Circuits and Systems, 2013, 4, 252-263
http://dx.doi.org/10.4236/cs.2013.43034 Published Online July 2013 (http://www.scirp.org/journal/cs)

High Speed Digital Oscillator Implementations Based on Advanced Arithmetic and Architecture Techniques

Ali Shatnawi, Mufleh Shatnawi
Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan
Email: ali@just.edu.jo, shatnawi.mufleh@gmail.com

Received September 24, 2012; revised December 1, 2012; accepted December 9, 2012

ABSTRACT

The advances of digital arithmetic techniques permit computer designers to implement high speed application specific chips. The currently produced digital circuits have demonstrated high performance in terms of several criteria, such as, high clock rate, short input/output delay, small silicon area, and low power dissipation. In this paper, we implement several sinusoidal generation methods to optimize their performance and output using advanced digital arithmetic techniques. In this paper, the implementations of advanced digital oscillator structures with and without pipelining are proposed. The synthesis results of the implementation with pipelining have proven that it is superior to other sinusoidal generation methods in terms of the maximum frequency and signal resolution. Hence, this method is used in the design of the proposed digital oscillator chip.

Keywords: Digital Oscillator; Single-Output Digital Oscillator; Multiple-Output Digital Oscillator; Advanced Arithmetic Techniques; Hardware Implementation

1. Introduction

A sinusoidal oscillator can be designed using analog or digital components. The sinusoidal parameters such as the frequency, amplitude and phase, are easier to control in digital oscillators than in analog ones. On the other hand, the amount of harmonic distortion in digital oscillators is higher than that in analog ones. This is obviously attributed to the resulting quantization error and digital arithmetic round-offs.

Digital sinusoidal oscillators are used in many applications, including communications, music synthesis, control, radar, and several digital signal processing applications. As the design and fabrication of digital integrated circuits are getting very systematic and relatively simple, the choice of the digital approach for sinusoidal oscillators has become very desirable.

The development of high speed digital signal processing units is very important in the implementation of real-time applications. This paper uses advanced digital arithmetic techniques along with advanced hardware design techniques to produce high-speed, high-frequency and digital oscillator chips. The organization of the paper will be as follows. Section 2 presents the motivation for this work, Section 3 discusses some of the approaches used in the design of digital oscillators, Section 4 introduces the number representation used, Section 5 presents some digital oscillators found in the literature, Section 6 analyses the oscillators used, Section 7 gives the implementation and results, and Section 8 concludes the paper.

2. Motivation

In the fields of communications, digital signal processing, and digital image processing, which are known to be computationally intensive, the performance of an implementation is decided by the efficiency of the critical arithmetic operations. Like most DSP applications, the performance of a digital oscillator is highly dependent on the efficiency of the multiplication unit. In other words, the optimization of the multiplication algorithm is critical in the implementation of the digital sinusoidal oscillators; especially, when the design is based on recursive algorithms.

In this paper, we will work on two issues to implement an efficient digital oscillator. In the first, we will device a methodology to implement a very fast multiplier, a critical component of the digital oscillator. In the second, we propose pipelined implementation of the entire design to enable a fast clock rate.

3. Digital Oscillator Generation Methods

In literature, several methods to implement a digital oscillator had been proposed. Among the most known methods are the following.

3.1. Look-Up-Table (LUT)

The samples of a general sinusoidal signal are stored in a ROM-like memory, and re-generated at a rate that is necessary to produce the desired frequency [1,2].

3.2. Recursive Method

In this method, a second-order recursive digital filter is used to generate the sequence of the output samples of the required waveform [3-10]. One of these methods is based on the following difference equation.

$$y(n) = 2\cos\theta \cdot y(n-1) - y(n-2), n \geq 0 \tag{1}$$

where $y(n)$ is the output sample number n, and $2\cos(\theta)$ is a multiplier coefficient. The value of this coefficient plays an important role in determining the frequency of the generated sinusoidal signal. The Z-transform of (1) is given by:

$$Y(z) = \frac{\left(2\cos(\theta) - z^{-1}\right)y(-1) - y(-2)}{1 - 2\cos(\theta) \cdot z^{-1} + z^{-2}} \tag{2}$$

Let the initial condition $y(-1) = 0$, then

$$Y(z) = \frac{-y(-2)}{1 - 2\cos(\theta) \cdot z^{-1} + z^{-2}} \tag{3}$$

But

$$Z\left[\sin(n\theta)\right] = \frac{z^{-1} \cdot \sin(\theta)}{1 - 2\cos(\theta) \cdot z^{-1} + z^{-2}} \tag{4}$$

By comparing (3) with (4), the Z-inverse of (3) will be given by

$$y(n) = \frac{-y(-2)}{\sin(\theta)} \cdot \sin(n+1)\theta \tag{5}$$

Equation (5) shows that θ carries information about the frequency of the sinusoidal signal generated by the digital oscillator. The angle θ can be rewritten as:

$$\theta = \omega T_S = 2\pi \cdot f \cdot T_S \tag{6}$$

where T_S is the time interval between consecutive samples of the generated sinusoidal signal, and f the desired frequency. The value of T_S determines the maximum frequency of the digital oscillator chip, that is $\left(f_s = 1/T_s\right)$. The generated sinusoidal signal is represented in a discrete form with a number of samples per cycle that is given by

$$N \cong \frac{2\pi}{\theta} \tag{7}$$

Combining (6) and (7), the number of samples can also be expressed as

$$N = \frac{1}{f \cdot T_s} \tag{8}$$

The above equations show that the frequency of the generated sinusoidal signal has a linear relationship with the maximum frequency of the digital oscillator.

4. Fixed-Point Number Representation and Fraction Manipulation

Some of the commercially available digital signals processing units have no hardware support for floating-point arithmetic, usually called Floating Point Unit (FPU), due to the cost limitation. Instead, they use software emulation for the implementation of floating-point operations. This can significantly limit the rate at which iterative real-time applications can execute.

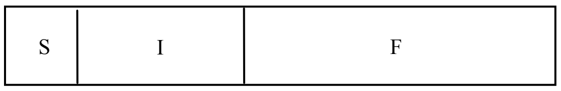
The cost of digital operations and the simplicity of the arithmetic design are highly affected by the choice of the numbering representation system. The fixed-point number representation is typically used when the hardware cost, speed, and hardware complexity are considered in the implementation of DSP processing units [3].

It should be noted that any digital arithmetic operation using fixed-point representation can be performed as integer digital arithmetic operation. This significantly improves the execution speed of the digital signal processing units and alleviates the software complexity due to software emulation used for implementing floating-point operations. On the other hand fixed-pint arithmetic suffers from the quantization error due to the finite-precision representation. However, in most digital signal processing application, such as digital oscillators and digital filters, the data and coefficients can be scaled to reduce the effect of the quantization error on the accuracy of the implemented systems [11].

In this study, we have chosen the sign-magnitude fixed point number representation for the implementation of the digital oscillator. It has been shown that the hardware implementation of sign-magnitude fixed point number representation is required less area compare to 1's complement or 2's complement [11]. This can be clearly seen that only one inverter is required to generate a negative number.

The format of the number representation, which is shown in **Figure 1**, is composed of a sign bit, *I* bits for the integer part, and *F* bits for the fraction part.

The number of fraction bits determines the upper and lower bounds of both the frequency range and the total



S: sign
I: number of integer bits
F: number of fraction bits

Figure 1. The sign-magnitude fixed-point number representation.

harmonic distortion of the implemented oscillator as will be shown later.

5. Digital Arithmetic Techniques and Related Work

The multiplication is a basic arithmetic operation in almost all digital signal processing applications. Digital signal processing systems require hardware multipliers to implement DSP algorithms efficiently. The speed of the multiplier directly impacts the speed of the digital processing units [12-14].

There are many fast multipliers in the literature that can be used in the design of an efficient digital oscillator [11,15-18]. Regardless of the multiplication method, the basic operation of a multiplication algorithm is the addition. In this section we consider the most common adder structures. To further optimize the addition operations of a multiplication, multi-operand addition techniques are used [19].

5.1. Carry Save Adder (3:2 Adder/3:2 Counter)

The Carry Save Adder (*CSA*) is classified as a redundant number system adder. The *CSA* Adder is used to add more than two numbers—binary vectors-together, say x, y, and z, and generate two binary numbers—vectors-$S+C$ such that $x+y+z=S+C$. This addition can be done in $O(1)$ time, since the carry bits are saved in the C vector and no carry propagation is required. The result of the multi-operand addition can be obtained by adding the finial binary vectors $(S+C)$ together using a conventional number system adder like the carry look-ahead adder.

The full adder (FA) can be thought of as a 1-bit *CSA*, as shown in **Figure 2** [14,19].

5.2. 4:2 Compressor

The "4-2 *carry-save module*" was proposed in [20]. It contains a combination of FA cells, which are interconnected to generate the output signal faster than the traditional *CSA* Adder, 3:2 Adder. The structure actually compresses five bits (input signals) into three bits (output signal); however, it is connected in such a way that four of the inputs are coming from the same bit position and one from the preceding bit position (known as carry-in, C_{in}). The output of such a 4:2 Compressor consists of one bit in the succeeding bit position (known as carry-out, C_{out}) and two output bits in the same bit position. The basic 4:2 Compressor structure is depicted in **Figure 3**.

This Compressor is widely used in a multiplier unit, since it compresses four partial product bits into two bits, and produces one carry bit. The structure of the compressor reduces the number of partial product bits by one half at each stage. The speed of such a 4:2 Compressor is represented by the speed of ***three XOR*** gates in series [15,21].

5.3. Multiplier Design

The digital arithmetic multiplication techniques can be divided into three stages: partial products generation stage, partial products summation stage, and the final addition stage. In this section we explain different implementations for the second stage. The second stage is the most critical in the determination of the overall speed of the multiplier. In this stage we can use one of multi-operand addition techniques (Linear tree, Wallace tree, and Compressor tree) to design a multiplier with the minimum area and a low latency. The final stage; however, is an addition using one of the fastest conventional adders, such as the carry look-ahead adder [11].

5.4. Array of Full Adder (FA)

In the multiplier, an array of FA's is used to add the partial products. The multiplier (X) and multiplicand (A) are

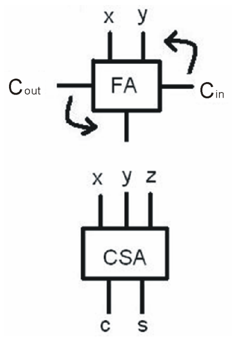


Figure 2. Full adder to carry save adder.

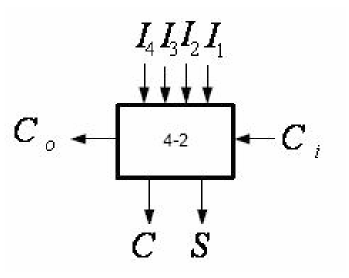


Figure 3. 4:2 Compressor.

unsigned integers with the same word-length, *n* bits. The product requires 2*n* bits to be represented, as shown in (9) (We refer to such multiplication by *n-by-n multiplication*).

$$
\begin{aligned}
&P = X \times A \\
&P = \sum_{i=0}^{n-1} x_i \cdot 2^i \times \sum_{j=0}^{n-1} a_j \cdot 2^j \qquad (9) \\
&P = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \left(x_i \cdot a_j \right) \cdot 2^{i+j}
\end{aligned}
$$

5.5. Linear Carry-Save Adder (CSA) Tree

The multiplier generates partial products using the modified Booth's recoding algorithm proposed in [14]. The generated partial products are summed up using Linear Carry-Save Adder Tree (LCSAT) as shown in **Figure 4**. In each level, we generate two bits of the final product result. The Carry Look-ahead Adder (CLA) is used in the final stage to generate the remaining bits of the result [11].

5.6. Wallace CSA Tree

In the multiplication operation, the addition of the partial products is the most time consuming process. We present a multi-operand addition technique to handle the procedure of repetitive addition, which is referred to as the Wallace tree, illustrated in **Figure 5**.

A Wallace Tree is composed of (3, 2) counters [14]. It achieves the highest degree of parallelism with a delay that grows with $O\left(\log_{3/2} n\right)$. Traditionally, Wallace Trees were not embraced by designers, because they are much harder to design and layout due to their irregular structures [11,14].

5.7. Compressor Tree

The 4:2 Compressor Tree has a more regular structure than the ordinary *CSA* tree. It is made of "3:2 Counter" counters, as the partial products are added up in the form of a binary tree. The compressor structure can be extended to the wanted number of partial products by creating a new structure or combining some existing ones.

In [16-18], the proposed algorithms differentiate between the fast and the slow inputs and outputs of the 3:2 *CSA* counter. Moreover, the 4:2 Compressor adders were used in the partial product summation tree, so as to reduce the delay of the parallel multipliers.

6. Digital Oscillator Analysis and Related Work

The most common methods to generate sinusoidal signals are the recursive methods using a second-order difference Equation (1). It has been shown in [1-10] that

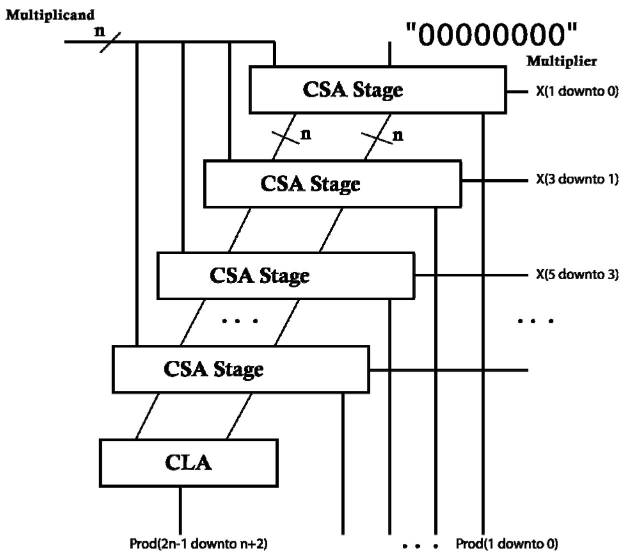


Figure 4. Multiplier based on the Linear Carry-Save adder.

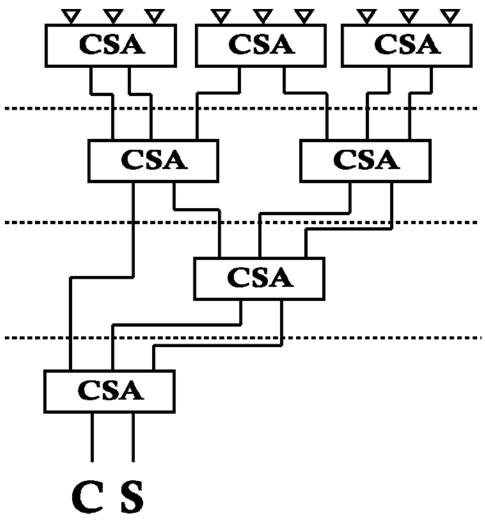


Figure 5. Adding 9-operands using WCSA.

using recursive method, it is difficult to generate a given desired frequency, and this difficulty increases as the desired frequency is reduced where the sensitivity and round-off errors increase by reducing the frequency. Many of recursive digital oscillators have been proposed in literature to increase the range of frequency that can be generated and to reduce the effect of round-off errors. These digital oscillators can be classified into the following:

- Single-Output Direct-Form Digital Oscillator
- Multiple-Output Direct-Form Digital Oscillator
- Complex Digital Oscillator with Integrator
- Combined Digital Oscillator

The direct-form oscillator is a single output oscillator that generates a sinusoidal signal using a second-order difference Equation (1) suffers from a round-off quantization error. The use of recursive methods increases the impact of this error on the output accuracy in the worst case, the quantization errors may accumulate to undesired levels, leading to major discrepancy between the generated output and the ideal output.

To improve the output signal generated from the direct-form oscillator is a single output oscillator, the feedback circuit of the first or second order is used to reduce the effect of the round-off error and increase the number of samples of the generated sinusoidal signal. The feedback circuit has improved the generated sinusoidal signal by reducing the amount of noise in the generated signal [3].

The Multiple-Output Direct-Form Digital Oscillator generates sine and cosine waveforms, namely $y_s(n)$ and $y_c(n)$, with a stable amplitude and fixed phase shift $\pi/2$. The multiple-output direct-form digital oscillator suffers from a round-off quantization error; the round-off error can be reduced by using a second order error feedback as shown in [4].

The combined digital oscillator depends on combining both oscillators that are capable of generating both the real (cosine) and imaginary (sine) components of the sinusoidal signal $e^{j\theta}$. It should be emphasized that each oscillator in the combined digital oscillator can generate sinusoidal signals independently from the other oscillator. In other words, the amplitude and frequency of each oscillator output can be designed separately.

The combined digital oscillator is capable to generate sinusoidal signals with lower frequencies compared to the multiple-output direct-form oscillator. This implies that the generated signal from the combined digital oscillator has more samples than the generated signal from individual multiple-output direct-form oscillator. It should be emphasized that the multiple-output direct-form oscillators that are used to build the combined digital oscillator have no error feedback circuits [5].

An enhancement on structure of combined digital oscillator has been introduced in [6] as shown in **Figure 6**. The proposed digital oscillator represents the multiplier coefficient $2\cos(\theta)$ using $b + 2$ bits; 1 bit for the sign, 1 bit for the integer part and b bits for the fractional part. It has been shown that the number of samples per cycle depends on the smallest value of θ, which is given by

$$\theta_{\min} = \cos^{-1}\left[\frac{1}{2}\left(2-2^{-b}\right)\right] \tag{10}$$

In the proposed structure, the value of the multiplier coefficient can be increased in steps of 2^{-b} over a specific range as given by:

$$\begin{aligned} &\cos\theta'_m = m\cdot 2^{-(b+1)} \\ &m \in \left\{1,2,3,4,\cdots,2^{(b-2)}\right\} \end{aligned} \tag{11}$$

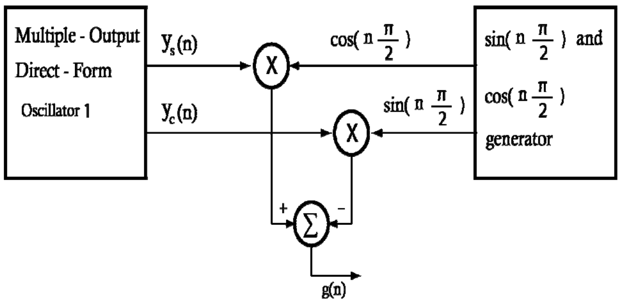


Figure 6. Block diagram of combined digital oscillator proposed in [6].

where θ'_m represents a quantized value of the actual analog value of θ at a certain value of m. When the value of m is small, θ'_m will be close to $\pi/2$. The combined digital oscillator evaluates the difference between the phases of both sinusoidal signals generated by the multiple-output direct-form oscillator and the two sinusoidal differential components, $\sin(n\pi/2)$ and $\cos(n\pi/2)$. The phase of the generated sinusoidal signal will then be evaluated using the formula $\theta_g = (\pi/2) - \theta'_m$. The value of θ_g is very small in comparison with both $\pi/2$ and θ'_m; therefore, the generated sinusoidal signal has a relatively large number of samples, $N \cong 2\pi/\theta_g$.

The complex digital oscillator with integrator structure consists of two digital integrators and two multipliers, arranged in a closed-loop fashion proposed in [7]. The gains for the integrators and multipliers coefficient values have been equivalently distributed over the whole structure of the digital oscillator to increase the frequency resolution and simplify the implementation of the structure. The proposed structure has a quantization error that is less than that of the direct-form digital oscillator. The real and imaginary components (the cosine and sine) of the signal $e^{j\theta}$ are generated.

The linear relationship between the generated frequency of sinusoidal signal and the maximum operation frequency of digital oscillator chip has been proven in [10], the combined digital oscillator of the multiple-output direct-form oscillator is proposed. The generated frequency values are expressed by:

$$\begin{aligned} &f_d = \frac{m\left(2^{-b}\right)}{2\pi} f_{clk} \\ &m = 1,2,\cdots,\left\lceil 2^{b/2}-1\right\rceil \end{aligned} \tag{12}$$

where f_{clk} represents the maximum operating frequency. It is shown in (12) that the generated frequency and the maximum frequency are linearly related.

7. Hardware Implementation and Results

The digital oscillator structures discussed in the previous section is described in VHSIC hardware description language (VHDL) [22,23]. To validate the functionality of these oscillators, we have simulated their implementations using the Mentor Graphic simulation tool Model Sim [24]. After simulation, we synthesized the structures using the Xilinx synthesis tool. The Synthesis process takes the conceptual VHSIC Hardware Description Language (VHDL) design definitions, and generates the logical or physical representation for the targeted silicon device. The FPGA implementation using VIRTEX-5 family was chosen for the synthesis process [25].

7.1. The Use of Pipelining

Pipelining is an advanced hardware technique that, in most cases, significantly improves the performance of digital arithmetic units. The cost of pipelining is usually embedded in the inter-stage buffers (registers). The performance gain is attributed to the large throughput obtained by breaking up the computation path over multiple logic stages with shorter latencies. This enables a faster clock rate but at the expense of a higher hardware complexity [26,27].

7.2. Synthesis Parameters

The following synthesis metrics are considered.

- **Maximum combinational path delay:** This metric represents the total combination delay of the critical path of the digital arithmetic unit.
- **Number of slice Look-Up-Tables (LUT):** The look-up-table represents the direct form implementation of a Boolean function. This metric measures the number of slices used for implementing the look-up-table. It is usually used as an indicator to the utilized layout area.
- **Number of slice registers:** This metric measure the number of slices used for implementing the required registers, which also indicates the utilized area.
- **Number of fully used bit slices:** This metric measures the actual number of used slices.
- **Maximum frequency:** The maximum operating frequency of the digital arithmetic unit.

It should be emphasized that the FPGAs are built from a grid of cells, called Configurable Logic Block (CLBs). Each CLB contains a number of Look-Up Tables (LUTs) and Flip-Flops. This grid is arranged into slices.

7.3. The Sign-Magnitude Multiplier Implementation and Synthesis Results

The multiplication algorithms have been studied in details for four fast multiplier schemes in terms of the maximum combinational path delay and the number of slice LUTs. **Tables 1** and **2** summarize the results for different multiplier schemes.

The synthesis results (**Figure 7**) show that the Compressor tree and the Wallace tree multipliers are superior to the other multipliers in terms of the maximum combinational path delay for large number of bits. It is also shown that the Compressor tree multiplier slightly outperforms the Wallace Tree multiplier.

Figure 8 shows that increasing the number of bits results in an increase in the number of slice LUTs. This increase is due to the need for a larger input/output look-up table when the number of bits is increased. Approximately all multiplier schemes require almost the same number of slice LUTs.

The Compressor tree multiplier has been used in implementing all the digital oscillator structures. It outper-

Table 1. Sign-magnitude multipliers, maximum combinational path delay (ns).

		Number of bits		
		9 bits (s, 8)	17 bits (s, 16)	33 bits (s, 32)
Multiplier schema	Array of full adder	13.157	25.487	49.986
	LCSA	8.810	13.680	23.397
	Wallace tree	10.411	14.743	21.376
	Compressor tree	8.975	13.415	21.233

Table 2. Sign-magnitude multipliers, number of slices LUTs.

		Number of bits		
		9 bits (s, 8)	17 bits (s, 16)	33 bits (s, 32)
Multiplier schema	Array of full adder	99	401	1573
	LCSA	118	421	1618
	Wallace tree	124	446	1561
	Compressor tree	130	422	1647

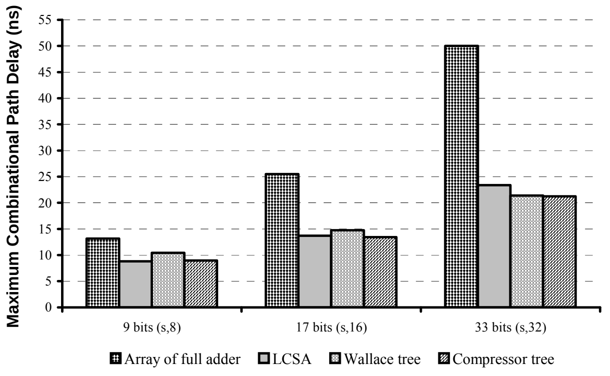


Figure 7. Multiplier results: maximum combination path delay (ns) vs word sizes for different multiplier schemes.

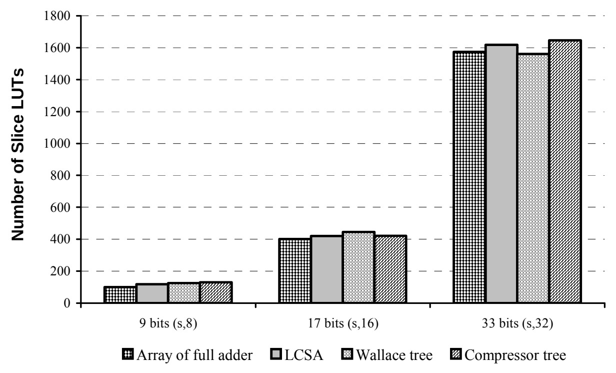


Figure 8. Multiplier results: number of slice LUTs vs multiplier schemes for different word size.

forms all other multiplier schemes in terms of the maximum combinational path delay without a significant increase in the number of slice LUTs. The next section presents some digital oscillator implementation using the Compressor tree multiplier.

7.4. Digital Oscillator Implementation and Synthesis Results

The implementation of the four recursive digital oscillators mentioned earlier is presented in this section. Some digital oscillator structures are implemented with a second-order error feedback as proposed in [3]. The error feedback is introduced to alleviate the quantization error and increase the number of samples per cycle. This will be further explained in the simulation section.

To distinguish between the digital oscillator structures, we give name them as follows.

- *Single-output oscillator*: It produces a single output. If feedback is employed, it will be called "*single-output oscillator with feedback*", and if pipelining is employed it will be preceded by the word "*pipelined*". The main structure of this oscillator was proposed in [3].
- *Multiple-output oscillators*: They are based on the digital oscillator structures proposed in [5]. This oscillator can also be constructed with error feedback or without error feedback and with or without pipelining.
- *Basic combined oscillator*: It is based on the oscillator proposed in [6]. This oscillator can be constructed using two symmetric multiple-output oscillators.
- *Basic complex oscillator* and *advanced complex oscillator*: They are based on the oscillators proposed in [7,9], respectively. Similarly, if pipelining is used these digital oscillators will be called "*basic complex pipelined oscillator*" and *advance complex pipelined oscillator*, respectively.
- *Advanced combined oscillator*: It is it constructed using a single modified multiple-output oscillator structure, arranged with a finite-state machine [10]. This oscillator can have the same variations as in the *basic digital oscillator*.

The pipeline structures for all the digital oscillators mentioned earlier are proposed in this paper. The digital oscillator structures with pipelining are compared with the digital oscillator structures without pipelining for all digital oscillator combinations. The digital oscillator structures are implemented with different word-size: 8, 16, and 32 bits.

Figure 9 shows the value of maximum frequency for different digital oscillator structure types with different word sizes. From these results, we can deduce the following.

- As number of bits is increased, the maximum frequency of each of the digital oscillator structures is decreased.
- The pipelined oscillator structures have higher maximum frequency than the non-pipelined structures.
- Considering the pipelined versions of all digital oscillator structures (except the basic and advanced digital oscillator), the maximum frequency of the digital oscillator without error feedback is slightly higher than that of the digital oscillator with error feedback.

Figures 10 and **11** show the number of slice register and the number of fully used bit slices for different oscillator structures, and for different number of bits, respectively. **Figures 10** and **11** imply the following.

- Both the number of fully used bit slices and the number of slice registers show the same behavior for the different oscillator structures.
- As the number of bits is increased, the number of fully used bit slices of all digital oscillator structure types increases. This is also true for the number of slice registers.
- All pipelined oscillator structures have higher numbers of fully used bit slices than the non-pipelined oscillator structures. This is also valid for the number of slice registers. This is a natural consequence of the use of pipelining, as it requires extra inter-stage registers.

Figure 12 shows the number of slice LUTs versus the different oscillator structure and for different number of bits. We can deduce the following.

- As number of bits is increased, the number of slice LUTs of all digital oscillator structures is increased. This is also valid for the number of slice registers.
- The pipelining does not affect the number of slice LUTs in all oscillator structure for all word sizes.

7.5. Digital Oscillator Simulation Results

This sinusoidal signal is generated using the following assumptions.

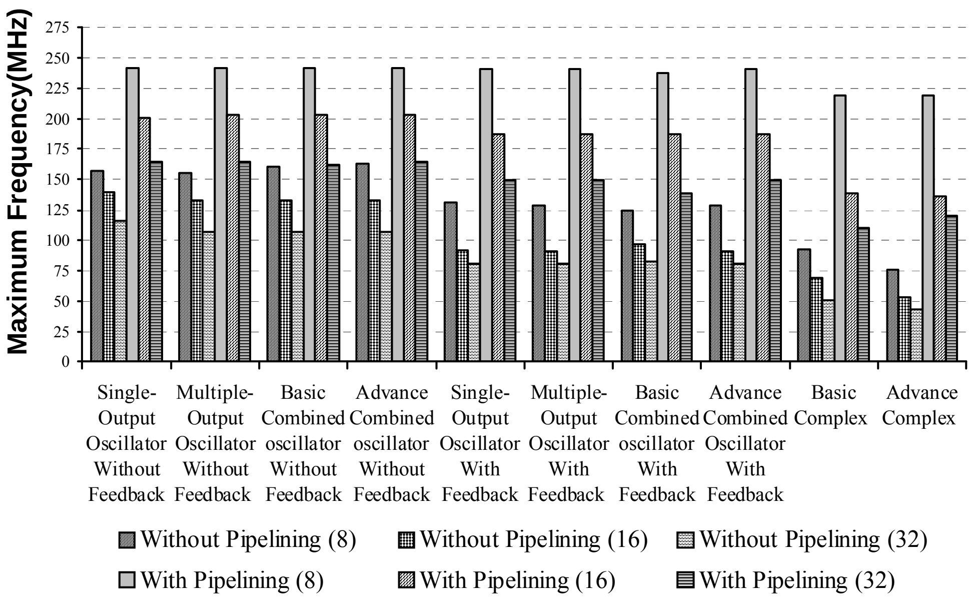


Figure 9. Maximum frequency vs oscillator types for different word sizes.

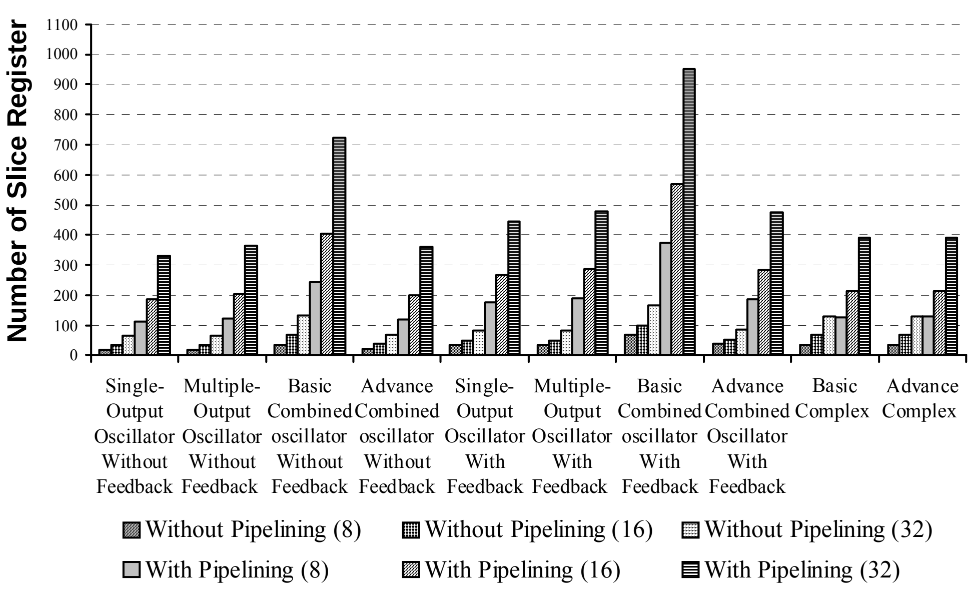


Figure 10. Number of slice register vs oscillator types for different word sizes.

- The number format is (**s**, **1**, **7**), that is, seven bits for the fraction part, one bit for the integer part, and one bit for the sign.
- The multiplier coefficient $2\cos\theta$ is determined using $2^F \cdot 2\cos\theta = 255$. Hence, the theoretical value for both θ, N are given by:

$$\begin{aligned}\theta_t &= \cos^{-1}(255/256) = \cos^{-1}(0.99609375) \approx 5.066 \\ N_t &= 360/5.066 \approx 71.062\end{aligned} \quad (13)$$

- The initial condition $y(-2)$ is assumed to be equal to (−11).

Figure 13 shows the sinusoidal signal generated by the single-output direct-form oscillator without the 2nd level error feedback. The generated number of samples in a complete cycle is $N_g \cong 2\pi/\theta_g$, where θ_g is the generated phase. $N_g \cong 46$ and $\theta_g \cong 7.826$. The generated number of samples is less than the expected theoretical number of samples due to the quantization error.

Figure 14 shows the sinusoidal signal generated by the single-output direct-form digital oscillator with 2nd level error feedback. The sinusoidal signal has a number of samples that is approximately equal to theoretical number of samples. $N_g \cong 71$ and $\theta_g \cong 5.07$.

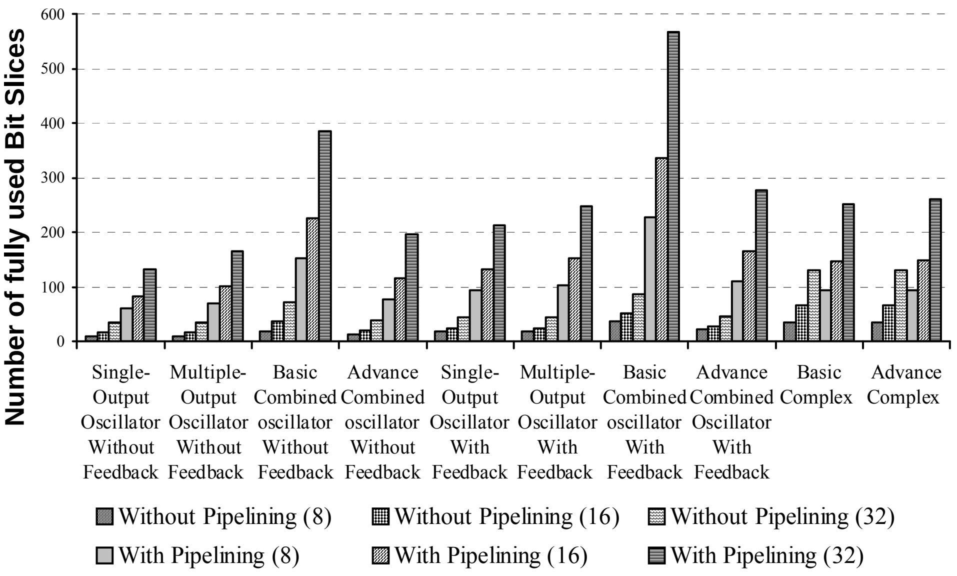


Figure 11. Number of fully used bit slices vs oscillator types for different word sizes.

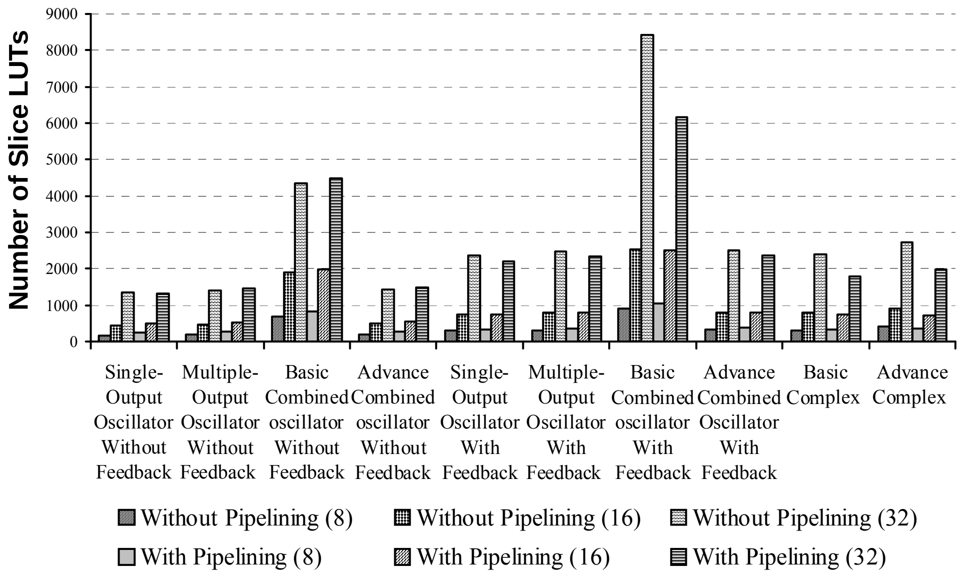


Figure 12. Number of slice LUTs vs oscillator types for different word sizes.

The sinusoidal signal generated by the advanced combined digital oscillator is shown in **Figure 15**. The multiple-output direct-form digital oscillator is used to generate a sinusoidal signal with the following parameters:

- Multiplier coefficient $2\cos\theta$: It is computed using $2^F \cdot 2\cos\theta = 7$. The theoretical value for both θ, N, are then given by:

$$\begin{aligned}\theta_t &= \cos^{-1}(7/256)\\ &= \cos^{-1}(0.02734375) \approx 88.433\\ N_t &= 360/88.433 \approx 4.071\end{aligned} \tag{14}$$

- Initial condition $y(-2)$: It is given by $y(-2) = -248$.

The advance combined digital oscillator depends on the phase difference between $\pi/2$ and the generated phase from several multiple-output direct-form digital oscillators. The above combined digital oscillator generates a sinusoidal signal with

$$\begin{aligned}\theta &\cong (\pi/2) - 88.433 \cong 1.567\\ N &= 360/1.567 = 230\end{aligned} \tag{15}$$

8. Conclusions

We have utilized advanced architecture and arithmetic techniques to implement digital oscillator.

The synthesis results have shown that the pipelined

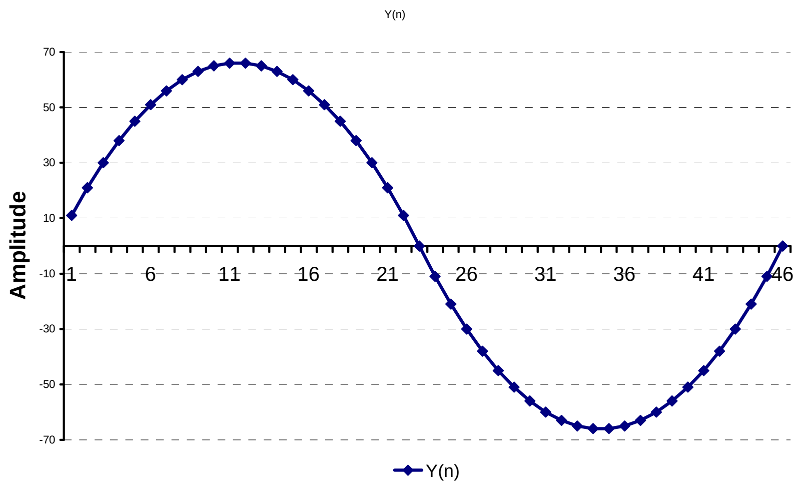


Figure 13. Sinusoidal signal generated by a single-output direct-form digital oscillator without 2nd level error feedback.

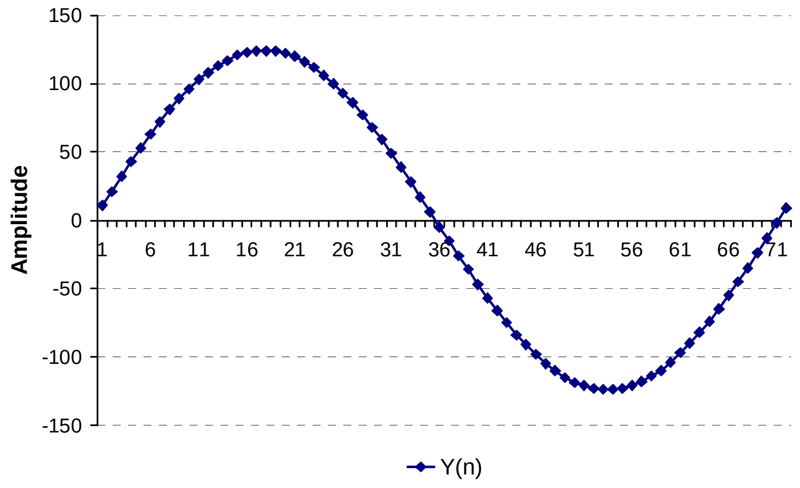


Figure 14. Sinusoidal signal generated by a single-output direct-form digital oscillator with 2nd level error feedback.

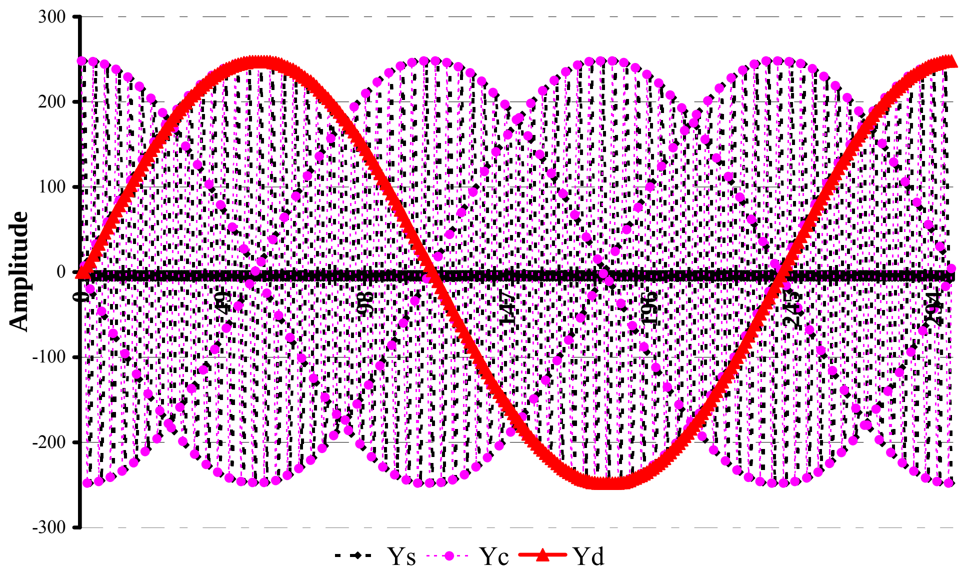


Figure 15. Sinusoidal signal generated by an advanced combined digital oscillator.

digital oscillator structures are superior in terms of the maximum frequency when compared with the non-pipelined ones. This has led to a significant enhancement of the generated sinusoidal signal in terms of the frequency and number of samples per cycle.

The simulation results of oscillator structures have shown that the combined digital oscillators proposed in [10] have produced sinusoidal signals with a large number of samples in comparison with the other digital oscillators. This makes the combined digital oscillator structure proposed in [10] the preferable digital oscillator structure among all digital oscillator structures.

It is to be noted that our work is sensitive to the arithmetic algorithms used. Thus, if faster arithmetic algorithms are proposed, new implementations for digital oscillators should be devised.

REFERENCES

[1] M. Schanerberger and S. S. Awad, "The Implementation of a Digital Sine Wave Oscillator Using the TMS320C25: Distortion Reduction and Applications," *IEEE Transactions on Instrumentation and Measurement*, Vol. 39, No. 6, 1990, pp. 870-873. doi:10.1109/19.65786

[2] M. Al-Ibrahim, S. Bataineh and A. Al-Khateeb, "Digital Sinusoidal Oscillators with High Frequency Resolution and Low Harmonic Distortion," *International Journal of Electronics*, Vol. 87, No. 10, 2000, pp. 1209-1218. doi:10.1080/002072100415657

[3] A. I. Abu-El-Haija and M. M. Al-Ibrahim, "Improving Performance of Digital Sinusoidal Oscillators by Means of Error Feedback Circuits," *IEEE Transactions on Circuits and Systems*, Vol. 33, No. 4, 1986, pp. 373-380. doi:10.1109/TCS.1986.1085932

[4] N. J. Fliege and J. Wintermantel, "Complex Digital Oscillators and FSK Modulators," *IEEE Transactions on Signal Processing*, Vol. 40, No. 2, 1992, pp. 333-342. doi:10.1109/78.124943

[5] M. M. Al-Ibrahim and A. M. Al-Khateeb, "Efficient Low Frequency Digital Sinusoidal Oscillator," *International Journal of Electronics*, Vol. 81, No. 2, 1996, pp. 159-169. doi:10.1080/002072196136823

[6] M. M. Al-Ibrahim and A. M. Al-Khateeb, "Digital Sinusoidal Oscillator with Low and Uniform Frequency Spacing," *IEE Proceedings of Circuits*, *Devices and Systems*, Vol. 144, No. 3, 1997, pp. 185-189. doi:10.1049/ip-cds:19971004

[7] M. M. Al-Ibrahim and A. M. Al-Khateeb, "Extremely Low Sensitivity Digital Sinusoidal Oscillator Structure," *International Journal of Electronics*, Vol. 85, No. 6, 1998, pp. 755-765. doi:10.1080/002072198133806

[8] A. A. Hiasat and A. M. Al-Khateeb, "New High-Resolution Digital Sinusoidal Oscillator Structure with Extremely Low Frequency and Sensitivity," *International Journal of Electronics*, Vol. 86, No. 3, 1999, pp. 287-296. doi:10.1080/002072199133427

[9] M. M. Al-Ibrahim (Jarrah), "A Multi Frequency Range Digital Sinusoidal Oscillator with High Resolution and Uniform Frequency Spacing," *IEEE Transactions on Circuits and Systems—II*: *Analog and Digital Signal Processing*, Vol. 48, No. 9, 2001, pp. 872-876.

[10] M. M. Al-Ibrahim, "A New Hardware-Efficient Digital Sinusoidal Oscillator with Low- and Uniform-Frequency Spacing," *Electrical Engineering*, Vol. 85, No. 5, 2003 pp. 255-260. doi:10.1007/s00202-003-0168-4

[11] M. D. Ercegovac and T. Lang, "Digital Arithmetic," Morgan Kaufmann Publishers, Burlington, 2003.

[12] C. K. Koc, "Parallel Canonical Recording," *Electronics Letters*, Vol. 32, No. 22, 1996, pp. 2063-2065. doi:10.1049/el:19961402

[13] A. D. Booth, "A Signed Binary Multiplication Technique," *Quarterly Journal of Mechanics and Applied Mathematics*, Vol. 4, No. 2, 1951, pp. 236-240. doi:10.1093/qjmam/4.2.236

[14] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, Vol. 13, No. 2, 1964, pp. 14-17. doi:10.1109/PGEC.1964.263830

[15] D. Villeger and V. G. Oklobdzija, "Evaluation of Booth Encoding Techniques for Parallel Multiplier Implementation," *Electronics Letters*, Vol. 29, No. 23, 1993, pp. 2016-2017. doi:10.1049/el:19931345

[16] V. G. Oklobdzija and D. Villeger, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology," *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, Vol. 3, No. 2, 1995, pp. 292-301.

[17] V. G. Oklobdzija, D. Villeger and S. S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," *IEEE Transactions on Computers*, Vol. 45, No. 3, 1996, pp. 294-306. doi:10.1109/12.485568

[18] P. F. Stelling and V. G. Oklobdzija, "Optimal Circuits for Parallel Multipliers," *IEEE Transactions on Computers*, Vol. 47, No. 3, 1998, pp. 273-285. doi:10.1109/12.660163

[19] A. Weinberger and J. L. Smith, "A Logic for High-Speed Addition," *National Bureau of Standards Circulation*, Vol. 591, 1958, pp. 3-12.

[20] A. Weinberger, "4:2 Carry-Save Adder Module," *IBM Technical Disclosure Bulletin*, Vol. 23, No. 8, 1981, pp. 3811-3814.

[21] D. Villeger and V. G. Oklobdzija, "Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products," 1993 *Conference Record of The Twenty-Seventh Asilomar Conference on Signals*, *Systems and Computers*, Vol. 1, 1993, pp. 781-784.

[22] P. J. Ashenden, "The VHDL Cookbook," Department of Computer Science, University of Adelaide, 1990.

[23] U. Heinkel, M. Padeffke, W. Haas, T. Buerner, H. Braisz, T. Gentner and A. Grassmann, "The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design Including VHDL-AMS," Wiley, Chichester, 2000.

[24] Introduction to ModelSim Simulation Software Tool.

http://www.model.com/

[25] Introduction to Xilinx Synthesize Software Tool. http://www.xilinx.com/

[26] D. A. Patterson and J. L. Hennessy, "Computer Organization and Design: The Hardware/Software Interface," 3rd Edition, Morgan Kaufmann Publishers, Burlington, 2004.

[27] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," 3rd Edition, Morgan Kaufmann Publishers (Elsevier), Burlington, 2002.