Research Problem: Fully Parallel Multipliers for $GF(2^m)$

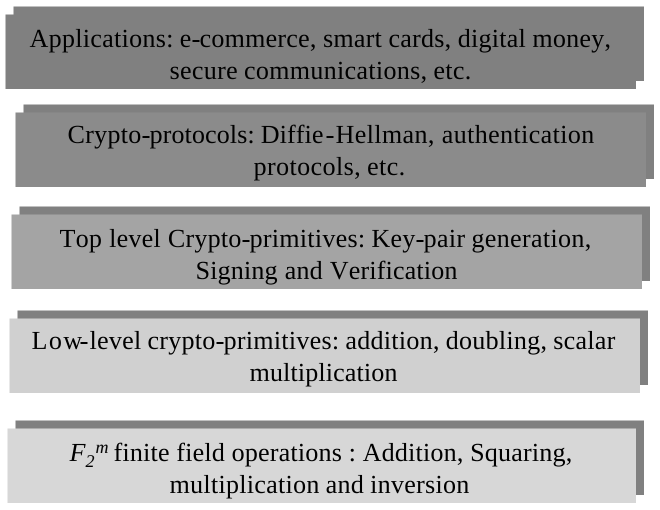
Francisco Rodríguez Henríquez
CINVESTAV-IPN
Computer Science Section
Av. IPN #2508 México, D.F. 07360, México
e-mail: francisco@cs.cinvestav.mx

Outline

Introduction

- Arithmetic over binary finite fields *GF*(2^m) has many important applications, particularly in the theory of error control coding and in cryptography.
- In a binary finite field, field addition, subtraction, multiplication and division are defined.
- Out of those four operations, multiplication is by far the most important of them. That is because most applications need to perform a large number of multiplications during the execution of the algorithms that conform their schemes.
- In the next slide we show a typical top-down model for modern cryptographic applications.

Modern Cryptosystems: A Top-Down Model


Applications: e-commerce, smart cards, digital money, secure communications, etc.

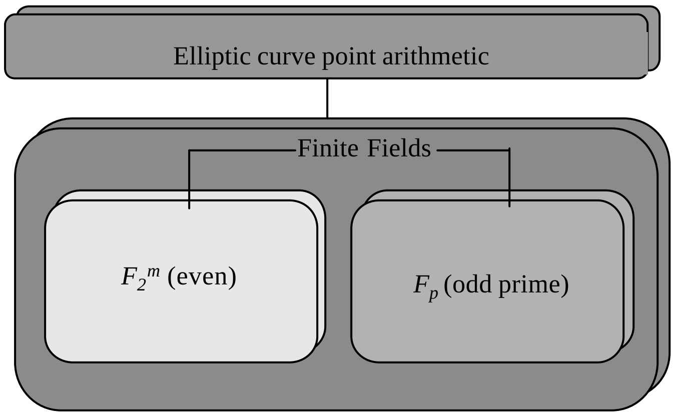
Crypto-protocols: Diffie-Hellman, authentication protocols, etc.

Top level Crypto-primitives: Key-pair generation, Signing and Verification

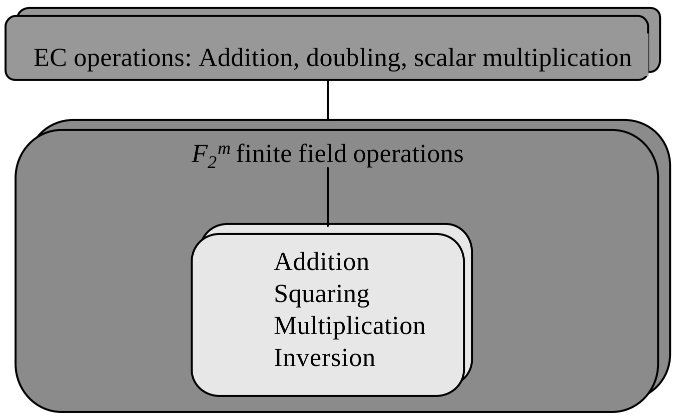
Low-level crypto-primitives: addition, doubling, scalar multiplication

F_{2^m} finite field operations : Addition, Squaring, multiplication and inversion

Elliptic curves over finite fields



Elliptic curves over finite fields



EC operations: Addition, doubling, scalar multiplication
F_{2^m} finite field operations
Addition
Squaring
Multiplication
Inversion

Two-steps Multipliers

In most algorithms the modular product is computed in two steps: polynomial multiplication followed by modular reduction. Let *A(x)*, *B(x)* and *(x)* Î *GF(2*^{m}*)* and *P(x)* be the irreducible field generator polynomial.

- In order to compute the modular product we first obtain the product polynomial *C(x)*, of degree at most *2m-2*, as

Polynomial product
2m-1 coordinates

$$\boxed{C(x) = A(x)B(x) = \left(\sum_{i=0}^{m-1} a_i a^i\right)\left(\sum_{i=0}^{m-1} b_i a^i\right)}$$

- Then, in the second step, a reduction operation is performed in order to obtain the *m-1* degree polynomial *C'(x)* is defined as

Reduction step
m coordinates

$$\boxed{C'(x) = C(x) \bmod P(x)}$$

Modular Multiplication in *GF(2m)*: Some Definitions

The field F_{2^m}

Let us consider a finite field $F=GF(2^m)$ over $K=GF(2)$.

Elements of F: Polynomials of degree less than m, with coefficients in K, such that,

$$\{a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\ldots+a_2x^2+a_1x+a_0 | a_i = 0 \text{ or } 1\}.$$

Fact: The field F has exactly $q-1=2^m-1$ nonzero elements plus the zero element.

Seminario de FPGAs
otoño 2003

Francisco Rodríguez Henríquez

Generating polynomial and polynomial basis

The finite field $F=GF(2^m)$ is completely described by a monic irreducible polynomial, often called generating polynomial, of the form

$$P(x) = x^m + k_{m-1}x^{m-1} + k_{m-2}x^{m-2} + \ldots + k_1 x + k_0$$

Where $k_i \in GF(2)$ for $i=0,1,\ldots,m-1$.

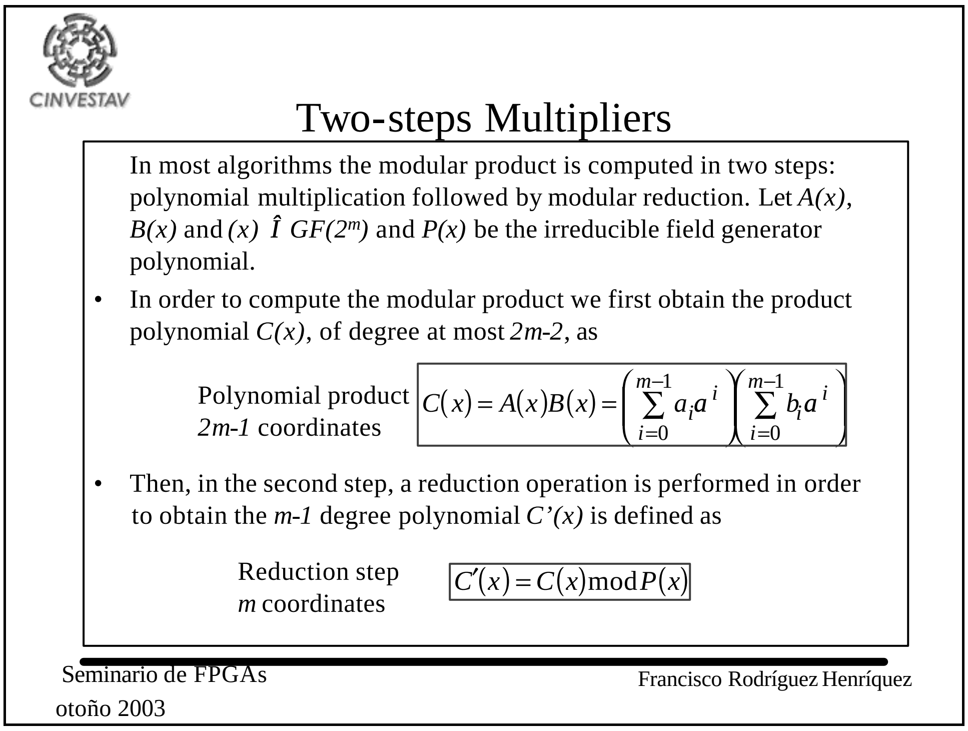
Let a be a primitive root of $P(x)$, i.e., $P(a) = 0$. Then, we define the polynomial or canonical basis of $GF(2^m)$ over $GF(2)$ using the primitive element a and its m first powers

$$\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\},$$

which happen to be linearly independent over $GF(2)$.

Seminario de FPGAs
otoño 2003

Francisco Rodríguez Henríquez



Two-steps Multipliers

In most algorithms the modular product is computed in two steps: polynomial multiplication followed by modular reduction. Let *A(x)*, *B(x)* and *(x)* Î *GF(2^m)* and *P(x)* be the irreducible field generator polynomial.

- In order to compute the modular product we first obtain the product polynomial *C(x)*, of degree at most *2m-2*, as

Polynomial product
2m-1 coordinates

$$\boxed{C(x) = A(x)B(x) = \left(\sum_{i=0}^{m-1} a_i a^i\right)\left(\sum_{i=0}^{m-1} b_i a^i\right)}$$

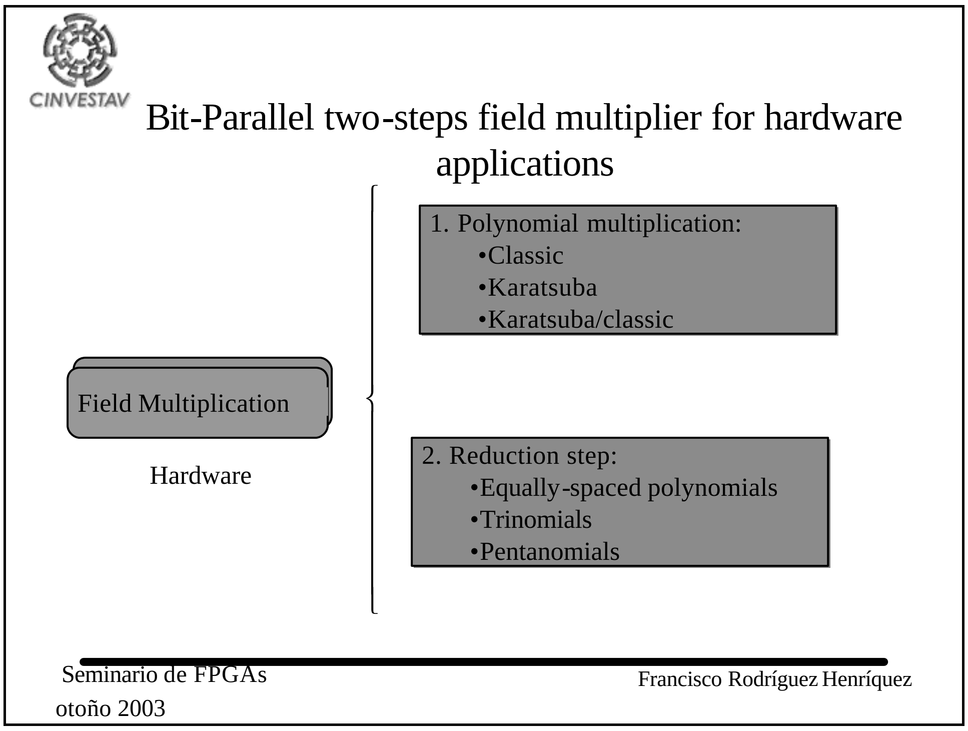
- Then, in the second step, a reduction operation is performed in order to obtain the *m-1* degree polynomial *C'(x)* is defined as

Reduction step
m coordinates

$$\boxed{C'(x) = C(x) \bmod P(x)}$$

Seminario de FPGAs
otoño 2003

Francisco Rodríguez Henríquez



Bit-Parallel two-steps field multiplier for hardware applications

Field Multiplication

Hardware

1. Polynomial multiplication:
   - •Classic
   - •Karatsuba
   - •Karatsuba/classic

2. Reduction step:
   - •Equally-spaced polynomials
   - •Trinomials
   - •Pentanomials

Seminario de FPGAs
otoño 2003

Francisco Rodríguez Henríquez

Multipliers performance criteria for hardware applications

- Usually, the measure of the performance for hardware implementations of the arithmetic operations in the Galois field *GF(2^m)* is the space and time complexities.
- Main performance criteria
  - Space complexity
    - Number of AND gates
    - Number of XOR gates
  - Time complexity
    - Circuit’s total gate delay

Polynomial multiplication: classical algorithm

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{m-2} \\ c_{m-1} \\ c_m \\ c_{m+1} \\ \vdots \\ c_{2m-3} \\ c_{2m-2} \end{bmatrix} = \begin{bmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_1 & a_0 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & a_{m-2} & \cdots & a_3 & a_2 \\ \vdots & & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-2} \\ b_{m-1} \end{bmatrix}$$

AND gates = m^2
XOR gates = $(m-1)^2$
Time delay = $T_A + \lceil \log_2 m \rceil T_X$

Special Case: Squaring

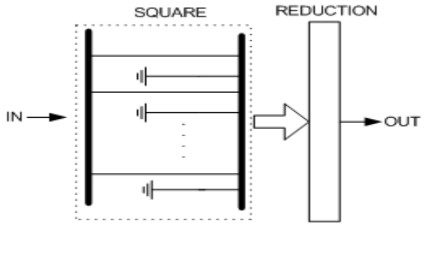
- Let *A* be an element of the finite field *F=GF*(2^5). Then, the square of *A* is given as,

$$
\begin{array}{ccccccccc}
 & & & & a_4a_3a_2a_1a_0 & * & a_4a_3a_2a_1a_0 & & \\
\hline
 & & & & a_4a_0 & a_3a_0 & a_2a_0 & a_1a_0 & a_0a_0 \quad + \\
 & & & a_4a_1 & a_3a_1 & a_2a_1 & a_1a_1 & a_0a_1 & \\
 & & a_4a_2 & a_3a_2 & a_2a_2 & a_1a_2 & a_0a_2 & & \\
 & a_4a_3 & a_3a_3 & a_2a_3 & a_1a_3 & a_0a_3 & & & \\
a_4a_4 & a_3a_4 & a_2a_4 & a_1a_4 & a_0a_4 & & & & \quad = \\
\hline
a_4 \; 0 & a_3 & 0 & a_2 & 0 & a_1 & 0 & a_0 &
\end{array}
$$

In general, for an arbitrary element *A* in the field *F=GF*(2^5), we have,

$$C(x)=A(x)A(x)=A^2(x)=\left(\sum_{i=0}^{m-1}a_ix^i\right)\left(\sum_{i=0}^{m-1}a_ix^i\right)=\sum_{i=0}^{m-1}a_ix^{2i}$$

Special Case: Squaring [by Nazar Saqib]



$$A=a_3x^3+a_2x^2+a_1x+a_0$$
$$A^2=a_6x^6+a_4x^4+a_2x^2+a_0$$

A = 1111
A^2= 1010101

$2^k n$-bit Karatsuba Multipliers

$2kn$-bit Karatsuba Multipliers

There are some asymptotically faster methods for polynomial multiplications, such as the Karatsuba-Ofman algorithm.

Discovered in 1962, it was the first algorithm able to accomplish polynomial multiplication under $O(m^2)$ operations.

Karatsuba's algorithm is based on the idea that the polynomial product $C=AB$ can be written as,

$$A = x^{\frac{m}{2}} A^H + A^L; \quad B = x^{\frac{m}{2}} B^H + B^L;$$

$$C = x^m A^H B^H + \left(A^H B^H + A^L B^L + \left(A^H + A^L\right)\left(B^H + B^L\right)\right) x^{\left\lceil \frac{m}{2} \right\rceil} + A^L B^L = x^m C^H + C^L$$

CINVESTAV

$2kn$-bit Karatsuba Multipliers

- last equation can be carried out at the cost of only 3 polynomial multiplications and four polynomial additions.
- Of course, Karatsuba strategy can be applied recursively to the three polynomial multiplications of last equation.
- By applying this strategy recursively, it is possible to achieve a polynomial complexity of $O\left(m^{\log_2 3}\right)$
- Best results can be obtained by combining classical method with Karatsuba strategy.

Seminario de FPGAs
otoño 2003

Francisco Rodríguez Henríquez

CINVESTAV

Procedure Kmul2^k(C, A, B)

Input: Two elements A, B ? $GF(2^m)$ with $m=rn=2^kn$, and where A, B can be expressed as,

$$A=x^{\frac{m}{2}}A^H+A^L, B=x^{\frac{m}{2}}B^H+B^L.$$

Output: A polynomial $C=AB$ with up to $2m$-1 coordinates, where $C=x^mC^H+C^L$..

```
0.  begin
1.     if ($r == 1$) then
2.         $C = mul\_n(A, B)$;
3.         return;
4.     for i from 0 to $\frac{r}{2}-1$ do
5.         $M_{Ai} = A_i^L + A_i^H$;
6.         $M_{Bi} = B_i^L + B_i^H$;
7.     end
8.     $mul2^k(C^L, A^L, B^L)$;
9.     $mul2^k(M, M_A, M_B)$;
10.    $mul2^k(C^H, A^H, B^H)$;
11.    for i from 0 to r - 1 do
12.        $M_i = M_i + C_i^L + C_i^H$;
13.    end
14.    for i from 0 to r - 1 do
15.        $C_{\frac{r}{2}+i} = C_{\frac{r}{2}+i} + M_i$;
16.    end
17. end
```

IASTED-CST may 2003

Francisco Rodríguez Henríquez

2*kn*-bit Karatsuba Multipliers

It can be shown that the space and time complexities of a m=2^kn-bit Karatsuba multiplier combined with a classical method are given as,

$$\text{XOR Gates} \le \left(\frac{m}{n}\right)^{\log_2 3}\left(n^2+6n-1\right)-8m+2;$$

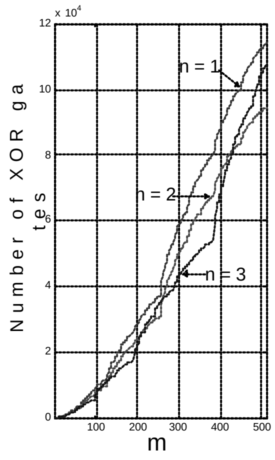
$$\text{AND Gates} \le \left(\frac{m}{n}\right)^{\log_2 3} n^2;$$

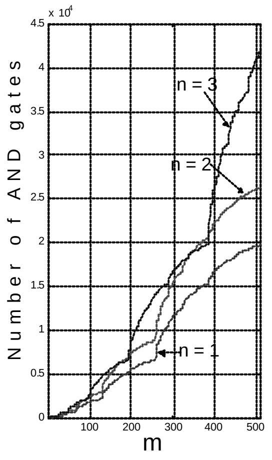
$$\text{Time Delay} \le T_{AND}+T_X\left(\log_2 n+k\right).$$

Space and Time complexities

m	r	n	AND gates	XOR gates	Time Delay	Area (NAND units)
1	1	1	1	0	T_A	1.26
2	1	2	4	1	T_X+T_A	7.2
4	1	4	16	9	$2T_X+T_A$	40.0
8	2	4	48	55	$6T_X+T_A$	181.5
16	4	4	144	225	$10T_X+T_A$	676.4
32	8	4	432	799	$14T_X+T_A$	2302.1
64	16	4	1296	2649	$18T_X+T_A$	7460.8
128	32	4	3888	8455	$22T_X+T_A$	23499.9
256	64	4	11664	26385	$26T_X+T_A$	72743.6
512	128	4	34992	81199	$30T_X+T_A$	222727.7

Space complexity of hybrid Karatsuba multipliers for arbitrary *m* using *n=1, 2, 3*





Binary Karatsuba Multipliers

Binary Karatsuba Multipliers

- Problem: Find an efficient Karatsuba strategy for the multiplication of two polynomials $A, B \in GF(2^m)$, such that $m = 2^k + d, d \neq 0$.
- Basic Idea: Pretend that both operands are polynomials with degree m' = $2^{(k+1)}$, and use normal Karatsuba approach for two of the three required polynomial multiplications, i.e., given

$$A = x^{\frac{m}{2}} A^H + A^L; \quad B = x^{\frac{m}{2}} B^H + B^L;$$

$$C = x^m A^H B^H + \left(A^H B^H + A^L B^L + \left(A^H + A^L\right)\left(B^H + B^L\right)\right)x^{\left\lceil \frac{m}{2} \right\rceil} + A^L B^L = x^m C^H + C^L$$

Binary Karatsuba Multipliers

- Compute the two 2^k-bit polynomial multiplications:

$$A^L B^L \text{ and;}$$
$$M = M_A M_B = \left(A^H + A^L\right)\left(B^H + B^L\right)$$

- While the remaining d-bit polynomial multiplication $A^H B^H$ can be computed using a $k' = \lceil \log_2(d) \rceil$ -bit Karatsuba multiplier in a recursive manner (since the leftover d bits can be expressed as, $d = 2^{k1} + d_1$).

Binary Karatsuba Multipliers

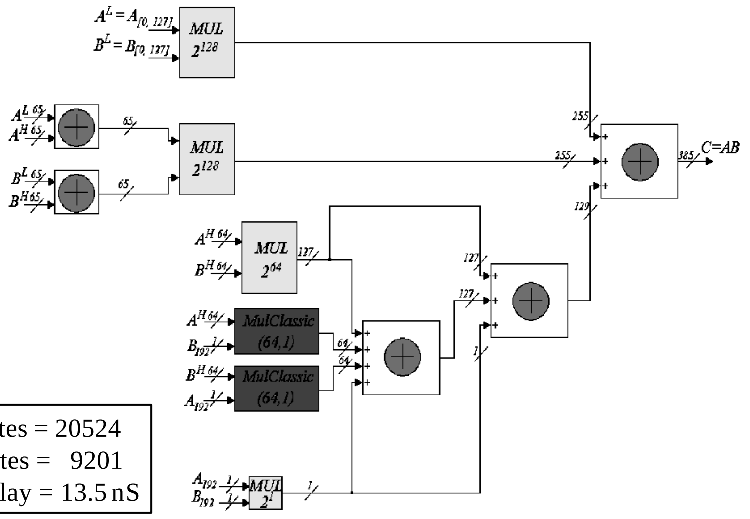
- The above outlined strategy yields a Binary Karatsuba scheme where the hamming weight of the original m will determine the number of recursive iterations to be used by the algorithm.

An Example

An Example

- As a design example, let us consider the polynomial multiplication of the elements A and $B \in GF(2^{193})$. Since $(193)_2$ = 11000001, the Hamming weight of m is h = 3. This will imply that we need a total of three iterations in order to compute the multiplication using the generalized m-bit binary Karatsuba multiplier. Additionally we notice that for this case, m = 193 $=2^7+65$.

193-bit binary Karatsuba Multiplier

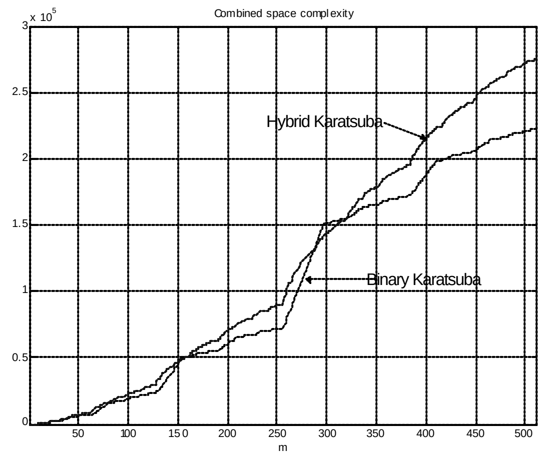


XOR gates = 20524
AND gates = 9201
Time delay = 13.5 nS

An Example

- Where we have assumed that the above circuit has been implemented using a 1.2μ CMOS technology, where we have that the time delays associated to the AND, XOR logic gates are given as: $T_A \cong T_x = 0.5$ nS.
- Next slide shows a comparison between the proposed binary Karatsuba approach and the more traditional hybrid approach discussed previously.

Binary and hybrid Karatsuba multipliers' area complexity



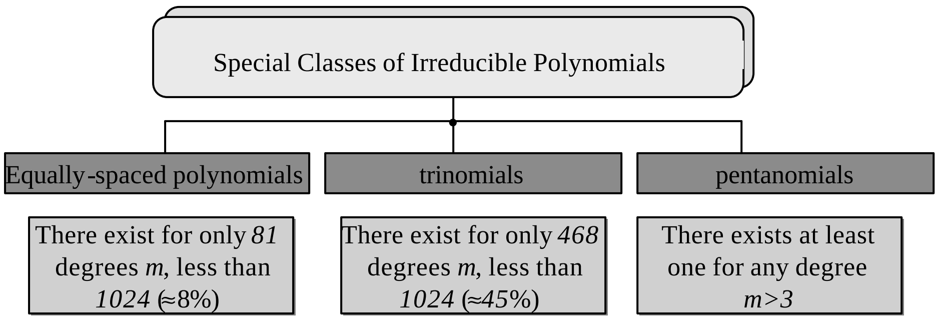
Second step: reduction

- Problem: Given the polynomial product *C(x)* with at most, *2m-1*, obtain the modular product *C'* with *m* coordinates, using the generating irreducible polynomial *P(x)*.

$$\boxed{C'(x) = C(x) \bmod P(x)}$$

- Using a general irreducible polynomial with Hamming weight (the number of nonzero terms) equal to *r* would require at most (*r-1*)(*m-1*) XOR gates, i.e., complexity *O*(*m*).
- The complexity of our schemes as applied to special classes of pentanomials (*r=5*) requires about *m* fewer XOR gates than the above prediction.

Bit-Parallel modular dual basis for hardware applications

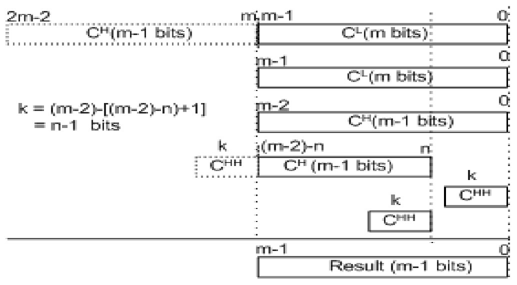

Special Classes of Irreducible Polynomials
Equally-spaced polynomials
trinomials
pentanomials

There exist for only *81* degrees *m*, less than *1024* (≈8%)

There exist for only *468* degrees *m*, less than *1024* (≈45%)

There exists at least one for any degree *m>3*

Squaring: Reduction Step FPGA Implementation [by Nazar Saqib]



Conclusions

Conclusions (1/2)

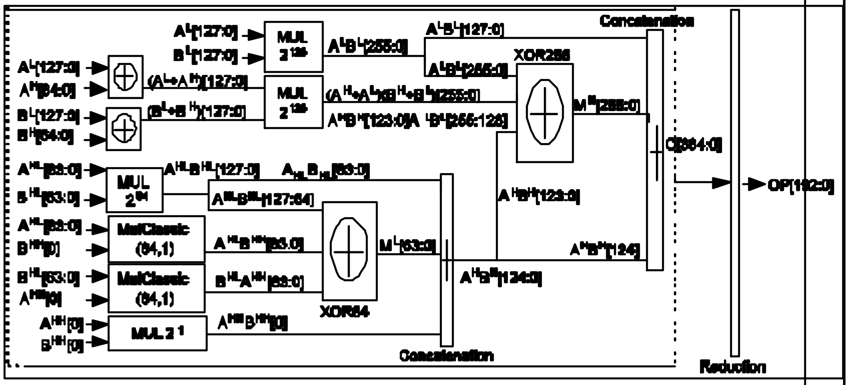
- In this paper we presented a new approach that generalizes the classic Karatsuba multiplier technique.
- The most attractive features of the new algorithm presented here is that the degree of the defining irreducible polynomial can be arbitrarily selected by the designer.

Conclusions (2/2)

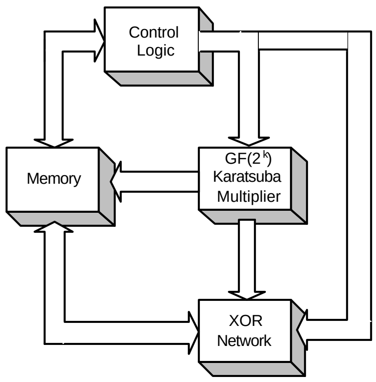
- Also the proposed multiplier leads to highly modular architectures and is thus well suited for VLSI implementations.
- As a future work, we are planning to implement in FPGA devices a sequential version of the strategy discussed here, as is shown in the next slide.

Field Multiplication: FPGA Implementation

Preliminary results yield a time delay of 50-70 ηSec and ≈9K Slices of hardware resources utilization.



Programmable binary Karatsuba Multiplier



Control Logic
Memory
GF(2^k) Karatsuba Multiplier
XOR Network