Project 2 – Design of a 4-bit Multiplier

Background

- **ALU** – Fundamental building block of a CPU
  - Digital circuit capable of performing arithmetic and logic operations
  - The multiplier in an ALU is built using adders and designed using the ‘add-and-shift’ algorithm

Background

- **Adder** – A digital circuit that performs addition on numbers
- **Full Adder** – An adder that accounts for carry-in and carry out values

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

Background

- Add and Shift Algorithm

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 5: Example of add-and-shift algorithm.

4-Bit Multiplier Theory

- Using two 74194 shift registers, we were able to set the Multiplicand and the Multiplier.
- SLSI and SRSI were kept clear. Modes of operation, S0 and S1, are were the same for both shift registers.
- The outputs of the multiplier were used as inputs for a 4:1 MUX.

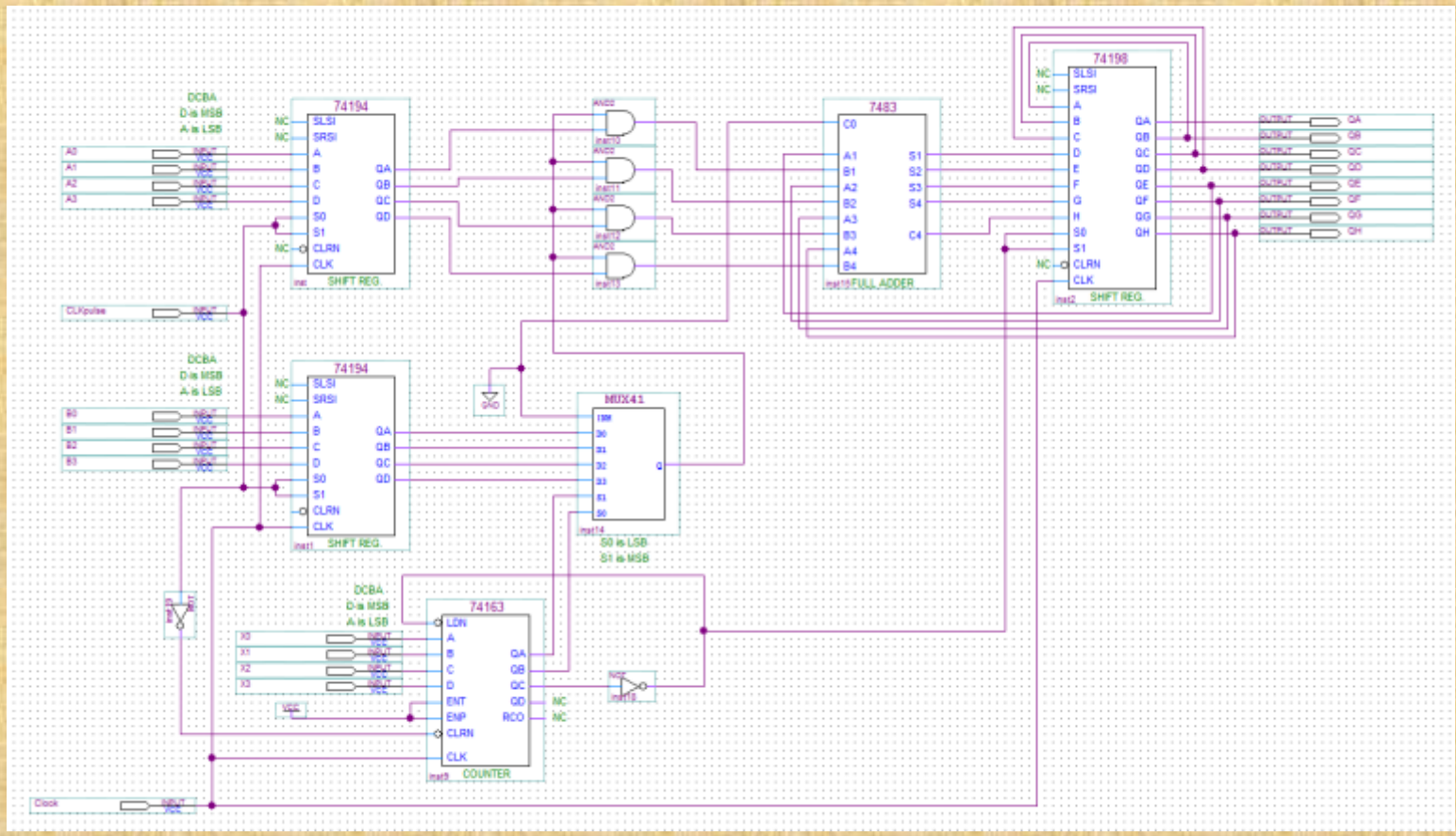
4-Bit Multiplier Theory

- Purpose of the 4:1 MUX is to select which bit the multiplier is multiplying the multiplicand by.
- Accomplished via 74163 synchronous up counter. The two least significant bits were used as select lines to the 4:1 MUX. When the counter reached the fourth clock pulse, the third most significant being inverted, which was fed to the LOAD line. With all outputs ‘1111’, the counter went back to state zero and started over again.
- Counter was cleared on first clock pulse.

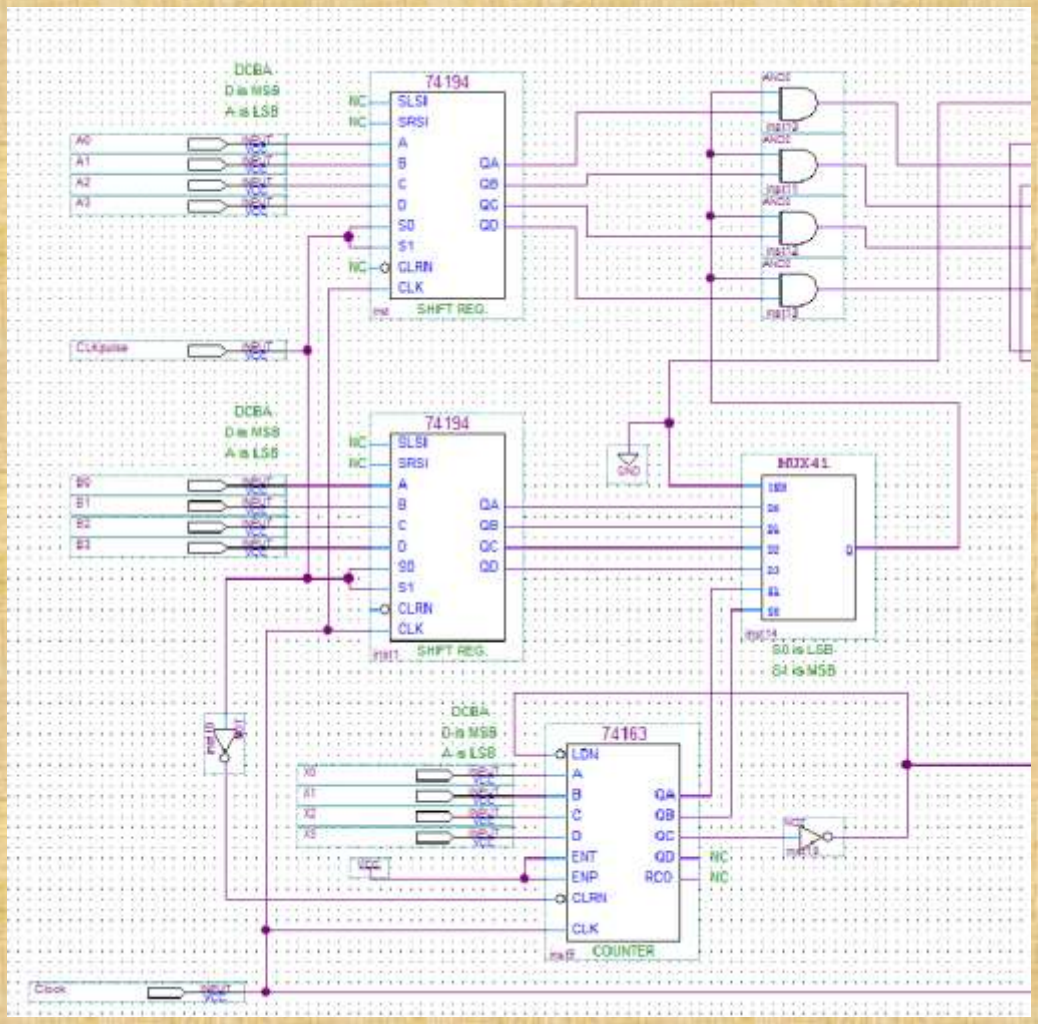
4-Bit Multiplier Theory

- Both multiplicand and multiplier go through four AND gates to the full adder.
- CarryIn for the full adder is grounded to zero.
- The shift register is shifting right. This is caused by our carryOut value set to the most significant bit.

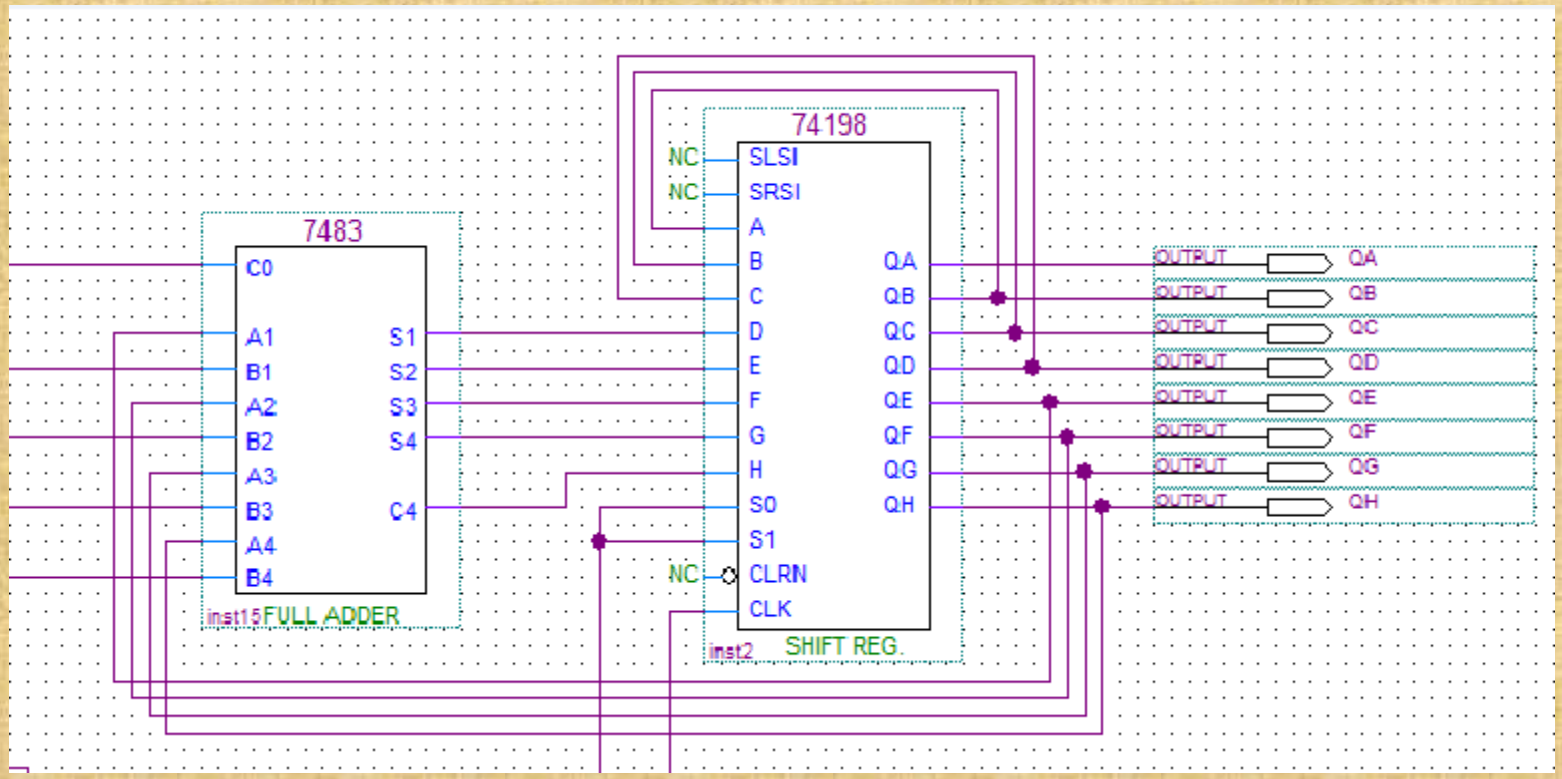
Implementation



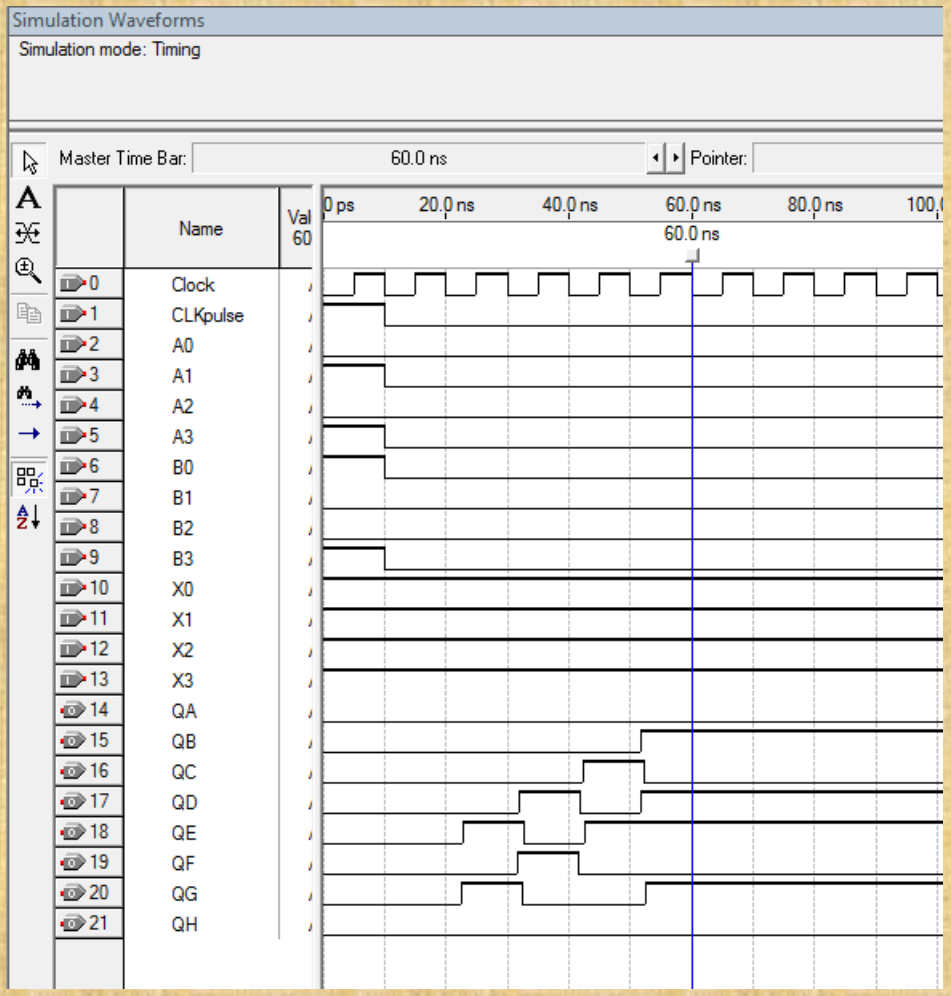
Implementation



Implementation



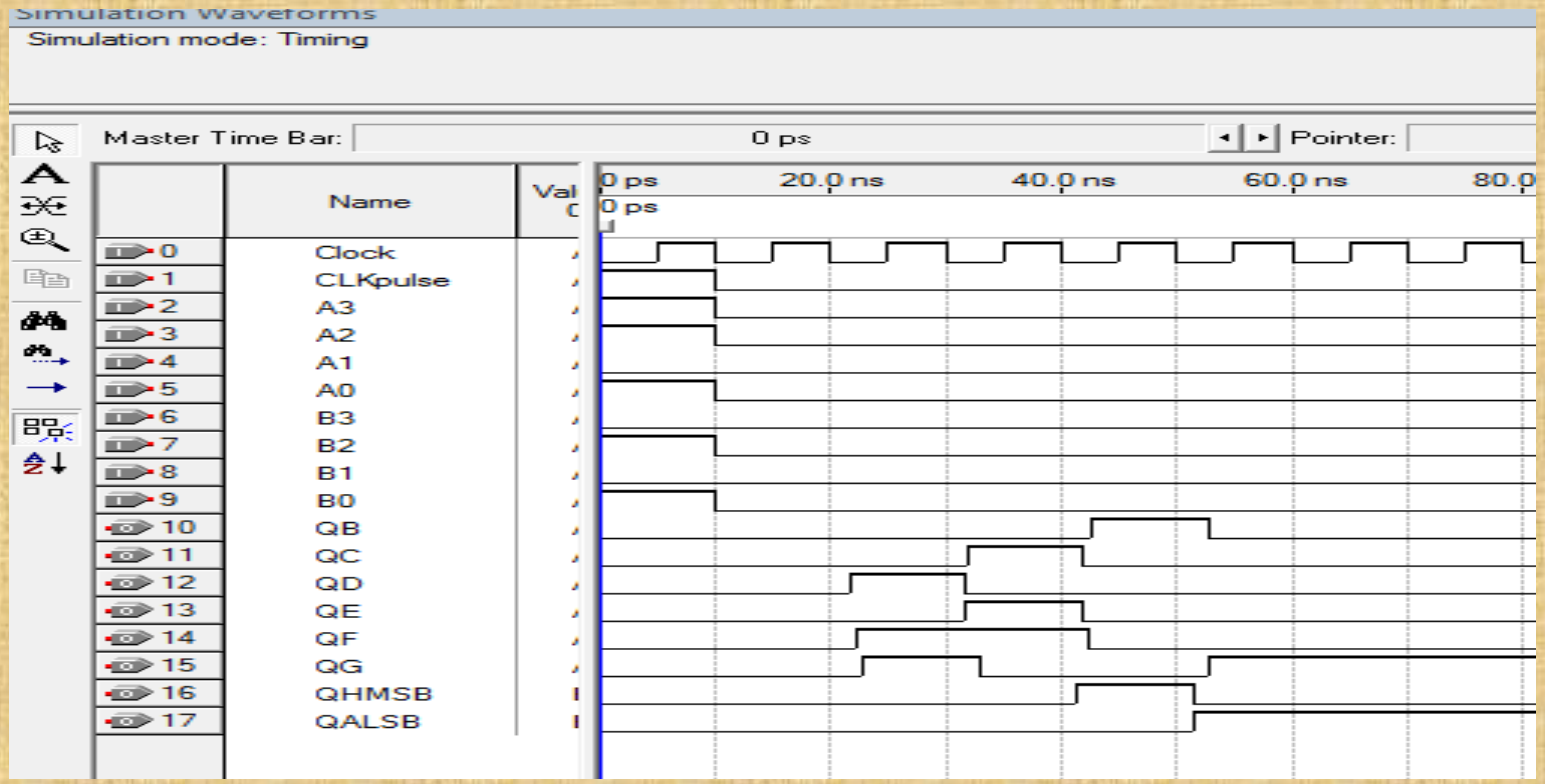
Simulation



10 times 9 where 10 is the multiplicand and 9 is the multiplier

A3 A2 A1 A0= 1010 : ten
B3 B2 B1 B0 = 1001: 9
Result :
QH QG QF QE QD QC QB QA
= 01011010
= 64+16+8+2=90

Simulation


Simulation Waveforms
Simulation mode: Timing
Master Time Bar:
0 ps
Pointer:
Name
Val
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0
0 Clock
1 CLKpulse
2 A3
3 A2
4 A1
5 A0
6 B3
7 B2
8 B1
9 B0
10 QB
11 QC
12 QD
13 QE
14 QF
15 QG
16 QHMSB
17 QALSB