Mode = 0
No addition is desired
"NO ADD" A
Shift 1 bit on right

Mode = 1
Add operation of multiplicand content with left shift register

We have now

As status register
Now shift bits to right
Now Status is

```
 0 0 1 0  
```

Now M = 0
No 'Add', but Shift Right, 
Operation is not needed 
As 3 bit operation is 
completed

Final Status

```
 0 0 1 0  1 0 0
```

: Final Number is 10100

Decimal Numbers

```
Multiplicand 101 → 5
Multiplier 100 → 4
```

Multiplication Output = 5x4 = 20
= 10100
ALGORITHM of Baugh-Wooley

MULTIPLIER

X and Y are the Multiplicand and Multiplier 2's Complemented Numbers. Then

\[ X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \]

\[ Y = -y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} y_j 2^j \]

Product \[ P = X \cdot Y \]
\[ P = X_{n-1} Y_{n-1} 2^{2n-2} + \sum_{l=0}^{n-2} \sum_{j=0}^{n-2} x_i y_j 2^{i+j} \]

\[ -X_{n-1} \sum_{j=0}^{n-2} y_j 2^{n+j-1} - Y_{n-1} \sum_{i=0}^{n-2} x_i 2^{n+i-1} \]

To Avoid Subtractor Cell use, we represent -ve quantities as

\[ -X_{n-1} \sum_{j=0}^{n-2} y_j 2^{n+j-1} = X_{n-1} \left[ -\frac{2^{n-2}}{2} + 2^{n-1} \sum_{j=0}^{n-2} y_j 2^{n+j-1} \right] \]
Block cell - 3

Block cell - 4

Block cell - 5
FIVE BASIC OPERATION BLOCKS

In

Baugh - Wooley Multiplier

\[ x_i \]
\[ \text{Block Cell - 1} \]

\[ x_i \cdot y_j \]

\[ x_i \cdot y_j \]

\[ x_i \cdot y_j \]

\[ \text{Block - cell - 2} \]
In this Multiplier we need

1. 2's Complement Generator

2. AND GATES to get Partial Products

3. Full Adders

To save area and also to improve speed

nxm bit Multiplier is arranged in an ARRAY-FORM.
In Booth's Multiplier we recode 2's Complement Nos.

Since we use Binary Nos, we observe that:

- j-long sequence of 1's is equivalent to
  \[ \Rightarrow (j-1) \text{ long sequence of 0's.} \]

Replacement of 1's by 0's reduce Partial Product Terms.
Weakness of Booth's Simple Recoding Technique:

It may create more 1's than initial number of 1's in the sequence after Recoding. This happens when in the number we have sparse 1's. Ex.

85 → 001010101 → Number
01,-1,1,-1,1,-1 → Booth Coded
MODIFIED BOOTH RECODING

Here \( x_0 - x_{n-1} \) are the Original Numbers

\( y_0 - y_{n-2} \) are Modified Booth Recoded Numbers.

Example here uses Radix-4 Scheme

What we learn: – 3 bits inspected & 2 bits get eliminated
<table>
<thead>
<tr>
<th>Inspected Bits</th>
<th>Recoded Bits</th>
<th>Recoded Digit times Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i$</td>
<td>$x_{i-1}$</td>
<td>$x_{i-2}$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example

Multiplicand $A$ \[0\ 0\ 1\ 1\ 0\ 1\] \[13\]
Multiplier $x$ \[1\ 1\ 1\ 0\ 1\ 0\] \[-6\]

Booth Recoding of $x$

\[\begin{array}{c}
1\ 1\ 1\ 0\ 1\ 0 \\
\hline
1\ 1\ 1\ 0\ 1\ 0 \\
1\ 1\ \boxed{-1}\ 1\ 0\ 1\ 0 \\
1\ 1\ 0\ \boxed{-1}\ 1\ 0\ 1\ 0 \\
1\ 1\ 0\ 0\ \boxed{-1}\ 1\ 0\ 1\ 0 \\
1\ 1\ 0\ 0\ 0\ \boxed{-1}\ 1\ 0\ 1\ 0 \\
\end{array}\]

0.A - 1A - 2A
Multiplicand A 0 0 1 1 0 1
Multiplier  x 1 1 1 0 1 0

OPERATION 0 -A -2A

Initial 0 0 0 0 0 0 0
Add (-2A) + 1 0 0 0 1 0 1

Shift 2 bits 1 1 0 1 0 0 1 0 1
Add (-1A) + 1 1 0 0 0 1 1 (0 0)

Neglect 1 1 0 1 1 0 0 0 1

Decimal
13
x -6
-78
Last: 1 0 1 1 0 0 0 1
Shift 2 bits: 1 1 1 0 1 1 0 0 0 1
Add (0.4) + 0 0 0 0 0 0 0 0 0 0

Sign Bit: 1 1 | 1 0 1 1 0 0 0 1 → -78
BOOTH ENCODER

Diagram of Booth Encoder with input bits $b_i, b_{i+1}, b_{i-1}$ and output bits $2x_i, x_i, M_i$. The diagram includes logic gates to represent the encoding process.
Partial Product Generator

\[ a_i \times x_i \]
\[ a_{i-1} \times 2x_i \]

Modified Version

Shifter uses Pass Gates & Buffer.