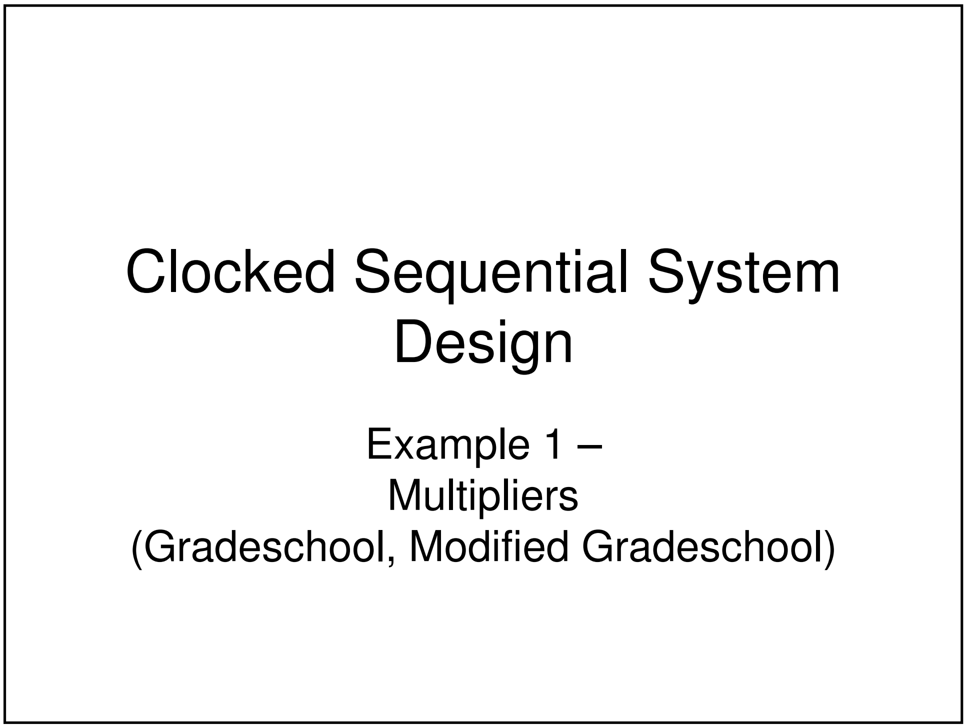
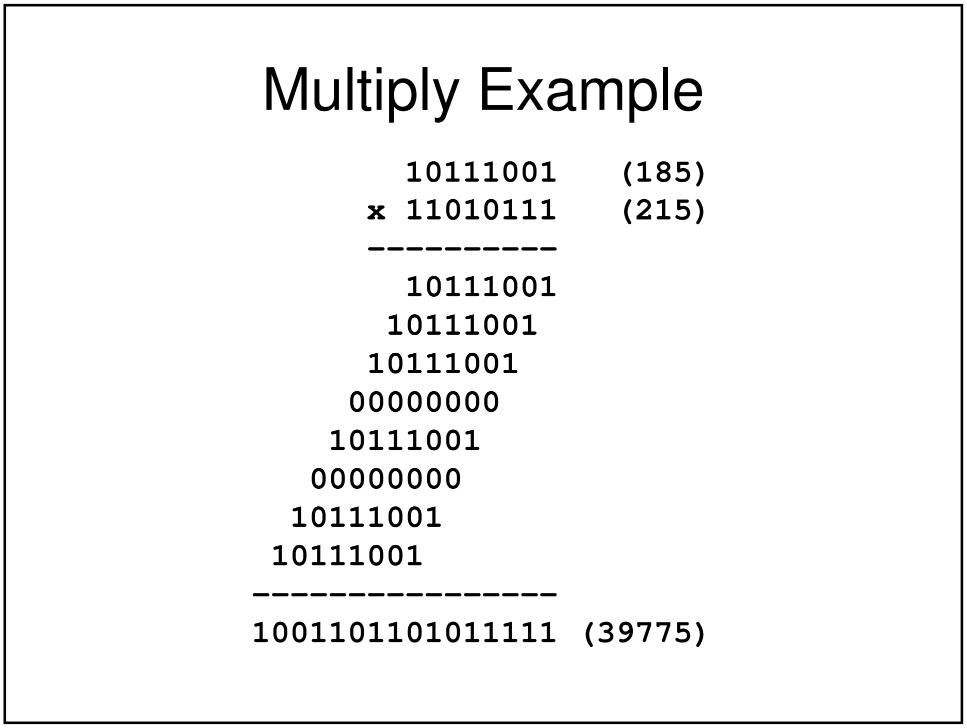


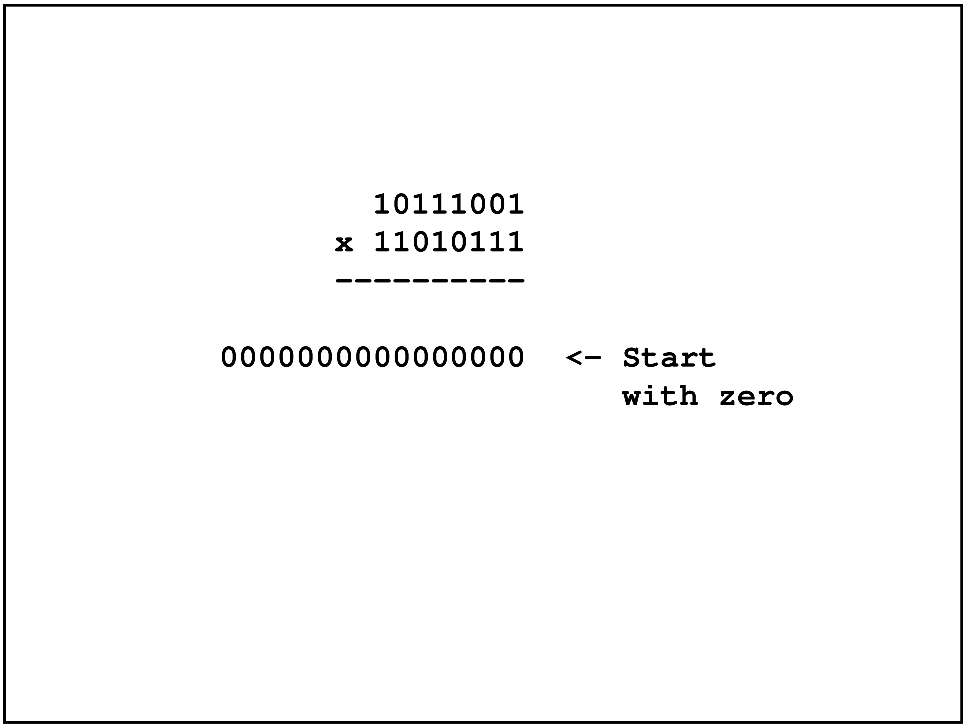
Clocked Sequential System Design

Example 1 –
Multipliers
(Gradeschool, Modified Gradeschool)

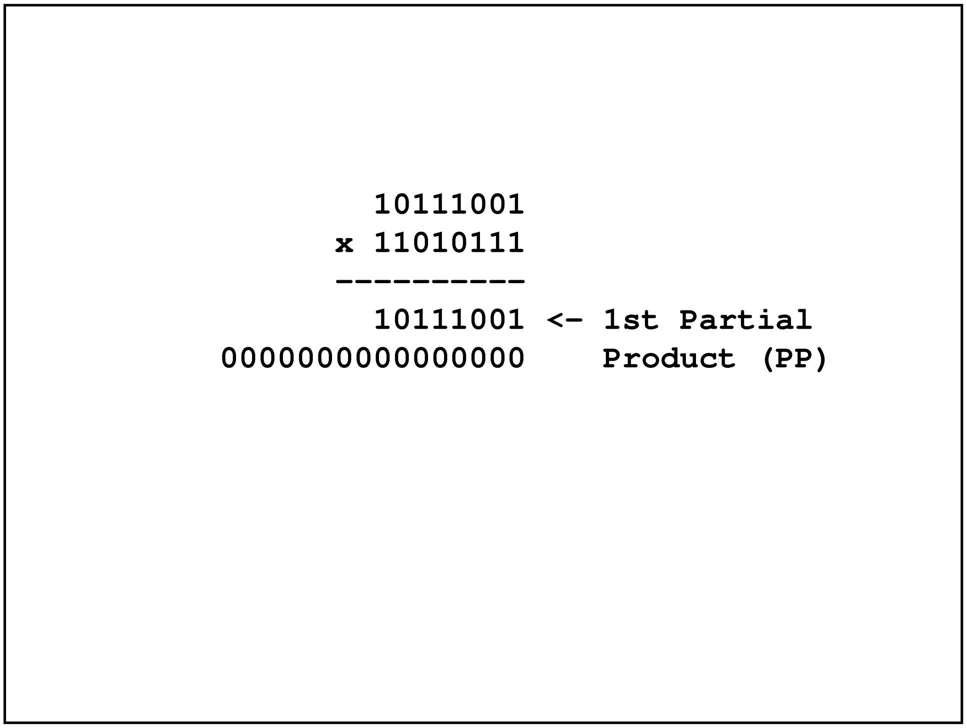


Multiply Example

```
        10111001   (185)
      x 11010111   (215)
      ----------
        10111001
       10111001
      10111001
     00000000
    10111001
   00000000
  10111001
 10111001
----------------
1001101101011111 (39775)
```



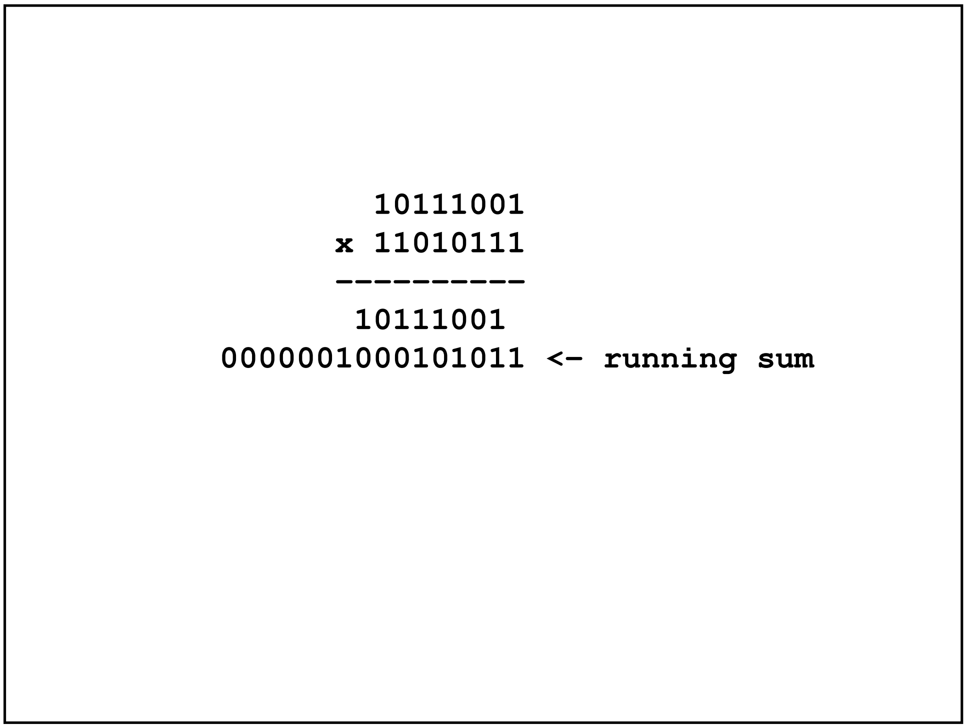
```
        10111001
      x 11010111
      ----------

0000000000000000  <- Start
                     with zero
```



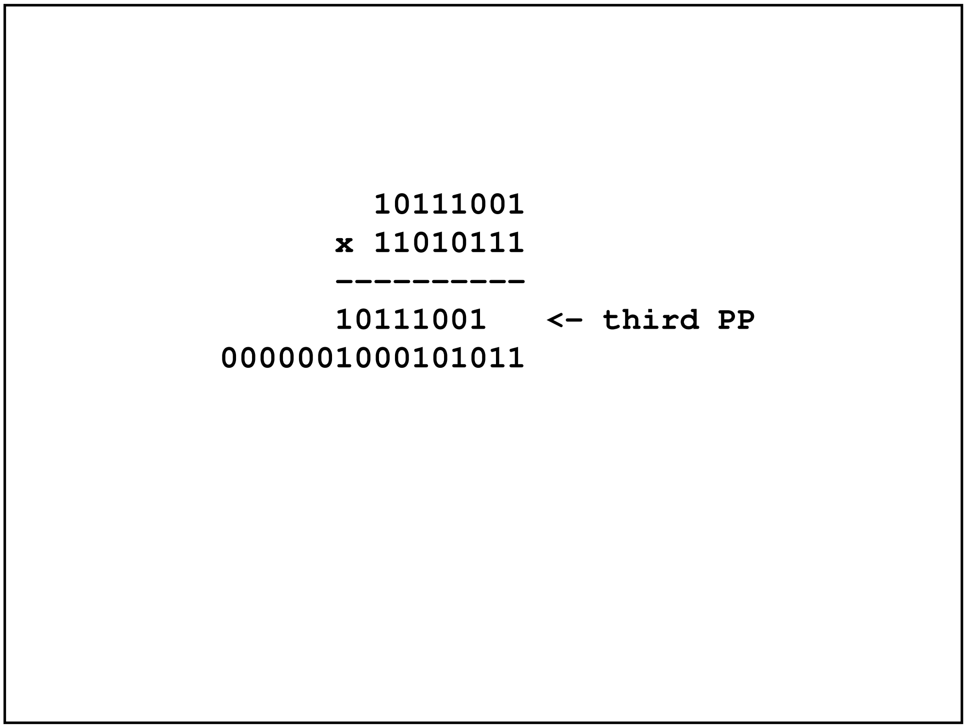
```
        10111001
      x 11010111
      ----------
        10111001 <- 1st Partial
0000000000000000    Product (PP)
```

```
        10111001
      x 11010111
      ----------
        10111001
0000000010111001 <- sum of Prod, PP
```

```
        10111001
      x 11010111
      ----------
       10111001 <- second PP
0000000010111001
```



```
        10111001
      x 11010111
      ----------
       10111001
0000001000101011 <- running sum
```

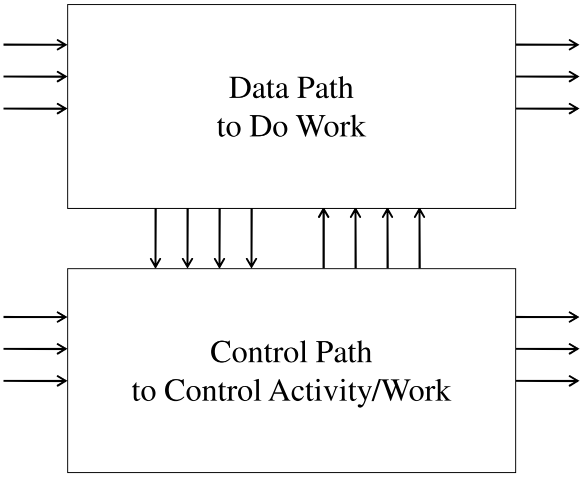


```
        10111001
      x 11010111
      ----------
      10111001   <- third PP
0000001000101011
```

```
        10111001
      x 11010111
      ----------
      10111001
0000001100001111 <- running sum
```

```
        10111001
      x 11010111
      ----------
     00000000    <- fourth PP
0000001100001111
```

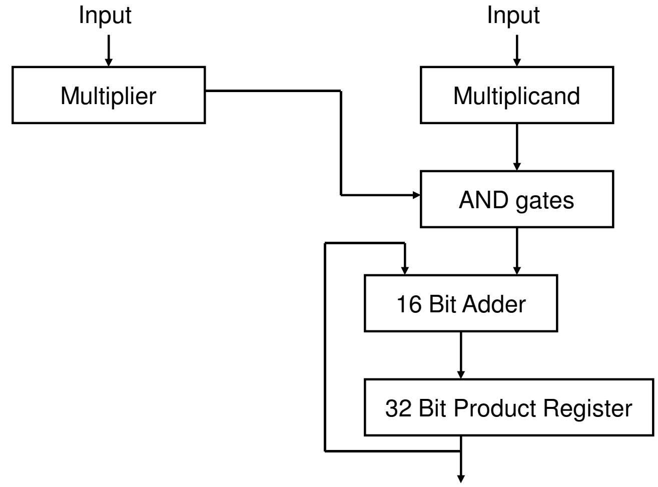
```
        10111001
      x 11010111
      ----------
     00000000
0000001100001111 <- running sum
```



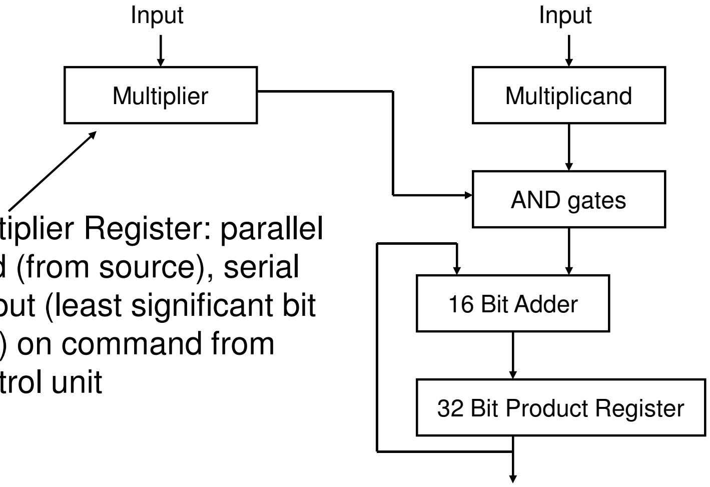
Data Path
to Do Work

Control Path
to Control Activity/Work

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)

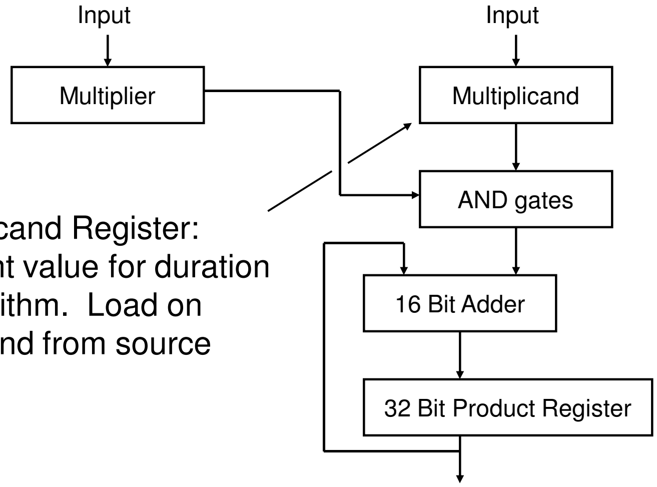


Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



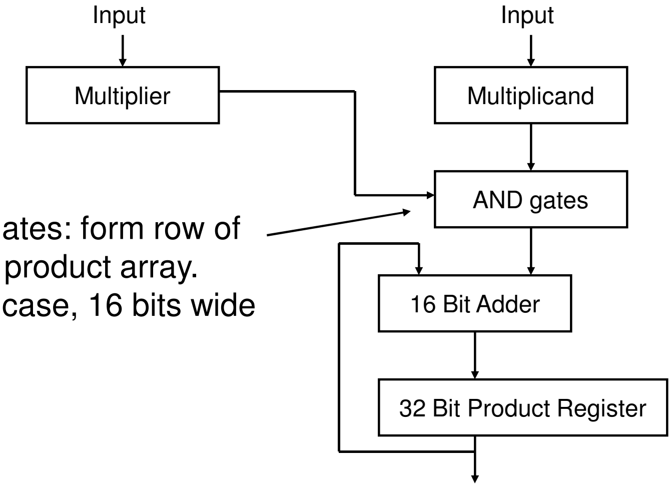
Multiplier Register: parallel load (from source), serial output (least significant bit first) on command from control unit

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



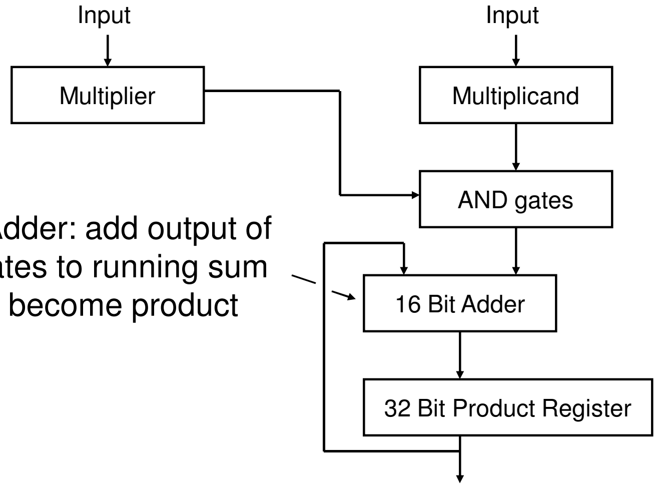
Multiplicand Register:
constant value for duration
of algorithm. Load on
command from source

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



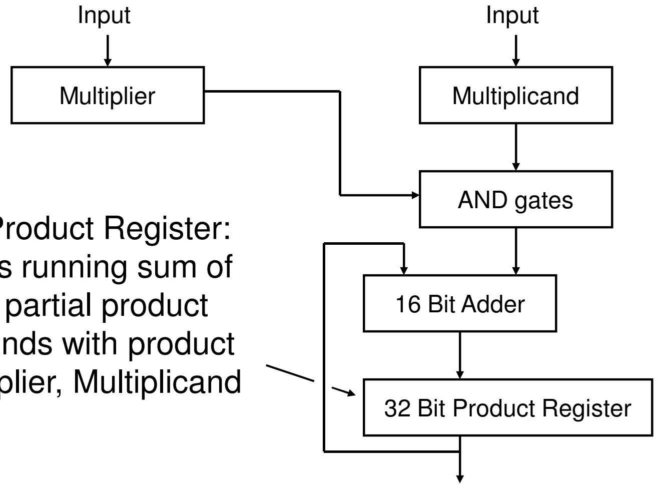
AND gates: form row of
partial product array.
In this case, 16 bits wide

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



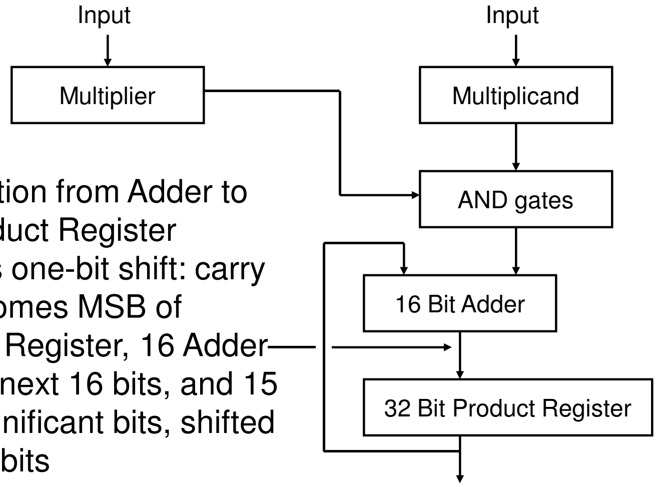
16 Bit Adder: add output of AND gates to running sum that will become product

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



32 Bit Product Register: contains running sum of rows of partial product array; ends with product of Multiplier, Multiplicand

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)



Connection from Adder to the Product Register includes one-bit shift: carry out becomes MSB of Product Register, 16 Adder outputs next 16 bits, and 15 least significant bits, shifted product bits

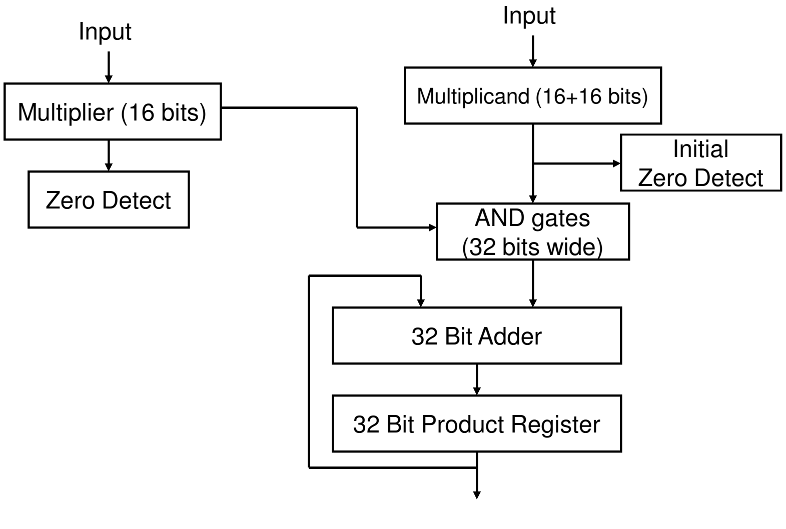
Register: asynchronous and synchronous behavior

```
NAME_OF_PROC:
process ( <clock, reset, set go here> ) is

begin

  if <asynchronous signals tested here> then
    <put set, reset stuff here>
  elsif RISING_EDGE ( clock ) then
     < put synchronous stuff here, in particular… >
    if <enabling condition> then
      <action/activity of register goes here>
    end if;
  end if;

end process NAME_OF_PROC;
```

Multiplication – Simple Gradeschool Algorithm for 16 Bits (32 Bit Result)

```
Step 1:
    Clear Product Register
    Clear Counter
    Load Multiplicand
    Load Multiplier
Step 2: (repeat 16 times)
    Increment Counter
    Load Product Register
    Shift Multiplier
Step 3:
    Done
```

Implement the Modified Gradeschool Multiplication Algorithm



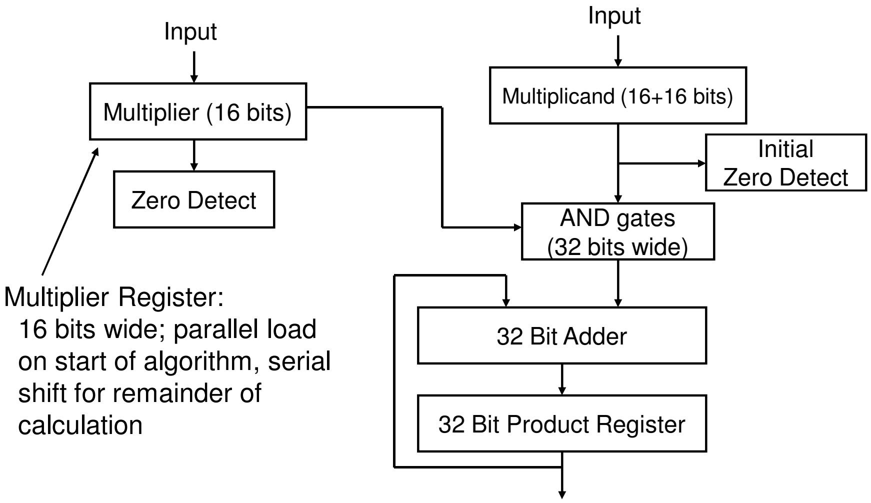
Biggest Unsigned Binary Multiply – 16 Bits × 16 Bits

```
                1111111111111111
                1111111111111111
                ----------------
                1111111111111111
               1111111111111111
              1111111111111111
             1111111111111111
            1111111111111111
           1111111111111111
          1111111111111111
         1111111111111111
        1111111111111111
       1111111111111111
      1111111111111111
     1111111111111111
    1111111111111111
   1111111111111111
  1111111111111111
 1111111111111111
--------------------------------
11111111111111100000000000000001
```

Biggest Unsigned Binary Multiply – 16 Bits × 16 Bits

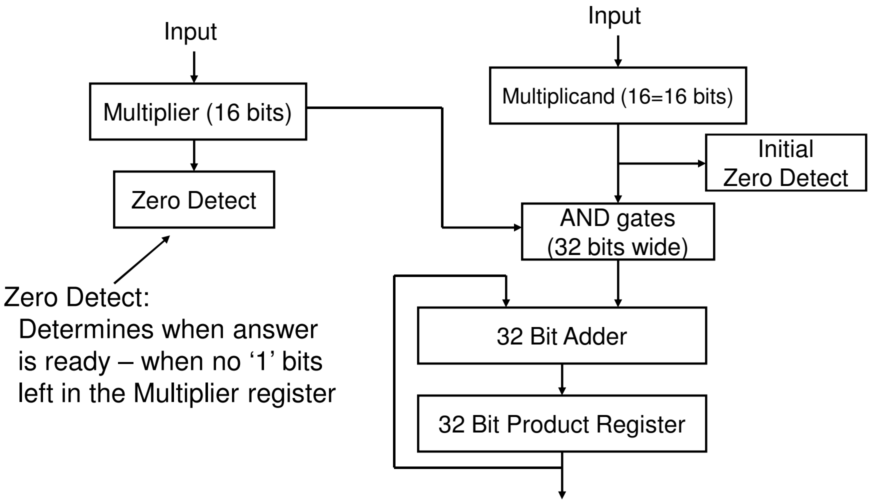
```
                1111111111111111
                1111111111111111
                ----------------
00000000000000001111111111111111
00000000000000011111111111111110
00000000000000111111111111111100
00000000000001111111111111111000
00000000000011111111111111110000
00000000000111111111111111100000
00000000001111111111111111000000
00000000011111111111111110000000
00000000111111111111111100000000
00000001111111111111111000000000
00000011111111111111110000000000
00000111111111111111100000000000
00001111111111111111000000000000
00011111111111111110000000000000
00111111111111111100000000000000
01111111111111111000000000000000
--------------------------------
11111111111111100000000000000001
```

Implement the Modified Gradeschool Multiplication Algorithm

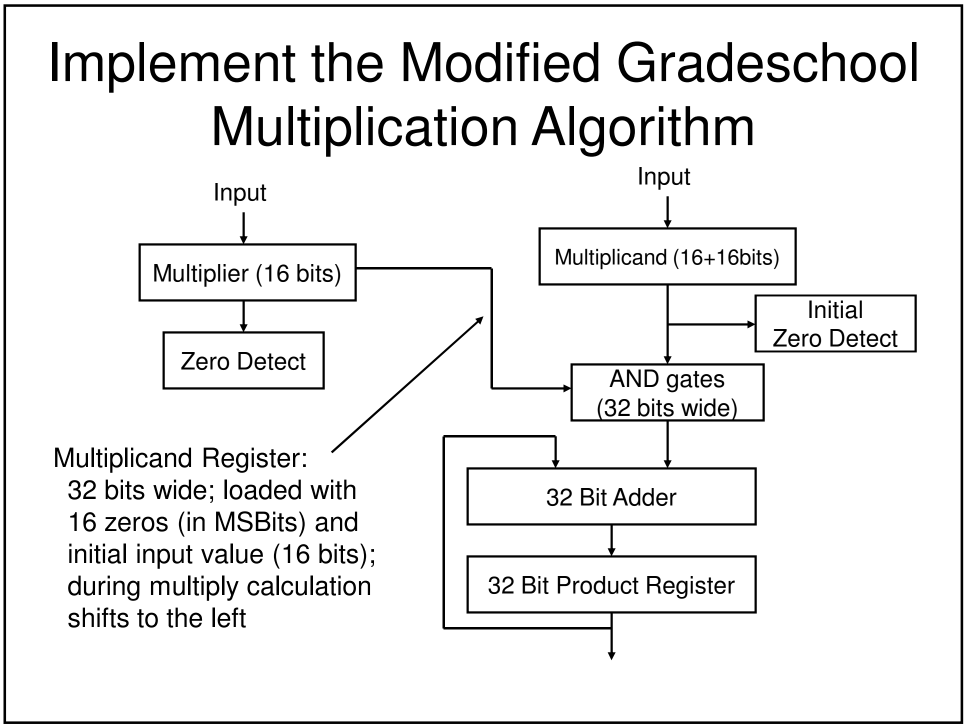


Multiplier Register:
16 bits wide; parallel load on start of algorithm, serial shift for remainder of calculation

Implement the Modified Gradeschool Multiplication Algorithm

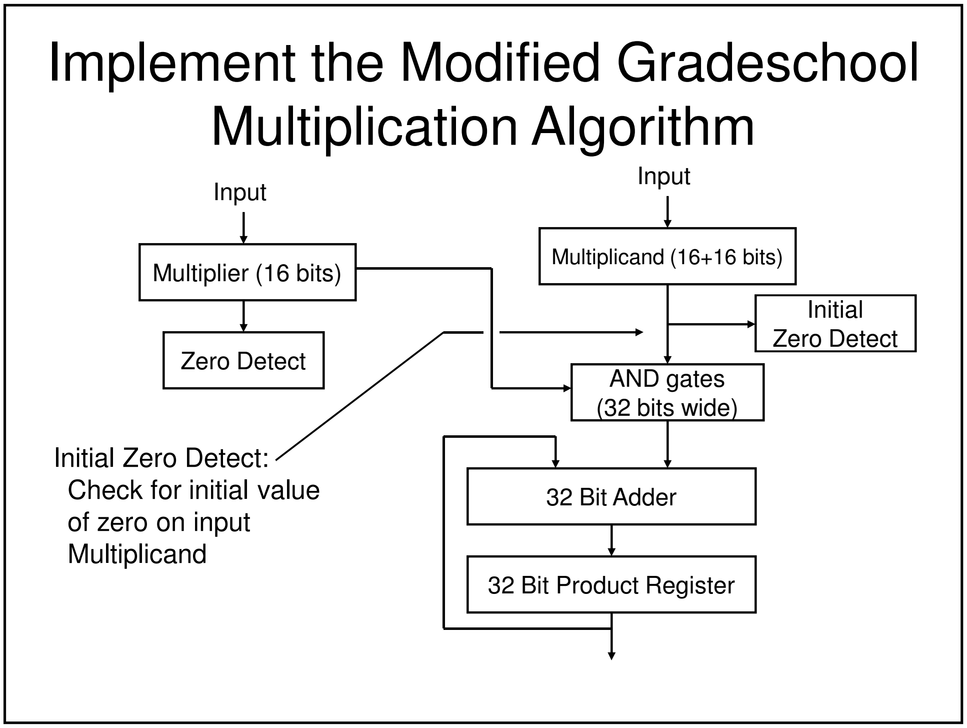


Zero Detect:
Determines when answer is ready – when no ‘1’ bits left in the Multiplier register



Implement the Modified Gradeschool Multiplication Algorithm

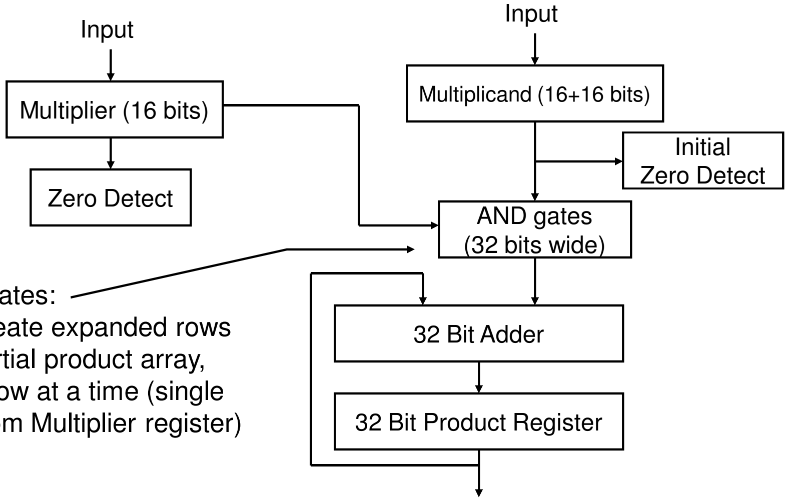
Multiplicand Register:
32 bits wide; loaded with 16 zeros (in MSBits) and initial input value (16 bits); during multiply calculation shifts to the left



Implement the Modified Gradeschool Multiplication Algorithm

Initial Zero Detect:
Check for initial value of zero on input Multiplicand

Implement the Modified Gradeschool Multiplication Algorithm



Input

Input

Multiplier (16 bits)

Multiplicand (16+16 bits)

Initial Zero Detect

Zero Detect

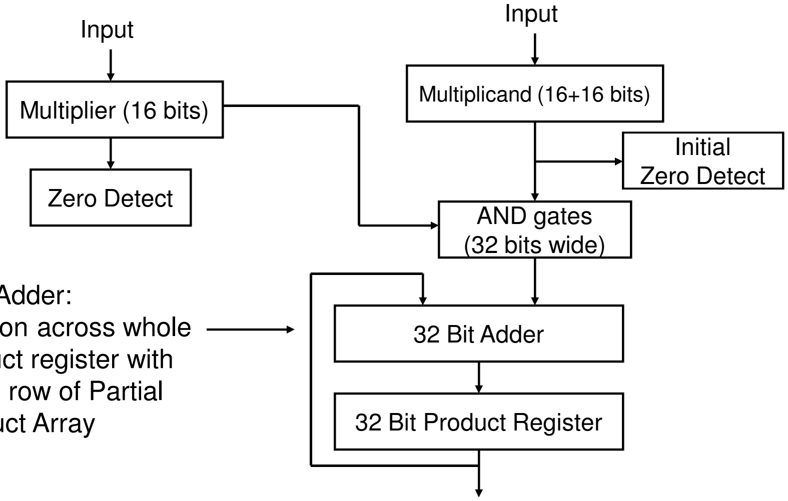
AND gates (32 bits wide)

32 Bit Adder

32 Bit Product Register

AND gates:
To create expanded rows of partial product array, one row at a time (single bit from Multiplier register)

Implement the Modified Gradeschool Multiplication Algorithm



Input

Input

Multiplier (16 bits)

Multiplicand (16+16 bits)

Initial Zero Detect

Zero Detect

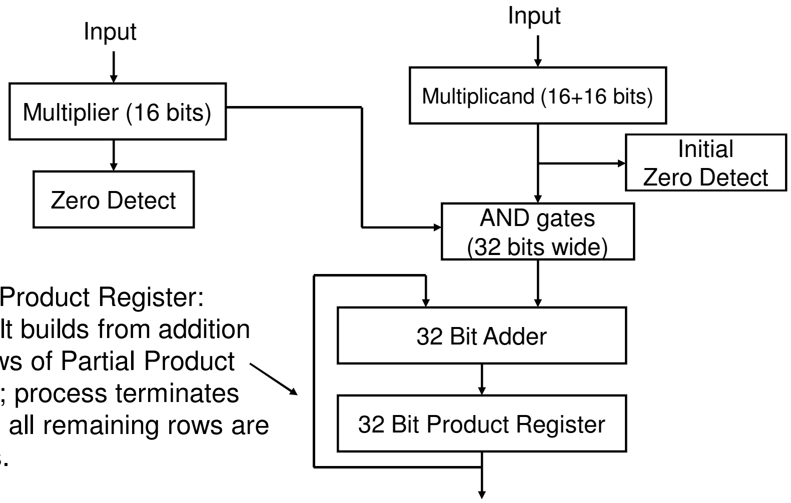
AND gates (32 bits wide)

32 Bit Adder

32 Bit Product Register

32 Bit Adder:
Addition across whole product register with entire row of Partial Product Array

Implement the Modified Gradeschool Multiplication Algorithm



32 Bit Product Register:
Result builds from addition of rows of Partial Product Array; process terminates when all remaining rows are zeros.

Multiplication – Modified Gradeschool Algorithm for 16 Bits (32 Bit Result)

Step 1:
  Clear Product Register
  Load Multiplicand
  Load Multiplier
Step 2:
  If MIER = 0 OR MCAND = 0, done
Step 3:
  Load Product Register
  Shift Multiplier
Step 4:
  If MIER = 0, then Done
  else Go to Step 3