# THE IMPLEMENTATION OF A DIRECT DIGITAL SYNTHESIS BASED FUNCTION GENERATOR USING SYSTEMC AND VHDL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UĞUR KAZANCIOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 2007

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Murat AŞKAR
Supervisor

Examining Committee Members

| | | |
|---|---|---|
| Assist. Prof. Dr. Haluk KÜLAH | (METU, EE) | |
| Prof. Dr. Murat AŞKAR | (METU, EE) | |
| Assist. Prof. Dr. Behzat ŞAHİN | (METU, EE) | |
| Assist. Prof. Dr. Çağatay CANDAN | (METU, EE) | |
| İrfan OKŞAR (M.Sc.) | (ASELSAN Inc.) | |

| | |
|---|---|
| Name, Last name | : Uğur KAZANCIOĞLU |
| Signature | : |

## ABSTRACT

## THE IMPLEMENTATION OF A DIRECT DIGITAL SYNTHESIS BASED FUNCTION GENERATOR USING SYSTEMC AND VHDL

KAZANCIOĞLU, Uğur
M.Sc., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Murat AŞKAR

February 2007, 101 pages

In this thesis, a direct digital synthesis (DDS) based function generator design module is presented, defined and implemented using two digital hardware modeling/design languages namely SystemC and VHDL. The simulation, synthesis and applicability performances of these two design languages are compared by following all digital hardware design stages. The advantages and open issues of SystemC based hardware design flow are emphasized in order to be a reference for future studies.

SystemC initially appeared as a modeling language like HDL design languages. In the last years, SystemC gained popularity also as a hardware design language and it is expected to become alternative to traditional design languages. Using a single platform for hardware modeling, design and verification reduces the spent time and cost.

The designed DDS function generator module supports standard I2C and UART communication protocols and it is in ready to use format for digital applications. In this thesis, the function generator module VHDL code is implemented into Xilinx FPGA and verified on the hardware platforms.

**Keywords**: SystemC, VHDL, SystemC Synthesis, Direct Digital Synthesis, Function Generator, FPGA

# ÖZ

## DOĞRUDAN SAYISAL SENTEZ TABANLI FONKSİYON ÜRETECİNİN SYSTEMC VE VHDL KULLANILARAK GERÇEKLENMESİ

KAZANCIOĞLU, Uğur
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Murat AŞKAR

Şubat 2007, 101 sayfa

Bu tezde SystemC ve VHDL isimli tasarım ve modelleme dilleri kullanılarak doğrudan sayısal sentez tabanlı fonksiyon üreteci modülü sunulmuş, tanımlanmış ve uygulanmıştır. Bu iki tasarım dilinin simülasyon, sentez ve uygulanabilirlik performansları tüm sayısal donanım tasarım evreleri takip edilerek karşılaştırılmıştır. SystemC tabanlı donanım tasarım akışının avantajları ve açık konuları ileriki çalışmalar için referans olması amacıyla vurgulanmıştır.

SystemC ilk olarak HDL tasarım dilleri gibi modelleme dili olarak ortaya çıkmıştır. SystemC son yıllarda donanım tasarım dili olarak da popülerlik kazanmış ve geleneksel tasarım dillerine alternatif olması beklenmektedir.

Donanım modelleme, tasarım ve doğrulama için tek bir platform kullanılması harcanan zaman ve maliyeti azaltmaktadır.

Tasarlanan fonksiyon üreteci modülü standart I2C ve UART haberleşme protokollerini desteklemektedir ve sayısal uygulamalar için kullanıma hazır formattadır. Bu tezde fonksiyon üreteci modülü VHDL kodu Xilinx FPGA içine uygulanır ve donanım platformlarında doğrulanır.

**Anahtar kelimeler :** SystemC, VHDL, SystemC Sentezi, Doğrudan Sayısal Sentez, Fonksiyon Üreteci, FPGA

## ACKNOWLEDGMENTS

I would like to express my deepest sense of gratitude to my supervisor Prof. Dr. Murat AŞKAR for his guidance and invaluable ideas.

I am deeply grateful to ASELSAN Inc. for providing tools and other facilities throughout this study.

I would like to forward my appreciation to all my friends and colleagues who contributed to my thesis with their continuous encouragement.

I would also like to express my profound appreciation to my family for their continuous support.

# TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ASIC | Application Specific Integrated Circuit |
| ASK | Amplitude Shift Keying |
| A/D | Analog to Digital |
| DAC | Digital to Analog Converter |
| DDS | Direct Digital Synthesis / Synthesizer |
| DSP | Digital Signal Processor |
| D/A | Digital to Analog |
| EDA | Electronic Design Automation |
| FPGA | Field Programmable Gate Array |
| FSK | Frequency Shift Keying |
| HDL | Hardware Description Language |
| HW | Hardware |
| IC | Integrated Circuit |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Intellectual Property |
| I2C | Inter-Integrated Circuit |
| NCO | Numerically Controlled Oscillator |
| OSCI | Open SystemC Initiative |
| PLD | Programmable Logic Device |
| PLL | Phase Locked Loop |
| PSK | Phase Shift Keying |
| RTL | Register Transfer Level |
| SNR | Signal to Noise Ratio |
| SoC | System On Chip |
| UART | Universal Asynchronous Receiver / Transmitter |
| VCD | Value Change Dump |
| VHDL | VHSIC Hardware Description Language |
| VHSIC | Very High Speed Integrated Circuit |
| VLSI | Very Large Scale Integrated Circuit |

# CHAPTER 1

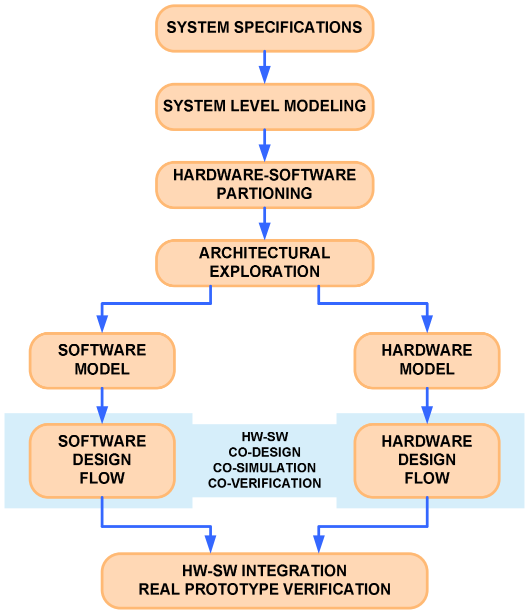
# INTRODUCTION

A direct digital synthesis (DDS) based function generator design core is defined, implemented and verified in this thesis. The function generator design is also a case study to investigate and compare the traditional HDL and new SystemC based design flows. Throughout the study, the traditional design flow will be followed using VHDL as a design language.

The electronic system design becomes more complex as the demand for complete systems increases. Today's modern technology enables producing a chip that holds all of the necessary electronic circuitry for a complete system. This technology is known as System on Chip (SoC) and the increased complexity of SoC technology addresses the design and verification at system level [1].

In traditional system design flow, the system is partitioned into hardware and software at early stages. Then, the hardware and software engineers design their respective components with the lack of communication. In some cases, the produced solution may not be the desired one. The integration of the hardware and software may lead problems due to isolation of the designs. Unavoidable results in this flow are higher cost and long design time.

In modern system design flow, system should be modeled at various levels of abstraction. This requirement appeared due to the fact that the complexity problem can be solved only using higher abstraction levels [2, 3, 4, 5]. The modern system languages should also constitute a single platform for both hardware and software designers. The hardware and software should be developed in parallel and this would remove painful hardware and software integration problem. Hence, modern system design flow (illustrated in Figure 1-1) is proposed as a solution for shortcoming of the traditional system design flow.



SYSTEM SPECIFICATIONS
SYSTEM LEVEL MODELING
HARDWARE-SOFTWARE PARTIONING
ARCHITECTURAL EXPLORATION
SOFTWARE MODEL
HARDWARE MODEL
SOFTWARE DESIGN FLOW
HW-SW CO-DESIGN CO-SIMULATION CO-VERIFICATION
HARDWARE DESIGN FLOW
HW-SW INTEGRATION REAL PROTOTYPE VERIFICATION

**Figure 1-1 Modern System Design Flow**

The new EDA (Electronic Design Automation) tools are proposed to replace the traditional system design flow with modern system flow. Moreover, various design languages have been appeared in the market and this leads to tremendous discussion on these languages [1, 6]. Open SystemC Initiative (OSCI) organization's SystemC language has been developed as a modern system level modeling and design language to simplify the system level design problems [7, 8, 9]. IEEE has approved SystemC (IEEE 1666) as electronic design standard based on the SystemC 2.1 Language Reference Manual. Since SystemC is supported by all of the major EDA companies, it seems to be dominant among the system level modeling and design languages.

SystemC supports a hierarchical system design environment. In this environment system designers can model and verify the designs expressed at true system levels of abstraction. This allows the system designer to make low-level designs or use higher abstraction levels. SystemC links the system model to hardware design implementation and verification [10]. The designers notice the advantage of reduced simulation time. Additionally, it can be used to simulate software algorithms. A wide range of simulation support enables SystemC to address all aspects of SoC design.

SystemC initially appeared as a modeling language like HDL design languages. The new EDA tools are proposed to directly synthesize the SystemC descriptions like HDL synthesis. There are not necessary numbers of well developed SystemC tools that support direct synthesis of SystemC descriptions. In place of these tools, SystemC descriptions are written in synthesizable format for which [11, 12] includes some descriptions and then SystemC to HDL translational tools are used [13, 14, 15, 16]. The traditional HDL and SystemC based hardware design flows are illustrated in Figure 1-2.

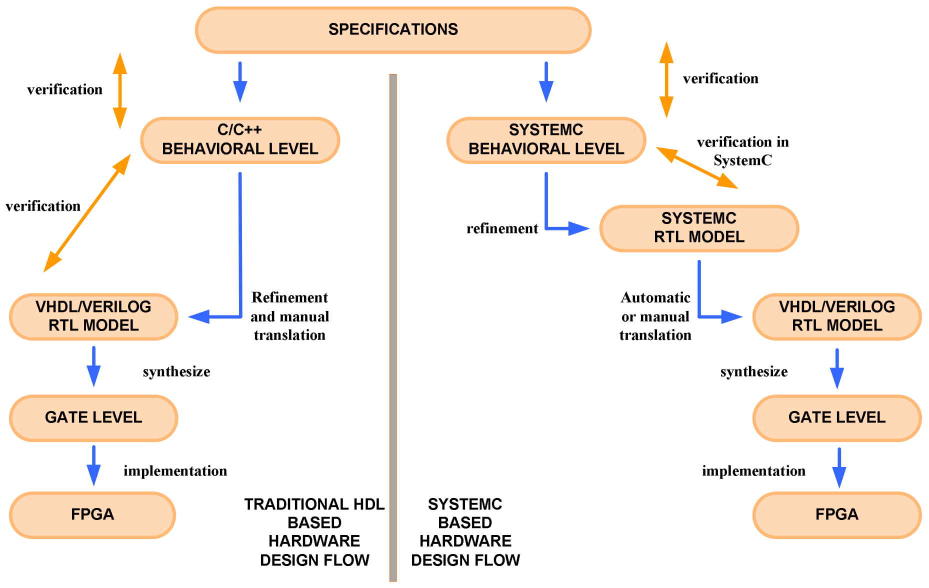


**Figure 1-2 Traditional HDL and SystemC Digital Hardware Design Flows**

The traditional design starts with developing behavioral level descriptions using C/C++. Then the models are rewritten manually using design languages such as VHDL and Verilog. The synthesis and implementation processes are applied after verification of the RTL model. Using the same design platform for behavioral and RTL models is an advantage for the designer, and this is satisfied by the modern SystemC based design flow. In traditional approach, the designer maintains both C and RTL models in different environments. However, refinement, debugging and RTL synthesis of a design can be done in a single SystemC platform. This reduces the time and effort. The translation of the SystemC RTL descriptions to HDL is a disadvantage in terms of the resource utilization and timing, but the new well developed tools having a direct SystemC description synthesis capability will dismiss this disadvantage.

In this study, both traditional and SystemC design flow is studied by implementing a DDS function generator having various periodical waveform generation capability.

Periodical waveform generation is a key function for all communication systems. Periodical waveforms account for many of the RF and microwave signals in communications, radar and test systems. Since the communication market is in rapid development, various ways in order to generate periodical waveforms were discovered. The sine wave is a well known periodical waveform. For long years, analog circuits such as Phase Locked Loop (PLL) synthesizers were used to generate a sine wave. However the developments in the high-density integrated circuit technology revolutionized the periodic waveform generation. In the last years, the periodic waveforms below 1GHz in frequency are generated mostly by the digital technique known as Direct Digital Synthesis (DDS). The new DDS chips with capability of generating higher frequencies increased the popularity of DDS technique more and more in the last years.

The frequency synthesizers use various periodical waveform production techniques and these can be grouped as (i) indirect, (ii) direct and (iii) hybrid. The indirect synthesizers use phase locked loop techniques to multiply the reference input clock. The output signal frequency is the multiplication of the input frequency with some constant. The output frequency increment of the PLL is reference clock frequency and this results in not satisfying resolution for many applications. On the other hand, these synthesizers are good at output spectrum and can generate output frequencies up to 10 GHz. The direct frequency synthesizers use digital techniques to generate waveforms. These synthesizers can generate lower frequencies but, they allow fine step sizes and more accurate frequency values. Hybrid synthesizers combine the advantages of the direct and indirect techniques to achieve both higher frequencies and fine step sizes.

The direct frequency synthesizers use DDS technique which lets generating sine waves at very precise frequencies [17, 18]. As the name implies, the analog sine wave is completely generated by digital circuits in this technique. The digitally quantized samples of the desired waveform are generated at the input reference clock frequency. The generated digital samples of the waveform are converted to analog signals using the D/A converters and filter circuits. The DDS technique has some superior advantages over classical PLL synthesizers. Some of these advantages are as follows:

- DDS technique allows for very fast frequency switching at a low cost. The frequency switching time can be in nanoseconds level.
- The waveform frequency is digitally adjustable with microhertz frequency resolution.
- The waveform phase and amplitude can be adjusted digitally.
- The implementation of DDS is easier than the classical synthesizers.
- The DDS core can be combined with additional signal processing blocks to make clock generators and modulators.

On the other hand, DDS technique has also disadvantages as stated below:

- The maximum output frequency is less than the clock source frequency.
- The digital generation of the sine wave results in distortion. The generated waveform is not spectrally pure [19].

- Spurious performance of DDS is dominated by the DAC. At higher speed DAC IC's linearity is not good.

The DDS implementation usually uses addressing of the samples contained in the sine look-up table. Since the samples are generated only by addressing, arbitrary periodical waveform generation is also possible with this technique. Only thing to do is to replace the sine look-up table with a look-up table that includes arbitrary wave samples. In this study, the DDS technique is applied only to generate sine wave samples.

There are many DDS commercial ICs operating at frequencies from 10 MHz to 1-2 GHZ in the market [20, 21, 22, 23, 24, 25]. There is a competition among the suppliers to present faster and more capable DDS chips. Analog Devices is one of these suppliers and it presents innovative and fast DDS chips in the last years. For instance, Analog Devices' AD9858 model [20] is a complete DDS with internal 10-bit DAC. It operates at clock rates up to 1 GHz. The AD9858 can actually work with clock rates up to 2 GHz; since it also includes a divide-by-two circuit on the clock input port. This reference clock frequency allows generating output waveform frequencies 400 MHz or more. Using its 32-bit frequency tuning words, the synthesizer achieves frequency resolution below 1Hz. The frequency tuning and control words of AD9858 are loaded via its 8-bit parallel or synchronous serial ports.

Today's fast changing electronic world requires that the whole digital components on the boards will be collected in one package. Among the several integration techniques, one of the most commonly used approaches is the utilization of Field Programmable Gate Arrays (FPGAs). This fact led to a tremendous acceleration to the Gate Array technology in the last years. Most of the digital designers prefer achieving all digitally complex computations with a single chip in a simple and effective way. FPGA technology does not only support logic based designs, it also supports DSP and processor based applications. As a result, major companies design various intellectual properties, abbreviated as IP, for FPGAs. There are also many companies preparing DDS IPs [26, 27, 28, 29]. These designs are in ready to use format and can be implemented into a target FPGA simply. For instance Xilinx's DDS Compiler V1.0 [27] is a complete DDS IP with adjustable frequency and output resolution. It also supports up to 16 independent channels.

The presented function generator module combines the most recent DDS system with controller and standard interface units. The designed DDS function generator core allows generating standard waveforms and making FSK/PSK modulation. The DDS function generator core is implemented into FPGA and verified with hardware based tests. During the design period, the most recent digital design environments are used.

The designed function generator module has two serial peripheral interfaces which are Inter Integrated Circuit (I2C) bus [30] and Universal Asynchronous Receiver/Transmitter (UART). These interfaces are used to load waveform configuration and design control parameters. The type, frequency, phase offset and amplitude parameters are sent to the function generator using these serial interfaces. If this function generator core is implemented into the packaged Application Specific Integrated Circuit (ASIC), then I2C interface usage will be helpful for communication with other ASICs. If the design is implemented into FPGA and will be tested for different design parameters by means of a computer, the UART interface will be very useful since the communication over UART are supported nearly by all computers. Additionally, a function generator having UART interface can communicate with nearly all microcontrollers and microprocessors.

The design of function generator using both SystemC and VHDL makes the comparison of different design flows possible. In this thesis, the power of these design languages in digital hardware applications is also observed and they are compared according to synthesis, simulation and applicability performances. The comparison is made by following all the design stages. The advantages and disadvantages of these two different hardware design cycles are emphasized to be helpful for future applications.

The organization of this thesis is as follows. In Chapter 2, the background information about SystemC and VHDL will be given. The syntax of these two languages will be compared. The basic information about DDS function generators and commercial DDS applications will be presented. In addition, the information about DDS methodology and the communication standards used in the design will also be given. These are necessary for the explanation of the design and interpretation of test results.

Chapter 3 covers full design description of the function generator module. First, the followed SystemC and VHDL based RTL design flows are presented. Then the design specifications and used hardware/software resources are given. The explanations of the function generator sub-modules are presented and the detailed data flow schematic of function generator module is given.

Chapter 4 covers the verification of the functional sub-modules and the whole function generator design. The generated SystemC and VHDL design descriptions are tested separately in software environments. Only the simulations which are really necessary to understand the functionality are given. The synthesis of function generator VHDL descriptions, implementation into FPGA and hardware based tests are left to end of this chapter.

Results of the study are presented in Chapter 5. The applied digital hardware design cycles and designed DDS function generator core are discussed and further suggestions are given for future studies.

# CHAPTER 2

# DESIGN LANGUAGES AND DDS FUNCTION GENERATORS

## 2.1 DESIGN LANGUAGES

### 2.1.1 VHDL

VHDL is an IEEE hardware description language. This language is developed with the goal to develop very high-speed integrated circuits. VHDL is one of industry's widely used standard languages used to describe digital systems. Another hardware description language Verilog is also widely used. Both of these languages allow describing and simulating complex digital systems.

VHDL is used mainly for development of Application Specific Integrated Circuit (ASIC). The synthesis tools transform the written VHDL code to gate-level netlist. This netlist defines the layout of the ASICs. VHDL codes are also written for FPGAs to describe the internal hardware structures. Since the synthesis of the codes gives suboptimal results, VHDL is not used for design of noncomplex Programmable Logic Devices (PLDs).

Digital systems can be represented using different levels of abstraction. There are four abstraction levels of a digital circuit. These levels can be ordered as behavioral, register transfer level (RTL), gate-level and layout. The functional description of the model is defined in the behavioral level. The behavioral descriptions can be simulated, but they may not be synthesized. The design is divided into concurrent and sequential elements in RTL. In this level, concurrent statements are executed in parallel as soon as data arrives at the input. Sequential statements are executed in the sequence that they are specified. RTL description uses only the little VHDL language constructs. A strict methodology is followed to write the design codes. The generated codes using RTL methodology are in synthesizable code format. Third abstraction level is named as gate-level. The design is represented as a netlist with logic gates and storage elements in this level. The last abstraction level is layout level. The different cells of the target technology are placed on the chip and the connections are routed. After the layout has been verified, the circuit is ready for the production process.

The entity is main concept of the digital designs in VHDL. A design entity can be divided in to two parts which are entity declaration and architecture body. The entity declaration defines the external interfaces of the design entity. It defines interaction of the different VHDL modules in a straightforward manner. The architecture body represents the internal descriptions of the design entity. The architectures can contain signals, processes and instantiations of other entities.

The statements within architecture operate concurrently. This led to define VHDL constructs (processes) to achieve necessary sequential behavior. A process consists of a sequence of statements, which are executed sequentially like in conventional programming languages, whereas the processes themselves are treated concurrently like other statements.

The signals are used for passing information among VHDL processes or entities. A process may read and write signals. It may be sensitive to signals. Signal assignments require a delay before the signal assumes its new value.

### 2.1.2 SYSTEMC

The SystemC class library has been developed to support system level design. It has been developed by a group of companies, universities and individuals forming the Open SystemC Initiative (OSCI). IEEE has approved

SystemC (IEEE 1666) as electronic design standard based on the SystemC 2.1 Language Reference Manual developed by OSCI [35].

SystemC is an open source C++ library that is emerging as a standard for high-level design and system modeling. It addresses the increasing complexity of SoC designs at system level. It provides to describe and simulate concurrent hardware constructs using ordinary C++ syntax. This leads to co-design and co-verification of hardware and software in a single environment.

Modules are the principal building blocks of a SystemC design hierarchy. A SystemC model usually consists of several modules which communicate via ports. Processes are the principal computation elements which fulfill necessary sequential behavior. They run concurrently with other processes. Events allow the synchronization between processes.

Ports of a module are the external interfaces that pass information to and from a module. They trigger actions within the module. Signals create the connections between the module ports allowing the modules to communicate. Channels are the communication elements of SystemC. They are generalized form of signals. Complex communication structures can be modeled using channels.

### 2.1.3 COMPARISION OF SYSTEMC AND VHDL SYNTAXES

Since SystemC allows modeling concurrent processes described by ordinary C++ syntax, it can be easily learned by the designers who already make applications using HDL and C++. SystemC has similarities to VHDL semantically, but it has a syntactical overhead compared to VHDL. Some basic language syntaxes of the VHDL and SystemC are compared in Table 2-1. Full adder's SystemC and VHDL codes are given as a comparison example in Table 2-2. SystemC words in capitals (SC MODULE, SC_CTOR or SC METHOD) are macros and hide the real C++ syntax to provide noncomplex syntax for SystemC.

**Table 2-1 Comparison of Some Basic SystemC and VHDL Syntaxes**

| SystemC Syntax | VHDL Syntax |
|---|---|
| (Logic Values) SC_LOGIC_0, SC_LOGIC_1, SC_LOGIC_X, SC_LOGIC_Z | (Logic Values) ‘0’, ‘1’ ‘X’, ‘Z’ |
| (Input & Output Port Declaration) sc_in<sc_logic> A, B; sc_out<sc_logic> C, D; | (Input & Output Port Declaration) A, B : in std_logic; C, D : out std_logic; |
| (Variable Declaration) sc_logic v_bit; sc_lv<3> v_vec; | (Variable Declaration) variable v_bit : std_logic; variable v_vec : std_logic_vector(2 downto 0); |
| (Signal Declaration) sc_signal<sc_logic> s_bit; sc_signal<sc_lv<3> > s_vec; | (Signal Declaration) signal s_bit : std_logic; signal s_vec : std_logic_vector(2 downto 0); |
| C = A & B; C = A \| B ; C = A ^ B; C = ~A; | C <= A and B; C <= A or B; C <= A xor B; C <= not A; |
| if (A= =SC_LOGIC_0) C = B; else if (A= = SC_LOGIC_1) D = B; else C = A; | if (A=’0’) then C <= B; elsif (A=’1’) then D <= B; else C <= A; end if; |
| sc_uint<2> tempK = K.read( ); switch(tempK){ case 0: C = B; break; case 1: D = B; break; default: C = A; } | case K is when “00” => C <= B; when “01” => D <= B; when others => C <= A; end case; |
| SC_METHOD (process); sensitive << clk.pos( ); | process (clk) ………. if ( clk’event and clk = ‘1’) then |
| SC_METHOD (process); sensitive << clk.neg( ); | process(clk) ………. if ( clk’event and clk = ‘0’) then |
| SC_METHOD (process); sensitive << A << B; | process (A, B) |

**Table 2-2 Comparison of SystemC and VHDL Full Adder Codes**

**SystemC Full Adder Code**

```
#include “systemc.h”
#include “half_adder.h”

SC_MODULE (full_adder)
{
        sc_in<sc_bit>   A, B, Cin;
        sc_out<sc_bit> Sum, Cout;

        sc_signal <sc_bit> sigC1,sigC2,sigS1;

        half_adder      H1;
        half_adder      H2;

        SC_CTOR (full_adder):
        H1 (“H1”), H2 (“H2”)
        {

                H1.A(A);
                H1.B(B);
                H1.C(sigC1);
                H1.S(sigS1);

                H2.A(sigS1);
                H2.B(Cin);
                H2.C(sigC2);
                H2.S(Sum);

                SC_METHOD (process);
                sensitive << sigC1 << sigC2;
        }

        void process( )
        {
                Cout    = sigC1 | sigC2;
        }
};
```

**VHDL Full Adder Code**

```
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
port(
        A, B, Cin       : in bit;
        Sum, Cout       : out bit);
end module;

architecture arch of full_adder is

signal sigC1, sigC2, sigS1 : bit;

component half_adder
port(
        A, B    : in bit;
        C, S    : out bit );
end component;

begin
H1: half_adder port map
{
        A       => A;
        B       => B
        C       => sigC1;
        S       => sigS1};
H2: half_adder port map
{
        A       => sigS1;
        B       => Cin
        C       => sigC2;
        S       => Sum};

process(sigC1, sigC2)
begin
        Cout    <= sigC1 or sigC2;
end process;
end arch;
```

## 2.2 DDS FUNCTION GENERATOR

Periodic waveforms are necessary for nearly all electronic applications. In order to produce sine, triangular, ramp and square wave outputs, the programmable periodic waveform generator ICs are used. The well-known periodical waveform is sine wave that can be generated by different design approaches such as PLL and DDS. With advances in digital technology DDS is now replacing PLL in professional applications and DDS becomes the main part of the function generators (waveform generators) due to the rapid development of the VLSI technology in the last years. The function generator design presented in this study is based on the DDS technique. In this section, the basic information about DDS and DDS function generators will be presented. In order to communicate with external world and receive the waveform configuration data, the I2C and UART serial communication modules are used as sub-modules of the designed function generator. The basic information about these communication standards is also given in this section.

The time-varying digital signals are generated using the DDS technique. This technique allows generating sine waves at very precise frequencies. In order to construct analog waveform, digital to analog conversion is performed but the analog conversion of the digital samples is not an application issue in this study.

DDS technique makes arbitrary periodical waveform generation possible as well as a sine wave generation. If the arbitrary periodical waveform sample values are loaded into the internal look-up table module in the DDS, the arbitrary periodical waveform with desired frequency and phase can be generated.

In the market, various DDS based function generator designs are present. However the basic design approach is similar to each other. The basic DDS function generator block diagram is illustrated in Figure 2-1. The presented block diagram combines DDS, communication interface and some internal logic in order to make externally configurable periodical waveform generation possible.

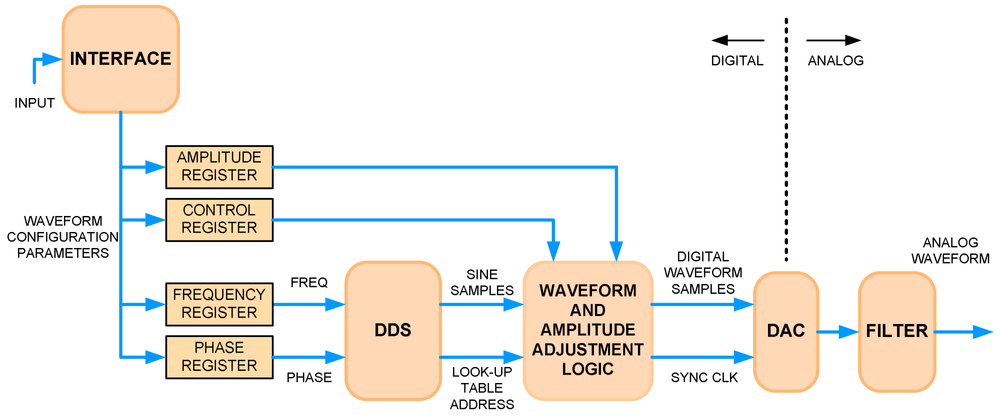


**Figure 2-1 Basic DDS Function Generator Block Diagram**

The interface modules of the function generators receive the waveform configuration parameters from outside environment in parallel or serial format according to the chosen communication standard. Then they load data into the function generator's waveform configuration registers which hold the waveform construction parameters. Various communication standards are utilized by DDS chip manufacturers. For instance, most of Analog Devices DDS ICs have Serial Peripheral Interface (SPI) which is a communication protocol used primarily for synchronous serial communication of host processor and peripherals. The clock frequencies of SPI devices can go up to some MHz and more. This is a sufficient communication speed for DDS applications. Some of Analog Devices' DDS ICs support only parallel bus interface for reception of configuration data. In the last years, more serial bus systems are preferred instead of a parallel bus, because of the simpler wiring advantage. Additionally, serial buses are becoming more common as improved technology enables them to transfer data at higher speeds. Serial transmission is a better option for IC producers also due to its cheap implementation advantage (fewer pins enables cheaper ICs). There are popular serial bus systems like I2C, CAN or USB which proved their effectiveness. In this study, I2C slave receiver and UART receiver modules provide the communication interface of the designed function generator. The background information about these standards is given in sections 2.2.2 and 2.2.3.

The heart of the DDS function generators is DDS module. This module receives the frequency and phase offset information from the configuration registers and produces the sine samples according to the desired settings. The background information about DDS is given in section 2.2.1.

Waveform selection and amplitude adjustments are done using internal logic according to the control and amplitude configuration register values. As illustrated in Figure 2-1, the internal logic receives sine samples, look-up table addressing signal, amplitude register value and control value, and then generates samples of selected waveform and synchronization clock for D/A conversion. The internal logic generates triangular, ramp and square wave using the look-up table addressing signal. The waveform amplitude is digitally adjusted by multiplication. The control register value defines the type of the periodical waveform. It is utilized as a multiplexer control signal to output the desired waveform.

The last part of the function generators is digital to analog conversion and filtering. The generated samples and synchronous clock signal are received by DAC in order to construct a stepwise waveform. The frequency of the synchronous clock determines the analog conversion rate. For function generators, the synchronous clock frequency is equal to the received reference clock frequency. However, the phase of the reference clock and synchronous clock can differentiate 180 degree according to the internal logic and DAC conversion timing. The principal aim is to make a conversion when data is ready at the inputs of DAC. The DAC output spectrum includes aliased images of the output signal. For instance, the lowest frequency of these aliased images is located at $F_{ref} - F_{out}$ (reference clock frequency minus output frequency). In order to suppress these images, low-pass filters are used.

There are various waveform generator and digital frequency synthesizer applications in the market. Some of them are designed as Intellectual Property (IP) and others are designed as integrated chip. The commercial ICs and IPs present in the market are given with their specifications in section 2.3.

## 2.2.1 DIRECT DIGITAL SYNTHESIZERS

The Direct Digital Synthesizer module is the heart of DDS function generators. DDS module produces a digital staircase approximation of a sinusoid in order to construct the sine wave. Generated samples are converted to analog signal and filtered to get pure wave at desired frequency. The staircase approximation of sinusoid is illustrated in Figure 2-2.

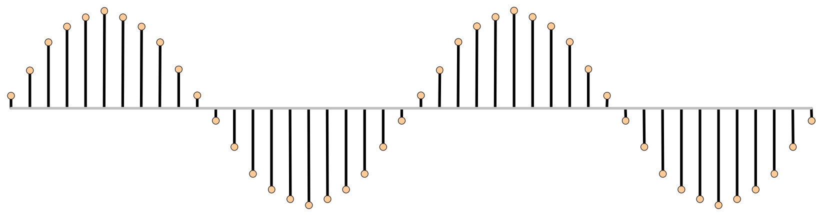


**Figure 2-2 Digital Staircase Approximation of A Sinusoid**

The DDS modules basically constitute of three main blocks. These are Numerically Controlled Oscillator (NCO), Sine Look-up Table and Digital to Analog Converter (DAC). The NCO comprises the increment register and phase accumulator logic. The increment register stores the binary value of frequency control register. The phase accumulator adds the phase increment value to its accumulator output value. The calculated accumulator output is used to address the look-up table which outputs the digital sample values of sine wave at current phase value.

In this study, the analog conversion of digital samples is not the main discussion area. The main discussion is done on the generation of digital samples according to the desired frequency and phase offset values. The basic DDS data flow diagram is illustrated in Figure 2-3. The presented figure does not include digital to analog conversion structures.

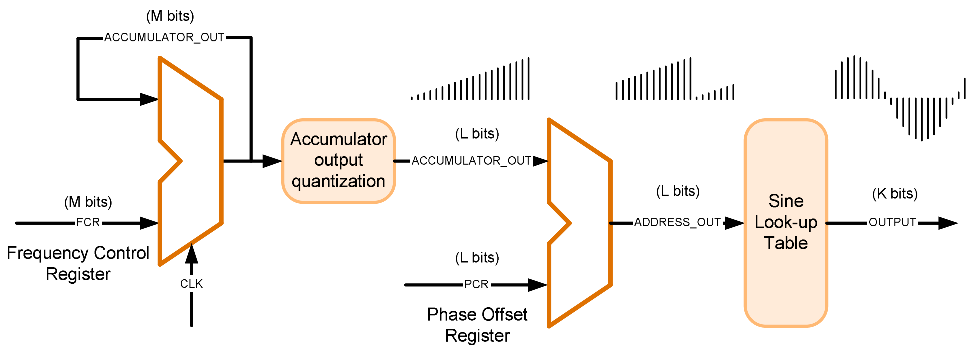


**Figure 2-3    Basic DDS Data Flow Diagram**

At each reference clock cycle, the phase accumulator integrates the phase increment value (frequency control register value) to the phase accumulator output value. The phase increment and accumulator output value are defined by the same number of bits. This number and used reference clock frequency determine the frequency resolution of DDS.

The full precision of the phase accumulator can not be used to index the look-up table due to the very large memory requirement. As a result, the phase accumulator output is quantized by filtering least significant bits. The quantization directly depends on the look-up table length. Nearly all DDS ICs offer phase offset register. The phase offset register value is added to the quantized phase accumulator output in order to make phase tuning possible.

In order to generate a periodical waveform at a constant frequency, a constant phase increment value is added to the phase accumulator at each reference clock cycle. A waveform at higher frequency can be generated if the phase increment value is larger. This situation can be explained as phase accumulator steps the faster through the look-up table. If the phase increment value is smaller, then the phase accumulator steps slower. As a result, a waveform at low frequency is generated.

The frequency of the waveform depends on the reference clock frequency, the phase increment register value and length of phase accumulator. The waveform frequency is calculated using the formula given below.

$$F_{out} = \frac{FCR \times F_{ref}}{2^m}$$

where

| | |
|---|---|
| $F_{out}$ | = DDS output waveform frequency |
| FCR | = phase increment (frequency control register value) |
| $F_{ref}$ | = reference clock frequency |
| m | = phase accumulator word length |

If the desired wave frequency is 500 Hz and the supplied reference clock frequency is 100 MHz, the phase increment value for 32-bit accumulator is calculated as below:

$$FCR = \frac{F_{out} \times 2^m}{F_{ref}} = \frac{500 \times 2^{32}}{100 \times 10^6} = 21475$$

The frequency resolution of the direct digital synthesizer is a function of the reference clock frequency and number of bits employed in phase accumulator. The frequency resolution is calculated using the formula given below:

$$\Delta f = \frac{F_{ref}}{2^m}$$

where Δf is the frequency resolution.

In order to have better frequency resolution, the number of bits employed in the phase accumulators is increased. In practical applications (see section 2.3 and Appendix B) the accumulator sizes are higher than 28 bits for fine frequency resolution. The Analog Devices' AD9959 chip [21] can be clocked up to 500 MHz and it can produce sine output around 200 MHz. This chip has 32-bit accumulator. When this chip is clocked with 500 MHz, the output frequency of this device can be adjusted with 0.12 Hz frequency resolution ( $\approx 500$ MHz / $2^{32}$ ).

There are also two parameters which have an effect on the quality of the waveform. These are the look-up table length and look-up table width. If the value of these parameters is increased, the output waveform resolution becomes better.

The sampling frequency determines the highest frequency that can be produced digitally. The Nyquist Theorem states that the highest frequency which can be generated accurately is less than half of the sampling rate. As a result, the highest frequency that can be generated by the DDS module is $F_{ref}$ / 2. The output frequency is limited with purity concepts in addition to the reference clock frequency. The generated waveform with two samples can not be pure enough, so more samples are required to improve quality.

Spectral Purity Concepts:

The output of the DDS includes the spurious signals. The spurious signal sources can be ordered as below:

1. The reference clock
2. Phase truncation
3. Angle to amplitude conversion
4. Digital to analog conversion

The reference clock is the principal input for DDS. All the signal generation processes are done synchronously with reference clock. The phase accumulator increases the phase accumulator value at each reference clock cycle.

As a result, the spectral characteristics of the reference clock have a direct effect on the output signal quality. The reference clock is also used for digital to analog conversion timing. If the reference clock is noisy, undesired magnitude reduction can be observed at the DAC output.

The phase accumulator must have a sufficient field width to span the desired frequency resolution. In general, a large number of bits are allocated for phase accumulator. Because, more bits for phase accumulator leads to better frequency resolution. However, the look-up table size restriction causes the quantization of the phase accumulator output. In order to reduce the chip die area (to have smaller look-up table), the least significant bits of the phase accumulator output are not used. The phase accumulator quantization level directly depends on the desired length of the look-up table.

The phase accumulator output quantization introduces time base jitter in the output waveform. The quantization and resultant spurious outputs are emphasized in DDS IP product sheets in detail. Xilinx's DDS Compiler V1.0 Product Specification sheet [27] indicates the phase accumulator output quantization for different look-up table examples clearly. It states that the waveform of 12-bit output by using 256 point look-up table results in nearly 48 dB spurious output. The waveform of 16-bit output using 256 point look-up table results nearly 48 dB spurious output again. These outputs lead to the fact that the change in number of output bits has no effect on the spurious output. It shows also that the waveform of 16 bit output by using 1024 point look-up table results 60 dB spurious output. This experience states that the change in look-up table size has direct effect on the spurious output. The look-up table sizes and the resultant spurious outputs are ordered below.

- 256 point look-up table => Approximate Spurious Output = 48 dB
- 512 point look-up table => Approximate Spurious Output = 54 dB
- 1024 point look-up table => Approximate Spurious Output = 60 dB
- 2048 point look-up table => Approximate Spurious Output = 66 dB

Some of the DDS design implements the angle to amplitude conversion using algorithmic techniques. This technique reduces the look-up table sizes and chip die area further. However, the algorithmic approximation results in higher spur levels.

Another spur source is digital to analog conversion. The quantization noise and distortion of DAC leads to spurious signals at the output. These are principally caused by not ideal switching characteristics of DAC. Typically DDS signal error in the output waveform is dominated by the performance of the DAC. The quantization noise is proportional to the DAC resolution. It is formulated as defined below.

$$\mathrm{SNR} = 6.02\mathrm{N} + 1.76\,\mathrm{dB}$$

where N is the DAC output resolution. S. Cheng and J. R. Jenses [19] give the detailed analysis of the spectral purity for DDS applications.

### 2.2.2 INTER-INTEGRATED CIRCUIT (I2C) BUS

In this study, the interfaces of DDS function generator are selected as Inter Integrated Circuit (I2C) Bus [30] and UART. The basic information about I2C bus serial communication standard is presented below.

I2C bus control modules provide an interface between I2C-compatible devices connected by way of the two-wire I2C serial bus. External components attached to the I2C bus serially transmit and/or receive serial data to/from the USART through the 2-wire I2C interface.

I2C bus application example is shown in Figure 2-4. Each I2C device is recognized by a unique address and can operate as either a transmitter or a receiver. A device connected to the I2C bus can be considered as the master or the slave when performing data transfers. A master initiates a data transfer and generates the clock signal SCL. Any device addressed by a master is considered a slave. The I2C bus interface module designed for the DDS function generator supports only slave mode of operation. It only receives the messages which are sent for its address.

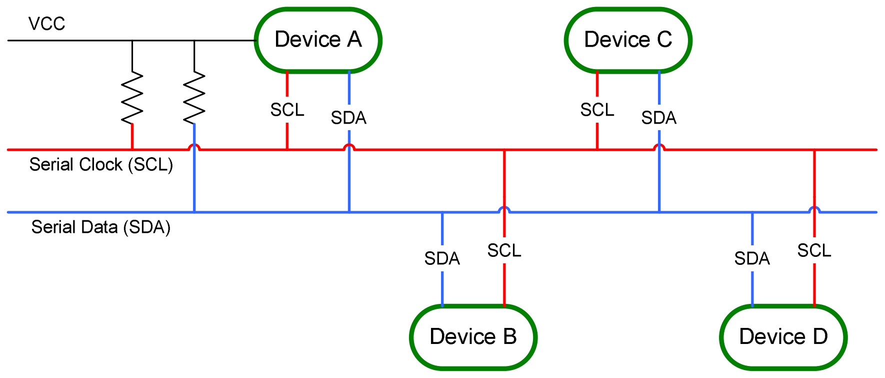


**Figure 2-4 I2C Bus Connection Diagram**

I2C data is communicated using the serial data pin (SDA) and the serial clock pin (SCL). Both SDA and SCL are bidirectional, and they must be connected to a positive supply voltage using a pull-up resistor as shown in Figure 2-4.

The master module starts the communication by pulling SDA pin to low as SCL pin is high. After start of communication, all the slave devices connected to the I2C bus start to wait slave address data. Then the first byte which consists of a 7-bit slave address and the R/W bit is sent by the master module. If one of the slave module addresses is equal to the data sent by master, target slave module makes an acknowledgement by pulling SDA pin to low. Other slave modules wait up to end of communication. The R/W bit determines the data transfer direction. If R/W is equal to 0, then the master transmits data to a slave, else if it is equal to 1, the master receives data from a slave. The most significant bit of the data package is transmitted first by the transmitters. The acknowledge bit (ACK) is sent from the receiver after each byte on the 9th SCL clock.

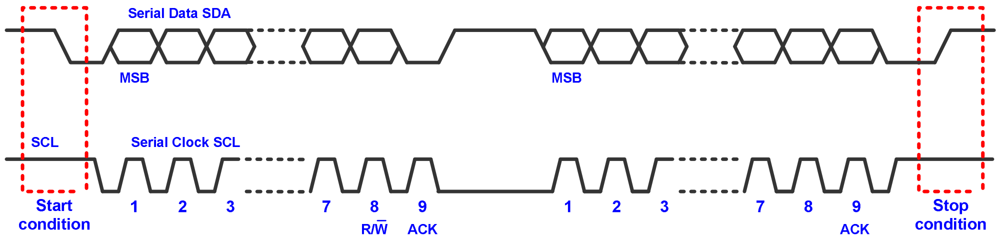


**Figure 2-5 I2C Bus Serial Communication Timing**

I2C bus serial communication timing is illustrated in Figure 2-5. The master device generates one clock pulse for each data bit for transmission. A START and STOP conditions are generated by the master. A START condition is a high-to-low transition on the SDA line while SCL is high. A STOP condition is a low-to-high transition on the SDA line while SCL is high. The bus is busy when START condition exists. The busy situation is removed when STOP condition occurs. Data on SDA must be stable during the high period of SCL as shown in Figure 2-5. The high and low state of SDA can only change when SCL is low, otherwise START or STOP conditions are generated.

### 2.2.3 UART

There are mainly two forms of serial transmission. These are UART (Universal Asynchronous Receiver/Transmitter) and USART (Universal Synchronous-Asynchronous Receiver/Transmitter). The synchronous serial transmission requires that receiver and transmitter modules share clock signal. The transmitter sends the data and clock signal so that the receiver knows the time to read the data. On the other hand, asynchronous serial transmission does not require the clock signal. In this type of transmission, the transmitter and receiver agree on timing by adding some useful bits to the data pack.

The UART controllers are main components for serial communication systems. The UART receiver takes the bits serially and packs them in byte format.

The UART transmitter takes bytes of data and transmits the bits in a sequential format. The word transmission always starts with the “START” bit. This bit informs the receiver that a word will be sent to it. Then the receiver tries to become synchronous with the transmitter and waits for a word. After the synchronization, the data bits of a word are received serially. Commonly, the least significant bit of the word is received first. After sending all the data, the transmitter can send “PARITY” bit for receiver to check correctness of reception. The transmission stops with sending “STOP” bit to the receiver. The timing of the RS232 serial communication is illustrated in Figure 2-6.

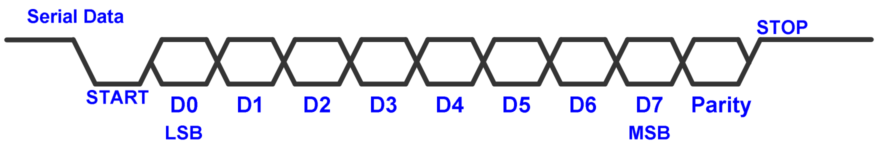


**Figure 2-6 RS232 Serial Communication Timing**

In most computer systems, the UART is connected to circuitry that generates signals that comply with the EIA RS232 specification. In this standard, the “START” bit is logic low and “STOP” bit is logic high. RS232 compatible devices usually transmit the least-significant bit first, immediately after the “START” bit. The most significant bit is transmitted last, followed by an optional parity bit.

## 2.3 COMMERCIAL DDS ICS AND IP MODULES

There are various commercial DDS ICs and IPs in the market. These designs differ from each other according to their capabilities. In Table 2-3, some commercial DDS and NCO ICs are listed. The waveform frequencies generated by listed commercial DDS and NCO ICs vary between 25 MHz and 1GHz. Most of the chips have 32-bit frequency register which is sufficient to adjust frequency with mHz frequency resolution. The ICs' phase offset registers differ in length between 2 and 16 bits. Most of the ICs can not control the amplitude of the waveform digitally. In the given list of ICs, only Analog Devices' AD9959 has the amplitude control capability. The output resolution of the ICs varies between 10 and 16 bits.

Analog Devices AD9833 is served as programmable function generator integrated chip into the market [22]. This programmable function generator is a fully integrated DDS chip. It is also a simple waveform generator capable of producing sine, triangular and square wave outputs. The waveform amplitude adjustment with this chip is not possible. The chip is configured via a 3-wire SPI serial interface. Analog Devices AD9959 consists of four DDS core that provides independent frequency, phase and amplitude controls [21]. In this chip, amplitude adjustment is done by multiplication of look-up table output. FSK, PSK and ASK modulation can be performed by applying source data to the input pin. The carrier is sine wave for these modulations.

IPs are used as the ready to use design modules for FPGA applications. The commercial DDS and NCO IPs of Xilinx, Altera and Lattice Semiconductor are listed in Table 2-4. IP modules allow configuring the length of the frequency adjustment register up to 32 bits. The phase offset of the waveform can be adjusted also using phase offset registers whose length can be configurable up to 32 bits also. IP modules do not allow adjustment of waveform amplitude.

Xilinx has introduced some DDS IPs into the market. Xilinx Logicore Numerically Controlled Oscillator V1.0.3 [26] is designed to generate digital samples of the sine wave. This design presents only a phase increment value input for frequency adjustment. Xilinx Logicore DDS Compiler V1.0 [27] combines various properties. It serves independent channels, phase offset definition and various output resolution. All of these properties can be set by the designer before adding design into a project.

**Table 2-3 Commercial DDS and NCO Integrated Circuits**

| IC Manufacturer and Model | Waveforms and Modulation | Reference Clock (MHz) | Frequency Resolution (bit) | Phase Offset (bit) | Amplitude Control (bit) | Output Resolution (bit) | Control Interface | Internal DAC | Number of Independent Channel |
|---|---|---|---|---|---|---|---|---|---|
| Analog Devices AD9959 | Sine | 500 | 32 | 14 | 10 | 10 | Serial | Yes | 4 |
| Analog Devices AD9858 | Sine | 1000 | 32 | 14 | No | 10 | Parallel | Yes | 1 |
| Analog Devices AD9833 | Sine Triangular Square | 25 | 28 | 12 | No | 10 | Serial | Yes | 1 |
| Qualcomm Q2240I-3S1 | Sine Arbitrary | 100 | 32 | No | No | 12 (sine) 14 (arb) | Parallel | No | 1 |
| Qualcomm Q2368 | Sine BFSK, BPSK, QPSK, 8-PSK | 135 | 32 | 3 | No | 12 | Serial or Parallel | No | 2 |
| Qualcomm Q2334 | Sine BFSK | 50 | 32 | 3 | No | 12 | Serial | No | 2 |
| Harris HSP45102 | Sine BPSK, QPSK, | 40 | 32 | 2 | No | 12 | Serial | No | 2 |
| Fairchild TMC2340A | Sine | 50 | 32 | 16 | 15 | 16 | Parallel | No | 1 |

**Table 2-4 Commercial DDS and NCO Intellectual Properties**

| Vendor | IP Name | Version | Supported FPGA Families | Frequency Resolution | Phase Offset | Amplitude Control | Output Resolution | Number of Independent Channel |
|---|---|---|---|---|---|---|---|---|
| Xilinx Logicore | NCO | 1.0.3 | Spartan-3, Spartan-3E Virtex-II, Virtex-II Pro Virtex-4, Virtex-5 | configurable 3-30 bits | configurable 3-10 bits | NO | configurable 4-16 bits | 1 |
| Xilinx Logicore | DDS | 5.0 | Spartan-3, Spartan-3E Virtex-II, Virtex-II Pro Virtex-4, Virtex-5 | configurable 3-32 bits | configurable 3-32 bits | NO | configurable 4-32 bits | configurable 1-16 channels |
| Xilinx Logicore | DDS Compiler | 1.0 | Spartan-3, Spartan-3E Virtex-II, Virtex-II Pro Virtex-4, Virtex-5 | configurable 3-32 bits | configurable 3-32 bits | NO | configurable 4-20 bits | configurable 1-16 channels |
| Altera MegaCore | NCO Compiler | 6.1 | Cyclone, Cyclone II HardCopy II HardCopy Stratix, Stratix, Stratix II, Stratix II GX, Stratix III, Stratix GX | configurable up to 32 bits | configurable | NO | configurable | configurable |
| Lattice ispLever Core | NCO | 2.2 | LatticeEC, LatticeECP LatticeECP2, LatticeECP2M LatticeSC, LatticeXP | configurable up to 32 bits | configurable 4-32 bits | configurable 4-32 bits | configurable 4-32 bits | configurable 1-16 channels |

# CHAPTER 3

# DESIGN OF DDS FUNCTION GENERATOR

## 3.1 FOLLOWED SYSTEMC AND VHDL RTL DESIGN FLOWS

In this study, the SystemC and VHDL based hardware design stages are followed separately. The main aim is to compare the SystemC and traditional HDL based design flows in terms of coding, synthesis and implementation performances. The SystemC initially appeared as a modeling language like HDL languages. However the new tools are proposed to directly synthesis of SystemC descriptions like traditional HDL synthesis. Since there is not sufficient number of well developed direct synthesis tools in the market, the SystemC descriptions are translated to VHDL code usually and then VHDL synthesis tools are used. In this study, only some SystemC descriptions are translated to VHDL due to synthesizable code writing restrictions. The advantages and disadvantages of SystemC design flow are well observed by making the same digital design using both SystemC and VHDL. The studied digital design cycles are illustrated in Figure 3-1. The hardware and software tools used at each step of the design are also given in the figure.

The first part of the digital hardware design job is to determine the specifications. The design specifications of the presented DDS function generator are given in section 3.3. The defined specifications contain necessary information about the waveform characteristics, modulation types, communication interfaces and common characteristics. The specification definition is necessary and very critical for development of the design plan. The specifications are determined in such a way that minimum numbers of change will be done after starting to design. All designers know that the change of specification affects the whole design period in a negative way.

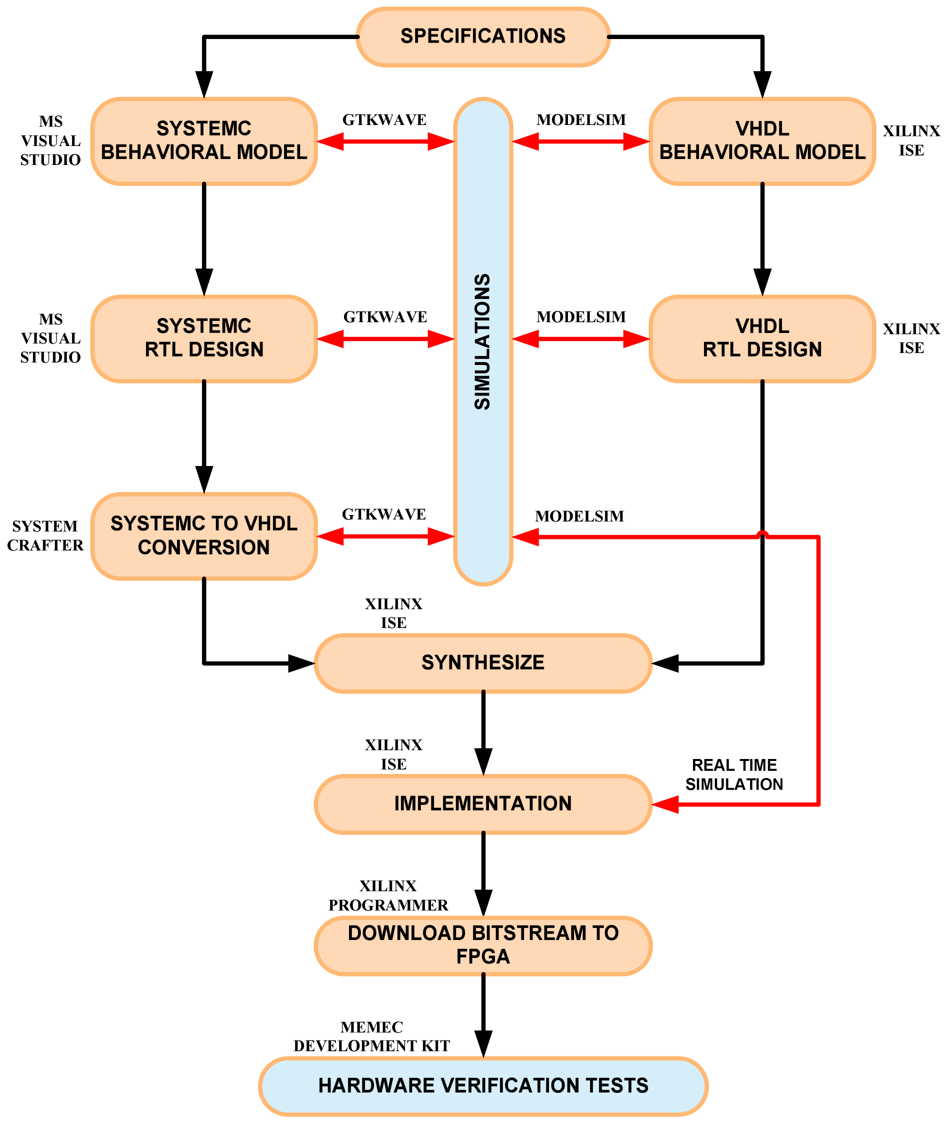


**Figure 3-1 Followed SystemC and VHDL RTL Design Flows**

After determination of the specifications, the behavioral models of the function generator are formed. The goal of the behavioral modeling is to facilitate the synthesis of the digital modules. In many digital design applications, it is not appropriate to describe whole design structure first. In place of difficult structure definition, the designer needs a behavioral description of a design. In this description method, the hierarchy of the modules and their functions are defined using the higher abstraction levels. The aim is to observe the functionality of the design. There is no need to describe the internal structures of the modules. There is an increasing awareness of the need for the behavioral models for specifying digital devices, since the demand for complex and well developed chips increases. This situation leads to present more talent design tools to the market and solve the complexity problem using higher abstraction levels.

In this thesis, the behavioral description of the function generator is done using both SystemC and VHDL. In order to simulate the behavioral models, test bench files are prepared. If the behavioral models are verified with simulations, design cycle passes to RTL block. Direct pass from specification block to RTL block is also possible. The RTL design modules of the function generator are presented in section 3.5. They have been implemented using both SystemC and VHDL. The simulations are done for each design block separately. The design test and verification is presented in Chapter 4.

SystemC based design cycle includes the SystemC to VHDL translation. The number of software that can synthesize the SystemC descriptions directly is insufficient. The companies which design such software do not share it with public. For this reason, the SystemC descriptions should be translated to VHDL code. As a translator tool, SystemCrafter’s SystemC synthesis software is used. The SystemC to hardware flow is presented in section 3.2.

In the VHDL based design cycle, the design modules are designed and verified using Xilinx ISE and ModelSim. The hardware verification tests are applied on the Memec Virtex-4 FX LC development board and Memec P160 Analog Module (Appendix C.1, C.2). The implementation is done for Xilinx’s Virtex4 series V4FX12 FPGA [31] which the development board includes.

## 3.2 SYSTEMC TO HARDWARE FLOW

In SystemC based design flow, the critical edge is to synthesize the generated SystemC descriptions. In the market, various SystemC synthesizers are used. Some of the them synthesize the SystemC descriptions written at behavioral level and others synthesize only the descriptions written at RTL level. Behavioral synthesis allows the designers to create hardware from un-timed high-level models quickly. It also enables them to verify the designs in less time. However, there are disadvantages of using this type of synthesis. Although the designers satisfy with the accurate function descriptions, this synthesis type is not good at allocation of hardware resources and timing.

In this thesis, some of the SystemC RTL descriptions are synthesized. The SystemC and traditional hardware design flows are illustrated in Figure 3-2. The presented figure shows RTL based synthesis but not behavioral synthesis.

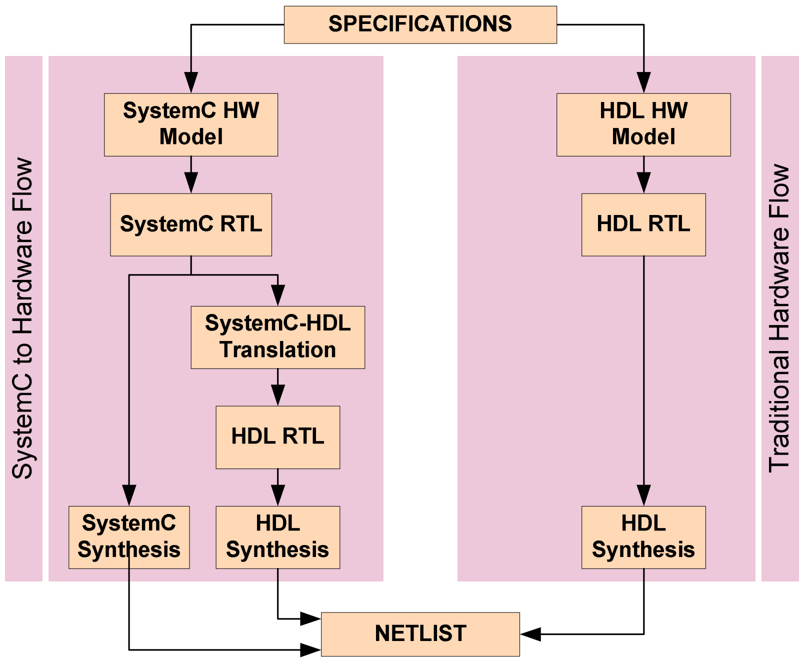


**Figure 3-2 SystemC and Traditional Hardware Design Flows**

The behavioral synthesis does not require additional SystemC model to RTL refinement. As illustrated in Figure 3-2, there are two paths to produce netlist using SystemC to hardware flow. The first path is direct synthesis of the SystemC RTL descriptions. The second path depends on the translation of the SystemC descriptions into HDL descriptions. In this path, the translated design codes are synthesized to netlist using the HDL based synthesis process.

The traditional RTL synthesis provides most powerful synthesis. The direct SystemC synthesis is also powerful but it has unsolved problems related with hardware resource usage and timing. Some tools support this type of synthesis but available tools are not in satisfactory quality and number when compared to HDL synthesis tools. In the future, more companies will make an effort to improve the present SystemC synthesis tools or create well developed tools. The SystemC synthesis with translation brings some disadvantages such as broken of SystemC design cycle and redundant logic generation by SystemC to HDL translator. Using more powerful HDL tools can be an advantage in terms of better synthesis performance. However, the designer usually does not become aware of better synthesis advantage due to generated redundant logic.

In order to translate the SystemC implementation, the designers select either manual recoding in VHDL or automatic translation. The manual recoding increases the probability of errors, either through mistranslation or misinterpretation. A high-performance synthesis tool, SystemCrafter SC, automates this process, by quickly translating SystemC to RTL VHDL.

In this study, SystemC descriptions of the UART baud generator are automatically translated to VHDL using SystemCrafter SC synthesis tool. The SystemCrafter SC presents a code generation and verification platform for digital designers. However it is used only to translate SystemC description in this study. The hardware models are generated and refined to RTL using Microsoft's Visual C++ 6.0. After translation of the SystemC descriptions, the generated VHDL descriptions are synthesized into netlist using Xilinx's synthesis tool.

SystemCrafter SC provides to design, debug and simulate the hardware and systems using existing C++ development environment. It lets to develop hardware and software design code and simulate it in a common platform. The hardware is synthesized to VHDL RTL for implementation using traditional design stages. SystemCrafter SC also writes a structural SystemC description of the synthesized circuit for verification [12].

SystemCrafter's SystemC to hardware flow is illustrated in Figure 3-3. The design flow includes the stages as defined below. After the last stage (VHDL synthesis and netlist generation), VHDL code is implemented for target technology and place and route simulations are done.

1. Developing SystemC hardware model and simulations
2. Translation of SystemC descriptions to VHDL RTL code
3. Gate-Level model simulations
4. VHDL synthesis

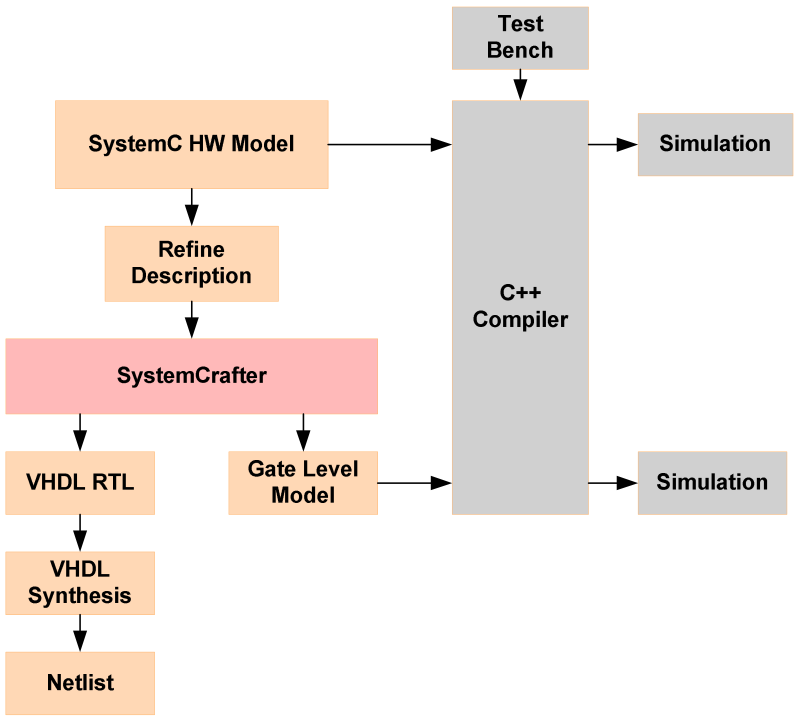


**Figure 3-3 SystemCrafter SC's SystemC to Hardware Flow**

The initial SystemC design descriptions are developed at first stage. The test bench module codes are written to test the generated SystemC descriptions. Then the design is simulated by combining the design and test bench modules in a single environment. SystemCrafter SC uses a C++ compiler such as Microsoft Visual C++ or GNU GCC to simulate the SystemC constructs. If the designer is satisfied with the results, the design is verified and flow passes to second stage. On the other hand, the unsuccessful simulations require the refinement of the design descriptions. The new test bench codes may be rewritten for refined design. This refinement process continues up to achieving successful design verification.

At the second stage, the SystemC descriptions are translated to the VHDL codes. At this stage, the critical point is having synthesizable SystemC descriptions. If the written descriptions are not in synthesizable format, they should be refined into synthesizable format. SystemC language is superset of C++. It comprises all C++ descriptions for simulation and modeling purposes. No synthesis tool can compile all the descriptions written using SystemC. This synthesis limitation is not the unique case for SystemC. The VHDL and Verilog also appeared as a modeling language. The digital designers who use these languages also should pay an attention to write the design descriptions in synthesizable format. For instance, the statements such as “wait for 500 ps” can not be synthesized using VHDL synthesis tools. The synthesis tool can not know desired timing. For this reason, the designer should use a counter and reference clock to wait for such a period of time. SystemCrafter places more restrictions over SystemC synthesis than the traditional design languages. When designing the same module with SystemC and HDL, the designer notices the differences of synthesizable subsets and difficulties in writing synthesizable descriptions using these languages.

At the gate-level model simulation stage, SystemC hardware descriptions are translated to gate-level SystemC models. The VHDL translation stage and gate-level model simulation stages are independent but gate-level simulations give an idea about success of generated VHDL codes. In this stage, the automatically generated gate-level models are simulated. If the designer is satisfied with the simulations of the generated gate-level SystemC models, the gate-level model simulation stage is completed. On the other hand, unsatisfactory simulation results require that the synthesizable descriptions should be refined again.

After running the gate-level model simulation, SystemCrafter SC generates a set of VHDL files. These files contain the synthesizable VHDL codes. The generated files are synthesized using the VHDL synthesis tools such as Xilinx Project Navigator. Before synthesis of the code, SystemCrafter's gate library, craft_gatelibrary.vhd, is added to the project of the synthesis tool as VHDL package. The product of the synthesis process is the gate-level netlist file.

At the final stage, generated gate-level netlist file is implemented for selected FPGA type. The place and route simulations are run after implementation process. If the design is also verified with these simulations, the programming file is prepared and FPGA is configured.

## 3.3 DDS FUNCTION GENERATOR DESIGN SPECIFICATIONS

The function generator module specifications are summarized in Table 3-1 and Table-3-2. Waveform and common characteristics of the function generator is given in Table 3-1. Modulation specifications and interface properties are given in Table 3-2. The design specifications are presented by assuming 100 MHz reference clock is supplied to the function generator module.

**Table 3-1 DDS Function Generator Design Specifications Part-1**

| **WAVEFORM CHARACTERISTICS** | | |
|---|---|---|
| Sine Wave | Frequency Range | 23 mHz - 40 MHz |
| Square Wave | Frequency Range | 23 mHz - 40 MHz |
| Triangular Wave | Frequency Range | 23 mHz - 10 MHz |
| Ramp Wave | Frequency Range | 23 mHz - 10 MHz |
| **COMMON CHARACTERISTICS** | | |
| Number of Output Channel | | 1 |
| Output Resolution | | 12-bit |
| Frequency Control Register (Frequency Register-1) | | 32-bit |
| Frequency Modulation Register (Frequency Register-2) | | 32-bit |
| Frequency Resolution | | 23 mHz |
| Phase Offset Control Register (Phase Register-1) | | 10-bit |
| Phase Modulation Register (Phase Register-2) | | 10-bit |
| Amplitude Adjustment Register | | 8-bit |
| Waveform Selection Register | | 5-bit |
| Look-up Table Holds | | Quarter sine wave |
| Look-up Table Size | | 256 x 12 bit |
| Minimum and Maximum Output Levels | | 0x000 and 0xFFF |
| Approximate Spurious Output | | 60 dB |
| Trigger Delay | | 4 clock cycles |

Table 3-2 DDS Function Generator Design Specifications Part-2

| **MODULATION** | | |
|---|---|---|
| Frequency Shift Keying (FSK) | Carrier Wave | Sinusoidal |
| | Source | External |
| Phase Shift Keying (PSK) | Carrier Wave | Sinusoidal |
| | Source | External |
| **INTERFACES** | | |
| UART | Work Type | Receiver only |
| | Baud Rate Adjustment | External |
| | Supported Baud Rates | 1200-38400 bps |
| | Data Bits | 8 |
| | Stop Bits | 1 |
| | Parity | None |
| | Flow Control | None |
| | Adjustable Function Generator Parameters | Frequency<br>Phase Offset<br>Amplitude<br>Waveform Selection |
| | Soft Reset Control | Yes |
| I2C | Standard Mode Support | Yes |
| | Fast Mode Support | Yes |
| | High Speed Mode Support | No |
| | Work Type | Slave Receiver only |
| | Supported Addressing | 7-bit |
| | Slave Module Address | “1001100” |
| | Adjustable Function Generator Parameters | Frequency<br>Phase Offset<br>Amplitude<br>Waveform Selection |
| | Soft Reset Control | No |

## 3.4 SOFTWARE AND HARDWARE RESOURCES USED IN DESIGN

All of the software tools used throughout the thesis is summarized in Table 3-3 below. Microsoft Visual Studio 6.0 is used as the development environment to design and verify the function generator core using the SystemC. The SystemC library (SystemC 2.0.1 release for this study) is used with Microsoft Visual Studio 6.0 to provide such a development environment. SystemC Win1.0 Beta and GTKWave Wave Analyzer V1.3.19 provide a wave viewer environment to verify the developed SystemC models. SystemCrafter is a synthesis tool for SystemC applications. In this study, it is only used to translate SystemC descriptions to VHDL.

**Table 3-3 Used Software for Design and Verification**

| Tools / Package | Usage |
|---|---|
| SystemC 2.0.1 [2] | SystemC library |
| Microsoft Visual Studio 6.0 [1] | C++ compiler |
| SystemC Win1.0 Beta [2] | SystemC design and verification environment |
| GTKWave Wave Analyzer 1.3.19 [2] | Wave viewer |
| SystemCrafter SC 2.0.0 [3] | SystemC synthesis tool |
| Matlab R2006a [1] | Mathematical computation and analysis tool |
| Xilinx ISE 7.1 [3] | Xilinx integrated synthesis and implementation tool |
| ModelSim XE III 6.0a [3] | Simulation tool |
| Microsoft Visual Studio .NET 2003 [1] | .NET platform |

([1]) Licensed to Aselsan Inc.

([2]) Free version

([3]) Trial version

In order to determine the sinusoidal sample values to generate sine wave, MATLAB R2006a mathematical computation and analysis tool is used. Xilinx Project Navigator ISE 7.1 software is used to write and synthesize VHDL description of the function generator. It is also used for implementation and programming of FPGA. To verify the generated VHDL design description, ModelSim XE III 6.0a is preferred as simulation software.

Microsoft Visual Studio .NET 2003 is used to make user interface software between the computer and the function generator hardware test platform. The function generator configuration parameters (frequency, phase offset, wave type and amplitude information) are sent to the hardware test platform in RS232 format using the control software developed on this platform.

The function generator VHDL description is tested and verified on the hardware test platform. All of the hardware tools used to test the design throughout this study is summarized in Table 3-4.

**Table 3-4 Used Hardware for Verification**

| Hardware | Usage |
|---|---|
| Memec Virtex-4 FX LC | Development kit |
| Memec P160 Analog Module | AD and DA converter module |
| Xilinx Platform Cable USB | Programmer over JTAG port |
| Oscilloscope | Signal voltage viewer |
| SMB-BNC Cable | Connects Memec P160 with oscilloscope |
| Test Computer | User interface software runs on |

Memec Virtex-4 FX LC Development Kit [32] and Memec P160 Analog Module [33] constitute the function generator module’s hardware test platform. Memec’s development kit includes Xilinx Virtex-4 series V4FX12 FPGA. The analog module includes two 12 bit 165 Msps D/A converters. The function generator design is implemented into the FPGA using the Xilinx’s Platform

Cable USB [34] which is a high-speed download cable that configures or programs all Xilinx FPGA, CPLD and ISP Configuration PROM. The specifications of Memec's development boards are given in Appendix C. The hardware test platform setup is presented in Chapter 4.

## 3.5 DDS FUNCTION GENERATOR MODULES

The modular and hierarchical architecture has been chosen for function generator design. Modularity and hierarchy help to simplify and organize a design project. Hierarchy allows the building of a design out of modules which themselves may be built out of (sub-)modules.

The modular architecture is the preferred method for digital systems since it helps to organize the system into logically distinct modules, such that the modules can be changed by enhanced modules. Different implementation alternatives can be examined for the modules in a simulation. Only the corresponding component instantiation needs to be changed for this in the overall model. The modularity helps different modules to work together, and enables easier maintenance and module replacement.

The DDS function generator module designed in this study includes the modules presented below. The names of the modules are given in the hierarchical order. In this study, both SystemC and VHDL design files are prepared for presented modules.

- Function Generator Module
    - DDS Module
        - DDS RAM Module
    - UART Receiver Module
    - UART Baud Generator Module
    - I2C Slave Receiver Module
    - Main Controller Module

### 3.5.1 DDS MODULE DESIGN

In direct digital synthesis based function generator design the direct look-up table method is utilized. This method is a trivial way of obtaining not only sinusoidal samples, but also any periodical function. In this approach, sampled (or calculated) amplitude values of one full period are stored in the memory. The phase output, which is obtained from the phase accumulator, is connected to the address inputs of the look-up table and the output samples are obtained from the data outputs of the memory. The frequency resolution may be increased by adding more address lines to the memory. By increasing the number of data output lines, the output resolution may also be increased. But increasing the number of address or data lines means the increase of the storage capacity of the memory device, which is an undesired situation.

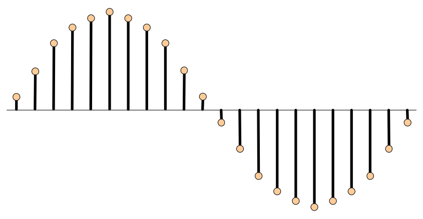


**Figure 3-4 Four Quadrant of Sine Function**

The samples of one full period sine wave are shown in Figure 3-4. In order to generate a sine wave, there is no need to hold all the samples in a memory. Symmetry properties of the sine can be used to reduce memory size. The quarter sine wave is enough to represent the full period sine wave as shown in Figure 3-5. Thus, the memory requirement is reduced to 1/4. Additionally, the maximum amplitude value of the quarter wave is half of the full wave. This property also results in one bit reduction for the memory width. As a result, the total memory reduces to 1/8. In this study, the quarter wave samples of a sine wave are loaded into the DDS RAM module. However, one bit amplitude reduction is not applied. The DDS RAM module holds a look-up table 256-word in length and 12-bit in width.

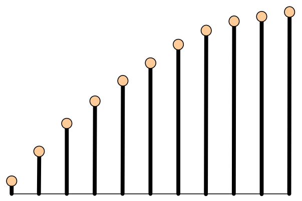


**Figure 3-5 First Quadrant Samples of Sine Wave**

The data flow diagram of designed DDS module is shown in Figure 3-6. The frequency control register and phase accumulator output are in 32-bit length. The designed DDS module can be clocked up to 125 MHz when it is implemented using Xilinx Virtex4 FPGA technology and it can produce a sine wave around 40 MHz. When this chip is clocked with 100 MHz reference clock, the output frequency can be adjusted with 0.023 Hz ( = 100MHz / $2^{32}$ ) resolution.

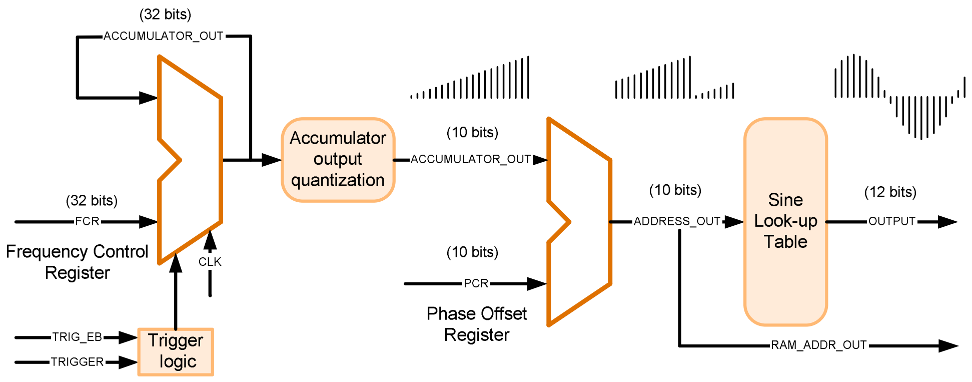


**Figure 3-6 Data Flow Diagram of Designed DDS Module**

As shown in Figure 3-6, DDS module contains trigger logic block in addition to the basic DDS. This logic block provides a trigger capability to the function generator design. When the trigger enable signal is set to logic high, the trigger signal is expected to start waveform generation. In order to make ramp, triangular or square wave generation possible, DDS module sends the look-up table addressing value to the output port.

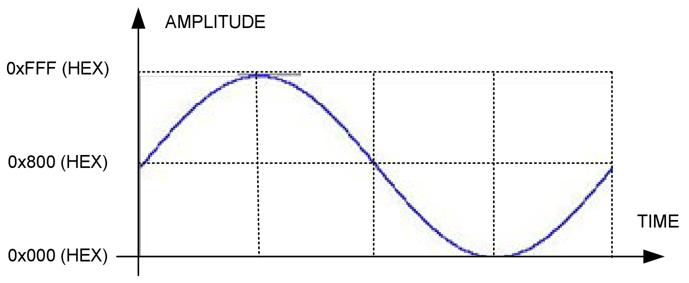


**Figure 3-7 Sine Wave Construction**

The designed DDS uses 1024 points to construct waveforms so the approximate spurious output becomes 60 dB as discussed in section 2.2. The maximum and minimum output values that can be generated by designed DDS module are shown in Figure 3-7.

### 3.5.2 I2C SLAVE RECEIVER MODULE DESIGN

The designed I2C slave receiver module has the following features:

- Slave receiver operational mode
- Compliance to the Philips Semiconductor I2C specification v2.1
- 7-bit device addressing mode
- Fast mode up to 400 kbps support
- Byte format transfer

The I2C slave receiver module provides an interface between the function generator and the I2C bus. The master device connected to the I2C bus has exclusive control over it. In order to control the function generator in this design, the master device transmits the waveform configuration parameters to the designed I2C bus slave receiver module. The parameters include the type, frequency, phase offset and amplitude information of periodical waveform.

To configure the function generator, first, the master device transmits the defined specific slave address value. Then it transmits the function generator configuration register sub-address value. As last package, the configuration data is transmitted to the slave receiver module. After each data package reception, the slave module gives acknowledgement. I2C bus configuration parameter reception format is shown in Figure 3-8.

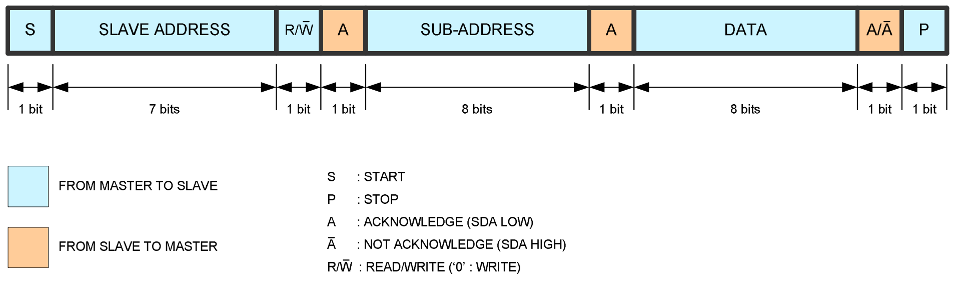


**Figure 3-8 I2C Bus Parameter Reception Format**

The sub-address data package points the configuration register which will hold the received data. The I2C register addresses and the data definitions are summarized in Table 3-5. The I2C slave receiver module waits for 14 bytes of configuration data. The I2C bus master device does not need to transmit each byte with slave address and sub-address. It can transmit all the data after the transmitting sub-address byte and this allows loading the configuration parameters to the function generator core module more quickly.

**Table 3-5 I2C Sub-Address Definitions**

| I2C Sub-Address | Data Definition |
|---|---|
| 1 | Wave Amplitude |
| 2 | Frequency Word-1 (bits 31-24) |
| 3 | Frequency Word-1 (bits 23-16) |
| 4 | Frequency Word-1 (bits 15-8) |
| 5 | Frequency Word-1 (bits 7-0) |
| 6 | Frequency Word-2 (bits 31-24) |
| 7 | Frequency Word-2 (bits 23-16) |
| 8 | Frequency Word-2 (bits 15-8) |
| 9 | Frequency Word-2 (bits 7-0) |
| 10 | “000000” + Phase Offset Word-1 (bits 9-8) |
| 11 | Phase Offset Word-1 (bits 7-0) |
| 12 | “000000” + Phase Offset Word-2 (bits 9-8) |
| 13 | Phase Offset Word-2 (bits 7-0) |
| 14 | “000” + Work Mode (bits 4-0) |

### 3.5.3 UART RECEIVER MODULE DESIGN

The designed UART receiver and baud generator modules have the following features:

- 8-bit data with non-parity and one stop bit
- Receive shift register
- Receive buffer register
- LSB-first data receive
- Receiver start-edge detection
- Programmable baud rate between 1200 Hz and 38400 Hz
- Data ready flag

The UART receiver and baud generator modules provide an interface between the function generator and an external system via one serial data pin, uart_sin. Using this input pin, the serial data in RS232 format is received by UART receiver module. When the start bit comes to the receiver module, it stimulates the baud generator module. The clock signal synchronized with input serial data is generated by the baud generator module. The receiver module receives the serial data at each positive edge of this synchronized clock signal and puts the bytes into the receive shift register. The connections of the UART receiver modules and signal directions are illustrated in Figure 3-9.

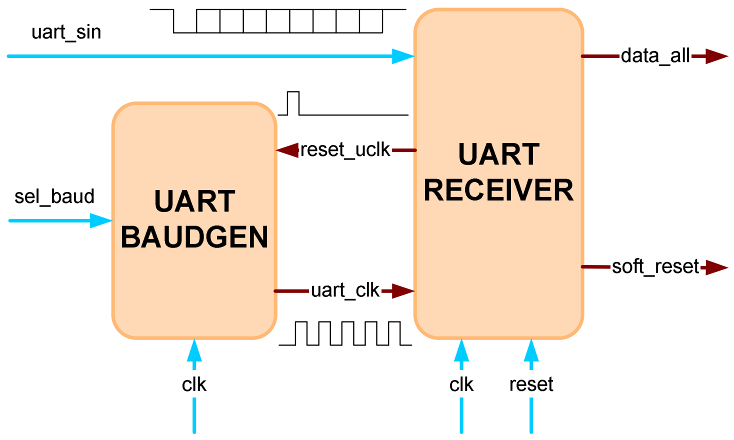


**Figure 3-9 UART Receiver Modules' Data Flow Diagram**

In UART mode, UART receiver module receives characters at a bit rate asynchronous to the transmitter device. For this reason, the designed receiver module has a start-edge detection capability. To receive the data, first, the receiver module transmits the trigger signal (reset_uclk) to the baud generator. Then, this module generates the synchronous clock signal at a selected baud rate.

The RS232 data transmission format, shown in Figure 3-10, consists of a start bit, eight data bits and one stop bit. It does not include parity bit. The synchronized clock makes a low-to-high transition at each center point of the serial data bit. This provides a clear data reception by the receiver module. The baud generator accomplishes this by receiving input baud rate signal (sel_baud). The baud generator module does not produce clock signal for start and stop bits.

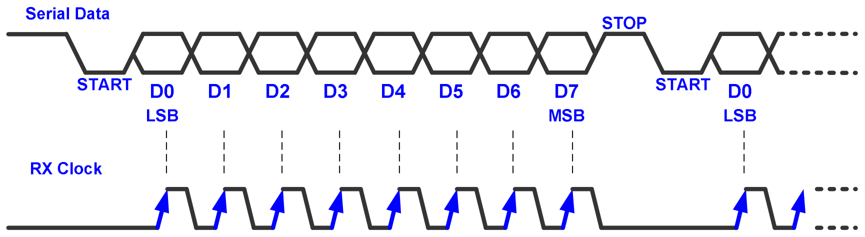


**Figure 3-10 RS232 Data and Clock Synchronization**

The UART receiver module is used to receive the waveform configuration parameters from an external system. The parameters include the type, frequency, phase offset and amplitude information of the waveform. The parameter data packages must be transmitted to the receiver module in a fixed format. The configuration parameter reception order of the designed receiver module is shown in Figure 3-11. Each configuration data package starts with configuration header (decimal 90) and ends with function generator's work mode. The configuration data package also includes one amplitude word, two frequency control words each of which has 32-bit length (4 bytes) and two phase offset words each of which has 10-bit length (represented by 2 bytes). The receiver module waits for 14 data bytes after receiving the configuration header byte.

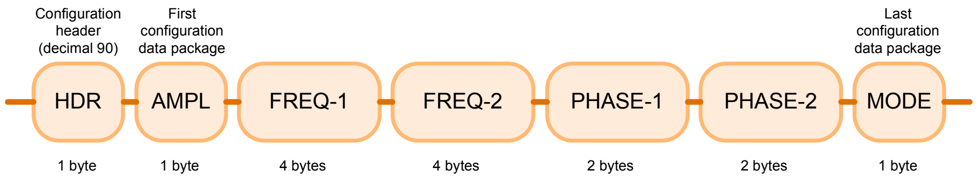

Configuration header (decimal 90)
First configuration data package
Last configuration data package
HDR
AMPL
FREQ-1
FREQ-2
PHASE-1
PHASE-2
MODE
1 byte
1 byte
4 bytes
4 bytes
2 bytes
2 bytes
1 byte

**Figure 3-11 Configuration Parameter Reception Order**

The waveform configuration parameters are hold in the shift register. The shift register data definitions are given in the Table 3-6. When all the configuration data is received, the Data Ready Flag bit (bit-113) is set to logic high and the main controller module receives the shift register data.

**Table 3-6 UART Shift Register Data Definitions**

| UART Shift Register Bit Number | Data Definition |
|---|---|
| 113 | Data Ready Flag |
| 112-104 | Wave Amplitude |
| 103-96 | Frequency Word-1 (bits 31-24) |
| 95-88 | Frequency Word-1 (bits 23-16) |
| 87-80 | Frequency Word-1 (bits 15-8) |
| 79-72 | Frequency Word-1 (bits 7-0) |
| 71-64 | Frequency Word-2 (bits 31-24) |
| 63-56 | Frequency Word-2 (bits 23-16) |
| 55-48 | Frequency Word-2 (bits 15-8) |
| 47-40 | Frequency Word-2 (bits 7-0) |
| 39-32 | "000000" + Phase Offset Word-1 (bits 9-8) |
| 31-24 | Phase Offset Word-1 (bits 7-0) |
| 23-16 | "000000" + Phase Offset Word-2 (bits 9-8) |
| 15-8 | Phase Offset Word-2 (bits 7-0) |
| 7-0 | "000" + Work Mode (bits 4-0) |

The UART baud generator module supports six different baud rate selections. The UART clock signals are generated from input reference clock (100 MHz) and their frequencies can be selected between 1200 Hz and 38400 Hz. The user must define the serial communication speed before start transmitting the configuration data. In order to do this, the user applies the baud rate selection signal to 4-bit length baud rate selection input of the function generator. Then this value is loaded into the baud rate register. The baud rate register value and related synchronized clock frequencies are summarized in Table 3-7.

**Table 3-7   UART Baud Rate Selection Table**

| Baud Rate Register Value | Generated Clock Frequency |
|---|---|
| “0110” | 38400 Hz |
| “0101” | 19200 Hz |
| “0100” | 9600 Hz |
| “0011” | 4800 Hz |
| “0010” | 2400 Hz |
| “0001” | 1200 Hz |

### 3.5.4 MAIN CONTROLLER MODULE DESIGN

The main controller module is a link between the communication modules and DDS module. It controls the I2C interface with transmitting module address and receiving the sub-address and data packages. The data packages hold the frequency, phase offset, type and amplitude information of a desired waveform. In the main controller module, the data packages are distinguished according to the sub-addresses received from I2C receiver module and written to the target configuration registers.

The main controller module has also an interface with the UART receiver module. The data came to the UART receiver module has been hold in 113-bit shift register. The main controller module receives the shift register value and writes the shift register’s data into configuration registers when data ready flag bit is set to ‘1’. In addition, FSK and PSK modulations are also controlled by this module. It switches the frequency and phase offset control values sent to the

DDS module according to the received modulation source input. The carrier wave for FSK and PSK is sine wave. In this module, there are two 32-bit frequency registers, two 10-bit phase registers, one 8-bit amplitude register and one 5-bit work mode register.

### 3.5.5 FUNCTION GENERATOR MODULE DESIGN

The function generator design module does not include only the sub-modules, but it also includes the internal processes to generate desired waveform. The internal processes in this module are illustrated in Figure 3-12.

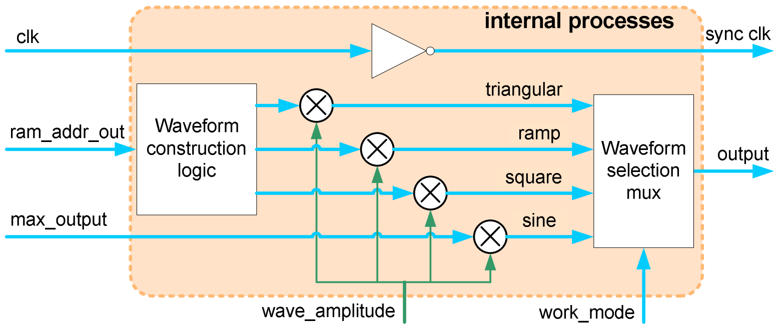


**Figure 3-12 Function Generator Module Internal Processes**

All of the properties of the function generator can not be implemented in the sub-modules. DDS module only generates the sine samples. However, the function generator must also build square, triangular and ramp waves. The duties of the internal processes can be ordered as below:

- Waveform selection
- The synchronization clock generation for DAC
- Square, triangular and ramp wave sample generation
- Output sample amplitude adjustment

**Table 3-8 Waveform Property Selection Table**

| Work Mode | Selected Waveform | Selected Frequency Register | Selected Phase Register |
|---|---|---|---|
| “00000” | Sine | 1 | 1 |
| “00001” | Sine | 1 | 2 |
| “00010” | Sine | 2 | 1 |
| “00011” | Sine | 2 | 2 |
| “00100” | Square | 1 | 1 |
| “00101” | Square | 1 | 2 |
| “00110” | Square | 2 | 1 |
| “00111” | Square | 2 | 2 |
| “01000” | Triangular | 1 | 1 |
| “01001” | Triangular | 1 | 2 |
| “01010” | Triangular | 2 | 1 |
| “01011” | Triangular | 2 | 2 |
| “01100” | Ramp | 1 | 1 |
| “01101” | Ramp | 1 | 2 |
| “01110” | Ramp | 2 | 1 |
| “01111” | Ramp | 2 | 2 |
| “00110” | Sine (FSK) | 1 & 2 | 1 |
| “00111” | Sine (PSK) | 1 | 1 & 2 |

The designed function generator has a capability of generating many standard waveforms with excellent frequency resolution. These waveforms can be ordered as sine, square, triangular and ramp waves. The function generator has also a capability of making frequency and phase shift keying. The Table 3-8 defines which frequency and phase register will be used for selected work mode.

In order to convert digital samples to analog signals, a DAC requires a reference clock as timing source in addition to sample data. The function generator produces samples at each rising edge of input clock. Since the analog conversion must be implemented when the sample data is ready, internal processes send the inverse of the reference clock to DAC.

The look-up table counter signal is used to generate square, triangular and ramp waves by internal processes. The address counter steps through the look-up table and completes its one cycle at desired period time. This periodic counter is used to build defined waveforms with a little modification.

The designed function generator can also make digital amplitude adjustment. The sample outputs of DDS are multiplied with the loaded amplitude constant in the internal processes.

## 3.6 INTEGRATION OF DESIGN MODULES

The function generator design file merges the main controller module with the interface units (I2C and UART modules) and DDS modules. The modularity architecture of the function generator lets to add new module such as a DDS RAM module that holds the samples of the arbitrary waveform. In addition, the interface modules can be replaced with desired modules without spending more effort.

The block diagram of the function generator is shown in Figure 3-13. The interface modules supply the configuration data of the waveform that will be generated. DDS module generates the samples of the sine wave. The duty of the main controller module is to become a bridge between the interface and DDS modules. It receives the data from external world using the I2C slave module and UART receiver modules. It sends the frequency and phase offset values to the DDS module. It also has duties related with modulation. The module combination is implemented in the function generator design file. All the designed modules are defined as sub-modules of the function generator. In order to connect the sub-modules with each other, the internal signals are also defined in the design file.

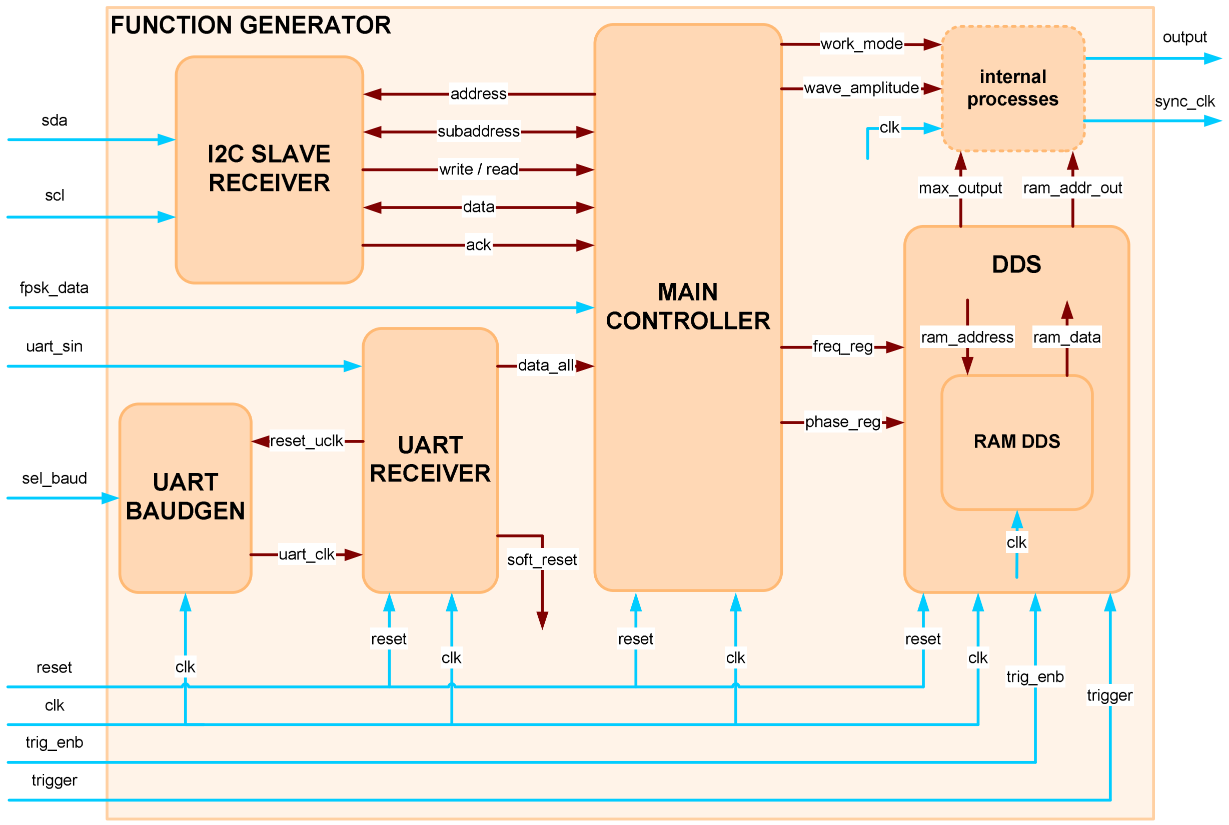


Figure 3-13  Function Generator Block Diagram

# CHAPTER 4

# IMPLEMENTATION OF DDS FUNCTION GENERATOR

## 4.1 TEST AND VERIFICATION METHODOLOGY

Before verification of the design on the hardware platforms, the function generator is verified in software environments. In order to verify design, the test bench-design approach is used. The test bench generates input signals to the design module and receives the output signals from it. The received outputs are compared with the expected results. The prepared test bench file does not need to become in a synthesizable form. The sample design module and its test environment are illustrated in Figure 4-1.

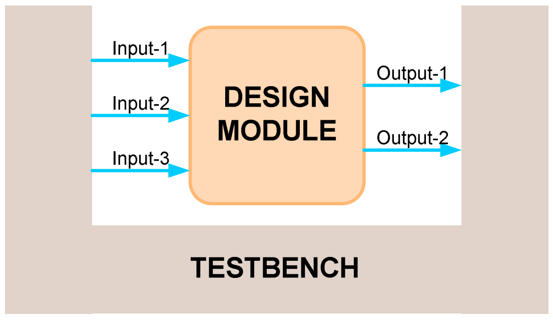


**Figure 4-1 Test Environment for Design Module**

The test bench module generates Input-1, Input-2 and Input-3 signals for the design module and receives Output-1 and Output-2 signals. The generated and received signals are observed in the simulation environment. If the received signals are expected signals, the design module is verified and can be used as a sub-module of the bigger designs or alone.

In this thesis, the design of the function generator is made using two different design languages named by SystemC and VHDL. The test environments for these two languages are different. As a result, the design descriptions written using SystemC and VHDL are simulated in different test environments.

Microsoft Visual C++ 6.0 is used to prepare design and test files in SystemC based hardware design. SystemC based design verification flow diagram is illustrated in Figure 4-2. In order to test the design module, the test bench file and main file are prepared. The main file contains test bench and design modules as its sub-modules. The main file is compiled and if there is no compilation error, the design project is built. The design environment generates an executable file. The user runs this executable file. Executable file generates the VCD file that holds the samples of simulated waveforms. GTKWave Wave Analyzer V1.3.19 software is used to open VCD file in order to observe simulation waveform.

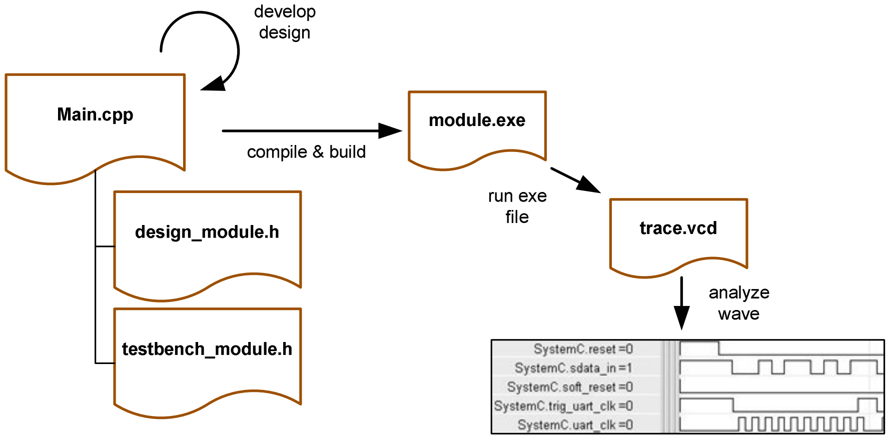


**Figure 4-2 Verification Flow for SystemC Based Designs**

In Xilinx Project Navigator, the verification of the HDL design module is done differently. In this program, the test bench module contains the design module as its sub-module (component) so there is no need for main file additionally. The generated test signals in the test bench module are given to the design module's input ports. The output ports of the design module are connected to the test signals. All the defined signals in the test bench file can be observed in the simulation environment. In this thesis, ModelSim XE III 6.0a is used as a HDL based simulation and debug environment. ModelSim is initialized from a single user interface on the Xilinx Project Navigator. This improves the productivity and facilitates the verification. The verification flow for HDL based designs is illustrated in Figure 4-3.

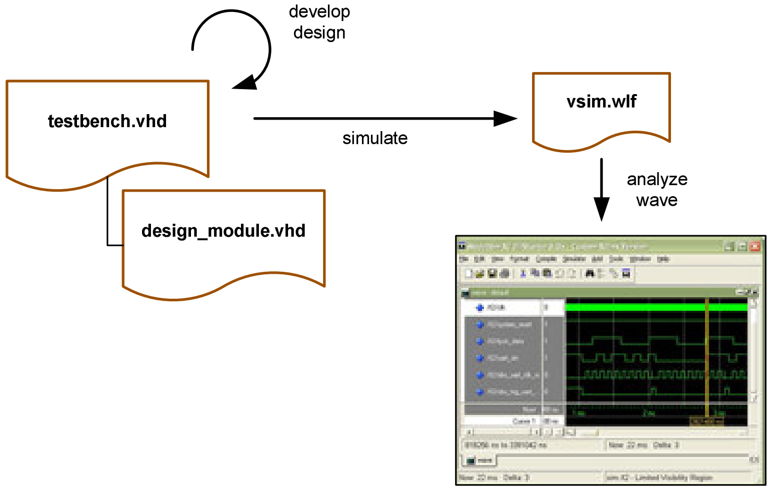


**Figure 4-3 Verification Flow for HDL Based Designs**

ModelSim is a comprehensive HDL simulation environment. It verifies the HDL source code by allowing the behavioral and timing model investigation of the digital designs. ModelSim is also a debugging environment and has a full language support for VHDL, Verilog, SystemVerilog and SystemC. Since the trial version of the ModelSim is used in this study, license restrictions do not let to simulate SystemC design codes. ModelSim is a very powerful simulation environment, and it is sometimes difficult to use this simulation environment. However Xilinx design environments can take care of launching ModelSim to simulate projects. When the designer launches ModelSim using Xilinx design environments, the wave window appears without any additional work. The wave window is the most important window of ModelSim. It contains waveforms for all input and output signals of the top-level design module. There are also dataflow window which can be used to observe the internal signals of top-level module. This window is very helpful during the design period. Because, the designer can observe all the internal signals and notice the design errors of sub-modules. There is no need to simulate all the sub-modules using separate test bench files, if applied test to top-level module is enough to verify whole design. In this study, this talented dataflow window of the ModelSim is used and the designed top-level module is simulated only using only two test bench files.

ModelSim verification flow is illustrated in Figure 4-4. The VHDL code is compiled into a VHDL library before it can be simulated. The simulator can not read VHDL source code directly. It only simulates a compiled database. In the compilation phase, compiler may point out some possible design or syntax errors. After compiling the design successfully, actual mistakes (design mistakes) can be observed in simulation environment.

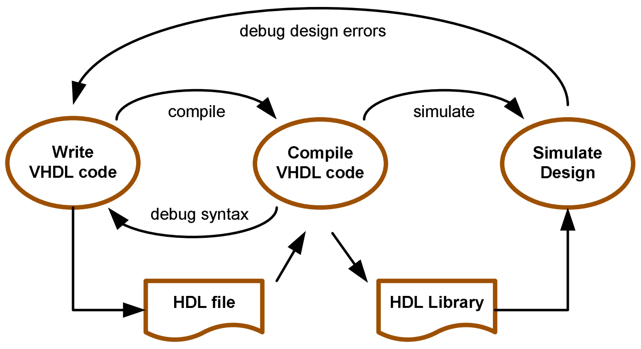


**Figure 4-4 ModelSim Verification Flow**

## 4.2 DESIGN IMPLEMENTATION USING SYSTEMC

### 4.2.1 DDS WAVEFORM GENERATION TEST

The DDS design module and its connections with the test bench module are illustrated in Figure 4-5. The RAM module which contains the sinusoidal samples is a sub-module of DDS module. In this section, the RAM module is not verified alone. The simulation results of DDS verify the functionality of the internal RAM module.

The DDS module has two output ports. One of them (ram_addr_out) transmits the RAM address value which is initially same as the defined phase register value. After the DDS starts to operate, the RAM address value is incremented by the defined frequency register value. The second output port (max_output) is used to send the sinusoidal samples generated by DDS. In order to verify DDS module, the frequency and phase values are sent from the test bench module. The trigger and trig_enb signals are also supplied to detect the trigger performance of the designed module. It is expected that the DDS module generates the sinusoidal samples at the rising edge of the trigger signals when trig_enb signal is logic high. It is also expected that the frequency and phase values are taken into account during the sample generation.

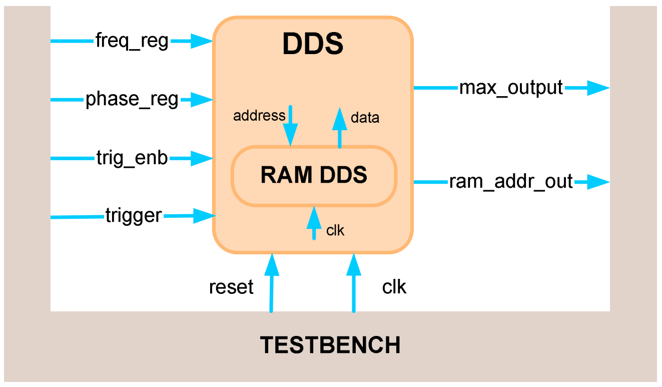


**Figure 4-5 DDS Module Test Environment**

The simulation waveform of the DDS module designed using SystemC is given in Figure 4-6. At the start of the simulation, reset condition is employed by logic high pulse for reset port. After a time, reset condition is removed. Since the trigger is enabled at the start of simulation, the sinusoidal sample generation does not start instantaneously. When the trigger signal is sent from the test bench module, the DDS launches to send the sinusoidal samples. There is maximum 4 clock cycles delay between trigger's rising edge and start of sample generation.

Since the phase register value is used as an offset value for addressing the RAM (look-up table), the initial value of the RAM address is equal to phase register value (0x333 in hexadecimal representation). The delay between the RAM addressing and sample generation is 2 clock cycles.

The second reset condition is employed in the presented simulation. At this condition, the RAM address output value is 0x000 and sinusoidal sample output value is 0x800. This output value represents a minimum analog voltage level. After a reset condition is removed, the sample generation starts instantaneously. Since the trig_enb is logic low (trigger is not enabled), the module does not look for a trigger signal to initialize the waveform generation.

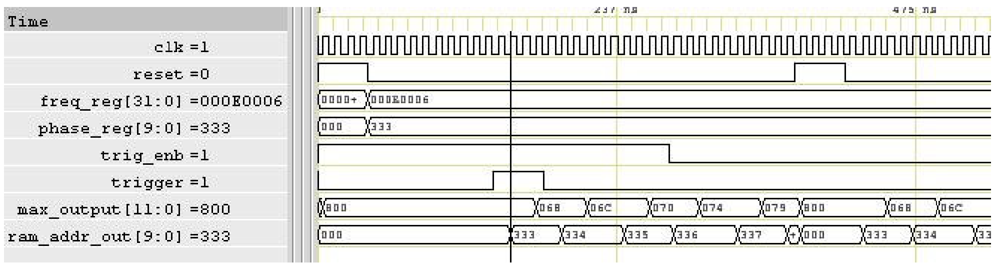


**Figure 4-6 DDS Operation**

In the presented simulation waveform, the samples are repeated for more than 3 clock cycles. At each clock cycle the addressed sinusoidal wave samples are sent from the DDS module. If the value of the frequency register is incremented, the number of repeated samples decreases.

### 4.2.2 I2C BUS COMMUNICATION TEST

I2C slave design module, I2C master test module and their connections with the test bench module are illustrated in Figure 4-7. I2C master module is used to generate desired I2C bus signals in order to test and verify the I2C slave receiver design module. The master module is not the sub-module of the function generator. It is controlled by the test bench module to transmit desired test data package in I2C bus standard format.

The slave receiver module receives sda and scl signals from the master module and generates the parallel data package. This package is received by the test bench module and compared with the test data sent by master module.

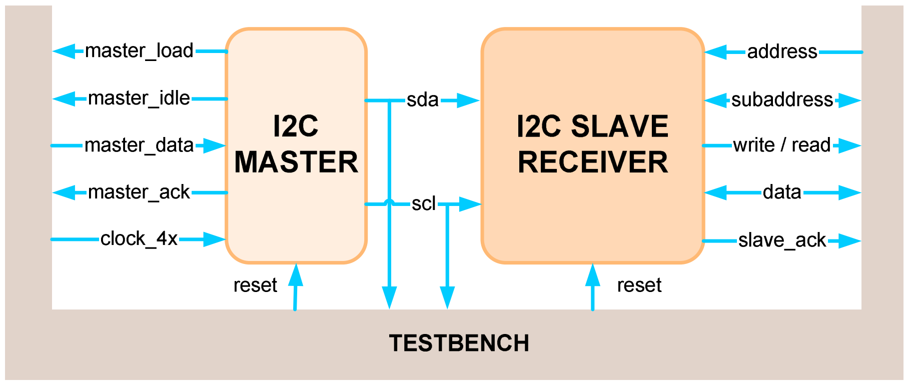


**Figure 4-7 I2C Slave Receiver Module Test Environment**

The simulation waveform of the master module design using SystemC is given in Figure 4-8. At the start of the simulation, the reset condition is employed by logic high pulse for reset port. At this condition, the master module generates idle signal (port is not used). After removing the reset condition, the master module makes its lookload signal logic high which a data package request from the test bench. The master module starts to send serial data by making high-to-low transition of scl when sda is low. In the presented figure, this condition is shown at the time of cursor. After the start condition, the data bits are sent serially (most significant bit is sent first) with the synchronization clock pulse.

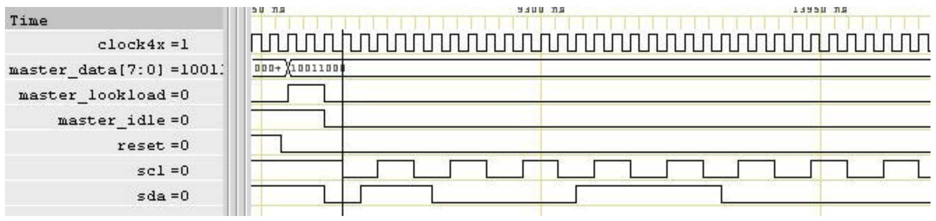


**Figure 4-8    I2C Bus Data and Clock Synchronization**

The simulation waveform of the I2C slave receiver module design using SystemC is given in Figure 4-9. The master module sends the address (0x98), sub-address (0x01) and data packages (from 0x02 to 0x1F) to the slave receiver module serially. If the received address signal is equal to the internal address register value, then the slave receiver module accepts the serial data. In this simulation, the address is verified first and then the sub-address and data package values are received. At the time of cursor, the first data package is sent by master module for being written to a register at sub-address 0x01. The sub-address value is incremented after each data. The I2C data reception ends with stop condition.

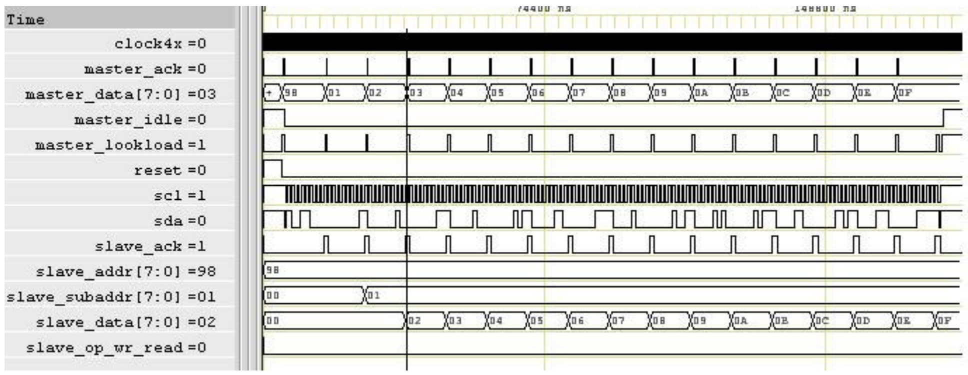


**Figure 4-9    Configuration Data Reception over I2C Bus**

### 4.2.3 UART COMMUNICATION TEST

#### UART Receiver Module Test

In order to test the UART receiver module, there is a need for a UART transmitter module. Because the observation of the receiver module performance would be possible only if the transmitter module sends the test data and it is compared with the data received by the receiver module. The UART receiver module and its connections with the test bench are illustrated in Figure 4-10. The test bench module contains the UART transmitter module.

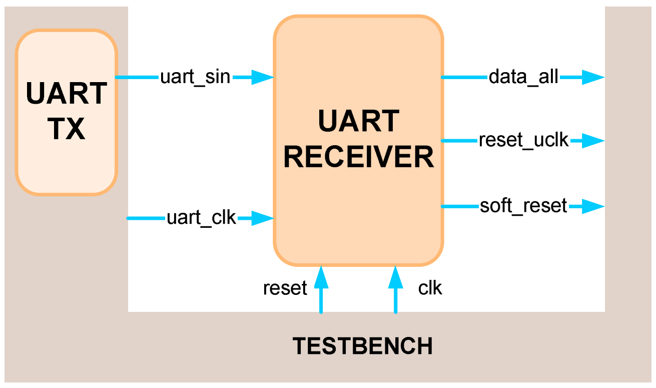


**Figure 4-10 UART Receiver Module Test Environment**

The data and clock synchronization simulation waveform of the UART receiver module is given in Figure 4-11. When the start bit of serial data arrives to the receiver module, it makes reset_uclk signal logic low. This is an initialization signal for baud generator module to start operation. The baud generator module starts to generate the synchronization clock after high-to-low transition of reset_uclk signal. The receiver makes reset_uclk signal logic high after each stop bits. This start and stop bit detection operation provides a perfect serial data reception by making serial data and clock synchronization. At each rising edge of clock, the receiver accepts the serial data value. At the reset condition (external hard reset), reset_uclk signal is made logic high by the receiver module and the synchronization clock pulses are not generated.

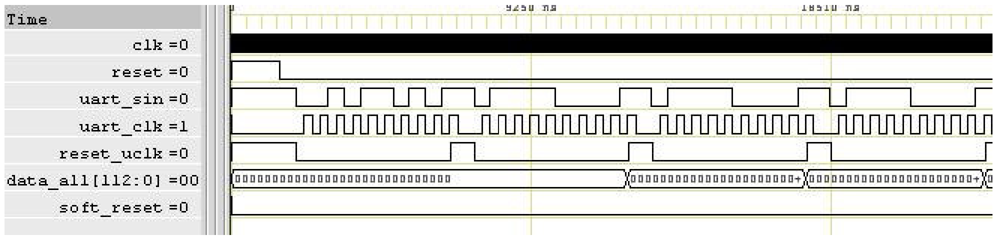


**Figure 4-11 UART Data and Clock Synchronization**

The configuration data reception simulation waveform of UART receiver module is given in Figure 4-12. The test bench transmits the configuration data serially to the receiver module as shown in the figure. The transmitted first byte is configuration header (decimal 90). After receiving the configuration header, the receiver module waits for the configuration data to load registers. The shift register signal (data_all) holds 113 bits data. After the receiver accepts 14 bytes of configuration data which contains frequency, phase offset, amplitude and waveform selection information, the configuration data reception is completed and the most significant bit of shift register signal (data all) is made logic high. The most significant bit is a warning for controller module and means that the data load has been completed and the new parameters can be used.

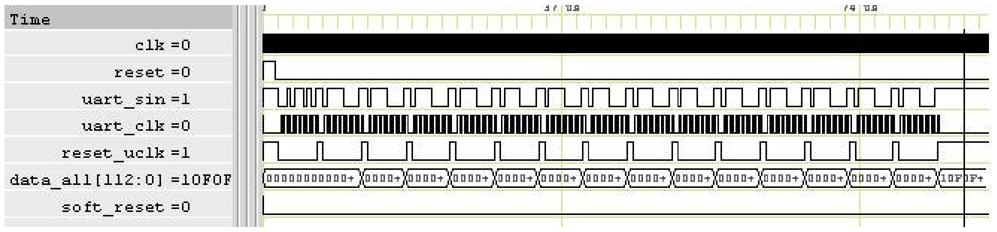


**Figure 4-12 Configuration Data Reception over UART**

## UART Baud Generator Module Test

The UART baud generator module and its connections with the test bench module are illustrated in Figure 4-13. The baud generator module generates the synchronization clock for data reception. The module receives three signals which are baud rate selection, clock initialization and system clock signal, and then transmits synchronization clock signal from its output port.

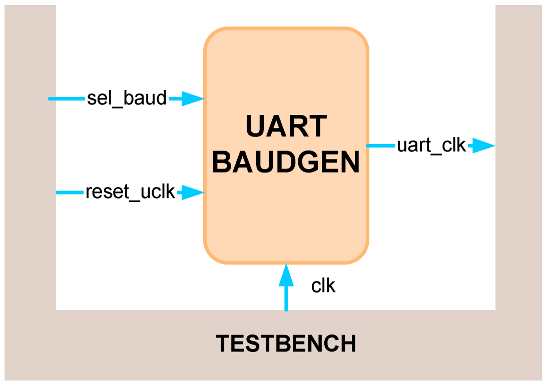


**Figure 4-13 UART Baud Generator Module Test Environment**

The synchronized clock generation of baud generator module is given in Figure 4-14. The baud rate selection signal determines the frequency of generated synchronization clock signal. The baud rate selection signal (sel_baud) is 04 initially as shown in the figure. This means that 9600 Hz baud rate is required. The module calculates and generates the desired synchronization UART clock using the system clock. As shown in the figure again, the sel_baud is made 05. This means that 19200 Hz baud rate is required and the module also generates this clock. The reset signal is supplied to the baud generator module from the receiver module and helps to receive serial data bit at its middle point. When the reset condition (reset_uclk is logic high), is present, the internal counters take initial values. After removing this condition the clock generation starts again.

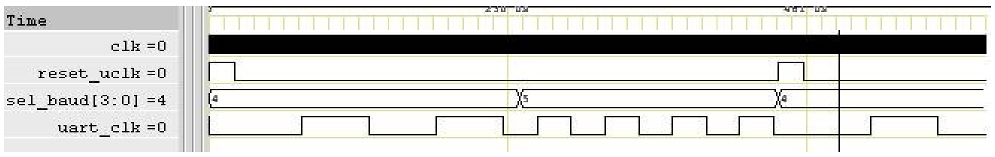


**Figure 4-14 UART Synchronization Clock Signal Generation**

### 4.2.4 MAIN CONTROLLER OPERATION TEST

The main controller module is a bridge between the interface modules and DDS module. Main controller module and its connections with the test bench module are illustrated in Figure 4-15. In this simulation, the test bench acts as if it is the UART receiver module and sends the configuration data signal (data_all) to the controller. The dotted lines represent unused signals at this simulation.

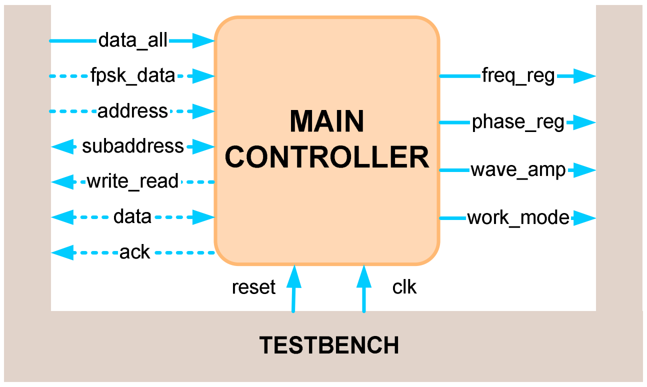
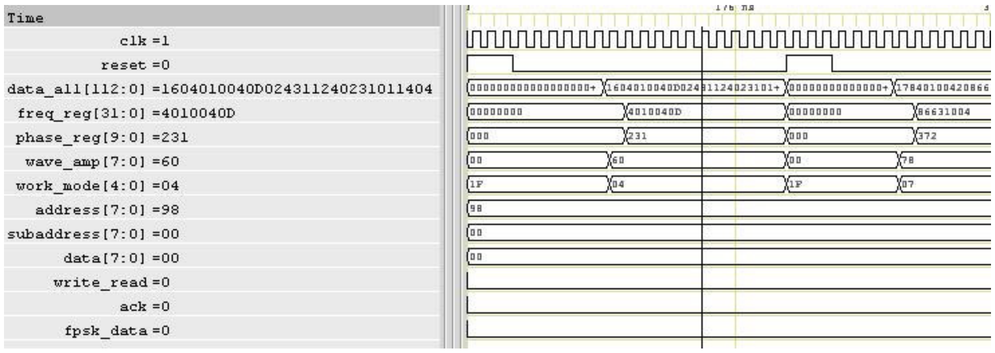


**Figure 4-15 Main Controller Module Test Environment**

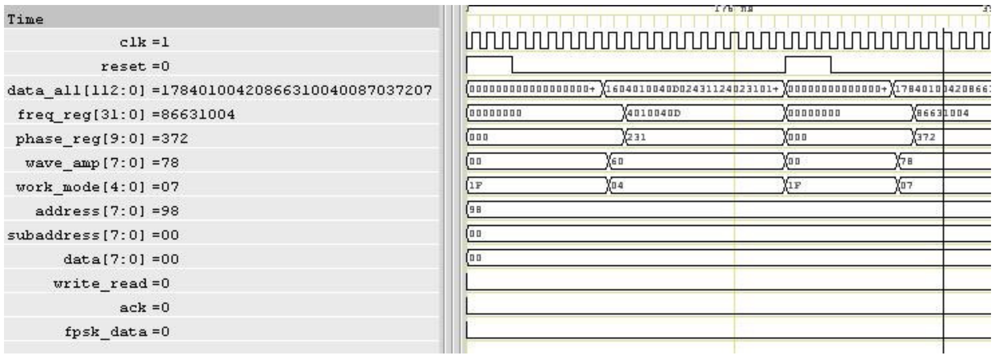
In order to verify the controller operations, the UART data sample is sent to the controller module by test bench. The mission of the controller module is to receive the configuration data from the interface modules and place the values of frequency, phase offset, amplitude and waveform selection data to the configuration registers. In Figure 4-16, the controller receives the UART data sample value (hex 1-60-4010040D-02431124-0231-0114-04) at the time of cursor. The waveform selection mode is 04 (means that square wave, frequency-reg1 and phase-reg1 are selected). The controller really outputs the frequency-reg1 (0x4010040D), phase-reg1 (0x231) with correct amplitude (0x60) values.



```
Time
                clk =1
              reset =0
data_all[112:0] =1604010040D024311240231011404
 freq_reg[31:0] =4010040D
 phase_reg[9:0] =231
  wave_amp[7:0] =60
 work_mode[4:0] =04
   address[7:0] =98
subaddress[7:0] =00
      data[7:0] =00
         write_read =0
                ack =0
          fpsk_data =0
```

**Figure 4-16  Main Controller Operation (Part-1)**

In Figure 4-17, the controller receives the UART data sample value (hex 1-78-40100420-86631004-0087-0372-07) at the time of cursor. The waveform selection mode is 07 (means that square wave, frequency-reg2 and phase-reg2 are selected). The controller really outputs the frequency-reg2 (0x86631004), phase-reg2 (0x372) with correct amplitude (0x 78) and work mode (0x07) values.



```
Time
                clk =1
              reset =0
data_all[112:0] =17840100420866310040087037207
 freq_reg[31:0] =86631004
 phase_reg[9:0] =372
  wave_amp[7:0] =78
 work_mode[4:0] =07
   address[7:0] =98
subaddress[7:0] =00
      data[7:0] =00
         write_read =0
                ack =0
          fpsk_data =0
```

**Figure 4-17  Main Controller Operation (Part-2)**

### 4.2.5 VERIFICATION OF THE WHOLE DESIGN

In this section, DDS function generator's sub-modules are supposed to be tested and verified. The applied tests cover only the verification of internal processes of function generator and whether the signals between sub-modules are truly connected or not. All of the signals and variables that are used in the design are not observed. As a result, if there are unexpected output values, the sub-modules that constitute the function generator must be tested and verified in their test environments again.

The block diagram of the function generator and its test environment are illustrated in Figure 4-18. The configuration test signals are generated by the UART transmitter module and the I2C master module. These modules simulate the external interfaces of the design. Using these simulated interfaces, the selected function generator configuration parameters are loaded into the design. The generated output waveform sample values are observed and compared with the expected results.

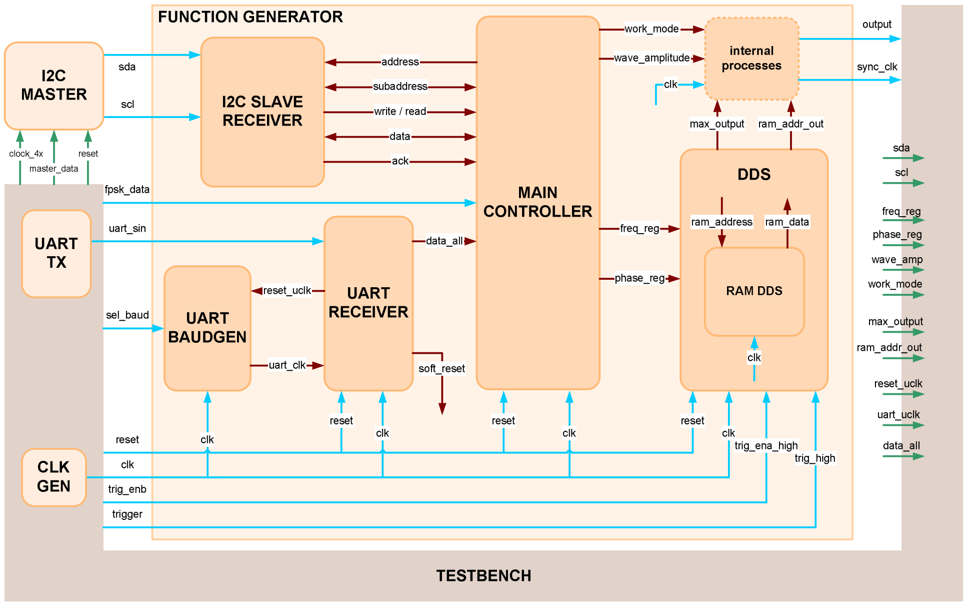


**Figure 4-18 Function Generator Test Environment**

The sample generation simulation of the function generator after loading the configuration data over I2C interface is shown in Figure 4-19. The configuration data is transmitted to the function generator design using I2C bus master module. The frequency, phase, amplitude and waveform selection signals of function generator are set according to the received sub-address value. The sample generation detail simulation is shown in Figure 4-20. The output waveform generation starts when trigger signal makes low-to-high transition. The RAM address signal's initial value (obs_ram_addr_out) is equal to the phase offset value (obs_phase_reg). The internal look-up table outputs the addressed sinusoidal samples (max_output) at each clock cycle.

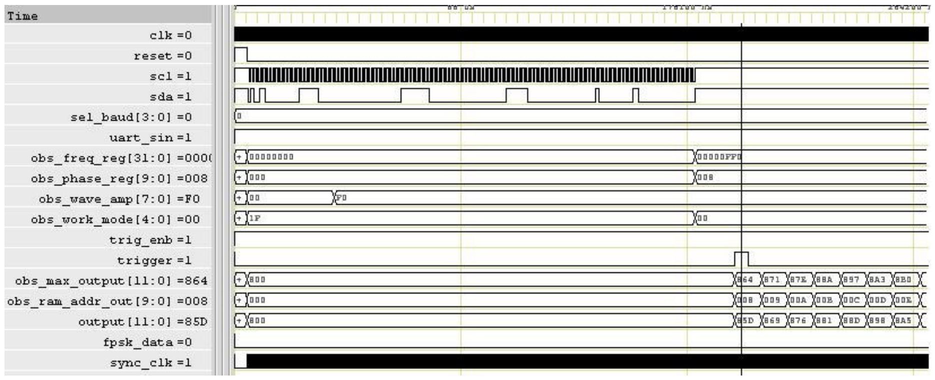


**Figure 4-19 Wave Generation Using I2C Interface**

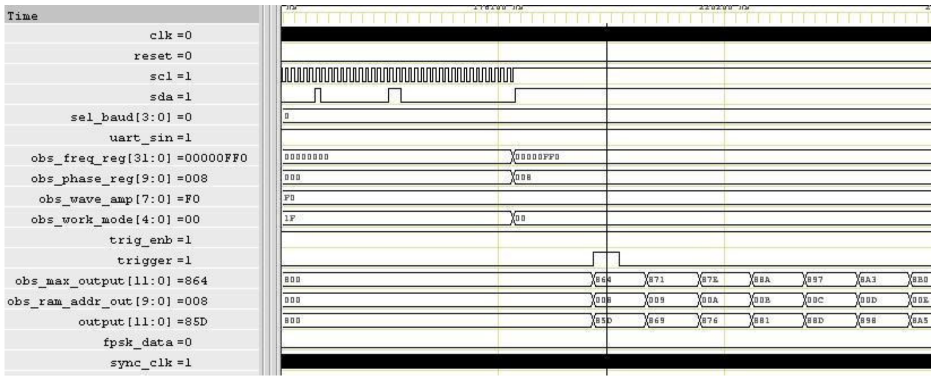


**Figure 4-20 Wave Generation Using I2C Interface in Detail**

The sample generation simulation of the function generator after loading the configuration data over UART interface is shown in Figure 4-21. The function generator receives the UART data serially and sets its configuration signals after data loading is completed. The sinusoidal sample values are calculated according to the desired amplitude after trigger signal arrives. As shown in Figure 4-22, the output value (0x85D) of the function generator is result of the multiplication of max_output (0x864) and amplitude value (0xF0) and division it by the maximum amplitude value (0xFF).

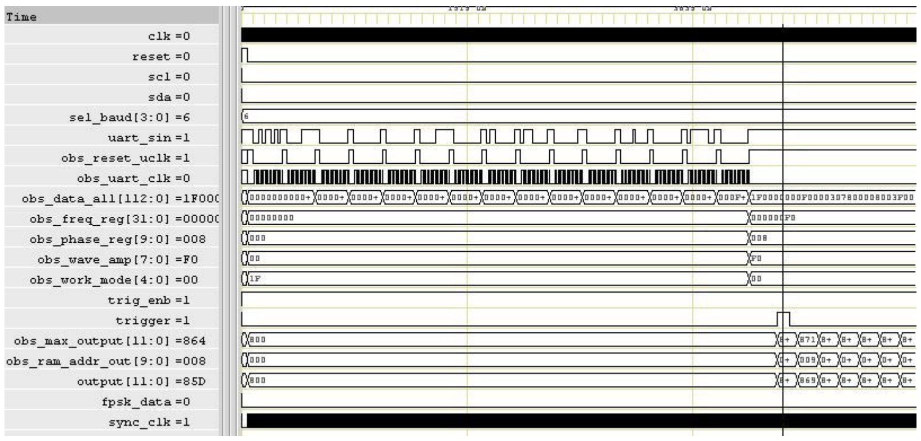


**Figure 4-21 Wave Generation Using UART Interface**

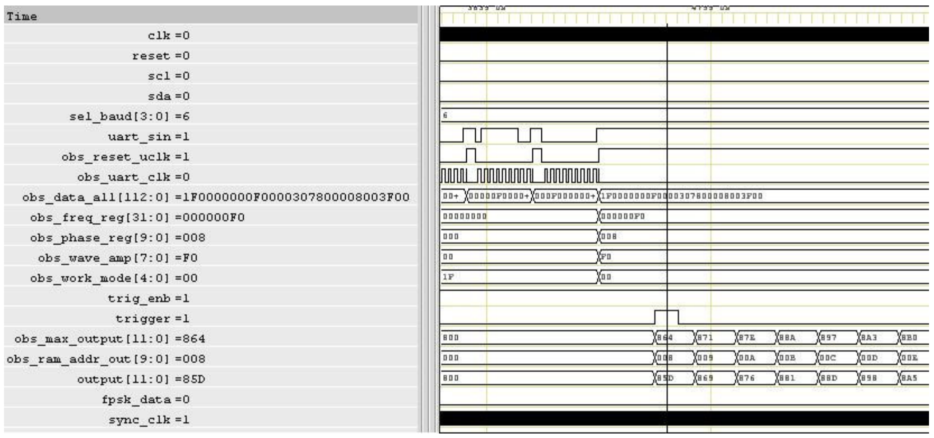


**Figure 4-22 Wave Generation Using UART Interface in Detail**

## 4.3 DESIGN IMPLEMENTATION USING VHDL

### 4.3.1 VERIFICATION IN SOFTWARE ENVIRONMENTS

The designed function generator has a capability of generating many standard waveforms with excellent frequency resolution. These waveforms can be ordered as sine, square, triangular and ramp waves. In addition, FSK and PSK modulations can be done using external data source and carrier sine wave. The function generator has two standard communication interfaces to receive operational data. This section includes verification results of the synthesized function generator VHDL descriptions in software environment.

In order to verify the function generator's VHDL design codes, two different test bench files were prepared. One of them was prepared to observe all output waveforms if I2C interface is utilized. It defines the function generator and I2C master module which is used to generate standard I2C signals as its sub-modules. It controls I2C master module to generated configuration data and receives the output signals from function generator. The second one was prepared to observe waveforms if UART interface is utilized. This test bench module supplies standard UART signals to the function generator core.

The configuration data is loaded into the function generator module using I2C interface as shown in Figure 4-23. The I2C master module transmits data and clock signals using its serial data port SDA and serial clock port SCL. The main controller module analyzes the received data, and then it refreshes its frequency, phase offset, amplitude and waveform selection register data. The configuration parameter values are refreshed after each data reception. The waveform sample generation starts when trigger signal makes low-to-high transition. The sine and square wave generation is shown in Figure 4-23 and Figure 4-24. As the new configuration values are loaded, the frequency of the waveform changes according to new values. The analog literal is selected for RAM addressing and output signals to observe the waveforms clearly.

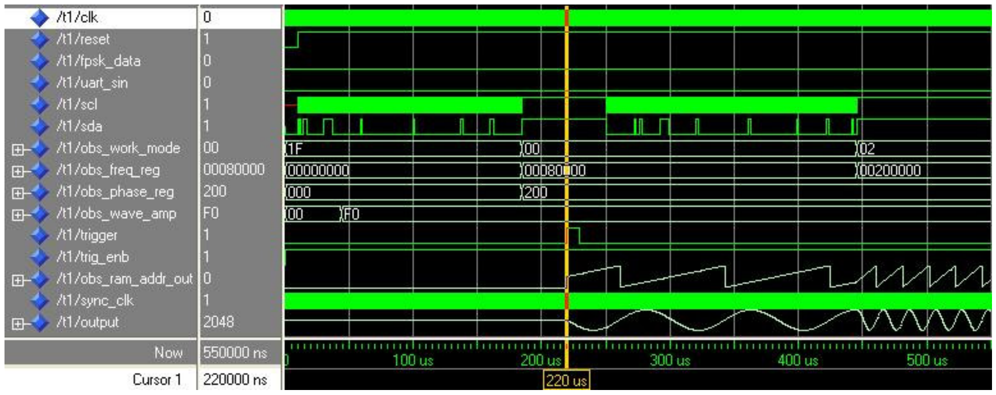


**Figure 4-23 Sine Wave Generation Simulation (Using I2C Bus)**

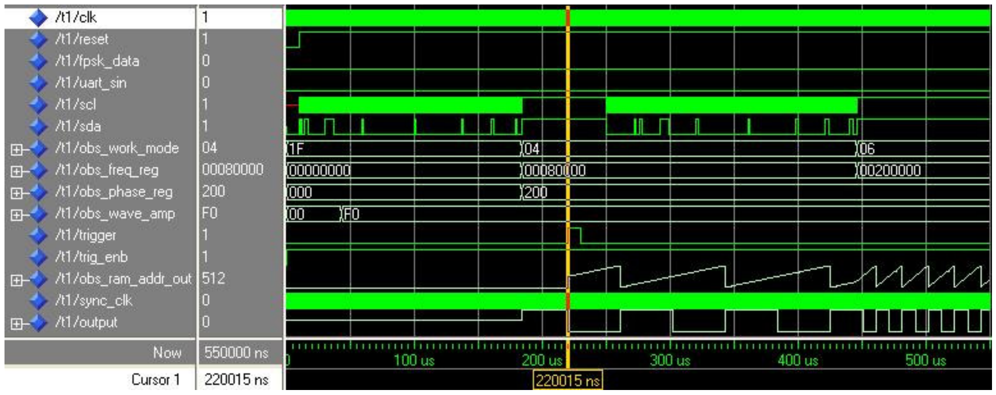


**Figure 4-24 Square Wave Generation Simulation (Using I2C Bus)**

The triangular wave generation is shown in Figure 4-25. In this simulation, the trigger is disabled by making trig_enb signal logic low. As a result, the function generator does not wait for trigger pulse. The waveform sample generation starts immediately after the receiving the configuration data. The ramp wave generation is shown in Figure 4-26. The look-up addressing signal is directly given to the output port in order to generate a ramp wave.

The configuration parameters are loaded two times for triangular and ramp wave simulation. At the second loading, the phase register value is changed in addition to the frequency value. As can be seen in the presented simulation waveforms, the output signal makes a glitch due to this phase offset change.

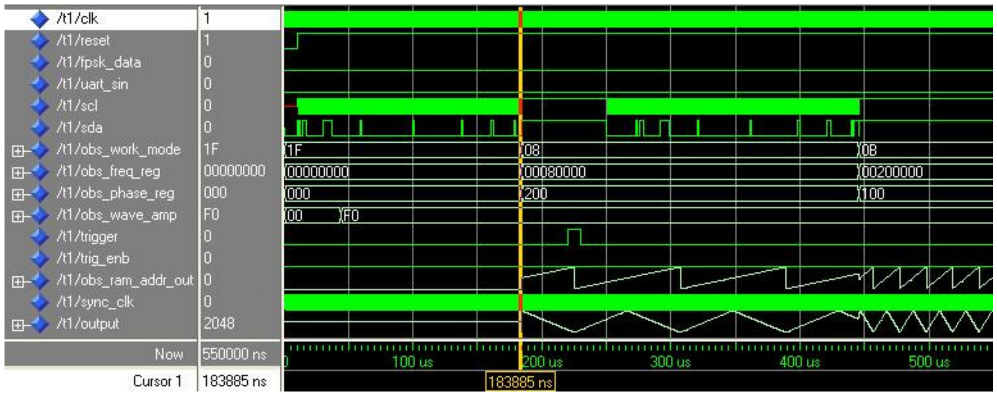


**Figure 4-25 Triangular Wave Generation Simulation (Using I2C Bus)**

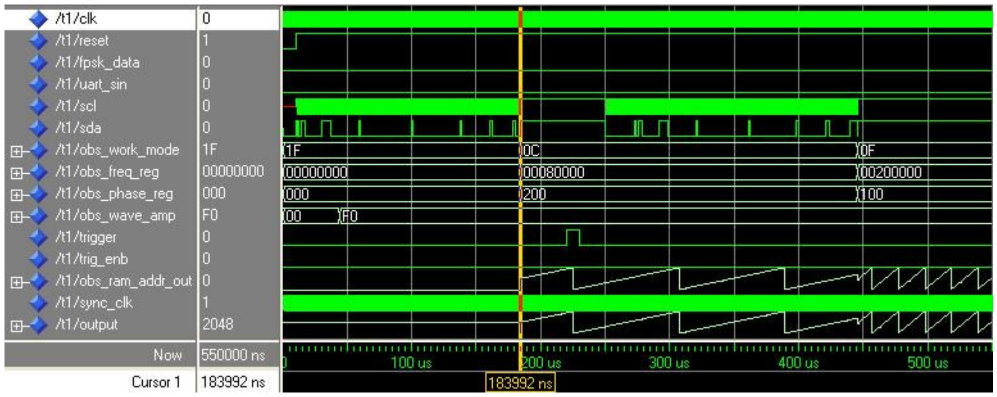


**Figure 4-26 Ramp Wave Generation Simulation (Using I2C Bus)**

The sine wave generation using UART interface is shown in Figure 4-27. In this simulation, the trigger is enabled by making trig_enb signal logic high. Then the configuration data is transmitted by the test bench module. When the transmission is completed, the configuration registers take their new values. The waveform generation does not start before trigger pulse is transmitted by test bench module. As can be seen in figure, output value and max_output (addressed look-up table output) value are different due to the amplitude scaling.

At the time of cursor, the test bench module transmits the soft reset data package (0x98). After reception of this information, the function generator resets its configuration registers and stops the sample generation. The soft reset signal is set only using UART interface and it is transmitted to all sub-modules of function generator as reset signal. When break reset data package (0x99) received, the function generator removes the soft reset condition. However the waveform generation does not start due to the loss of old configuration data. The new configuration data must be loaded to initialize the waveform generation again.

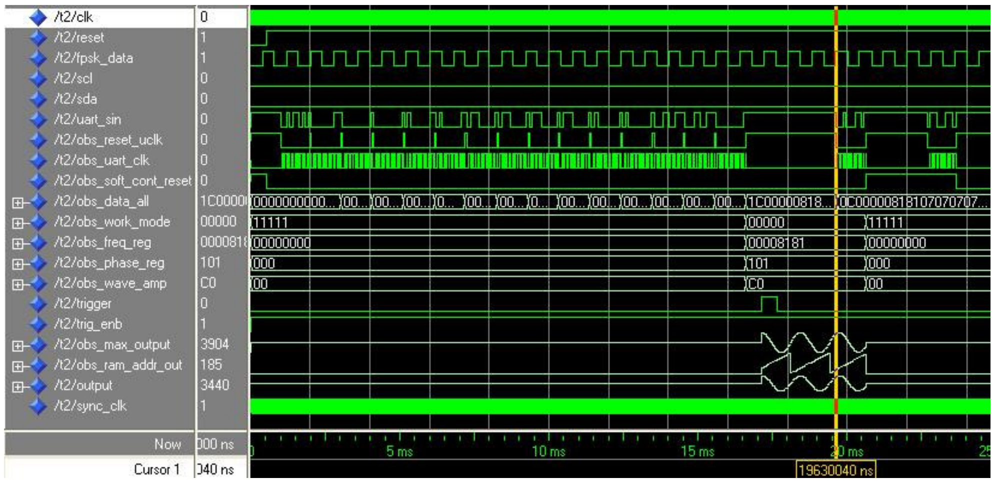


**Figure 4-27  Sine Wave Generation Simulation (Using UART)**

## 4.3.2 VERIFICATION ON HARDWARE

The DDS function generator design was verified by testing it after implementing into FPGA. The function generator descriptions were written in synthesizable format using VHDL. Xilinx's Virtex-4 V4FX12 was selected as target FPGA. The hand-written VHDL descriptions were synthesized using Xilinx XST software. The estimated device utilization summary of function generator descriptions is given in Table 4-1. These values are supplied by Xilinx XST synthesis tool and reported after the synthesis of design descriptions. The available resources colon of the presented table represents the target FPGA's hardware resources.

**Table 4-1 Device Utilization Summary for Function Generator VHDL Code**

| Logic Utilization | Used | Available | Utilization |
|---|---|---|---|
| Number of Slices | 513 | 5472 | 9% |
| Number of Slices Flip Flops | 334 | 10944 | 3% |
| Number of 4 input LUTs | 883 | 10944 | 8% |
| Number of bonded IOBs: | 22 | 320 | 6% |

In order to verify the function generator design, the development boards should have FPGA and D/A converter. Memec Virtex-4 FX12 LC Development Board [32] and Memec P160 Analog Module [33] were selected as hardware resources to test the design. The basic properties and pictures of these boards are given in Appendix C.1 and Appendix C.2.

Memec's development board contains Xilinx's Virtex4 series V4FX12 FPGA [31]. The Xilinx Virtex-4 family is revolutionized the fundamentals of FPGA economics. It presents three application-domain-optimized platforms. These can be ordered as logic, DSP and embedded processing based platforms. Virtex-4 FPGAs deliver breakthrough performance at the lowest cost and offer a compelling alternative to ASICs.

Memec's other board, analog module, was used with the development board. The analog and development boards are combined to form analog application board. The analog board contains high speed A/D and D/A converters. When the analog and development boards are merged, the converters and FPGA has been connected with digital I/O channels. The analog module does not only receive the digital data for D/A converters, it also receives the synchronization clock signals from the development board. These clock signals are used by D/A converters and determine the analog conversion rate. The analog module has also power connections with the development board.

Since the VHDL descriptions are generated using Xilinx's ISE software, the usage of Xilinx's FPGA is an advantage in terms of software support, quick and easy programming. Xilinx ISE software supports full code-to-FPGA processes for Xilinx FPGAs. The code-to-FPGA steps can be ordered as below:

1. Syntax check of description
2. Synthesis of description
3. Implementation (Translate - Map – Place and Route)
4. Generation of programming file
5. Programming the device

In order to generate programming file, the function generator design should have been implemented for the selected FPGA. In this study, the development board contains Xilinx V4FX12 FPGA and the design is implemented for this FPGA technology. Before the implementation, the constraint file is written to define hardware I/O connections. If the constraints are not defined, the implementation is done according to the synthesis tool's defaults. The implementation constraint file includes the definitions ordered below:

1. Timing constraints
2. Package pin assignments
3. Area constraints

Implementation process consists of translating, mapping, placement and routing of a logical design into the targeted Xilinx FPGA. At this stage, the logical design file, which is output of the synthesis stage, is converted to a native circuit description (NCD file). This file contains hierarchical components used to develop the design and the Xilinx primitives. The implementation includes the processes ordered below:

1. Translate (NGDBuild)
2. Mapping (MAP)
3. Placement and Routing (PAR)

The Translate process merges all of the input netlists and design constraints and outputs a Xilinx native generic database (NGD) file, which describes the logical design reduced to Xilinx primitives. The Map process maps the logic defined by an NGD file into FPGA elements, such as CLBs and IOBs. The output design is a native circuit description (NCD) file that physically represents the design mapped to the components in the Xilinx FPGA. The Place and Route process takes a mapped NCD file, places and routes the design and produces an NCD file that is used as input for bitstream generation. The Generate Programming File process produces a bitstream for Xilinx device configuration. After the design is completely routed, the device is configured so that it can execute the desired function. Xilinx's code-to-FPGA design flow is illustrated in Figure 4-28. Input and output files for each flow step are defined in figure.

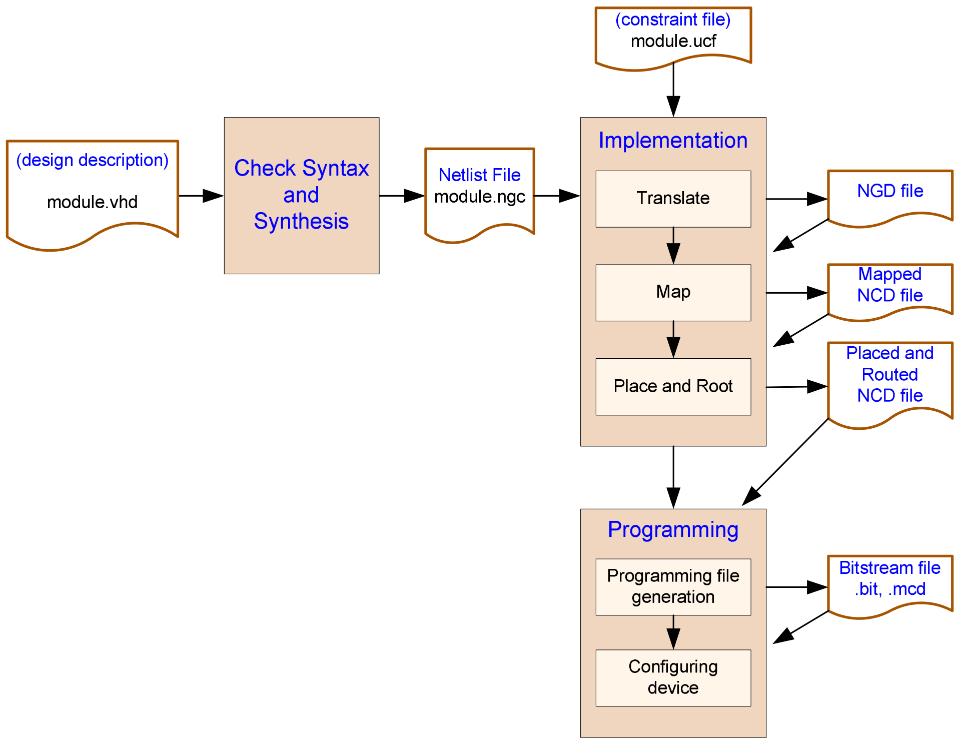


**Figure 4-28 Xilinx's Code-to-FPGA Design Flow**

FPGA is programmed using Xilinx's Platform Cable USB [34]. The hardware test setup used in this study is illustrated in Figure 4-29. The test setup includes the Memec's development board, Memec's analog module, Xilinx's programmer, oscilloscope and test computer. Memec's boards are combined to form waveform generation hardware and Xilinx's programmer is connected to JTAG port of Memec's development board. After following setup steps defined above, FPGA is programmed using Xilinx ISE environment and the test setup becomes ready for function generator design verification.

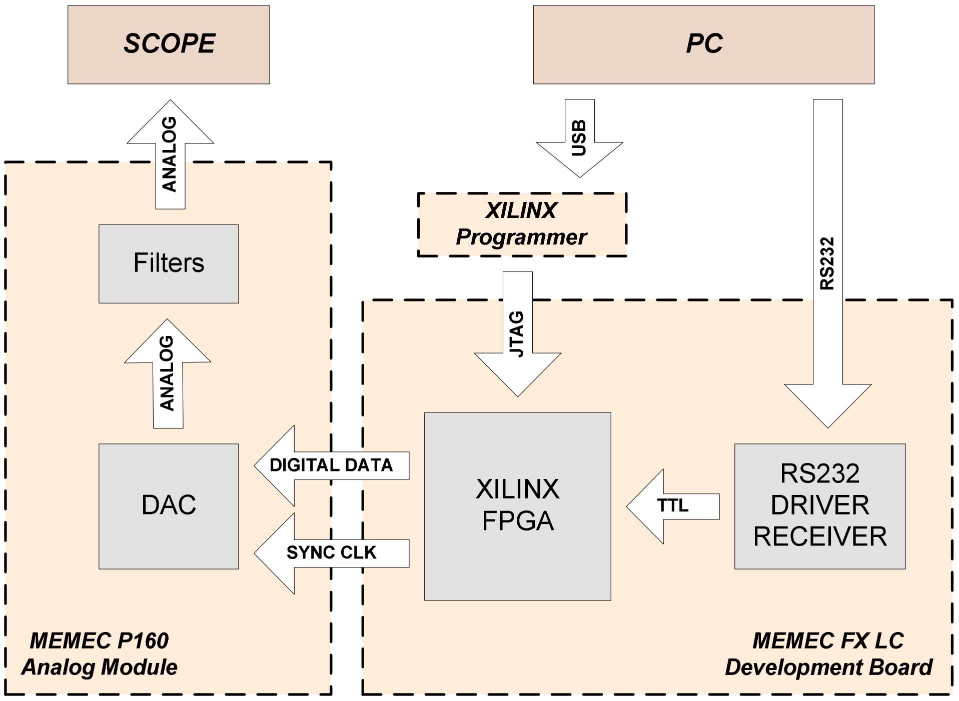
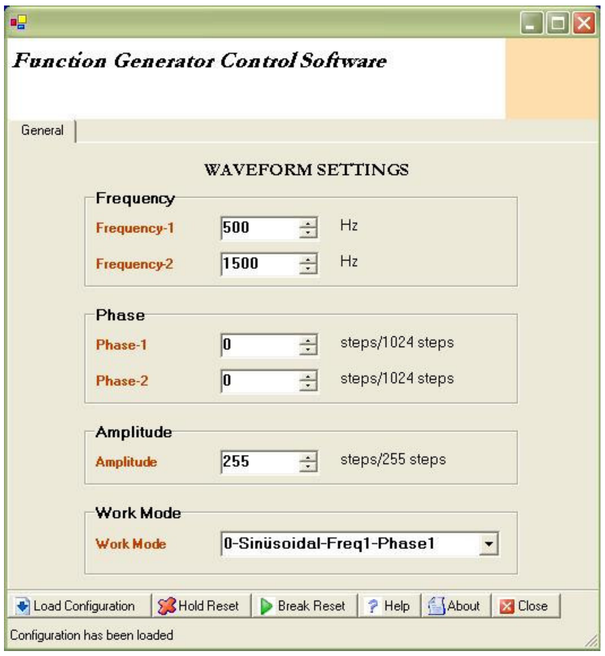


**Figure 4-29 Hardware Test Setup**

In order test the function generator core, the waveform configuration parameter should have been sent to the FPGA. The Memec's development board [32] has physical serial port. By setting a communication channel between the development board and the computer using the serial ports, the waveform configuration parameters would be sent to the development board.

The application software was prepared in order to control the serial port of the computer. This software allows communication with the function generator core over UART. The user interface of the software is shown in Figure 4-30.


*Function Generator Control Software*

General

WAVEFORM SETTINGS

Frequency
Frequency-1: 500 Hz
Frequency-2: 1500 Hz

Phase
Phase-1: 0 steps/1024 steps
Phase-2: 0 steps/1024 steps

Amplitude
Amplitude: 255 steps/255 steps

Work Mode
Work Mode: 0-Sinüsoidal-Freq1-Phase1

Configuration has been loaded

**Figure 4-30  Function Generator Control Software with UART Interface**

In order to load the configuration parameters using this software, the serial port of the development board is connected to the computer's serial port. The frequency, phase offset, amplitude and work mode parameters are selected on the user interface of the control software. When "Load Configuration" button is pressed, the loading is executed. In order to reset the function generator (soft reset), "Hold Reset" button is pressed. When "Break Reset" button is pressed, the reset condition is removed. The hardware verification test setup and generated waveforms by DDS function generator are shown in Figure 4-31, 32, 33, 34.

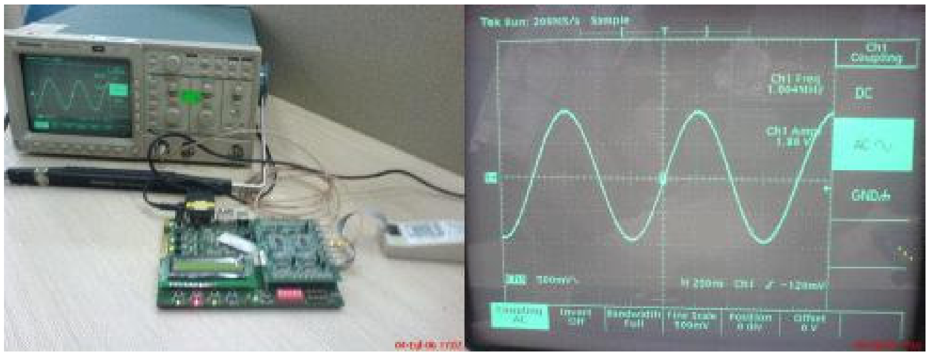


Figure 4-31  Sine Wave Generation on Hardware

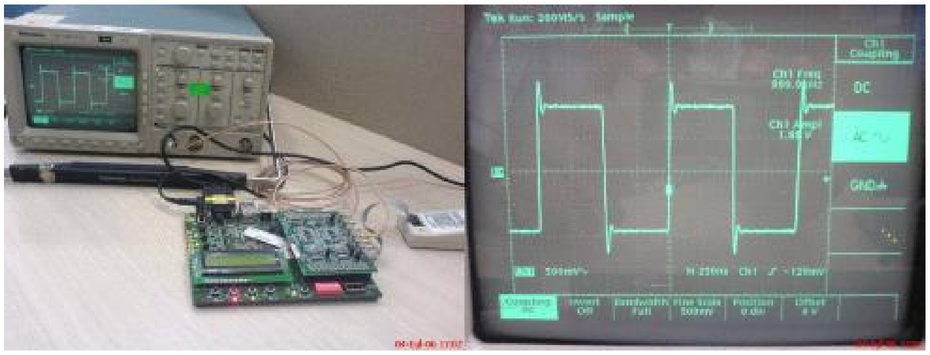


Figure 4-32  Square Wave Generation on Hardware

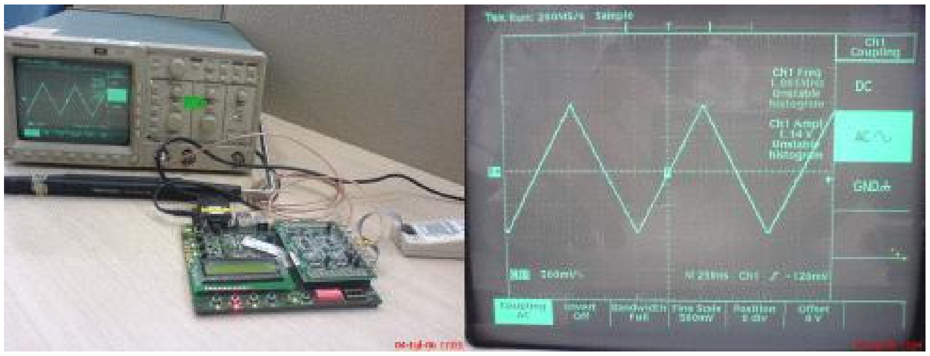


Figure 4-33  Triangular Wave Generation on Hardware

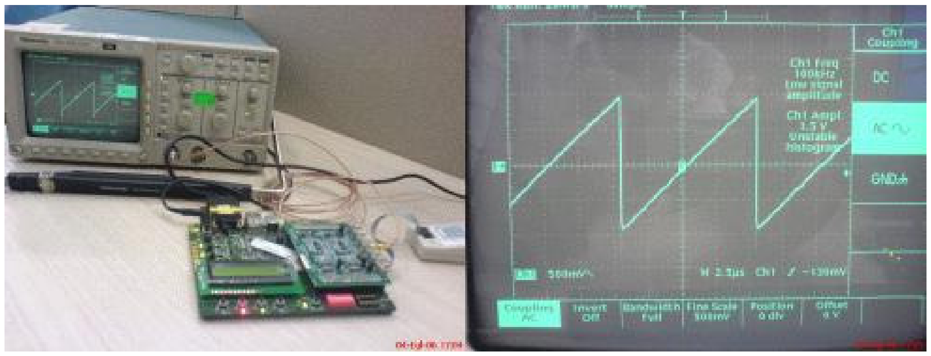


**Figure 4-34 Ramp Wave Generation on Hardware**

## 4.4 SYNTHESIS OF UART BAUD GENERATOR SYSTEMC CODE

In this section, the synthesis results of the UART baud generator SystemC description are presented. The VHDL description of the same module is also synthesized to compare the synthesis performances of two synthesis flows.

An important point is the fact that UART baud generator module descriptions were written in synthesizable RTL format using SystemC and VHDL. The SystemCrafter's synthesis tool translated the SystemC descriptions to VHDL code. The hand-written VHDL code and translated VHDL code were synthesized using Xilinx's synthesis tool.

After an automatic translation of the SystemC descriptions to VHDL with SystemCrafter SC, generated codes are synthesized using Xilinx Project Navigator. First, a new design project is created in Xilinx Project Navigator. Then Xilinx's Spartan-2E XC2S50E is selected as target FPGA and Xilinx XST is chosen as a synthesis tool. All of the VHDL design files that describe the synthesized circuit are added to the source directory of the project location.

An additional gate library file namely craft_generator.vhd is supplied as a part of the SystemCrafter distribution. It can be obtained from the SystemCrafter SC's install directory. In order to synthesize the automatically translated VHDL files, the gate library file is also added to the project as a VHDL package file. This file includes the SystemC library of gate-level descriptions prepared by the SystemCrafter SC.

The synthesis flow using Xilinx XST synthesis tool is illustrated in Figure 4-35. The Xilinx XST receives the SystemCrafter's VHDL package and design description file as input and generates the netlist file that describes the gate-level structure of the design as output.

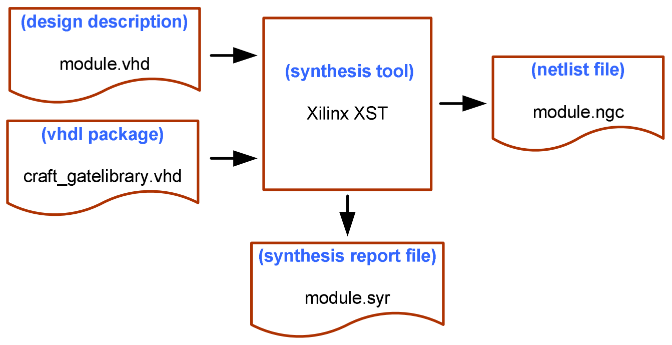


**Figure 4-35 Synthesis of Translated VHDL Code Using Xilinx XST**

Xilinx XST generates a synthesis report describing hardware utilization and timing. The estimated device utilization summary for automatically translated UART baud generator VHDL code is given in Table 4-2. The device utilization summary for hand-written VHDL code is given in Table 4-3.

**Table 4-2 Device Utilization Summary for Translated VHDL Code**

| Logic Utilization | Used | Available | Utilization |
|---|---|---|---|
| Number of Slices | 26 | 768 | 3% |
| Number of Slice Flip Flops | 23 | 1536 | 1% |
| Number of 4 input LUTs | 41 | 1536 | 2% |
| Number of bonded IOBs: | 7 | 4 | 3% |
| Number of GCLKs: | 1 | 1 | 25% |

**Table 4-3 Device Utilization Summary for Hand-written VHDL Code**

| Logic Utilization | Used | Available | Utilization |
|---|---|---|---|
| Number of Slices | 12 | 768 | 1% |
| Number of Slice Flip Flops | 17 | 1536 | 1% |
| Number of 4 input LUTs | 22 | 1536 | 1% |
| Number of bonded IOBs: | 7 | 4 | 3% |
| Number of GCLKs: | 1 | 1 | 25% |

As shown in Table 4-2 and Table 4-3, the translated code requires more hardware resource than hand-written code. The SystemCrafter SC synthesis tool adds redundant logic descriptions when translating the SystemC descriptions to VHDL. Xilinx XST synthesis tool removes most of this redundant logic. However, the reduction of the logic is not in satisfactory level. In this experiment, hand-written code requires nearly the half of the resources required by translated code.

New well developed synthesis tools can use hardware resources more effectively. However, it is nearly impossible to reach the same level of hand-written hardware utilization using current synthesis tools.

## 4.5 COMPARISON OF SYSTEMC AND VHDL DESIGN FLOWS

In this study, the digital design flow of SystemC RTL and VHDL RTL is evaluated in terms of code to synthesized gate-level netlist transformation. In traditional design flow, the algorithmic concept designers use C/C++ for modeling. This is followed by architectural level design using VHDL or Verilog. Since there is a gap related with higher abstraction level support of VHDL and Verilog, SystemC was born to close this abstraction level gap. However SystemC should show its advantages for RTL synthesis to be replaced with the traditional languages. In this section, the comparison is tried to be made for SystemC and VHDL function generator designs in terms of synthesis and implementation.

For VHDL implementation of the function generator, Xilinx's ISE was used. For SystemC implementation, the SystemC library version 2.0.1 was employed with Microsoft Visual C++ 6.0 which lets developing and debugging of the design code. The simulations of the VHDL and SystemC design codes were done in different platforms. It can be observed that there are advantages and disadvantages of using SystemC language in digital hardware design flow. The comparison of VHDL and SystemC based hardware design flows is made when presenting advantages and disadvantages of SystemC design flow. Advantages of using SystemC design flow can be stated as below:

- SystemC proves its importance as being a common language in the whole design process. It provides that the designer does not need to switch design languages using SystemC direct synthesis flow (direct design flow from behavioral model to gate-level model).
- Since it provides various abstraction levels with a single language, the test benches can be reused at various modeling levels.
- SystemC eases the design refinement while passing from higher abstraction level to lower abstraction levels. The designer does not need to recode all the interfaces and hierarchies designed at higher abstraction levels.
- SystemC shortens development cycles with its fast simulation and debug support. A high-level simulation engine is required to obtain fast simulation speed. SystemC achieves this using standard C/C++ development environment. VHDL function generator design is simulated using ModelSim XE III 6.0a. It turned out to be significantly slower than the SystemC simulation environments.

Disadvantages of using SystemC design flow can be stated as below :

- In the market, there are lots of high-performance HDL synthesis tools and a little number of well developed SystemC tools. Current SystemC synthesis tool performances are not satisfactory.
- There are problems with error definitions reported by SystemC library. Some error origins are not found with justifiable expenses.
- There are various SystemC tools that translate the SystemC descriptions to VHDL or Verilog. However each has special recommendation in order to write synthesizable SystemC code. This damages the think of vendor independent SystemC language.
- Translation of the SystemC descriptions to VHDL generates redundant logic and results in utilization of more hardware resources when implementation into hardware. SystemCrafter's synthesis tool was used to translate SystemC descriptions of function generator. It showed that the hardware resource utilization for translated VHDL code is more than resource utilization for hand-written VHDL code.
- After translation of SystemC descriptions to VHDL, the designer can not make refinements in the generated VHDL code easily. The translated code includes redundant logic and has a different writing style that can not be traced easily.
- Synthesizable VHDL description may include more than one process each of which may have different signals in the sensitivity list. However, SystemCrafter's synthesis tool requires that all the processes in the module must have same source in their sensitivity lists. This restricts the effective process utilization and undermines the idea of writing same synthesizable code using SystemC language easier. During the translation of SystemC descriptions, each process defined in the module is written into different design files and this makes tracing the whole design code difficult.

# CHAPTER 5

# DISCUSSION AND CONCLUSION

In this thesis, a direct digital synthesis (DDS) based function generator design core having I2C and UART communication interfaces is presented, defined and implemented using two digital hardware modeling/design languages namely SystemC and VHDL. The simulation, synthesis and applicability performances of these two design languages are compared by following all hardware design stages which starts with specification definition and ends with generation of gate-level netlist. The VHDL function generator descriptions are also implemented into FPGA and verified with hardware tests. SystemC to hardware flow is followed by translating UART baud generator SystemC descriptions to VHDL and synthesizing the translated code. The synthesis performances of the hand-written and translated VHDL codes are compared in terms of approximate hardware resource utilization during implementation into hardware. The design of DDS function generator using both SystemC and VHDL enables observing SystemC language's power and potential in the hardware design flow. The weak aspects of SystemC design flow are also observed and presented in this study.

The specifications of designed DDS function generator is determined upon a detailed research about the present commercial DDS, Numerically Controlled Oscillator (NCO) and function generator ICs or Intellectual Properties (IPs) in the market. There are limited numbers of function generator ICs that use

DDS technique and generate higher frequency outputs. Moreover, most of these ICs do not allow the digital adjustment of waveform amplitude. Various DDS and NCO IPs are present in the market however; ready to use DDS function generator IP has not been found. Moreover, most of DDS and NCO IPs do not allow amplitude adjustment also. The designed DDS function generator core combines DDS module and popular standard serial communication interfaces namely I2C and UART. The integration of I2C receiver module within a DDS function generator gives a great flexibility for remote control of the design by simple microprocessors supporting I2C. In the market, ICs or IPs which integrate DDS and I2C has not been found and this integration is proposed in this study. The designed function generator presents amplitude adjustment, external trigger for waveform generation and FSK/PSK modulation capabilities also. It is verified with adequate number of test bench modules using both SystemC and VHDL verification environments. Besides, VHDL design description is implemented into FPGA and verified with hardware based tests. Since the designed core has UART communication interface, the function generator's output waveform configuration parameters have been loaded to FPGA from a personal computer's UART port.

The DDS modules seem to become popular for periodical waveform generation applications in the last years. Periodical waveforms are generated by function generators that use various waveform production techniques. They can generate various waveform outputs at a desired frequency, phase and amplitude. These waveforms are generated for different applications such as new product testing, clock signal generation and carrier wave generation for communication systems. The function generator implemented in this study uses DDS technique to generate periodical waveforms. As the name implies, the analog sine wave is completely generated by digital circuits using this technique. The designed function generator combines DDS core module with two serial peripheral interfaces (I2C and UART), internal controller module, waveform generation logic (for waves except sine wave) and amplitude control logic.

A DDS function generator core written using VHDL is implemented into Xilinx's Virtex4 series V4FX12 FPGA. The simulation results presented in Chapter 4 show that the function generator design suggested in this study is applicable to the FPGAs. The implementation results show that the maximum internal logic delay of function generator for this FPGA technology is smaller than 8 ns and this allows operation using nearly 125 MHz DDS reference clock. Maximum frequency of periodical waveform output is 40 MHz for this reference clock rate. It is observed that function generator core proposed in this study can be implemented into FPGAs having limited hardware resources and controlled by processors having I2C or UART serial interfaces. The presented function generator core can generate higher frequency outputs when compared with Analog Device's AD9833 programmable waveform generator IC which achieves maximum 12.5 MHz output. Besides, this waveform generator IC does not allow digital amplitude adjustment. The presented function generator design is also a case study sample to compare SystemC and VHDL hardware design flows. The design can be more developed with dedicated effort to prepare and add innovative design modules to present structure.

SystemC appeared to provide a modeling platform for system level designs. The ability to model entire system using a single platform is a critical capability for system designers. In this platform, the designers can iterate the designs faster and more cost effectively. In the last years, SystemC gained popularity as a hardware design language which can replace traditional HDL based design languages such as VHDL and Verilog. Since using a single platform and single language for hardware modeling, design and verification will be very effective in time and cost. Its capability in hardware and software co-verification and co-design is also an advantage for electronics industry. SystemC is supported by various vendors to synthesize the written code directly. However, the present synthesis tools are not satisfactory in performance when compared to traditional HDL synthesis tools.

Since both SystemC and VHDL hardware design flows are followed during this study, the different design and verification platforms had to be utilized due to lack of design tools supporting both design flows. The design descriptions are written using Microsoft Visual Studio 6.0 and Xilinx ISE design development environment. The SystemC simulation waveforms are observed using GTKWave waveform viewer software after the generation of simulation file. The VHDL simulations are realized using ModelSim simulation tool.

In this study, SystemC to hardware flow is also illustrated by synthesizing the UART baud generator module into hardware. There are two different approaches for synthesizing SystemC descriptions. These are (i) direct synthesis of the SystemC descriptions and (ii) synthesis of VHDL codes after SystemC-VHDL translation. The first one is a simple, very effective approach, but unfortunately there is not sufficient number of well developed tools in today's electronic market. For this reason, first the SystemC descriptions have been translated to the VHDL codes and then VHDL-based synthesis cycle is followed. As the translation software, SystemCrafter SC SystemC synthesis tool is used.

The automatic translation of SystemC descriptions results in introduction of redundant logic code and the designer may not trace the generated VHDL code. The modifications may not be done on automatically generated VHDL code easily. The modifications must be done in the SystemC descriptions, and then the descriptions must be translated to VHDL again. During the synthesis, some of this redundant logic is removed depending on the capability of the synthesis tool. However it is observed that SystemC to VHDL translation is not an effective solution from the removing redundancy point. In this study, it is shown that the translated VHDL code utilizes hardware resources nearly twice as much as hand-written VHDL code. However, this hardware utilization can be reduced by more powerful translation software or direct synthesis tools.

When writing descriptions with the synthesizable subset of the SystemC language, coding becomes similar to the syntax of VHDL. However there are more rules and limitations in order to write synthesizable SystemC code when it is compared with VHDL. For instance, SystemCrafter SC synthesis tool does not accept processes that include different signals in their sensitivity list. Due to these limitations, whole function generator description could not be translated to

VHDL. The designs including only sequential processes which are sensitive to a single and same clock edge (like microprocessor applications) can be easily implemented using SystemC. However, DSP applications generally include both combinatorial logic and sequential processes sensitive to many different signals. Hence, it is very difficult to write the synthesizable SystemC descriptions for such applications.

Finally, SystemC brings several advantages such as low simulation times, effective modeling and hardware/software co-design/co-verification. However it has even unsolved problems defined in this study. The SystemC to hardware design flow will likely be further investigated in the future. The designed DDS function generator core allows generating standard waveforms and making FSK/PSK modulation. The core is controlled using I2C and UART interfaces and has trigger and amplitude scaling capabilities. The modular architecture of the core lets to improve the design sub-modules or replace them with new ones. The design architecture is suitable to make arbitrary waveform generation possible. On the other hand, in addition to the presented design, the sizes of the look-up table and phase accumulator can be enlarged to have better resolutions. The look-up table holding the arbitrary waveform samples can be added and its words can be refreshed by loading the samples of various waveforms using communication interfaces. Besides, high speed Serial Interface Peripheral (SPI) modules may be added to load the waveform configuration data more quickly.

# REFERENCES

[1] A. Habibi, S. Tahar, “A Survey: System-on-a-Chip Design and Verification”, Technical Report, Electrical and Computer Engineering Department, Concordia University, January 2003

[2] F. Bruschi, F. Ferrandi, “Synthesis of complex control structures from behavioral SystemC models”, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE’03), pp. 112-117, 2003

[3] N. Calazans, E. Moreno, F. Hessel, V. Rosa, F. Moraes, E. Carara, "From VHDL Register Transfer Level to SystemC Transaction Level Modeling: a Comparative Case Study", Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI’03), pp. 355-360, September 2003

[4] R. Schutten, “Raising the Level of Abstraction Reduces System-on-Chip Verification”, Synopsys Inc., March 2004

[5] S. Swan, “SystemC Transaction Level Models And RTL Verification”, Design Automation Conference, 43rd ACM/IEEE, pp. 90-92, July 2006

[6] G. Martin, "SystemC and the Future of Design Languages: Opportunities for Users and Research", Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI’03), pp. 61-62, September 2003

[7] Open SystemC Initiative, “Functional Specification for SystemC Version 2.0”, 2002

[8] Open SystemC Initiative, “SystemC Version 2.0 User’s Guide, Update for SystemC 2.0.1”, 2002

[9] Open SystemC Initiative, “SystemC 2.0.1 Language Reference Manual Revision 1.0”, 2003

[10] I. Yarom, G. Glasser, “SystemC Opportunities in Chip Design Flow”, Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2004), pp. 507-510, December 2004

[11] Synopsys Inc., “Describing Synthesizable RTL in SystemC”, May 2001

[12] SystemCrafter Inc., SystemCrafter SC User Manual Version 2.0.0

[13] Synopsys Inc., “Synopsys CoCentric SystemC Compiler: RTL User and Modeling Guideline”, 2001

[14] J. Saul, “Using SystemC and SystemCrafter to Implement Data Encryption”, Xcell Journal, Issue 58, pp.32-34, 2006

[15] Prosilog Inc., “Prosilog’s SystemC Compiler Datasheet”, 2002

[16] C. Cote, Z. Zilic, “Automated SystemC to VHDL Translation in Hardware/Software Codesign”, 9th International Conference on Electronics, Circuits and Systems, pp. 717-720, September 2002

[17] L. Cordesses, “Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 1)”, IEEE Signal Processing Magazine, pp. 50-54, July 2004

[18] L. Cordesses, “Direct Digital Synthesis: A Tool for Periodic Wave Generation (Part 2)”, IEEE Signal Processing Magazine, pp. 110-112, September 2004

[19] S. Cheng, J. R. Jensen, R. E. Wallis, G. L. Weaver, “Further Enhancements to the Analysis of Spectral Purity in the Application of Practical Direct Digital Synthesis”, IEEE International Ultrasonics, Ferroelectronics, and Frequency Control Joint 50th Anniversary Conference, pp. 462-470, 2004

[20] Analog Devices Inc., AD9858 1 GSPS Direct Digital Synthesizer Datasheet, 2003

[21] Analog Devices Inc., AD9959 4 Channel 500 MHz DDS with 10-bit DACs Datasheet, 2005

[22] Analog Devices Inc., AD9833 Programmable Waveform Generator Datasheet, 2003

[23] Qualcomm Inc., Synthesizer Products Data Book, August 1997

[24] Harris Semiconductor, HSP45102 12-Bit Numerically Conrolled Oscillator Datasheet, 1996

[25] Fairchild Semiconductor, TMC2340 Digital Synthesizer Datasheet, September 2000

[26] Xilinx Inc., Logicore Numerically Controlled Oscillator V1.0.3 Datasheet, December 1999

[27] Xilinx Inc., Logicore DDS Compiler V1.0 Datasheet, September 2006

[28] Altera MegaCore, NCO Compiler User Guide, December 2006

[29] Lattice ispLever Core, Numerically Controlled Oscillator IP Core User Guide, December 2006

[30] Philips Semiconductors, I2C Bus Specifications, January 2000

[31] Xilinx Inc., Virtex-4 User Guide, October 2006

[32] Memec Inc., Memec Virtex-4 FX12 LC Development Board User’s Guide, April 2005

[33] Memec Inc., Memec P160 Analog Module User Guide, July 2003

[34] Xilinx Inc., Platform Cable USB Product Specifications, June 2006

[35] IEEE Computer Society, “IEEE Standard SystemC Language Reference Manual”, New York, USA, March 2006

# APPENDIX A

# STRUCTURE OF CD-ROM DIRECTORY

The source codes and executable files of the simulations performed in this study are given in the CD attached at the back cover of this thesis. The table gives the folder structure of the CD.

**Table A-1 Structure of CD-ROM Directory**

| | | |
|---|---|---|
| SystemC_Files/ | Includes SystemC design and simulation files | |
| | SRC/ | SystemC Source Files |
| | TB/ | SystemC Test Bench Files |
| | MAIN/ | SystemC Main Files |
| | SIM/ | SystemC Simulation Results |
| VHDL_Files/ | Includes VHDL design and simulation files | |
| | SRC/ | VHDL Source Files |
| | TB/ | VHDL Test Bench Files |
| | SIM/ | VHDL Simulation Results |
| | HARD/ | Hardware Verification |
| SC_to_VHDL/ | Includes SystemC-to-VHDL translation files | |
| | SC_SRC/ | SystemC Source Files |
| | VHDL_SRC/ | Translated VHDL Source Files |

# APPENDIX B

# REFERENCE DESIGNS

## B.1 ANALOG DEVICES AD9833

The AD9833 is a low power programmable waveform generator capable of producing sine, triangular and square wave outputs. The specifications and functional block diagram of AD9833 are given below.

- Digitally programmable frequency and phase
- 0 MHz to 12.5 MHz output frequency range
- 28-bit frequency resolution (0.1 Hz for 25 MHz reference clock)
- 12-bit phase offset resolution
- 3-Wire Serial I/O Port (SPI) Interface

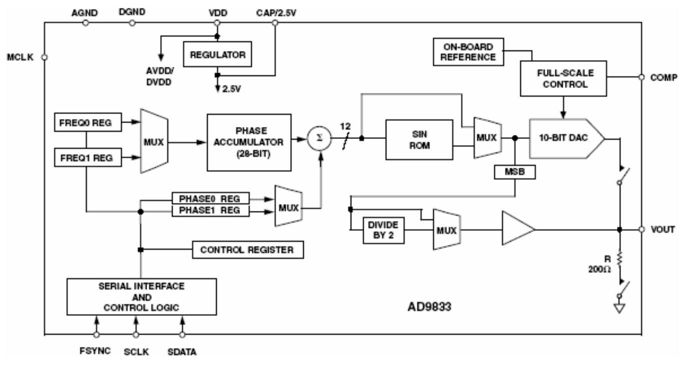


**Figure B-1 Analog Devices AD9833 Functional Block Diagram**

## B.2 ANALOG DEVICES AD9959

The AD9959 consists of four DDS cores that provide independent frequency, phase and amplitude control between channels. The specifications and functional block diagram of AD9959 are given below.

- Four synchronized DDS channels
- Independent frequency/phase/amplitude control between channels
- Linear frequency/phase/amplitude sweeping capability
- Individually programmable DAC full scale currents
- Four integrated 10-bit D/A converters
- 32-bit frequency tuning resolution frequency registers
- 14-bit phase offset resolution
- 10-bit output amplitude scaling resolution
- Serial I/O Port (SPI) with enhanced data throughput

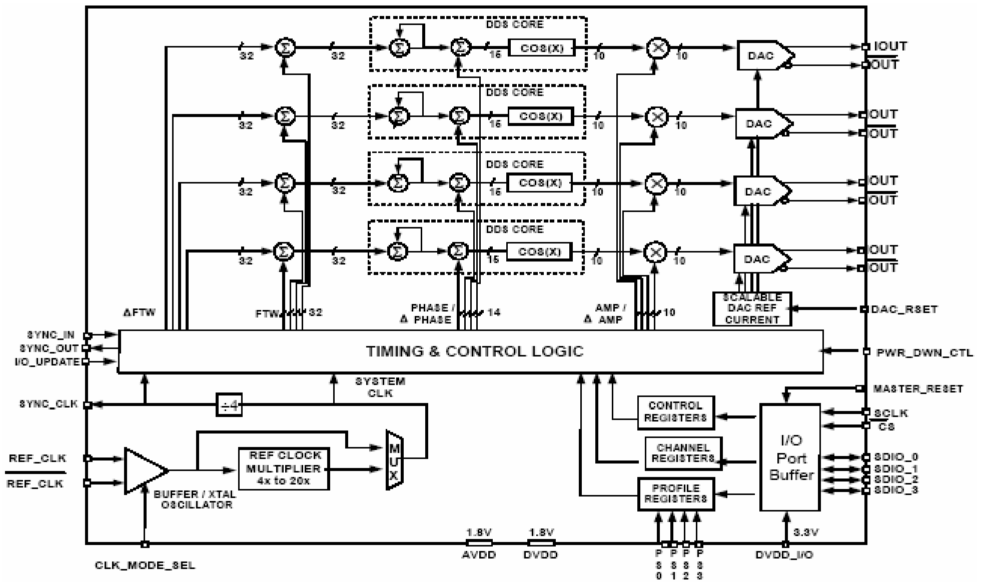
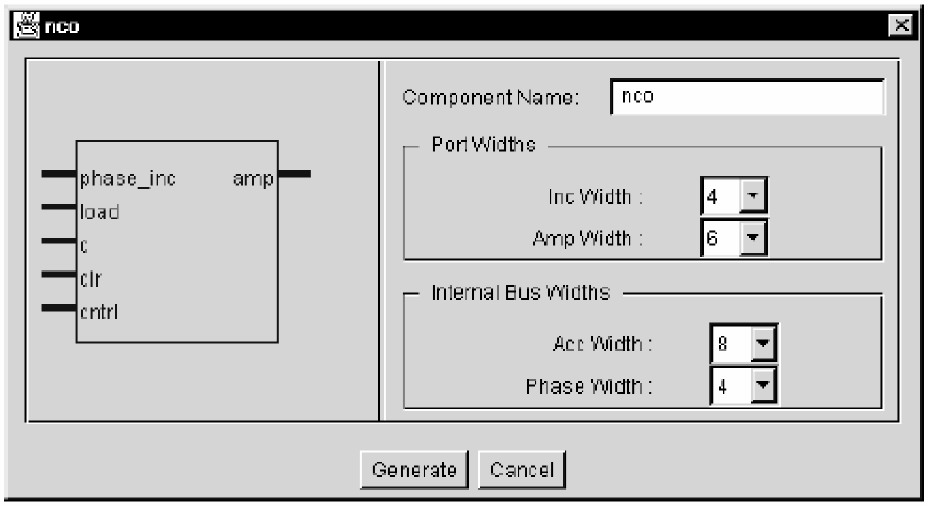


**Figure B-2 Analog Devices AD9959 Functional Block Diagram**

## B.3 XILINX NUMERICALLY CONTROLLED OSCILLATOR V1.0.3

Xilinx’s Numerically Controlled Oscillator (NCO) IP module generates a digital “staircase” approximation to a sine (or cosine) wave, the frequency of which is determined by the input phase increment value. The output can either be used directly, for example, by a digital multiplier, or can be passed into a Digital-to-Analog Converter (DAC) for use in the analog domain. The specifications and parameterization window snapshot of the NCO IP are given below.

- Input phase increment resolution from 3-30 bits
- Output amplitude resolution from 4-16 bits
- Frequency resolution control via phase accumulator word (3-30 bits)
- Phase noise control via programmable phase resolution (3-10 bits)
- Sine/cosine outputs available
- Excellent for high-speed I/Q modulation/demodulation



Component Name: nco

Port Widths

Inc Width : 4

Amp Width : 6

Internal Bus Widths

Acc Width : 8

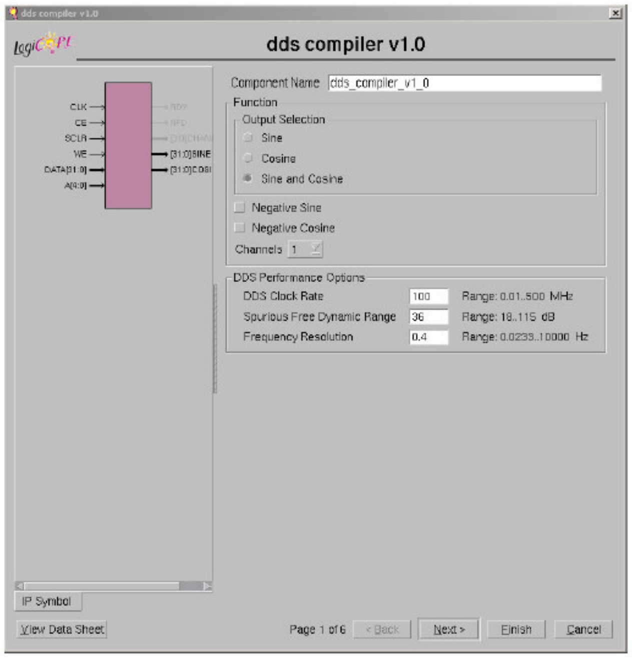
Phase Width : 4

Figure B-3 Xilinx NCO V1.0.3 Parameterization Window

## B.4 XILINX DDS COMPILER V1.0

Xilinx's DDS Compiler IP module is the newest DDS IP in the market and it presents spurious specifications for DDS applications. The specifications and parameterization window snapshot of the DDS Compiler IP are given below.

- Sine, cosine, or quadrature outputs
- Look-up table can be allocated to distributed or block memory
- Frequency resolution control via phase accumulator word (3-32 bits)
- Output amplitude resolution from 4-20 bits
- Support up to 16 independent channels
- Optional phase offset capability allows multiple synthesizers with precisely controlled phase differences



dds compiler v1.0

Component Name: dds_compiler_v1_0

Function

Output Selection: Sine [✓ Sine and Cosine] Cosine

Negative Sine ☐

Negative Cosine ☐

Channels: 1

DDS Performance Options

| | | |
|---|---|---|
| DDS Clock Rate | 100 | Range: 0.01..500 MHz |
| Spurious Free Dynamic Range | 36 | Range: 18..115 dB |
| Frequency Resolution | 0.4 | Range: 0.0233..10000 Hz |

**Figure B-4 Xilinx DDS Compiler V1.0 Parameterization Window**

# APPENDIX C

# HARDWARE TEST TOOLS

## C.1 MEMEC P160 ANALOG MODULE

The Memec Design P160 Analog Module kit includes the following:

- Two 12-bit 53-Msps A/D converters
- AC coupled, single-ended, 1 to 1.5 Vp-p analog input
- Low pass input filter with fc = 19.4 MHz (for A/D converters)
- Two 12-bit 165 Msps D/A converters
- Single-ended, 2 Vp-p analog output, AC coupled output optional
- Low pass output filter with fc = 28.4 MHz (for D/A converters)

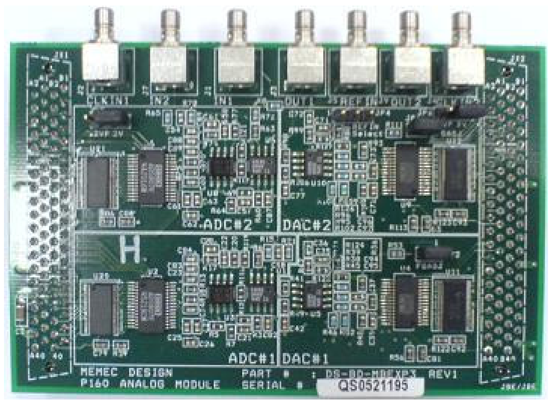


**Figure C-1 Memec P160 Analog Module**

## C.2 MEMEC FX LC DEVELOPMENT BOARD

The Memec Development kit includes the following:

- Xilinx XC4VFX12-10FF668 FPGA
- 64MB of DDR SDRAM and 4MB of Flash
- 10/100/1000 Ethernet PHY
- On-board 100MHz LVTTL Oscillator and LVTTL Oscillator Socket
- P160 Connectors and LCD Panel
- Platform Flash configuration PROM
- PC4 JTAG Programming/Configuration Port
- SystemACE™ Module Connector
- CPU JTAG Port, CPU Debug Port, RS232 Port and USB-RS232 Bridge
- Four User LEDs , Four User Push Button Switches and DIP Switch

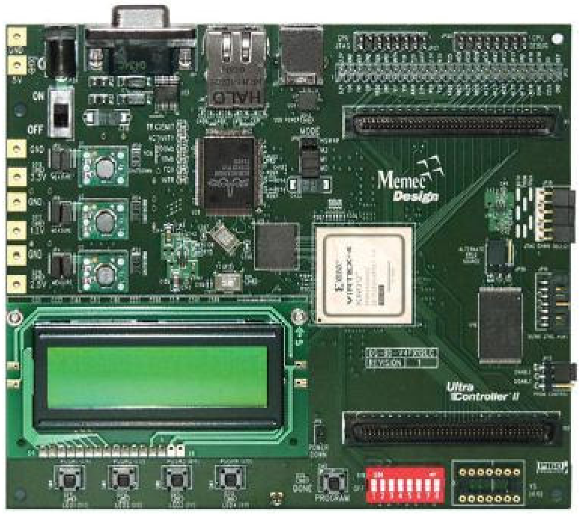


**Figure C-2 Memec Virtex-4 FX LC Development Kit**