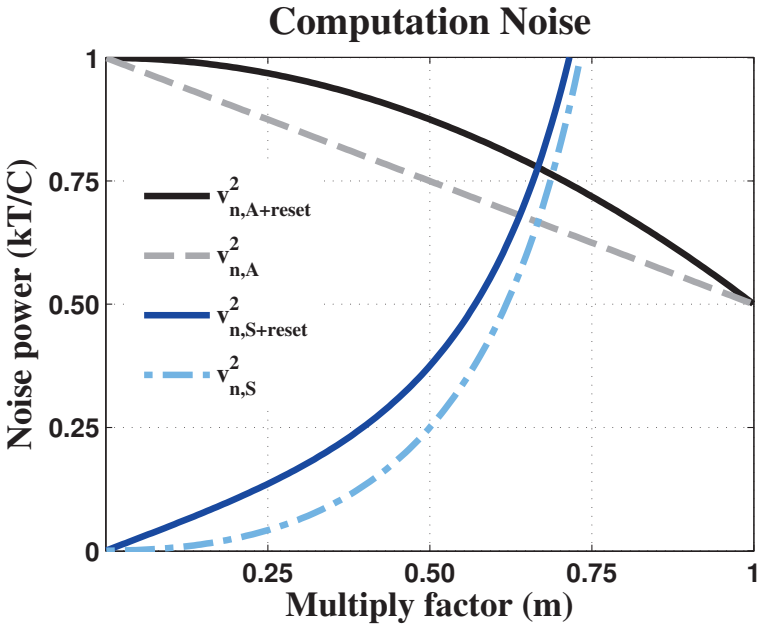


Figure 5.11: 2-pt share and multiply operand and stealing capacitor noise

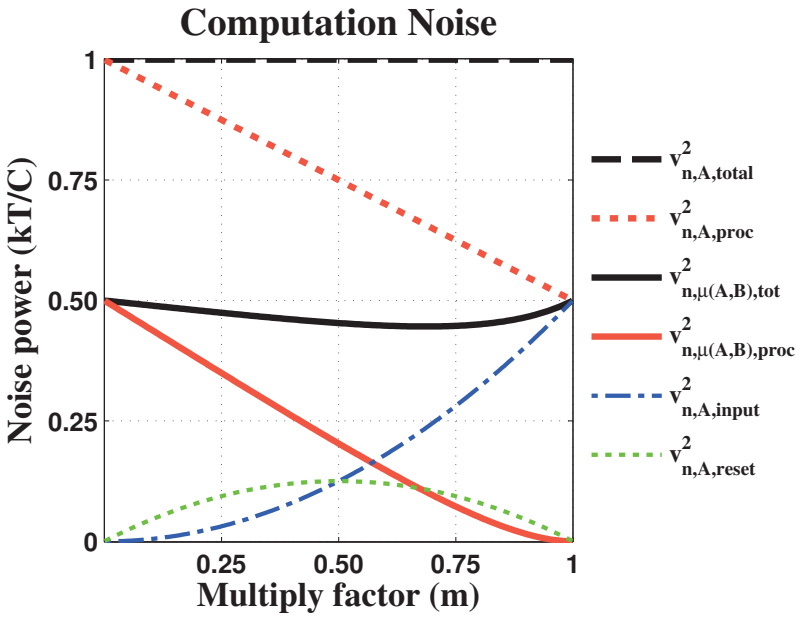


Figure 5.12: 2-point share and multiply operand noise averaging with correlation

Due to correlation (Eq. (5.26)), averaging two outputs' processing noise gives

$$\overline{v^2_{n,\mu(A,B),\text{proc}}} = \frac{1}{4}\left(1+\rho_{A,B,\text{proc}}\right)\left(\overline{v^2_{n,A}} + \overline{v^2_{n,B}}\right) \tag{5.34}$$

Reset and input noise does not benefit from averaging because they are perfectly correlated among the operand outputs due to the charge-sharing operation. With this in mind, averaging two output operand copies with input and reset noise gives the results shown in Eqs. (5.35) and (5.36) with correlation coefficients (5.29) and (5.30).

$$\begin{aligned}\overline{v^2_{n,\mu(A,B),\text{proc+res}}} &= \frac{1}{4}\left(1+\rho_{A,B,\text{proc}}\right)\left(\overline{v^2_{n,A}} + \overline{v^2_{n,B}}\right) + \overline{v^2_{n,\text{reset}}} \\ &= \left(1+\rho_{A,B,\text{reset}}\right)\left(\tfrac{1}{4}\left(\overline{v^2_{n,A}} + \overline{v^2_{n,B}}\right) + \overline{v^2_{n,\text{reset}}}\right)\end{aligned} \tag{5.35}$$

$$\begin{aligned}\overline{v^2_{n,\mu(A,B),\text{total}}} &= \tfrac{1}{4}\left(1+\rho_{A,B,\text{proc}}\right)\left(\overline{v^2_{n,A}} + \overline{v^2_{n,B}}\right) + \overline{v^2_{n,\text{input}}} + \overline{v^2_{n,\text{reset}}} \\ &= \tfrac{1}{2}\left(1+\rho_{A,B,\text{total}}\right)\overline{v^2_{n,\text{out},C}}\end{aligned} \tag{5.36}$$

When the output noise correlation calculated in Sec. 5.4 is considered, the computation noise due to averaging the two operand-capacitor outputs is shown in Fig. 5.12. The processing noise on each operand with and without input and reset noise: $v^2_{n,A,\text{total}}$ (Eq. (5.32)), and $v^2_{n,A,\text{proc}}$ (Eq. (5.19)), respectively, are plotted as dashed lines. The results from Eqs. (5.34) and (5.36) are plotted as $v^2_{n,\mu(A,B),\text{total}}$ and $v^2_{n,\mu(A,B),\text{proc}}$ for the operand-averaged result both with and without reset and input noise, respectively. The total operand-averaged noise value dips below 0.5 due to the differential correlation of the processing noise.

5.6 Conclusions

Analysis of the different noise sources was performed in this chapter. It was shown how the computation noise could be separated into different contributing components. The total noise matched the expected value of kT/C for each capacitor of size C. Also, the noise from a cascade of computations never exceeeded that of a single computation. Statistical models of the noise in a system can also be useful for purposes such as the LTI model shown in Appendix B. Modeling methods, such as the one used for CRAFT (Sec. 6.4.1), can properly simulate system-level noise with proper correlation.

Chapter 6

CRAFT: Charge Re-use Analog Fourier Transform

6.1 Introduction

In this chapter, we discuss the design of a wideband digitizer introduced in Chapter 2. The digitizer is based on a frequency domain divide and conquer approach that could be followed by multiple ADCs that digitize the input in the frequency domain. For this, we propose to utilize an RF sampler followed by a discrete-time Fourier transform engine to perform channelization of the wideband RF input. The use of RF samplers and subsequent discrete-time processing, prior to digitization, provide a number of advantages in deep sub-micron CMOS processes including high linearity, programmability, large bandwidth, robustness to jammers, immunity to clock jitter, low power ADCs, etc. [21–23]. Recently, discrete time radio receivers using RF sampling have been demonstrated using CMOS technology for Bluetooth [24], GSM/GPRS [25], WLAN [26, 27], and SDR-type applications [28, 29].

We use a discrete time DFT as a functionally equivalent linear phase N-path filter

(see Fig. 6.1) to perform channelization [30]. This scheme reduces both the speed and dynamic range of the ADCs, and, by virtue of being minimal phase, allows for simple reconstruction in the digital domain using an IFFT without any loss of information. A few current based analog DFT filters have been designed recently [31, 32]. However, these designs are speed-limited, and consume significant power (Table 6.4) minimizing the overall gains. Additionally, these use active devices for signal processing, and are therefore expected to be more non-linear at high operating speeds. In this work, we perform a charge domain DFT to significantly reduce the power overhead, making a DFT-based wideband digitizing front-end feasible.

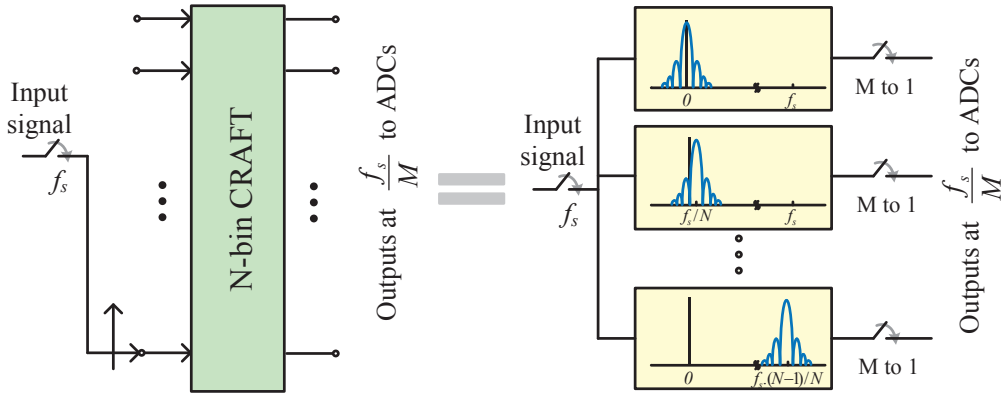


Figure 6.1: The functional equivalence of the DFT with frequency-domain outputs and a time-domain N-path complex-bandpass filter bank

This paper describes the design of such a DFT filter, CRAFT (Charge Re-use Analog Fourier Transform), based on passive switched capacitors. It performs an analog domain 16 point DFT running at input rates as high as 5GS/s and uses only 3.8mW, or 12.2pJ of 16 point DFT conversion. The design was first presented in [33, 34] and has been expanded upon in [35]; this chapter includes further the details of the implementation, modeling and mitigation of non-idealities, design methodology and optimization, and

additional measurement results.

As shown in Fig. 6.2, an architecture with a CRAFT RF front-end reduces the ADCs' speeds (by N), at a negligible power overhead. The ADC dynamic range is also reduced due to the removal of out-of-band signals per ADC. The impact of the CRAFT front-end on the ADC input bandwidth and dynamic range for well-spread signals is shown in Fig. 6.3 where the expected ADC requirements for a 5GS/s input are plotted amongst measured ADC implementations [1]. As seen in Fig. 6.3, the CRAFT front-end reduces the required speed of each ADC as well as their input dynamic range through channelization. This brings the ADC requirements from being infeasible (top right corner) toward being achievable (bottom left half) thereby solving the critical wide-band digitizing problem in SDRs. In contrast, a time interleaved ADC approach on reduces the speed requirements of each ADC without reducing the dynamic range as shown by the vertical arrow in Fig. 6.3; consequently the total ADC power remains approximately unchanged.

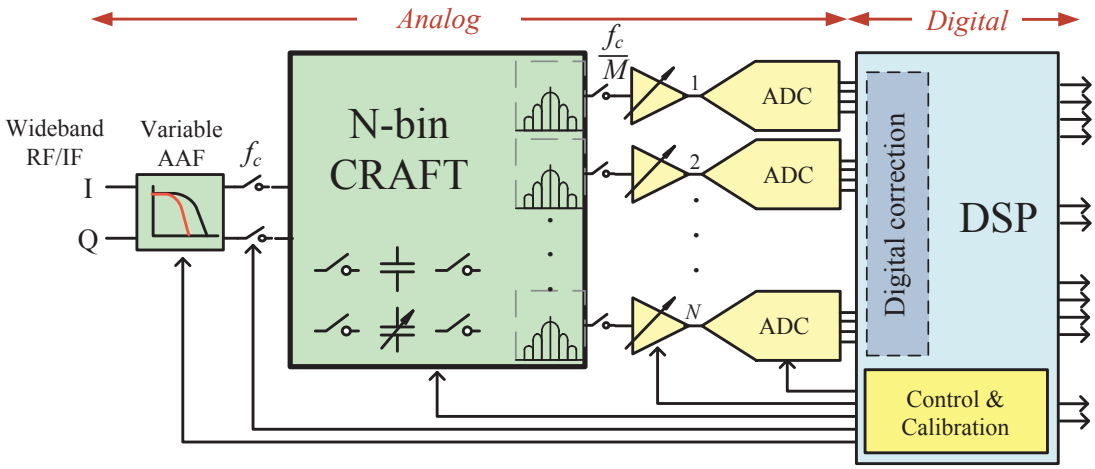
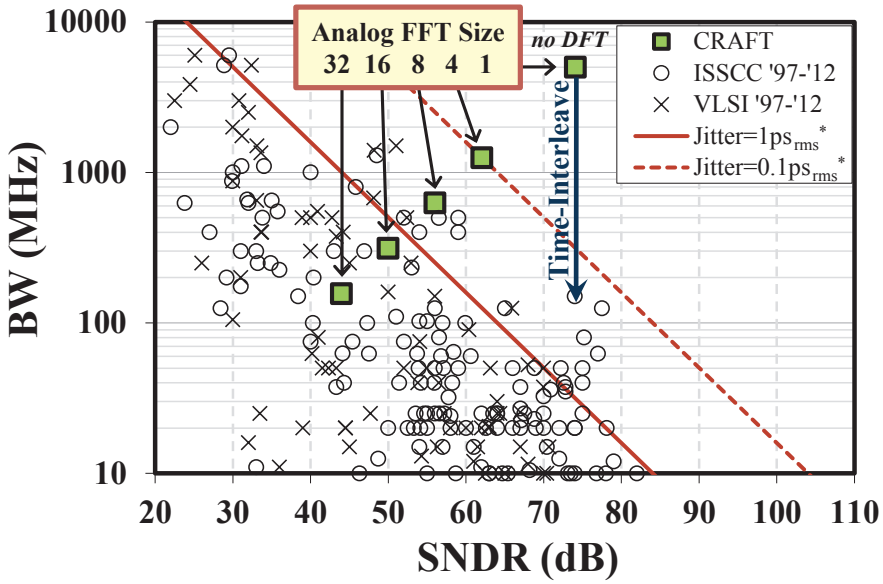


Figure 6.2: An envisioned wideband sensing architecture for SDR using CRAFT

For dynamic range calculations, signals are assumed to be distributed in frequency.



* Assumes white noise (see [23] for details on assumption)

Figure 6.3: ADC requirements feasibility vs. bin size of CRAFT front-end

Breaking the input up into equal frequency channels reduces the PAPR. In general, an N path channelization of spread signals reduces the dynamic range by N times[1], causing an N times dynamic range reduction for the ADCs.

As seen, the CRAFT front-end brings the required ADC specifications from being infeasible (top right corner) toward being achievable (bottom left half). By solving this critical broad-band digitizing problem, CRAFT is expected to enable the realization of wideband SDRs in the future.

The chapter is organized as follows. First, the conceptual charge-based passive design utilized in CRAFT is described. The CRAFT implementation is then discussed in detail, including a brief description of the on-chip circuitry that interfaces the design for performance measurement. The circuit performance is ultimately limited by several

[1]Even in the case the signals are congested, but not directly on-bin, the DFT provides sufficient signal spreading for this approximation to roughly hold.

non-idealities. Modeling of these non-idealities, and design techniques and circuit optimization to mitigate them are discussed. Measurement results from a prototype 16 point 5GS/s design are then presented, followed by conclusions.

6.2 CRAFT Design Concept

CRAFT operations are based on charge re-use. Once sampled, the charge on a capacitor is shared and re-shared with other charge samples such that the resulting mathematical manipulation is an in-place DFT. Since the design operates using only toggling switches (transistors), low power and high speeds are made feasible. Additionally, due to the dynamic nature of the switch power consumption, the power scales with frequency, supply, and technology in a digital-like fashion.

The computation to be performed is a time-to-frequency transform defined as follows:

$$\textbf{DFT:} \quad X(k) = \sum_{n=0}^{N-1} x(n) W^{kn} \tag{6.1}$$

where W is defined as $W = e^{-\frac{2\pi j}{16}}$. The desired 16 point DFT ($N = 16$) can also be represented as a linear matrix operation on a vector of length 16 as shown below:

$$\vec{\mathbf{X}} = \frac{1}{k}\,\mathbf{F}\,\vec{\mathbf{x}} \tag{6.2}$$

where k is a scaling factor, relative to the mathematical linear operation, due to attenuation inherent in the charge-domain operations. When $\mathbf{F}$ is a 16×16 matrix all DFT outputs are simultaneously calculated as expanded below:

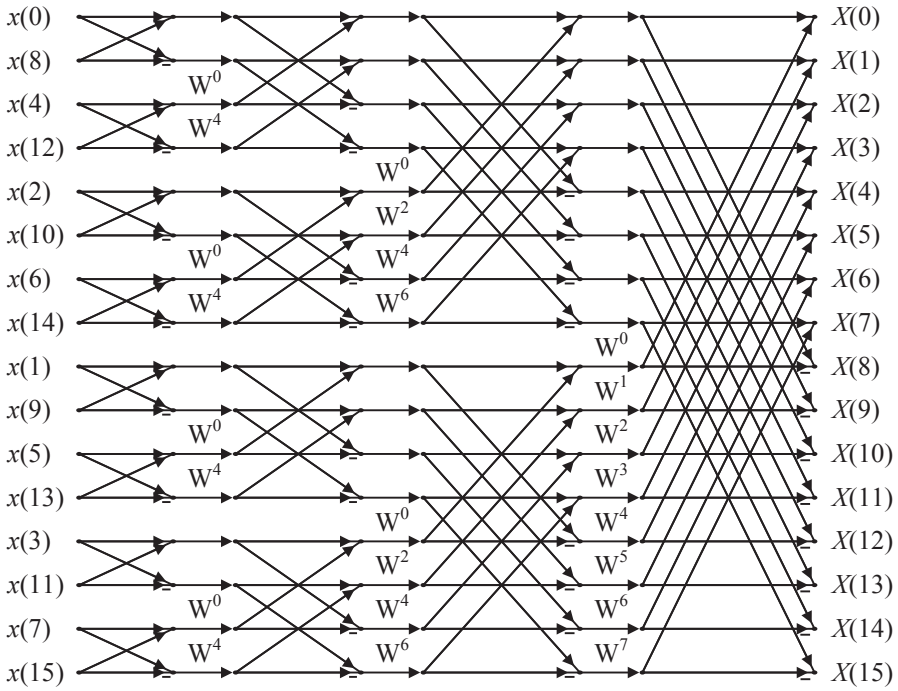
$$\Rightarrow \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{15} \end{bmatrix} = \frac{1}{k} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W & W^2 & \cdots & W^{15} \\ 1 & W^2 & W^4 & \cdots & W^{30} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W^{15} & W^{30} & \cdots & W^{225} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{15} \end{bmatrix} \tag{6.3}$$

where x_i are the inputs to the DFT engine $x(i)$, and X_k are outputs $X(k)$. The equation is further simplified by noting the symmetry and periodicity of the powers of W. These properties are utilized to formulate the FFT algorithm as an efficient implementation to calculate the DFT outputs in $N \log_2 N$ operational complexity rather than the N^2 complexity of multiplication by $\mathbf{F}$.

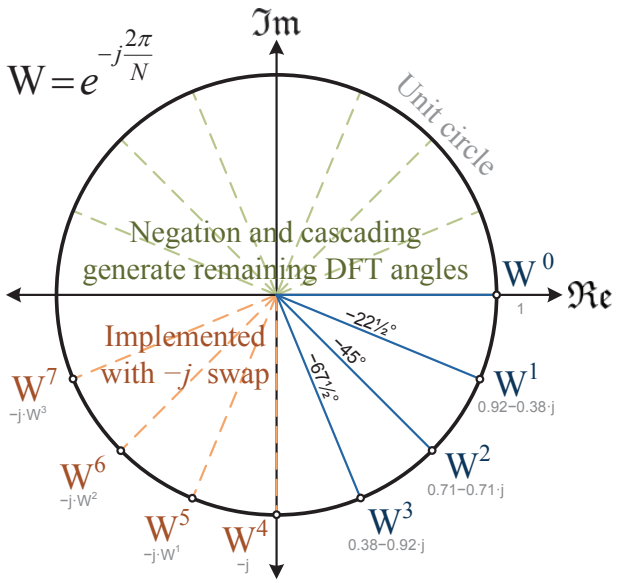
Fig. 6.4(a) shows the flowchart representation of the radix-2 decimation-in-time FFT algorithm used in CRAFT. As seen in Fig. 6.4(a), and by its definition as a linear operation, the FFT uses only two types of operations: addition (and subtraction), and multiplication by twiddle factors. These twiddle factors are shown as powers of W in the figure, where $W = e^{-\frac{2\pi j}{16}}$, that are equally spaced points on an unit circle in the complex plane as shown in Fig. 6.4(b). As a result, for every scaling factors, W^k, $\Re\{W^k\} \leq 1, \forall k$ and $\Im\{W^k\} \leq 1, \forall k$. Since passive computations inherently attenuate the signal, these operations are particularly suited for sub-unity scaling.

For performing the CRAFT operations, the following charge domain computations are used. Addition is performed by sharing the charges on two capacitors as shown in Fig. 6.5(a). Multiplication is performed by stealing charge away from a capacitor using an appropriately-sized stealing capacitor (C_s) as shown in Fig. 6.5(b). These two simple operations form the basis of all operations in CRAFT. The full set of charge-domain mathematical operations performed are shown in Fig. 6.6.

These operations are utilized in the CRAFT processor as shown in Fig. 6.6. Row 1 shows a single-ended, 2-point share operation used for addition and subtraction. Sub-unity multiplication, if required for scaling or twiddle-factors, is based on charge stealing as shown in row 2 of Fig. 6.6 as a share and multiply operation (single-ended, 2-point implementation). Negation is performed by swapping the positive and negative operand wires as shown in row 3, while multiplication by j ($=\sqrt{-1}$) is performed by swapping the appropriate wires of the real and imaginary components as shown in row 4 of Fig. 6.6.

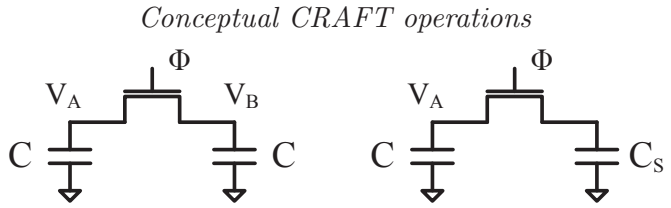


(a) 16 point radix-2 decimation-in-time FFT



(b) DFT twiddle factors

Figure 6.4: A 16 point radix-2 decimation-in-time FFT (a), and DFT twiddle factors coefficients on the unit circle (b)



(a) Add (charge-share) (b) Scale (charge-steal)

Figure 6.5: Fundamental implementation of operations used by CRAFT: 2-point addition (a), and 1-point sub-unity scaling (b)

These techniques are combined to perform a complex multiplication, as shown in row 5. An example butterfly operation containing general complex multiplications is also shown in row 6.

Each general butterfly operation requires two clock phases for the two stages of the share and complex multiply operation, as denoted by different colors. However, in the CRAFT engine, we optimize the butterfly implementations such that only five, instead of eight, clock phases are utilized for the four butterfly stages of our radix-2 FFT design. This optimization is discussed further in Sec. 6.3. The optimized butterfly blocks are then used to construct the 16 point CRAFT engine shown in Fig. 6.7. As shown, the following different types of butterflies are utilized: 16 of 1-stage share (Type 1), six of 1-stage scalar multiply (Type 2A), two of 1-stage complex multiply (Type 2B), five of 1-stage scalar multiply (Type 3A), one of 1-stage complex multiply (Type 3B), and two of 2-stage complex multiply (Type 3C). The circuit schematics for these butterflies are shown in Fig. 6.8.

The in-place nature of CRAFT computations means that they are inherently destructive; therefore, multiple copies of each data value are required for multiple operations. Since a radix-2 FFT algorithm performs the DFT with a minimum number of operations per operand (less copies required), it is selected for CRAFT.

Conceptually, the FFT is computed as follows: The input signal is sampled onto

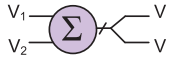
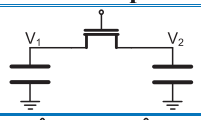
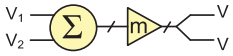
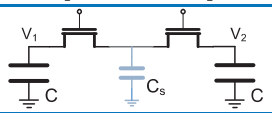
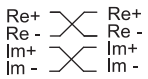
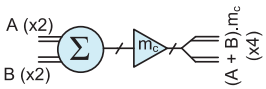
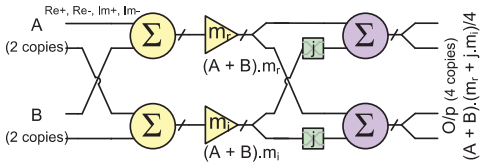
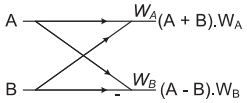
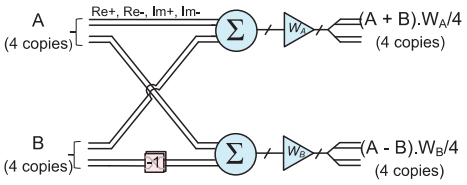
	Function name	Symbol used	Implementation
1	**Share**		 $V = \frac{V_1 + V_2}{2}$
2	**Share & multiply**		 $V = (V_1 + V_2).m$ $m = \frac{C}{2 \cdot C + C_s}$
3	**Negate**		$A \times (-1) = B$ 
4	**Multiply by '*j*'** $j = \sqrt{-1}$		$A \times j = B$ 
5	**Share & complex multiply**	 $m_c = m_r + j \cdot m_i$	
6	**Butterfly**		

Figure 6.6: Charge-domain mathematical operations used by the CRAFT processor

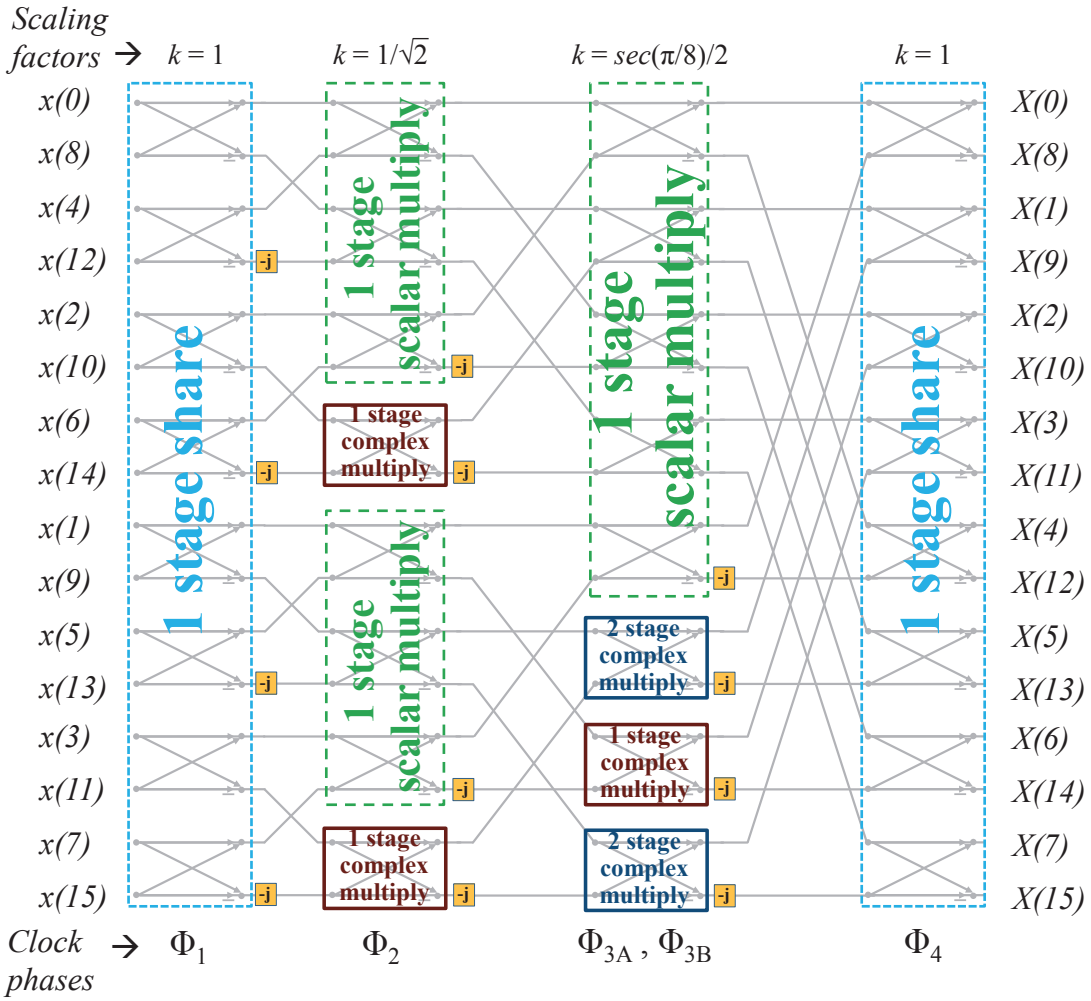


Figure 6.7: The 16 point CRAFT processor implementation algorithm showing the different butterflies used, the clock phases (Φ), and the scaling factors (k) per stage

capacitors. Since each input is operated on twice in an FFT butterfly, and twice for complex operations, four copies of each sample are required. Also, considering *I, Q* (=2) and differential (=2) inputs, the 16 point FFT requires 16×2 (complex math) $\times$ 2 (butterfly branches) $\times 2$ (*I, Q*) $\times 2$ (pseudo-differential) $= 256$ sampling capacitors.

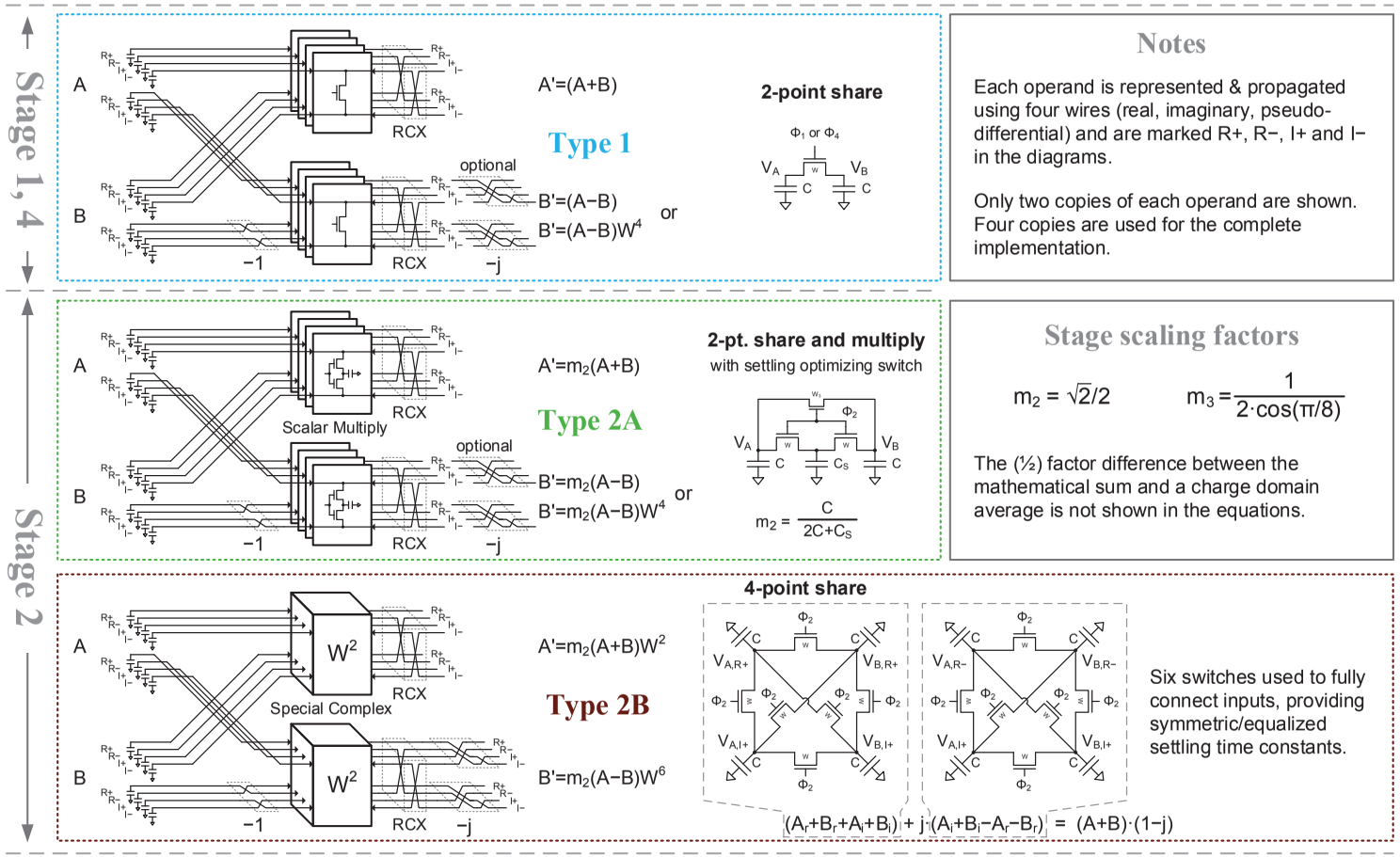


Figure 6.8: The four FFT stages and their component types of butterflies are shown. Note that the capacitors holding the operands for each buttefly (at the left of the diagram) are the sampling capacitors. *(continued)*

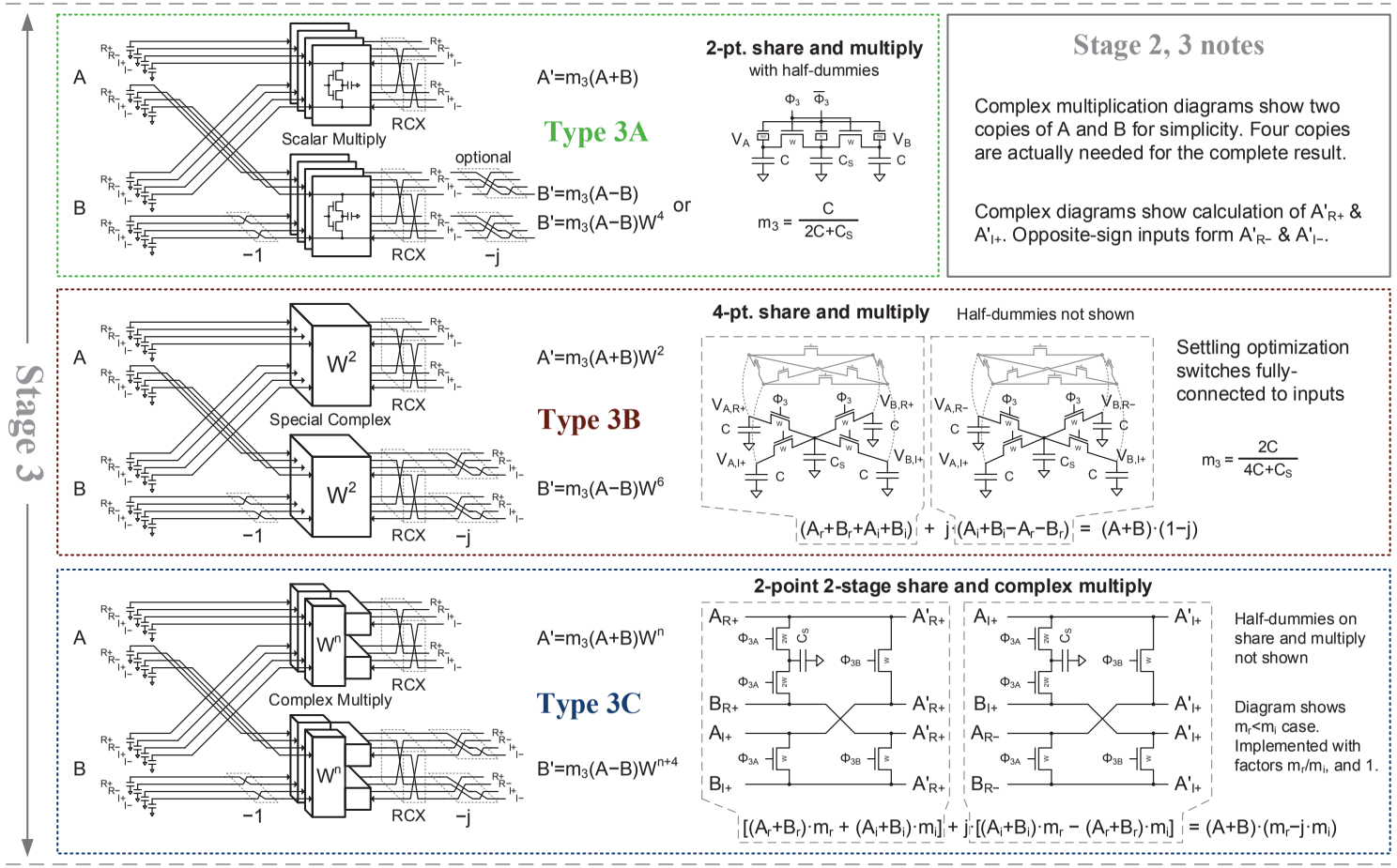


Figure 6.8: The four FFT stages and their component types of butterflies are shown. Note that the capacitors holding the operands for each butterfly (at the left of the diagram) are the sampling capacitors.

6.3 CRAFT Implementation

The CRAFT engine is implemented as shown in Fig. 6.9. This illustration closely emulates the layout floorplan. Images of the full on-chip implementation layout, as well as that of only the processing core are shown in Fig. 6.10 andFig. 6.11 respectively. The processing core, shown inFig. 6.9, includes 4 ($= \log_2(16)$) stages of FFT butterflies similar to those represented in the signal flow graph in Fig. 6.7. Also in Fig. 6.9, note how the butterflies types are differentiated to correspond with those in Fig. 6.7 and in Fig. 6.8. As shown, the core design is supplemented with input voltage samplers for the capacitive input array and output latches and multiplexers for data read-out. A block-level detailed description of the figure follows later in this section.

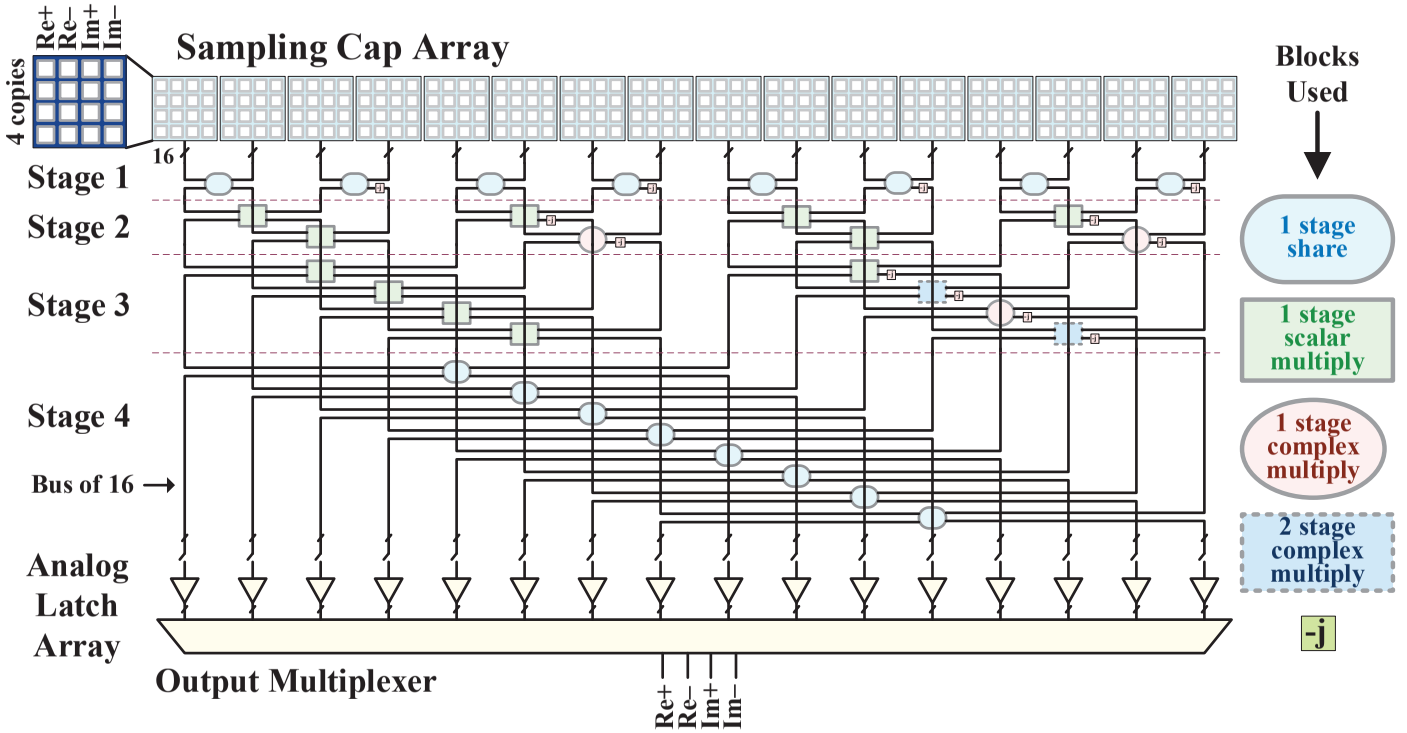


Figure 6.9: A layout-mimicing schematic diagram of CRAFT showing the different elements of the design

All the blocks shown are controlled and clocked by two state machines. The timing diagram for the operations is shown in Fig. 6.12. The sampling and processing operations, requiring 16 and five phases respectively, are allowed equal amounts of time in anticipation of an interleave-by-2 implementation. The longer processing phase clocks allow for more settling time, lowering both the error and the power consumption in the computations. As currently implemented, the reset phase prevents contiguous time window conversions in an interleave-by-2 design. This is only a trivial limitation of our implementation as the processing clock durations can be easily reduced for only a marginal increase in power[2]. Similarly, the processing phases can be further reduced to obtain a lower latency design at the expense of higher power. For the energy optimized design described in this work, the optimization of the duration of each phase is discussed in Sec. 6.4.

Dynamic optimization is possible through digital control of the clock phases by state machines. In particular, at lower clock rates the switches provide more settling than is necessary for the same computational accuracy. Supply voltage or processing clock period modification allow some of this "extra" computational accuracy to be traded for power savings. This is useful as some other block, such as the input sampler or ADC, may now be limiting the system's lower dynamic range bound.

6.3.1 Sampler

Sixteen banks of sixteen sampling switches and capacitors compose an array, as shown in Fig. 6.9, and capture the input in a time-interleaved fashion as controlled by the 16 sample phases in Fig. 6.12. Fig. 6.13(c) shows the pseudo-differential sampler design used in CRAFT. Compared to other traditional approaches shown in Fig. 6.13(a)

[2]Reducing the clock duration requires wider switches for the same settling accuracy. This translates into increased switching power dissipation and non-ideal effects due to device parasitic capacitances.

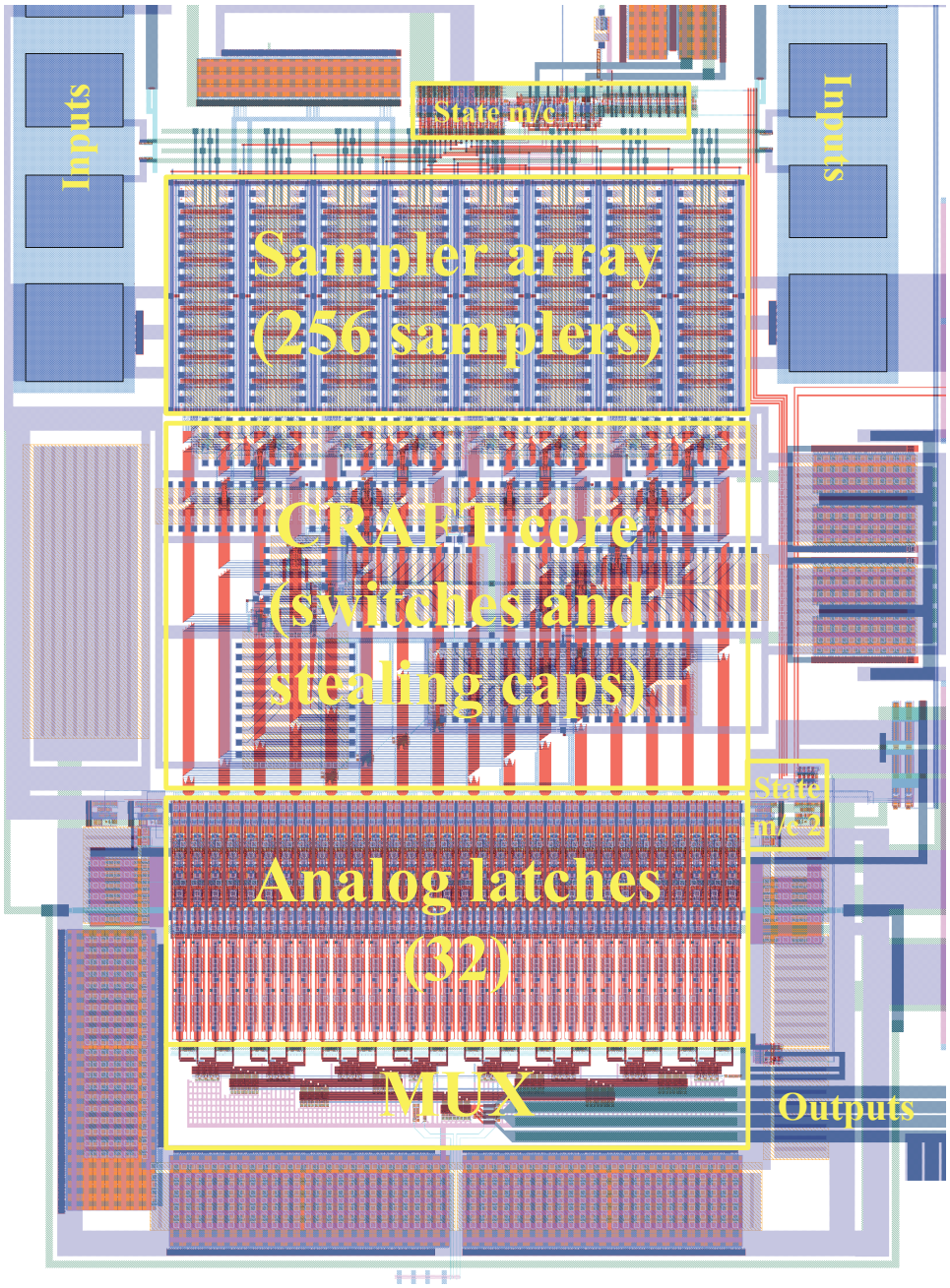


Figure 6.10: A screenshot of the CRAFT implementation and test structures layout

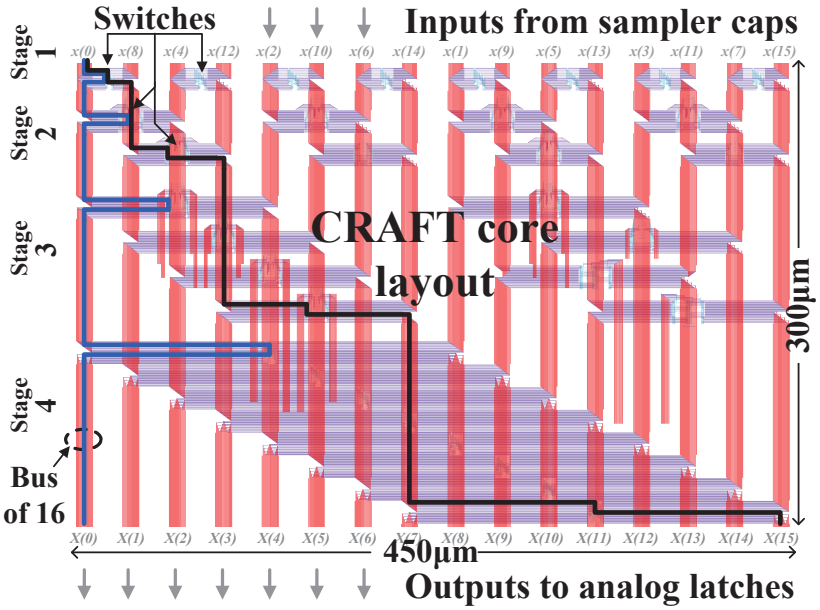


Figure 6.11: A screenshot of the CRAFT processing core layout with two highlighted example sampling capacitor distribution cases to be nominally matched for parasitics

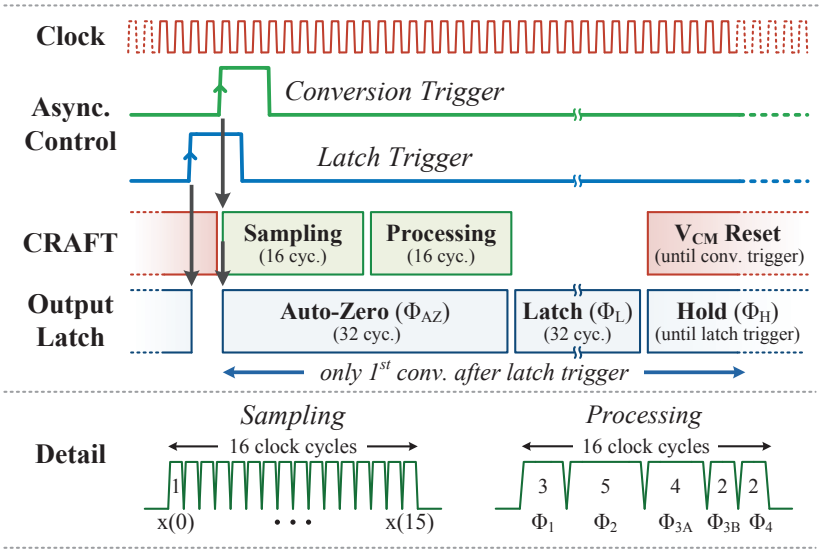


Figure 6.12: System timing diagram showing: clock, trigger, sampling, processing, reset, and output analog latch timing

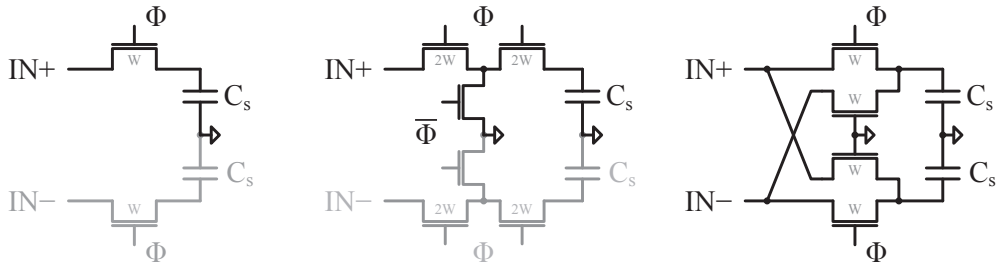
and (b), this topology provides excellent signal feedthrough cancellation for differential signals.

Isolating the inputs from the sampling capacitors is particularly important during any charge-domain processing to prevent corruption of operations and results. Because of signal rewiring and the use of a pseudo-differential operand representation, signal feedthrough of different polarities and from both the I and Q inputs may combine for an unpredictable and undesired impact on the computational accuracy (non-linearity).

During the sampling hold mode, when the switches are off, the following mechanisms cause signal feedthrough and leakage: (a) capacitive feedthrough due to the appreciable overlap capacitance (C_{ov}) to sampling capacitance ratio: C_{ov}/C ratio, and (b) sub-threshold leakage dependent on the input as well as the sampled and held voltages (see Fig. 6.13(a)).

In wideband samplers, active circuits are often used in the sampler and may also be employed to reduce signal feedthrough. Techniques range from combining with an inverted signal in a single-ended design [36] to using active feedforward cancellation in a differential design [37].

A traditional passive way of improving isolation is by grounding the intermediate node while in hold mode as shown in Fig. 6.13(b) [38]. This reduces the capacitive feedthrough as well as the input dependence of sub-threshold leakage. However, to maintain the same on-resistance, switch widths must double, power consumption increases by more than 4X, and the total parasitic loading of the input driver increases. Additionally, the matching of the sampling capacitance is impacted negatively. This is due to the fact that the total sampling capacitance including parasitics ($C_{\text{total}} = C_{\text{cap}} + C_{\text{para}}$) has a larger voltage-dependent component (C_{para}). For the requirements in CRAFT, a simulated $\sim$50dB isolation improvement compared to (a) can be realized for a 4.1X power increase.



(a) Basic (b) T-switch (c) Proposed

Figure 6.13: The proposed off-state feedthrough cancellation sampler (c), compared with two traditional pseudo-differential samplers (a), (b)

In comparison, the proposed sampler configuration (Fig. 6.13(c)) uses matched cross-coupled dummy feedthrough paths from the differential inputs to cancel the capacitive signal feedthrough at the sampler output nodes without any additional switching or active power consumption. Moreover, sub-threshold input leakage is differentially-suppressed for small hold voltages as a path to both IN+ and IN- generate a common-mode effect. This additional benefit is useful in light of the in-place nature of the charge-domain operations. In particular, computations with small results will be provided more protection from sub-threshold sampler conduction leakage, helping maximize computational dynamic range. Finally, parasitic loading is only slightly increased compared to (a) and the additional parasitic capacitance on the output nodes has a lesser voltage-dependence since the cross-coupled switches' C-V characteristic is operating in the negative V_{gs} region.

Simulations suggest that the proposed design in Fig. 6.13(c) cancels feedthrough and provides isolation primarily limited by local device and layout mismatch. For example, a matching of 0.1% (3σ) for C_{ov} and layout parasitics achieves 75dB average (64dB $\mu+3\sigma$) isolation compared to Fig. 6.13(a).

The choice of the size of the sampling capacitor is a trade-off between the power and

the integrated noise in the circuit. Larger capacitors have lower noise at the expense of requiring switches of lower resistance (larger width) for equal sampling performance and processing speed. For capacitor size selection, the non-linearity goal in the circuit is approximately 60dB and the integrated noise level is placed lower than this so that the circuit is not noise limited. To meet this condition, the sampling capacitors are sized at $200fF$ each such that the total output referred $k\mathrm{T}/C$ noise contribution by the CRAFT engine due to the reset, sampling, and processing operations is 63.4dB below the output full-scale. Details of the noise contribution of the processing operations is discussed in Sec. 6.4.

As shown in Fig. 6.9, 16 buses of 16 lines are connected to the sampling capacitors, and run through the CRAFT core. These distribution wires are permanently connected to the sampling capacitors. Consequently, their parasitic capacitance is part of the total sampling capacitance and needs to be accurately matched. Additionally, since the operations are performed in-place, the outputs appear on the sampling capacitors at the end of the processing phase.

Inputs are provided using 50Ω terminated GSSG probe-pads, which feed the sampler array. When integrated with a wideband RF or IF front-end, the removal of these extra termination ease the power and drive requirements of the front-end. The large signal-swing desired at the sampler input is no different than if it were the sampler of a conventional ADC following the RF front-end.

As shown in Fig. 6.9, each input sample is stored using a set of 16 capacitors (four copies of pseudo-differential, complex inputs). For a 16 point FFT, 16 sets are used totaling 256 capacitors. The 256 sampling switches are designed to run at 5GS/s with a total differential input swing of 1.2V and target 60dB SFDR in simulation. Extracted simulations including signal source, parasitics loading, and sampling clock fall-time effects resulted in 55–60dB SFDR over the entire input bandwidth with full-scale inputs.

Inputs at −6dBFS, as was used for calibration, supported 73–85dB SFDR in simulation.

The design samples and processes signals using a pseudo-differential representation; two capacitors, each with voltages ranging from 0 to $2V_{cm} = V_{cm} \pm V_p$, give a full-scale signal range of $V_{pp,\text{diff}} = 4V_{cm}$. Advantages of this arrangement include: maximized swing while using nMOS-only switches, junction leakage issues inherent to below-ground swing are avoided, wire swapping can be used for performing negation without bottom-plate switches, capacitors can use a shared bottom-pate for considerable area savings, and signal-independent non-idealities, like clock feedthrough, appear as common-mode. Although not used in this work, bottom plate sampling is a possible method to alleviate charge injection errors in the passive sampler [39].

6.3.2 CRAFT Core

The CRAFT core performs the FFT operation in four ($= \log_2(16)$) stages as represented in Fig. 6.7. This is accomplished using the five processing clock phases shown in Fig. 6.12. The CRAFT operation, represented in matrix form as shown in Eq. (6.2), can be separated into the four constituent processing stages, denoted by $\mathbf{S_i}$ where i is the stage number, in Eq. (6.4). This decomposition matches the signal flow graph in Fig. 6.4(a) and Fig. 6.7.

$$\mathbf{F} = \tfrac{1}{16} \cdot \mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1\mathbf{S}_{\text{bitrev}} \tag{6.4}$$

As each $\mathbf{S_i}$ contains only a single stage of computations, it easily maps to a set of in-place charge-domain operations. The $2^4 = 16$ division factor is from the inherent scaling by $\frac{1}{2}$ in each addition (charge-averaging share) operation. The $\mathbf{I}_{\text{bitrev}}$ and $\mathbf{S_i}$ matrices are expanded in Appendix A.

As described in Fig. 6.6, each butterfly is conceptually implemented using switches and scaling capacitors to perform a set of share and multiply operations on the input

samples. Moreover, the *a priori* knowledge of the exact butterfly operations are exploited to hard-wire a number of modifications that reduce the number of clock phases required per stage, and limit the signal attenuation inherent to passive computation. The implementation details of these improvised butterflies in each of the four stages is shown in Fig. 6.8 and described below in detail.

Stage 1

The first stage requires only share operations, as shown in Fig. 6.4(a), and uses a single processing-clock cycle (Φ_1), as shown in Fig. 6.12. The linear operation performed is represented by the term $\frac{1}{2}\cdot\mathbf{S}_1$ expanded in Appendix A. The scaling factor of $\frac{1}{2}$ is due to the use of charge-averaging operations as opposed to mathematical sum operations.

The butterflies in this stage are nearly identical with the only difference being a wire-swapping based multiplication by '$-j$' (W^4) on some outputs as shown in Fig. 6.7, and Fig. 6.9. The implementation of the butterflies are shown in the top of Fig. 6.8 (Type 1). Note that the magnitude of the multiplicand in this stage is unity. Due to the trivial scaling factors, both multiplication and complex operation are completely avoided as shown in Fig. 6.8.

For the Type 1 butterfly implementation, two operands A and B perform a 2-point share operation. Each operand is represented using 2 copies of real and imaginary pseudo-differential voltages ($\Re\pm$ and $\Im\pm$) stored on 8 capacitors[3]. The corresponding ($\Re\pm$ & $\Im\pm$) charges on these capacitors are shared when the stage 1 processing clocks are enabled. Differential RC settling-error cancellation (RCX), as discussed in Sec. 6.4.3, is performed using wire-swapping at the outputs.

[3]Two copies of this butterfly diagram in Fig. 6.8 are implemented in the actual circuit to provide 4 copies to support the complex butterfly stages utilized in stages 2 and 3.

Stage 2

The second stage comprises primarily share and multiply operations as shown in Fig. 6.4(a), and is represented by the $\frac{1}{2}\cdot\mathbf{S}_2$ matrix in Appendix A.

In this stage two types of operations are performed: share and multiply by unity, including W^0 and W^4 using '$-j$' wire-swaps, and share and multiply by $W^2 = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}j$. We note that a share followed by multiplication by a complex twiddle factor $(m_r - jm_i)$ with equal real and imaginary parts ($m_r = m_i = m$) can be simplified [40] as shown:

$$\underbrace{(A+B)\cdot(m_r - jm_i)}_{\text{share and complex mult.}} = \underbrace{(A_r + B_r + A_i + B_i)\cdot m}_{\text{4-point share and multiply}} + j\underbrace{(-A_r - B_r + A_i + B_i)\cdot m}_{\text{4-point share and multiply}}$$
$$= [(A_r + jA_i) + (B_r + jB_i)]\cdot m(1-j)$$

As a result, a 2-stage complex multiplication (Fig. 6.6 row 6) can be replaced with a single-stage share and multiply operation with 4 operands. Moreover, this share and multiply operation can be implemented with a scaling factor of m=1 to reduce attenuation. With this scaling factor modification, the stealing is effectively removed and it can now be replaced by a purely sharing operation, as shown in Fig. 6.8 (Type 2B). The pseudo-differential nature of the input operands is used along with hard-wiring of wiring-swaps to perform the desired computation using all of input operands and copies and generate all of the required output copies and polarities. For perfect symmetry and equalized differential-settling between every pair of capacitors in the 4-point operation, the fully-connected switching configuration as shown in Type 2B is used.

The scaling due to a charge-domain 4-point share operation is $\frac{1}{4}$ since it has an averaged voltage output rather than a mathematical sum. So, the maximum magnitude that can be implemented for a butterfly with W^2 twiddle factor is $\frac{\sqrt{2}}{4}$ using this stage scaling factor normalization technique. The other butterflies in this stage, which

originally required a scaling factor of unity, should scale by $\frac{1}{\sqrt{2}}$ for equal magnitude, remembering that charge-averaging of two operands provides an additional factor of $\frac{1}{2}$ compared to a mathematical sum. This scaling is obtained using a 2-point share and multiply operation shown in Fig. 6.8 (Type 2A). The third switch, in Type 2A, is used for power-efficiently improving the differential settling as discussed in Sec. 6.4.3.

This improvisation using a 4-point share operation eliminates one clock phase, thereby providing for more settling time, and minimizes the mathematical attenuation inherent to charge-computations for this stage of the FFT by $\sqrt{2}$.

Stage 3

The third stage utilizes share and multiply operations as shown in Fig. 6.4(a) and is represented by the $\frac{1}{2}\cdot\mathbf{S}_3$ matrix in Appendix A.

As seen in the figure, the twiddle factors in this stage include $W^1 = j\cdot W^5 = \cos(\frac{\pi}{8}) - j\cdot\sin(\frac{\pi}{8})$, and $W^3 = j\cdot W^7 = \sin(\frac{\pi}{8}) - j\cdot\cos(\frac{\pi}{8})$. These butterflies require the share and complex multiplication shown in Fig. 6.6 row 5, and consume two clock cycles: Φ_{3A} and Φ_{3B} in Fig. 6.12. Also, due to the supplementary and orthogonal nature of these pairs of twiddle factors, only two sizes of stealing capacitors, and m=$\cos(\frac{\pi}{8})$ and m=$\sin(\frac{\pi}{8})$, are necessary. In conjunction with '$-j$' wire-swaps, this duplication helps reduce mismatch and minimize the number of systematic error sources relative to an ideal FFT. The implementation of these butterflies are shown in Fig. 6.8 (Type 3C). As shown, the complex multiplication is implemented in 2 stages. The entire operation, and FFT stage, is scaled by $1/m_i$ in order to reduce the total attenuation by $sec(\pi/8)$. Consequently, only one scaling capacitor is necessary for this butterfly type as shown.

For the butterflies with trivial twiddle factors (W^0 and W^4) the unity scaling is performed with a share operation as in earlier stages. However, due to the modified FFT stage factor from the attenuation reduction technique used in the complex butterflies,

these butterflies require a constant scaling factor to have proper magnitude. On top of that, a 2-stage operation introduces a $\frac{1}{2}$ factor difference between addition and averaging twice. So, like in stage 2, a 2-point share and multiply operation is used instead and $m{=}\frac{1}{2}\sec(\frac{\pi}{8})$ to match the magnitude of the complex butterflies. These butterflies, shown in Fig. 6.8 (Type 3A), utilize both Φ_{3A} and Φ_{3B} for their operations. Half-dummy switches are used as shown to cancel clock feedthrough and reduce charge injection (details in Sections 6.4.5 and 6.4.6).

Multiplication by W^2 is performed using a technique similar to the 4-point share used in stage 2. However, for stage 3, scaling is required to match the inherent attenuation in the Type 3C butterflies in this stage. Therefore, a stealing capacitor is utilized in the Type 3B butterflies in a 4-point share and multiply operation. As the mathematical attenuation of a 2-stage 2-point complex operation matches that of a one-stage 4-point share and multiply, a scaling factor of $m{=}\sqrt{2}\sec(\frac{\pi}{8})$ is necessary for a W^2 butterfly of proper magnitude. The optimized switching configuration used is determined and shown in Fig. 6.8 (Type 3B). The additional switches fully-connected between the inputs improve the differential settling in a power efficient manner. Half-dummy switches added for clock feedthrough cancellation were also used but are omitted to avoid clutter in the diagram.

The increase in stage 3 power consumption due to half-dummy switches was deemed necessary to suppress of the impact of charge-injection from this stage on the FFT output computation non-linearity.

Stage 4

The fourth stage comprises only share operations as shown in Fig. 6.4(a) and is represented by the $\frac{1}{2}{\cdot}\mathbf{S}_4$ matrix in Appendix A. These butterflies are identical in type to the butterflies in stage 1 and are not redrawn in Fig. 6.8.

Using the improvised share and multiply techniques, only five clock phases are used for FFT computation. Moreover, the total charge-domain attenuation in the CRAFT core is limited to 0.383 ($= 1 \times \frac{1}{\sqrt{2}} \times \frac{1}{2}\sec(\frac{\pi}{8}) \times 1$) as shown in Fig. 6.7. This attenuation is 6.1X (15.7dB) superior compared to an unmodified implementation[4].

Note that the butterfly implementation depicted in Fig. 6.9 imitates the signal flow shown in the last row of Fig. 6.6; as a result, after each operation, half the wires return to their bus while the rest continue on the other bus. To equalize the sampler wiring parasitics, the switches are always placed midway between two operand buses. Two example wires, one always returning to its own bus while the other always shifting on to the other operand bus, are highlighted in the layout screenshot in Fig. 6.9. As seen, the two wire lengths (and their associated parasitics) are nominally matched. Also, the series wiring resistance must be considered as it comes in series with the switch resistance and sufficient voltage settling must be ensured during the switch on period.

6.3.3 Output Latch

On the far end of the processing core, the wiring buses connect through switches to analog latches to save the outputs prior to being read out (Fig. 6.9). Thirty-two (16×2 for real and imaginary) of these latches capture the entire FFT output. These analog latches are needed only for measurement purposes to prevent capacitor leakage due to the slow sequential read-out and for driving the large load presented by the off-chip ADCs. Once ADCs are integrated, the CRAFT engine's pre-sampled outputs will directly interface with the ADCs, obviating the need for these latches.

To match the CRAFT performance, a two-stage, folded-cascode, differential operational transconductance amplifier (OTA) with 70dB gain and 900MHz UGF is designed.

[4] An unmodified implementation uses only generic complex scalar butterflies (Fig. 6.6, row 6).

The OTA is used in a differential switched-capacitor analog-latch, shown in Fig. 6.14, with a closed-loop gain of 2.5 providing a −7.8dBV full-scale output. It is based on a switched-capacitor instrumentation amplifier [17] modified for a capacitor-based input and having different input and output common-mode voltages. The corresponding clock phases used are shown in Fig. 6.12. The latch clocks are enabled for only the first CRAFT conversion after being triggered. The latch performs offset cancellation during the CRAFT sampling and processing phases (Φ_{AZ}), and latches the CRAFT output with a 10τ settling accuracy during the next 32 clock phases (Φ_L). Finally, the output load is connected and the output is held (Φ_H) for output read until the external measurement system requests a new latching event. Switched-capacitor based common-mode feedback for the OTAs is performed during the Φ_{AZ} phase.

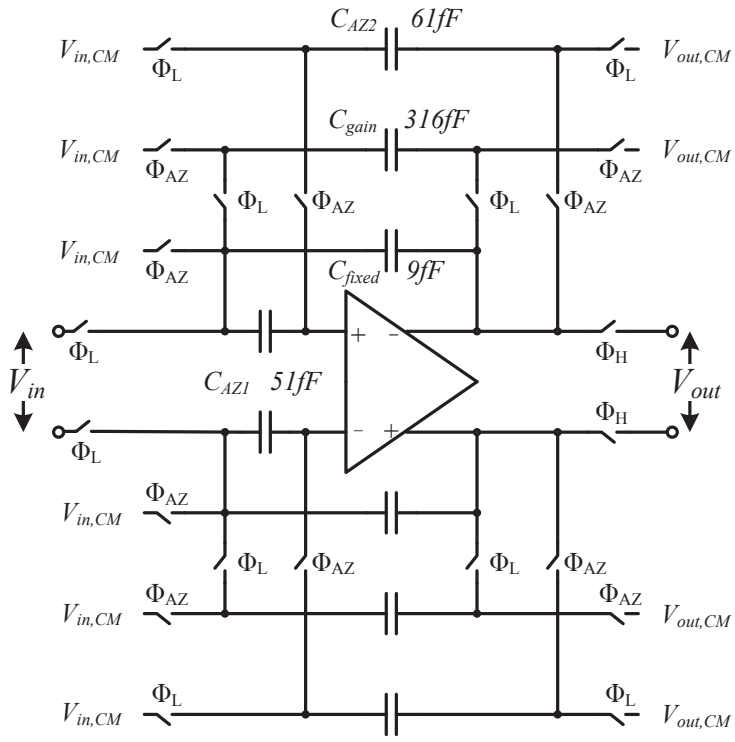


Figure 6.14: The analog latch used in a 32-element array to save CRAFT results

6.3.4 State Machines

As shown in Fig. 6.15, the sampling array, CRAFT processing engine, and output testing interface require multiple clocks to operate and interface with the testing equipment. The input clock is used to generate all internal signals. State machine 1 (SM1) is externally triggered to initiate a conversion. It generate 16 sampling clock phases followed by the processing clocks to operate the CRAFT core switches, as shown in Fig. 6.12. Another state machine (SM2) uses handshaking with SM1 and an external trigger to determine when CRAFT outputs are valid. It subsequently generates the clocks for the analog latch array to save the first CRAFT conversion after being triggered. The latched outputs are then observed sequentially using the integrated low-resistance analog multiplexer (16×4 to 1×4 for differential real and imaginary outputs from one bin). This setup allows asynchronous operation between the conversion and latch triggering.

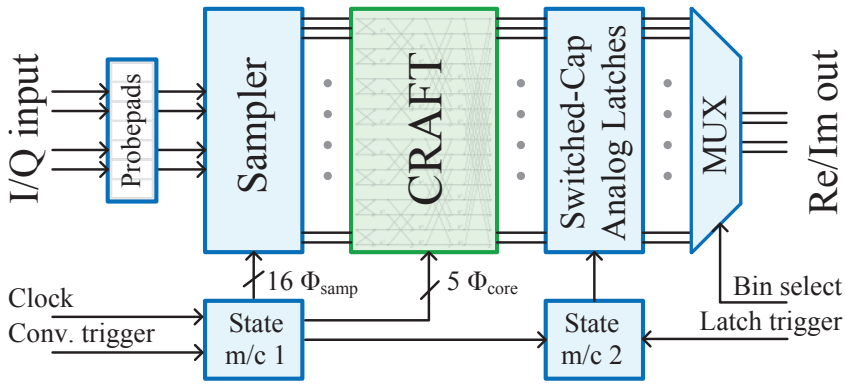


Figure 6.15: CRAFT processor and supporting on-chip circuits

6.4 Circuit Non-idealities and Optimization

As described, control of the design relies heavily on digital state machines and the schematic and layout regularity and wiring complexity approaches that of digital designs. Despite these properties, the CRAFT engine is a sensitive analog circuit, and still remains vulnerable to circuit non-idealities including noise, matching, parasitics, and device non-linearity. Consequently, accurate modeling of all these non-idealities is critical to optimize charge-domain operations for large dynamic range. Moreover, switched capacitor circuit noise, charge-injection, and charge-accumulation are not adequately modeled in the Spectre models available to us. Therefore, to isolate the impact of these effects per stage, it was necessary to enable/disable them independently in simulation.

For this design, CRAFT-specific models of dominant non-idealities (noise, settling, clock-feedthrough, charge-injection, etc.) are developed in MATLAB®. Each non-ideality is modeled as an independent error source. These error sources can be enabled and disabled independently in simulation for each stage of the CRAFT implementation to enable isolation of the impact of each error source. Also, *a priori* knowledge of the operations is utilized to devise novel circuit techniques such as correlated noise reduction, differential settling error cancellation, and high-linearity switches. Performance is optimized using system simulations that model both the non-idealities and the new circuit techniques. This enables the high dynamic range even at 5GS/s. The different sources of non-ideality, their modeling, and mitigation techniques used for CRAFT are outlined below.

The different operations in CRAFT can be distinguished into the reset, input sampling, and subsequent processing phases. In this section, the effect of noise in the sampler as well as the processing core is discussed. The effect of the sampler providing inputs to CRAFT is then briefly outlined. This is followed by a discussion on other non-linearities

in the CRAFT core processing[5] switches, namely: incomplete computational settling, operand-dependent processing-switch resistance, clock-feedthrough, and charge injection and absorption [20]. Modeling methods are described and mitigation techniques, both independently developed and those outlined in previous literature [1, 41], are utilized.

6.4.1 Noise

Sampling Noise

Motivation During a voltage sampling operation, noise, of power $k\mathrm{T}/C$, manifests as a hold-value disturbance as the sampling switch opens. Noise due to the reset operation of the CRAFT core (sampling capacitors reset to V_{cm}) does not matter[6] in this implementation since voltage sampling over-writes it. In this design, 4×2 capacitors, $200f$F each, are used to sample four copies of each input pseudo-differentially. The full-scale input is $V_{pp,\mathrm{diff}} = 1.2V$. This sampler has a sampled-noise voltage of $\sqrt{\overline{v_n^2}} = V_{n,rms} = 144\mu V$ on each of the four single-ended copies (−63.4dBFS). As the noise is uncorrelated, averaging these copies at the output and forming a differential output gives a total noise of $V_{n,rms} = 144 \cdot \frac{1}{\sqrt{4}} \cdot \sqrt{2} = 102\mu V$ (−72.4dBc) for a full-scale single-tone input.

Modeling Sampling noise effects are included into our MATLAB®system simulation model. After sampling or resetting, a zero-mean Gaussian random variable with $\sigma = V_{n,rms} = \sqrt{k\mathrm{T}/C}$ is added to the each sample capacitor's final value.

[5]The sampling process and its related non-idealities have been well-studied in literature, hence the short section and focus on the processing non-idealities.

[6]This is not true for a current-domain sampler: noise during the reset phase disturbs the initial voltage of the capacitor upon which integration is performed during the sampling period. Then, when the switch opens at the end of the sampling period another noise event occurrs.

Mitigation The sampling capacitor size sets both the sampler's noise as well as the baseline for the processing noise because the same capacitors are used for computations. Since the processing core has attenuation, the sampling operation is not a dominant source of the total tabulated noise. Therefore, the capacitor size is selected to be 200fF/capacitor based on the calculated total output-referred noise.

Processing Noise

Motivation Similar to the sampling operation, noise from the CRAFT core switches modify the final value of computations. The noise power and correlation depends on the type of charge-domain operation: number of operands, implementation method, and multiplication factor. The derivation and analysis of noise in different charge-domain computation configurations is detailed in Chapter 5 and listed in Table 5.1.

At the end of every 2-point share operation, $\frac{1}{2}\cdot k\mathrm{T}/C$ noise power is added to each output copy. Interestingly, this instantaneous sampled noise on the two capacitors arising from a single switch is completely correlated (equal magnitude, opposite signs). Similarly, the noise sampled on the capacitors during a share and multiply operation arise from a set number of switches, and are therefore partially differentially correlated.

In addition to processing noise, these operations are affected by noise sampled on the CRAFT core stealing capacitors, C_s, at the end of the reset phase. This reset phase charge error on C_s distributes to the operand capacitors during share and multiply operations, giving a common mode noise term by the end of a settled computation. This is covered in Sec. 5.3.

Finally, note that noise of the input operands (for e.g., from the previous stage of operations) gets averaged by the settling of the subsequent (charge-averaging) operation. The statistical nature of the noise effects described above make both hand analysis and accurate modeling possible.

Modeling For modeling noise, zero-mean Gaussian random variables are used for both reset and processing phases. The noise on the operand outputs is a combination of the noise generated by each switch. The magnitude and sign of the linear combination is dictated by the switches' noise-transfer functions (NTF) for the particular switch configuration used as detailed in Chapter 5. This modeling method generates the proper noise correlation on the output copies, that, when averaged over multiple simulations, provide the expected output referred noise power. For example, for a 2-point share operation (row 1 of Fig. 6.6), the integrated rms noise voltages at each output is given by $V_{n,rms} = \sqrt{0.5 \times kT/C}$, but are perfectly differentially correlated. This configuration is used in stages 1 and 4 of CRAFT, shown in the Type 1 configuration in Fig. 6.8.

Mitigation Noise of the later stages of the CRAFT engine is reduced due to copy averaging. Four copies of each output are available, and are shared to reduce the total noise power by four. Moreover, correlation, ρ, between these copies can be exploited for noise reduction if the noise-correlated pairs appear at the same output. In our design, the copy averaging before the latching operation is used to reduce stage 3 and stage 4 noise contributions to the output.

Total Output Noise The noise contribution of each stage is computed analytically and tabulated in Table 6.1. The attenuation (A_v) reduces the output referred noise by A_v^2. The total single-ended, copy-averaged output referred noise is computed as shown in the last column yielding a total output-referred noise of $0.26 \cdot \frac{kT}{C}$ (-63.9dBFS per differential $\Re$, $\Im$ output for a noise EVM of -61.9dBFS[7]). Compared to the sampling operation without processing or attenuation, CRAFT has a noise figure of 8.5dB.

[7] $\sqrt{\Re^2 + \Im^2}$ is a chi distribution with 2 degrees of freedom. For $X_i \sim \mathcal{N}(0,1)$ it has mean $\sqrt{\pi/2}$ (=1.96dB).

Table 6.1: Summary of noise contribution in CRAFT

Stg.	**Noise sources**	$\overline{V_n^2}$/copy	$\mathbf{A_v}$	$\boldsymbol{\rho_{out}}$	$P_{n,\mathbf{out}} = \left(\frac{1}{4}\right)\overline{V_n^2}\mathbf{A_v^2}(1+\boldsymbol{\rho})$
	Sampler (4-copy)	$1.000 \cdot \frac{kT}{C}$	0.383	0	$0.0366 \cdot \frac{kT}{C}$
1	2pt. share	$0.500 \cdot \frac{kT}{C}$	0.383	0	$0.0183 \cdot \frac{kT}{C}$
2	2pt. sh/mult. (m=0.707)	$0.750 \cdot \frac{kT}{C}$	0.541	0	$0.0549 \cdot \frac{kT}{C}$
3	2pt. sh/mult. (m=0.541)	$0.854 \cdot \frac{kT}{C}$	1	-0.29	$0.1521 \cdot \frac{kT}{C}$
4	2pt. share	$0.500 \cdot \frac{kT}{C}$	1	-1	0
	Total output noise including reset operation				$\mathbf{0.2619} \cdot \frac{kT}{C}$

6.4.2 Input-dependent Sampling-switch Resistance

Motivation Any corruption of the inputs to CRAFT combine with the processing core non-idealities to form the dynamic-range-defining non-linearity floor of the outputs. During the sampling process, the finite resistance of the sampling switch introduces a first-order low-pass response on the input signal at the sampling capacitor (consider Fig. 6.13(a)). This pole causes frequency-dependent attenuation and phase shift of the signal sampled. Additionally, the switch on-resistance is voltage dependent, causing the filtering effect to be a non-linear function of the input voltage. This non-linearity creates unwanted distortion that is often quantified by the total harmonic distortion (THD). In differential designs even harmonics are suppressed [19, pg. 452] which often leaves the 3rd harmonic as the dominating contribution to the THD. The amplitude of the spurious 3rd harmonic tone with respect to the desired fundamental (carrier) tone is specified in dBc (dB with respect to carrier.) This metric is used to compare the simulated sampler performance with the requirements. Techniques to reduce the resistance variation over the input voltage swing range, such as constant-V_{GS} samplers [42], are useful to further improve linearity but have degrading performance when operating at very high speed [43].

Modeling The sampling process is modeled in MATLAB®using a single input-pole to match the attenuation and phase shift of the design. Non-linearity modeling of the sampler was not included since the goal is to optimize the CRAFT processing core. The sampler's non-linearity was independently optimized using a conventional circuit simulator.

Mitigation If the sampling resistance is minimized, such that first-order pole is much higher than the highest frequency content of the input signal, the resistance-induced attenuation and phase shift due to the pole is minimized. However, for spectrum sensing applications, where the input bandwidth is typical large, this requires very large sampling switches and increased power consumption in the sampler. Additionally, large switches increase charge injection and clock feedthrough effects which can only be mitigated to first-order by using half-dummy switches [44]. Also, the non-linearity of the sampling switch resistance becomes a critical performance limiter, particularly when large swing is desired to maximize SNR.

For CRAFT, $W/L = \frac{3.92\mu m}{60nm}$ nMOS-only switches were used for a -3dB pole of approximately 6GHz. The proximity of this pole the upper frequency content of CRAFT (2.5GHz) is a reasonable compromise because the worst-case effects of 0.8dB attenuation and a 20° phase shift do not affect our ability to measure the performance of CRAFT. Also, the fixed phase and amplitude effects of the sampler are compensated for in the calibration method detail in Sec. 6.6.1. Extracted simulation of the sampler for different input frequencies and amplitudes showed it roughly matched the -60dB SNDR design goal of the CRAFT core. The off-state capacitive feedthrough of the sampler due to the switch size is canceled using the design shown in Fig. 6.13(c).

6.4.3 Incomplete Settling

Motivation For an RF discrete-time signal processor, very high sampling and processing speeds are mandated. The speed of operations in CRAFT directly trade-off with the settling accuracy of the operations. The switch size can be increased to allow better settling; however, this not only increases the power consumption, but also causes larger charge-injection and clock-feedthrough errors. To optimize the computation accuracy, the time-domain settling characteristics give useful insight. The settling equations are separated into ideal and error components, with settling speeds depending on the switch configuration. Consequently, different switch topologies can be contrasted by comparing their settling time-constants (τ) and total number of switches (power).

For example, in CRAFT, stages 1–4 have average simulated settling-time to time-constant ratios of: $\Gamma_{S1} = 7$, $\Gamma_{S2} = 4$, $\Gamma_{S3} = 5$, and $\Gamma_{S4} = 4$ respectively.

For the modeling and mitigation sections that follow, two aspects are considered. First, models are developed for a single-ended settling scenario. Later, this model is extended to the pseudo-differential operation utilized in CRAFT. Mitigation techniques, such as RC settling-error cancellation (RCX), are developed based on these models and exploitable correlation between error sources. Parts of the more general discussion of incomplete settling, from Sec. 4.5, are included here with additional CRAFT-specific analysis, and conclusions.

Modeling

(i) *Single-ended settling:* Consider two operand capacitors, each with a capacitance, C, and voltages of $V_a(t)$ and $V_b(t)$. A 2-point share operation, implemented as in row 1 in Fig. 6.6 and Fig. 6.8 (Type 1), has a settling response as shown in Eq. (6.5), and is plotted in Fig. 6.16 (see $V_a^+(t)$ and $V_b^+(t)$). The input capacitors (C), holding initial values of v_{a0} and v_{b0}, are connected by a switch of resistance

R_{sw} at $t = 0$. This yields a settling error time-constant: $\tau_d = R_{sw}C/2$.

$$\begin{aligned} V_a(t) &= \tfrac{1}{2}\left(v_{a0} + v_{b0}\right) + \tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d} \\ V_b(t) &= \underbrace{\tfrac{1}{2}\left(v_{a0} + v_{b0}\right)}_{\text{ideal result}} - \underbrace{\tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d}}_{\text{settling error}} \end{aligned} \tag{6.5}$$

A 2-point share and multiply operation, as shown in Fig. 6.6, has the settling response of Eq. (6.6). The input capacitors (C), with initial values v_{a0} and v_{b0}, are connected to the stealing capacitor (C_s, with no initial value[8]) by switches of resistance R_{sw}. The input-to-stealing-capacitor ratio defines the scaling factor of the operation: $m = \frac{2}{(2+C_s/C)}$.

$$\begin{aligned} V_a(t) &= \overbrace{\tfrac{1}{2}\left(v_{a0} + v_{b0}\right) m \left[1 + \left(\tfrac{1-m}{m}\right) e^{-t/\tau_s}\right]}^{\text{scaled-sum term}} + \overbrace{\tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d}}^{\text{difference term}} \\ V_b(t) &= \underbrace{\tfrac{1}{2}\left(v_{a0} + v_{b0}\right) m}_{\text{ideal result}} \underbrace{\left[1 + \left(\tfrac{1-m}{m}\right) e^{-t/\tau_s}\right]}_{\text{scaling error factor}} - \underbrace{\tfrac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d}}_{\text{difference settling error}} \end{aligned} \tag{6.6}$$

The difference and sum-settling time-constants for a 2-point share and multiply operation (2-switch variant), τ_d and τ_s respectively, are

$$\tau_d = R_{sw}C \qquad\qquad \begin{aligned} \tau_s &= R_{sw} \cdot (C \parallel C_s/2) \\ &= R_{sw}C \cdot (1 - m) \end{aligned} \tag{6.7}$$

(ii) *Pseudo-differential settling:* Pseudo-differential operands are described by the relationships of Eq. (6.8), where V^+ and V^- are the voltages on separate capacitors for the positive and negative components of the operand.

[8]Initial stealing capacitor value $v_{k0} = 0$, or $v_{k0} = V_{cm}$ where V_{cm} is the common mode voltage used as reference for all operations. Note that due to reset operation noise this initial value deviates from the ideal null value. Since this system is linear, this effect can be treated separately.

$$\begin{aligned} V_A &= V_a^+ - V_a^- \\ V_B &= V_b^+ - V_b^- \end{aligned} \tag{6.8}$$

Consequently, charge-domain operations are performed separately on these components. The time-domain settling response of these differential operands is shown in Fig. 6.16. For example, $V_A(t)$ is composed of two share and multiply settling operations as shown below. Using Eq. (6.6) with $m' = m\left[1 + \left(\frac{1-m}{m}\right) e^{-t/\tau_s}\right]$ and $\epsilon = \frac{1}{2}\left(v_{a0} - v_{b0}\right) e^{-t/\tau_d}$, we can expand Eq. (6.8) as follows:

$$\begin{aligned} V_A(t) &= \overbrace{\left[\tfrac{1}{2}\left(v_{a0}^+ + v_{b0}^+\right) m'^+ + \epsilon^+\right]}^{V_a^+(t)} - \overbrace{\left[\tfrac{1}{2}\left(v_{a0}^- + v_{b0}^-\right) m'^- + \epsilon^-\right]}^{V_a^-(t)} \\ &= \left(v_{a0}^+ + v_{b0}^+\right) \underbrace{\tfrac{1}{2}(m'^+ + m'^-)}_{m'_d} + \underbrace{(\epsilon^+ - \epsilon^-)}_{\epsilon_d} \end{aligned} \tag{6.9}$$

The differential nature ($v_{a0}^+ = -v_{a0}^-$ and $v_{b0}^+ = -v_{b0}^-$) allows this share and multiply expression to be written in a form similar to the single-ended expressions $V_a^+(t)$ and $V_a^-(t)$ within Eq. (6.9) ($m' = m'_d = 1$ for a share operations).

$$\begin{aligned} V_A(t) &= \left(v_{a0}^+ + v_{b0}^+\right) m'_d + \epsilon_d \\ V_B(t) &= \left(v_{a0}^+ + v_{b0}^+\right) m'_d - \epsilon_d \end{aligned} \tag{6.10}$$

For the differential operands, scaling error and inter-copy error, m'_d and ϵ_d shown below, now include the settling effects of the positive and negative components of the pseudo-differential representation.

$$\begin{aligned} m'_d &= \tfrac{1}{2}(m'^+ + m'^-) = m\left[1 + \left(\tfrac{1-m}{m}\right) \tfrac{1}{2}(e^{-t/\tau_s^+} + e^{-t/\tau_s^-})\right] \\ \epsilon_d &= (\epsilon^+ - \epsilon^-) = \left(v_{a0}^+ - v_{b0}^+\right) \tfrac{1}{2}(e^{-t/\tau_d^+} + e^{-t/\tau_d^-}) \end{aligned} \tag{6.11}$$

For small operand swings, $\tau_d^+ \approx \tau_d^-$ and the error's relationship to the two single-ended operations (pos. and neg.) is clearest: $m'_d \approx m'$ and $\epsilon_d \approx 2\epsilon$.

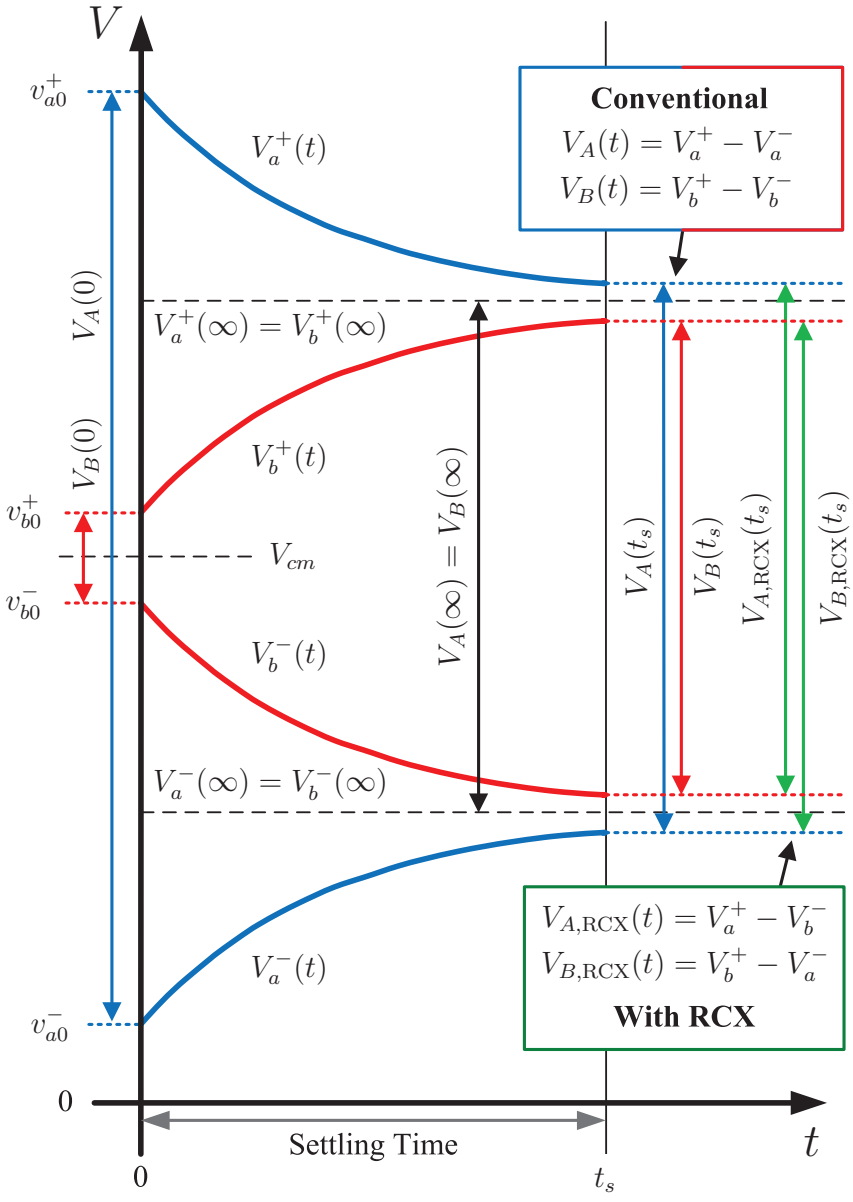


Figure 6.16: An example pseudo-differential charge-domain operation (share, m=1) settling showing how RCX conceptually reduces differential settling error

Mitigation

(i) *Single-ended settling:* The settling accuracy can be improved through the addition of a third switch ($R_{sw,3}$) for the share and multiply operation, as shown in Fig. 6.8 (Type 2A). This improves the differential settling error of the two operands by providing an alternate settling path. Note, from Fig. 6.8 (Type 2A), the widths of the main sharing switches (having a resistance, R_{sw}) effectively occur in series for the differential settling equation. As a result, increasing the third switch width (having a resistance, $R_{sw,3}$) improves the differential settling twice as power-efficiently[9] compared to widening the main switches (see Eq. (6.12) below).

$$\tau_d' = (2R_{sw} \parallel R_{sw,3}) \cdot (C/2) = R_{sw}C \cdot \left(\frac{1}{1+2R_{sw}/R_{sw,3}}\right) \tag{6.12}$$

(ii) *Pseudo-differential settling:* Considering these settling expressions and the situation under small operand swings, a method for RC settling error cancellation (RCX) is developed. Recognizing that V_a^- and V_b^- settle to the same value but have ϵ error of opposite sign, they can be interchanged to form the new differential operands below (compare with Eq. (6.8)).

$$\begin{aligned} V_{A,\mathrm{RCX}}(t) &= V_a^+(t) - V_b^-(t) \\ V_{B,\mathrm{RCX}}(t) &= V_b^+(t) - V_a^-(t) \end{aligned} \tag{6.13}$$

This is implemented using only wire swapping as shown in Fig. 6.16. As shown, swapping the output wires between the copies greatly reduces differential settling error. While the pseudo-differential "inputs" $(+,-)$ to the operation are (V_a^+, V_a^-) and (V_b^+, V_b^-), the capacitors holding the "output" results (two copies) are (V_a^+, V_b^-) and (V_b^+, V_a^-). This rewiring causes only a slight change to Eq. (6.10) for the two output copies as seen below.

[9]For our switch-sizing choice, the settling-power product $((e^{\tau_d})^2 \times P_{sw})$ improves $\sim$ 3dB.

$$\begin{aligned} V_{A,\mathrm{RCX}}(t) &= \left(v_{a0}^{+} + v_{b0}^{+}\right) m_d' + \epsilon_{d,\mathrm{RCX}} \\ V_{B,\mathrm{RCX}}(t) &= \left(v_{a0}^{+} + v_{b0}^{+}\right) m_d' - \epsilon_{d,\mathrm{RCX}} \end{aligned} \tag{6.14}$$

When written in this similar form $\epsilon_{d,\mathrm{RCX}} = \left(v_{a0}^{+} - v_{b0}^{+}\right) \frac{1}{2}(e^{-t/\tau_d^{+}} - e^{-t/\tau_d^{-}})$. The swap of the $(e^{-t/\tau_d^{+}} + e^{-t/\tau_d^{-}})$ term of ϵ_d (Eq. (6.11)) into $(e^{-t/\tau_d^{+}} - e^{-t/\tau_d^{-}})$ allows the differential-settling error to be canceled for small operand values.

$$(e^{-t/\tau_d^{+}} - e^{-t/\tau_d^{-}}) \xrightarrow{\tau_d^{+} \approx \tau_d^{-}} 0$$

Also, it can be shown that $\epsilon_{d,\mathrm{RCX}} < \epsilon_d$:

$$\begin{aligned} \epsilon_d = (\epsilon^{+} - \epsilon^{-}) \qquad & \epsilon_{d,\mathrm{RCX}} = (\epsilon^{+} + \epsilon^{-}) = (\epsilon^{+} - \epsilon^{-}) \left[\frac{\epsilon^{+} + \epsilon^{-}}{\epsilon^{+} - \epsilon^{-}}\right] \\ & \epsilon_{d,\mathrm{RCX}} = \epsilon_d \cdot \tanh\left(t \left(\frac{\tau_d^{+} - \tau_d^{-}}{2\tau_d^{+}\tau_d^{-}}\right)\right) \end{aligned} \tag{6.15}$$

This means that RCX always yields a net improvement[10]. RCX is used in all the CRAFT butterflies as seen in Fig. 6.8. Simulations show a net improvement of about 10dB in the FFT settling error from RCX.

6.4.4 Operand-dependent Processing-switch Resistance

Motivation The on-resistance of the nMOS-only processing switches are dependent on operand voltages. This deviation from the nominal switch resistance value, $R_{sw,\mathrm{nom}} = R_{triode}(V_G, V_D, V_S) = R_{triode}(V_{dd}, V_{cm}, V_{cm})$, causes a range of settling time-constants to be realized depending on the operand value (switch drain/source voltage) combinations. A simulation example of this effect on a pseudo-differential 2-point share operation is

[10] It is important to understand that the inter-copy error is not actually being canceled, it is just partially translated into a common-mode component. In case a stage with common-mode rejection follows, this settling error is rejected.

visible in Fig. 4.9. Furthermore, the time-variation of node voltages cause each R_{sw} to vary during the charge-domain computations. For example, a 2-point share operation (single-ended, operand nodes V_a and V_b) has one switch with resistance shown below.

$$R_{sw}(t) = R_{triode}(V_G(t), V_D(t), V_S(t)) = R_{triode}(V_{dd}, V_a(t), V_b(t))$$

Consequently, the systems of differential equations for share and scale and multiply operations, Eqs. (3.2) and (3.6) respectively, no longer have solutions as previously shown, which assume no time or node voltage dependence of τ.

Modeling Modeling methods for computation settling with voltage-dependent switch resistance are detailed in Sec. 4.6. Iterative evaluation of the circuit's differential equations is one method discussed. The second method, iterative application of the differential equations' solution, provides an additional level of abstraction from the specifics of the implementation. This technique is illustrated for share and share and multiply operations in Eqs. (4.30) and (4.32), respectively. Both of these simulation techniques are faster and less complex than device-level circuit simulations. They come with the caveat that they neglect device parasitics and potential non-linearity caused by them (i.e. voltage-dependent parasitic capacitances) However, in the context of large-signal swing for high SNR, the effects of switch resistance variation are dominant.

MATALB®modeling of all the charge-domain operations in CRAFT utilizes both of these simulation methods for all charge-domain operations in the system. Iterative evaluation the differential equations is the primary technique while, in the case of 2-point share operations, iterative exponential method yields equally accurate results with a speed advantage. Higher-order terms, such as the first and second derivative of operand node voltages and derivatives of the switch resistances, are used to increase simulation accuracy and reduce the number of iterations (time steps) required.

Mitigation It is important to ensure operand-dependent switch resistance does not significantly degrade computational error (dynamic range) relative to the ideal incomplete-settling design targets. Therefore, CRAFT simulates this effect in all charge-domain operations to validate the system's dynamic range.

Separately, simulations of individual charge-domain operations were used to compare the nominal settling desired ($n = t_s/\tau$) with the amount achieved with switch resistance variation (mean and variation of $t_s/\tau_{\text{effective}}$) to ensure tha the desired compuation accuracy bound is not exceeded. This also allowed the quantitative evaluation of the benefit of RCX error cancellation. Additionally, the effect of trade-offs in switch voltage headroom (V_{dd}, V_{cm}, and signal swing) upon dynamic range and switch energy consumption were compared and optimized. All of this was performed without the need for computation phase times (t_s) or switch sizes to be determined since the settling timescale was normalized by $R_{sw,\text{nom}}C$, and settling goals were expressed in a number-of-time-constants (n) format.

Details of the calculation of effective settling-constants and how it was linked to incomplete settling without voltage dependence is covered in Sec. 4.6. This allows the analysis and notation of Eqs. (6.10), (6.14), and (6.15) to be retained while still keeping the familiar $n{\cdot}\tau$ settling form as seen in Eq. (4.33).

6.4.5 Clock Feedthrough

Motivation Clock signals used to toggle the CRAFT core switches feeds through the nMOS-switch overlap capacitances to the operand and stealing capacitors, C or C_s, causing a charge-domain error on the operand values. Layout parasitic capacitance provides an additional path between clock routing and operand nodes and therefore should be matched for an equal effect (common-mode in a differential design).

Each charge-domain operation consists of one rising and one falling edge applied to

the switches. In the case of complete settling, share operations (with equal operand capacitors C) generate equal and opposite disturbances ($+\Delta$ and $-\Delta$) on the operands as the switch closes and opens. This common-mode shift and subsequent restoration yields no net effect on the charge-domain operation.

However, for share and multiply operations, the operand capacitors C are not necessarily equal to the stealing capacitor C_s causing them to suffer unequal disturbances (say, Δ and Δ_s respectively) during a switching event's edges. The rising-edge clock-feedthrough charge is shared during the operation phase. However, when the switches are opened, a $-\Delta_s$ falling-edge clock feedthrough is placed on the stealing capacitor, which becomes disconnected and cannot redistribute. This means that not all of the charge added to the computation during the rising-edge feedthrough can be removed from the outputs, as in a share operation.

Any residual voltage step is a clock feedthrough based computation error. Its magnitude depends on capacitor sizes, the operation's multiplication factor m (determining C_s), the switch configuration, and the switch size. Analysis of this error source is covered in Sec. 4.2.

Modeling A MATLAB®model of the CRAFT clock feedthrough is utilized as follows. A rising-edge feedthrough voltage step is applied to input operands and stealing capacitors. The perturbed voltages then perform the charge-domain computation. Another falling-edge feedthrough voltage step is then applied. The magnitude of each voltage step is a function of the operand's implementation, switch sizes, node capacitances, and the supply voltage.

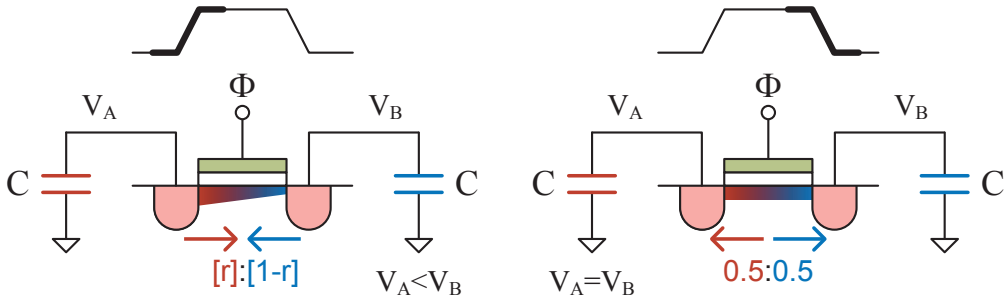
Mitigation Clock feedthrough is minimized by adding half-dummy switches. This comes at the cost of doubling switching power and requiring differential clocks. Determining when this expense is necessary is a critical optimization step which requires the ability to isolate it as an error source in specific processing stages to analyze the effect on the total error. The capabilities of our MATLAB®modeling allow this simulation and evaluation technique. To keep clock feedthrough effects below the required SNDR, we determined that half-dummy switches are only necessary in stage 3 of the CRAFT implementation.

6.4.6 Charge Injection and Absorption

Motivation During passive switched-capacitor operations, turning on MOS switches causes charge absorption from the source and drain nodes as device channels are formed. Similarly, turning off the MOS switches causes this channel charge to be injected back these nodes. This causes perturbations of the voltages held at these high-impedance nodes. Analysis of this error source is covered in Sec. 4.3.

The charge injection error depends on the impedance on each terminal, the clock transition time relative to the channel transient time constant, and the node time constants, as discussed in [20]. For charge absorption, in addition to the the factors mentioned, the operand node voltages play an important part as shown in Fig. 6.17(a). Due to unequal initial node voltages, the switch spends some of it's turn-on transition time forming a channel in the saturation region of operation and then continues into the triode operating regime. As a result of the sharing operation, the switch being on for a while, this unequal charge absorption error is distributed among the capacitors causing a constant error. Due to the high speed of operation in CRAFT, incomplete settling allows some residual charge absorption error to be present. However, this small error is attenuated by the factor $e^{-t_s/\tau}$. As a result, this error only needs to be modeled if the

highest level dynamic range performance is required.



(a) Absorption (saturation & triode) (b) Injection (deep triode)

Figure 6.17: Charge absorption (a) and charge injection (b) in a share operation

Charge injection, as shown in Fig. 6.17(b), occurs at the end of the share operation and causes impedance and fall-time dependent errors on the final computation values. It applies an equal voltage-step disturbance to all the output operand copies (under the assumption that the operation's implementation has this desireable feature). For the pseudo-differential representation used the positive and negative components have different charge-injection magnitudes because different voltages across their channel capacitance modify the quantity of charge injected. This different step error manifests as both a common-mode level shift, and a differential error. For small signal swings the error is primarily common-mode. However, as the swing increases its differential fraction grows because of the stronger voltage-dependence. This the is the primary error mechanism for charge-injection in a differential system and necessitates its modeling.

Modeling For modeling the charge injection and absorption, the clock edge is assumed to be much slower than the minority carrier time constants. For charge injection, an equal split is assumed at the end of the sharing operation (equal voltages and equal impedances on both nodes) as shown in Fig. 6.17(b). For the share and multiply operation, the model developed in [20] is utilized. For charge absorption, the effect of

the instantaneous node voltages is included in the model. This error is then distributed as the charge-domain operation settles, and the remnant of this error due to incomplete settling is included in the output values.

Mitigation Like clock feedthrough, half-dummies are an effective way of reducing the effects of channel charge[11]. However, apart from doubling the power consumption, their primary limitation is that they work best only when the charge-split is even. This occurs only for share operations with small input operand swings. Attempting to adjust the dummy weighting to compensate for unequal charge split due to unequal impedances (stealing capacitor) would reduce the effectiveness of the clock-feedthrough cancellation effect they provide.

To minimize the effect of the residual charge absorption disturbance, improving t_s/τ without increasing the switch channel charge Q_{ch} is desired. Without changing τ, the computations settling time constant (accuracy), this can be accomplished by: increasing t_s (the settling time), decreasing switch length, or choosing a V_{dd} for a minimum C_{ch}/R_{sw}, as described in Sec. 6.5. This improvement also benefits the magnitude of the charge-injection effect at the end of the computation.

6.5 Design Methodology and Optimization

For the CRAFT processing engine, computational speed, dynamic range, and operating power trade-off with each other. The analysis of design non-idealities, discussed in Sec. 6.4, represents a complex design space with different trade-offs associated with each error source and the particular mitigation techniques utilized. This section outlines a design and optimization methodology used in the CRAFT design to achieve superior

[11]Note, that in passive charge-domain operations, switch channel charge must be handled without the advantages available to traditional switched-capacitor circuits such as virtual grounds or low impedance nodes.

performance. For this implementation, the following specifications and constraints were assumed:

1. A 5GS/s input rate with an interleave-by-two CRAFT engine for processing contiguous windows: This provides a total processing time of $16 \times \frac{1}{5\text{GHz}} = 3.2\text{ns}$.

2. A 60dB (10bit) dynamic range is chosen as a target specification.

Using these goals, the CRAFT engine is optimized for processing power. The design methodology is divided into an architecture choice based on the constraints listed, followed by an energy optimization procedure.

6.5.1 Architecture Choice

For the architecture chosen the processing time (16 sample periods for interleave by two) is initially assumed to be shared equally among the 5 clock phases. This assumption is revisited during the energy optimization procedure described later. Also, a nominal V_{dd} for all the stages is assumed as an initial choise and is optimized later. Based on these assumptions, the following design choices are made:

1. *Input swing:* The maximum input swing, $V_{sig,\max}$ for the sampler is chosen to achieve -60dBFS non-linearity while running at 5GS/s. This determines the peak-to-peak input swing to be used as the input full scale. For use with a nMOS switch based processing core the common mode voltage, V_{cm}, is set at $\frac{1}{2}V_{sig,\max}$.

2. *Capacitor size:* The sampling capacitor size is selected such that the noise floor from the sampling operation is at least 10dB lower than required for the target SNDR of 60dB. This dictates a sampling capacitor size of 200fF.

3. *Attenuation:* Attenuation degrades FFT performance. Fortunately, some techniques to minimize the attenuation, as discussed in Sec. 6.3, are available. Consequently, all techniques that mitigate attenuation are incorporated for improved performance.

4. *Dummy switches:* The effect of clock feedthrough and charge injection for each stage on the overall SNDR is simulated. Dummy switches are selected for stages where the overall SNDR is otherwise not met.

5. *Settling optimizing switches:* The overall computational settling error is simulated, and settling optimization switches are chosen for stages where their effect is overtly beneficial to SNDR performance.

6. *Sampler switch size:* The minimum switch size that provides adequate sampler settling and non-linearity for the required SNDR is determined.

7. *Settling:* The minimum per-stage computation settling accuracy required for the overall SNDR is selected. For CRAFT, the following amounts of nominal computational settling were chosen for stages 1–4: 7τ, 4τ, 5τ, and 4τ, respectively.

6.5.2 Energy Optimization

The exact switch sizing in each stage, as well as the V_{dd} employed, trade-off with the total energy consumption per processing operation. The energy optimization algorithm employed is outlined below.

1. *Supply voltage:* In short channel devices velocity saturation affects nMOS switches. The triode resistance of the switch is approximated in deep triode[12] as:

[12]For the devices used in CRAFT, $p = 0.50$ provided an accurate empirical fit.

$$R_{\text{triode}} \propto \left(\frac{1+U_0 V_{ov}}{V_{ov}}\right) \approx \left(\frac{1}{V_{ov}}\right)^p$$

Using this approximation, the switch resistance is $R_{sw} \propto W^{-1}\,(V_{ov})^{-p}$. For a constant R_{sw}, $W \propto (V_{ov})^{-p} = (V_{dd} - (V_{tn} + V_{cm}))^{-p}$. In order to calculate the energy per switch operation, we compute $\frac{1}{2}C_{gs}V_{dd}^2 \propto W \cdot V_{dd}^2$. Using these equations, we compute the energy per switch operation for a constant switch on-resistance:

$$E_{sw}\Big|_{\text{const.}R_{sw}} \propto \frac{V_{dd}^2}{(V_{dd} - (V_{tn} + V_{cm}))^p} \tag{6.16}$$

This is plotted in Fig. 6.18. As seen, this curve has a unique minimum energy. The minimum occurs at $V_{dd,\text{opt}} = \frac{2}{2-p}\,(V_{tn} + V_{cm})$. Naturally, this minimum coincides with the typical supply voltage in this technology to optimize digital energy per speed (consider this nMOS switch as part of an inverter). This optimum V_{dd} is then customized per stage depending on the varying operand voltage swings as a result of attenuation[13]. Additionally, if alternate sample rates or time allocations may be used, different V_{dd} settings allow separate optimization in those modes while implemented switch sizes remain fixed.

2. *Switch size:* For calculating the optimal switch width note, that each V_{dd} corresponds to a particular switch width (for a given resistance) on the constant-resistance plot. The maximum allowable nominal resistance can be calculated based on the required settling and allocated time chosen in Sec. 6.5.1. Therefore, from the V_{dd} chosen above, and the maximum allowable resistance, the optimal switch width, W, for each stage, is calculated. The effects of wiring resistance across the processing core should be considered.

[13]For the optimization described, it is assumed that 2 different V_{dd}s are available for optimization to cover a general case. The optimization algorithm can be easily modified for the specific case using a single or multiple V_{dd}s, based on availabilitiy.

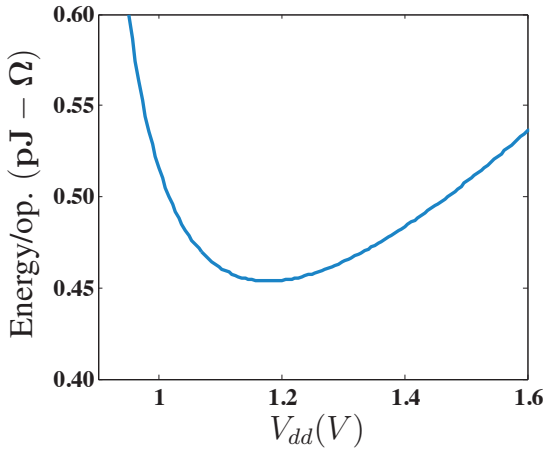


Figure 6.18: Optimization of switch-energy per operation (contant R_{sw})

3. *Time allocation:* Note that the energy per stage is dependent on the required switch resistance, and consequently, the time allocated per stage. The per-stage time allocated is now considered as the last optimization variable, and is re-distributed (instead of the equal distribution assumed earlier) to optimize the total energy further. For the new allocated times, new optimal switch widths are determined. This step is effective because stages have varying relative computational complexity and are implemented using different types of charge-domain operations. Therefore, some stages require a larger number of switches or particularly low switch resistance for computational accuracy.

   For CRAFT, the following processing phase times, as shown in Fig. 6.12, were chosen to minimize power: $3T_s$, $5T_s$, $4T_s$, $2T_s$, and $2T_s$ (Φ_1, Φ_2, Φ_{3A}, Φ_{3B}, and Φ_4), where T_s is the sampling clock period (1/5GHz) and the total processing duration is $16T_s$ for an interleave-by-two implementation.

4. *Half-dummies:* Finally, the use of half-dummies in specific stages is revisited. Although unlikely, it might be possible to achieve the required SNDR by providing more time (reducing switch width and the resulting clock feedthrough and charge

injection) to the SNDR-limiting stage instead of adding half-dummies. While this may require other stages to have shorter duration (larger switches and more energy consumption), the potential of removing half-dummies would cut this stage's power by 2. This possibility is also verified before finalizing the design.

6.5.3 LTI Model

In order to perform high-level simulations, the CRAFT engine, followed by ADCs for digitization, can be approximated using a linear time invariant model as shown in Fig. 6.19. Noise disturbances as well as systematic mismatch variation followed by digital correction can be included in this model. This is useful for evaluating system level impacts of noise and matching. Furthermore, the impact of other high-level design tradeoffs, like

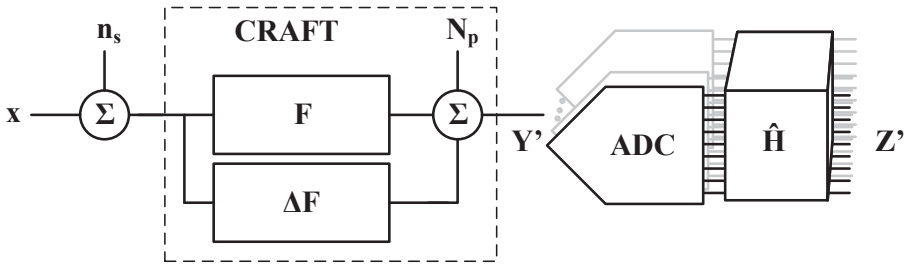


Figure 6.19: An LTI model of the CRAFT operation with digitization and correction

incomplete settling, can be included if linearized (i.e. without operand voltage dependence). Non-linear error components or effects can be approximated and included using random variables with mean and variance determined from separate simulations.

A brief description of the components shown in Fig. 6.19 are tabulated in Table 6.2. Further details can be found in Appendix B. Note that effects such as: settling error, charge injection and absorption, and clock feedthrough are not included in this model.

Table 6.2: Summary of variables in LTI model

Variable	Description	Expression
$\mathbf{x}$	Input (16 samples)	$x(i), \forall\ i = [0, 15], i \in \mathbb{Z}$
$\mathbf{n}_s$	Sampler noise (pseudo-differential)	$\sigma^2_{n_{s,i}} = 2k\mathrm{T}/C$ C = capacitance/copy
$\mathbf{F}$	Ideal FFT	$\mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1$, see Appendix A
$\mathbf{\Delta F}$	Linear FFT implementation error	See Appendix B
$\mathbf{F}'$	FFT approximation	$\mathbf{F}+\mathbf{\Delta F}$, see Appendix B
$\mathbf{N}'_\mathbf{p}$	Equiv. processing + reset noise at CRAFT output	See Appendix B
$\mathbf{N_q}$	Equiv. quantization noise at ADC input	See Appendix B
$\hat{\mathbf{H}}$	Correction matrix	$\mathbf{F}(\hat{\mathbf{F}}')^{-\mathbf{1}}$, see Sec. 6.6.1
$\mathbf{Y}'$	Digitized output	$(\mathbf{F}+\mathbf{\Delta F})\,(\mathbf{x}+\mathbf{n_s})+\mathbf{N}'_\mathbf{p}+\mathbf{N_q}$, see Appendix B
$\mathbf{Z}'$	Digitized and corrected output	$\hat{\mathbf{H}}\,\mathbf{Y}'=\hat{\mathbf{X}}+\hat{\mathbf{N}}_\mathbf{s}+\hat{\mathbf{N}}_\mathbf{p}+\hat{\mathbf{H}}\,\mathbf{N_q}$, see Appendix B

6.6 Measurement Results

The design has been implemented in the IBM 65nm CMOS process. Measurement results are shown in Fig. 6.22−6.28. The measurements shown are after correction to compensate for systematic offsets due to parasitics. The calibration process used, and possible improvement are detailed below.

6.6.1 Calibration

The accuracy of the CRAFT operations are dependent on the matching between the capacitors used to realize them as discussed in Sec. 4.4. Any systematic mismatch between the capacitors causes computation errors that reduce the dynamic range of the system relative to its intended linear operation. However, since these errors are systematic and input independent, it is possible to calibrate out the errors after conversion to digital. This is because any improper charge-domain computation due to mismatched operand

weights (capacitors) is still a linear operation. For our measurements, we use a simple calibration technique described below.

First, the non-idealities in the CRAFT operation are represented using a modified FFT matrix, $\mathbf{F'}$, that includes the effects of mismatch and represents the non-ideal CRAFT linear transform (see Eq. (6.4)). The resultant outputs are thus represented by $\vec{\mathbf{X}}'$ as below (compare with Eq. (6.2)).

$$\vec{\mathbf{X}}' = \frac{1}{k}\,\mathbf{F}'\vec{\mathbf{x}} \tag{6.17}$$

A simple calibration method is determined by observing that the 16 point FFT matrix, $\mathbf{F}$, contains of 256 elements. The application of $\mathbf{F}$ to a set of 16 inputs samples provides a set of 16 outputs corresponding to 16 linear equations of a DFT. Consequently, 16 mutually independent sets of inputs: vectors $\vec{\mathbf{x}}_\mathbf{i}$, with entries $x_i(t)$ will form 256 independent equations. Therefore, using the sets of measured results, $\vec{\mathbf{X}}'_\mathbf{i}$, with entries $X'_i(k)$, all 256 elements of the non-ideal $\mathbf{F'}$ matrix can be determined when all input and output vectors are composed as shown below.

$$\begin{aligned}\left[\vec{\mathbf{X}}'_\mathbf{0} \quad \vec{\mathbf{X}}'_\mathbf{1} \quad \ldots \quad \vec{\mathbf{X}}'_\mathbf{15}\right] &= \frac{1}{k}\,\mathbf{F}'\left[\vec{\mathbf{x}}'_\mathbf{0} \quad \vec{\mathbf{x}}'_\mathbf{1} \quad \ldots \quad \vec{\mathbf{x}}'_\mathbf{15}\right] \\ \mathbf{X}'_{cal} &= \frac{1}{k}\,\mathbf{F}'\hat{\mathbf{x}}_{cal}\end{aligned} \tag{6.18}$$

For convenience, we generate orthogonal inputs based on separate one-tone signals. Note that in a fully integrated implementation, perfect tones are not easily available on-chip. However, the sampling array can be preloaded with samples of linearly independent (not necessarily orthogonal) inputs that would only require a low resolution DAC and can be used instead to provide similar calibration accuracy.

To generate the linearly independent inputs, 15 separate single-tone inputs, each centered on a non-DC bin i, are applied to the system. Due to the AC coupled nature

of the test setup, the 16th independent input stimulating the DC bin cannot be used. Rather than introduce an input at the sampling rate to alias to the DC bin an ideal DC response is assumed. This calibration data set is applied to Eq. (6.18) as shown below, where $x_i(t)$ is the time-domain input for a pure I/Q tone placed on the i-th bin, and $X'_i(k)$ is its frequency-domain response as output by CRAFT.

$$\begin{bmatrix} 1 & X'_1(0) & X'_2(0) & \dots & X'_{15}(0) \\ 0 & X'_1(1) & X'_2(0) & \dots & X'_{15}(1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & X'_1(15) & X'_2(15) & \dots & X'_{15}(15) \end{bmatrix} = \frac{1}{k}\,\mathbf{F'} \begin{bmatrix} 1 & x_1(0) & x_2(0) & \dots & x_{15}(0) \\ 1 & x_1(1) & x_2(0) & \dots & x_{15}(1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1(15) & x_2(15) & \dots & x_{15}(15) \end{bmatrix} \tag{6.19}$$

The calibration inputs levels are at -6dBFS (dB of full-scale) so as to reduce circuit non-linearity effects introduced during calibration. This 6dB reduction in input amplitude suppresses the sampler's non-linearity (primarily 3rd order) by an additional 12dB to reduce its influence on the procedure. Also, it is important to minimize processing non-linearity specific to the calibration input vectors from being included in the computed estimate of $\mathbf{F'}$. Additional phase diversity of the input vectors among many measurement runs is used to further spread and average processing non-linearity. The phase of the peak on-bin term is used to align and synchronously (phase-aligned in post processing) average 512 measurement runs, for each $\mathbf{X'_i}$, to reduce noise.

Since the sampled input values on-chip cannot be known, the ideal time-domain input calibration vectors, $\vec{\mathbf{x}}_\mathbf{i}$, are estimated based on the magnitude and phase of the measurement bin peak $X'_i(i)$. This is appropriate because spurious content in the uncorrected CRAFT outputs appears on bins other than the on-bin input. As uncorrected on-bin outputs are assumed correct to first order, their value provides useful information about the amplitude and phase modification the sampler has applied to the expected input. Additionally, since the measured (through the CRAFT transform) phase and

magnitude of the on-bin terms are used, the attenuation caused by the FFT implementation (k term of non-ideal FFT: $\vec{\mathbf{X}}' = \frac{1}{k}\,\mathbf{F}'\vec{\mathbf{x}}$) is incorporated. Finally, an oscilloscope was used to verify that the AWG outputs as supplied to the probes had amplitude flatness and matching delay over frequency. It is in this manner that the sampler frequency response and residual test setup effects are removed or incorporated into the correction.

After determining an estimate of $\mathbf{F}'$ using the calibration data sets, a one-time correction matrix, $\hat{\mathbf{H}}$ is computed as shown below:

$$\begin{aligned} \hat{\mathbf{F}}' = \mathbf{X}'_{cal}\,\hat{\mathbf{x}}_{cal}^{-1} \quad \Rightarrow \quad \hat{\mathbf{H}} &= \mathbf{F}(\hat{\mathbf{F}}')^{-1} \\ &= \mathbf{F}\,\hat{\mathbf{x}}_{cal}\,(\mathbf{X}'_{cal})^{-1} \end{aligned} \tag{6.20}$$

where $\hat{\mathbf{x}}_{cal}$ and $\mathbf{X}'_{cal}$ are the calibration input and output matrices composed in Eq. (6.18). All subsequent measurements across different magnitude and frequency inputs are then corrected by applying $\hat{\mathbf{H}}$ to CRAFT measurements, $\vec{\mathbf{X}}'$:

$$\begin{aligned} \hat{\mathbf{H}}\,\vec{\mathbf{X}}' \quad \Rightarrow \quad \hat{\vec{\mathbf{X}}} &= \hat{\mathbf{H}}\,(\mathbf{F}'\,\vec{\mathbf{x}}) \\ \hat{\vec{\mathbf{X}}} &\approx \mathbf{F}\,\vec{\mathbf{x}} \end{aligned} \tag{6.21}$$

Improved calibration

While the calibration technique used was performed on measurements offline, real-time correction in the digital domain does not require repeated inversion of a 16×16 matrix. Since the correction is for systematic linear computation errors in the charge-domain implementation it only needs to be determined once. In cases requiring a level of performance where variation in systematic error may become important (i.e. with temperature) it would still only periodically need to be recomputed.

It is clear from the implementation details that a number of butterflies are identical (see Fig. 6.7), and are therefore well matched to each other. In fact, a detailed analysis

reveals, that if the sampling capacitors are well-matched, the total number of coefficients in the FFT matrix of equation (6.4) that are in error is only 10 rather than all 256 elements in **F**. This makes digital correction a viable option with minimal power overhead since even a full-complexity 16×16 matrix multiply is not necessary. The digital correction can also be absorbed into the first stage of DSP that follows the ADC to further reduce the calibration overhead. For example, if an IFFT is used to reconstruct time-domain samples from the CRAFT outputs, the corrections can be incorporated by modifying some of the digital butterfly operations to invert the imperfect operations performed in the charge-domain.

6.6.2 Test Setup

The test setup is shown in Fig. 6.20. Differential I and Q inputs are generated using the Tektronix AWG-7122B arbitrary waveform generator. Low-pass filters of 7th order with a cutoff frequency of 5.5GHz are used on the AWG outputs to perform reconstruction and remove sampling image tones. The inputs are probed into the chip using 50Ω GSSG probes. On-chip resistive terminations combine with the effective switching load of the sampling capacitors ($R_{\text{eff}} = 1/(f_s C) \approx 250\Omega$) to provide the input matching. The latched differential outputs are selected with an analog mux and buffered off-chip before being captured by external ADCs controlled by an FPGA programmed using LabVIEW®.

Note that the CRAFT processor runs at an input/output rate of 5GS/s per I and Q channel. Additionally, the outputs are analog values held on capacitors. Due to the very high speed of operation, the limitations in the number of I/O pins, and the high-impedance nature of the outputs they are first latched using the OTA-based analog latches described in Sec. 6.3.3, and multiplexed out at a slower rate limited by the external ADC. The CRAFT processing speed is not compromised due to the output

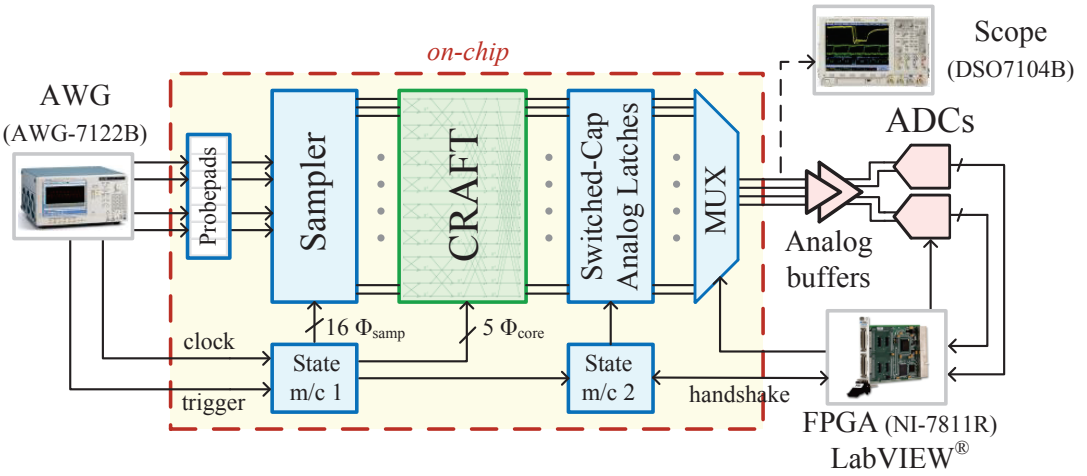
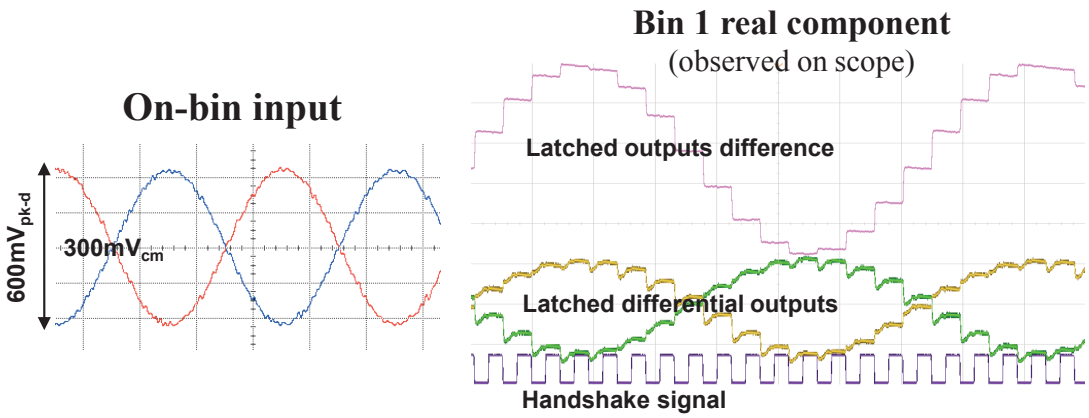


Figure 6.20: External test setup and supporting circuits for the CRAFT processor

read-out limitation. This output multiplexing and read-out is performed by an FPGA (NI-7811R) programmed using LabVIEW®. The outputs are first buffered using discrete high-speed low-noise differential instrumentation amplifiers, then digitized by two external 16-bit ADCs (AD7621) interfaced with the FPGA. The entire output data chain was designed to have the lowest possible non-ideality floor and has a noise level of -73dB relative to the full-scale output coming from the chip.

Using this scheme, RF inputs that are synchronous as well as asynchronous to the clock, can be captured and phase aligned. The latter is particularly important to emulate practical scenarios while still allowing noise averaging to be performed.

The time-domain input and output characteristics, as observed by the oscillocope (Agilent DSO7104B) shown in Fig. 6.20, for a signal on the first bin is shown in Fig. 6.21. The input signal is sampled using a progressively shifting phase ($2\pi/16$) upon every FFT conversion causing the bin 1 output to rotate periodically as shown in the figure.



(a) Example on-bin input

(b) Output (rotation due to asynchronous triggering)

Figure 6.21: Time-domain latched CRAFT outputs, single-bin input, with fractional triggering rate for periodic phase rotation of output

6.6.3 On-bin 1-tone Measurements

Fig. 6.22 shows the CRAFT output magnitudes ($\sqrt{\Re^2 + \Im^2}$) with a 312.5MHz ($= \frac{5\text{GHz}}{16}$) single-tone input at 5GS/s. Curve I shows the measured uncalibrated output magnitude across all 16 bins. Curve II in Fig. 6.22 shows the calibrated plot depicting the circuit RMS noise floor (including the integrated noise of the analog latches) at approximately -46dB. To explore the non-linearity floor, a synchronous average of 512 measurements is used. The resultant Curve III shows the non-linearity floor with 43dB SFDR[14] and 48dB SNDR[15]. Note that the SNDR and SFDR also signify the out-of-band rejection of the FFT as a filter for on-bin (orthogonal) signals while, as a general filter, it has the

[14]SFDR for a one-tone test is calculated as the difference between a full-scale on-bin signal and the largest off-bin output caused by the CRAFT non-linearity/calibration errors.

[15]SNDR is calculated as follows:

$$SNDR = 20 \times \log_{10}\left(\frac{\sqrt{\sum_{k=1}^{N} V_{\text{ideal}}^2(k)}}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\{V_{\text{meas}}(k) - V_{\text{ideal}}(k)\}^2}}\right) \tag{6.22}$$

where N is the number of FFT bins. Since the outputs are observed/digitized on a per bin basis, the total noise + distortion (N+D) in the denominator is averaged over N bins to yield the average per bin.

expected *sinc* filter response due to the sampled inputs being provided to CRAFT without windowing. The nonlinearity predicted by simulations of the standalone CRAFT engine is shown in Curve IV. Note that Curve III not only includes the non-idealities in CRAFT, but, unlike Curve IV, also includes the non-idealities of the 8-bit resolution AWG inputs, the sampler non-linearity, and systematic and random sampling jitter. Unfortunately, despite the use of state-of-the-art test equipment, the limitations in the input and output test setup severely limit the observable non-idealities in CRAFT.

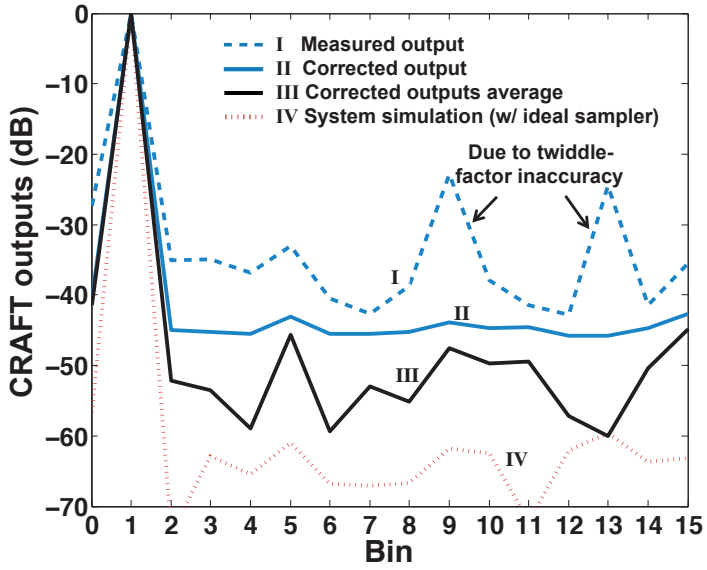


Figure 6.22: CRAFT outputs with a single-tone 362.5MHz *I/Q input at 5GS/s*

Ideally, all bins except bin 1 should be at the noise floor, but due to systematic inaccuracies in parasitic extraction, significant uncompensated interconnect capacitances on the stealing capacitors of this CRAFT version resulted in FFT twiddle factor errors. This causes systematic input-independent computation errors of a few butterflies to be larger than expected under corrected verification practices. These mathematical errors propagate to the outputs and take the appearance of spurious tones. A design revision would improve the uncorrected measurements outputs (spurious tone magnitude)

by approximately an order of magnitude and bring the accuracy of the system's linear transform in line with expectations based on the charge-domain matching (capacitance) achieved. Digital correction would further improve the transform's accuracy to the extent that it is limited by noise and non-linearity. All subsequent measurement results use outputs after off-line digital correction by the method shown in Sec. 6.6.1

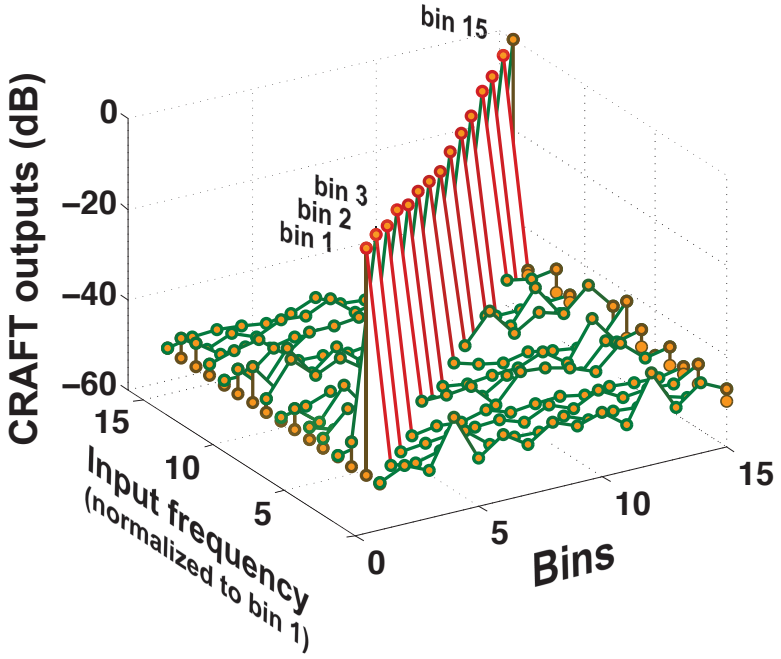
SNDR Variation

One-tone measurements for each FFT bin were performed as shown in Fig. 6.23(a) and the resulting SNDR at 1, 3, and 5 GS/s for a one-tone input frequency placed at each bin is shown in Fig. 6.23(b). The average SNDR across bins is $\approx$ 50dB at 1 & 3 GHz and degrades to 47dB at 5GHz while SNDR better than 45dB is maintained across all frequencies. The SNDR and SFDR measurements are tabulated in Table 6.3. Theses results show that CRAFT provides 7–8 bits of spectrum sensing resolution in the digital back-end over a 5GHz (2.5GHz$\times$2 due to *I, Q* inputs) frequency range.

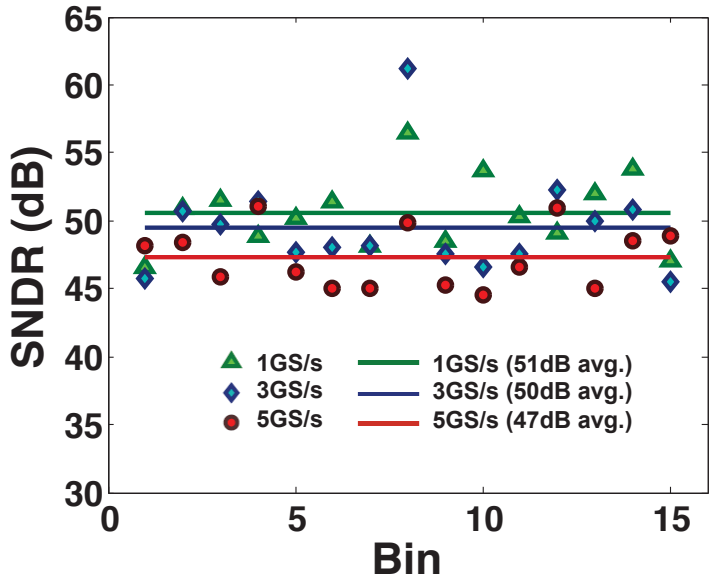
The CRAFT output magnitudes with varying input amplitudes for a single-tone, on-bin 312.5MHz input at 5GS/s are shown in Fig. 6.24. The input is varied over a 18dB range. As seen, the circuit is limited by the noise floor for low input amplitudes, while it becomes non-linearity limited at larger input amplitudes. Note that Curve III in Fig. 6.22 represents a cross-section (one specific input amplitude) of Fig. 6.24.

Fig. 6.25 plots the output SNDR versus the input amplitude. A fourth-order fit shows a linear SNDR improvement with increasing amplitude before being limited by a combination of the processing core non-linearities, sampler non-linearity, and the AWG resolution (8 bits). While $-$6dBFS inputs were used for calibration there is still 1–2 dB SNDR improvement as the input amplitude increases before the combined non-linearity of previously mentioned sources begin to dominate.

Fig. 6.26 plots the measured noise plus distortion (non-linearity) floor versus the



(a) Single-tone tests of all 15 non-DC bins



(b) SNDR vs. single-tone bin at 3 different sampling rates

Figure 6.23: Measurement results: 1-tone measurements (a), and SNDR results across bins and operating rates (b)

Table 6.3: Table of CRAFT SNDR and SFDR with 1-tone, on-bin, 0dBFS inputs

1-tone bin	**1GS/s**		**3GS/s**		**5GS/s**	
	SNDR	**SFDR**	**SNDR**	**SFDR**	**SNDR**	**SFDR**
1	46.6	37.7	45.8	35.8	48.1	42.6
2	50.9	42.8	50.7	43.8	48.4	39.7
3	51.5	44.7	49.8	44.8	45.8	41.8
4	48.9	37.9	51.4	40.2	51.1	41.1
5	50.2	44.4	47.7	40.7	46.2	42.5
6	51.4	44.8	48.0	40.1	45.0	36.3
7	48.1	40.2	48.2	42.6	45.1	40.1
8	56.5	51.2	61.2	57.2	49.9	42.8
9	48.5	41.3	47.5	41.2	45.2	39.6
10	53.7	47.8	46.6	39.6	44.5	35.6
11	50.3	44.9	47.6	40.0	46.6	43.3
12	49.2	37.9	52.2	41.4	50.9	41.6
13	52.0	46.2	50.0	45.5	45.0	39.9
14	53.8	46.6	50.9	43.0	48.5	38.8
15	47.1	39.0	45.4	35.9	48.9	42.3
Min	46.6	37.7	45.4	35.8	44.5	35.6
Max	56.5	47.8	61.2	45.5	51.1	43.3
Mean	**50.6**	**43.2**	**49.5**	**42.1**	**47.3**	**40.5**
Sigma	**2.7**	**4.1**	**3.8**	**5.0**	**2.3**	**2.3**

input amplitude. In the figure, a level below which noise dominates is evident followed by a steady increase due to sampling and processing non-linearity. Since it is an absolute measurement, rather than referenced to the input signal level like SNDR, it is useful for analyzing the SNDR results. It is calculated from the measurments here as: **N+D** (dBFS) = **Input** (dBFS) − **SNDR** (dB). Without the use of external attenuators, the AWG output resolution became a limitation when generating low input amplitudes. This causes the large variation at the noise floor as well as the limitation to −35dBFS. If the input signal range were extendend lower, it is expected that SNDR would cross zero at approximately −53dBFS (extrapolating Fig. 6.25)

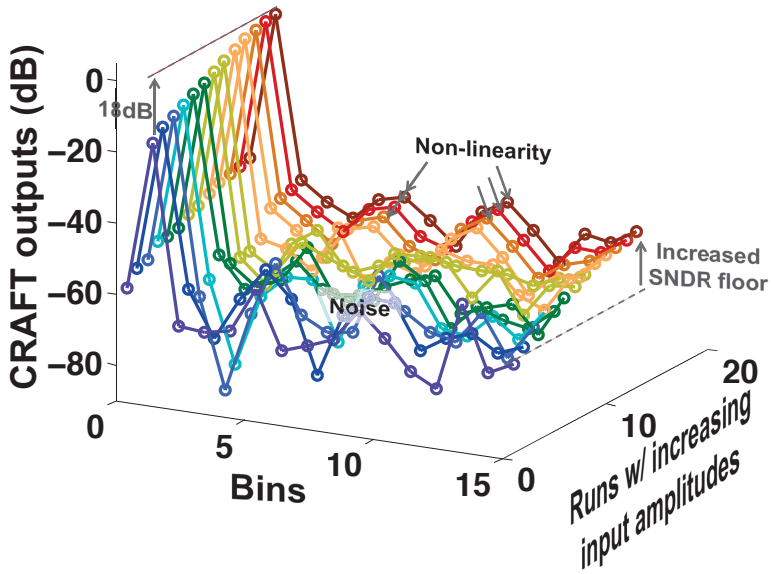


Figure 6.24: CRAFT noise and non-linearity with increasing input amplitude at 5GS/s

6.6.4 On-bin 2-tone Measurements

Results from a two-tone test with tones on adjacent bins are shown in Fig. 6.27. Two separate full-scale (0dBFS) single-tone measurements are plotted. Assuming a preceding AGC, the peak (time-domain) amplitude of the two-tone input signal is normalized to that of a single tone in the 1-tone tests as shown. The two one-tone tests are superposed and scaled in amplitude to match that of the two-tone measurement result. The difference between the superposition and the measured two-tone output is indicative of the computation non-linearity introduced.

The relative increase in bins 13 and 14 is likely due to imperfectly corrected twiddle factor errors in stage 2 of the CRAFT engine. For signals on bins 1 and 2 non-ideal computational results in specific butterflies of stage 2 would propagate errors to their conjugate frequency pairs. Note that the mathematical effect of these twiddle factor errors, leaking signal on to the negative of the signal frequency, is identical to the effect of I-Q mismatch errors in a homodyne receiver.

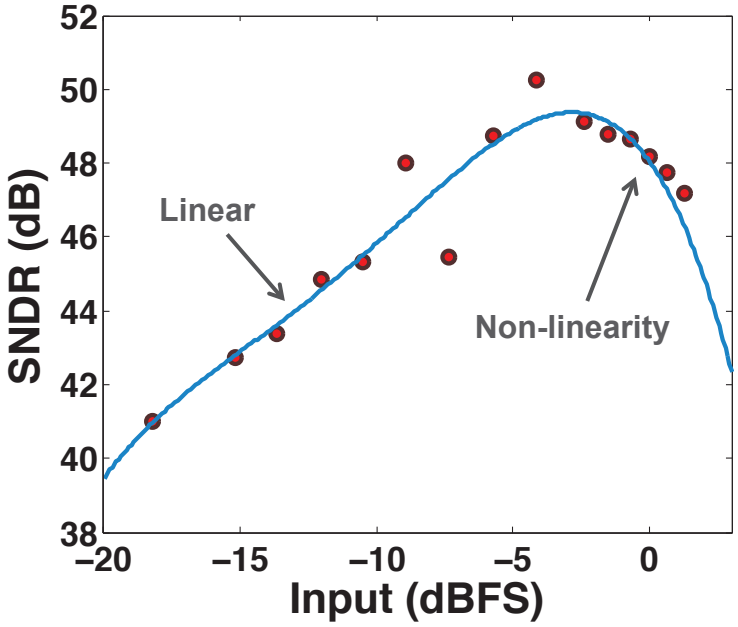


Figure 6.25: Single-tone SNDR vs. input amplitude at 5GS/s

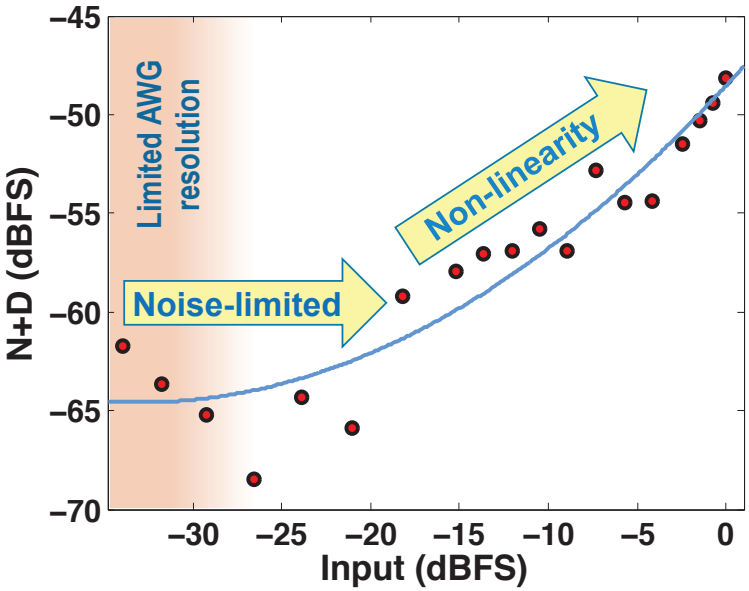


Figure 6.26: Single-tone noise plus distortion (N+D) vs. input amplitude at 5GS/s

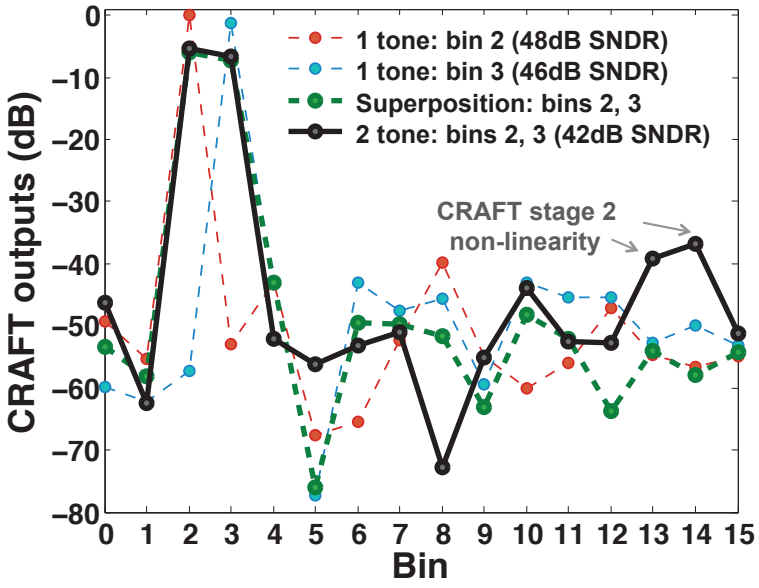


Figure 6.27: A two-tone non-linearity test of CRAFT

6.6.5 Power Consumption

The CRAFT core consumes 12.2pJ/conv. and uses 3.8mW of power when interleaved by two for a 5GS/s input and 5GS/s aggregate output. Measurements of the energy consumption versus supply voltage and frequency are depicted in Fig. 6.28. These measurements clearly show the expected digital-like relationship of the CRAFT energy with frequency and supply voltage. This further corroborates our premise that CRAFT is expected to respond favorably to technology scaling.

Updated operating and implementation techniques have been identified to reduce power consumption of this design approximately in half. This is primarily accomplished by gating some processing clocks during the core reset operation, and reducing local and global clock routing parasitic capacitance.

A die photograph of the CRAFT chip is shown in Fig. 6.29. The 256 capacitor sampling array plus supporting digital state machines consume only 0.13mm^2. This

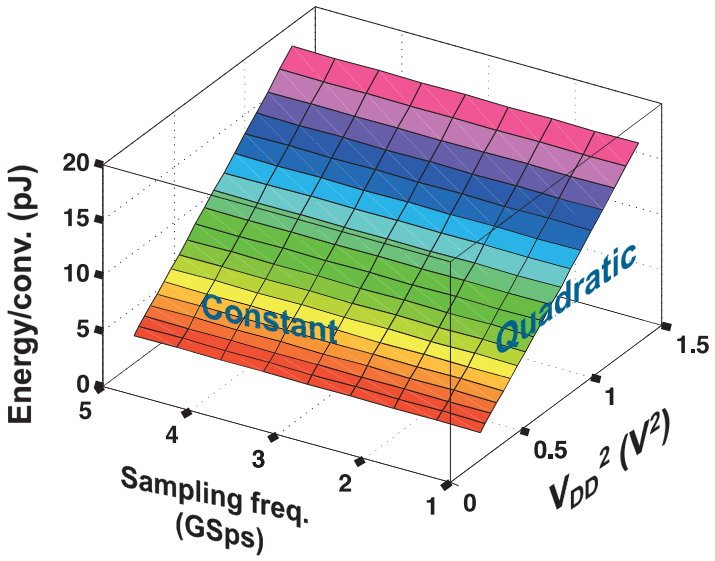


Figure 6.28: Measurements: CRAFT's digital-like energy relation with f_s and V_{dd}

area is a few times larger than that of a sampling array for an interleave-by-16 time-interleave ADC design due to the CRAFT core's need for operand copies. Nevertheless, it is still relatively small, even when combined with the CRAFT processing core which occupies an area of only $300\mu\text{m} \times 480\mu\text{m} = 0.144\text{mm}^2$ as shown.

6.7 Conclusion

This chapter describes a wideband ultra-low power RF front-end channelizer based on a passive, switched-capacitor, 16-point FFT. The design is based on a charge re-use technique that enables it to run at speeds of 5GS/s with a 47dB SNDR and is capable of transforming a 5GHz (I/Q) signal while consuming only 3.8mW (12.2pJ/conv.).

The chapter details the design, optimization, and implementation of the CRAFT processing engine. It also describes the on-chip interface circuitry and the test setup required to test such a high-speed, high dynamic range system. Non-idealities in the CRAFT computations are discussed, analytical models are derived, and new circuit

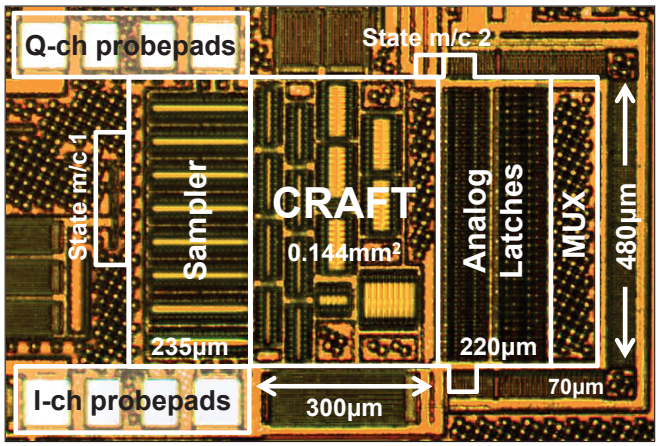


Figure 6.29: Die photo of CRAFT and supporting circuits in 65nm CMOS

techniques are developed for reducing and mitigating them. These techniques can be easily extended to other passive switched-capacitor circuits to improve their performance. Measurement results are then presented.

Table 6.4 compares the CRAFT performance with one digital and two analog domain FFT implementations. As shown, CRAFT operates at speeds 5X faster than previous state-of-the-art designs. Additionally, it consumes better than 28X lower energy. As an RF channelizer, it is expected to reduce digitization requirements enabling wideband digital spectrum sensing. As a result, it helps advance the state-of-the-art for wideband SDR architectures.

Moreover, this technique of performing passive charge-domain operations prior to sampling opens up a large number of possibilities. The CRAFT engine enables low power wide-band digitization of RF signals by performing N-path filtering before digitization (effectively a frequency-inteleave rather than time-interleave arrangement of conventional high-speed digitizers). This frequency-domain filtering and digitization can be utilized for spectrum sensing and real-time analysis in form factors and power envelopes that have been prohibitive (i.e. portable devices). Naturally, CRAFT and its

derivatives will benefit future cognitive radio, signal intelligence applications, as well as electronic warfare applications. Finally, by virtue of its orthogonal frequency domain transform characteristics, the CRAFT engine can be used in phased arrays utilizing the FFT for beamforming, or in spatio-spectral beamformers [45].

Table 6.4: Table for comparison with other FFT implementations

Ref	**Domain**	**bins**	**SNDR** (dB)	**Power** (mW)	**Speed** (GS/s)	**E/conv.**[*] (pJ/conv.)	**E/conv.**[*] ratio
[31]	Current	64	–	389	1.2	345.8	28X
[32]	Current	8	36	19	1.0	405.3	33X
[46]	Digital	128	51[†]	175	1.0	1600	131X
This work	**Charge**	**16**	**47**[‡]	**3.8**	**5.0**	**12.2**	**1X**

[*] Scaled for complexity similar to the scaling used in [32]
[†] 8.5bit ENOB assumed for 10-bit internal word length
[‡] After twiddle factor correction

Chapter 7

Conclusions & Contributions

The dream of a cognitive radio, first presented 15 years ago, is growing closer with increasing pace. Novel RF circuit architectures and circuit techniques have been developed in pursuit of a non-trivial functional realization. Currently, reconfigurable RF architectures are the preferred method of achieving wideband operation. Nevertheless, these designs fall short of the ideal cognitive radio, and lack the tremendous versatility in early descriptions [7].

In this thesis, circuit architectures and techniques in recent literature are discussed. Their suitability for spetrum sensing in cognitive radio applications is briefly reviewed. New architectures and circuits are described, and measurement results from a prototype design demonstrate promise through wideband sampled analog processing for frequency-domain digitization.

7.1 Remarks

SDR Spectrum Sensing:

1. The digital intelligence of current cognitive radios outstrips the RF front-end

performance and versatility. Wideband digitaization of an RF signal is therefore a limiting factor of current designs.

2. To reduce dynamic range and power requirements, analog domain signal processing before digitization is necessary. Discrete time signal processing is popular due to its compatibility and similarity with digital signal processing.

3. For spectrum sensing, wideband architectures and RF circuits must provide consistent performance across frequency. As a result of the large bandwidth of interest, circuit linearity and power becomes critical.

CRAFT Technology Scaling: As expected, measurements show CRAFT power consumption scales linearly with sample rate and quadratically with supply. This corroborates our claim that CRAFT is expected to respond favorably to technology scaling. The CRAFT energy consumption in lower technology nodes, for fixed design performance and input swing, is predicted to decrease as detailed in Table 7.1.

Table 7.1: CRAFT scaling with technology (est.)

Technology (L)	*65 nm*	*45 nm*	*32 nm*
nMOS thresh. (V_{tn}) [47]	0.57	0.51	0.37
Supply* (V_{core})	1.30	1.25	1.10
Computation switches†	0.39	0.19	0.09
Local clock routing†	0.21	0.15	0.10
Clock distribution‡	0.40	0.40	0.40
Total dynamic cap (C)	1.00	0.73	0.60
$\boldsymbol{E}_{\text{norm.}} \propto \boldsymbol{C}\boldsymbol{V}_{\text{core}}^2$	**1.00**	**0.68**	**0.43**

* For sufficient processing switch voltage headroom
† Gate area scales by $1/L^2$, gate wiring by $1/L$
‡ To first-order, does not scale with technology

CRAFT Improvements: The CRAFT core consumes 12.2pJ/conv. and uses 3.8mW of power when interleaved by two for a 5GS/s input and 5GS/s aggregate output.

Updated operating and implementation techniques have been identified to reduce power consumption of the current CRAFT design in half. This is primarily accomplished by gating unnecessary processing clocks during the core reset operation, and reducing local and global clock routing parasitic capacitance. Additionally, the following issues and solutions have been identified for future design iterations:

1. Improve sampler nonlinearity so it doesn't limit processing core performance.
2. Core wiring bus shielding for parasitic coupling capacitance reduction to improve matching and reduce digital correction complexity.
3. Core wiring bus reordering for fine-grained matching of capacitances.
4. Gm sampling for integrated AAF and increased sampling jitter tolerance.
5. Data windowing for improved out-out-band filtering.

CRAFT Applications: Immediate and future applications which benefit from high-speed analog-domain FFTs (or other linear transforms) have been identified:

1. Wideband spatio-spectral beamforming [15].
2. Fully-analog receivers (passive SC analog baseband demodulation).
3. BPSK, QPSK modulation/demodulation using low resolution DAC/ADC.
4. High speed OFDM charge-domain modulation and demodulation.
5. MIMO receivers and transmitters.
6. Polyphase FFT for improved fiter rejection.

7.2 Research Contributions

In this thesis, the following major contributions were made:

Passive Switched-Capacitor Computations

1. Passive SC computation elements are introduced and behavior analyzed.
2. Noise accumulated during charge-domain operations is quantified. Output noise correlation, and the advantages it provides, is determined. The noise of a cascade of computations is shown to not increase.
3. Linear and nonlinear error sources are identified and analyzed. This includes: systematic mismatch, clock feedthrough, charge injection, and incomplete settling including voltage dependent effects.
4. Error mitigation and circuit-level cancellation techniques are developed.
5. Nonlinear settling-error simulations are performed. Optimization techniques to maximize computational dynamic range are discussed.

Charge-Domain RF Front-end for Low-power Digitization

1. RF front-end sampled charge processing was explored using analytical system models to ease digitization.
2. A wideband spectrum sensing architecture based on a passive switched-capacitor DFT analog processing front-end was developed. This architecture was shown to be capable of wideband, low-power digitization in the frequency domain.
3. Non-idealities in passive computations were explored and modeled at the system-level. Calibration techniques and an LTI model were designed.

4. An ultra-low power, high-speed charge-domain FFT prototype for an SDR receiver front-end was designed and demonstrated in measurements [33, 35]. The prototype demonstrates orders of magnitude improvement in performance over other competing architectures and circuits.

References

[1] B. Murmann. ADC performance survey 1997-2012 (ISSCC & VLSI Symposium). *http://www.stanford.edu/£\sim£murmann/adcsurvey.html* , 2012.

[2] B. Fette. *Cognitive Radio Technology*. Newnes, 2006.

[3] NTIA. U.S. frequency allocations. *http://www.ntia.doc. gov/osmhome/allochrt.pdf*.

[4] D. Cabric, I.D. O'Donnell, M.S.-W. Chen, and R.W. Brodersen. Spectrum sharing radios. *IEEE Circuits and Systems Magazine*, 6(2):30 – 45, 2006.

[5] Federal Communications Commission. In the matter of facilitating opportunities for exible, efcient, and reliable spectrum use employing cognitive radio technologies, report and order. *FCC 05-57, ET Docket No. 03-108*, May 2005.

[6] T. Ulversoy. Software defined radio: Challenges and opportunities. *IEEE Communications Surveys Tutorials*, 12(4):531 – 550, 2010.

[7] J. Mitola III. Cognitive radio: an integrated agent architecture for software defined radio dissertation. *Royal Institute of Technology, Stockholm*, 2000.

[8] I. D. O'Donnel and R. W. Brodersen. An ultra-wideband transceiver architecture for low power, low rate, wireless systems. *IEEE Transactions on Vehicular Technology*, pages 1623 – 1631, September 2005.

[9] S. Hoyos, B.M. Sadler, and G.R. Arce. Analog to digital conversion of ultra-wideband signals in orthogonal spaces. In *IEEE Conference on Ultra Wideband Systems and Technologies*, pages 47 – 51, November 2003.

[10] S. Hoyos, B.M. Sadler, and G.R. Arce. Broadband multicarrier communication receiver based on analog to digital conversion in the frequency domain. *IEEE Transactions on Wireless Communications*, 5(3):652 – 661, March 2006.

[11] S. R. Valezquez, T. Q. Nguyen, S. R. Broadstone, and J. K. Roberge. A hybrid filter bank approach to analog-to-digital conversion. *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pages 116 – 119, October 1994.

[12] W. Namgoong. A channelized digital ultrawideband receiver. *IEEE Transactions for Wireless Communications*, pages 502 – 510, May 2003.

[13] T-L Hsieh, P. Kinget, and R. Gharpurey. A rapid interference detector for ultra wideband radio systems in 0.13μm CMOS. *IEEE Radio Frequency Integrated Circuits Symposium*, pages 347 – 350, April 2008.

[14] M. Elbadry, B. Sadhu, J. Qiu, and R. Harjani. Dual channel injection-locked quadrature LO generation for a 4GHz instantaneous bandwidth receiver at 21GHz center frequency. *IEEE Radio Frequency Integrated Circuits Symposium*, pages 333–336, June 2012.

[15] S. Kalia, M. Elbadry, B. Sadhu, S. Patnaik, J. Qiu, and R. Harjani. A simple, unified phase noise model for injection-locked oscillators. *IEEE Radio Frequency Integrated Circuits Symposium*, June 2011.

[16] T. H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 2003.

[17] R. Unbehauen and A. Cichocki. *MOS Switched-Capacitor and Continuous-Time Integrated Circuits and Systems: Analysis and Design*. Springer-Verlag, 1989.

[18] Roubik Gregorian and G.C. Temes. *Analog MOS Integrated Circuits for Signal Processing*. John Wiley & Sons, Inc., 1986.

[19] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2002.

[20] Y. Ding and R. Harjani. A universal analytic charge injection model. In *IEEE International Symposium for Circuits and Systems*, volume 1, pages 144 – 147, 2000.

[21] Z. Ru, E.A.M. Klumperink, and B. Nauta. On the suitability of discrete-time receivers for Software-Defined Radio. In *IEEE International Symposium on Circuits and Systems*, pages 2522 – 2525, May 2007.

[22] A. Abidi. The path to the Software-Defined Radio receiver. *IEEE Journal of Solid-State Circuits*, pages 954 – 966, May 2007.

[23] V.J. Arkesteijn, E.A.M. Klumperink, and B. Nauta. Jitter requirements of the sampling clock in software radio receivers. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(2):90 – 94, February 2006.

[24] K. Muhammad, D. Leipold, B. Staszewski, Y.-C. Ho, C.M. Hung, K. Maggio, C. Fernando, T. Jung, J. Wallberg, J.-S. Koh, S. John, I. Deng, O. Moreira, R. Staszewski, R. Katz, and O. Friedman. A discrete-time bluetooth receiver in a 0.13 μm digital CMOS process. In *IEEE International Solid-State Circuits Conference*, pages 268 – 527 vol.1, February 2004.

[25] K. Muhammad, Y.C. Ho, T. Mayhugh, C.M. Hung, T. Jung, I. Elahi, C. Lin, I. Deng, C. Fernando, J. Wallberg, S. Vemulapalli, S. Larson, T. Murphy,

D. Leipold, P. Cruise, J. Jaehnig, M.C. Lee, R.B. Staszewski, R. Staszewski, and K. Maggio. A discrete time quad-band GSM/GPRS receiver in a 90nm digital CMOS process. In *IEEE Custom Integrated Circuits Conference*, pages 809 – 812, September 2005.

[26] D. Jakonis, K. Folkesson, J. Dbrowski, P. Eriksson, and C. Svensson. A 2.4-GHz RF sampling receiver front-end in 0.18-μm CMOS. *IEEE Journal of Solid-State Circuits*, 40(6):1265 – 1277, June 2005.

[27] F. Montaudon, R. Mina, S. Le Tual, L. Joet, D. Saias, R. Hossain, F. Sibille, C. Corre, V. Carrat, E. Chataigner, J. Lajoinie, S. Dedieu, F. Paillardet, and E. Perea. A scalable 2.4-to-2.7GHz Wi-Fi/WiMAX discrete-time receiver in 65nm CMOS. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 362 – 619, February 2008.

[28] R. Bagheri, A. Mirzaei, S. Chehrazi, M.E. Heidari, Minjae Lee, M. Mikhemar, Wai Tang, and A.A. Abidi. An 800-MHz–6-GHz Software-Defined wireless receiver in 90-nm CMOS. In *IEEE Journal of Solid-State Circuits*, volume 41, pages 2860 – 2876, December 2006.

[29] Z. Ru, E.A.M. Klumperink, and B. Nauta. Discrete-time mixing receiver architecture for RF-sampling Software-Defined Radio. *IEEE Journal of Solid-State Circuits*, 45(9):1732 – 1745, September 2010.

[30] F. Harris, Chris Dick, and Michael Rice. Digital receivers and transmitters using polyphase filter banks for wireless communications. *IEEE Transactions on Microwave Theory and Techniques*, pages 1395 – 1412, April 2003.

[31] F. Rivet, Y. Deval, J.-B. Begueret, D. Dallet, P. Cathelin, and D. Belot. The experimental demonstration of a SASP-based full software radio receiver. *IEEE Journal of Solid-State Circuits*, pages 979 – 988, May 2010.

[32] M. Lehne and S. Raman. A 0.13-μm 1-GS/s CMOS discrete-time FFT processor for ultra-wideband OFDM wireless receivers. *IEEE Transactions on Microwave Theory and Techniques*, pages 1639 – 1650, November 2000.

[33] B. Sadhu, M. Sturm, B. M. Sadler, and R. Harjani. A 5GS/s 12.2pJ/conv. analog charge-domain FFT for a software defined radio receiver front-end in 65nm CMOS. *IEEE Radio Frequency Integrated Circuits Symposium*, pages 39–42, June 2012.

[34] R. Harjani, B. Sadhu, and M. Sturm. Multi-stage charge re-use analog circuits. *U.S. Patent (filed)*, June 2012.

[35] B. Sadhu, M. Sturm, B. M. Sadler, and R. Harjani. Analysis and design of a 5GS/s analog charge-domain FFT for an SDR front-end in 65nm CMOS. *IEEE Journal of Solid State Circuits*, 2013.

[36] I.-H. Wang, J.-L. Lin, and S.-I. Liu. 5-bit, 10 GSamples/s track-and-hold circuit with input feedthrough cancellation. *Electronics Letters*, 42(8):457–459, April 2006.

[37] A. Boni, A. Pierazzi, and C. Morandi. A 10-b 185-MS/s track-and-hold in 0.35-μm CMOS. *IEEE Journal of Solid-State Circuits*, 36(2):195–203, feb 2001.

[38] B. Murmann. EE315B: VLSI Data Conversion Circuits. *Stanford University*, 2011.

[39] N. J. Guilar, F. Lau, P. J. Hurst, and S. H. Lewis. A passive switched-capacitor finite-impulse-response equalizer. *IEEE Journal of Solid-State Circuits*, 42(2):400 – 409, February 2007.

[40] K.W. Martin. Complex signal processing is not complex. *IEEE Transactions on Circuits and Systems I*, 51(9):1823 – 1836, September 2004.

[41] G. Wegmann, E.A. Vittoz, and F. Rahali. Charge injection in analog MOS switches. *IEEE Journal of Solid-State Circuits*, 22(6):1091 – 1097, December 1987.

[42] A.M. Abo and P.R. Gray. A 1.5-V, 10-bit, 14.3-MS/s CMOS pipeline analog-to-digital converter. *IEEE Journal of Solid-State Circuits*, 34(5):599–606, May 1999.

[43] S.M. Louwsma, A.J.M. van Tuijl, M. Vertregt, and B. Nauta. A 1.35 GS/s, 10 b, 175 mW time-interleaved AD converter in 0.13 μm CMOS. *IEEE Journal of Solid-State Circuits*, 43(4):778–786, April 2008.

[44] C. Eichenberger and W. Guggenbühl. Dummy transistor compensation of analog MOS switches. *IEEE Journal of Solid-State Circuits*, 24(4):1143 – 1146, August 1989.

[45] S. Patnaik, S. Kalia, B. Sadhu, M. Sturm, M. Elbadry, and R. Harjani. An 8GHz multi-beam spatio-spectral beamforming receiver using an all-passive discrete time analog baseband in 65nm CMOS. In *IEEE Custom Integrated Circuits Conference*, September 2012.

[46] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee. A 1-GS/s FFT/IFFT processor for UWB applications. *IEEE Journal of Solid-State Circuits*, pages 1726 – 1735, August 2005.

[47] ITRS: Model for assessment of CMOS technologies and roadmaps (MASTAR). *http://www.itrs.net/models.html*, 2011.

Appendix A

CRAFT matrices

The linear operation performed by the DFT is written as in Eq. (6.2). When the computation is performed in a stage-wise manner, as is done by an FFT, it can be decomposed into a sequence of operations as shown below:

$$\textit{FFT:} \qquad \vec{\mathbf{X}} = \mathbf{F}_4\mathbf{F}_3\mathbf{F}_2\mathbf{F}_1\mathbf{I}_{\text{bitrev}}\vec{\mathbf{x}}$$

These four (radix-2, 16-point FFT) stages, $\mathbf{F}_1$ through $\mathbf{F}_4$, are shown below and are represented graphically in Fig. 6.4(a). $\mathbf{I}_{\text{bitrev}}$ is an identity matrix modified to perform bit-reverse ordering of the input vector, $\vec{\mathbf{x}}$, for the decimation-in-time algorithm.

$$\mathbf{I}_{\text{bitrev}} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

$$\mathbf{F}_1 = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & W^4 & -W^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & W^4 & -W^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^4 & -W^4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^4 & -W^4
\end{bmatrix}$$

$$\mathbf{F}_2 = \begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & W^0 & 0 & W^0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & W^2 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & W^4 & 0 & -W^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & W^6 & 0 & -W^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^0 & 0 & W^0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & W^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^4 & 0 & -W^4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^6 & 0 & -W^6
\end{bmatrix}$$

$$\mathbf{F}_3 = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^0 & 0 & 0 & 0 & W^0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^1 & 0 & 0 & 0 & W^1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & W^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 & 0 & 0 & 0 & W^3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^4 & 0 & 0 & 0 & -W^4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^5 & 0 & 0 & 0 & -W^5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^6 & 0 & 0 & 0 & -W^6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^7 & 0 & 0 & 0 & -W^7
\end{bmatrix}$$

$$\mathbf{F}_4 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}$$

For CRAFT, this sequence of operations represents the cascade of in-place processing operations performed. Therefore, the four-stage, charge-domain, radix-2, 16-point, decimation-in-time FFT implemented by CRAFT (see Eq. (6.4)) is written as:

$$\textit{CRAFT:} \qquad \vec{\mathbf{X}} = \tfrac{1}{16} \cdot \mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1\mathbf{I}_{\text{bitrev}}\vec{\mathbf{x}}$$

where $\mathbf{I}_{\text{bitrev}}$ was shown previously, and $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$, and $\mathbf{S}_4$ are shown below. They differs from the FFT matrices due to the attenuation, charge-averaging, and stage scaling factor effects of the implementation. These matrices are also rewritten in a manner that matches the implementation show in Fig. 6.7 and described in Secs. 6.2 and 6.3.2.

$$\mathbf{S}_1 = \mathbf{F}_1$$

$$\mathbf{S}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix}
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & W^2 & 0 & W^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -j & 0 & j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -jW^2 & 0 & jW^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & W^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -j & 0 & j & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -jW^2 & 0 & jW^2
\end{bmatrix}$$

$$\mathbf{S}_3 = \left(\frac{1}{2\cos(\frac{\pi}{8})} \right) \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^1 & 0 & 0 & 0 & W^1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^2 & 0 & 0 & 0 & W^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & W^3 & 0 & 0 & 0 & W^3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & j & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -jW^1 & 0 & 0 & 0 & jW^1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -jW^2 & 0 & 0 & 0 & jW^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -jW^3 & 0 & 0 & 0 & jW^3
\end{bmatrix}$$

$$\mathbf{S}_4 = \mathbf{F}_4$$

Appendix B

FFT LTI model

The complete CRAFT+ADC digitization operation, including systematic twiddle factor inaccuracies and noise, can be approximated using an LTI model as discussed in Section 6.5.3. Fig. B.1 and Table B.1 have been reprinted here for clarity and details of each component of the model are covered in this appendix.

As discussed in Chapter 6 and detailed in Appendix A, the 16-point FFT ($\mathbf{X} = \mathbf{F}\,\mathbf{x}$) is implemented as $\mathbf{X} = \frac{1}{16}\,\mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1\mathbf{I}_{\text{bitrev}}\mathbf{x}$, where $\frac{1}{16}$ is due to the inherent scaling of charge-based operations (averaging vs. addition).

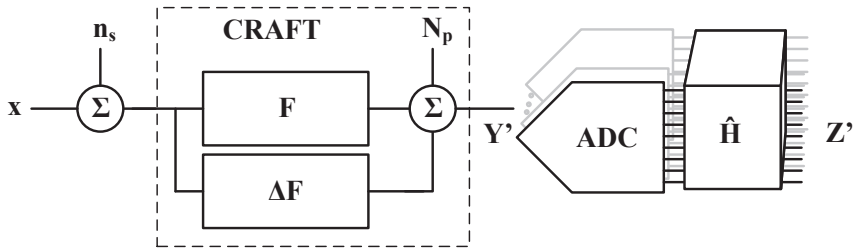


Figure B.1: An LTI model of the CRAFT operation with digitization and correction

Table B.1: Summary of variables in LTI model

Variable	Description	Expression
$\mathbf{x}$	Input (16 samples)	$x(i), \forall\ i = [0, 15], i \in \mathbb{Z}$
$\mathbf{n}_s$	Sampler noise (pseudo-differential)	$\sigma^2_{n_{s,i}} = 2k\mathrm{T}/C$ C = capacitance/copy
$\mathbf{F}$	Ideal FFT	$\mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1$, see Appendix A
$\mathbf{\Delta F}$	Linear FFT implementation error	See Appendix B
$\mathbf{F}'$	FFT approximation	$\mathbf{F}+\mathbf{\Delta F}$, see Appendix B
$\mathbf{N}'_\mathbf{p}$	Equiv. processing + reset noise at CRAFT output	See Appendix B
$\mathbf{N_q}$	Equiv. quantization noise at ADC input	See Appendix B
$\hat{\mathbf{H}}$	Correction matrix	$\mathbf{F}(\hat{\mathbf{F}}')^{-\mathbf{1}}$, see Sec. 6.6.1
$\mathbf{Y}'$	Digitized output	$(\mathbf{F}+\mathbf{\Delta F})\,(\mathbf{x}+\mathbf{n_s})+\mathbf{N}'_\mathbf{p}+\mathbf{N_q}$, see Appendix B
$\mathbf{Z}'$	Digitized and corrected output	$\hat{\mathbf{H}}\,\mathbf{Y}'=\hat{\mathbf{X}}+\hat{\mathbf{N}}_\mathbf{s}+\hat{\mathbf{N}}_\mathbf{p}+\hat{\mathbf{H}}\,\mathbf{N_q}$, see Appendix B

B.1 Systematic Twiddle Factor Error

The linear errors in the FFT matrix due to systematic capacitor mismatch (sampling and stealing capacitors) can be represented using matrix $\mathbf{F}'$ rather than the ideal matrix $\mathbf{F}$ ($= \frac{1}{16}\,\mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1$), producing $\mathbf{X}'$ at the output, where

$$\mathbf{X}' = \mathbf{F}'\,\mathbf{x} = \tfrac{1}{16}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\mathbf{x}$$

and $\mathbf{I}_{\text{bitrev}}$ is absorbed into $\mathbf{S}'_1$ for notation simplicity.

Then, by expanding each non-ideal stage ($\mathbf{S}'_i$) as an ideal stage ($\mathbf{S}_i$) plus the systematic error ($\mathbf{E}_i$) it can be rewritten as:

$$\begin{aligned}\mathbf{X}' &= \tfrac{1}{16}\,(\mathbf{S}_4+\mathbf{E}_4)\,(\mathbf{S}_3+\mathbf{E}_3)\,(\mathbf{S}_2+\mathbf{E}_2)\,(\mathbf{S}_1+\mathbf{E}_1)\,\mathbf{x}\\ &= (\mathbf{F}+\Delta\mathbf{F})\,\mathbf{x}\end{aligned}$$

where,

$$\begin{aligned}\Delta\mathbf{F} = \tfrac{1}{16}\big(&\mathbf{E}_4\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1 + \mathbf{E}_4\mathbf{E}_3\mathbf{E}_2\mathbf{S}_1 + \mathbf{E}_4\mathbf{E}_3\mathbf{S}_2\mathbf{E}_1 + \mathbf{E}_4\mathbf{S}_3\mathbf{E}_2\mathbf{E}_1 + \mathbf{S}_4\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1 + ... \\ &\mathbf{E}_4\mathbf{E}_3\mathbf{S}_2\mathbf{S}_1 + \mathbf{E}_4\mathbf{S}_3\mathbf{E}_2\mathbf{S}_1 + \mathbf{S}_4\mathbf{E}_3\mathbf{E}_2\mathbf{S}_1 + \mathbf{E}_4\mathbf{S}_3\mathbf{S}_2\mathbf{E}_1 + \mathbf{S}_4\mathbf{E}_3\mathbf{S}_2\mathbf{E}_1 + ... \\ &\mathbf{S}_4\mathbf{S}_3\mathbf{E}_2\mathbf{E}_1 + \mathbf{E}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1 + \mathbf{S}_4\mathbf{E}_3\mathbf{S}_2\mathbf{S}_1 + \mathbf{S}_4\mathbf{S}_3\mathbf{E}_2\mathbf{S}_1 + \mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{E}_1\big)\end{aligned}$$

B.2 Noise

The CRAFT operation is linear despite the presence of twiddle factor errors. Consequently, the noise in the system can be considered independently using linear superposition of the individual noise sources: the sampler, processing computations, and the ADC, as shown below.

Sampler Noise

Noise of a sampling operation modifies the system input values to $\mathbf{x_n}$. This sampling noise power can be treated as an independent random variable, $\mathbf{n_s}$, combining with the ideal final sample values as: $\mathbf{x_n} = (\mathbf{x} + \mathbf{n_s})$. When passed through CRAFT, the complete effect of sampling noise can be isolated as an independent term, $\mathbf{N'_s}$, as shown below.

$$\begin{aligned}\mathbf{X}' &= \mathbf{F}'\,\mathbf{x_n} = \mathbf{F}'\,(\mathbf{x} + \mathbf{n_s}) \\ &= \mathbf{F}'\,\mathbf{x} + \mathbf{N'_s}\end{aligned} \tag{B.1}$$

where,

$$\mathbf{N'_s} = (\mathbf{F} + \Delta\mathbf{F})\,\mathbf{n_s} = \mathbf{F}'\,\mathbf{n_s}$$

Therefore, sampling noise can be an additive term either before or after CRAFT.

Note that the outputs of the CRAFT operation are interpreted in the frequency domain. Therefore, any mismatch in the transfer function gains to each outputs will cause a coloration of the output referred sampler noise. As shown below, the expected

sampler noise appears at the FFT output as white noise plus additional colored noise terms due to the unequal gains of $\Delta\mathbf{F}$ at the different output bins.

$$\begin{aligned}\mathbf{X}' &= \mathbf{F}'\,\mathbf{x_n} = (\mathbf{F}+\Delta\mathbf{F})\,(\mathbf{x}+\mathbf{n_s}) \\ &= \underbrace{\mathbf{F}\,(\mathbf{x}+\mathbf{n_s})}_{\text{white}} + \underbrace{\Delta\mathbf{F}\,(\mathbf{x}+\mathbf{n_s})}_{\text{colored}}\end{aligned}$$

Finally, it is noteworthy that with digital correction, $\hat{\mathbf{H}}\,\mathbf{X}'$, sampling noise is restored to being purely white as shown below.

$$\hat{\mathbf{X}} = \hat{\mathbf{H}}\,\mathbf{X}' = \mathbf{F}(\hat{\mathbf{F}}')^{-1}\mathbf{X}' \approx \mathbf{F}(\mathbf{x}+\mathbf{n_s})$$

Processing Noise

Each stage of charge-domain processing, $\mathbf{S}_i$, adds noise, $\mathbf{N}_i$, at its completion. Noise from the core reset operation disturbs the initial stealing capacitor values but can be treated as a final-value noise since the system is linear. Therefore, CRAFT, including reset and processing noise, is expanded below.

$$\begin{aligned}\mathbf{X}' &= \frac{1}{2}\,\mathbf{S}'_4\left(\frac{1}{2}\,\mathbf{S}'_3\left(\frac{1}{2}\,\mathbf{S}'_2\left(\frac{1}{2}\,\mathbf{S}'_1\mathbf{x}+\mathbf{N}_1\right)+\mathbf{N}_2\right)+\mathbf{N}_3\right)+\mathbf{N}_4 \\ &= \tfrac{1}{16}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\mathbf{x}+\mathbf{N}'_\mathbf{p} \\ &= \mathbf{F}'\,\mathbf{x}+\mathbf{N}'_\mathbf{p}\end{aligned} \tag{B.2}$$

where,

$$\mathbf{N}'_\mathbf{p} = \frac{1}{8}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{N}_1+\frac{1}{4}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{N}_2+\frac{1}{2}\,\mathbf{S}'_4\mathbf{N}_3+\mathbf{N}_4$$

This relationship shows that the cascaded processing noise can be modeled as an output-referred noise, $\mathbf{N}'_\mathbf{p}$, that is composed of linear combinations of independent random variables (elements of $\mathbf{N}_i$). Any systematic implementation errors causing $\mathbf{S}'_i$ to deviate from $\mathbf{S}_i$ will only modify the weighting of these linear combinations.

The effect of digital correction on processing noise is discussed later.

ADC Noise

The ADC quantizers add independent noise ($\mathbf{N_q}$) to every output of the CRAFT operation, and are interpreted as frequency domain noise perturbations, given by:

$$\mathbf{X'} = \mathbf{F'}\,\mathbf{x} + \mathbf{N_q} \tag{B.3}$$

In the case where all the ADCs have equal gains, and the white quantization noise approximation holds, $\mathbf{N_q}$ can be approximated to be white. With per-bin AGCs before the ADCs, entries of $\mathbf{N_q}$ are scaled by the reciprocal of the gains. If the output is transformed back to the time domain, the non-uniform quantization noise amplitude creates a colored effect.

All Noise Sources

The effects of all noise sources can be included using linear superposition. Sampling, processing, and ADC quantization noise are represented in the equation below.

$$\begin{aligned}\mathbf{Y'} &= \mathbf{F'}(\mathbf{x} + \mathbf{n_s}) + \mathbf{N'_p} + \mathbf{N_q} \\ &= (\mathbf{F} + \Delta\mathbf{F})\mathbf{x} + \mathbf{N'_s} + \mathbf{N'_p} + \mathbf{N_q}\end{aligned}$$

B.3 Digital correction:

The output, $\mathbf{Z'}$, after digital correction, is computed below.

$$\mathbf{Z'} = \hat{\mathbf{H}}\,\mathbf{Y'} = \hat{\mathbf{X}} + \hat{\mathbf{N}}_\mathbf{s} + \hat{\mathbf{N}}_\mathbf{p} + \hat{\mathbf{H}}\,\mathbf{N_q} \tag{B.4}$$

This assumes that the correction uses an accurate estimate of the imlementation, is represented by the relationships $\hat{\mathbf{F}}' \approx \mathbf{F'}$ and $\hat{\mathbf{S}}'_i \approx \mathbf{S}'_i$. Consequently,

$$\hat{\mathbf{N}}_\mathbf{s} = \hat{\mathbf{H}}\,\mathbf{N}'_\mathbf{s} = \mathbf{F}(\hat{\mathbf{F}}')^{-1}\mathbf{N}'_\mathbf{s} = \mathbf{F}(\hat{\mathbf{F}}')^{-1}\mathbf{F}'\mathbf{n_s} \approx \mathbf{F}\,\mathbf{n_s}$$

and,

$$\begin{aligned}\hat{\mathbf{N}}_\mathbf{p} &= \hat{\mathbf{H}}\,\mathbf{N}'_\mathbf{p} = \mathbf{F}(\hat{\mathbf{F}}')^{-1}\mathbf{N}'_\mathbf{p} \\ &= \mathbf{F}\,(\hat{\mathbf{S}}'_4\hat{\mathbf{S}}'_3\hat{\mathbf{S}}'_2\hat{\mathbf{S}}'_1)^{-1}\left[\frac{1}{8}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{N}_1 + \frac{1}{4}\,\mathbf{S}'_4\mathbf{S}'_3\mathbf{N}_2 + \frac{1}{2}\,\mathbf{S}'_4\mathbf{N}_3 + \mathbf{N}_4\right] \\ &\approx \mathbf{F}\left[\frac{1}{8}\mathbf{S}'^{-1}_1\mathbf{N}_1 + \frac{1}{4}\left(\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_2 + \frac{1}{2}\left(\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_3 + \left(\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_4\right]\end{aligned}$$

Furthermore, if the implementation has little error from the ideal transform ($\mathbf{S}'_i \approx \mathbf{S}_i$), the approximation below shows that the processing noise is restored to its ideal value.

$$\begin{aligned}\hat{\mathbf{N}}_\mathbf{p} &\approx \mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{S}_1\left[\frac{1}{8}\mathbf{S}'^{-1}_1\mathbf{N}_1 + \frac{1}{4}\left(\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_2 + \frac{1}{2}\left(\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_3 + \left(\mathbf{S}'_4\mathbf{S}'_3\mathbf{S}'_2\mathbf{S}'_1\right)^{-1}\mathbf{N}_4\right] \\ &\approx \frac{1}{8}\,\mathbf{S}_4\mathbf{S}_3\mathbf{S}_2\mathbf{N}_1 + \frac{1}{4}\,\mathbf{S}_4\mathbf{S}_3\mathbf{N}_2 + \frac{1}{2}\,\mathbf{S}_4\mathbf{N}_3 + \mathbf{N}_4 = \mathbf{N}_\mathbf{p}\end{aligned}$$

In summary, Eq. (B.4) shows that digital correction correctly genrates an estimate, $\hat{\mathbf{X}}$, of the desired ideal, noiseless transform $\mathbf{X} = \mathbf{F}\,\mathbf{x}$. Additionally, the colored sampling and processing noise terms are restored to their ideal output-referred values: $\hat{\mathbf{N}}_\mathbf{s} \approx \mathbf{F}\,\mathbf{n_s}$ and $\hat{\mathbf{N}}_\mathbf{p} \approx \mathbf{N_p}$. Quantization noise, $\hat{\mathbf{H}}\,\mathbf{N_q}$, is slightly modified after correction. Rather than being completely independent between output bins, the correction matrix causes some weighted combining of outputs. However, since the implementation should match the desired ideal transform to many bits of accuracy, this effect is minimal.

Appendix C

Glossary and Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This appendix defines jargon terms in a table of acronyms and their expansion.

C.1 Acronyms

Table C.1: Acronyms

Acronym	Meaning
A/D	Analog to Digital
ADC	Analog to Digital Converter
AGC	Automatic Gain Control
ASIC	Application-Specific Integrated Circuit
	Continued on next page

Table C.1 – continued from previous page

Acronym	Meaning
ASP	Analog Signal Processing
AWGN	Additive White Gaussian Noise
BB	BaseBand
BJT	Bipolar Junction Transistor
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
BW	BandWidth
CMOS	Complementary Metal Oxide Semiconductor
CR	Cognitive Radio
CRAFT	Charge Reuse Analog Fourier Transform
CT	Continuous-Time
D/A	Digital to Analog
DAC	Digital to Analog Converter
dB	Decibel
dBc	Decibel normalized to carrier

Continued on next page

Table C.1 – continued from previous page

Acronym	Meaning
dBFS	Decibel normalized to Full Scale
dBm	Decibel normalized to 1mW
DSA	Dynamic Spectrum Access
DC	Direct Current, 0 Hz
DFT	Discrete Fourier Transform
DNL	Differential Non-Linearity
DR	Dynamic Range
DSP	Digital Signal Processing
DT	Discrete-Time
EM	Electro-Magnetic
EMI	Electromagnetic Interference
FET	Field Effect Transistor
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array

Continued on next page

Table C.1 – continued from previous page

Acronym	Meaning
HPF	High Pass Filter
I/O	Input/Output
I−Q	In phase − Quadrature phase
IC	Integrated Circuit
IF	Intermediate Frequency
IIR	Infinite Impulse Response
INL	Integral Non-Linearity
LNA	Low Noise Amplifier
LO	Local Oscillator
LPF	Low Pass Filter
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
OFDM	Orthogonal Frequency Division Multiplexing
OpAmp	Operational Amplifier
OTA	Operational Transconductance Amplifier
QPSK	Quadrature Phase-Shift Keying

Continued on next page

Table C.1 – continued from previous page

Acronym	Meaning
PA	Power Amplifier
PCB	Printed Circuit Board
PLL	Phase Locked Loop
PSK	Phase Shift Keying
Q	Quality factor
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RC	Resistor-Capacitor
RF	Radio-Frequency
Rx	Receiver
SASP	Sampled Analog Signal Processor
SC	Switched-Capacitor
SDR	Software Defined Radio
SFDR	Spurious Free Dynamic Range
SNDR	Signal to Noise and Distortion Ratio

Continued on next page

Table C.1 – continued from previous page

Acronym	Meaning
SNR	Signal to Noise Ratio
SR	Software Radio
TV	Television
Tx	Transmitter
UWB	Ultra-WideBand
UGF	Unity Gain Frequency
VGA	Variable Gain Amplifier
WLAN	Wireless Local Area Network