The 1st Regional Conference of Eng. Sci. NUCEJ Spatial ISSUE vol.11,No.1, 2008 pp91-97

Design And Implementation Of High Speed Complex Multiplier Using Fpga

Ali Mohammed Hassan Al-Bermani
College of Information Engineering, Al-Nahrain University
e-mail: alicom1980@yahoo.com

Raya Kahtan Mohamed
College of Information Engineering, Al-Nahrain University
e-mail: . rayait2005@yahoo.com

Abstract

Multiplication is an important part in real-time digital signal processing (DSP). The present work deals with the design and implement of complex multiplier/mixer using Field Programmable Gate Array (FPGA) chip with low cost and high speed.

Two devices of FPGA are chosen to implement the design; to achieve the task of mixer system implementation. The rules that are important for such implementation are proposed in order to reach the minimum cost and high speed requirement for the individual component of mixer system. These components are software simulated using VHDL language, with software called MODELSIM version SE-EE5.4a. Since mixer is important in any digital receiver because of high speed need, so different multiplier method are proposed with different data resolution and different worst case of additional noise. To achieve high speed data, a parallel tree multiplier is used with Wallace tree method which is optimal in speed but it has a complicated routing that makes it impractical to implement, because of this, we present a modification for fast parallel multiplier using both Wallace tree and Booth algorithm to achieve a sufficient design for most of DSP application. The proposed design of mixer is simulated using ISE4.1i and results in successful achievement of its desired specification. The final implementation of programmable (4, 8, 16, 32 and 64) bit mixer data input resolution is achieved using Virtex-II devices and also implemented in LP-2900 CPLD device. The resulting performance depending on multiplier method are viewed in mixer cost. However, the routing is much more regular with great reduction in FPGA cost and it is achieved for the desired mixer when compared with other methods.

Keywords: *Digital Communication, VHDL, FPGA, ISE4.1i, Virtex-II, Wallace tree, Booth algorithm.*

1. Introduction

The function of the mixer is to multiply the incoming signal by the locally generated sinusoid to shift the spectrum of the signal. A straightforward implementation uses two multipliers, one for each the sine and cosine terms [1]. The mixer can mix either a real or complex input with the Numerical Control Oscillator (NCO) sine and cosine outputs. For down-conversion, the mixer performs the equation:

$$\text{Mixer Output} = (\text{Ii} + \text{j Qi})(\text{Cos} - \text{j Sin}) \quad (1)$$

This can also be represented in the following form:

$$\text{Mixer Output} = ((\text{Ii Cos}) + (\text{Qi Sin})) + \text{j}((\text{Qi Cos}) - (\text{Ii Sin})) \quad (2)$$

It is evident that this equation requires four multipliers, an adder and a subtractor. When a real input is desired, the Q input path is replaced with a fixed value of 0x0000. Then the above equation is reduced to:

$$\text{Mixer Output} = \text{Ii}(\text{Cos} - \text{j Sin}) \quad (3)$$

The Mixer can operate in three modes: Normal Mix Mode, Disable Oscillator Mode, and Disable Data Mode. In the "Normal Mix Mode," the mixer mixes the input data with the NCO cosine and sine outputs. The "Disable Oscillator Mode" allows the input data to bypass the mixer and flow directly through to the output. This is accomplished by replacing the NCO cosine with a constant value of 0xFFFF and sine with a constant value of 0x0000. The "Disable Data Mode" allows the NCO cosine and sine outputs to bypass the mixer and flow directly through to the output.

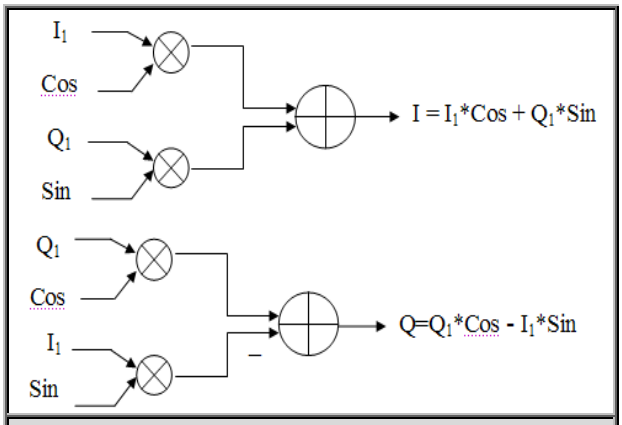

Figure (1) Mixer block diagram [2]

This is accomplished by replacing the input data "I" with a constant value of 0xFFFF and the "Q" with 0x0000. The complex multiplier block diagram is shown in fig.1.

2. Multiplication in FPGA [3, 4]

Multiplication is basically a shift add operation. There are, however, many variations on how to do it. Some are more suitable for FPGA use than others. Bit –parallel multiplier are of two main types, array and tree multipliers. Because of speed and power consideration, the selection, here, is a tree multiplier structure. There are many tree structures one of them is Wallace tree.

A Wallace tree is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder tree does, the Wallace tree sums up all the bits of the same weights in a merged tree. Usually full adders are used, so that 3 equally weighted bits are combined to produce two bits: one (the carry) with weight of n+1 and the other (the sum) with weight n. Each layer of the tree, therefore, reduces the number of vectors by a factor of 3:2. The tree has as many layers as is necessary to reduce the number of vectors to two (a carry and a sum).The structure for this type of multiplier is shown in fig.2

Wallace tree is a tree of carry-save adders. A carry save adder consists of full adders like the more familiar ripple adders, but the carry output from each bit is brought out to form second result vector rather than being wired to the next most significant bit. The carry vector is 'saved' to be combined with the sum later, A Wallace tree multiplier is one that uses a Wallace tree to combine the partial products from a field of 1x n multipliers (made of AND gates).

If the Wallace tree combined with a fast adder can offer a significant advantage there. As Wallace tree is optimal in speed but it has a complicated routing, which in makes it impractical to implement since the cells in the tree has different loads and must be individually optimized, so a modification for fast parallel multiplier using both Wallace tree and Booth algorithms the overturned-Stairs adder is one of the modification of the Wallace tree which has the same speed of Wallace tree and is sufficient for most DSP and communication application. However, the routing is much more regular.

3. Wallace tree with booth algorithm multiplier (WBM)

Wallace Tree algorithm used to sum the partial products in reduced time and Booth algorithm has been used to reduce the number of partial products generated in a multiplication process. Booth observed that when strings of '1' bits occur in the multiplicand the number of partial products can be reduced by using subtraction So, when both algorithms are combined in one multiplier, we can expect a significant reduction in computing multiplications .The delay is proportional to log(n) for Wallace Tree multiplier as shown in fig.3 which represent the relation between the number of input bits and the delay in Wallace Tree multiplier [5,6]

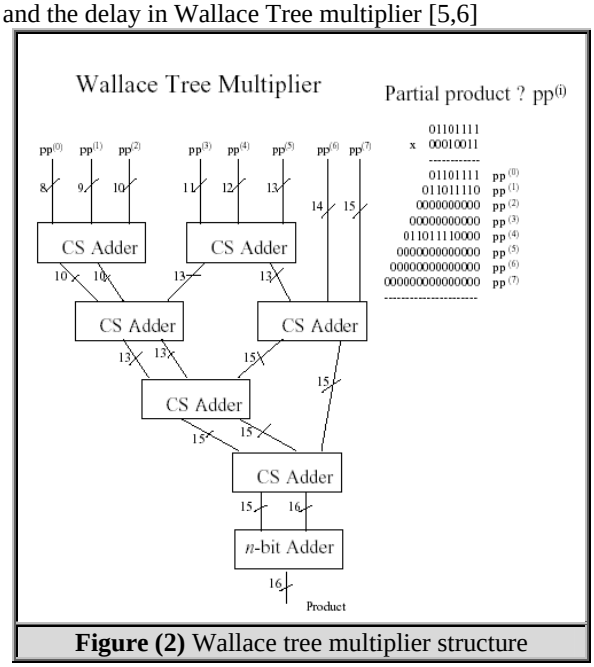

Figure (2) Wallace tree multiplier structure

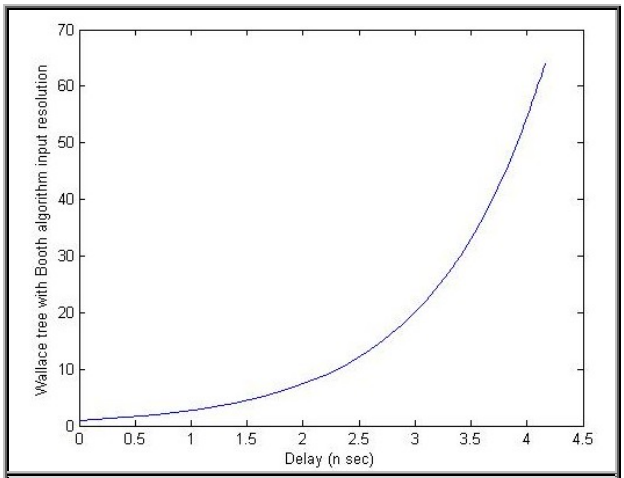

Figure (3) Wallace tree with Booth algorithm delay processing

4. The noise in digital system

There are multi sources of additional noise in receivers such as that due to multiplication and addition that will be generated in the output data. So the worst case of additional noise can be described by the following equation. [7]

$$NW = -20\log 2^{n-2} \approx -6n + 12 \qquad (4)$$

The worst level of additional noise in digital system for different resolution (different number of bits) is shown in fig.4. From this figure we can see the level of additional noise for 8-bit case is highest (-36 dB), which is not suitable for most application of digital systems.

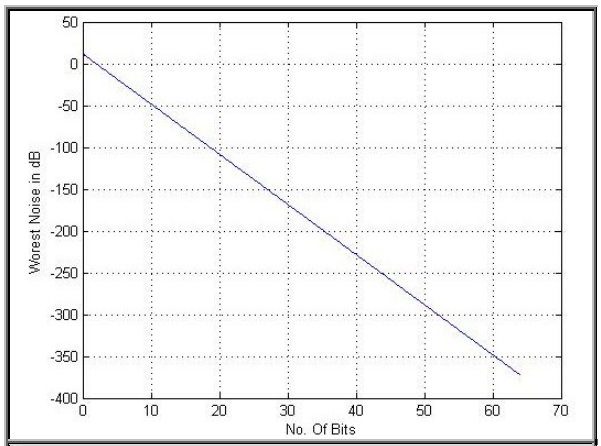

Figure (4) The worst noise in Mixer system

However 12-bit design give critical implementation (-60 dB) which need a careful design and it may not be suitable for all normal application. So more than 12-bit can cover all the normal application and it give ability to make any approximation in the circuit implementation of mixer part.

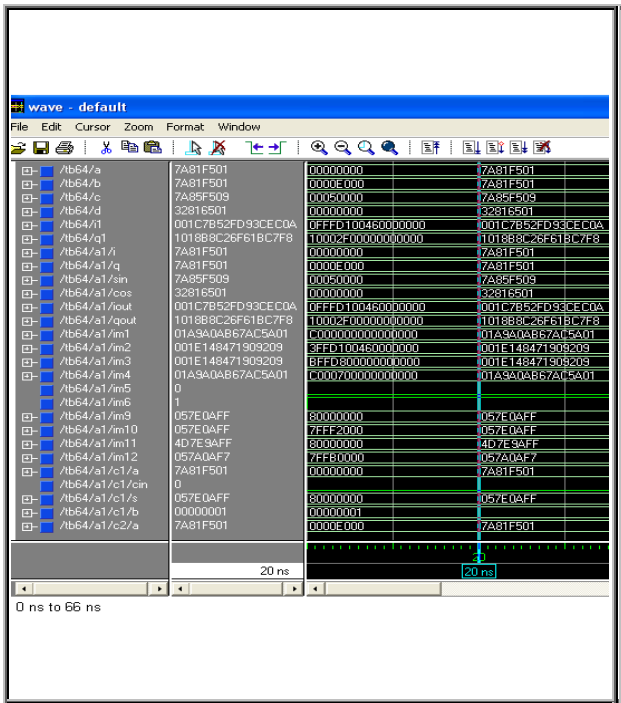

Figure (5) 32-bit Mixer test in hexadecimal values.

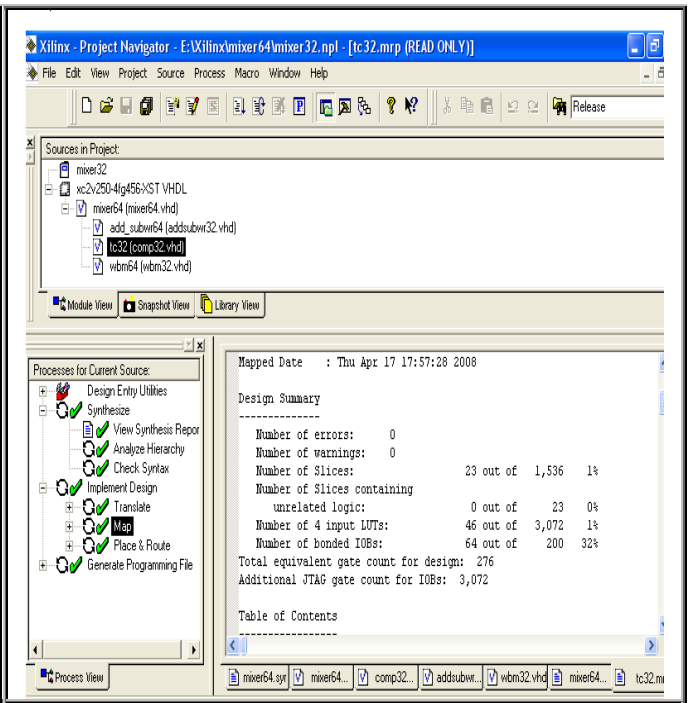

Figure (6) ISE 4.1i run window for 16-bit mixer

5. System implementation and device selection

The proposed design and implementation of mixer system is presented as five designs of mixer which simulated depending on multiplier method and resolution, hence it implemented using Virtex-II FPGA where all stages are choosing the target device [8]. The software used to implement the functions and all the components are written in VHDL code. All components are combined and checked for errors in syntax [9, 10]. Finally all the designed systems are synthesized and ready to be delivered to the target device. As all vertix-II devices have the same specifications except for their number of slices. The first selected device is XC2V250 but for extra resolution of mixer, we use another device XC2V1500 in order to cover this implementation. Also CPLD Altera with LP2900 device kit was used for real download implementation.

5.1 Waveform Test

Fig.5 show 32-bit complex multiplier waveform test in hexadecimal values output conversion
In Fig.6 we can see the implementation of the designed 16-bit mixer using device (XC2V250-4fg 456).

6. CORE RESOURCE UTILIZATION

Tables 1 and 2 show a comparison between two's complement programs of (3 bit, 7 bit, 15 bit ,31 bit, and 63 bit input) using XC2V250-4fg 456

Table (1) Two's complement synthesize report

Design statistics and cell usage	**2'C 3-bit input**	**2'C 7- bit input**	**2'C 15-bit input**	**2'C 31-bit input**	**2'C 63-bit input**
IOs	**8**	**16**	**32**	**64**	**28**
BELs	**4**	**10**	**22**	**46**	**94**
LUT2	**1**	**3**	**7**	**15**	**31**
LUT3	**1**	**5**	**13**	**29**	**60**
LUT4	**2**	**2**	**2**	**2**	**2**
IO Buffers	**8**	**16**	**32**	**64**	**128**
delay	**8.860 ns**	**11.026 ns**	**16.706 ns**	**28.066 ns**	**50.595 ns**
Levels of logic	**3**	**5**	**9**	**17**	**33**

Table (2) Two's complement Map report

Design summary	2'C 3-bit input	2'C 7-bit input	2'C 15-bit input	2'C 31-bit input	2'C 63-bit input
No. of slices	2 out of 1,536 0.130%	5 out of 1,536 0.325%	11 out of 1,536 0.716%	23 out of 1,536 1.497%	47 out of 1,536 3.06%
No. of 4_input LUTs	4 out of 3,072 0.130%	10 out of 3,072 0.325%	22 out of 3,072 0.716%	46 out of 3,072 1.497%	93 out of 3,072 3.03%
No. of bonded IOBs	8 out of 200 4%	16 out of 200 8%	32 out of 200 16%	64 out of 200 32%	128 out of 200 64%
Total equivalent gate count for design	24	60	132	276	558
Additional JTAG Gate count for IOBs	384	768	1,536	3,072	6,144

Tables 3 and 4 show a comparison between Wallace tree with Booth algorithm multiplier (WBM) programs for (4 bit, 8 bit, 16 bit, 32 bit, and 64 bit input) using XC2V250-4fg 456.

Table (3) WBM synthesize report

Design statistics and cell usage	WBM 4 bit input	WBM 8 bit input	WBM 16 bit input	WBM 4 32 bit input	WBM 4 64 bit input
IOs	16	32	64	128	256
IO Buffers	8	16	32	64	128

Table (4) WBM Map report

Design summary	**WBM 4 bit input**	**WBM 8 bit input**	**WBM 16 bit input**	**WBM 4 32 bit input**	**WBM 4 64 bit input**
Number of bonded IOBs	**8 out of 200 4%**	**16 out of 200 8%**	**32 out of 200 16%**	**64 out of 200 32%**	**128 out of 200 64%**
Additional JTAG Gate count for IOBs	**384**	**768**	**1,536**	**3,072**	**6,144**

Tables 5 and 6 show a comparison between Adder/subtractor programs (8 bit, 16 bit, 32 bit, 64 bit ,and 128 bit input) using XC2V250-4fg 456.

Table (5) Adder/Subtractor synthesize report

Design statistics and cell usage	Add/Sub 8 bit input	Add/Sub 16 bit input	Add/Sub 32 bit input	Add/Sub 64 bit input	Add/Sub 128 bit input
IOs	26	50	98	194	386
BELs	16	32	64	129	258
LUT2	1	1	1	1	1
LUT3	1	1	1	1	1
LUT4	14	30	62	126	254
IO Buffers	26	50	98	194	386
delay	18.068 ns	27.792 ns	48.348 ns	89.529 ns	168.228 ns
Levels of logic	10	18	34	67	131

Table (6) Adder/Subtractor map report

Design summary	Add/Sub 8 bit input	Add/Sub 16 bit input	Add/Sub 32 bit input	Add/Sub 64 bit input	Add/Sub 128 bit input
No. of slices	9 out of 1,536 0.586%	17 out of 1,536 1.106%	33 out of 1,536 2.148%	66 out of 1,536 4.297%	131 out of 1,536 8.529%
No. of 4_input LUTs	16 out of 3,072 0.5208%	32 out of 3,072 1.0416%	64 out of 3,072 2.0833%	129 out of 3,072 4.1992%	258 out of 3,072 8.3984%
No. of bonded IOBs	26 out of 200 13%	50 out of 200 25%	98 out of 200 49%	194 out of 200 97%	386 out of 200 93%(error occurs in this case)
Total equivalent gate count for design	96	192	384	774	1,548
Additional JTAG Gate count for IOBs	1,248	2,400	4,704	9,312	18,528

Table (7) Mixer Map report using XC2V1500-5bg575

Design summary	**Mixer 34 bit input**	**Mixer 8 bit input**	**Mixer 16 bit input**	**Mixer 32 bit input**	**Mixer 64 bit input**
No. of bonded IOBs	**18 out of 392 4%**	**34 out of 392 8%**	**66 out of 392 16%**	**130 out of 392 33%**	**258 out of 392 65%**
Additional JTAG Gate count for IOBs	**864**	**1,632**	**3,168**	**6,240**	**12,384**

A comparison between Wallace tree with Booth algorithm multiplier and normal multiplier which are shown in tables 8 and 9.

Table (8) Synthesize report for two types of multipliers

Design statistics and cell usage	**Normal multiplier**	**Wallace Tree with Booth algorithm multiplier**
IOs	16	16
BELs	30	1
LUT2	10	No LUT used
LUT3	2	No LUT used
LUT4	17	No LUT used
IO Buffers	16	8
delay	15.741 ns	0.602 ns

Table (9) Map report for two types of multipliers

Design summary	**Normal multiplier**	**Wallace Tree with Booth algorithm multiplier**
Number of slices	**15 out of 7,680 1%**	**No slice used**
Number of 4_input LUTs	**29 out of 15,360 1%**	**No LUT used**
Number of bonded IOBs	**16 out of 392 4%**	**8 out of 392 2%**
Additional JTAG Gate count for IOBs	**768**	**384**

Then the Wallace tree with Booth algorithm multiplier used in the mixer program was replaced by the normal multiplier and compared the results which are shown in tables 10 and 11 using XC2V1500-5bg575.

Table (10) Mixer synthesize report for two types of multipliers

Design statistics and cell usage	**Mixer with Normal multiplier**	**Mixer with Wallace Tree and Booth algorithm multiplier**
IOs	**30**	**30**
BELs	**149**	**2**
LUT2	**7**	**No LUT used**
LUT3	**763**	**No LUT used**
LUT4	**77**	**No LUT used**
IO Buffers	**30**	**18**

Table 11 Mixer map report for two types of multipliers

Design summary	Mixer with Normal multiplier	Miser with Wallace Tree and Booth algorithm multiplier
Number of slices	82 out of 7,680 1%	No Slice used
Number of 4_input LUTs	147 out of 15,360 1%	No LUT used
Number of bonded IOBs	30 out of 392 7%	18 out of 392 4%
Additional JTAG Gate count for IOBs	1,440	864

The FPGA floorplanner editor results of mixer of 32 bit input are shown in fig.7

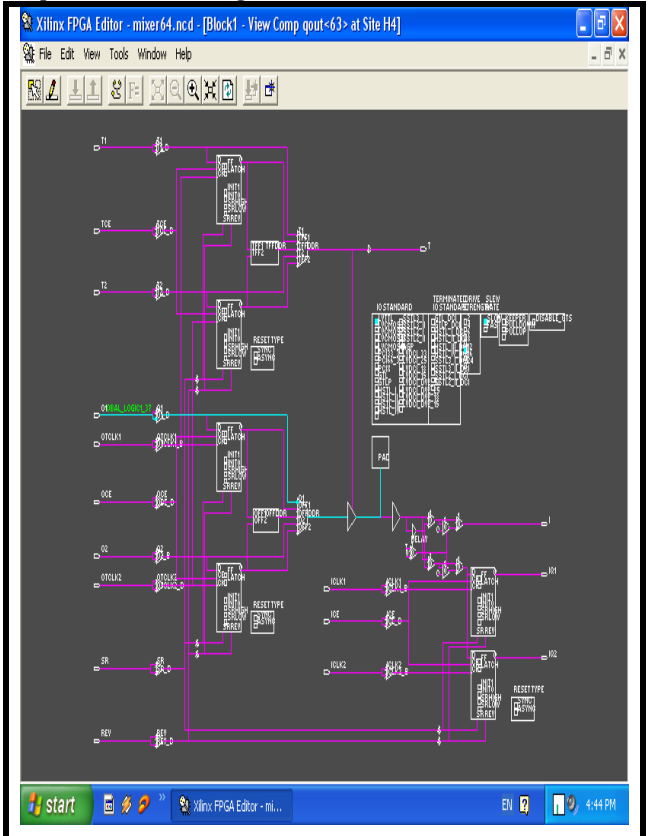

Fig.7 FPGA editor result of a 32 bit input mixer

7.CONCLUTIONS

The main features considered for the designed complex multiplier/mixer is the low complexity to be suitable for implementation using FPGA chip. Different resolutions of Wallace tree multiplier are considered with modification of Booth algorithms to achieve more promising enhancement multiplication in FPGA.

Mixer who implemented using Wallace tree is optimal in speed but it has a complicated routing, which in makes it impractical to implement since the cells in the tree has different loads and must be individually optimized, so a modification was consider with fast parallel multiplier using WBM where overturned-Stairs adder is one of the modification of the Wallace tree which has the same speed of Wallace tree and is sufficient for most DSP and communication application.

A comparison between normal Mixers with high speed processing WBM Mixer were considered.

REFERENCES

[1] Applications Digital Radio "High Performance Digital Down-Converters for FPGAs", by Ray Andraka President, Andraka Consulting Group, Inc. ray@andraka.com

[2] NOVA engineering Inc. "Complex Multiplier/Mixer Megafunction" November 1996. http://www.nova-eng.com

[3] Andraka Consulting Group "Multiplication in FPGA", 08/13/00. http://www.andraka.com

[4] Weidong Li and Lars Wanhammar " VHDL Code Generator for a Complex Multiplier". {weidongl, larsw}@isy.liu.se.

[5] Saket A. Jamkar," ARITHMETIC ARRAYS FOR RECONFIGURABLE FABRICS", UNIVERSITY OF WISCONSIN-MADISON, 2004. http://www.ece.wisc.edu.

[6] "Booth Encoded Wallace-tree multiplier" http://www.eecs.tufts.edu

[7]"J.A. Wepman,J.R. Hoffman" ," RF and IF Digitization in Radio Receivers: Theory, Concepts, and Example " March 1996.

[8] Xilinx Inc. "Virtex-II platform FPGA Overview" products and solutions, 2003
0 0.http://www.xilinx.com

[9] "Introductory VHDL: from simulation to synthesize", prentice Hall Xilinx Design Series, prentice Hall, 2001.

[10] "VISIC Hardware Description Language", January 2008, http://en.wikipedia.org

الخلاصة

ان عملية الضرب هي عملية مهمة في اعطاء نتيجة واقعية لتطبيقات معالجة الاشارة الرقمية.يتعامل هذا البحث مع تصميم وتطبيق الة عقدية لضرب الارقام (complex multiplier (mixer باستخدام شريحة الكترونية قابلة للبرمجة (FPGA) بكلفة قليلة وسرعة عالية. اختير جهازين من ال (FPGA) لتطبيق التصميم والوصول لمهمة الضارب العقدي. تم افتراض الشروط المهمة لهذا التطبيق لغرض الوصول الى اقل كلفة واعلى سرعة. استخدمت لغة ال(VHDL) لتصميم البرامج وتم تطبيقها في برنامج (MODELSIM) من نوع(SE_EE.5a) . بما ان الضارب العقدي هو جزء مهم في اي
مستلم رقمي، استخدم ضارب من نوع شجرة والس للوصول الى السرعة العالية مع انه ضارب معقد وغير عملي في التطبيق.لذلك تم اجراء تعديل عليه باستخدام ضارب شجرة والس مع خورازمية بوث للوصول الى التصميم المطلوب في معظم تطبيقات معالجة الاشارة الرقمية. تم الحصول على نتائج جيدة بمحاكاة تصاميم الضارب العقدي باستخدام برنامج (ISE4.1i). للحصول على الدقة المطلوبة، استخدمت مدخلات البرامج المصممة (٤ بت، ٨ بت ، ١٦ بت، ٣٢ بت ، ٦٤ بت) وطبقت باستخدام (Vertix II) كما طبقت في
(LP-2900 CPLD) .يتوضح اداء النظام المصمم من خلال
الكلفة الناتجة.