Adaptive Spread Spectrum Clock Generator using Delta-Sigma Digital-to-Time Converter

Ramin Khatami

February 26, 2015

Dedication

Here, i would like to first thank my mother Fatemeh Alvandi. Without her support and special care and ultra effort she put for our education and well being, i would not be the person i am today.

Second, i would like to thank my father Mohammad. He passed away before i start my journey as an engineer but his memories are always with me and i have always felt his presence around me.

I also like to thank my brother Ali. From very early days in my childhood up to this day; he has been a true big brother and has always motivated me to push myself hard to move one step forward and advance in whatever task i am doing; in his own special way.

I also like to give a very special thank to Mr. Ishikawa; our laboratory technician who has always guided me during my academic life at Gunma University and kindly walked me though many hurdles in past 3 years. His carefulness advises in these years helped me to not get lost in the csharp curves and fall into the deep valley of ignorance.

Finally and most importantly, i would like to thank my professor and supervisor: Professor Haruo Kobayashi, who is the most admiring person i have seen. It is not exaggeration if i say he has been the most important and influential person on me so far and comparable to the role of my father. I have always thought of him as a role model and felt the closest feeling a student can feel toward his/her professor. Being in his laboratory and having the chance to take his lessons and advice were probably the greatest opportunity i have ever acquired in my life up to this point. Without him and without his (fatherly) extra care about my well being in all ways, i would not be the person i am today and without his personal care and deep knowledge of circuit this research would not be instantiated and as result-full, as it is today.

Dear Professor Kobayashi, your hard working and kindness are things i will hold dear through rest of my life.

Contents

Preface

This research has studied and proposed a novel Converter method for use in spread-spectrum generation. Thanks to this method, one can select and exclude certain spectrum bands from spreading spectrum.Proposed method in this research is utilizable in advanced form of spread spectrum clock generation and also applicable to power circuits switching electromagnetic interference reduction/customization.

Built upon proposed algorithms and introduced convictions in this research; an auto-configurable EMI exclusion method has also been introduced.This method has capability of adaptively excluding certain band from spreading spectrum in regards to detectable environmental signal functioning frequencies. This innovation removes the need for circuit designer to choose certain bands for exclusion; as it is done automatically and open their hands to focus on their design only and not spending too much time considering their circuits EMI effects on other surrounding circuit components.

The converter method is a unique algorithms to convert a Digital signal to Analog in time domain called Digital-to-Time converter (abr. DTC). As we show later DTC circuits proposed here are best fit to solve our problem and any problem where time domain is concerned.

This method is essentially fast (reachable at high clock frequency) and flexible (programmable) by taking full advantages of digital circuit simplicity.

Acknowledgements

During the course of this research many, directly or indirectly, have contributed most notably Professor Kobori at Oyama National College of technology. I would also like to thank Koichi Hamashita and Jun-ichi Matsuda for their kind support of this project.This research was also funded and supported by Adaptable and Seamless Technology Transfer Program through target-driven R&D, by Japan Science and Technology Agency (JST).

Chapter 1

Spread Spectrum Clock

Abstract: Spread Spectrum Clock generation is a commonly used algorithm to suppress clocks electromagnetic interference noise, first introduced by a printing company in 1970
Keywords: Clock, EMI, Spread Spectrum Clock Generation

1.1 Clock Signal

Electronics devices use clocks to synchronize data as they move from one place to another place or spot, or measure occurrence of values in time. A clock circuit is is usually a circuit utilizing the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise square wave with 50% duty rate (Fig. 1.1) with fixed frequency.

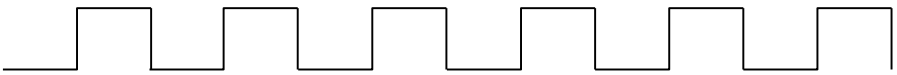


Figure 1.1: Square Wave

It is a bold statement, but clock circuit is the most important component of electronics circuits, in charge of synchronizations and adjuster which circuits functionality and accuracy depends upon it.

A clock circuit essentially functions on a certain frequency and radiates an energy that may interfere with other devices in vicinity. This radiation is called electromagnetic interference or EMI in short.

1.2 Electromagnetic Interference (EMI)

Electromagnetic interference or in short EMI is a noise that affects an electrical circuit functionality by either electromagnetic induction or electromagnetic radiation, emitted from another **source** in vicinity. This disturbance can lead to interruption, obstruction, and/or degrade the effective performance of the original circuit. These effects can range from a simple degradation of transmitting data in the circuit to a total loss of it. The source we mentioned above could be any object, not necessarily electrical circuit, that carries shifting currents; therefore even the Sun or the Northern Lights can have EMI degradation effects (in fact both of these two aforementioned objects have a extensive effect in degradation of Integrated circuits although they are not matter of discussion in this research).

Clocks EMI can potentially affects the reception of AMFM radio, cell phone, television or any form of wireless transfer of data, functioning in its spectrum existence area.

There are many cures for EMI problem but the one we are interested in and focus-on in this research is spread spectrum technique.

All communication watchdogs around the globe that regulate broadcast emission such as in radio or TV (like Interstate Commerce Commission, FCC in US) enforce a limit on EMI for all electronic equipment. Failing to fulfill this requirement, called EMI compliance test , prevents manufacturers from entering the market and/or causes delay of final products shipment; and solving it is a very time consuming and costly task. Therefore assurance of compliance with EMI limits starts at the very early stage of device manufacturing [[1]]. But staying below EMI limit is becoming more and more difficult these days by high increase in clock functioning frequency. Spread spectrum clock generator is a counter-measure which serves as a way to lower EMI [[2]]. It slows down or slows up clock speed within a few percent of its target frequency, thus hammering its EMI peak by spreading it across a range of frequencies (Fig. 1.2).

1.3 Spread Spectrum

The vast majority of digital devices do not require a clock at a fixed, constant frequency. As long as the minimum and maximum clock times are respected (ref. 1.3), the time between clock edges can vary widely from one edge to the next and back again. Such digital devices work just as

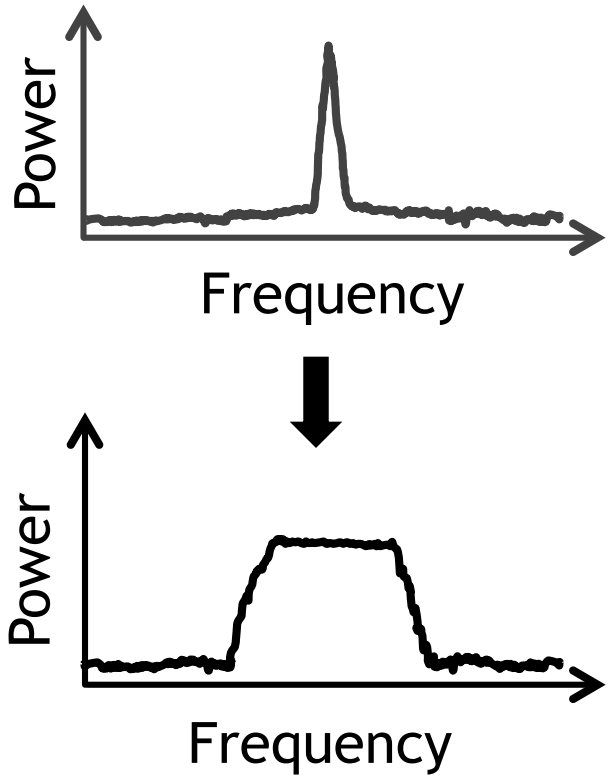


Figure 1.2: Spread Spectrum Clock Generator

well with a clock generator that dynamically changes its frequency, such as spread-spectrum clock generation. Devices that use static logic do not even have a maximum clock time; such devices can be slowed down and paused indefinitely, then resumed at full clock speed at any later time.

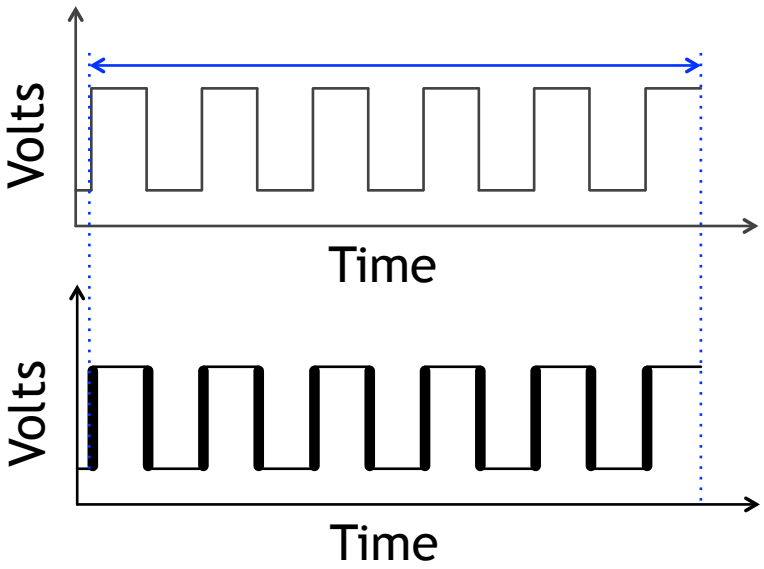
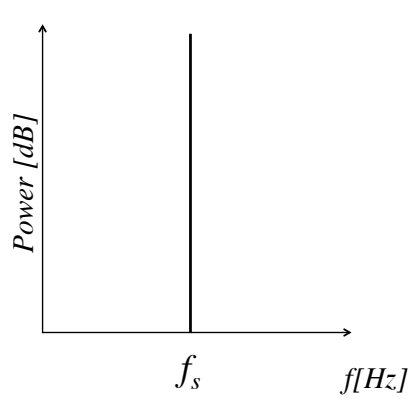


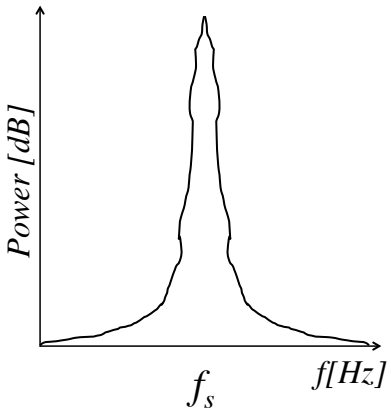
Figure 1.3: Spread Spectrum generated Clock Vs. original clock length

Spread-spectrum clock generation (SSCG) is used in synchronous digital circuits, to reduce the spectral density of the electromagnetic interference (EMI) that these systems generate. A synchronous digital circuit is a circuit that utilize a a clock signal and due its periodic nature, has a very sharp frequency spectrum peak. In fact, a perfect clock signal would have all its energy concentrated at a single frequency (the the functioning frequency plus its harmonics) (fig. 1.4) .

Practical synchronous digital systems radiate electromagnetic energy on a number of narrow bands spread on the clock frequency and its harmonics, resulting in a frequency spectrum that, at certain frequencies, can exceed the regulatory limits for electromagnetic interference (e.g. those of the FCC in the United States, JEITA in Japan and the IEC in Europe). For instance in case of Japan, consumer electronics and industrial equipments EMI limit is



(a) Ideal clock frequency spectrum



(b) In practice clock frequency spectrum

Figure 1.4: Ideal clock spectrum Vs. in Practice clock signal spectrum

slightly different but it is below 40 [dBμV/m] v.s. 50 [dBμV/m] up to 230 MHz frequency and 47 [dBμV/m] v.s. 57 [dBμV/m] above that) (Fig. 1.5)

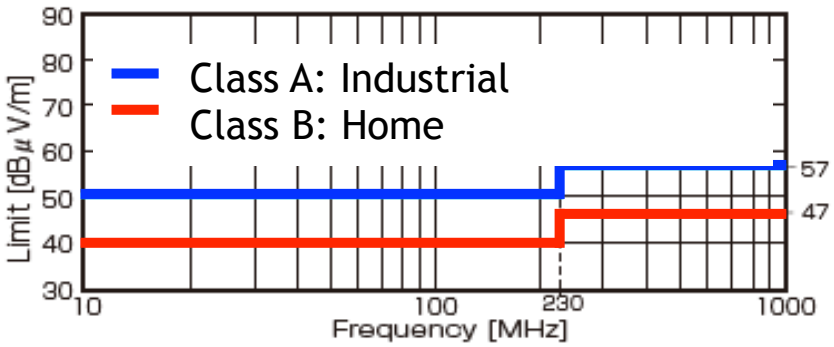


Figure 1.5: EMI Regulation (CISPR22) in Japan

Spread-spectrum clocking avoids this problem by using one of the methods previously described to reduce the peak radiated energy, therefore its electromagnetic emissions comply with electromagnetic compatibility (EMC) regulations.(Fig. 1.6)

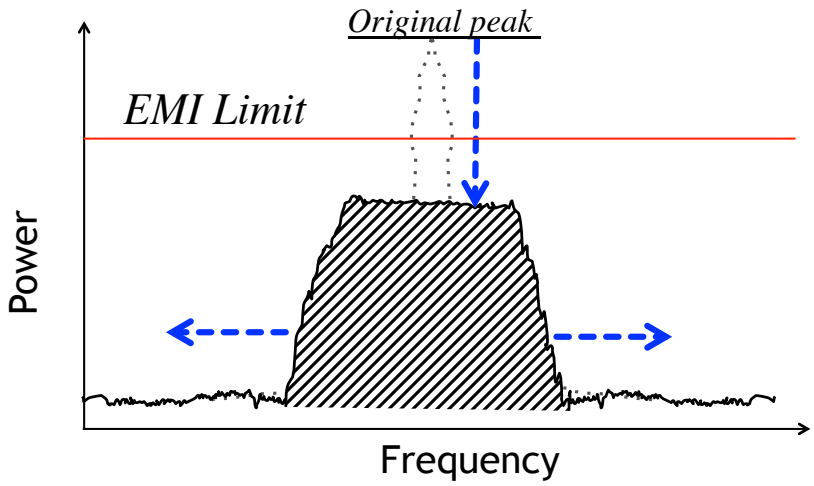


Figure 1.6: Clock's Spreaded spectrum

SSCG has become a common technique to gain regulatory approval because it requires only simple equipment modification. Because of faster clock speed and increasing integration of circuits within portable devices and their small size restraints, traditional and passive measures to reduce EMI, such as capacitors or metal shielding have been excluded. With recent overload in small and portable devices, SSCG method popularity is on the rise.

However, spread-spectrum clocking, like other kinds of dynamic frequency change, can also create challenges for designers. Principal among these is clock/data misalignment, or clock skew. It should be noticed that SSCG does not reduce total radiated energy in fact it might add little bit to id, and therefore systems are not necessarily less likely to cause interference. Because of this the spreader EMI also can also cause trouble such as in case of of external valuable weak radio signal presence in the spreaded spectrum band (Fig. 1.7).

Distributing this same energy into a larger bandwidth prevents systems from putting enough energy into any one narrow band to exceed the statutory limits. The usefulness of this method as a means to reduce real-life interference problems is often debated, since it is perceived that spread-spectrum clocking hides rather than resolves higher radiated energy issues by simple exploitation of loopholes in EMC legislation or certification procedures. This situation results in electronic equipment sensitive to narrow bandwidth(s) experiencing much less interference, while those with broadband sensitivity,

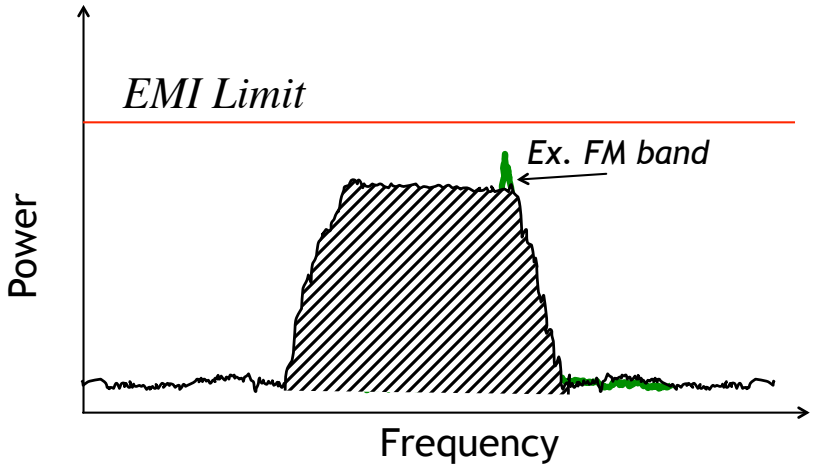


Figure 1.7: Spread spectrum Problem

or even operated at other frequencies (such as a radio receiver tuned to a different station), will experience more interference.

1.4 Summary

In This chapter we introduced Electromagnetic Interference or EMI concept and briefly talked about clocks EMI generation. we also explained and explored spread spectrum clock generation technique benefits in clocks EMI reduction and its usage drawbacks in the circuit.

Chapter 2

Delta Sigma Digital to Time Convertor

Abstract: Delta Sigma Digital to time converter is a unique circuit for conversion of digital signal to time signal. in this chapter we'll talk extensively about delta sigma technique. converter circuit and method of bringing digital signal to time domain
Keywords: Delta-Sigma, Digital-to-Time, converter, time domain

2.1 converters

In Electronics, Digital electronics is a name of field where the representation of signal is a series of discontinues discrete values. digital electronics has gained so much popularity in past several decades because it is easier to manipulate and reproduce a number of known state values rather than continues range of infinite values. To bring natural signal to/from digital domain we need converters

2.1.1 ADC

An analog-to-digital converter is circuit to convert a continues data quantity to a set of discrete numbers which usually involves sampling and quantization process. Accuracy of a ADC converter is usually measured with its sampling frequency,the number of times a continued value is measured in one second and its resolution which determines with how many discrete value the continues data is represented. ADCs other properties can be inferred or calculated according to these characteristics.

2.1.2 DAC

Most signal in the nature are Analog such as sound, temperature, light, etc. To gather these data and use them efficiently in digital circuits and digital electronic sub branches one need a fast and reliable way to convert the signal from and to digital domain. These devices are called signal converter and whether if its role is to change a an analog signal to digital signal or analog signal its name changes. the latter is called digital-to-analog converter or in short ADC or A/D and the later is called digital-to-analog converter or DAC or D/A.

the main purpose of these two are essentially same. they both convert the signal for easier comprehension, weather it is mathematically or computationally as it is the case for ADC or it is to make it easier for the end user to actually feel the result as it is the case in DAC. In Electronics, a converter is a circuit that converts a signal from or to Analog value.

2.2 Delta Sigma Method

Delta-Sigma ($\Delta\Sigma$; or Sigma-Delta, $\Sigma\Delta$) modulation is a digital signal processing, or DSP method for encoding analog signals into digital signals as found in an ADC. It is also used to transfer higher-resolution digital signals into lower-resolution digital signals as part of the process to convert digital signals into analog. In a conventional ADC, an analog signal is integrated, or sampled, with a sampling frequency and subsequently quantized in a multi-level quantizer into a digital signal. This process introduces quantization error noise. The first step in a delta-sigma modulation is delta modulation. In delta modulation the change in the signal (its delta) is encoded, rather than the absolute value. The result is a stream of pulses, as opposed to a stream of numbers as is the case with PCM. In delta-sigma modulation, the accuracy of the modulation is improved by passing the digital output through a 1-bit DAC and adding (sigma) the resulting analog signal to the input signal, thereby reducing the error introduced by the delta-modulation.

This technique has found increasing use in modern electronic components such as converters, frequency synthesizers, switched-mode power supplies and motor controllers, primarily because of its cost efficiency and reduced circuit complexity. Both analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) can employ delta-sigma modulation. A delta-sigma ADC first encodes an analog signal using high-frequency delta-sigma modulation, and then applies a digital filter to form a higher-resolution but lower sample-frequency digital output. On the other hand, a delta-sigma DAC encodes a

high-resolution digital input signal into a lower-resolution but higher sample-frequency signal that is mapped to voltages, and then smoothed with an analog filter. In both cases, the temporary use of a lower-resolution signal simplifies circuit design and improves efficiency.

In brief, because it is very easy to regenerate pulses at the receiver into the ideal form transmitted. The only part of the transmitted waveform required at the receiver is the time at which the pulse occurred. Given the timing information the transmitted waveform can be reconstructed electronically with great precision. In contrast, without conversion to a pulse stream but simply transmitting the analog signal directly, all noise in the system is added to the analog signal, reducing its quality.

Each pulse is made up of a step up followed after a short interval by a step down. It is possible, even in the presence of electronic noise, to recover the timing of these steps and from that regenerate the transmitted pulse stream almost noiselessly. Then the accuracy of the transmission process reduces to the accuracy with which the transmitted pulse stream represents the input waveform.

Delta-sigma modulation converts the analog voltage into a pulse frequency and is alternatively known as Pulse Density modulation or Pulse Frequency modulation. In general, frequency may vary smoothly in infinitesimal steps, as may voltage, and both may serve as an analog of an infinitesimally varying physical variable such as acoustic pressure, light intensity, etc. The substitution of frequency for voltage is thus entirely natural and carries in its train the transmission advantages of a pulse stream. The different names for the modulation method are the result of pulse frequency modulation by different electronic implementations, which all produce similar transmitted waveforms.

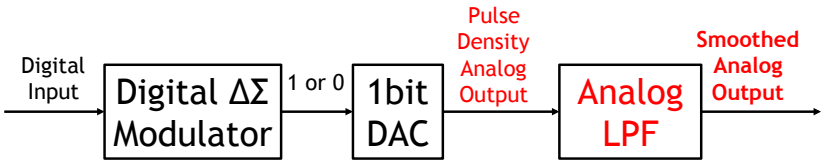
The ADC converts the mean of an analog voltage into the mean of an analog pulse frequency and counts the pulses in a known interval so that the pulse count divided by the interval gives an accurate digital representation of the mean analog voltage during the interval. This interval can be chosen to give any desired resolution or accuracy. The method is cheaply produced by modern methods; and it is widely used.

2.3 Digital-to-Time convertors

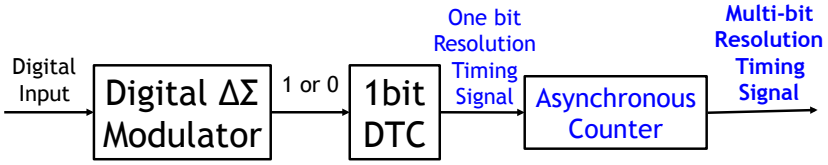
2.3.1 Intorduction to $\Delta\Sigma$ DTC

Traditionally, information has been processed and encoded in the voltage domain; however, more recently information encoding in time domain has

considerably gained popularity. Data conversion in theory is simply the process of working with signals in different domains. The DTC is a converter device which maps a digital value to a timing signal.



(a) ΔΣ DAC configuration.



(b) ΔΣ DTC configuration.

Figure 2.1: ΔΣ DAC and ΔΣ DTC analogy.

Basic distinction between DTC and DAC is the domains where they function;the DTC operates in time domain, whereas the DAC operates in voltage domain.

The process of converting signal from digital to analog (or vice verse) usually involves many techniques including filtering and smoothing of the signal before and after the conversion. In this regard, our proposed DTC is in full compliance with its conventional methods. The DTC includes a digital ΔΣconverter (Fig. 2.1), and there samples are interpolated with a low pass filter (LPF). In time domain, a LPF is used to smooth the signal by cutting high frequency components.

The delta-sigma DTC idea is - to our knowkedge - only used for very recent work [13] which employs DTC driven phase signal conversion for automatic test equipment (ATE) applications. This work differs in two main points from it; First, our proposed methods apply pulse cycle and width modulation with some innovative ways in addition to phase modulation, and also utilize an asynchronous counter to perform as a low pass filter in time domain which -due to its digital nature- is very simple compared to PLL.

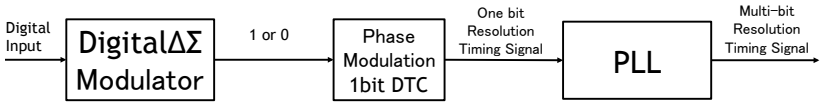
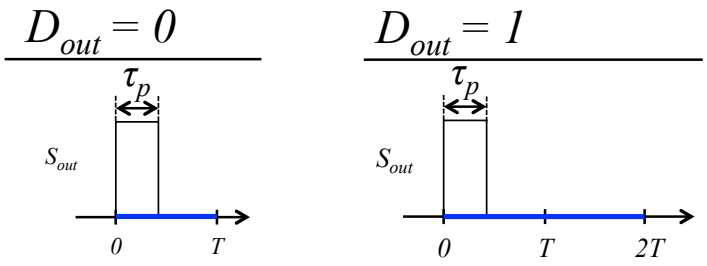


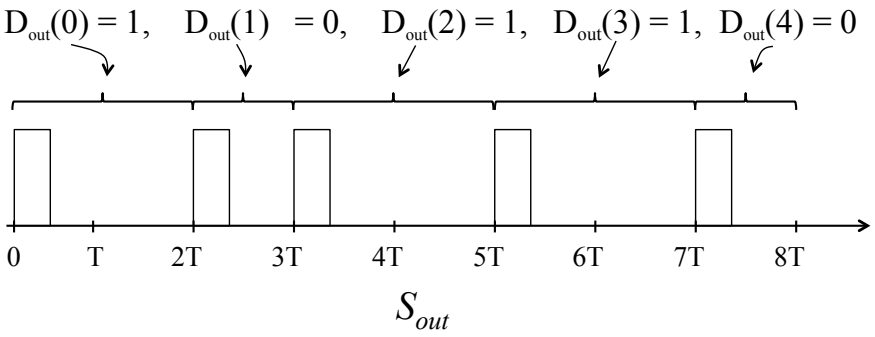
Figure 2.2: Delta Sigma DTC Analogy.

2.3.2 ΔΣPCMDTC

We consider Pulse Cycle Modulation (PCM), which is a manipulation of representing each timing signal pulse cycle (Figs. 2.3, 2.4).



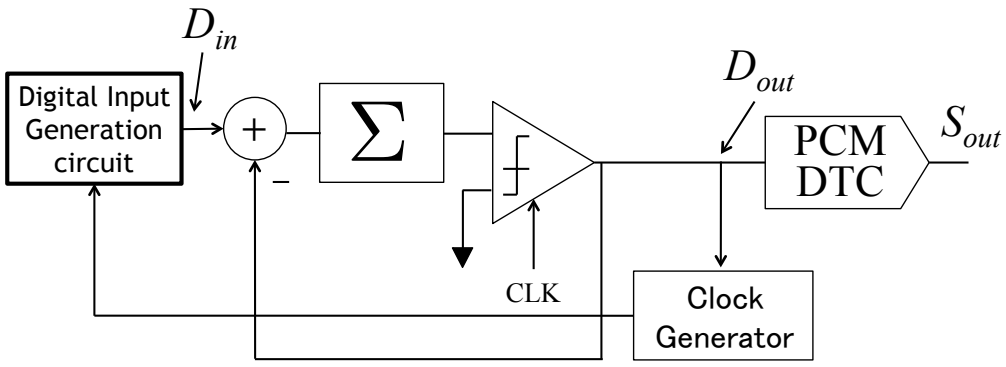
(a) Time signal representation for digital "1" & "0".



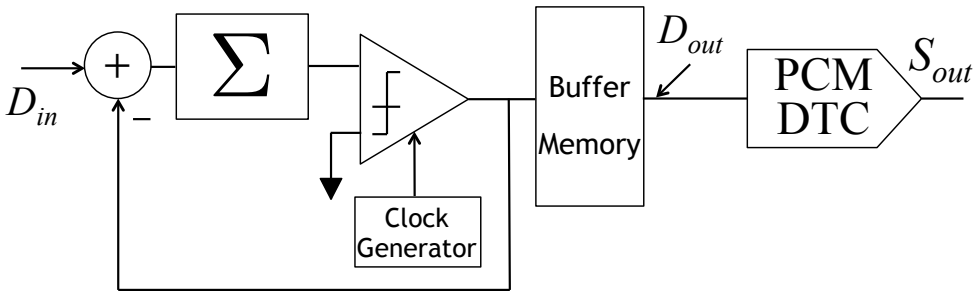
(b) Time signal of 10110 by represented by left time signals.

Figure 2.3: PCMDTC example.

For example, digital "0" is mapped to a time signal with cyclic period of T while "1" is with 2T; then in case of digital input sequence D=10110, the output signal is shown in Fig. 2.3. Block diagram of the ΔΣPCM DTC is illustrated in Fig. 2.4.



(a) PCMΔΣDTC block diagram.



(b) Alternative PCMΔΣDTC block diagram.

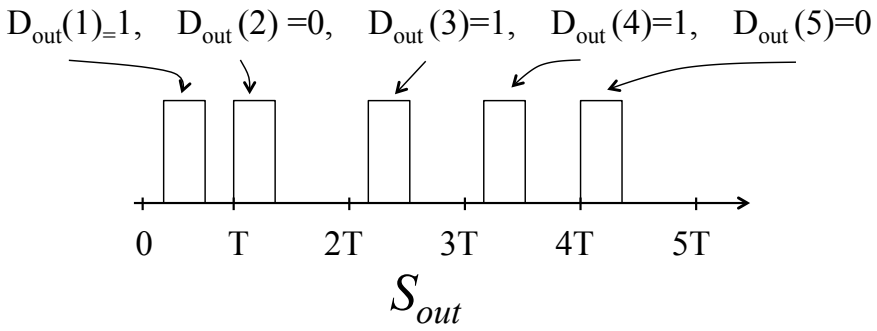
Figure 2.4: PCMDTC circuit block diagram.

Because of unlimited variable periods which we can choose from frequency, PCMDTC could be superior to other following methods in regard to multi-bit modulation.

2.3.3 ΔΣPPMDTC

Pulse Position Modulation (PPM) ΔΣDTC encodes each digital signal value by shifting output pulse signal beginning position (Figs. 2.5, 2.6). So for instance, if we have output pulse with the frequency of f, digital "0" is mapped to pulse with zero shifting position (phase= 0), and digital "1" is mapped to pulse with shifting position of a constant C (where $C \leq T$). A sample modulation of digital value similar to PCMDTC (10110) is shown in Fig. 2.5.

(a) Time signal representation for digital "1" & "0".



(b) Time signal of 10110 represented by left time signals.

Figure 2.5: PPMDTC example.

Two major differences and benefits of PPMDTC compared to the previous method are as follows: First, output signal length is independent of numbers

of "0"'s and "1"'s. Second, it consists of only a delay element and digital multiplexer (Fig. 2.6).

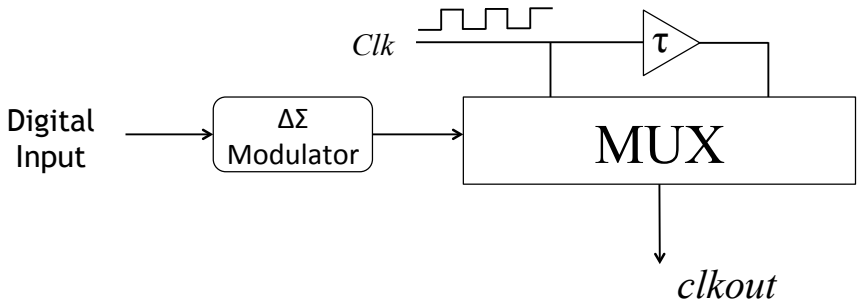


Figure 2.6: PPMDTC circuit block diagram.

This circuit capability of handling high-frequency signal in digital circuit might be game changer factor to choose it over other methods in applications.

2.3.4 ΔΣPWMDTC

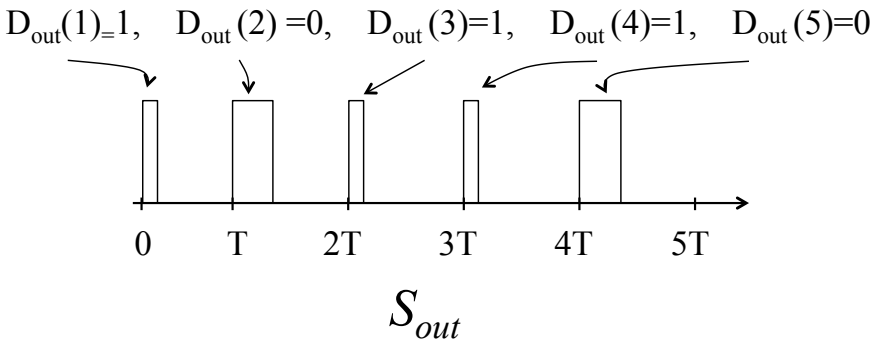
Pulse Width Modulation (PWM) DTC changes the output signal width based on digital input value (Figs. 2.7, 2.8).

Its implementation may be a little bit complex, but its benefits surface up when it is used in conjuncture with other methods, which we will discuss later in this section.

2.3.5 ΔΣPRJDTC

Our proposed Pseudo Random Jitter (PRJ) DTC is very similar to pulse cycle modulator (PCM), because, like PCM, the major distinction between two timing output signals for two distinct digital inputs are pulse frequency (cycle period). However in PRJ, output signal frequency changes arbitrarily (or pseudo randomly) between two (or more for multi-bit DTCs) constant values (Fig. 2.9).

(a) Time signal representation for digital "1" and "0".



(b) Time signal of 10110 represented by left time signals.

Figure 2.7: PWMDTC example.

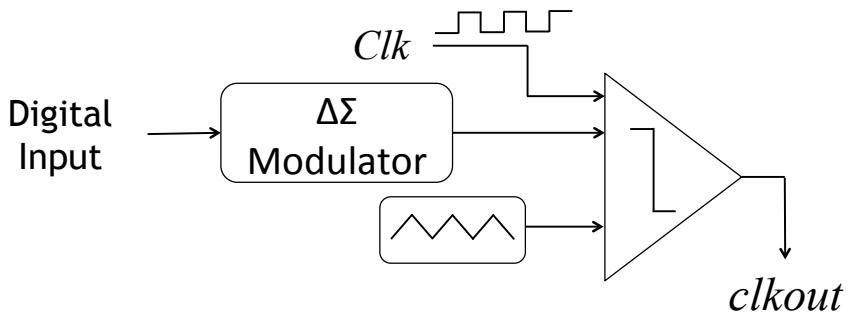


Figure 2.8: PWMDTC circuit block diagram.

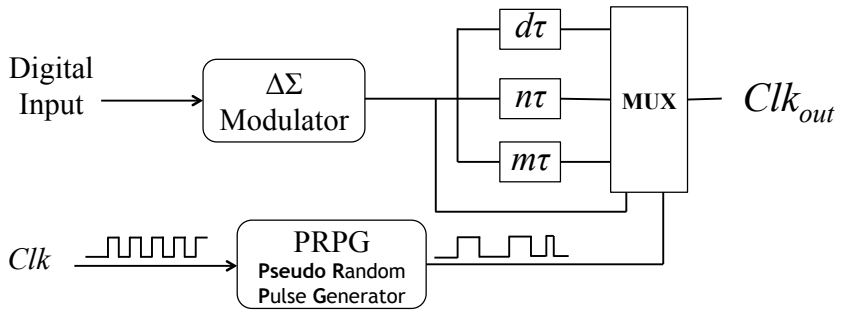


Figure 2.9: PRJDTC circuit block diagram.

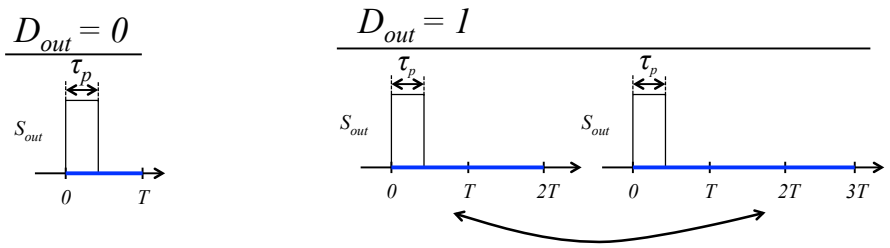
This technique can be achieved by randomly delaying output signal so that it looks like a big jitter in output pulse. Fig. 2.10 shows a sample output of PRJ DTC for a digital input sequence of 10110.

2.3.6 Other ΔΣDTC methods

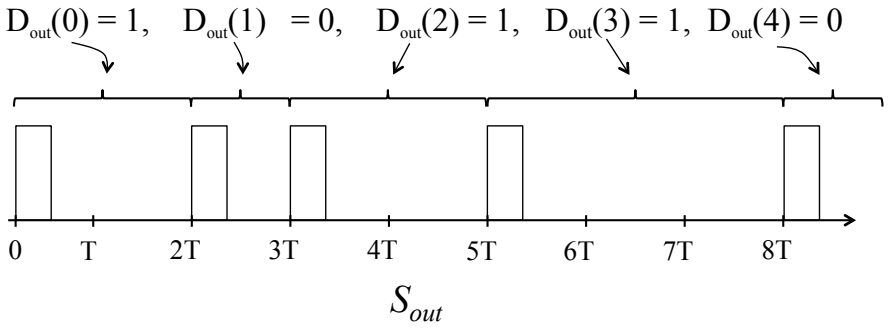
We can extend our proposed DTC methods by combining of PCM, PPM, PWM or randomly changing any of three main characteristics (cycle, position and width) for more effective SSCGs.

2.4 Summary

Digital to time converter is a method to convert a digital value to timing signal, meaning it brings a digital high and digital low value where they are identified by their voltage height regardless of their timing, to time domain signal where pulses height only holds meaning when we look at its timing. In this chapter we introduced several technique to achieve this such as PCM, PPM, PWM, PRJ,... . we also went into these methods configuration and illustrated our point with easy to follow examples



(a) Time signal representation for digital "high" and "low".



(b) Time signal of 10110 represented by left time signals.

Figure 2.10: PRJDTC example.

Chapter 3

SSCG with ΔΣDTC

Abstract: Here we talk about the design of our proposed Delta-Sigma Digital-to-Time converter and its result in simulations.

Keywords: PCMΔΣDTC, PPMΔΣDTC, PWMΔΣDTC, PRJΔΣDTC, compound ΔΣDTC

3.1 Spread Spectrum using various DTCs

3.1.1 SSCG Simulation Methodology

Sine wave with frequency of f_s/N, sampled in N points with sampling frequency f_s is fed to DTC as digital input.

After being noise-shaped by a first-order delta-sigma converter, output has been digital-to-time modulated according to the relevant method. Original clock (without sigma-delta DTC modulation) signal base peak power and its harmonics reach to 66dB (Fig. 3.1).

3.1.2 PCMΔΣDTC Results

Spread spectrum with PCM DTC suppresses spectrum peak significantly and also creates notches at some locations.

The power spectrum of previously introduced signal by various DTC has been presented in Figs. 3.2, 3.3, 3.4, 3.5, where we derive those notch locations by Eq. (3.1). Here we assume that periods T_H and T_L corresponding to digital

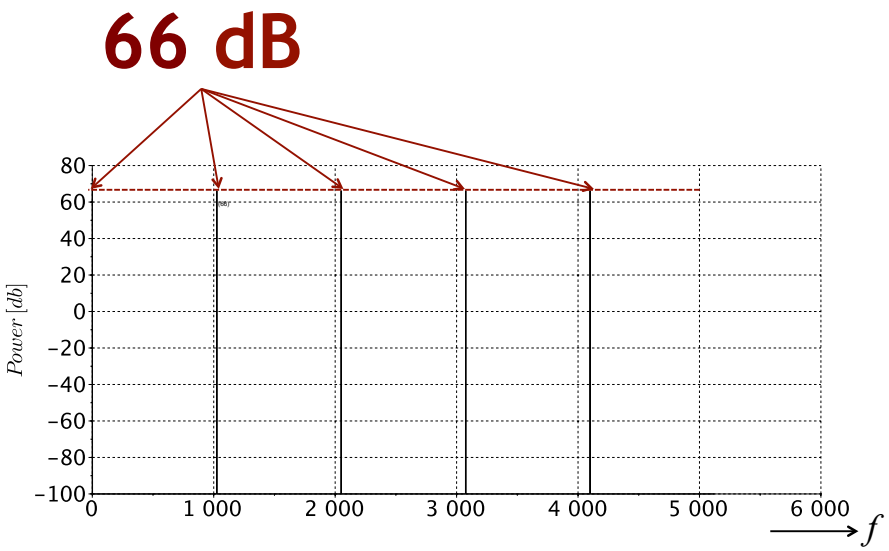


Figure 3.1: Base peak power without DTC modulation.

signals "1" and "0" are integer (n_H, n_L) multiple of constant minimum base period (T_C), respectively.

$$f_{notch} = \frac{K \times (n_H + n_L)}{2|n_H - n_L|} f_s. \tag{3.1}$$

Here $K = |n_H - n_L| - 1, |n_H - n_L| - 2, \cdots, 1$ and
n_H and n_L are positive integers, defined as $n_H = T_H/T_C$, $n_L = T_L/T_C$.

Notice that this equation, as well as other methods' equations, has been extracted by observing statistical probability of notch locations in experimented simulation results.

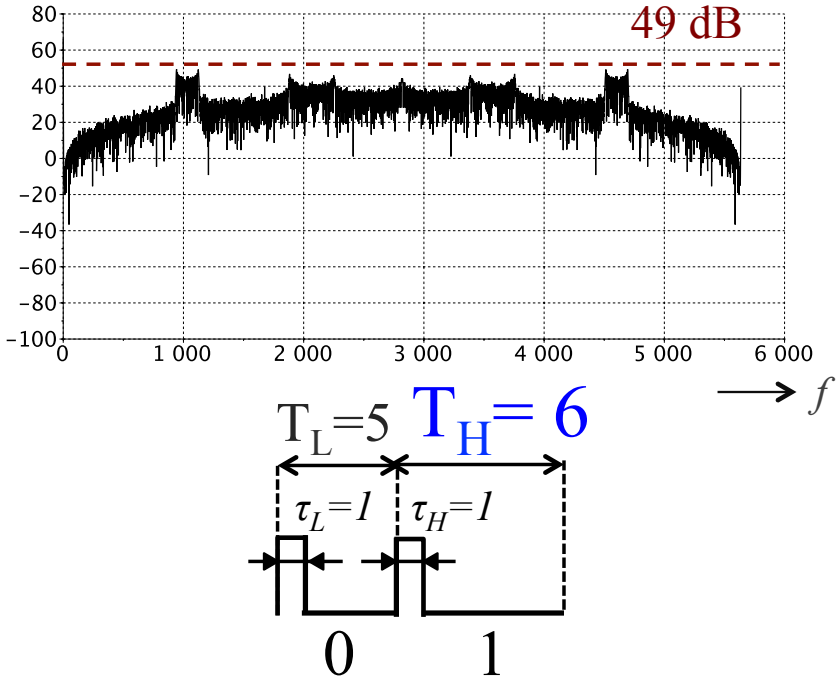


Figure 3.2: $T_H = 6$

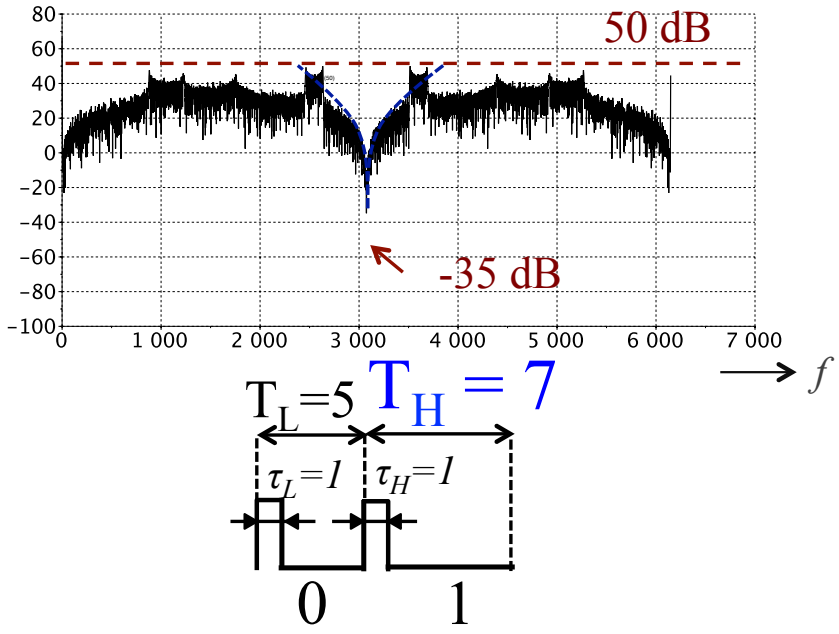


Figure 3.3: $T_H = 7$

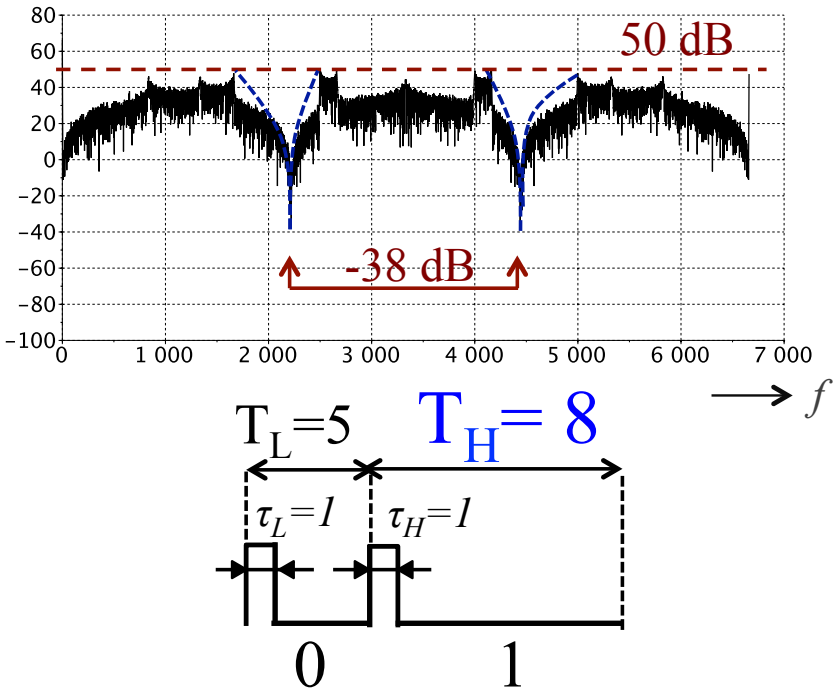


Figure 3.4: $T_H = 8$

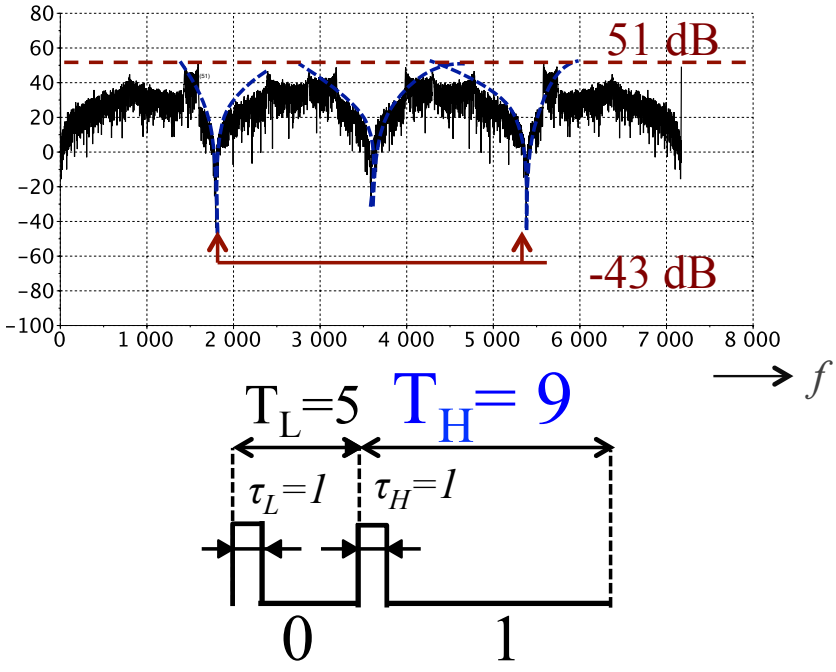


Figure 3.5: $T_H = 9$

3.1.3 PPMΔΣDTC Results

Spread spectrum power of the PPM DTC method is illustrated in Figs. 3.6, 3.7, 3.8, 3.9. If we assume that each pulse period (T_H, T_L) is integer (n_H, n_L) multiple of base period T_C and each pulse phase (ϕ_H, ϕ_L corresponding to digital signals "1", "0" respectively) is integer (q_H, q_L) multiples of constant minimum base period (T_C), then we observe that PPM DTC has capability to lower noise in particular bandwidth given by Eq. (3.2), although PPM DTC influence on signals peaks may not be sufficient.

$$f_{notch} = \frac{K}{|q_H - q_L|} f_s. \tag{3.2}$$

Here $K = |q_H - q_L| - 1, |q_H - q_L| - 2, \cdots, 1$ and
q_H and q_H are positive integers, defined as $q_H = n_H(\phi_H/2\pi)$, $q_L = n_L(\phi_L/2\pi)$.

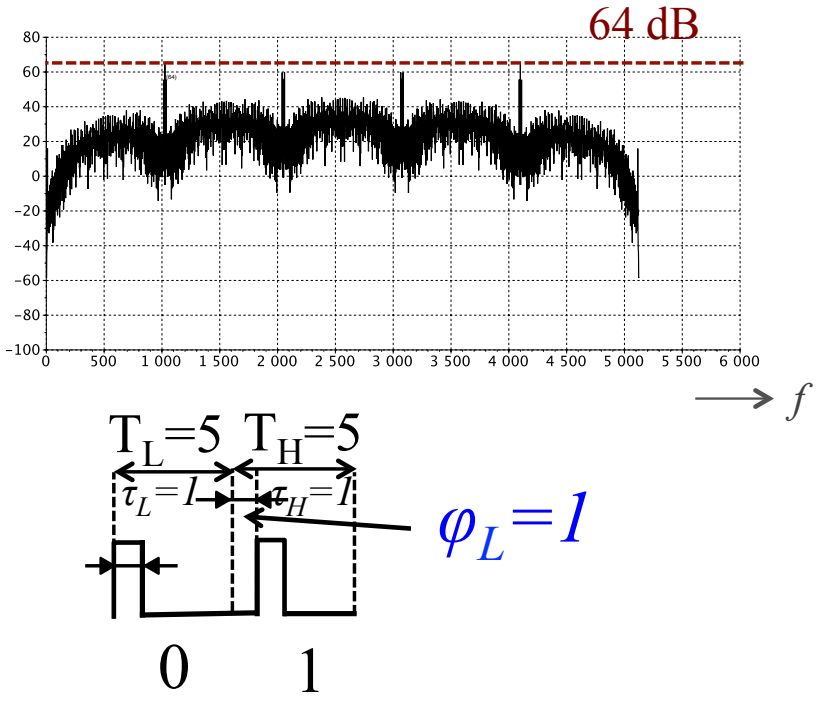


Figure 3.6: $q_H = 1$

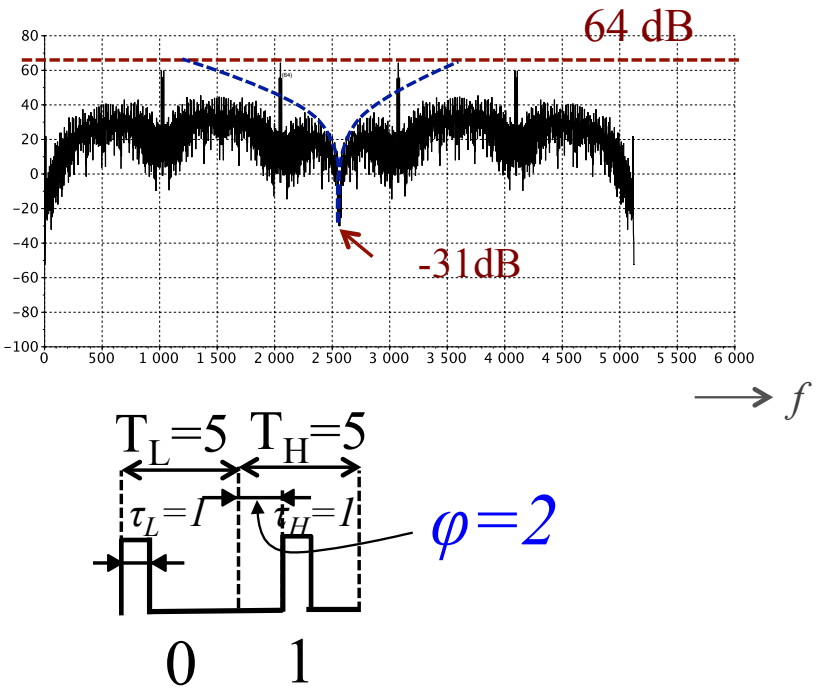


Figure 3.7: $q_H = 2$

3.1.4 PWMΔΣDTC Results

Fig. 3.10 to Fig. 3.13 shows the demonstration of PWM method; in the same manner as PPM DTC, PWM may not have any notable performance on peak reduction, but it creates deep notches in certain bands pretty well, whose locations are given by Eq. (3.3).

$$f_{notch} = \frac{K}{|m_H - m_L|} f_s. \tag{3.3}$$

Here $K = |m_H - m_L| - 1, |m_H - m_L| - 2, \cdots, 1$ and
m_H and m_L are positive integers, defined as $m_H = \frac{\tau_H}{T_C}$, $m_L = \frac{\tau_L}{T_C}$.

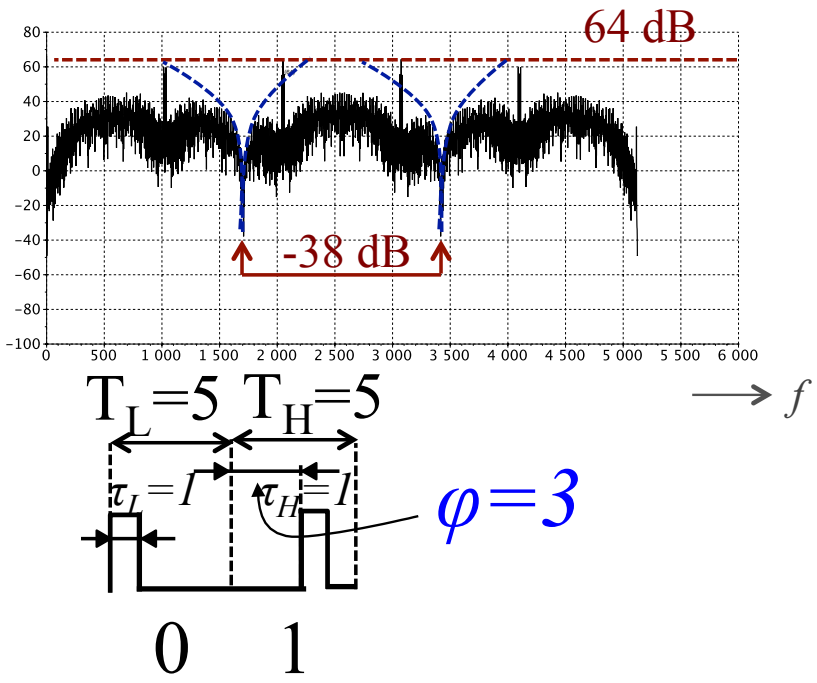


Figure 3.8: $q_H = 3$

3.1.5 PRJΔΣ DTC Results

Finally, Fig. 3.14 through Fig. 3.17(a) illustrates sample signal of generated/-modulated PRJDTC. We observe that this method is very effective in lowering system signals peaks and yields notch if carefully designed. Set the pulse period corresponding to digital "0" to be T_L which is integer (n_L) multiples of constant minimum base period (T_C), and also design so that the pulse period corresponding to digital value "1" arbitrarily alters between T_{H1} and T_{H2}, which are integer (n_{H1} , n_{H2}) multiple of T_C. Then we found that the notch frequency locations are determined by Eq. (3.4).

$$f_{notch} \simeq K(\frac{4n_L + p + q}{4G})f_s. \tag{3.4}$$

Here $K = G-1, G-2, \cdots, 1$ and
G is the greatest common divisor between p and q and $p = |n_{H1} - n_L|$, $q = |n_{H2} - n_L|$.

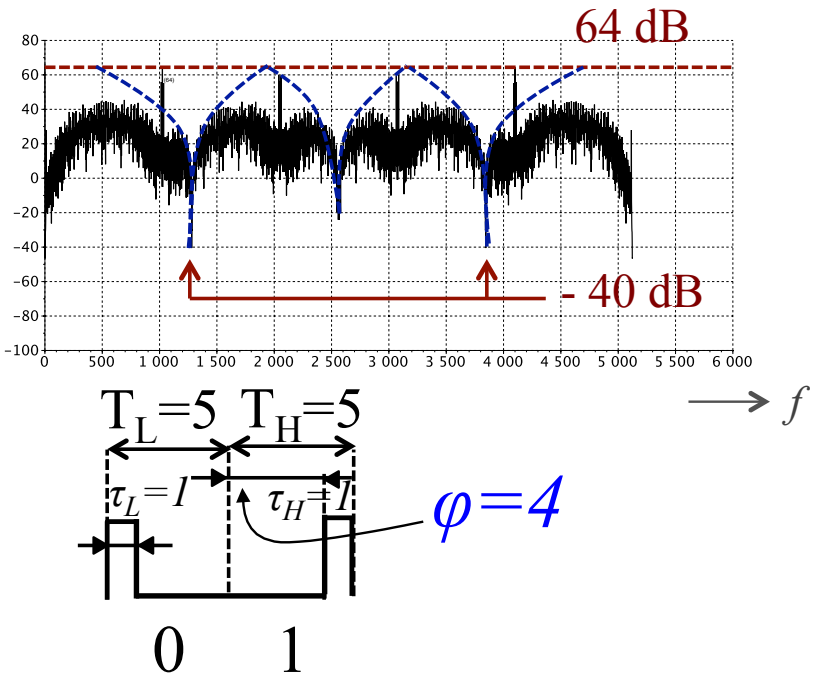


Figure 3.9: $q_H = 4$

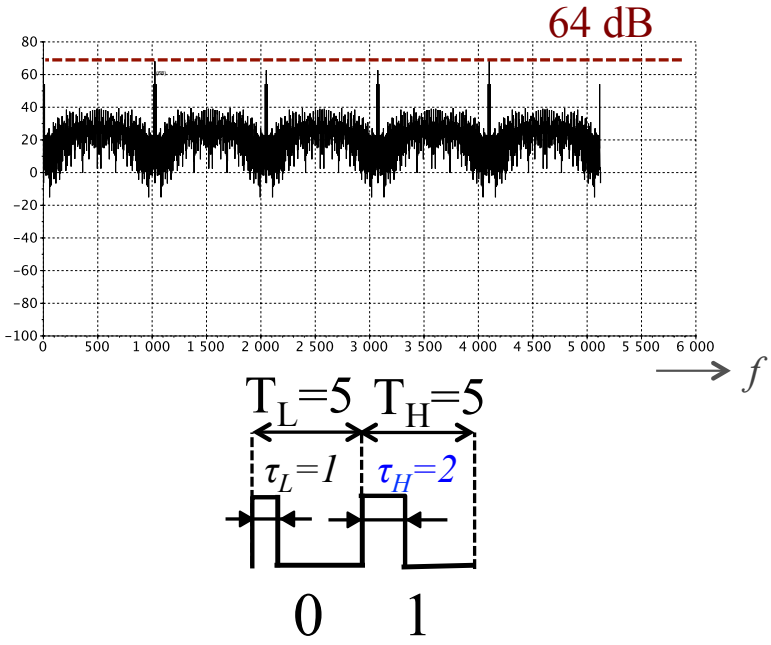


Figure 3.10: $\tau_H = 2$

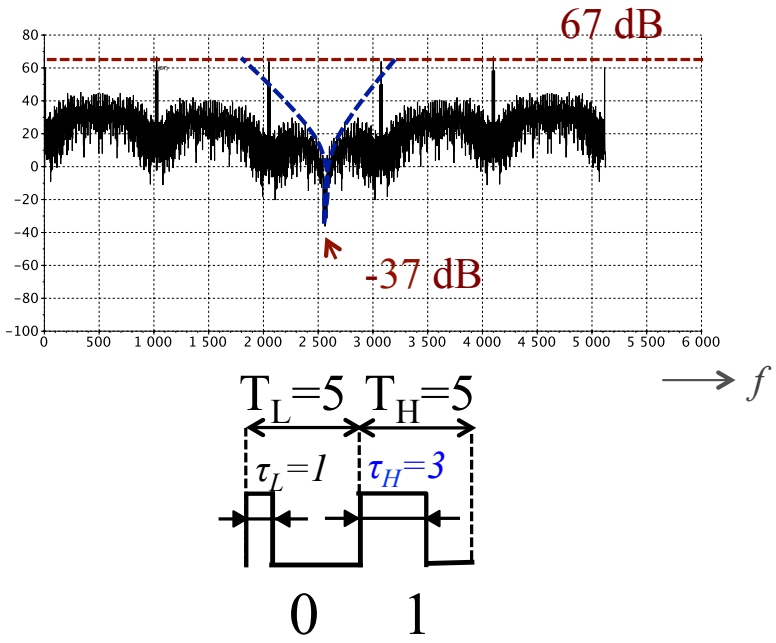


Figure 3.11: $\tau_H = 3$

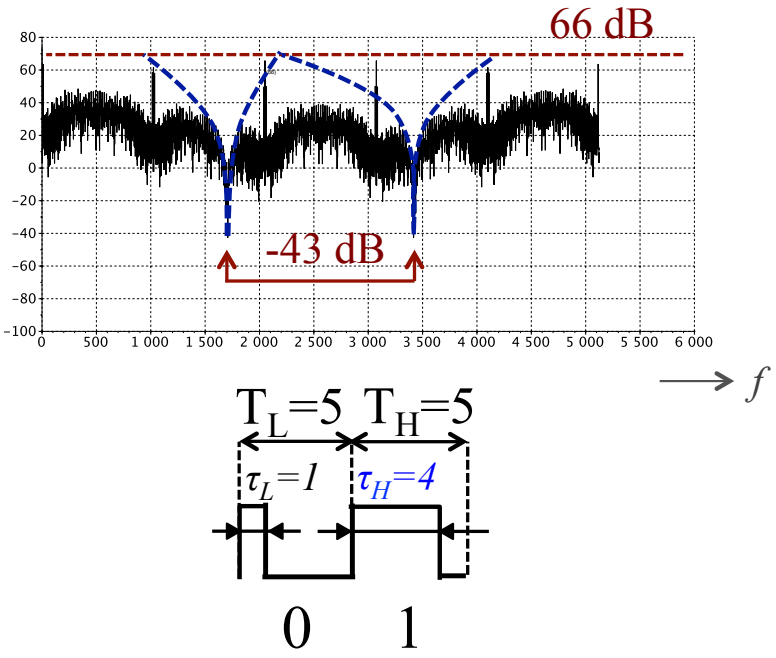


Figure 3.12: $\tau_H = 4$

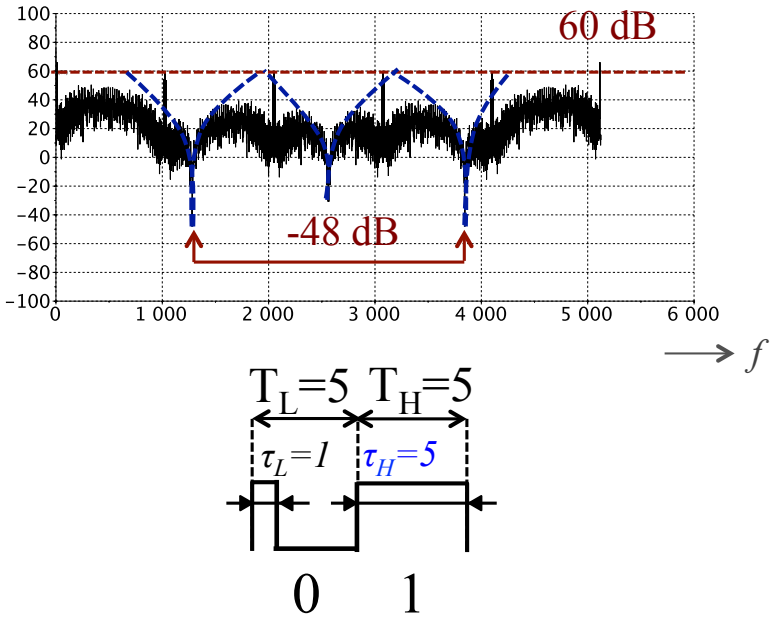


Figure 3.13: $\tau_H = 5$

3.1.6 Compound ΔΣ DTC Results

For the matter of completeness in Fig. 3.18, 3.19, a compound method of PWMDTC + PRJDTC is shown. We notice the affect of PWM DTC in hammering signal high in side bands of the notches in Figs. 3.18.

3.2 Summary

In this chapter we showed previous chapter methods simulation results and we introduced numerically derived notch location equations. Based on this equation circuit designer can easily choose a certain band and use a DTC converter in the circuit to exclude a certain band from spreading spectrum in desired circuit.

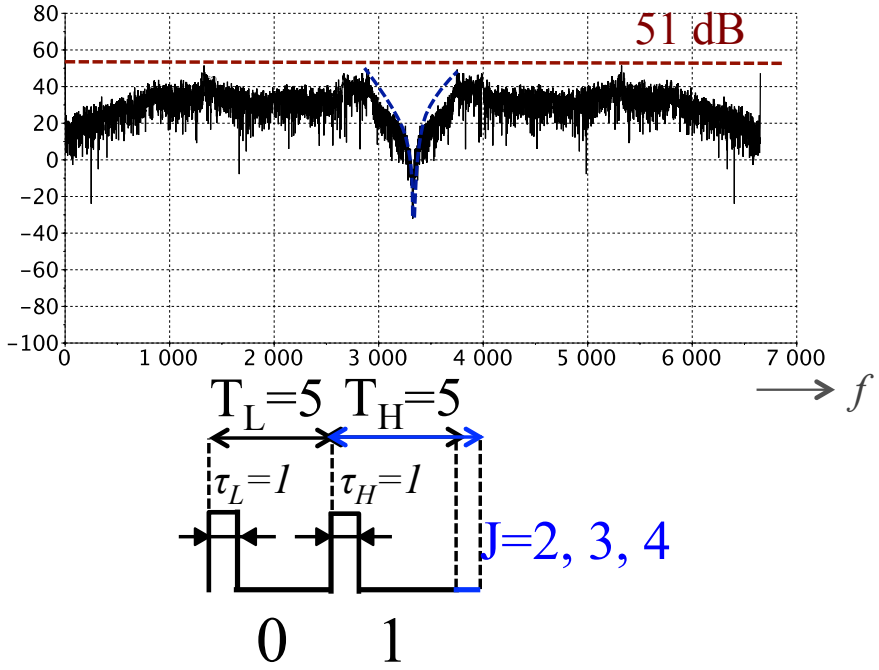
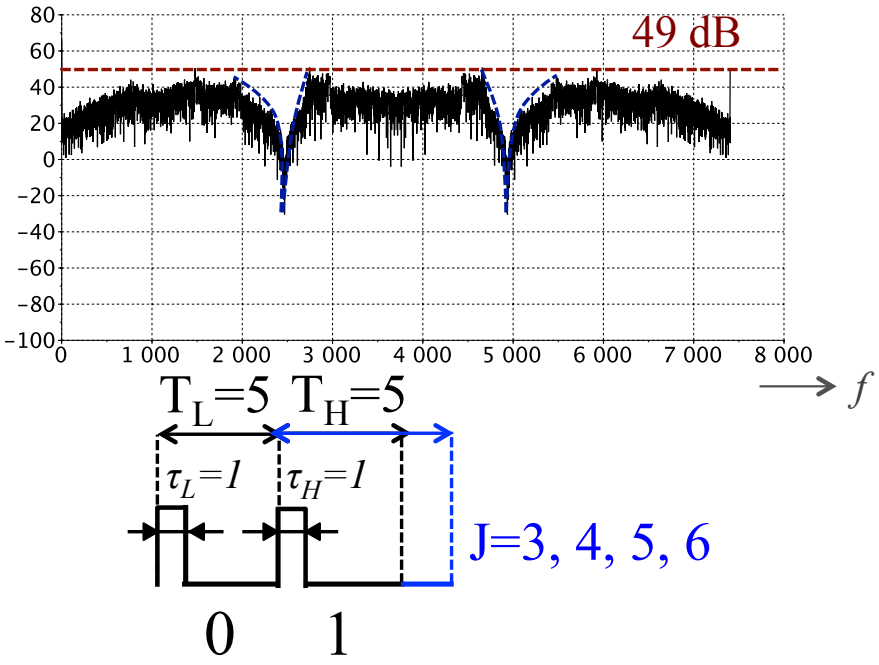
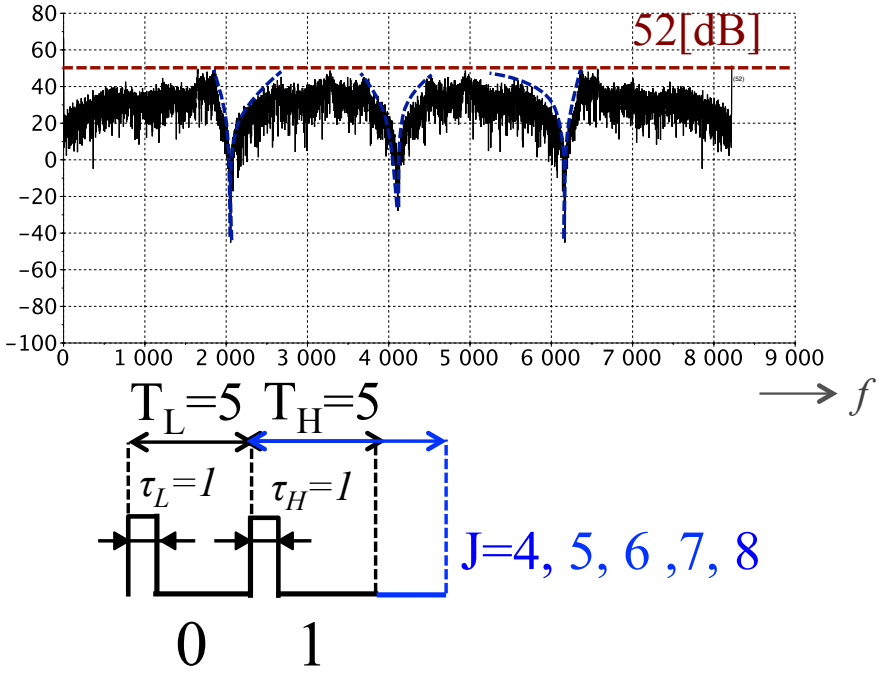


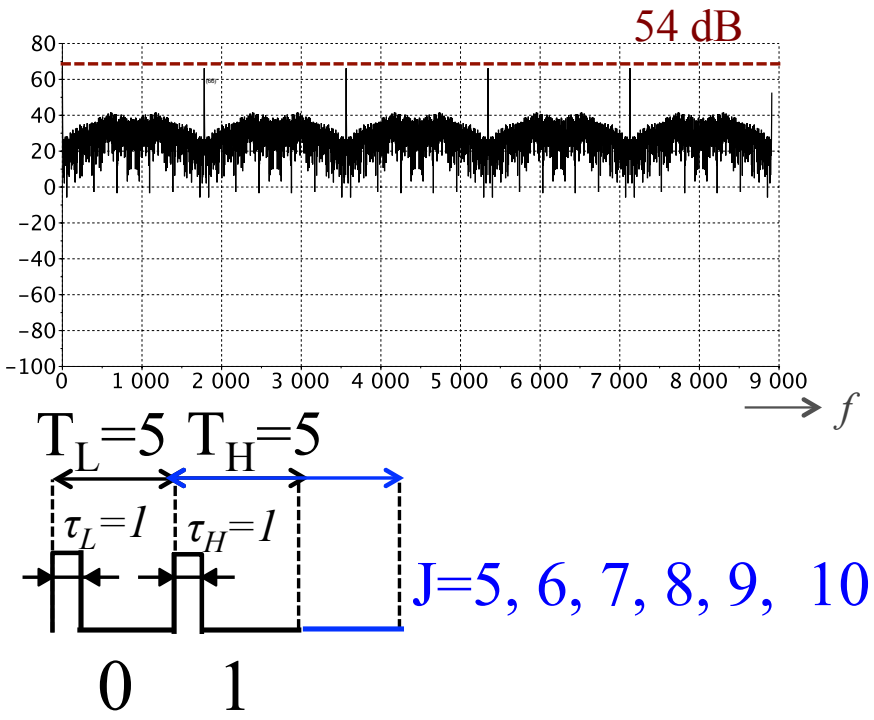
Figure 3.14: $J = 2, 4$



(a) $J = 3, 6$


52[dB]
80
60
40
20
0
-20
-40
-60
-80
-100
0
1 000
2 000
3 000
4 000
5 000
6 000
7 000
8 000
9 000
f
T_L=5 T_H=5
τ_L=1
τ_H=1
J=4, 5, 6 ,7, 8
0
1

(a) $J = 4, 8$


54 dB
80
60
40
20
0
-20
-40
-60
-80
-100
0
1 000
2 000
3 000
4 000
5 000
6 000
7 000
8 000
9 000
f
T_L=5 T_H=5
τ_L=1
τ_H=1
J=5, 6, 7, 8, 9, 10
0
1

(a) $J = 5, 10$

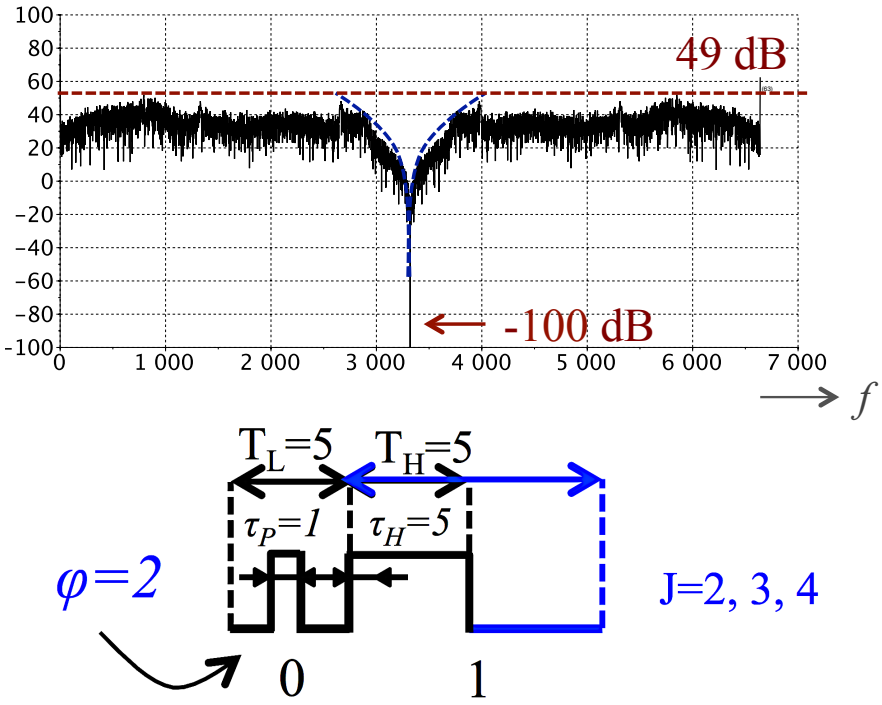


Figure 3.18: $J = 2, 4$ & $\tau_H = 5$

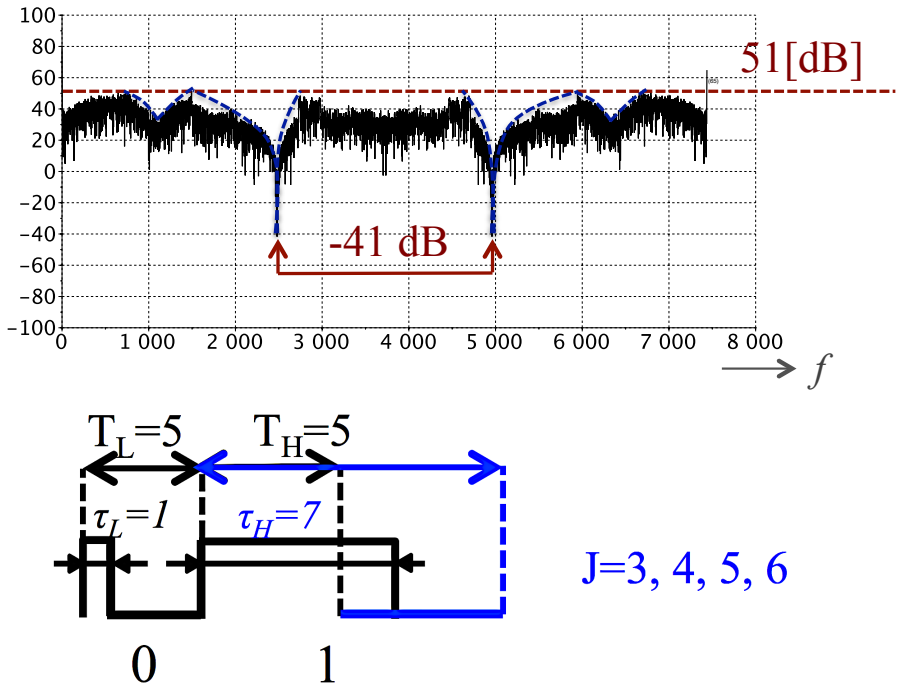


Figure 3.19: $J = 3, 6$ & $\tau_H = 7$

Chapter 4

Adaptive DTC Technology

4.1 Adaptive DTC Idea

In our research we find out that SSCG technique we have introduced previously [9] is a perfect match (or by some means the only available solution) for adaptive clock spread spectrum as it is a simple digitally implementable method to convert a clock based on digital values and is functional at high frequencies, implementable fully with digital circuit. With a delta-sigma DTC algorithm implemented in programmatically configurable digital circuit, location of the required exclusion spectrum bands can be sensed (by a switch or other measures) and DTC algorithm parameters can change automatically and output clocks shape can change on the fly (Fig. 4.1).

Adaptive SSCG can utilize only one form of many delta sigma DTCs proposed such as PCM or PPM or it can utilize multi methods and choose the method dynamically base on set conditions. (Fig. 4.2).

For instance, in case of PCMDTC, based on external factor 1 to 3, PCM configuration changes adaptively and in result notch relocates accordingly (Fig. 4.3). In contrast to formerly introduced SSCG, delta-sigma DTC methods, it dynamically computes output pulse timing signals characteristics and shifts the DTC method internally without affecting overall circuit design architecture.

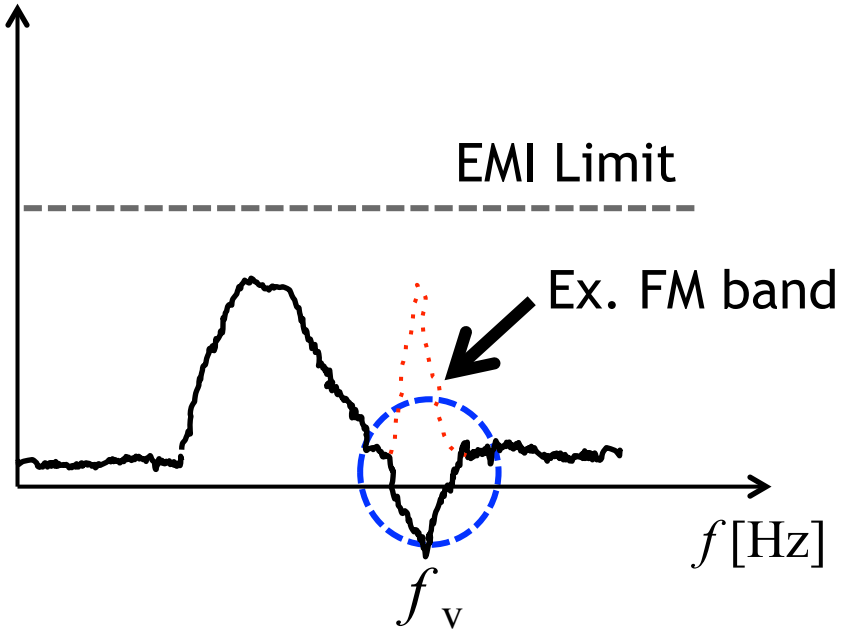


Figure 4.1: Spread Spectrum Problem

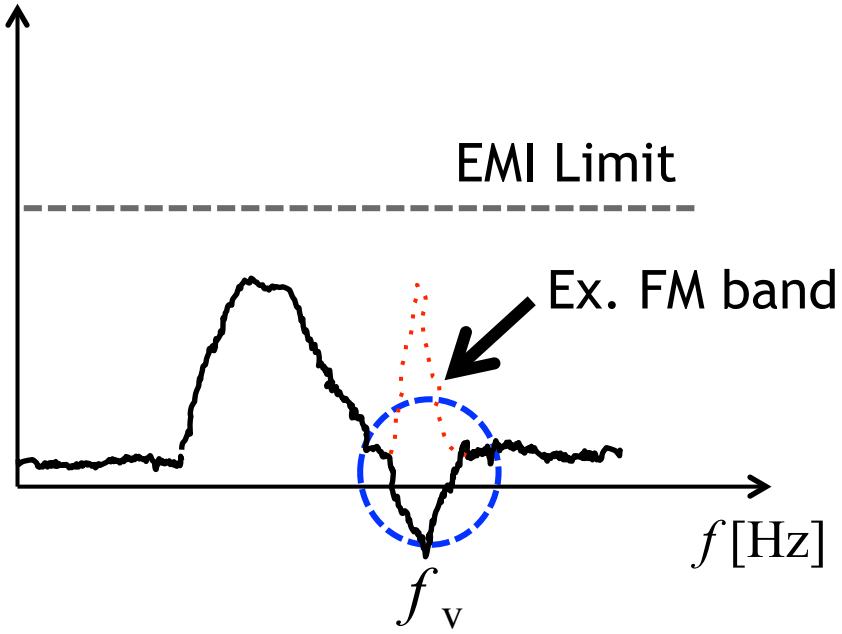


Figure 4.2: Adaptive Spread Spectrum analogy

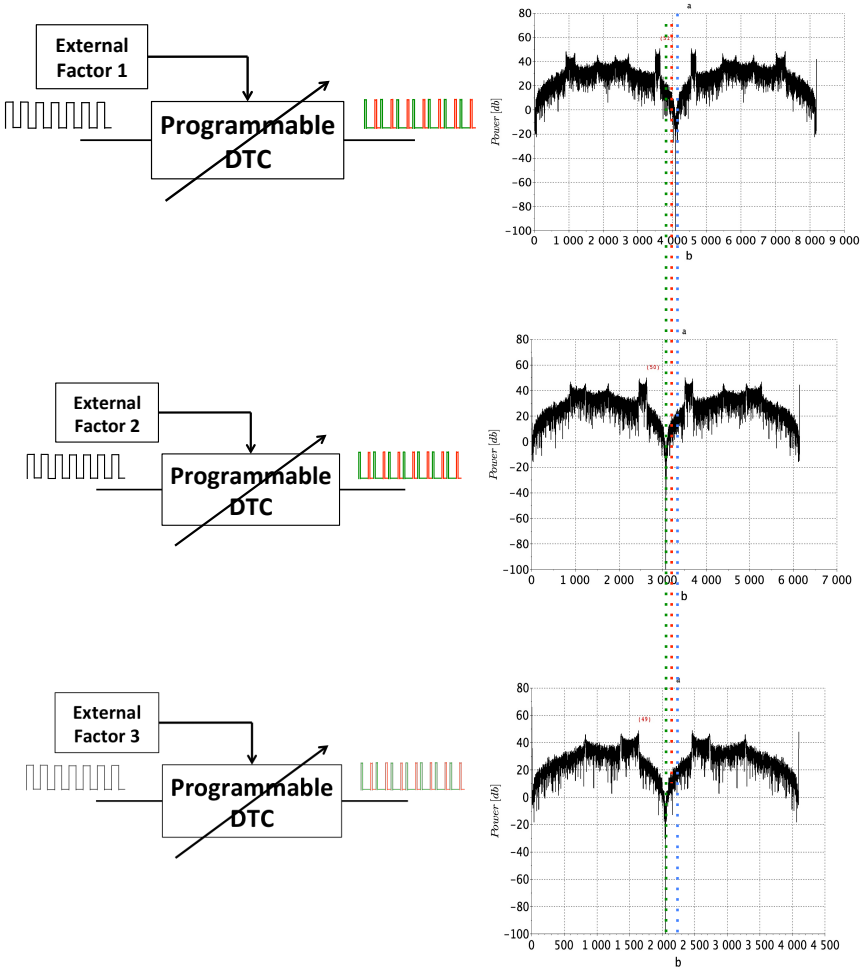


Figure 4.3: Factor 1 in Programmable DTC and its Spread Spectrum bands

4.2 Summary

In this research we have introduced the recent development in time domain signal analysis and current situating in its usage in DTCs. Then we talk about a method built over introduced SSCG with exclusive noise band to adaptively and dynamically change output pulses characteristics in order to tailor-out or exclude certain bandwidth from spreading.

Offered method opens up broad range of opportunities for circuit designer to prioritize their task of passing EMI compliance test for their unit without worrying too much about its later usage and effects on other surrounding electrical circuit equipment's.

The feasibility of current method has been verified by numerical analysis and

its real world bench marking is in process of implementation in FPGA.

Appendices

Appendix A

Publication List

[1] Ramin Khatami, Haruo Kobayashi, Yasunori Kobori, "Delta-Sigma Digital-to-Time Converter For Band-Select Spread Spectrum", Key Engineering Materials.

[2] Ramin Khatami, Haruo Kobayashi, "Delta-Sigma Digital-to-Time converter and It's Application For Band-Select Spread Spectrum" 58th System LSI Joint seminar, Tokyo University of Agriculture and Technology(Oct. 18, 2014).

[3] Ramin Khatami, Haruo Kobayashi, Yasunori Kobori, "Band Select Spread Spectrum Clock Generation with Delta-Sigma Digital Time Convertor Circuit"", IEE Electronic Circuit Conference, Yokohama, Japan (Mar. 7, 2014).

[4] Ramin Khatami, Haruo Kobayashi, "Spread spectrum clock generation noise band selection technology", JST New Technology Presentation Meeting, Tokyo, Japan (Mar. 6, 2014).

[5] Ramin Khatami, Haruo Kobayashi, Yasunori Kobori, "Delta-Sigma Digital-to-Time Converter For Band-Select Spread Spectrum Clock", 5th International Conference on Advanced Micro-Device Engineering (AMDE2013) , Gunma, Japan (Mar. 3-4, 2014).

[6] Ramin Khatami, Haruo Kobayashi, Yasunori Kobori Gunma University, Japan, "Delta-Sigma Digital-to-Time Converter For Band-Select Spread Spectrum Clock", 5th International Conference on Advanced Micro-Device Engineering (AMDE2013) Kiryu, Japan (Dec. 19, 2013).

[7] Ramin Khatami, Haruo Kobayashi, Nobukazu Takai, Yasunori Kobori, Tetsuji Yamaguchi, Eiji Shikata, Tsuyoshi Kaneko and Kimio Ueda, "Delta-Sigma Digital-to-Time Converter and its Application to SSCG", The 4th IEICE International Conference on Integrated Circuits Design and Verification, Ho Chi Minh City, Vietnam (Nov. 15-16, 2013).

[8] Ramin Khatami, Haruo Kobayashi, Yasunori Kobori, Takai Nobuhiro "Delta-Sigma Digital-to-Time Convertor consideration", 3rd IEE Tokyo Branch Tochigi-Gunma joint Research Presentations, Utsunomiya Prefecture, Japan (Mar. 2-Mar.1, 2013).

[9] Ramin Khatami, Fatemeh Hassani, Takuya Arafune, Yasunori Kobori, Haru Kobayashi, "Spread Spectrum Clock Generator With Adaptive Band Exclusion", 5th IEE Tokyo Branch Tochigi-Gunma joint Research Presentations, Utsunomiya Prefecture, Japan (March. 2-Mar.3, 2015).

Appendix B

Simulation Settings

All programs in this research have been constructed using Scilab, an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language developed by Scilab Enterprises and instantiated back in 1990s by French Institute for Research in Computer Science and Automation (INRIA) and Ecole des Ponts ParisTech.

Simulation results in this research were produced using following codes with Scilab version 5.5.1 and simulation tool box (PC) configurations as follow.

Model IMac(21.5-inch, Late 2009)
OS Yosemite (Version 10.10.2)
Processor 3.06 GHz Intel core 2 Duo
Memory 8 GB 1067 MHz DDR3
Graphics NVIDIA GeForce 9400 256 MB

B.1 Digital-to-Time Converter logic

```
//DTC.sci
function [DTCPall]=RandDTC(DigiDec, N, M, Phi, jitter)
Leng = length(DigiDec);
// all DTC Time output for each input signal
s = 1
jit1= jitter(1)
jit2= jitter(2)
DTCPall = zeros(1,Leng*(N(2)+jit2));
for i=1:Leng ;
    //index for current value
    curvalidx = DigiDec(i)+1;
```

```
    n = N(curvalidx);
    m = M(curvalidx);
    ph = Phi(curvalidx);
    //total length= N
    //high length = M
    //start zero length = phi
    //end zero length = N-(phi+M) + random jitter for HIGH and LOW
    endzerolength = n-(ph+m)

    //if jitte is here endzerolength = endzerolength + random jitter
    if (jit2 > jit1+1) then
        if curvalidx==1 then
            if rand()>=0.5 then
                endzerolength = endzerolength + jit1;
            end
        elseif curvalidx==2
            if rand()>=0.5 then
                endzerolength = endzerolength + jit2;
            end
        end
    end

    curvalidx
    //temp
    tempDTC = [zeros(1,ph) ones(1,m) zeros(1,endzerolength)];
    tempDTClen = length(tempDTC);
    DTCPall(s:s+tempDTClen-1) = tempDTC;
    s= s+tempDTClen
end
    DTCPall = DTCPall(1,1:s-1);
endfunction
```

B.2 Delta Sigma

```
// Kobayashi Labratory - Gunma University
// Ramin Khatami
// Feb 2015

//---Delta Sigma function---//
//here we dont use order and integrator
function [out, Dout, DigiDec]=DelSig(sig)
```

```
Vref=1;
Vlow=0;

//delta sigma parameters
err = 0;
a = 0;

//Delta Sigma analog output
out = zeros(1,length(sig));

//Delta Sigma Digital output
Dout = zeros(1,length(sig));

//decimal representation of Delta Sigma Digital Output
DigiDec = zeros(1,length(sig));

for i=2:length(sig)
    // error is newly input signal - previously output signal
    err(i) = sig(i)-out(i-1);
    a(i) = err(i)+a(i-1);

    //quantization
    if( a(i) >= 0) then
        out(i) = 1;
        DigiDec(i) = 1;
        Dout(i)= 1;
    else
        out(i) = -1;
        DigiDec(i) = 0;
        Dout(i)= 0;
    end
end

endfunction
```

B.3 Plotter Source code

```
////////////////////////////////////////////////////////////////////////////////////////////
//plots
function [aa]=Stylize(subject,xTitle,yTitle,Auto,bounds)
```

```
    title(subject,"color","black","fontsize",5);
    xlabel(xTitle,"fontsize",6,"color","black");
    ylabel(yTitle,"fontsize",6,"color","black");
    xset("font",1,6);
    //change font
    xgrid(1);
    aa = gca();
    aa.children.children.thickness =2;
    aa.parent.background= -2;
    a = gcf()
    a.figure_size = [1280 920];
    //a.figure_size = [800 600];

    sleep(100)
    if(Auto==1)
        BB = aa.data_bounds;
        xmin=BB(1,1);
        ymin=BB(1,2)+BB(1,2)/20;
        xmax=BB(2,1);
        ymax=(BB(2,2)+BB(2,2)/20);
        bounds= [xmin,ymin;xmax, ymax];
    end
    aa.data_bounds=bounds;
endfunction

function []=SaveThis(fileName)
    //EPS export
        sleep(200);
        xs2jpg(gcf(), fileName+'.jpg');
endfunction

//plot signal
function [f]=plotSig(t,sig,alone)
    if(alone)
        f = figure();
    else
        f = gca();
    end
    plot2d(t,sig);
endfunction

function [f]=plotDelSigOut(out,alone)
    if(alone)
```

```
        f = figure();
    else
        f = gca();
    end
    plot2d2(0:length(out)-1, out);
endfunction

//plot Average Del Sig out
function [f]=plotDAOut(out,alone)
    if(alone)
        f = figure();
    else
        f = gca();
    end

    DA = [];
    for i=1:length(out)
        DA(i)= [mean(out(1:i))];
    end

    plot2d(0:length(DA)-1,DA);
endfunction


//plot digital quantized signal
function [f]=plotDigi(DigiDec,alone)
    if(alone)
        f = figure();
    else
        f = gca();
    end
    Qn = (max(DigiDec)+1)^(1/2);

    plot2d2(0:length(DigiDec)-1, DigiDec);

    aa = gca();
    aa.y_ticks =
        tlist(['ticks','locations','labels'],0:1:Qn+1,string(dec2bin(0:1:Qn+1)));
endfunction



//plot DTC signal
```

```
function [f]=plotDTC(DTCPall,alone)
    if(alone)
        f = figure();
    else
        f = gca();
    end
    plot2d2(0:length(DTCPall)-1, DTCPall);
endfunction



//fft and log10 of fft
function [f]=plotFFT(DTCFFT, alone)
    if(alone)
        f = figure();
    else
        f = gca();
    end
    //[imax jmax]= max(DTCPallFFTAbslog);
    //[imax2 jmax2]= max(DTCPallFFTAbslog(2:($/4)+1));
    //temperoraily fftN = ceil((length(DTCPallFFTAbslog)+1)/2);
    //temperoraily plot2d(0:fftN-1,DTCPallFFTAbslog(1:fftN));
    //plot2d(DTCPallFFTAbslog);
    plot2d2(0:length(DTCFFT)-1,DTCFFT);
    [imax jmax]= max(DTCFFT(10:$));
    xstring(jmax+5,imax+5,msprintf('(%.f)', imax));
    txt = gce();
    txt.font_size = 4;
    txt.font_style = 0;
    txt.font_color = 5;
    [imin jmin]= min(DTCFFT(10:$));
    xstring(jmin+10,imin-5,msprintf('(%.f)', imin));
    txt = gce();
    txt.font_size = 4;
    txt.font_style = 0;
    txt.font_color = 2;
    //xstring(jmax2+10,imax2-10,msprintf('(%.f)', imax2));

endfunction
```

B.4 Additional Functionality

```
//looper
function [bi] = dec2bi(de, biNum)
    bi = strtod(strsplit(string(dec2bin(de,biNum))));
endfunction

function [dec] = bi2dec(bi)
    dec = bin2dec(strcat(string(bi(1:$)),''));
endfunction
```

B.5 GUI Program

```
s = [];
out = [];
dout = [];
digidec = [];
dtcpal = [];
dtcfft = [];
function update_plot(update_idx)
    global manu_ui_el VALUES notch_method_list
    VALUES_STATUS = zeros(1,length(VALUES));
    for k = update_idx
        VALUES_STATUS(k) = 1
    end

    manu_ui_el(11).string= "Computing...";

    for i=find(VALUES_STATUS==1)
        if i >1 then
            VALUES(i) = strtod(strsplit(manu_ui_el(i).string,","));
        end
    end

    //if all zero value has changed
    if manu_ui_el(1).value ==1 then
        manu_ui_el(2:10).enable = "off";
        notch_method_list.enable = "off";
        manu_ui_el(4).enable = "on";
        notallzero = 0;
    else
```

```
    manu_ui_el(2:10).enable = "on";
    notch_method_list.enable = "on";
    notallzero = 1;
end

//if sampling freqhas changed
sig_chg_flg = 1;
if (VALUES_STATUS(2)==1 | VALUES_STATUS(3)==1 |
    VALUES_STATUS(8)==1) then
    fsamp = VALUES(8);
    T=1/fsamp;
    t = (0:T:1);
    F = VALUES(2);
    A = VALUES(3);
    if(A==0)
        sig = A*ones(1,length(t));
    else
        sig = A*sin(2*%pi*F*t);
    end
    //recalculte delta sigma
    [Out, Dout, DigiDec]= DelSig(sig);
    sig_chg_flg = 1;
else
    sig = s;
    Out = out;
    Dout = dout;
    DigiDec = digidec;
    sig_chg_flg = 0;
end

//if signal or dtc parameter has changed calculate dtc and
    redraw graph
if (sig_chg_flg==1 | VALUES_STATUS(4) | VALUES_STATUS(5) |
    VALUES_STATUS(6) | VALUES_STATUS(7)) then
    DTCPall = RandDTC(notallzero*DigiDec, VALUES(4), VALUES(5),
        VALUES(6), VALUES(7));
    DTCFFT = 20*log10(2*abs(fft(DTCPall)+1.e-6));
    //////
    // wait untill computation is done so that
    drawlater();
    //get a handel of second graph on screen if it exists,
        creat it if not
    graph_fig = scf(1);
```

```
        //delete former data
        if (graph_fig.children~=[]) then
            delete(graph_fig.children);
        end
        graph_fig.figure_position = [430 0];

        //draw fft plot
        plotFFT(DTCFFT,0);
        Stylize("a", "", "$Power \,
            [db]$",0,[0,-100;length(DTCFFT), max(DTCFFT)+5]);
        drawnow();
    else
        DTCPall = dtcpal;
        DTCFFT = dtcfft;
    end

    //reset values update status
    for i=find(VALUES_STATUS==1)
        VALUES_STATUS(i) = 0;
    end
    manu_ui_el(11).string= "Finished.";
    [s, out, dout, digidec, dtcpal, dtcfft]= resume(sig, Out, Dout,
       DigiDec, DTCPall, DTCFFT);
endfunction

function save_plot()
    global VALUES NAMES manu_ui_el
    //define file name
    fileName = "";
    if (manu_ui_el(1).value ==1) then
        fileName= "all_Zero";
    else
        //create a string from values
        for i= 2:size(NAMES,"*")
            fileName =
                fileName+NAMES(i)+strcat(string(VALUES(i)),'_')+"_"
        end
    end
    dt=getdate();
    //dircetory name in YYYY_MM_DD_HH
    saveDir =
        srcHome+"/img/"+string(dt(1))+"_"+string(dt(2))+"_"+string(dt(6))+"_"+string(dt(7))
    //create directory if it does not exist
```

```
    if(~isdir(saveDir))
        mkdir(saveDir);
    end
    //save file as eps
    xs2pdf(gcf(), saveDir+fileName+'.pdf');
endfunction

function selct_notch()
    global manu_ui_el VALUES notch_method_list
    f = VALUES(8)+1
    fnotch = VALUES(9)
    if notch_method_list.value ==1 then
        nH = VALUES(4)(1)
        nL = VALUES(4)(2)

    end
    for i=find(VALUES_STATUS==1)
        VALUES_STATUS(i) = 0;
    end
endfunction
```

B.6 GUI Programs Logic

```
// Kobayashi Labratory - Gunma University
// Ramin Khatami
// Feb 2015
//---cleaning and setting requirments--//
//clear concole screen
clc;
//clear variables in memory
clear;
//delete showing windows
xdel(winsid());
//raise scilab memory size to the max size
stacksize("max")
// show numbers in normal formats
format 'v';
// set present working directory as the srcHome for use in saving
    images, etc
srcHome = pwd();
```

```
//---bring external source files---//
exec( srcHome+'/add_func_v3.sci');
exec( srcHome+'/delsig_v3.sci');
exec( srcHome+'/dtc_v3.sci');
exec( srcHome+'/plotter_v3.sci');
exec( srcHome+'/gui_func_v3.sci');

global manu_ui_el VALUES notch_method_list
//---initial setting for the GUI program---//

//---dtc configuration--//
// array of string defining each value names to show in the program
NAMES = ["All Zero","Fsig", "A", "N","M", "φin", "Jitter",'Fsamp',
    'notch'];
// placholder for dtc configurations value

VALUES = list()

//all zero value
VALUES(1)=0;
//Fs : sinusoid signal frequency: Integer(ex. 1)
VALUES(2)=1;

//A: sinusoid signal Amplitude: Integer(ex. 1)
VALUES(3)=1;

//N: mutiplee of constant width for each pulse period: Integer
    array(ex. 5[for 0 in 1 bit dtc], 5[for 1 in 1 bit dtc])
VALUES(4)=[5 5]; //floor(4*rand(1,(2^VALUES(7).entries)-2))+5];

//M: mutiplee of constant width for each pulse high status length,
    it shoudl be lower than N: Integer array(ex. 0[for digital LOW
    in 1 bit dtc], 1[for digital HIGH in 1 bit dtc])
VALUES(5)=[1 1]//ones(1,2^VALUES(7).entries);

//φin: mutiplee of constant width for each pulse staritng high
    delay, it shoudl be lower than N+m: Integer array(ex. 0[for
    digital LOW in 1 bit dtc], 1[for digital HIGH in 1 bit dtc])
VALUES(6)=[0 0];

//random Jitter: similar to φin except φin varies randomely for
    defined status over 0: Integer array(ex. 0,0)
VALUES(7)=[0 0]; //no jitter
```

```
//[Hz]sampling frequency
VALUES(8)=2^10-1;

//notch: nothc location frequency: Integer (ex. 1025)
VALUES(9)=1025;

//if a function has been redifned do not throw error
funcprot(0);


//---draw graphical user interface---//
//get monitor screen size in pixel ex: [1,1,(width)1920, (height)
    1080]
screen_size = get(0,"screensize_px");
//screen size width
size_x = screen_size(3);
//screen size hight
size_y = screen_size(4);
//distance from top of the screen in gui app
top_offset = 0*size_y/100;
//distance between text inputs in the gui app
dist1 =6*size_y/100;

//define first window,; it is a place where dtc configurations are
    changed
//variable for text input boxes to show to the user and get the
    user inpout in gui app

manu_ui_el=[];
// variable for tag of text inputs
text_tag = [];

h_graph = figure( ...
"dockable"      , "off",...
"infobar_visible" , "off",...
"toolbar"       , "none",...
"menubar_visible" , "off",...
"menubar"       , "none",...
"default_axes"  , "off",...
"layout"        , "gridbag",...
"visible"       , "on");
//this window is as high as the screen size and 300px wide
```

```
h_graph.figure_size = [430 600];
//it takes a while for scilab to draw the window, there for wait
    100ms to go to next step
sleep(100)
// position config window to the left top side of the screen
h_graph.figure_position = [0 0];

//Create the constraint for nested uicontrols
c = createConstraints(...
    "gridbag",[1, 1, 1, 1], ...
    [1, 1], ...
    "both", "center", ...
    [0, 0], [50, 50]);
pos_y = 1;

c.grid= = [1 ,pos_y ,4,1];
manu_ui_el(1) = uicontrol(h_graph, ...
    "style","checkbox", ...
    "Min",0, ...
    "Max",1, ...
    "string",NAMES(1), ...
    "fontsize",16, ...
    "value", 1, ...
    "callback","update_plot(1);", ...
    "constraints", c);

//put control buttons label on screen
for i =2:size(NAMES,"*")
    pos_y = pos_y + 1;
    c.grid = [1,pos_y,1,1];
    text_tag(i) = uicontrol(h_graph, ...
    "style","text", ...
    "string", NAMES(i), ...
    "fontsize", 16, ...
    "constraints", c);

    c.grid = [2,pos_y,4,1];
    c.weight = [10,1];
    manu_ui_el(i) = uicontrol(h_graph, ...
    "style","edit", ...
    "Enable", "off", ...
    "string",strcat(string(VALUES(i)),', '), ...
    "fontsize",16, ...
```

```
    "callback","update_plot("+string(i)+");", ...
    "constraints", c);
end

//enable M text box to change output pulse width
manu_ui_el(3).enable = "on";
manu_ui_el(9).callback = "selct_notch()"

pos_y = pos_y + 1;
c.grid = [1,pos_y,5,1];
//notch creation mehtod chose
notch_method_list = uicontrol(h_graph, 'style','listbox',
    "callback" , "selct_notch()","Enable","off",'constraints', c);
set(notch_method_list,'string', "PCM|PPM|PWM|PRJ")
// fill the list
set(notch_method_list, 'value', 1);

//calculate
pos_y = pos_y + 1;
c.grid = [1,pos_y,1,1];

manu_ui_el(10) = uicontrol(h_graph, ...
    "style","pushbutton", ...
    "string", 'Compute', ...
    "Enable", "off", ...
    "fontsize",16, ...
    "callback","update_plot([1,2,3,4,5,6,7,8,9,10]);", ...
    "constraints", c);

//save graph
c.grid = [3,pos_y,1,1];
h_Save = uicontrol(h_graph, ...
    "style","pushbutton", ...
    "string", "Save", ...
    "fontsize",16, ...
    "callback","save_plot()", ...
    "constraints", c);

// stop simulation and exit
c.grid = [5,pos_y,1,1];
h_Stop = uicontrol(h_graph, ...
    "style","pushbutton", ...
    "string", "Stop", ...
```

```
    "fontsize",16, ...
    "callback","xdel(0)", ...
    "constraints", c);
//show what is going on

pos_y = pos_y +1;
c.grid = [1, pos_y, 5, 1];
manu_ui_el(11) = uicontrol(h_graph, ...
    "style","text", ...
    "string","Finished.", ...
    "fontsize",16, ...
    "constraints", c);

update_plot([1,2,3,4,5,6,7,8,9,10]);
```

Bibliography

[1] S. Ben Dhia, M. Ramadani, E. Sicard, Electromagnetic Compatibility of Integrated Circuits, Springer, 2006

[2] C. D. Hoekstra, "Frequency Modulation of System Clocks for EMI Reduction", Hewlett-Packard Journal, no.13, pp.1- 7 Aug. 1997.

[3] R. Schreier, G. C. Temes, *Understanding Delta-Sigma Data Converters*, IEEE Press (2005).

[4] S. Uemori, M. Ishii, H. Kobayashi, D. Hirabayashi, Y. Arakawa, Y. Doi, O. Kobayashi, T. Matsuura, K. Niitsu, Y. Yano, T. Gake, T. J. Yamaguchi, N. Takai, "Multi-bit Sigma-Delta TDC Architecture with Improved Linearity", Journal of Electronic Testing : Theory and Applications, Springer, Issue 6, pp.879-892 (Dec. 2013).

[5] H.G. Skinner, K. P. Slattery, "Why Spread Spectrum Clocking of Computing Devices is not Cheating", IEEE International Symposium on Electromagnetic Compatibility, vol.1, pp. 537-544, Montreal (Aug. 2001).

[6] C. D. Hoekstra, "Frequency Modulation of System Clocks for EMI Reduction", Hewlett-Packard Journal, no.13, pp.1-7 (Aug. 1997).

[7] "SSCG Spread Spectrum Clock Generator", Fujitsu Semiconductor Corp Technical Report, AD04-00041-4E (Oct. 2011).

[8] A. Kumar, P. Madaan, "Reducing EMI in Digital Systems Through Spread Spectrum Clock Generators", Cypress Semiconductor Corp Technical Report (Feb. 2011).

[9] M. Iijima, S. Miyata, Y. Miyazaki, K. Okada, T. Saitou, "Spread Spectrum Clock Generation Circuit, Jitter Generation Circuit and Semiconductor Device", U.S. Patent: US7095260 B2 (2006).

[10] I. Mori, Y. Yamada, S. A. Wibowo, M. Kono, H. Kobayashi, Y. Fujimura, N. Takai, T. Sugiyama, I. Fukai, N. Onishi, I. Takeda, J. Matsuda, "EMI Reduction by Spread-Spectrum Clocking in Digitally-Controlled DC-DC Converters", IEICE Trans. Fundamentals, volE92-A, no.4, pp.1004-1011 (April 2009).

[11] T. Daimon, H. Sadamura, T. Shindou, H. Kobayashi, M. Kono, T. Myono, T. Suzuki, S. Kawai, T. Iijima, "Spread-Spectrum Clocking in Switching Regulators for EMI Reduction", IEICE Trans. Fundamentals, vol. E86-A, no. 2, pp.381-386 (Feb. 2003).

[12] H. Sadamura, M. Namekata, M. Kono, H. Kobayashi, N. Ishikawa, "EMI Reduction Technique of Switching Regularities and Its Measurement Verification", IEICE Trans. vol. J86-C, no.11, pp.1169-1176 (Nov. 2003)

[13] K. Chuai, S. Aouini, G. W. Roberts, "A Low Cost ATE Phase Signal Generation Technique for Test Applications", IEEE International Test Conference , pp.1-10, Austin (Nov. 2010)

Index