Proposed Design and Implementation of a Schematic FPGA-BASED Binary Arithmetic Multiplier[1]

Dr. Yazen A. Khalil[2]

Abstract

This article presents a proposed design and implementation of an 8-bit Arithmetic Multiplier based on FPGA (Field Programmable Gate Array). The design is implemented a schematic FPGA way using CPLD (Complex Programmable Logic Device) development board SN-PLDE2. The development board contains an FPGA device EPF8282ALC4-4 (5000 gates account) of ***Altera*** FLEX8000 family (Flexible Logic Element MatriX) with the other necessary peripherals. The proposed design is achieved under MAXplus2 V10.1 software for FPGA programming. The designed arithmetic multiplier is tested using an experiment board (SN-PLDE3A). The results show both efficient usage and high performance including the accuracy and the fast operation.

Keywords

Arithmetic Multiplier, FPGA, FLEX

الخلاصة

تقدم هذه المقالة تصميما مقترحا وتنفيذاً لضاعف حسابي الكتروني (8-bit Arithmetic Multiplier) باستخدام تقنيات **مصفوفة البوابات القابلة للبرمجة حقلياً (FPGA)**. يتم تنفيذ التصميم باستخدام الطريقة التخطيطية (schematic) باستخدام لوحة منظومة الأجهزة المنطقية المبرمجة (PLD) والذي يحوي مصفوفة البوبات القابلة للبرمجة حقليا من نوع (Altera) برقم (EPF8282ALC4-4) المتكون من 5000 بوابة منطقية قابلة للبرمجة من عائلة FLEX8000 مع الملحقات الضرورية الأخرى. وتم تنفيذ التصميم المقترح باستخدام البرنامج (MAXplus2) الاصدار (10.1). تم اختبار التصميم باستخدام لوحة التجربة (SN-PLDE3A) والملحقة باللوحة الرئيسية. وأظهرت النتائج الاستخدام الفعال الأداء العالي بما في ذلك الدقة والسرعة.

[1] This paper was presented in the Engineering Conference of Control, Computers and Mechatronics Jan. 30-31/2011, University of Technology.

[2] University of Koya - College of Engineering

1. Introduction

The process of a binary multiplication with paper and pencil is illustrated in Figure (1). The multiplicand is multiplied in turn by each digit of the multiplier. These partial products are then added with due consideration for the differing numerical significance of each digit of the multiplier. Each partial product is either identically zero or equal to the multiplicand, depending on whether the multiplier digit is 0 or 1 respectively [1, 2].

					d_3	d_2	d_1	d_0		
					1	0	1	0	Multiplicand	
		x			1	1	0	1	Multiplier	
	0	0	0	0	0	0	0	0	Partial product initially	
					1	0	1	0	d0=1	
Shift				0	0	0	0		d1=0	Partial Prod.
Shift			1	0	1	0			d2=1	
Shift +		1	0	1	0				d3=1	
	1	0	0	0	0	0	1	0	Final Product	

Figure (1) Binary Multiplication Example

2. The Proposed Design

As shown in Figure (1), the binary arithmetic multiplication process needs binary multiplication, shifting, and addition processes. Therefore; the main parts of the proposed design should involve a 8-bit full adder (8ADD), an 8-bit multiplier shift register constructed with two 4-bit shift registers, an accumulator (MUL8ACC) constructed with four PIPO shift registers, pulse generator and a control pulse generator. All above parts will be detailed in the following sub-sections. The complete proposed design is shown in Figure (2).

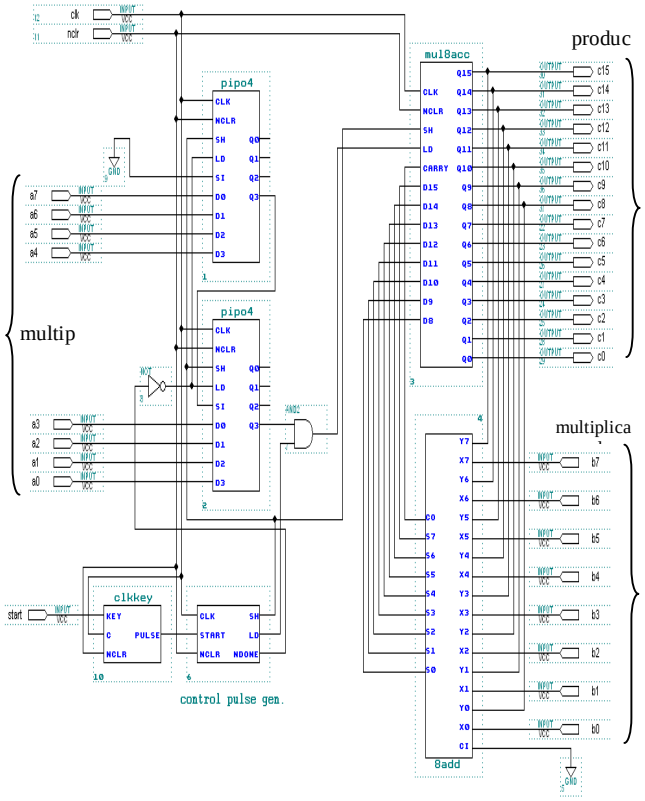

Figure (2) The Proposed Design of 8-bit

The 8-bit multiplicand (b0-b7) is fed to the adder input (X0-X7). The accumulator output (Q8-Q15) is fed to the adder input (Y0-Y7). With the LD control signal, the adder output (S0-S7) and its carry out are loaded to the inputs (D8-D15) of the accumulator (MUL8ACC). The 8-bit multiplier (a0-a7) is parallel-loaded into the multiplier shift register.

The pulse generator (clkkey) generates a starting pulse to drive the 8-bit multiplier. The control pulse generator is used to generate the control pulses for

the multiplication process. The following sections will give the details of each part mentioned above.

2.1 Multiplier Shift Register

This part is constructed with two 4-bit shift registers as shown in Figure (2). Its main job is the serial shift process of the multiplier bits to determine whether the load signal to the LD input of MUL8ACC is generated or not according to the following AND process.

LD(acc) = Multiplier bit • LD(control signal)

For 1 multiplier bit leads to pass the LD signal to load adder output into the accumulator.

2.2 The 8-bit Full Adder (8ADD)

This part is constructed with two 4-bit adders. Each one of these 4-bit adders involves four ordinary full adders as shown in Figure (3). Its job is to implement the required addition process of the partial products till is reachs the final product.

2.3 Accumulator (MUL8ACC)

The accumulator part of this design consists of four 4-bit PIPO shift registers and its control logic circuit as shown in Figure (4). It has two main jobs; the first one is to accumulate (temporary storage) the partial products after each multiplication process with the multiplier bit. The second one is the one bit shifting process that is required after each multiplication process as described in Figure (1), [1, 3].

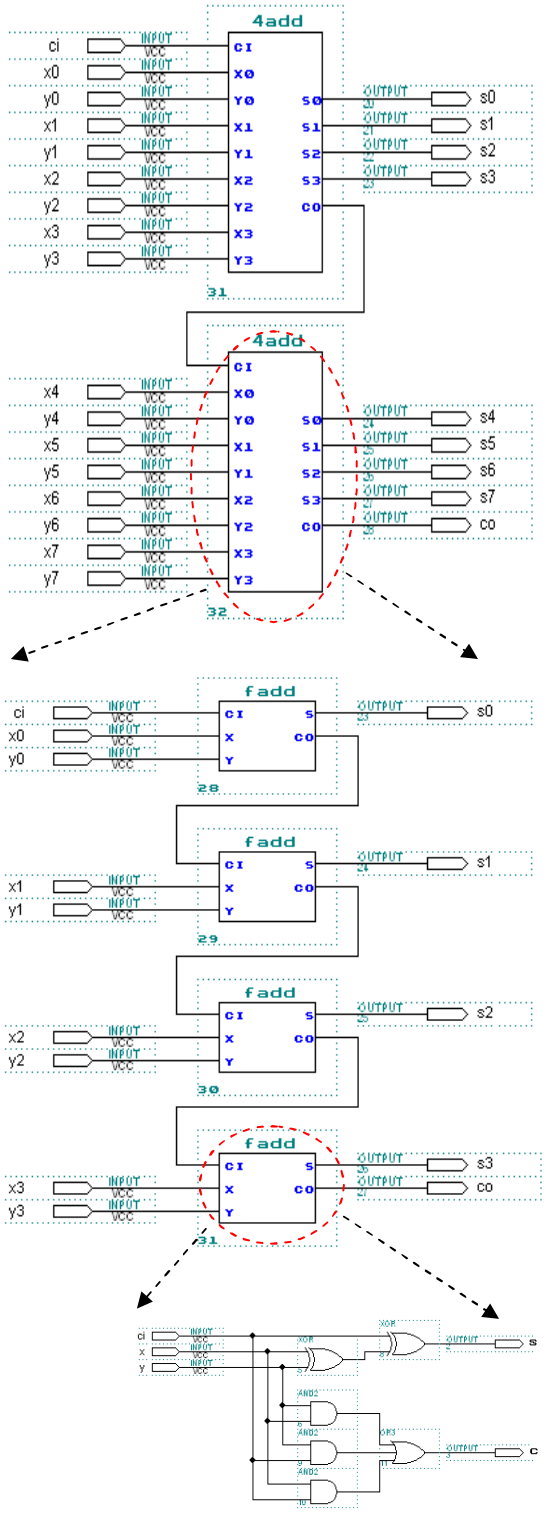

Figure (3) 8-bit full adder (8ADD)

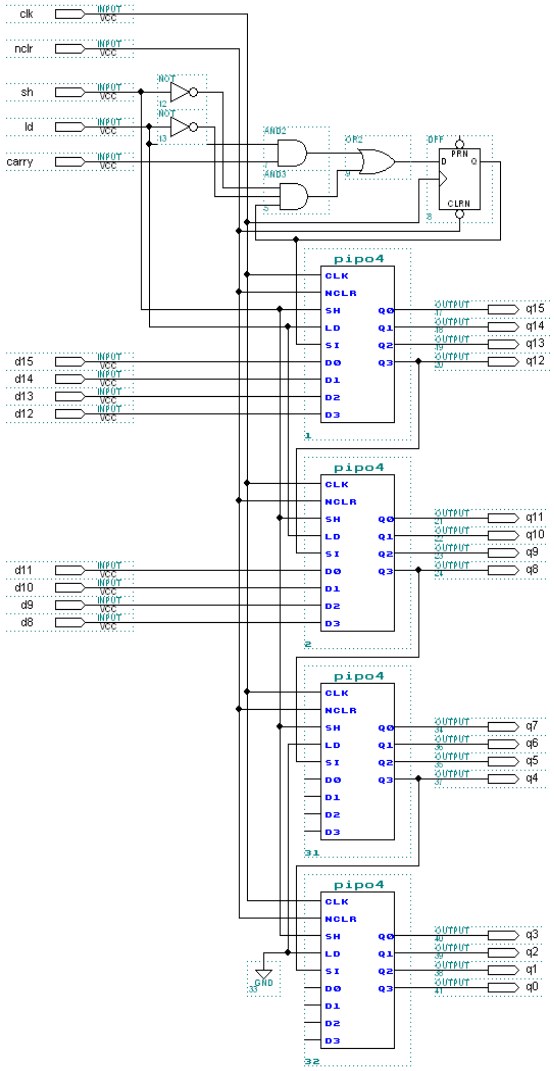

Figure (4) Accumulator

The 4-bit shift register (pipo4) that is used in the accumulator is shown in Figure (5)

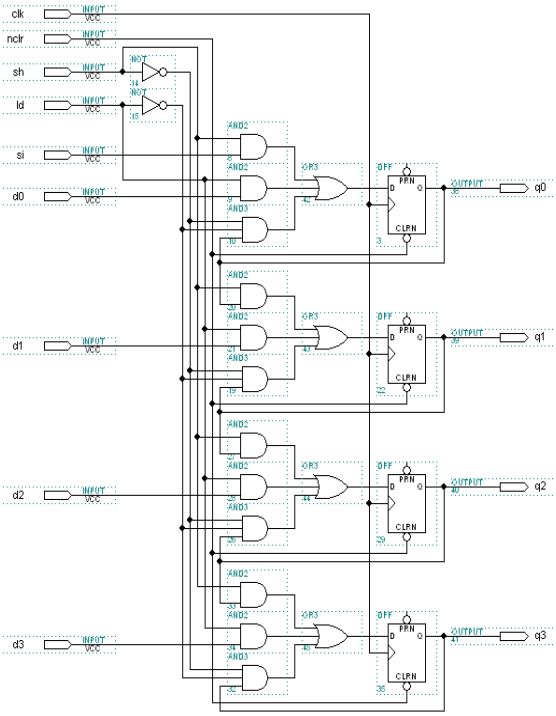

Figure (5) 4-bit Shift register

2.4 The Pulse Generator (clkkey)

Figure (6) shows the pulse generator (clkkey) that generates a starting pulse to drive the 8-bit multiplier. Figure (7) shows its timing diagram. Initially, all flip-flop outputs are cleared. If a START signal (logic 1) is applied to the KEY input, the flip-flop outputs Q1=1 and Q2=D2=$\overline{Q1}$. KEY=1 on the rising edge of clock pulse no. 1. On the rising edge of clock pulse no. 2, the Q1 output remains at 1 and the Q2 output returns to 0 because D2= $\overline{Q1}$. KEY=0. Hence, a pulse appears at the Q2 output.

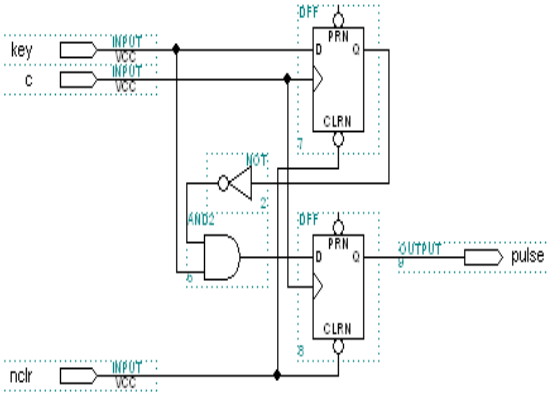

Figure (6) Start pulse generator

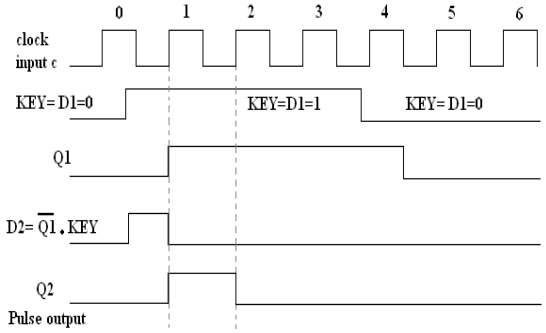

Figure (7) Timing diagram of clkkey

2.5 The Control Pulse Generator

The control pulse generator shown in Figure (8) is used to generate the control pulses for the multiplication process. The START pulse comes from the PULSE output of clkkey. On the rising edge of the third clock pulse, a logical high appears at the DONE output and flip-flop output, and arrives at the EN and CLRN (clear) inputs of the divide-by-16 counter to start counting. The DONE output holds at high for the succeeding clock pulses until the 17^{th} clock pulse arrives. During this counting period, the counter output changes from 0000 to 1111; load operation (LD=1) performs at every even-numbered clock pulse, and shift operation (SH=1) performs at every odd-numbered clock pulse. Therefore, 16 clock pulses are needed for an 8-bit multiplication.

From the logic diagram shown in Figure (8), the Boolean equations for SH and D are given by:

$$D = START \,\&\, \overline{Q} + Q \,\&\, \overline{Q0 \,\&\, Q1 \,\&\, Q2 \,\&\, Q3}$$

$$SH = Q \,\&\, Q0$$

$$LD = Q \,\&\, \overline{Q0}$$

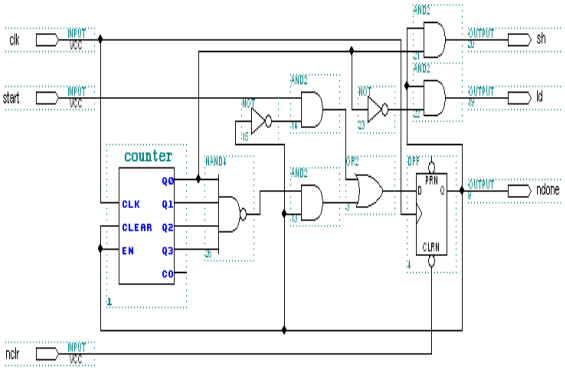

Figure (8) Control pulse generator

When the START pulse appears at the start input, the Q output is clocked to a logical 1, which drives the 4-bit synchronous binary counter to counting up: After 16 clock pulses, $Y = \overline{Q0 \,\&\, Q1 \,\&\, Q2 \,\&\, Q3} = 0$. The Q output returns to 0 to terminate the counting. The timing diagram of the control pulse generator is shown in Figure (9).

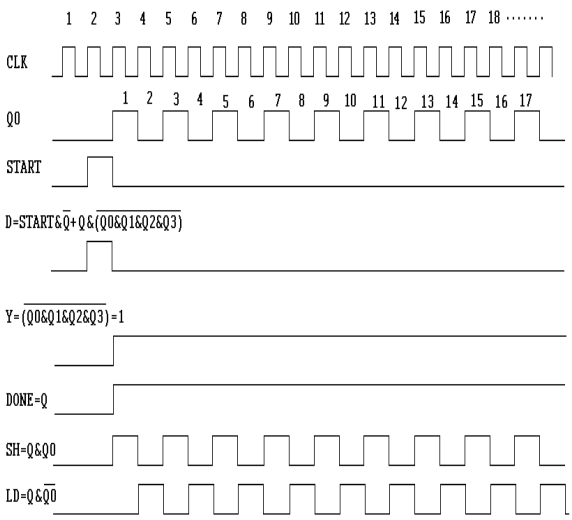

Figure (9) Timing diagram of the Pulse generator

The 8-bit multiplier has eight shift-and-add cycles. Each cycle is divided into shift and addition operations [4]. During shift operation, both the multiplier a0-a7 and the partial product Q0-Q15- are shifted by one position. If the multiplier bit at D0 is 1, addition operation is then performed. The shift operation is performed at every odd-numbered clock pulse when Q0=1 and SH=1, the addition operation is performed at every even-numbered clock pulse when Q0=0 and LD=1. During this operation, the adder output S7-S0 is loaded into the accumulator D8-D15.

3. Experimental Test

- Implement the proposed design with the software MAXPlus2 V10.1, as shown in Figure (10) and save it as graphic editor file format (.gdf).

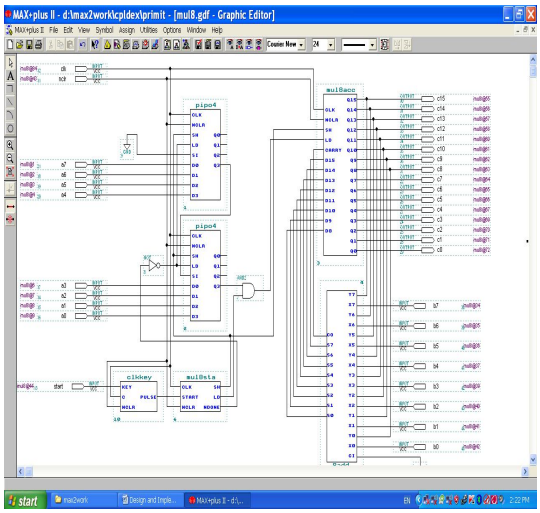

Figure (10) the proposed Design as appear in maxplus2 software

- Compile the design and assign FPGA pins to I/O devices. As shown in Figure (11).

- Download the design into the CPLD / FPGA development system CIC-310 with appropriate downloader software (DNLD82.exe) by using the RS-232 COM Port as shown in Figure (12), this done by the aid of the *Altera* applications note for FPGA devices [5].

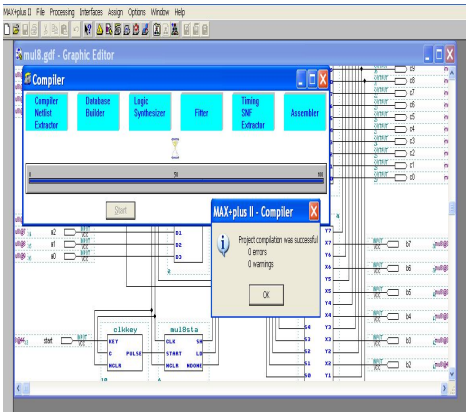

Figure (11) Compiler window

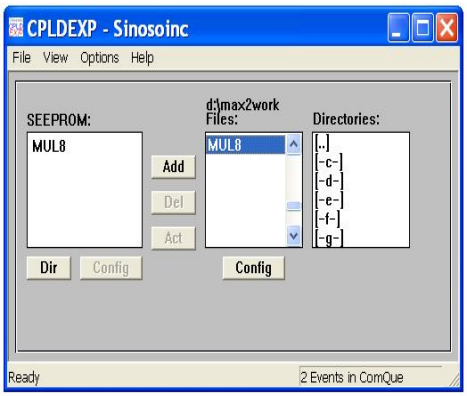

Figure (12) Downloader window

- Set the input values of the multiplicand a7-a0 and the multiplier b7-b0. Apply clock pulses by pushing and releasing the pulse generator equipped with the development system and recording the partial products and the final product and checking its validity.

4. Conclusions

Arithmetic Binary Multipliers can be implemented in FPGAs, providing both efficient usage and high performance. The Altera FLEX8000 family is particularly well suited for this application. There is a range of options available to the user when designing multipliers: speed, area, accuracy and the design method.
During this design and implementation, the following points can be concluded:

- The Altera development tools and the FLEX8000 devices can effectively create multipliers to meet the individual requirements.
- The advantage of the programmable logic over fixed-function logic is that the devices use much less board space for an equivalent amount of logic.
- With programmable logic, designs can be readily changed without rewiring or replacing components.
- A logic design can generally be implemented faster and with less cost with programmable logic than with fixed-function ICs.
- The proposed design can be used to build the decimal multipliers.

References

[1] Floyd T. L. , 2006, "Digital Fundamentals", ninth edition, Pearson Prentice Hall, Page(s) 628-648.

[2] Neto H. C. and Vestias M. P. , 2008 "Decimal Multiplier On FPGA Using Embedded Binary Multipliers", Field Programmable Logic and Applications International Conference, Page(s): 197 – 202.

[3] Beuchat J. L. and Muller J. M. , 2008, "Automatic Generation of Modular Multipliers for FPGA Applications", IEEE Transactions On Computers, Vol. 57, No. 12, Page(s): 1600 – 1613.

[4] Beuchat J. L. and Muller J. M., 2004, "Modulo m Multiplication-Addition: Algorithms and FPGA Implementation," Electronics Letters, vol. 40, no. 11, pp. 654-655.

[5] Altera Application Note 306, for FPGA devices, July, 2004.