STUDY OF MULTIPLICATION ALGORITHMS IN COMPUTERS

Mayadevi P.A

[1] *Assistant Professor, Computer Science & Engineering Department, Viswajyothi college of Engineering and Technology, Kerala, India*

ABSTRACT

Data are manipulated by arithmetic instructions in a digital computer to produce the solution of complex engineering problems. These arithmetic instructions perform arithmetic calculations by using an arithmetic processor and which is the part of processor unit. From the four basic operations (addition, subtraction, multiplication, division…) it shows how to perform the basic multiplication operation in signed- magnitude representation.

Keyword: - *Signed-magnitude, Multiplicand, Multiplier, Partial Product, Booth etc….*

1. INTRODUCTION

Computer architecture deals with the various functional units and their interaction to satisfy the user requirements. But the interconnection o various hardware components to form a computer system is known as computer organization [1]. It is the study of various arithmetic algorithms and the implementation of them with various digital hardwire. Here consider only one basic operation such as multiplication for different types of data. The fixed point binary data are represented as:

- Sign magnitude representation
- Signed 2's compliment representation

The basic operations that are required for multiplication are to perform successive addition and shifting the data in registers to the right by one bit.. A counter register is used to store a number which is equal to the number of right shifts[1].

2. ALGORITHM FOR MULTIPLICATION

In sign magnitude representation two fixed point binary numbers are multiplied by successive shift and add operations.

An example of such an illustration is

```
13 X          1 1 0 1        Multiplicand
 9            1 0 0 1        Multiplier
---------------------------
              1 1 0 1
            0 0 0 0
          0 0 0 0            +
        1 1 0 1
---------------------------
127     1 1 1 0 1 0 1        Product
```

Looking at the least significant bit of the multiplier first, if it is 1, copied down the multiplicand otherwise zeros are copied down. In successive lines, the numbers copied are shifted one position to the left. Finally the product is formed as a sum of these numbers[1].

The sign of the multiplier and the multiplicand determines the sign of the product. If the signs are identical, the sign of the product is positive, otherwise it is negative[1].

Hardware Implementation

The equipments /hardware required for multiplication is as follows:

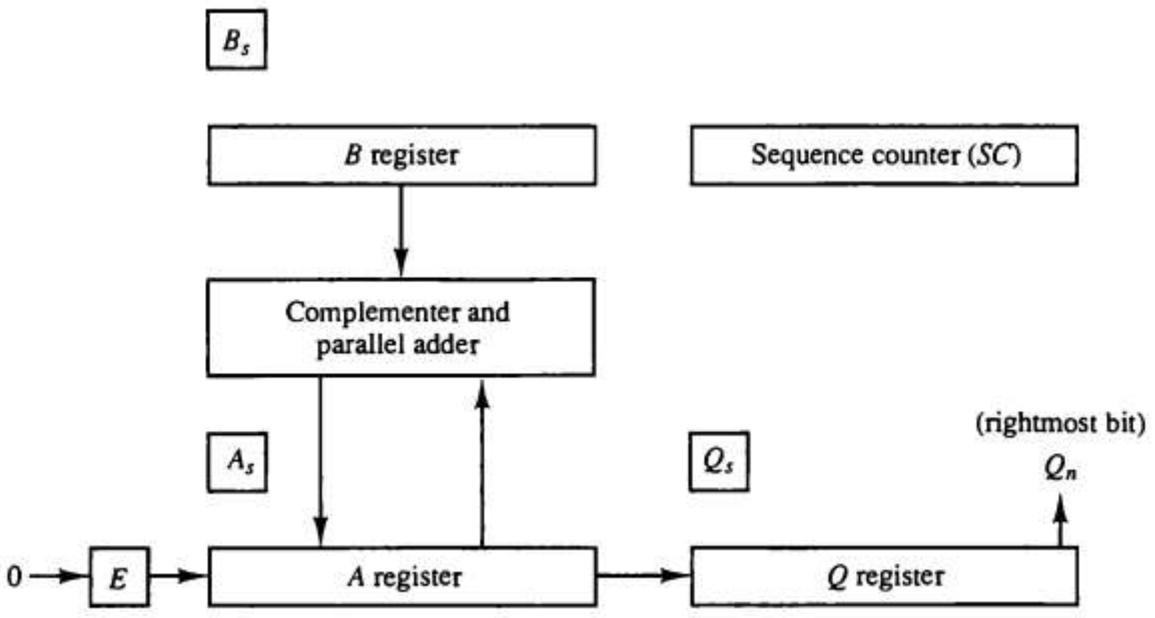

Fig-2.1: Hardware Configuration for Multiplication

The multiplier is stored in register Qr and the sign of the multiplier is stored in Qsn. The multiplicand and its sign are stored in registers Br and Bsn respectively. The counter Register (CR) is a register that contains a value equal to the number of bits in the multiplier. The Accumulator (AC) register is initially cleared as zero. After forming each partial product the counter is decremented by 1. When the counter value reaches zero, the product is available in registers AC, Qr and the sign of the product is available in Qsn[1].

Algorithm

The Algorithm for multiplication is given.

Initially the multiplicand and the multiplier are stored in respective registers and compare their signs. Then consider the lower order bit of the multiplier, if it is 1, the multiplicand is added with the present partial product in AC. If it is 0 there is no change in the partial product. Then shift register EAC,Qr one bit position to the right. Decrement the counter register by one and check the value, if it is not zero, repeat the process and forms a new partial product. When it reaches 0, the final product is available in registers AC and Qr[2].

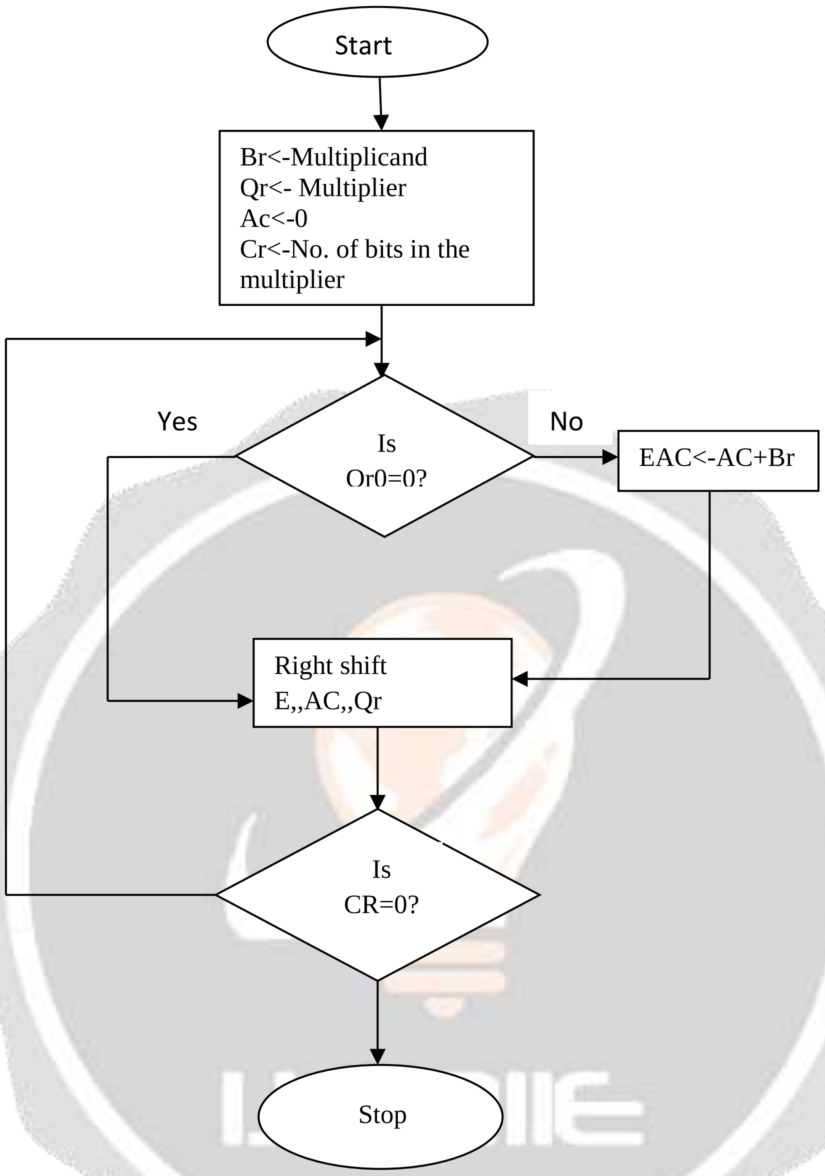

Fig-2.2 signed number multiplication

Example :

Multiplicand=1101	E	A	Q	SC
Multiplier=1001	0	0000	1001	100
Qn=1 Add B		<u>1101</u>		
Partial product		1101		
Shift right EAQ	0	0110	1100	011
Qn=0 Shift right EAQ	0	0011	0110	010

Qn=0 Shift right EAQ	0	0001	1011	001
Qn=1, Add B		1101		
Partial Product		1110		
Shift right EAQ	0	0111	0101	000

Final product in AQ = 1110101

2.2 Booth's Algorithm

It is the procedure for multiplication of binary numbers that are represented in signed 2's compliment form. The basic principle of Booth's algorithm states that, if the multiplier contains strings of zeros there is no need of addition of multiplicand to the partial product but just shifting. The string of one's in the multiplier with bit weight 2^p to weight 2^r treated as $2^{p+1}-2^r$[5].

In booth algorithm, examine the multiplier bits and according to that add or subtract the multiplicand to the partial product or left unchanged then perform the shift operation[5]. These are based on the following rules[1]:

1. If the least significant 1 is encountered from a string of 1s in the multiplier then subtract the multiplicand from the partial product.

2. If the first 0 (previous 1) is encountered from a string of 0s in the multiplier then the The multiplicand is added to the partial product.

3. If the multiplier bit is identical to the previous bit there is no need to change the partial product.

<u>Hardware Implementation[1]</u>

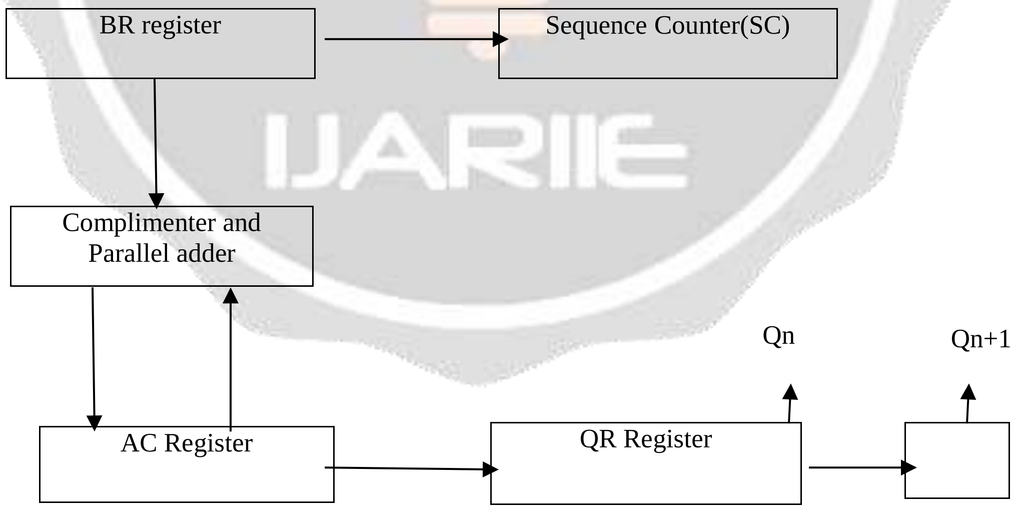

Fig-2.3: Hardware Configuration for Booth's Multiplication

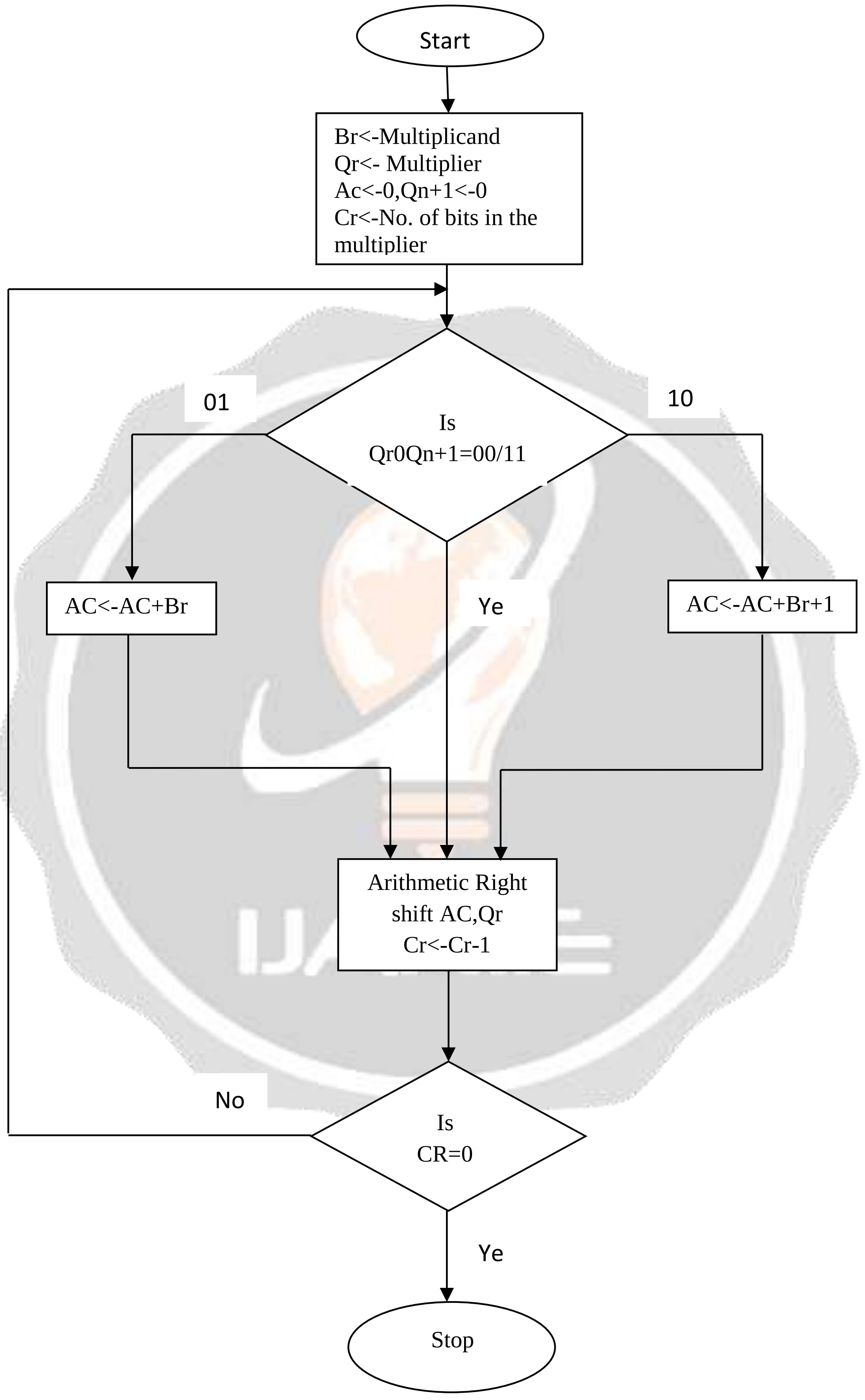

Fig-2.4 Booth's Multiplication Algorithm

Initially the Accumulator register AC and the bit Qn+1 are cleared to 0 and the counter register is set as a number equal to the number of bits in the multiplier. Then compare the two bits of the multiplier Qn0 and Qn+1 with 10 if they are equal means that the first 1 is encountered from the string of 1s so it performs the subtraction of the multiplicand from the partial product. If the bits are same as 01 means that first 0 is encountered from the string of 0s so it performs addition of multiplicand to the partial product. If the bits are 00 or 11 there is no need to change the partial product. Then perform arithmetic shift operation. The arithmetic shift is to be performed over the partial product and the multiplier including the bit Qn+1[6].

Example :

		BR=11000	AC	QR	Qn+1	SC
Qn	Qn+1	BR+1=01000	00000	10100	0	101
0	0	arshr	00000	01010	0	100
0	0	arshr	00000	00101	0	011
1	0	Subtract BR	01000			
			01000			
		Arshr	00100	00010	1	110
0	1	Add BR	11000			
			11100			
		Arshr	11110	00001	0	001
1	0	Subtract BR	01000			
			00110			
		Arshr	00011	00000	0	000

Product = 1100000

3. CONCLUSIONS

Generation and accumulation of partial products are the two basic operations in multiplication process. So to speed up the multiplication process it requires to reduce number of partial product and to increase the accumulation[6]. If smaller the number of partial products it reduces the complexity and the accumulation of the partial product. Apply these two solutions simultaneously by using Booths multiplication algorithm**.** So it is the fastest multiplication algorithm as compared with signed-magnitude representation[6]. In booth's algorithm positive or negative numbers are represented in its 2's compliment form and the sign bits are not stored separately from other bits[1].

4. REFERENCES

[1]. Computer System Architecture, Third Edition by M.Morris Mano
[2]. Computer Architecture from microprocessor to supercomputer by Behrooz Parhami
[3]. Harpreet Singh Dhillon and Abhijit Mitra "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic" in International Journal of Computational and Mathematical Sciences 2:2 2008.
[4]. Purushottam D. Chidgupkar and Mangesh T. Karad "The Implementation of Vedic Algorithms in Digital Signal Processing".
[5]. Aviral Mittal "Booth Multiplier Implementation of Booth‟s Algorithm using Verilog RT
[6]. *(PDF) Different Multiplication Algorithm and Hardware*.