1

32-BIT CSD MULTIPLIER

Sidhant Priyadarshi

***Abstract*—This is the implementation of 32-bit CSD multiplier in which multiplicand remains the Binary input and multiplier into booth is CSD representation of binary multiplier Binary input and multiplier into booth is CSD representation of binary multiplier complement.So,when we convert binary into CSD -1 is represented as 1 which I its 2's compliment.Multiplication is caried out using booth technique and simulation is done using Xilinx ISE Design Suit 14.7.**

I. INTRODUCTION

CSD is an elegant method to implement digital multipliers in a more efficient way. This article reviews the basics of CSD and its implementation.There are a number of algorithms to convert a binary number to CSD. The simple approach suitable for paper-and-pencil conversion from unsigned binary representation to CSD is to search the number from LSB to MSB, find a string of 1s followed by a 0.

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed 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.

II. METHODS

II.1 BOOTH MULTIPLIER ALGORITHM[1]

Find 3 ×-(4), with m = 3 and r =-(4), and x = 4 and y = 4:m = 0011,

refer fig:1

-m = 1101, r = 1100

A = 0011 0000 0

S = 1101 0000 0

P = 0000 1100 0

Perform the loop four times:

P = 0000 1100 0. The last two bits are 00.

P = 0000 0110 0. Arithmetic right shift.

P = 0000 0110 0. The last two bits are 00.

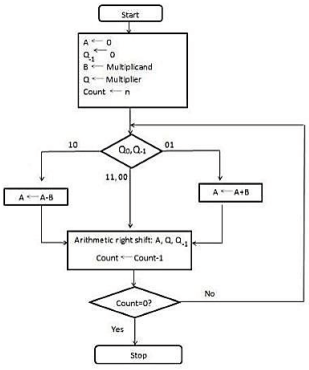

Fig. 1. BOOTH ALGORITHM

P = 0000 0011 0. Arithmetic right shift.

P = 0000 0011 0. The last two bits are 10.

P = 1101 0011 0. P = P + S.P = 1110 1001 1.

Arithmetic right shift.

P = 1110 1001 1. The last two bits are 11.

P = 1111 0100 1. Arithmetic right shift.

The product is 1111 0100, which is-(12).

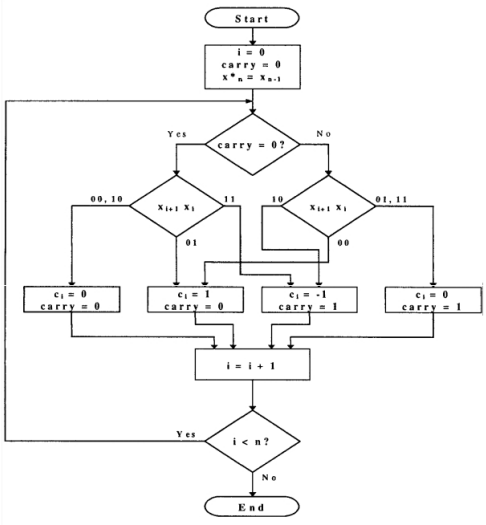

Fig. 2. BINARY TO CSD

II.2 BINARY TO CSD[2]

CSD representation of 1010111.

Use fig:2

In the first iteration, we find 0111 and replace it with 100-1,and we obtain 101100-1.
In the second iteration, we find 011 and replace it with 10-1.Hence, we have 110-100-1.
Now, there is another string 11 To find the CSD equivalent of this iteration, we need to add a 0 to the leftmost position of 110-100-1
Obviously, this will not change the value of this number, but it will allow us to replace
011 with 10-1
Finally, we obtain the CSD representation as
10-10-100-1.

II.3 CSD MULTIPLIER

To get the output using CSD representation:
II.3.A we need to convert the binary multiplier into CSD multiplier and multiplicand remains in binary refer section II.2.
II.3.B follow the steps as shown in section II.1.

III. VERILOG IMPLEMENTATION

III.1 VERILOG CODE TO CONVERT BINARY TO CSD

```
module csdigit(
input signed[14+2:0]a,
output reg signed [14+1:0]c
);
integer i;
reg carry=0;
always @(a or i or carry)
begin
for(i=0;i¡16;i=i+1)
begin
if(carry == 0)
Begin
if(a[i+1],a[i]==2'b00 ——a[i+1],a[i]==2'b10)
begin
c[i] = 1'b0;
carry = 0;
end
else if(a[i+1],a[i]==2'b01)
begin
c[i] = 1'b1;
carry = 0;
end
else if(a[i+1],a[i]==2'b11)
begin
c[i] = -1'b1;
carry = 1;
end
else
begin
c = a;
carry = 0;
end
end
else
begin
if(a[i+1],a[i]==2'b01 ——a[i+1],a[i]==2'b11)
begin
c[i] = 1'b0;
carry = 1;
end
else if(a[i+1],a[i]==2'b00)
begin
c[i] = 1'b1;
carry = 0;
end
else if(a[i+1],a[i]==2'b10)
begin
c[i] = -1'b1;
carry = 1;
end
else
begin
c = a;
carry = 0;
end
end
end
end
endmodule
```

III.2 VERILOG CODE FOR CSD MULTIPLIER USING BOOTH TECHNIQUE

```
module bmul
( input signed [15:0] m,
input signed [16:0] q,
output reg signed[31:0] p
);
wire signed [15:0]qin;
csdigit csd(.a(1'b0,q[14],q[14:0]),.c(qin));
reg [15:0]mn;
integer i;
reg qn1;
reg [1:0]t;
initial begin
p = 32'h0;
qn1 = 0;
end
```

```
always @(m,qin,p)
begin
for(i=0;i¡16;i=i+1)
begin
t = qin[i],qn1;
mn = -m;
case(t)
2'b10 :begin
p[31:16] = p[31:16] + mn;
p = p¿¿1;//right shift 1 bit
p[31] = p[30];
qn1 = qin[i];
end
2'b01 : begin
p[31:16] = p[31:16] + m;
p = p¿¿1;//right shift 1 bit
p[31] = p[30];
qn1 = qin[i];
end
2'b00,2'b11:begin
p = p¿¿1;//right shift 1 bit
p[31] = p[30];
qn1 = qin[i];
end
default : begin end
endcase
end
if(qin==16'h8000)//2power16/2 = (8000)base16 its 2s complement is also same thats y we need to add this
begin p = -p;
end
end
endmodule
```

IV. RESULTS

IV.1.BINARY TO CSD CONERSION:

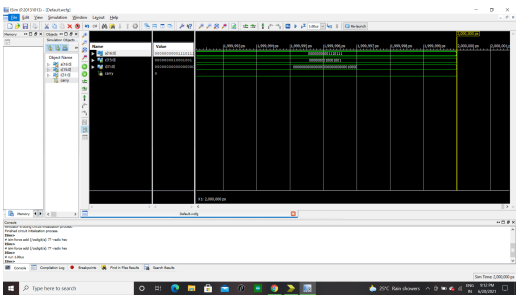

Fig. 3. BINARY TO CSD O/P

IV.2.CSD MULTIPLIER:

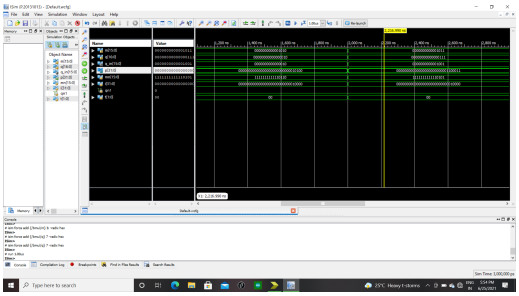

Fig. 4. CSD MULTIPLIER O/P

V. CONCLUSIONS

The basic idea to introduce CSD was to reduce the numbers of one's and increase the numbers of zero's of the binary number. While we do Multiplication number of one's in the input increases the number of steps i.e.; add and shift .To avoid the the method of CSD is introduced ,Which helped us in decreasing the hardware requirement of adder circuit and more efficient and fast way of multiplication.

REFERENCES

[1] D. Chandel, G. Kumawat, P. Lahoty, V. V. Chandrodaya, and S. Sharma, "Booth multiplier: Ease of multiplication," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 326–330, 2013.

[2] R. Guo and L. S. DeBrunner, "A novel fast canonical-signed-digit conversion technique for multiplication," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 1637–1640.