Digital Design Lab
EEN 315 Section 3G

Lab 2
Design of a Four-Bit Multiplier

Group 5
Nathan Paternoster
(Partners: Michael Finale, Daniel Bennett)

Sayan Maity, TA

University of Miami

November 19th, 2013

Abstract

An **ALU** (Arithmetic Logic Unit) is a digital circuit that is capable of performing integer arithmetic and logic operations. It is a widely used circuit, found in every CPU and most microprocessors. ALU's can be designed to be very simple and provide only basic addition and subtraction functions or can be much more complex and provide a wide variety of functions. The focus of this project is to design one component of an ALU, a **multiplier** circuit. Our multiplier will be a four-bit by four-bit multiplier. To design this circuit we will use the **add-and-shift** algorithm. This algorithm will multiply two N-bit numbers and will have a processing speed of O(N) = 2N. The circuit will be built using a **full adder**, a digital circuit that is capable of adding two binary numbers.

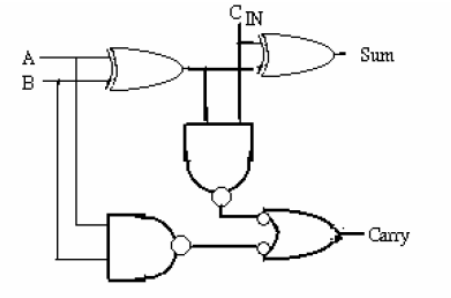
We will simulate a 4-bit multiplier on Altera's Quartus II program and also program the circuit into Altera's UP2 board.

Table of Contents

Overview

A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The circuit will be built using binary adders. The truth table for a *full adder* is shown. In this 1-bit by 1-bit adder, A and B are the inputs and S is the sum. C_O and C_i are the carry-out and carry-in bits. These account for overflow. This table demonstrates the fundamental method of binary addition. The circuit for a full adder is also shown. A *half-adder* will not consider a carry-in.



Full adders can be cascaded to perform n-bit addition. A 3-bit adder is shown. ALU's use these circuits to perform binary *addition*.

Subtraction can be performed in ALU's in nearly the same way when signed integers are represented using 2's complement. In common subtraction: 4 (minuend) – 3 (subtrahend) = 1 (result). In binary, adding 4 (100) to the 1's complement of 3 (~~011~~ → 100) plus 1 (101) will produce 1001. The result must be only three bits so the MSB will be removed and the correct result of 1 (001) is produced.

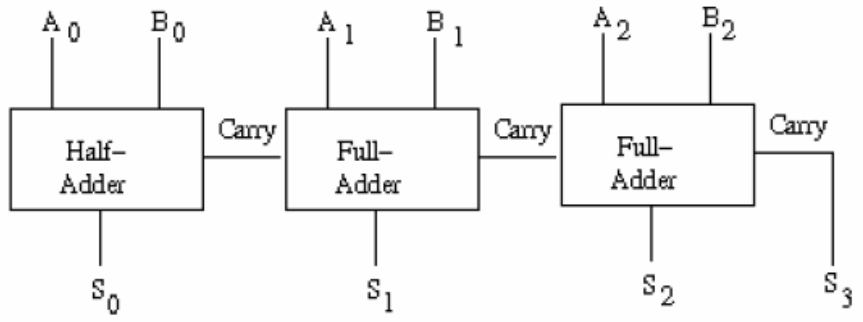


Figure 3: A 3-bit binary adder. The first bit requires only a half adder because a carry-in is not yet considered.

Binary *multiplication* in an ALU may be implemented with the *add-and-shift* algorithm. In this algorithm we consider multiplication to be the summation of

partial sums. The inputs will be n-bits each and the multiplied result will be 2n-bits. In common addition: 2 (*multiplicand*) x 3 (*multiplier*) = 6 (result). The algorithm states that there will be two operations for each n bit. The first will be adding the partial sum to the summation result thus far and the second will be shifting the summation result to the right. When the n bit of the multiplier is 0 the number to be added will be 0. When the n bit of the multiplier is 1 the number to be added is the multiplicand. A 4-bit example of this algorithm is shown.

0101	5	Step 0:			0	0	0	0	0	0	0	0
X1101	x13	Step 1:	Add	+	0	1	0	1				
01000001	65				0	1	0	1	0	0	0	0
			Shift		0	0	1	0	1	0	0	0
		Step 2:	Add	+	0	0	0	0				
					0	0	1	0	1	0	0	0
			Shift		0	0	0	1	0	1	0	0
		Step 3:	Add	+	0	1	0	1				
					0	1	1	0	0	1	0	0
			Shift		0	0	1	1	0	0	1	0
		Step 4:	Add	+	0	1	0	1				
					1	0	0	0	0	0	1	0
			Shift		0	1	0	0	0	0	0	1

Figure 4: The add-and-shift algorithm.

Step 1 considers the LSB of the multiplier. Since it is 1 the number to be added is the multiplicand. The partial sum is then shifted to the right. Step 2 considers the 2nd digit of the multiplier. Since it is 0 the number to be added is 0. The partial sum is then shifted to the right. There will be 4 total steps because the numbers being multiplied are n = 4 bits. The total number of operations is then O(4) = 2(4) = 8.

Objectives

To introduce the design of arithmetic circuits. To understand the advantages of register storage. To introduce the concepts of control logic. To introduce programmable logic.

Equipment

Description	Chip Number	Quantity
Quad 2-Input NAND Gates	7400	2
Dual 4-line to 1-line Multiplexers	74153	1
Synchronous 4-bit Counters	74163	1
4-bit Binary Full Adder	7483	1
4-bit Bi-directional Universal Shift Registers	74194	2
8-bit Bi-directional Shift Registers	74198	1

- *Note: Since we are simulating this circuit, these components are for design purposes only.*

Description

Once the shift-and-add algorithm is understood the circuit can be designed to accomplish both the adding and shifting steps. An 8-bit shift register will be connected back to a 4-bit full adder to keep the result summation in memory. The inputs A_1, A_2, A_3, and A_4 of the full adder will accept the four most significant bits of the 8-bit shift register's output as their input. These pins are Q_E, Q_F, Q_G, and Q_H.

The other inputs of the full adder, B_1, B_2, B_3, and B_4 will accept the next four bits of the input to be added. The carry-in pin of the adder will be grounded because only whole integers will be added in our multiplier. The four output bits of the sum of the adder along with the carry-out bit will be connected with the five most significant input bits of the 8-bit shift register (D-H). The other three input bits of the shift register will receive the next three most significant output bits of the shift register (Q_B, Q_C, and Q_D) in order to keep the 8-bit partial sum in memory.

The four external input bits (A_0, A_1, A_2, and A_3[MSB]) of the multiplicand and the four input bits of the multiplier (B_0, B_1, B_2, and B_3[MSB]) will each be fed into their own 4-bit shift register. These will act as memory storage devices where each step of the add-and-shift algorithm is controlled by a single clock. The control lines (S_0 and S_1) for each register will be connected to the same 'pulse' input. This will be set to '1' for the first clock pulse, which will allow the external parallel inputs to be loaded. After this initial loading the rest of the operations will be done within the circuit and the control lines will be '0'.

The four output bits of the multiplier's shift register will be connected to a 4:1 MUX whose select lines are controlled by a 74163 synchronous counter. The counter's four inputs are all set to '1' and the ENT and ENP enable pins are set to '1'. The CLRN pin's default state is '1' but it will receive a '0' on the first pulse, which is the inverse of the 'pulse' input. The 4:1 MUX will be responsible for selecting each bit of the multiplier to multiply the multiplicand by. This will be accomplished by setting the two select lines of the MUX to be the two least significant bits of the counter's output. The inverse of the third output pin of the counter will be connected the LDN (load) input. When the output of the counter reaches four this bit will be 1 and will feed a 0 back into the load input. This will load the '1111' input back into the counter, causing it to return to start 0 and begin counting up again. This will cause the counter to only count from 0 to 3.

The four outputs of the multiplicand shift register will be fed into AND gates.

The second input to the AND gates will be the output of the 4:1 MUX. The outputs of the four AND gates will be the next 4-bit number to be added to the partial sum. They will be fed into the full adder.

All three shift registers and the counter will be controlled by the same clock. The control lines of the 8-bit shift register will connected to the inverse of the third bit of the counter. When the third bit of the counter is equal to '1' it signifies the end of the add-and-shift algorithm. When the counter has reached 4 all the bits of the multiplier have already been considered.

On the programmable UP2 board, the input pins will be assigned to pins on the board. The four bits of the multiplier and four bits of the multiplicand will be assigned to eight switches. The 'pulse' input will be assigned to a separate switch and the clock input will be assigned to a key. The output will be assigned to eight consecutive LEDs.

Specifications

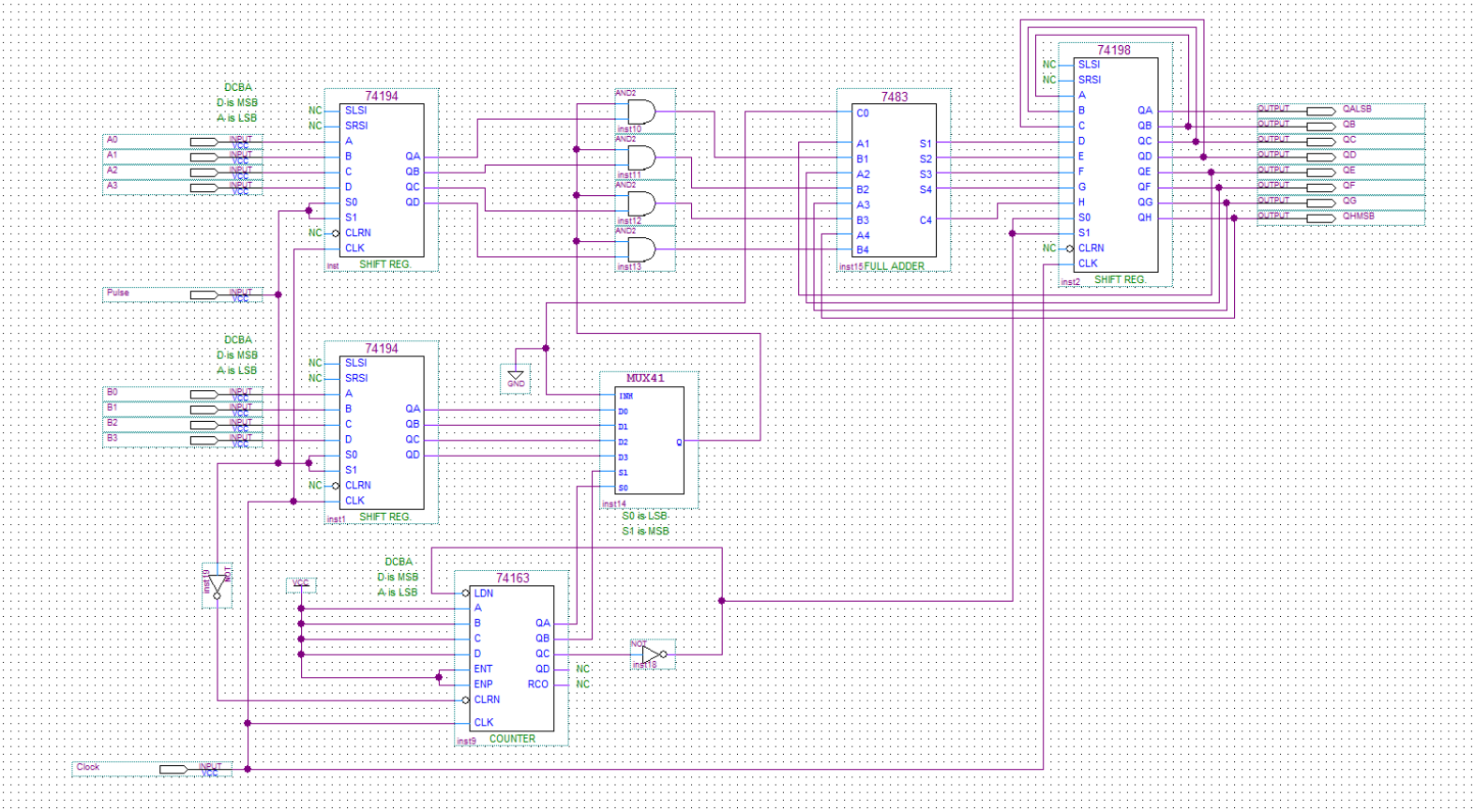
Must design and implement a 4-bit multiplier using the add-and-shift algorithm. Both inputs must be fed from the same switches, toggled by two different push buttons, and stored in two different registers. An additional switch must be used to signal the start of multiplication. The design must include a 'DONE' signal to indicate the end of multiplication.

On the UP2 board, must use SRAM. Must use the on board clock to synchronize the circuit. Must use LEDs to display the output.

Design Synthesis

None

Complete Logic Diagram

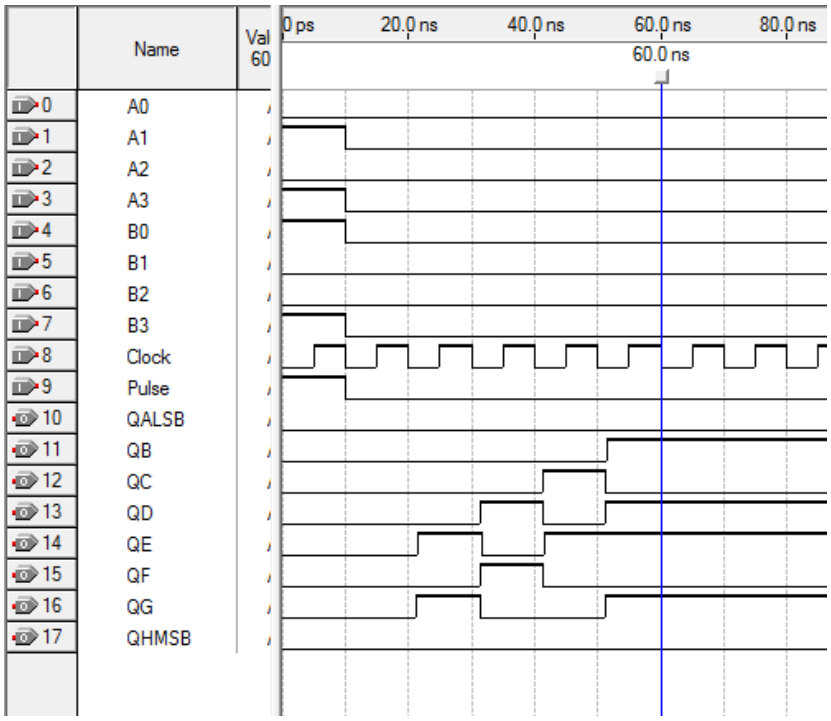


Results and Simulations

Simulation

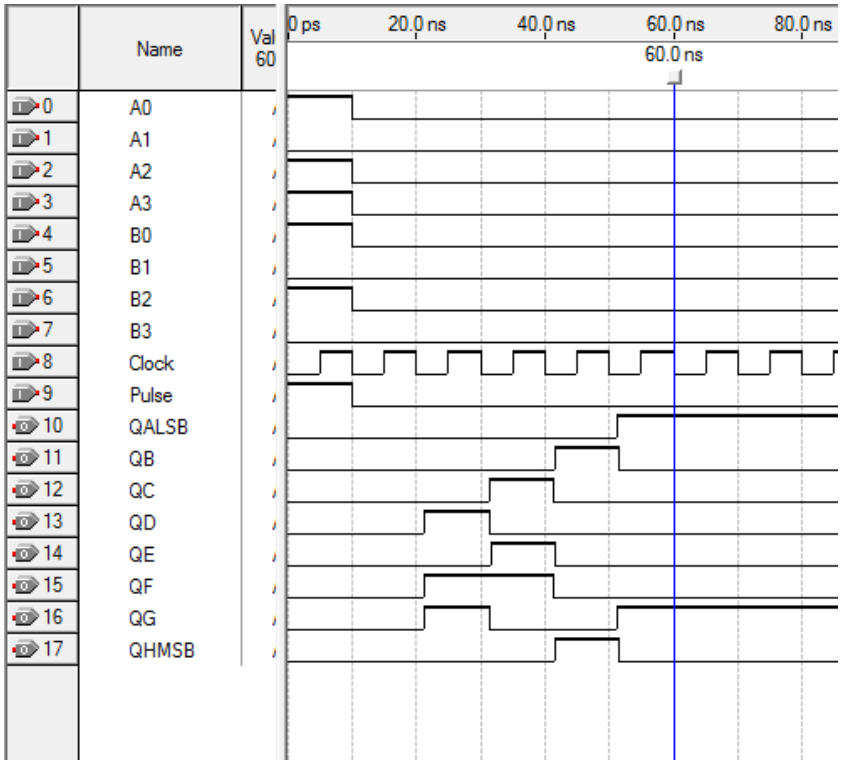
a) 10*9

$A_3A_2A_1A_0$ represent the multiplicand 10 (1010) and $B_3B_2B_1B_0$ represent the multiplier 9 (1001). The pulse is high for the first clock cycle. The final answer is 90 (01011010).



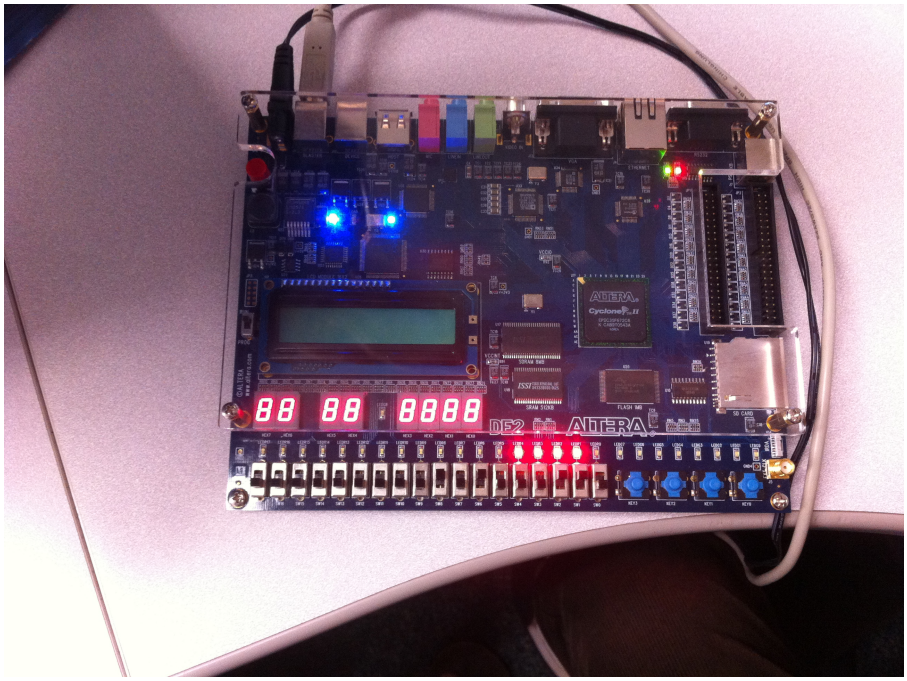
b) 13*5

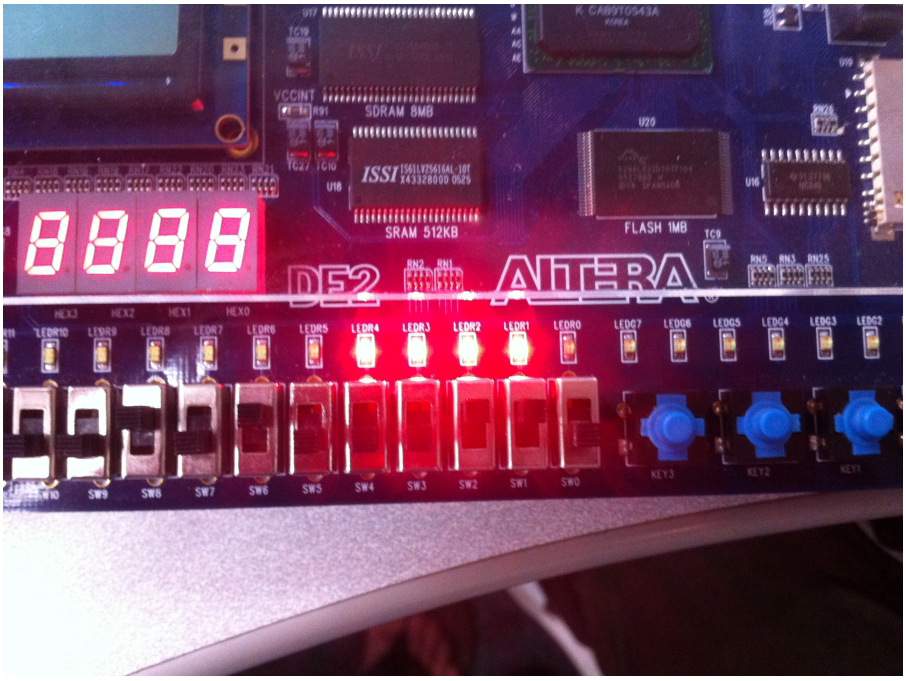
$A_3A_2A_1A_0$ represent the multiplicand 13 (1101) and $B_3B_2B_1B_0$ represent the multiplier 5 (0101). The pulse is high for the first clock cycle. The final answer is 65 (01000001).



Implementation

10 (1010) * 3 (0011) = 30 (00011110)





Answers to the questions in the lab handout

None

Conclusion

This lab provided a fairly comprehensive background to arithmetic circuits. We learned how to design a multiplier circuit and by extension an adder and subtracter circuit. We encountered difficulty when designing the counter and 4:1 MUX's pin connections, but managed to work out the problems after more rigorous testing. I learned about the importance of ALU's in many applications and how to use a programmable board to manually test circuits.

Works Cited

None

Signed OFF

(Included)