Journal of Engineering and Development, Vol. 9, No. 2, June (2005) ISSN 1813-7822

Low Cost and High Speed Look-Up Table Implementation of Xilinx FPGA

Dr. Dhafer R. Zaghar
Computer & Software Eng. Dept., College of Eng.
Al-Mustansiriya University, Baghdad, Iraq

Asst. Prof. Dr. Khamis A. Zidan
Computer & Software Eng. Dept., College of Eng.
Al-Mustansiriya University, Baghdad, Iraq

Dr. Laiyth M. Al-Rawi
Computer & Software Eng. Dept., College of Eng.
Al-Mustansiriya University, Baghdad, Iraq

Abstract

There are two methods to implement LUT up to 7-bit depends on the type of Xilinx chip hardware and the software that can use in design and the code generation. The first method implements LUT as a RAM. This method gives high speed and requires a very high cost.

The second method implements LUT as logic gates. This method requires special software and gives a low speed implies.

This paper proposed a modification to the second method that will save the speed of the first method and low cost of the second method. It depends on the design of the LUT. Therefore it will not require special software.

الخلاصـــــــة

هناك طريقتان لبناء الجداول الرقمية LUT سعة ٧-بت تستند إلى تقنية Xilinx والى البرامجيات المستخدمة في تصميم وتوليد الشفرة. الطريقة الأولى المستخدمة لبناء الجداول الرقمية على شكل ذاكرة RAM وهذه الطريقة تكون عالية الكلفة وذات سرعة عالية. إما الطرية الثانية فتستخدم البوابات المنطقية وهذه تحتاج إلى برمجيات خاصة وتكون إبطا من الطريقة السابقة.

في هذا البحث تم اقتراح طريقة لتحوير وتطوير الطريقة الثانية بحيث تعطي نفس سرعة الطريقة الأولى مع كلفة مقاربة للطريقة الثانية دون الحاجة إلى برمجيات خاصة.

Keywords
DSP, FPGA, LUT, Virtex-II, Xilinx, CLB, bit cell, gate cell

1. Introduction

A large number of system development and integration companies, labs, and government agencies (hereafter referred to as "the community") exist that have traditionally produced DSP applications requiring rapid development and deployment as well as ongoing design flexibility. These applications are generally low-volume and frequently specific to defense and government requirements. This task has generally been performed by software applications on general-purpose computer [1]. Often these general-purpose solutions are not adequate for the processing requirements of the applications and the designers have been forced to employ solutions involving special-purpose hardware acceleration capabilities. These special-purpose hardware accelerators come at a significant cost. The community does not possess the large infrastructure or volume requirements necessary to produce or maintain special-purpose hardware.

Additionally, the investment made in integrating special-purpose hardware makes technology migration difficult in an environment where utilization of leading-edge technology is critical and often pioneered. Recent improvements in Field Programmable Gate Array technology have made FPGA's a viable platform for the development of special-purpose digital signal processing hardware [1], while still allowing design flexibility and the promise of design migration to future technologies [2]. Many entities within the community are eyeing FPGA-based platforms as a way to provide rapidly deployable, flexible, and portable hardware solutions.

Introducing FPGA components into DSP system implementations creates an assortment of challenges across system architecture and logic design, where system architects may be available, skilled logic designers is a scarce resource. There is a growing need for tools to allow system architects to be able to implement FPGA-based platforms with limited input from logic designers. Unfortunately, getting designs translated from software algorithms to hardware implementations has proven to be difficult [2].

Current efforts like VHDL can be processed by traditional FPGA design flows. Other tools use derived languages based on C such as Handel-C [3], C++ extensions such as System C [4], or Java classes such as JHDL [5]. These tools give designers the ability to more accurately model the parallelism offered of the underlying hardware elements. While these approaches attempt to raise the abstraction level for design entry, many experienced logic designers argue that these higher levels of abstraction do not address the underlying complexities required for efficient hardware implementations.

Another approach has been to use "block-based design" [6] where system designers can behaviorally model at the system level, and then partition and map design components onto specific hardware blocks which are then designed to meet timing, power, and area constraints. An example of this technique is the Xilinx System Generator for the Math Works Simulink Interface [7]. Using this tool, a system designer can "develop high performance DSP systems for Xilinx FPGA's. Designers can design and simulate a system using **MatLab**, Simulink, and a Xilinx library of bit/cycle-true models. The tool will then automatically generate synthesizable Hardware Description Language (HDL) code mapped to Xilinx pre-optimized algorithms" [7]. However, this block-based approach still requires that the designer be intimately involved with the timing, and control aspects of cores in addition to being able to execute the back-end processes of the FPGA design flow.

2. Architectural Description of Virtex-II Array [8-11]

The Virtex-II user-programmable gate array, shown in **Fig.(1)**, comprises two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

i. CLBs provide the functional elements for constructing logic.
ii. IOBs provide the interface between the package pins and the CLBs.
iii. CLBs interconnect through a general routing matrix (GRM).

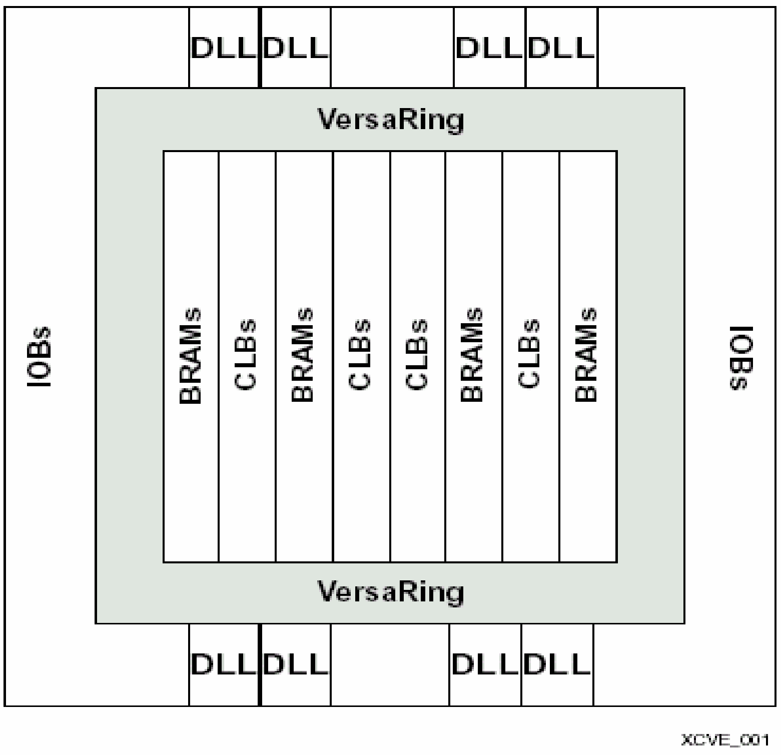

Figure (1) Virtex-II architecture overview

The GRM comprises an array of routing switches located at the intersections of horizontal and vertical routing channels. Each CLB nests into a Versa Block that also provides local routing resources to connect the CLB to the GRM.

The Versa Ring I/O interface provides additional routing resources around the periphery of the device. This routing improves I/O routability and facilitates pin locking. The Virtex-II architecture also includes the following circuits that connect to the GRM.

i. Dedicated block memories (BRAMs) of 4096 bits for each block.
ii. Clock DLLs for clock-distribution delay compensation and clock domain control.
iii. 3-State buffers (BUFTs) associated with each CLB that drive dedicated segmentable horizontal routing resources.

Values stored in static memory cells control the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can be reloaded if necessary to change the function of the device.

Input/Output block in the Virtex-II IOB select I/O inputs and outputs that support a wide variety of I/O signaling standards.

There is three IOB storage elements function either as edge-triggered D-type flip-flops or as level-sensitive latches. Each IOB has a clock signal (CLK) shared by the three flip-flops and independent clock enable signals for each flip-flop. In addition to the CLK and CE control signals, the three flip-flops share a Set/Reset (SR). For each flip-flop, this signal can be independently configured as a synchronous Set, a synchronous Reset, an asynchronous Preset, or an asynchronous Clear. The output buffer and all of the IOB control signals have independent polarity controls.

Input path in the Virtex-II IOB input path routes the input signal directly to internal logic and/ or through an optional input flip-flop. An optional delay element at the D-input of this flip-flop eliminates pad-to-pad hold time. The delay is matched to the internal clock-distribution delay of the FPGA, and when used, assures that the pad-to-pad hold time is zero. Output path includes a 3-state output buffer that drives the output signal onto the pad. The output signal can be routed to the buffer directly from the internal logic or through an optional IOB output flip-flop.

The 3-state control of the output can also be routed directly from the internal logic or through a flip-flop that provides synchronous enable and disable.

3. Xilinx FPGA CLB Architecture [8-11]

The basic building block of the Virtex-II Configurable Logic Block (CLB) is the logic cell (LC). A LC includes a 4-input function generator, carry logic, and a storage element. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop as shown in **Fig.(2)**. Each Virtex-II CLB contains four LCs, organized in two similar slices, as shown in **Fig.(3)**.

In addition to the four basic LCs, the Virtex-II CLB contains logic that combines function generators to provide functions of five or six inputs. Consequently, when estimating the number of system gates provided by a given device, each CLB counts as 4 LCs.

Virtex-II function generators are implemented as 4-input look-up tables (LUTs). In addition to operate as a function generator, each LUT can provide a 16 x 1-bit synchronous RAM. Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM.

F5 in **Fig.(3)** multiplexer output in each slice combines the function generator outputs. This combination provides either a function generator that can implement any 5-input function, a 4:1 multiplexer, or selected functions of up to nine inputs.

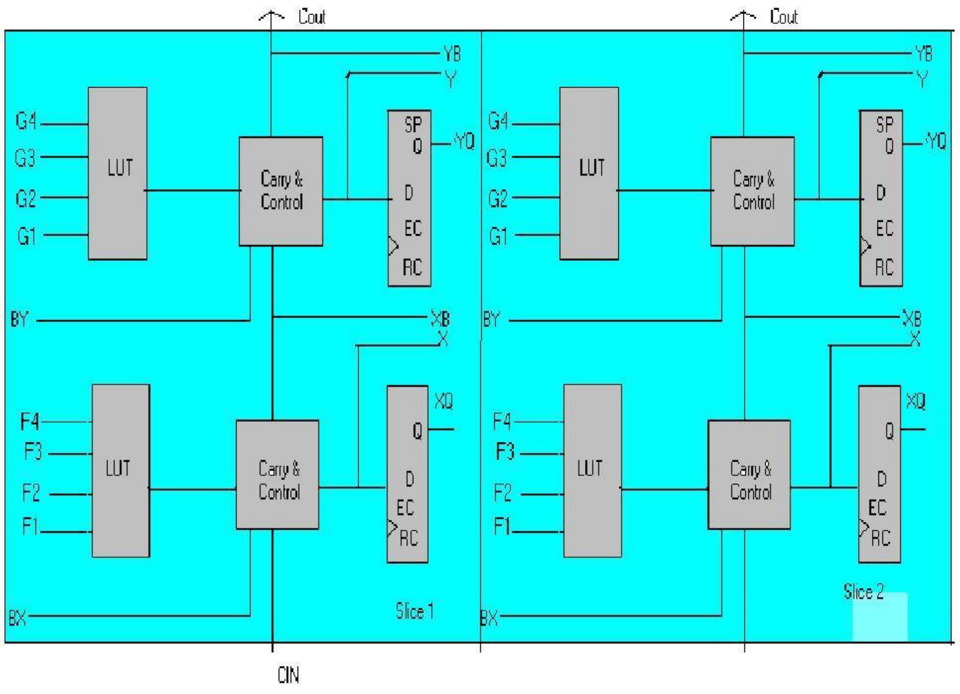

Figure (2) Detailed view of virtex-II slice

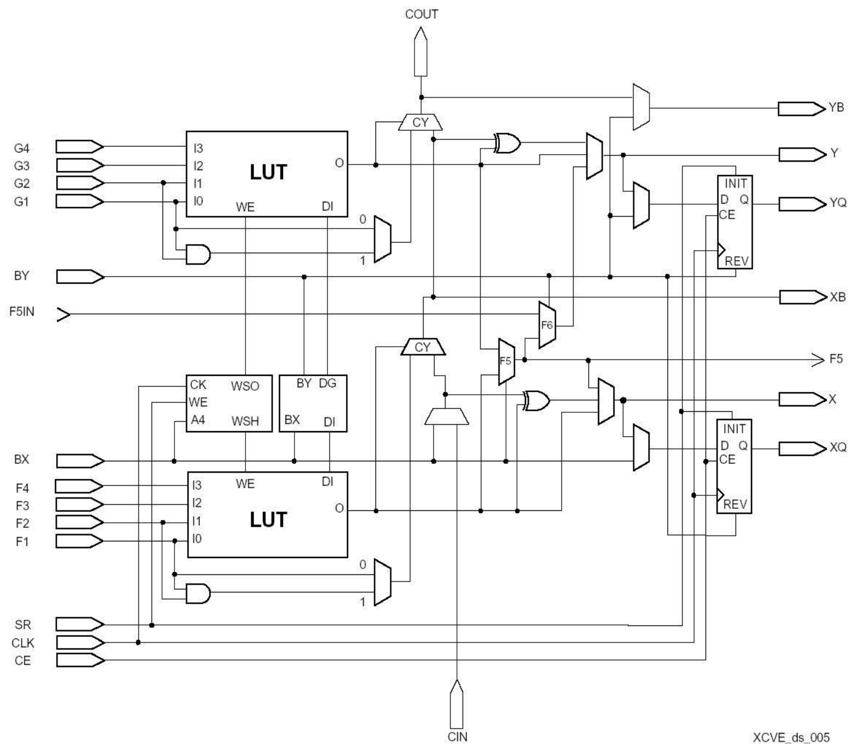

Figure (3) A simplified structure of a CLB of virtex-II

Similarly, F6 multiplexer output combines the outputs of all four-function generators in the CLB by selecting one of the F5-multiplexer outputs. This permits the implementation of any 6-input function, an 8:1 multiplexer, or selected functions of up to 19 inputs. Each CLB has four direct feedthrough paths, two per slice. These paths provide extra data input lines or additional local routing that does not consume logic resources.

4. Conventional Circuit Design using Xilinx FPGA [8]

The characteristics of FPGA give a set of rules to reduce the cost and increase the speed of DSP systems are different from the rules that used in Boolean design. These rules are varied from FPGA to other depending on the type of FPGA. The rules that can be to built an optimal cost and speeds in Xilinx FPGA are that:

i. 1-Divided the large circuit to small sub circuits with input pins less than 7-inputs, because the large input function is uncontrollable space and delay design.

ii. 2-Each sub circuit has 4-inputs for minimum cost, because each 4-inputs function need to one cell and one delay.

iii. 3-Each sub circuit has 6-inputs for maximum speed, because each 6-inputs function need to four cells (one CLB) and one delay.

iv. 4-Each sub circuit has 5-inputs for optimal cost and speed, because each 5-inputs function need to two cells and one delay.

v. 5-When the above rules are not satisfied try to use 3-inputs and not increase the input number over 6 inputs, because each 7-inputs function need to eight cells and two delay, when each 1,2,3-inputs function need to one cell and one delay.

5. LUT Architecture [11-13]

The Lockup Table (LUT) function is a function to convert the linear input data to sinusoidal output data. The basic architecture for the Lockup Table (LUT) is a ROM that stores the output data when the input data represents the address of ROM. The implementation of LUT by using ROM gives low cost LUT, but the LUT is a part from DSP system has other components which give high cost in ROM implementation. Thus the DSP is a part from larger systems not suitable to build in ROM. This mean that there is two solutions, first built LUT by ROM and other parts of over all system by using other technique such FPGA. That is meaning the system needs to moor than one chip and interfacing between these chips that increase the cost and complexity and reduce the speed. The second solution is redesigning the LUT to become suitable with the other parts technique (Xilinx FPGA).

6. The Proposed Implementation Method

This method depends on the separation of the nx1 LUT to (n-6) sub LUT (SLUT). Each SLUT is a 6x1 LUT depending on the possibility of Xilinx FPGA to implement any 6x1 LUT in one CLB. Then, a constant form of a digital function to match the parts of the LUT will be used.

For example an 8x1 LUT shown in Fig.(4), has a function S that depend on 8 variables (ABCDEFGH). This function will generate a 16x16 cell Karnof map. It will divide to 4 units each one is an 8x8 cell sub_Karnof map that has a function (Ti) depending on 6 variables (CDEFGH).

$$S = f(A\,B\,Ti) = \overline{A}\,\overline{B}\,T0 + \overline{A}\,B\,T1 + A\,\overline{B}\,T2 + A\,B\,T3 \ldots\ldots\ldots\ldots\ldots\ldots \quad \textbf{(1)}$$

Each sub_function Ti will require one CLB (4 cells) so the collection function *f* will be required to one CLB (4 cells). This is mean that the total cost of the 8x1 LUT is 20 cells (5 CLBs) as shown in **Fig.(5)**.

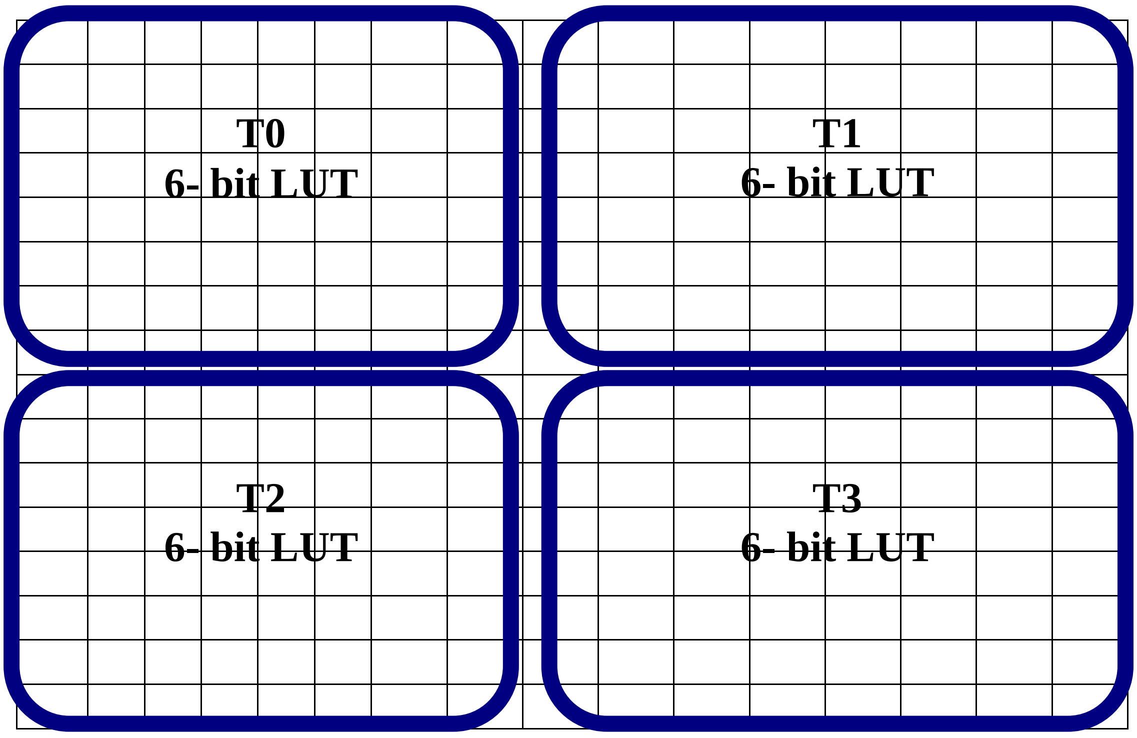

Figure (4)16x16 cell karn of map

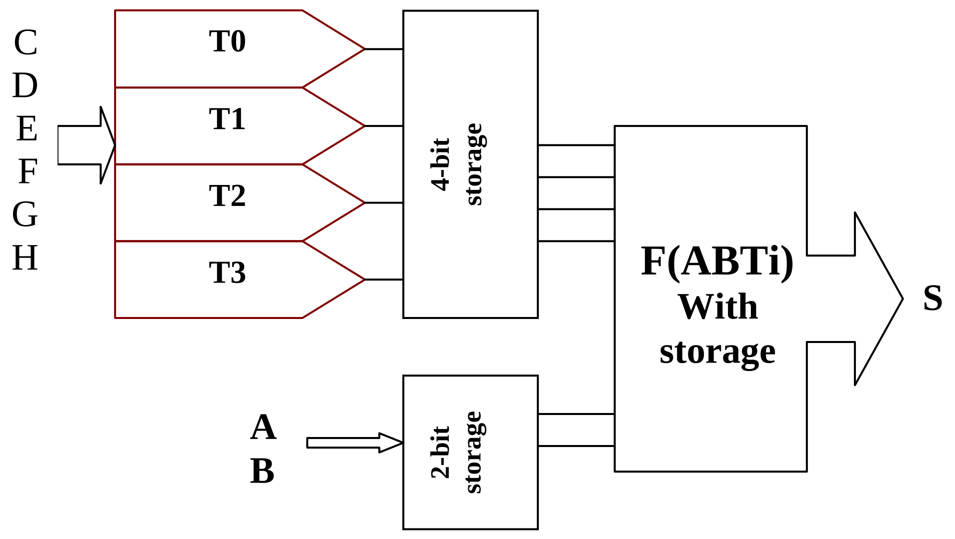

Figure (5) Proposed method for 8x1 LUT

7. Proposed Case Study for Evaluation

The implementation of 10x10-bit LUT using v999999 chip using ISA4.1I directly will require 10,000 cells with a maximum speed 250 MHz. However, the conversion of the data of LUT to Boolean equation (that required to a hard work) will require 640 cells with a maximum speed 50 MHz. Thus the use of the proposed method 10x10-bit LUT will require 1030 cells with a maximum speed 250 MHz but with 4 clock delay between input and output data as shown in **Table (1)**.

Table (1) The cost, speed and RGV for the three methods

Method	*Total cost /cell*	*Speed /MHz*	*RGV*
Direct	10000	250	1
Boolean equations	650	50	3
Proposed	1030	250	9.7

In table (1), the RGV is the rational gain value that can be calculated as follow:

$$RGV = \frac{cost2}{cost1} * \frac{speed1}{speed2} = \frac{10000}{cost1} * \frac{speed1}{250} \quad \text{.........................} \ (2)$$

8. Conclusions

The DSP systems have different functions with LUT. These systems will have a cost problem because the architecture of LUT is suitable with ROM. However the other parts such as adders have a high cost when implemented using ROM. The implementation of the system using Xilinx FPGA give a low cost for all parts except the LUT will has a very high cost. Therefore, the implementation of DSP systems using FPGA gives a total cost less than ROM. This paper shows the method of implement a very low cost and high speed LUT using Xilinx FPGA.

The Xilinx FPGA is a good tool to build the DSP systems, but not all DSP circuits are suitable with the architecture of Xilinx FPGA such as LUT and some circuits. It needs to be redesigned to become suitable with conventional circuit design using Xilinx FPGA.

The good partitions to LUT will give a low cost and high-speed DSP system in one chip FPGA. The DDS is an example has a DSP circuits suitable with FPGA and other circuit suitable with ROM and the old design use two chips first is FPGA and the other ROM or a very large FPGA chip. However, by using the proposed method the circuit of DDS built by using small to medium single FPGA chip.

9. References

1. R. Lauwereins, M. Engels, M. Adé, and J. Peperstraete, ***"Grape-II: A System-Level Prototyping Environment for DSP Applications"***, IEEE Computer, Vol. 28, No. 2, February 1995, pp. 35-43.

2. P. Banerjee, et. al., ***"MATCH: A MATLAB Compiler for Configurable Computing Systems"***, Technical Report, Center for Parallel and Distributed Computing, Northwestern University, Aug. 1999.

3. J. Gerlach, and W. Rosenstiel, ***"System Level Design using the System C Modeling Platform"***, http://www.systemc. org/papers/sda-2000.

4. P. Bellows, and B. Hutchings, ***"JHDL- An HDL for Reconfigurable Systems"***, Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines, April 1998, pp. 175-184.

5. H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, ***"Surviving the SOC Revolution: A Guide to Platform-Based Design"***, Kluwer Academic Publishers, 1999.

6. Xilinx System Generator v2.1 for Simulink Reference Guide, Xilinx, 2000.

7. J. Donaldson, ***"From Algorithm to Hardware-The Great Tools Disconnect"***, COTS Journal, October 2001, pp. 48-54.

8. R. Andraka's Website: http://users.ids.net/~randraka.

9. U. Sjֲ strֲ m, M. Karlsson, and M. Hֲ rlin, ***"A Digital Down Converter Chip"***, In Proc. of European Signal Processing Conference EUSIPCO'96, Vol. 1, Trieste, Italy, Sept. 1996, pp. 284–287.

10. L. Wanhammar, ***"DSP Integrated Circuits"***, Academic Press, 1999.

11. M. Vadim, ***"Frequency Synthesis Theory and Design"***, Third addition, Wiley & Sons, New York, 1975.

12. D. R. Zaghar, ***"A Hybrid Frequency Synthesizer using Digitally Controlled Oscillator in Microwave Band"***, M.Sc. Thesis, Baghdad University, 1997.

13. N. Lindlbauer, ***"Application of FPGA's to Musical Gesture Communication and Processing"***, M.Sc. Thesis, Berkeley, 1999.

List of Symbols

CLB:	configurable logic block.
DDS:	direct digital synthesizer.
DLL:	delay locked loop.
DSP:	digital signal processing.
FPGA:	Field programmable gate array.
HDL:	Hardware Description Language.
IOB:	input/output buffer.
JHDL:	Java classes HDL.
LC:	logic cell.
LUT:	look-up table.
RGV:	Rational gain value.
VHDL:	very high speed HDL.