NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING

MINI PROJECT

FINAL REVIEW

PROJECT BY

NIKHIL DWIVEDI 1NH17EC729
RITHWIK SHOME 1NH17EC738
D GOWTHAM 1NH16EC021

8 BIT BOOTH MULTIPLIER

OBJECTIVE

To design a 8 bit booth multiplier that multiplies two binary numbers.

INTRODUCTION

- Booth's multiplication algorithm is a multiplication algorithm that multiplies two binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck College in Bloomsbury, London. Booth's algorithm is of interest in the study of computer architecture.
- Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is the standard technique used in chip design, and provides significant improvements over the "long multiplication" technique.

REQUIREMENTS

- XILINIX 12.1
- SPARTAN 6 FPGA

LITERATURE SURVEY

Following publications were used to complete this project.

- Booth Multiplier: Implementation of Booth's Algorithm using Verilog RTL by Aviral Mittal.
- An 8 Bit by 8 Bit Booth Multiplier by Rick Fenster – 22597217
- Simulation of Booth Multiplier with Verilog-XL November 30, 2011 Robert D'Angelo & Scott Smith Tufts University Electrical and Computer Engineering.

PROCEDURE

- Take M as multiplicand.
- Take Q as multiplier.
- Consider a 1-bit register Q-1which is initialized to 0.
- Consider a register A which is initialised to 0.

CONDITION

1. If $Q_0\,Q_{-1}$ are same i.e. 00 or 11 then, perform arithmetic right shift by 1 bit.
2. If $Q_0\,Q_{-1}$ = 10 then perform

   $A \leftarrow A - M$

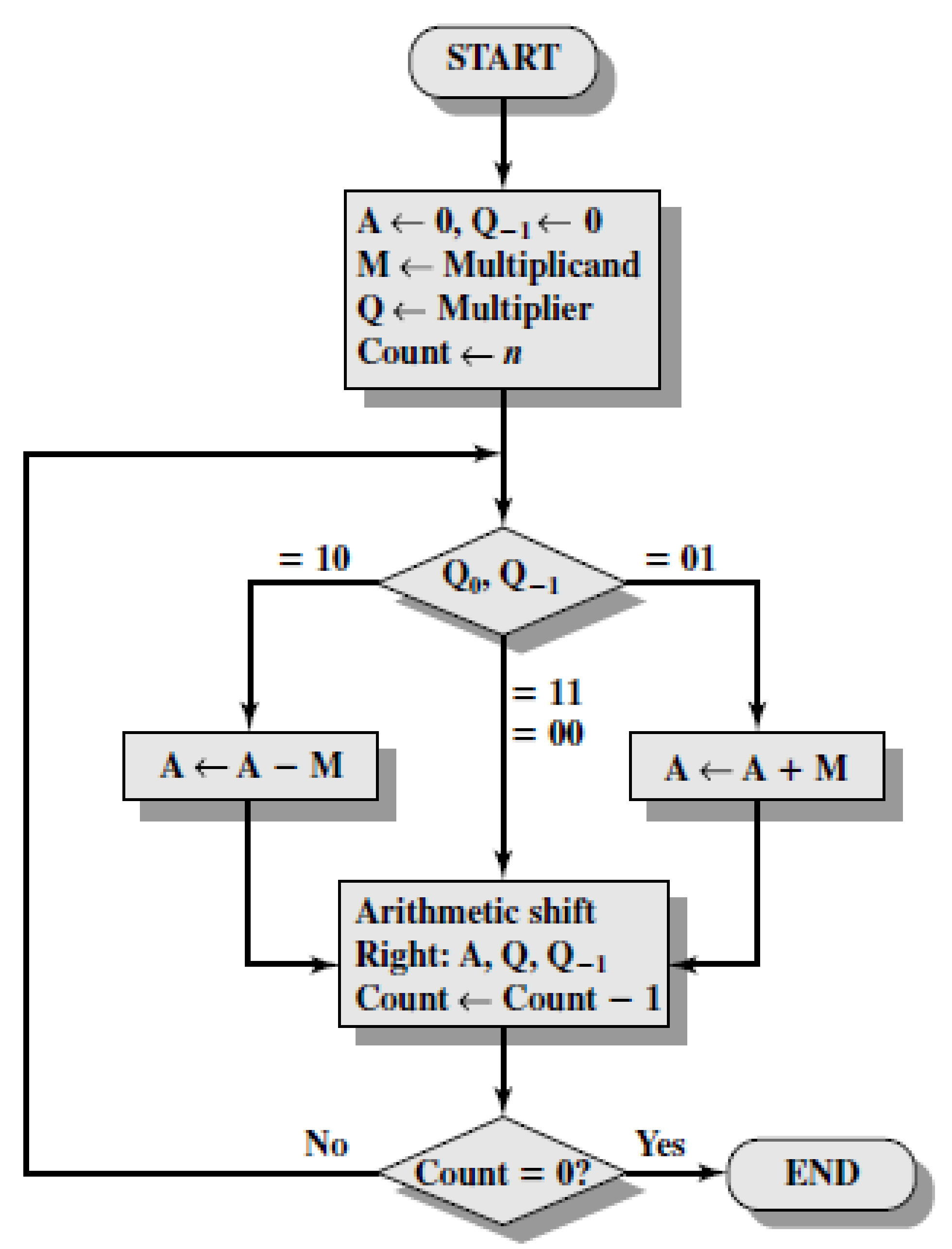
   And then perform arithmetic right shift.
3. If $Q_0\,Q_{-1}$ = 01 then perform

   $A \leftarrow A + M$

   And then perform arithmetic right shift.

Q_0	Q_{n-1}	OPERATION PERFORMED
0	0	Shift
0	1	Add M
1	0	Subtract M
1	1	Shift

FLOW CHART



START
A ← 0, Q−1 ← 0
M ← Multiplicand
Q ← Multiplier
Count ← n
= 10
Q0, Q−1
= 01
= 11
= 00
A ← A − M
A ← A + M
Arithmetic shift
Right: A, Q, Q−1
Count ← Count − 1
No
Count = 0?
Yes
END

For example:

Consider two numbers 6 and 2 and we have to perform their multiplication by using Booth's algorithm.

Here 6 is multiplicand (M) and 2 is multiplier (Q).

Now write 6 and 2 in binary form.

M = 6 = 0 1 1 0

Q = 2 = 0 0 1 0 (Q_3, Q_2, Q_1, Q_0)

Booth's algorithm calculates the product in n steps where n is the number of bits used to represent the numbers.

INITIALISE	A	B	Q_{-1}	OPERATIONS
	0 0 0 0 ↓↘↘↘↘	0 0 1 0 ↘↘↘↘	0	
Step 1.	0 0 0 0	0 0 0 1 ↓	0 ↓	Arithmetic right shift
Step 2.	1 0 1 0 ↓↘↘↘↘ 1 1 0 1	0 0 0 1 ↘↘↘↘ 0 0 0 0	0 1	A←A-M Then shift
Step 3.	0 0 1 1 ↓↘↘↘↘ 0 0 0 1 ↓↘↘↘↘	0 0 0 0 ↘↘↘↘ 1 0 0 0 ↘↘↘↘	1 0	A←A+M Then shift
Step 4.	0 0 0 0	1 1 0 0 In binary, 12 = 1100 Hence 3*2 = 12	0	Arithmetic right shift

3 → Q → Multiplier

7 → M → Multiplicand

	A	Q	Q_{-1}
1.	00000000	00000011	0
	11111001	00000011	0
	11111100	10000001	1
2.	11111100	10000001	1
	11111110	01000000	1
3.	11111110	01000000	1
	00000101	01000000	1
	00000010	10100000	0
4.	00000010	10100000	0
	00000001	01010000	0
5.	00000001	01010000	0
	00000000	10101000	0
6.	00000000	10101000	0
	00000000	01010100	0
7.	00000000	01010100	0
	00000000	00101010	0
8.	00000000	00101010	0
	00000000	00010101	0

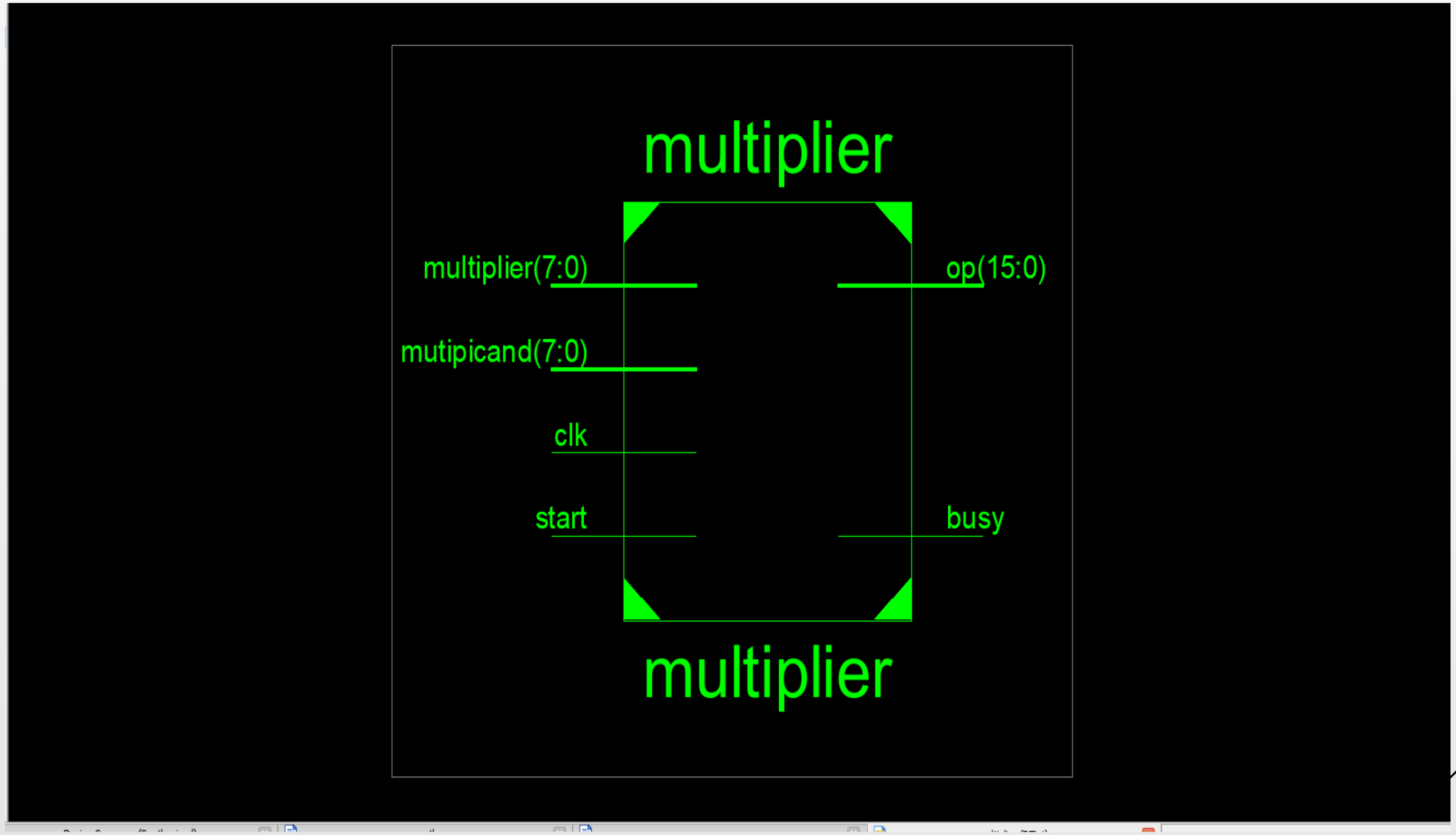
Ans = 21

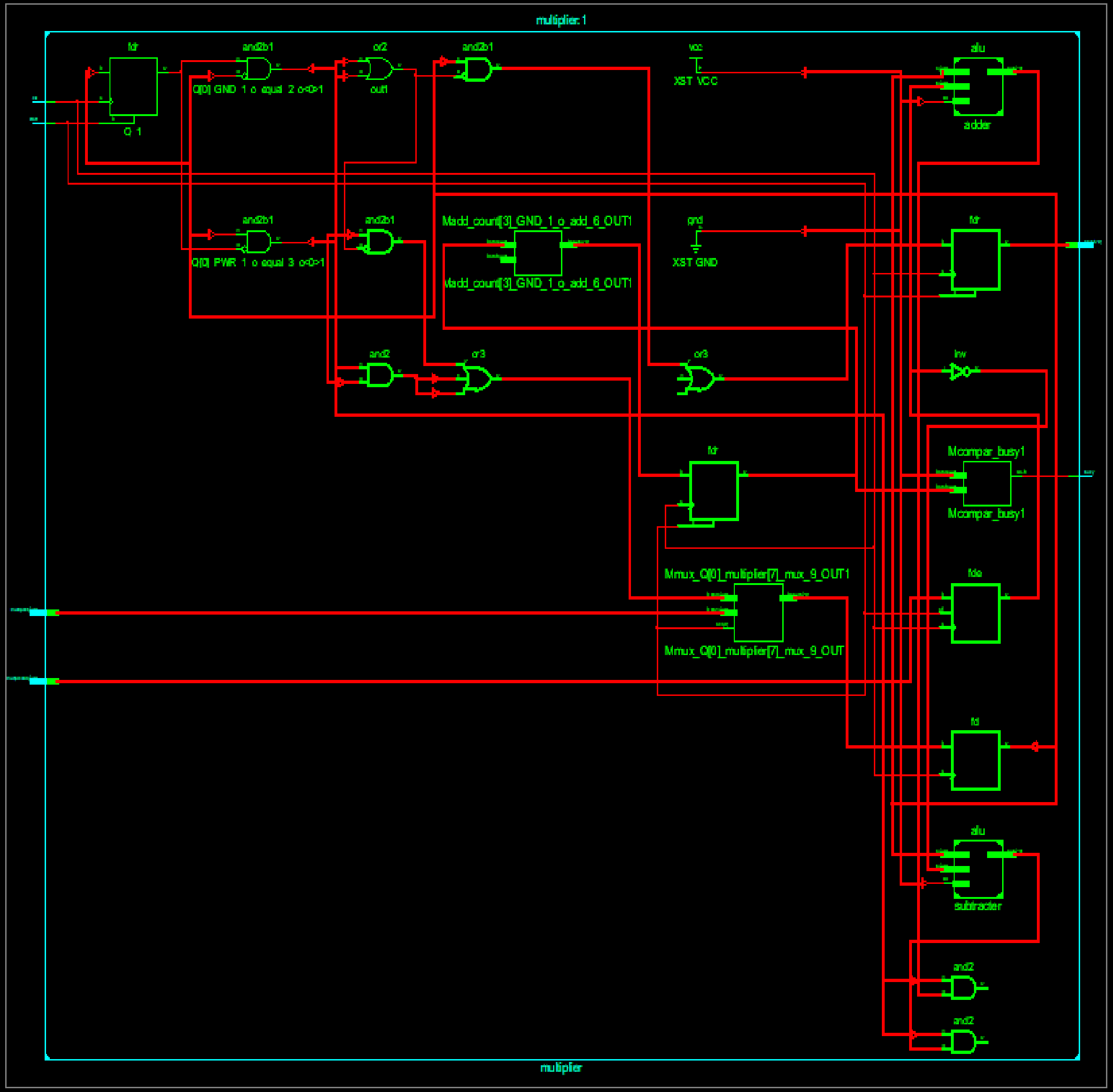
A − M = A + 2's M

```
   00000111
   11111000
 +        1
 ----------
   11111001
```

RESULTS OBTAINED

RTL SCHEMATIC

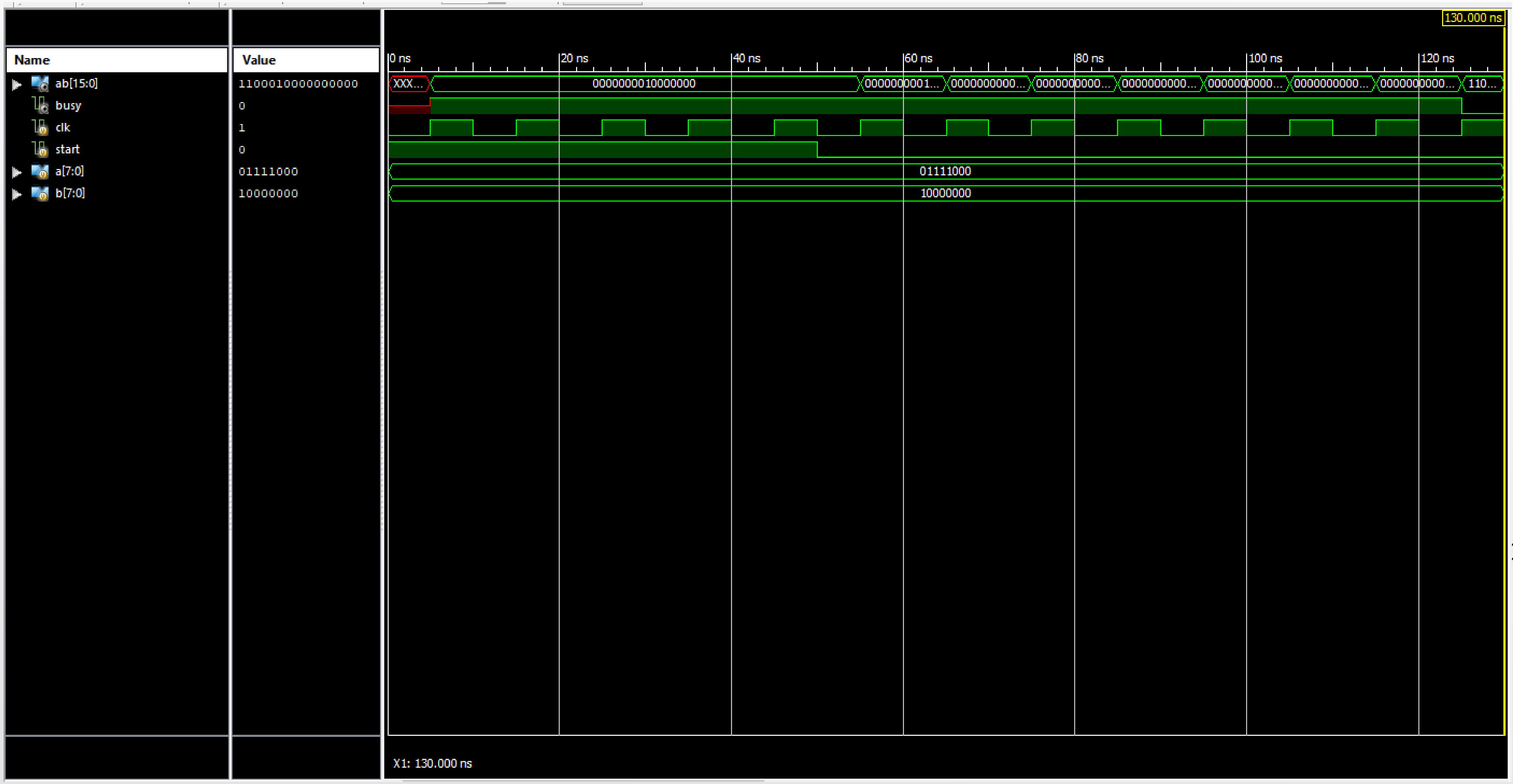



multiplier:1
fdr
and2b1
or2
vcc
alu
XST VCC
adder
Q[0] GND 1 o equal 2 o<0>1
out1
Madd_count[3]_GND_1_o_add_6_OUT1
gnd
XST GND
Q[0] PWR 1 o equal 3 o<0>1
and2
or3
inv
Mcompar_busy1
fde
Mmux_Q[0]_multiplier[7]_mux_9_OUT1
Mmux_Q[0]_multiplier[7]_mux_9_OUT
fd
subtracter
multiplier

6*3

130.000 ns
Name
Value
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
ab[15:0]
0000000000010010
X...
0000000000000110
000000...
111111...
111111...
000000...
000000...
000000...
000000...
0...
busy
0
clk
1
start
0
a[7:0]
00000011
00000011
b[7:0]
00000110
00000110

128*128



130.000 ns
Name
Value
ab[15:0]
1100010000000000
busy
0
clk
1
start
0
a[7:0]
01111000
b[7:0]
10000000
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
XXX...
0000000010000000
0000000001...
0000000000...
110...
01111000
10000000
X1: 130.000 ns

APPLICATIONS

- It is the standard technique used in chip design.
- It is used in ALU unit of computer to calculate multiplication of signed and unsigned number in binary form. This actually tells that how computer internally calculate signed number multiplication.
- It is a technique that allows for smaller, faster multiplication circuits.

FUTURE SCOPE

- Radix-4 binary multiplier is more faster than radix-2 multiplier but its speed and efficiency can be increased significantly in the future.
- The algorithm can be used for both positive and negative integer so it is used very widely as compared to other multipliers.

THANK YOU