Spread Spectrum Modulation System For Burst Mode DC-DC Converters

by

Ji Zhang

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 12, 2006

Author_____
Department of Electrical Engineering and Computer Science
May 12, 2006

Certified by_____
David A. Walch
Design Engineer
VI-A Company Thesis Supervisor

Certified by_____
David J. Perreault
Associate Professor
M.I.T. Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Spread Spectrum Modulation System For Burst Mode DC-DC Converters

by

Ji Zhang

Submitted to the
Department of Electrical Engineering and Computer Science

May 12, 2006

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis develops a spread spectrum switching system for DC-DC converters operating in burst mode. Burst mode DC-DC converters have high efficiency under low-power conditions in applications such as cell phones and notebook computers, but often produce noise in the audible range. This thesis explores a frequency modulation scheme and transient control that attenuates audible noise harmonics while minimizing the tradeoff for converter regulation, efficiency, and output voltage ripple.

M.I.T. Thesis Advisor: David J. Perreault
Title: Associate Professor

VI-A Company Thesis Supervisor: David Walch
Title: Design Engineer

Acknowledgements

This thesis is the achievement of a collective effort. I would like to thank my mentors, colleagues, and friends, who stood with me through triumphs and challenges. This project started when John Bazinet shared his vision, and David Salerno provided the idea behind this thesis. Professor David Perreault steered my research and refined my analysis with his breadth of knowledge. David Walch guided my technical direction with terrific supervision. Eddy Wells volunteered his robust design as a test subject so that proof of this thesis was possible. James McDonald spent too many hours in laboratory finding optimizations and helping me debug. John Canfield donated a database of related subject experience, and Nitzan Gadish donated her knowledge of and notes on circuits. There was never a day when Rich Cook did not suggest a design idea. Mark Jordan, David Loconto, David Olsen, John King, Bill Martin, John Ziegler, George Humphrey, Mark Belch, Bert Hepfinger, and Brian Harnedy imparted their precious analog expertise. Joe Demers and Hellmuth Witte deserve the credit for building the testing core. Fred Viana always knew where to look for technical assistance.

My parents sacrificed too much to give me the opportunity to be at M.I.T. Adam Kumpf's passion for innovation and Michael Glick's ingenuity are what motivated me to pursue this project from start and execute until end.

Without the inspiration from any of these people, this thesis would not have been possible.

Contents

List Of Figures

List Of Tables

Chapter I: Introduction

Overview

Power management has become a center of attention in the Analog IC community both in terms of breakthroughs as well as profits. Modern portable devices such as laptops, cellular phones, and PDAs have challenging power requirements but only have a handful of batteries from which they can source power. Analog solutions bridge the gap between complex power needs and simple power supplies.

DC-DC converters are especially important since commercial power sources, such as batteries, are limited to a range of discrete voltages. The list of available voltages becomes increasingly limited as size and physical property requirements come into play. Minimizing noise (output voltage and current ripple) and improving efficiency (input power vs. output power) in DC-DC converters become hotly contested areas as portable device and power sources scale down in size, yet power demands increase with product complexity.

This thesis encompasses (1) development of a spread spectrum system to attenuate audible noise for burst mode DC-DC converters, (2) study on the efficacy of spread spectrum techniques to reduce audible noise, and (3) documentation of the effects of such techniques on converter efficiency and output ripple.

For practicality purposes, this project is implemented with discrete components, but the goal of this thesis is to develop a systematic technique and analyses that may be incorporated into future converters or be introduced in standalone circuits. The frequency modulation system is robust enough such that it may be augmented to existing DC-DC converters for testing. LTC3458 is a high efficiency buck-boost converter that uses programmable burst mode operation and is a suitable test candidate.

Chapter II: Background

Basic Switching Power Converter Topology

Switching-mode DC-DC converters are generally designed for high efficiency. The basic architecture of switching-mode DC-DC converters consists of energy storage devices and some sort of a controllable switch (typically a diode, thyristor, BJT, or MOSFET) that channels the power from the input source (see Figure 2.1). Buck converters step down the input voltage, and boost converters step up the input voltage.

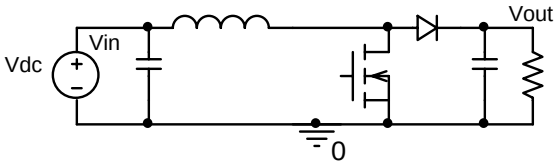


Figure 2.1: Basic circuit for a DC-DC boost converter.

A common method to control the switch is pulsewidth modulation (PWM), in which the output voltage is scaled by varying the duty cycle (the fraction of on-time) of the switch. Through rectification and low-pass filtering, the pulse train output of the converter is transformed into an essentially DC value.

The switch controller can vary in complexity depending on the application. If the input voltage is fixed and known, the controller can simply be an oscillator with fixed frequency and fixed duty cycle. However, a feedback control loop is needed for a robust design.

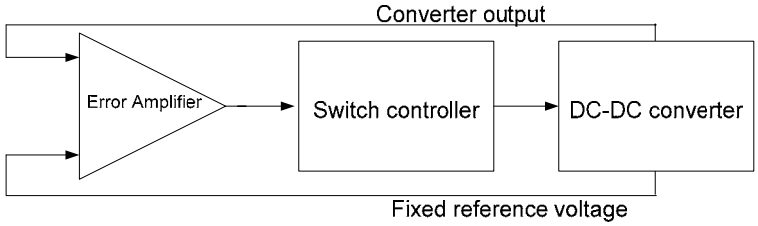


Figure 2.2: Basic control loop for a DC-DC converter.

In Figure 2.2, the output of the power converter is compared to a fixed voltage through an error amplifier. Typically a downscaled version of the converter output is actually fed into the error amplifier since the reference voltage is lower than the actual output of the converter. The switch controller, after sensing whether the output of the converter is too high or too low, can increase or decrease the duty cycle of the switch and stabilize the converter output.

One of the methods of improving power converter efficiency at light loads is to employ burst mode (also known as sleep mode and standby mode), in which the converter almost completely shuts down and only sends full throttles of current when necessary to maintain a specified output voltage window. Examples of DC-DC converters that employ burst-mode to improve light-load efficiency include the LTR2500, ZXCP330E6, and LM2770. This is particular useful since the quiescent current of the DC-DC converter is greatly reduced.

Switching Noise

Assuming that a power converter is operating in periodic steady state, the power converter's periodic switching frequency causes voltage and current ripple at harmonic frequencies. Noise with frequencies spanning 20Hz to 20kHz lies within the human auditory range. The primary goal of this project is to attenuate switching audible noise.

Specifically, the audible noise in a switching DC-DC converter originates from the vibrating inductor. Figure 2.3 shows that by the right hand curl rule, the magnetic field in a solenoid inductor is straight through the inductor while current flows circularly through the inductor coil. By the right hand rule, the magnetic force, which is orthogonal to both the current and the magnetic field, points directly in and out (perpendicular to the coil-walls) of the inductor. An AC current component thus causes an alternating force that causes the inductor to expand and contract.

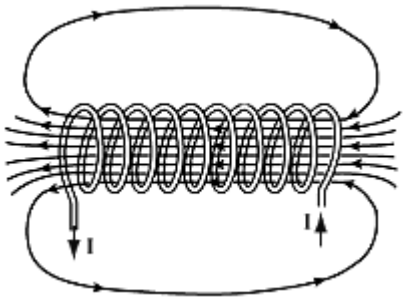


Figure 2.3: Magnetic field around a solenoid.

A rough estimate for the human threshold of hearing is 0 dB sound pressure level (dBspl), which is approximately equal to 20uPa [1]. The loudness of sound relates positively with audio power. In this case the components of electrical power generate the audio power. The loudness of the inductor noise is proportional to the voltage ripple, which is proportional to the current ripple that generates the magnetic force. Although this relationship is not linear, human ear sensitivity is not uniform across different frequencies, and acoustic sensitivity depends on the medium of transmission, for convenience this project uses a measure of output voltage ripple, dbV, to describe the relative strength of audible noise. Zero dBspl of sound does not mean that there is no noise; it merely indicates that under average conditions this is the typical threshold of hearing. In particular applications, such as various cell phones, even small noises in the audio spectrum can be amplified in audible noise.

Most modern PWM power converters are specifically designed with switching frequencies above 20kHz in order to avoid noise in the audio spectrum [2]. Converters operating in burst mode, however, have switching periods that are dependent on the load. Burst mode switching periods often drop into the audio spectrum and result in piercing tones.

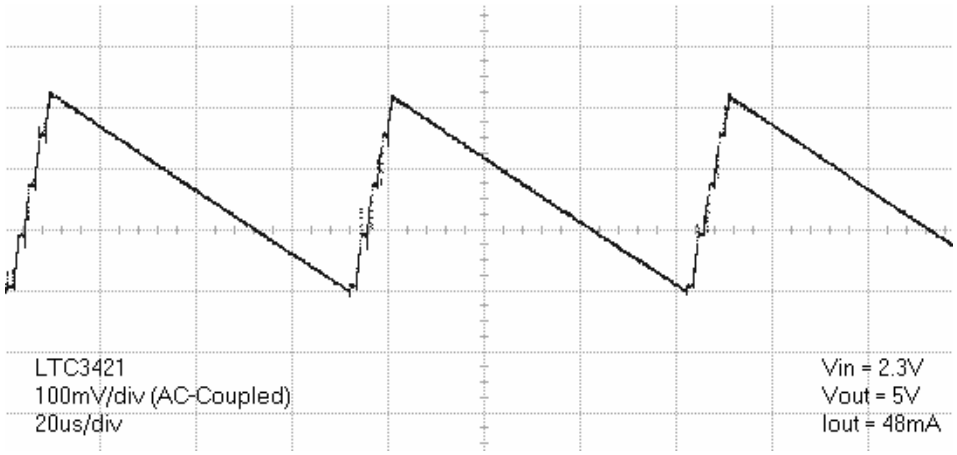


Figure 2.4: Typical output voltage waveform of a burst-mode DC-DC converter.

Figure 2.4 shows the regulated output voltage Linear Technology's LTC3421 DC-DC converter operating in burst mode. Within every switching period, rising segments correspond with the on-phase of the power converter, and falling segments represent the sleeping phase of the converter. Even though the output waveform is not a perfect sinusoid, the high-pitched audible tone that the converter generates sounds very clear and noticeable from 1 foot away.

Since Burst Mode frequency varies depending on load, the problematic noise does not fall under distinctive patterns. Adding passive band-limit filters to eliminate audible noise not only can add an enormous amount of layout space but also may not be effective since there is no preset frequency band to target.

Chapter III: Previous Solutions

Programmable burst mode

The simplest solution to reducing audible switching noise in burst mode is simply to exit burst mode when appropriate. The burst mode disable function is usually implemented with an on-chip input that the application must accommodate. DC-DC converter ICs LTC3421, LM2650, and MAX1677, for example, have the option to completely avoid burst mode and be forced into traditional continuous or synchronized switching mode. The tradeoff between Burst noise and quiescent current becomes an issue for application designers. Typically, a microcontroller selects continuous or synchronized mode when a device is in operation and selects Burst mode when the device is in standby [3]. Unfortunately this workaround does not directly address the audible noise problem and also requires a devoted process on a continuously monitoring microcontroller.

Pseudorandom modulation schemes

Since the drastic improvement of microcontrollers in the 1980s, substantial research has been published in synthesis of different spread spectrum modulation approaches. The randomized modulation concept originated in 1970 in [4] as an effort to soften harmonic spikes since older DC-DC converters' switching speeds did not push PWM switching frequencies past the audio spectrum. Described in [5], acoustic applications for pseudorandom modulation were originally aimed at high power motors. For this document the terms random modulation and spread spectrum (SS) serve the same purpose and are used interchangeably.

In [6] a microcontroller with a pseudorandom bit sequence injects noise into a DC-DC converter's feedback error comparator and successfully attenuates harmonic spikes. In [7] a follow-up frequency spectrum analysis of the same setup supports the previous method of randomizing switching frequencies.

Very similar microcontroller lookup table randomization schemes are also applied to PWM motor drivers. In [8] a PWM motor driver's harmonic spike at 500Hz on

the oscilloscope is completely attenuated after a random number generator is attached to the PWM generator to spread the switching bandwidth.

In [9] a state-based switching randomization algorithm based on Markov chains imposes randomization properties based on time domain properties. For example, for a power converter, it can be programmed that there are high probabilities that long duty cycles follow short duty cycles. In this example, the average voltage ripple is reduced since there is less probability of consecutive extra-long off-time for the switch. The Markov chain random modulation approach gives the capability to actively condition the time-domain properties of the switching waveform in addition to shaping the frequency spectrum.

The paper that classifies and analyzes different modulation algorithms is [10]. Most randomized modulation schemes, including the above methods, are categorized as stationary modulation-- meaning that the underlying reference switching pattern being dithered does not change. It is also assumed that the manner through which the randomization is generated does not change between switching cycles. While most published works on randomized modulation use empirical evidence, [10] presents some analysis methods that may be useful. A key conclusion from [10] is that randomized modulation techniques can be effective to eliminate spikes for narrowband applications but are not so practical for wideband needs. For this project, the audible noise does lie in narrowband harmonics, but the exact harmonic frequency changes.

No published work is found regarding the application of spread spectrum techniques to attenuate audible noise on converters in burst mode. However, some modern DC-DC converters such as the LTC3251 and the MAX1703 contain an on-chip spread spectrum system (for PWM mode) to eliminate high frequency (above 1 MHz) electromagnetic interference (EMI) that can interfere with RF circuitry.

Chapter IV: Overview of New Switching Modulation System

In designing a DC-DC converter switching frequency modulation system, there are 6 foreseeable challenges. Each of the following will be addressed in the upcoming sections.

(1) The available supply voltage may either be too low or have too much ripple to power a DSP chip or a microcontroller. Furthermore, the converter may be designed to power a DSP or microcontroller.

(2) Mentioned in [10], randomized switching modulation can lead to substantial increases in converter output ripple.

(3) Also mentioned in [10], spread spectrum techniques are most practical for converters with deterministic switching since randomized modulation do not seem to be effective at eliminating harmonics across a wide spectrum. Unfortunately, switching frequency in burst mode varies since it is load dependent.

(4) There is no substantive documentation on the efficacy of spread spectrum techniques for burst mode converters.

(5) There is no published study of the effects of adding randomization on the efficiency of low power burst mode converters.

(6) Converters typically use burst mode for low power applications, and the addition of a current-draining randomization system makes the converter less desirable.

To accommodate (1) potentially low or rippling supply voltages, the modulation system may have to be implemented as either an analog system or mixed signal system with basic digital components such as shift registers. There are three basic components to this system, as shown in Figure 4.1:

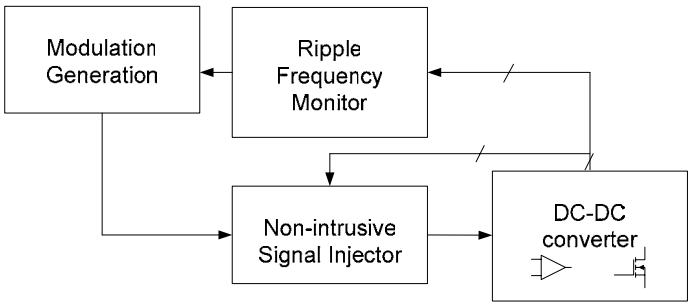


Figure 4.1. Block diagram of frequency modulation system.

The entire modulation system, from an ideal signal processing perspective, is modeled in Matlab and described in chapter V. The goal of modeling in Matlab is to efficiently discover and analyze different ways of reshaping the converter burst mode output. Chapter VI describes microcontroller implementations of the feasible Matlab models. Although programmed spread spectrum control may be inefficient, it does provide insight into some potential challenges in implementation as well as gauge the accuracy of the Matlab simulations. Chapters VII, VIII, and IX describe in detail the analog circuit implementation of the three components in the spread spectrum system.

Signal Injector

The signal injector component, chapter VII, is a time-domain control system as well as the interface between the rest of the modulation system and the original DC-DC converter. The injected waveform is superimposed on the output voltage feedback signal at the input of the error amplifier. It is non-intrusive in that latching the signal injector onto a converter does not change the fundamental properties of the converter (burst mode threshold, output shape, output DC voltage). Figure 4.2 shows a block diagram of the injector implementation aimed at preventing the randomized modulation from severely worsening the output voltage ripple. An on/off block shuts off the injected signal if the injected signal is causing excess ripple at the wrong time. For example, a random interference signal that will force the converter to release a burst of current should not be injected while the converter is already bursting. The randomization signal is also level shifted and scaled to appropriate proportions such that if it does pass through the

on/off gate and interfere with the converter's burst control, it does it mildly and does not change general output waveform shape of the converter.

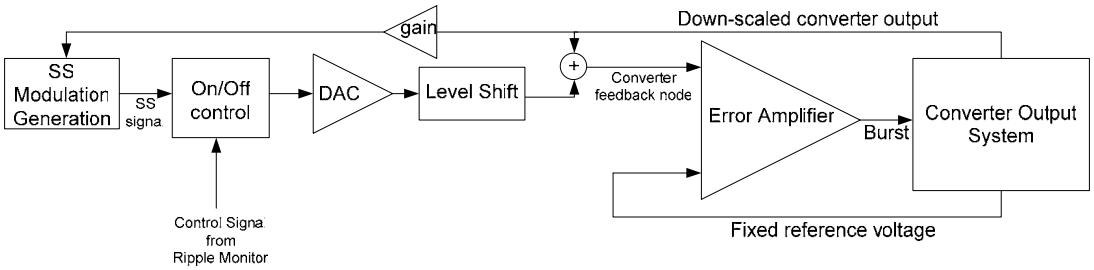


Figure 4.2: Block diagram of non-intrusive signal injector.

The easiest way to inject an interference signal into an existing DC-DC converter with discrete components is by altering the feedback error-compensation network of the DC-DC converter. This, however, is by no means non-intrusive in certain application-specific converter configurations. To preserve the properties of the original converter, required work may involve bypassing feed-forward capacitors and readjusting the output DC voltage calibration of the converter in addition to having the on/off, level shift, gain, and control blocks in Figure 4.2.

Ripple Monitor

The ripple frequency monitor, chapter VIII, in essence, is a frequency to voltage converter that accommodates external programming (via circuitry) for ease of use. At the same time, the ripple monitor controls the clock for the modulation generator component since it dictates the target output frequency band. It is assumed that:

(a) Different bands of pink and intensities of white noise are used to treat different burst mode harmonics.

(b) The human ear can have vastly different sensitivities to different audio harmonics [10].

It is more efficient to use shorter bands of pink noise to shape the burst mode frequency spectrum simply because implementation of the modulation generation

component is simpler. The ripple monitor chooses an optimal band and intensity of noise to accommodate (3) the load-dependent harmonics in burst mode.

The exact way that the ripple monitor system reshapes the converter output frequency spectrum depends on the application need. In applications where audio noise is amplified and interfering with other subsystems, the goal may be to purely suppress one harmonic. Matlab models and Monte-Carlo simulations (in chapter V) of burst-mode control with injected band-limited noise reveal patterns of desirable noise vs. burst frequency.

In instances where the audible noise from the converter itself is the problem, it sometimes is advantageous to spread the output frequency spectrum in directions where the human ear is less sensitive. Figure 4.3 is the standard Robinson-Dadson graph illustrating the vastly varying sensitivity of the human ear to sounds of different frequencies. In Figure 4.3, each contour represents the power of sound that is perceived to be equally loud at different frequencies. It is clear that, among study subjects, high frequency and low frequency sounds are on average more difficult to hear than sounds in the 1-5kHz range.

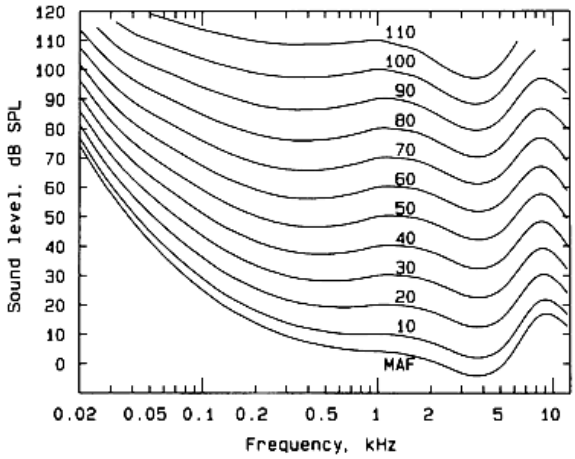


Figure 4.3: The Robinson-Dadson equal loudness curves.
MAF: Binaural minimum audible field (threshold of hearing).

The Robinson-Dadson curves suggest that spreading a sound harmonic's amplitude may not have predictable effects on the human perceived loudness, pitch, and quality of the sound. The actual effects of spread spectrum on perceived sound will obtained experimentally. For applications where the DC-DC converter's pure

audible noise is the main focus, a control circuit to prevent reshaping of the frequency spectrum in the wrong directions can be of help.

In measuring the (4) efficacy of randomized modulation in both simulation and experimentally, measuring the output voltage ripple and its Fourier transform is the most quantitatively accurate feasible measure. If actual audible noise is to be measured in Pascals (mechanical pressure), a silent environment, sound sensors that are equally sensitive across the audio frequency spectrum, and accountability for the directional and distance gain differences are required. Of course, both quantitative and qualitative observations of spread spectrum noise are needed for a comprehensive analysis of the effects of spread spectrum (chapter IX). Consideration must be given, however, to the fact that human feel for audible noise attenuation varies since auditory sensitivity varies from person to person.

Modulation Generation

The (5) effects of randomization on burst mode efficiency depends on the degree to which the converter's output ripple worsens. If the dominant frequency-dependent loss mechanism may be approximated as proportional to the burst frequency, and if the output voltage and the mean switching frequency remain the same, the total amount of energy dissipated in the basic converter components does not change. This assumes that the variation in frequency does not cause inductor saturation or significant magnetic core loss. Change, if any, in the efficiency of the converter is measured by keeping the converter's output current constant (with a current source) and reading the input supply current.

In the work here, losses due to the frequency modulation system are not included in the efficiency calculations since the entire system is built with discrete components while the converter itself is an integrated circuit. To address (6) the low power challenges, the entire frequency modulation system, especially the modulation generation, must be built with careful consideration for quiescent power consumption.

Since there are power supply limitations, using preprogrammed lookup tables on an external microcontroller to generate pseudorandom noise is not a feasible final

solution. Instead, the two methods of creating pseudorandom noise that are explored here are amplified Zener noise circuits and linear feedback shift registers (LFSR). Emphasis will be on the topological circuit design of a generic frequency modulator rather than development of a new discrete algorithm for randomization.

Chapter V: Matlab Modeling and Simulation

Simulating Converter Output

A typical DC-DC converter burst mode output voltage waveform essentially looks like a sawtooth waveform in steady state. When the power converter shuts off to conserve energy, the output voltage drifts downward until it is below the converter's burst mode low-voltage threshold. Figure 5.1 is a typical application circuit of the burst mode operation step-up converter LTC3458. The feedback (FB) pin is a voltage-divided output. The application in Figure 5.1 has a reference voltage of 1.22V, meaning that the converter starts to fire bursts of current when the voltage on the FB pin drops below 1.22V. Noise introduced at the FB voltage will be amplified in the converter output since the FB pin is the result of downscaling the output.

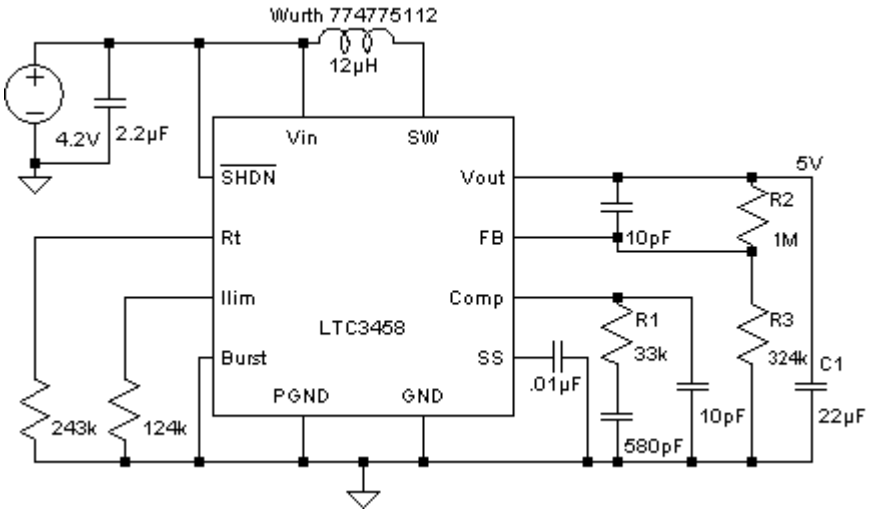
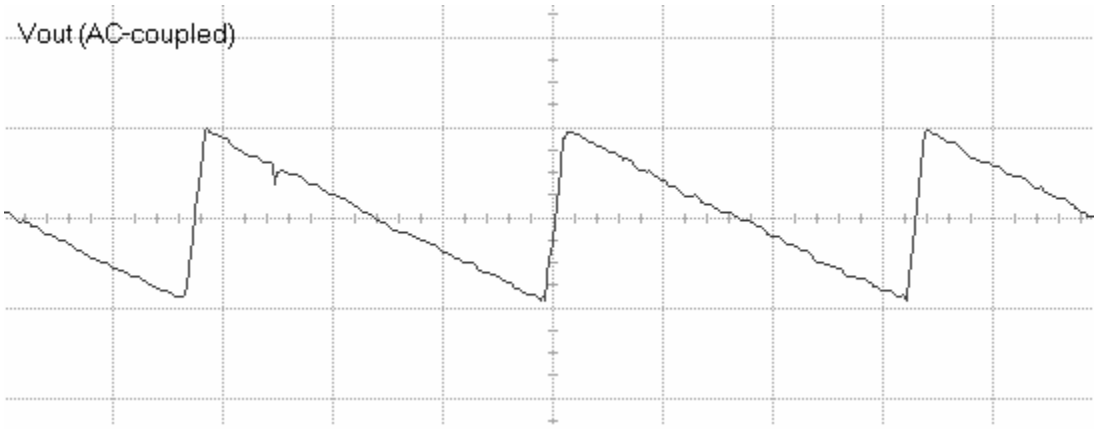


Figure 5.1: LTC3458 with 5V output in burst mode.

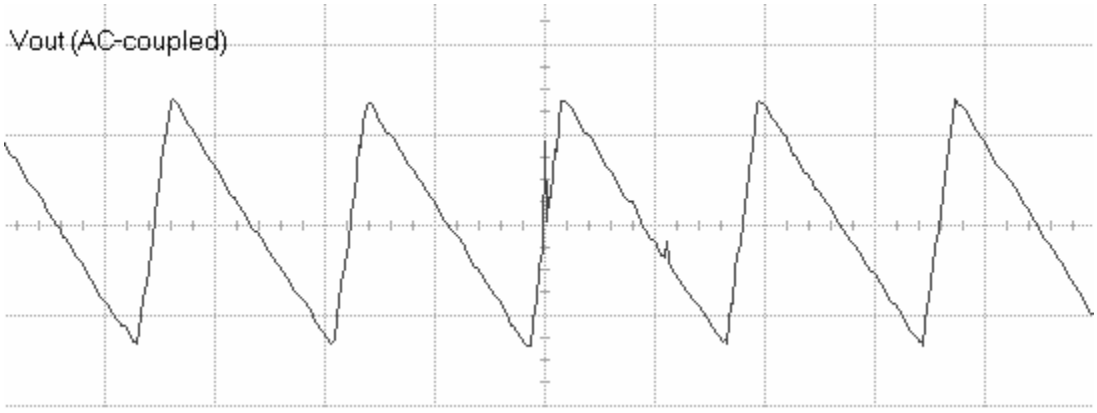
The burst pin is grounded to force the converter into burst mode (a resistor-capacitor combination here can set a threshold at which the converter switches between burst mode and PWM mode). The I_{LIM} resistor sets the converter current limit. The SHDN pin is a simple on/off control for the converter. The SS capacitor is used for the converter's soft start. The R_T pin sets the PWM switching frequency and the COMP pin is linked to the output of the error amp that monitors output voltage. In burst mode, the R_T connection is not relevant, and the COMP output is disabled to save power.

Figure 5.2 compares the output of the LTC3458 with the same above configuration but under different loads. The only different between the two scenarios is that the converter under a heavier load has a steeper rate of voltage falloff and that the ripple is slightly greater in amplitude since the load is greater.



50us/div 100mv/div

(a)

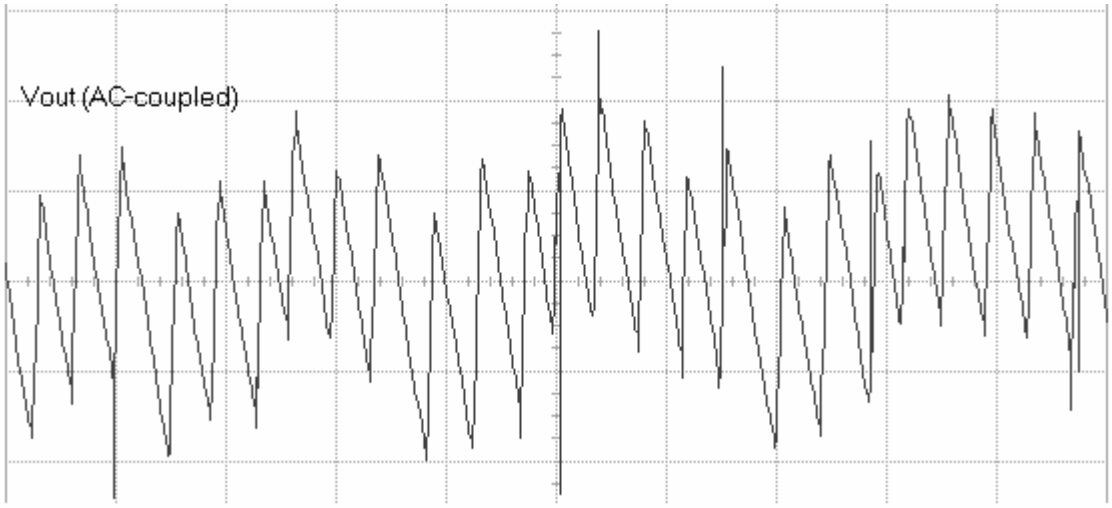


50us/div 100mv/div

(b)

Figure 5.2: AC-coupled output of LTC3458 with two different loads.

Figure 5.3 compares the output of the LTC3458 with the same load but the diagram on the left has a $51mV_{pp}$ AC-coupled square wave directly applied to the FB pin. The injected square wave does not significantly directly couple to the V_{OUT} pin through the resistor connecting the FB pin to the output. Instead, the effects of the injected noise that is reflected at V_{OUT} is the result of noise at the inputs to the feedback comparator.



200us/div 100mv/div

Figure 5.3: AC-coupled output of LTC3458 with spread spectrum switching control.

For example, a +25.5mV$_{DC}$ signal that is directly injected to the FB node should induce V$_{OUT}$ to *fall* an additional 100mV before a current burst is triggered. When the positive signal is applied to the FB pin, the input into the feedback error amplifier is artificially raised above what it really should be (a scaled down version of V$_{OUT}$). This is as if the reference voltage for the error amplifier has been lowered.

$$V_{FB}(dependent) = V_{OUT}(independent) \cdot \left(\frac{324k}{324k+1M}\right)$$

$$\Delta V_{OUT}(dependent) = -\Delta V_{FB}(independent) \cdot \left(\frac{324k+1M}{324k}\right)$$

$$= -25.5mV \cdot \left(\frac{324k+1M}{324k}\right) = -100mV$$

If a constant +25.5mV is applied to the FB node, then the peak voltage to which V$_{OUT}$ climbs is also reduced by 100mV. The entire output waveform is essentially a level shifted version of the original V$_{OUT}$. Thus, a fairly simple model of a DC-DC converter voltage output behavior with injected noise at the Feedback node can be summed up as an approximate sawtooth waveform:

(a) Output voltage slowly drifts lower. The rate of drift (slope of waveform) is approximately linear and inversely proportional to the converter load. The voltage drift actually should model a capacitor discharge (exponential dropoff), but the period of the output waveform is short enough that a linear approximation is accurate enough.

(b) When a downscaled version of the output voltage reaches below a reference level, the converter turns on for a short duration (on chips such as the LTC3458 each burst segment lasts roughly .5us) its current burst, causing the output voltage to rapidly rise. The level to which the output voltage rises depends on the number of burst pulses that take place. The converter stops bursting current when the feedback voltage has been brought back up to an acceptable level.

(c) A lighter load causes each burst pulse to push the output voltage higher than if there were a heavier load. Most DC-DC converters only allow complete pulses. Runt pulses may lessen the average output voltage ripple but degrade converter efficiency.

Figure 5.4 is a flow diagram of the Matlab simulation that models a burst DC-DC converter with the addition of an extra component that injects pseudorandom noise into the system.

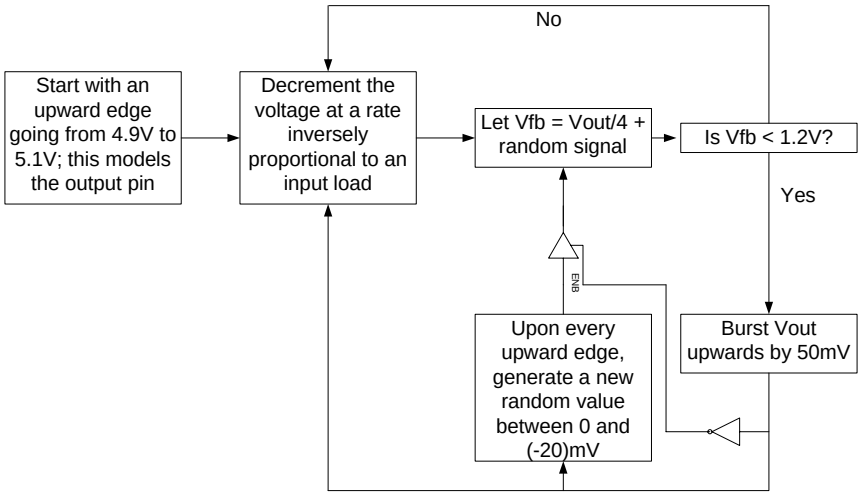


Figure 5.4: Flow chart of Matlab simulation of modified DC-DC converter.

Note that there is a tristate buffer between the pseudorandom noise generator and the actual power converter feedback system. This gate models part of the non-intrusive signal injector. The goal of this buffer is to have a programmable control for controlling the worsening of the converter output voltage ripple due to added noise. Figure 5.5 compares the simulated outputs of the Matlab converter model with (5.5c) and without (5.5d) the noise control buffer.

The two main differences between a spread spectrum system with and without a noise control buffer are (1) the output voltage ripple and (2) the presence of a noticeable voltage ceiling on the buffered system. Details regarding the specific construction and operation of the noise control buffer are described in chapter IX. For relevance to this simulation, the only the black-box behavior of this controller is needed.

In the modified converter with a noise control buffer, the pseudorandom noise is negatively level shifted such that the frequency of burst can only increase since the negative-level injected noise at the feedback pin will only decrease the input into the converter error amplifier. When the converter is active on an up-burst, however, the noise control buffer prevents negative noise from entering the system so that the converter does not burst too much. The noise control buffer effectively puts a ceiling on the maximum voltage level to which the converter can burst and thus acts as a voltage ripple control. The Matlab code to generate the converter output models are in Appendix A.

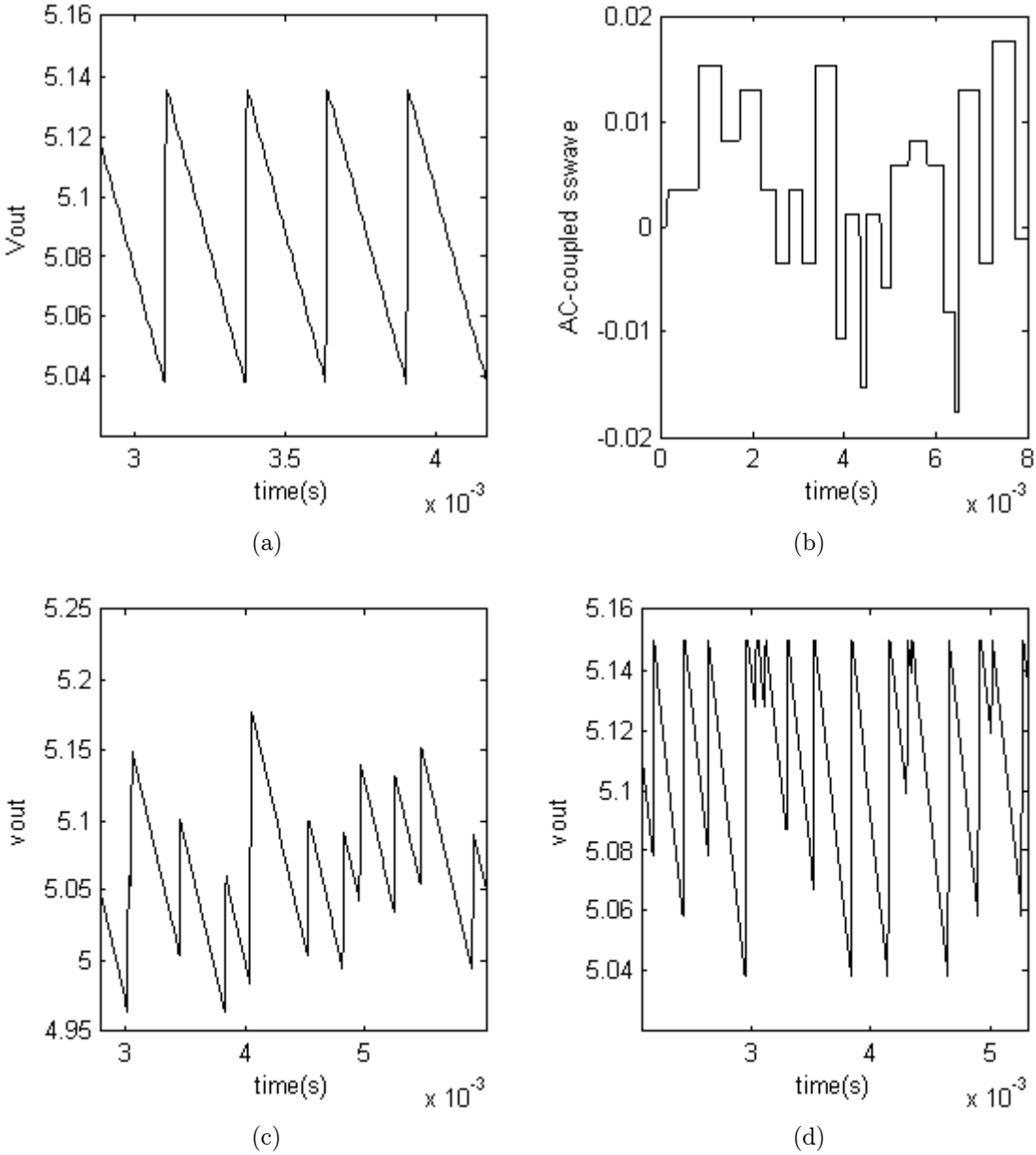
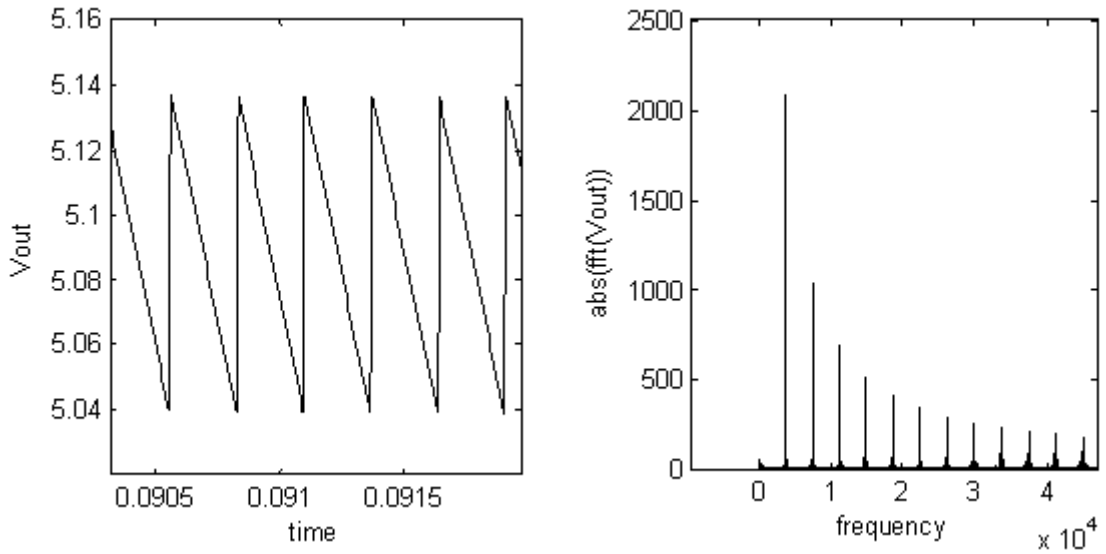


Figure 5.5: (a) The simulated converter output. (b) The simulated 4-bit resolution pseudorandom bit sequence. (c) The simulated converter with an AC-coupled noise injected into its feedback pin. (d) The simulated converter with a level-shifted noise signal injected into its feedback pin and a ripple control system.

Analysis of Simulation

The Fourier transform of the unperturbed converter output illustrates that the simulated waveform is clear sawtooth wave. Figure 5.6a is a time-domain plot of a 4kHz converter output simulation, and Figure 5.6b is the Discrete Fourier transform of the same converter output.



(a) (b)

Figure 5.6: (a) The simulated converter output.
(b) The Discrete Fourier transform of the simulated converter output

A perfect sawtooth wave with fundamental frequency ω_0 and period T can be represented by the equation:

$$Vout(t) = \frac{1}{2} + \sum_{n=1}^{\infty} \frac{1}{n} \sin\left(\frac{nt\pi}{2T}\right)$$

The sawtooth waveform is a sum of sine waves at every integer multiple of the fundamental frequency ω_0. The sine wave at the n-th multiple of ω_0 is scaled down by $1/n$, a property that is well illustrated in Figure 5.8b.

Purer sawtooth waves have more distinct harmonic components. If a sawtooth wave were to be distorted with injected pseudorandom noise, then its integer multiple harmonics in the frequency domain do not look as sharp, as in Figure 5.9 because the energy in each frequency harmonic is spread to nearby frequencies.

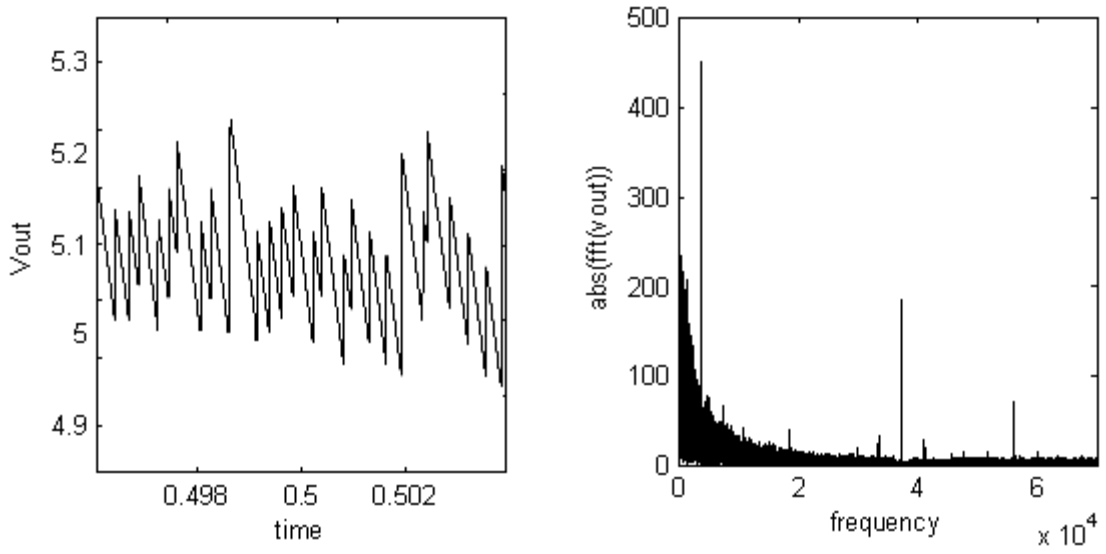


Figure 5.7: (a) The simulated converter output with pseudorandom noise.
(b) The Fourier transform of the simulated converter output.

Even though the output voltage ripple increases from approximately 100mV to 250mV when uniformly-distributed 10Hz-100kHz pseudorandom noise is injected into the system, the magnitude of the harmonic frequencies of the spread spectrum converter are greatly reduced (compare Figures 5.6b and 5.7b). Figure 5.7b shows from a Fourier transform perspective that the original audio harmonics in the converter are not completely suppressed.

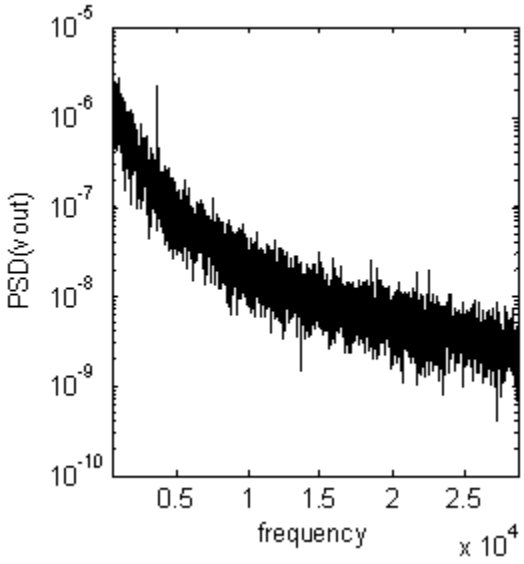


Figure 5.8: PSD of converter output with pseudorandom noise

Figure 5.8 shows the power spectral density (PSD) of the spread spectrum converter output on a logarithmic scale, which highlights the fact that the concentration of power at ω_0 has not been completely spread to neighboring frequencies to create a smooth PSD. The powers at higher harmonic frequencies, however, are adequately dispersed.

The converter harmonics can be spread more effectively by increasing the amplitude of the noise and thus increasing the maximum voltage ripple on the converter output. The tradeoff between converter output voltage ripple and spread spectrum efficacy can appear very obvious when comparing the different power spectrum densities of converters with different magnitudes of pseudorandom noise. Figure 5.9 compares the PSD of the converters with different maximum output ripple voltages and with the same load. Figure 5.9a is the output of a converter with 250mV range of output voltage ripple and Figure 5.11b is the output of a converter with a 400mV range of output voltage ripple. Figure 5.9c, which shows a distinct audio harmonic at roughly 4kHz, is the PSD corresponding to the converter in 5.9a. Figure 5.9d, which does not have any distinct audio harmonics, is the PSD of the converter in 5.9b.

The impact of increased converter output voltage ripple on the characteristic of the audible noise, however, is a complex matter. The actual audible noise in most DC-DC converters is the result of vibrations of the inductor, which originates from channeling current through the inductor. Assuming that the load remains constant, when the voltage ripple becomes greater, the maximum current through the inductor of the power converter does not change. More bursts of current are fired, however, in order to charge the load and push the output voltage to greater amplitudes.

One interesting point to note is that many burst mode DC-DC converters do not allow partial bursts of current. Runt current pulses typically lower the efficiency of the DC-DC converter and therefore many converters contain logic to prevent prematurely ending current pulses. Since in some DC-DC converters there is a minimum length of time for each current pulse, correspondingly there is a minimum output voltage ripple as well as a limited resolution of the different voltage ripple levels in converter steady state.

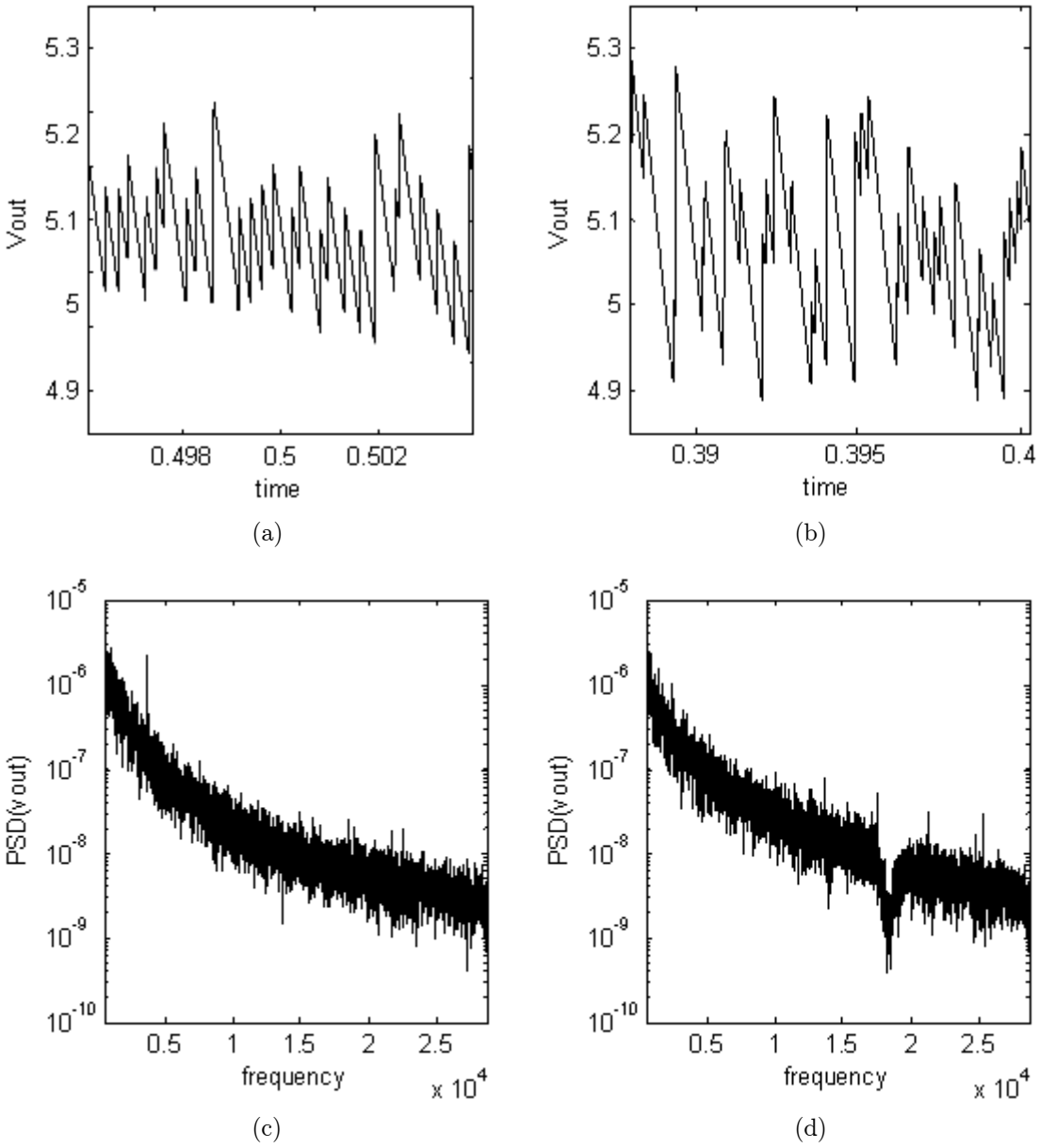
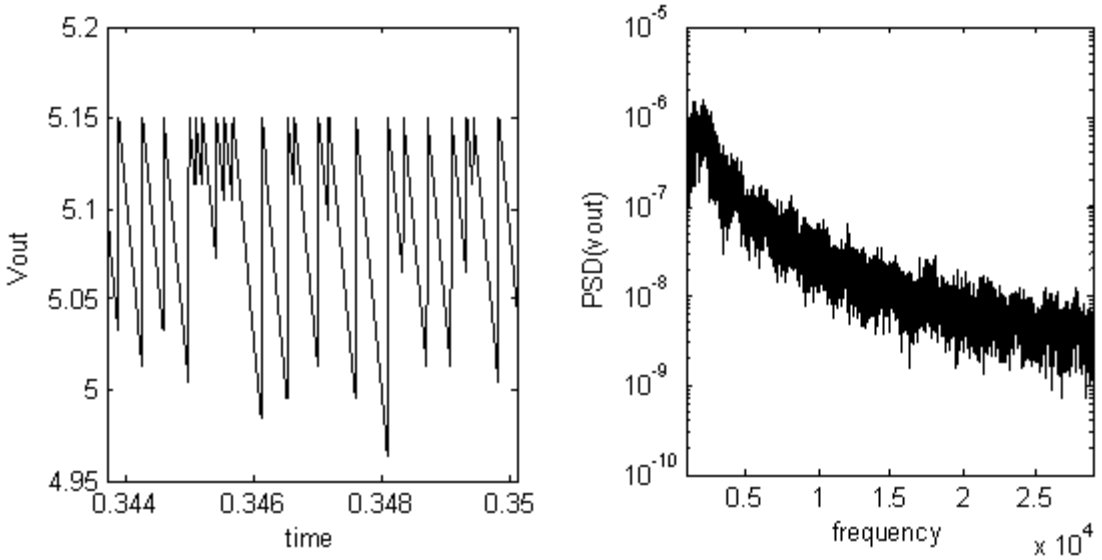


Figure 5.9: (a) Simulation of a converter with 250mV of output voltage ripple.
(b) Simulation of a converter with 400mV of output voltage ripple.
(c) PSD of the converter in (a). (d) PSD of the converter in (d).

The same level of spread spectrum efficacy can be achieved for the spread spectrum converter with a noise control buffer. Figure 5.10 shows the simulated V_{OUT} of such a converter along with the PSD of the output waveform. The PSD of this system does not appear to contain distinct harmonics, indicating that, from a PSD perspective, the system is effective at dithering the audible harmonics in the

corresponding non-spread spectrum system. The difference in this case, however, is that there is a ceiling for V_{OUT} and that the maximum output ripple voltage is controlled.



(a) (b)

Figure 5.10: (a) Simulation of a converter with a noise control buffer.
(b) The PSD of the converter output in (a).

Model Inaccuracies

Pseudorandom vs. Random

One of the factors that the above Matlab model does not capture is the pseudorandom aspect of a non-theoretical spread spectrum system. Chapter VII describes in detail the limitations of the pseudorandom bit sequence generator and its impact. The Matlab model uses a random number generator as random as the intrinsic Matlab seed, which is more "random" than the actual pseudorandom generator.

Voltage Level Resolution

It is difficult to implement many different analog voltage levels in the simulation, construct a digital control system, as well as keep the number of samples and Matlab processing requirement at manageable levels. The actual spread spectrum system involves the use of a DAC to convert the pseudorandom bit sequence into analog voltage levels for compatibility with the converter error compensation system. The exact analog levels are carefully controlled with the DAC and adjusted based on converter and speed requirements. This property is not well modeled in the Matlab simulation; hence, Figure 5.10, for example, only shows a limited number of voltage levels. In practice, when there is a fine resolution of analog voltages, there is even more spread spectrum efficacy. Chapter X, testing and analysis, describes the effects and tradeoffs of voltage resolution.

External Noise

The most well-designed power converters are still susceptible to external noise. For spread spectrum purposes, however, external sources of noise can often be beneficial. For example, the actual Fourier spectrum of a converter output may resemble the basic shape of the Matlab simulation in Figure 5.8b but have a much fuzzier output. If external noise is low enough such that it helps the dithering of audio harmonics and yet does not affect the efficiency of the converter or the voltage ripple level, then the external noise essentially benefits the spread spectrum system. Unfortunately, such external noise is unpredictable, and the goal of this project is to develop a controlled system for spread spectrum. The issue of external noise, however, must be addressed to analyze the effectiveness of the converter spread spectrum system under different operating conditions. This discussion is in chapter X.

Chapter VI: Microcontroller Modeling And Testing

Purpose

As mentioned in Chapter III, *Previous Solutions*, prior spread spectrum modulation implemented for macro-power systems are often conveniently implemented digitally with either DSPs or microcontrollers. The fastest and most economical way to peek at potential results from this spread spectrum system is to implement a converter switching controller digitally. Different implementation details that Matlab simulations may not model accurately can quickly be observed in a physical system. For future reference, any modifications to the current system can easily be tested and implemented to a high degree of accuracy by editing microcontroller code.

Setup

The chosen microcontroller is a Microchip PIC16F876A since it is at this time available free of cost. A readily available Communications Port (for PCs) PIC16PRO programmer with public ROM burning software PICALLW is used to flash Assembly code onto the PIC microcontroller. The working word-for-word code is in Appendix C, and this section describes the microcontroller programming from a high abstraction level.

Besides economic and availability reasons, the main benefits of the PIC16F876A are that it has Analog to Digital Converter (ADC) inputs, PWM outputs, and an acceptable clock speed (16MHz with an external crystal oscillator). The ADC is essential as part of the feedback voltage and ripple monitor, and the high clock speed is crucial to generating and changing PWM analog voltage levels at a speed lower than the PWM carrier pulse frequency.

Traditional microcontroller implementations use a lookup table to generate a pseudorandom bit sequence. While the PIC16F876A has more than enough RAM to store an acceptable random number table, the current setup borrows a pseudorandom bit sequence from a readily available LTC6902 spread spectrum

oscillator chip. Figure 6.1 shows a configured LTC6902 that generates the pseudorandom bit sequence. The maximum switching frequency is governed by the equation:

$$f_{max} = \frac{R_{SET}}{20k\Omega} \cdot \frac{N \cdot M}{10MHz}$$

$N = 1$ by setting the PH pin to 0

$M = 1$ by setting the DIV pin to 0

The frequency spreading percentage is governed by the equation:

$$Spreading = 100 \cdot \frac{f_{max} - f_{min}}{f_{max}}$$

$N = 1$ by setting the PH pin to 0

$M = 1$ by setting the DIV pin to 0

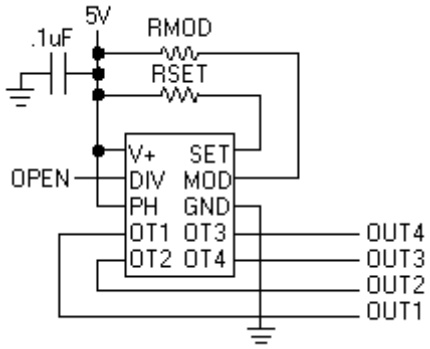


Figure 6.1: Pseudorandom bit sequence generator.

Preliminary Results

For testing, the clock of the microcontroller is always set to a frequency greater than or equal to that of the burst frequency of the power converter. This ensures that at every burst cycle the converter feedback sees a potentially different pseudorandom noise level.

Without a spread spectrum system, the burst mode system audible tone from LTC3421 and LTC3458 DC-DC converters are very noticeable. When the microcontroller pseudorandom noise PWM output is connected to a converter feedback node, the distinct burst mode tone clearly becomes a fan-like pink noise.

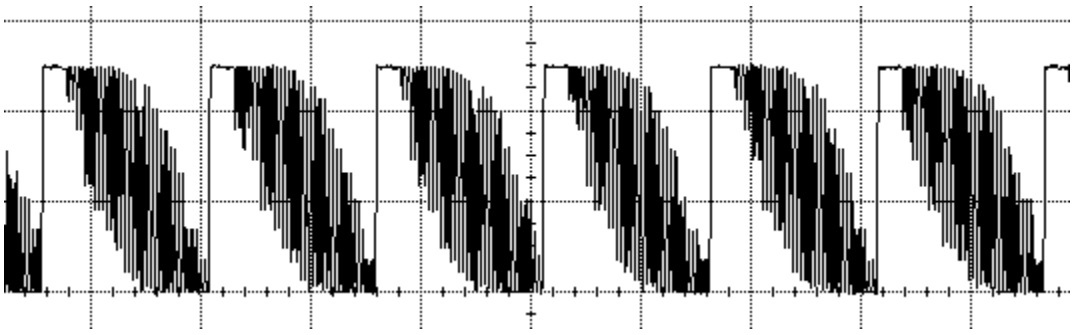
The exact intensity of the noise is not measured at this time, but what is more important is that the basic nature of the audible noise is definitely altered. Also, it is clear that injecting greater magnitudes of pseudorandom noise has greater impact on the audible noise attenuation. This is somewhat obvious since too much noise simply distorts the DC-regulation beyond recognition.

Limitations

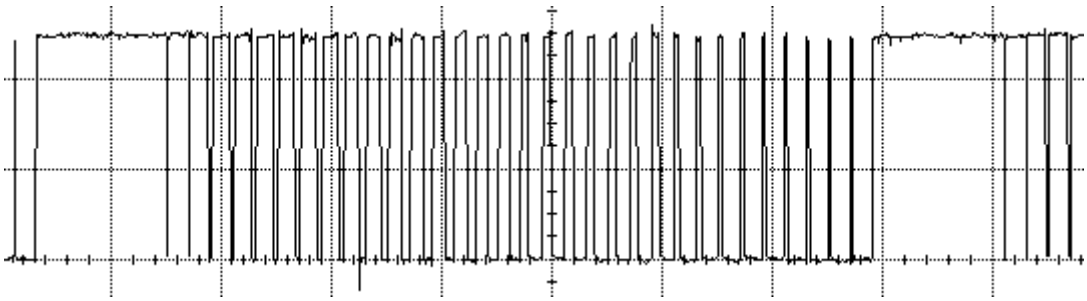
The most noticeable limitation of programming a microcontroller for spread spectrum control is that the microcontroller's PWM oscillation frequency is not fast enough. The PIC16F876A uses an external 16MHz crystal oscillator for clocking but the microprocessor's internal processing requires 4 clock periods for each PWM cycle (each voltage level transition, upslope, level, downslope, level) requires one instructional clock cycle. This causes the microprocessor's PWM switching frequency to be 4MHz at the fastest.

Since the actual spread spectrum system dithers the harmonic frequencies to 16 surrounding frequencies, the microcontroller needs to generate 16 different PWM voltages. If the PWM is run at the maximum 4MHz, then the PWM duty cycle only has two values- 100% and 0%. To accommodate a 16 duty-cycle resolution, the PWM frequency is set at a fixed 250kHz (4MHz/16).

The PWM frequency of 250kHz is an order of magnitude higher than the audio range and thus higher than the maximum required frequency for the spread spectrum system, but 250kHz does not yield smooth waveforms. Figure 6.2 shows a PIC16F876A microcontroller cycling through its 16 PWM outputs at the maximum frequency. It is clear from Figure 6.2a that the microcontroller PWM output is able to span the entire range of voltages from 0-5V. The output waveform, if ideal, is supposed to be a sawtooth waveform, but the digital PWM switching is visibly reflected as the waveform is not smooth at all. The reason for this is highlighted in Figure 6.2b, a time-magnified view of the same sawtooth waveform in Figure 6.2a. Figure 6.2b shows one period of the sawtooth waveform in Figure 6.2a and shows a gradual, 16-piece transition from 100% duty cycle to 0% duty cycle.



(a) 100us/div, 2V/div



(b) 20us/div, 2V/div

Figure 6.2: (a) PIC16F876A cycling through 16 different PWM duty-cycles.
(b) Time-zoomed view of one period of the output in (a); highlighting decreasing duty cycle.

With a random sequence generation lookup table, the microcontroller can only be a rough substitute for the modulation generation module of the spread spectrum system. The microcontroller cannot substitute for any of the analog control subsystems since the ADC of the microcontroller is even slower than the PWM control. Nevertheless, the microcontroller model shows that a spread-spectrum system can indeed affect burst-mode DC-DC converter audible noise in a systematic way.

Chapter VII: Pseudorandom Modulation System

Linear Feedback Shift Registers (LFSR)

Linear Feedback Shift Registers are a common method of generating pseudorandom bit sequences [11]. The LFSR system mainly consists of using an n-bit length shift register and XOR-ing specific combinations of bit outputs in cascaded chain back into the carry-input of the shift register. The carry-out bits that are included in the XOR network are called "taps", and the stored values in the registers are the "states". Figure 7.1 illustrates an example of a LFSR and Table 7-1 shows the corresponding LFSR state transitions. The initial input is set to 1 since the zero-state results in a 1-state loop.

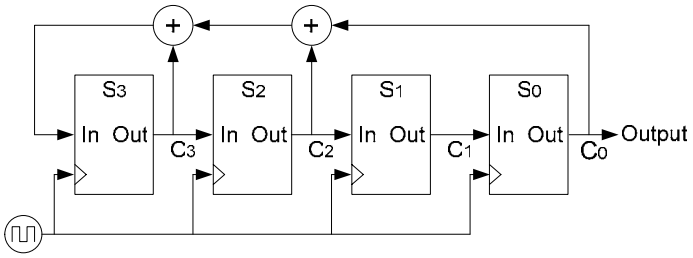


Figure 7.1: 4-bit LFSR with bits C_0, C_2, and C_3 as the taps.

function S3 = (C3 ^ (C0 ^ C2))

Time	S3	S2	S1	S0	C0 (out)
0	1	0	0	0	-
1	1	1	0	0	0
2	0	1	1	0	0
3	1	0	1	1	0
4	0	1	0	1	1
5	0	0	1	0	1
6	0	0	0	1	0
7	1	0	0	0	1

Table 7-1: State transitions of 4-bit LFSR with bits C_0, C_2, and C_3 as the taps.

For an n-bit sequence, there are $(2^n - 1)$ possible states. The zero-state is omitted. The LFSR in Figure 7.1 is an example of an unoptimized 4-bit LFSR since only 8 states are produced when there are 16 possible states. The LFSR in Figure 7.2 is an example of an optimized LFSR since all 16 possible states.

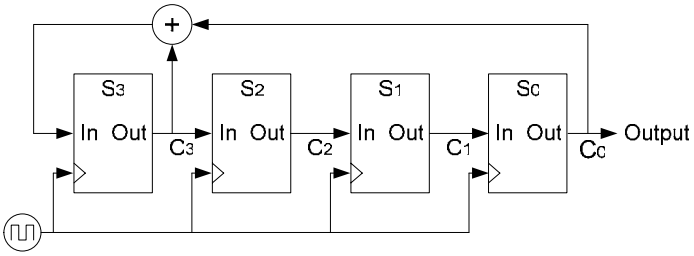


Figure 7.2: 4-bit LFSR with first and last output as the taps.

function S3 = (C0 ^ C3)

Time	S3	S2	S1	S0	C0 (out)
0	1	0	0	0	-
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	0	1	1	1	1
5	1	0	1	1	1
6	0	1	0	1	1
7	1	0	1	0	1
8	1	1	0	1	0
9	0	1	1	0	1
10	0	0	1	1	0
11	1	0	0	1	1
12	0	1	0	0	1
13	0	0	1	0	0
14	0	0	0	1	0
15	1	0	0	0	1

Table 7-2: 4-bit LFSR with first and last output bits as the taps.

In [12] there is a detailed explanation of the method to find optimal taps for all-lengths of LFSRs. To give a brief summary of the described optimization method: model the LFSR as a polynomial, and the LSFR is optimal if the polynomial is primitive. The polynomial can be written as:

$$f(x) = c_0 + c_1 x + c_2 x^2 + \ldots c_{n-1} x^{n-1} + x^n$$

$$c_x = \begin{Bmatrix} 1 \text{ if bit x is tapped} \\ 1 \text{ for x = n and x = 0 by definition} \\ 0 \text{ else} \end{Bmatrix}$$

The polynomial models of the LFSRs in Figure 7.1 and 7.2 can be modeled as:

$$f_{7.1}(x) = x^4 + x^3 + x^2 + 1 = (x+1)(x^3 + x + 1)$$
$$f_{7.2}(x) = x^4 + 1$$

The criteria for a polynomial to be primitive and the corresponding LFSR to be optimal, as [12] describes, are:

(a) let the period of f(x) be the smallest m for which f(0) divides $x^m + 1$
(b) an irreducible polynomial is a polynomial that cannot be factored
(c) a primitive polynomial is an irreducible polynomial with period 2^n-1

The LFSR in Figure 7.1 is not irreducible and therefore is not primitive. Table 7-3 lists the minimal number of taps for the optimal n-bit LFSR systems. The specific tap combination for each n-bit LFSR is not necessarily unique, but using two taps (1 XOR gate) has obvious chip layout area and power consumption benefits.

The required length of the LFSR depends on the clock that feeds the registers. Since the LFSR cycles through at most 2^n-1 possible states before repeating, there are harmonics created in the audible frequency spectrum (20-20kHz) when the LFSR is not long enough. For the spread spectrum DC-DC converter, the clock feeding the registers is at the fastest just beyond the audible spectrum at 20kHz. Even though 20-20kHz is the typical audible frequency range, it is safer to set 24kHz as the clock frequency and 16Hz as the repeat frequency and exceed the minimum requirements. A 10-bit LFSR barely meets the 20-20kHz bound, and adding one more register does not greatly increase the amount of circuitry. With more conservative bounds, the LFSR for the spread spectrum DC-DC converter must be at least 11-bits long:

$$\text{LFSR periodic frequency} = \frac{f_{clock}}{2^n - 1}$$
$$f_{clock} \geq 24k\text{Hz}$$
$$\frac{f_{clock}}{2^n - 1} \leq 16\text{Hz}$$
$$\frac{24k\text{Hz}}{20\text{Hz}} \leq 2^n - 1$$
$$\underline{n} \geq 11$$

From table 7-3, an optimal 11-bit LFSR has taps at bit 11 (output) and bit 1 (second bit from the input). Such a LFSR cycles through 2047 possible states, and with a 20kHz clock, it has a period of 9.78Hz, which is well outside the audible range. This implementation, however, assumes that there is one LFSR state transition for every period of the converter burst. In reality, there may be several state transitions for every converter burst period (described later). If, for example, there are two state transitions for every burst period, then the above calculations for the LFSR period must be halved. The length of the LFSR must be longer to accommodate the effective LFSR cycle frequency.

If, for example, the length of the LFSR is 13, then period of the LFSR cycle, compared with the period of an 11-bit LFSR, is quadrupled. If the 11-bit LFSR accommodates a burst-mode converter with a LFSR that changes state once per burst cycle, then the 13-bit LFSR accommodates up to 4 state changes per burst cycle. Figure 7.3 shows an example of a LFSR that sometimes switches states twice and sometimes once per DC-DC converter output cycle. Since the converter switching cycle can more than one LFSR period, and since only the LFSR output at the burst-time of the converter is effectively sampled by the converter error controller, an n-bit LFSR can potentially generate less than 2^n-1 converter burst thresholds.

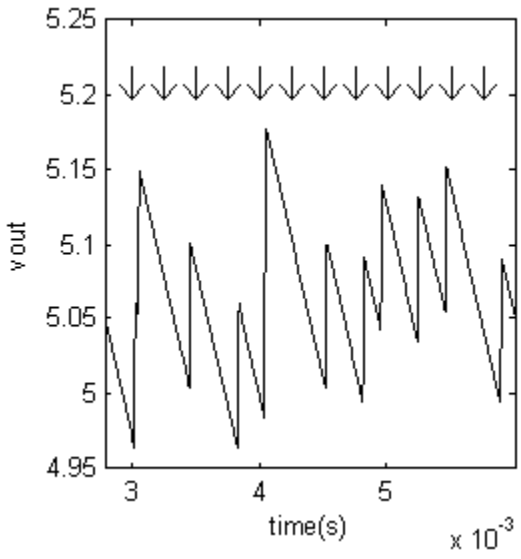


Figure 7.3: Simulated converter voltage output with arrows representing the LFSR clock.

Unfortunately the number of XOR gates required for optimal 12, 13, and 14-bit LFSRs is three instead of one. For sake of convenience the LFSR used for this thesis' spread spectrum system is an optimal 15-bit LFSR with only 1 XOR. From an IC and layout perspective, a 13-bit LFSR may be more appropriate since additional flip-flops may consume more power and require more layout area than additional XOR gates.

Figure 7.4 shows a straightforward implementation of a 15-bit LFSR with discrete components. 74HC595s are 8-bit serial-in and parallel-out shift registers. There is a simple soft-start circuit attached to the serial input of the input end of the LFSR. During power-up, the soft-start forces a high value into the LFSR input to break the LFSR out of the zero-loop. R_{10} and R_{11} can be made even bigger and C_1 can be made smaller to conserve power as long as the soft-start forces the high voltage for long enough a time and the LFSR actually absorbs the high voltage into its first state. It does not matter whether the forced high voltage lasts for more than one clock cycle (thus forcing a different start state) since the optimized LFSR cycles through every possible combination of states.

No. of bits	States	Taps
2	3	0 1
3	7	0 2
4	15	0 3
5	31	1 4
6	63	0 5
7	127	0 6
8	255	1 2 3 7
9	511	3 8
10	1023	2 9
11	2047	1 10
12	4095	0 3 5 11
13	8191	0 2 3 12
14	16383	0 2 4 13
15	32767	0 14
16	65535	1 2 4 15
17	131071	2 16
18	262143	6 17
19	524287	0 1 4 18
20	1048575	2 19
21	2097151	1 20
22	4194303	0 21
23	8388607	4 22
24	16777215	0 2 3 23
25	33554431	2 24
26	67108863	0 1 5 25
27	134217727	0 1 4 26
28	268435455	2 27
29	536870911	1 28
30	1073741823	0 3 5 29

Table 7-3[1]: Shortest optimal taps for LFSRs.

[1] R. Sung, A. Sung, P. Chan, and J. Mah, "Linear Feedback Shift Register," University of Alberta Electrical Engineering, http://www.ee.ualberta.ca/~elliott/ee552/studentAppNotes/1999f/Drivers_Ed/lfsr.html.

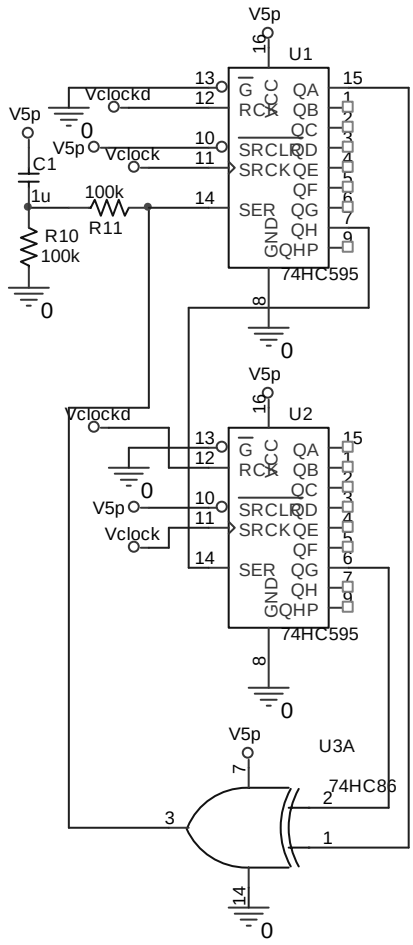


Figure 7.4: 11-bit LFSR implemented with discrete components.

A main advantage of LFSRs is that they are straightforward to implement. Basic flip-flops and logic can be constructed purely from MOS transistors, making very feasible to implement in an IC. Adding additional registers to the LFSR does not take up much additional layout space.

Another benefit of LFSRs is that they have very low power and supply voltage requirements, even in discrete form. Both 74HC86 XOR gates and 74HC595 DIP packages come in 2V versions. When no spread spectrum is desired, the switching power for the shift register can be avoid simply by silencing the clock input to the register.

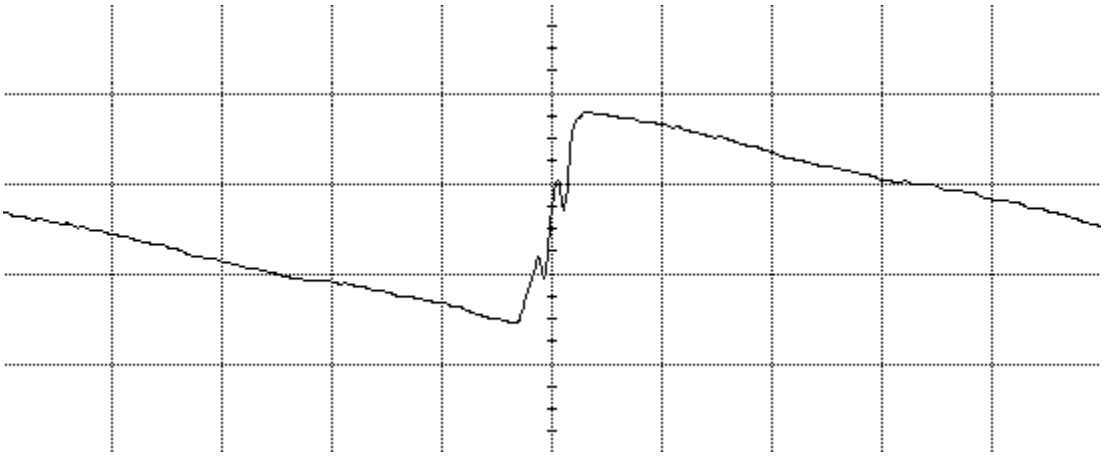
Clock generation is a source of difficulty for implementing LFSRs. In synchronized switching converters, the LFSR can simply borrow (and upscale and downscale as needed) the switch control signal from the converter's internal oscillator. In burst

mode DC-DC converters, however, the internal oscillator of the converter is shut down to conserve quiescent power.

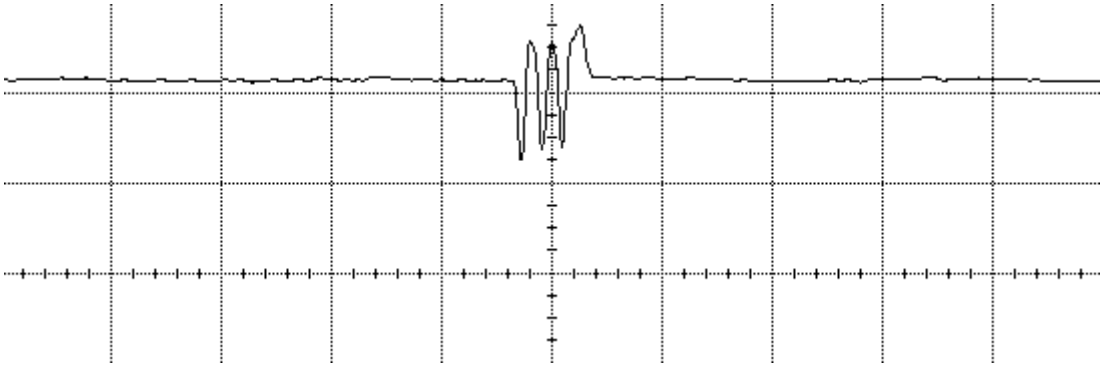
In a DC-DC converter IC environment, the converter's error internal amplifier output can be fed into the LFSR as a clock. In a discrete components setting, the voltage on the converter's inductor is used as a clock. When the converter initiates a burst of current, there is an energy transfer through the inductor, and the voltage on the inductor momentarily goes low. Since the voltage on some inductors tend to have spikes, and since the clock input should put as little capacitance load on the inductor, a voltage buffer stage (which is also a key component in the output ripple monitor, described in the next chapter) couples the inductor pin to the clock input of the LFSR. The simplest yet effective buffer consists of two MOS inverters in series.

Figure 7.5a shows the V_{OUT} pin of a LTC3458 DC-DC converter operating in burst mode. Figure 7.5b shows the corresponding inductor voltage. There is a shuffling of current during every burst; henceforth, the voltage on the inductor oscillates for every burst of current that the converter outputs. In Figure 7.5, there are three total bursts of current in one burst mode cycle. Correspondingly, there are three observed 2V dips in the voltage of the inductor in Figure 7.5(b). The output of the buffer in Figure 7.5(c) corresponds to the change in voltage on the inductor. This signal in Figure 7.5(c) feeds into the LFSR as a clock. In this example there are three voltage spikes in one burst period, and thus there are three LFSR state changes per converter burst period. Unfortunately this leads to power inefficiency since there are unnecessary switching power losses.

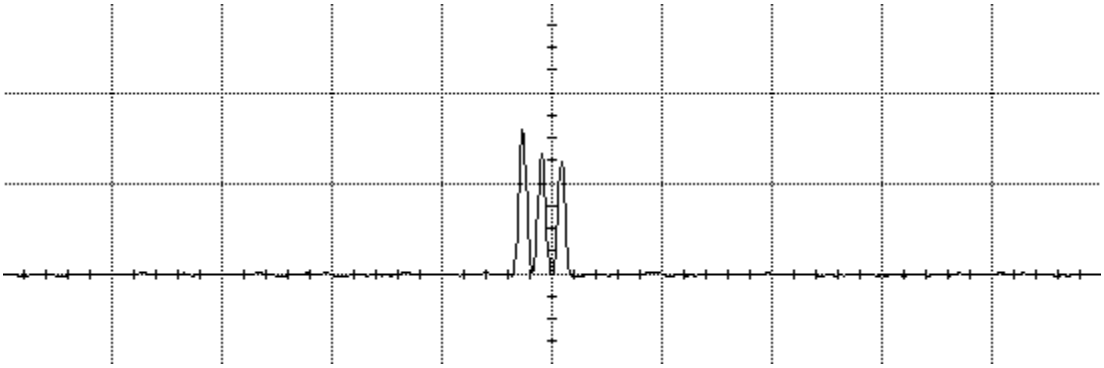
A possible modification to using the raw current burst indicator on the inductor pin as the LFSR clock is to low-pass this signal to shorten the necessary LFSR length. Using a filter, however, may not be power efficient when contrasted to lengthening the LFSR digital chain.



(a) 20us/div, 2V/div



(b) 20us/div, 2V/div



(c) 20us/div, 2V/div

Figure 7.5: (a) LTC3458 V_{OUT} AC-coupling.
(b) Corresponding inductor voltage (SW pin); DC-coupling.
(c) Output of the buffer and the generated clock signal; DC-coupling.

Avalanche Noise Amplifier

An analog noise-generation alternative to using LFSRs is the amplification of pink noise generated from the avalanche breakdown in a PN junction. According to [13] pp. 755-756, the holes and electrons in the depletion region of a reverse-biased PN junction constantly collide with silicon atoms and create more hole-electron pairs (avalanche effect). When enough hole-electron pairs are present, a random burst of current can occur.

Figure 7.6 shows the schematic for a pink-noise generator using a reverse biased diode, the base-emitter junction of transistor Q2. The circuit, an adaptation of the frequency dithering system in [14], consists of the noise source, Q2, and two op-amp gain stages. For practicality, the only power supply used is a single ended 5V. Consequently, the op-amps are configured in a single supply configuration. Capacitors C_4 and C_{44}, in series with the non-inverting op-amp feedback resistors R_4 and R_{44}, respectively, allows the op-amp to act as a DC-buffer and an AC-amplifier. The signal inputs to the op-amps are biased around 2.5V to avoid potential clipping issues. Capacitors C_3 and C_5 are feed-forward capacitors that increase the upper bandwidth of the op-amps.

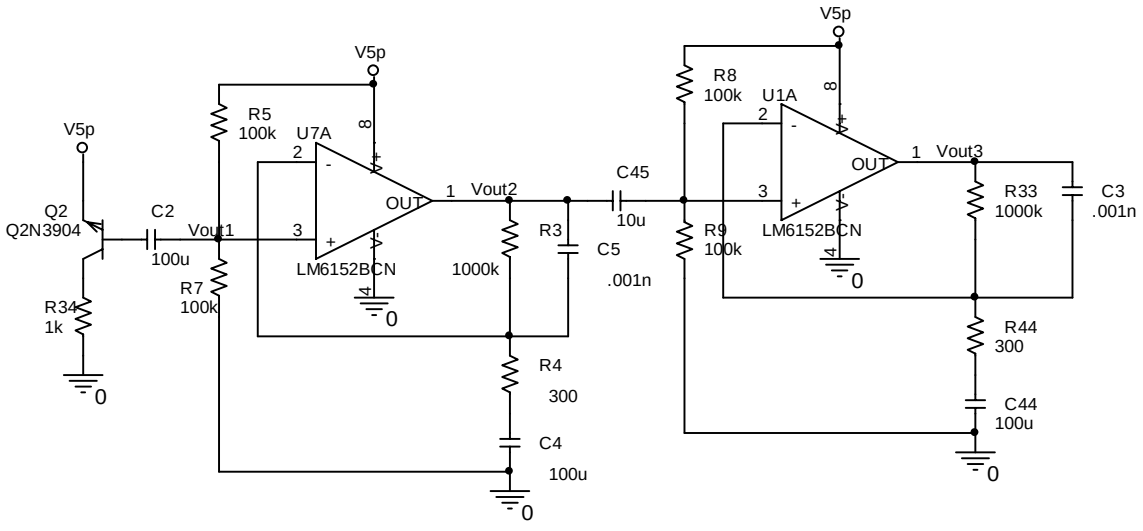


Figure 7.6: Schematic for a pink-noise generator utilizing avalanche breakdown noise in a PN junction.

A low voltage Zener diode or a noise diode can be substituted for the reverse-biased transistor junction as the noise source. According to p. 755 of [13], PN junctions reverse-biased with greater voltages provide noise proportional to the voltage-squared; hence many published Zener diode noise generator circuits use supply voltages greater than 15V. In an IC setting, the supply voltage may be very limited and the straightforward remedy to this issue is to increase the amplification in the gain stages.

Fortunately the bandwidth of the pink noise for this spread spectrum application only needs to be 20-20kHz. The open-loop gain-bandwidth product of the 6152 op-amp is approximately 75MHz, which allows each op-amp stage to be configured for a maximum closed-loop gain of 3750. The above circuit configures each op-amp stage for a gain of 3333, and cascades two equal-gain stages for an overall mid-band gain of 1.1×10^7. Figure 7.7 is a noise simulation of the circuit in Figure 7.6, and it illustrates that the simulated bandwidth meets the 20-20kHz requirement. The noise simulation in SPICE includes thermal and shot noise (explained in the next section) as well. If the high-frequency bandwidth is not enough, a straightforward solution is to divide the amplifier into more than two stages. This, of course, consumes more power.

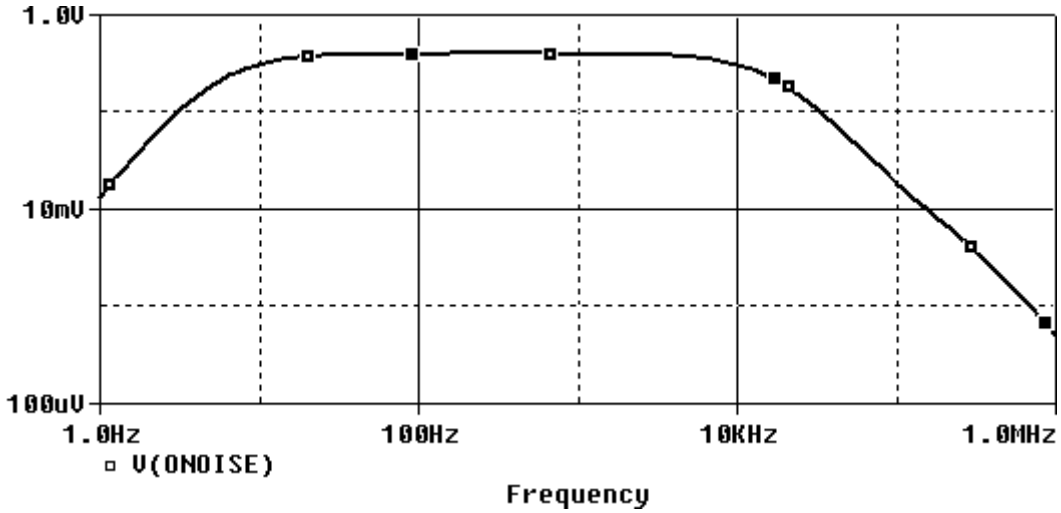
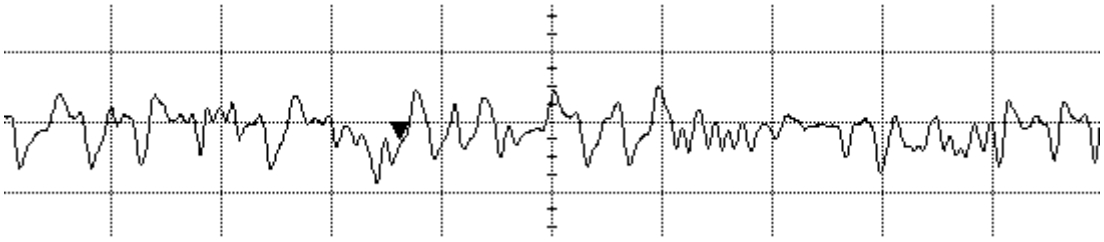


Figure 7.7: Noise AC simulation of the circuit in Figure 7.5.

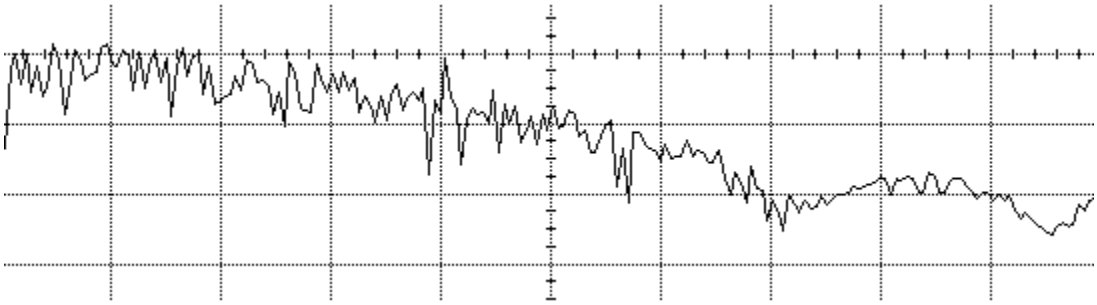
The low frequency roll-off in the above AC simulation is caused by the AC-coupling capacitors C_4, and C_{44}. AC-coupling capacitors C_2 and C_{45} are in series with large resistances (the parallel combination of the biasing resistors and the op-

amp input resistance), and therefore do not contribute significantly to the low-frequency bandwidth. To accommodate 20Hz the capacitors need to be relatively big. Another potential disadvantage of dividing the amplifier to more gain stages is that each gain stage requires an additional large coupling capacitor.

Figures 7.8a and 7.8b are test captures of the Figure 7.5 circuit output. Figure 7.8a shows the band-limited time-domain scope waveform-display. Figure 7.8b, the FFT of the waveform in 7.8a, confirms that the amplified noise is indeed white in the 20-20kHz range since the frequency spectrum is relatively flat. The high-frequency portion of Figure 7.8b follows the simulation in Figure 7.7.



(a) 100mV/div, 2kHzus/div, centered at 10kHz



(b) 20dB/div, 2kHzus/div, centered at 10kHz

Figure 7.7: (a) 20MHz band-limited time-domain scope capture of the avalanche noise generator. (b) FFT of the noise in waveform in (a).

A challenge in building a noise source with an avalanche diode is that the peak amplitude of the noise varies from device to device. Since the spread spectrum system must regulate the magnitude of the noise in order to regulate the DC-DC converter's output ripple, building a precise white noise generator in this fashion is extremely difficult in concept. Imposing a ceiling and a floor on the output is not a directly viable regulation scheme since it forces the noise to clip and thus changes the frequency shape of the noise.

Avalanche Noise Clipped Amplifier

A variation of the avalanche noise generator simply substitutes a comparator instead of the last op-amp gain stage or simply has an overall gain so high that the output reaches the output limit voltage level. If the amplifier output is then put through a low-pass filter, then this system essentially becomes a pseudorandom bit-sequence generator like a LFSR. Just like a LFSR, a DAC is needed at the output of a clipped avalanche noise amplifier to translate the bit-sequence into multiple voltage levels.

Unlike the LFSR, however, the avalanche noise clipped amplifier does not have parallel outputs. A sample-hold circuit is needed to capture successive time-interval values of the bit-sequence in order to feed 2^n possible input values into the DAC, where n is the length of the captured bit-sequence. Figure 7.9 shows the sample-and-hold scheme. Although the sample-hold circuit does not need to be accurate (since the input is essentially digital), the sample-hold must finish a complete capture cycle with a minimum 20kHz. If the bit-length for each capture cycle is 4, for example, then the sample-hold needs to sample at 80kHz.

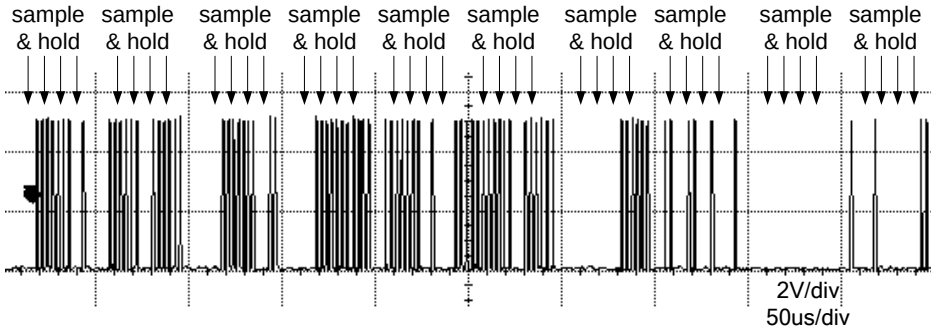


Figure 7.9: Scope capture time-domain output waveform of a clipped avalanche noise generator.

Glitch with Jitter Generator

There are many ways to double or triple a clock's frequency without actually modifying the clock generator itself. A frequency multiplier is needed if for example, the clock for a LFSR is not fast enough or if the clock for the sample-hold circuit is not fast enough (since the only available burst-mode DC-DC converter

on-chip clock source is the feedback error amplifier output). A straightforward and flexible way to increase the switching frequency by generating glitches is to feed a cascade of delay elements into a collapsing tree of XOR gates, as in Figure 7.10.

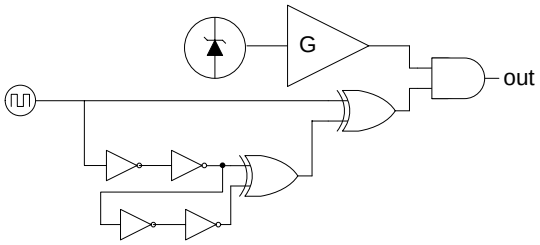


Figure 7.10: Gate-level diagram of a random glitch generator.

In Figure 7.10, an input clock signal is delayed through a series of inverters and then is fed into XOR gates. Assuming that the circuit is designed such that longest propagation delay through the longest chain of inverters is less than the duty cycle-period of the clock, after the rising edge of the clock signal passes through the upper-most XOR gate, the output of that gate remains high until the time-delayed clock signal propagates to the same location. In Figure 7.10, after the clock propagates through the longest chain of inverters and to the final XOR gate again, the output of that XOR gate is restored to its original high value. This essentially creates two rising clock edges for every clock period. The series of inverters can be lengthened according to the pattern in 7.10 to create additional glitches and increase the output clock rate.

The Zener diode and amplification stage at the top of Figure 7.10 is an optional component. By combining avalanche noise and any XOR output into an AND gate in Figure 7.10, the glitch output at that XOR gate has a probability of not passing through. This introduces a method of creating a pseudorandom bit-sequence whose length and characteristics (such as maximum bit-length and probability of occurrence for each individual bit within a cycle) that is easily controlled.

Other Sources of Noise

Described in [13] are burst noise, thermal noise, and flicker noise, all three of which are smaller low frequency sources. Since the amplitude-distributions of these noises are not Gaussian, amplifying these noise does not provide a white or adequately pink noise source. A filter system can be used to flatten the frequency spectrum for these noises, but it is much easier to simply account for these noises when amplifying avalanche noise and adjust the low-frequency bandwidth of the amplifier.

For example, as mentioned in [13], the spectral density of combined burst, flicker, thermal, and avalanche noises is often higher at the low frequencies. The spectral density can roughly be flattened simply by decreasing the low-frequency gain of the noise amplifier. In the circuit in Figure 7.5, the low-frequency cutoff can be raised by decreasing the value of the AC-coupling capacitors.

Table 7-4 compares the different pseudorandom noise generators described in the earlier portions of this chapter. LFSR is the pseudorandom noise generator of choice, since its implementation is flexible and straightforward.

Circuit Type	Comparative Advantages	Comparative Disadvantages
LFSR	Simple logic	Needs clock
	Easily modified	Needs a DAC
Avalanche noise amplifier	No clock needed	Supply voltage needs to be high
	Fewer transistors	Unreliable output amplitude
Clipped noise amplifier	Fewer transistors	Needs a sample-hold
	Easier bandwidth requirement	Needs a DAC
Glitch generator	Difficult timing requirements	Needs a VCO
	Controlled probability distribution	Needs a DAC

Table 7-4: Comparison of different pseudorandom noise generators.

Chapter VIII: Non-Intrusive Noise Injection And Transient Control

Digital to analog conversion and level shift

The avalanche noise amplifier mentioned in the previous chapter produces an AC-coupled output that must be level-shifted to the correct level before being injected into the feedback node of the DC-DC converter. Since testing of this project's spread spectrum system works with the LTC3442 and LTC3421 DC-DC converters, on which the feedback pins are centered on 1.22V, the level-shift circuitry is designed around the 1.22V reference.

Described in chapter V, injection of AC-coupled noise can effectively dither the frequency domain harmonics at the cost of increased converter output voltage ripple. To directly AC-couple avalanche noise into a DC-DC converter, only a voltage buffer is needed. An emitter-follower or source-follower can act as a coarse level shifter.

Translating a pseudorandom bit-sequence output from a LFSR into a level-shifted requires a DAC stage. Fortunately no sample-hold circuit is needed for LFSRs since LFSRs' intermediate bits are all stored and the shift register system has parallel outputs.

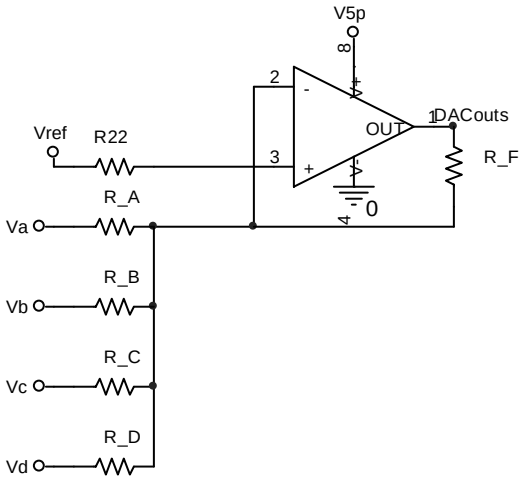


Figure 8.1: Single-supply op-amp DAC and level-shifter circuit.

An n-bit input DAC outputs 2^n different voltage levels; a 4-bit input DAC has a 16-level resolution. Figure 8.1 shows a single-stage op-amp DAC that converts a 0-5V digital signal into 16 different analog voltage levels within a 200mV range centered on a DC voltage that is proportional to the reference voltage.

To solve for the general DAC transfer function:

$$V_{OUT} = A_V\left(V_+ - V_-\right)$$
$$= A_V\left(V_{REF} - V_-\right)$$
$$\frac{V_- - V_A}{R_A} + \frac{V_- - V_B}{R_B} + \frac{V_- - V_C}{R_C} + \frac{V_- - V_D}{R_D} + \frac{V_- - V_{OUT}}{R_F} = 0$$
$$\rightarrow \begin{pmatrix} R_B R_C R_D R_F\left(V_- - V_A\right) \\ + R_A R_C R_D R_F\left(V_- - V_B\right) \\ + R_A R_B R_D R_F\left(V_- - V_C\right) \\ + R_A R_B R_C R_F\left(V_- - V_D\right) \\ + R_A R_B R_C R_D\left(V_- - V_{OUT}\right) \end{pmatrix} = 0$$
$$\rightarrow V_-\left(R_B R_C R_D R_F + R_A R_C R_D R_F + R_A R_B R_D R_F + R_A R_B R_C R_F + R_A R_B R_C R_D\right)$$
$$= \begin{pmatrix} V_A\left(R_B R_C R_D R_F\right) + V_B\left(R_A R_C R_D R_F\right) + V_C\left(R_A R_B R_D R_F\right) + V_D\left(R_A R_B R_C R_F\right) \\ + V_{OUT}\left(R_A R_B R_C R_D\right) \end{pmatrix}$$
$$\rightarrow V_{OUT} = \begin{pmatrix} A_V V_{REF} \\ - A_V\left(\dfrac{V_A R_B R_C R_D R_F + V_B R_A R_C R_D R_F + V_C R_A R_B R_D R_F + V_D R_A R_B R_C R_F}{R_B R_C R_D R_F + R_A R_C R_D R_F + R_A R_B R_D R_F + R_A R_B R_C R_F + R_A R_B R_C R_D}\right) \\ - A_V\left(\dfrac{V_{OUT} R_A R_B R_C R_D}{R_B R_C R_D R_F + R_A R_C R_D R_F + R_A R_B R_D R_F + R_A R_B R_C R_F + R_A R_B R_C R_D}\right) \end{pmatrix}$$

let $\gamma = R_B R_C R_D R_F + R_A R_C R_D R_F + R_A R_B R_D R_F + R_A R_B R_C R_F + R_A R_B R_C R_D$

$$\rightarrow V_{OUT}\left(1 + A_V \frac{R_A R_B R_C R_D}{\gamma}\right)$$
$$= \begin{pmatrix} A_V V_{REF} \\ - A_V \dfrac{V_A R_B R_C R_D R_F + V_B R_A R_C R_D R_F + V_C R_A R_B R_D R_F + V_D R_A R_B R_C R_F}{\gamma} \end{pmatrix}$$
$$\rightarrow V_{OUT} = V_{REF}\left(\frac{\gamma}{R_A R_B R_C R_D}\right) - V_A\left(\frac{R_F}{R_A}\right) - V_B\left(\frac{R_F}{R_B}\right) - V_C\left(\frac{R_F}{R_C}\right) - V_D\left(\frac{R_F}{R_D}\right)$$

For this particular DAC the output voltage should be in the form :

$$V_{OUT} = 1.2 - kV_A - 2kV_B - 4kV_C - 8kV_D$$

where V_{OUT} has a maximum of 1.2V and a minimum of 1V,
and the inputs are either 5V or 0V.

$$\rightarrow .2 = 5k + 2{\cdot}5k + 4{\cdot}5k + 8{\cdot}5k$$

$$\rightarrow k = 1/375$$

$$\rightarrow R_A = 375R_F,\ \ R_B = \frac{375}{2}R_F,\ R_C = \frac{375}{4}R_F,\ R_D = \frac{375}{8}R_F$$

$$\gamma = R_B R_C R_D R_F + R_A R_C R_D R_F + R_A R_B R_D R_F + R_A R_B R_C R_F + R_A R_B R_C R_D$$

$$\rightarrow \gamma = \frac{1}{2}\cdot\frac{1}{4}\cdot\frac{1}{8}k^3 {R_F}^4 + \frac{1}{1}\cdot\frac{1}{4}\cdot\frac{1}{8}k^3 {R_F}^4 + \frac{1}{1}\cdot\frac{1}{2}\cdot\frac{1}{8}k^3 {R_F}^4 + \frac{1}{1}\cdot\frac{1}{2}\cdot\frac{1}{4}k^3 {R_F}^4 + \frac{1}{1}\cdot\frac{1}{2}\cdot\frac{1}{4}\cdot\frac{1}{8}k^4 {R_F}^4 = \frac{1}{4}{R_F}^4$$

$$\rightarrow V_{OUT} = V_{REF}\left(\frac{\frac{15}{64}\cdot 375^3 {R_F}^4 + \frac{1}{64}\cdot 375^4 {R_F}^4}{\frac{1}{1}\cdot\frac{1}{2}\cdot\frac{1}{4}\cdot\frac{1}{8}\cdot 375^4 {R_F}^4}\right) - \frac{1}{375}V_A - \frac{2}{375}V_B - \frac{4}{375}V_C - \frac{8}{375}V_D$$

$$\rightarrow V_{OUT\,\max} = V_{REF}\left(\frac{15}{375} + 1\right)$$

$$\rightarrow V_{REF} \approx 1.15385\text{V}$$

One major advantage of using this type of op-amp DAC for prototyping is that the DAC's maximum output voltage is only affected by the input reference voltage if the feedback resistor R_f is kept constant. Also, the output range of the DAC can be scaled by changing R_f or by scaling R_A through R_D. Modifying the output DC bias through adjusting the reference voltage does not change the output range, and modifying the output range through scaling R_A through R_D does not change the output DC bias. Changing R_f, however, complicates matters since it affects both the output range and the output DC bias.

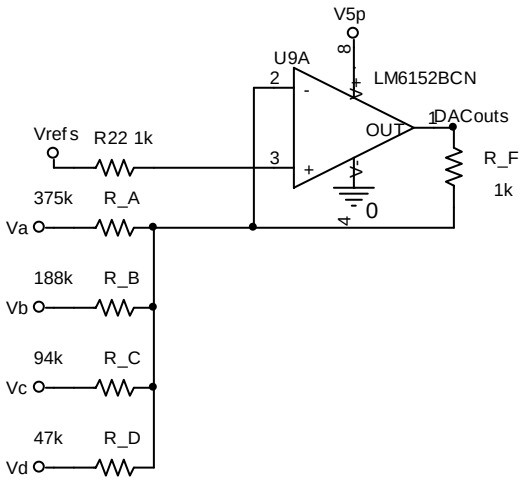


Figure 8.2: Op-amp DAC with configured output range and output DC bias.

For the DAC circuit in Figure 8.2, the output voltage range for 0-5V inputs is [1.0V, 1.2V], and the transfer function is:

$$V_{OUT} = V_{REF}\left(\frac{15}{375}+1\right) - \frac{1}{375}V_A - \frac{2}{375}V_B - \frac{4}{375}V_C - \frac{8}{375}V_D$$

Figure 8.3 shows a simulation of the DAC circuit in Figure 8.2. The four input inputs, V_A through V_D, are staggered such that the DAC output voltage cycles through all 16 possible output voltage levels in one period. Described in [15] and in [16], The current-scaling R-2R-2^nR topology is the most straightforward DAC implementation since it requires the fewest parts. The disadvantage of this topology is that the input resistors may need to be large multiples of the feedback resistor, such as in this case. With the same non-inverting op-amp, an R2R ladder could also be used, with the advantage that the input resistors do not need to be monstrously large with increasing resolution. Other topologies, such as current-switching and charge-scaling DACs, can run at faster speeds, but for this application and proof of concept the simplest DAC solution suffices.

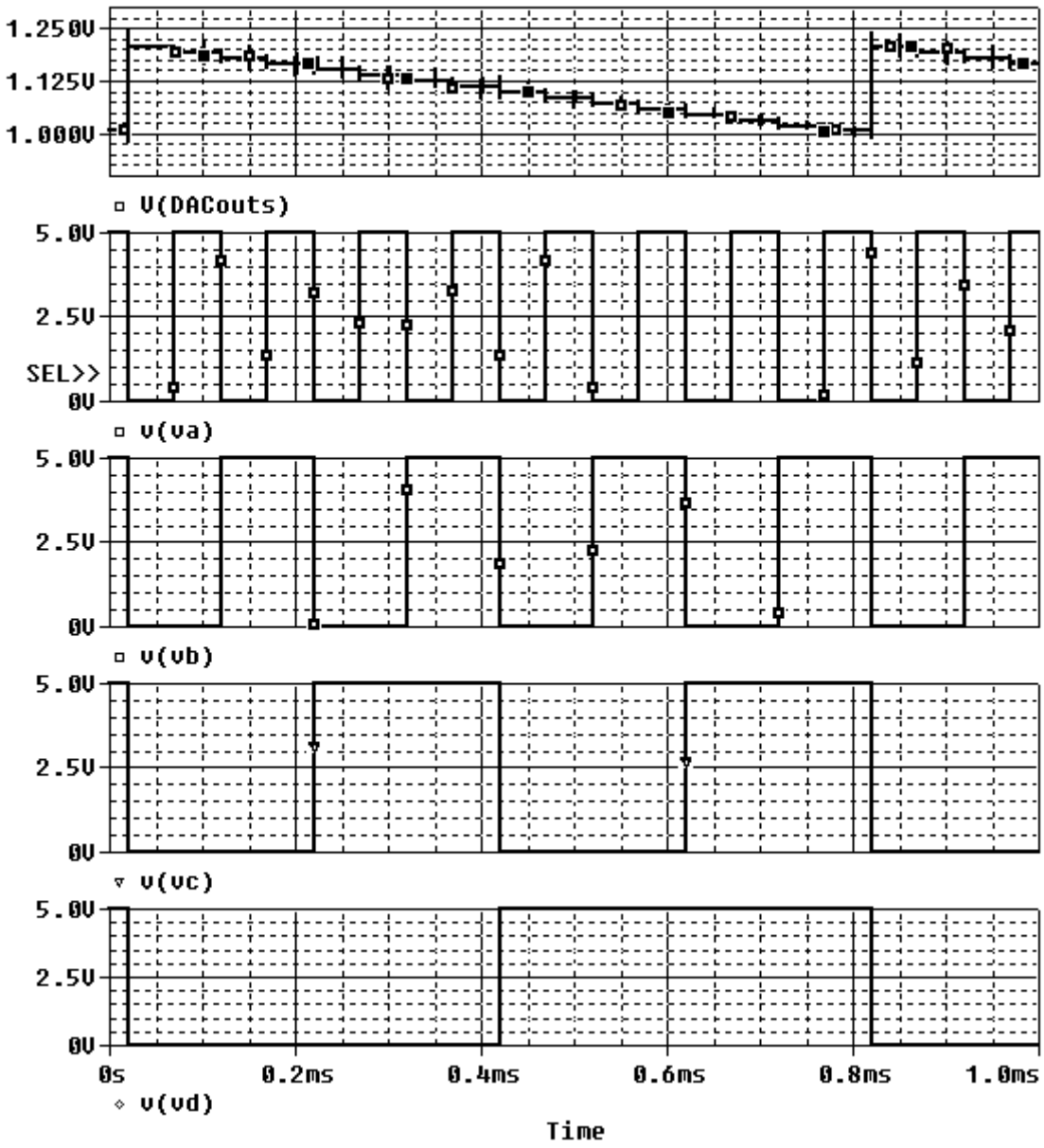


Figure 8.3: Simulation of op-amp DAC circuit in Figure 8.2.

On/Off Noise Injection Control

Mentioned in chapter III, the purpose of adding a noise injection transient control is to keep the converter output voltage ripple in check. For AC-coupled noise, both a voltage floor and ceiling should be implemented to keep the output voltage in an absolute range. For level-shifted noise, however, only one of the two bounds may be necessary with an on/off control switch.

Either an imposed voltage floor or ceiling can limit the voltage ripple by the same amount. Implementing a voltage floor or ceiling involves systems that are different in subtle ways, and both cases have advantages and disadvantages. The biggest argument against voltage floors is that in burst mode, the downward drift time of the converter can vary greatly between a few microseconds and milliseconds, potentially making timing requirements for imposing a voltage floor difficult. The biggest argument against voltage ceilings is that many converters do not allow runt pulses (partial current bursts) in order to improve efficiency. Consequently, it may not be possible to impose a true "hard ceiling". For example, if a burst of current occurred when the voltage is very close to the ceiling, the voltage will shoot up past the ceiling. The maximum value that the voltage will reach, however, is the one current burst's worth of voltage above the voltage ceiling. In this sense, there is still a maximum voltage that can come out of the converter, but this voltage depends on the height of the voltage step caused by one full current burst.

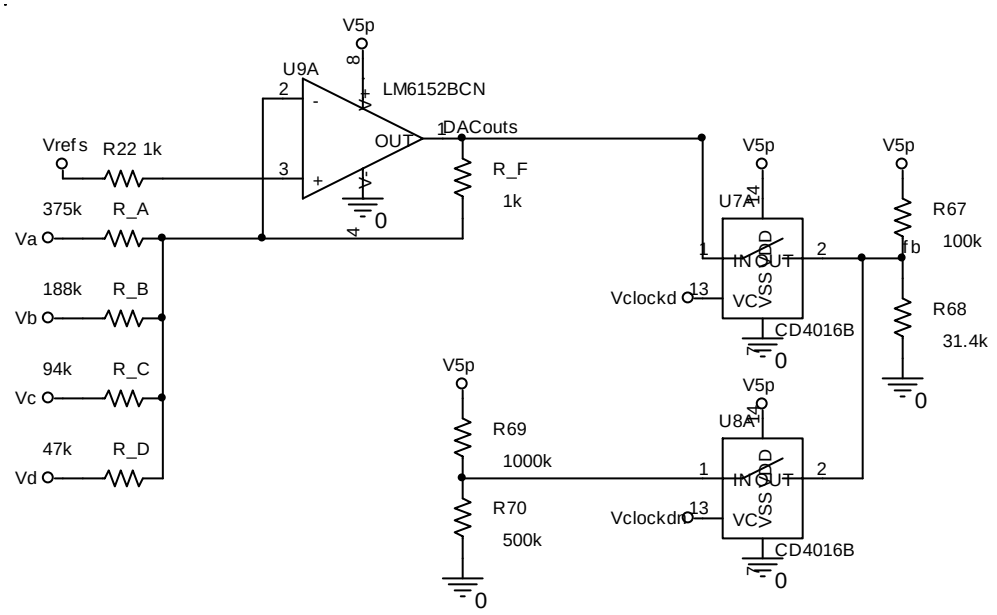


Figure 8.4: Two analog switches forming an analog mux acting as an on/off switch.

The most robust and straightforward way of imposing a floor or ceiling is by passing the injection noise through an analog switch. Figure 8.4 shows the schematic of the pseudorandom noise output DAC coupled through two analog switches that form an analog mux. The switch imposes either a ceiling or a floor on the converter output by not letting the injected noise pass either during

current-bursting or during voltage down-drifting. If, for example, the analog switch does not let the noise signal pass when the converter's output is drifting down, then the converter's feedback error amplifier will always trigger at the default preset reference voltage, assuming that the error amplifier's hysteresis is not affected by what happens during current bursting.

A shortcut alternate solution to the analog switch is to split the feedback node's voltage divider into smaller series resistors and put a FET in parallel with one of them, as in Figure 8.5. Shorting resistors between the feedback pin and ground decreases V_{FB}, and shorting resistors between the feedback pin and V_{OUT} increases V_{FB}. Either On-resistance of the FET that acts as the shorting switch or the resistor that the FET shorts must be scaled such that parallel combination of the original feedback resistor and the saturated FET brings V_{FB} to the desired level. If this shorting FET is used as a pseudo-off switch that nullifies injected pseudorandom noise, and since the output of the DAC is a level-shifted signal that is lower than the reference 1.22V at the feedback pin, the FET should be in parallel with a resistor between V_{OUT} and V_{FB}. Unfortunately, using pull-up MOSFETs to short resistors between V_{OUT} and V_{FB} is very difficult with discrete components since the FET suffers from Body effect.

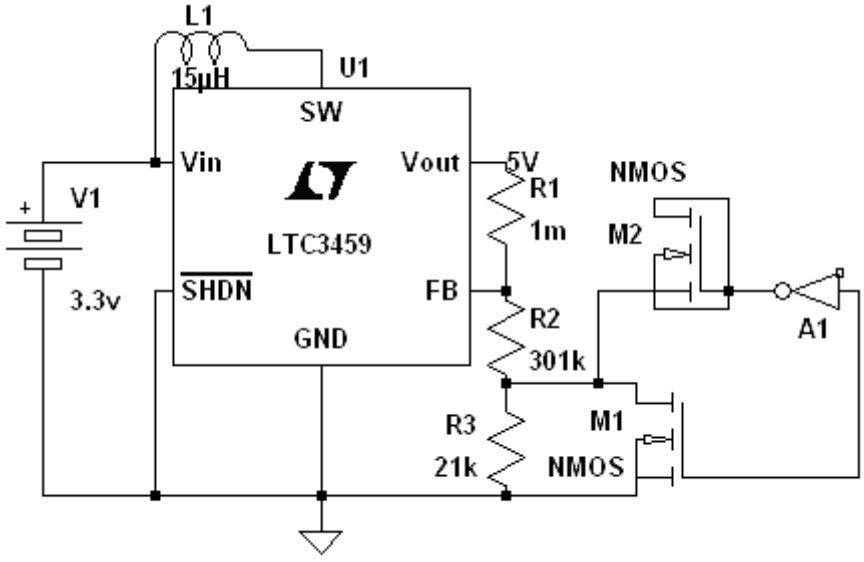


Figure 8.5: Method of changing V_{FB} via shorting the feedback resistors.

In Figure 8.5, the AC-coupled inverter, A1, that bridges the gate and drain of the M1 FET, nullifies gate-drain coupling of the M1 FET. FET M2's drain and source are shorted such that M2 just acts like a capacitor that matches the parasitic capacitors of M1.

Extending the idea of using FETs to short feedback resistors is the circuit in Figure 8.6, in which the need for a 4-bit DAC can be eliminated if FET Body-effects are handled through proper resistor and FET scaling. In Figure 8.6, the digital inputs into the FET switch system are the same pseudorandom bit-sequence from before. In the multiple-FET-switch configuration, however, there is a bottleneck for the signal path, and multiple analog switches need to be applied.

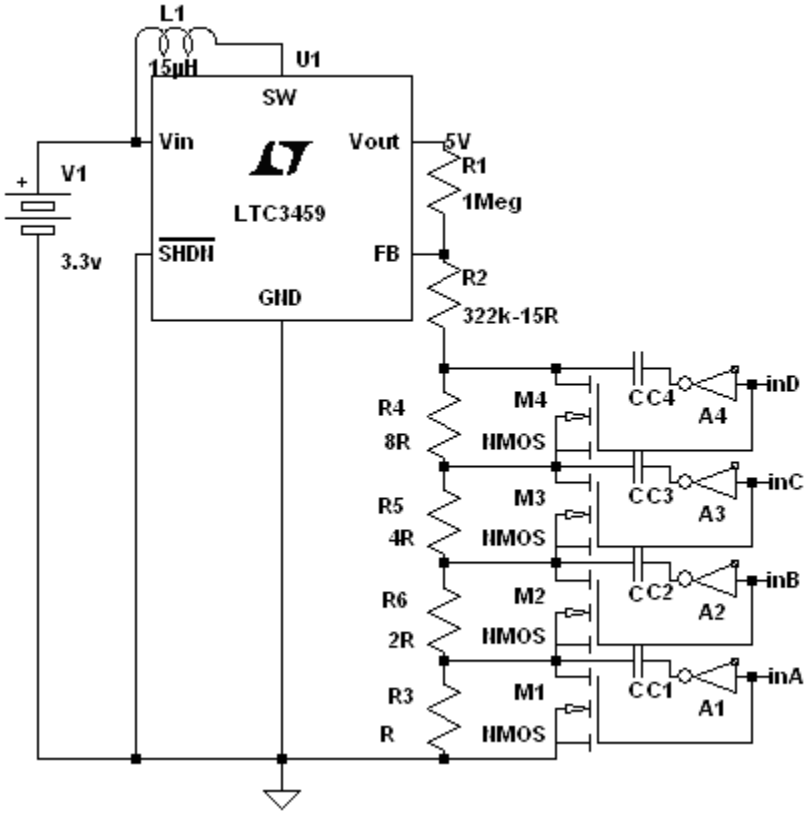


Figure 8.6: Method of discretizing V_{FB} via shorting the feedback resistors.

On/Off Signal

If operating in off-during-burst mode, the spread spectrum system needs to detect and shut off the noise injection approximately within 2us after the start of a current burst, as indicated by the voltage step-size in Figure 7.4. Otherwise, if the injected noise is below the reference voltage 1.22V, then the error amplifier will be fed a lower voltage than the pure feedback voltage and the converter will release too many current bursts. The analog switches in Figure 8.4 have maximum propagation delays of 200ns; thus, the speed bottleneck rests in the logic circuitry that generates the on/off clock.

Chapter III noted that a voltage ceiling is more desirable than a voltage floor since the current bursts occur in discrete packets and adding noise during a current burst runs into voltage granularity problems. Earlier sections of this chapter noted that a hard voltage ceiling is unfortunately much more difficult to implement. Similarly to having a choice of analog switch or analog mux implementations, there are also different methods to generate the on/off or selector control signal.

The most straightforward method is to pass the converter's inductor pin voltage through a high input impedance voltage buffer. If the voltage buffer is a simple series of two inverters, then the intermediate inverted signal can conveniently be used to feed the negative selection signal in the analog mux, as in Figure 8.7.

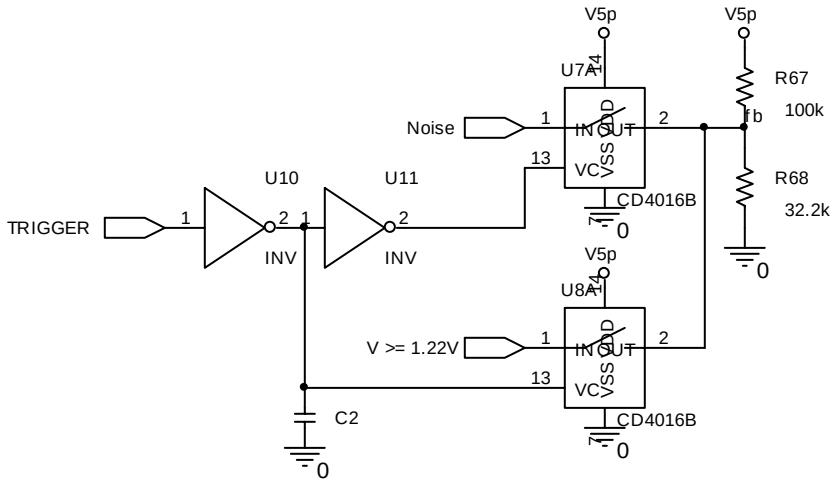
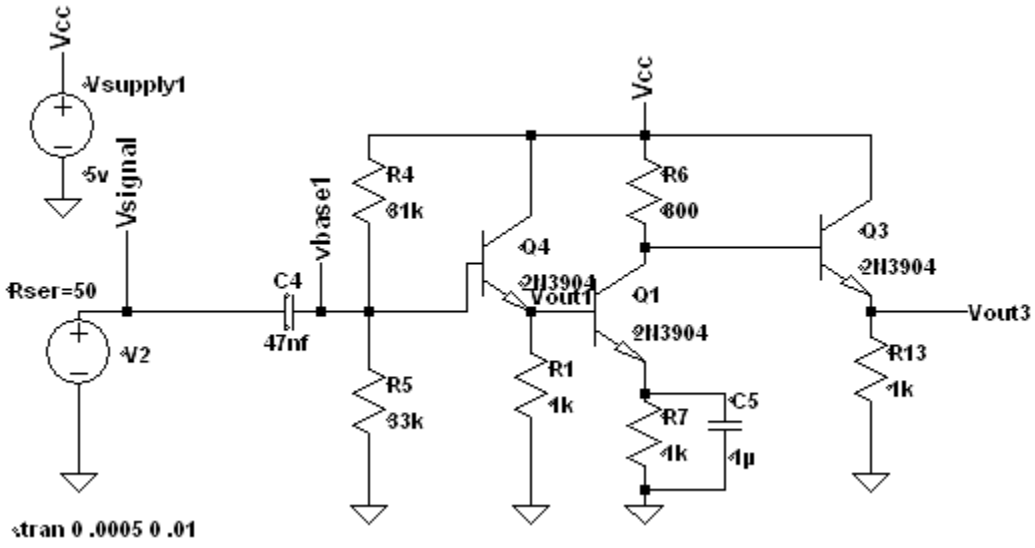


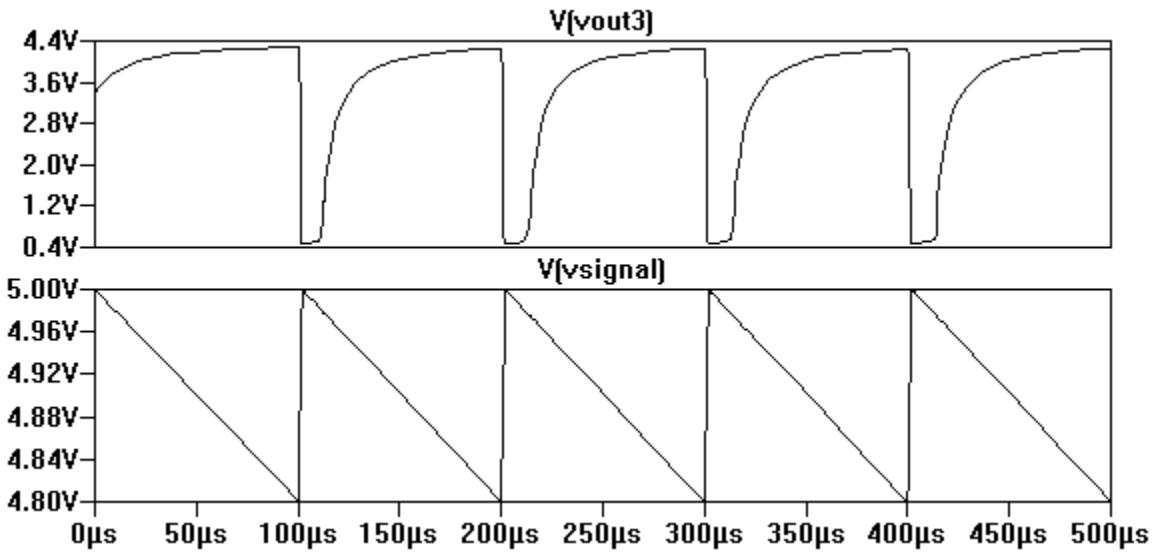
Figure 8.7: Using converter's inductor voltage to feed into mux selector.

Directly coupling the inductor voltage to the input trigger in the above circuit is the simplest and fastest implementation, but rests on the premise that the change in inductor voltage is above the threshold voltage of the inverters. Also to note from Figure 7.4 is that within one switching converter period, the inductor voltage can shuffle multiple times since the current bursts are broken down into small and discrete units. Since the noise should be completely cut off during the entire current-burst phase, the inductor signal must go through a low-pass filter formed with capacitor C_2. Capacitor C_2 is intentionally not placed on the inductor to prevent possible LC effects. On the IC level, if there is an available control signal, such as the error comparator output, then the low-pass filter may not be necessary.

Alternatively the converter output sawtooth waveform can be amplified into a square wave. Figure 8.8a shows an example of a simple discrete component amplifier that shapes a 200mV sawtooth waveform into an approximate square wave. The important feature to note is that the output of the amplifier immediately inverts the moment that the converter output swings up. The output of the amplifier, in Figure 8.8b, effectively acts as the on/off switch to the analog mux.



(a)



(b)

Figure 8.8: 5V 1MHz amplifier with discrete components: (a) circuit (b) transient simulation.

A third way to generate the analog mux on/off signal is to detect the slope of the converter output voltage. Figure 8.9 shows the comparator-delay combination, in which the comparator tracks the difference in power converter output at two different points in time. When a burst of current occurs, the slope of the output voltage becomes positive, and the output of the comparator flips, shutting off the pseudorandom noise. When the power converter stops bursting current, the slope of the output voltage waveform becomes negative and the output of the comparator flips again, turning on the pseudorandom noise. This method, in concept, is the most accurate, but also most complex because of the comparator.

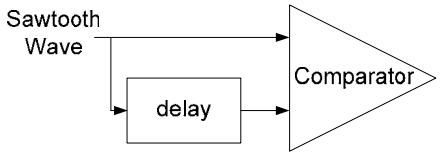


Figure 8.9: Block diagram of simple slope detector.

All three methods (direct error-amp or inductor voltage coupling, output waveform amplifier, and slope detector) serve the same purpose. From a design perspective, the simplest and most power-efficient (direct coupling) is the easiest and best choice.

Chapter IX: Testing And Analysis

There are four metrics by which the spread spectrum converter is measured in this thesis: audible noise, output ripple, regulation, and efficiency. Since this spread spectrum system is implemented with discrete components, it is not possible to directly measure the effect of adding of such a system on the quiescent current of a DC-DC converter. Tested in the following section is the LTC3458 DC-DC converter, operating in burst mode with 5V output. The testing was conducted at room temperature at Linear Technology's New Hampshire design center. All original DC-DC converter components were surface mount and soldered onto a PC board. The additional control parts of the circuit, which includes the LFSR, on/off controller, DAC, and SW-pin inverters, are all DIP ICs or through-hole discrete components soldered onto a single-sided vectorboard. The complete circuit is in Figure 9.1 below:

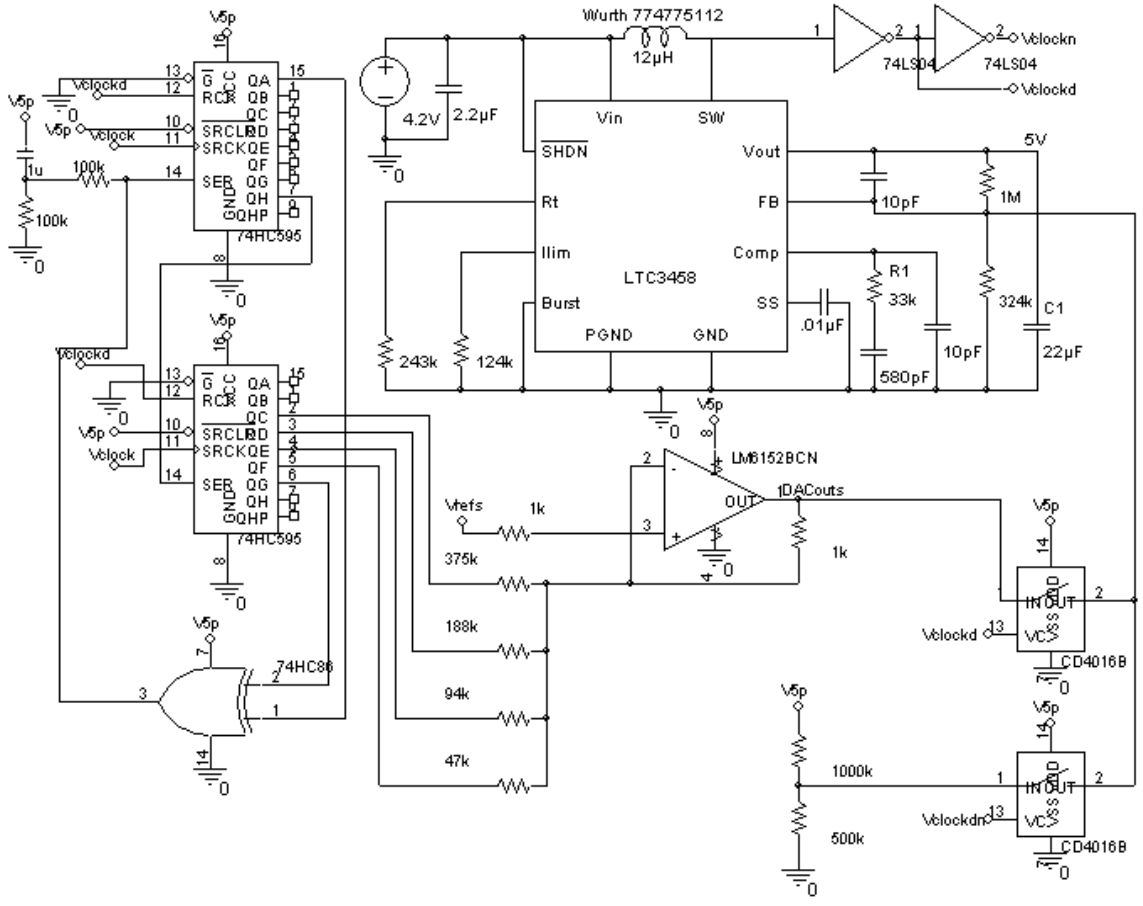
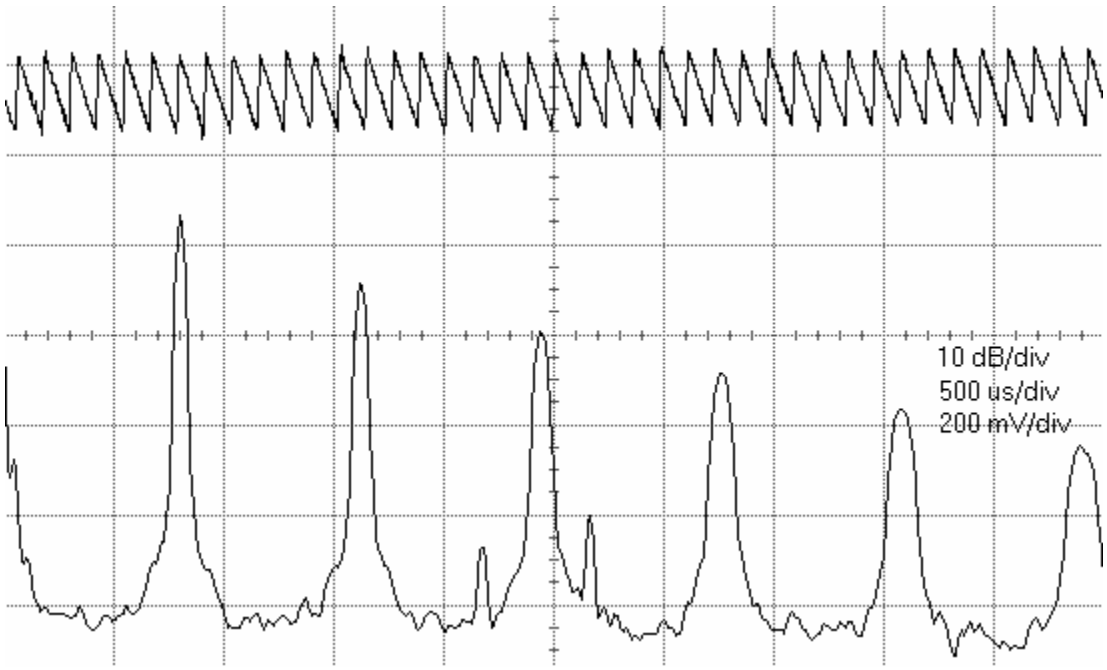


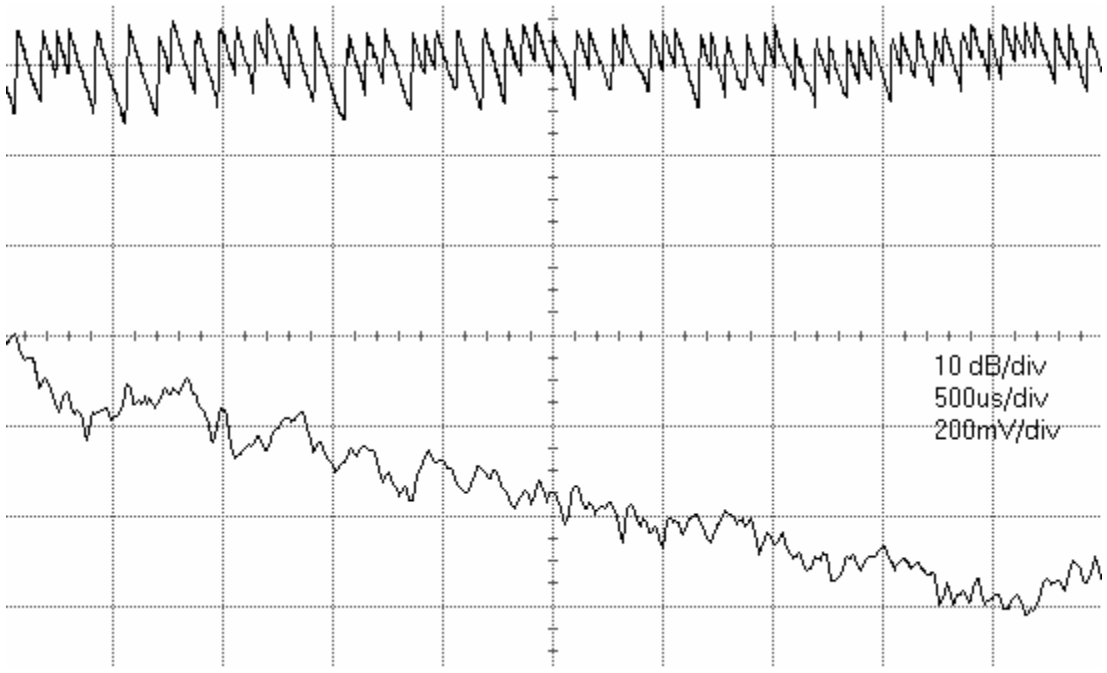
Figure 9.1: The complete spread-spectrum burst-mode LTC3458 circuit.

In burst mode, the LTC3458 produces a clear tone that is audible to the naked ear from 1 foot away. With a spread spectrum system, the sound is slightly quieter, but the noise becomes "swooshing" and white noise-like. Figures 9.1a and 9.1b highlight the output waveform differences between the vanilla LTC3458 and a spread-spectrum LTC3458. The FFT of the output waveforms indicates that the unmodified LTC3458 produces distinct harmonics. The modified LTC3458 can produce an output at the same voltage, with the same load, but with a much smoother FFT. The peak harmonic of the modified LTC3458 is also approximately 18dB less than that of the unmodified LTC3458, which explains quieter output.

The spread spectrum system in figure 9.2b shows a slight increase in ripple voltage when compared to that in figure 9.2a, even though an output voltage ceiling is imposed on the system by the time-domain noise control. On close inspection, the output voltage overshoots the ceiling whenever the downward drift portion of the output voltage is very short. The reason that the converter output always overshoots in this situation is that there exists a minimum burst length. When the spread spectrum system forces LTC3458 to prematurely burst, and if the output voltage has barely settled, it is inevitable that the output voltage will overshoot the imposed ceiling. The LTC3458 does not allow runt current bursts since runt pulses seem to lower efficiency. With this in mind, the modified LTC3458 actually always has worse maximum output peak-to-peak voltage ripple than a vanilla LTC3458, as shown in figure 9.3. The time-average output ripple of the modified LTC3458, however, is roughly 20% less than that of the unmodified LTC3458. Shown in figure 9.4 is the time-averaged RMS ripple voltage of LTC3458 when a spread spectrum system is added. The time-averaged RMS ripple voltage is calculated by taking the root-mean-squared difference of the output voltage from the average DC-value over 100ms at 5000 samples per second.



(a)



(b)

Figure 9.2: (a) Unmodified LTC3458 output with FFT running with 16mA load and 5V output; the upper waveform is the output voltage and the bottom waveform is its FFT
(b) Spread spectrum LTC3458 output with FFT with 16mA load and 5V output.; the upper waveform is the output voltage and the bottom waveform is its FFT

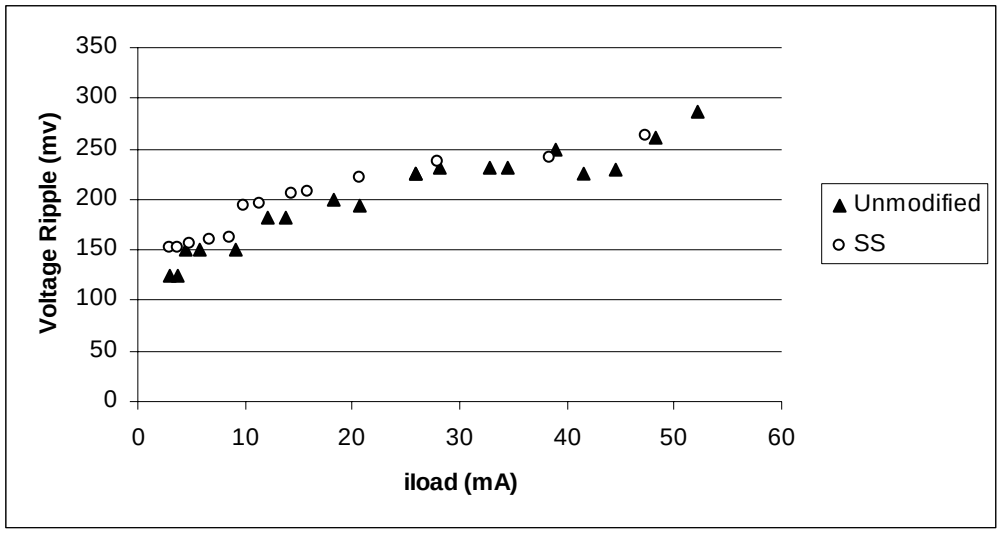


Figure 9.3: Maximum output voltage ripple of LTC3458 with 5V output.

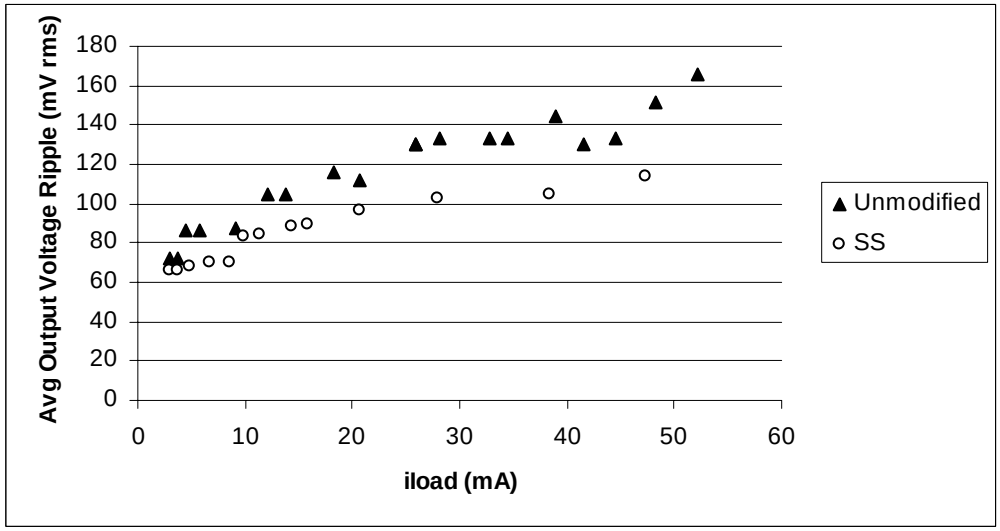


Figure 9.4: 500ms time-averaged output voltage ripple of LTC3458 with 5V output.

When coupled with a spread spectrum system, the output regulation of the LTC3458 actually improves. Figure 9.5 compares the regulation characteristics of a vanilla LTC3458 and a modified LTC3458. As the load current increases, LTC3458's regulation falls at an approximately linear rate (for the low power range). When spread spectrum is added, the regulation falls at a slightly slower rate since the spread spectrum output voltage always reaches the preset ceiling but rarely drops as low as the original floor.

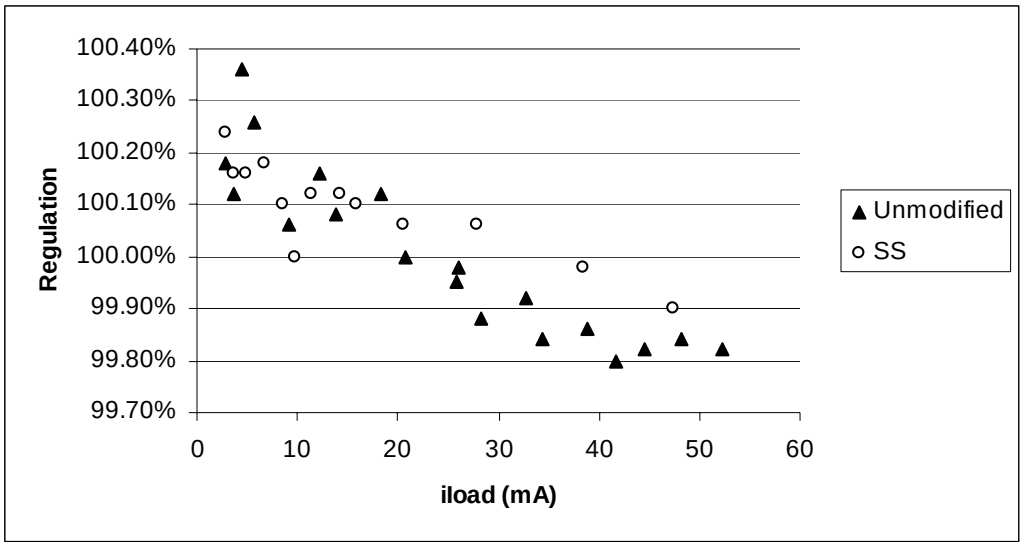


Figure 9.5: Output regulation of LTC3458 with 5V output; regulation is defined as Vout/5V.

Since the average burst rate for a spread-spectrum LTC3458 is higher, the efficiency of the spread spectrum LTC3458 consistently underperforms when compared to an unmodified LTC3458. Efficiency, defined as $V_{OUT}I_{LOAD}/V_{IN}I_{IN}$, is directly correlated with regulation (V_{OUT}/5V). The modified LTC3458 is more efficient than the original LTC3458, but figure 9.6 clearly shows the efficiency measurements of a vanilla LTC3458 against a spread spectrum LTC3458. The power measurements in the efficiency calculation were obtained with a voltmeter and an ammeter at the DC-input as well as at the DC-output of the LTC3458.

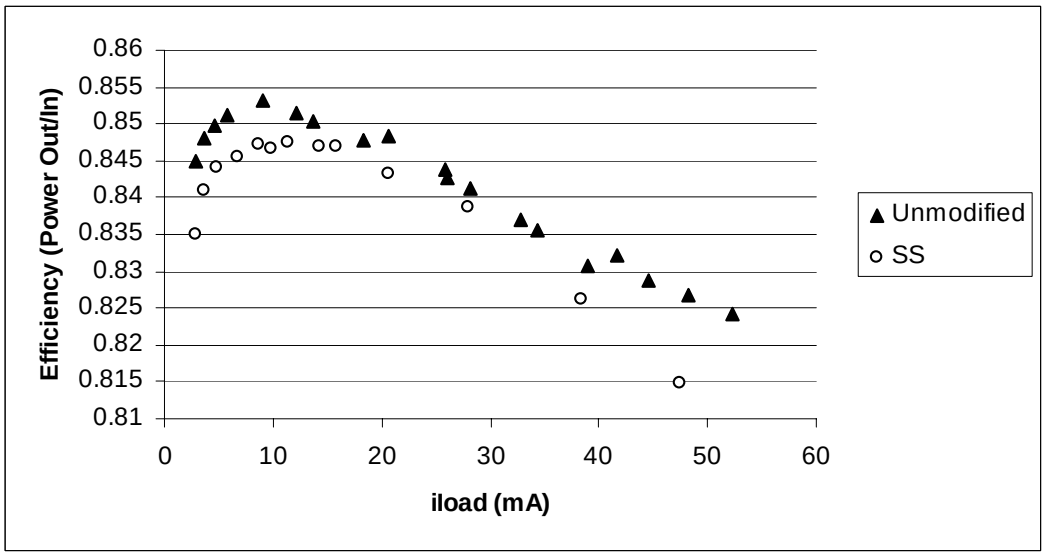


Figure 9.6: Efficiency of LTC3458 with 5V output.

The hidden variable is the time-average frequency. LTC3458's efficiency decreases with frequency. With a spread spectrum system, LTC3458's average burst mode frequency is higher for every value of load current. The increase in frequency and decrease in efficiency are results of imposing an output voltage ceiling and floor. If there were no ceiling and floor and pure white noise were injected into the converter burst mode control, then the mean burst frequency and efficiency would not change. Table 9-1 contains the data used to generate the above graphs.

iIn (mA)	Vin (V)	Win (mW)	iLoad	Vout	Wo (mW)	RpM (mV)	RpA (mV)	Effic	Reg
5.246	3.3	17.31	2.92	5.009	14.63	125	72	0.845	100.18%
6.636	3.3	21.90	3.71	5.006	18.57	125	72	0.848	100.12%
8.124	3.3	26.81	4.54	5.018	22.78	150	87	0.850	100.36%
10.296	3.3	33.98	5.77	5.013	28.93	150	87	0.851	100.26%
16.115	3.3	53.18	9.07	5.003	45.38	151	87	0.853	100.06%
21.621	3.3	71.35	12.13	5.008	60.75	181	105	0.851	100.16%
24.501	3.3	80.85	13.74	5.004	68.75	181	105	0.850	100.08%
32.64	3.3	107.71	18.24	5.006	91.31	200	115	0.848	100.12%
36.88	3.3	121.70	20.65	5	103.25	193.7	112	0.848	100.00%
46.44	3.3	153.25	25.88	4.9975	129.34	225	130	0.844	99.95%
46.715	3.3	154.16	25.99	4.999	129.92	225	130	0.843	99.98%
50.77	3.3	167.54	28.22	4.994	140.93	231.5	134	0.841	99.88%
59.21	3.3	195.39	32.74	4.996	163.57	231	133	0.837	99.92%
62.31	3.3	205.62	34.42	4.992	171.82	231	133	0.836	99.84%
70.84	3.3	233.77	38.9	4.993	194.23	250	144	0.831	99.86%
75.59	3.3	249.45	41.6	4.99	207.58	225	130	0.832	99.80%
81.4	3.3	268.62	44.6	4.991	222.60	230	133	0.829	99.82%
88.2	3.3	291.06	48.2	4.992	240.61	262	151	0.827	99.84%
95.8	3.3	316.14	52.2	4.991	260.53	287	166	0.824	99.82%
The following are measurements of the modified LTC3458									
5.275	3.3	17.41	2.9	5.012	14.53	153	66	0.835	100.24%
6.714	3.3	22.16	3.72	5.008	18.63	153	66	0.841	100.16%
8.775	3.3	28.96	4.88	5.008	24.44	157	68	0.844	100.16%
12.08	3.3	39.86	6.73	5.009	33.71	161	70	0.846	100.18%
15.376	3.3	50.74	8.59	5.005	42.99	163	71	0.847	100.10%
17.537	3.3	57.87	9.8	5	49.00	193	84	0.847	100.00%
20.367	3.3	67.21	11.38	5.006	56.97	195	84	0.848	100.12%
25.577	3.3	84.40	14.28	5.006	71.49	205	89	0.847	100.12%
28.436	3.3	93.84	15.88	5.005	79.48	207	90	0.847	100.10%
37.054	3.3	122.28	20.61	5.003	103.11	222	96	0.843	100.06%
50.43	3.3	166.42	27.9	5.003	139.58	237	103	0.839	100.06%
70.29	3.3	231.96	38.34	4.999	191.66	241	104	0.826	99.98%
88.06	3.3	290.60	47.4	4.995	236.76	263	114	0.815	99.90%

Table 9-1: Measurements of the LTC3458 with 3.3V input and 5V output in burst mode.

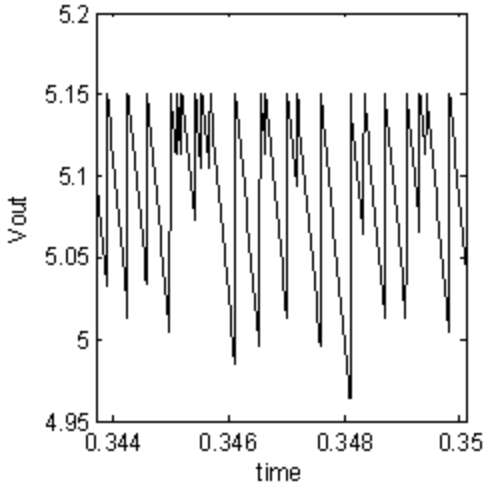
Chapter X: Conclusion

The spread spectrum modulation system for burst mode DC-DC converters effectively whitens the converter output frequency spectrum, decreases the average output voltage ripple, and faintly improves regulation at the tradeoff of slightly worse maximum ripple and approximately 0.5% lower efficiency. Although spread spectrum is unable to completely silence the burst mode audible noise, it transforms the burst mode audible tones into a slightly more pleasant pink noise and significantly lowers output frequency harmonics.

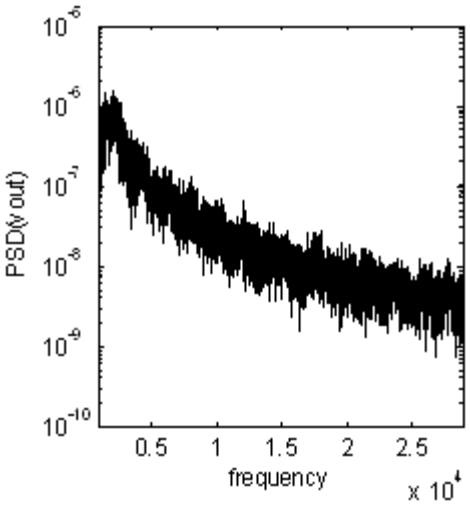
With a few minor caveats described in Chapter IX, the Chapter V Matlab model accurately represents this spread spectrum converter, as shown in the comparison of Figures 5.10 and 9.1 on the next page.

While there are many methods of whitening a DC-DC converter's output frequency spectrum including Avalanche noise amplification systems and glitch generators. The spread spectrum system described this thesis, using linear feedback shift registers, appears the simplest to implement. Furthermore, the linear feedback register system allows the flexibility of coupling a feedback control system that reshapes the converter output in the time domain.

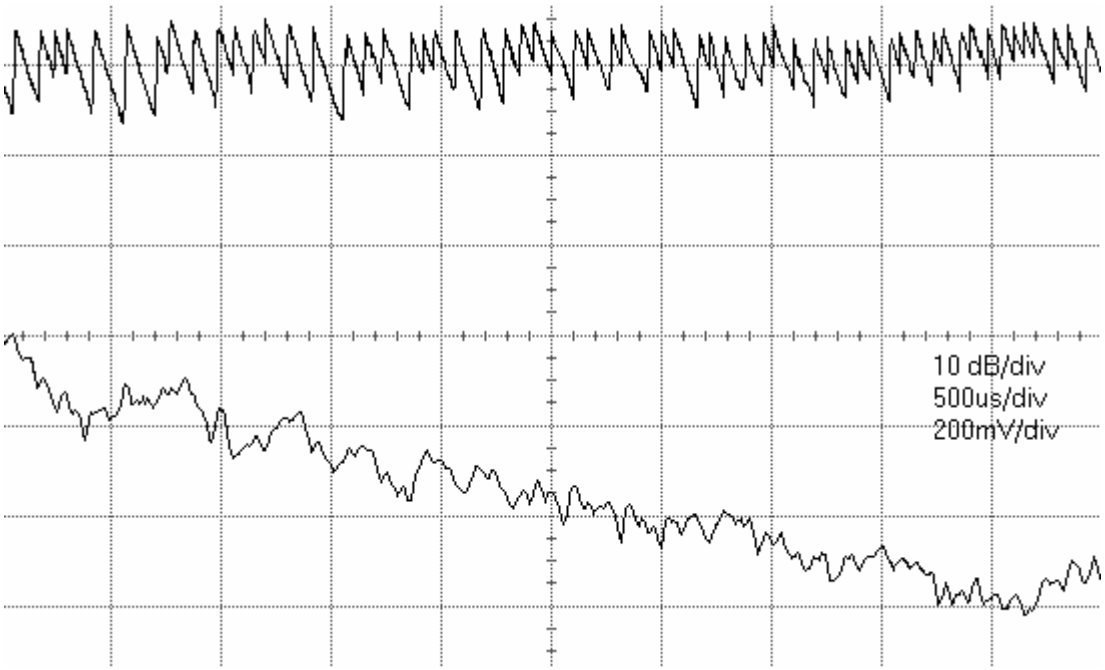
Although the components of this spread spectrum system, such as the DAC, LFSR, and time-domain controller, can be further optimized, this thesis successfully explores a systematic approach at implementing a burst mode spread spectrum system and effectively found its tradeoffs.



5.10a



5.10b



9.2b

Figures 5.10a, 5.10b, and 9.2b: Simulated and actual converter outputs.

Appendix A: Matlab Code

Fast-Fourier Transform and Data Plotter- fastfft.m

```
function [vFrequency, vAmplitude] = fastfft(vData, SampleRate, Plot)

%FASTFFT   Create useful data from an FFT operation.
%   Usage: [vFrequency, vAmplitude] = fastfft(vData, SampleRate,
[Plot])
%
%   (no plot will be shown if the last input == 0 or is not included)
%
%   This function inputs 'vData' as a vector (row or column),
%   'SampleRate' as a number (samples/sec), 'Plot' as anything,
%   and does the following:
%
%     1: Removes the DC offset of the data
%     2: Puts the data through a hanning window
%     3: Calculates the Fast Fourier Transform (FFT)
%     4: Calculates the amplitude from the FFT
%     5: Calculates the frequency scale
%     6: Optionally creates a Bode plot
%
%   Created 7/22/03, Rick Auch, mekaneck@campbellsville.com
%   Modified 8/21/05, Ji Zhang, aceji@mit.edu


%Make vData a row vector
if size(vData,2)==1
    vData = vData';
end

%Calculate number of data points in data
n = length(vData);

%Remove DC Offset
vData = vData - mean(vData);

%Put data through hanning window using hanning subfunction
vData = hanningt(vData);

%Calculate FFT
vData = fft(vData);

%Calculate amplitude from FFT (multply by sqrt(8/3) because of effects
of hanning window)
vAmplitude = abs(vData)*sqrt(8/3);

%Calculate frequency scale
vFrequency = linspace(0,n-1,n)*(SampleRate/n);

%Limit both output vectors due to Nyquist criterion
DataLimit = ceil(n/2);
```

```
vAmplitude = vAmplitude(1:DataLimit);
vFrequency = vFrequency(1:DataLimit);

if exist('Plot', 'var')==1 & Plot~=0
    plot(vFrequency, vAmplitude);
    title('Bode Plot');
    xlabel('Frequency (Hz)');
    ylabel('Amplitude');
end


%-------------------------------------------------------------------------
--------------------
%Hanning Subfunction
function vOutput = hanningt(vInput)
% This function takes a vector input and outputs the same vector,
% multiplied by the hanning window function

%Determine the number of input data points
n = length(vInput);

%Initialize the vector
vHanningFunc = linspace(0,n-1,n);

%Calculate the hanning funtion
vHanningFunc = .5*(1-cos(2*pi*vHanningFunc/(n-1)));

%Output the result
vOutput = vInput.*vHanningFunc;
%temp = hanning(n);

%vOutput = vInput'.*temp;
```

Optimal Spread Spectrum Ratio & Frequency Finder- findoptimalloops.m

```
%%%
% Ji Zhang
% 2005-08-02 last ed.
% Sawtooth Waveform Generator: Optimal results Searcher
% DC-DC Burst Mode Emulation Part 1a.

format short g;

trials = 50;
optimaltable = zeros(10,3);
% 1st column = burst freq
% 2nd column = spread freq
% 3rd column = PSD(burst_w_spread)/PSD(burst_no_spread)

for burst_freq_mult = 1:trials  % high value for overnight

    sprintf ('** %d out of %d mainloops **', burst_freq_mult,trials)

    burst_freq = burst_freq_mult*200;


    results = loopwaveformgenerate(burst_freq);
    % search for the best results
%    temp = results(:,4);
%    results(:,4) = results(:,1);
%    results(:,1) = temp;     % easier for sorting

    [minvalues, minindices]=min(results);


    %%%% finding best ss frequency %%%%
    best_ss_index = minindices(4);
    ss_freq = results(best_ss_index,2);
    optimaltable(burst_freq_mult,2) = ss_freq;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%% calculating PSD reduction %%%%
%    originalPSD=results(1,4);
%    newPSD=minvalues(4);
%    PSDratio = newPSD/originalPSD;
    PSDratio = results(best_ss_index,4);
    optimaltable(burst_freq_mult,3) = PSDratio;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    %%%% recording burst frequency %%%%
    trueburstfreq = results(best_ss_index,1);
    optimaltable(burst_freq_mult,1) = trueburstfreq;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


end %for
```

Linear Feedback Shift Register Simulation- linearshiftregsim.m

```
%%%
% Ji Zhang
% 2005-08-11 last ed.
% Linear Shift Register Simulation
% This is the pseudorandom code simulation
% DC-DC Burst Mode Emulation Part 2a.

format short g;

sizereg = 7;      % size of the register being tested
trials = 10000;
Fs = 65536; % 2^17
Amplitude = 1;
tlen = 1;   % length of time
dc_bias = 0;
t = [0:Fs*tlen-1]/Fs;   % generating the time axis
output=zeros(Fs,1);

register = zeros(sizereg,1);
register(1) = 1;  %0000 is the lockup state
% position 1 is the bit being pushed out by convention

for stepnumber = 1: 3 :Fs-3

    output(stepnumber) = register(1);     %save the output
    output(stepnumber+1) = register(1);   %save the output
    output(stepnumber+2) = register(1);   %save the output
    output(stepnumber+3) = register(1);   %save the output

    % the 4-bit case
%    input = xor(xor(register(1),register(3)), register(4));
    % input is a function of xor additions

    % the 7-bit case
    input = xor(register(1),register(7));

    for movebit = 1:(sizereg-1)
      register(movebit) = register(movebit+1);
    end %for
    register(sizereg) = input;

end % for stepnumber

%hanning window & scaling
output = output-mean(output);
output= output.*hanning(length(output));

freq = [0:Fs-1];
%plot(t,output);

pwelch(output,[],[],[],Fs);

%temp = abs(fft(output));
%plot(freq(1:Fs/2), temp(1:Fs/2));  % the other half is nonsense
```

Linear Feedback Shift Register Simulation- linearshiftregsim.m

```
% Ji Zhang
% 2005-08-02 last ed.
% Sawtooth Waveform Generator: Results Collector
% DC-DC Burst Mode Emulation Part 2a.

% Given a burst_frequency, this function finds the
% best band-limited random noise center frequency

function results = loopwaveformgenerate(burst_freq)

format short g;

trials = 50;            % sets the number of trials
granularity = 200;      % hz
startfreq = 0;          % default; cannot be changed
% 1st column of results = burst frequency
% 2nd column of results = noise center frequency
% 3rd column of results = new dominant frequency
% 4th column of results = overall suppression
% 5th column of results = suppression of old frequency
% 5th column is not implemented yet

%%% careful about setting upper & lower bounds

[f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveformgenera
te(.00001,burst_freq);

results = zeros(trials+1,4);
results(:,1) = maxVOUT_Cfq;
results(1,2) = 0;       %as if no interference signal
results(1,3) = maxVOUTfq;
results(1,4) = maxVOUT/maxVOUT_C;

for j = 1 : trials

    sprintf ('%d out of %d sub_1_loops', j, trials)

    if j~=0
      [f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveform
generate(j*granularity,burst_freq);
    else
      [f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveform
generate(j*granularity,burst_freq);
    end %if

    results(j+1,2) = maxVOUT_ssfq;
    results(j+1,3) = maxVOUTfq;
    results(j+1,4) = maxVOUT/maxVOUT_C;

end %for
```

Converter Output Simulation Automation Add-on- loopwaveformgenerate.m

```
% Ji Zhang
% 2005-08-02 last ed.
% Sawtooth Waveform Generator: Results Collector
% DC-DC Burst Mode Emulation Part 2a.

% Given a burst_frequency, this function finds the
% best band-limited random noise center frequency

function results = loopwaveformgenerate(burst_freq)

format short g;

trials = 50;            % sets the number of trials
granularity = 200;      % hz
startfreq = 0;          % default; cannot be changed
% 1st column of results = burst frequency
% 2nd column of results = noise center frequency
% 3rd column of results = new dominant frequency
% 4th column of results = overall suppression
% 5th column of results = suppression of old frequency
% 5th column is not implemented yet

%%% careful about setting upper & lower bounds

[f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveformgenera
te(.00001,burst_freq);

results = zeros(trials+1,4);
results(:,1) = maxVOUT_Cfq;
results(1,2) = 0;       %as if no interference signal
results(1,3) = maxVOUTfq;
results(1,4) = maxVOUT/maxVOUT_C;

for j = 1 : trials

    sprintf ('%d out of %d sub_1_loops', j, trials)

    if j~=0
      [f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveform
generate(j*granularity,burst_freq);
    else
      [f,maxVOUTfq,maxVOUT,maxVOUT_C,maxVOUT_Cfq,maxVOUT_ssfq]=waveform
generate(j*granularity,burst_freq);
    end %if

    results(j+1,2) = maxVOUT_ssfq;
    results(j+1,3) = maxVOUTfq;
    results(j+1,4) = maxVOUT/maxVOUT_C;

end %for
```

Converter Output Simulation Core- waveformgenerate.m

```
% Ji Zhang
% 2005-10-03 last ed.
% Sawtooth Waveform Generator & Spectrum Generator
% DC-DC Burst Mode Emulation Part 1a.

%function [Fs, maxVOUTfq, maxVOUT, maxVOUT_C, maxVOUT_Cfq,
maxVOUT_ssfq] = waveformgenerate(sswavefreq, burst_freq)

% tic;      % start timer

% *note sswavefreq & burst_freq are not accurate measures;
% there exist granularity errors at higher frequencies
% take the actually generated frequency via fft rather than the input
freq!


%clear all
Fs = 131072; %2^17; a power of 2 makes discrete fft & PSD easier
           %the higher this number, the finer the granularity
           %Fs needs to be higher than 2^17 to have accuracy up to
20kHz
A=.3;       %Amplitude
f0=1000;    %Frequency [Hz]
tlen=1;     %Length [s];      needs to be 1 or else dimensions are
messed up
freq=[0:Fs-1];

%%%%%% Waveform properties

dc_bias = 5.2;
burst_threshold = .3;
%load = 10000;    % the less the load the faster the sawtooth
oscillator_multiplier = .3; % how much the square wave is multiplied
spread_ratio= .3;


load = Fs*(1/burst_freq)*100/burst_threshold;

%burst_freq = (burst_threshold/100)*load*(1/Fs);
% this is the default burst frequency
% threshold = (100/load)*x, where x is in units of time



t=[0:Fs*tlen-1]/Fs; %Time axis
%xt=A*(mod(2*pi*f0*t+(pi/180)*phi,2*pi)/pi-1);   %Sawtooth signal x(t)

%**************************************
%fft frequency-axis calibration
%generate the fft of a known function
%and graph the fft; then use the same fft
%procedures to graph other functions
%sinewave = sin(2*pi*5000*t); %freq = 5k
```

```
%xt=A*(mod(f0*t,1)/pi);   %Sawtooth signal x(t)
%xt=xt*-1;
%xt=xt+dc_bias;

% Spread Spectrum

%generate square wave
%K=.00004;
%sqt=A*.3*(mod(f0*t,K)<K/2);
%sqt=sqt-A*.3/2;

sswave=zeros(1,Fs);
%sswave=sqt;
sswaveused=sswave;
sswave(1) = 0;
%sqt=generic square wave
%xt=generic sawtooth wave
%sswave=spread spectrum wave

%spread the square wave here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%adding the two signals
%vout=xt;
vout=zeros(1,Fs);
%vout(1)=dc_bias;
vout(1) = 5.1;
vout_clean=vout;
size_xt=size(vout);
%size_xt=size(xt);

% initial loop conditions
last_jump = 1;
next_jump = 1;
just_burst = 0;

for i = 2 : size_xt(2)

    vout(i) = vout(i-1)-25/load;    % vout's downward drift

    % for reference
    vout_clean(i)=vout_clean(i-1)-25/load;

    comparator_in_clean = dc_bias - burst_threshold;

    if (vout_clean(i-1)/4.2 < 1.2)
      vout_clean(i) = vout_clean(i) + .1;
    end %if
%    if vout_clean(i-1) <= comparator_in_clean % error checker
%        vout_clean(i) = min(vout(i-1)+1,dc_bias);        % burst % the
burst should not be constant
%    end %if


    %%%%%%%%%%% 16-bit DAC model %%%%%%%%%%%
```

```
    %%%% AC-coupled sswave %%%%%%%
    sswave(i) = sswave(i-1);  %sswave doesn't change until burst
%{
    if (vout(i-1)/4.2 + sswave(i) < 1.2)
      vout(i) = vout(i) + .1; % burst up by 50mV
%        sswave(i) = (ceil(16*rand)-8.5)/220;
        sswave(i) = (ceil(16*rand)-8.5)/425;
    end %if
%}

    %%%% Ceiled AC-coupled sswave %%%%%%
    sswave(i) = sswave(i-1);
    if (vout(i-1)/4.2 + sswave(i) < 1.2)

      vout(i) = 5.15;   % burst up by 50mV
      sswave(i) = (ceil(16*rand)-8.5)/425;
    end %if


%% $$ here
%{


    %generate the spread spectrum square wave

    sswave(i)=sswave(i-1);
    if next_jump<=i;                    % if we reach a transition
point

        sswave(i)=-1*sswave(i-1);       % flip step the sswave
        sswave(i-1)=-1*sswave(i-1);       % flip step the sswave

      temp=rand;
      if temp<.5
          temp=-1;
      else
          temp=1;
      end %if

        next_jump = last_jump + .5*Fs/sswavefreq +
.5*temp*Fs/sswavefreq*(rand*spread_ratio);

        last_jump = next_jump;
    end %if

    % one problem is that if the interference signal increases the
feedback
    % pin voltage, then vout drops too low and makes a huge jump next
time
    % this increases the output ripple significantly (in testing)

    % killing positive portions of the input signal
    if (sswave(i) > 0)
      sswaveused(i) = 0;
    else
      sswaveused(i) = sswave(i)*2;  % since we truncated 1/2 signal
```

```
    end %if

%    sswaveused(i) = sswave(i);     % default w/o ripple reduction

    comparator_in = dc_bias - burst_threshold -
sswaveused(i)*oscillator_multiplier;


% this is wrong; we should be checking comparator_in, not vout
% since a small signal applied to comparator_in may not reflect at vout

    if vout(i-1) <= comparator_in        % error checker
        vout(i) = vout(i-1)+.10;        % burst % the burst should not
be constant
%        vout(i) = min(vout(i-1)+1,dc_bias);        % burst % the burst
should not be constant

    end %if

%}




end %for



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% outputting results
pwfreq = 1/(t(2)-t(1));       %the frequency axis for pwelch
                         %also the sampling frequency Fs
sswaveused = sswave;

%subplot(3,3,7);
[tempy_ss,tempx_ss] = pwelch(sswave,[],[],[],pwfreq);
tempy_ss(1:66) = 0;
tempy_ssaudio = tempy_ss;
[maxVOUT_ss,maxVOUT_ssfq]=max(tempy_ssaudio);

% maxVOUT_Cfq is just an index; we have to scale it
maxVOUT_ssfq=maxVOUT_ssfq*(tempx_ss(2)-tempx_ss(1));

%subplot(3,3,8);
[tempy_voutc,tempx_voutc] = pwelch(vout_clean,[],[],[],pwfreq);
tempy_voutc(1:66) = 0;
tempy_voutcaudio = tempy_voutc;
[maxVOUT_C,maxVOUT_Cfq]=max(tempy_voutcaudio);

% maxVOUT_Cfq is just an index; we have to scale it
maxVOUT_Cfq=maxVOUT_Cfq*(tempx_voutc(2)-tempx_voutc(1));


%subplot(3,3,9);
[tempy_vout,tempx_vout] = pwelch(vout,[],[],[],pwfreq);
tempy_vout(1:66) = 0;
tempy_voutaudio = tempy_vout;
```

```
[maxVOUT,maxVOUTfq]=max(tempy_voutaudio);
% maxVOUT_fq is just an index; we have to scale it
maxVOUTfq=maxVOUTfq*(tempx_vout(2)-tempx_vout(1));


r_freq = maxVOUTfq;
r_mag = maxVOUT;


%disp('press key to continue');
%pause;
sprintf('new peak PSD/old peak PSD = %d', maxVOUT/maxVOUT_C)
sprintf('peak OLD PSD freq = %d', maxVOUT_Cfq)
sprintf('peak NEW PSD freq = %d', maxVOUTfq)


%%%%%% OPTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Keeping the different capabilities of this simulator modular
%% is important in speeding up the simulator when a particular
%% component is not needed
%% Uncomment the function/option to enable the option
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%
%% GRAPHER
%% turn this off if running Monte Carlo since this is too slow!
waveformgrapher(freq, pwfreq, t, sswave, sswaveused,
vout_clean,vout,tempy_voutc, tempx_voutc, tempy_vout, tempx_vout,
tempy_ss, tempx_ss);
%% waveformgrapher(freq, pwfreq, t, sswave, sswaveused,
vout_clean,vout,tempy_voutc, tempx_voutc, tempy_vout, tempx_vout,
tempy_ss, tempx_ss);

%%%%%%%%
%% AUDIO
% waveformsounder(Fs,sswave,vout_clean,vout);
%% waveformsounder(Fs,sswave,vout_clean,vout);

%%%%%%%%
%% TIMER; compares processing time if any of the options are enabled
%toc
```

Converter Output Simulation Grapher Add-on- waveformgrapher.m

```
% Ji Zhang
% 2005-08-01 last ed.
% Sawtooth Waveform Generator & Spectrum Grapher
%  - Generates 9 subplots for waveformgenerate()
% DC-DC Burst Mode Emulation Part 1b.

function void =waveformgrapher(freq, pwfreq,
t,sswave,sswaveused,vout_clean,vout,tempy_voutc, tempx_voutc,
tempy_vout, tempx_vout, tempy_ss, tempx_ss)

% outputting results
pwfreq = 1/(t(2)-t(1));       %the frequency axis for pwelch
                        %also the sampling frequency Fs

ixt=find(t<=.008);            %Subset of time axis

subplot(3,3,1);
plot(t(ixt),sswaveused(ixt),'-b');

subplot(3,3,2);
plot(t(ixt),vout_clean(ixt),'-g');

subplot(3,3,3);
plot(t(ixt),vout(ixt),'-r');

subplot(3,3,4);
temp=abs(fft(sswave));
temp(1) = 0;
plot(freq,temp);  %fft looks better on linear scale
%semilogy(freq,temp);
title('fft(sswave)');

subplot(3,3,5);
temp=abs(fft(vout_clean));
temp(1) = 0;
plot(freq,temp,'-g');
%semilogy(freq,temp,'-g');

title('fft(vout_clean)');
%[maxvout_c,maxvout_cfq]=max(temp)

subplot(3,3,6);
temp=abs(fft(vout));
temp(1)=0;  % for some reason fft screws up the first fft
semilogy(freq,temp,'-r');
title('fft(vout)');


subplot(3,3,7);
plot(tempx_ss,tempy_ss,'-b');
%semilogy(tempx,tempy,'-b');
title('PSD(sswave)');
```

```
subplot(3,3,8);
plot(tempx_voutc,tempy_voutc,'-g');
%semilogy(tempx_voutc,tempy_voutc,'-g');
title('PSD(vout_clean)');


subplot(3,3,9);
%temp=fft(vout);
%temp=temp.*conj(temp);
%temp(1)=0;
%semilogy(freq,temp,'-r');
plot(tempx_vout,tempy_vout,'-r');
%semilogy(tempx,tempy,'-r');
%temp = pwelch(vout,[],[],[],pwfreq);
%temp(1) = temp(2); % hack
%size_temp = size(temp);
%size_temp = size_temp(1);
%xaxis = 1:size_temp;
%xaxis = xaxis*pwfreq/size_temp;
%semilogy(xaxis,temp,'-r');

%title('PSD(vout)');

%r_freq = maxVOUTfq;
%r_mag = maxVOUT;

%%%%%%%% manual testing %%%%
%close; % close the figure

%plot(t(ixt),sswaveused(ixt),'-k');
```

Converter Output Simulation Sound Add-on- waveformsounder.m

```
% Ji Zhang
% 2005-07-29 last ed.
% Sawtooth Waveform Generator & Spectrum Sounder
%  - Generates audio for waveformgenerate()
% DC-DC Burst Mode Emulation Part 1c.

function void = waveformsounder(Fs,sswave,vout_clean,vout)

sound(sswave-mean(sswave),Fs);
disp('press key to continue; sound vout clean');
pause;
sound(vout_clean-mean(vout_clean),Fs);
disp('press key to continue; sound vout real');
pause;
sound(vout-mean(vout),Fs);
disp('*** done ***');
```

Appendix B: Microcontroller Code

```
; Sawtooth generator
; 2005.08.24 last ed.
; 2005.08.22 star
; Ji Zhang aceji@mit.edu
; PIC16f876a
; this is essentially a PWM duty ratio cycler

      list p=16f876a
      ; Include file, change directory if needed
    include "p16f876a.inc"

      __config _HS_OSC & _WDT_OFF & _BODEN_OFF & _LVP_OFF & _PWRTE_OFF
& _DEBUG_OFF & _CP_OFF & _WRT_OFF


counterx equ 0x30
countery equ 0x31
countleftx equ 0x32
countlefty equ 0x33

delaytempA equ 0x3A
delaytempB equ 0x3B
delaytempC equ 0x3C

d1 equ      0x40
d2 equ      0x41
d3 equ      0x42
tmp000      equ 0x43

w_temp equ 0x70
status_temp equ 0x71


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;start

      org 0x0000
      goto SETUP
;     goto Interrupt_Handler

      org 0x0004        ; interrupt codes start at 0x0004
Interrupt_Handler
      ;     There is no interrupt



      org 0x0010  ; jump to second page to start code if needed
;     This is essentially the start of the non-interrupt portion
SETUP

;     SET THE PWM PERIOD
;     pwm period = [PR2 + 1]*4*Tosc*TMR2 Prescale value
```

```
;     try 4 for TMR2 prescale; other values work, too
;     Tosc = 1/4000000 = 2.5e-7
;     period/4 since we are using 16MHz clock instead of default 4MHz
;     PR2 = (period*4) / (4 * Tosc * TMR2 Prescale) -1
;     PR2 = (period*4) / (4 * 2.5*10^-7 * 4) -1
;     freq = 500KHz --> PR2 = (4/500000)/( 4*2.5*10^-7*4) -1 = 1
;     at 500KHz there is only 8-step resolution from 0-100% duty cycle
;     so let's try 250KHz for a 16-step resolution
;     PRS = (4/250000)/(4*10^-6) = 3

      bsf STATUS, RP0
      bcf STATUS, RP1
;     movlw d'3'  ; when TMR=PR2-->end of period
      movlw d'124'      ; for a frequency of 8kHz
      movwf PR2
      bcf STATUS, RP0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;     then change the duty cycle to shape the waveform
;     (4*time_on) since we are using 16MHz clock instead of 4MHz
;     duty cycle = (4*time_on) / (Tosc * TMR2 Prescale)
;              = (4*time_on) / ( 2.5 * 10^-7 * 4 )
;                (4*time_on) = CCPR1L:CCP1CON<5:4>
;
;     SET THE PWM initial duty cycle by writing to CCPR1L and
CCP1CON<5:4> BITS
;     CCPR1L = upper 8 bits of the 10 bit duty cycle
;     CCP1CON<5:4> = lower 2 bits
;     Duty cycle = (CCPR1L:CCP1CON<5:4>) * Tosc*TMR2 Prescale
;     CCPR1L:CCP1CON<5:4> = (4/1000000) / (2.5e-7 * 4) = 4 = 0b100
;     freq of on = 1/(bits*10e-6 / 4)
;     the min freq of on = 4MHz, and 4MHz/PWMfreq = resolution
;     at PWM of 500KHz, we have 1/8 resolution<F9>
;     at 500KHz there is only 8-step resolution from 0-100% duty cycle
;     so let's try 250KHz PWM for a 16-step resolution
;     -> duty cycle ranges from 250KHz (100% duty cycle) to (4MHz min
duty cycle)
      movlw b'00000001'
      movwf CCPR1L      ;set bits 9-2
      bcf CCP1CON,CCP1X ;set bit 1
      bcf CCP1CON,CCP1Y ;set bit 0


;     MAKE CCP1 pin into an output
      bsf STATUS,RP0
      movlw b'11110011' ;C3 used for debugging
      ANDWF TRISC
      bcf STATUS,RP0

;     Set TMR2 prescale and enable TMR2 by writing to T2CON
;     TMR2 prescale are T2CON<1:0>
;     TMR2 enable is bit 2
      movlw b'00000101' ;TMR2=on, prescale = 1:4
      movwf T2CON

;     Configure the CCP1 module for PWM
```

```
      movf CCP1CON,W
      andlw b'00110000' ;mask everything except previously set Duty
Cycle bits
      iorlw b'00001111' ;Enable PWM
      movwf CCP1CON       ;1000Hz square wave with 50% duty cycle out
of RC2



      goto  loop


      loop        ;commenting this disables the PWM cycling;
      bcf   STATUS, C

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;     SET THE PWM PERIOD
;     pwm period = [PR2 + 1]*4*Tosc*TMR2 Prescale value
;     try 4 for TMR2 prescale; other values work, too
;     Tosc = 1/4000000 = 2.5e-7
;     period/4 since we are using 16MHz clock instead of default 4MHz
;     PR2 = (period*4) / (4 * Tosc * TMR2 Prescale) -1
;     PR2 = (period*4) / (4 * 2.5*10^-7 * 4) -1
;     freq = 500KHz --> PR2 = (4/500000)/( 4*2.5*10^-7*4) -1 = 1
;     at 500KHz there is only 8-step resolution from 0-100% duty cycle
;     so let's try 250KHz for a 16-step resolution
;     PRS = (4/250000)/(4*10^-6) = 3

      bsf STATUS, RP0
      bcf STATUS, RP1
      movlw d'3'  ; when TMR=PR2-->end of period
;     movlw d'124'      ; for a frequency of 8kHz
      movwf PR2
      bcf STATUS, RP0

      movlw b'00111110'
      movwf       CCPR1L      ;set bits 9-2
      bsf   CCP1CON,CCP1X ;set bit 1
      bcf   CCP1CON,CCP1Y ;set bit 0
      call  Delaysetup  ;reset delaytempA
      call  Delay

;     goto  loop  ; commenting this gets rid of the PWM cycling

;     testing

;     cycle the duty cyle
;     charge to full duty cycle
;0
      movlw b'00000100'
      movwf       CCPR1L      ;set bits 9-2
      bcf   CCP1CON,CCP1X ;set bit 1
      bcf   CCP1CON,CCP1Y ;set bit 0
```

```
      call  Delaysetup  ;reset delaytempA
      call  Delay

;-1
      movlw       b'00000011'
      movwf CCPR1L
      bsf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call  Delay

;-2
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay

;-3
      bcf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay

;-4
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-5
      movlw       b'00000010'
      movwf CCPR1L
      bsf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-6
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-7
      bcf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-8
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-9
      movlw       b'00000001'
      movwf CCPR1L
```

```
      bsf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-10
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-11
      bcf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-12
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-13
      movlw       b'00000000'
      movwf CCPR1L
      bsf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-14
      bcf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay
;-15
      bcf   CCP1CON,CCP1X ;set bit 1
      bsf   CCP1CON,CCP1Y ;set bit 0

      call  Delaysetup  ;reset delaytempA
      call Delay

;reached the lowest voltage; have to reset now!


;     call  SetupADCz
;     call  ReadADCz

      goto  loop

Delaysetup

;     (delay_time+1)*resolution = period of sawtooth
;     +1 since we need to burst back up the voltage in one cycle7
;     0x1A = delay for 2e-5 seconds
;     0x0D = delay for 1e-5 seconds
;     0x06 = delay for 5e-6 seconds
```

```
      movlw 0x08
      movwf delaytempA
      return

Delay
      decfsz      delaytempA, f
      goto  Delay ; loop
      return            ; else return



SetupADCz
        bsf     STATUS,RP0      ;bank 1
        bcf     STATUS,RP1
 ;       movlw   H'00'
;        movwf   TRISC           ;portc [7-0] outputs
        clrf    ADCON1          ;left justified, all inputs a/d
        bcf     STATUS,RP0      ;bank 0
        movlw   B'01000001'     ;Fosc/8 [7-6], A/D ch0 [5-3], a/d on
[0]
                                          ;default inchannel is a0
        movwf   ADCON0
            return

ReadADCz
                                ;wait for acquision time (20uS)
                                ;(non-critical for this test)

        bsf     ADCON0,GO       ;Start A/D conversion
Waitz
        btfsc   ADCON0,GO       ;Wait for conversion to complete
        goto    Waitz

        movf    ADRESH,W        ;Write A/D result to PORTC
;        movwf   PORTC           ;LEDs
        movwf     tmp000                ; MSB output in tmp0.
(highest 8 bits)
        return

      end
```

References

[1] R. Nave, Georgia State University HyperPhysics 2005, "Sound Intensity," http://hyperphysics.phy-astr.gsu.edu/hbase/sound/intens.html

[2] A. Stankovic, G. Verghese, D. Perreault, "Analysis and Synthesis of Randomized Modulation Schemes for Power Converters," IEEE Transactions on Power Electronics, Volume 10, Issue 6, November 1995, pp. 680-693.

[3] D. Bell, "Micropower DC-DC Converters Promote Battery-Life in Handhelds." http://www.commsdesign.com/showArticle.jhtml?articleID=16505372, July 9, 2002.

[4] P. Clark, "Self-Commutated Thyristor DC-to-DC Converter," IEEE Transactions on Magnetics, Volume 6, Issue 1, March 1970, pp. 10-15.

[5] T. Habetler and D. Divan, "Acoustic Noise Reduction in Sinusoidal PWM Drives Using a Randomly Modulated Carrier," IEEE Transactions on Power Electronics, Volume 6, July 1991, pp. 356-364.

[6] T. Tanaka, T. Ninomiya, and K. Harada, "Random-Switching Control in DC-DC Converters," IEEE PESC Proc, 1989, pp. 500-507.

[7] T. Tanaka and T. Ninomiya, "Random-Switching Control for DC-DC Converters: Analysis of Noise Spectrum," Power Electronics Specialists Conference, 1992. PESC '92 Record, 23rd Annual IEEE 29 June-3 July 1992, pp. 579-586 vol.1.

[8] J. Bourgeois, "PWM Technique for Acoustic Noise Reduction in Power Applications," Applied Power Electronics Conference and Exposition, March 7-11 1993, pp. 141-145.

[9] A. Stankovic, G. Verghese, D., Perreault, "Markov Chain Controlled Random Modulation of Switching Signals in Power Converters," United States Patent 5510698, April 23, 1996.

[10] A. Stankovic and H. Lev-Ari, "Randomized Modulation in Power Electronic Converters," Proceedings of the IEEE Volume 90, Issue 5, May 2002, pp. 782–799.

[11] B. Walter, Conversations on Linear Feedback Shift Registers, Chelmsford, MA: LTC Design Engineer, 2005

[12] W. Cherowitzo, "Linear Feedback Shift Registers," University of Colorado at Denver Math 4410: Coding Theory and Cryptology, http://www-math.cudenver.edu/~wcherowi/courses/m5410/m5410fsr.html.

[13] P. Gray, P. Hurst, S. Lewis, and R. Meyer, Analysis and Design of Analog Integrated Circuits. New York: Wiley, 2001.

[14] S. Hageman, Frequency Dithering Enhances High-Performance ADCs. EDN, August 18, 2005, pp. 84-86.

[15] P. Horowitz and W. Hill, Art of Electronics, The, Cambridge University Press: 1980, Sec. 9-16.

[16] K. Lundberg, Becoming One with the Transistor. Unpublished, 2005.