A New Approach to Subquadratic Space Complexity Parallel Multipliers over Extended Binary Fields

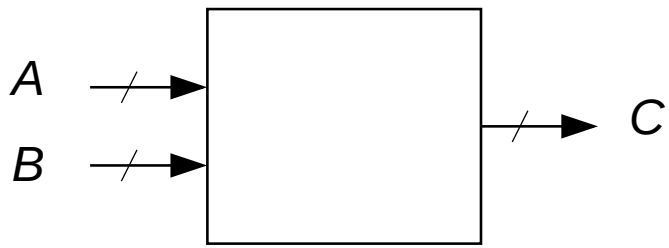
M. A. Hasan
joint work with H. Fan

University of Waterloo

December 5, 2006

Assumptions

- For efficient multiplication over GF(2^n), there is a 'good' field defining polynomial (e.g., trinomial or pentanomial)
- Multiplier inputs (say A and B) are arbitrary and available in full ($2n$ bits) at the same time



- Simple bit-parallel realization (no memory, no reuse of circuits in a single multiplication)

Hardware complexities

- space: number of logic gates– two-input ANDs ($\#AND$) and two-input XORs ($\#XOR$)
- time: maximum delay due to gates only (time for AND– T_A and time for XOR– T_X)

Arithmetic complexities

- number of GF(2) multiplications (= $\#AND$)
- number of GF(2) additions (= $\#XOR$)

Field multiplication using polynomial operations

Step 1 Polynomial multiplication:

$$S(x) = A(x)B(x) = \left(\sum_{i=0}^{n-1} a_i x^i\right)\left(\sum_{i=0}^{n-1} b_i x^i\right)$$

where $\deg S(x) \leq 2n-2$

Step 2 Polynomial reduction: Obtain $C(x)$ reducing $S(x)$ modulo field defining polynomial (say $F(x)$ of degree n)

Remarks:

- For Step 1 using the schoolbook method:

$$\#AND = n^2; \quad \#XOR \approx n^2$$
$$\text{Gate delay} \approx (\log_2 n) T_X$$

- For a low weight $F(x)$, Step 2 is much less expensive

Improvement using Karatsuba's trick

Let $A(x)$ and $B(x)$ be two-term polynomials. Then

$$(a_1x + a_0)(b_1x + b_0) =$$
$$(a_1b_1)x^2 + \{[(a_1 + a_0)(b_1 + b_0)] + [a_1b_1 + a_0b_0]\}x + a_0b_0$$

Notes:

- Number of coefficient multiplications is reduced from four to three
- If applied recursively to n-term $A(x)$ and $B(x)$
  - $\#AND = n^{\log_2 3} \approx n^{1.58}$
  - $\#XOR \leq 6n^{1.58} - 8n + 2$
  - Gate delay $\approx (3 \log_2 n)T_X$

Field multiplication using matrix-vector product

Step 1 Form binary $n \times n$ matrix $\mathbf{Z} = (A, Ax, \cdots, Ax^{n-1}) \bmod F(x)$

Step 2 Obtain matrix vector product $C = \mathbf{Z}B$

Notes:

- Step 1 requires less than nw_F XORs, where w_F is the number of nonzero coefficients in $F(x)$
- For straightforward matrix-vector product, Step 2 requires about n^2 ANDs and n^2 XORs

Example 1

Consider GF(2^4) and let $F(x)$ be $x^4 + x^1 + 1$. Then $C = \mathbf{Z}B$ becomes

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Example 2

If $F(x)$ is changed to $x^4 + x^3 + 1$, then

$$\mathbf{Z} = \begin{pmatrix} a_0 & a_3 & a_2 + a_3 & a_1 + a_2 + a_3 \\ a_1 & a_0 & a_3 & a_2 + 2_3 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 + a_3 & a_1 + a_2 + a_3 & a_0 + a_1 + a_2 + a_3 \end{pmatrix}$$

Remarks:

- Matrix $\mathbf{Z}$ does not appear to have any special form
- Delay $\leq (\log_2 n) T_X$. Delay can be reduced to $1T_X$ by using 'good' $F(x)$ and/or by using shifted-polynomial basis

Multiplication using coordinate transformation

For an arbitrary $F(x)$ of degree n, there exists matrix $\mathbf{U}$ such that

$$\mathbf{U}C = \mathbf{U}\mathbf{Z}B = \mathbf{T}B$$

where $\mathbf{T} = \mathbf{U}\mathbf{Z}$ is an $n \times n$ Toeplitz matrix.
Remarks:

- $\mathbf{T}$ of n^2 entries is completely defined by its $2n - 1$ entries
- Transformation matrix $\mathbf{U}$ can be defined using $F(x)$ and has been used for changing a field element's representation from *polynomial* to *triangular* basis
- Such transformation also applies to GF(q^n) where q is prime or a power of prime

Example 1 Cont'd.

$$\mathbf{UZ} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}$$

$$= \begin{pmatrix} a_0 + a_3 & a_3 + a_2 & a_2 + a_1 & a_0 + a_1 + a_3 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}$$

For Example 2, the following matrix gives the desired transformation.

$$\mathbf{U} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Taking advantage of Toeplitz structure

Consider the 4×4 Toeplitz matrix from Example 1:

$$\mathbf{T} = \begin{pmatrix} a_0 + a_3 & a_3 + a_2 & a_2 + a_1 & a_0 + a_1 + a_3 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}$$

- Two-way split gives

$$\mathbf{T} = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_0 \\ \mathbf{T}_2 & \mathbf{T}_1 \end{pmatrix}$$

  where sub-matrices $\mathbf{T}_0$, $\mathbf{T}_1$ and $\mathbf{T}_2$ are each individually a 2×2 Toeplitz matrix
- Sum of two Toeplitz matrices results in another Toeplitz matrix

Toeplitz matrix-vector product ($n = 2^i$):

$$\mathbf{T}V = \begin{pmatrix} \mathbf{T}_1 & \mathbf{T}_0 \\ \mathbf{T}_2 & \mathbf{T}_1 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \end{pmatrix} = \begin{pmatrix} P_0 + P_2 \\ P_1 + P_2 \end{pmatrix}$$

where

$$\begin{aligned} P_0 &= (\mathbf{T}_0 + \mathbf{T}_1)V_1, \\ P_1 &= (\mathbf{T}_1 + \mathbf{T}_2)V_0, \\ P_2 &= \mathbf{T}_1(V_0 + V_1). \end{aligned}$$

Remarks:

- Number of matrix-vector products has been reduced from four to three
- Some simple tricks reduce the cost of $\mathbf{T}_0 + \mathbf{T}_1$ and $\mathbf{T}_1 + \mathbf{T}_2$ by 25%

For $n = 2^i$ $(i > 0)$:

$$\#AND = n^{\log_2 3} \approx n^{1.58}$$

$$\#XOR = 5.5n^{\log_2 3} - 6n + 0.5$$

$$\text{Gate delay} = (2 \log_2 n)T_X + T_A$$

Toeplitz matrix-vector product ($n = 3^i$)

$$\mathbf{T}V = \begin{pmatrix} \mathbf{T}_2 & \mathbf{T}_1 & \mathbf{T}_0 \\ \mathbf{T}_3 & \mathbf{T}_2 & \mathbf{T}_1 \\ \mathbf{T}_4 & \mathbf{T}_3 & \mathbf{T}_2 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} P_0 + P_3 + P_4 \\ P_1 + P_3 + P_5 \\ P_2 + P_4 + P_5 \end{pmatrix},$$

where

$$\begin{aligned} P_0 &= (\mathbf{T}_0 + \mathbf{T}_1 + \mathbf{T}_2)V_2, & P_3 &= \mathbf{T}_1(V_1 + V_2), \\ P_1 &= (\mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_3)V_1, & P_4 &= \mathbf{T}_2(V_0 + V_2), \\ P_2 &= (\mathbf{T}_2 + \mathbf{T}_3 + \mathbf{T}_4)V_0, & P_5 &= \mathbf{T}_3(V_0 + V_1). \end{aligned}$$

- Number of matrix-vector products has been reduced from nine to six
- Some simple tricks reduce the cost of matrix additions

For $n = 3^i$ $(i > 0)$:

$$\#AND = n^{\log_3 6} \approx n^{1.63}$$

$$\#XOR = \frac{24}{5} n^{\log_3 6} - 5n + \frac{1}{5},$$

$$\text{Gate delay} = (3 \log_3 n) T_X + T_A$$

Dealing with arbitrary n

Let p_i and p_j be two small primes (i.e., 2 and 3 in this talk) for which we have efficient formulae for matrix-vector product $\mathbf{T}V$.

- If $p_i p_j | n$, let
  #XOR for formula p_i: $\mathcal{S}^{\oplus}_{p_i}(n) = h_i \mathcal{S}^{\oplus}_{p_i}(n/p_i) + k_i n + l_i$.

  Let $n = p_i p_j t$. #XOR of algorithms $\Omega(n, p_j, p_i)$ is:

  $$\mathcal{S}^{\oplus}_{\Omega(n,p_j,p_i)} = h_i[h_j \mathcal{S}^{\oplus}(t) + k_j p_j t + l_j] + k_i n + l_i.$$

  Compare $\Omega(n, p_j, p_i)$ and $\Omega(n, p_i, p_j)$ and choose
  E.g., $n = 6t$. #XOR are
  $63t - 4 + 18 \cdot \mathcal{S}^{\oplus}(t)$ (first 2 next 3) and
  $66t - 7 + 18 \cdot \mathcal{S}^{\oplus}(t)$ (first 3 next 2).
- If none of the p_i's divide n, then use padding & deleting.

Field multiplication using MVP (re-stated)

Step 1 Form binary $n \times n$ matrix $\mathbf{Z} = (A, Ax, \cdots, Ax^{n-1}) \bmod F(x)$

Step 1′ Transform $\mathbf{Z}$ to $\mathbf{T} = \mathbf{UZ}$

Step 2 Obtain matrix vector product $\mathbf{T}B$, which is equal to $\mathbf{U}C$

Step 2′ Obtain C from $\mathbf{U}C$

Efficient coordinate transformation

Trinomial: $F(x) = x^n + x^v + 1$

$$\mathbf{U}_{eff} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_{(n-v)\times(n-v)} \\ \mathbf{I}_{v\times v} & \mathbf{0} \end{pmatrix}$$

Consider the following from Example 1

$$\mathbf{Z} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}$$

$$\mathbf{U}_{eff}\mathbf{Z} = \begin{pmatrix} a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \\ a_0 & a_3 & a_2 & a_1 \end{pmatrix}$$

The following Toeplitz matrix was obtained earlier from $\mathbf{Z}$ of Example 1:

$$\mathbf{UZ} = \begin{pmatrix} a_0 + a_3 & a_3 + a_2 & a_2 + a_1 & a_0 + a_1 + a_3 \\ a_1 & a_0 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_3 \end{pmatrix}$$

Pentanomial: $F(u) = u^n + u^{v+1} + u^v + u^{v-1} + 1$

NIST recommended ECDSA fields:
$GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$, $GF(2^{571})$,

(n, v)=(163, 67), (163, 69), (163, 71), (283, 24), (283, 133),
(571, 104), (571, 230).

$$\mathbf{U}_{eff} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_{(n-v)\times(n-v)} + \mathbf{J}_{(n-v)\times(n-v)} \\ \mathbf{I}_{v\times v} + \mathbf{J}^T_{v\times v} & \mathbf{0} \end{pmatrix},$$

where $J_{0,v-1} = 1$ in $\mathbf{J}_{v\times v}$.

U is also simple for equally spaced polynomials.

Comparison

Hardware multipliers ($F(x) = x^n + x^k + 1$ with $n = b^i$)

b		#XOR	Gate delay
2	Kara	$6n^{\log_2 3} - 6n + k - 1$	$(3\log_2 n + 1)T_X + T_A$
	MVP	$5.5n^{\log_2 3} - 5n - 0.5$	$(2\log_2 n + 1)T_X + T_A$
3	Kara	$\frac{80}{15}n^{\log_3 6} - \frac{16}{3}n + k - 1$	$(4\log_3 n + 1)T_X + T_A$
	MVP	$\frac{72}{15}n^{\log_3 6} - 4n - \frac{4}{5}$	$(3\log_2 n + 1)T_X + T_A$

Software implementations

- As simple as Karatsuba: $\approx$ 100 lines ANSI C
- Smaller look-up table: 0.5 kilobytes vs. 128 kilobytes
- Faster: speedup by 20-30% for $n = 2^i$ $(6 < i < 18)$

Complexity of one recursive loop for input size s

	XOR	Read	Write
Kara	$5s$	$6s$	$7s$
MVP	$3.5s$	$6s$	$4s$

Other bases

Dual and triangular bases

Polynomial basis P and a secondary basis S (i.e., dual or triangular) can be combined for multiplication

If the multiplier inputs A and B are w.r.t. S and P respectively, then the following holds

$$\begin{aligned} \tilde{C} &= (\tilde{A}, \tilde{A}x, \cdots, \tilde{A}x^{n-1})B \\ &= \tilde{\mathbf{T}}B \end{aligned}$$

Remarks:

- Toeplitz matrix $\tilde{\mathbf{T}}$ is obtained directly
- No extra arithmetic is needed for $\tilde{C}$

Normal basis

- Optimal normal bases (ONBs) are known to have a lower arithmetic complexity
- For multiplication over GF(2^n) using an ONB, the conventional approach requires $O(n^2)$ arithmetic operations over GF(2)
- For type I ONB, it can be shown that

$$C' = (\mathbf{T}' + \mathbf{R}')B'$$

- For type II ONB:

$$C' = (\mathbf{T}' + \mathbf{H}')B'$$

ONB	$\#AND$	$\#XOR$	Gate delay
I	$n^{\log_2 3} + n$	$5.5n^{\log_2 3} - 4n - 0.5$	$(2\log_2 n + 1)T_x + T_A$
II	$2n^{\log_2 3}$	$11n^{\log_2 3} - 12n + 1$	$(2\log_2 n + 1)T_x + T_A$

Summary

- Presented a matrix-vector product (MVP) based approach to subquadratic multiplication over GF(2^n)
- Compared to the Karatsuba based scheme, MVP approach reduces gate count by 8% and gate delay by 33%
- For software implementation, we obtained 20-30% speedup for $n = 2^i$ $(6 < i < 18)$
- The MVP approach can be used with dual, triangular and normal bases
- It can be applied to non-binary fields
- Details are in tech reports at www.cacr.math.uwaterloo.ca