A Novel Approach for Generating Digital Chirp Signals Using FPGA Technology for Synthetic Aperture Radar Applications

Von der Naturwissenschaftlich-Technischen Fakultät der
Universität Siegen
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

Von

M. Sc. Ashraf Samarah

1. Gutachter: Prof. Dr.-Ing. habil. Otmar Loffeld
2. Gutachter: Prof. Dr.-Ing. Dietmar Ehrhardt

Vorsitzender: Prof. Dr. rer. nat. Christoph Ruland

Tag der mündlichen Prüfung: 07.05.2012

Dedication

To my loving family

Acknowledgments

I would like to express my heartiest gratitude and deep appreciation to my advisor Prof. Dr. Otmar Loffeld for his consistent help, skillful guidance and attention that he devoted during the course of this work. Even with his eventful schedule, he always found time to counsel me on scientific matters and extended his caring support even while away from his office. I am indebted to Prof. Loffeld for all that I had the advantage to learn from him, and his numerous encouragements that contributed to the accomplishment of this work.

I ardently extend my warm thanks to Prof. Dr. Dietmar Ehrhardt for accepting to be my second supervisor, and would like to express my deep appreciations to him for his support.

On a special note I would like to thank Dr. Holger Nies and Dr. Stefan Knedlik for their guidance and advice during the course of this research. I thank them because they were always willing to help me and answer all of my inquiries. In addition, I will not forget to express my sincere gratitude to the SAR group for their support, reassuring communications and friendly words of encouragement. I must also thank Mr. Alaa Al Bashar for his assistance during this work.

I acknowledge and am thankful for the research scholarship for doctoral studies at the University of Siegen by the International Postgraduate Program (IPP), and am grateful for the support and facilities provided by ZESS (Center for Sensor Systems). I would also like to thank Mrs. Niet-Wunram who was always there to provide any help that I needed during my stay at ZESS.

Table of Contents

List of Figures

List of Tables

Dissertation Abstract

In this dissertation a novel digital chirp signal generator is proposed, analyzed, and realized. The new system generates digital chirp signals with the lowest level of spurious harmonic distortion, less memory size and low hardware complexity in comparison with other systems and techniques reported in the literature.

In this improved digital chirp generator the start frequency and phase can be controlled by the initial content of the counter and the accumulator. Furthermore, the sweep rate can also be controlled by means of location and size of the address lines. The proposed system is a hybrid of the digital chirp generator and the system using the methodology of the piecewise polynomial interpolation based on the direct digital frequency synthesizer. Moreover, an optimization technique is applied to enhance the performance of this chirp generator and to avoid the attenuation in the speed of its operations.

The new digital chirp generator uses a clock to trigger the counter (first integrator) and after that its output feeds the accumulator (second integrator), the decimal value of selected digital lines of the content of the accumulator, which represents the phase, is then used to calculate the value of the chirp sine using the interpolator. This interpolator uses predetermined interpolation coefficients to fit the sine wave from the calculated phase instead of using a predetermined waveform stored in a big size memory. This implies, that a smaller look-up table for sine and cosine functions is used in comparison with the previous techniques.

A new improved parallel processing technique is proposed in order to increase the bandwidth of the chirp signal up to 320 MHz and more based on the used level of the parallelism.

As a comparison with the look-up table method, the size of the ROM in the new method is reduced by a factor of more than 128 when using 12 address lines, and Spurious Free Dynamic Range (SFDR) reaching 100.9 dBc.

The system is realized using the Innovation Integration X5-TX platform with FPGA Xilinx VIRTEX-5 used with the parallel processing technique to generate a chirp signal with high bandwidth up to 320MHz using 200 MHz clock frequency.

Keywords: *Direct Digital Frequency Synthesizer, Digital Chirp signal, piecewise Parabolic-Polynomial Interpolation, Spurious harmonic distortion, Parallel processing technique, and FPGA.*

Kurzfassung

In dieser Dissertation wird ein neuartiger Chirp-Signal-Generator vorgeschlagen, analysiert und realisiert. Das neue System erzeugt digitale Chirp-Signale mit im Vergleich zu den aus der Literatur bekannten Systemen und Techniken geringsten harmonischen Verzerrungen, reduziertem Speicherbedarf und niedrigerem Hardware-Aufwand.

Bei diesem verbesserten digitalen Chirp-Signal-Generator kann die Startfrequenz und Phase durch den initialen Inhalt des Zählers und des Akkumulators gesteuert werden. Darüber hinaus kann auch die Steigung der Frequenzrampe durch die Anzahl und die Beschaltung der Adressleitungen gesteuert werden. Das vorgeschlagene System ist ein Hybrid aus einem digitalen Chirp-Generator und einem System, das auf einem direkten digitalen Frequenzsynthesizer basiert und die Methode der stückweisen polynomialen Interpolation verwendet. Des Weiteren wird eine Optimierungstechnik angewandt, um die Leistung des Chirp-Generators zu verbessern und eine Verlangsamung der Rechengeschwindigkeit zu vermeiden.

Der neue digitale Chirp-Generator verwendet einen, um einen Taktgenerator Zähler (ersten Integrator) zu takten, dessen Ausgang den Akkumulator (zweiten Integrator) speist. Der Dezimalwert ausgewählter digitaler Leitungen des Akkumulatorinhalts, welcher die Phase repräsentiert, wird dann verwendet, um den Wert des Chirp-Signals mittels des Interpolators zu berechnen. Dieser Interpolator verwendet im Voraus berechnete Interpolationskoeffizienten, um die Sinusfunktion aus der berechneten Phase zu erzeugen, anstatt eine vorgegebene, in einem großen Speicher abgelegte Wellenform zu verwenden. Dies hat zur Folge, dass im Vergleich zu den bisherigen Verfahren eine kleinere Look-Up-Tabelle für die Sinus- und Kosinusfunktionen verwendet wird.

Eine neue, verbesserte parallele Verarbeitungstechnik wird vorgeschlagen, um die Bandbreite des Chirp-Signals auf bis zu 320 MHz und mehr (je nach Grad der Parallelisierung) zu vergrößern.

Im Vergleich zur Methode mit Look-Up-Tabellen wird die Größe des Speichers durch die neue Methode um einen Faktor von mehr als 128 bei Verwendung von 12 Adressleitungen reduziert, und der störungsfreie dynamische Bereich (Spurious Free Dynamic Range, SFDR) erreicht einen Wert von 100,9 dBc.

Das System wurde realisiert auf der Plattform „Innovation Integration X5-TX“ mit einem Xilinx VIRTEX-5 FPGA. Verwendet wurde die parallele Verarbeitungstechnik, um mit einer Taktfrequenz von 200 MHz ein breitbandiges Chirp-Signal mit einer Bandbreite von bis zu 320 MHz zu erzeugen.

Schlagwörter: *Direct Digital Frequency Synthesizer, Digital Chirp signal, piecewise Parabolic-Polynomial Interpolation, Spurious harmonic distortion, ParallelVerarbeitungstechnik, and FPGA.*

Abbreviations and Symbols

Abbreviation	***Meaning***
BSP	Board Support Package
CDMA	Code Division Multiple Access
DACs	*Digital to Analog Converters*
DCLK	Data Clock
DDFS	*Direct Digital Frequency Synthesizer*
DDR	*Double Data Rate*
DLL	Delay Lock Loop
FIW	*Frequency Input word*
FPGA	*Field Programmable Gate Array*
LFM	*Linear Frequency Modulated*
LPF	*Low Pass Filter*
LSB	*Least Significant Bit*
LUT	*Look Up Table*
LVDS	Low Voltage Differential Signal
MSB	*Most Significant Bit*
PA	*Phase Accumulator*
PCI	Peripheral Component Interconnect
PLL	*Phase Looked Loop*
PM	*Phase Modulated*
PSAC	*phase-to-sine-amplitude converter*
QDR	Quad Data Rate
RA	*Radar Altimeter*
RF	Radio Frequency
ROM	*Read Only Memory*
SAR	*Synthetic Aperture Radar*
SBSRAM	*Synchronous Burst Static RAM*
SCMF	*Sine Cosine Mapping Function*
SFDR	*Spurious Free Dynamic Range*
SPW	*Start Phase Word*
SRW	*Sweep Rate Word*
TDMA	Time Division Multiple Access

VHDL	VHSIC *(Very High Speed Integrated Circuit)* Hardware Description Language
XMC	XDR (External Data Representation) Memory Controller

Symbol	***Meaning***
L	*Word length of the phase accumulator*
μ	*fractional delay*
$h_I(t)$	*finite-duration impulse response of a fictitious*
$x(m)$	*sequence of signal samples*
V_k	*Interpolation coefficients*
t_p	*one period of system clock*
$\beta(T)$	*The phase stored in the accumulator*
$S_{\min}$	*lowest required sweep rate*
$t_{\max}$	*the maximum sweep time*
f_s	*Sampling frequency*
k_n	*Constant used to adjust the unit of each series*
b_n	*Fixed coefficients numbers*
f_0	*Starting frequency*
f_{clk}	*Clock frequency*
ϕ_0	*Starting phase*
$f_{\max}$	*Higher desired frequency component*
θ	*Phase accumulator*
θ'	*Normalize phase accumulator*
$f_{\min}$	*The minimum synthesizable frequency*
f_{out}	*The output frequency*
$\alpha,\ S$	*chirp rate*
T_p	*chirp pulse duration*
ω_0	*Baseband bandwidth*

w_1	*start frequency*
$w(t)$	*The instantaneous frequency of a linearly swept signal*
$\phi(t)$	*instantaneous phase*
ϕ_0	*Start phase*
p	*Least significant bits of the phase accumulator*
K	*Most significant bits of the phase accumulator*
x	*The phase angle, represented as a fraction in the interval [0, 1)*
r	*is the degree of the polynomial approximation,* $r \geq 1$
s	*is the number of piecewise continuous polynomial segments,* $s \geq 1$
c_{ki}	*are the polynomial coefficients*
x_k	*is the lower bound of the kth piecewise continuous segment*
$y(kT_i)$	*The interpolants*
$y(t)$	*The signal after re-sampling*
y_k, m_k, *and* p_k	*Interpolation coefficients*
c	*The word length of the variable* γ
AC(T)	*The Content of the accumulator*
AC_o	*Initial content of the accumulator*
CO(T)	*The content counter*
CO_o	*Initial content of the counter*
$\text{mod}_{2\pi}(x)$	*Modulus* 2π *for the phase x, it reduces x to the range* $[0,2\pi]$

1 Introduction

Linear Frequency Modulated (LFM), chirp or sweep signals are defined as sinusoidal signals whose frequency increases or decreases over a certain amount of time, and both the phase and frequency must be specified for all time. These signals are required for earth remote sensing (Synthetic Aperture Radar- SAR, Radar Altimeter- RA), planetary remote sensing, and are required for several applications like target velocity estimation [1], phase coding of sweep signals in communication applications, system characterization, radar (especially in Synthetic Aperture Radar), sonar, acoustic digital imaging, and the determination of system response with network analyzers [2, 3]. A description of the mathematical formulation for these types of signals can be found in chapter 3.

There are only a few methods applicable for generating digital chirp signals. One method is to store it in a predetermined waveform high-speed digital memory. The main limitation of this method is the time-bandwidth product. Another approach is based on the calculation of binary words [4], corresponding to the analytical expression of the quadratic phase of a sampled linear sweep. These calculations must be performed in real time; as a drawback, the maximum attainable frequency will only be in the kilohertz range. For these reasons, Pedersen presented his architecture to circumvent these disadvantages [5], which are based on real time digital evaluation of the phase of the desired sweep signal and then reading its value from a look-up table (LUT) of length L ($L = 2^K$). Other constraints which appeared as main disadvantages of Pedersen chirp generator are the high level of spurious harmonic distortion and the big size of the memory. The detailed description of this generator is given in chapter 3.1.

Chapter 3.2 reports another chirp generator in which fractional addressing is utilized to reduce the level of the harmonic distortion. The major disadvantage of this generator is that it requires the implementation of two independent chirp generators and then evaluating the phase difference between the two generated sweep signals. The hardware requirements, therefore, make its implementation costly.

Our proposed system provides an extremely low level of spurious harmonic distortion, and minimizes the size of the generator's memory. At the same time the architecture is a hybrid of the digital sweep generator and the system using piecewise polynomial interpolation based on direct digital frequency synthesizer (DDFS). The interpolator uses predetermined interpolation coefficients to fit the sine wave from the calculated phase instead of using a predetermined waveform which is stored in a big sized memory. This implies that a smaller look-up table for the sine and cosine functions is used compared to existing architectures with minimum hardware overhead, and the computation of the sinusoidal values is performed by a piecewise parabolic and extended by a piecewise-polynomial interpolation (approximation) structure. Therefore, only interpolation coefficients are stored in the memory. Chapter 4 shows the analysis and formulation of the direct digital frequency synthesizer as reported in the literature, also included in chapter 4 is a short view of ROM compression techniques.

Chapter 5 documents the definition of the problem and introduction to the solution, while chapter 6 presents the complete solution, the methodology of implementing the new architecture, simulation and theoretical implementation.

The theoretical work is realized and put to practical implementation by applying the approach and generating the digital chirp signal on the "Innovative X5-TX" platform, which is manufactured by Xilinx.

Chapter 7 documents the hardware implementation and the generation of the digital chirp signal based on the proposed method. In the same chapter a smart parallel processing technique is completely derived, implemented and exploited in order to increase the generated chirp signals' bandwidth up to 320 MHz by means of 200 MHz clock frequency for the applications of Synthetic Aperture Radar (SAR) using a hybrid system based on direct digital frequency synthesizer. The conclusion of the acquired results and published contributions are shown in chapter 8.

2 Chirp Signals

To begin, let us explain what the chirp signal is, and discuss why we want to use it.

The chirp signal was given its name because it sounds like the chirp of a bird when played through a speaker. The word chirp in the dictionary means: a short, high pitched sound, such as that made by a small bird or insect.

Chirp signals are also known as Linear Frequency Modulated (LFM) signals, and are angle modulated sweeping signals. These signals sweep through the entire frequency bandwidth B[Hz] from one end to the other in form of a sinusoidal waveform of a constant amplitude and within a certain time T[s]. If this sequence of frequencies is swept from the lowest to the highest frequency limit, it is called an *Up chirp*, while in the opposite direction it is a *Down chirp*.

A chirp signal, in mathematical term, is defined as:

$$s(t) = a(t)\exp(j\beta t + j\alpha t^2) \tag{2.1}$$

Where $a(t) = 1 \; for \; 0 \le t \le t_p$ and zero otherwise; t_p is the chirp pulse duration [6]. For an efficient explanation we will consider that both α and β are positive quantities; and α or 2α is the chirp rate, and the chirp pulse is a phase modulated (PM) signal.

We can define the instantaneous frequency of the chirp pulse within the interval $0 \le t \le t_p$ by differentiating the phase of the chirp signal in (2.1) with respect to the time; this is shown as follows:

$$\begin{aligned}\omega_{ip}(t) &= \frac{d}{dt}(\beta t + \alpha t^2) \\ &= \beta + 2\alpha t\end{aligned} \tag{2.2}$$

Note that with $\alpha > 0$, the instantaneous frequency is an increasing function of time; in this case the chirp is called upsweep. The minimum value of $\omega_{ip}(t)$, is β, and its maximum is $\beta + 2\alpha t_p$. The spectral support band of a chirp signal is approximately bounded by these minimum and maximum values, where $|S(f)| > 0$ for $\omega \in [\beta, \beta + 2\alpha t_p]$. Thus, the carrier (midfrequency) of a chirp pulse is $\omega_c = \beta + \alpha t_p$, where its baseband bandwidth is $\pm\omega_0 = \pm\alpha t_p$.

Why do we want these kinds of signals particularly for radar applications?

The answer of this question is based on two requirements. Firstly, we want to consider how a radar system operates. A short burst of radio frequency energy is emitted from a directional antenna and then targets aircrafts and other items reflect some of this energy back to a radio receiver, which is located next to the transmitter. Since radio waves travel at a constant velocity, the elapsed time between the transmitted and received signals provides the distance to the target. This brings up the first requirement for the pulse: it needs to be as short as possible. For example, a 1 microsecond pulse provides a radio burst about 300 meters long. This means that the distance information we obtain with the system will have a resolution of about the same length. If we want better distance resolution, we need a shorter pulse [7].

The second requirement comes as the following: if we want to detect and measure objects farther away, more energy is needed in the pulse. Unluckily, *more energy* and *shorter pulse* are conflicting requirements. The electrical power needed to supply a pulse is equal to the energy of the pulse divided by the pulse length. Requiring both more energy and a shorter pulse makes electrical power handling a limiting factor in the system. The output stage of a radio transmitter can only handle a certain peak power without destroying itself.

Chirp signals present a way of breaking this limitation. Earlier than the impulse reaches the last point of the radio transmitter, it is passed through a chirp system. Instead of bouncing an impulse off the target aircraft, a chirp signal is used. After the chirp echo is received, the signal is passed through a matched filter, and restores the signal to an impulse. This allows the portions of the system that measure distance to see short pulses, while the power handling circuits see long duration signals. This type of wave shaping in form of matched filtering is a fundamental part of modern radar systems.

From a theoretical point of view, chirp signals provide an astonishing number of advantages. They substantiate the following ideal features of a fundamental nature in communications engineering:

- They have a quasi ideal rectangular spectrum to utilize the channel's capacity and to offer an optimal lowest spectral power density compared to all other existing transmission signals.
- They are programmable with respect to processing gain, which means that it is possible to achieve determinable distances in ranging while at the same time suppress adaptively disturbances and noise.
- All three main modulation modes can be applied at the same time, each of which contributes specific physical parameters for optimal transmission properties, as follows:

  FM (Frequency modulation) contributes a robust transmission by a big time bandwidth product and also guarantees an ideal spectrum shaping and processing gain.

AM (Amplitude modulation) which is generated by the transformation of the chirp signal, contributes the ideal spectrum as well as an ideal envelope function. The kind of this function is a *sinc* function with the shortest duration possible at any given bandwidth, in order to use time effectively. This means if we apply the AM technology to chirp signal which is generated by the transformation of the chirp signal in order to change the rectangular amplitude of chirp signal by using a given window function practically. Therefore, it is possible to achieve both an ideal rectangular spectrum and an ideal envelop function. If this amplitude modulation is proper, then this envelop function can be a sinc function.

PM (Phase modulation) contributes the capability to transfer single bits in BPSK (Binary phase-shift keying), QPSK (Quadrature phase-shift keying) or a higher multiphase angle modulation mode. It allows the transmission of bits by a combination of multi chirp modulation [8].

- They allow a high resolution on time axis and are, therefore, the best suited for ranging.
- They enable systems that provide a very short latency by asynchronously working correlative transmission systems.
- Chirp signals prove the ability to superpose these long signals to allow the data rate and bit energy to vary adaptively or to generate multi chirps in different combinations, which results in other advantages.
- They can be processed in an analogue way to realize low power solutions.
- Chirp Spread Spectrum (CSS) signals are resistive against overloading if chirps do not overlap.

- Chirp Spread Spectrum (CSS) signals are approximately resistive against multi paths effects.
- Chirp Spread Spectrum (CSS) signals do not fail because of Doppler shifting, but they provide wrong distances!
- Chirp Spread Spectrum (CSS) signals can be processed asynchronously and present advantages in comparison to other systems which need to be synchronized. This facility lowers latency and improves coexistence ability.

3 State of the Art Methods for Generating Digital Chirp Signals

Several applications require linear chirp (sweep) signals, i.e., a sweep where both phase and frequency must be specified for all times. These applications are Synthetic Aperture Radar (SAR), systems characterization, sonar, acoustic imaging, etc [9-12]. Other usages may lie in phase coding of sweep signals in communication applications. A well known use of digital chirp signals is the determination of the systems response with network analyzers. As the excitation signal itself is used for the synchronous demodulation, and a coherent measurement is in fact performed without actually knowing the phase.

Digital chirp generators exhibit the advantages of digital techniques, i.e., stability, flexibility and low cost. In addition, the parameters of a digital by generated sinusoidal are easy to control. In this chapter, we will discuss as a literature survey the generation of digital chirp signals using different methods and techniques.

3.1 Pedersen's Chirp Generator

In 1990, Pedersen [5] proposed his method in order to generate digital chirp signals, based on real time digital evaluation of the phase of the desired swept signal, and then reading its value from a look-up table (LUT) of length $L(L = 2^K)$. The methodology of this technique is based on a digital approach, which minimizes the restrictions of previously known techniques, i.e., limited frequency range or limited signal duration. This digital chirp system performs the following functions:

a. Real time generation of the phase of a linearly swept signal.

b. Extraction of mod (2π) from the total phase.

c. Generation of the desired sine or cosine swept signals by means of a LUT.

When the phase of a linearly swept signal is produced, both start phase and start frequency are specified. The sweep rate can be varied over a wide range in multiples of two. However, arbitrary sweep rates can be obtained by allowing the clock frequency to be variable. Moreover, by employing two sweep modules the in-phase and quadratic-phase components of a chirp signal may be produced. In addition, multiple coherent digital chirp signals can be produced by driving several sweep modules from the same clock.

In the following section, we will show the mathematical description, which required and needed to understand the novelty of the approach presented later.

3.1.1 Mathematical Description of the Chirp Generator

As a mathematical description for the chirp signal, we will consider $w(t)$ as the instantaneous frequency of a linearly swept signal with start frequency of w_1 and sweep rate of S (Hz/s). The description of $w(t)$ is given as:

$$w(t) = 2\pi St + w_1 \qquad (3.1)$$

The corresponding instantaneous phase is obtained by integrating (3.1) :

$$\phi(t) = \pi St^2 + w_1 t + \phi_0 \qquad t \geq 0 \qquad (3.2)$$

$\phi(t)$ contains a quadratic term corresponding to the linearly varying frequency (S is the sweep rate), a linear term corresponding to the start frequency, and a constant term ϕ_0 corresponding to the start phase.

The generation of the coherent digital chirp signal is achieved by applying a double integration for the constant sweep rate in the time domain in order to obtain (3.1) and (3.2), then extracting of mod (2π) from the total phase, after that the generation of the desired sine or cosine swept signals by means of the look-up table will take place.

Figure 3.1 illustrates the functional block diagram of Pedersen's chirp generator, which produces a quadratic phase function by digitally integrating a clock signal twice.

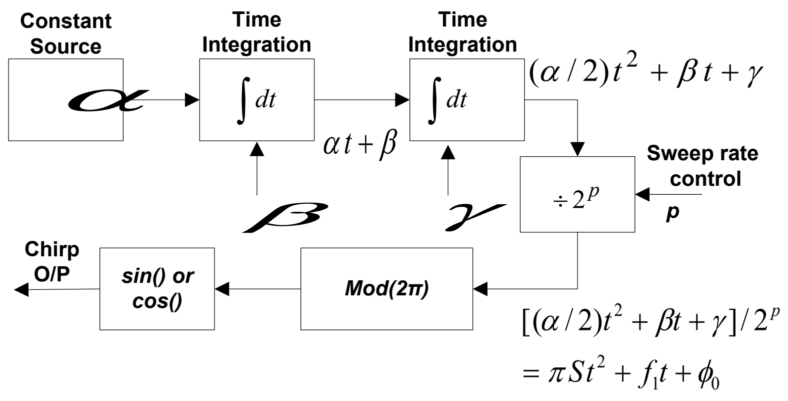


Figure 3.1: Functional block diagram of the digital chirp generator

A constant value of α is applied to the first integrator with the initial value of α. The output $(\alpha t+\beta)$ is applied to the second integrator with the initial value equal to γ. Then the result of the second integration is a quadratic function and equal to $(\alpha/2)t^2+\beta t+\gamma$. The sweep rate is reduced by factors of 2, because of dividing the second integrator's output by 2^P , where P is a specified integer. This gives a phase function equal to $[(\alpha/2)t^2+\beta t+\gamma]/2^P$. If $\alpha/2^P=2\pi$, $\beta/2^P=f_1$, and $\gamma/2^P=\phi_0$, with a comparison we can achieve the following equation:

$$\phi(t) = [(\alpha/2)t^2 + \beta t + \gamma]/2^p \tag{3.3}$$

This equation is identical to (3.2). Now, taking the 2π modulus of $\phi(t)$ and performing a sine or cosine operation will generate a linear sweep in digital form in which the start phase and the start frequency have been preset.

The actual digital implementation is shown in Figure 3.2, where the two integrations are performed by means of a counter and an accumulator. Both of them (the counter and the accumulator) should typically be 32 to 36-bits wide.

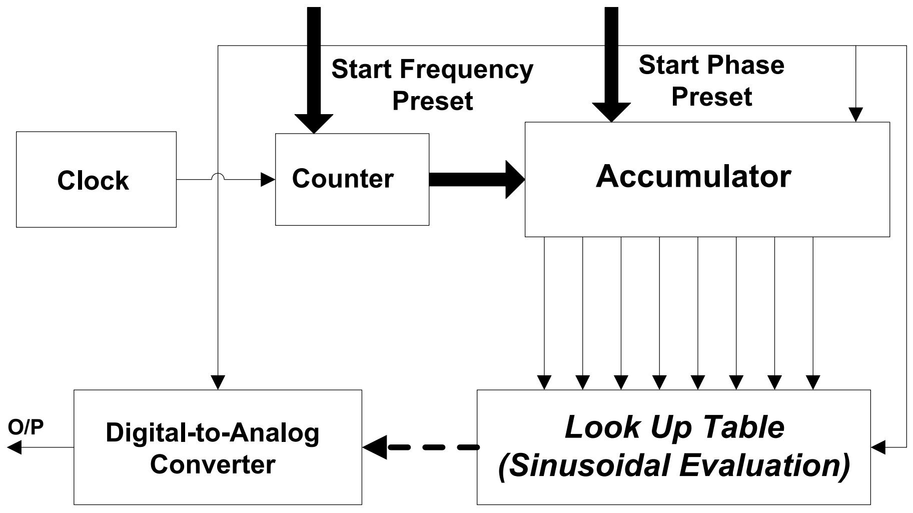


Figure 3.2**:** Simplified Block diagram of the implementation of the digital sweep generator

Presetting the counter and the accumulator permits a specified start frequency and start phase, respectively. A subset (typically 8 bits) of neighboring outputs from the accumulator constitutes the address lines to a look-up table (high speed RAM or PROM) in which the desired function is stored, typically a sine or a cosine function. However, rectangular or triangular waveforms may be stored instead. The output of the look-up table is sent to a high speed digital-to-analog converter.

The sweep rate is determined by selecting the output lines from the accumulator to be used as address lines for the look-up table. Selecting address lines that contain the more significant bits of the accumulator output lines will produce a low sweep rate, while output lines containing the least significant bits will produce high sweep rate.

The frequency of the clock in Figure 3.2 determines the highest frequency that the digital sweep generator can generate. If one assumes that the minimum number of samples required determining one cycle of the waveform is 5, and a maximum frequency of 10 MHz is required, then the clock frequency should be at least 50 MHz.

A low-pass (LP) filter is placed after the digital-to-analog converter to remove the periodicity of the spectrum and the sampling effects in the time domain. The cutoff frequency for the LP filter should be chosen in order that the signal will not suffer any amplitude or phase effects due to the LP filter.

In order to understand the way of operation in this generator, the 8 least significant bits from the accumulator are the address lines to the look-up table memory. Therefore, the encoded 8-bit output from the accumulator can have values from 0 to 255 and is designed to represent a phase between 0 and 2π radians where 0 corresponds to 0 radian and 255 corresponds to $(255/256)\times 2\pi$ radians.

As the binary value contained in the 8 least significant bits of the accumulator increases quadratically towards (11111111), the output of the PROM goes through the values of a full sine cycle. When the first "overflow" from these 8 bits into the 9th bit of the accumulator occurs, one cycle of the sine wave has been completed. The frequency of the sine wave at a given point in time is determined by the number of clock pulses required to produce the "overflow." With every repeated overflow, the selected set of accumulator output lines has cycled through a phase change of 2π. The phase changes in a quadratic fashion, and, by requiring at least 5 samples per cycle, the maximum frequency of the sweep signal has been reached when only 5 clock pulses are needed to increase the content of the 8 bits from one “overflow” to the next.

When the entered lines to the look-up table (sinusoidal evaluation) are the lines 0-7 instead of the lines 1-8, then the time to generate an “overflow” is doubled, or, consistently, the sweep rate is abridged by the factor two. The slowest sweep rate is achieved when the 8 most significant bits from the accumulator are used as address lines. At any given clock frequency and sweep rate, the accumulator output lines which are less significant than the 8 lines to the look-up table represent accuracy that is not utilized; in other words, some of these less significant lines would be used if the word-length in the look-up table were to be increased. The address lines greater than the 8 bits to the look-up table represent an unused, or unnecessary, part of the accumulator. Thus at the chosen sweep rate, there would be no change in the system performance if the accumulator were reduced in bit width until the 8-bit output lines constituted the most significant bits.

As shown below, the width (in bits) of the counter and the accumulator determines the minimum sweep rate achievable. Since the maximum frequency is specified by the clock frequency, changing the start frequency from dc to some higher frequency and assuming a fixed minimum sweep rate will produce a reduced sweep time.

The counter must never overflow during the time of one sweep. For example, a 32-bit wide counter is needed in order to have sweep duration of 30 seconds when a 100-MHz clock is used. A 32-bit wide counter will, as far as the counter is concerned, allow a 480 seconds sweep. However, if the accumulator is also kept at a width of 36 bits, it will limit the duration of the sweep to about 120 s, which is nonetheless, sufficient for most applications [5].

3.1.2 Derivations of the parameters for the digital chirp generator

The parameters of the chirp signal have been defined in the previous section. A unit of system time t_p is specified as one period of system clock. A new system time function then is defined as $T = t / t_p$ = time measured in clock cycled.

The basis for the quantitative analysis is the quadratic look-up table. Therefore, a new phase function $\theta(t)$ is defined as:

$$\theta(t) = \mathrm{mod}_{2\pi}\{\phi(t)\} = \phi(t) - 2\pi n \qquad (3.4)$$

where n is an integer such that $0 \le \theta(t) \le 2\pi$.

The three basic components of the digital system to be considered are the clock, the counter, and the accumulator, as shown in Figure 3.2. The clock produces a pulse every t_p seconds or equivalently, the clock output is 1 when expressed as a function of the system time T. Since the counter is incremented by 1 every t_p seconds, it is equivalent to an integrator having a constant input.

As the counter is preset with the value X_0 the output, $X(T)$ is given by:

$$X(T) = \int_0^T dT + X_0 = T + X_0 \tag{3.5}$$

Where $X(T)$ is the counter content, X_0 is the initial content of counter, and at every clock cycle, the accumulator will add the current counter output to the accumulator value of the previous clock cycle:

$$Y(T) = X(T) + Y(T-1) \tag{3.6}$$

$Y(T)$ is the accumulator content and $Y(T)$ in (3.6) may be considered the result of a differentiation of $Y(T)$, as shown below, leading to an expression for $Y(T)$.

$$\begin{aligned} \frac{dY(T)}{dT} &= \frac{Y(T) - Y(T-1)}{1} = X(T) \\ dY(T) &\cong X(T)dT \end{aligned} \tag{3.7}$$

$$\begin{aligned} Y(T) &= \int_0^E X(T)dT = \int_0^T (T + X_0)dT \\ &= T^2/2 + X_0T + Y_0 \end{aligned} \tag{3.8}$$

The content of the accumulator $Y(T)$ can be considered as a phase value. However, since only a subset, k of the output lines from the accumulator is sent to the look-up table, the phase of the sweep signal in general is not equal to the phase stored in the accumulator. In the following derivations, $\beta(T)$ represents the phase, stored in the accumulator.

It is important to note that the k least significant bits of the accumulator can contain a phase range of $0 - 2\pi$ radians $\beta(T)$ is defined from the numerical value of the content of the accumulator as follows:

$$\beta(T) = 2\pi \frac{Y(T)}{2^k} = 2\pi \left(\frac{T^2/2 + X_0 T + Y_0}{2^k} \right) \tag{3.9}$$

If the accumulator is n-bits wide, the maximum phase value that can be stored is $2\pi(2^{n-k})$ radians. Therefore, equation (3.9) is valid as long as $0 \le \beta(T) \le 2\pi(2^{n-k})$ rad. For the implementation of the sweep system, $k = 8$ has been used. The values stored in the look-up table correspond to only one sine (or cosine) wave, and as seen from (3.3) the phase of the sweep signal is, therefore, equal to $\theta(T) = \mathrm{mod}_{2\pi}\{\phi(T)\}$.

The address size, k of the sine look-up table determines the accuracy of the phase data to the look-up table. Of course, k cannot exceed the size of the accumulator, and the k bits typically constitute only a small fraction of the total number of accumulator output lines. The selection of the k bits used as address lines is defined by the parameter, p as illustrated in Figure (3.3), where p is the position of the least significant bit of the accumulator output which is used as an address line into the look-up table. The smallest value that p can assume is $p = 0$. The k input lines into the sine look-up table are then bits $[p, p+1, p+2, ..., p+k-1]$. Since the k input lines are shifted to the left by p bits, relative to the least significant bit of the accumulator, $\theta(T)$ is equal to $\mathrm{mod}\,2\pi\{\beta(T)/2^k\}$. From (3.8) one finds:

$$\theta(T) = \mathrm{mod}_{2\pi}\left\{ 2\pi \frac{T^2/2 + X_0 T + Y_0}{2^k} \right\} \tag{3.10}$$

The unused accumulator bits $0, 1, 2, \dots, \mathrm{p} - 1$ represent the accuracy and they are not utilized. As seen in Figure 3.3 below, the bits $[p + k, p + k + 1, p + k + 2, \dots, n - 1]$ are not used either; because these bits represent a count of the number of completed cycles, which is unrelated to the signal generation.

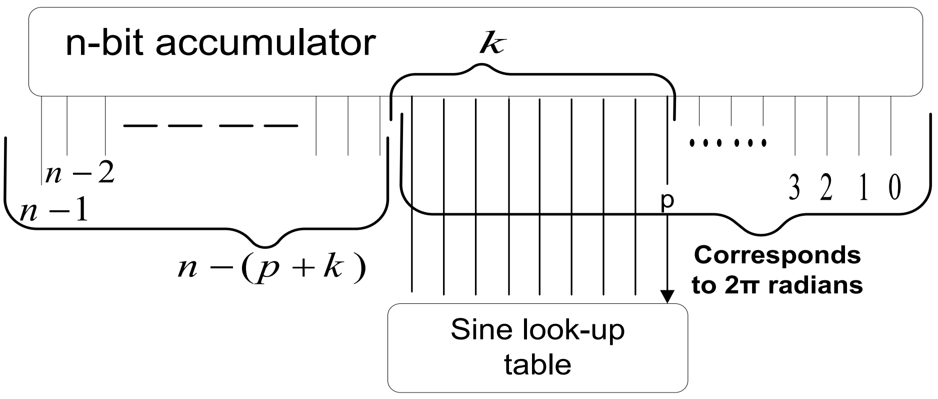


Figure 3.3**:** Definition of the output lines from the accumulator

The sine look-up table has the size of $2^k \times q$, where q is the number of bits used to represent the sine data in the PROM table, typically $k = q$. Since $\sin[\theta(T)]$ varies from -1 to 1 and the data in the look-up table vary from 0 to $2^q - 1$, the binary words from the look-up table to the analog-to-digital converter take the form of:

$$V_{out}(T) = \frac{(2^q - 1)\cdot(\cos[\theta(T)] + 1)}{2} \tag{3.11}$$

Recalling that $T = t/t_p$ and θ can be converted from the system time, then $\theta(t)$ will be as follows:

$$\theta(t) = \mathrm{mod}_{2\pi}\left\{\frac{(t^2/2t_p^{\ 2}) + X_0(t/t_p) + Y_0}{2^{p+k}} 2\pi\right\} \tag{3.12}$$

By restating the phase function of a linear sweep $\phi(t)$ and the relationship between $\phi(t)$ and $\theta(t)$, as given in equations (3.2) and (3.4):

$$\phi(t) = \pi S t^2 + f_1 t + \phi_0 \tag{3.2}$$

$$\theta(t) = \mathrm{mod}_{2\pi}\{\phi(t)\} = \phi(t) - 2\pi n \tag{3.4}$$

It can be seen that the phase function in (3.12) is the mod 2π of the phase function in (3.2). However, the time dependence of (3.12) and (3.2) is the same.

By comparing (3.12) and (3.2), the expressions for the variables in (3.2) can be shown as the following:

$$\begin{aligned} S &= \left[(2^{p+k})t_p{}^2\right]^{-1} \quad Hz/s \\ &= 2\pi\left[(2^{p+k})\cdot t_p^2\right] \quad rad/s^2 \end{aligned} \qquad (3.13)$$

$$f_0 = X_0[(2^{p+k})t_p{}^2]^{-1} \quad rad \qquad (3.14)$$

$$w_0 = 2\pi\cdot X_0[(2^{p+k})t_p{}^2]^{-1} \quad rad/s \qquad (3.15)$$

Another form for the equations (3.13) and (3.14), respectively is shown as follows:

$$S = (f_{clk})^2/2^{p+k} \quad Hz/s \qquad (3.16)$$

$$f_0 = (X_0\cdot f_{clk})/2^{p+k} \quad Hz \qquad (3.17)$$

The other important values are given in(3.18) and(3.19)

$$\phi_0 = \frac{2\pi\cdot Y_0}{2^{p+k}} \quad rad \qquad (3.18)$$

$$f(t) = St + f_0 = t[(2^{p+k})t_p{}^2]^{-1} + X_0[(2^{p+k})t_p]^{-1} \quad Hz \qquad (3.19)$$

The sweep parameters in terms of the digital systems parameters are given by equations (3.13) -(3.19). These are the desired results of the quantitative analysis.

The sweep rate, given in (3.16), allows only variations in steps of two. However, it is clearly possible to select the three signal parameters S, f_0 and ϕ_0 freely when f_{clk}, CO_0, and AC_0 are considered as dependent variables. The difference in this approach is that f_{clk} has been made a dependent variable. Thus based on (3.16) to(3.18) the following expressions for f_{clk}, CO_0 and AC_0 can be found:

$$f_{clk} = \sqrt{2^{p+k} \cdot S} \tag{3.20}$$

$$X_0 = \frac{f_0}{f_{clk}} 2^{p+k} = \frac{f_0}{\sqrt{S}} \sqrt{2^{p+k}} \tag{3.21}$$

$$Y_0 = \frac{\varphi_0}{2\pi} 2^{p+k} \tag{3.22}$$

The expressions in (3.21) and (3.22) can be used to determine X_0 and Y_0 for the desired start frequency, f_0 and start phase, ϕ_0 when the clock frequency is fixed. The stop frequency f_{stop} may be implemented by having a comparator generate a "clock disable" pulse when the content of the counter has reached a value, $X_{\max}$ corresponding to f_{stop}. It can be seen that if the initial counter contents, X_0 corresponds to f_0, then in a linear fashion $X_{\max}$ corresponds to f_{stop}:

$$X_{\max} = \frac{f_{stop}}{f_{clk}} 2^{p+k} = \frac{f_{stop}}{\sqrt{S}} \sqrt{s^{p+k}} \tag{3.23}$$

3.1.3 Bit number requirements of components

From the sampling theorem, the Nyquist sampling rate, equal to twice the highest desired frequency component, $f_{\max}$ determines the minimum sampling rate needed to accurately reproduce a waveform. However, in a practical implementation, it is customary to choose a sampling rate 5 times higher than the highest frequency component. Since one sample per system clock period will be obtained:

$$\begin{aligned} f_{clk} &= 1/t_p = 5f_{\max} \\ f_{\max} &= \frac{f_{clk}}{5} = \frac{1}{5t_p} \end{aligned} \tag{3.24}$$

The highest clock frequency where the system can operate is the only constraint on maximum frequency. The lowest required sweep rate, $S_{\min}$ determines the necessary bit number of the accumulator and the counter. Assuming that the sweep starts from dc, $S_{\min}$ may be described in terms of $f_{\max}$ and the maximum sweep time, $t_{\max}$ as follows:

$$\begin{aligned} S_{\min} &= \frac{\Delta f}{\Delta t} = \frac{(f_{\max} - 0)}{t_{\max}} \\ &= f_{\max}/t_{\max} = f_{clk}/5t_{\max} = 1/5t_p t_{\max} \end{aligned} \tag{3.25}$$

When $S = S_{\min}$ and $p = p_{\max}$.

The necessary bit number of the accumulator is n and it is given by:

$$n = p_{\max} + k \tag{3.26}$$

As will be shown in the results later, the main disadvantages of Pedersen's chirp generator are the high level of spurious harmonic distortion and the large memory size. Note that, the spurious harmonic distortion appeared as a result of discarding the least significant bits of the phase accumulator.

3.2 Digital Chirp Generator based on the fractional bits technique

This section demonstrates a technique based on real-time evaluation of the chirp signal, with the phase equal to the phase difference between two independent chirp signals and two different sweep rates S_1 and S_2 [13]. The chirp signals with sweep rates S_1 and S_2 are generated by two separate digital LUT chirp generators.

Generally, digital generators have the disadvantage of increased harmonic distortion caused by different numerical errors, the most significant of which comes from implementing the fractional addressing to improve the frequency resolution. Fractional addressing implies that the LUT is addressed by the bits of both: the fractional and integer parts of the address register [14, 15]. However, in Pedersen's chirp generator, the fractional part is not used; thus, the produced signals have spurious harmonic distortion [14, 15]. This problem can be modeled as nonuniform sampling (i.e. a nonuniform number of samples per period of the chirp signal, and hence, non-periodic chirp signals). One method of reducing the distortion caused by nonuniform sampling [14, 15] is by linear interpolation of the missing samples [16, 17].

In the fractional bits technique, this is addressed by a sample pointer $m = I + F$ to LUT with length L, where I and F are the integer and fractional parts, respectively. This chirp generator uses the trigonometric identity method, which is based on expanding $sin[(2\pi/L)(I+F)]$ as follows:

$$\sin\left[\frac{2\pi}{L}(I+F)\right] = \sin\left(\frac{2\pi}{L}I\right)\cos\left(\frac{2\pi}{L}F\right) + \cos\left(\frac{2\pi}{L}I\right)\sin\left(\frac{2\pi}{L}F\right) \tag{3.27}$$

The value of the total phase of $s(t) = A\cos(\pi S t^2 + 2\pi f_0 t + \phi_0)$, where S is the sweep rate, f_0 as a start frequency and ϕ_0 is the initial phase, can be considered as the sample pointer

m with I and F being the integer part and fractional part of the total phase, respectively. The integer-fraction technique utilizes two digital chirp generators. Each digital chirp generator can be considered as a Pedersen chirp generator. The analysis of this structure starts when the content of the two accumulators at time t is given by:

$$AC_i\left(\frac{t}{t_p}\right) = \frac{t^2}{2t_p^2} + CO_{oi}\frac{t}{t_p} + AC_{oi} \qquad i = 1,2 \qquad (3.28)$$

Where CO_{oi} and AC_{oi} are the start frequency and start phase of the i^{th} digital chirp generator in Figure 3.4.

Since a subset, K_i, of the output lines from the accumulators are sent to the look-up tables and the K_i input lines into the look-up tables are shifted to the left by p_i bits, the total phase of the i^{th} chirp signal can be shown in the following by:

$$\theta_i\left(\frac{t}{t_p}\right) = \frac{AC_i\left(\frac{t}{t_p}\right)}{2^{(p_i+K_i)}} \qquad i = 1,2 \qquad (3.29)$$

Where $L_i = 2^{K_i}$. The values of the total phase in (3.29) can be considered as the sample pointer $m_i = I_i + F_i$, i=1,2 with I_i and F_i being the i^{th} chirp generator integer and fractional parts. The chirp or sweep signals from LUT1 and LUT2 are represented by the following equations:

$$\sin\left(\frac{2\pi}{L_1}m_1\right), \qquad \cos\left(\frac{2\pi}{L_1}m_1\right) \qquad (3.30)$$

$$\sin\left(\frac{2\pi}{L_2}m_2\right), \qquad \cos\left(\frac{2\pi}{L_2}m_2\right) \qquad (3.31)$$

The sweep rates generated by both digital chirp generators are given by:

$$S_1 = \frac{f_{clk}^2}{2^{p_1} L_1} \quad Hz/s \tag{3.32}$$

$$S_{21} = \frac{f_{clk}^2}{2^{p_2} L_2} \quad Hz/s \tag{3.33}$$

Modulation techniques can be used as in Figure 3.4 to obtain the phase difference between two sweep functions as obtained by:

$$\sin\left[\frac{2\pi}{L_1}m_1 - \frac{2\pi}{L_2}m_2\right] = \sin\left(\frac{2\pi}{L_1}m_1\right)\cos\left(\frac{2\pi}{L_2}m_2\right) - \cos\left(\frac{2\pi}{L_1}m_1\right)\sin\left(\frac{2\pi}{L_2}m_2\right) \tag{3.34}$$

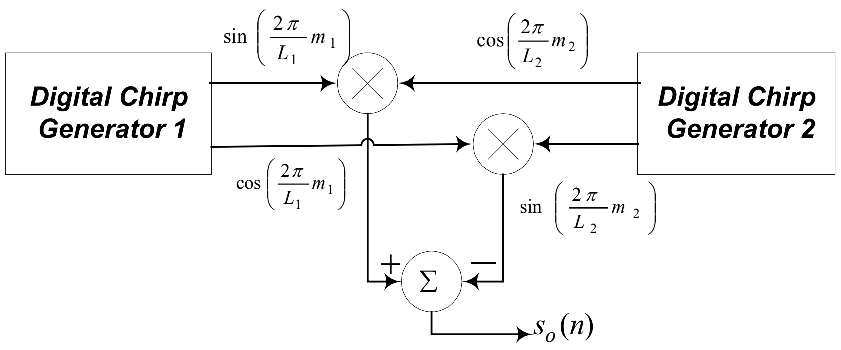

Digital Chirp Generator 1
Digital Chirp Generator 2
$\sin\left(\frac{2\pi}{L_1}m_1\right)$
$\cos\left(\frac{2\pi}{L_2}m_2\right)$
$\cos\left(\frac{2\pi}{L_1}m_1\right)$
$\sin\left(\frac{2\pi}{L_2}m_2\right)$
+
−
Σ
$s_o(n)$

Figure 3.4: Schematic diagram of the fractional bit digital chirp generator structure $s_o(n) = \sin[(2\pi/L_1)m_1 - (2\pi/L_2)m_2]$

The output chirp signal, which is denoted by $S_o(m_1, m_2)$, is given by:

$$S_o(m_1, m_2) = \sin\left[2\pi\left(\frac{m_1}{L_1} - \frac{m_2}{L_2}\right)\right] \qquad L_2 > L_1 \tag{3.35}$$

The generated phase of the chirp generator in this section is written as:

$$\theta_g(m_1, m_2) = 2\pi\left(\frac{m_1}{L_1} - \frac{m_2}{L_2}\right) \qquad rad \tag{3.36}$$

The generated sweep rate, of this chirp generator, can be shown by using (3.36) and is given by:

$$S_g = f_{clk}^2\left(\frac{1}{2^{p_1}L_1} - \frac{1}{2^{p_2}L_2}\right) \qquad Hz/s \tag{3.37}$$

Consequently, the start frequency and start phase can also been written as:

$$f_{go} = f_{clk}\left(\frac{CO_{o1}}{2^{p_1}L_1} - \frac{CO_{o2}}{2^{p_2}L_2}\right) \qquad Hz \tag{3.38}$$

$$\theta_{go} = 2\pi\left(\frac{AC_{o1}}{2^{p_1}L_1} - \frac{AC_{o2}}{2^{p_2}L_2}\right) \qquad rad \tag{3.39}$$

If we select identical addresses, this means that $p_1 = p_2 = p$, then (3.37) can be written in the following form

$$S_g = \frac{f_{clk}^2}{2^p}\left(\frac{L_2 - L_1}{L_2 L_1}\right) \qquad Hz/s \tag{3.40}$$

By assuming $L_1 = L_2 - 1$ and L_2 is divisible by 4, it is clear that L_1 is not divisible by 4, then (3.40) can be written as

$$S_g = \frac{f_{clk}^2}{2^p L_2 L_1} \qquad Hz/s \tag{3.41}$$

In conclusion, comparing the sweep rate of this chirp generator and the Pedersen sweep rate, it is clear that this chirp generator is equivalent to Pedersen's generator with table length $L = L_2 L_1$. This chirp generator is more suitable for the applications, which need an extremely low sweep rate and it can be calculated based on (3.41). Moreover the chirp generator mentioned in this section can reach sweep rates lower than that of Pedersen by a factor of $(2^K - 1)$ assuming that the Pedersen LUT length is equivalent to the LUT length of the second chirp generator ($L_2 = 2^K$) as in Figure 3.4.

4 Direct Digital Frequency Synthesizer

4.1 Introduction

The first step of generating digital chirp signals starts with implementing a Direct Digital Frequency Synthesizer (DDFS) and combining it within the hybrid architecture (proposed architecture) in order to generate chirp signals with a quadratic phase and extremely high quality.

This chapter describes in detail the basic implementation, methodology and the practical consideration of the DDFSs, which generate single phase or quadrature sinusoids with excellent frequency resolution, good spectral purity and phase continuity on switching [18]. DDFS plays an important role, both in modern communication systems and in measurement instrumentation.

The DDFS, viewed as a single functional unit, accepts a normalized input frequency control word (FCW) and generates a sequence of samples of sine and cosine functions having the precise frequency dictated by the input FCW. Fast frequency switching is crucially important in modern wireless communication systems such as Time Division Multiple Access/Code Division Multiple Access TDMA/CDMA digital cellular systems and spectrum-spread wireless LANs. For example, the TDMA system may require that the carrier frequency have to be switched during a signal slot, that is, the change must be accomplished within 100 μs.

The advantages of the digital components, which can be relied upon, include direct processing control, high programmable ability, small area, fast switching speed, low phase noise, excellent stability, and low power consumption, (which is essential requirement in wireless devices), and reduced device complexity. This type of circuit has improved over the past three decades as silicon technology has matured, and became widely applied in modern communication systems.

4.2 The traditional DDFS

The Pedersen methodology applies the modification to the phase generation circuitry of the direct digital frequency synthesizer in order to produce the synthesized chirps.

The first model of DDFS was introduced by Tierney *et al.* [4] in 1971. As shown in Figure 4.1, it consists of two main blocks namely as phase accumulator (PA) and phase-to-sine-amplitude converter (PSAC) or sine/cosine generator. The PA block consists of an over-flowing adder and a feedback register. The second block consists of a PSAC which is traditionally implemented digitally using a ROM and can be followed by a digital to analog converter (DAC) and a low pass filter (LPF) if an analog output is desired.

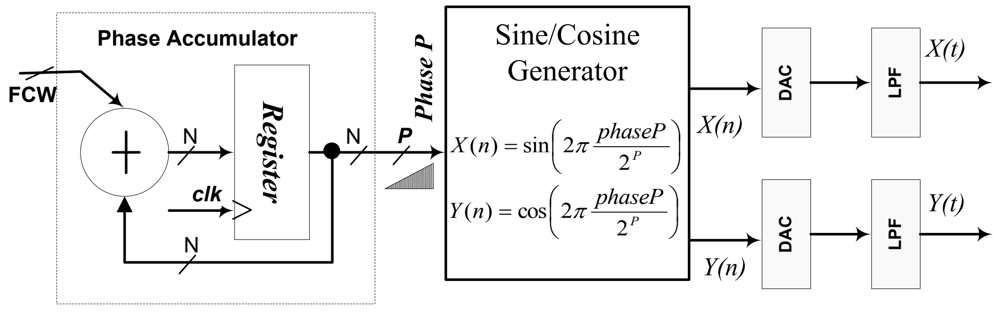

Phase Accumulator
FCW
N
clk
N
Register
N
P
Phase P
Sine/Cosine Generator
$X(n)=\sin\left(2\pi\frac{phaseP}{2^P}\right)$
$Y(n)=\cos\left(2\pi\frac{phaseP}{2^P}\right)$
X(n)
Y(n)
DAC
LPF
X(t)
DAC
LPF
Y(t)

Figure 4.1: Simplified schematic of a quadrature DDFS

The PA is a variable-increment N-bit counter as shown in Figure 4.1, the output is increased by Frequency Control Word (FCW) for every successive clock pulse interval, $t_{clk}=1/f_{clk}$. It produces a ramp by integrating the value of the FIW, which varies in the interval $[0,2^{N-1}]$. As shown in Figure 4.1, the frequency of the generated sine wave is controlled by (FCW) as shown in Figure 4.2, and the following equation shows the frequency relationships of the DDFS structure.

$$\begin{aligned} f_{\min} &= \frac{f_{clk}}{2^N} \\ f_{out} &= f_{\min} \bullet FCW \qquad 0 \le FCW \le 2^{N-1} \end{aligned} \tag{4.1}$$

$f_{\min}$ is the minimum synthesizable frequency, f_{clk} is the clock frequency, N is the word length of the phase accumulator, f_{out} is the output frequency, FCW is the frequency input control word, and the frequency resolution of the synthesizer is $\Delta f = \dfrac{f_{clk}}{2^N}$.

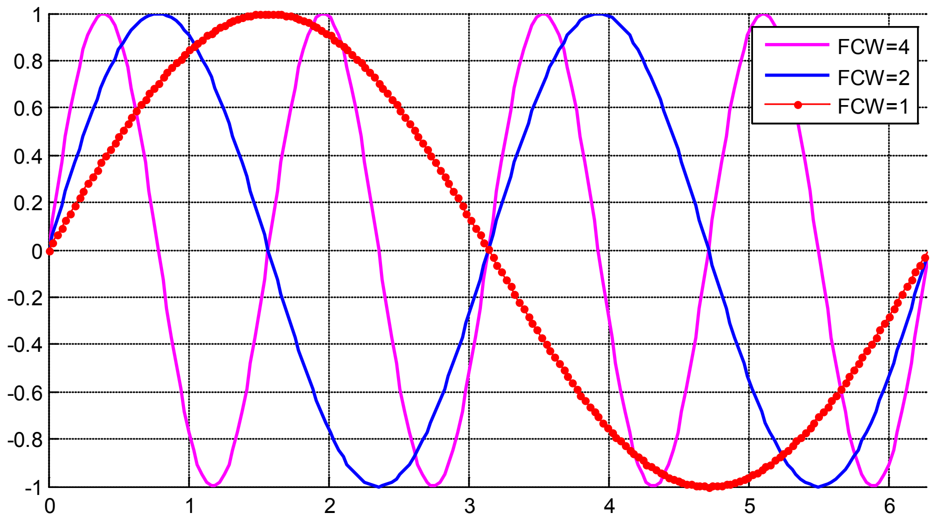


Figure 4.2: Effect of FCW values on output sine wave.

When $\phi(n)$, the content of the PA, is greater than 2π then the accumulator overflows and resets to 0. The period of the sine signal is represented by $T = \dfrac{1}{f_{out}} = \dfrac{2^N}{FCW} \times t_{clk}$, which means that the larger values of FCW the phase will be increased at faster rates. Consequently, higher frequencies will be generated.

The sine/cosine generator maps the output data of the PA to the approximated (quantized) sine amplitude. The function of the DAC is to convert the output sine wave to an analog signal form. The spectral purity of the sine wave is also enhanced by the LPF, which attenuates and filters out unwanted high frequency signals that are present due to the digital approximation.

4.3 A demonstration for the functionality of DDFS

The DDFS in Figure 4.1 generates the outputs corresponding to an input *FIW*. To understand the operation of the DDFS, let us assume that the PA is implemented using N = 8-bits and an infinite precision PSAC in which the amplitudes of the samples of the sinusoidal signal are not quantized. The 8-bit PA output value represents an angle corresponding to one of the 256 equally spaced angles from $[0-2\pi)$. For example, the 8-bit PA output '00000000' represents the angle 0 radians, and ‘11111111’ represent the angle $255\times 2\pi/256 = 2\pi - \pi/128$.

Let us assume that the DDFS input *FCW* = '00000100', which corresponds to an angular frequency of $4\times 2\pi/256 = \pi/32$ radians/sample, is applied at time $n=0$ and that the PA output at $n=0$ has been reset to '00000000'. Thereafter, at each clock pulse, the adder in the PA will add the preceding PA output stored in the register to the input *FCW*. This will generate successive angles $0, \pi/32, \pi/16, 3\pi/32, \pi/8, ...$ and so forth. At $n=63$ the PA content is ‘11111100’ and adding FCW to this content an overflow will occur and the PA output is set to ‘111111111’ at $n=64$. This sequence of 64 samples form a complete cycle of the sinusoidal waveform, the adder will start again from ‘00000000’ and so on.

If the input *FCW* is changed from ‘00000100’ to ‘00001000’ (i.e., the *FCW* is doubled from $\pi/32$ radians/sample to $\pi/16$ radians/sample) immediately after $n=64$. The overflowing accumulator will begin incrementing the PA output according to the new *FCW* and will generate output values corresponding to the following sequence of angles $0, \pi/16, \pi/8$ and so forth for $n = 64, 65, 66, \ldots$ respectively [19].

The PSAC following the PA in Figure 4.1 generates the DDFS outputs by taking the binary words sequence at PA output and performing a mapping from the angles to their corresponding sine and cosine values. The DDFS outputs for the PA output sequence generated from the input $FCW = 00000100$ for $0 \leq n \leq 64$ and the input $FCW = 00001000$ for $64 \leq n \leq 96$ are given in Figure 4.3. The outputs for $0 \leq n \leq 64$, plotted by circles in Figure 4.3, correspond to the first FCW of $\pi/32$ radians/sample, while the outputs for $64 \leq n \leq 96$, plotted by asterisks in Figure 4.3, correspond to the second FCW of $\pi/16$ radians/sample.

As illustrated in Figure 4.3, the frequency of output sinusoids for $n \geq 64$ is twice the frequency of output sinusoids for $n < 64$. This is due to the doubling of the input FCW, occurring at $n = 64$, which causes the phase accumulator to sweep the unit circle twice as fast. This figure shows also the phase continuity in both sine and cosine functions. This figure also shows the phase continuity in the case of sine and cosine functions.

The overflowing characteristic of the phase-accumulator adder performs the necessary $\bmod(2\pi)$ operations, mapping all phase angles greater than 2π to their corresponding angles between zero and 2π [19].

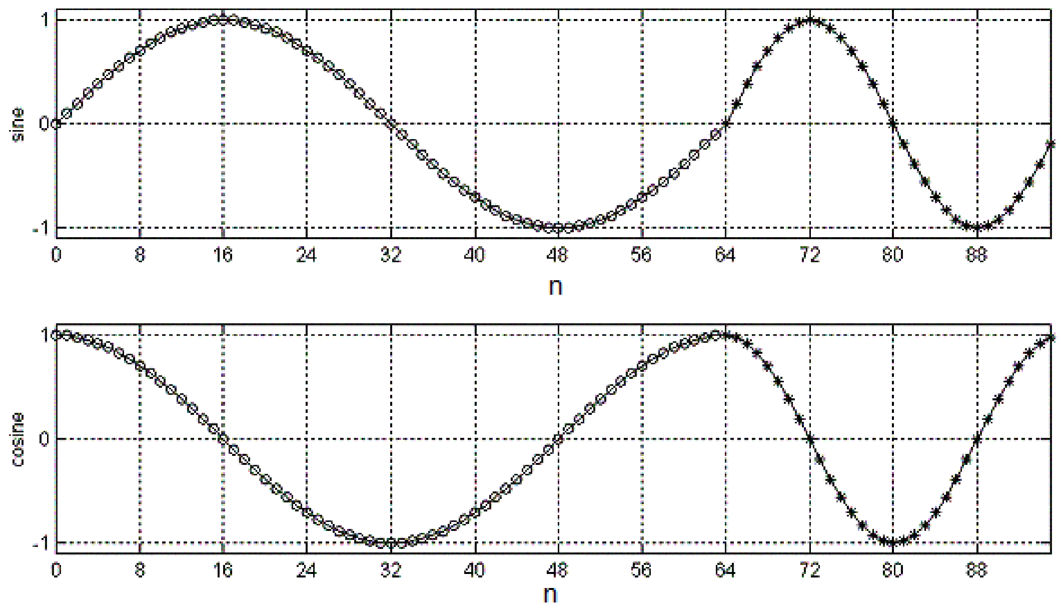


Figure 4.3: Sine and cosine DDFS outputs generated from the PA outputs

4.4 Practical considerations for DDFS

In order to limit the complexity of the Phase to Sine Amplitude Converter (PSAC), it should be built with a reasonable amount of hardware; the PA output is typically truncated before being fed to the PSAC, as shown in Figure 4.4, where only M bits out of N bits are retained. This phase truncation causes errors (deterministic, periodic errors, often referred to as noise) at the DDFS output formed as a set of spurious frequencies (output signal components at undesired frequencies). Furthermore, the PSAC has finite precision outputs and its implementation may employ approximations and quantization operations that also generate deterministic noise at the DDFS output, hence, contribute another set of spurious frequencies. Spurious frequencies are often referred to as spurs [20]. The spurs caused by phase truncation are referred to as phase-truncation spurs, while the spurs resulting from PSAC imperfections are referred to as PSAC spurs.

The sine and cosine functions have symmetry properties that are exploited in DDFS design to reduce the complexity of the PSAC implementation. In general, the complexity of the PSAC implementation grows with an increasing number of PSAC input bits k as in Figure 4.4. For example, if the PSAC accepts the M bits as the address to a LUT with pre-computed and stored values for the corresponding sine and/or cosine functions, each increase of one bit in M would increase the LUT size by a factor of two. There are many alternative and more efficient implementation techniques for the PSAC such as those reported in [21-24]. In these techniques, the complexity of the PSAC grows with increasing M, but may not grow exponentially as in the direct LUT case.

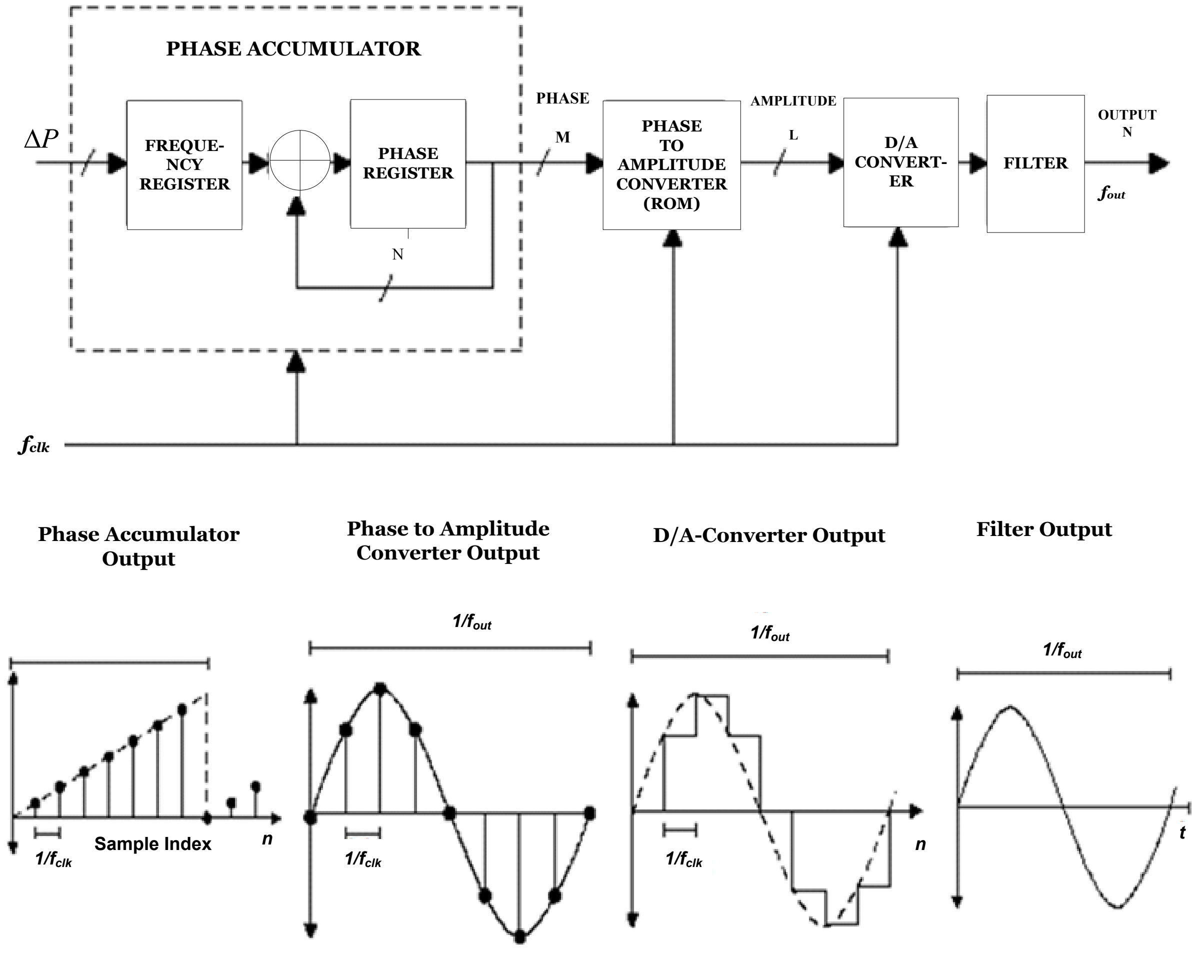


Figure 4.4: General structure of a practically realizable DDFS.

4.5 ROM Compression Techniques

The phase to sine amplitude converter is a digitally implemented device which converts digital phase input from the PA to output amplitude, which is often a sine look-up table (LUT) stored on a Read Only Memory (ROM). The LUT size is 2^N words and contains quantized amplitude values of one cycle of the waveform to be generated. The truncated PA output represents an address word that indicates the position in the LUT to be addressed in order to read the amplitude of the waveform, which is stored in the ROM. The frequency and amplitude resolution can be improved by increasing the number of memory locations and the length of memory words, in other words by a larger ROM size. Unfortunately, larger a ROM size means higher power consumption and lower speed. For this reason, many techniques for ROM reduction, which can achieve high frequency resolution and sufficient spectral purity, have been presented. The most important ROM compression techniques are exploitation of sine function symmetry, Taylor Series Approximation [24], Sunderland and its modified Architecure [21], and Nicholas Architecture [25].

5 Problem Definition and Introduction to the Solution

5.1 Problem definition

The goal is to implement a simple chirp generator with a maximum Spurious Free Dynamic Range (SFDR), less power consumption, fast operations and low cost hardware complexity. All of the previous methods for generating chirp signals reported in literature are suffering from at least from one of the features mentioned above.

The generation of the digital chirp signal based on the LUT method causes a big sized memory in addition to the spurious harmonic distortion as a result of neglecting some of the least significant bits of the generated quadrature phase, as mentioned in the previous methods. For the generation of this phase a counter and an integrator has been used to achieve the required phase, as shown in Figure 5.1. Here the presetting of the counter initial value (CO_0) and that of the accumulator (AC_0) specify the start up frequency ω_1 and the starting phase ϕ_0.

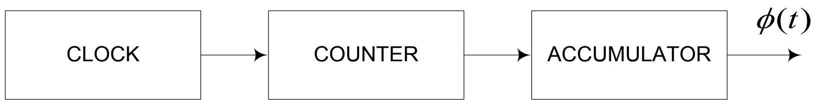


Figure 5.1: The generation of the quadratic phase $\phi(t)$

The chirp signal is a sinusoidal function of a phase $\phi(t)$ with a quadratic term of time t corresponding to linearly varying frequency (the sweep rate, S). As shown in the following equation:

$$\sin(\phi(t)) = \sin(\pi S t^2 + w_1 t + \phi_0) \tag{5.1}$$

The approach of Pedersen is based on real time digital evaluation of the phase of the desired chirp signal, and then reading its value from a look-up table (LUT) with the length ($L = 2^K$). The main disadvantage of Pedersen chirp generator is the high level of spurious harmonic distortion and the large size of the ROM [3, 26]. A chirp generator was reported in chapter 3.2 in which the fractional bits are utilized to interpolate the sample values that are not stored in the LUT and therefore increase the effective LUT length. This technique is known as fractional addressing, and was utilized to reduce the level of spurious harmonic distortion. The major disadvantage of the digital chirp generator in chapter 3.2 is that it requires the implementation of two independent chirp generators and then evaluating the phase difference between the two generated chirp signals. The hardware requirements, therefore, make its implementation costly.

For Synthetic Aperture Radar (SAR) applications, it is hard to generate a digital chirp signal with 200-400MHz bandwidth using one thread as in the previous methods e.g. Pedersen… etc.

5.2 Introduction to the solution

Some compression techniques as mentioned in chapter 4.5 are used to reduce the size of the required memory, and manage this issue, and then are used to generate a chirp signal; they solve the issue of the big sized memory, but at the same time another problem appears like the level of the spectral purity, hardware complexity, etc. These issues became the core of our problem and pushed us to implement an alternative chirp signal generator with a novel methodology which handles the above mentioned drawbacks.

The new methodology intends to implement a polynomial evaluation unit based on the piecewise parabolic interpolation technique to reconstruct the sinusoidal function with small sized memory, then combine it with a hybrid system which consists of Pedersen's method and the polynomial evaluation unit as an efficient a ROM compression technique. Ultimately, the small size of the memory can avoid the slowdown of the chirp generator's operation.

A further enhancement for the new architecture is applied in order to increase the spectral purity of the generated chirp signal. This enhancement is based on increasing both the order of the interpolation equation and the number of sections per quarter in the generated sinusoidal function (increasing the number of samples per quadrant), furthermore applying an optimization technique on the harmonics calculations.

The realization of the proposed system using the technology of the FPGA technology will show a promising architecture, because it is implemented with the less hardware complexity than the other generator in the literature.

As a solution for the high bandwidths digital chirp signals, a smart solution will be presented to increase the bandwidth in order to be suitable for SAR applications. This solution is summarized by implementing and using the parallel processing technique and generates four threads instead of one.

6 A novel and Improved Digital Chirp Generator

6.1 Introduction

The improved generator is based on the direct digital frequency synthesizers' (DDFS) technique not on the technique of the phase-locked loops (PLL), because there are a number of advantages offered by DDFS over PLL implementations that are very important in terms of performance improvement and superior capabilities. Some of these advantages are:

1- ***Fast Switching Speed*:** The DDFS is an open loop circuit without any feedback. A DDFS may be tuned between any two frequencies in one clock period (pipelined), which is typically less than 100 ns [20]. Certainly, the switching speeding cannot be less than the overall latency of the digital gates. Being in the nano second range, the tuning latency of the DDFS is significantly less than other types of synthesizers such as PLL.

2- ***High Frequency Resolution:*** when $\Delta f = f_{clk} / 2^N$, it can be seen that we can get precise frequency tuning by increasing the number of bits of the PA. Most DDFS have frequency resolutions in the Hz and μHz levels.

3- ***Low Phase Noise:*** The stability of the output frequency is determined by the stability of the reference frequency. Therefore, the phase noise is equal to or less than the reference frequency. Generally, the reference frequency is generated by a fixed crystal oscillator. Thus, the phase noise can be dramatically reduced.

4- ***Phase Continuity:*** if a new *FCW* is applied to the input of the PA, the DDFS will rapidly synthesize the new frequency. However, due to nature of the PA, the phase continuity is inherently guaranteed.

5- ***Stability of Output Frequency over a Wide Bandwidth:*** The minimum output frequency of DDFS is the minimum resolution of the reference clock frequency, or the resolution of the PA. According to Nyquist sampling theorem, a DAC can recover the signal if the frequency is less than half of the reference clock frequency [20]. Therefore, the maximum output frequency is equal to $f_{clk} / 2$.

Generally, there are two possible ways to improve DDFS: speed up the PA, and reduce the size of the ROM. These possible implementations are intended and directly related to obtaining higher speeds (higher frequencies or higher frequency resolutions) and reducing power consumption. In this thesis, we are going to deal with the reduction of the ROM size in the chirp generator, since ROM is a major consumer of system power and typically occupies a large portion of the chip area [27].

6.2 The Proposed Architecture of the Digital Chirp Signal Generator

The proposed system is a hybrid of the digital chirp generator and the system using interpolation based direct digital frequency synthesizer as published in [28-30]. The interpolator uses predetermined interpolation coefficients to fit the sine wave from the calculated phase instead of using a predetermined waveform stored in a big sized memory. This implies that a smaller look-up table for the sine and cosine functions is used compared to existing architectures with minimum hardware overhead, and the computation of the sinusoidal values is performed by a piecewise parabolic and extended to a piecewise-polynomial approximation structure. Thus, only interpolation coefficients are stored in the memory. As a matter of fact, this work came to avoid the consequent problems of the large ROMs. The large ROMs slow down the chirp generator’s operation and increase the power consumption [31]. The speed of the Chirp's operation depends on the circuit complexity and the hardware overhead. Therefore, our proposed system with smaller ROM has less hardware and circuit complexity which avoids the slowdown of the generator's operations. Figure 6.1 shows the block diagram of the proposed digital chirp generator.

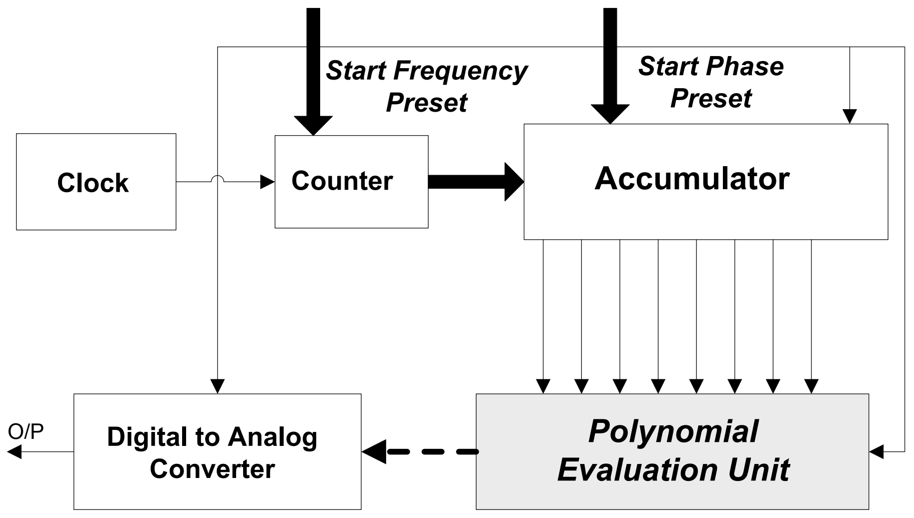


Figure 6.1: Block diagram of the proposed digital chirp generator

In this chirp generator the start frequency and phase can be controlled by the initial content of the counter and the accumulator. Additionally, the sweep rate can also be controlled by the location and size of the address lines. This digital chirp generator uses the clock to trigger the counter (first integrator) and feeds the output to the accumulator (second integrator). In general, the phase of the sweep signal is not equal to the value of the phase stored in the accumulator because only a subset *k* of the output lines from the accumulator is sent to the look-up table as in Pedersen's method. The proposed architecture, as shown in Figure 6.1, consists of a clock, counter, accumulator, and the decisive polynomial evaluation unit's block, in addition to the digital to analog converter, respectively.

The polynomial evaluation unit contains the evaluating methodology for the sinusoidal waveforms, which are based on both exploiting the sinusoidal symmetry, and piecewise polynomial approximation techniques as a compression technique. In the following sections we will explain this compression technique in detail.

As we will see later in the section 7.4, the proposed architecture as in Figure 6.1 will be improved and modified in order to achieve the goal of generating the chirp signal with high bandwidth using the parallel processing technique.

6.3 Quadrant Compression Technique Using the Piecewise Parabolic Interpolation

This technique uses the symmetry of the sine wave with the piecewise polynomial interpolation simultaneously, where the implementation of the sine mapping function is depicted in Figure 6.2, where the angle within [0, π/2] is represented by the LSBs (*P*-2), which feeds the piecewise polynomial interpolator in order to calculate the sine function. The two MSBs of the input phase determine in which quadrant the input phase lies in and perform its computations.

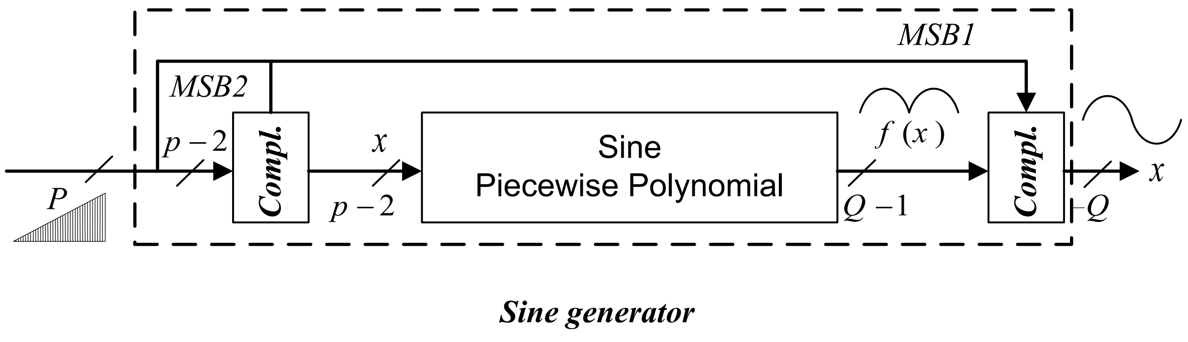
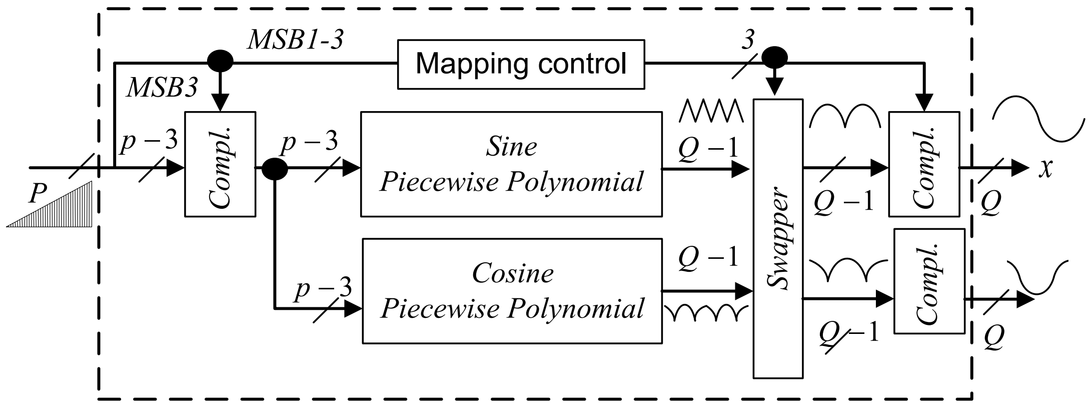


Figure 6.2: Quadrant compression architecture for single phase chirp generator

The quadrant compression architecture for single phase generator can be extended to compute the sine and cosine functions by using two piecewise polynomial interpolators. This extension is illustrated in Figure 6.3 in order to compute the sine and cosine functions in the [0, π/4] interval. If the input phase represents the angle in the [π/4, π/2] interval, then the input of the sine and cosine blocks is complemented while the outputs of sine and cosine blocks are swapped, since $\sin(\pi/4+x)=\cos(\pi/4-x)$ and $\cos(\pi/4+x)=\sin(\pi/4-x)$ [31].

Similar swapping and/or inversion operations take place if the input phase represents an angle belonging to second, third and fourth quadrant. The third MSB of the phase P is used to invert the input of both sine and cosine blocks, while the three MSBs of phase P are used to properly swap and/or invert their outputs.



Sine/Cosine generator

Figure 6.3: Quadrant compression architecture for quadrature phase chirp generator

The polynomial interpolation is an important and well known technique for computing very high precision approximations of the elementary functions, including exponential, logarithm, square-root, and various transcendental functions. This technique will be applied to our proposed chirp generator in order to make it more efficient than the other generators with respect to the following important aspects: the size of the memory, the spurious harmonic distortion, complexity, and flexibility.

This architecture has four parameters that need to be adjusted in order to have a minimum low level of spurious harmonic distortion, and low hardware complexity. This system is also proposed to be able to generate a chirp signal with high bandwidth reaching up to 400MHz in order to use it in Synthetic Aperture Radar's applications. These parameters are summarized with the degree of the polynomial approximation, the number of the piecewise-continuous polynomial sections, the position of the segment bound, and the method for generating the interpolation coefficients in order to reconstruct the sinusoidal waveform. This methodology with the corresponding parameters will be explained in detail in the coming sections.

Several DDFS design approaches have been proposed based on the following expression with various degrees of complexity [32]. All design approaches are specific cases of (6.1); it differs in only four main points. These points are the degree of the polynomial approximation, the number of piecewise-continuous polynomial sections per quadrant, the position of the bounds, and the method of calculating the polynomial coefficients c_{ki}. The expression in (6.1) shows the relationship of these parameters.

$$\sin(\frac{\pi x}{2}) \cong \begin{cases} \sum_{i=0}^{r} c_{oi}(x-x_o)^i & x_0 \le x < x_1 \quad (x_0 = 0) \\ \sum_{i=0}^{r} c_{1i}(x-x_1)^i & x_1 \le x < x_2 \\ \vdots & \\ \sum_{i=0}^{r} c_{ki}(x-x_k)^i & x_k \le x < x_{k+1} \\ \vdots & \\ \sum_{i=0}^{r} c_{(s-1)i}(x-x_{s-1})^i & x_{s-1} \le x < x_s \quad (x_s = 1) \end{cases} \tag{6.1}$$

where

x : is the phase argument, represented as a fraction in the interval $[0,1)$

r: is the degree of the polynomial approximation, $r \ge 1$

s : is the number of piecewise continuous polynomial segments, $s \ge 1$

c_{ki} : are the polynomial coefficients

x_k : is the lower bound of the k^{th} piecewise continuous segment.

In the case where the quadrant is divided into equal length sections, the segment's lower bound x_k is calculated by:

$$x_k = \frac{k}{s} \qquad k \in \{0,1,2,\cdots,s\} \tag{6.2}$$

In addition, if s is equal to an integer power of two, then the subtraction $(x - x_k)$, for $x_k \le x \le x_{k+1}$ is accomplished trivially by truncating the $\log_2(s)$ *MSB*s from x.

6.4 The derivation of the used interpolants in the proposed digital chirp generator

In the implementation of the proposed chirp generator, we will employ the piecewise polynomial approximation to further compress the ROM size of our generator, while maintaining or exceeding the spectral purity of the techniques mentioned before. As a hybrid system based on the direct digital frequency synthesizer, the techniques in [31, 33] will be adopted to achieve the new chirp generator with a good performance.

The interpolated values $y(kT_i)$ can be achieved by entering the signal samples *x(m)* through a DAC, a time continuous filter, and then re-sampling the analog signal $y(t)$ at $t= kT_i$, as shown in Figure 6.4. It shows a hybrid analog/digital model for the interpolation filter, and converts the samples to a sequence of weighted analog impulses, which are applied to a time-continuous interpolating filter with impulse response $h_I(t)$.

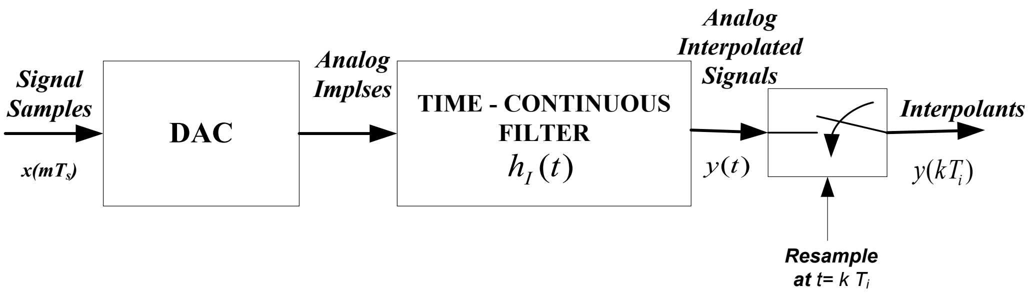


Figure 6.4: Rate conversion with time-continuous filter

The output of the filter is shown in (6.3) :

$$y(t) = \sum_{m} x(m) h_I(t - mT_s) \qquad y(t) \neq x(t) \tag{6.3}$$

Where *x(m)* is a sequence of signal samples taken at intervals T_s. Then after re-sampling $y(t)$ at $t= kT_i$, where T_i is synchronized with the signal symbols, and $h_I(t)$ is the impulse response. So the interpolated (the new samples) are given by (6.4):

$$y(kT_i) = \sum_{m} x(mT_s) h_I(kT_i - mT_s) \tag{6.4}$$

In order to map the interpolation image, we have to recognize that *m* is a signal index and the filter index is defined by (6.5)**:**

$$i = \text{int}\left[\frac{kT_i}{T_s}\right] - m \tag{6.5}$$

where int[z] means largest integer not exceeding z.

The base point index and the fractional interval are given by (6.6) and (6.7) respectively

$$m_k = \text{int}\left[\frac{kT_i}{T_s}\right] \tag{6.6}$$

$$\mu_k = \frac{kT_i}{T_s} - m_k \qquad 0 \leq \mu_k \leq 1 \tag{6.7}$$

The equation shown in (6.4) can be arranged and reformed as in (6.8) , if we know that $m = m_k - 1$, $(kT_i - mT_s) = (i + \mu_k)T_s$ and the interpolants are re-sampled at time. In this case the following equation as shown in (6.8) is representing the fundamental equation for digital interpolation of data signals:

$$\begin{aligned} y(kT_i) &= y\left[(m_k + \mu_k)T_s\right] \\ &= \sum_{i=I_1}^{I_2} x\left[(m_k - i)T_s\right] h_I\left[(i + \mu_k)T_s\right] \end{aligned} \tag{6.8}$$

where $\{x(m)\}$ is a sequence of signal samples taken at interval T_s, μ_k is a fractional shift, m_k is the base point index, and $h_I(t)$ is a finite-duration impulse response of a continuous time analog interpolating filter as it is shown in (6.9). This kind of interpolation filter can be efficiently implemented using a new equivalent structure as in the Farrow structure [34].

The implementation assumes that the impulse response is a piecewise defined polynomial in each T_s segment (section) with $i \in [I_1, I_2]$.

$$h_I(t) \triangleq h_I\left[(i + \mu_k)T_s\right] = \sum_{n=0}^{N} b_n(i)\mu_k^n \tag{6.9}$$

The coefficients $b_n(i)$ are fixed numbers, independent of μ_k, determined by the filter's impulse response $h_I(t)$. These coefficients are chosen to provide the widest pass-band and the strongest attenuation at multiples of the sampling frequency thus reforming robust interpolation. This operation and the sample time relations (Base point pivoting) are illustrated in Figure 6.5.

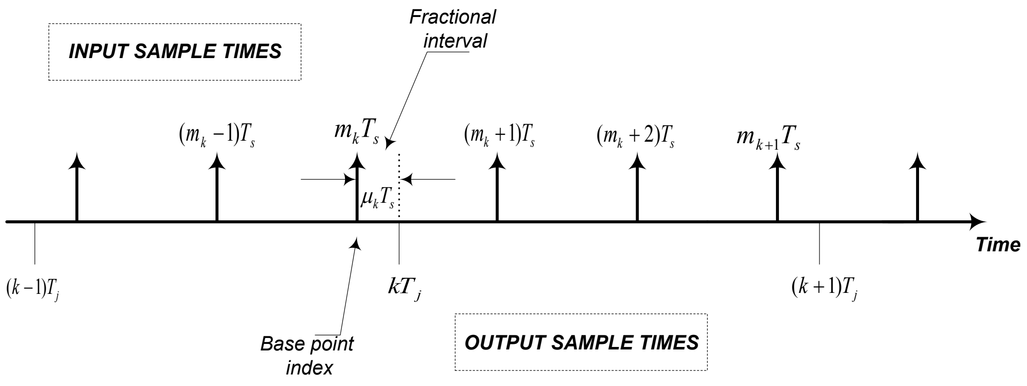


Figure 6.5: Base point pivoting

We can consider $y(kT_i)=y(k)$, after we substitute (6.9) in (6.8) and rearrange the terms to show that the interpolation can be performed as the following:

$$\begin{aligned} y(k) &= \sum_{i=I_1}^{I_2} x(m_k-i)\sum_{n=0}^{N} b_n(i)\mu_k^n \\ &= \sum_{n=0}^{N} \mu_k^n \sum_{i=I_1}^{I_2} b_n(i)x(m_k-i) \\ &= \sum_{n=0}^{N} \mu_k^n v(n), \qquad wh\ ee\ \ v(n)=\sum_{i=I_1}^{I_2} b_n(i)x(m_k-i) \end{aligned} \tag{6.10}$$

In this work we will consider the generation of the sine chirp signal; so technically, the sine wave is divided into four basic quadrants; each quadrant is subdivided into sections as shown in Figure 6.6.

The compression ratio can first be measured by a simple case in which only four sections (four base points) per quadrant are considered as illustrated in Figure 6.6. In this case, for each base point, three interpolation coefficients are required (V_1, V_2, V_3). This needs a ROM size of 4 (base points) × 3 (coefficients) = 12 words, by assuming that each word consists of 10 bits, then the total ROM size will be 120 bits.

By exploiting the symmetry of the sine/cosine wave, only the parameters that are relevant to one quadrant need to be stored, instead of storing values of the sine function directly in the ROM as in the traditional method, the interpolation coefficients will be stored.

The Farrow structure as in [35] receives the input samples from the sine wave and performs the interpolation based on the neighboring three points. Therefore, for each base point (sample of the sine), three interpolation coefficients are computed. These interpolation coefficients are used according to (6.10) in order to compute the value of the sine wave at any arbitrary fractional delay specified by μ_k .

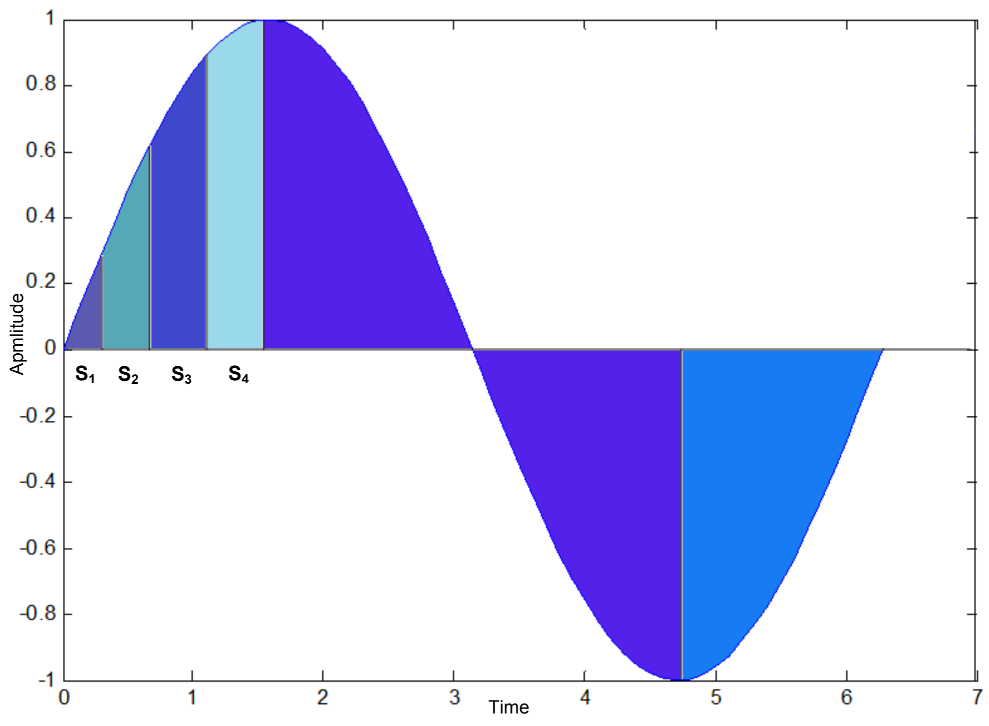


Figure 6.6: Sine wave partitioning

The implementation of the structure, which is based on the piecewise parabolic interpolation is described as the following:

$$y(2,k) = V_1(2) + \mu_k V_2(2) + \mu_k{}^2 V_3(2) \tag{6.11}$$

where μ_k is the fractional shift, *n=2* is the base point index and k is the fractional delay index. The interpolation calculation has been pre-computed for a specified number of base points (sections) of the sine wave and stored in a ROM. Each word in the ROM consists of (V_1, V_2, V_3) which are in conjunction with the fractional delay μ_k, and can be used to calculate intermediate points between two base points.

As we will see later, when we increase the number of interpolation coefficients, the error will be further decreased. Equation (6.12) is a polynomial equation derived from (6.11); for a third order interpolation:

$$y(n,k) = V_1(n) + \mu_k V_2(n) + \mu_k{}^2 V_3(n) + \mu_k{}^3 V_4(n) \tag{6.12}$$

Where μ_k is the fractional shift, n is the base point index and k is the fractional delay index. Let us express in Figure 6.7 the piecewise polynomial interpolation of the sine function in $[0, \pi/2]$ for the third order approximation to be compatible with Figure 6.2 and Figure 6.3, and it will be inside the polynomial evaluation unit.

The $p-2$ LSBs of the input phase (representing an angle within $[0, \pi/2]$) feed the piecewise-polynomial interpolator, are used to compute the sine function. The two MSBs of input phase are used to determine the quadrant in which the input phase lies and to perform accordingly the required complementation [18]. The signal x represents the input phase $[0, \pi/2]$ scaled to a binary fraction in the interval $[0,1]$. The range $[0,1]$ is subdivided in s

sub-intervals, with $s = 2^u$. The u MSBs of x encode the segment starting point x_k and are used as an address to the small look-up tables that store the polynomial coefficients. The remaining bits of x represent the offset $x - x_k$.

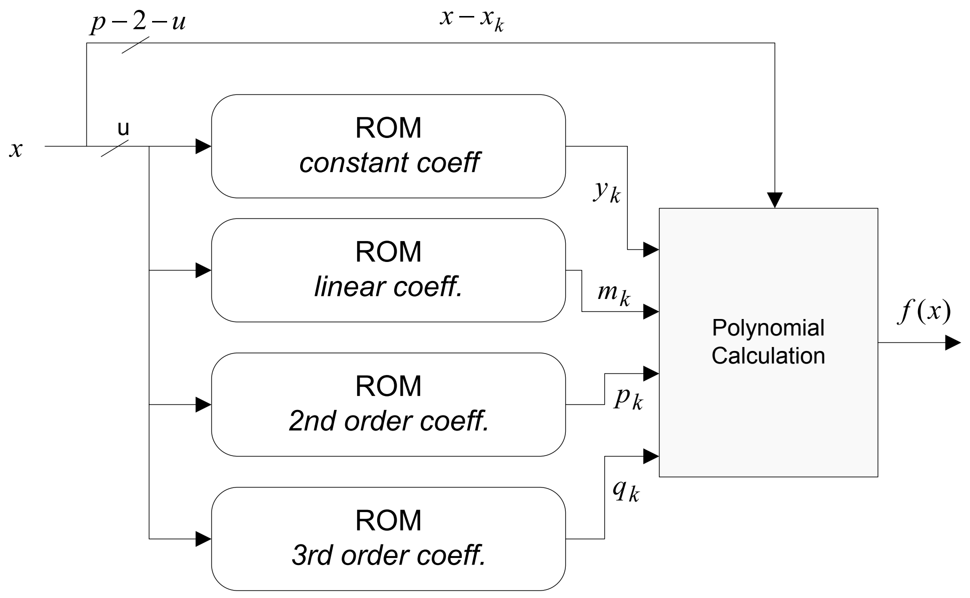


Figure 6.7: Piecewise-polynomial interpolation of sine function in $[0, \pi/2]$ for a third-order approximation

The polynomial calculation block computes the piecewise polynomial approximation of the sine function as follows:

$$f(x) = y_k + m_k(x - x_k) + p_k(x - x_k)^2 + q_k(x - x_k)^3$$
$$\mathit{for}\ x_k \le x \le x_{k+1;\ k=1\ldots s} \qquad (6.13)$$
$$\mathit{with}\ x_1 = 0;\ x_s = 1$$

For a piecewise parabolic approximation's case $q_k = 0$, while we have $p_k = 0$ and $q_k = 0$ for a piecewise-linear approximation.

6.5 Spurious Free Dynamic Range (SFDR)

The Spurious Free Dynamic Range (SFDR) is the usable dynamic range of a DAC before spurious noise interferes or distorts the fundamental signal. SFDR is the measure of the difference between the fundamental and the largest harmonically or non-harmonically related spur from DC to the full Nyquist bandwidth (*$f_S/2$*). In short, it is defined as the amplitude ratio between the wanted sinusoid and the largest undesired frequency component, and it is generally considered to be the most important parameter for communication systems applications [36]. Figure 6.8 shows how SFDR is measured correctly (usually it is measured in dBc). As shown in Figure 6.9, 95.6 dBc(1) is the SFDR of the generated chirp signal for order 2 and 16 section/quad, read from the plot.

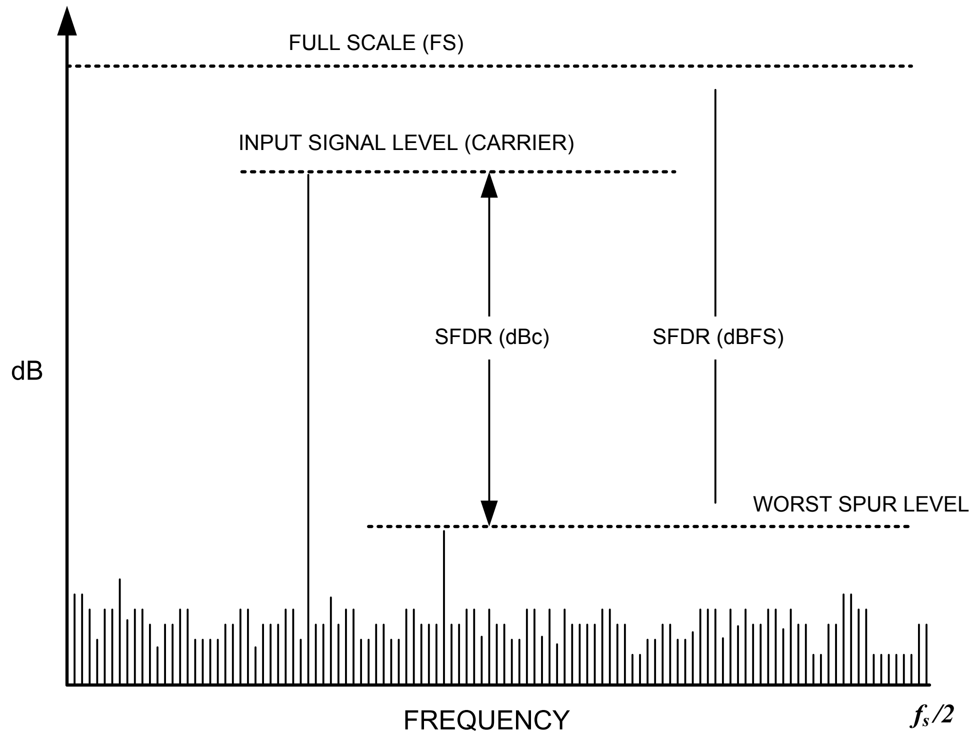


Figure 6.8: Measure of Spurious Free Dynamic Range (SFDR), as measured in [37]

1) dBc (decibels relative to carrier) is a measure of the strength of an instantaneous signal at radio frequency . Suppose a signal has an unmodulated-carrier power of P_0 watts and a modulation signal power of P watts at some specified instant in time. Then the instantaneous modulated signal strength in dBc, symbolized S_{dBc} ,is: S dBc=10 $\log_{10}(P/P_0)$

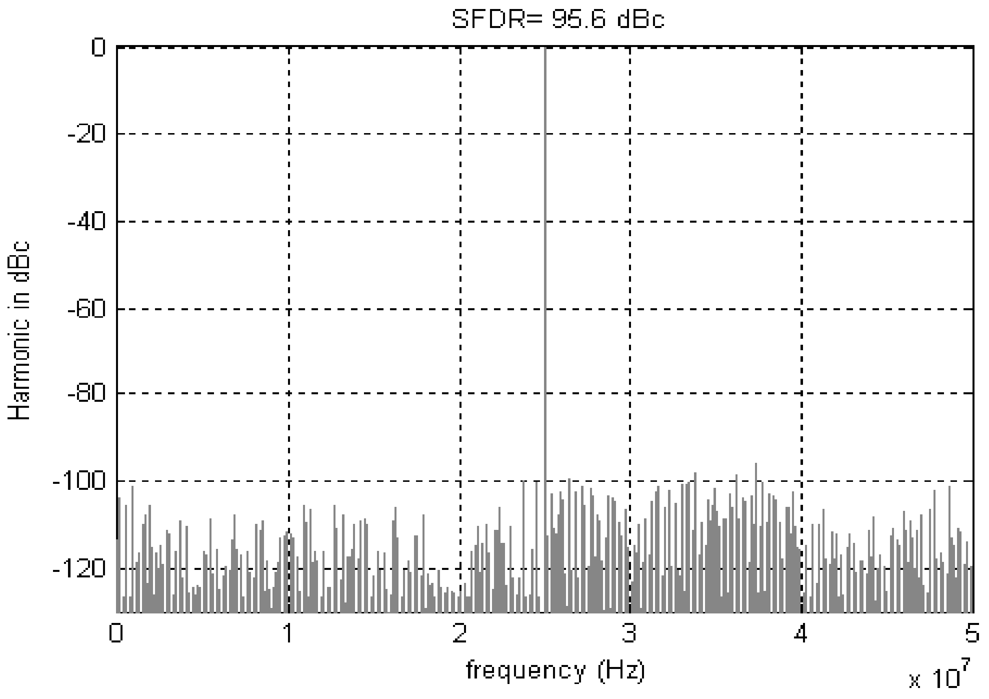


Figure 6.9: Output spectrum with 16 sections/quad

6.6 Harmonics Calculation

In this section we will analyze how to calculate the coefficients. Let us consider that the output of the digital chirp generator $g(x)$ in terms of a Fourier series to be as the following:

$$g(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos(\frac{2n\pi}{T}x) + b_n \sin(\frac{2n\pi}{T}x)\right) \tag{6.14}$$

We will consider here the generation of the sine chirp signal, and we will assume that $g(x)$ as a period of T=4 with an odd symmetry. Therefore the output can be represented by Fourier series as follow:

$$g(x) = \sum_{n=1}^{\infty} b_n \sin(\frac{n\pi x}{2}) \tag{6.15}$$

Since $g(x)$ has quadrant symmetry, it's even harmonics are zero. The amplitude of the odd harmonics are given by:

$$b_n = 2\int_0^1 f(x)\sin(\frac{n\pi x}{2})dx \qquad n\ odd \tag{6.16}$$

By substituting (6.13) in (6.16) with some calculations detailed in the appendix, we will obtain the following:

$$b(n) = \frac{4}{n\pi}g(n) - \frac{8}{(n\pi)^2}h(n) - \frac{32}{(n\pi)^3}l(n) + \frac{192}{(n\pi)^4}m(n) \tag{6.17}$$

for odd n and 0 otherwise, with:

$$g(n) = \sum_{k=0}^{s-1} \alpha_k \cos(\frac{kn\pi}{2s}) \tag{6.18}$$

$$h(n) = \sum_{k=1}^{s} \beta_k \sin(\frac{kn\pi}{2s}) \tag{6.19}$$

$$l(n) = \sum_{k=0}^{s-1} \gamma_k \cos(\frac{kn\pi}{2s}) \tag{6.20}$$

$$m(n) = \sum_{k=0}^{s-1} \delta_k \cos(\frac{kn\pi}{2s}) \tag{6.21}$$

The coefficients α_k, β_k, γ_k, and δ_k in equations (2.17), (2.18), and (2.19) are, respectively, given by:

$$\alpha_k = \begin{cases} y_{k+1} - y_k - \frac{m_k}{s} - \frac{p_k}{s^2} + \frac{q_k}{s^3}, & k = 1\ldots s-1 \\ y_1, & k = 0 \end{cases} \tag{6.22}$$

$$\beta_k = \begin{cases} m_{k+1} - m_k - 2\frac{p_k}{s} - 3\frac{q_k}{s^2}, & k = 1 \ldots s-1 \\ -m_s - 2\frac{p_s}{s} - 3\frac{q_s}{s^2}, & k = s \end{cases} \tag{6.23}$$

$$\gamma_k = \begin{cases} p_{k+1} - p_k - 3\frac{q_k}{s}, & k = 1 \ldots s-1 \\ p_{1,} & k = 0 \end{cases} \tag{6.24}$$

$$\delta_k = \begin{cases} q_{k+1} - q_k, & k = 1 \ldots s-1 \\ -q_s, & k = 0 \end{cases} \tag{6.25}$$

For a piecewise-quadratic interpolation, $q_k = 0$, the coefficients δ are zero and the function $m(n)$ in (6.17) vanishes. In the case of piecewise-linear interpolation, both p_k and q_k will be equal to zero and consequently $l(n)$ and $m(n)$ will vanish too. Equation (6.17) will reduce to the following:

$$b(n) = \frac{4}{n\pi} g(n) - \frac{8}{(n\pi)^2} h(n) \tag{6.26}$$

From (6.18)-(6.21), it can be seen that the functions $g(n)$, $h(n)$, $l(n)$, and $m(n)$ are periodic in n with period $4s$. Moreover, $g(n)$ and $l(n)$ have even symmetry, whereas $h(n)$ and $m(n)$ have odd symmetry

$$\begin{aligned} g(n) &= g(4s+n) = g(-n) = g(4s-n) \\ h(n) &= h(4s+n) = -h(-n) = -h(4s-n) \\ l(n) &= l(4s+n) = l(-n) = l(4s-n) \\ m(n) &= m(4s+n) = -m(-n) = -m(4s-n) \end{aligned} \tag{6.27}$$

As a result, the SFDR for the proposed system with the optimization is increased from 97.6 dBc (in the non-optimized case) to 100.9 dBc. Both cases are illustrated in Figure 6.10 and Figure 6.11

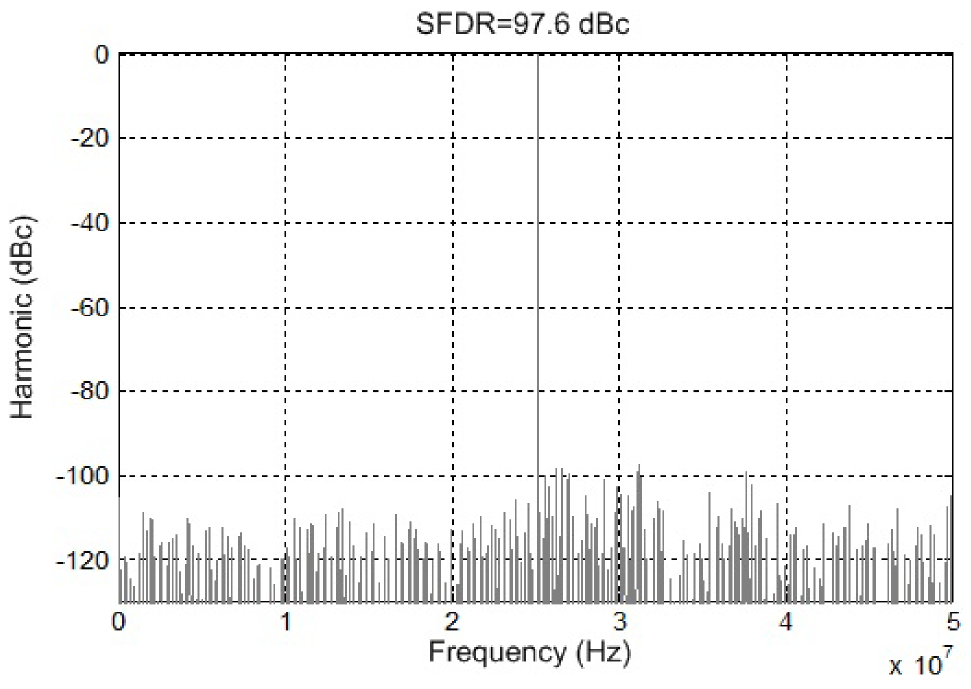


Figure 6.10: Output spectrum for 32 sections per quadrant with non-optimized parabolic approximation

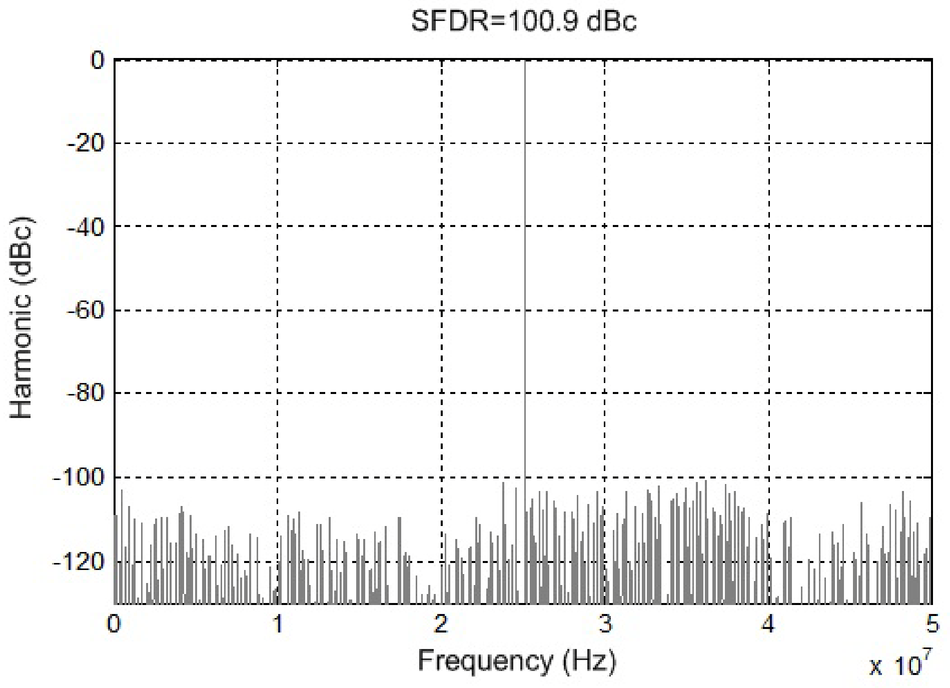


Figure 6.11: Output spectrum for 32 sections per quadrant with optimized parabolic approximation.

In Table 1, we give a comparison between this work and the other techniques in terms of the memory size and SFDR; we can demonstrate that our proposed method presents the best solution for the application that simultaneously needs a small size for the ROM and high spectral purity for the signal.

Table 1: Comparison of ROM size and SFDR for the proposed method and the other compression techniques in the literature

Architecture	**ROM size (Bit)**	**SFDR (dBc)**
This work	480	97.2
Langlois and AL-Khalili [36]	448	84.2
Eltawil and Babak [33]	600	80.1
Belaour et al.[24]	960	77
Nicholas et al. [25]	3072	90.3
Tan and Samueli [38]	16382	96.4

As a result of an optimized piecewise parabolic interpolation with 16 sections per quadrant and 10 bits word size, a value of SFDR 97.2 dBc can be achieved. Based on the tradeoff theory a higher value of SFDR can be obtained by increasing the number of sections per quadrant.

6.7 Implementation and Simulation Results of the proposed chirp generator

The chirp generator as seen in Figure 6.12 is tested by simulating Figure 6.16 and Figure 6.17 within the polynomial evaluation unit in order to generate the optimum chirp. The counter has been set as a 32 bit unsigned up counter.

The initial counter value CO_0 can be set directly in the block properties. The clock source for the counter is defined as the system clock. The accumulator has been set as 32 bit with wrapping output in case of overflow. Also, the initial value of the accumulator AC_0 can be set directly in its properties dialogue box.

For selecting the right bits-set for achieving the required sweep rate *S*, the block *Slice* has been used, which extract a sequence of length *k* bits starting from LSB bit with an offset *p* and represents it as new data value.

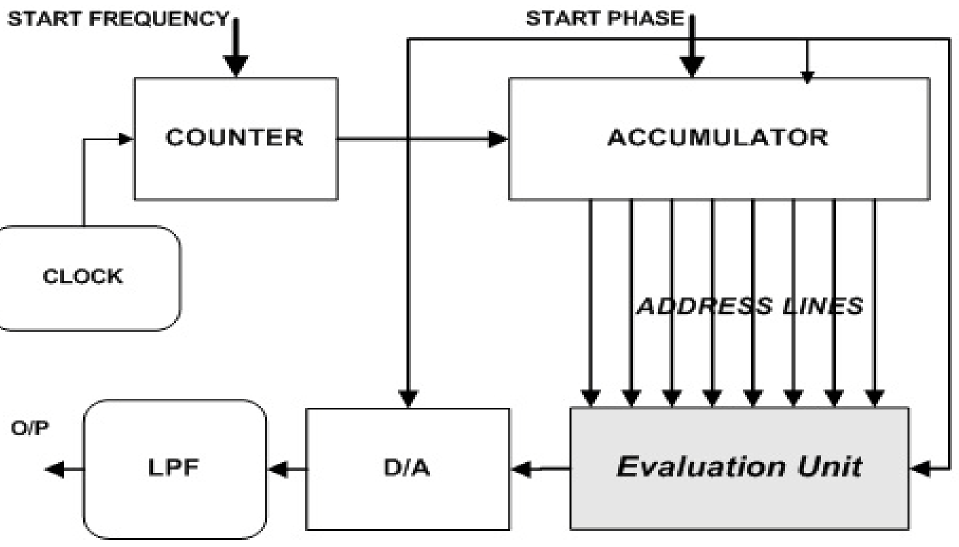


Figure 6.12: Block diagram of the proposed digital chirp generator

The implementation of the sections' interpolation is depicted in Figure 6.17, while the exploitation of the sinusoidal symmetry architecture is achieved by simulating Figure 6.16. These architectures were implemented based on Horner's scheme, which is based on an algorithm for efficiently evaluating the polynomials form and simultaneously describes a manual process by which one may approximate the roots of a polynomial equation. The Horner's scheme can also be viewed as a fast algorithm for dividing a polynomial by a linear polynomial [39].

The following figures show the results after each block in the proposed system. Figure 6.13 shows the linear phase appears after the counter or the first accumulator, and Figure 6.14 illustrates the quadratic phase is produced after applying the accumulator. At the end, a chirp signal is generated from the interpolation evaluation unit, this is depicted in Figure 6.15.

The architectures of Figure 6.16 and Figure 6.17 are combined under the polynomial evaluation unit to be an alternative solution to the structure of Farrow, which is shown in Figure 6.18 [34].

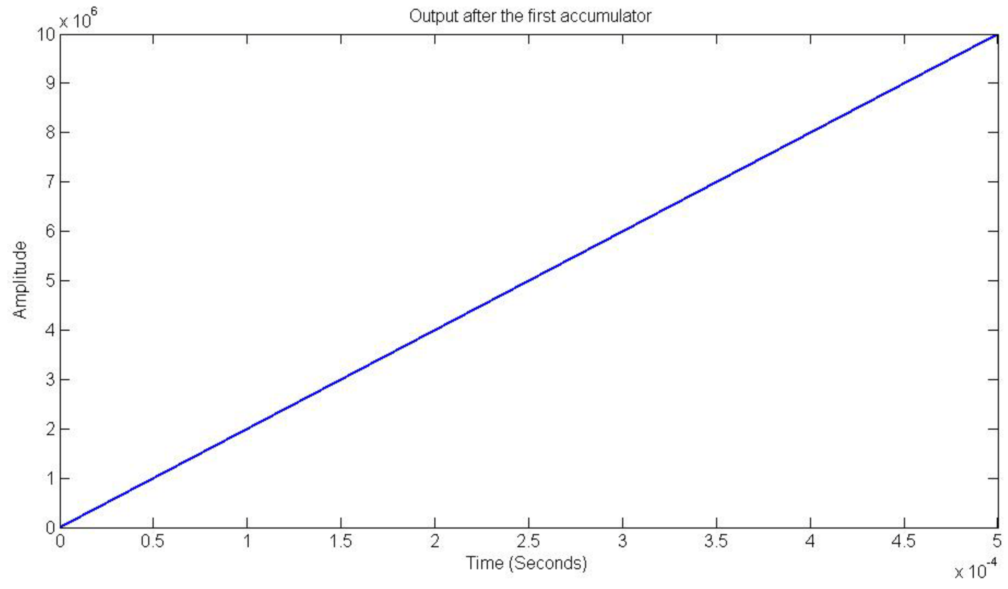


Figure 6.13: The output after the counter or the first accumulator

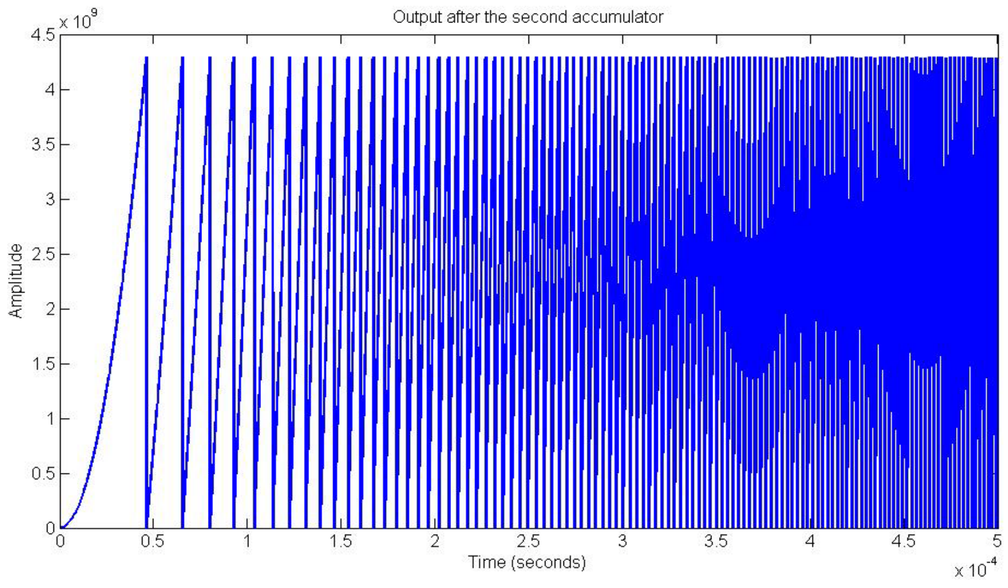


Figure 6.14: The output of the second accumulator

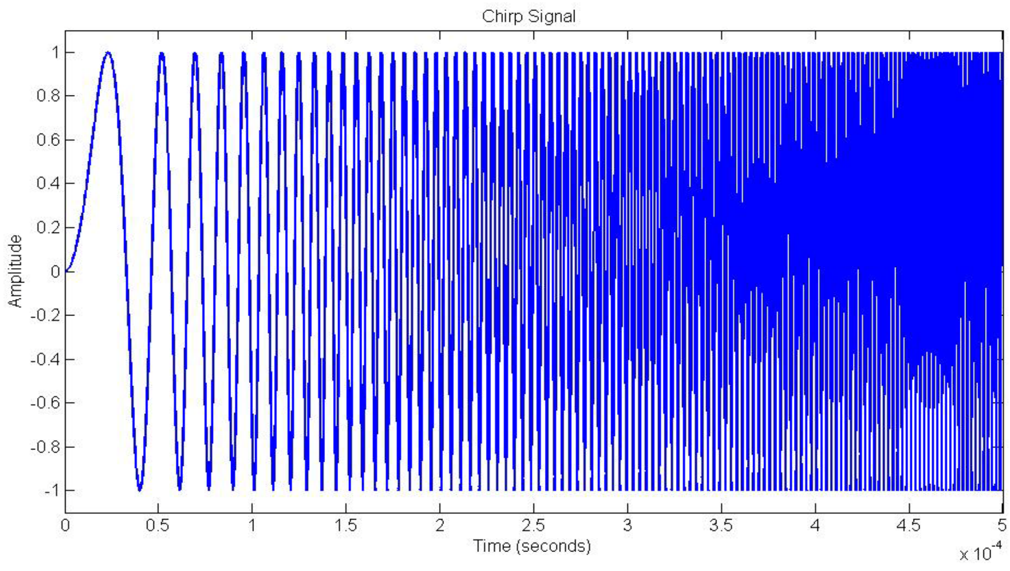


Figure 6.15: The output of the evaluation unit (chirp signal)

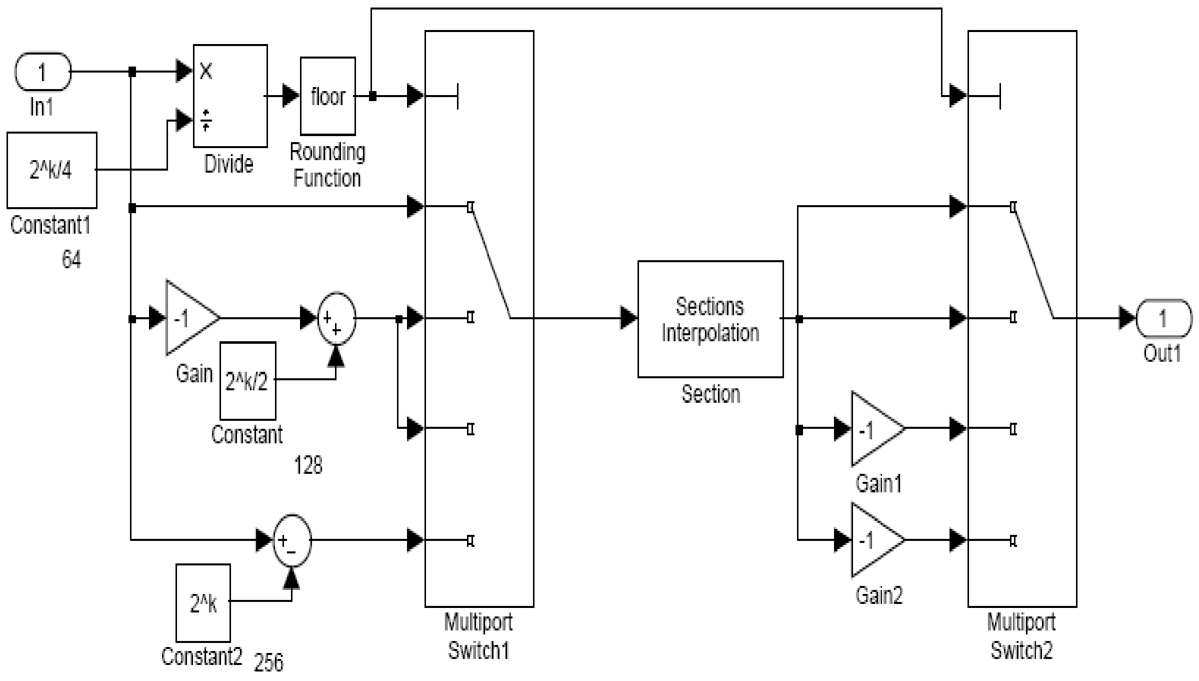


Figure 6.16: The block diagram of the simulated structure works as Farrow structure to generate the sine wave

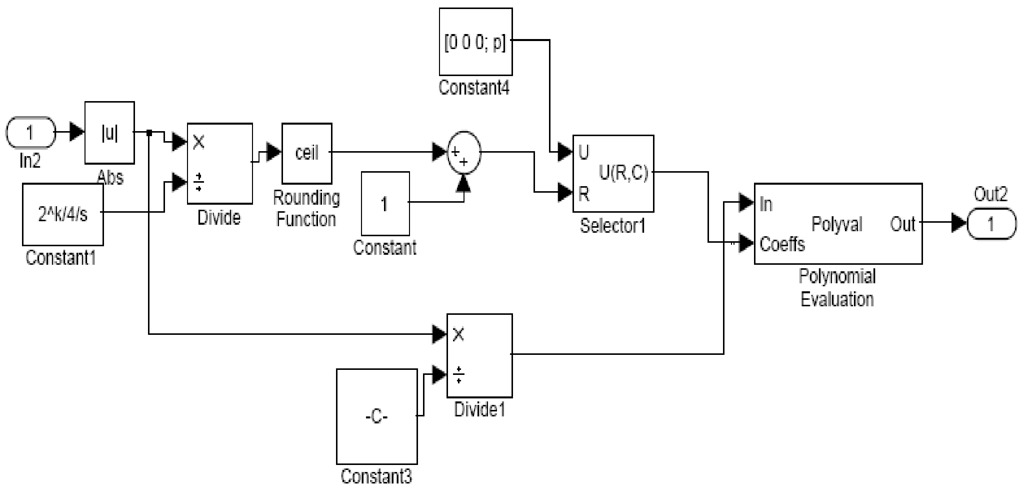


Figure 6.17: The implementation of the section interpolation in Figure 6.16

The proposed architecture is a multirate filter structure which has advantages to make it favourable and more practical than the Farrow structure; these advantages are summarized by choosing a high order FIR filter, simplicity in the implementation, and flexibility with installed parameters in order to generate the interpolation coefficients and reshape the optimum sinusoidal signal, The following figure shows the structure under ROM's mask.

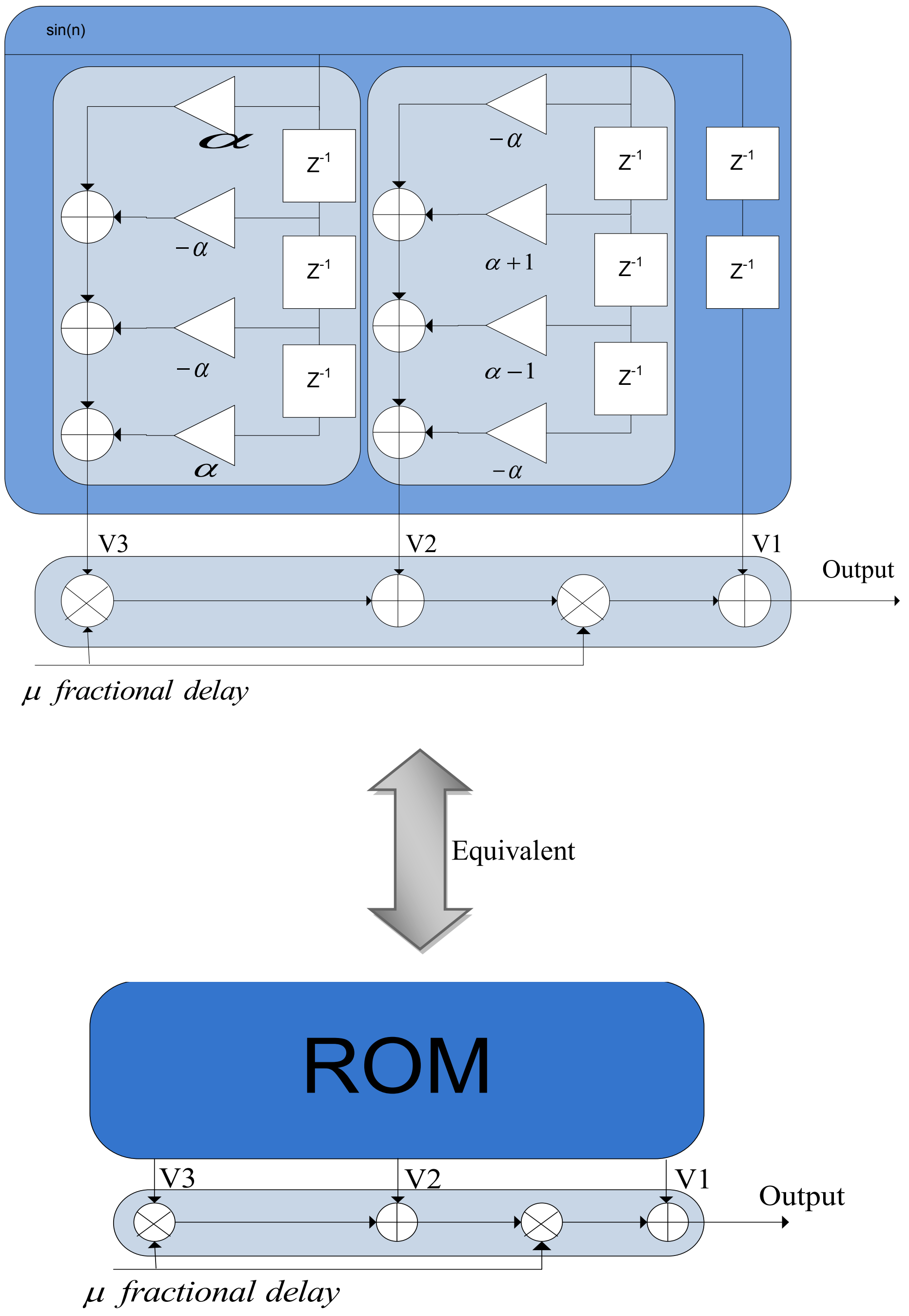


Figure 6.18: Farrow structure

A MATLAB program was written to implement the Pedersen system, where the direct LUT method was used in his approach. The program also evaluated the spectrum based on the proposed method. Assuming $fclk=1/t_p$= 1 GHz, and the chirp signal start from DC CO_0= 0 to reach f_{max}=125 MHz, with 8 address lines $(k=8)$. Therefore, the maximum sweep rate is S_{max}=390x10^{12} Hz/s.

Figure 6.19 and Figure 6.20 show the amplitude spectra of the error sequence belonging to Pedersen's chirp generator based on the LUT technique and the proposed generator with the method based on the piecewise polynomial approximation technique using 16 sections per quadrant. These figures show the efficiency of the interpolation technique, where the level of the spectral purity of the signal is greatly increased.

The obtained results show that as the value of the least significant bits increases, more undesirable spurious harmonic components are generated in Pedersen chirp generator samples, while the piecewise polynomial interpolation method compensate this increment in the distortion by increasing the number of sections per quadrant and the order of the interpolation equation. Thus, the proposed method is an efficient technique which improves the accuracy of the digital chirp signal generator and the spectral purity of the generated signal.

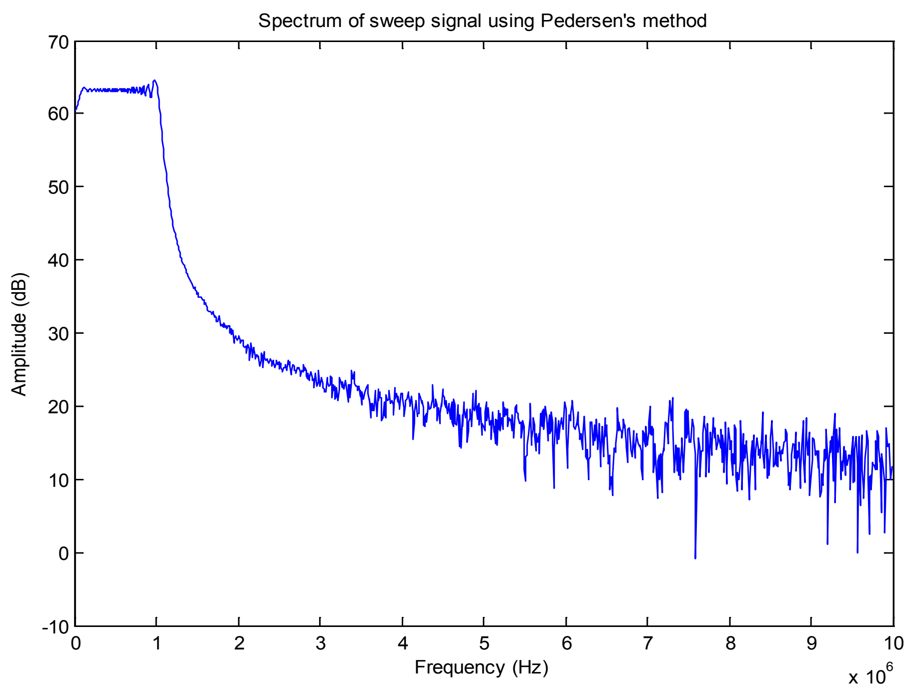


Figure 6.19: Output spectrum of the chirp signal using Pedersen’s method

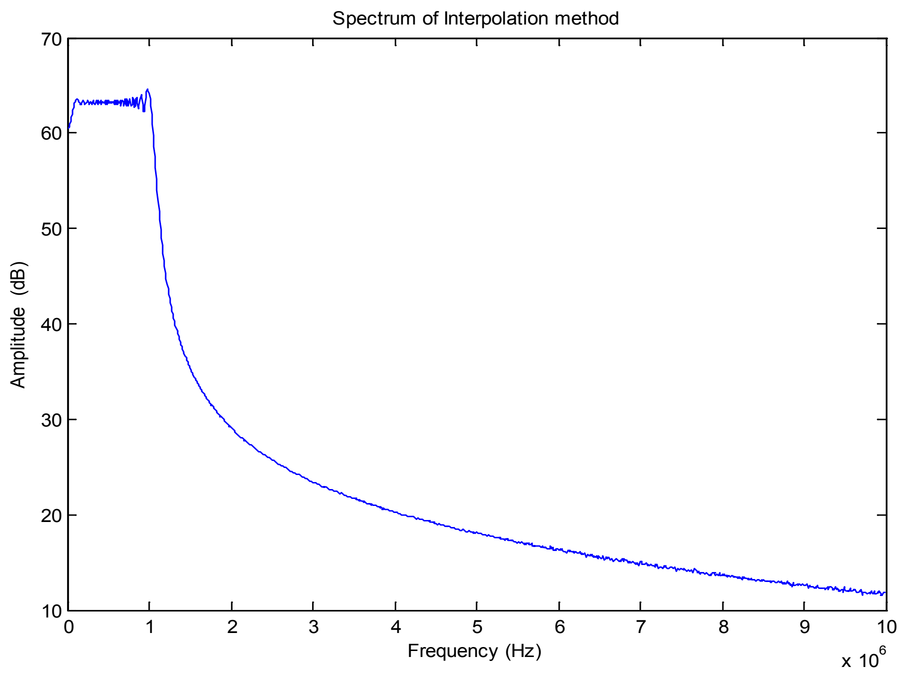


Figure 6.20: Spectrum of the chirp signal using the interpolation method

In the case of a piecewise linear interpolation, only two interpolation coefficients will be generated for each section; therefore the expression in (6.13) will be reformed as in the following:

$$f(x) = \begin{Bmatrix} y_0 + m_0(x - x_0) & x_0 \le x < x_1 & (x_0 = 0) \\ y_1 + m_1(x - x_1) & x_1 \le x < x_2 & \\ \vdots & & \\ y_k + m_k(x - x_k) & x_k \le x < x_{k+1} & \\ \vdots & & \\ y_{s-1} + m_{s-1}(x - x_{s-1}), & x_{s-1} \le x < x_s & (x_s = 1) \end{Bmatrix} \tag{6.28}$$

Figure 6.21 shows the spectra of the signal in the case of linear interpolation approximation, with a different number of sections per quadrant, where the number of section plays an important role in decreasing the spurious harmonic distortion.

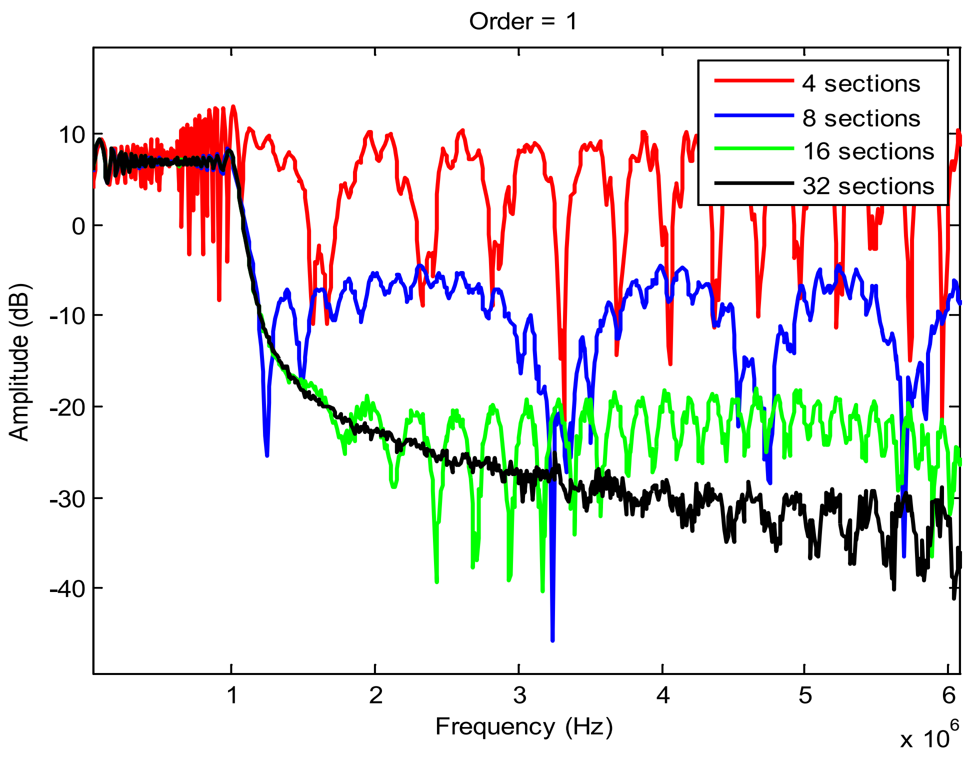


Figure 6.21: Signal spectra in the case of linear interpolation with different number of sections/quadrant

To begin, we have to generate and acquire the interpolation coefficients, and the number of these interpolation coefficients is controlled by the number of sections per quadrant in addition to the order of interpolation equation. For instance, the piecewise parabolic interpolation coefficients with different number of sections per quadrant are shown in tables 1- 4. In the case of 4 sections per quadrant we have to generate $3\times4=12$ coefficients. For 16 sections per quadrant, we have to generate $3\times16=48$ coefficients, etc.

We have noticed from our calculations that the number of sections per quadrant's parameter has no serious effect on the results, when the order of the interpolation equation is equal to four or higher.

Table 2: Interpolation coefficients for 4 secs/quad

V1	V2	V3
-3.8551	6.3797	-0.0005
-10.9489	7.2571	-0.0286
-16.3758	8.5948	-0.11□7
-19.3096	9.6690	-0.2103

Table 4: Interpolation coefficients for 8 secs/quad

V1	V2	V3
-1.9485	6.3078	-0.0001
-5.7405	6.5450	-0.0039
-9.3119	6.9909	-0.0179
-12.5255	7.5922	-0.0462
-15.2577	8.2734	-0.0887
-17.4036	8.9415	-0.1408
-18.8806	9.4924	-0.1922
-19.6321	9.8178	-0.2274

Table 5: Interpolation coefficients for16 secs/quad

V1	V2	V3
-0.9835	6.2895	-0.0000
-2.9108	6.3501	-0.0005
-4.8101	6.4692	-0.0024
-6.6630	6.6432	-0.0065
-8.4518	6.8671	-0.0135
-10.1592	7.1341	-0.0240
-11.7688	7.4360	-0.0381
-13.2650	7.7634	-0.0560
-14.6334	8.1055	-0.0774
-15.8610	8.4507	-0.1017
-16.9358	8.7865	-0.1280
-17.8474	9.0997	-0.1549
-18.5872	9.3769	-0.1808
-19.1480	9.6044	-0.2039
-19.5244	9.7687	-0.2218
-19.7128	9.8566	-0.2321

Table 3: Interpolation coefficients for 32 secs/quad

V1	V2	V3
-0.4995	6.2848	-0.0000
-1.4671	6.3002	-0.0001
-2.4312	6.3305	-0.0003
-3.3894	6.3757	-0.0008
-4.3395	6.4353	-0.0018
-5.2791	6.5089	-0.0032
-6.2059	6.5960	-0.0053
-7.1179	6.6959	-0.0080
-8.0126	6.8080	-0.0115
-8.8881	6.9313	-0.0159
-9.7422	7.0649	-0.0211
-10.5728	7.2078	-0.0272
-11.3779	7.3590	-0.0343
-12.1556	7.5171	-0.0424
-12.9040	7.6810	-0.0513
-13.6214	7.8492	-0.0612
-14.3059	8.0205	-0.0719
-14.9559	8.1933	-0.0834
-15.5700	8.3661	-0.0956
-16.1465	8.5373	-0.1083
-16.6841	8.7054	-0.1214
-17.1815	8.8687	-0.1348
-17.6376	9.0255	-0.1483
-18.0511	9.1742	-0.1617
-18.4212	9.3130	-0.1747
-18.7469	9.4403	-0.1871
-19.0274	9.5542	-0.1987
-19.2621	9.6532	-0.2091
-19.4504	9.7356	-0.2181
-19.5918	9.7997	-0.2254
-19.6860	9.8438	-0.2306
-19.7328	9.8664	-0.2333

As we will notice, Figure 6.22 and Figure 6.24 are devoted to proving two important factors in our method which should be adjusted in order to achieve the minimum harmonic distortion. These factors are the order of the interpolation equation and the number of sections per quadrant.

We can clearly see in Figure 6.22 how important increasing the order of the interpolation equation is in order to decrease the spurious harmonic distortion. Figure 6.24 shows the spectra of the chirp signals with different number of sections per quadrant and with third order interpolation equation

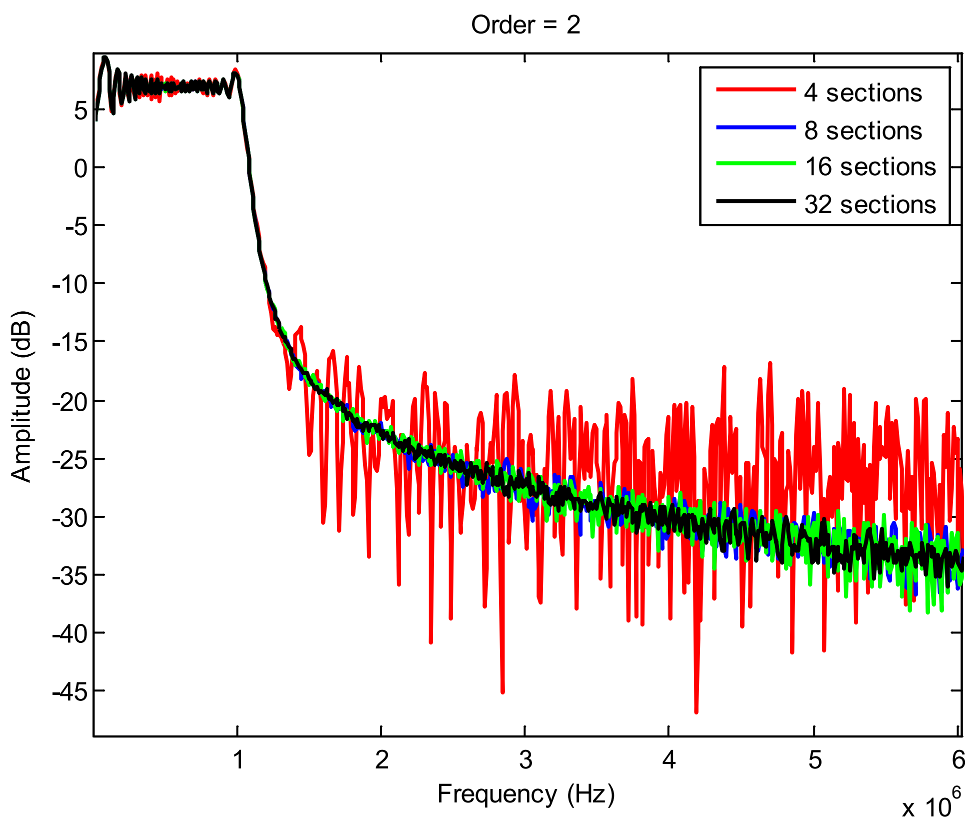


Figure 6.22: Signal spectra in the case of piecewise parabolic interpolation with different number of sections/quadrant.

In the following figure, we can see the generated chirp signal by our proposed method and at the same time show the phase continuity of the signal.

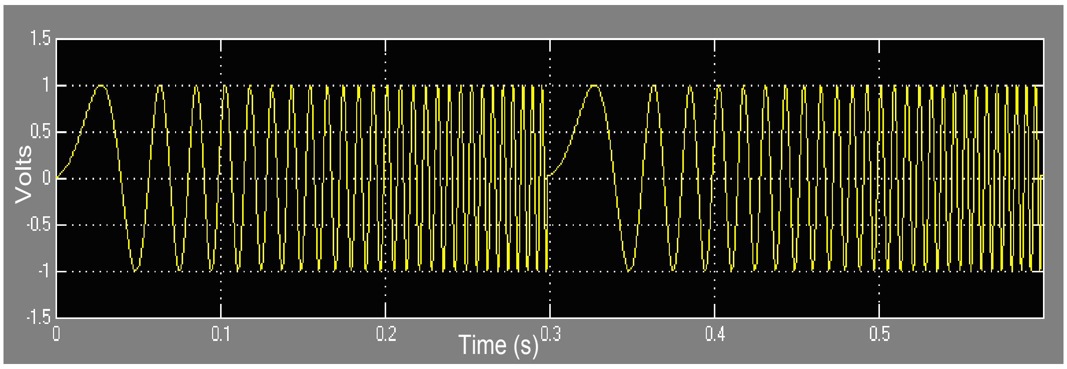


Figure 6.23: The generated coherent swept signal by the new technique (2 periods)

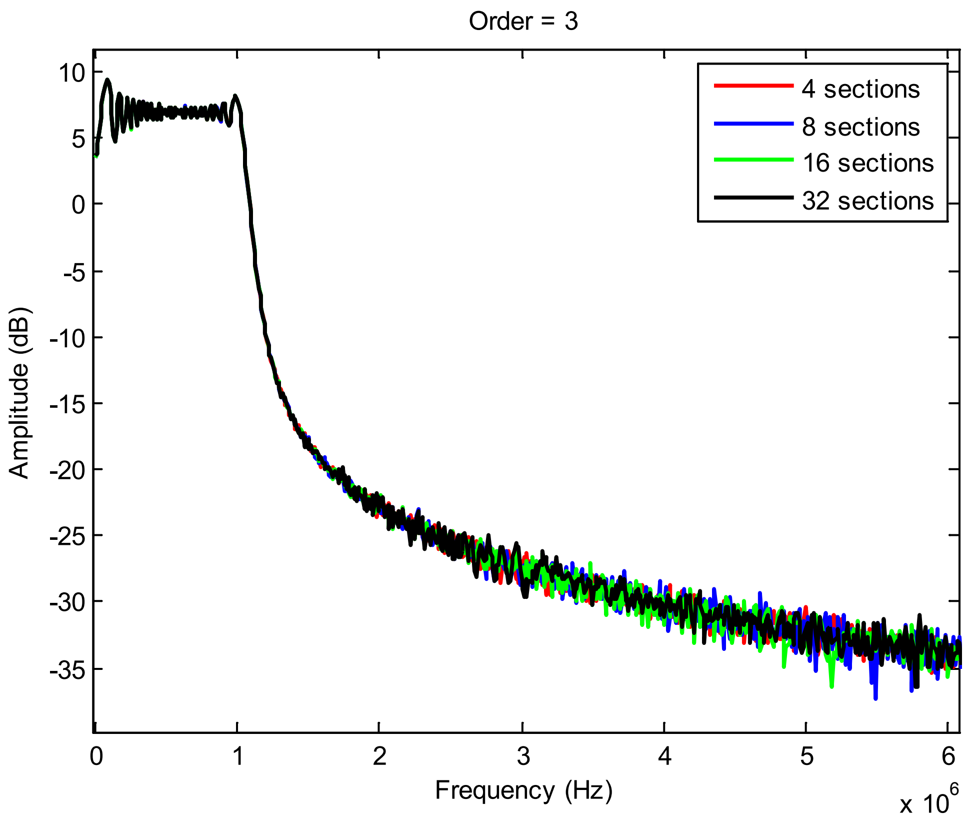


Figure 6.24: Signal spectra in the case of third order interpolation equation and different number of sections/quadrant

7 *Hardware Implementation of the Proposed System*

7.1 Introduction

This chapter shows the practical aspect of the dissertation and the realization of the proposed system using the Field Programmable Gate Array (FPGA) technology. Here we will provide a complete description of the implementation of the proposed interpolation method for generating the chirp signal and the used tools. Our goal, by the end of this chapter, is to show the hardware realization of the digital chirp generator with a bandwidth of up to 320MHz.

The second section will show the road map for using the platform and the summary for the most important features of this kit, while the third section shows step by step the implementation and programs for the proposed digital chirp generator.

The last section of this chapter shows a smart and unique parallel processing technique for overcoming the limited speed's problem of the FPGA.

7.2 Features of the used Platform

7.2.1 X5-TX Platform

X5-TX, as shown in Figure 7.1 and described in [40], is a flexible all in one solution platform that includes all required hardware, firmware and software for demanding applications involving generating complex high speed arbitrary waves used in wireless transmitters, RADAR, etc.

This platform provides a high level of abstraction that allows an easier and speedy development without any significant RTL design knowledge (VHDL or Verilog), which makes it a powerful tool for ongoing research [40].

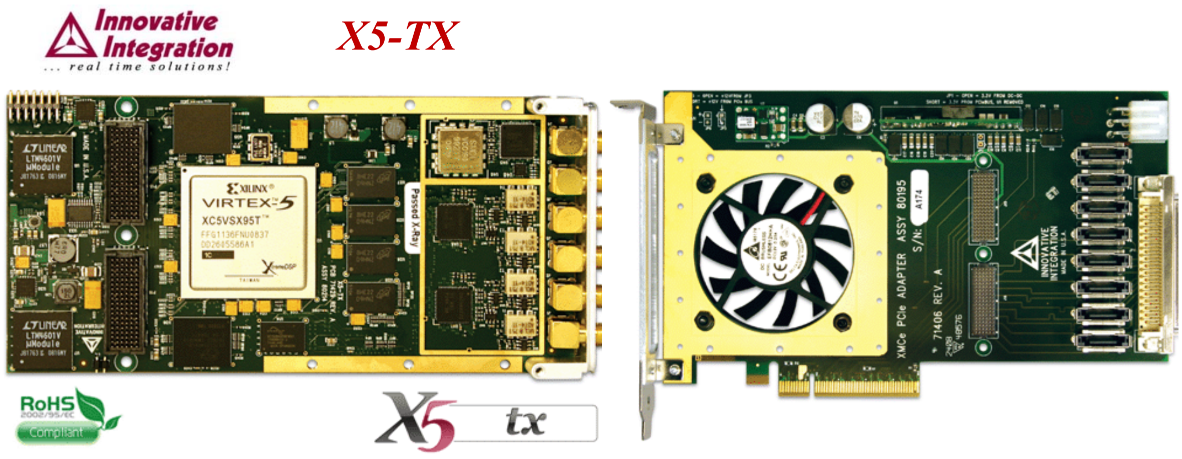


Figure 7.1: Xilinx VIRTEX-5 Innovative Integration X5-TX

X5-TX features a high speed DAC with a bandwidth up to true 1 GSPS and 16 bit resolution, a powerful Virtex5 FPGA computing core, large external DRAM, SRAM memory, and eight lanes PCI-Express host interface for a fast data transfer rate with the host PC, as illustrated in Figure 7.2. It can be fully customized using *VHDL* and/or *MATLAB* using the *Framework Logic toolset*. The *MATLAB* Board Support Package (*BSP)* supports real-time hardware-in-the-loop development using the graphical *Simulink* environment with *Xilinx System Generator* [41, 42].

The synthesized waveform data can be fed to the DAC using one of the following methods:

- Stream mode: system playbacks data stream from the host PC via PCI-Express interface which is limited by the PCI-Express bandwidth. It gives 2 GBPS with 250 MBPS for 8 lanes.
- Pattern mode: Where the system playbacks pattern data stored in the platform memory.

- IP logic: the system calculates the waveform data using logic implemented in the on-board Virtex5 FPGA. The maximum clock is one of the limitations of this type.

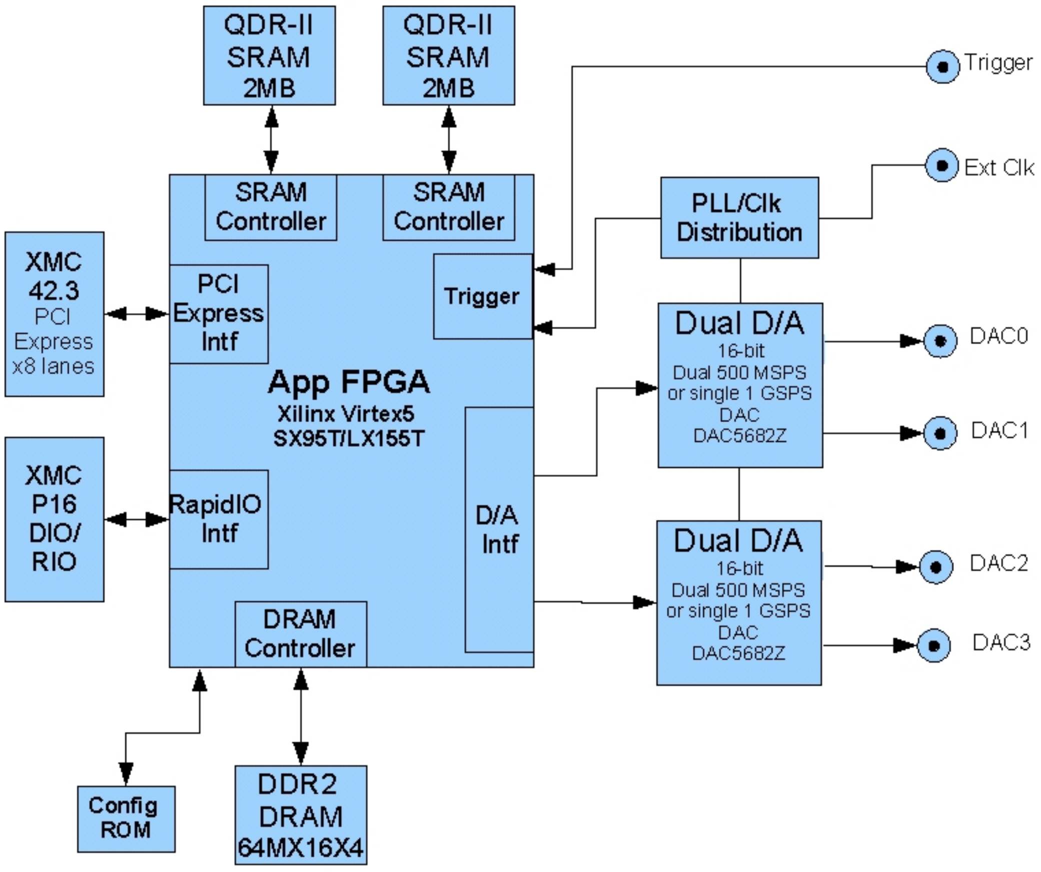


Figure 7.2: X5-TX Block Diagram

7.2.2 DAC5682Z (Digital-to-Analog Converter)

The DAC5682Z is a dual-channel 16-bit 1.0 GSPS digital-to-analog converter (DAC) with wideband Low Voltage Differential Signal (LVDS) data input, integrated 2x/4x interpolation filters, on-board clock multiplier and internal voltage reference. DAC5682Z, as illustrated in Figure 7.3, offers superior linearity, noise, crosstalk and PLL phase noise performance [43].

DAC5682Z integrates a wideband LVDS port with on-chip termination. The on-chip delay lock loop (DLL) simplifies LVDS interfacing by providing skew control for the LVDS input data clock.

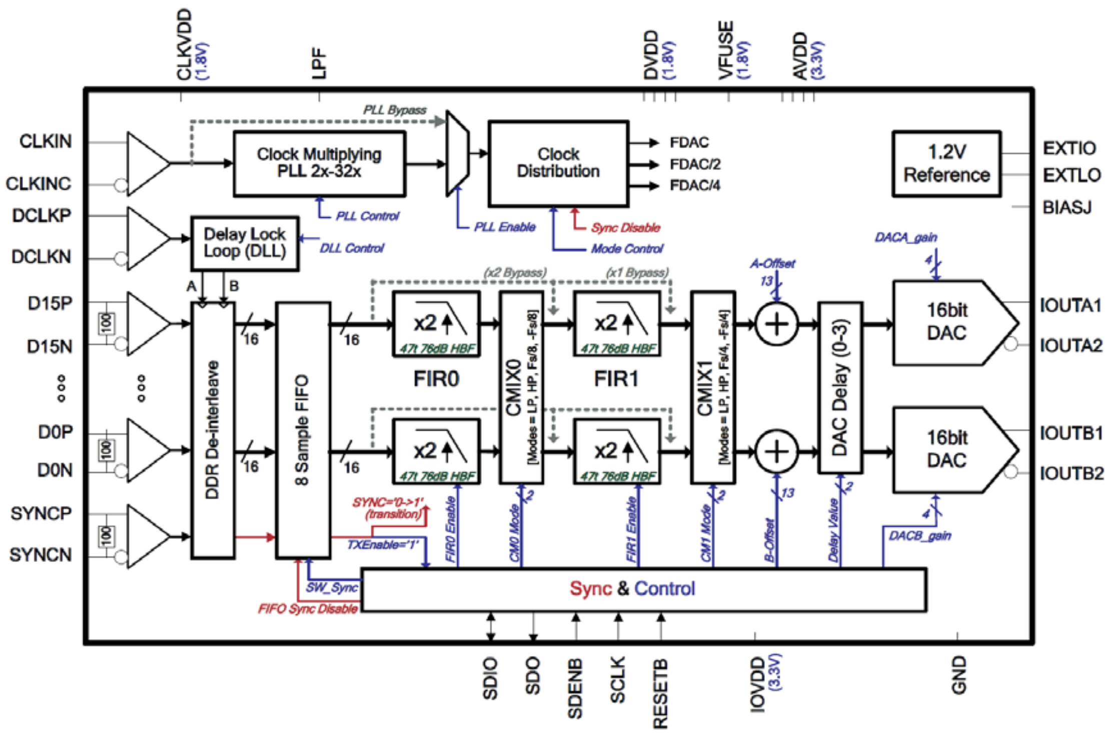


Figure 7.3: DAC functional block diagram

LVDS half-rate data clock (DCLKP/DCLKN) is provided by the FPGA and it is generated by a toggling data bit to maintain LVDS data to DCLK timing alignment. LVDS data relative to DCLK is input using Double Data Rate (DDR) switching and using both rising and falling edges as shown in Figure 7.4.

Interfacing very high-speed LVDS data and clocks presents a big challenge to system designers as they have unique constraints and are often implemented with specialized circuits in order to increase the bandwidth. One such specialized LVDS circuit used in many FPGAs and ASICs is a SERializer-DESerializer (SERDES) block. For interfacing to the DAC5682Z, only the SERializer functionality of the SERDES block is required. SERDES drivers accept lower rate parallel input data and output a serial stream using a shift register at a frequency multiple of the data bit width. For example, a 4-bit SERDES block can accept parallel 4-bit input data at 250 MSPS and output serial data 1000 MSPS.

External clock distribution for FPGA and ASIC SERDES drivers often have a chip-to-chip system constraint of a limited input clock frequency compared to the desired LVDS data rate. In this case, an internal clock multiplying PLL is often used in the FPGA or ASIC to drive the high-rate SERDES outputs. Due to this possible system clocking constraint, the DAC5682Z accommodates a scheme where a toggling LVDS SERDES data bit can provide a “data driven” half-rate clock (DCLK) from the data source. A DLL on-board the DAC is used to shift the DCLK edges relative to LVDS data to maintain internal setup and hold timing.

To increase bandwidth of a single 16-bit input bus, the DAC5682Z assumes Double Data Rate (DDR) style interfacing of data relative to the half-rate DCLK. Figure 7.4 provides an example of implementation using FPGA-based LVDS data and clock interfaces to drive the DAC5682Z. In this example, an assumed system constraint is that the FPGA can only receive a 250 MHz maximum input clock while the desired DAC clock is 1000 MHz. A clock distribution chip such as the CDCM7005 or the CDCE62005 is useful in this case to provide frequency and phase locked clocks at 250 MHz and 1000 MHz.

From the example provided by Figure 7.4, driving LVDS data into the DAC using SERDES blocks requires a parallel load of 4 consecutive data samples to shift registers. Color is used in the figure to indicate how data and clocks flow from the FPGA to the DAC5682Z. The figure also shows the use of the SYNCP/N input, which along with DCLK, requires 18 individual SERDES data blocks to drive the DAC’s input data FIFO that provides an elastic buffer to the DAC5682Z digital processing chain [43].

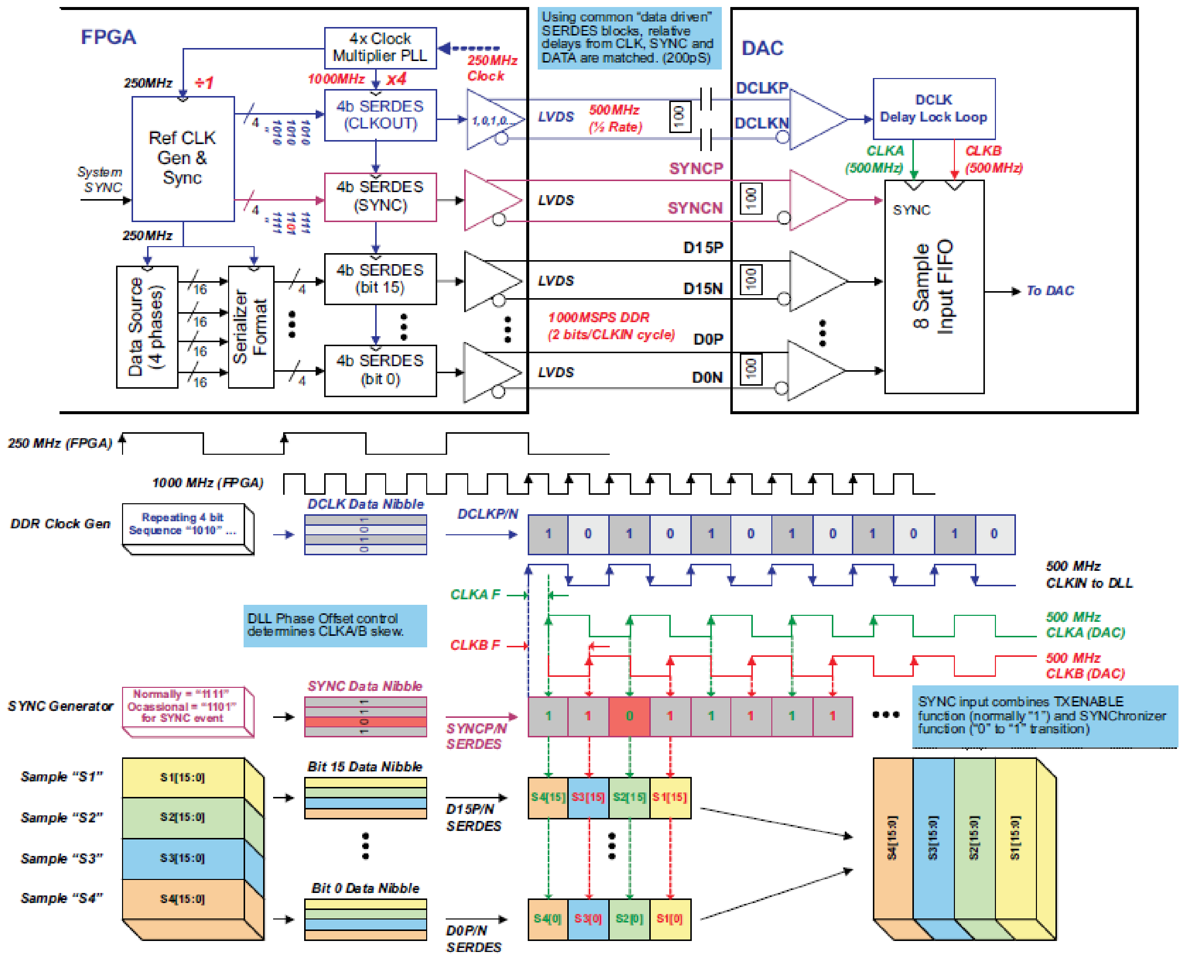


Figure 7.4: Data Flow to DAC (taken from [43])

Due to the sampled nature of high-speed DAC's, the well known *sin(x)/x* or (*sinc(x)*) response can significantly attenuate higher frequency output signals. Figure 7.5 shows the normalized *sinc* attenuation roll-off with respect to the final DAC sample rate in 4 Nyquist zones. For example, if the final DAC sample rate F_S = 1.0 GSPS, then a tone at 440MHz is attenuated by 3.0 dB. Although the *sinc* response can create challenges in frequency Planning, one side benefit is the natural attenuation of Nyquist images.

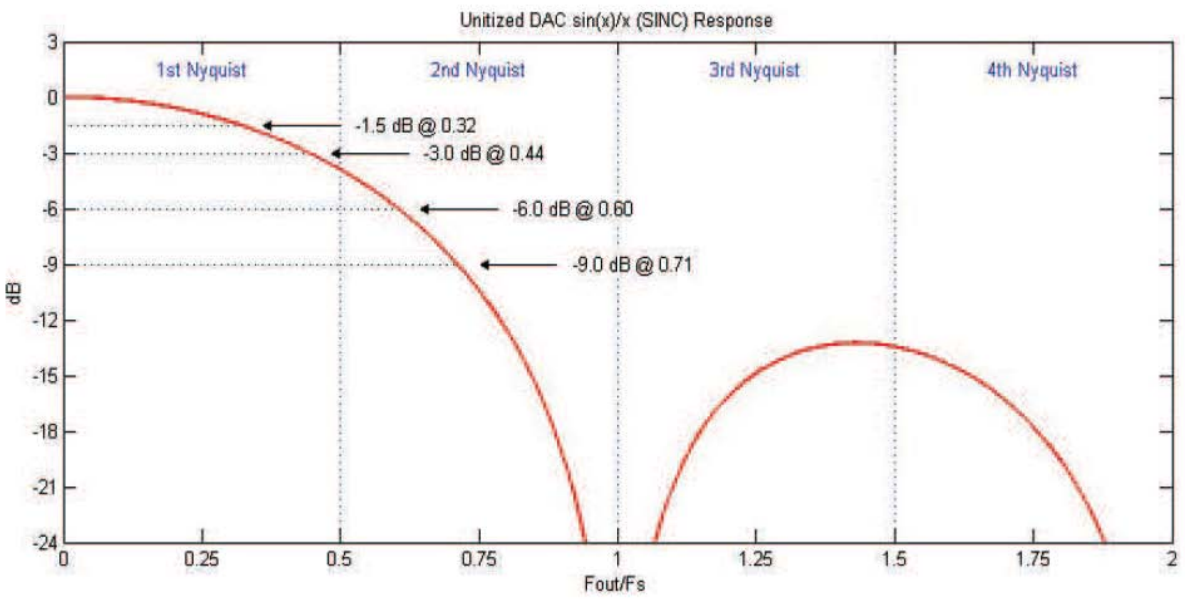


Figure 7.5: Sinc response of the DAC

7.2.3 The Used FPGA

A Xilinx Virtex5 SX95T, with external 512 MB DDR2 DRAM and 4MB QDR-II memory, provides a very high performance DSP core for demanding applications such as RADAR and direct RF digitizing [44]. X5-TX platform has the close integration of the analog IO, memory and host interface with the FPGA enables real-time signal processing at rates exceeding 300 GMAC/s. The X5 XMC modules couple Innovative's powerful Velocia architecture with a high performance, 8-lanes PCI Express interface that provides over 1 GB/s sustained transfer rates to the host. Private links to host cards with >1.6 GB/s capacity using P16 are provided for system integration[45].

7.2.4 Software and the used blocksets in the HDL language

X5-TX can be programmed using an HDL language (VHDL, Verilog) or Simulink accompanied with System Generator tool from Xilinx. Another possibility is to develop a part of the design in Simulink and the rest using HDL language and then combine the two parts.

The advantages of using Simulink over the HDL language is the higher level of abstraction that allows the design to be automatically compiled into an FPGA logic at the push of a button without any significant RTL design knowledge. Also, it provides access to underlying FPGA resources through low level abstractions, allowing in principle the construction of highly efficient FPGA designs.

The System Generator tool allows device-specific hardware designs to be constructed directly in Simulink modeling environment. It allows designs to be composed from a variety of ingredients. Data flow models, HDL codes and functions derived from MATLAB can be used side-by-side, simulated together, and synthesized into working hardware.

The main difference between Simulink FPGA-targeted designs and typical Simulink designs is the usage of special blocks for the logic that should be implemented in the FPGA, called System Generator blocks. These blocks can be either Xilinx standard blocksets or 3rd party ones, and they are described as follows:

Xilinx Standard Blockset: It provides abstractions of mathematical, logic, memory, and DSP functions that can be used to build and debug high performance DSP systems in Simulink. Xilinx blockset contains functions for signal processing such as FIR Filters and FFTs, error correction (i.e. Viterbi decoder, Reed-Solomon encoder/decoder), arithmetic, memories (e.g. FIFO, RAM,) and digital logic. There are also blocks that provide interfaces to other software tools (e.g., FDATool, ModelSim).

Innovative Integration Blockset: *It* is a 3rd party blockset containing functions for accessing various features on X5-TX board such as DDR memory, ADC, DAC, SBSRAM, Rocket IO, PCI-Express…etc. They can be thought of drivers for the X5-TX hardware.

Bit-True and Cycle-True Modeling: Using the SysGen blocks guarantees that the simulations are bit-true and cycle-true. To say a simulation is bit-true means, that at the boundaries (i.e., interfaces between System Generator blocks and non-System Generator blocks,) a value produced in simulation is bit-for-bit identical to the corresponding value produced in the hardware. To say a simulation is cycle-true means, that at the boundaries, corresponding values are produced at corresponding times. The boundaries of the design are the points at which System Generator gateway blocks exist. When a design is translated into hardware, Gateway In (respectively, Gateway Out) blocks become top-level input (resp., output) ports.

Timing and Clocking: The designs in the System Generator are discrete time systems. In other words, the signals and the blocks that produce them have associated sample rates. A block's sample rate determines how often the block is awoken (allowing its state to be updated). The System Generator sets most sample rates automatically. A few blocks, however, set sample rates explicitly or implicitly[46].

A simple System Generator model, as depicted in Figure 7.6, illustrates the behaviour of discrete time systems. Consider the model shown below; it contains a gateway that is driven by a Simulink source (Sine Wave,) and a second gateway that drives a Simulink sink (Scope) [47].

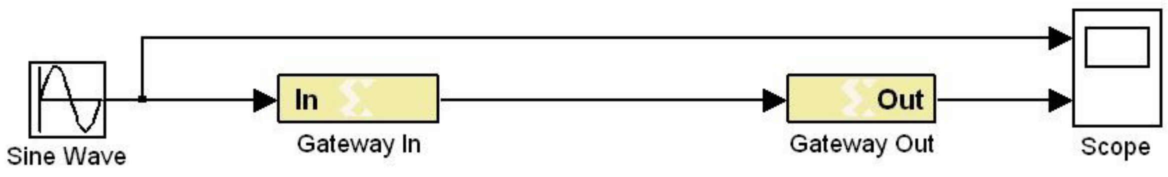


Figure 7.6: A simple System Generator model

The *Gateway In* block is configured with a sample period of one second. The *Gateway Out* block converts the Xilinx fixed-point signal back to a double (so it can be analyzed in the Simulink scope) but does not alter sample rates. The scope output, as shown in Figure 7.7, shows the unaltered and sampled versions of the sine wave [46].

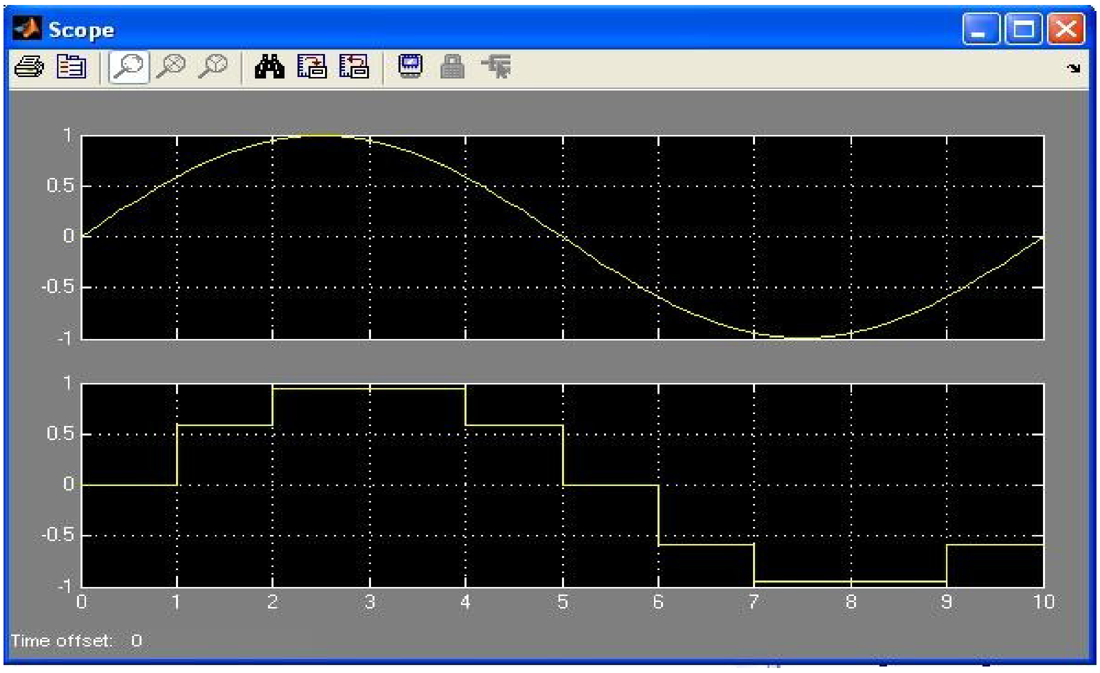


Figure 7.7: unaltered and sampled versions of the sine wave

According to the design flows using the system generator, which can be used in many settings, the following is a brief description for each setting:

- ***Algorithm Exploration:*** SysGen can be used for algorithm exploration, design prototyping and model analysis. It helps to get a feel for the design problems that are likely to be faced, and perhaps to estimate the cost and performance of an implementation in hardware. In this setting, a designer assembles key portions of the design without worrying about fine points or detailed implementation with little need to translate the design into hardware. Simulink blocks and MATLAB (.m) code provide stimuli for simulations and for analyzing results.

- ***Hardware co-simulation:*** Creating a "**FPGA-in-the-Loop**" simulation target, the System Generator provides hardware co-simulation, which incorporates a design running in an FPGA directly into a Simulink simulation. When the design is simulated in Simulink, the results for the compiled portion are calculated in hardware and delivered to Simulink.
- ***Implementing a Complete Design:*** In this setting, the whole system is implemented using SysGen, where SysGen translates the design into HDL and generates the required files to process the HDL code using downstream tools. Figure 7.8 shows the design hierarchy of the work process on the platform, which can achieve all the previous settings.
-

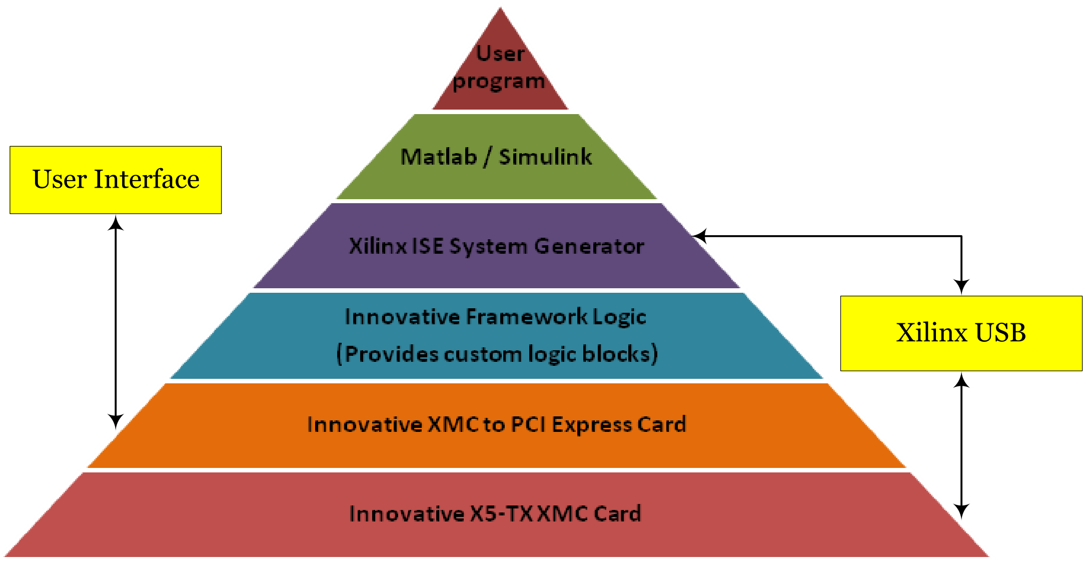

User program
Matlab / Simulink
Xilinx ISE System Generator
Innovative Framework Logic (Provides custom logic blocks)
Innovative XMC to PCI Express Card
Innovative X5-TX XMC Card
User Interface
Xilinx USB

Figure 7.8: Design Hierarchy (edited and taken from [46])

Automatic code generation: Using SysGen provides different kind of compilation depending on the purpose of usage:

- Two types of Netlists, HDL Netlist and NGC Netlist
- Bitstream - produces an FPGA configuration bitstream that is ready to run in a hardware FPGA platform
- EDK Export Tool - for exporting to the Xilinx Embedded Development Kit
- Various varieties of hardware co-simulation (i.e. Innovative Integration products)
- Timing Analysis - a report on the timing of the design

7.3 Programming the FPGA on the platform X5-TX to generate the chirp signal

In this section we will show the description of the Simulink model and develop the equivalent synthesizable model of the chirp signal and after that combine it with the X5-TX infrastructure. In order to be able to execute the previously described algorithm for generating the chirp signal on FPGA, an equivalent synthesizable system should be developed using only Xilinx blocks in Simulink.

7.3.1 Description of the Simulink model

As mentioned in chapters 2 and 6, the chirp signal is a sinusoidal function with a quadratic phase term of time *$\phi(t)$*, and it is corresponding to linearly varying frequency (the sweep rate, S), as shown in the following equation:

$$\sin\phi(t) = \sin(\pi S t^2 + w_1 t + \phi_0) \qquad t \geq 0 \tag{7.1}$$

where ω_1, ϕ_0 are the starting frequency and starting phase, respectively.

For the implementation of this equation, a counter and an integrator have been used to generate the required phase, where the presetting of the counter initial value is CO_0 and the one of the accumulator is AC_0, specifying the start up frequency ω_1 and the starting phase ϕ_0.

Figure 7.9 shows the system described in the previous chapter, which will be a reference model and the starting point for developing the FPGA-synthesizable model.

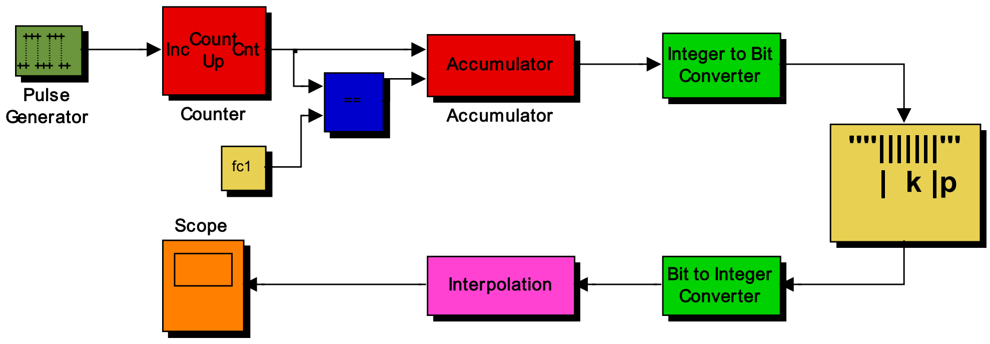


Figure 7.9: Simulink model of the proposed system

7.3.2 Developing the equivalent synthesizable model

The interpolation block in the last figure contains the compression algorithm that has the following functions:

- Exploitation of the sine function symmetry
- Execution of the piecewise polynomial interpolation algorithm

For implementing the reference model shown in Figure 7.9 in FPGA, an FPGA-synthesizable model should be developed by replacing each block with an equivalent from the Xilinx library. Figure 7.10 shows the equivalent of the model depicted in Figure 7.9.

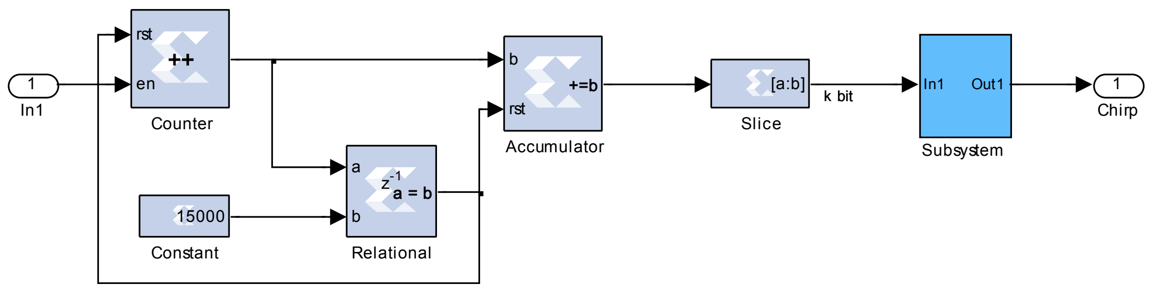


Figure 7.10: Equivalent FPGA implementation of the Simulink model of Figure 7.9

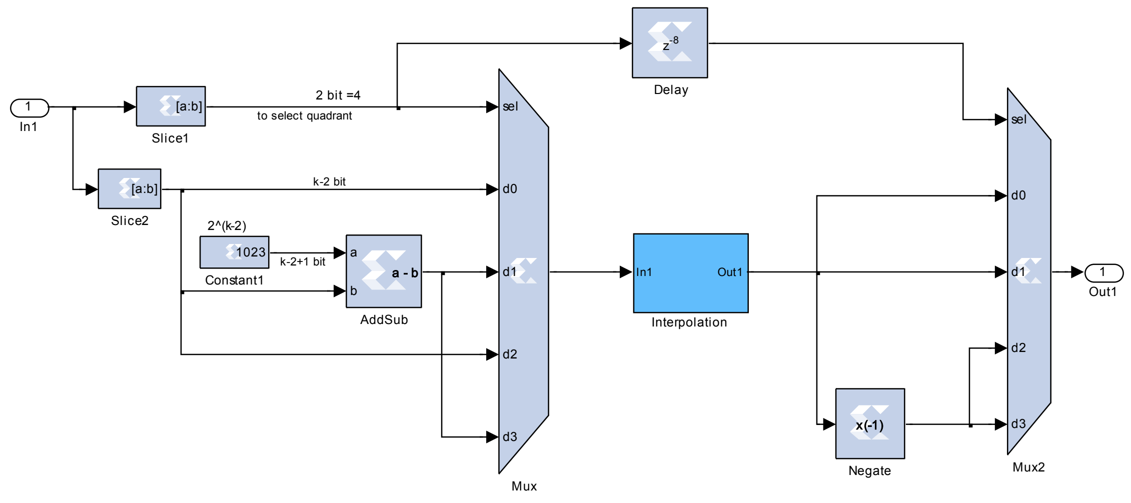
The counter and the accumulator in the reference model are directly replaced with their equivalent from Xilinx library.

Now, the implementation of the *Interpolation* block is done by two steps:

- Achieving the quarter compression
- Piecewise polynomial (parabolic) interpolation

In Figure 7.11 an equivalent sub-system to the one of Figure 7.10 is depicted. The *Slice1* block extracts the most two significant bits of the phase $\phi(t)$ which can be used to indicate in which quarter the phase $\phi(t)$ is located. *Slice2* block extracts the rest bits of the phase $\phi(t)$ and produces $\phi_{-2}(t)$. The *Mux* block is used to find the relevant angel of the phase $\phi_{-2}(t)$ in the first quarter based on the quarter number. When the phase $\phi(t)$ is located in the first or third quarter, it is simply routed directly to the *Interpolation* block. However, in the case of the second and fourths quarter the phase is complemented by subtracting 2^{k-2} from the $\phi_{-2}(t)$ before it is routed to the *Mux* output.

The sine value of phase $\phi(t)$ is identical to the sine value of its relevant angle $\phi_{-2}(t)$ in the first and second. However, it should be negated when $\phi(t)$ is in the third or fourth quarter. The value of the *sine* is delivered through *Mux2* block which uses the quarter number to select the right value from its input.



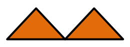
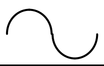
				
Phase in the 1st quarte of the sine	Generating the phase for each quadrant	The *Mux* block is used to find the relevant angel of the phase $\phi_{-2}(t)$ in the first quarter based on the quarter number	The sine value of phase $\phi(t)$ is identical to the sine value of its relevant angle $\phi_{-2}(t)$ in the first and second. However, it should be negated when $\phi(t)$ is in the third or fourth quarter	The value of the *sine* is delivered through *Mux2* block which uses the quarter number to select the right value from its input 2

Figure 7.11: Equivalent FPGA architecture to the Simulink architecture for the proposed chirp generator

Piecewise polynomial algorithm

The algorithm is implemented by dividing the first quarter into $s = 2^q$ sections, in each section the sine value is calculated by quadratic interpolation using a dedicated parabolic curve for each section specified by a set of coefficients b_n. To find out in which section the current phase is located, the phase value is divided by a factor $\frac{2^{k-2}}{s}$ and rounded to the next higher value. The result represents the q MSB bits in $\phi_{-2}(t)$. This q is used as an input to the *Selector* block which selects the right set of coefficients from the LUT in *Constants* block. The *Polyval* block as shown in Figure 6.17 calculates the sine value of the phase $\phi_{-2}(t)$ using parabolic interpolation, which is defined by the corresponding section coefficients and the scaled value of $\phi_{-2}(t)$. Scaling is performed by dividing $\phi_{-2}(t)$ by $\frac{2^{k-2}}{1/4}$.

To implement the interpolation algorithm mentioned above in FPGA, an equivalent circuit is developed as in Figure 7.12.

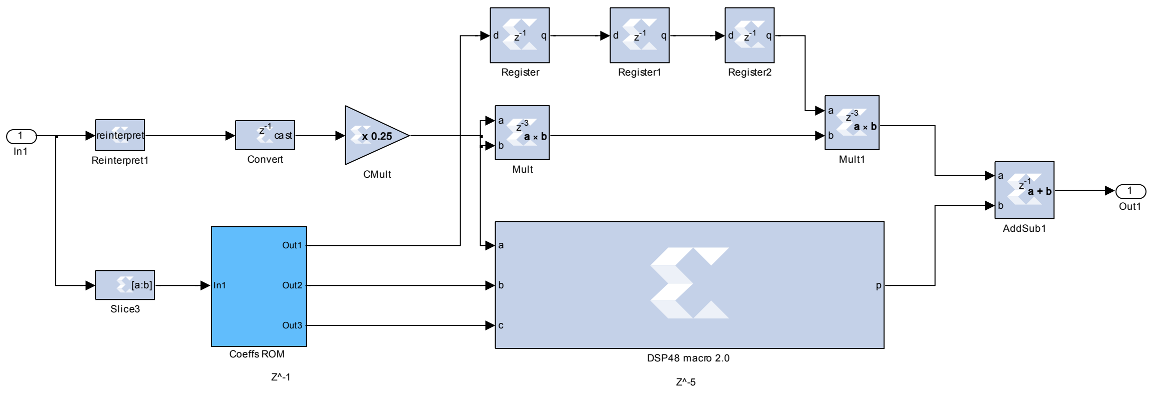


Figure 7.12: The FPGA implementation of the interpolation algorithm

The *Slice3* block extracts the q MSB bits which are used as the address lines for the *Coeff ROM* block which is the LUT that stores a set of three coefficients for each section. At the same time, Figure 7.12 includes the implementation of the *Coff ROM* block.

In Figure 7.12, the *Reinterpret1* and the *CMult* blocks act as a divider; first it moves the binary point to the left $k-2$ digit positions and then divides it by 4. The *Convert* block changes the data type of the signal from unsigned 10_10 to signed 11_10 where the additional bit is added to accommodate the sign bit.

The interpolation calculation is performed, and the DSP48 macro 2.0 block calculates the $\mathrm{b.x+c}$ part where the $\mathrm{a.x^2}$ part is calculated separately as shown in Figure 7.13.

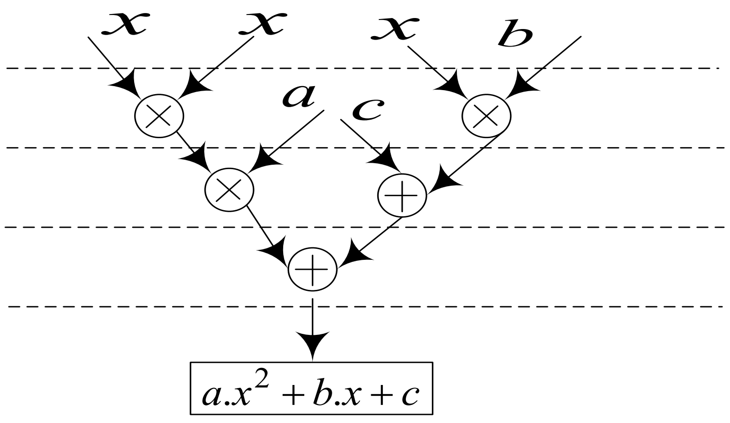


Figure 7.13: The performance of the interpolation calculations

Synchronizing the inter-results:

Different delays have been inserted in the system to compensate the different processing time in each block. For example, in case of $\phi_{-2}(t)$ is crossing from a section to the next one, a new value of x will be available before the new 3 coefficients sets corresponding to the new section are available at the output of the *Coeff ROM* block, therefore a delay of one clock is implemented in the *Convert* block. Similarly, a delay of 3 clocks is inserted by using 3 register blocks with 1 clock delay for each to compensate the delay in the *Mult* which needs 3 clock cycles to complete its operation. The complete design of the chirp signal generator is packed in a subsystem as shown in Figure 7.14.

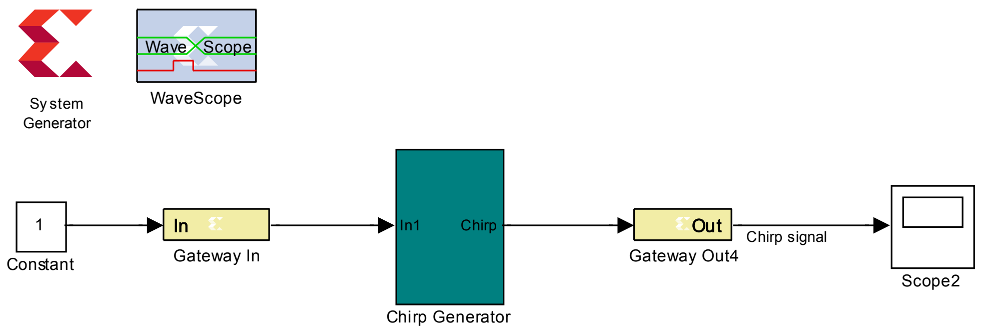


Figure 7.14: The complete FPGA design of the chirp generator

Before sending the chirp signal data stream to the DAC, the data should first be converted to another form in order to fit the DAC, and to achieve a suitable conversion the following points should be taken into account:

- Chirp generator output is 34 bit fixed point number with 30 bit after the binary point.
- Data codes to the DAC are 16 bit signed integer.

To scale the chirp generator data to fit the DAC, the data stream is converted to 16 bit fixed-point number with 14 bit after binary point. To get the 16 bit singed integer out of the 16 bit fixed point number, a multiplication by 2^{14} should be executed. A faster implementation can be done by simply reinterpreting the data as 16 bit singed integer which matches the DAC data type. This conversion is shown in Figure 7.15.

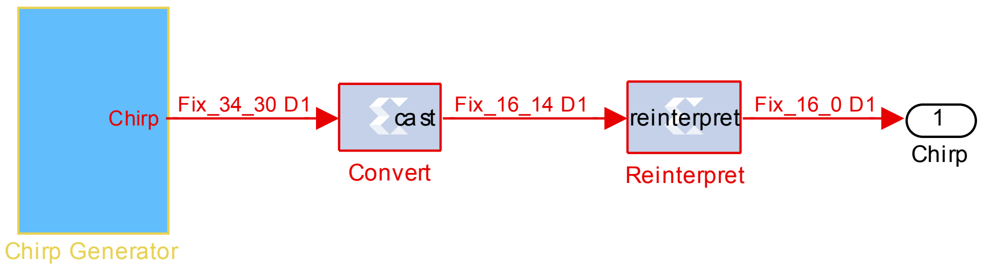


Figure 7.15: The block diagram of converting the data in order to fit the DAC

Finally, the block of the chirp generator is integrated within the card logic; this can be done by adding it right before the data samples enter the DAC interface component. The other logic is used to configure the DAC and the other on-board components. The whole implementation of the proposed chirp generator based on the FPGA technology and based on the Xilinx libraries is shown in Figure 7.17.

As a result of the implementation we have successfully generated the chirp signal based on the piecewise parabolic interpolation using the technology of the FPGA. Figure 7.16

presents the generated sine chirp signal, with clock frequency equal to 250 MHz, CO_0 = 100 Hz, AC_0 = 0, and the bit width of the counter and accumulator is 32 bit.

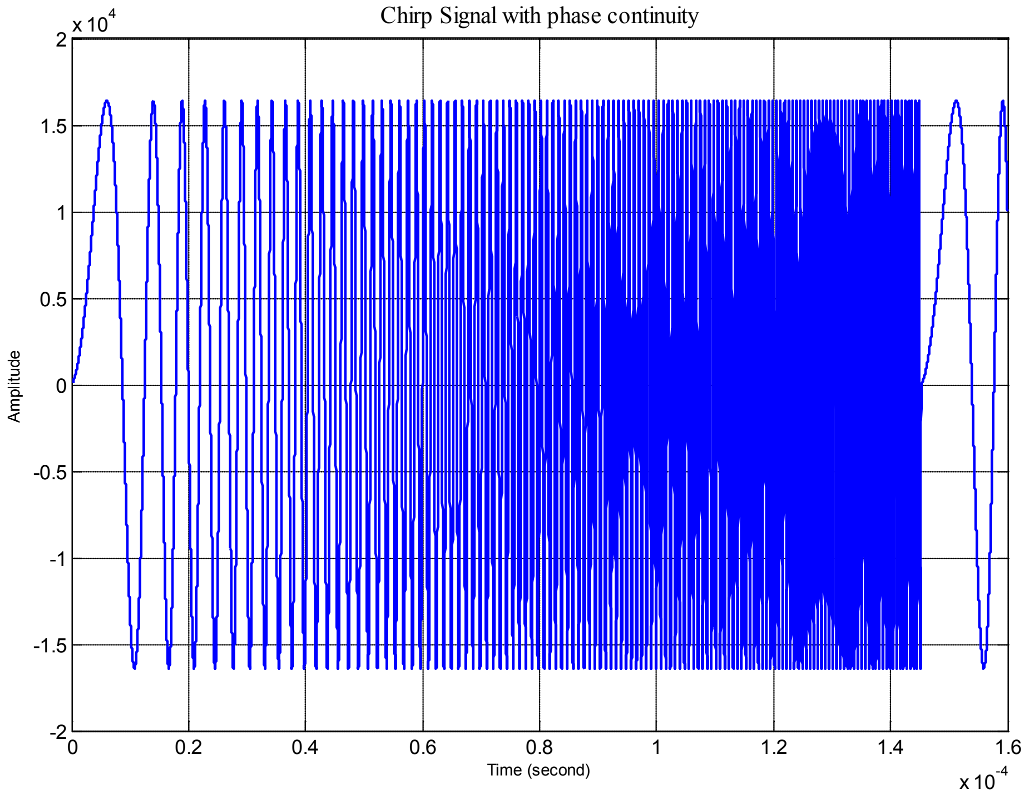


Figure 7.16: The generated chirp signal based on the proposed method and using the FPGA technology

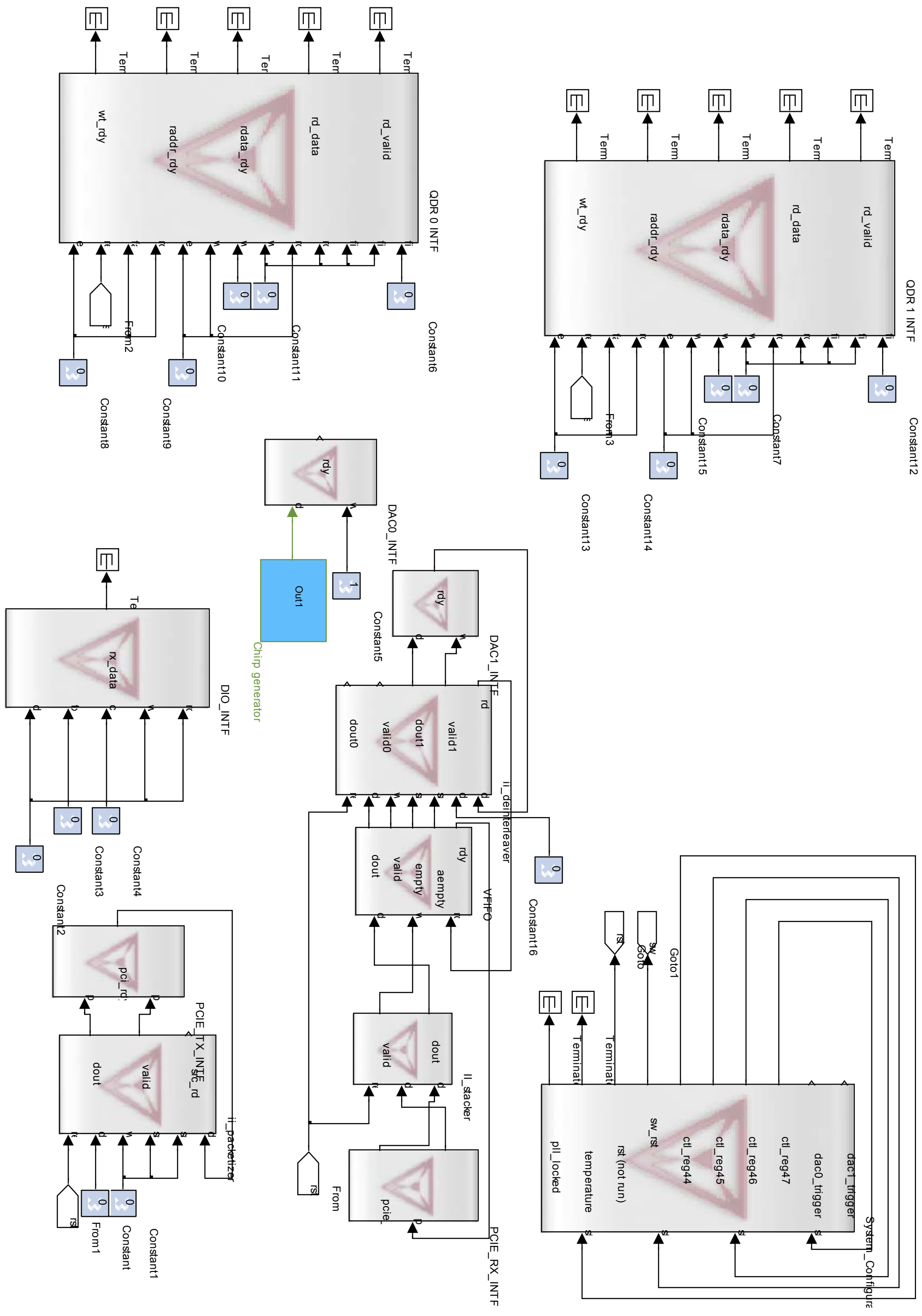


Figure 7.17: The complete implementation of the proposed chirp generator using the libraries of Xilinx

7.4 A parallel processing technique and overcoming the limited speed's problem of the FPGA

The parallel processing (multi-processing) technique is used to increase the sampling rate by replicating hardware so that multi inputs can be processed in parallel and multi outputs can be produced at the same time. Note that, multiple outputs are computed in parallel in a clock period and the effective sampling speed is increased by the level of parallelism.

7.4.1 The complete derivation and implementation for the parallel processing technique with level of four threads

When T_{sys_clk} is the system clock, then n= t/ T_{sys_clk} ; n=0, 1, 2,…, and assuming that the system is capable of generating 1 sample per clock , thus Ts = T_{sys_clk} ; Ts sampling period , Fs = 1/ Ts.

Our X5-TX card has a system clock of T_{sys_clk} =1/250 MHz ➔ Fs = 250 MHz, and a 1GSPS DAC.

To be able to feed the DAC with 1GSPS using 250 MHz system clock, the system should be able to generate 4 samples per clock. That means the system throughput should be 4 times more than the system frequency. Now, to satisfy this criterion, either the system clock should be increased to 1000 MHz, which is not possible due to hardware restrictions, or a parallel processing structure (technique) should be applied, using 4 threads to calculate 4 samples per system clock.

One issue facing the usage of parallel structure is, that at a specific time step n, there is only one data available for processing, not 4. This data is the system clock which forms the input to the counter/accumulator.

To solve this problem a prediction scheme is used, that will calculate the input of each thread at time step n.

To simplify the solution, the design process will be split into two steps:

- Design of a prediction scheme for the counter (1st Accumulator) (linear output)
- Design of a prediction scheme for the 2nd accumulator (quadratic output)

First step: Design of a prediction scheme for the counter (1st Accumulator) (linear output)

Let *f(n)* be the function of the counter output:

$$f(n) = n \times FCW \tag{7.2}$$

When $f(0{:}n)=\{0,\ 0+FCW,\ 2\times FCW,\ \ldots,\ n\times FCW\}$

$$= \{0,\ FCW,\ 2\times FCW,\ \ldots,\ n\times FCW\}$$

where:

n is the time step of the counter (1st accumulator); n=0, 1, 2, …

FCW: frequency control word (step size).

Therefore, the output function of the counter with time step (n+1) is given by:

$$\begin{aligned} f(n+1) &= (n+1)\times FCW \\ &= n\times FCW + FCW \end{aligned} \tag{7.3}$$

It is required for our implementation to define the four functions *f0, f1, f2, f3* that fulfill (7.5). These four functions will be defined according to two cases of *FCW,* the first when FCW equal one and the second when FCW is bigger than one.

In the case of *FCW=1,* those functions are given by:

$$f_p(n) = \begin{Bmatrix} (0,\ 1,\ 2,\ 3), \\ (4,\ 5,\ 6,\ 7), \\ (8,\ 9, 10,\ 11), \\ . \\ . \\ . \end{Bmatrix} \tag{7.4}$$

When FCW is bigger than one, considering the number of threads (q) is equal to 4, then the formula of the case $q = 4$ is given by:

$$f_p(n) = \begin{Bmatrix} (0, 0 + FCW, 0 + 2\times FCW, 0 + 3\times FCW), \\ (4\times FCW, 4\times FCW + FCW, 4\times FCW + 2\times FCW, 4\times FCW + 3\times FCW) \\ . \\ . \\ . \\ (f0, f1, 2, f3) \end{Bmatrix} \tag{7.5}$$

Note that

$f0 = q\times n\times FCW$	$= (q\times n)\times FCW$
$f1 = q\times n\times FCW + FCW$	$= (q\times n+1)\times FCW$
$f2 = q\times n\times FCW + 2\times FCW$	$= (q\times n+2)\times FCW$
$f3 = q\times n\times FCW + 3\times FCW$	$= (q\times n+3)\times FCW$

So, the required 4 functions of the counter/accumulator are given by:

$$(f0, f1, f2, f3) = \begin{pmatrix} q\times n\times FCW, q\times n\times FCW + FCW, \\ q\times n\times FCW + 2\times FCW, q\times n\times FCW + 3\times FCW \end{pmatrix} \tag{7.6}$$

Figure 7.18 shows a simplified block diagram of the parallel processing structure with a level of four threads and the throughput of the counter at each thread.

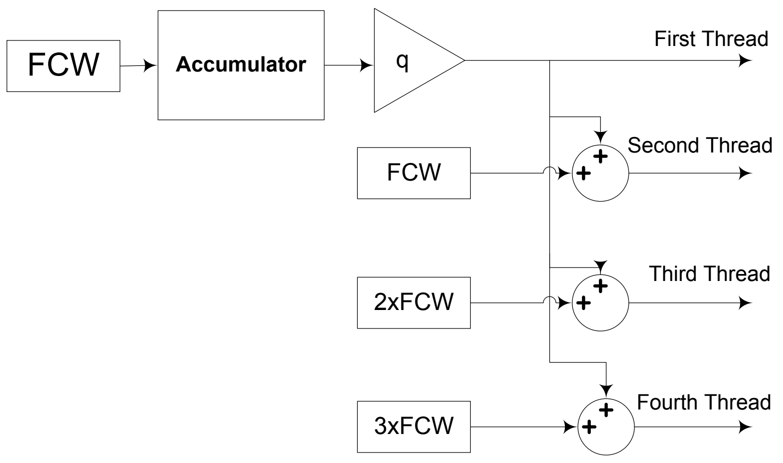


Figure 7.18: Simplified block diagram for the throughput at each thread

For simplicity, *k* can be used as time step instead of *n* in thread functions as (*k0, k1 , k2, k3*): *k0 = q×n, k1 = q×n+1, k2 = q×n+2,* and *k3 = q×n+3.* Thus, the time step for a specific thread is given as follows:

$$k_i = q \times n + i \qquad i = 0,1,\ldots,q-1 \tag{7.7}$$

where:

q: number of threads

n: time step relative to clock time

i: thread number, *i*=0, 1, .., *q*-1

Table 6: Time step results for the 1st and 2nd accumulator

n	*k*	*k0, k1, k2, k3*
0	0	0, 1, 2□ 3
1	4	4, 5, 6, 7
…		…
n	*q×n*	*q×n, q×n+1, q×n+2, …, q×n+q-1*

Writing $f1 \ldots fq$ in terms of k gives:

$$\begin{aligned} f\,0 &= k\,0 \times FCW \\ f\,1 &= k\,1 \times FCW \\ f\,2 &= k\,2 \times FCW \\ f\,3 &= k\,3 \times FCW \end{aligned} \qquad (7.8)$$

Figure 7.19 shows the generated single tune (mono frequency) signal based on the parallel processing technique with a level of four threads. To clearly show the parallel processing technique, Figure 7.19 has focused on the first two quadrants of the generated sine signal in order to prove the derived equation in the previous sections, for example if we take the blue pulse we will notice that it occupies only 1 fourth of the system clock period [1/200MHz], on the other hand the other pulses do the same repetition in order to form the whole sine wave. This figure shows simply how each thread completes the other in order to form the whole sine signal.

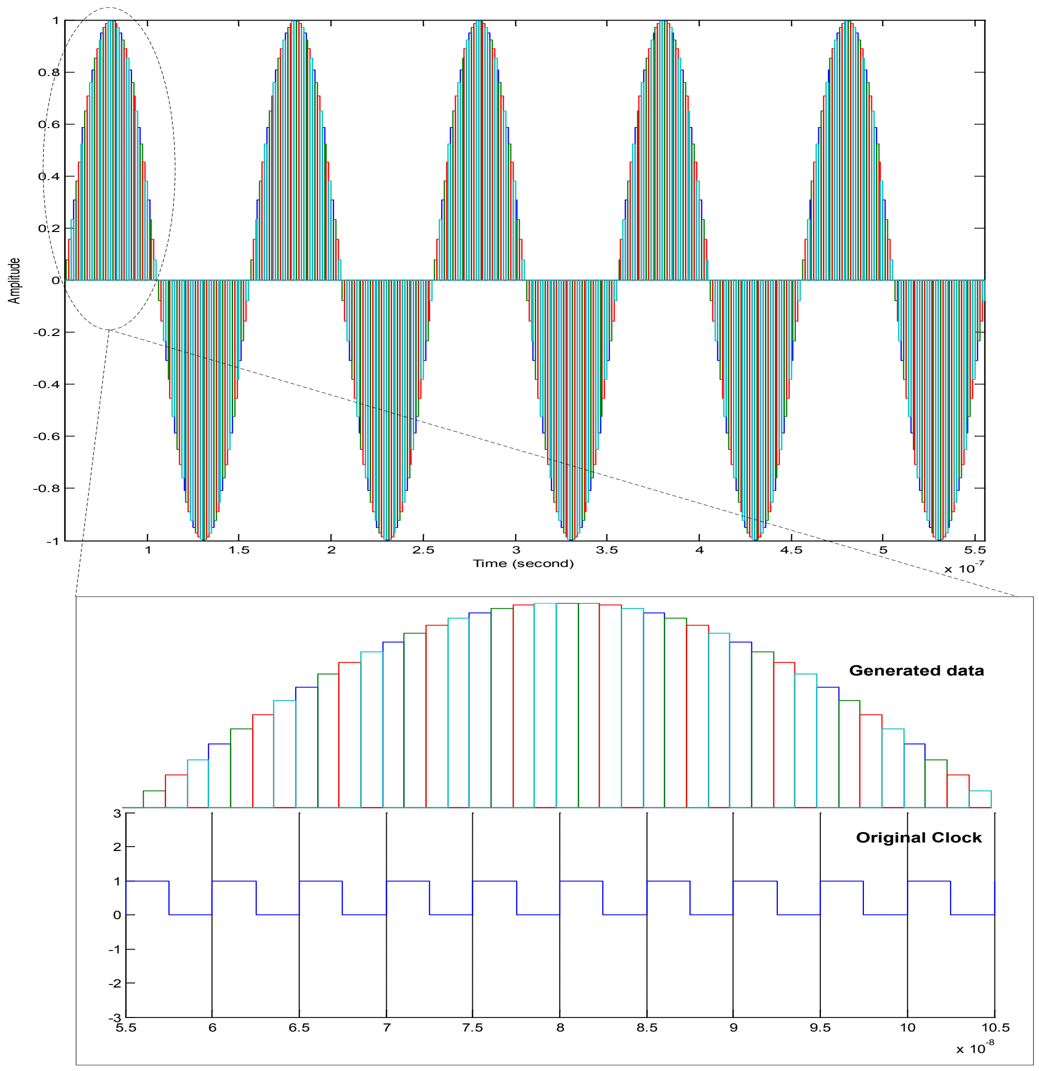


Figure 7.19: The generated single tune signals using 4 threads parallel processing technique

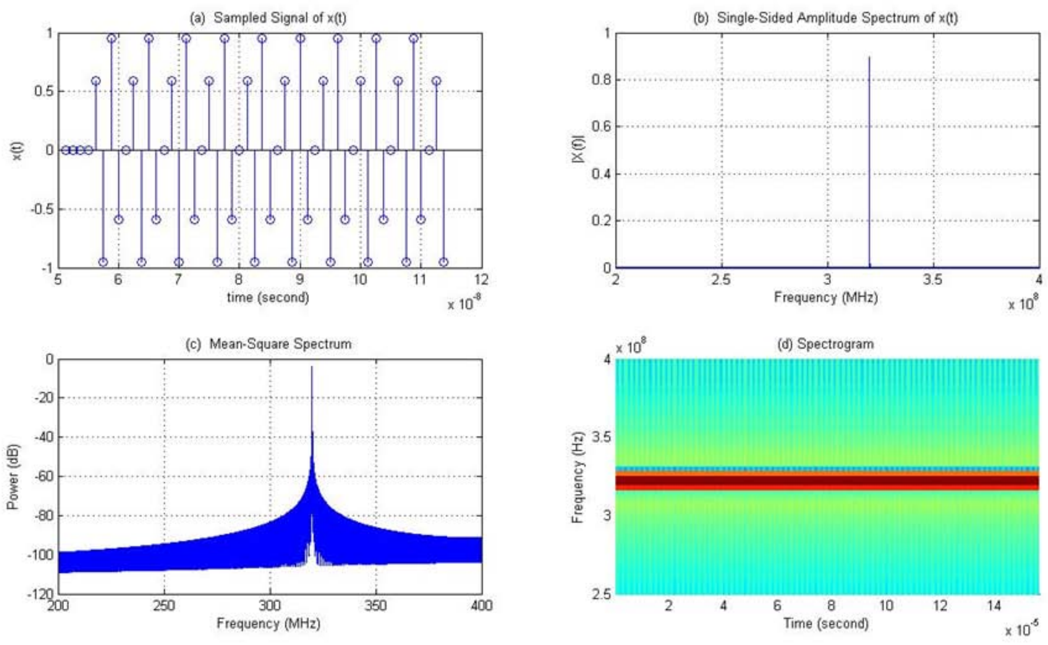


Figure 7.20: The generated single tune signal with 320 MHz. (a) Sampled Signal (b) Single-Sided Amplitude Spectrum (c) Mean Square Spectrum (d) The spectrogram

Second step: *d*esign of a prediction scheme for the 2nd accumulator (quadratic output)

Using the output of the previous step to feed *q* accumulators in *q* threads will lead to erroneous results, because each accumulator should accumulate all counter values, not only the one of the respective thread, for example in case of 4 thread system, the 1st thread will accumulate the values: 0, 4 ,8, 12,…

To work around this problem, a correction scheme should be implemented to compensate the deficit from the required accumulator output because of the missed accumulation steps. In the generation of the chirp signal, the constant value will be the sweep rate word (SRW) instead of the frequency control word.

The Spurious-Free Dynamic Range of the 125 MHz generated single tune signal reaches 52 dB, this is clearly shown in the following figure.

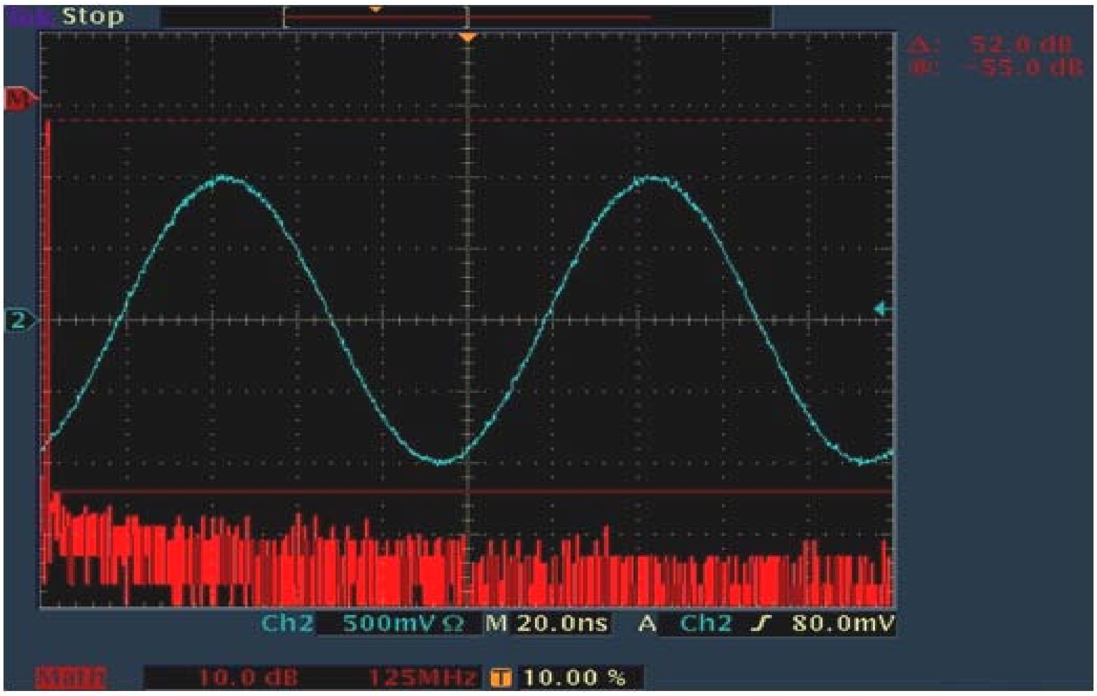

Stop
Δ: 52.0 dB
@: -55.0 dB
M
2
Ch2 500mV Ω M 20.0ns A Ch2 ⨍ 80.0mV
Math 10.0 dB 125MHz 10.00 %

Figure 7.21: Spurious Free Dynamic Range (SFDR) of the generated signal with 125 MHz.

Now, Let *g(n)* be the output function of the accumulator and it is given by:

$$\begin{aligned} g(n) &= \sum_{n=0}^{n-1} f(n) \\ &= \left(\sum_{n=0}^{(n-1)-1} f(n) \right) + f(n-1) \\ &= g(n-1) + f(n-1) \\ &= g(n-1) + (n-1) \times SRW \end{aligned} \tag{7.9}$$

Now, it is required to define the q functions $g_0, g_1, g_2, ... g_q$ that represent the output of the accumulator at time steps $k, (k+1), (k+2), (k+q-1)$ respectively, where:

$$g_i(k_i) = g_i(k_i - 1) + f(k_i - 1) \tag{7.10}$$

In case of multithreading, the value of the function g_i at time step $(k_i - 1)$, which should represent the previous value, is not available for the thread, because in a single thread the calculations are done at steps equal q, therefore the previous value in a thread i is $g_i(k_i - 4)$.

Therefore (7.10) should be written using $g_i(k_i - 4)$ terms only, which should be as in the following:

$$\begin{aligned} g(k-1) &= g(k-2+f)(k-2 \quad) \\ &= g(k-3)+f(k-3)+f(k-2 \quad) \\ &= g(k-4+f)(k-4+f)(k-3)+f(k-2 \end{aligned} \tag{7.11}$$

The Substitution of (7.11) in (7.10) gives:

$$g_i(k_i) = g_i(k_i-4+f)(k_i-4+f)(k_i-3)+f(k_i-2+f)(k_i-1) \tag{7.12}$$

The replacing of the value of $f(k)$ in (7.12) gives:

$$\begin{aligned} g_i(k_i) &= g_i(k_i-4)+(k_i-4)\times SRW+(k_i-3)\times SRW \\ &\quad +(k_i-2)\times SRW+(k_i-1)\times SRW \\ g_i(k_i) &= g_i(k_i-4)+(4k_i-6)\times SRW \end{aligned} \tag{7.13}$$

We can also write (7.13) to be as follows:

$$\begin{aligned} g_i(k_i) &= g_i(k_i-4)+4(k_i+4-4)\times SRW-6\times SRW \\ &= g_i(k_i-4)+4(k_i-4)\times SRW+16\times SRW-6\times SRW \\ &= g_i(k_i-4)+4(k_i-4)\times SRW+10\times SRW \end{aligned} \tag{7.14}$$

The complete implementation of the previous mathematical analysis is shown in Figure 7.22

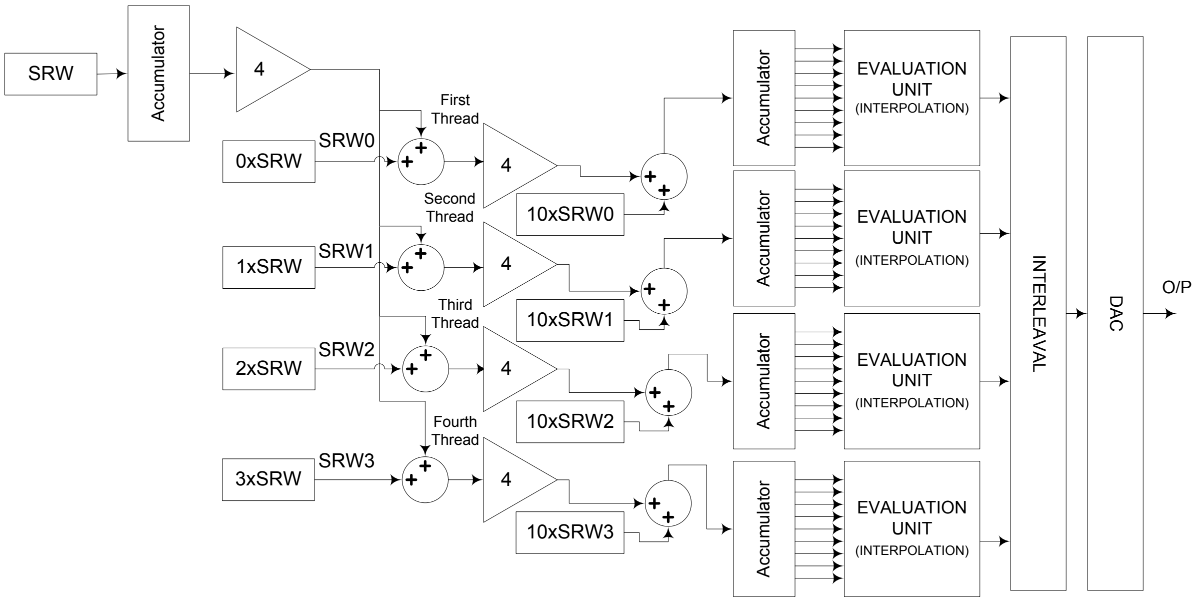


Figure 7.22: The generation of the digital chirp signal using 4 threads parallel processing structure and the interpolation methodology

A chirp signal is successfully generated by using the methodology of piecewise parabolic interpolation and a parallel processing technique. This chirp signal is illustrated in Figure 7.23. Figure 7.24 shows the spectrum of the chirp signal and the linearly swept-frequency signal from 0 up to 320 MHz. The proposed system (in the hardware implementations) has the ability to increase the bandwidth by increasing the level of the parallel processing (number of threads), if the hardware features (e.g. clock of the DAC) permit.

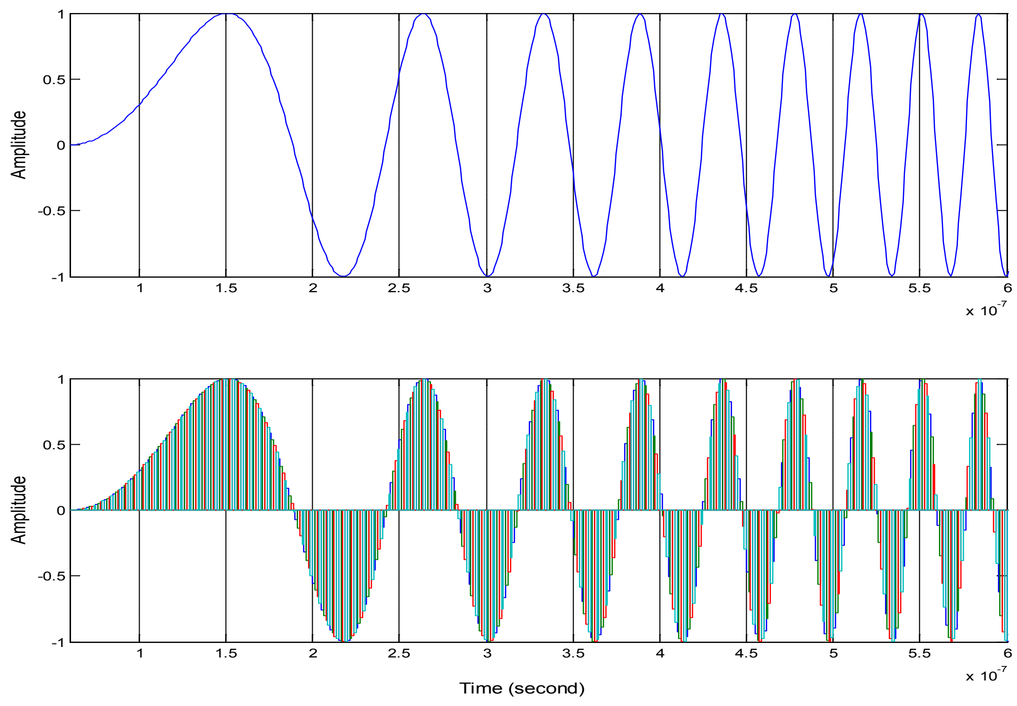


Figure 7.23: The generated chirp signal using 4 threads parallel processing technique and the interpolation methodology.

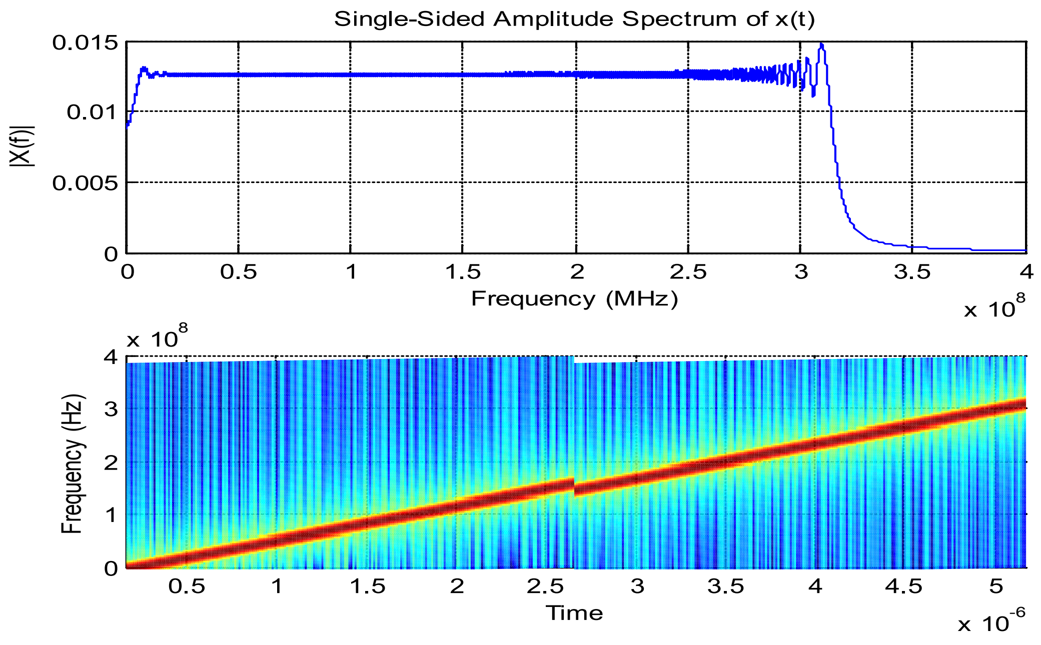


Figure 7.24:A 320 MHz digital chirp signal using the proposed method: single sided amplitude spectrum, (b)The energy spectrogram

8 Conclusions and Contributions

8.1 Conclusions

In this dissertation, a new digital chirp signals generator is proposed based on the methodology of the piecewise polynomial approximation and using FPGA technology. We have realized our new architecture via X5-TX innovation Xilinx card to generate a sine chirp signal.

The proposed system is a hybrid of the digital sweep generator and the system using interpolation based direct digital frequency synthesis. The interpolator uses predetermined interpolation coefficients to fit the sine wave from the calculated phase instead of using a predetermined waveform stored in a big sized memory.

The proposed digital chirp generator showed an extremely low level of the spurious harmonic distortion and at the same time reduced both the hardware complexity and memory size of the LUT. The new size of the ROM is reduced by a factor of more than 128 when using 12 address lines, and Spurious Free Dynamic Range (SFDR) reaches 100.9 dBc. This chirp signal generator is comparable with other methods that implement the look-up table method.

The system is realized using the FPGA technology, The Innovation integration X5-TX platform with FPGA Xilinx VIRTEX-5 was used to generate a chirp signal with high bandwidth reaches 320MHz.

8.2 Contributions

During the course of PhD research, the other has significantly contributed to the scientific knowledge in the area of waveform generation and signal processing. Some of the achievements made through this research are listed below:

1. A novel method, for generating chirp signals based on piecewise- polynomial interpolation for radar application with reduction in the hardware complexity and the spurious harmonic distortion, is proposed.
2. The novel approach and the proposed theoretical aspects are realized using the technology of FPGA. This contribution makes this work really important, because it will be integrated within the HITCHHIKER system, which gives ZESS the ability to operate its own SAR sensor.
3. A unique parallel processing technique with 4 threads is derived and implemented by the FPGA technology in order to generate high bandwidth digital chirp signals for radar applications.
4. A reduction in the size of the memory (with respect to the previous methods in the literature), less hardware complexity, and high SFDR are important parameters have been achieved to make our system (architecture) unique.
5. Extension for the architecture using an optimization method in order to achieve further enhancement in the error metrics like SFDR and Total harmonic distortion.

The contributions to the scientific work have produced the following publications:

1. Samarah, A. Albashar, A. and Loffeld, O.: ‘A Unique Parallel Processing Technique for Generating Digital Chirp Signals for the Applications of Synthetic Aperture Radar’, *IEEE Proc.EMS2011, Madrid, November 16-18, 2011, 185-190*

2. Samarah, A. and Loffeld, O.: An Improved Digital Chirp Generator Using Optimized Piecewise Parabolic Interpolation. *In the proceeding of Conf EUSAR2010, Aachen-Germany, 7-10 June, 2010, 253-256.*

3. Samarah, A. and Loffeld, O.: High-Performance Coherent Digital Sweep Oscillator Using Piecewise Parabolic Interpolation. *IEEE Proc. ECCDT, August 2009, 461-465.*

4. Samarah, A. Al-Ibrahim, M. and Loffeld, O.: An Efficient Method for Generating Coherent Digital Sweep Signals. *IEEE proc. CICSYN, India, July 2009, 85-97.*

5. Samarah, A. Loffeld, O. Shahab, W. and Al-Ibrahim, M.: A Digital Sweep (Chirp) Generator with Extremely Small Memory Size and High Level of the Spurious Free Dynamic Range, *International Journal of Simulation, Systems, Science and Technology (IJSSST), UK. Vol. 11, No.1, 9-15, January 2010.*

6. Wang, R. et al. Samarah, A.: A Bistatic Point Target Reference Spectrum for General Bistatic SAR Processing, *IEEE Geosciences and Remote Sensing Letters, Vol.5, NO.3, July 2008, 517-521.*

7. Wang, R. et al. Samarah, A.: Analysis and Processing of spaceborne/Airborne Bistatic SAR data’, in Proc. *IGRASS, Boston, USA 3(1), pp. 597-600, July 2008.*

8. Samarah, A. Albashar, A. and Loffeld, O.: ‘A Realization for a Low Cost Digital Chirp Generator for SAR Application Using FPGA Technology’. Under preparation

Appendix

The detailed derivation of (6.17)-(6.25) is achieved by substituting (6.13) in (6.16) and based on [31], we have

$$b_n = \sum_{k=1}^{s} J'_{k,n} + J''_{k,n} + J'''_{k,n} + J''''_{k,n} \tag{1}$$

Where

$$J'_{k,n} = \int_{x_k}^{x_{k+1}} 2y_k \sin\left(\frac{n\pi x}{2}\right) dx \tag{2}$$

$$J''_{k,n} = \int_{x_k}^{x_{k+1}} 2m_k (x - x_k) \sin\left(\frac{n\pi x}{2}\right) dx \tag{3}$$

$$J'''_{k,n} = \int_{x_k}^{x_{k+1}} 2p_k (x - x_k)^2 \sin\left(\frac{n\pi x}{2}\right) dx \tag{4}$$

$$J''''_{k,n} = \int_{x_k}^{x_{k+1}} 2q_k (x - x_k)^3 \sin\left(\frac{n\pi x}{2}\right) dx \tag{5}$$

By solving integrals recalling that $x_k = (k-1)/s$ one obtains

$$J'_{k,n} = \frac{4y_k}{n\pi}\left[\cos\left(\frac{(k-1)n\pi}{2s}\right) - \cos\left(\frac{kn\pi}{2s}\right)\right] \tag{6}$$

$$J''_{k,n} = \frac{-4m_k}{n^2\pi^2 s}\begin{bmatrix} n\pi\cos\left(\frac{kn\pi}{2s}\right) \\ +2s\left(\sin\left(\frac{(k-1)n\pi}{2s}\right) - \sin\left(\frac{kn\pi}{2s}\right)\right)\end{bmatrix} \tag{7}$$

$$J'''_{k,n} = \frac{-4p_k}{n^3\pi^3 s^2}\begin{bmatrix} 8s^2\cos\left(\frac{(k-1)n\pi}{2s}\right) \\ +(n^2\pi^2 - 8s^2)\cos\left(\frac{kn\pi}{2s}\right) - 4n\pi s\sin\left(\frac{kn\pi}{2s}\right)\end{bmatrix} \tag{8}$$

$$J''''_{k,n} = \frac{-4q_k}{n^4\pi^4 s^3}\begin{bmatrix} n\pi(n^2\pi^2 - 24s^2)\cos\left(\frac{kn\pi}{2s}\right) \\ +6s\left((8s^2 - n^2\pi^2)\sin\left(\frac{kn\pi}{2s}\right) - 8s^2\sin\left(\frac{(k-1)n\pi}{2s}\right)\right)\end{bmatrix} \quad (9)$$

From (1) and (6)-(9), collecting terms in $\cos\left(\frac{kn\pi}{2s}\right)$; $\cos\left(\frac{(k-1)n\pi}{2s}\right)$; $\sin\left(\frac{kn\pi}{2s}\right)$ *and* $\sin\left(\frac{(k-1)n\pi}{2s}\right)$ one obtain

$$b_n = \sum_{k=1}^{s}\begin{bmatrix} A_{k,n}\cos\left(\frac{kn\pi}{2s}\right) + B_{k,n}\cos\left(\frac{(k-1)n\pi}{2s}\right) \\ C_{k,n}\sin\left(\frac{kn\pi}{2s}\right) + D_{k,n}\sin\left(\frac{(k-1)n\pi}{2s}\right)\end{bmatrix} \quad (10)$$

Where

$$A_{k,n} = \frac{-4}{n^3\pi^3 s^3}\begin{bmatrix} n^2\pi^2 s^2 m_k + (n^2\pi^2 s - 8s^3)p_k \\ +n^2\pi^2 q_k - 24s^2 q_k + n^2\pi^2 s^3 y_k\end{bmatrix} \quad (11)$$

$$B_{k,n} = \frac{4}{n^3\pi^3}\left[n^2\pi^2 y_k - 8p_k\right] \quad (12)$$

$$C_{k,n} = \frac{8}{n^4\pi^4 s^2}\begin{bmatrix} n^2\pi^2 s(sm_k + 2p_k) \\ +3(n^2\pi^2 - 8s^2)q_k\end{bmatrix} \quad (13)$$

$$B_{k,n} = \frac{8}{n^4\pi^4}\left[24q_k - n^2\pi^2 m_k\right] \quad (14)$$

Since n is odd, then the term $\cos\left(\frac{n\pi}{2}\right)$ will vanish, therefore (10) can be written as follows:

$$b_n = B_{1,n} + \sum_{k=1}^{s-1}\begin{bmatrix} (A_{k,n} + B_{k+1,n})\cos\left(\frac{kn\pi}{2s}\right) \\ +(C_{k,n} + +D_{k+1,n})\sin\left(\frac{kn\pi}{2s}\right) + C_{s,n}\sin\left(\frac{n\pi}{2}\right)\end{bmatrix} \quad (15)$$

By substituting (11)-(14) in (15), we get

$$
\begin{aligned}
b_n &= \frac{4}{n\pi} y_1 - \frac{32}{n^3\pi^3} p_1 \\
&+ \frac{4}{n\pi} \sum_{k=1}^{s-1} \left[-y_k + y_{k+1} - \frac{m_k}{s} - \frac{p_k}{s^2} - \frac{q_k}{s^3} \right] \cos\left(\frac{kn\pi}{2s} \right) \\
&+ \frac{32}{n^3\pi^3} \sum_{k=1}^{s-1} \left[-p_{k+1} + p_k + 3\frac{q_k}{s} \right] \cos\left(\frac{kn\pi}{2s} \right) \\
&+ \frac{8}{n^2\pi^2} \sum_{k=1}^{s-1} \left[m_k - m_{k+1} + 2\frac{p_k}{s} + 3\frac{q_k}{s^2} \right] \sin\left(\frac{kn\pi}{2s} \right) \\
&+ \frac{192}{n^4\pi^4} \sum_{k=1}^{s-1} [q_{k+1} - q_k] \sin\left(\frac{kn\pi}{2s} \right) \\
&+ \frac{8}{n^2\pi^2} \sum_{k=1}^{s-1} \left[(m_s + 2\frac{p_s}{s}) + 3\frac{q_s}{s^2} \right] \sin\left(\frac{n\pi}{2} \right) \\
&- \frac{192}{n^4\pi^4} q_s \sin\left(\frac{n\pi}{2} \right)
\end{aligned}
\qquad (16)
$$

We achieved the formulae (6.17)-(6.25) after rearranging (16) .

Bibliography

[1] J. E. Wilhjelm and P. C. Pedersen, "Target velocity estimation with FM and PW echo ranging Doppler systems-Part I: system analysis," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control,* vol. 40, pp. 373-380, 1993.

[2] P. C. Pedersen, "Digital sweep generator," in *U.S. Patent Application*, 1988.

[3] R. G. Plumb, and Ma, H, "Swept frequency reflectometer design for in-situ permittivity measurement," *IEEE Trans. Instrumentation. Measurement,* vol. 42, pp. 730-734, 1993.

[4] J. Tierney, C. M. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Trans. Audio Electroacoust,* vol. AU-19, pp. 49-57, Mar. 1971.

[5] P. C. Pedersen, "Digital generation of coherent sweep signals," *IEEE Trans. Instrumentation and Measurement,* vol. 39, pp. 90-95, 1990.

[6] M. Soumekh, *Synthetic Aperture Radar Signal Processing with Matlab Algorithms*. United States: John Wiley & Sons, Inc., 1999.

[7] S. W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing," San Diego, CA: California Technical Publishing, 1997.

[8] N. T. GmbH, "http://www.nanotron.com/EN/support/FAQ/FAQ-01.htm#pgfId-1043694," Berlin, Germany, 2011.

[9] C. E. Cook and M. Bernfield, *Radar Signals*. New York: Academic, 1967.

[10] G. K. Lewis, "Chirped PVDF tranducers for medical ultrasound imaging," in *Ultrasonics Symp*, 1987, pp. 879-884.

[11] S. M. Robinowitz, C. H. Gager, C. E. Muehe, and C. M. Johanson, "Applications of digital technology to radar," in *Proc. IEEE*, 1985, pp. 325-339.

[12] T. Yamamoto, S. Fujii, and Y. Aoki, "Holographic imaging system using wideband chirped ultrasound," *Acoustical Imaging,* vol. 13, pp. 435-445, 1984.

[13] A. Hiasat and A. Al-Khateeb, "Efficient digital sweep oscillator with extremely low sweep rates," *Proc. IEE Circuits, Devices, and Systems,* vol. 145, pp. 409-414, 1998.

[14] Y.C.Jeng, "Digital spectra of nonuniformly sampled signals: Digital look-up tunable sinusoidal oscillator," *IEEE Trans. Instrumentation and Measurement,* vol. 37, pp. 358-362, 1988.

[15] Y. C. Jeng, "Digital spectra of nonunifor,ly sampled signals: fundamentals and high-speed waveform digitizers," *IEEE Trans. Instrum. Measur.,* vol. 37, pp. 241-251, 1988.

[16] D. Garcia, "Precision digital sine generation with TMS32010," in *digital signal processing applications with the TMS320 family* Dallas: TX(Texas Instruments, 1986) 1986, pp. 269-289.

[17] M. Schanerberger and S. Awad, "The implementation of digital sine wave oscillator using the TMS320C25: Distortion reduction and applications," *IEEE Trans. Instrumentaion and Measurement,* vol. 39, pp. 870-873, 1990.

[18] B. G. Goldberg, "Direct Digital Frequency Synthesis Demystified," Eagle Rock, VA: LLH Technol.Pub., 1999.

[19] A. Torosyan, "System For Analysis And Design Of Direct Digital Frequency Synthesizers." vol. 7532989, 2004.

[20] H. T. Nicholas and H. Samueli, "An analysis of the output spectrum of direct digital frequency synthesizers in the presence phase-accumulator truncation," in *Proceedings of the 41st Annual Frequency Control Symposium*, 1987, pp. 495-502.

[21] D. A. Sunderland, R. A. Strauch, S. S. Wharfield, H. T. Paterson, and C. R. Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications," *IEEE Journal of Solid-State Circuits,* vol. 19, pp. 497-505, 1984.

[22] A. I. Abu-El-Haija and M. M. Al-Ibrahim, "Digital oscillators having low sensitivity and round off errors," *IEEE Transactions on Aerospace and Electronic Systems,* vol. AES-22, pp. 23-32, 1986.

[23] A. Madisetti, A. Kwentus, and A. N. Willson, "A 100-MHz 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range," *IEEE Journal of Solid-State Circuits,* vol. 34, pp. 1034-1043, 1999.

[24] M. S. Bellaouar, A. M. O'brecht, and M. I. E. Fahim, "Low-power direct digital frequency synthesis for wireless communications," *IEEE Journal of Solid-State Circuits,* vol. 35, pp. 385-390, 2000.

[25] H. Nicholas and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25-micron CMOS with -90-dB spurious performance," *IEEE Journal of Solid-State Circuits,* vol. 26, pp. 1959-1969, Dec 1991.

[26] P. C. Pedersen, P. A. Lewin, and L. B. Jorno, "Application of time-delay spectrometry for calibration of ultrasonic transducer," *IEE Transactions on Ultrasonic, Ferroelectrics, and frequency control,* vol. 35, pp. 185-205, 1988.

[27] J. M. Langlois and D. Al-Khalili, "ROM Size Reduction with Low Processing Cost for Direct Digital Frequency Synthesis," in *PACRIM 2001*, Victoria, Canada, 2001, pp. 287-290.

[28] A. Samarah, M. Al-Ibrahim, and O. Loffeld, "An Efficient Method for Generating Coherent Digital Sweep Signals," in *CICSYN2009*, India, 2009, pp. 85-97.

[29] A. Samarah and O. Loffeld, "High-Performance Coherent Digital sweep Oscillator Using Piecewise Parabolic Interpolation," in *Proc. IEEE European Conference on Circuit Theory and Design* Turkey, 2009, pp. 461-465.

[30] A. Samarah and O. Loffeld, "An Improved Digital Chirp Generator Using Optimized Piecewise Parabolic Interpolation," in *EUSAR2010*, Aachen, Germany, 2010, pp. 253-256.

[31] D. De Caro and A. Strollo, "High-Performance Direct Digital Frequency Synthesizers Using Piecewise-Polynomial Approximation," *IEEE Trans. Circuits and Systems,* vol. 52, pp. 324-337, Feb. 2005.

[32] J. M. P. Langlois and D. Al-Khalili, "Phase to sinusoid amplitude conversion techniques for direct digital frequency synthesis," in *IEE Proc.-Circuits, Devices Syst*, 2004, pp. 519-528.

[33] A. Eltawil and D. Babak, "Interpolation based direct digital frequency synthesis for wireless communications," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2002, pp. 73-76.

[34] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circuits ans Systems*, Espoo, Finland, 1988, pp. 2641-2645.

[35] L. Erup, F. Gardner, and R. Harris, "Interpolation in Digital Modems - Part II: Implementation and Performance," *IEEE Transaction on Communications,* vol. 41, pp. 998-1008, June 1993.

[36] J. P. Langlois and D. Al-Khalili, "Novel Approach to the Design of Direct Digital Frequency Synthesizers Based on Linear Interpolation," *IEEE Trans. Circuits and Systems-II: ANALOG AND DIGITAL SIGNAL PROCESSING,* vol. 50, pp. 567-578, 2003.

[37] J. Garcia, S. G. LaJeunesse, and D. Bartow, "Measuring Spurious Free Dynamic Range in a D/A Converters," in *TB326* FL: Intersil, 1995.

[38] H. S. L.K. Tan, "200 MHz quadrature digital synthesizer /mixer in 0.8um CMOS," *IEEE Journal of Solid-State Circuits,* vol. 30, pp. 193-200, March 1995.

[39] Wikipedia, "Horner scheme," Wikipedia Contributors, 2011.
[40] I. Integration, "X5-TX User's Manual," California, 2009.
[41] I. Integration, "Product Guide: DSP, Data Acquisition, and Embedded Control." vol. 21.
[42] I. Integration, "X5-TX Frame Work Logic User Guide," California, 2009.
[43] T. Instrument, "16-Bit, 1.0 GSPS 2x-4x Interpolating Dual-Channel DAC," in *DAC5682Z*, 2011.
[44] Xilinx, "Virtex-5 Family Overview_Product Specification," 2009.
[45] Xilinx, "Virtex-5 FPGA User Guide." vol. V5.3, 2010.
[46] Xilinx, "System Generator for DSP_User Guide," in *UG640*, V 12.3 ed, 2010.
[47] I. Integration, "X5-MATLAB BSP MANUAL," Rev 1.0 ed California, 2010.