Booth's Algorithm:-

→ It is a multiplication algorithm that multiplies 2 signed binary numbers in 2's complement notation.

Conventional multiplication method:-

→ Shift-and-add Multiplication. For n-bit multiplication, we iterate n times.

→ Add '0' (or) the Multiplicand to the 2n-bit partial product (depending on the next bit of the multiplier)

→ Shift the 2n-bit partial product to the right.

→ i.e, we need n-additions and n-shift operations.

Ex:

```
    0 1 0 1 1 0
        1 0 1
  -------------
    0 1 0 1 1 0
  0 0 0 0 0 0  x
0 1 0 1 1 0  x  x
-----------------
      Result
```

→ Booth's algorithm is an improvement, where we avoid the additions when we have consecutive 0's (or) 1's are in the Multiplier.

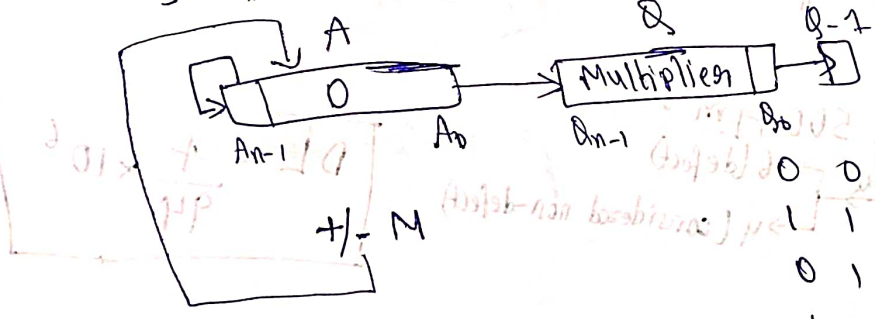
⇒ Faster process:-

Algorithm:-

→ We inspect (Q_i, Q_{i-1}) of the multiplier:-

- If the bits are same (00 or 11), we only shift the partial product.
- If the bits are 01, we do an addition and then shift.
- If the bits are 10, we do a subtraction and then shift.

→ Initially Q_{i-1}, i.e Q_{-1} is assumed to be zero.



0 0 } only shift.
1 1 }

0 1 → Add and shift.

1 0 → Subtract and shift.

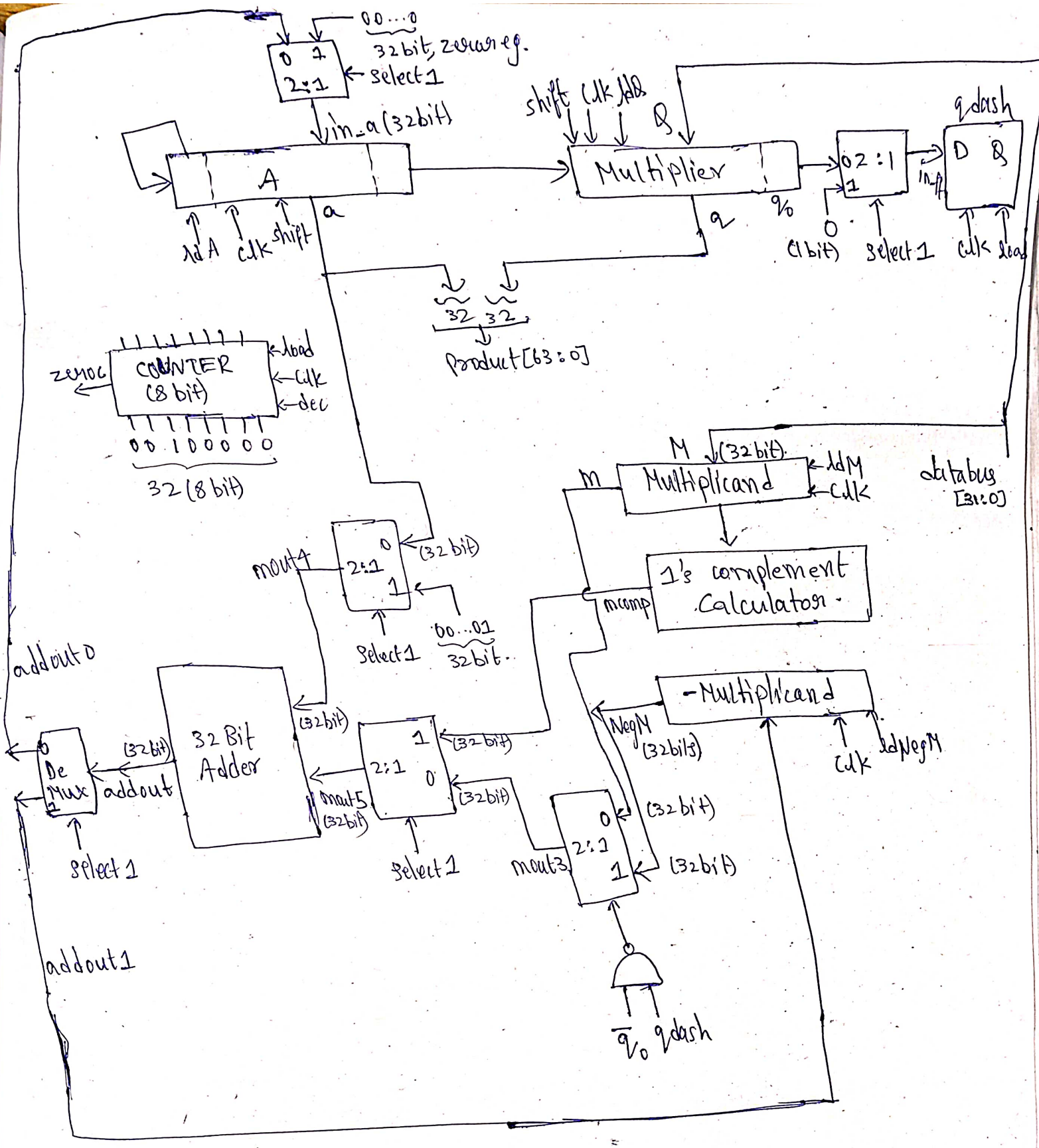
Example:- $(-10) \times (13)$ → Assume 5 bit numbers.

$M : (10110)_2$

$-M : (01010)_2$

$Q : (01101)_2$

A	Q	Q-1	
0 0 0 0 0	0 1 1 0 1	0	→ Initialization
0 1 0 1 0	0 1 1 0 1	0	→ A = A − M } step 1
0 0 1 0 1	0 0 1 1 0	1	→ shift
1 1 0 1 1	0 0 1 1 0	1	→ A = A + M } step 2
1 1 1 0 1	1 0 0 1 1	0	→ shift


00...0
32 bit, zeroes eg.
0
1
2:1
select1
vin_a(32bit)
A
ldA
clk
shift
a
shift clk ldB
Q
Multiplier
q
q%
2:1
0
1
(1 bit)
select1
qdash
D
Q
clk
load
32
32
Product[63:0]
COUNTER
(8 bit)
zeroc
load
clk
dec
00 100000
32 (8 bit)
M
(32 bit)
ldM
clk
Multiplicand
m
databus
[31:0]
mout4
2:1
0
1
(32 bit)
select1
00...01
32 bit.
1's complement
Calculator
mcomp
addout0
32 Bit
Adder
(32bit)
De
Mux
addout
select1
addout1
1
2:1
0
(32 bit)
mout5
(32bit)
select1
-Multiplicand
NegM
(32bits)
clk
ldNegM
mout3
(32bit)
(32bit)
$\overline{q}$
q
qdash

State Diagram :