Digital Signal Processing with Signal-Derived Timing: Analysis and Implementation

Yu Chen

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

Abstract

Digital Signal Processing with Signal-Derived Timing: Analysis and Implementation

Yu Chen

This work investigates two different digital signal processing (DSP) approaches that rely on signal-derived timing: continuous-time (CT) DSP and variable-rate DSP. Both approaches enable designs of energy-efficient signal processing systems by relating their operation rates to the input activity.

The majority of this thesis focuses on CT-DSP, whose operations are completely digital in CT, without the use of a clock. The spectral features of CT digital signals are analyzed first, demonstrating a general pattern of the quantization noise spectrum added in CT amplitude quantization. Then the focus is narrowed to the investigations of the system characteristics and architecture of CT digital infinite-impulse-response (IIR) filters, which are barely studied in the previous work on this topic. This thesis discusses and addresses previously unreported stability issue in CT digital IIR filters with the presence of delay-line mismatches and proposes an innovative method to design high-order CT digital IIR filters with only two tap delays. Introducing an event detector allows the operation rate of a CT digital IIR filter to closely track the input activity even though it is a feed-

back system. For the first time, the filtered CT digital signal is converted to a synchronous digital signal. This facilitates integrating the CT digital filter and conventional discrete-time systems and expands the applications of the former. This discussion uses a computationally efficient interpolation filter to improve the signal accuracy of the synchronous digital output. On the circuit level, a new delay-cell design is introduced. It ensures low jitter, good matching, robust communication with adjacent circuits and event-independent delay.

An integrated circuit (IC) with all these ideas adopted was fabricated in a TSMC 65 nm LP CMOS process. It is the first IC implementation of a CT digital IIR filter. It can process signals with a data rate up to 20 MHz. Thanks to the IIR response and the 16-bit resolution used in the system, the implemented filter can achieve a frequency response much more versatile and accurate than the CT digital filters in prior art. The implemented system features an agile power adaptive to input activity, varying from 2.32 mW (full activity) to 40 μW (idle) with no power-management circuitry.

The second part of the thesis discusses a variable-rate DSP capable of processing samples with a variable sampling rate. The clock rate in the variable-rate DSP tracks the input sampling rate. Compared to a fixed-rate DSP, the proposed system has a lower output data rate and hence is more computationally efficient. A reconstruction filter with a variable cutoff frequency is used to reconstruct the output. The signal-to-noise ratio remains fixed when the sampling rate changes.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Prof. Yannis Tsividis, my advisor, for inspiring my interest in integrated circuit design, for his patient guidance of my research, and for always answering my questions in a timely manner. The past six years was a tough journey. But I never felt hopeless, because he could always point out to me a way out of the difficulties.

I would like to thank my thesis committee members: Profs. Rajit Manohar, Thao Nguyen, Mingoo Seok and John Wright. Special thanks to Prof. Rajit Manohar, for his guidance on asynchronous digital circuit design and his amazing tool for design verification. Also to Prof. Thao Nguyen, for offering a wonderful Advanced Digital Signal Processing course, which inspired my interest in this topic.

A lot of other people have to be especially mentioned, because things would have been very different without them. I want to thank Prof. Steven Nowick for teaching me the basics of asynchronous digital circuit design and Prof. Kinget for offering great circuit design classes and for his helpful discussions of my circuits. I want to thank my colleague, Sharvil Patil, for the countless discussions we had and for the joy that resulted from our collaboration over the past few years. I want to thank Jianxun Zhu for teaching me numerous things, e.g. PCB design, soldering, testing, etc., and Yang Xu for his insightful discussions. I want to thank Kevin Tien for answering my CAD tool questions and for other enjoyable conversations. I also want to thank Rabia Yazicigil, Ning Guo, Josh Kim, Jayanth Kn, Baradwaj Vigraham, Christos Vezyrtzis, Chun-wei Hsu and Karthik Tripurari for their helpful discussions and assistance.

Finally, I want to thank my family. I thank my parents; their support kept me going. I thank my wife for her love, her support, and for undergoing this tough journey with me.

Chapter 1

Introduction

Digital signal processors (DSPs[1]) are widely used in modern electronic systems because they are highly programmable and robust to noise. Conventional DSPs process discrete-time (DT) digital samples generated by a DT analog-to-digital converter (ADC[2]). As shown in Fig. 1.1, A DT-ADC samples an analog input with a uniform-rate clock and then quantizes the samples to discrete values. To accommodate the worst-case scenario, the sampling clock frequency must be at least twice the input's maximum possible bandwidth. The clock frequency is fixed regardless of the actual input activity. This implies that both conventional DSPs and DT-ADCs must always operate on a clock rate chosen for the worst case. On the other hand, many real-world signals have time-varying activities. For example, speech signals can contain frequency components as high as 10 kHz, but the signal power is usually concentrated in a narrower band of around 3.5 kHz [1]. Biosignals, such as those found in electrocardiography (ECG), are sporadic signals containing short intervals with

[1] "DSP" is used as an abbreviation for both "digital signal processor" and "digital signal processing."

[2] "ADC" is used as an abbreviation for both "analog-to-digital converter" and "analog-to-digital conversion."

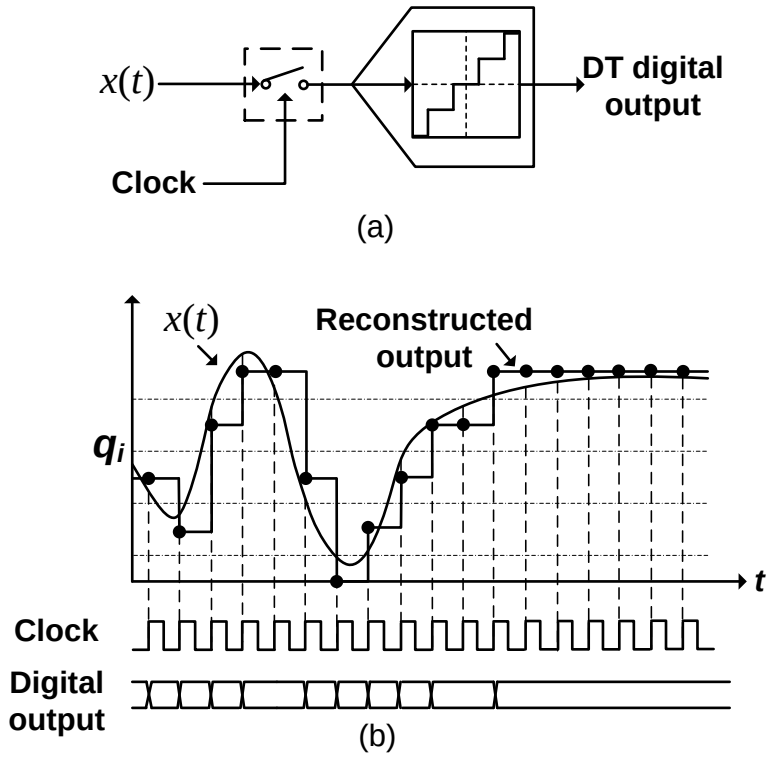


Figure 1.1: An example of a DT-ADC. (a) Block diagram. (b) Operation of the CT-ADC. q_i are quantization levels.

high activity and long periods of silence. Sampling and processing these signals with a uniform worst-case clock rate is unnecessarily energy inefficient.

Two approaches have been implemented to improve the signal-tracking property of the data conversion: continuous-time (CT) ADC [2, 3, 6] and variable-rate (VR) ADC [1, 8–11]. A CT-ADC converts an analog input into a set of CT binary waveforms without sampling. One common CT-ADC is level-crossing sampling (LCS) [3, 12–14]. This scheme can be modeled as an ideal quantizer (Fig. 1.2(a)). Whenever the input $x(t)$ crosses one of the quantization levels q_i, the amplitude of the quantized signal $x_q(t)$ changes. Because the amplitude of the quantized signal has a finite number of values, the signal can be represented by a set of CT binary waveforms (Fig.

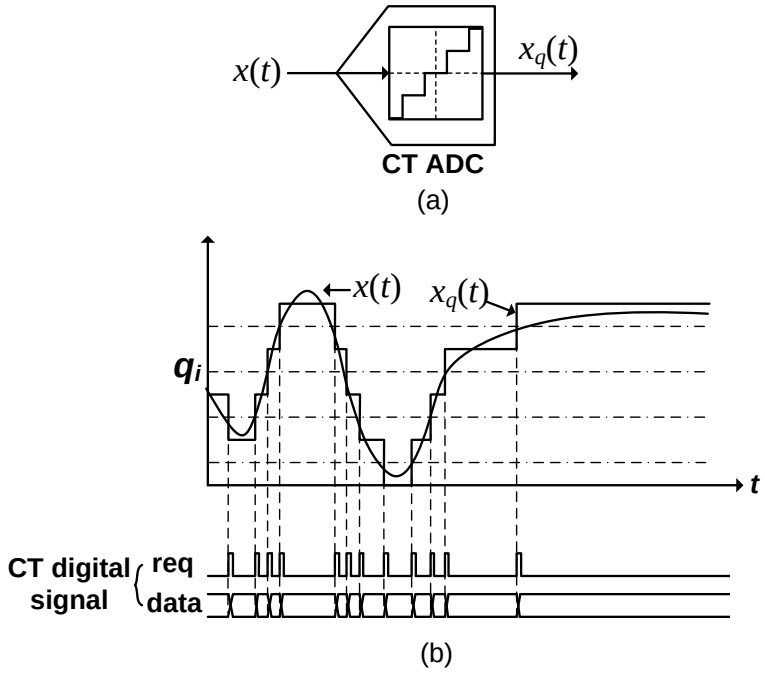


Figure 1.2: An example of a CT-ADC. (a) The block diagram. (b) Operation of the CT-ADC. q_i are quantization levels.

1.2(b)). The time instants at which the amplitude changes are recorded by the pulses on the *req* (request) bit. *Req* is effectively signal-derived timing, which asks the following stage (e.g., a DSP) to process the data. The combination of the timing *req* and the amplitude value *data* forms a CT digital signal. We call an update in the data value of a CT digital signal an *event*. Notice from Fig. 1.2(b) that the event rate tracks the input. When $x(t)$ varies quickly, it crosses quantization levels more frequently, which generates a high event rate. When $x(t)$ varies slowly, the event rate decreases accordingly. In the extreme case where $x(t)$ is almost quiescent, as shown on the right of 1.2(b), no event is generated for a long time. The nonuniform interevent intervals in a CT digital signal are key to the signal representation.

A VR-ADC converts an analog input into a DT digital output, but with a nonuniform sampling

rate. Similar to a conventional DT-ADC, it is composed of a sampler followed by a quantizer (Fig. 1.3(a)). The difference between the two lies in the sampling clock. In the conventional DT-ADC (Fig. 1.1), the clock has a fixed sampling rate. By contrast, the VR-ADC clock's sampling rate is variable. Its exact value is determined by the activity detector. The detector takes both $x(t)$ and a base clock as its inputs and derives a VR clock according to the input activity. The base clock has a fixed rate which is always the highest in the system. Its operations are shown in Fig. 1.3(b). When $x(t)$ varies quickly, the VR clock operates at its highest rate, the same as the base clock, and the DT digital samples are generated with the highest rate. When $x(t)$ varies slowly, the VR clock slows to an integer fraction of the base clock, which results in lower sampling rate. The VR clock is a signal-derived timing.

Both CT-ADCs and VR-ADCs generate events/samples whose rate tracks the input activity. In a CT-ADC, an event is generated at the level-crossing and hence is free of quantization error at those time instants. Its signal-derived timing, *req,* is not necessarily aligned to any timing grid, which results in an aliasing-free output. In addition, its event rate can go as low as zero if there is no activity in the input at all. In contrast, the VR clock is derived both from the input and a base clock. Samples are quantized in both the amplitude and the time domains and hence suffer from quantization error and aliasing. In addition, even when the input has no activity, the VR clock's lowest rate is not usually zero.

We use the term "event" in our discussions of CT digital signals. A CT digital signal is defined in the entire time domain; "events," which are defined as the updates of the signal value, are only part of the signal. Another part of the information is recorded in the interevent intervals, during

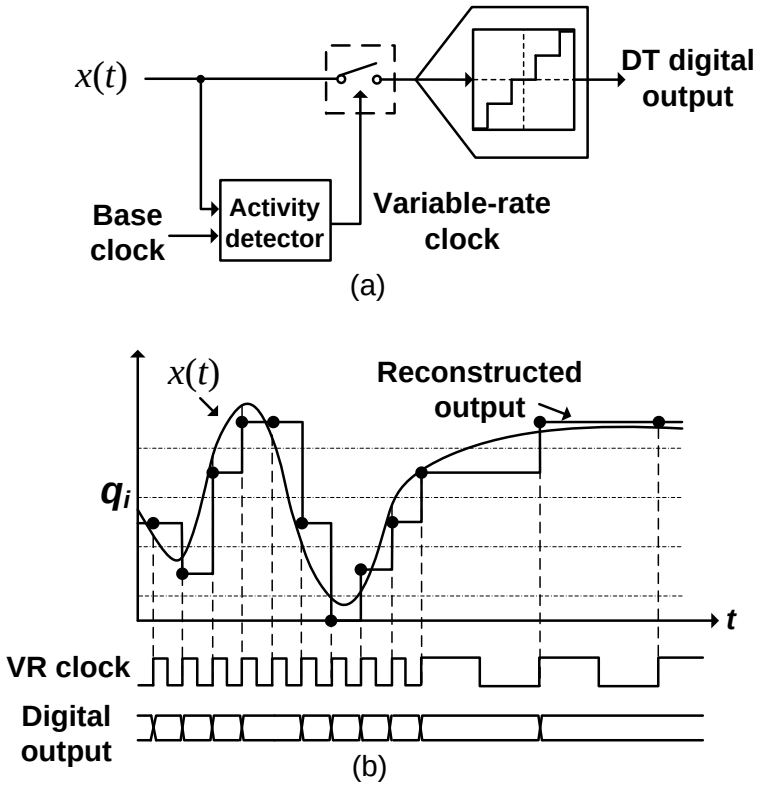


Figure 1.3: An example of a VR-ADC. (a) The block diagram. (b) Operation of the VR-ADC. q_i are quantization levels.

which the signal value doesn't change. By contrast, we use the term "sample" in the discussions of DT digital signals. A DT digital signal contains meaningful information only at the discrete sampling instants when samples are generated.

Despite their differences, both data converters offer the promise of energy-efficient systems by relating the event/sample rate to the input activity. This work discusses approaches to design energy-efficient DSPs in both schemes.

1.1 DSPs with signal-derived timing

Fig. 1.4(a) shows the structure of a conventional DT-DSP. It is composed of arithmetic blocks (i.e., the multipliers and the adders) and a tapped delay line. Because the preceding DT-ADC generates digital samples with a uniform rate and the DT-DSP operates under the same clock rate, the tapped delay line can simply be built with a chain of D flip flops (DFFs). Each time the delay line shifts the digital samples down by one position, it effectively implements a delay of one sample interval (i.e., one clock cycle). However, this delay line doesn't work for CT digital signals from CT-ADCs or VR DT digital signals from VR-ADCs because their event/sample intervals are nonuniform. New hardware is needed to handle these CT and VR DT digital signals.

The DSP processing a CT digital signal is called a CT-DSP (Fig. 1.4(b)). It is similar to a DT-DSP except for the delay line. Because the events can occur at any time in a CT digital signal, not aligned to any uniform timing grid, it must rely on the CT delay blocks to implement tap delays. Both *req* and *data* are delayed by the CT delay blocks. The original signal-derived timing *req* further derives several new timings req_i, $i = 1, 2, \ldots, K$, that trigger the operations at each CT-DSP

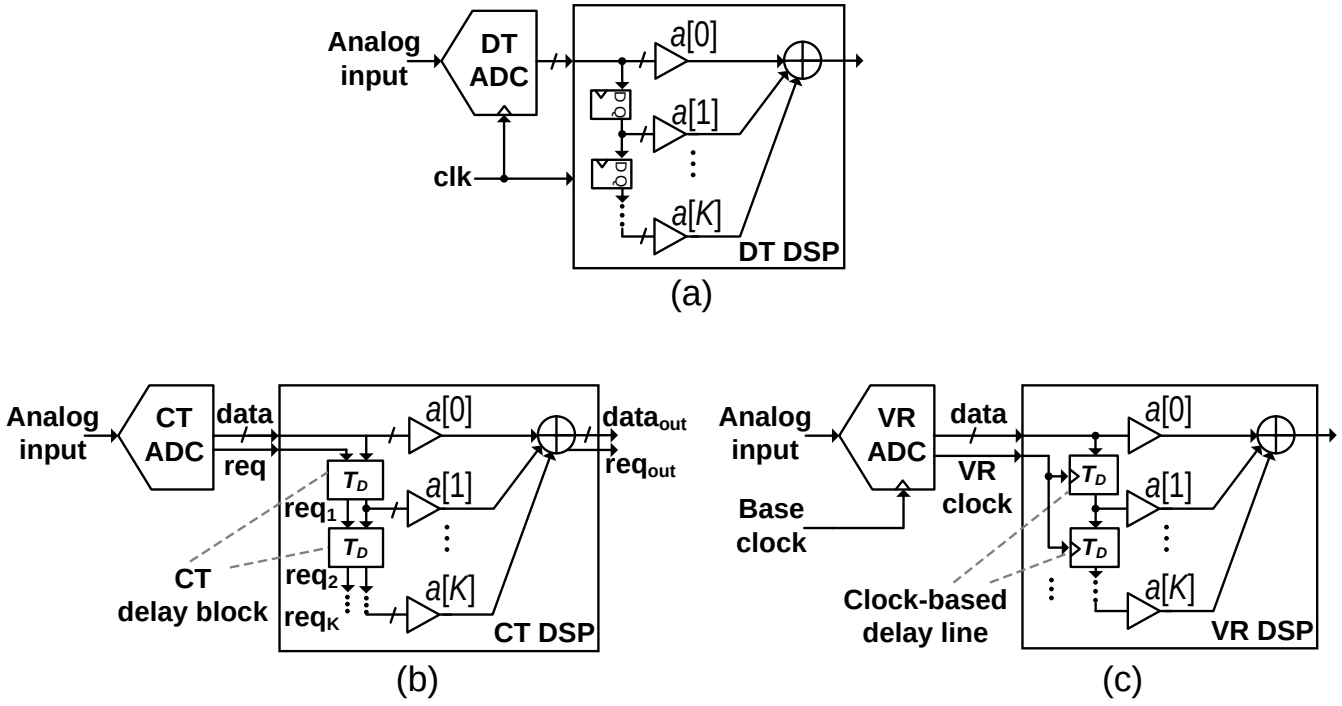


Figure 1.4: Block diagrams of various systems. (a) DT-ADC/DT-DSP. (b) CT-ADC/CT-DSP. (c) VR-ADC/VR-DSP.

tap. Because req_i, $i = 1, 2, \ldots, K$, are delayed versions of *req,* their activities also track the input, as do the operations in the CT-DSP.

The signals at the internal nodes and the output of a CT-DSP are also CT digital. An event at these nodes also means an update of their data values. The completion of the update is indicated by a rising edge on the timing bit *req*. For example, in Fig. 1.4(b), when an input event arrives, it triggers an update at the output of the CT-DSP to reflect the data change at the input. An event is generated there. Note that an event at the output of the CT-DSP can result in a zero change in $data_{\text{out}}$. It is possible that two input events arrive at two different taps of a CT-DSP filter at the same time, but their amplitude changes happen to cancel each other out. For example, when these two input events have the same amount of amplitude change, but see $+1$ and -1 tap coefficients, respectively. They trigger a new event at the CT-DSP output, but the value of $data_{\text{out}}$ is unchanged.

Fig. 1.5 shows one possible implementation of a CT delay block. It is composed of a one-bit CT delay line delaying *req* by T_{D}, and an asynchronous FIFO for temporary data storage [15]. The *data* of a CT digital signal is stored into the FIFO through the write terminals (W) when it first enters the delay block. T_{D} later, the pulses on the timing signals arrive at the end of the one-bit CT delay line – i.e., req_{i+1}. They read out *data* through the read terminals (R). As a result, the block implements a T_{D} delay between $data_{i+1}$ and $data_i$.

The minimum event interval, termed the granularity, is determined by the accompanying CT-ADC. If the CT-ADC is a level-crossing sampler, the granularity is the minimum interval between the instants when an input crosses two successive quantization levels. Normally, it is much smaller than the tap delay T_{D} [3]. So it is very likely that a delay block is delaying multiple events with

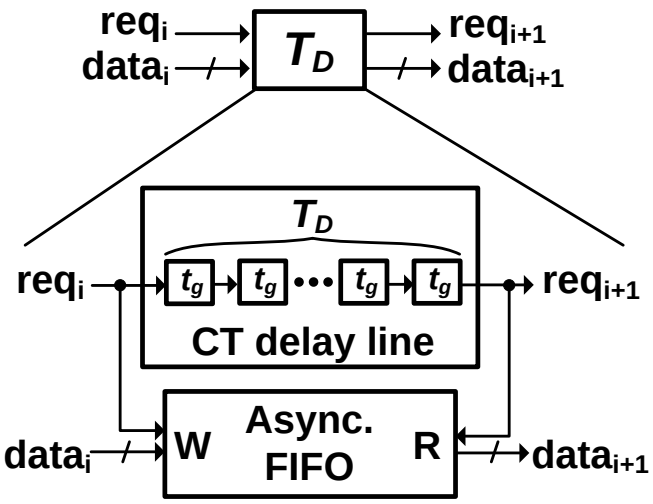


Figure 1.5: The structure of a CT delay block.

close intervals at the same time. In order to properly handle this crowded traffic, the one-bit CT delay line is built with a chain of smaller-delay cells, t_g. The value of t_g must be no larger than the granularity of the input CT digital signal.

A CT-DSP built with the CT delay blocks in Fig. 1.5 can process CT digital signals from various types of analog-to-digital conversions, e.g., pulse-width modulation [4], delta modulation [3], sigma–delta modulation, LCS, etc. In this thesis, we will use LCS as an example, because of its signal-tracking feature. It is a known problem that the output of a LCS can have a high event rate when the analog input moves quickly. Recently developed techniques [5–7] can greatly reduce the event rate without compromising conversion accuracy.

The DSP processing a VR DT digital signal is called a VR-DSP (Fig. 1.4(c)). Although the samples from a VR-ADC have a nonuniform rate, they are aligned to a timing grid defined by the highest-rate base clock. Hence, one can still implement the delay line with a clock and conventional

digital circuits (e.g., DFFs). The challenge in designing an energy-efficient VR-DSP is that its operations should track the input activity. So the operations in a VR-DSP have to rely on the VR clock, rather than the base clock. The key of such design is to adaptively configure the delay lines so that the tap delay is independent of the instantaneous rate of the VR clock. More details about the design of a VR-DSP are provided in Chapter 7.

1.2 Prior art and contributions of this work

The first successful integrated-circuit (IC) implementation of the CT-DSP was published by Schell et al. in [3]: a 16^{th}-order FIR filter for voice applications. [16] provided extensive theoretical analysis for CT digital signal processing, and [17] improved the frequency capability by five orders of magnitude and implemented a CT-DSP for GHz-range applications. However, both [3] and [17] relied on specific coding schemes: delta modulation in [3] and per-edge signal encoding in [17]. This limits the applicability of their designs. [15] implemented a CT-DSP with an open input format. It can accept different types of modulations, both synchronous and asynchronous. For the first time, this design offered a solution to delay a multiple-bit CT digital data in an energy- and hardware-efficient way. It implemented a kHz-range CT-DSP whose frequency response remained intact while processing inputs with different sampling rates.

All the previous attempts on CT-DSPs only resulted in FIR filters. The only work toward a CT IIR filter suffered from high noise and large discrepancies between the measured and designed frequency responses [18]. In addition, the filter order in all previous CT-DSPs is limited by the fact that a large number of power-hungry, area-consuming delay lines, proportional to the filter order, is

needed for their implementation. Finally, their outputs are nonsynchronous and hence incompatible with DT systems.

The work presented in this thesis solves all of these problems. The contribution of this work include

- Spectral analysis of the CT-ADC and -DSP.
- Integrated-circuit implementation of the first CT digital IIR filter.
- A design method using a small, fixed number of delay taps to design high-order CT digital IIR filters.
- A CT-to-DT converter facilitating integration of the CT digital filter with conventional DT systems.
- Analysis of the CT digital IIR filter stability issue in the presence of delay-line mismatches. A practical solution is provided to prevent instability in CT digital IIR filters.
- An event-detection technique that allows a closed-loop CT digital IIR filter's power consumption to track the input activity.
- A VR-DSP solution that can process DT digital signals with a variable sampling rate. The VR-DSP has a frequency response independent of the instantaneous sampling rate, and has a power consumption tracking the input activity.

1.3 Thesis outline

This thesis discusses the design of both CT-DSPs and VR-DSPs, with the majority of the content (Chapters 2–6) focusing on the former, while Chapter 7 focuses on the latter.

Chapter 2 analyzes the spectral feature of CT-ADC and DSP. It analyzes the error introduced in CT amplitude quantization. Spectral features of the quantization error added on band-limited signals are highlighted. The effects of synchronization and time coding on the error spectrum are discussed. Finally, the chapter analyzes the error at the output of a CT-DSP.

Chapters 3–6 discuss the implementation of a CT digital IIR filter system. Chapter 3 describes the design's system-level considerations. Chapter 4 discusses the design of digital delay cells. Chapter 5 gives the circuit implementations of the various blocks. Chapter 6 presents measurement results.

Chapter 7 discusses the design considerations of the VR-DSP. An illustrative example is provided. A reconstruction filter with a variable cutoff frequency is also introduced for output reconstruction.

Chapter 8 offers some suggestions for future work.

Chapter 2

Spectral Analysis of the CT-ADC and -DSP

This chapter analyzes the error introduced in CT amplitude quantization and highlights spectral features of the quantization error added to band-limited signals. The chapter also discusses the effects of synchronization and time coding on the error spectrum and analyzes the error at the output of a CT-DSP. This chapter is based on [19].

2.1 Introduction

The traditional view that quantization amounts to an additive and independent source of white noise has mainly resulted from the culture of DT signals; to produce a DT signal from a CT input, both input sampling and quantization are performed. In contrast to this, in a purely CT analog-to-digital converter (ADC) only quantization occurs [20] (Fig. 2.1). In this chapter, we show how CT amplitude quantization can be analyzed as a memoryless (nonlinear) transformation of continuous

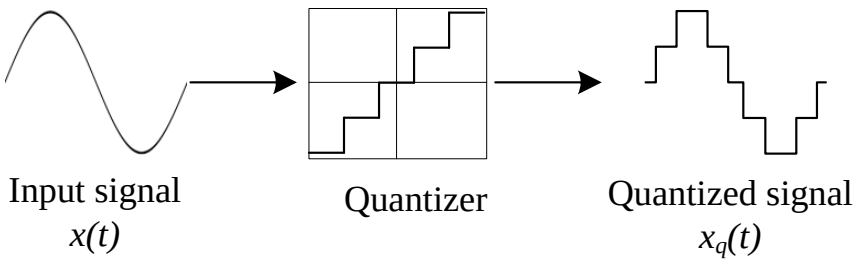


Figure 2.1: A CT amplitude quantization operation.

waveforms. This yields harmonic-based, rather than noise oriented, spectral results. The resulting spectrum is discussed in Section 2.2.

While purely CT signal processing has advantages [20], one still needs to consider and evaluate the effects of an eventual sampling of the signals thus obtained for interface compatibility with standard data processing (including storage, transmission and off-line computation). Section 2.3 addresses this issue.

As mentioned in [20], one can store the nonuniformly spaced event times t_k by finely quantizing the time axis, which results in "pseudocontinuous" operation. A basic question when doing this is how to choose the resolution of time quantization such that the accuracy of the resulting signal is comparable with that of a purely CT system. Section 2.4 discusses this problem. Finally, the results obtained for CT quantizers are extended to the output of a CT-DSP in Section 2.5.

2.2 CT amplitude quantization

CT amplitude quantization (Fig. 2.1) converts the input $x(t)$ to a piecewise constant signal $x_q(t)$ and allows for further digitizing. Basically, $x_q(t)$ is an instantaneous transformation of $x(t)$, which

can be expressed as

$$x_q(t) = Q(x(t)), \tag{2.1}$$

where Q is the scalar function of quantization. The error signal $e(t) = x_q(t) - x(t)$ is then also a memoryless function of $x(t)$,

$$e(t) = E(x(t)), \tag{2.2}$$

where E is the error function of Q defined by

$$E(x) = Q(x(t)) - x(t). \tag{2.3}$$

The above error signal, $e(t)$, is defined with the implicit assumptions that it does not contain a signal component or a constant, and the quantization does not add delay. In the more general case, the error must be defined as follows. The mean square error (MSE) is obtained by a minimization algorithm over a, b and τ, which compensates for the gain, DC offset and time delay of the quantization, respectively:

$$MSE = \min_{a,b,\tau} \left\{ \overline{\left(x_q(t) - ax(t-\tau) - b\right)^2} \right\}. \tag{2.4}$$

Once the three parameters are determined, the error signal can then be defined as

$$e(t) = x_q(t) - ax(t-\tau) - b. \tag{2.5}$$

For simplicity, this chapter assumes that quantizers have a unity gain, zero DC shift and zero time delay, so that $e(t) = x_q(t) - x(t)$. However, the principles presented are also valid for other cases.

After reviewing basic knowledge on the error function, $E(x)$, of standard scalar quantizers, we will tackle the spectral analysis of $e(t)$ when $x(t)$ is a single tone and then when it is composed of two tones.

2.2.1 Quantization characteristics

Quantization consists of approximating an input number x by the nearest value from a predetermined finite set of quantum levels. In general, these levels are not necessarily uniformly spaced, as is the case in signal companding. In this chapter, we concentrate on the simpler but standard case of uniform spacing. Fig. 2.2(a–b) shows the two typical transfer functions, Q, of uniform scalar quantizers [21]. They are both staircase functions. The midrise quantizer of Fig. 2.2(a) has a discontinuity at $x = 0$. Meanwhile the midtread quantizer of Fig. 2.2(b) has a quantum level at $x = 0$ and is preferred in event-driven systems, as it can tolerate small input imperfections around zero (DC offset, noise, etc.) and maintain a zero output in the absence of a signal. An odd number of quantum levels is however necessary to achieve symmetry around the origin. With a quantization step size Δ and an N-bit resolution of quantum levels, the single-ended full-scale amplitude of the midtread quantizer is

$$A_{\mathrm{FS}} = \left(2^{N-1} - \frac{1}{2}\right)\Delta,$$

whereas it is $2^{N-1}\Delta$ with the midrise quantizer.

Fig. 2.2(c–d) shows the corresponding error functions, $E(x)$. They are both odd periodic saw-

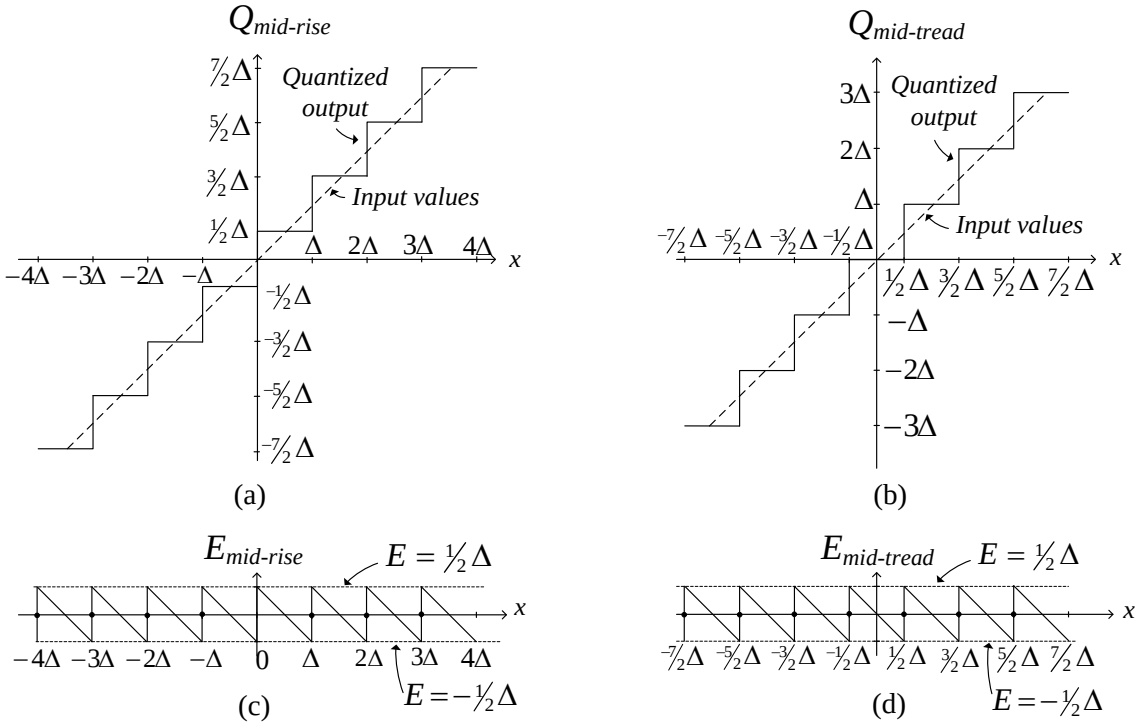


Figure 2.2: Quantization characteristics and error functions. (a) Characteristic of midrise quantizer. (b) Characteristic of midtread quantizer. (c) Quantization error of midrise quantizer. (d) Quantization error of midtread quantizer.

tooth functions of period Δ. When x is a random variable that is uniformly distributed over a range of a multiple of Δ in length, $E(x)$ is known to be uniformly distributed between $-\frac{\Delta}{2}$ and $\frac{\Delta}{2}$ [22]. In this case, $E(x)$ yields the classical variance value of $\frac{\Delta^2}{12}$.

When the above statistical assumption on the input is not satisfied, one must resort to a deterministic analysis of $E(x)$. As a periodic function, $E(x)$ can be expanded in a Fourier series. With midtread quantizers, the expansion is [23, 24]

$$E(x) = \Delta \sum_{k=1}^{\infty} (-1)^k \frac{\sin(2k\pi x/\Delta)}{k\pi}. \tag{2.6}$$

Removing $(-1)^k$ from this equation yields the series corresponding to midrise quantizers. In the

following discussion however, we will focus on the midtread characteristic and choose by default the error expansion of (2.6).

2.2.2 Single-tone input

We now analyze the quantization error signal when $x(t)$ is a single-tone (sinusoidal) input. Fig. 2.3 shows both the quantized output $x_q(t) = Q(x(t))$ and the quantization error signal $e(t) = E(x(t))$ for a six-bit midtread quantizer with a full-scale single-tone input. As illustrated in part (c) of this figure, the analysis of $e(t)$ can be divided into three different regions [25].

1. The bell-like pulses formed by the quantization of the sine wave around its peaks and troughs
2. The sawtooth patterns arising from the quantization of the sine wave around its zero-crossings, where it is close to an ideal ramp
3. The transition parts between the above two regions

Although the amplitude values of a sinusoid are not uniformly distributed [26] (maximal density values being obtained at the extreme values of the sinusoid, as expected from Fig. 2.3(c)), the probability distribution of $e(t)$ is still modeled as uniform in $\left[-\frac{\Delta}{2}, \frac{\Delta}{2}\right]$ [22]. In the following, "power," denoted by P, refers to a mean-square value. With a full-scale sinusoidal input, the total error power of $\frac{\Delta^2}{12}$ resulting from this model appears to be satisfactorily accurate in practice. Given the power of a full-scale sinusoidal input of $\frac{A_{\text{FS}}^2}{2}$, the signal-to-error ratio (*SER*) of the quantized output is

$$SER = \frac{P_{\text{signal}}}{P_{\text{error}}} = \frac{A_{\text{FS}}^2/2}{\Delta^2/12}.$$

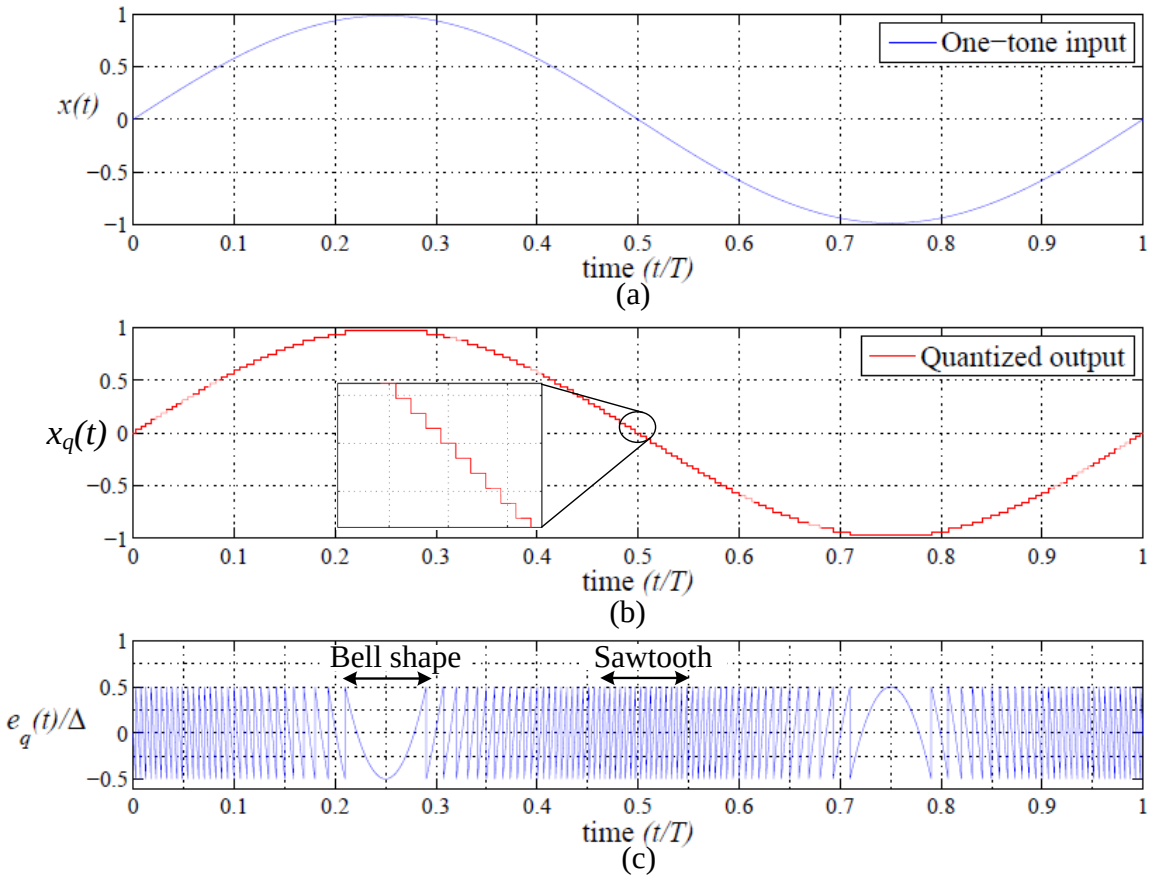


Figure 2.3: A quantized sinusoid waveform and its quantization error. The quantizer has a six-bit resolution. (a) A full-scale sinusoidal input; (b) the quantized output waveform (the piecewise-constant waveform is more clearly shown in the zoomed-in plot.); (c) the quantization error signal. It's amplitude is normalized to the quantization step Δ.

With an N-bit midtread quantizer, $A_{\text{FS}} = (2^{N-1} - \frac{1}{2})\Delta$. Thus, the *SER* in decibels yields the expression

$$SER_{\text{dB}} = 10\log_{10}\frac{\left((2^{N-1} - 1/2)\Delta\right)^2/2}{\Delta^2/12} \approx 6.02N + 1.76\,\text{dB},$$

where the approximation assumes $2^N \gg 1$.

In the frequency domain, the bell-like pulses and the sawtooth waveform have very different contributions to the spectrum. Fig. 2.4 shows the spectrum of the error signal, $e(t)$, in Fig. 2.3(c).

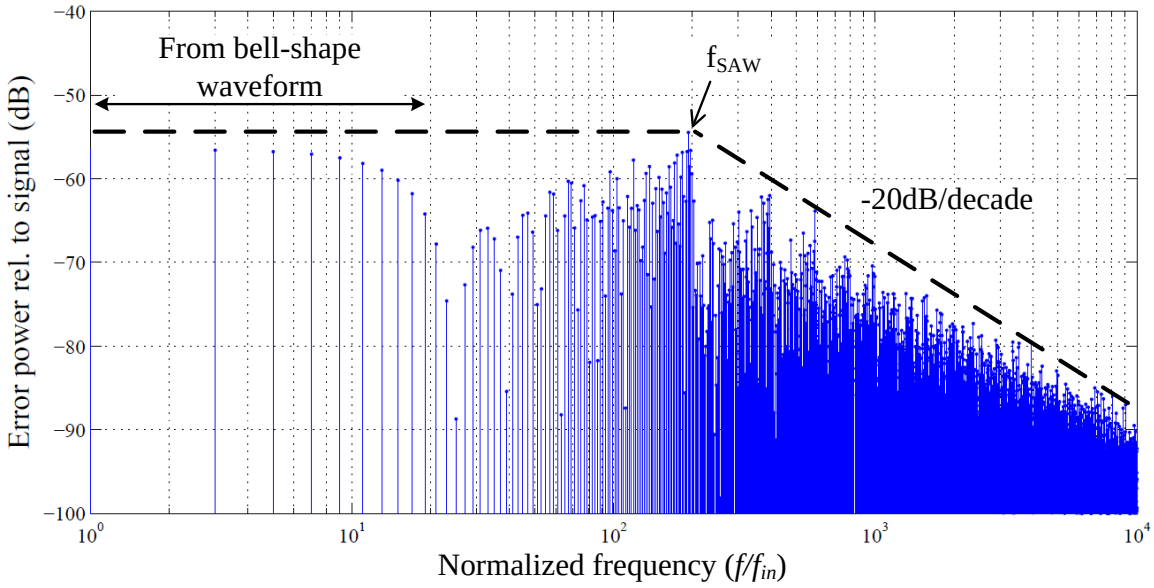


Figure 2.4: Power spectrum of the quantization error in Fig. 2.3(c) relative to signal power. The error signal is obtained by quantizing a full-scale sinusoidal signal with a six-bit midtread quantizer. The x-axis shows the frequency normalized to f_{in} and is plotted on a logarithmic scale.

Since $e(t)$ is periodic with the same period as the input, its spectrum is composed of harmonics, with the first harmonic at the input frequency. Since the quantization error function $E(x)$ is odd-symmetric around the origin and the input signal $x(t)$ has so-called "half-wave symmetry," the corresponding error signal, $e(t)$, also has "half-wave symmetry," and thus the resulting discrete spectrum contains only odd-order harmonics. The bell-like pulses are periodic in the time domain at the input frequency and contribute mostly low-order harmonics. [27] observed that small variations in the amplitude of the input may cause large changes in the bell-like pulses. Thus, the power of these low-order harmonics is very sensitive to the input amplitude. The fast-varying sawtooth part contributes mostly to the high-frequency part of the error power. Let the minimum sawtooth duration, T_{SAW}, be the shortest time for the input to cross a quantization interval, and define the sawtooth frequency, f_{SAW}, as its inverse. For a band-limited input, T_{SAW} is approximately $\frac{\Delta}{\left|\frac{dx}{dt}\right|_{\max}}$,

which implies that

$$f_{\mathrm{SAW}} = \frac{\left|\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{\max}}{\Delta}. \tag{2.7}$$

For an input of the form $x(t) = A_{\mathrm{in}} \sin(2\pi f_{\mathrm{in}} t)$,

$$f_{\mathrm{SAW}} = \frac{2\pi A_{\mathrm{in}} f_{\mathrm{in}}}{\Delta}. \tag{2.8}$$

With a full-scale input, $A_{\mathrm{in}} = \left(2^{N-1} - \frac{1}{2}\right)\Delta$ and hence

$$f_{\mathrm{SAW}} = 2\pi\left(2^{N-1} - \frac{1}{2}\right) f_{\mathrm{in}} \approx 2^N \pi f_{\mathrm{in}} \tag{2.9}$$

The last approximation assumes $2^N \gg 1$, which is satisfied in most practical cases. In the given example, $N = 6$. This leads to $f_{\mathrm{SAW}} = 201 f_{\mathrm{in}}$ (indicated in Fig. 2.4). The spectrum beyond f_{SAW} contains the harmonics of the fundamental components below f_{SAW}. A qualitative observation is in order. In Fig. 2.4, one can observe that the amplitude of the components up to f_{SAW} can roughly be thought to have a constant envelope, while the amplitude of the components beyond f_{SAW} can be seen to drop off, on average, with a -20 dB/decade slope. The reason for the value of this slope will be discussed below.

Assume the single-tone input has a full-scale amplitude A_{FS}, an arbitrary input frequency f_{in} and an arbitrary phase shift ϕ:

$$x(t) = A_{\mathrm{FS}} \sin(2\pi f_{\mathrm{in}} t + \phi). \tag{2.10}$$

By substituting (2.10) into (2.6), the quantization error becomes

$$e(t) = \Delta \sum_{k=1}^{\infty} (-1)^k \frac{1}{\pi k} \sin\left(\frac{2\pi k A_{\mathrm{FS}}}{\Delta} \sin\left(2\pi f_{\mathrm{in}} t + \phi\right)\right). \tag{2.11}$$

A complete Fourier series expansion of $e(t)$ can then be obtained using Bessel functions and Jacobi-Anger expansion [24, 28, 29]. This process is involved and thus is not included in this chapter. We only highlight some important results. The term of (2.11) corresponding to $k = 1$ is

$$e_1(t) = -\frac{2\Delta}{\pi} \sum_{m=1, m\,\mathrm{odd}}^{\infty} J_m\left(\frac{2\pi A_{\mathrm{FS}}}{\Delta}\right) \sin\left[m(2\pi f_{\mathrm{in}} t + \phi)\right],$$

where $J_m(.)$ is the Bessel function of the first kind of order m. This is a sum of all the odd-order harmonics of the input frequency with amplitudes $J_m\left(\frac{2\pi A_{\mathrm{FS}}}{\Delta}\right)\frac{2\Delta}{\pi}$. According to [28],

$$J_m(\beta) \approx 0,\ m > \beta + 1. \tag{2.12}$$

Thus, $J_m\left(\frac{2\pi A_{\mathrm{FS}}}{\Delta}\right) = J_m\left(2\pi\left(2^{N-1} - \frac{1}{2}\right)\right)$ is negligible for $m > 2^N\pi$. In other words, all the significant tones of $e_1(t)$ have frequencies below $2^N\pi f_{\mathrm{in}}$, which is exactly the f_{SAW} defined in equation (2.9).

The k^{th} order term of $e(t)$ in (2.11) can be expressed as

$$e_k(t) = (-1)^k \frac{2\Delta}{k\pi} \sum_{m=1, m\,\mathrm{odd}}^{\infty} J_m\left(\frac{2k\pi A_{\mathrm{FS}}}{\Delta}\right) \sin\left[m(2\pi f_{\mathrm{in}} t + \phi)\right].$$

This has the same form as $e_1(t)$ with a factor of $1/k$ in front of the summation and a factor of k inside the argument of the Bessel function. The $(-1)^k$ in front does not affect the amplitude. Using again (2.12), we find the significant tones of $e_k(t)$ are present only up to kf_{SAW}. The frequency range $[(k-1)f_{\mathrm{SAW}}, kf_{\mathrm{SAW}}]$, which we call the k^{th} f_{SAW} band, contains the significant tones from the k^{th} and higher order terms only, those resulting from $\sum_{i=k}^{\infty} e_i(t)$. However, because of the scaling factor $1/k$, the power of the tones within each k^{th} f_{SAW} is mostly determined by the term of the lowest order $e_k(t)$. They are approximately k^2 times lower than the tones within the first f_{SAW} band (i.e., $[0, f_{\mathrm{SAW}}]$). This result is better observed on the plot of Fig. 2.5, plotted using a linear frequency axis. A staircase-like spectrum is observed. Each step has the same width, f_{SAW}. Also, the average power in each k^{th} f_{SAW} band is k^2 lower than the tones in the first f_{SAW} band. This is consistent with the observed slope of $-20\,\mathrm{dB/decade}$ indicated on the logarithmic scale of Fig. 2.4. Fig. 2.5 presents zoomed-in plot around f_{in} to more clearly show the discrete feature of the spectrum. Only harmonics are present, while no component exists at any in-between frequency.

Power within $[0, f_{\mathrm{SAW}}]$

It was observed in [27] that the error power below f_{SAW} dominates the entire quantization error power. The reason is that the tones beyond f_{SAW} come from the higher order terms of equation (2.11) and thus their power is reduced by $1/k^2$. From (2.11), it can be found numerically that 71% of the total error power lies in the band $[0, f_{\mathrm{SAW}}]$. It is also interesting to estimate this number using the staircase model of Fig. 2.5. According to this rough model, the power spectral density in the k^{th} f_{SAW} band is assumed flat and proportional to $1/k^2$. As the error tones are uniformly spaced

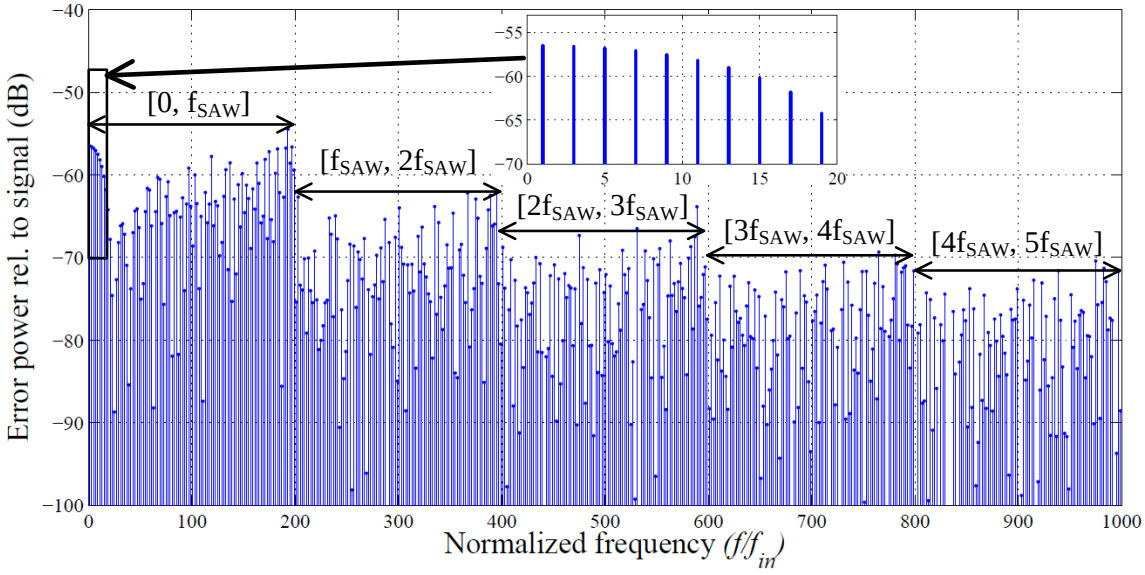


Figure 2.5: Power spectrum of the quantization error in Fig. 2.3(c) relative to signal power. The error signal is obtained by quantizing a full-scale sinusoidal signal with a six-bit midtread quantizer. The x-axis shows the frequency normalized to f_{in} and is plotted on a linear scale. A zoomed-in plot shows a narrow spectrum around f_{in} (i.e., $[0, 20f_{\text{in}}]$).

(with a distance of $2f_{\text{in}}$), the error power within the k^{th} f_{SAW} band decreases in $1/k^2$ with k. The ratio of power that lies in $[0, f_{\text{SAW}}]$ is then $1/\sum_{k=1}^{\infty} k^2 = 1/\frac{\pi^2}{6} \simeq 61\%$, comparable to the numerical result above.

2.2.3 Two-tone input

Assume now that the input is a two-tone signal, $x(t) = A_{\text{in1}} \sin(2\pi f_{\text{in1}} t) + A_{\text{in2}} \sin(2\pi f_{\text{in2}} t)$, assuming zero phase between the two. The sawtooth frequency, f_{SAW}, can still be derived from (2.7):

$$f_{\text{SAW}} = \frac{2\pi A_{\text{in1}} f_{\text{in1}} + 2\pi A_{\text{in2}} f_{\text{in2}}}{\Delta}.$$

By defining the weighted-averaged frequency

$$f_{\text{in,avg}} = \frac{A_{\text{in1}} f_{\text{in1}} + A_{\text{in2}} f_{\text{in2}}}{A_{\text{FS}}},$$

one obtains the equivalent expression

$$f_{\text{SAW}} = \frac{2\pi A_{\text{FS}} f_{\text{in,avg}}}{\Delta},$$

which coincides with f_{SAW} of a full-scale single-tone input of frequency $f_{\text{in,avg}}$ according to (2.8). Fig. 2.6(a) plots the spectrum of the quantization error, $e(t)$, resulting from this input in the case where $A_{\text{in1}} = A_{\text{in2}} = \frac{A_{\text{FS}}}{2}$ and $N = 6$. The spectrum now contains not only harmonics, but also intermodulation products. The tones are uniformly spaced by a distance of $\Delta f = f_{\text{in2}} - f_{\text{in1}}$ in frequency. Given the bit resolution $N = 6$, $f_{\text{SAW}} = \pi\left(2^{N-1} - \frac{1}{2}\right) = 201 f_{\text{in,avg}}$. This value is consistent with the value of f_{SAW} obtained from the graphical method by finding the transition frequency between the flat and descending parts of the spectrum. Such agreement will also be observed with more complicated inputs in Section 2.2.4. It can be found numerically that the power below the maximum sawtooth frequency in this two-tone example is 78% of the total quantization-error power.

The total error power is about $\frac{\Delta^2}{12}$, which results in a $SER = 34.8\,\text{dB}$ – i.e., $10\log_{10}\frac{A_{\text{in1}}^2/2 + A_{\text{in2}}^2/2}{\Delta^2/12}$. However, if a narrow baseband is considered, only a few harmonics and intermodulation components fall in band and the resulting in-band *SER* can be much higher. Fig. 2.6(b) shows an example baseband, $[0, 2f_{\text{in,avg}}]$. A limited number of tones are evenly distributed within the baseband, with no component exists at any other frequency. The in-band *SER* can be found by summing the rel-

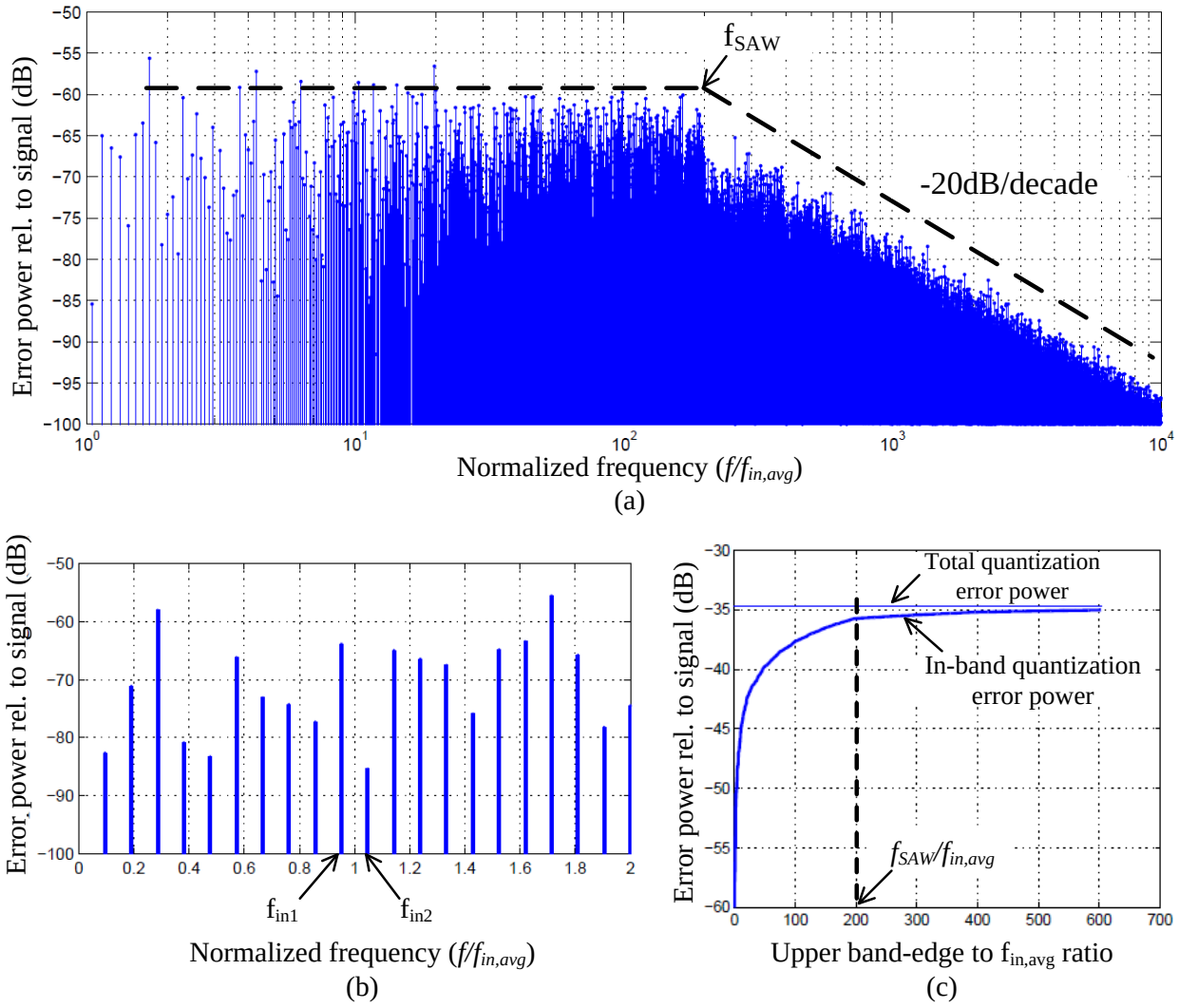


Figure 2.6: Quantization error relative to signal power in the two-tone test. The quantizer has a six-bit resolution. Full load of the quantizer is achieved by setting the amplitudes of the two input tones as half of A_{FS}. (a) A wide spectrum of the quantization error plotted on a logarithmic scale. The frequency axis is normalized to $f_{in,avg}$. (b) A narrow baseband spectrum of the quantization error on linear scale. The frequency axis is normalized to $f_{in,avg}$ and shows the range from 0 to 2. The locations of the two input tones are shown. (c) In-band quantization error power relative to signal power as a function of the ratio of the upper band-edge frequency to $f_{in,avg}$.

ative power of all these tones, which is 51 dB. The choice of baseband is rather arbitrary in this two-tone case, where the input signal does not occupy a full band. We can choose a baseband with almost any width and find the corresponding in-band *SER*. Fig. 2.6(c) shows the in-band quantization error power as a function of the ratio of baseband's upper band-edge frequency to $f_{\text{in,avg}}$. As the band-edge moves to high frequencies, more and more harmonics and intermodulation tones fall inside the baseband, increasing the resulting in-band error power. Although the curve is monotonically increasing, its rate of change drops considerable once the band-edge is larger than f_{SAW}. This is because the error tones beyond f_{SAW} contain little power. The horizontal line on the top shows the total quantization error power relative to the signal power. It is the ultimate value of the relative error power when the baseband is infinitely wide.

2.2.4 Band-limited Gaussian inputs

We now consider a more complex class of signals, namely band-limited signals with a Gaussian amplitude probability density function. The probability of such a random input overloading the quantizer is negligible (less than 1 in 10,000) as long as its root-mean-square amplitude, A_{RMS}, is less than a quarter of the quantizer's full-scale range A_{FS} [30, 31]. The quantization-error spectrum of this signal has features similar to one- and two-tone inputs: 1) It has a corner frequency, which we denote by $f_{\text{SAW,Gaussian}}$, and 2) beyond $f_{\text{SAW,Gaussian}}$, the spectrum follows a -20 dB/decade slope. $f_{\text{SAW,Gaussian}}$ can be calculated by averaging the f_{SAW} contributed by each input tone [27]:

$$f_{\text{SAW,Gaussian}} = \overline{f_{\text{SAW}}} = \frac{2\pi A_{\text{RMS}} \overline{f_{\text{in}}}}{\Delta}.$$

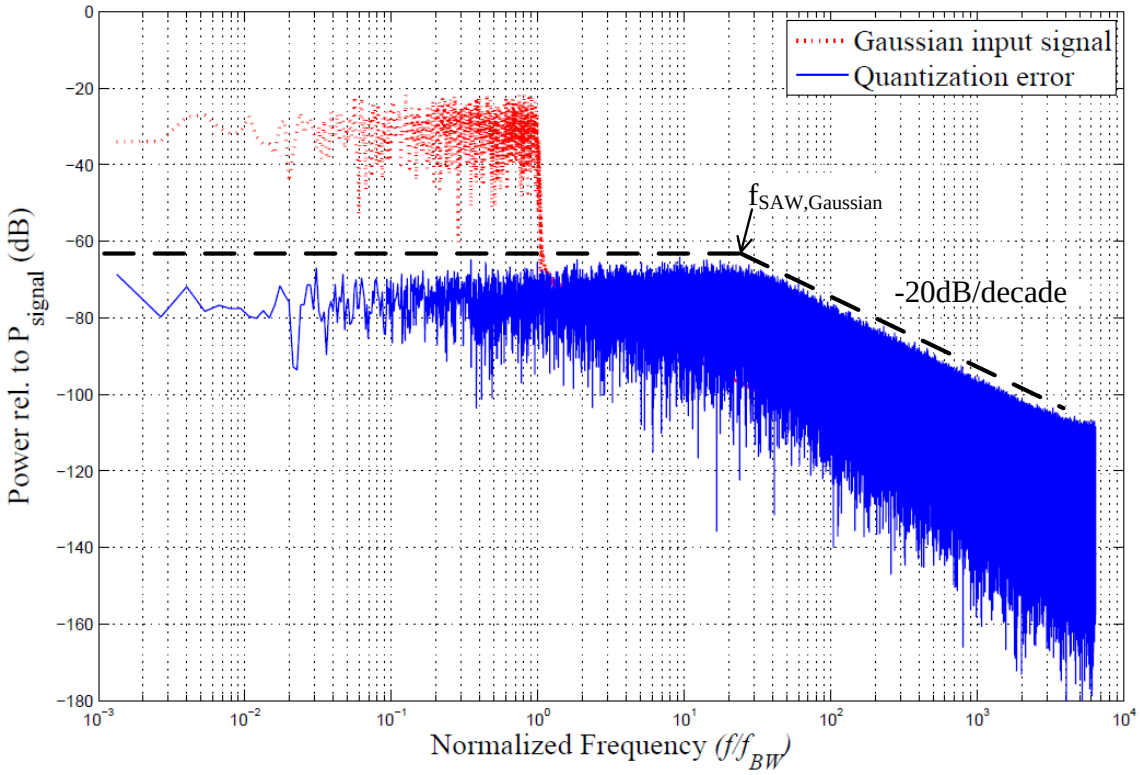


Figure 2.7: Power spectra in the band-limited Gaussian input case. The root-mean-square amplitude, A_{RMS}, is set as one fourth of A_{FS}. The quantizer has a six-bit resolution. The x-axis shows the frequency normalized to f_{BW} and is plotted on a logarithmic scale. The red dotted line is the power spectrum of the input signal. It is generated by low-pass filtering a Gaussian white noise with a cutoff frequency of 500 Hz. The solid line shows the power spectrum of the quantization error. Both power values are normalized to P_{signal}, the total power of the input signal.

Calling f_{BW} the maximum input frequency, we obtain $f_{\mathrm{SAW,Gaussian}} = \pi A_{\mathrm{RMS}} f_{\mathrm{BW}}/\Delta$.

Fig. 2.7 shows an example spectrum of the quantization error (solid line). In contrast to the one- and two-tone cases, the spectrum is continuous. A_{RMS} is chosen to be $A_{\mathrm{FS}}/4$ and thus $f_{\mathrm{SAW,Gaussian}} = \pi 2^{N-3} f_{\mathrm{BW}}$. For $N = 6$, $f_{\mathrm{SAW,Gaussian}} \approx 25 f_{\mathrm{BW}}$. This is consistent with the plot of the quantization error. Beyond $f_{\mathrm{SAW,Gaussian}}$, a -20 dB/decade slope is apparent. Integrating the error power over the entire frequency axis, we find the *SER* equals to 29 dB. In the same figure, the spectrum of the band-limited Gaussian input is also plotted (dotted line). It's power spectrum is white up to f_{BW}.

Beyond that, its power drops significantly (due to filtering). In this band-limited Gaussian case, the signal band is well defined as $[0, f_{\mathrm{BW}}]$. The bandwidth of the quantization error is much wider than that of the input signal. Integrating the error power within the signal band, we find the in-band *SER* to be 45 dB, which is 16 dB higher than the *SER* calculated over the entire frequency axis.

To summarize Section 2.2, f_{SAW} plays an important role in the spectral characterization of any CT quantizations of band-limited signals. The value of this quantity can be obtained from equation (2.7). Within $[0, f_{\mathrm{SAW}}]$, the error power spectral density is almost flat. Beyond f_{SAW}, the spectral density drops with a slope of -20 dB/decade. The frequency range $[0, f_{\mathrm{SAW}}]$ contains the majority of the total power of the quantization error [27, 32], roughly in the range of 70% to 80%.

2.3 Uniform sampling of quantized CT signals

2.3.1 Uniform sampling context and issues

While CT amplitude-quantized signals are free from aliasing, the step length in their piecewise-constant waveforms varies, preventing their simple integration into standard systems operating under a fixed clock rate. Resolving this issue requires a uniform sampling of the CT outputs. This inevitably brings back the issue of aliasing. In fact, because CT quantization is an instantaneous operation, permuting the order of amplitude quantization and time sampling gives the same result [33]. Indeed, the sample of the amplitude-quantized signal $x_q(t) = Q(x(t))$ at an instant t_k is $x_q(t_k) = Q(x(t_k))$, which is equal to the quantized version of the sample of $x(t)$ at t_k. This makes the system equivalent to a conventional ADC involving sampling and quantization because the two op-

erations can be interchanged. In spite of this system equivalence, there are still two strong reasons for using CT systems. 1) The major part of the hardware in a CT system is event-driven and thus dissipates power more efficiently compared to clocked systems. 2) Using oversampling to improve the *SER* is straightforward with CT systems. Only the sampling clock frequency of the output synchronizer needs to be increased. In conventional systems on the other hand, oversampling requires increasing the clock rate of the entire signal-processing chain.

2.3.2 Analysis of aliased quantization error – a staircase model

Next, we investigate how aliasing degrades signal accuracy when a CT-quantized signal is sampled. It is usually believed that in-band quantization error can be arbitrarily reduced by increasing sampling frequency. We will see that there is a limit to this error reduction.

Assume a band-limited input, $x(t)$, of maximum frequency f_{BW} is quantized and then uniformly sampled at a frequency $f_{\mathrm{s}} \geq 2f_{\mathrm{BW}}$. The sampled signal includes an quantization error component:

$$e_{\mathrm{s}}(t) = \sum_{k=-\infty}^{\infty} e(nT_{\mathrm{s}})\,\delta(t-kT_{\mathrm{s}}),$$

where $e_{\mathrm{s}}(t)$ is the CT quantization error signal and $T_{\mathrm{s}} = \frac{1}{f_{\mathrm{s}}}$. We wish to evaluate the power of $e_{\mathrm{s}}(t)$ that lies in the input baseband, $[0, f_{\mathrm{BW}}]$. Denoting by $E(f)$ the Fourier transform of $e(t)$, we have

$$E_{\mathrm{s}}(f) = E(f) * \sum_{n=-\infty}^{\infty} \delta(f - nf_{\mathrm{s}}). \tag{2.13}$$

As a result, the in-band error power is the sum of the error power in a bandwidth of $\pm f_{\mathrm{BW}}$

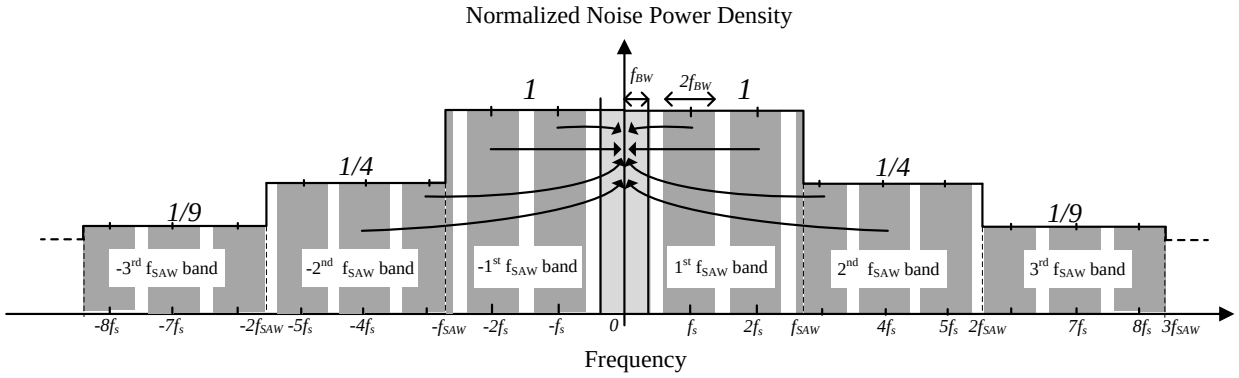


Figure 2.8: Staircase-modeled error spectrum before aliasing. The height represents the power spectrum density and is normalized to the value of the first f_{SAW} band. The dark parts represent the bands, which will be shifted into the baseband after sampling. In this example, f_{s} is assumed to be a fraction of f_{SAW}; thus, each f_{SAW} band contains multiple dark parts.

around each harmonic of the sampling frequency f_{s} [27]. This convolution operation is visually explained in Fig. 2.8. As was explained in Section 2.2, $|E(f)|^2$ along the frequency axis can be reasonably modeled as a staircase function whose steps correspond to the f_{SAW} bands of successive orders. The heights of the steps represent the power spectrum density and decrease as $1/k^2$. The dark areas represent the frequency bands of width $2f_{\mathrm{BW}}$ centered around the multiples of f_{s} (excluding the zero multiple). As (2.13) indicates, after $e(t)$ is sampled at frequency f_{s}, all the dark parts are shifted into the baseband $[-f_{\mathrm{BW}}, f_{\mathrm{BW}}]$ and added to its original power.

Assuming the aliased components are independent, we can estimate the total aliased error power, P_{al}, from the double-sided spectra:

$$P_{\mathrm{al}} \approx 2\sum_{k=1}^{\infty} N_k \frac{P_{\mathrm{bb}}}{k^2}, \tag{2.14}$$

where P_{bb} is the original error power within the baseband before aliasing and N_k is the number of

positive multiples of f_s that fall in the k^{th} f_{SAW} band. The factor of two is due to the two-sided spectrum: In signed frequency, the k^{th} f_{SAW} band includes the interval $[(k-1)f_{\text{SAW}}, kf_{\text{SAW}}]$ and its negative counterpart. The total quantization error power in $e_s(t)$ is

$$P_{\text{total}} = P_{\text{al}} + P_{\text{bb}}. \tag{2.15}$$

In (2.14), P_{bb} is solely dependent on the input and the quantizer. Meanwhile, N_k depends on the ratio of $\frac{f_s}{f_{\text{SAW}}}$.

When $\frac{f_s}{f_{\text{SAW}}} \ll 1$, the aliased error power P_{al} is dominated by the part coming from the first f_{SAW} band where power density is the highest. Thus, equation (2.15) can be reduced to

$$P_{\text{total}} \approx 2N_1 P_{\text{bb}} + P_{\text{bb}}.$$

With $N_1 \gg 1$ (which is true in most oversampling scenarios), it can be further reduced to

$$P_{\text{total}} \approx 2N_1 P_{\text{bb}}.$$

The worst value of SER_{dB}, denoted by $SER_{\text{dB,worst}}$, is achieved at Nyquist sampling where the entire error power is aliased into the baseband. As long as $\frac{f_s}{f_{\text{SAW}}} \ll 1$, doubling f_s halves N_1 and thus decreases P_{total} approximately by a factor of 2. A 3 dB improvement in SER_{dB} is expected.

When $\frac{f_s}{f_{\text{SAW}}} \gg 1$, P_{al} results from high-order f_{SAW} bands, and is consequently negligible com-

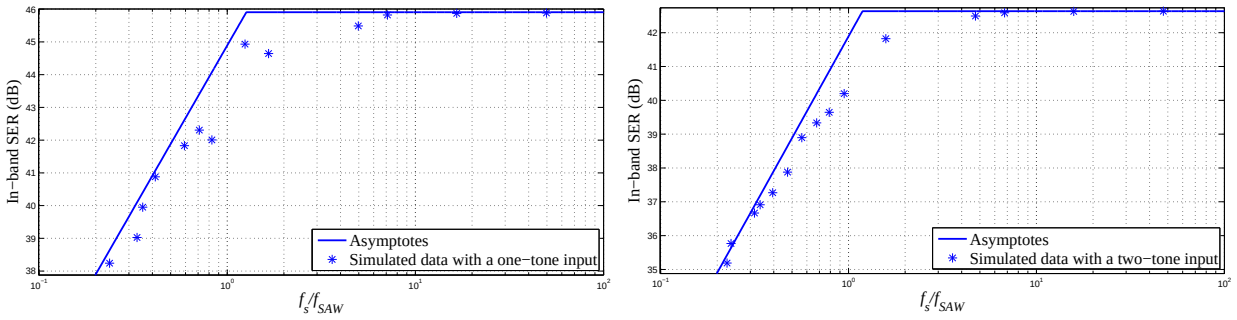
pared to P_{bb}. In this case, equation (2.15) reduces to

$$P_{\text{total}} \approx P_{\text{bb}}.$$

The error power becomes independent of the sampling frequency and SER_{dB} reaches its highest value, denoted by $SER_{\text{dB,best}}$. At infinite oversampling, the result converges to the CT-quantization error power. P_{total} *does not* go to zero, as one would have assumed by applying the "3 dB reduction for each doubling of f_{s}" rule.

As this analysis shows, the plot of SER_{dB} versus $\frac{f_{\text{s}}}{f_{\text{SAW}}}$ has two asymptotes. As shown in Fig. 2.9. For $\frac{f_{\text{s}}}{f_{\text{SAW}}} \ll 1$, the plot asymptotically approaches a straight line of slope 10 dB/decade (i.e., 3 dB/octave), starting from the lowest point of $SER_{\text{dB,worst}}$. For $\frac{f_{\text{s}}}{f_{\text{SAW}}} \gg 1$, SER_{dB} approaches $SER_{\text{dB,best}}$.

Fig. 2.9 also shows the results of Matlab experiments to verify the staircase model. One-tone and two-tone inputs are used with a six-bit quantizer. The measured points satisfactorily reproduce the asymptotic trends in the regions $\frac{f_{\text{s}}}{f_{\text{SAW}}} \ll 1$ and $\frac{f_{\text{s}}}{f_{\text{SAW}}} \gg 1$, with, however, some local deviations that can be attributed to three factors: The model assumes a flat spectrum within each f_{SAW} band, which is not exactly true. The assumption implied by the power additivity in (2.14) – i.e., that all the aliased components are independent – is not rigorously satisfied. Finally, (2.14) assumes that each band $[nf_{\text{s}} - f_{\text{BW}}, nf_{\text{s}} + f_{\text{BW}}]$ entirely falls into one f_{SAW} band of order k. Fig. 2.8 shows that this latter assumption fails when nf_{s} is close to the boundaries of an f_{SAW} band (e.g., $n = 3$).



(a) One-Tone Test (b) Two-Tone Test

Figure 2.9: Theoretical asymptotes and simulation results of SER_{dB} using the staircase model. Both cases use a six-bit midtread quantizer. (a) One-tone test with $f_{\mathrm{BW}} = 20f_{\mathrm{in}}$. (b) Two-tone test with $f_{\mathrm{BW}} = 20f_{\mathrm{in,avg}}$.

2.4 CT-ADC with time coding

The event times t_k in CT-ADCs are nonuniformly spaced and can have arbitrary values. Time coding (a term adopted from [34]) is necessary when one wants to store the nonuniform samples [20]. A high-frequency clock can be used to quantize the time intervals between any two successive events $\Delta_k = t_k - t_{k-1}$. An interesting question is how high the clock frequency should be. To answer this question, we note that the output of such a CT-ADC with time coding is equivalent to a clocked ADC with the same sampling frequency; thus, the staircase model in Fig. 2.8 applies. According to the previous result that the aliased error power becomes negligible when $\frac{f_{\mathrm{s}}}{f_{\mathrm{SAW}}} \gg 1$, the time discretization needs to be sufficiently fine so that the in-band *SER* is comparable to the CT case with no sampling. Thus, the sampling frequency has to be several times higher than f_{SAW} to ensure the highest in-band *SER*.

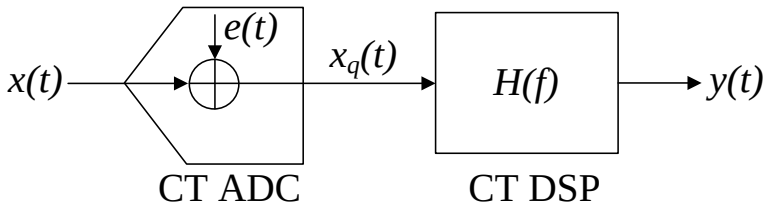


Figure 2.10: A CT signal processing system composed of a CT-ADC and a CT-DSP.

2.5 Error analysis of a CT-DSP

The error at the output of a CT-DSP is analyzed next. Fig. 2.10 shows a signal processing system composed of a CT-ADC and a CT-DSP. Mathematically, the CT-ADC (the quantizer) can be represented as a summer with an additional input $e(t)$, which is the quantization error signal. The output of the quantizer $x_q(t)$ can then be expressed as

$$x_q(t) = x(t) + e(t).$$

Since the CT-DSP is a linear-time-invariant (LTI) system, both the input and the error are processed by the same transfer function $H(f)$ of the DSP. The Fourier transform of the CT-DSP output, $Y(f)$, can be expressed as

$$Y(f) = X(f)H(f) + E(f)H(f), \tag{2.16}$$

where $X(f)$ and $E(f)$ are the Fourier transforms of $x(t)$ and $e(t)$, respectively. The error at the output of a CT-DSP is the second term in the equation, whose power spectrum can be represented as $|E(f)|^2|H(f)|^2$.

The CT-DSP consists of an FIR or an IIR filter with a uniform tap delay T_{D} [20]. This results

in a periodic frequency response with a period of $1/T_{\mathrm{D}}$. While the bandwidth of $X(f)$ is less than half this period, $|E(f)|^2$ remains dominant in the whole interval $[0, f_{\mathrm{SAW}}]$, which typically covers several $1/T_{\mathrm{D}}$ periods. Thus, the power spectrum $|E(f)|^2|H(f)|^2$ of the filtered quantization error signal periodically yields regions of high values at least several times outside the input baseband. Interfacing the output of the CT-DSP with standard (synchronous) data-processing blocks requires uniform sampling synchronized with the clock of those blocks. To obtain an in-band *SER* comparable with the nonsampling case, a sampling frequency at least higher than f_{SAW} should be used to prevent any significant out-of-band error power being aliased into the baseband.

Chapter 3

System-Level Considerations of CT Digital IIR Filter Systems

3.1 Introduction

This chapter describes the system-level considerations of a CT digital IIR filter system design. An illustrative example first explains the instability issue introduced by delay-line mismatches. Such an issue has been neither observed nor analyzed by any previous work. This chapter provides a practical solution to solve this issue. Then a new way of designing high-order CT digital IIR filters is introduced. It allows one to use only two tap delays to implement CT IIR filters with any order higher than two. Following that, this chapter also discusses an interpolator used in the system in preparation for CT-to-DT conversion. The interpolator suppresses the noise and distortion power

in the repetitive passbands of the IIR filter's frequency response by employing multiple sections of CT digital FIR filters.

3.2 Preventing instability in CT digital IIR filters

Potential instability and its causes. In contrast to DT digital filters, which use a clock to realize the tap delays, CT digital filters rely on CT delay lines to produce the required timing. In filters with an order greater than one, multiple delay elements are required, and mismatches between them are of concern. In a FIR filter, mismatches only cause errors in the frequency response. In CT digital IIR filters, however, mismatches can cause instability, as will now be explained via an example.

Consider as an example, the second-order IIR filter in Fig. 3.1(a), with a unit pulse at its input. Assume that $b_1 = -1.5$, $b_2 = 0$, so that only the smaller loop is active. This system is unstable; the upper-loop gain is greater than one, which causes the output to progressively become larger and larger, without bound. However, when the lower tap coefficient is changed to $b_2 = -0.6$, the system becomes stable, as can be deduced from its unit pulse response in Fig. 3.1(b). The reason is that the output from the two paths, which are aligned and spaced apart by T_{D}, combine in such a way that the output is prevented from growing. The stability of this system can also be verified with conventional z-domain analysis, which shows that the system poles are inside the unit circle.

Unfortunately, when mismatches exist, the above alignments of the signals from the two taps, which had "saved the show" above, cannot be expected to occur. This is illustrated using the system in Fig. 3.2(a), where T_{D2} is assumed to be slightly different from T_{D1} due to mismatches. The pulses

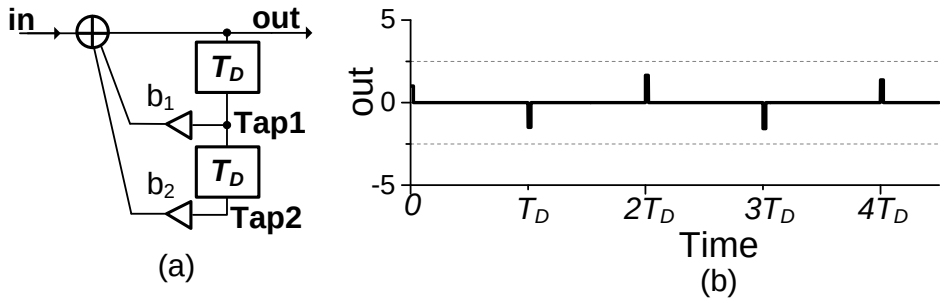


Figure 3.1: An ideal second-order IIR filter. (a) Block diagram. (b) Unit pulse response for $b_1 = -1.5$, $b_2 = -0.6$.

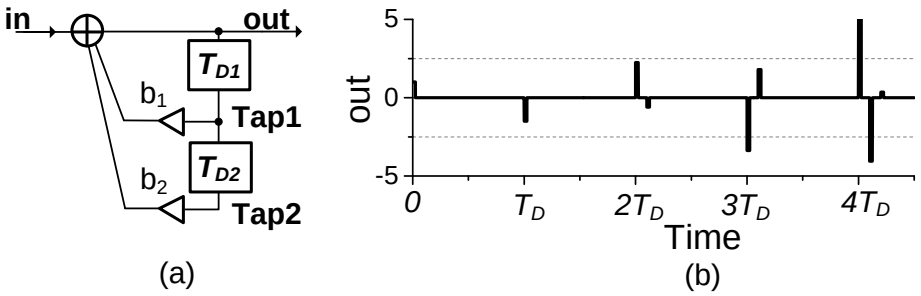


Figure 3.2: A second-order IIR filter with tap delay mismatch. (a) Block diagram. (b) Unit pulse response for the same b_1 and b_2 values as in Fig. 3.1.

being returned from the two taps no longer coincide in time, as shown in Fig. 3.2(b) for a mismatch of 10% (this value is exaggerated for clarity in the graph). The pulses no longer combine as a group, which would have yielded the result in Fig. 3.1(b). Each pulse appears isolated and keeps growing. This instability will be mathematically proved in Appendix A. It can be verified that if each pulse cluster (shown around multiples of T_{D}) is allowed to coincide by reducing the mismatch to 0, the response of Fig. 3.1(b) results.

Solution. Since mismatches prevent the clusters from aligning with each other, we can force the pulses to coincide by grouping them. This can be achieved by holding the pulses that appear at the adder input, and only generating an output when all anticipated pulses have arrived; we can correct

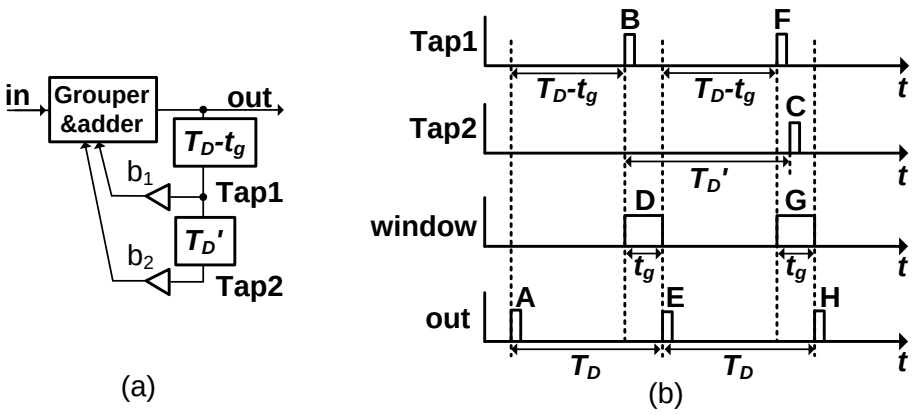


Figure 3.3: A second-order IIR with event grouping. (a) Block diagram. (b) Timing diagram.

for the delay (t_g) caused by this pulse holding and grouping by modifying the upper delay element to $T_D - t_g$ as shown in Fig. 3.3(a). Fig. 3.3(b) illustrates the operation. Assume that Pulse A in the output passes through the top delay and appears at Tap1 after a delay $T_D - t_g$, as Pulse B. This pulse does two things: 1) It goes through the bottom delay of length T_D' (of value to be discussed below), causing Pulse C to appear at Tap2, and 2) it activates a grouping window D of length t_g. Now consider the output of the grouper/adder. Since, in this example, the only pulse present in the first window is B, with no other pulse to be added to it, this pulse appears at the output of the adder, after a delay t_g, as Pulse E. The total delay caused by going from Pulse A, through the loop, to Pulse E, is $(T_D - t_g) + t_g = T_D$, as is marked between Pulses A and E. Similarly, Pulse E is delayed and appears at Tap1 as Pulse F, activating a new window, G. Now there are two pulses to be grouped and added together: F and C. The result appears at the end of the second window, as H. It can be seen from the marked intervals that H follows E with a delay T_D.

The value of T_D' is in principle not critical, as long as Pulse C falls within Window G, but both its position and the width of the window can be optimized given information on mismatches. Thus,

if τ represents the expected worst-case delay tolerance, the window should be at least as wide as $t_g = 2\tau$, and T'_D should be such that Pulse C nominally falls in the center of the window, which implies that $T'_D = T_D + t_g/2$.

In addition to the pulses originating on Tap1 and Tap2 as in the above example, we may also have an input pulse (not shown in Fig. 3.3), which is also allowed to activate a grouping window. If no other pulse arrives during that window, the input pulse appears at the output after a delay t_g. If, however, a feedback pulse originating on Tap1 arrives during that window, the window is extended by t_g from such arrival, so that the above approach, which is essential for stability, is not compromised. At the end of this extended window, all pulses, including the input pulse, are grouped and added together. This can cause a small input-dependent variation of the grouping delay; but as long as t_g is much smaller than T_D, the resulting distortion is small.

Since pulses arriving within a same grouping window are combined together, the length of the window t_g determines the minimum interval between pulses getting out of the grouper. It also constrains the input-signal granularity, which should be no less than t_g. Otherwise, two successive input events can collide in one window. Throwing away either event results in a distortion. On the other hand, the input sees a variable window length, which can be slightly longer than t_g. The longest window an input event can see is $2t_g$, which occurs when a feedback event extends a grouping window right before the window closes. Thus, even if the input signal granularity is t_g, the collision can still happen. If the collision happens, the earlier input event is replaced by the newer one to keep the data updated.

This grouping solution has been tested with exhaustive simulations, and has been found to

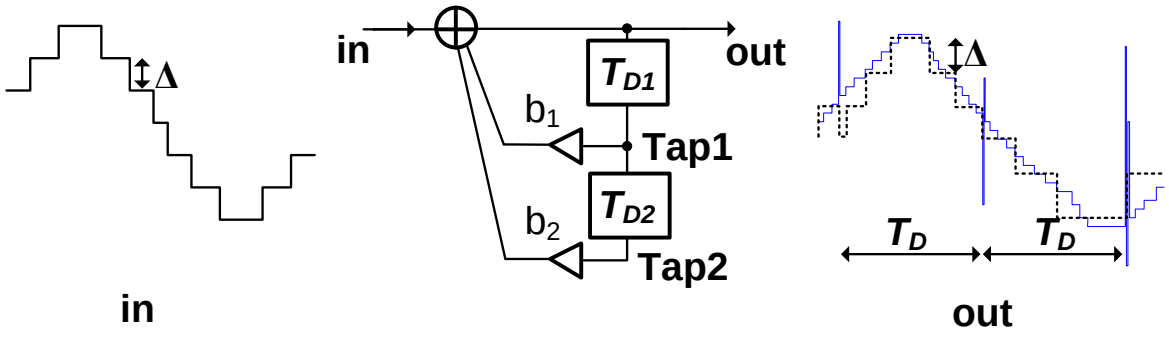


Figure 3.4: Input and output waveforms of a second-order IIR filter with mismatched delay lines. $b_1 = -1.5$, $b_2 = -0.6$. $T_{D1} = T_D$, $T_{D2} = 1.1T_D$. The solid and dash lines at the output show the filtered output with and without the delta modulation respectively.

maintain stability as expected in the presence of delay mismatches. If the input granularity requirement is satisfied, the input collision scenario happens very rarely and the resulting distortion is small.

3.2.1 Discussion of the previous CT digital IIR filter design

To the best of the author's knowledge, the only attempt at CT digital IIR filter design has been in [18]. It implemented a second order CT digital IIR filter. Surprisingly, although it didn't use the grouping method proposed above, the resulting IIR filter was still stable. On the other hand, the filtered output contained a substantial amount of noise. We explain the reasons in this section.

Fig. 3.4 shows typical input and output waveforms of a second order CT digital IIR filter with mismatched delay lines. Its input is a quantized sinusoidal waveform with a uniform quantization step, which is denoted as Δ. The filtered output waveform is shown as the solid line on the right. Because of the misalignment of the responses at Tap1 and Tap2, the output waveform contains spikes. Initially, these spikes are spaced by approximately T_D. However, both the magnitudes and

the width of the spike clusters grow along the time, which ultimately messes up the entire output waveform.

These spikes do not appear at the output of the IIR filter in [18], thanks to the use of a digital delta modulator in the feedback paths, right after the adder. The design in [18] assumes that the adder output does not change by more than 1 Δ at any time (the reason of which can be found in [18]). Hence, a delta modulator can be used to encode the adder output in only two bits: the timing bit and UP/DN. The digital delta modulator truncates the output of the adder so that the magnitude of the least significant bit (LSB) equal to 1 Δ. The modulator monitors the LSB and only generates a feedback event when the value of LSB changes. The resulting output waveform is shown as the dashed line on the right of Fig. 3.4. Because the output waveform can only go up or down by 1 Δ, the growing of the spikes are hence limited. This explains why the output waveforms do not increase without a bound in [18].

On the other hand, the original filtered output shown by the solid line in Fig. 3.4 has shown that the unmodulated output waveform can have a magnitude change more than 1 Δ. Hence, the assumption in [18] that the output waveform can only change by no more than 1 Δ is not correct. Whenever the spikes occur, the digital delta modulator cannot capture the full magnitude change of the output waveform. The modulator output only changes by up to 1 Δ. More seriously, the change directions of modulator's output may be opposite to the change directions of the spikes. An example of such case is shown near the first spike in Fig. 3.4. This is because that the delta modulator generates output based on partial information. The correct output should be the one with

the spikes; thus, compared to that, the delta modulator's output contains a large amount of error. This explains the high noise power in the measured output in [18].

3.3 Digital signal processor with signal-derived timing

In previous CT digital filters [3, 15, 17], the number of tap delays is the same as the filter order. We introduce a new design method that allows one to implement high-order CT digital IIR filter with only two tap delays. As shown in Fig. 3.5(a), a sixth-order CT digital IIR filter can, in principle, be implemented with cascaded second-order sections in direct-II form. Signals at the internal nodes of a CT-DSP are also CT digital. An event at these internal nodes means an update of their data values. The completion of the update is indicated by a pulse on its timing bit *req* (see Fig. 1.4(b)). For example, in Fig. 3.5(a), when an input event arrives, it triggers an update at node p to reflect the data change at the input. Assume first delay-free arithmetic and perfectly matched delays. The new event at p also triggers new events at u and v simultaneously. Events at node p, u and v share the same timing. Since the delays in all taps are identical, events at node p_1, u_1 and v_1 also share the same timing. So do events at p_2, u_2 and v_2. Besides the input, the IIR filter only has three distinguishable timing signals. They are all present in the shadowed part in the first section of the IIR filter. Its timing path can then be separated out and shared by the rest of the system, which results in an equivalent implementation in Fig. 3.5(b). The timing block is composed of two tap delays in feedback to generate the timing needed to realize the infinite impulse response. The signal-derived req_{in} further derives three more timing signals: req_W, req_{R1} and req_{R2}. They trigger operations in the data path, including arithmetic operations and three FIFOs with write (W) and

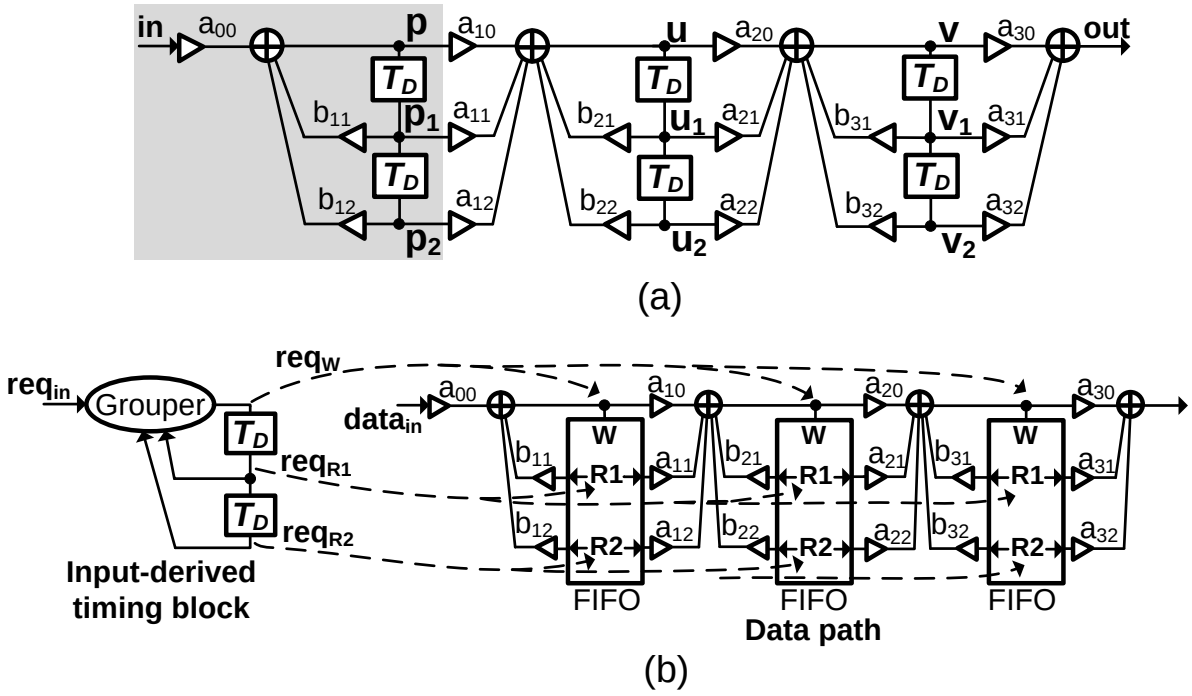


Figure 3.5: A sixth-order CT IIR implemented with cascaded second-order sections. (a) Block diagram. (b) Implementation with shared timing signals derived from input.

two independent reads (R1 and R2) each. This method of using only four timing signals to control the entire system can be applied to CT digital IIR filters of any order higher than two.

The T_D ($2T_D$) delay in the data path is realized by writing data into asynchronous FIFOs and reading it out T_D ($2T_D$) later. Since there is an one-to-one correspondence between a pulse at the timing signals and a data stored in the FIFOs, it is extremely important not to lose any pulses during the propagation of the timing signals in the system. This is ensured by using four-phase handshaking communication protocols [35] between blocks. We will discuss this in more detail in Chapter 4.

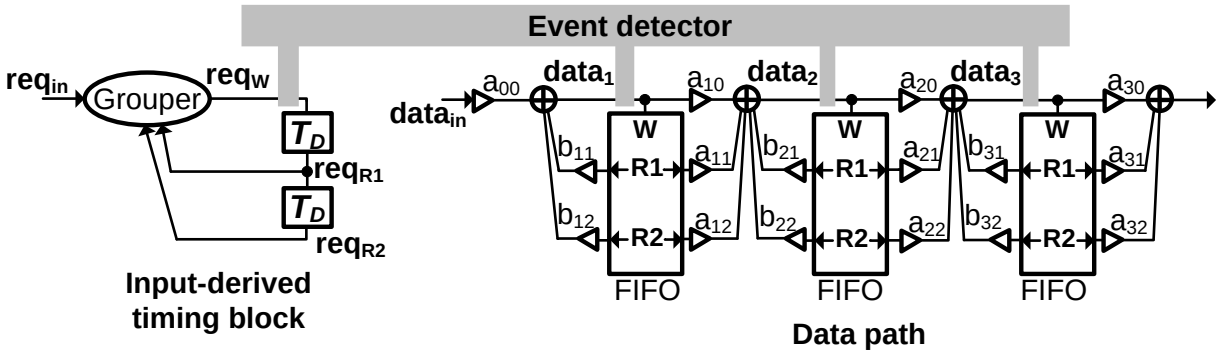


Figure 3.6: A sixth-order CT IIR with an event detector.

3.3.1 Event detection

Because the timing block in Fig. 3.5(b) contains close-loop paths, once an event enters, it keeps going around the loops without a stop. If there is no change at the outputs of the arithmetic blocks, these activities generate redundant events that waste energy. To remedy this, an event detector is introduced to monitor the signals in the feedback loops. Shown in Fig. 3.6, the event detector sits across the timing block and the data path. It monitors the data which will be stored into the FIFOs (i.e., $data_1$, $data_2$, and $data_3$) and controls the timing bit req_W. When a pulse occurs on req_W, it tries to write data, which are associated with this event, into the three FIFOs. If the values of $data_1$, $data_2$, and $data_3$ are all the same as the previous event, the current event is identified as redundant. The event detector prevents the pulse entering the delay lines in the timing block. It also prevents the pulse triggering write operations in the FIFOs. This event is effectively eliminated from the system. As long as any one of the three data is different from the previous event, the current event is not redundant.

3.4 Interpolation filter for synchronization

Although the operations of a CT digital IIR filter track the input activity, the varying event intervals of the CT digital signal at its output prevent its integration with DT systems. Such integration becomes possible if a synchronous output is generated by sampling the CT signal with a clock. Because of the use of tap delays, the frequency response of the CT digital IIR filter is periodic. The repetitive passbands preserve distortion and noise power. It stays out of band if the CT digital signal is the final output as in [15]. Once sampled, it is aliased back and degrades *SNDR*. A computationally efficient interpolator employing FIR filters is used before sampling to alleviate the issue [36, 37] (Fig. 3.7). It is composed of four first-order FIR sections. The filter implements notches at $(k+16n)/T_{\mathrm{D}}$, $k=2,\ldots,15$, $n=1,2,\ldots$ (Fig. 3.7(b)). These notches suppress the distortion and noise power, and push away the first intact repetitive passband to $16/T_{\mathrm{D}}$. Configuring the tap delays and coefficients, one can design a similar interpolation filter for high-pass and bandpass cases [37].

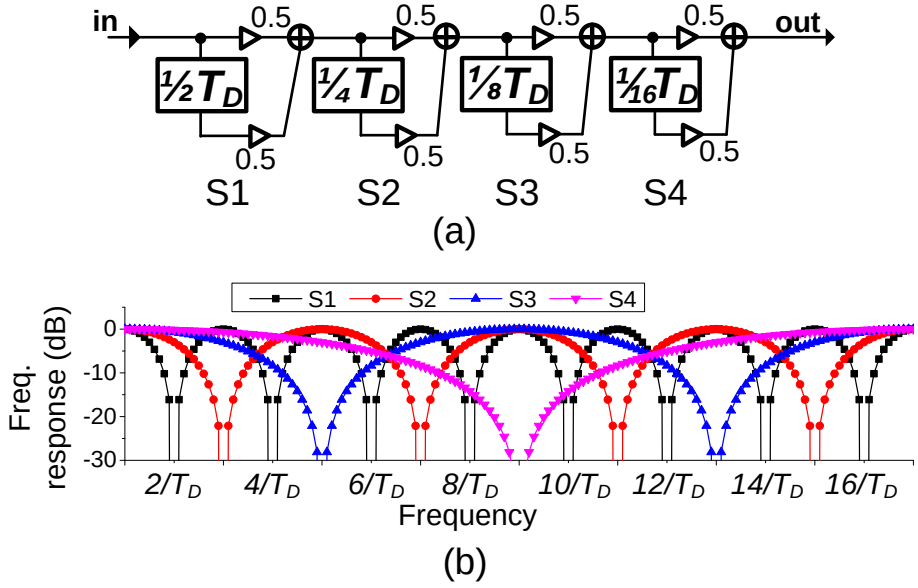


Figure 3.7: Interpolation filter. (a) Block diagram. (b) Frequency responses of each section.

Chapter 4

Design of Tunable Digital Delay Cells

4.1 Introduction

As we have introduced in Chapter 1, tunable digital delay cell is the most important building block in CT-DSP systems. It is often expected to have the following attributes:

- Wide tuning range;
- Good matching between identically laid out cells;
- Low jitter;
- Signal-independent delay;
- Robust communication, in order to guarantee correct propagation of every input event.

The existing designs [38–40] have shown satisfactory results in terms of tuning range. However, none of these work tried to optimize delay line matching. A poor delay line matching only results in

an inaccurate frequency response in CT digital FIR filters [3, 15, 17]. However, it can be disastrous in CT digital IIR filters, as we explain in Section 3.2. This chapter presents a widely tunable delay cell with good matching properties. It also has low jitter and a robust communication channel to adjacent circuits. Previously unreported effects that result in signal-dependent delay are discussed and eliminated.

This chapter is based on [41].

4.2 Prior art

We begin with a discussion of prior art, both because we use some of its features, and in order to point out its limitations, which are addressed in this work.

We focus on delaying digital information. Pure analog delay cells are possible [42], but the required constant bias current makes them energy inefficient for our purposes. Delay cells composed of cascaded inverters are not easily tunable. Delays based on thyristor-like circuits using a current source to charge a capacitor consume low power and have good tuning range [38–40], and will be the focus of this paper.

In CT-DSP applications, a delay line is used to implement a long delay with a fine granularity, and hence is composed of a chain of smaller-delay cells. This is shown in Fig. 4.1(a). Each cell passes an event (here defined as an up-going digital pulse edge) to the next stage through an asynchronous four-phase handshaking to avoid glitches [35]. Initially, all *req* signals are low and all *ack* signals are high. When the $(k-1)^{\text{th}}$ cell finishes delaying an event, it passes the event to the k^{th} cell by pulling up *reqi* in the latter. Once a delay operation is triggered in the k^{th} cell, its *acki* is pulled

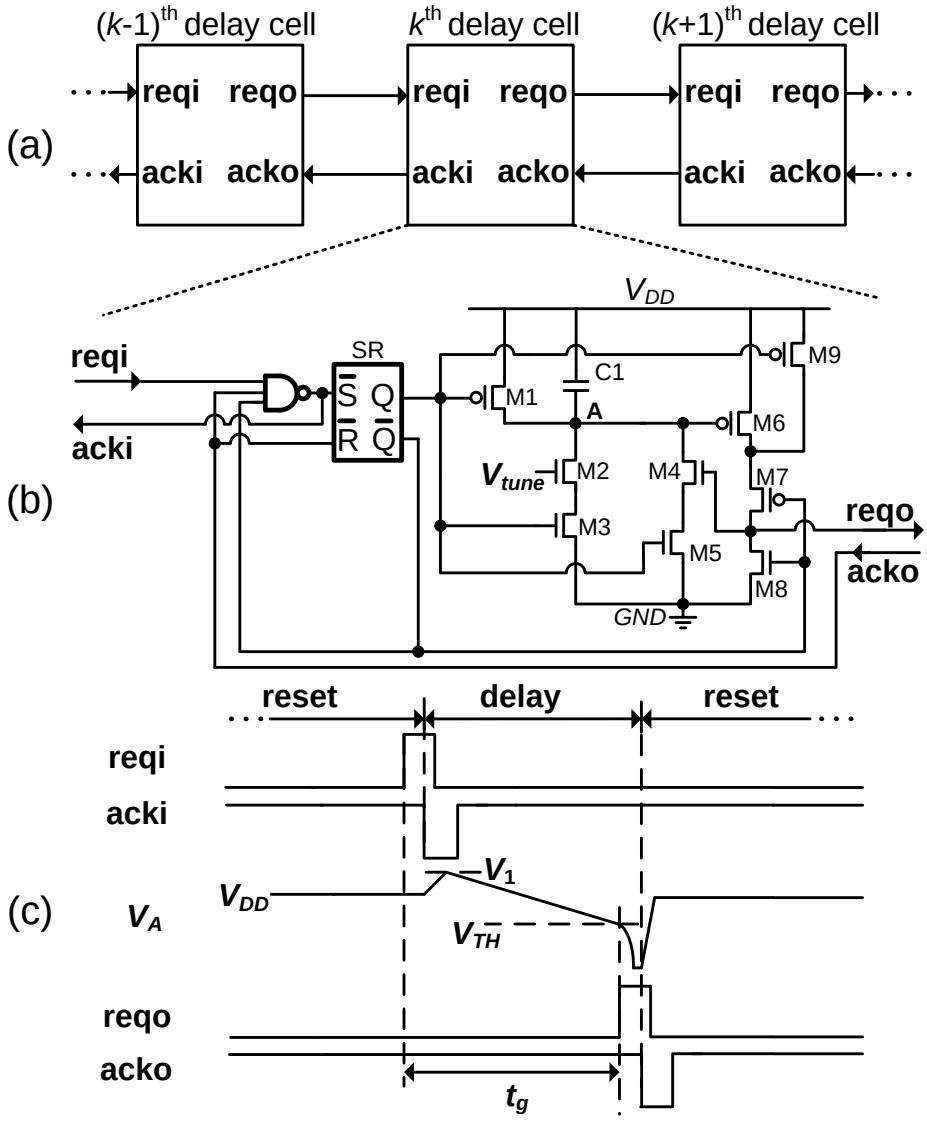


Figure 4.1: Delay cells. (a) A delay line composed of several delay cells for fine time granularity; (b) schematic of one delay cell; (c) timing diagram.

down to acknowledge this to the $(k-1)^{\text{th}}$ cell. That cell then quickly resets itself and pulls down *reqi* of the k^{th} cell. Following that, *acki* of the k^{th} cell is pulled up to conclude the event transfer between the two stages. After the k^{th} cell finishes delaying the event, the same handshaking takes place between the k^{th} and $(k+1)^{\text{th}}$ delay cells.

The schematic of one delay cell is shown in Fig. 4.1(b) [38–40]. A rising edge on *reqi* from the previous cell sets the latch and moves the cell into the delay phase; *acki* is pulled down to acknowledge the previous cell. The voltage on node A jumps to V_1 initially, due to capacitive feedthrough when M1 is turned off. The capacitor C1 is then slowly charged through M2–3. M4–8 form a transistor-based thyristor [38] to monitor V_A. Once V_A falls below the threshold of the thyristor, M4 is turned on to quickly pull V_A to *GND*. The thyristor provides a sharp rising edge on *reqo*. When the next delay cell enters the delay phase, it pulls *acko* down to reset the SR latch. The delay cell shown enters the reset phase. M3 is turned off and M1 is turned on to discharge C1. When V_A goes back to V_{DD}, the cell is fully reset, where it remains in anticipation of the next input event.

The time difference between the rising edges of *reqi* and *reqo* is the cell delay, denoted by t_{g}. It is equal to

$$t_{\text{g}} = C(V_1 - V_{TH})/I \tag{4.1}$$

where C is the capacitance of C1 and I is the DC current of M2. The delay can be tuned through I by changing the gate voltage of M2, V_{tune}. The energy consumed by each delay is dominated by the charging and discharging of the capacitor, and equals CV_{DD}^2.

Although the design described can be widely programmable, it suffers from three issues. First,

energy is wasted since, after V_A crosses V_{TH}, the circuit keeps charging C1 unnecessarily. Second, the implementation of the input asynchronous channel has a potential hazard: when an event arrives at the input of the current delay cell, its *acki* is pulled down to set its SR latch and to reset the previous delay cell. If the reset delay of the previous cell happens to be longer than the delay of the SR latch, *acki* is pulled up to stop the reset operation even before that operation is completed. This error is fatal, because the previous delay cell is not able to accept new events anymore. Third, delay variations due to device mismatches is not considered and minimized in this circuit. In this work, we introduce a new delay cell which resolves all the above issues.

4.3 Proposed delay cell

4.3.1 Eliminating energy waste

The first version of the proposed delay cell is shown in Fig. 4.2(a) (improvements to it will be discussed later). At the output, three inverters are used to replace the thyristor in Fig. 4.1. Fig. 4.2(b) shows the timing diagram. At the end of first delay phase, when V_A crosses the threshold of the first inverter (M5–6), *reqo* is pulled up. Once the next stage enters the delay phase, it pulls *acko* down. This resets the SR latch shown, and turns M3 off and M1 on. As a result, V_A is discharged to V_{DD} from the vicinity of V_{TH}, rather than from *GND* as in Fig. 4.1; energy is saved. The second half of the timing diagram will be explained later. V_{TH} is determined by the strengths of M5 and M6, while the following two inverters (M7–10) provide gain and correct logic polarity. As V_A slowly

moves near V_{TH}, it causes a crow-bar current through M5 and M6. Sizing M6 to be weak limits the current and hence alleviates this issue.

4.3.2 Ensuring robustness

At the input of the delay cell, a C-element [43] is introduced to improve robustness (Fig. 4.2(a)). This is a memory circuit that outputs a high when both inputs are high; outputs a low when both inputs are low; and, if the two inputs are different, holds the previous output value. Initially, the delay cell is in reset phase with Q low and Qb high. The C-element's output is low. Once a rising edge occurs on *reqi*, the C-element's output is pulled up and *acki* is pulled down. This sets the SR latch and resets the previous stage. When the operation in the latch completes, Qb is pulled down; when the previous stage finishes resetting, *reqi* is pulled down. Thanks to the use of the C-element, the order of these two events no longer matters. The C-element goes back to low only when both events occur. This assures that *acki* goes back to high only after the reset operation in the previous stage is completed. The design is insensitive to gate delay variations, as long as the following two timing assumptions are satisfied: 1. The falling delay of Qb in the SR latch, plus the falling delay of the C-element should be shorter than the rising delay of Q plus the delay of node A decreasing from V_{DD} to V_{TH}, plus a handshaking delay at *reqo*/*acko* interface. This assures that once a delay cell enters the delay phase, the C-element's output can go to zero before the delay cell starts the next reset phase. 2. The rising delay of the NAND gate should be shorter than the rising delay of Qb in the SR latch plus the rising delay of the C-element. This assures that the set and reset

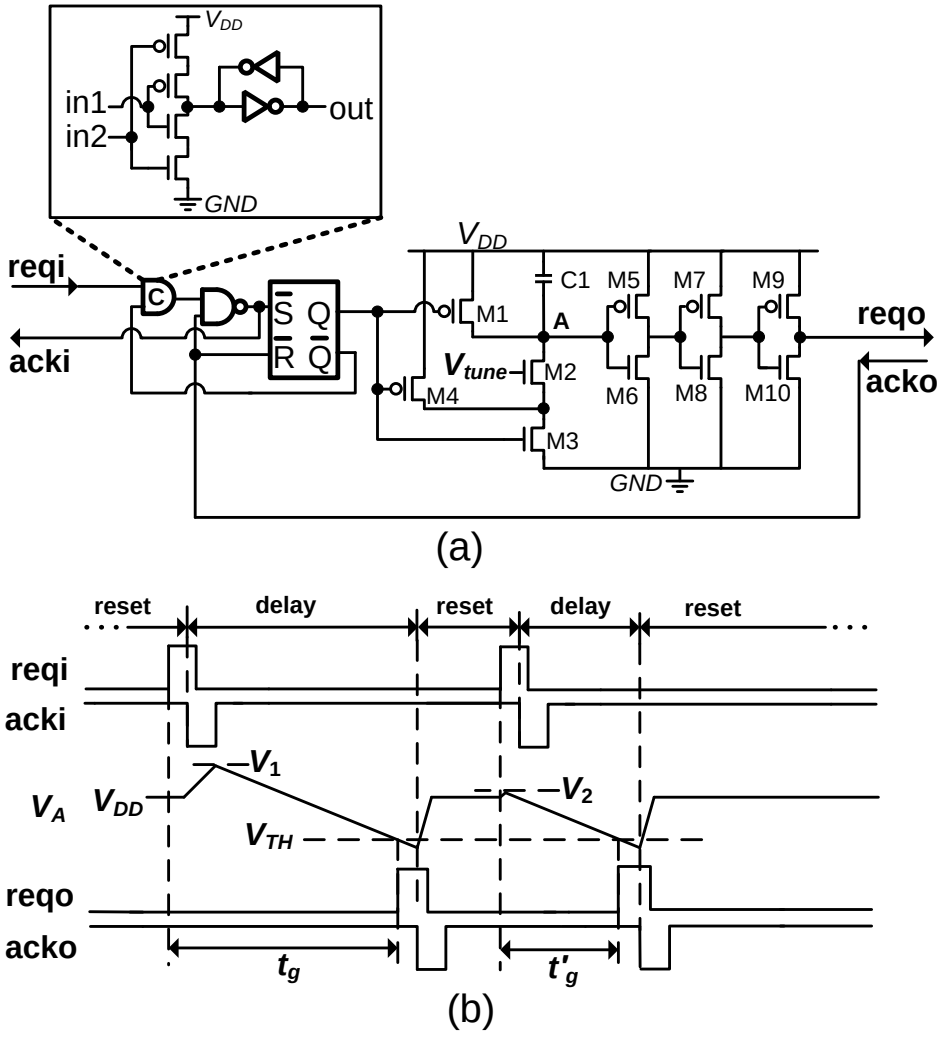


Figure 4.2: Baseline delay cell design. (a) Schematic; (b) timing diagram.

terminals of the SR latch are never on simultaneously. Fortunately, both timing assumptions can be very easily satisfied.

4.3.3 Sizing the current source for good matching

Device M5 and M6 cannot be made very large, as this affects the parasitic capacitance of node A. With reasonable small values for them (see Table 4.1 below), simulations show that the variation of the delay around its nominal value is dominated by the current source (M2) and the capacitor (C1) in Fig. 4.2. The capacitor size needs to be adequate for good matching and low jitter but not larger, as this would necessitate a large current for charging it and would waste energy. A pMOS capacitor working in the depletion region, with a capacitance of 13 fF, was found adequate based on these considerations and assuming a nominal delay of 25 ns.

A delay cell was designed in a 65 nm LP CMOS process to study the relations between delay variation and the size and bias of the current source (M2 in Fig. 4.2). Fig. 4.3(a) shows the ratio of standard deviation of the delay to its mean value as a function of the transistors gate area $A_G = WL$. One hundred Monte Carlo simulations were run, with fixed V_{tune} and W/L so that the transistor is at the same inversion level when its gate area is changed; the behavior seen is as expected. Fig. 4.3(b) shows the same ratio as a function of W/L, keeping the gate area fixed. Its bias current is provided through a current mirror fed with a constant current. The mismatch is seen to worsen as W/L increases, because this cause the transistor to move from strong to weak inversion, where matching for constant current is known to be poor. It is clear from Fig. 4.3 that a small delay

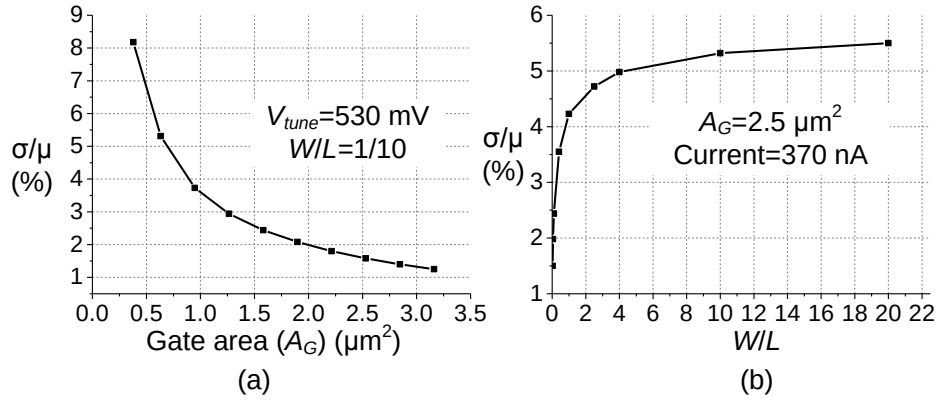


Figure 4.3: Delay standard deviation over mean. (a) As a function of area (b) As a function of W/L.

Device	M2	M5	M6	M7	M8	M9	M10
W(nm)	700	800	120	120	120	120	120
L(nm)	10,000	200	400	60	60	60	60

Table 4.1: Devices sizes of the delay cell in Fig. 4.2.

variation requires large A_G and small W/L. Since $A_G = WL$, or $L^2 = AG/(W/L)$, a large L is needed.

The same choice benefits the jitter performance. A large A_G keeps down flicker noise, while a transistor in strong inversion has a lower white current noise than in weak inversion, for a given current.

Based on the above guidelines, we choose M2 to have a length of 10 μm and a width of 0.7 μm. With $V_{tune} = 630\,\text{mV}$, and a capacitor of 13 fF, this achieves a nominal delay of 24.9 ns and $\sigma/\mu = 1.2\%$. Some of the final devices sizes are summarized in Table 4.1. The sizes of the switches M1, M3 and M4 are 200 nm (W)/60 nm (L).

4.3.4 Identifying, and eliminating, signal-dependent delay

Unfortunately, the resulting design has signal-dependent delay, as illustrated in Fig. 4.2(b). When the first event enters the delay cell while the cell is fully reset, V_A first jumps up to V_1 due to charge injection from M1 and then decreases to V_{TH}. The cell implements a nominal delay t_g. At the end of the first delay phase, *acko* is pulled down to reset the delay cell. However, when the second event enters the delay cell, V_A jumps to V_2, which is lower than V_1 for reasons explained below. It now takes V_A a shorter time to reach V_{TH}, which results in a t'_g shorter than the nominal value t_g. The difference between V_1 and V_2 is due to the use of the very long transistor (M2) as explained with the aid of Fig. 4.4, which shows the charge distributions when the transistor is fully on (left case) and fully off (right case). In both cases, the gate, drain and body of the transistor are tied to V_{tune}, V_{DD} and *GND* respectively. When the delay cell is in the delay phase, the transistor is on with an inversion layer present (left in the figure). When the cell enters the reset phase, the source terminal is pulled to V_{DD} in order to eliminate the inversion layer; channel electrons exit the device through both the drain and the source terminals. Because the transistor is very long, the electrons have to travel a long distance to reach either terminal, which takes time. There exists a latency between the change of the bias voltages and the change of the charge distribution in the transistor. This phenomenon is a nonquasistatic effect, and its latency is inversely proportional to the square of the channel length [44]. If the delay cell reenters a new delay phase before the procedure completes, some electrons which have not escaped remain in the channel. At the beginning of the second delay phase (second part of Fig. 4.2(b)), positive charges are injected into node A due to capacitive feedthrough from M1. Some of the positive charges are now neutralized by the electrons left over

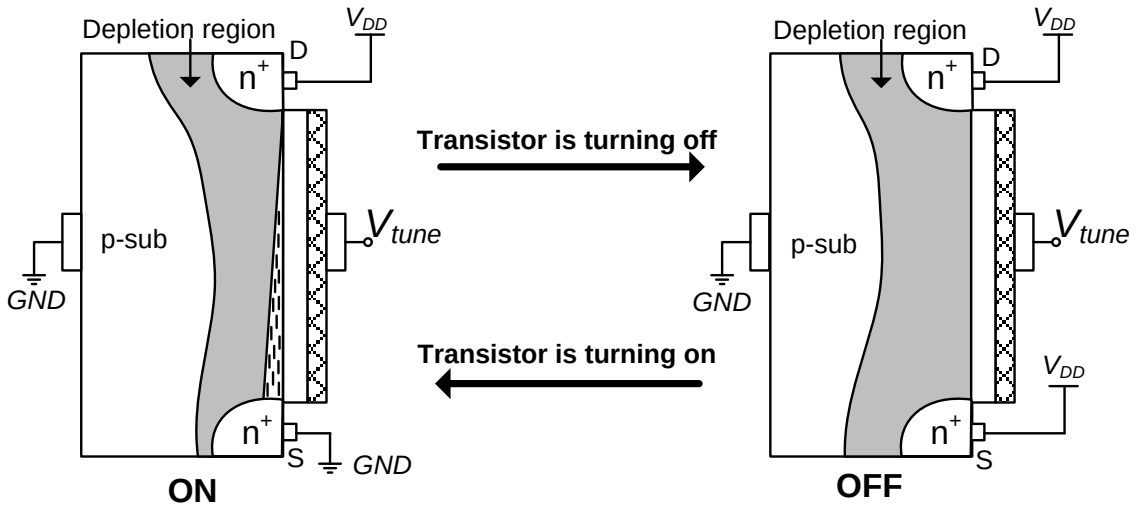


Figure 4.4: Charge distributions of nMOS transistor current source (M2 in Fig. 4.3) when it is fully on (left) and fully off (right).

in the channel, and thus the overshoot V_2 is smaller than V_1. The exact value of V_2 depends on the amount of left-over electrons in the channel, which depends on the spacing between two input events. Hence the delay becomes signal-dependent.

A similar effect exists in the reverse direction, when the transistor is turned from off to on. The latency in this direction becomes part of the delay implemented by the delay cell and hence is not harmful.

The nonquasistatic effects just described can be alleviated by using a shorter length for M2, but this compromises matching. A solution is shown in Fig. 4.5. The original transistor M2 in Fig. 4.2 is split into five transistors with equal lengths (M21-M25). When the delay cell enters the reset phase, the five transistors are turned off in parallel so that the electrons in the channels travel a distance which is five times shorter. When they are turned on, the five devices are connected in series, which is equivalent to a long device whose length is the sum of the individual lengths. In this

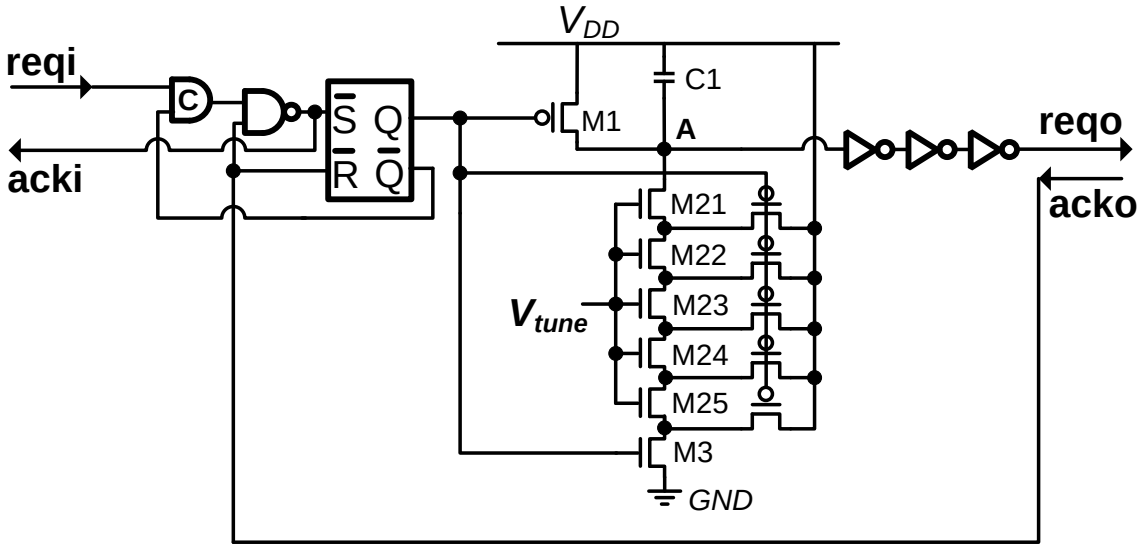


Figure 4.5: Schematic of the modified delay cell.

design, we choose the five transistor to have a width of 0.5 µm and a length of 1 µm. The aggregated length of the transistors is smaller than that of M2 in Fig. 4.2, to make sure each transistor does not suffer from nonquasistatic effects; for this reason, W has also been reduced. With $V_{tune} = 600\,\text{mV}$, the delay cell achieves a nominal delay of 25.1 ns and $\sigma/\mu = 2.3\%$.

4.4 Measurement results of delay cells

Fig. 4.6 shows the photo of the die, with the area containing 128 delay cells of the type shown in Fig. 4.5. The layout of one cell is shown on the right. The chip is fabricated in a 65 nm LP CMOS process. The delay of one delay cell can be tuned by adjusting V_{tune}. As shown in Fig. 4.7(a), a wide tunability range, from 5 ns to 10 µs, is achieved. Automatic tuning of a delay cell using an external clock as a reference is not used here, but is straight-forward [40]. Fig. 4.7(b) shows the

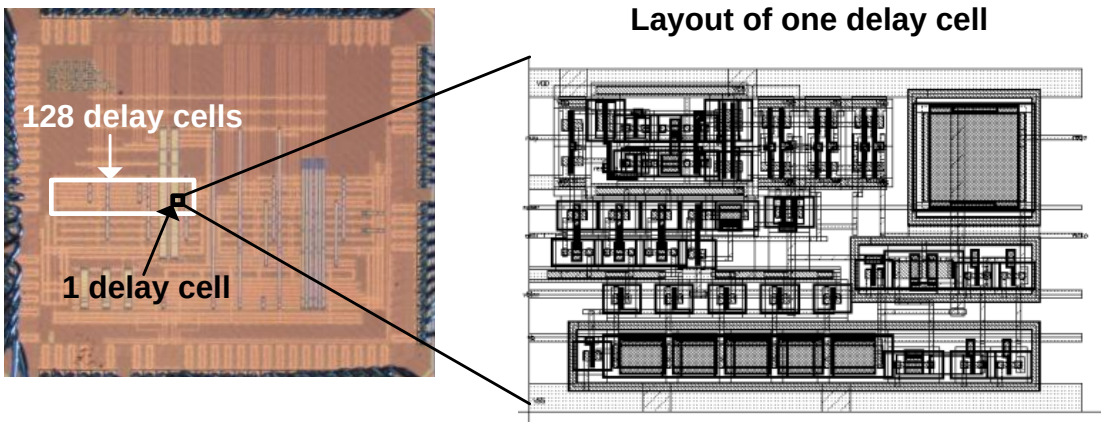


Figure 4.6: Die photo (left) and layout of one delay cell (right).

relation between the delay and the input event spacing. We feed two successive events into one delay cell when it is completely reset, and measure the delay of the second event. The curve with a large variation is for the design in Fig. 4.2, and the flat curve for the design in Fig. 4.5. It is clear that the solution described in Section 4.3.4 solves the problem of signal-dependent delay.

A comparison of this work to delay cells in prior art [38–40, 45] is summarized in Table 4.2; the results shown are measured unless indicated otherwise. Table 4.3 compares jitter performance with multiple delay cells cascaded. The design achieves low jitter and good matching, and provides a robust communication interface. The price to pay for these improvements are a larger area and a higher power consumption. These penalties can be alleviated by simplifying the delay cell design in applications where 4-phase handshaking is not necessary.

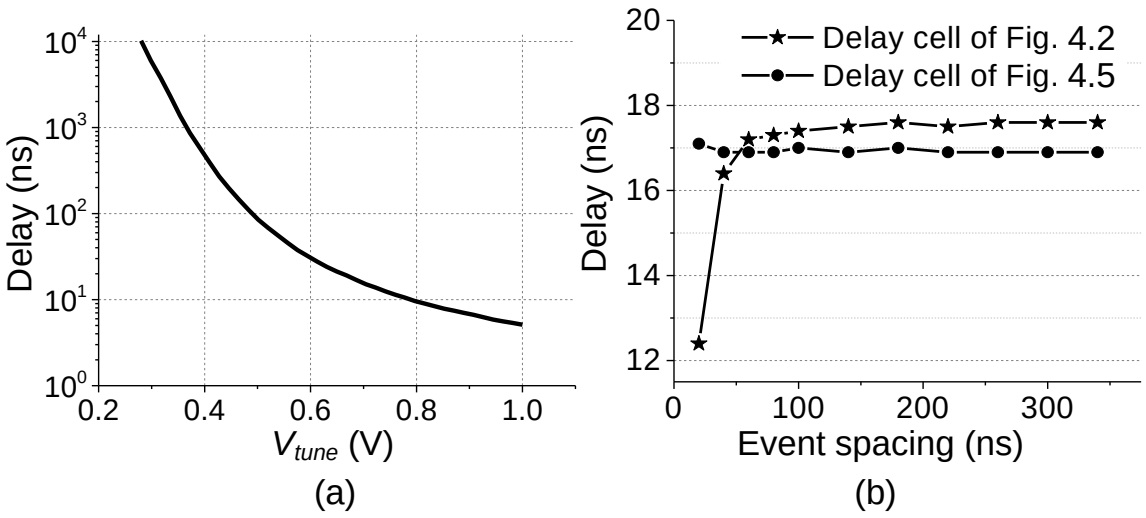


Figure 4.7: Measured delay-cell responses. (a) One cell (Fig. 4.5) versus V_{tune}. (b) One cell versus input event spacing for the cells of Fig. 4.2 and Fig. 4.5.

Parameter	Vezyrtzis	Chang	Schell	Kurchuk *	**This work**
V_{DD}	1 V	2 V	1 V	1.2 V	**1.2 V**
Tuning range	15-500 ns	2.5 ns-76.3 ms	5-1000 ns	0.3-300 ns	**5 ns-10 µs**
Matching	-	-	-	12.4%	**2.3%*†**
Energy/delay	50 fJ	60 fJ	50 fJ	20 fJ	**83 fJ‡**
Area	-	-	36 µm^2	-	**97 µm^2**
Input-independent delay	Yes	No	Yes	Yes	**Yes**
Robustness	Potential hazard	No handshaking	Potential hazard	No handshaking	**Hazard-free handshaking**

*Simulation results.
†Standard deviation over mean value; 100 Monte Carlo simulations; V_{tune}=625 mV.
‡Simulation; measured chip power consistent with simulations.

Table 4.2: Comparison of different delay cells.

	[7]	[10] *	**This work**
Number of delay cells	10	4	**20**
Jitter (rms/mean)	0.33%	0.3%	**0.065%**

*Simulation results.

Table 4.3: Jitter of delay cell cascades.

4.5 Conclusion

Design considerations for digital delay cells with good matching, low jitter, and robust communication interfaces have been presented. A design of a widely tunable delay cell following the guidance presented has been described, achieving a 5X smaller jitter than prior art, and a delay mismatch of 2.3%. Nonquasistatic effects on signal-dependent delay have been addressed and prevented in one of the designs presented.

Chapter 5

Implementation of a CT Digital IIR Filter System

5.1 Overall system architecture

Fig. 5.1 shows the block diagram of the complete system. The CT digital IIR filter implements the desired transfer function. An interpolation filter is used in preparation for CT-to-DT conversion. It is composed of CT digital FIR filters which suppress the noise and distortion power in the repetitive passbands of the IIR filter's frequency response. The CT-to-DT converter converts the filtered signal into a synchronous digital output compatible with conventional DT systems.

A four-phase bundled data protocol [35] is used for communications between and within blocks to assure robust communications between stages. Rather than using a pulse, a rising edge on *req* is now used to represent the occurrence of an event; a falling edge on *ack* means that an event is

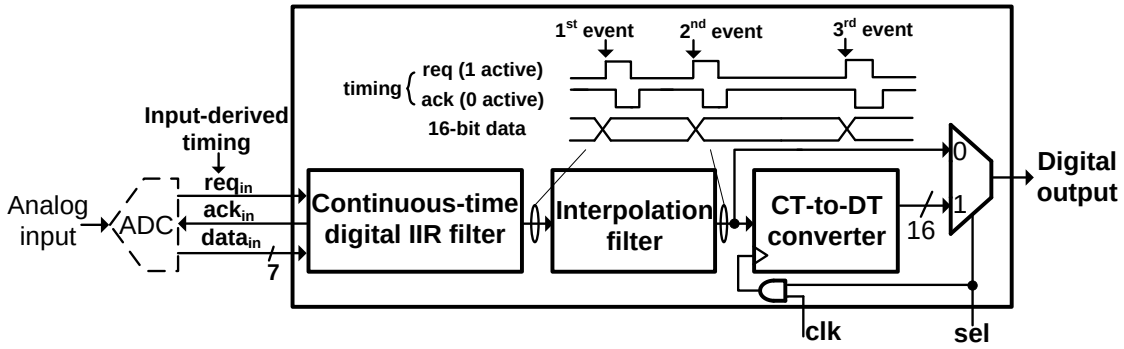


Figure 5.1: Top-level block diagram of the CT digital IIR filter system.

accepted by the following block. For example, if the *req* between the IIR filter and the interpolation filter is pulled high, which means an new event occurs at that node. A request is sent to the interpolation filter. Once the interpolation filter receives the event, it acknowledges the IIR filter by pulling *ack* down. Now that the event has safely propagated to the following stage, the request can be released: *req* is pulled down and *ack* is pulled high. This rising edge on *ack* concludes the four-phase handshake, which allows the next event to occur at this node. Although asynchronous digital techniques are used, a CT digital filter is different from an asynchronous digital filter. The timing distances between events, which contain critical information in CT digital signals, are preserved during their propagation in the former. An asynchronous digital filter is not capable of doing so. Except for the seven-bit input at $data_{\text{in}}$, a 16-bit word length is used everywhere to minimize truncation error. The signal-derived timing, req_{in}, triggers the operations in the entire system.

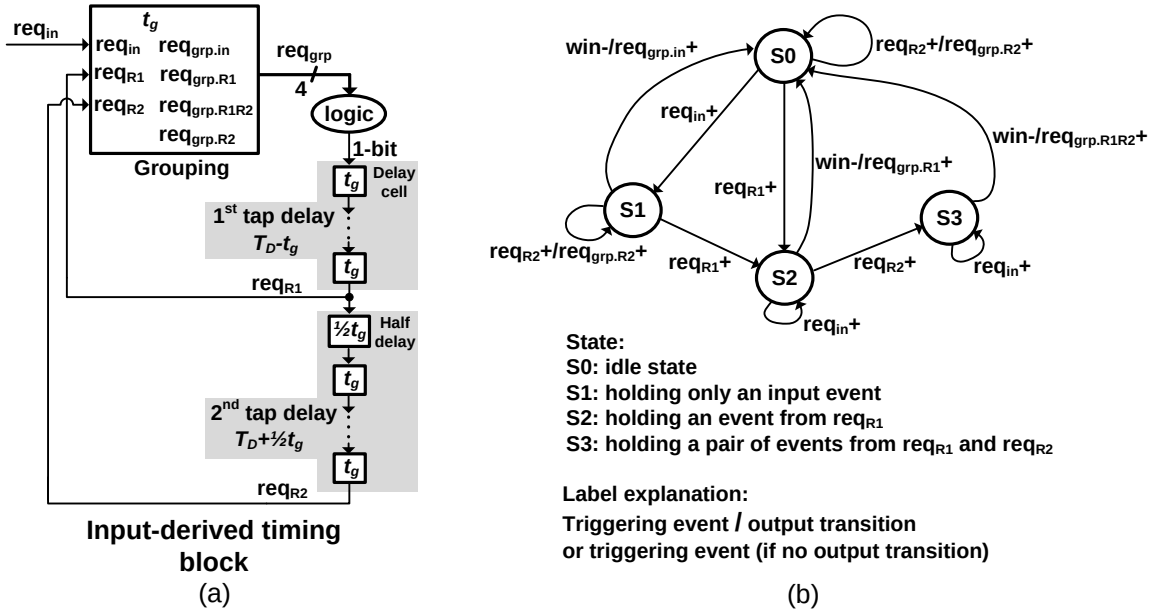


Figure 5.2: The input-derived timing block. (a) Architecture. (b) State diagram of the grouping block.

5.2 Architecture of the CT digital IIR filter

Section 3.3 introduces a method which allows one to design a high-order CT digital IIR filter with only two tap delays. It separates the system into a shared timing block and a data path. This chapter discusses the practical implementations of each part in detail.

5.2.1 Timing block

Fig. 5.2(a) shows the architecture of the timing block with mismatch of delay lines considered. The grouping block and an extra half-delay cell in the second tap delay implement the grouping solution introduced in Section 3.2. The length of the grouping window determines that the minimum interval of events going into the feedback paths is t_g. To handle this traffic, the two tap delays are composed of a cascade of smaller-delay cells with a delay of t_g. The grouping block takes req_{in},

req_{R1} and req_{R2} as its inputs and generates a four-bit output, req_{grp}, each bit of which triggers different FIFO operations in the data path.

Fig. 5.2(b) illustrates the grouping algorithm as a state diagram. Initially the block is in the idle state, S0. When an input event arrives, a rising edge on req_{in} (denoted by req_{in}+ in Fig. 5.2(b))[1] moves the block into S1, meaning that it holds and only holds an input event. Meanwhile, a grouping window is activated. If no other events arrive during it, the window closes (denoted by win-) after t_{g} and the block moves back to S0. In the meantime, $req_{grp.in}$ is pulled up (denoted by $req_{grp.in}$+). If a feedback event arrives at req_{R1} during the window, it pulls up req_{R1} (denoted by req_{R1}+) and moves the timing block into S2, meaning that it holds a feedback event from req_{R1}. The transition from S1 to S2 extends the grouping window by t_{g} from the arrival of the feedback event. If no feedback event arrives at req_{R2} during the window, the block moves back to S0 once the window closes and pulls $req_{grp.R1}$ up (denoted by $req_{grp.R1}$+). Otherwise, the event arriving at req_{R2} pulls up req_{R2} (denoted by $req_{R2}+$) and moves the block from S2 to S3. The new state means that the block holds a pair of feedback events from req_{R1} and req_{R2}. The transition from S2 to S3 does not affect the window. Upon the close of the window, $req_{grp.R1R2}$ is pulled up (denoted by $req_{grp.R1R2}$+) and the block moves back to S0. It is possible that an input event arrives when the grouping block is in S2 or S3. If so, that event is held along with the feedback events without affecting the grouping window or the block state.

The grouping solution introduced in Section 3.2 requests the event at req_{R2} must arrive during S2, within the window triggered or extended by the previous event at req_{R1}. This is ensured by

[1]To keep the discussion simple, we only show the rising transitions of *req* signals. In a practical implementation, a four-phase handshaking is completed soon after each transition, which finally pulls the *req* signals back to zero so that the block can be ready to accept another event.

designing the nominal value of the second tap delay to be $t_g/2$ longer than the first tap delay, and their mismatches to be smaller than $t_g/2$. On the other hand, we retain the grouper's capacity to handle req_{R2} events arriving at another time. This is reserved for the event detection function introduced in Section 3.3.1. When an event arrives at req_{R2} during S0 or S1, it immediately pulls $req_{grp.R2}$ up (denoted by $req_{grp.R2}$+) without affecting the block state or any ongoing grouping window.

5.2.2 Data path

The architecture of the data path with nonzero arithmetic and FIFO delays is shown in Fig. 5.3. Because the adders (except the first one) use the result of their previous adder as an input, the data path is implemented in an asynchronous pipeline to maintain a high throughput. Four adders of the sixth-order IIR filters are spread over four stages (DFF4 to DFF7) so that they can operate concurrently. The multipliers preceding each adder are placed into the same stage where the adder sits to not increase the number of stages in the pipeline. A small number of stages requires fewer control signals and thus simplifies the design.

The pipeline starts when an event arrives at req_{in}, req_{R1} or req_{R2}, activating the grouping window in the timing block and holding the data in DFF1, DFF2 or DFF3, respectively. The read operations in FIFO1 always start t_g before the feedback event actually arrives at req_{R1} or req_{R2}. This allows a t_g timing margin for read operations in FIFO1. At the end of the window, one of the bits in req_{grp} is pulled up, moving any held data in DFF1–3 into DFF4 and triggering new arithmetic operations. Meanwhile, depending on which bit of req_{grp} is pulled up, new data may be

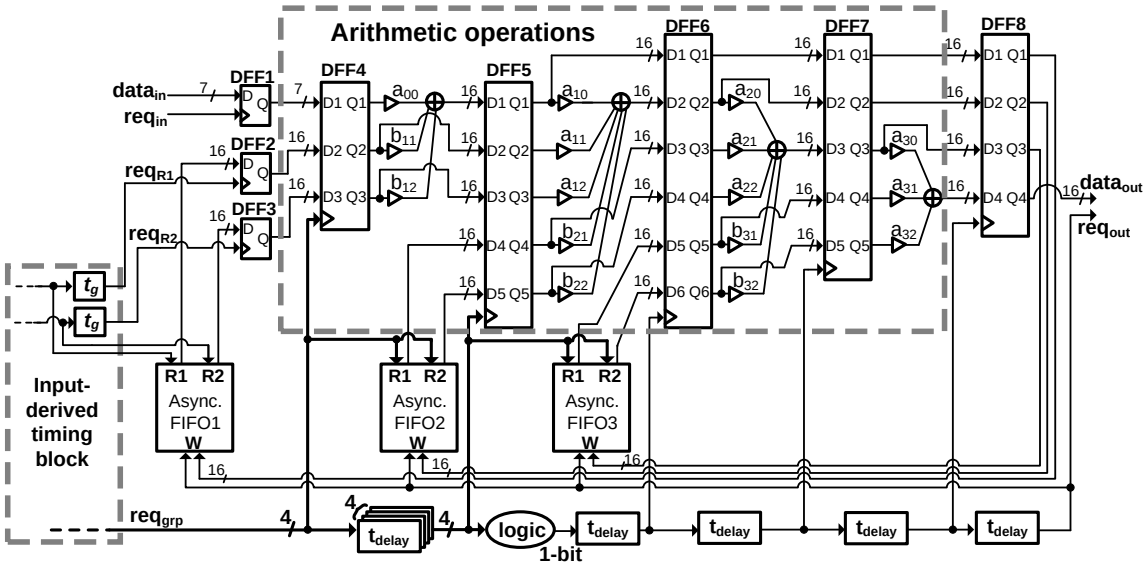


Figure 5.3: Architecture of the data path in a pipeline.

read out from FIFO2 to prepare for the arithmetic operations in next stage. If $req_{grp.in}$ is high, the grouping window holds only an input event and hence no data is read from FIFO2. If $req_{grp.R1}$ is high, a feedback event from req_{R1} is held in the grouper. So a data is read out from R1 of FIFO2. If $req_{grp.R1R2}$ is high, data is read out from both R1 and R2 of FIFO2. After t_{delay}, req_{grp} propagates to DFF5, latching the result from the previous stage into DFF5. Meanwhile, FIFO3 repeats the operation that just occurred in FIFO2. After FIFO3, no further FIFO operations are needed and the four bits in req_{grp} can be safely merged into one with a combinational logic. New data is generated at dataout after $5t_{\text{delay}}$. Three intermediate results of the feedback paths are stored in the FIFOs at the same time, upon the assertion of req_{out}. The value of t_{delay} should be longer than the interstage propagation delay in the pipeline.

Notice that req_{grp} goes through a t_{g}-based delay line in the timing block (Fig. 5.2), and a t_{delay}-based delay line in the data path (Fig. 5.3). These two delay lines have different design

requirements. Because t_g determines the length of a grouping window, according to our conclusions in Section 3.2, t_g needs to be at least twice the worst-case mismatch between two delay blocks. The lower boundary of t_{delay}, on the other hand, is constrained by the propagation delays in the arithmetic operations. The minimum value of t_{delay} decreases with advances in CMOS process, while the choice of t_g cannot benefit from them. However, if t_{delay} is chosen to be the same as t_g, we can combine the two delay lines and save hardware. The nominal value of t_g is 25 ns in this design. We fabricated the prototype in TSMC 65 ns LP CMOS process. 25 ns is much longer than the propagation delay of any stage in the data path. Hence, choosing t_{delay} to be t_g gives the data path enough timing margin for operations. In addition, combining the two delay lines also allows the use of the event detector introduced in Section 3.3.1, which will be discussed in the next section. Fig. 5.4 shows the implementation with the two delay lines combined.

5.2.3 Event detector

In Section 3.3.1, we introduced the event detector to monitor the feedback signals in the IIR filter and to eliminate redundant events. It is inserted into the feedback path, right before the FIFOs into which feedback data are written (Fig. 5.4). Its implementation is shown in Fig. 5.5. When an event enters the event detector, it triggers the left delay cells. The data on in1, in2 and in3 are saved in DFF9 and compared with the data of the previous event stored in DFF10. The comparison delay should be shorter than $t_g/2$ so that the result can be correctly latched. After a t_g delay, req_{EDI} is pulled up. If any of the three data values is different from the previous ones, CMP is low and req_{EDO} is pulled up accordingly. The event passes into the right stage as normal. If all three values

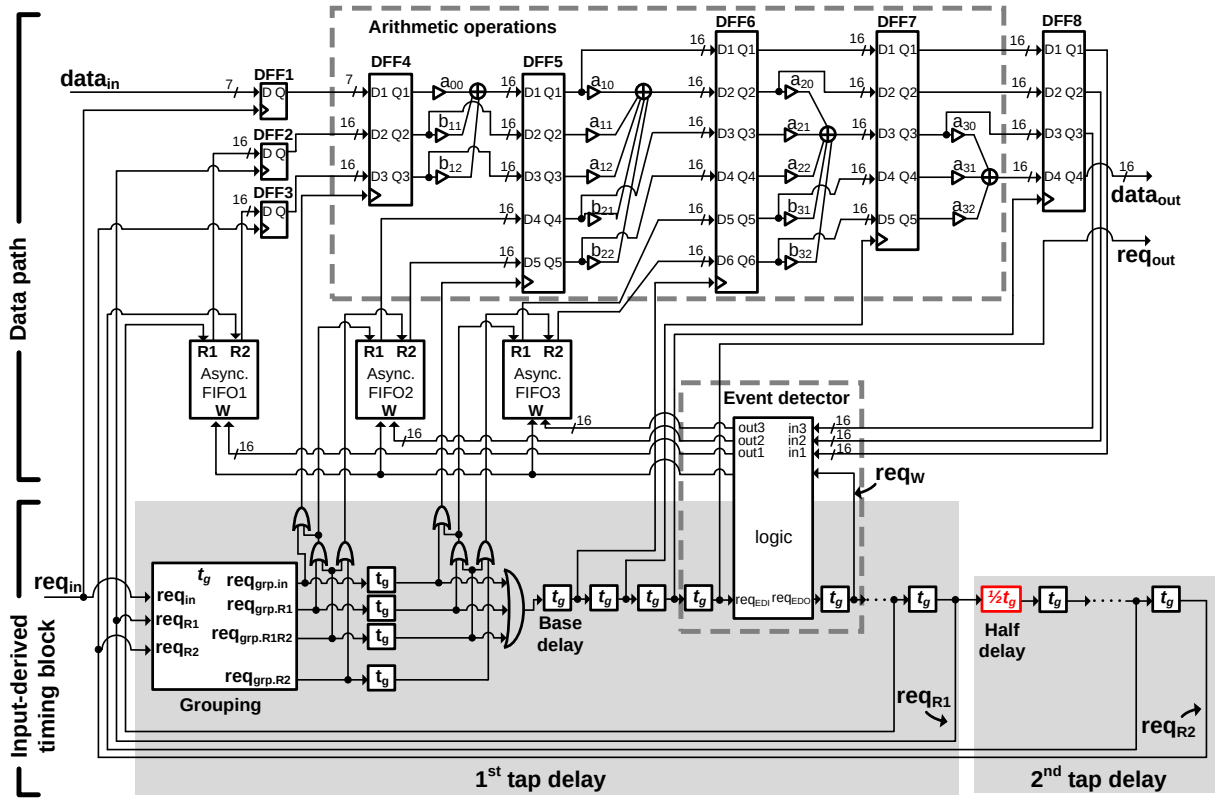


Figure 5.4: Architecture of the sixth-order CT digital IIR filter with shared delay lines.

are the same, the event is identified as redundant. CMP is set high to prevent the event from entering the right stage. The redundant event is effectively eliminated from the filter. A self-acknowledge is conducted in this case to keep the asynchronous pipeline working.

By default, the comparators in the event detector take all 16 bits of the inputs for comparison. In many low-resolution applications, it is not necessary to maintain a 16-bit resolution. The comparator is designed such that its resolution can be programmed from nine to 16 bits, with eight different options. When a low-resolution is used, a data is more easily considered as redundant and the IIR filter is more power efficient. The price is a high quantization noise power in the signal.

The grouping solution introduced in Section 3.2 specifies that the events from Tap2 should

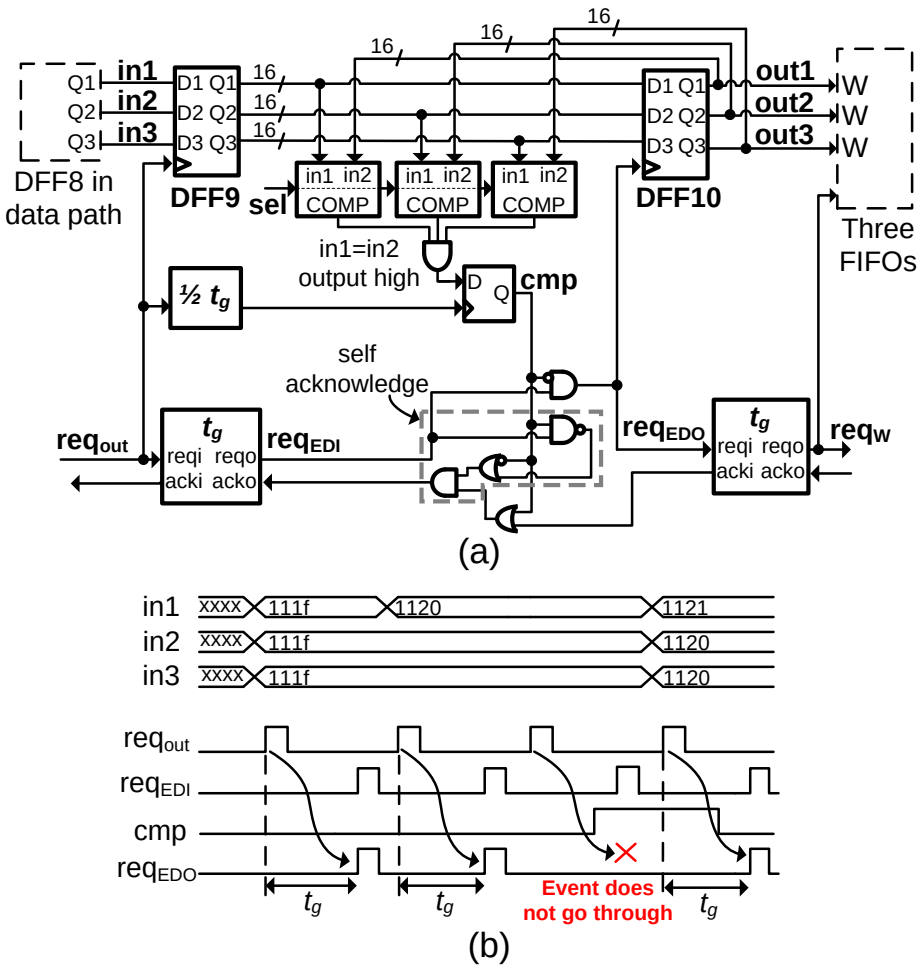


Figure 5.5: Event detector. (a) Architecture. (b) Timing diagram.

always arrive within the window activated by the previous event from Tap1. Notice from Fig. 5.4 that the event detector is only inserted in the first tap delay. It is possible that after an event is eliminated from the first tap delay, its paired event in the second tap delay may arrive at the grouper outside of any window, which violates the requirement. The solution to this is to throw away this left-alone event. When the left-alone event from Tap2 arrives at the grouping block, $req_{grp.R2}$ in Fig. 5.4 is pulled up immediately, without triggering a grouping window. $req_{grp.R2}$ and its delay version are only responsible for reading data associated with the event from R2 of FIFO2 and

FIFO3. The timing signals do not trigger DFF4 and DFF5. Thus, the data does not participate in any arithmetic operations.

5.2.4 Delay cell

The delay cell of the CT digital IIR filter system uses the design in Fig. 4.5 in Chapter 4.

5.2.5 Half-delay cell

The implementation of a half-delay cell is exactly the same as that of the delay cell in Fig. 4.5, except M21-25 are twice as wide. When the capacitor is charging, twice as much current goes through the capacitor, which results in a delay of approximate $t_g/2$.

5.2.6 Grouping block in the CT digital IIR filter

Fig. 5.6 shows the schematic of the grouping block in the IIR filter. It is composed of three channels, with interconnections between them. The feedforward and R1 channels are essentially two delay cells. The two current source transistors, M4 and M8, are each split into five short transistors in series to avoid nonquasistatic effect. However, they are drawn as two single transistors to keep the figure simple. The state of the block is stored in three registers: SR1, SR2 and DFF1. In S0, Q of all latches and DFFs are low; the outputs of all C-elements are low; all *req* signals are low and *ack* signals are high. When an input event arrives, it triggers a delay operation in the feedforward channel by pulling up $Q1$. The block enters S1. t_g later, Input A of the arbiter [47] is pulled up. While $Q2$ stays low, the arbiter pulls $req_{grp.in}$ up and the input event is moved to the next stage.

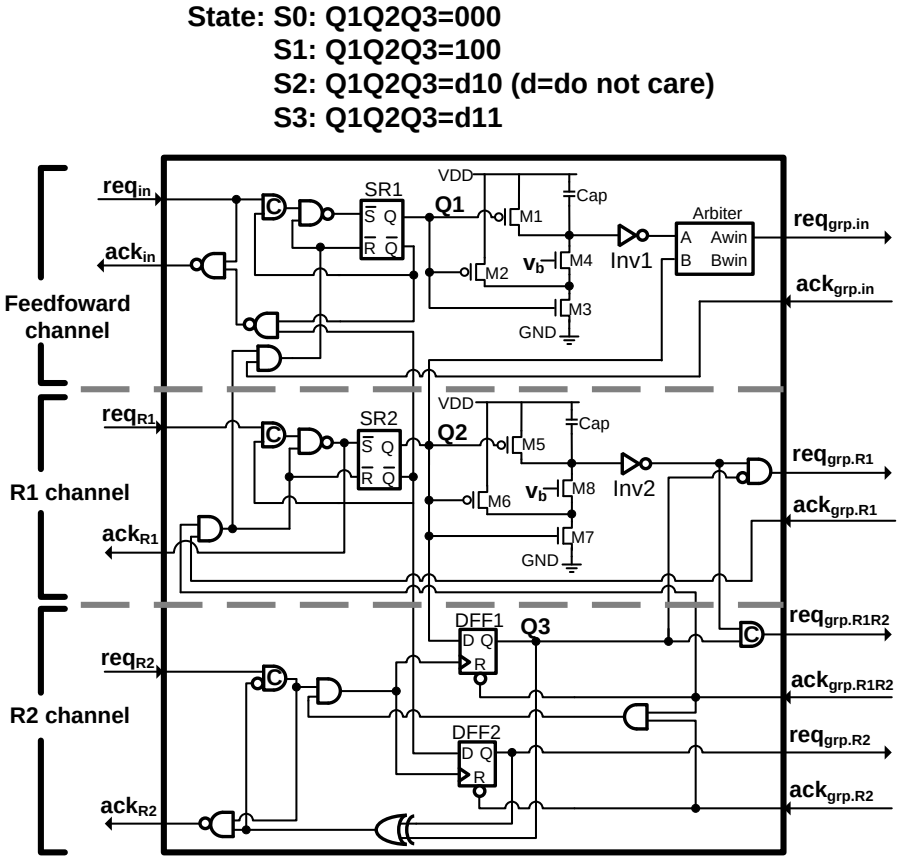


Figure 5.6: Circuit implementation of the grouping block in IIR filter.

After the following stage accepts the event, $ack_{grp.in}$ is pulled down to reset the feedforward channel. The block moves back to S0. If an event arrives at req_{R1} during S1, the grouper enters S2. $Q2$ is pulled up to prevent $req_{grp.in}$ being pulled up through the arbiter. Meanwhile, it triggers a delay operation in the R1 channel. t_g later, Inv2 is pulled up. While $Q3$ stays low, $req_{grp.R1}$ is pulled up. However, if an event arrives at req_{R2} during S2, the block enters S3 and $Q3$ is pulled up. Instead of $req_{grp.R1}$, $req_{grp.R1R2}$ is pulled up upon the completion of the delay operation in the R1 channel. In either case, an acknowledge signal will be received soon to reset the block back to S0. If an event arrives at req_{R2} while $Q2$ is low (i.e., $\overline{Q2}$ is high), DFF2 outputs a high which pulls $req_{grp.R2}$ up

immediately. Soon after that, $ack_{grp.R2}$ is pulled low to reset DFF2. The transition of DFF2 from reset to set to reset again completes very quickly without affecting the state of the grouper.

5.2.7 Asynchronous FIFO

The architecture of the asynchronous FIFO with one write (W) and two read (R1 and R2) channels is shown in Fig. 5.7. It is composed of 128 (4 columns by 32 rows) 16-bit-word cells. The SRAM cell is based on the fully static design in [48], but contains two read channels. Fig. 5.8 shows the timing diagram of both write and read operations. They are triggered by the timing signals req_W, req_{R1} and req_{R2}, which are not synchronized to a clock. A write operation starts when req_W is pulled up. A bundled data encoding scheme [35] ensures that $data_W$ is ready slightly before the timing signal. *Wen* is pulled up to latch $data_W$ into DFF1 and starts writing the cells pointed by the address $addr_W$. After $t_g/2$, *wen* is pulled down and $addr_W$ increases by one to prepare for the next write operation. The write delay should be shorter than $t_g/2$ so it can be completed before the $addr_W$ changes. A read operation starts when either of req_{R1} or req_{R2} is pulled up. Fig. 5.8(b) illustrates an example in which req_{R1} is pulled up. *Ren*1 is then pulled up to read the SRAM cells pointed by the address $addr_{R1}$. After a read delay shorter than $t_g/2$, data is ready on bus_{R1}. $t_g/2$ after req_{R1} is pulled up, *ren*1 is pulled down and latches the new data into DFF3. Meanwhile, $addr_{R1}$ increases by one in order to prepare for next read operation. The three address information $addr_W$, $addr_{R1}$ and $addr_{R2}$ are internal signals in the three address decoders. They increase by one at the falling edges of *wen*, *ren*1 and *ren*2, respectively. The three addresses are decoded into 32-bit row-enable and four-bit column-enable signals to select the desired cells. Because the

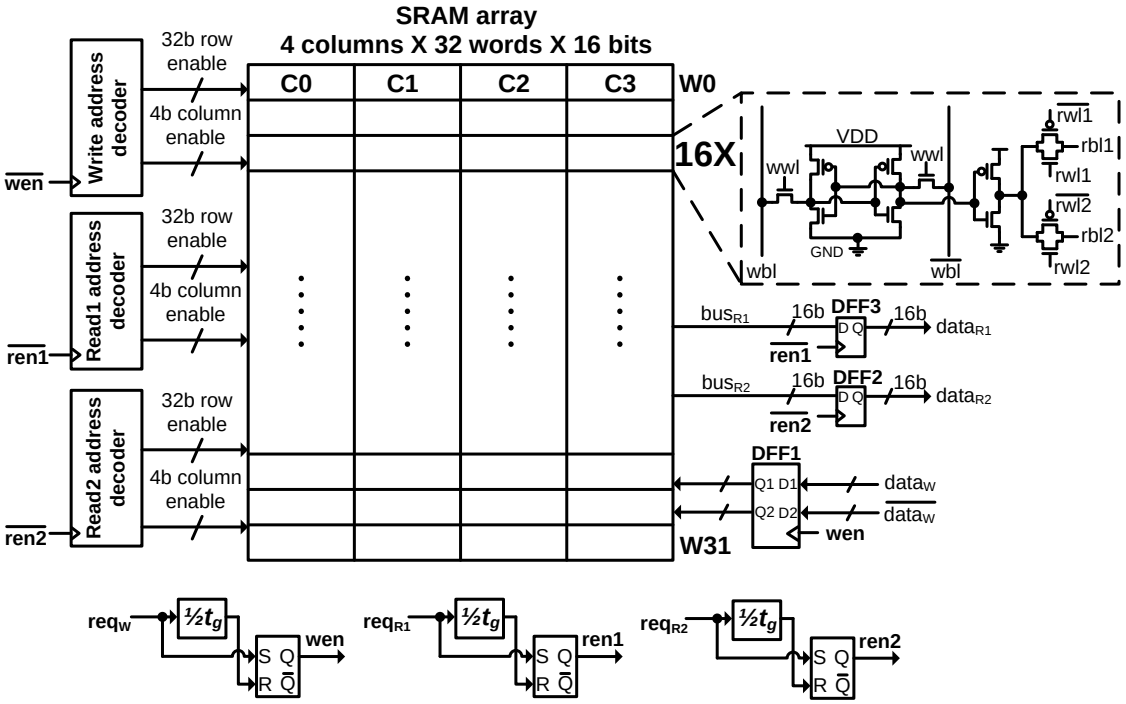


Figure 5.7: Architecture of asynchronous FIFO with one write and two read channels.

three decoders operate independently, the asynchronous FIFO supports one write and two reads operations simultaneously.

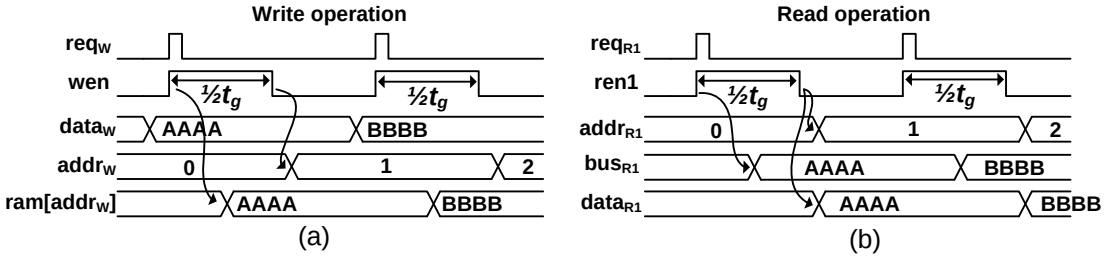


Figure 5.8: Timing diagrams of write and read operations for the asynchronous FIFO.

5.2.8 Arithmetic blocks

The CT digital IIR filter's multipliers and adders are synthesized using standard digital design tools: Verilog, Synopsys® Design Compiler and Cadence® Encounter. This is the first time that synthesize tools are used to implement the arithmetic blocks of a CT digital filter, which saves significant time and design effort. The only design constraint is that the propagation delay of each stage in the pipeline should be smaller than t_g. In this design, the nominal value of t_g is 25 ns. It is relatively easy to meet this requirement in the 65 nm TSMC LP CMOS process.

5.3 Interpolation filter

Introduced in Section 3.4, the interpolation filter is composed of multiple first-order CT digital FIR filters. The architecture of a first-order FIR section is shown in Fig. 5.9. To keep the design modular, its delay line is also composed of a cascade of t_g delay cells. It requires that the minimum interval of events both coming from its previous stage and going to its next stage be no smaller than t_g. On the other hand, the adder may see two events arriving at its two inputs arbitrarily closely. To meet the granularity requirement at the output of the adder, we use a grouping block similar to, yet simpler than, the one in Section 3.2 to combine these close events. When an event arrives at either input of the grouping block, it triggers a grouping window with a length of t_g. Other events arriving within the window are grouped with the leading event and moved to the next stage together at the end of the window.

The operations of the FIR section start when an input event arrives. req_{in} in Fig. 5.9 triggers the

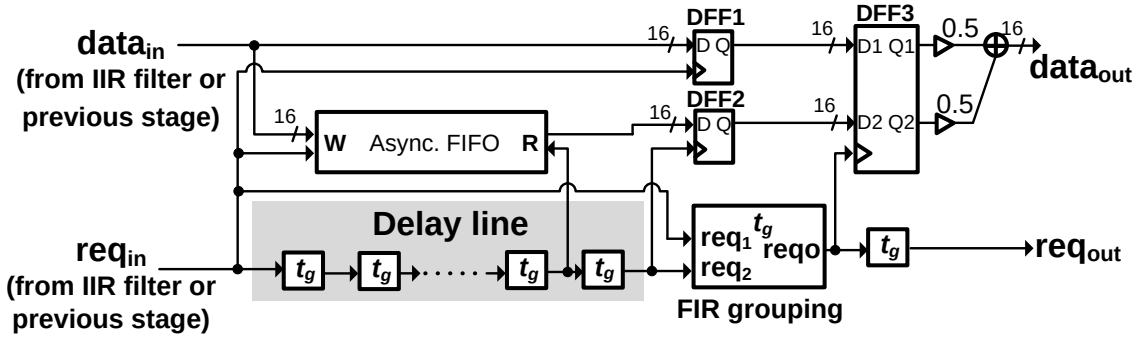


Figure 5.9: Architecture of one FIR section in interpolation filter.

delay line and stores $data_{\text{in}}$ into the FIFO. It also activates a window in the FIR grouping block, and holds $data_{\text{in}}$ in DFF1. At the end of the window, data in DFF1 and DFF2 are moved into the next stage for arithmetic operations. The rising edge on req_{out} identifies the validity of $data_{\text{out}}$. It also moves the output into the next FIR section. Similar grouping and arithmetic operations are triggered once an event arrives at the end of the delay line. In a first-order filter, only one FIFO read channel is required.

5.3.1 Grouping block in the interpolation filter

The grouping block in the interpolation filter is simply a delay cell with a slightly different input interface. Shown in Fig. 5.10, it has two req/ack input communication channels. Whenever an event from either channel arrives, it sets the SR latch and triggers a delay operation. The corresponding *ack* is then pulled down to reset its previous stage. t_{g} later, delay operation completes and *reqo* is pulled up. The time difference between the rising edges of the input *req* signal and *reqo* is the grouping window. If an event arrives at the other *req* during the window, it is properly acknowl-

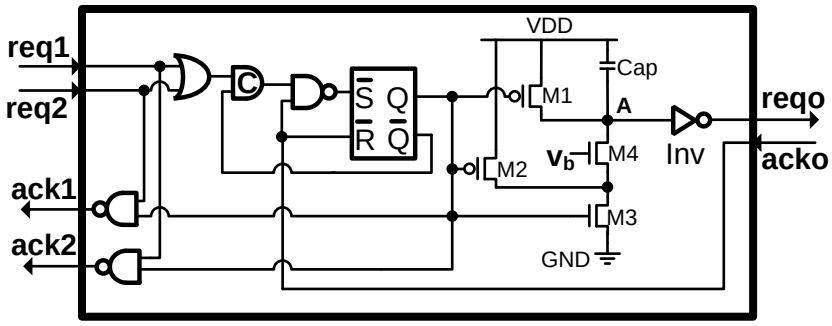


Figure 5.10: Circuit implementation of the grouping block in interpolation filter.

edged by pulling the other *ack* down. At the end of the window, both events are moved to the next stage.

5.4 CT-to-DT converter

Directly sampling the CT digital signal with a clocked DFF converts it into a synchronous digital signal. Traditionally, two DFFs are used in series to minimize the possibility of the final output being in a metastable state [49]. Although the synchronous output is ensured to be in a stable state with a good confidence, it is not necessarily in a correct state. If any bit of a data settles to a wrong state, we call the amplitude difference between the wrong sample and the corresponding correct sample a sampling error. This introduces error power at the synchronous output.

We can rely on a characteristic of the CT digital signal to alleviate this issue. Notice that a CT digital signal has a high data rate. The minimum space between two events is t_g. On the other hand, the tap delay of an IIR filter defines the baseband as $[0, 1/(2T_D)]$. So the maximum input frequency to the IIR filter is $1/(2T_D)$, which is much smaller than the effective event rate, $1/t_g$, in this design.

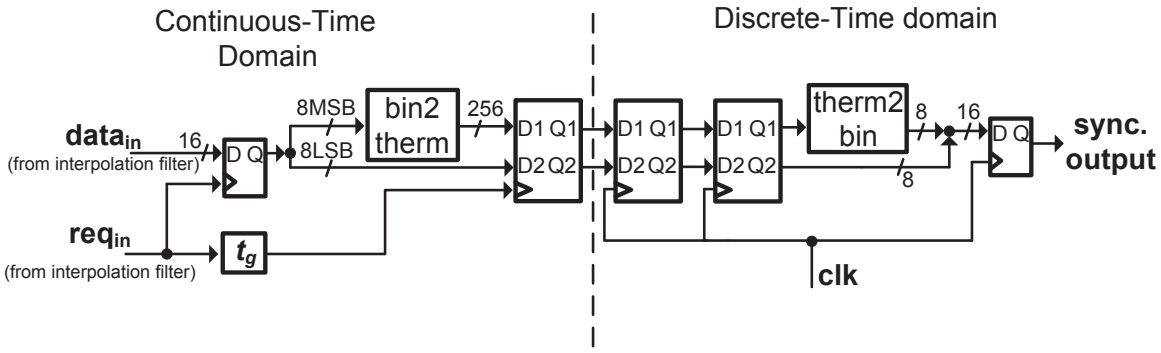


Figure 5.11: Architecture of the CT-to-DT converter.

As a result, the amount of amplitude change between two successive events in a CT digital signal is limited to a certain range, which can be estimated by

$$\Delta V \leq \text{Max Slope} \cdot t_{\text{g}} = \frac{A_{\text{FS}}\pi}{T_{\text{D}}} \cdot t_{\text{g}}. \tag{5.1}$$

In this design, the output resolution is 16-bit, $A_{\text{FS}} = 2^{15}$ LSB, T_{D} is 1 µs, and t_{g} is 25 ns. This results in a ΔV smaller than $2^{11.3}$ LSB. The parameters in the above equation are unchanged if the event detector is programmed to have a resolution lower than 16-bit, as is the absolute value of ΔV. Knowing that, we can convert the binary CT digital signal into a thermometer signal before sampling. Because ΔV never reaches full scale, only a subset of all the thermometer bits can change at any time. The three most significant bits are hence immune to sampling error. Fig. 5.11 shows the implementation of the CT-to-DT converter. To reach a comprise between accuracy and the hardware cost, only the eight most significant bits of the CT signal are converted into thermometer code. After sampling, the thermometer code is then converted back into binary.

Chapter 6

Measurement Results of the CT Digital IIR Filter System

This chapter presents measurement results of the CT digital IIR filter system. Unless otherwise noted, the nominal test condition uses a supply voltage of 1.2 V. The results are collected at room temperature. The test chip is fabricated in a 65 nm TSMC LP CMOS process. The die photo is shown in Fig. 6.1. The design occupies an active area of 0.64 mm^2.

6.1 Description of the measurement setup

The setup for measurements is shown in Fig. 6.2. An analog input is converted in to a CT digital signal either by a CT-ADC or by a DT-ADC following by a DT-to-CT conversion (which will be explained below). The output of the CT digital IIR filter system is a multiple-bit digital signal. It

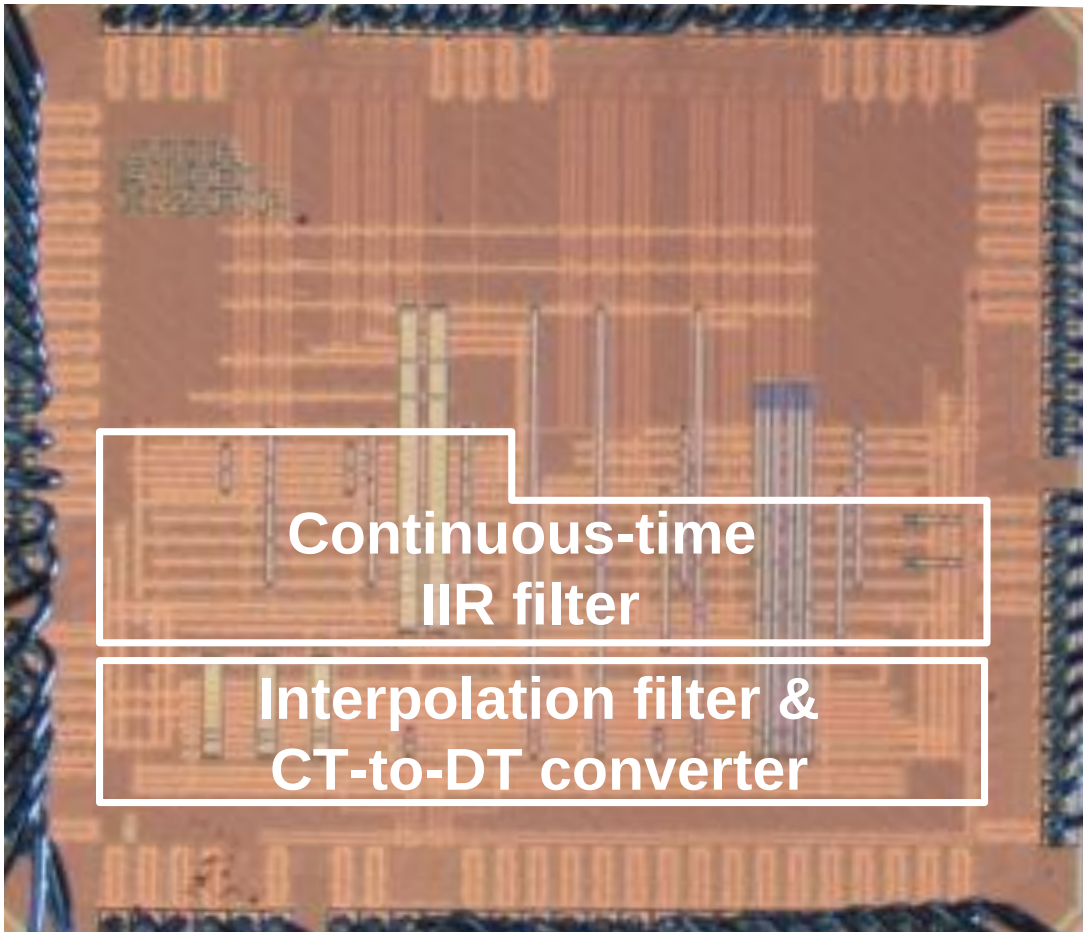


Figure 6.1: Die photo of the CT digital IIR filter system.

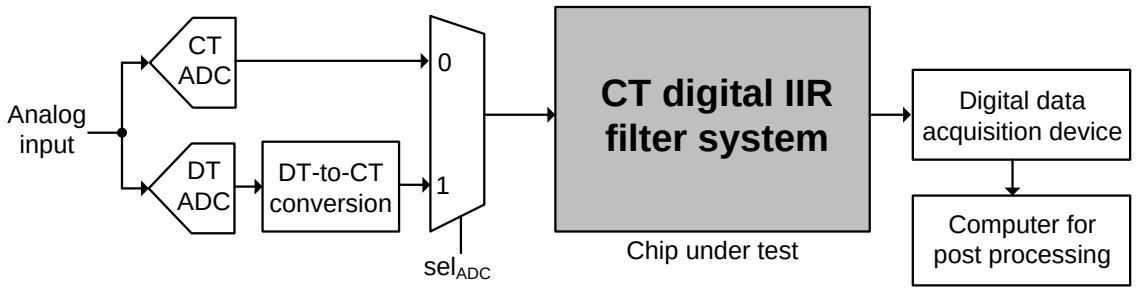


Figure 6.2: Measurement setup.

is acquired by a digital data acquisition device with a 500 MHz sampling frequency. After that, the acquired digital data are postprocessed in a computer.

The setup is composed of four PCBs. Fig. 6.3 shows a photograph of it. The board on the right implements a LCS CT-ADC [3], which converts an analog input into a CT digital signal. The chip under test (CUT) sits on the left board. On the same board, there is a DT-ADC. The two small boards on the bottom generate the configuration bits which will be stored in the CUT's scan chain. They also generate the system-reset signal.

Ideally, we would use a CT digital signal generated from the CT-ADC as the system input for all measurements. However, because it is built with discrete components with long propagation delays, the CT-ADC can only accept a full-scale input up to 5 kHz. To fully measure the performance of the CT digital IIR filter, we need a better signal source. The DT-ADC serves this purpose. We use a Texas Instrument® ADC08L060 for the DT-ADC. It accepts a maximum sampling frequency of 60 MHz. Although the original output of a DT-ADC is DT digital samples, we can convert it into a CT digital signal with a zero-order hold block. The converted output is still composed of binary waveforms, but they are defined in CT: a CT digital signal. Actually, it is not necessary to introduce

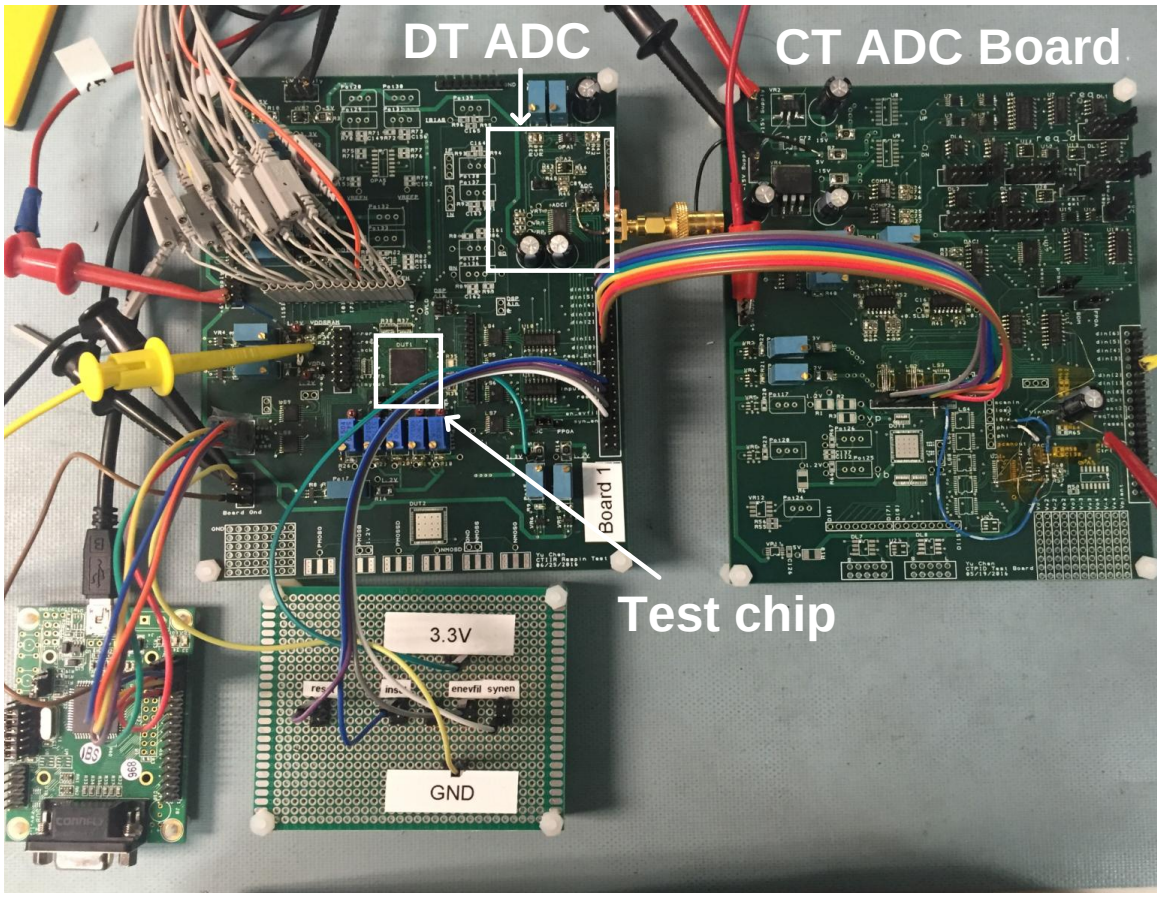


Figure 6.3: PCB boards for test.

an explicit zero-order hold block. The digital output of a CT-ADC naturally holds its value between two clock edges, as shown in Fig. 6.4. The timing bit of a CT digital signal, *req*, is derived from the clock. Feeding the clock into an one-shot circuit results in the required *req*. Whenever there is a new data generated at the output of the DT-ADC, a pulse is also generated on *req*. The delay $\Delta 1$ determines the pulse width. The delay $\Delta 2$ determines the delay between a rising edge on the clock and the rising edge on *req*. It should be long enough so that when *req* is pulled up, the data bits are fully settled. Both the CT-ADC and the DT-ADC have a seven-bit resolution.

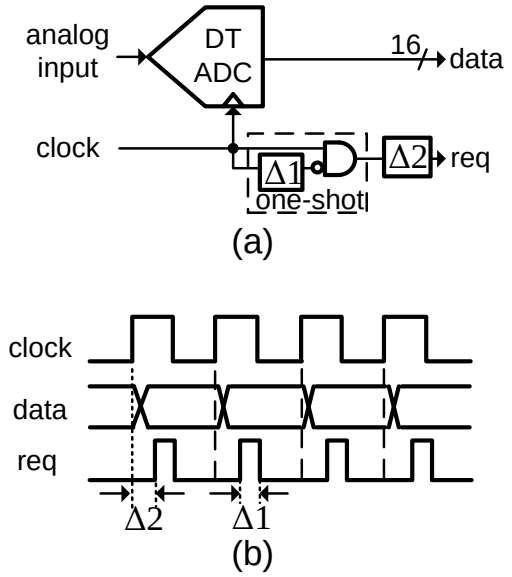


Figure 6.4: DT to CT conversion. (a) Schematic. (b) Operations.

6.2 System configuration and calibration

Fig. 6.5 shows that the CT digital IIR filter system is composed of three blocks, a sixth-order CT digital IIR filter, an interpolation filter and a CT-to-DT converter. All three blocks contain delay elements that require bias voltages. In this design, the t_g delay cells in the IIR filter's second tap delay share a bias voltage V_{b2}; the half-delay cells in the second tap delay and all FIFOs share a bias voltage V_{bhalf}. All other delay elements, including the grouping block, the IIR filter's first tap delay, the delay cells in the interpolation filter and the CT-to-DT converters, share the same bias voltage V_{b1}. Normally, these three bias voltages are connected together. By changing V_{b1}, V_{b2} and V_{bhalf}, we can change the delays of these delay elements.

The system can be configured through the scan chain. The system has two modes: normal operation and test mode. In normal operation, we can configure the tap coefficients and the number of delay cells in the two tap delays to change the CT digital IIR filter's frequency response. The

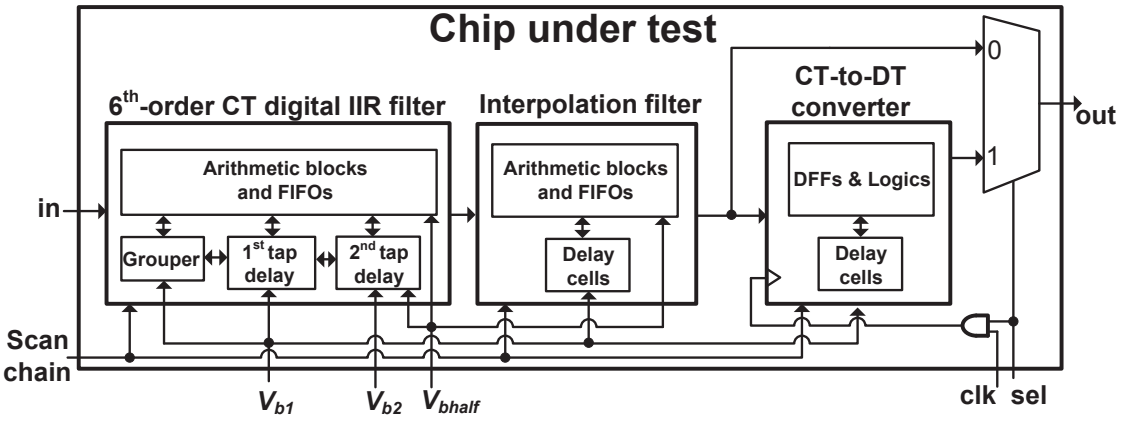


Figure 6.5: Block diagram of the chip under test.

number of FIR sections (0–4) in the interpolation filter and the number of delay cells in the tap delay of each FIR section can also be configured. We can change the notch locations in the frequency response of the interpolation filter through these configurations. Finally, we can choose either to use the binary-to-thermometer conversion or not in the CT-to-DT converter. A *sel* signal from outside the chip can enable or disable the CT-to-DT converter and choose the system output. In the test mode, we can select any delay element in the system and measure its delay value.

6.2.1 Calibration of delay lines

We designed the nominal values of the tap delay, T_{D}, and the smaller delay cell, t_{g}, to be 1 µs and 25 ns, respectively. To avoid systematic mismatches between the two tap delays in the CT digital IIR filter, we include 39 delay cells in the first tap delay and 40 delay cells in the second tap delay. One less cell used in the first tap delay compensates for the delay introduced in the grouping block, which is also 25 ns.

The delay lines in the CT digital IIR filter system can be tuned using an automatic tuning

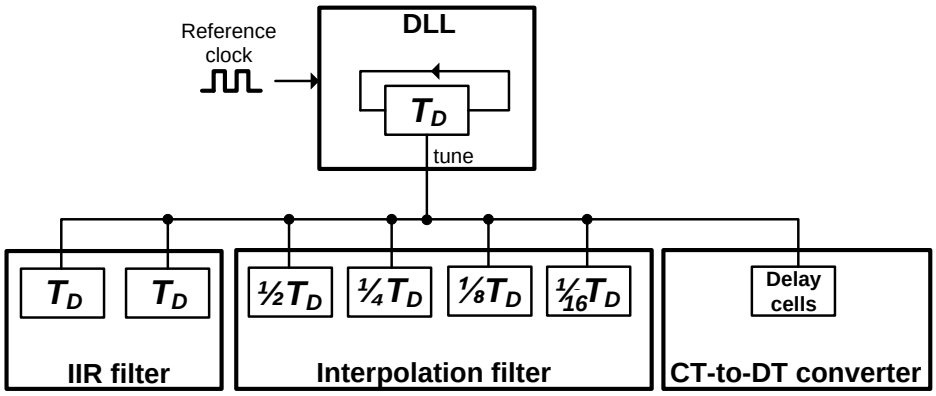


Figure 6.6: Example setup of automatic tuning of the delay lines.

mechanism [18, 40]. Fig. 6.6 shows an example of the automatic tuning setup. The bias voltages of all delay elements in the IIR filter, the interpolation filter and the CT-to-DT converter are tied to the same node, the voltage of which is provided by a delay lock loop (DLL). The DLL contains a T_{D} delay line configured as an oscillator. The cycle time of the oscillation is compared to a 1 MHz reference clock. When the oscillation cycle time equals to 1 µs, the DLL locks and the voltage on its tune terminal is the desired bias voltage for the other delay lines. Although we did not implement an automatic tuning system on our chip, below we describe a manual tuning that emulates this automatic tuning.

We configure the delay elements in the CT digital IIR filter as shown in Fig. 6.7. The grouping block and the first tap delay are configured as a closed loop, acting like the oscillator in the DLL in Fig. 6.6. The second tap delay is configured as an open loop. It takes the output from Tap1 as its input. At the beginning of the calibration, we feed a single pulse into the system through req_{in}. Since the system contains a close loop, the pulse keeps going through the loop with a cycle time of the loop delay. The target of the calibration is to set the loop delay to be 1 µs. In the meantime, we

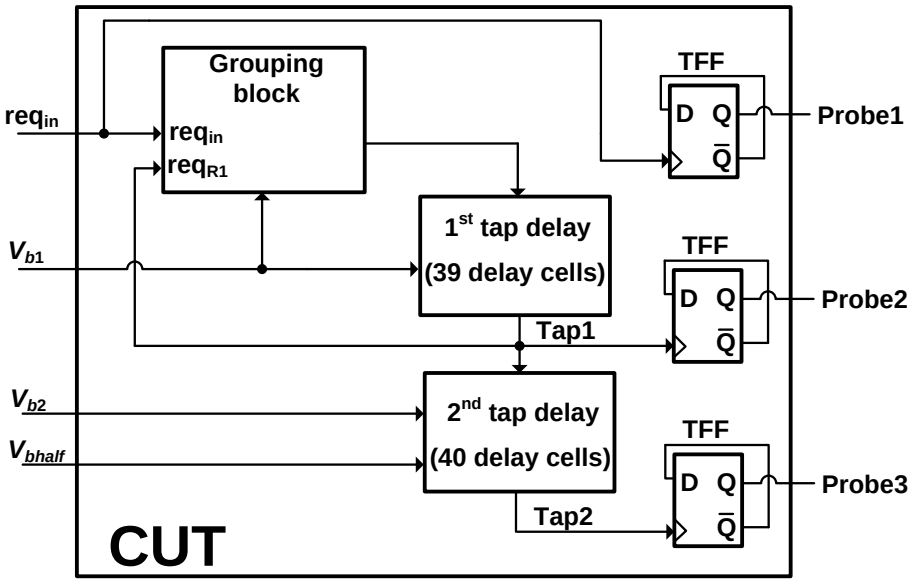


Figure 6.7: Configuration for delay-line calibration.

need to make sure that the delay of the second tap delay is within (1 µs, 1.025 µs). This condition guarantees that the events arriving at Tap2 are always within the grouping windows triggered by their paired events arriving at Tap1 (please refer to Section 3.2 for the details). Since pulses are very narrow and difficult to read out of the chip, req_{in}, Tap1 and Tap2 connect to a toggling flip-flop (TFF), which converts pulses to edges.

The delays are measured with the aid of an oscilloscope. Fig. 6.8 shows an example of the waveforms captured by the oscilloscope. The top, middle and bottom curves are the measurements at probe1, probe2 and probe3 in Fig. 6.7, respectively. The loop delay is the timing distance between any two successive edges on the middle curve, which is calculated by the oscilloscope's internal clock. The value is displayed on the right-most column in Fig. 6.8. The delay of the second tap delay is the timing distance between an edge on the middle curve and the following edge with the same direction on the bottom curve. By tuning the three bias voltages (V_{b1}, V_{b2} and V_{bhalf}),

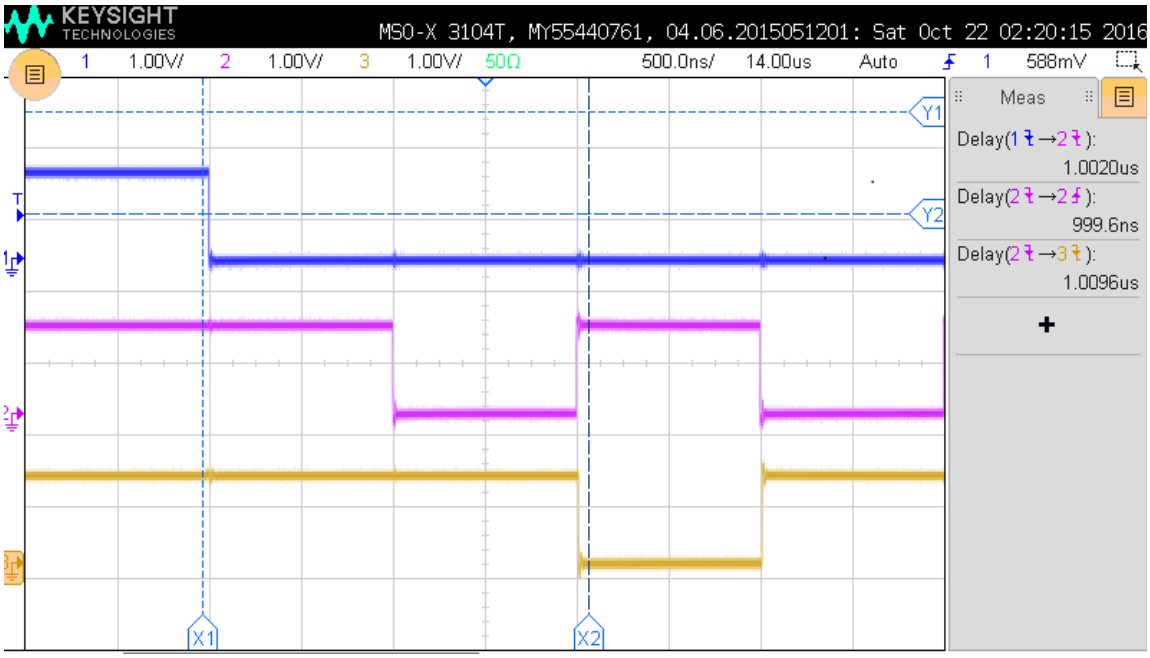


Figure 6.8: Delay-line measurements after tuning.

we can tune the delays to the desired values. Normally, all three bias voltages are connected together. We change the shared bias voltage to adjust the absolute delay values of the delay lines. The system relies on the matching of the delay lines to meet the relative timing requirement. The reason for keeping the three tuning voltages separate is to have the flexibility to tune the two delay lines in the IIR filter separately in case matching is poorer than expected. In the measurement, we find that when the bias voltages are connected together and equal to 0.627 V, the loop delay and the delay of the second tap delay are 999.6 ns and 1009.6 ns. This satisfies the relative timing requirement between the two delay lines. Thanks to the event grouping, as long as the relative timing requirement is fulfilled, the frequency response of the IIR filter is insensitive to mismatches between the two tap delays in the IIR filter. The response is only determined by the loop delay in

Fig. 6.7 and the tap coefficients. It is impossible to tune the loop delay to exactly 1 µs. However, this small difference (0.04%) has a negligible effect on the filter's frequency response.

The delay lines in the interpolation filter also share the same bias voltage as the delay lines in the IIR filter. The exact frequency locations of the notches implemented by the interpolation filter depend on the absolute values of the delay lines in the interpolation filter. Their accuracies rely on the matchings of the delay lines. On the other hand, the exact locations of these notches have a negligible effect on the in-band (i.e., [0, 500 kHz]) frequency response of the entire system. This is because all the notches are far from the in-band range. The in-band frequency response of the interpolation filter is always close to 0 dB.

6.2.2 Timing-dependent delay

The results reported in this chapter were collected in two measurements. The first measurement was done from July to August in 2016; the second was in December 2016. A timing-dependent delay was observed in the second measurement. After a delay line was calibrated using the above procedure, the exact delay amount of the delay line varied with the time. For example, we calibrated the second delay line in the IIR filter to have an initial delay value of 1.016 µs. Then we fed dense input events into the delay line. The input events had a uniform event spacing of 40 ns, which is slightly larger than the granularity of the delay line. Fig. 6.9 plots the measured delay value versus the index of the event. As we can see, later events see shorter delays than the early events. The delay value finally becomes constant with a value around 1.000 µs. This phenomenon was only observed during the second set of measurements, and not in the first set of measurements. Unfortunately, we

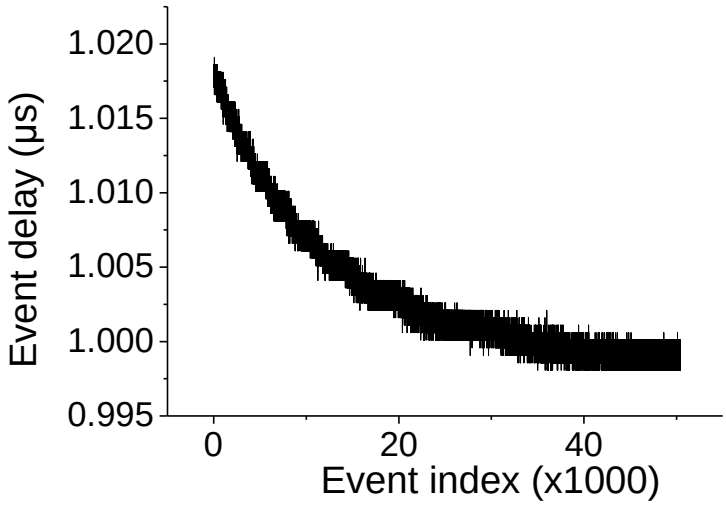


Figure 6.9: Event delay versus event index.

weren't able to find the reason of it; it is possible that some sort of changes occurred in-between the two sets of measurements. To tackle this timing-dependent delay problem in the measurement, during calibration, we set the values of the delay lines to be slightly longer than the desired values. This assured that the delay line can have the desired delay values during normal operation.

6.3 Frequency response

The CT digital IIR filter's frequency response is measured after the above calibrations are completed. The CT digital input signal is generated from the DT-ADC with the DT-to-CT conversion. The input event rate, determined by the ADC's sampling clock frequency, is 20 MHz. A full-scale analog signal is supplied at the input of the DT-ADC. Its frequency is swept from 10 kHz to 980 kHz. Two different digital outputs are collected: one from the output of the interpolation filter, the other from the output of the CT-to-DT converter (see Fig. 6.5). The signal at the output of the

interpolation filter is CT digital. A data-acquisition device with a 500 MHz sampling frequency is used to acquire the signal. Because the sampling frequency is much higher than the bandwidth where the majority of the signal power is located (980 kHz), the acquired DT digital signal is a very good representation of the original CT digital signal (the reason is given in Section 2.3). As an example, in Fig. 6.10(a), we show an acquired output signal when the analog input signal is 55 kHz. We do postprocessing in the computer to convert the digital data to their corresponding amplitude values. The output of the CT-to-DT converter is a DT digital signal, which is synchronized with a clock of 1 MHz (this clock is different from the sampling frequency of the DT-ADC). An example of the acquired DT output is shown in Fig. 6.10(b). Both responses are found by FFT analysis with a Hann Window [50]. The frequency responses of the CT digital IIR system when the IIR filter is configured as low-pass, high-pass, bandpass and bandstop are plotted in Fig. 6.11. Once the output is converted in a synchronous digital signal, the frequency response can only be measured up to half of the clock frequency, i.e., 500 kHz. In the low-pass and bandpass cases, the first repetitive passband around $1/T_{\mathrm{D}}$ is suppressed by the notches formed in the interpolation filter.

In the low-pass and high-pass frequency responses, the stopband rejection is better than 80 dB; this is a 45 dB improvement over previous work [3, 15], due to both the IIR response and the larger number of internal bits retained. With the low-pass configuration, the *SNDR* for full-scale ($1.6\,V_{pp}$) input signals in the passband is 54 dB or above.

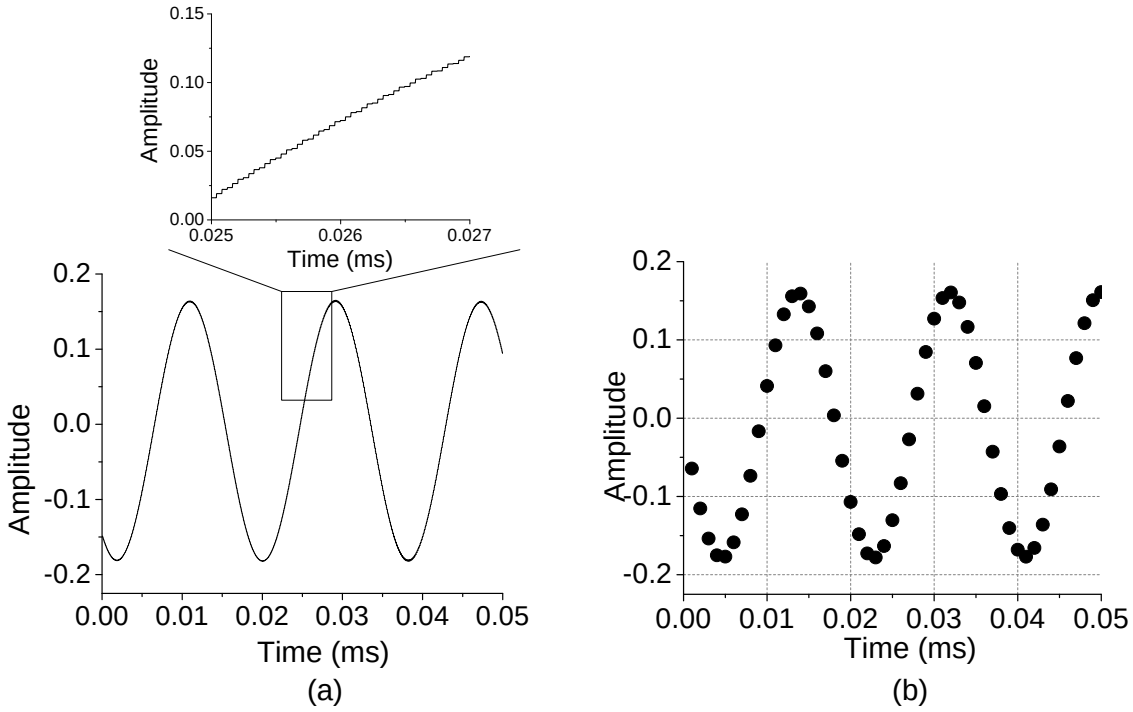


Figure 6.10: Comparison of the CT and DT waveforms at the interpolation filter and the CT-to-DT converter. (a) An example CT digital waveform acquired at the output of the interpolation filter. The zoomed-in plot shows more details of the waveform. (b) An example DT digital waveform acquired at the output of the CT-to-DT converter. Both waveforms are postprocessed by computer.

6.3.1 Two-tone test

The linearity of the designed filter is also tested with full-scale two-tone inputs. The in-band and out-band IM3 components are -59 dB and -58 dB respectively. Fig. 6.12 shows the input and output spectra of the filter with two out-band full-scale tones at 970 kHz and 980 kHz. These two tones are in the repetitive passband of the IIR filters frequency response, but are attenuated by the notches implemented in the interpolation filter. The output spectrum clearly demonstrates the alias-free feature of the CT digital filter. No components are aliased back into the baseband. By

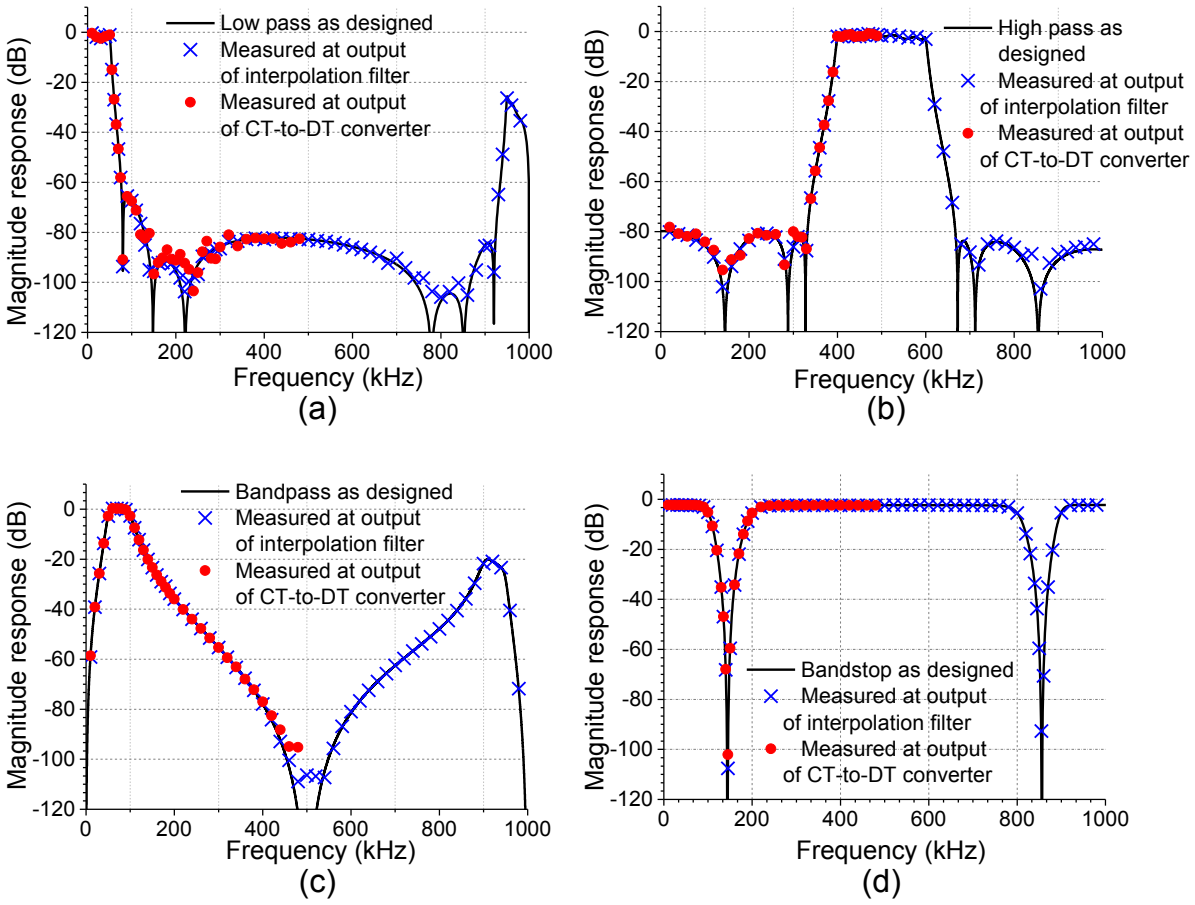


Figure 6.11: Frequency responses. (a) Low pass. (b) High pass. (c) Bandpass. (d) Bandstop.

contrast, if the same input is fed into a DT digital filter with the same frequency response, full aliasing will be observed.

6.3.2 Effect of the interpolation filter

The effect of the interpolation filter is better illustrated in Fig. 6.13. The gray curve shows the spectrum at the input of the interpolation filter (i.e., the output of the CT digital IIR filter) when the IIR filter is configured for low-pass operation as in Fig. 6.11(a). The testing input is a full-

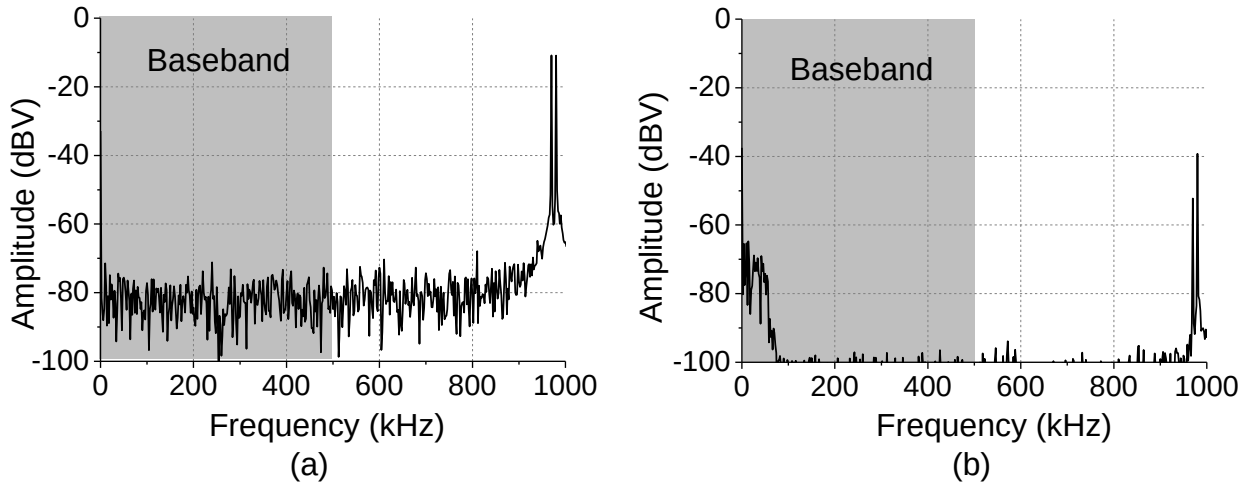


Figure 6.12: Out-band two-tone test. (a) Spectrum of the filter input. (b) Spectrum of the filter output, before CT-to-DT converter.

scale 5 kHz sine wave sampled at 20 MHz. With such a high oversampling ratio, the resulting DT digital signal has a spectrum very similar to a CT digital signal (see Section 2.3). Noise and distortion power are preserved at the repeating passbands, which are around 1 MHz, 2 MHz, ... The black curve shows the spectrum at the output of the interpolation filter. Clearly, the noise and distortion power are suppressed by the notches formed in the interpolation filter. Conducting FFT analysis on both signals, we find the *SNDR*s at the interpolator input and output from 0 to 250 MHz (the sampling frequency of the data acquisition device is 500 MHz) are 47.9 dB and 56.5 dB respectively. The interpolation filter increases the *SNDR* by 9.6 dB.

6.4 Power consumption

We measured the CT digital IIR filter system's power consumption under different input event rates. The system is configured as a low-pass filter whose frequency response is the same as shown

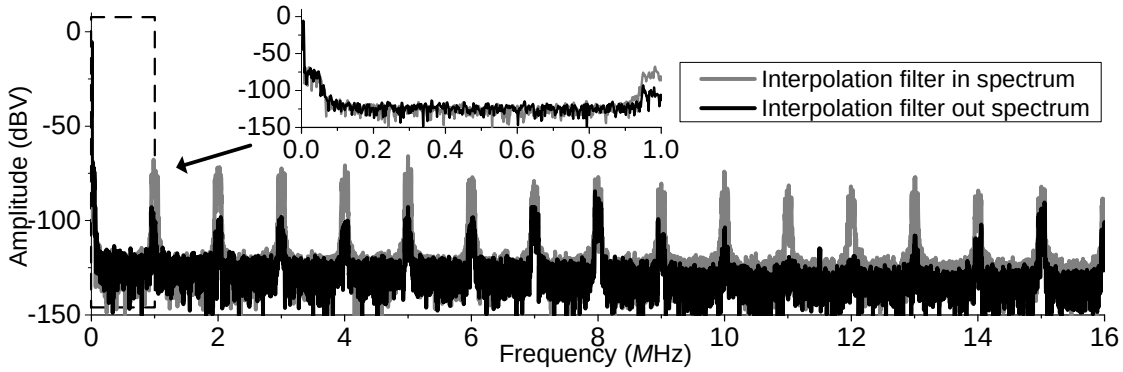


Figure 6.13: Spectra at input and output of the interpolating filter.

in Fig. 6.11(a). The input event rate is varied by sweeping the sampling clock frequency of the DT-ADC from DC to 20 MHz. The power consumption of the system versus the input event rate is shown in Fig. 6.14. When the event detector is on, it eliminates redundant events in the feedback loops. Power strongly decreases when input activity decreases. When the event detector is off, redundant events keep going around the feedback loops and maintain a high event rate regardless of the input. On the other hand, when the input rate is low and the event detector is off, the data values in the system are less likely to change. Hence, the power is still weakly dependent on the input. The power consumption of the system when the input event rate is 20 Msps is broken down in Table 6.1.

6.4.1 Test with a speech signal

We also tested the CT digital IIR system with a speech signal. The speech signal is converted to a CT digital signal by the LCS CT-ADC on the separate board. Events on the CT digital signal are generated only when the speech signal crosses one of the ADC's quantization levels. The CT

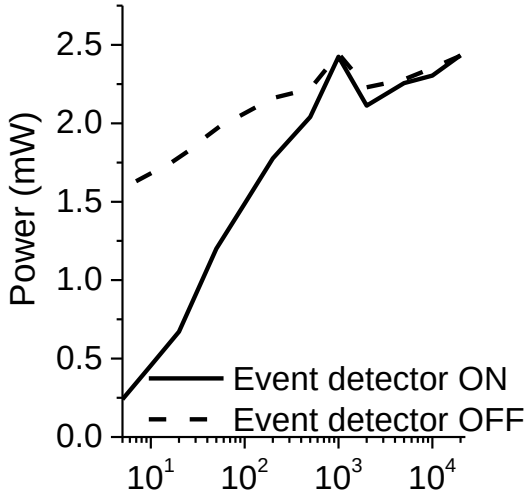


Figure 6.14: Power consumption versus input event rate.

digital IIR filter is configured as a low-pass filter with the same frequency response as it is in Fig. 6.11(a). The event detector is turned on, and 14 bits are used for data comparisons in the event detector. The speech signal and the system's instantaneous power consumption are shown in Fig. 6.15. When the input is active, the power can go high to 600 μW. When the input is static, the DSP settles to a quiescent state with redundant events eliminated by the event detector, dropping the power to the leakage level (40 μW). The average power is 83 μW.

Different from the CT digital FIR filters whose power consumption is determined only by the input activity [3, 15, 17], the power consumption of a CT digital IIR filter is determined by both the input activity and the activity in the filter's feedback loops. This explains why the power temporarily goes up when the input becomes less active at the end of the first burst of speech.

Dynamic power @ 20MHz input data rate (mW)	2.32
IIR filter	1.62
Delay line	0.41
SRAM	0.82
Arithmetic blocks	0.40
Interpolation filter & CT-to-DT converter	0.70
Leakage (mW)	0.04

Table 6.1: Power breakdown.

6.5 Performance summary and comparison to other work

The parameters of the CT digital IIR system are summarized in Table 6.2. The delay line is only a small portion of the system, both in area and power. The delay line area per tap is one third that in [3].

Table 6.3 compares the presented work to CT and DT digital processors in prior art [3, 15, 17, 51–53]. A conventional figure of merit, FoM_1 [17], for comparing different signal processors is given by

$$FoM_1 = \frac{Power}{f_{\max} 2^{ENOB} N}, \tag{6.1}$$

where $f_{\max}$ is the maximum input event rate, N is the filter order, and $ENOB = \frac{SNDR-1.76}{6.02}$. One issue with FoM_1 is that it treats the filter order for FIR and IIR filters in a same way. However, it is well known that to satisfy a given filtering requirement, IIR filters generally require much lower order than FIR filters [54]. Hence, treating their order in a same way results in a figure of merit which favors FIR filters. To tackle this problem, we introduce a modified figure of merit, FoM_2, as

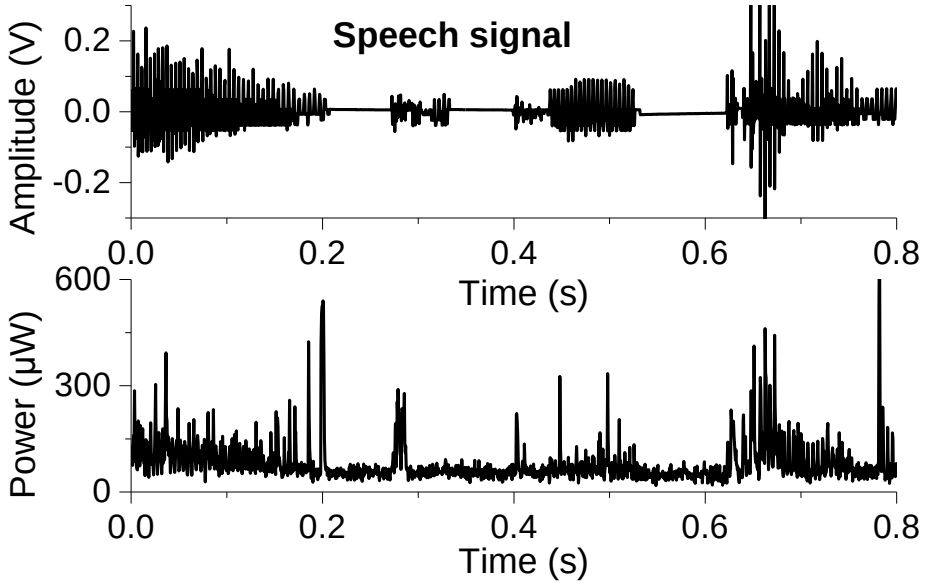


Figure 6.15: Plot of input speech signal (top) and the instantaneous power consumption (bottom) of the chip.

follows:

$$FoM_2 = \frac{FoM_1}{K}. \tag{6.2}$$

When the filter is a FIR filter, $K = 1$ and $FoM_2 = FoM_1$; if it is instead an IIR filter, K is the ratio of the orders of a FIR filter and an IIR filter when the two satisfy the same filtering requirement. For example, to implement the low-pass frequency response plotted in Fig. 6.11(a), which has $T_D = 1\,\mu s$, passband edge at 50 kHz, stopband edge at 88 kHz and stopband rejection more than 80 dB, the minimum orders required by a FIR filter and an IIR filter are 67^{th} and 6^{th} respectively. Hence, $K = \frac{67}{6} = 11.2$.

Table 6.3 shows that FoM_2 of our system beats all the CT digital designs in the prior arts including the state-of-the-art design in Ref. [17]. In addition, for the first time, it implements an IIR architecture, signal-derived timing, and synchronous output. The system's 16-bit resolution is

Process/supply voltage	65 nm/1.2 V
Input resolution	7 bits
Filter coefficient resolution	10 bits
DSP arithmetic/output res.	16 bits
T_D/t_g	1 µs/25 ns
Max. FIFO depth	128
Max. input data rate	20 MHz
SNDR*	57.7 dB
Core area	0.64 mm^2
IIR filter	0.38 mm^2
Delay line	0.05 mm^2
Interpolation filter	0.18 mm^2
CT-to-DT converter	0.08 mm^2

*Measured at output of the CT-to-DT converter, when the input is a full-scale 5kHz 7b sine wave and the IIR filter is low pass with T_D=1us and a cutoff frequency of 50kHz.

Table 6.2: Summary of the chip.

much higher than the others [3, 15, 17, 51]. Its FoM_2 also compares well with those of DT digital filters, but it avoids the clock and hence is free of aliasing. It features an agile power adaptation to input activity, which varies from 2.32 mW (full activity) to 40 µW (idle), with more than a 50× range, with no power-down circuitry.

Parameter	O'hAnnaidh	Tohidian	Agarwal	Schell	Kurchuk	Vezyrtzis	**This work**
Signal domain	DT analog		DT digital	CT digital			**CT digital**
Output/ clock freq.	Analog	Analog	Sync/ 2.1 GHz	Async.	Async.	Async.	**Sync/ 1 MHz***
Type/order	FIR/15th	IIR/7th	FIR&IIR/3rd	FIR/15th	FIR/5th	FIR/15th	**IIR/6th**
Process	45 nm	65 nm	32 nm	90 nm	65 nm	130 nm	**65 nm**
Supply	1.1 V	1.2 V	1 V	1 V	1.2 V	1 V	**1.2 V**
DSP arithmetic resolution	6 bits	8 bits	8 bits	8 bits	4 bits	8 bits	**16 bits**
Stop-band rejection	22 dB	>100 dB	NA	35 dB	15 dB	25 dB	**80 dB**
SNDR	33 dB	41 dB	50 dB	47-62 dB	20-22 dB	40-51 dB	**>54 dB**
Max. input rate	3.2 Gsps	800 Msps	2.1 Gsps	16.7 Msps	45 Gsps	10 Msps	**20 Msps**
Power (Min/Max)†	- / 48 mW**	- / 1.96 mW**	- / 24 mW**	310 µW/ 3 mW	270 µW/ 9.2 mW	70 µW/ 3.3 mW	**40 µW/ 2.32 mW**
FoM_1 (Min/Max)†	- / 27 fJ	- / 4 fJ	- / 15 fJ	1.2 fJ/ 12 fJ	0.1 fJ/ 4.3 fJ	1.6 fJ/ 76 fJ	**0.6 fJ/ 33 fJ**
FoM_2 (Min/Max)†	- / 27 fJ	- / 0.36 fJ	- / 1.3 fJ	1.2 fJ/ 12 fJ	0.1 fJ/ 4.3 fJ	1.6 fJ/ 76 fJ	**0.05 fJ/ 2.9 fJ**
Antialiasing filter required for complete system?	Yes	Yes	Yes	No	No	No	**No**

***For CT-to-DT converter only. ****Not including clock generation and anti-aliasing filter.
†Varying with signal activity; without power down circuitry.

Table 6.3: Comparison of presented work with state-of-art CT and DT digital filters.

Chapter 7

Design Considerations for VR Digital Signal Processing

7.1 Introduction

As already mentioned in Chapter 1, conventional fixed-rate ADC and DSP systems have to operate at the Nyquist frequency (i.e., twice the input bandwidth) or faster to avoid aliasing. However, for most real world signals, there are variations in the frequency content within the observation interval [8]. In practice, a spectrum is obtained by applying FFT on a finite-length interval of a signal. We define the width from zero to the maximum frequency which has a significant power as the "local bandwidth." For example, speech signals contain frequencies up to 10 kHz, but in most of the time their "local bandwidth" is up to 3.5 kHz [1]. Extensive studies have been done on the ADC side to take the advantage of a varying "local bandwidth." We only list a few here [1, 8–11]. The

VR-ADCs implemented in these works acquire samples with a sampling rate changing according to the "local bandwidth." Thus, a good amount of power is saved due to a low average sampling rate, while information is still accurately preserved.

On the other hand, the question as to how to process VR samples remains open. [9] develops an algorithm to extract features from an adaptively-sampled ECG signal. However, the techniques used in this specialized application do not apply in the general case. [10] introduces an integrated solution to do VR sampling and filtering. However, it requires a complicated algorithm to determine the clock rate, which adds a large power overhead. Hence, there is a need for an energy-efficient DSP which can process samples with a variable sampling rate. Its clock rate has to track the input sampling rate to maintain a low average activity, while its transfer function and signal-to-noise ratio should be independent of the sampling rate.

A conventional DSP does not satisfy the above requirements. Fig. 7.1 shows a VR-ADC followed by a conventional DSP. Since the focus of this work is on the DSP side, the details of the ADC are omitted for simplicity. Only a sampler with a VR clock f_{s} is shown. The DSP is implemented as a K^{th} order FIR filter with tap coefficients $h[k]$, $k = 0, \ldots, K$. Its tap delay T_{s} and the sampling period are completely linked, and in fact they are equal, and this is at the root of the problem: depending on the input "local bandwidth," f_{s} varies and thus the frequency response scales with the sampling frequency.

This chapter is based on the ISCAS paper [55].

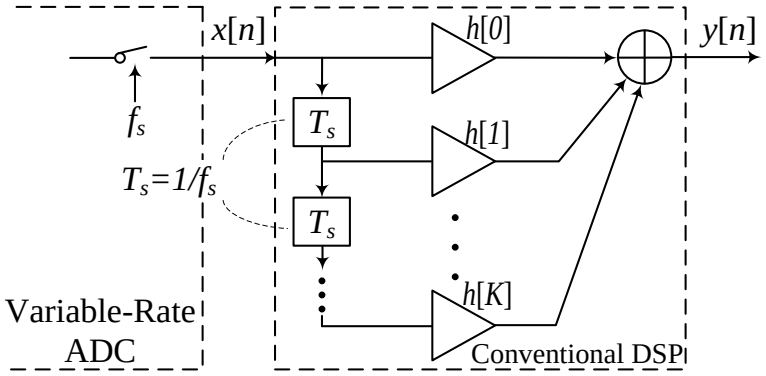


Figure 7.1: A VR-ADC followed by a conventional DSP.

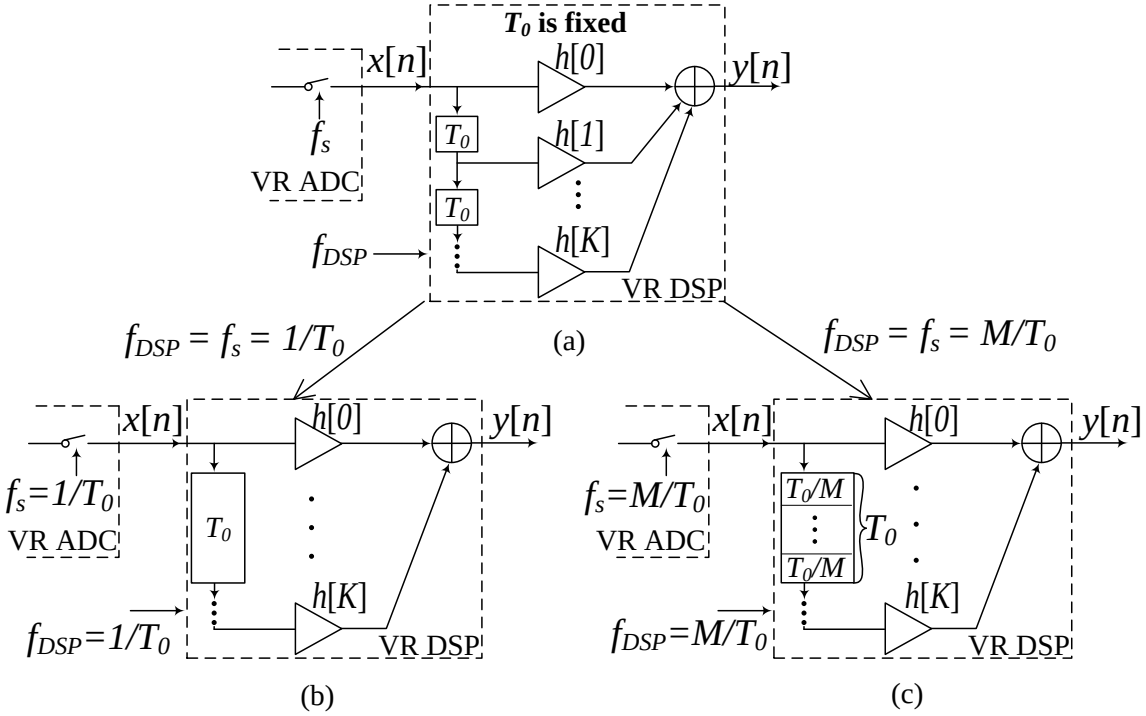


Figure 7.2: Variations on a VR-DSP. (a) The basic model. (b) In a constant-sampling-rate region with $f_s = f_{DSP} = 1/T_0$. (c) In a constant-sampling-rate region with $f_s = f_{DSP} = M/T_0$.

7.2 VR-DSP

The solution to the problem just mentioned is to *decouple the tap delay from the sampling clock.*

CT-DSPs implemented in [15] and the previous several chapters of this thesis have this feature.

This work discusses a way to do this in DT. We introduce a VR-DSP shown in Fig. 7.2(a). The clock rate of the VR-DSP is denoted by f_{DSP}. It is equal to the sampling clock f_{s} for most of the time, with some exceptions which will be discussed in Section 7.2.2. *Its tap delay T_0 is independent of the choice of f_{s}.* Implemented as a K^{th} order FIR filter, it has a fixed transfer function:

$$H(s) = \sum_{k=0}^{K} h[k](e^{sT_0})^{-k} \tag{7.1}$$

We now consider several special cases.

7.2.1 Constant-sampling-rate regions

A constant-sampling-rate region in a VR-DSP is defined as the case in which the input sampling rate has been constant for a long time. In that case, all the samples stored in the delay line of the DSP have the same timing distance. In this region, f_{DSP} is equal to f_{s} in Fig. 7.2(a). Fig. 7.2(b) shows the case where these quantities are selected to be $1/T_0$[1]. The operation in this case becomes the same as that of a conventional DSP. Because T_0 is equal to one clock cycle, the VR-DSP configures its delay line such that each tap contains one delay cell. The delay cell realizes a delay equal to one clock cycle. The output $y[n]$ is computed from

$$y[n] = \sum_{k=0}^{K} h[k]x[n-k]. \tag{7.2}$$

The transfer function corresponding to this equation is identical to (7.1).

[1] How the sampling frequency is determined according to the local bandwidth is not a focus of this paper. Readers interested in it can refer to [1, 8–10].

Fig. 7.2(c) shows the case when a fast sampling clock is used. Assume $f_s = f_{DSP} = M/T_0$, where M can be any integer larger than 1. Because T_0 is equal to M clock cycles, a VR-DSP configures its delay line such that each tap is composed of M delay cells. Each cell realizes a delay equal to one sampling cycle. Since one input sample is generated in each cycle, each delay cell stores one sample. The output $y[n]$ is computed from:

$$y[n] = \sum_{k=0}^{K} h[k]x[n-Mk] \tag{7.3}$$

This corresponds to the transfer function:

$$H(s) = \sum_{k=0}^{K} h[k](e^{s\frac{T_0}{M}})^{-Mk} = \sum_{k=0}^{K} h[k](e^{sT_0})^{-k}$$

which is again identical to (7.1). The key for a VR-DSP to implement a transfer function independent of f_s is that it can adaptively configure its delay line so that a constant tap delay T_0 is kept.

7.2.2 Sampling-rate-transition regions

A sampling-rate-transition region starts when f_s changes. The time intervals between samples before and after this change are different. Since both intervals will coexist in the delay line for a certain amount of time, the VR-DSP needs to properly configure its delay line to maintain a constant tap delay T_0, as well as to avoid distortion. In the following discussion, we assume the sampling

rate switches between two options $1/T_0$ and M/T_0 for simplicity. The principles presented are also valid for more sophisticated scenarios.

A fast-rate-to-slow-rate transition

Consider a VR-DSP operating in a constant-sampling-rate region with $f_s = M/T_0$. A fast-rate-to-slow-rate transition starts when the DSP detects the sampling rate decreasing to $1/T_0$. However, the DSP clock rate f_{DSP} is not allowed to change immediately: because the minimum timing distance between two successive samples in the delay line is T_0/M, the clock has to maintain a T_0/M cycle time to preserve this information. Otherwise, if a slower clock is used, the inherent timing information in the sample sequence will be distorted. In fact, this transition period is the only time when f_{DSP} is different from f_s. Since the timing distance in the following samples is T_0, which is equal to M clock cycles, any two consecutive samples have to spaced by M delay cells. This leaves vacant spots in-between. In each of the $M-1$ cycles following the time f_s switches, a dummy sample equal to the previous input sample is used to fill in the vacant spot at the input of the delay line. In the M^{th} cycle, a new input sample is generated. Thus, no dummy filling is needed. The operations in these M cycles then repeat $K-1$ times. In the end, the delay line of the VR-DSP contains K new input samples, with $M-1$ dummy samples inserted between any successive two of them. Because all these K input samples are spaced by T_0, the VR-DSP decreases f_{DSP} to $1/T_0$. Following this, it reconfigures the delay line to have one delay cell in each tap and discards all the dummy samples. This does not affect the timing distance between the samples because each delay

cell realizes a delay of T_0. From then on, the VR-DSP enters a constant-sampling-rate region with $f_s = f_{DSP} = 1/T_0$.

A slow-rate-to-fast-rate transition

A slow-rate-to-fast-rate transition occurs when a VR-DSP detects that f_s increases from $1/T_0$ to M/T_0. The procedure is just the opposite of the fast-rate-to-slow-rate transition. Originally, each tap contains one delay cell and stores one sample. As soon as the change is detected, each tap is reconfigured to contain M delay cells. At the same moment, each existing sample is copied $M-1$ times to fill up the vacant spots in the tap it belongs to. In the meantime, f_{DSP} increases to M/T_0 to be consistent with f_s. From then on, the VR-DSP operates with a fast clock rate of M/T_0.

To summarize the operations in sampling-rate-transition regions, we emphasize that as long as there are any samples acquired with a fast sampling rate in the delay line, the VR-DSP operates at the same fast rate to preserve the timing information in the samples. A VR-DSP with more than two clock rate options would have to operate with the highest rate at which the samples in the delay line are acquired.

7.2.3 An illustrative example

As an illustration, we consider an 80 ms input signal. As indicated at the top of Fig. 7.3, before 20 ms and after 40 ms, the input frequency is 4.1 kHz and $f_s = 50$ kHz. The signal in-between has a frequency of 16.4 kHz and $f_s = 200$ kHz. It is not necessary to use those high sampling frequencies in VR-DSPs; we choose this combination for illustration purpose as, within a short duration, a large

amount of samples are acquired, allowing the details to become visible. The waveforms around the two transition instants are shown in Fig. 7.3 (a) and (b). Because we know the exact instants when the input "local bandwidth" switches, we can make them coincident with the sampling frequency switching instants to simplify our discussion. Note that the details of how the sampling rate is switched in the input ADC is not the subject of this paper, the reader is referred to [1, 8–10]. The VR-DSP is implemented as a 10^{th} order low-pass FIR filter with a cutoff frequency at 18 kHz. Several filtered outputs are shown in Fig. 7.3 (c)-(f). Ideal outputs are shown by solid lines; these are generated by feeding the input signal into a CT filter with the same transfer function. (c) and (d) plot the output samples in the slow-rate-to-fast-rate and fast-rate-to-slow-rate transitions, respectively. In the parts far from the transition instants, the output samples from the VR-DSP match the ideal curve very well. Deviations from the ideal curve are observed in the transition parts; error is introduced by dummy samples. Dummy samples are not directly acquired from the input signal, but are reconstructed from the previous input sample, as discussed above. This introduces an error between the sample sequence and the input signal.

We also consider a fixed-rate DSP with a clock rate of 200 kHz for comparison. Its output samples in two transient periods are plotted in (e) and (f). Although a fixed-rate DSP generates better results in the transient regions, a VR-DSP has a much lower average output sample rate. In this case, the average sample rate of the VR-DSP is 2.3$\times$ lower than the fixed-rate DSP, which implies significant savings of computations and thus of power consumption. In general, the average sample rate of a VR-DSP is input dependent. A shorter duration of the high-frequency part of the input results in a lower average sample rate. For example if the length of the 16.4 kHz part in the

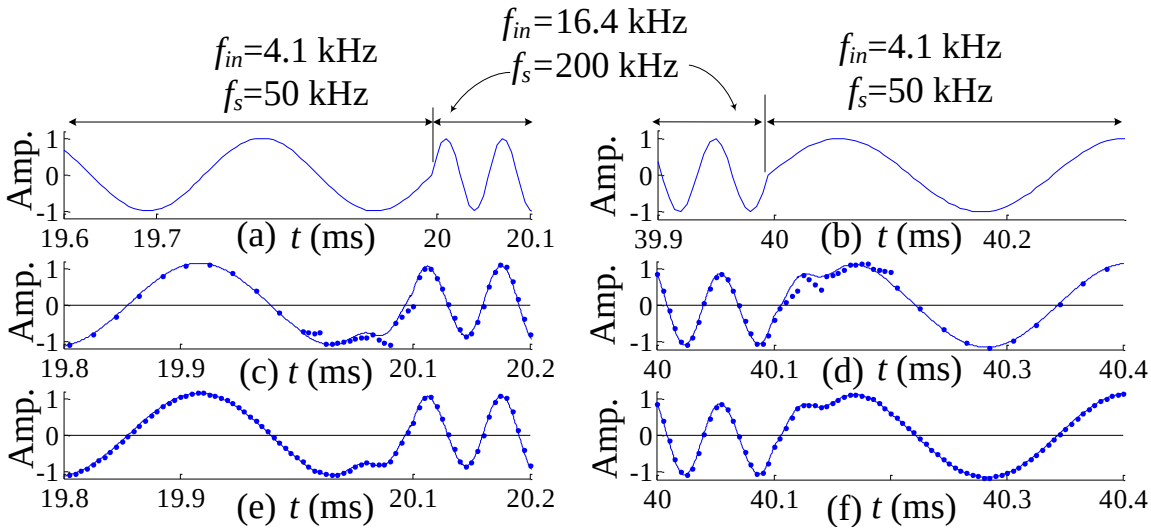


Figure 7.3: Sample input and output signals around frequency transitions. (a) Low to high frequency. (b) High to low frequency. In (c)–(f), the solid line shows the ideal waveform for reference; the dots represent the output samples from different DSPs. (c) A slow-rate-to-fast-rate transition in a VR-DSP. Both f_s and input frequency change at $t = 20$ ms. (d) A fast-rate-to-slow-rate transition in the VR-DSP. Both f_s and input frequency change at $t = 40$ ms. However, the output rate doesn't change until the end of the transition, which is at about $t = 40.2$ ms. (e) A slow-input-to-fast-input transition in a fixed-rate DSP. (f) A fast-to-slow transition in a fixed-rate DSP.

example above shrinks from 20 ms to 5 ms, the output rate becomes 3.3× lower than the fixed-rate DSP.

7.2.4 Remarks on required rate

A VR-DSP with a transfer function described in equation (7.1) has a repetitive frequency response, with a period of $1/T_0$. Within each frequency response period, the frequency responses in the two half periods are symmetrical. Thus, the maximum input bandwidth that the DSP can process is $1/2T_0$. On the other hand, since T_0 is the unit delay in (7.1), $f_s = 1/T_0$ is the lowest sampling rate one can use in the ADC. Any sampling rate lower than $1/T_0$ will cause the repetition of the frequency response to start earlier than $1/T_0$. Fig. 7.4(a) shows the case when the "local bandwidth"

of an input is low. Because of the frequency response constraint, the minimum sampling rate has to be $1/T_0$. Fig. 7.4(b) shows the case when the "local bandwidth" reaches the maximum, i.e., $1/2T_0$; this demands a higher sampling rate, M/T_0, with M an integer larger than 1. This implies an oversampling ratio of M. When quantization noise is considered, a sampling-frequency dependent noise floor is introduced. Assume the samples have a fixed resolution. Quantization introduces a noise with a total power of $V_{LSB}^2/12$, where V_{LSB} corresponds to the one least significant bit. Consider Fig. 7.4(a), when the sampling frequency is low; the noise floor is high because the total noise power is distributed within a narrow frequency interval $[0, 1/2T_0]$. When the sampling frequency is high, as shown in Fig. 7.4(b), the noise floor is low as the same amount of power is distributed over a wider range $[0, M/2T_0]$. On the other hand, an input with a low "local bandwidth" can be reconstructed by a reconstruction filter with a low cutoff frequency as shown in Fig. 7.4(a). Correspondingly, when the "local bandwidth" of the input is high, a filter with a high cutoff frequency should be used for reconstruction. By linking the cutoff frequency of the reconstruction filter with the sampling frequency of the VR-DSP, we can make sure that a fixed amount of quantization noise power is included in the passband of the reconstruction filter.

7.3 Reconstruction of VR samples

Signal reconstruction is needed in applications where a CT output is required. A CT signal or a uniform-rate sample sequence which is reconstructed from a VR sample sequence allows the use of the Fourier transform. This enables one to do frequency-domain analysis on the signal. In a conventional fixed-rate DSP system, the impulse response of a fixed-shape reconstruction filter is

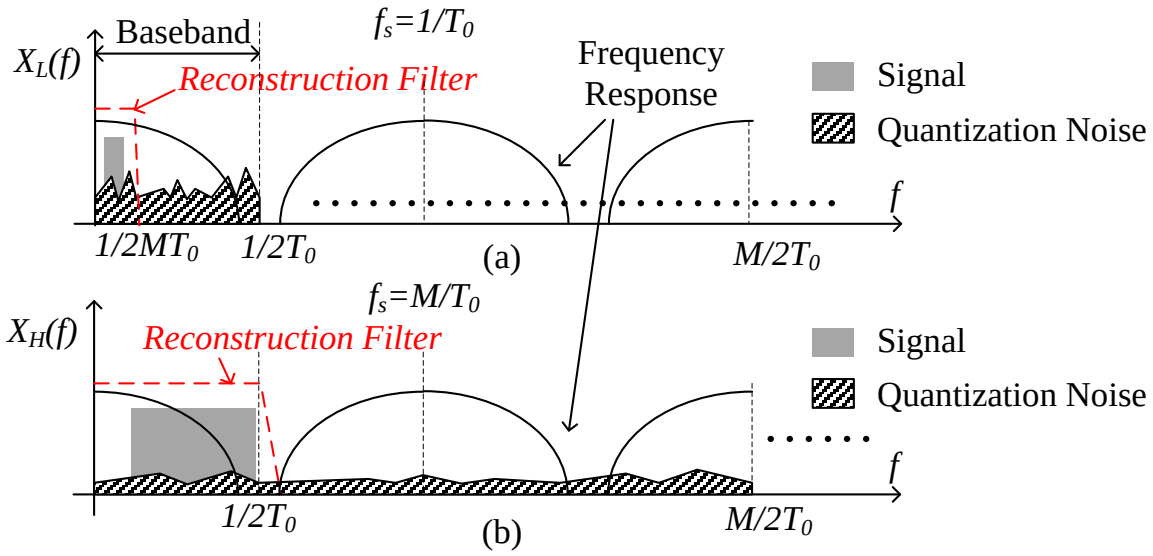


Figure 7.4: DSP frequency response, noise power spectral density, and reconstruction filter frequency response, for low and high sampling rates.

convolved with a uniform-rate sample sequence to generate the reconstructed output. One of the commonly used reconstruction filters is a brick-wall low-pass filter, whose impulse response is a sinc function [56]. An important feature of this function is when it reconstructs an output at one of the sampling instants, its zero-crossing points are coincident with the positions of all the other samples. Thus, the reconstructed output at that point is exactly equal to the sample value. However, this feature is no longer valid when reconstructing a VR sample sequence, whose intersample distance is nonuniform. This introduces error in the reconstructed output at the sampling instants. In order to solve this problem, we modify the sinc function to track the sampling frequency at the reconstruction instant as shown in Fig. 7.5. The stems on the time axis are the time stamps of the samples. Magnitude information is not shown for simplicity. Initially the sampling rate is M/T_0. After $t = t_s$, the sampling rate switches to $1/T_0$. The left and right sinc curves are used to reconstruct outputs at instants which are far from the transition instant t_s. The reconstruction filter

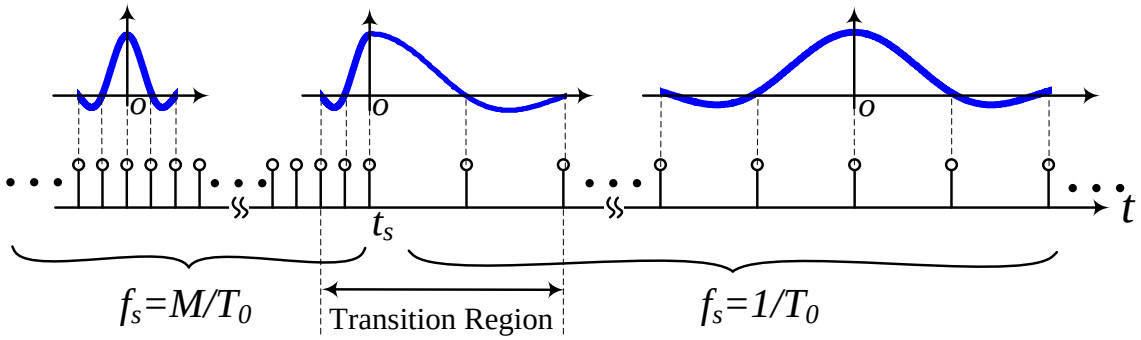


Figure 7.5: Reconstruction using sinc with a variable cutoff frequency at three different instants.

in each region, has an impulse response given by:

$$h_r(t) = \text{sinc}(t f_s) \tag{7.4}$$

In this case, the cutoff frequency of the reconstruction filter is always equal to $f_s/2$. As shown in Fig. 7.5, no matter the sampling frequency is high or low, the distances between two zero-crossing points in these sinc functions are equal to the intersample distances. In practice, only a finite number of samples in the vicinity of the reconstruction instant can participate in the operation. Thus, the sinc function has to be truncated. However, jumps are observed at t close to the switching instant t_s, due to the sudden change of f_s. To reconstruct an output around this transition region, we use a modified approach. We scale the two parts of a sinc function at the two sides of $t = t_s$ with different factors so that the distances between two zero-crossing points are the same as the interval-sample intervals on each side of $f = t_s$. The "sinc" function in the middle of Fig. 7.5 illustrates this.

To show the results of the reconstruction process, we reconstruct the output samples from the VR-DSP in 7.2.3. We compare the reconstructed results both with and without the transition-region modification. In both cases, 101 samples participate in the reconstruction. The results are shown

in Fig. 7.6. A solid line shows the output with the modification, while a dash line is without the modification. The dashed line has large error around the sampling rate switching instant at 20 ms. A zoomed-in plot around that instant is provided to clearly show the discontinuity in the output. On the other hand, the reconstructed output shown as the solid line coincides with the samples at all the sampling instants.

Next we show the reconstructed output against the ideal output in Fig. 7.7. In the constant-sampling-rate regions close to the two ends, the two waveforms match very well. In the sampling-rate-transition regions, an error is obvious. It is introduced by the dummy samples in the VR-DSP operation.

Finally, we consider quantization noise and assume the resolution of the input samples is eight bits. We built a reconstruction filter using the method described above but chose its cutoff frequency to be $f_s/8$. In the high-sampling-rate case in Section 7.2.3, the cutoff frequency is coincident with the edge of the baseband. We used this filter to reconstruct the output of the VR-DSP. The baseband SNR in the constant-sampling-rate regions are summarized in Table 7.1. We also reconstructed the samples from the fixed-rate DSP and listed its SNR for reference. As shown in the table, the SNR of the VR-DSP is almost a constant for different f_s and it is very close to the value in fixed-rate DSP case.

Although the sampling rate in the above example varies only once, the method introduced also applies to cases with multiple changes, provided that the time distances between two changes are apart by at least one transient interval.

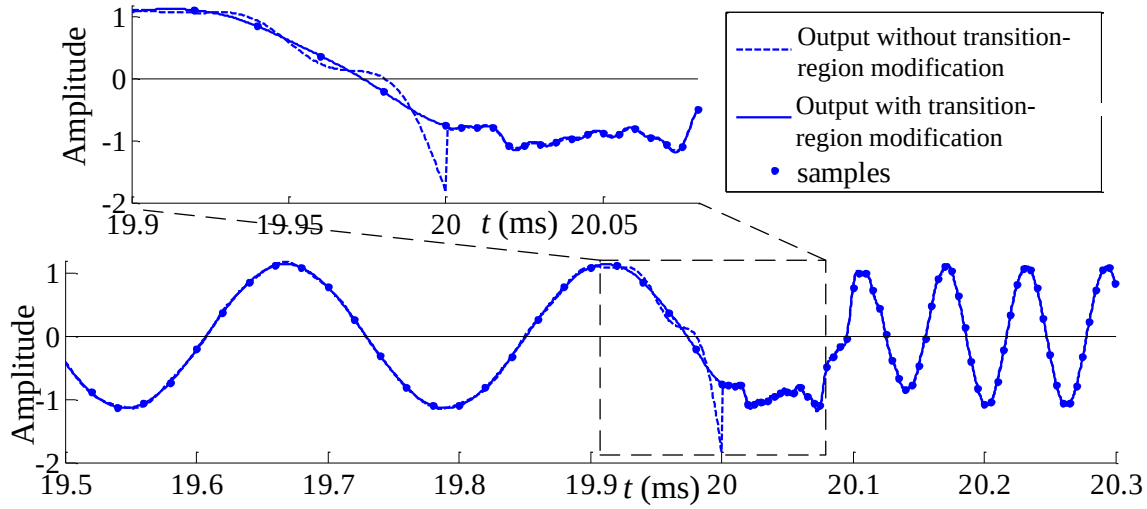


Figure 7.6: Comparison of two reconstruction methods.

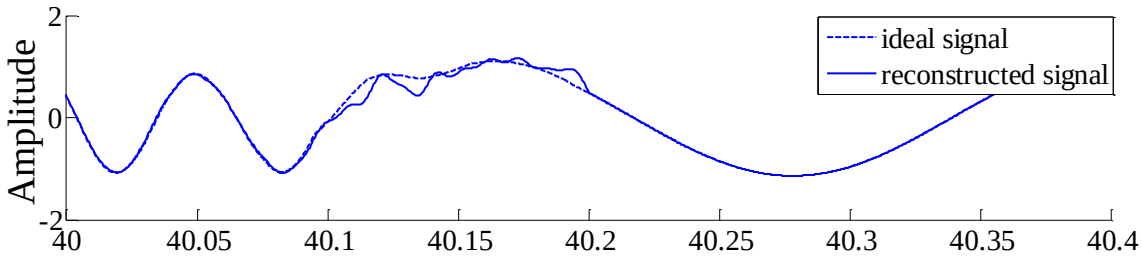


Figure 7.7: Comparison of the reconstructed and the ideal output.

Table 7.1: In-band SNR comparison

	VR-DSP		Fixed-Rate DSP	
Sampling Freq.	200 kHz	50 kHz	200 kHz	200 kHz
Input Freq.	12.8 kHz	4.1 kHz	12.8 kHz	4.1 kHz
SNR	57.8 dB	57.5 dB	57.7 dB	58.7 dB

7.4 Conclusion

A VR-DSP which can process samples with a VR according to the input local bandwidth is described. We also provide a method for accurately reconstructing the output from the output samples. Design considerations and limitations of these systems are discussed in the paper. Simulation

results confirm that a VR-DSP can process signals with a significantly lower average rate compared to fixed-rate DSPs, while maintain a comparable SNR in the output.

Chapter 8

Suggestions for Future Work

This chapter suggests some avenues for improvement to the work presented in this thesis.

8.1 Improvement to event detection in the CT digital IIR filter

As we discussed in Section 3.3.1 and 5.2.3, an event detector is used to monitor the traffic in the feedback paths of a CT digital IIR filter to eliminate redundant events. However, introducing a detector occasionally breaks the assumption of the event-grouping method, that an event arriving at the end of second tap delay can always be grouped with its paired event arriving at the end of first tap delay. An event may arrive at the end of second tap delay while its paired event in the first tap delay is eliminated by the event detector. In this case, our current solution is to throw away the event at the second tap delay. However, this adds error power to the signals. An improvement to the event detection which can avoid introducing this error is desired. A possible solution is shown in Fig. 8.1. In the data path, two FIFOs are used in each of the three second-order sections, in contrast

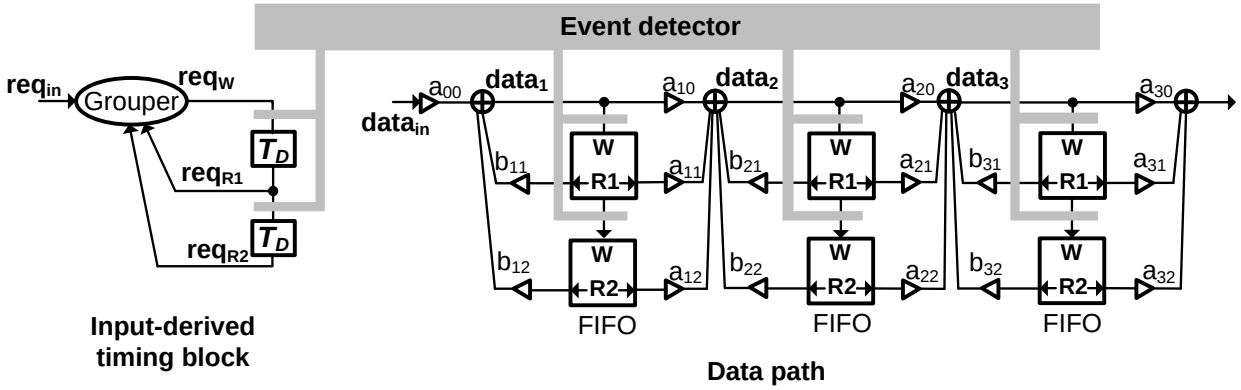


Figure 8.1: A possible implementation of event detection with no error power.

to one each in the original case in Fig. 3.6. The new event detector monitors all the input data which will be written into the six FIFOs and control the inputs of the two delay lines in the timing block. If the event detector detects that an event trying to write to the top three FIFOs is redundant and its paired event trying to write to the bottom three FIFOs is also redundant, the detector eliminates both events from the loop. As long as either of this pair of events is not redundant, both are preserved.

8.2 A CT digital IIR filter with one tap delay

In Chapter 3, we introduced a method that allows one to design a high-order CT digital IIR filter using two tap delays and showed that mismatches between the two delay lines can cause the IIR filter to be unstable. Although an event-grouping method was introduced to address this issue, it still requires the matching of the two delay lines to be relatively good. In this section, we improve the method so that one can design a high-order CT digital IIR filter using only *one* tap delay. A CT digital IIR filter based on this new method is completely free of the matching issue. Consider the shared timing block in a CT digital IIR filter in Fig. 3.5, which we replot in Fig. 8.2(a). An

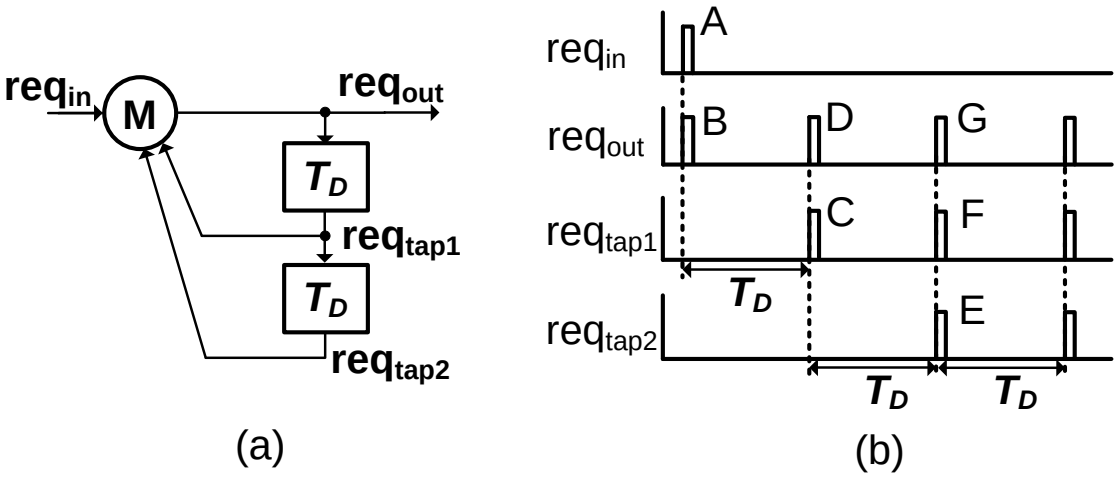


Figure 8.2: A shared timing block. (a) Shared timing block. (b) Operations.

M block is used to replace the grouper in the original plot. M stands for merging block with a zero delay. When an event (here denoted as a pulse on the timing signals) arrives at any of the merging block's three inputs (req_{in}, req_{Tap1} or req_{Tap2}), the block generates a pulse at its output, req_{out}, immediately. Fig. 8.2(b) shows the operations of the timing block after Pulse A enters. The merging block immediately generates Pulse B at req_{out}. Pulse B enters the top delay line and results Pulse C at req_{Tap1} after a T_{D} delay. Pulse C does two things: 1) It enters the merging block and generates Pulse D at req_{out} immediately. 2) It also enters the bottom delay line and generates Pulse E at req_{Tap2} after a T_{D} delay. When Pulse E is generated at req_{Tap2}, Pulse F is generated at req_{Tap1} at the same time. Pulse F is the outcome of Pulse D passing through the top delay line. Pulses E and F both enter the merging block and result in Pulse G. Every T_{D} after that, a pulse is generated at req_{out}, req_{Tap1} and req_{Tap2} at the same time.

From Fig. 8.2, it is clear that, except for Pulse C, all the pulses at req_{Tap1} coincide with pulses at req_{Tap2}. So, naturally, one asks if it is necessary to use two delay lines to generate the pulses at

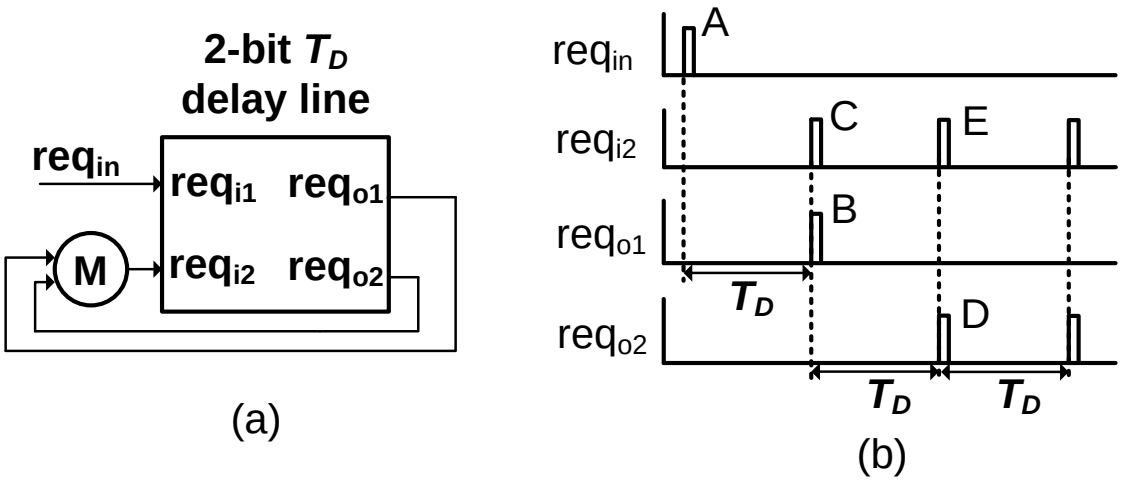


Figure 8.3: A two-bit T_{D} delay line. (a) Timing block. (b) Operations.

req_{Tap1} and req_{Tap2}. The answer is "No" if one can implement a two-bit delay line as shown in Fig. 8.3(a). The two-bit delay line has two inputs and two outputs. If a pulse enters through req_{i1}, the delay line generates a pulse at req_{o1} after a T_{D} delay. If a pulse enters through req_{i2}, the delay line generates a pulse at req_{o2} after a T_{D} delay. We configure the two-bit T_{D} delay line in a loop as shown in Fig. 8.3(a). Fig. 8.3(b) shows its operations after Pulse A enters the req_{i1} terminal. After a T_{D} delay, Pulse B is generated at req_{o1}. Pulse B goes into the merging block and immediately generates Pulse C at req_{i2}. After another T_{D} delay, the delay line generates Pulse D at req_{o2}. Pulse D also goes into the merging block and immediately generates Pulse E at req_{i2}. Every T_{D} after that, pulses are simultaneously generated at req_{i2} and req_{o2}. Comparing Fig. 8.2(b) and 8.3(b), one can see that merging the pulses at req_{in}, req_{o1} and req_{o2} in Fig. 8.3(b) generates a signal equivalent to req_{out} in Fig. 8.2(b), and that merging the pulses at req_{o1} and req_{o2} in Fig. 8.3(b) generates a signal equivalent to req_{Tap1} in 8.2(b). Finally, req_{o2} in Fig. 8.3(b) is equivalent to req_{Tap2} in Fig. 8.2(b). Later, we will show that the merging operation can be implemented very simply. It is clear now

that all the timing signals in Fig. 8.2(b) can be derived from the timing signals in Fig. 8.3(b), yet the structure in Fig. 8.3(b) uses one less delay line.

8.2.1 Implementation of two-bit T_{D} delay line

The functions of a two-bit T_{D} delay line are described above. Of course, one can implement the functions using two parallel one-bit T_{D} delay lines. This is, unfortunately, not a good way because the implementation suffers from the same matching issue and saves no hardware compared to the original design. The implementation we propose in this section promises to be free of the matching issue and to save half of the hardware compared to the original timing block in Fig. 8.2(a). Similar to a one-bit delay line, a two-bit delay line is composed of a cascade of two-bit t_{g} delay cells so that it can delay events with a granularity of t_{g} (Fig. 8.4(a)). Each two-bit t_{g} delay cell has two four-phase handshaking communication channels at its input and its output. Output Channel 1 (req_{o1}, ack_{o1}) of the $(k-1)^{\mathrm{th}}$ stage is connected to Input Channel 1 (req_{i1}, ack_{i1}) of the k^{th} stage. Output Channel 2 (req_{o2}, ack_{o2}) of the $(k-1)^{\mathrm{th}}$ stage is connected to Input Channel 2 (req_{i2}, ack_{i2}) of the k^{th} stage. A two-bit t_{g} delay cell has two functions: If an event arrives at its Input Channel 1, the delay cell generates an event at its Output Channel 1 after a t_{g} delay. If an event arrives at its Input Channel 2, the cell generates an event at its Output Channel 2 after a t_{g} delay.

The schematic of a two-bit t_{g} delay cell is shown in Fig. 8.4(b). Initially all the C-elements' outputs are zero. Q and Qb of the three SR latches are zero and high, respectively. When an event enters the delay cell from Input Channel 1, req_{i1} is pulled up. ack_{i1} is pulled down accordingly to acknowledge the event. In the meantime, this transition on ack_{i1} sets both SR0 and SR1. The delay

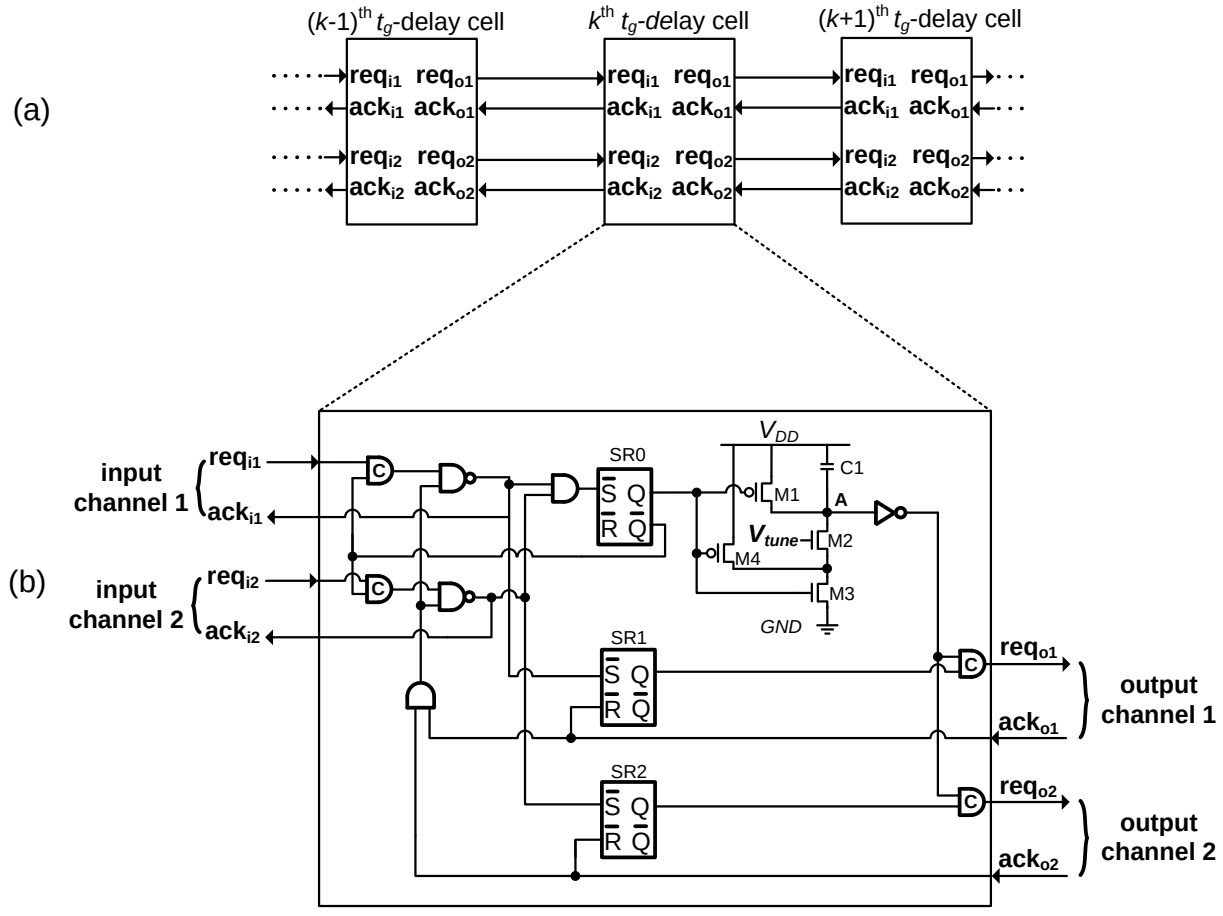


Figure 8.4: A two-bit delay line composed of several delay cells for fine time granularity. (a) Timing block. (b) schematic of one delay cell.

cell enters the delay phase. The high Q of SR1 represents that this ongoing delay operation is triggered from Channel 1. The mechanism of generating a delay relies on a current source charging a capacitor, which is exactly the same as in Fig. 4.5 (the split transistors in Fig. 4.5 are combined into one in Fig. 8.4(b) for simplicity). After a t_g delay, the output of the inverter is pulled high. Because Q of SR1 is high, req_{o1} is pulled high accordingly. An event is generated at Output Channel 1. Once the next delay cell enters into the delay phase, ack_{o1} is pulled down to reset both SR0 and SR1. A C-element is used to generate req_{o1}. It ensures that only when both SR1 and the delaying components (M1–M4 and C1) are completely reset will the req_{o1} fall to zero. An event entering the delay cell from Input Channel 2 triggers similar operations in the delay cell. The event sets SR0 and SR2 in this case. The high Q of SR2 represents that the ongoing delay phase is triggered from Channel 2. At the end of the delay operation, the event is passed into next delay cell through Output Channel 2. The delay cell uses two independent input/output channels to propagate events in the two channels separately. The key for this design is that the two channels share the same delaying resource (M1–M4 and C1 in Fig. 8.4(b)), so the delays of the two channels always match.

When the delay cell is in a delay phase, it cannot accept another incoming event from either of its two channels. This requires the input events of the delay cell on its two channels to be separated by at least t_g. This requirement can be fulfilled by using a grouping block at the very beginning of the two-bit delay line, as shown in Fig. 8.5(a). The grouping block should group all events that arrive at the inputs of the delay line within a t_g timing window into one event. As with a regular two-bit delay cell, the grouper also has two separate input and output channels. Their connections with the first two-bit t_g delay cell are also shown in Fig. 8.5(a). Normally, the two

channels in a two-bit delay line are symmetric. However, when it is used as a timing block for an IIR filter as shown in Fig. 8.3(a), Channel 2 of the two-bit delay line is configured as a closed loop. For an IIR filter, the loop delay of this closed loop determines the denominator of the filter's transfer function, which is critical to the system's stability. Hence, it is important to keep the loop delay event independent. This requires the grouping block to have the operations as plotted in Fig. 8.5(b)–(d).[1] When an event arrives at the grouper's req_{i1}, it triggers a timing window of length t_{g}, as shown in Fig. 8.5(b). Since no other events arrive during the window, an event is generated at the grouper's req_{o1} at the end of the window. A similar operation occurs if an event arrives at req_{i2} and no other events arrive during the window. An event is generated at req_{o2} after a t_{g} delay. Fig. 8.5(c) shows the case of an event arriving at req_{i1} in the middle of a window triggered by a previous event at req_{i2}. The grouper groups the two events and generates an output event at req_{o2} at the end of the window. Fig. 8.5(d) shows the case of an event arriving at req_{i2} in the middle of a window triggered by a previous event at req_{i1}. The grouper extends the grouping window by a new t_{g}. At the end of the extended window, an event is generated at req_{o2}. By extending the grouping window, the grouper guarantees that the delay from the arrival of an event at req_{i2} to the generation of the corresponding event at req_{o2} is always t_{g}.

Fig. 8.6 shows that the schematic of the grouping block is basically a simplified version of the grouping block in the CT digital IIR filter in Fig. 5.6. The "feedforward channel" and "R1 channel" in Fig. 5.6 are used as "Channel 1" and "Channel 2" in Fig. 8.6. Since the operations of the two channels have been described in detail in Section 5.2.6, we do not repeat them here.

[1] Only the operations associated with *req* are plotted to keep the plot simple.

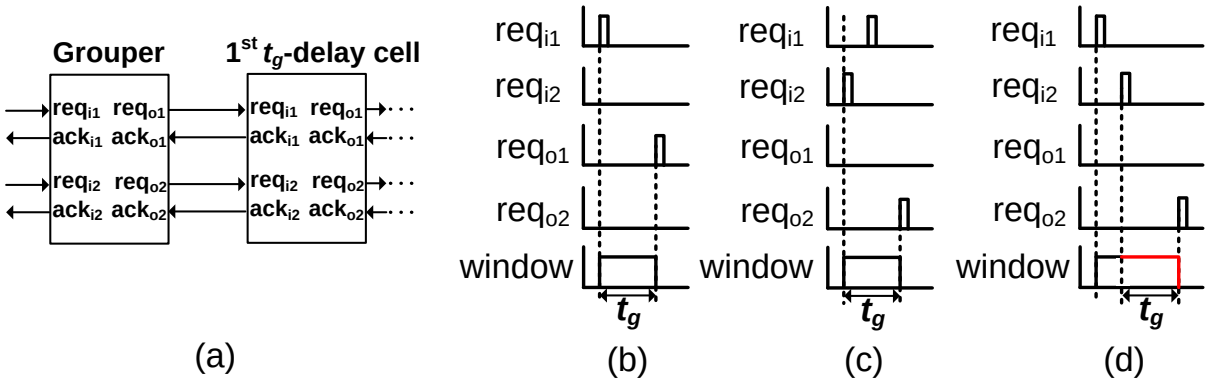


Figure 8.5: Grouping block in a two-bit delay line. (a) Its input/output interfaces and their connections with the first t_g delay cell. (b)–(d) Its operations in three different cases.

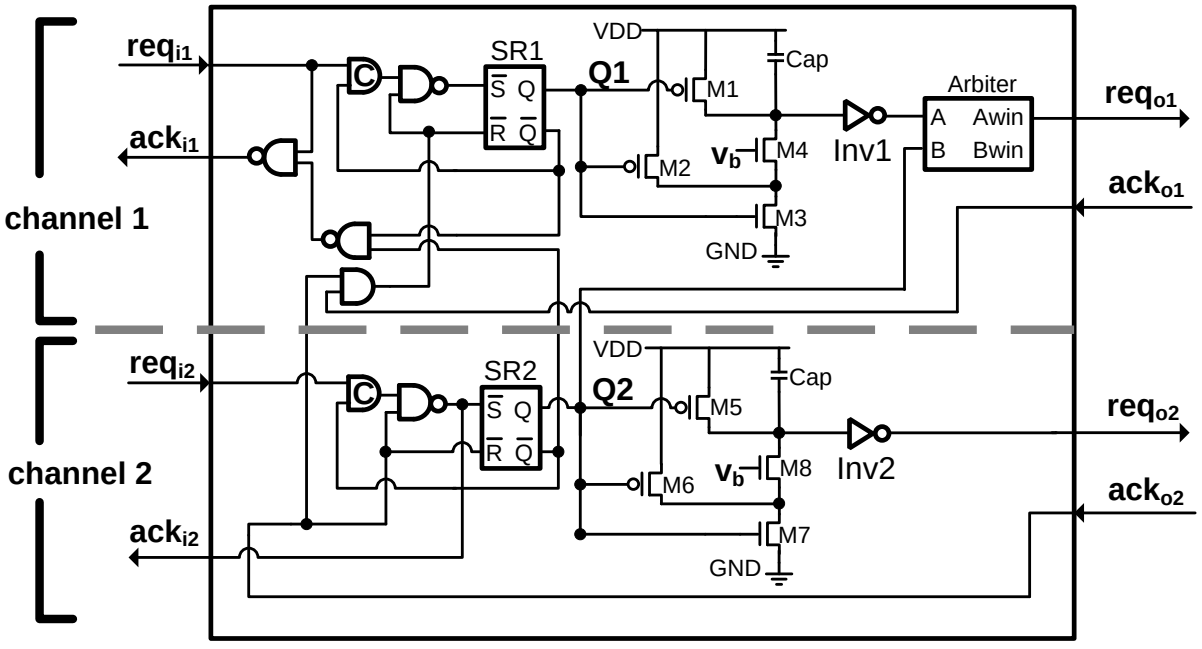


Figure 8.6: Schematic of the grouping block in Fig. 8.5(a).

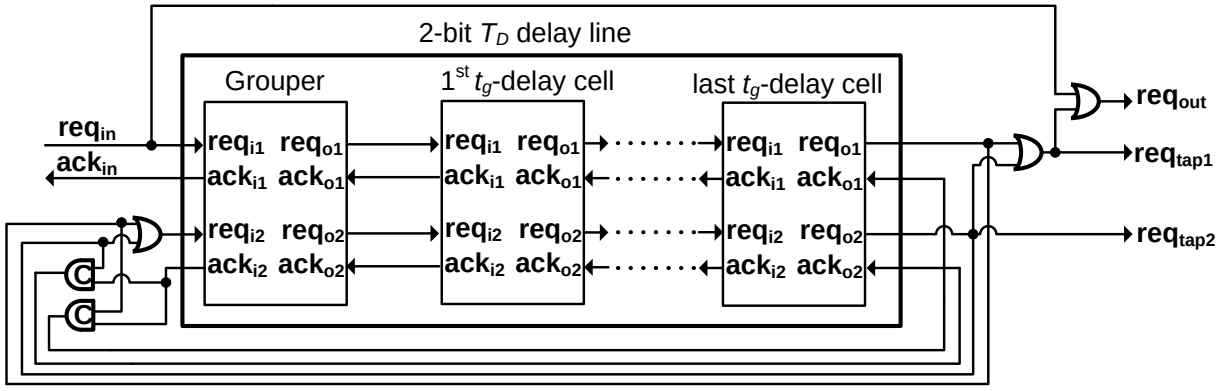


Figure 8.7: A two-bit T_{D} delay line is configured as the CT digital IIR filter's timing block.

8.2.2 A two-bit T_{D} delay line as the CT digital IIR filter's timing block

Fig. 8.7 shows the detailed connections of Fig. 8.3(a), which configures a two-bit T_{D} delay line as the timing block of a CT digital IIR filter. The merging block combining the two inputs at req_{i2} of the grouper can be simply implemented with an OR gate. Two C-elements are used to generate the corresponding acknowledge signals. As we described above, the signals that will be finally used to control the data path of a CT digital IIR filter – req_{out}, req_{Tap1} and req_{Tap2} in Fig. 8.2 – can simply be derived from the timing signals generated by the timing block in Fig. 8.7. They can be obtained with two OR gates as shown in the rightmost delay cell of Fig. 8.7.

8.3 A complete signal processing chain with a VR sigma–delta modulator and a VR-DSP

Chapter 7 introduces the VR-DSP. A complete signal-processing chain, including both a VR-ADC and a VR-DSP, is desired. As we discussed in Chapter 7, to implement a VR-DSP with a fre-

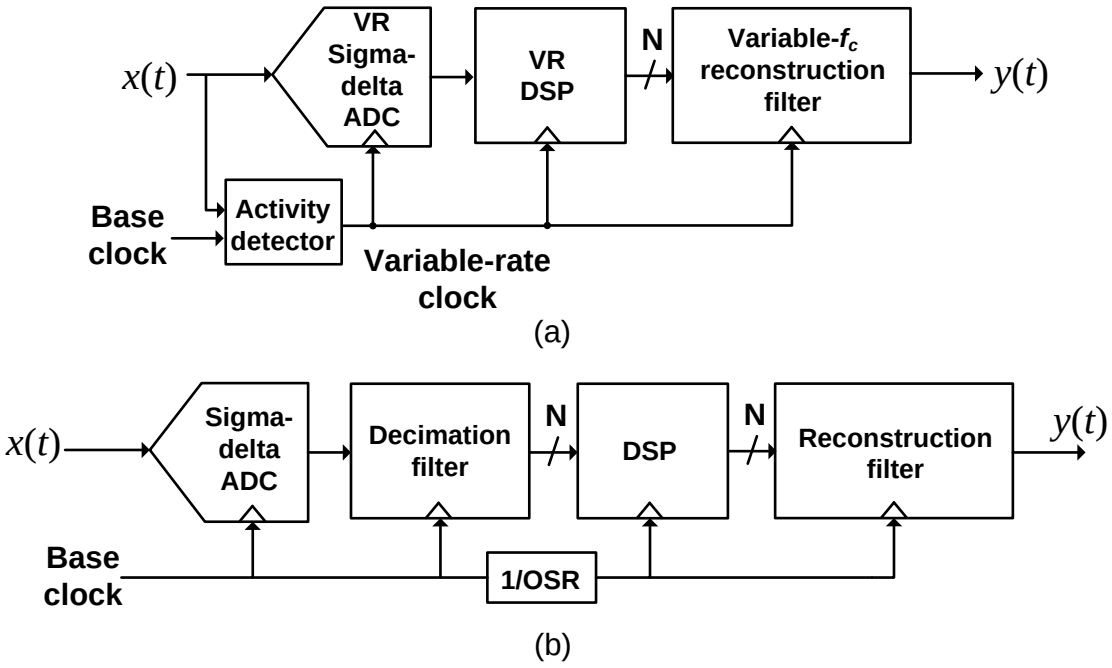


Figure 8.8: Two signal-processing chains compared. (a) Using a VR clock. (b) Using a fixed-rate clock.

quency response independent of the instantaneous sampling frequency, the preceding ADC must be oversampled. One naturally thinks of using a sigma–delta ADC. One possible implementation of a signal-processing chain composed of a VR sigma–delta ADC and a VR-DSP is shown in Fig. 8.8(a). According to the "local bandwidth" of the input signal, $x(t)$, the VR clock adaptively changes its instantaneous value. The VR-DSP, which operates at the same VR clock as the VR sigma–delta ADC, can directly process the samples generated by the ADC, without going through a decimation filter. At the end of the signal-processing chain, a reconstruction filter with a variable cutoff frequency tracking the instantaneous sampling rate ensures a fixed signal-to-noise ratio as we explained in Section 7.2.4.

As a comparison, we show a conventional signal-processing chain with fixed-rate sigma–delta

ADC and DSP in Fig. 8.8(b). Both the sigma–delta ADC and the decimation filter work at the highest clock rate in the system, the base clock rate. After the decimation filter, the sample rate decreases by a factor of OSR. Hence, both the DSP and the reconstruction filter work at a clock rate that is 1/OSR as fast. Assume the base clock rates in the two systems are the same. Also assume that the VR clock only changes between two clock rates: the base clock rate (when the input has high-frequency contents) and 1/OSR of the base clock rate (when the input only has low-frequency contents). Comparing the hardware and the average operation rates in the two systems in Fig. 8.8, we find that the fixed-rate scheme has a higher operation rate in the ADC and a lower operation rate in the DSP and the reconstruction filter than the average operation rate in the VR scheme. However, the extra decimation filter, which operates at the highest clock rate in the fixed-rate scheme, is not needed in the VR scheme. One advantage of the VR scheme, as we have shown in Chapter 7, is that the operation rate depends on the input activity. If the input signal has low-frequency contents most of the time, the average operation rate of the VR clock can be as low as 1/OSR of the base clock rate, which is close to the operation rate in the DSP and reconstruction filter in the fixed-rate scheme. As a result, the VR scheme has big advantages in terms of power and hardware consumptions benefiting from a much lower average operation rate in the ADC and the avoidance of a decimation filter.

Bibliography

[1] A. Mostafa *et al.*, "Adaptive sampling of speech signals," *IEEE Transactions on Communications,* vol. 22, pp. 1189–1194, Sep. 1974.

[2] Y. Tsividis, "Continuous-time digital signal processing," in *Electronics Letters,* vol. 39, no. 21, pp. 1551–1552, 16 Oct. 2003.

[3] B. Schell and Y. Tsividis, "A continuous-time ADC/DSP/DAC system with no clock and with activity-dependent power dissipation," *IEEE Journal of Solid-State Circuits,* vol. 43, no. 11, pp. 2472–2481, Nov. 2008.

[4] Leon W. Couch, *Digital and Analog Communication Systems.* Pearson; Jan. 19, 2012.

[5] M. Kurchuk and Y. Tsividis, "Signal-dependent variable-resolution quantization for continuous-time digital signal processing," *IEEE International Symposium on Circuits and Systems,* Taipei, 2009, pp. 1109–1112.

[6] C. Weltin-Wu and Y. Tsividis, "An event-driven clockless level-crossing ADC with signal-dependent adaptive resolution," in *IEEE Journal of Solid-State Circuits,* vol. 48, no. 9, pp. 2180–2190, Sept. 2013.

[7] P. Martnez-Nuevo, S. Patil and Y. Tsividis, "Derivative level-crossing sampling," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 62, no. 1, pp. 11–15, Jan. 2015.

[8] T. Kurp *et al.*, "An adaptive sampling scheme for improved energy utilization in wireless sensor networks," *2010 IEEE Instrumentation and Measurement Technology Conference (I2MTC),* pp. 93–98, May 2010.

[9] H. Kim, C. Van Hoof, and R. Yazicioglu, "A mixed signal ECG processing platform with an adaptive sampling ADC for portable monitoring applications," *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC,* pp. 2196–2199, Aug. 2011.

[10] S. M. Qaisar *et al.*, "Computationally efficient adaptive rate sampling and filtering," in *EUSIPCO,* vol. 7, pp. 2139–2143, 2007.

[11] W. Dieter *et al.*, "Power reduction by varying sampling rate," *Proceedings of the 2005 International Symposium on Low Power Electronics and Design,* pp. 227–232, Aug 2005.

[12] J. Mark and T. Todd, "A nonuniform sampling approach to data compression," *IEEE Transactions on Communications,* vol. 29, no. 1, pp. 24–32, Jan. 1981.

[13] J. Foster and T. K. Wang, "Speech coding using time code modulation," *IEEE Proceedings of Southeastcon '91,* Williamsburg, VA, 1991, vol. 2, pp. 861–863.

[14] N. Sayiner, H. V. Sorensen and T. R. Viswanathan, "A level-crossing sampling scheme for A/D conversion," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 43, no. 4, pp. 335–339, Apr. 1996.

[15] C. Vezyrtzis *et al.*, "A flexible, event-driven digital filter with frequency response independent of input sample rate," *IEEE Journal of Solid-State Circuits,* vol. 49, pp. 2292–2304, Oct. 2014.

[16] Bob Schell and Yannis Tsividis. "Analysis and simulation of continuous-time digital signal processors." *Signal Processing* vol. 89, no. 10, pp. 2013–2026, 2009.

[17] M. Kurchuk, C. Weltin-Wu, D. Morche and Y. Tsividis, "Event-driven GHz-range continuous-time digital signal processor with activity-dependent power dissipation," *IEEE Journal of Solid-State Circuits,* vol. 47, no. 9, pp. 2164–2173, Sept. 2012.

[18] B. Schell, *Continuous-time digital signal processors: Analysis and implementation.* Ph.D. thesis, Department of Electrical Engineering, Columbia University, 2008.

[19] Y. Chen, M. Kurchuk, N. Thao, and Y. Tsividis, "Spectral analysis of continuous-time ADC and DSP," in *Event-Based Control and Signal Processing,* M. Miskowicz (Ed.), CRC Press, 2015.

[20] Y. Tsividis, M. Kurchuk, S. Nowick, B. Schell, and C. Vezyrtzis, "Event-based data acquisition and digital signal processing in continuous time," in *Event-Based Control and Signal Processing,* M. Miskowicz (Ed.), CRC Press, 2015.

[21] A. Gersho and R. M. Gray, *Vector quantization and signal compression.* Springer, 1992.

[22] S. Haykin, *Communication Systems.* Wiley, 2001.

[23] H. E. Rowe, *Signals and Noise in Communication Systems.* Van Nostrand, 1965.

[24] N. M. Blachman, "The intermodulation and distortion due to quantization of sinusoids," *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 33, pp. 1417–1426, Dec 1985.

[25] H. Pan and A. Abidi, "Spectral spurs due to quantization in Nyquist ADCs," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 51, pp. 1422–1439, Aug 2004.

[26] T. Claasen and A. Jongepier, "Model for the power spectral density of quantization noise," *IEEE Transactions on Acoustics, Speech and Signal Processing,* vol. 29, no. 4, pp. 914–917, 1981.

[27] M. Kurchuk, *Signal encoding and digital signal processing in continuous time.* PhD thesis, Columbia University, 2011.

[28] P. Z. Peebles, *Communication System Principles.* Addison-Wesley, Advanced Book Program, 1976.

[29] V. Balasubramanian, A. Heragu and C. Enz, "Analysis of ultralow-power asynchronous ADCs," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 3593–3596, May 2010.

[30] W. R. Bennett, "Spectra of quantized signals," *Bell System Technical Journal,* vol. 27, no. 3, pp. 446–472, 1948.

[31] N. S. Jayant and P. Noll, *Digital coding of waveforms: Principles and applications to speech and video,* ACM, 1984.

[32] D. Hand and M. S. Chen, "A non-uniform sampling ADC architecture with embedded alias-free asynchronous filter," *Proceedings of the Global Communications Conference 2012,* pp. 3707–3712.

[33] Y. Tsividis, "Digital signal processing in continuous time: A possibility for avoiding aliasing and reducing quantization error," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 2, pp. ii–589–92, May 2004.

[34] J. Foster and T.-K. Wang, "Speech coding using time code modulation," in *IEEE Proceedings of Southeastcon '91,* pp. 861–863, 1991.

[35] J. Sparso and S. Furber, *Principles of Asynchronous Circuit Design.* Kluwer Academic, 2002.

[36] Sanjit Kumar Mitra, *Digital signal processing: A computer-based approach.* McGraw-Hill/Irwin, 2001.

[37] Y. Neuvo, Dong Cheng-Yu and S. Mitra, "Interpolated finite impulse response filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. 32, no. 3, pp. 563–570, Jun. 1984.

[38] B. S. Chang, G. Kim and W. Kim, "A low voltage low power CMOS delay element," *Twenty-first European Solid-State Circuits Conference,* Lille, France, 1995, pp. 222–225.

[39] B. Schell and Y. Tsividis, "A low power tunable delay element suitable for asynchronous delays of burst information," *IEEE Journal of Solid-State Circuits,* vol. 43, no. 5, pp. 1227–1234, May 2008.

[40] C. Vezyrtzis, *Continuous-Time and Companding Digital Signal Processors Using Adaptivity and Asynchronous Techniques.* PhD thesis, Columbia University, 2011.

[41] To be published.

[42] E. Burlingame and R. Spencer, "An analog CMOS high-speed continuous-time FIR filter," *Proceedings of the 26th European Solid-State Circuits Conference,* Stockholm, Sweden, 2000, pp. 288–291.

[43] David E. Muller, "Asynchronous logic and application to information processing." *Proceedings of the Symposium on the Application of Switching Theory in Space Technology,* 1963, 289–297.

[44] Y. Tsividis and C. McAndrew, *Operation and Modeling of the MOS Transistor.* Oxford University Press, 2011.

[45] M. Kurchuk and Y. Tsividis, "Energy-efficient asynchronous delay element with wide controllability," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems,* Paris, 2010, pp. 3837–3840.

[46] K. van Berkel, *Handshake Circuits: An Asynchronous Architecture for VLSI Programming.* Cambridge University Press, 1993

[47] C. L. Seitz, “System timing,” *Introduction to VLSI systems,* 1980, pp. 218–262.

[48] D. Kim, G. Chen, M. Fojtik, M. Seok, D. Blaauw and D. Sylvester, “A 1.85 fW/bit ultra low leakage 10T SRAM with speed compensation scheme,” *2011 IEEE International Symposium of Circuits and Systems (ISCAS),* Rio de Janeiro, 2011, pp. 69–72.

[49] R. Ginosar, “Metastability and synchronizers: A tutorial,” *IEEE Design and Test of Computers,* vol. 28, no. 5, pp. 23–35, Sept.–Oct. 2011.

[50] F. J. Harris, “On the use of windows for harmonic analysis with the discrete Fourier transform,” *Proceedings of the IEEE,* vol. 66, no. 1, pp. 51–83, Jan. 1978.

[51] A. Agarwal *et al.*, 11 A 320 mV-to-1.2 V on-die fine-grained reconfigurable fabric for DSP/media accelerators in 32 nm CMOS,” *ISSCC Digest,* pp. 328–329, 2010.

[52] E. O’hAnnaidh, E. Rouat, S. Verhaeren, S. L. Tual and C. Garnier, “A 3.2GHz-sample-rate 800mHz bandwidth highly reconfigurable analog FIR filter in 45nm CMOS,” *ISSCC Digest*, pp. 90-91, 2010.

[53] M. Tohidian, I. Madadi and R. B. Staszewski, “A 2mW 800MS/s 7th-order discrete-time IIR filter with 400kHz-to-30MHz BW and 100dB stop-band rejection in 65nm CMOS,” *ISSCC Digest*, pp. 174-175, 2013.

[54] L. R. Rabiner, J. F. Kaiser, O. Herrmann and M. T. Dolan, ”Some comparisons between fir and iir digital filters,” in *The Bell System Technical Journal*, vol. 53, no. 2, pp. 305-331, Feb. 1974.

[55] Y. Chen and Y. Tsividis, “Design considerations for variable-rate digital signal processing,” *2016 IEEE International Symposium on Circuits and Systems (ISCAS),* Montreal, QC, 2016, pp. 2479–2482.

[56] A. V. Oppenheim, R. W. Schafer and J. R. Buck, *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series, 1999.

Appendix

Appendix A

Stability Analysis of CT Digital IIR Filter in the Laplace Domain

With conventional systems which only involve transfer functions that are rational in s, one can determine their stability from the pole locations. As long as there is at least one pole located in the right half plane of the Laplace domain, the corresponding systems are unstable. In fact, this relation of a systems stability to its pole location is also valid when the system involves transfer functions which contain exponential functions in s. This is because regardless of the types of functions in s, a pole in the right half plane always corresponds to an unbounded impulse response. Hence pole locations will be our primary tool to determine the stability of a CT digital filter.

In Section 3.2, we have shown that an originally stable IIR filter becomes unstable once delay line mismatches are present. In this appendix, we will mathematically prove this result. We reuse

the example in Section 3.2, as shown in Fig. A.1(a). The transfer function is:

$$H(s) = \frac{e^{2sT_D}}{e^{2sT_D} + 1.5e^{sT_D} + 0.6}. \tag{A.1}$$

In order to judge the stability of the above system, we need to know the locations of its s-domain poles. Although the system has an infinite number of poles, they correspond to a finite number of e^{sT_D} values which make the denominator of Eqn A.1 zero. From the modulus of these e^{sT_D} values, we can deduce the locations of the s-domain poles. If a system is unstable, it must have at least one s-domain pole located in the right half plane of the Laplace domain. Meanwhile, e^{sT_D} can be expanded to be $e^{\alpha T_D} * e^{j\beta T_D}$. Because the s-domain pole in the right half plane contains a positive real part and T_D is a physical delay which must be positive, $\alpha T_D > 0$ and hence $|e^{\alpha T_D}| > 1$. Thus, the corresponding e^{sT_D} value which makes the denominator of the system's transfer function zero must have a modulus larger than 1.

The only two values of e^{sT_D} making the denominator zero are $e^{sT_D} = -0.75 \pm j0.19$. All the values of $s = \alpha + j\beta$ which satisfies the equations are the poles of Eqn A.1. $e^{\alpha T_D}$ represents the magnitudes of these poles. Since $|-0.75 \pm j0.19| = 0.77 < 1$, α must be negative to let the poles' modulus smaller than 1. Hence, all the s-domain poles of Eqn A.1 must have negative real parts and hence are located in the left half plane in the Laplace domain. Such a system is stable.

When delay lines are not identical, as we show in Fig. A.1(b) where the top delay line is T_D and the bottom delay line becomes $T_D + \tau$, the frequency response of the IIR filter changes to

$$H(s) = \frac{e^{2sT_D+\tau}}{e^{2sT_D+\tau} + 1.5e^{sT_D+\tau} + 0.6}. \tag{A.2}$$

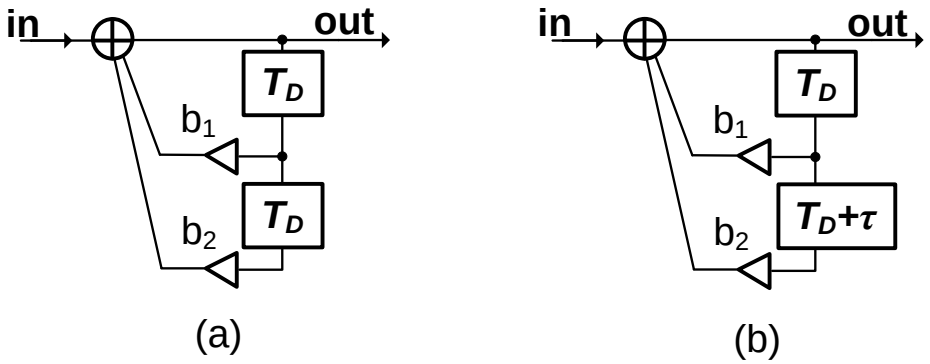


Figure A.1: Second-order IIR filters with $b_1 = -1.5$, $b_2 = -0.6$. (a) Identical delay lines. (b) Nonidentical delay lines.

As an example, we assume $\tau = 0.1T_D$. The transfer function becomes

$$H(s) = \frac{(e^{\frac{sT_D}{10}})^{21}}{(e^{\frac{sT_D}{10}})^{21} + 1.5(e^{\frac{sT_D}{10}})^{11} + 0.6}. \tag{A.3}$$

It is straightforward to find all values of $e^{\frac{sT_D}{10}}$ which makes the denominator of Eqn A.3 zero. Within all such values of $e^{\frac{sT_D}{10}}$, the one with the maximum modulus is $e^{\frac{sT_D}{10}} = -1.01 + j0.32$, whose modulus is $|e^{\frac{sT_D}{10}}| = 1.06$. Its corresponding s-domain poles must have positive real parts. Since the IIR system has at least one s-domain pole located in the right half plane, the system is unstable.

To better understand the effect of delay mismatch on system stability, we sweep τ from $-0.5T_D$ to $0.5T_D$, with a step of $0.01T_D$. The stability of the filter in each case can be determined using the above method. We find the maximum modulus of the values of $e^{\frac{sT_D}{100}}$ which make the denominator of the system transfer equal to zero. If it is larger than one, at least one s-domain pole of the system is located in the right half plane and hence the system is unstable. We plot the maximum modulus of the values of $e^{\frac{sT_D}{100}}$ which make the denominator of system transfer equal to zero versus τ in Fig.

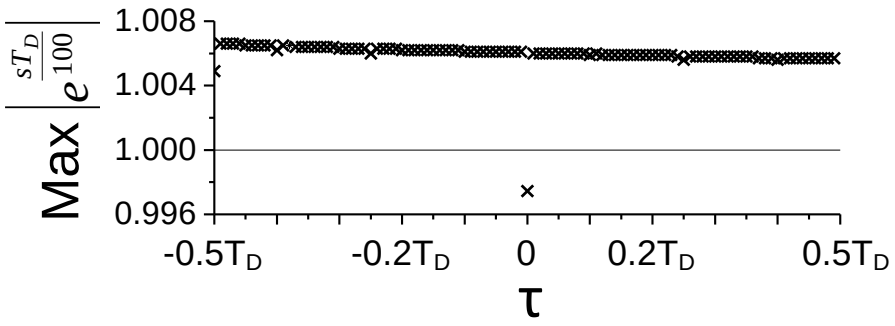


Figure A.2: Max modulus of $e^{\frac{sT_D}{100}}$ versus delay mismatch.

A.2. It is clear that the stability of the system does not change continuously with delay mismatch. The system is stable only when delay lines are perfectly matched, i.e. $\tau = 0$.