Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8351

DIGITAL PRINCIPLES AND SYSTEM DESIGN
(COMMON TO CSE AND IT)

UNIT No. 2

2.2 BINARY MULTIPLIER

BINARY MULTIPLIERS

- Binary computer does exactly the same multiplication as decimal numbers do, but with binary numbers.
- In binary encoding each long number is multiplied by one digit (either 0 or 1), and that is much easier than in decimal, as the product by 0 or 1 is just 0 or the same number.
- Therefore, the multiplication of two binary numbers comes down to calculating partial products (which are 0 or the first number), shifting them left, and then adding them together (a binary addition, of course):

```
        1011  (this is 11 in binary)
      x 1110  (this is 14 in binary)
      ======
        0000  (this is 1011 x 0)
       1011   (this is 1011 x 1, shifted one position to the left)
      1011    (this is 1011 x 1, shifted two positions to the left)
+    1011     (this is 1011 x 1, shifted three positions to the left)
   =========
    10011010  (this is 154 in binary)
```

- This is much simpler than in the decimal system, as there is no table of multiplication to remember: just shifts and adds.

- This method is mathematically correct and has the advantage that a small CPU may perform the multiplication by using the shift and add features of its arithmetic logic unit rather than a specialized circuit.

- The method is slow, however, as it involves many intermediate additions,these additions are time-consuming. Faster multipliers may be engineered in order to do fewer additions; a modern processor can multiply two 64-bit numbers with 6 additions , and can do several steps in parallel.

- The second problem is that the basic school method handles the sign with a separate rule ("+ with + yields +", "+ with − yields −", etc.).

Modern computers embed the sign of the number in the number itself, usually in the two's complement representation. That forces the multiplication process to be adapted to handle two's complement numbers, and that complicates the process a bit more.

Unsigned numbers

- For example, suppose we want to multiply two unsigned eight bit integers together: *a*[7:0] and *b*[7:0].
- We can produce eight partial products by performing eight one-bit multiplications, one for each bit in multiplicand *a*:

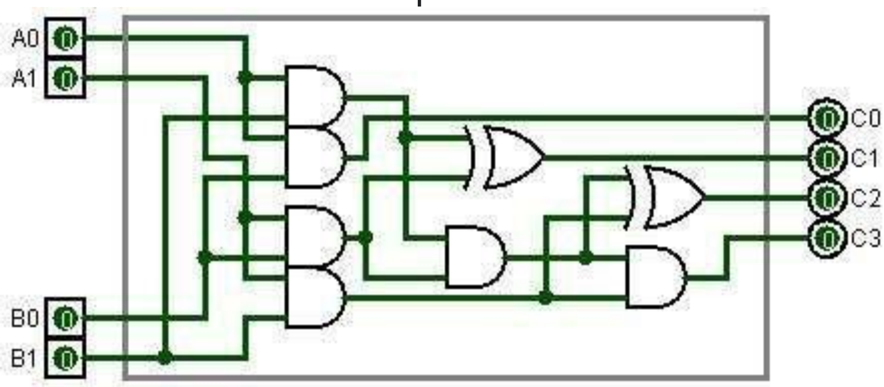
p0[7:0] = a[0] × b[7:0] = {8{a[0]}} & b[7:0]
p1[7:0] = a[1] × b[7:0] = {8{a[1]}} & b[7:0]
p2[7:0] = a[2] × b[7:0] = {8{a[2]}} & b[7:0]
p3[7:0] = a[3] × b[7:0] = {8{a[3]}} & b[7:0]
p4[7:0] = a[4] × b[7:0] = {8{a[4]}} & b[7:0]
p5[7:0] = a[5] × b[7:0] = {8{a[5]}} & b[7:0]
p6[7:0] = a[6] × b[7:0] = {8{a[6]}} & b[7:0]
p7[7:0] = a[7] × b[7:0] = {8{a[7]}} & b[7:0]

- where {8{a[0]}} means repeating a[0] (the 0th bit of a) 8 times .
- To produce our product, we then need to add up all eight of our partial products, asshown here:

```
                     p0[7] p0[6] p0[5] p0[4] p0[3] p0[2] p0[1] p0[0]
                   + p1[7] p1[6] p1[5] p1[4] p1[3] p1[2] p1[1] p1[0] 0
                 + p2[7] p2[6] p2[5] p2[4] p2[3] p2[2] p2[1] p2[0] 0    0
               + p3[7] p3[6] p3[5] p3[4] p3[3] p3[2] p3[1] p3[0] 0     0     0
             + p4[7] p4[6] p4[5] p4[4] p4[3] p4[2] p4[1] p4[0] 0     0    0     0
           + p5[7] p5[6] p5[5] p5[4] p5[3] p5[2] p5[1] p5[0] 0     0     0     0   0
         + p6[7] p6[6] p6[5] p6[4] p6[3] p6[2] p6[1] p6[0] 0     0     0    0     0     0
       + p7[7] p7[6] p7[5] p7[4] p7[3] p7[2] p7[1] p7[0] 0     0     0     0     0   0      0
         ---------------------------------------------------------------------------------------
P[15] P[14] P[13] P[12] P[11] P[10]  P[9]  P[8]  P[7]    P[6] P[5]   P[4]  P[3] P[2]  P[1]
                                            P[0]
```

- In other words, *P*[15:0] is produced by summing *p*0, *p*1 << 1, *p*2 << 2, and so forth, to produce our final unsigned 16-bit product.

Example circuit



2 bit by 2 bit binary Multiplier