POLYNOMIAL MULTIPLIER MODULE FOR $GF(2^4)$

It is not allowed to use a ROM-based implementation of the nibble multiplication of the two elements in $GF(2^4)$ $modulo$ $(x^4 + x + 1)$ as described in the Mini-AES specification.

The following is a simplified algorithm for this operation that is based on finding the Greatest Common Divisor (GCD) for polynomials using the Extended Euclidean algorithm for polynomials:

1. Initialize three registers:
    a. A Multiplier register (MPR-R) of 8 bits to store the value of the multiplier
    b. A Multiplicand Register (MCD-R) of 4 bits to store the value of the multiplicand, and
    c. A result register (Rslt) of 8 bits and initialize it to 0x00
2. If the multiplicand is greater than 0:
    a. If the LSB of the MCD-R is 1➔ Set (Rslt) = (Rslt) XOR (MPR-R).
    b. Shift the MPR-R left by 1.
    c. If the 5th bit of the MPR-R register is set ➔ Set (MPR-R) = (MPR-R) XOR (0x03).
    d. Shift the MCD-R right by 1 bit.
3. Go to step 2 and repeat until the MCD-R is 0.
4. When the MCD-R is 0, your output = (Rslt) AND (0X0F).

Example: Assume Multiplier = 0x9 (MPR = 0x9) and Multiplicand of 7 (MCD = 0x7)

Step	**MPR-R**	**MCD-R**	**Rslt**
1.	0x09 = 0000 1001	0x7 = 0111	0x00 = 0000 0000
2.a	0x09 = 0000 1001	0x7 = 0111	(0x00 xor 0x09) = 0x09 = 1001
2.b	0x12 = 0001 0010	0x7 =0111	0x09
2.c	(0x12 xor 0x03) = 0001 0010 ^ 0000 0011 0x11 = 0001 0001	0x7	0x09
2.d	0x11	0x3 = 0011	0x09
Repeat			
2.a	0x11 = 0001 0001	0x3 = 0011	(0x09 xor 0x11)= 0x18 = 0001 10000
2.b	0x22 = 0010 0010	0x3	0x18
2.c (skipped)			
2.d	0x22	0x1 = 0001	0x18
Repeat			
2.a	0x22 = 0010 0010	0x1 = 0001	0x18 xor 0x22 = 0001 1000 ^ 0010 0010 = 0x3a = 0011 1010
2.b	0x44 = 0100 0100	0x1	0x3a
2.c (skipped)			
2.d	0x44	0x0 = 0000	0x3a
3. skipped			
4.			0x3A and 0x0F = 0011 1010 & 0000 1111 =0x0A

From the above process we get the correct value for the operation, 0x9 * 0x7 = 0xA. To implement this algorithm into hardware, we must have at a minimum the following signals:

Clock – A necessary signal due to the shifting operations that must take place in the algorithm

Reset – To reset the internal state of the registers

Load – Required for the loading of values present on multiplier and multiplicand ports into the appropriate registers

Multiplier and Multiplicand – Signals containing the 4-bit values to be multiplied together

Output – Signal containing the final value from computation

Done – Should be activated to indicate the value of the operation is valid at the output

The easiest way to implement this algorithm is by using behavioral modeling (a clocked always block that contains sequential conditional statements)