New Bit-Level Serial $GF(2^m)$ Multiplication Using Polynomial Basis

Hayssam El-Razouk and *Arash Reyhani-Masoleh*

Presented by: Arash Reyhani-Masoleh
Department of Electrical and Computer Engineering
Western University, London, Ontario, Canada

22nd IEEE Symposium on Computer Arithmetic, 2015

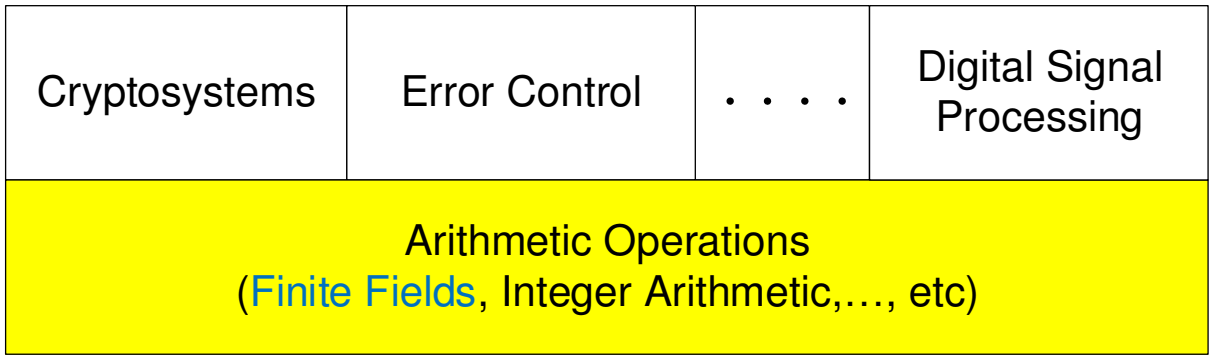
Outline

Outline

Applications of Finite Fields

Many applications use arithmetic operations over $GF(2^m)$

- Cryptography: elliptic curve, AES
- Error control coding: Reed-Solomon code
- Digital signal processing.



Outline

Polynomial Basis (PB) Representation

- There are different bases to represent a field element.
  - Polynomial basis, normal basis, dual basis, etc.
- PB offers efficient multiplications compared to other bases.
- To construct a PB:
  - Find $p(x)$: an irreducible polynomial of degree m over $GF(2)$.
  - Then, $\{\alpha^{m-1}, \ldots, \alpha, 1\}$ is known as the PB, where $p(\alpha) = 0$.
  - Any field element $A = (a_{m-1}, \cdots, a_1, a_0) \in GF(2^m)$, $a_i \in \{0, 1\}$, is represented w.r.t. the PB as

$$A = \sum_{i=0}^{m-1} a_i \alpha^i.$$

Polynomial Basis (PB) Representation

- There are different bases to represent a field element.
  - Polynomial basis, normal basis, dual basis, etc.
- PB offers efficient multiplications compared to other bases.
- To construct a PB:
  - Find $p(x)$: an irreducible polynomial of degree m over $GF(2)$.
  - Then, $\{\alpha^{m-1}, \ldots, \alpha, 1\}$ is known as the PB, where $p(\alpha) = 0$.
  - Any field element $A = (a_{m-1}, \cdots, a_1, a_0) \in GF(2^m)$, $a_i \in \{0, 1\}$, is represented w.r.t. the PB as

$$A = \sum_{i=0}^{m-1} a_i \alpha^i.$$

Polynomial Basis (PB) Representation

- There are different bases to represent a field element.
  - Polynomial basis, normal basis, dual basis, etc.
- PB offers efficient multiplications compared to other bases.
- To construct a PB:
  - Find $p(x)$: an irreducible polynomial of degree m over $GF(2)$.
  - Then, $\{\alpha^{m-1}, \ldots, \alpha, 1\}$ is known as the PB, where $p(\alpha) = 0$.
  - Any field element $A = (a_{m-1}, \cdots, a_1, a_0) \in GF(2^m)$, $a_i \in \{0, 1\}$, is represented w.r.t. the PB as

$$A = \sum_{i=0}^{m-1} a_i \alpha^i.$$

Outline

Arithmetic Operations over $GF(2^m)$

Let A and B be two field elements represented in the PB.

- **Addition**, $A+B$, is bit-wise XOR operations of their coordinates.

$$\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A} + \underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B} = \underbrace{\sum_{k=0}^{m-1} (a_k \oplus b_k) \alpha^k}_{A+B}.$$

- **Finite field Multiplication**, $C = A \cdot B \bmod p(\alpha)$, is more complex than addition.

1. Polynomial multiplication of A and B.

$$\left(\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A} \right) \cdot \left(\underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B} \right) = \underbrace{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \alpha^{i+j}}_{A \cdot B}.$$

2. Modulo reduction of α^{i+j} if $i+j \geq m$.

Arithmetic Operations over $GF(2^m)$

Let A and B be two field elements represented in the PB.

- **Addition**, $A+B$, is bit-wise XOR operations of their coordinates.

$$\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A} + \underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B} = \underbrace{\sum_{k=0}^{m-1} (a_k \oplus b_k) \alpha^k}_{A+B}.$$

- **Finite field Multiplication**, $C = A \cdot B \bmod p(\alpha)$, is more complex than addition.

1. Polynomial multiplication of A and B.

$$\left(\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A}\right) \cdot \left(\underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B}\right) = \underbrace{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \alpha^{i+j}}_{A \cdot B}.$$

2. Modulo reduction of α^{i+j} if $i+j \geq m$.

Arithmetic Operations over $GF(2^m)$

Let A and B be two field elements represented in the PB.

- **Addition**, $A+B$, is bit-wise XOR operations of their coordinates.

$$\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A} + \underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B} = \underbrace{\sum_{k=0}^{m-1} (a_k \oplus b_k) \alpha^k}_{A+B}.$$

- **Finite field Multiplication**, $C = A \cdot B \bmod p(\alpha)$, is more complex than addition.

1. Polynomial multiplication of A and B.

$$\left(\underbrace{\sum_{i=0}^{m-1} a_i \alpha^i}_{A}\right) \cdot \left(\underbrace{\sum_{j=0}^{m-1} b_j \alpha^j}_{B}\right) = \underbrace{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \alpha^{i+j}}_{A \cdot B}.$$

2. Modulo reduction of α^{i+j} if $i+j \geq m$.

Arithmetic Operations over $GF(2^m)$- Continued

- Exponentiation is computed by repeating squaring and multiplication.

$$A^e = \prod_{i=0}^{m-1} A^{e_i 2^i}.$$

- Exponent in radix-2 is $e = \sum_{i=0}^{m-1} e_i 2^i$.
- Inverse obtained if $e = 2^m - 2$ (Fermat's Little Theorem).

Therefore, **multiplication** is the main arithmetic operation.

Arithmetic Operations over $GF(2^m)$- Continued

- Exponentiation is computed by repeating squaring and multiplication.

$$A^e = \prod_{i=0}^{m-1} A^{e_i 2^i}.$$

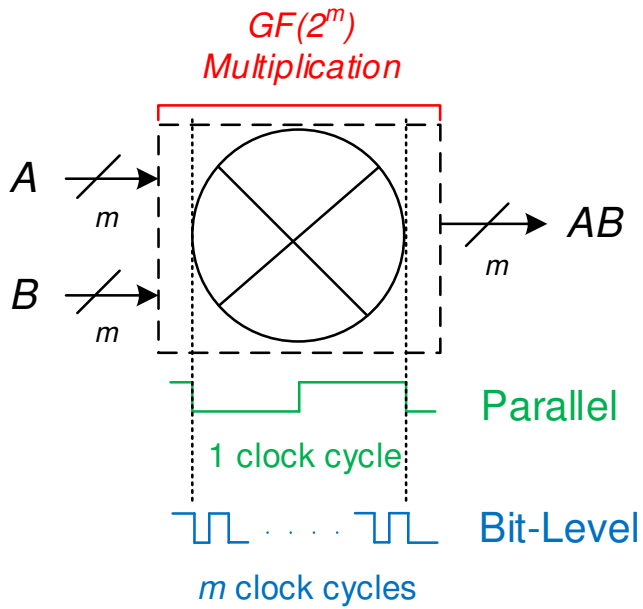
- Exponent in radix-2 is $e = \sum_{i=0}^{m-1} e_i 2^i$.
- Inverse obtained if $e = 2^m - 2$ (Fermat's Little Theorem).

Therefore, **multiplication** is the main arithmetic operation.

Categories of $GF(2^m)$ Multipliers

There are two general categories for $GF(2^m)$ multipliers:

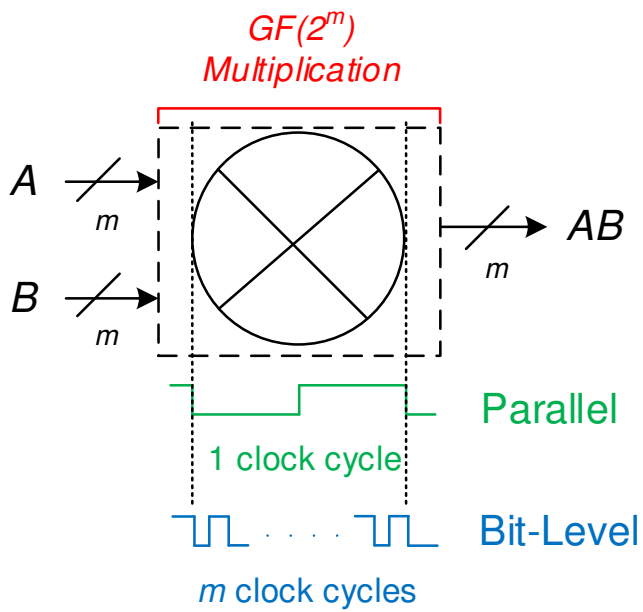
- Parallel computations.
  - High area and throughput
  - Result in a single clock cycle.
  - Attractive for high performance applications.
- Bit-level (BL) computations.
  - Lowest area
  - Result in m clock cycles
  - Attractive for resource constrained applications.



Categories of $GF(2^m)$ Multipliers

There are two general categories for $GF(2^m)$ multipliers:

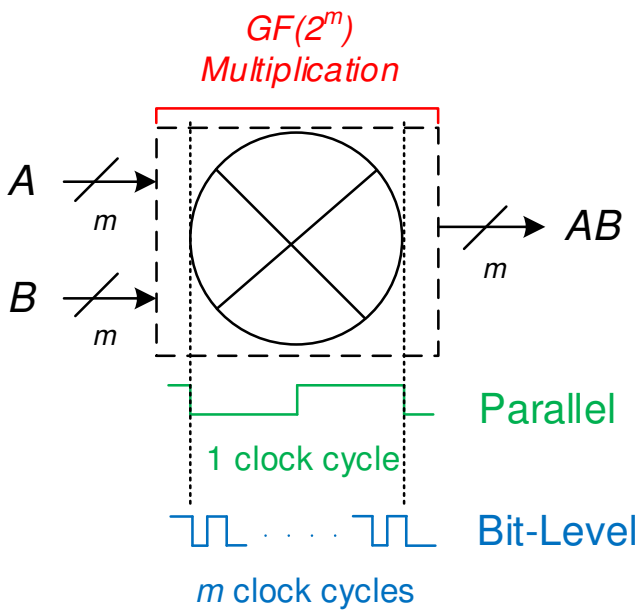
- Parallel computations.
  - High area and throughput
  - Result in a single clock cycle.
  - Attractive for high performance applications.
- Bit-level (BL) computations.
  - Lowest area
  - Result in m clock cycles
  - Attractive for resource constrained applications.



Categories of $GF(2^m)$ Multipliers

There are two general categories for $GF(2^m)$ multipliers:

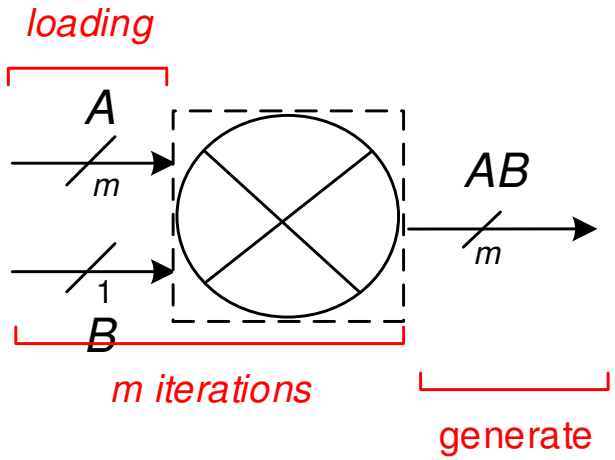
- Parallel computations.
  - High area and throughput
  - Result in a single clock cycle.
  - Attractive for high performance applications.
- Bit-level (BL) computations.
  - Lowest area
  - Result in m clock cycles
  - Attractive for resource constrained applications.



Bit-Level Multiplication Schemes

1. Bit-level Serial-in Parallel-out (BL-SIPO) [1]:

- One input, $A \in GF(2^m)$, is pre-loaded before computation.
- Another input, B, enters bit-by-bit during computation.
- Output is generated in parallel after m clock cycles.

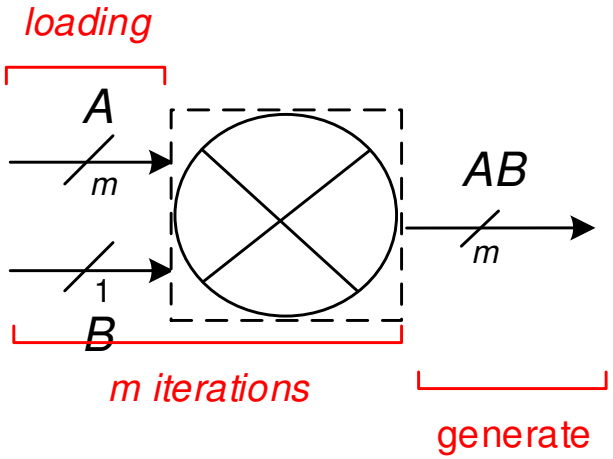


[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989

Bit-Level Multiplication Schemes

1. Bit-level Serial-in Parallel-out (BL-SIPO) [1]:

- One input, $A \in GF(2^m)$, is pre-loaded before computation.
- Another input, B, enters bit-by-bit during computation.
- Output is generated in parallel after m clock cycles.

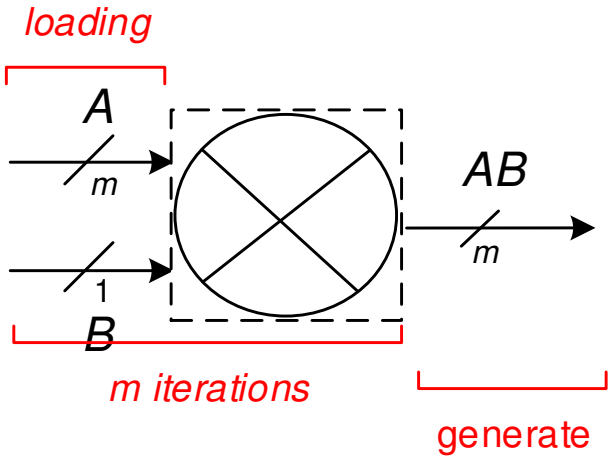


[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989

Bit-Level Multiplication Schemes

1. Bit-level Serial-in Parallel-out (BL-SIPO) [1]:

- One input, $A \in GF(2^m)$, is pre-loaded before computation.
- Another input, B, enters bit-by-bit during computation.
- Output is generated in parallel after m clock cycles.

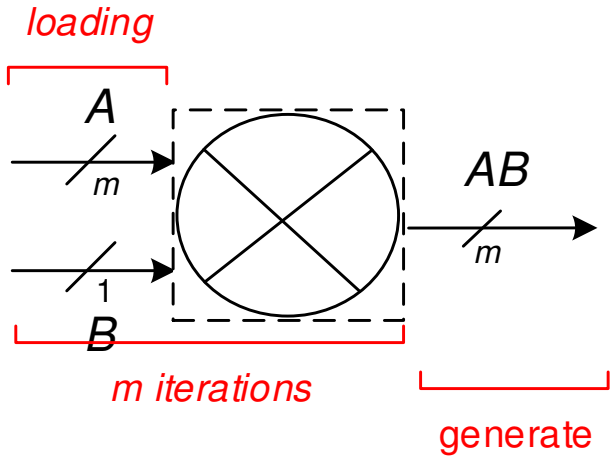


[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989

Bit-Level Multiplication Schemes

1. Bit-level Serial-in Parallel-out (BL-SIPO) [1]:

- One input, $A \in GF(2^m)$, is pre-loaded before computation.
- Another input, B, enters bit-by-bit during computation.
- Output is generated in parallel after m clock cycles.

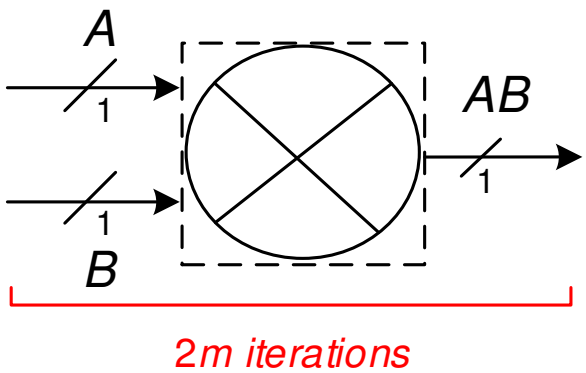


[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989

Bit-Level Multiplication Schemes - Continued

2. Bit-level Serial-in Serial-out (BL-SISO) [2, 3]:

- Inputs are entered bit-by-bit
- Output is generated bit-by-bit
- It takes $2m$ clock cycles to generate the result.

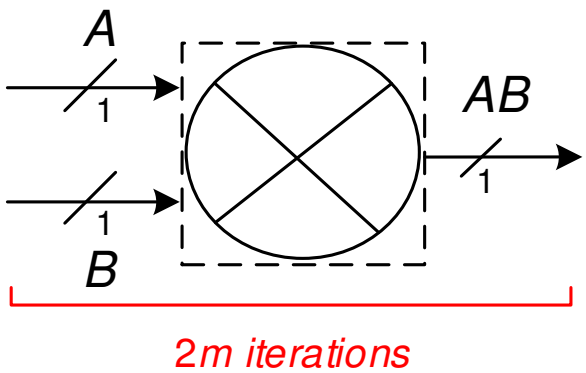


[2] M. Hasan and V. Bhargava, "Division and Bit-Serial Multiplication over GF(q^m)," *Computers and Digital Techniques, IEE Proceedings E*, vol. 139, pp. 230–236, May 1992
[3] A. Al-Khoraidly and M. K. Ibrahim, "Finite field serial-serial multiplication/reduction structure and method," Apr. 2009

Bit-Level Multiplication Schemes - Continued

2. Bit-level Serial-in Serial-out (BL-SISO) [2, 3]:

- Inputs are entered bit-by-bit
- Output is generated bit-by-bit
- It takes $2m$ clock cycles to generate the result.

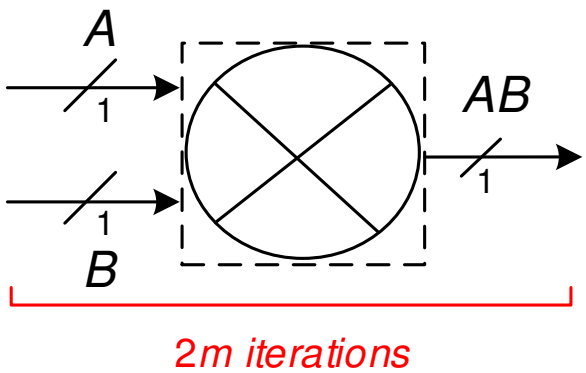


[2] M. Hasan and V. Bhargava, "Division and Bit-Serial Multiplication over GF(q^m)," *Computers and Digital Techniques, IEE Proceedings E*, vol. 139, pp. 230–236, May 1992
[3] A. Al-Khoraidly and M. K. Ibrahim, "Finite field serial-serial multiplication/reduction structure and method," Apr. 2009

Bit-Level Multiplication Schemes - Continued

2. Bit-level Serial-in Serial-out (BL-SISO) [2, 3]:

- Inputs are entered bit-by-bit
- Output is generated bit-by-bit
- It takes $2m$ clock cycles to generate the result.

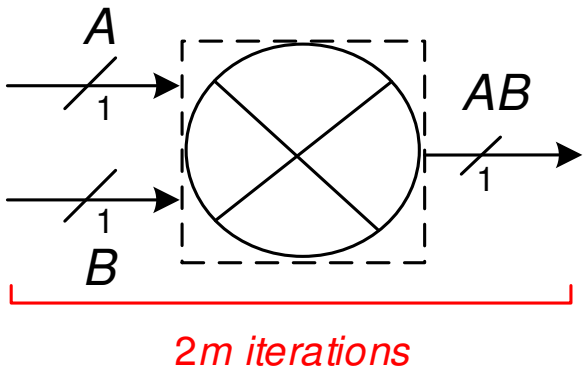


[2] M. Hasan and V. Bhargava, "Division and Bit-Serial Multiplication over GF(q^m)," *Computers and Digital Techniques, IEE Proceedings E*, vol. 139, pp. 230–236, May 1992
[3] A. Al-Khoraidly and M. K. Ibrahim, "Finite field serial-serial multiplication/reduction structure and method," Apr. 2009

Bit-Level Multiplication Schemes - Continued

2. Bit-level Serial-in Serial-out (BL-SISO) [2, 3]:

- Inputs are entered bit-by-bit
- Output is generated bit-by-bit
- It takes $2m$ clock cycles to generate the result.

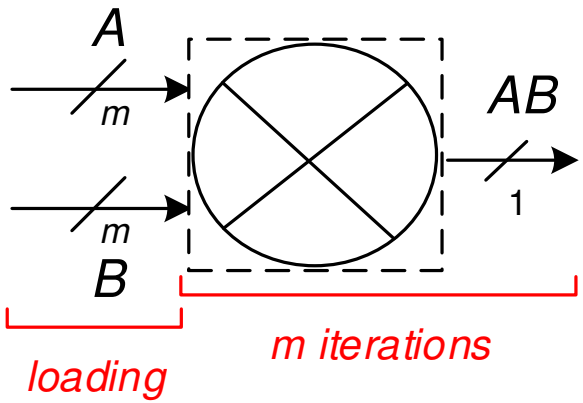


[2] M. Hasan and V. Bhargava, "Division and Bit-Serial Multiplication over GF(q^m)," *Computers and Digital Techniques, IEE Proceedings E*, vol. 139, pp. 230–236, May 1992
[3] A. Al-Khoraidly and M. K. Ibrahim, "Finite field serial-serial multiplication/reduction structure and method," Apr. 2009

Bit-Level Multiplication Schemes - Continued

3. Bit-level Parallel-in Serial-out (BL-PISO) [4].

- Two inputs are pre-loaded before computations.
- One output bit is generated per a clock cycle.
- All output bits are generated after m clock cycles.

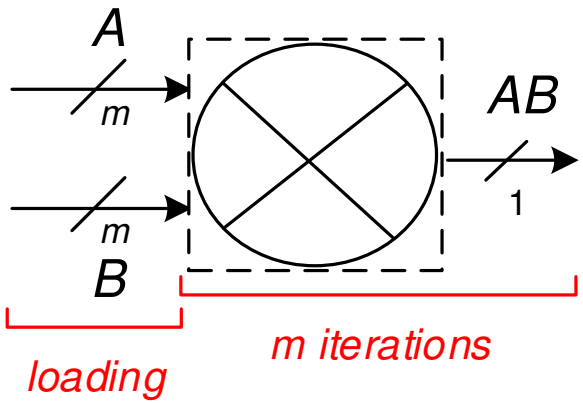


[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008

Bit-Level Multiplication Schemes - Continued

3. Bit-level Parallel-in Serial-out (BL-PISO) [4].

- Two inputs are pre-loaded before computations.
- One output bit is generated per a clock cycle.
- All output bits are generated after m clock cycles.

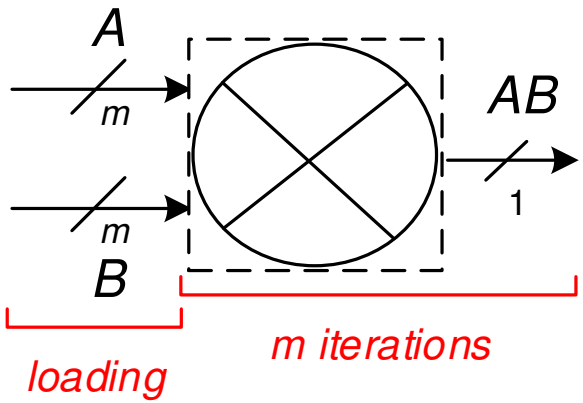


[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008

Bit-Level Multiplication Schemes - Continued

3. Bit-level Parallel-in Serial-out (BL-PISO) [4].

- Two inputs are pre-loaded before computations.
- One output bit is generated per a clock cycle.
- All output bits are generated after m clock cycles.

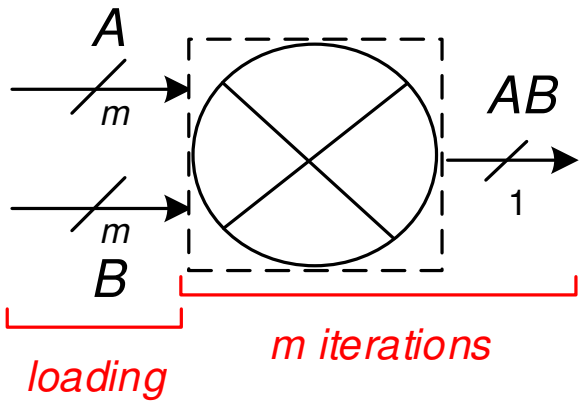


[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008

Bit-Level Multiplication Schemes - Continued

3. Bit-level Parallel-in Serial-out (BL-PISO) [4].

- Two inputs are pre-loaded before computations.
- One output bit is generated per a clock cycle.
- All output bits are generated after m clock cycles.

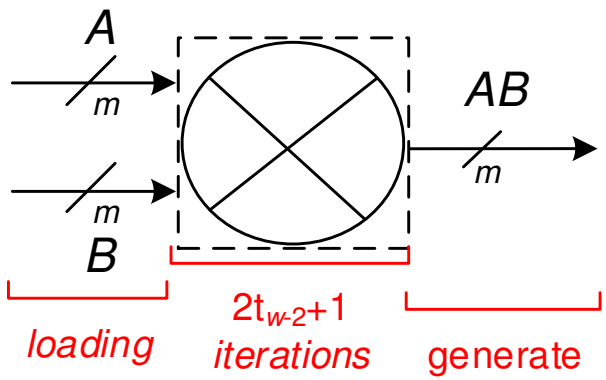


[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008

Bit-Level Multiplication Schemes - Continued

4. Bit-level Parallel-in Parallel-out (BL-PIPO) [5]:

- Two inputs are pre-loaded before computations.
- Output is generated in parallel after $2t_{\omega-2}+1$ clock cycles.
  - Field polynomial is $p(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$.

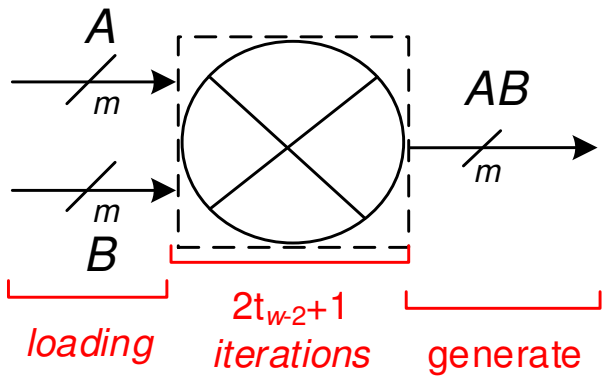


[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Bit-Level Multiplication Schemes - Continued

4. Bit-level Parallel-in Parallel-out (BL-PIPO) [5]:

- Two inputs are pre-loaded before computations.
- Output is generated in parallel after $2t_{\omega-2}+1$ clock cycles.
  - Field polynomial is $p(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$.

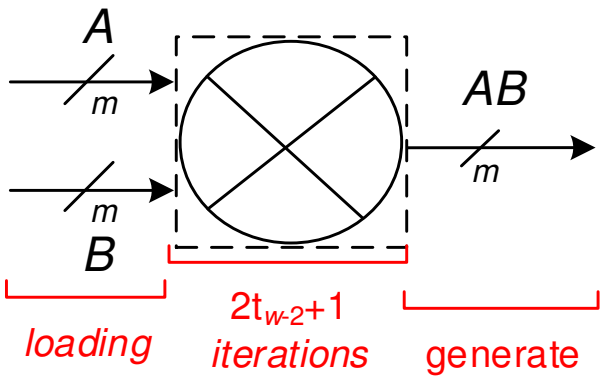


[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Bit-Level Multiplication Schemes - Continued

4. Bit-level Parallel-in Parallel-out (BL-PIPO) [5]:

- Two inputs are pre-loaded before computations.
- Output is generated in parallel after $2t_{\omega-2}+1$ clock cycles.
  - Field polynomial is $p(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$.



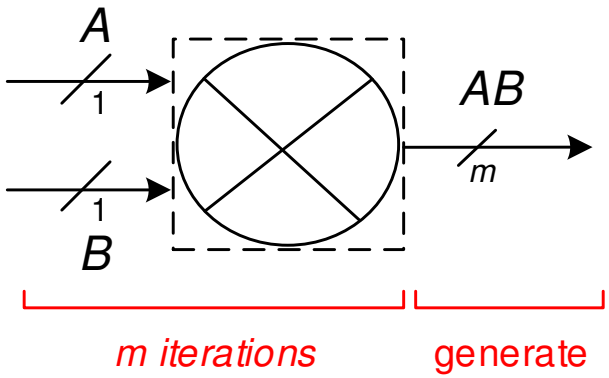
[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

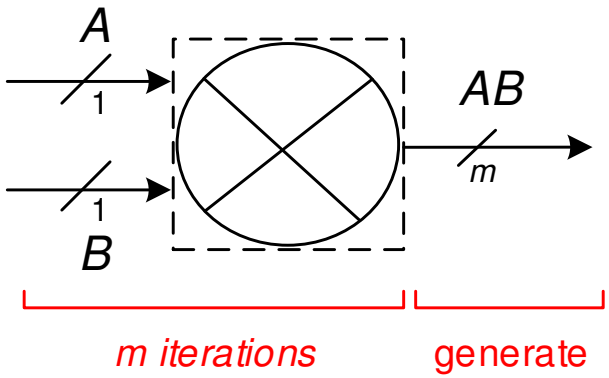
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

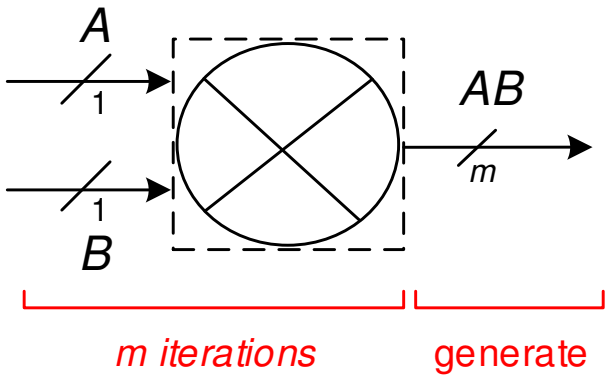
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

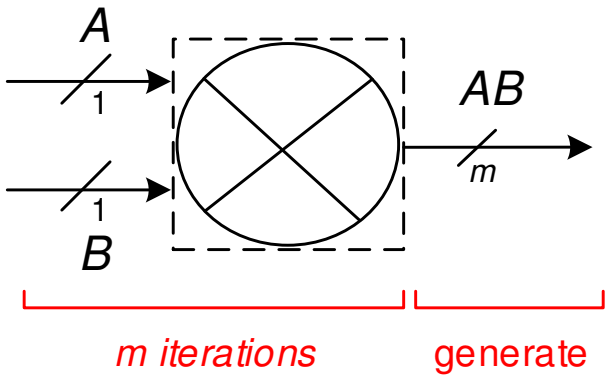
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

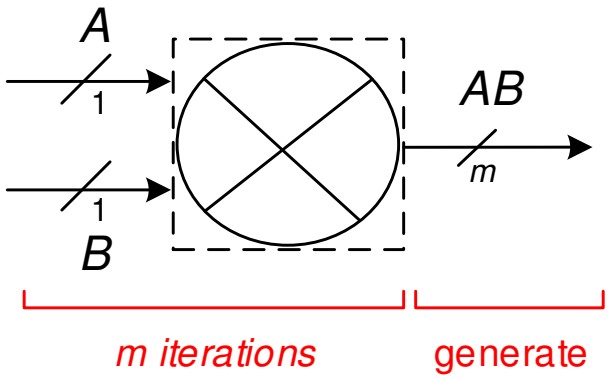
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

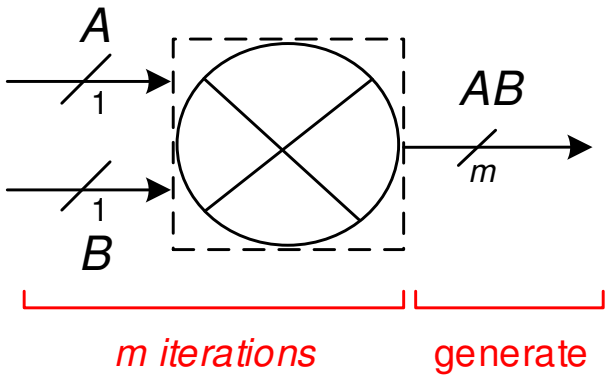
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

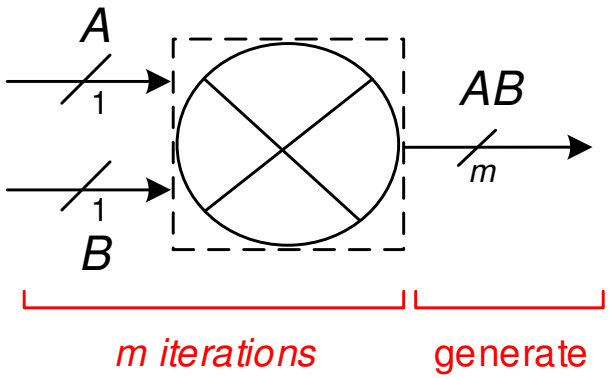
[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Bit-Level Multiplication Schemes - Continued

5. Bit-level Fully Serial-in Parallel-out (BL-FSIPO):

No pre-loading is required; multiplication is performed while inputs enter.

- Requires low input data-path resources.
- Output generated in parallel after m cycles.
- Improves throughput in input data-path constrained applications.
- Exists only for Normal Basis [6].



In this paper, a new BL-FSIPO multiplier using PB is proposed.

[6] G.-L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1383–1386, 1989

Outline

MSB-first Construction of Field Elements

Let $A = (a_{m-1}, \cdots, a_1, a_0) = \sum_{i=0}^{m-1} a_i \alpha^i \in GF(2^m)$

The following equation constructs A bit-by-bit starting at MSB-first.

$$A^{(i)} = a_{m-1-i} + A^{(i-1)}\alpha.$$

- Iterate $A^{(i)} \in GF(2^m)$ from $i = 0$ to $m-1$.
  - $A^{(-1)} = 0 = (0, \cdots, 0, 0)$.
  - $A^{(0)} = a_{m-1} = (0, \cdots 0, a_{m-1})$
  - $A^{(1)} = a_{m-1}\alpha + a_{m-2} = (0, \cdots, a_{m-1}, a_{m-2})$
  - $A = A^{(m-1)} = (a_{m-1}, \cdots, a_1, a_0)$.

MSB-first Construction of Field Elements

Let $A = (a_{m-1}, \cdots, a_1, a_0) = \sum_{i=0}^{m-1} a_i \alpha^i \in GF(2^m)$

The following equation constructs A bit-by-bit starting at MSB-first.

$$A^{(i)} = a_{m-1-i} + A^{(i-1)}\alpha.$$

- Iterate $A^{(i)} \in GF(2^m)$ from $i = 0$ to $m-1$.
  - $A^{(-1)} = 0 = (0, \cdots, 0, 0)$.
  - $A^{(0)} = a_{m-1} = (0, \cdots 0, a_{m-1})$
  - $A^{(1)} = a_{m-1}\alpha + a_{m-2} = (0, \cdots, a_{m-1}, a_{m-2})$
  - $A = A^{(m-1)} = (a_{m-1}, \cdots, a_1, a_0)$.

MSB-first Construction of Field Elements

Let $A = (a_{m-1}, \cdots, a_1, a_0) = \sum_{i=0}^{m-1} a_i \alpha^i \in GF(2^m)$

The following equation constructs A bit-by-bit starting at MSB-first.

$$A^{(i)} = a_{m-1-i} + A^{(i-1)}\alpha.$$

- Iterate $A^{(i)} \in GF(2^m)$ from $i = 0$ to $m-1$.
  - $A^{(-1)} = 0 = (0, \cdots, 0, 0)$.
  - $A^{(0)} = a_{m-1} = (0, \cdots 0, a_{m-1})$
  - $A^{(1)} = a_{m-1}\alpha + a_{m-2} = (0, \cdots, a_{m-1}, a_{m-2})$
  - $A = A^{(m-1)} = (a_{m-1}, \cdots, a_1, a_0)$.

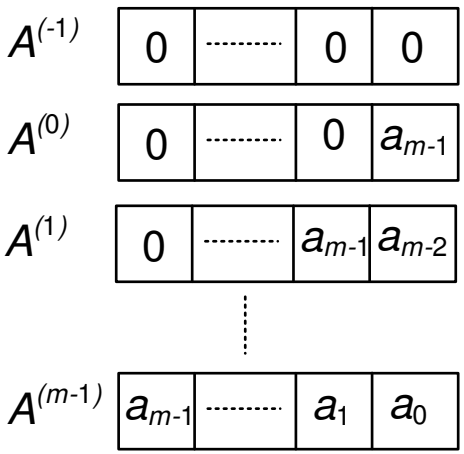
MSB-first Construction of Field Elements

Let $A = (a_{m-1}, \cdots, a_1, a_0) = \sum_{i=0}^{m-1} a_i \alpha^i \in GF(2^m)$

The following equation constructs A bit-by-bit starting at MSB-first.

$$A^{(i)} = a_{m-1-i} + A^{(i-1)}\alpha.$$

- Iterate $A^{(i)} \in GF(2^m)$ from $i = 0$ to $m-1$.
  - $A^{(-1)} = 0 = (0, \cdots, 0, 0)$.
  - $A^{(0)} = a_{m-1} = (0, \cdots 0, a_{m-1})$
  - $A^{(1)} = a_{m-1}\alpha + a_{m-2} = (0, \cdots, a_{m-1}, a_{m-2})$
  - $A = A^{(m-1)} = (a_{m-1}, \cdots, a_1, a_0)$.



Key Formulation for the Proposed Multiplier

Proposition 1: Let A and B be two arbitrary $GF(2^m)$ elements represented in the PB generated by the degree m irreducible polynomial $p(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$ with ω nonzero terms. Let us define $C_i = A^{(i)} B^{(i)} \bmod p(\alpha)$, where $A^{(i)}$ and $B^{(i)}$ are given as follows

$$A^{(i)} = a_{m-1-i} + A^{(i-1)}\alpha$$
$$B^{(i)} = b_{m-1-i} + B^{(i-1)}\alpha$$

where $i = 0, \ldots, m-1$, $A^{(-1)} = B^{(-1)} = 0$, and α is the root of $p(x)$. Then, based on the following recurrence on C_i, one can compute the multiplication of A and B, as $AB = C_{m-1}$:

$$C_i = a_{m-1-i} b_{m-1-i} + \left(a_{m-1-i} B^{(i-1)} + b_{m-1-i} A^{(i-1)} \right) \alpha + C_{i-1}\alpha^2 \bmod p(\alpha).$$

$i = 0, \ldots, m-1$, where $C_{-1} = A^{(-1)} B^{(-1)} \bmod p(\alpha) = 0$.

Outline

Proposed Architecture

$$C_i = a_{m-1-i}b_{m-1-i} + \left(a_{m-1-i}B^{(i-1)} + b_{m-1-i}A^{(i-1)}\right)\alpha + C_{i-1}\alpha^2 \bmod p(\alpha).$$

- Multiplication by α is just a left shift.
- Only reduction is due to multiply by α^2.
- Three registers X, Y, and Z are initially cleared.
- Inputs are entered to X and Y serially from MSB.
- After m clock cycles Z contains $AB \bmod p(\alpha)$.

Proposed Architecture

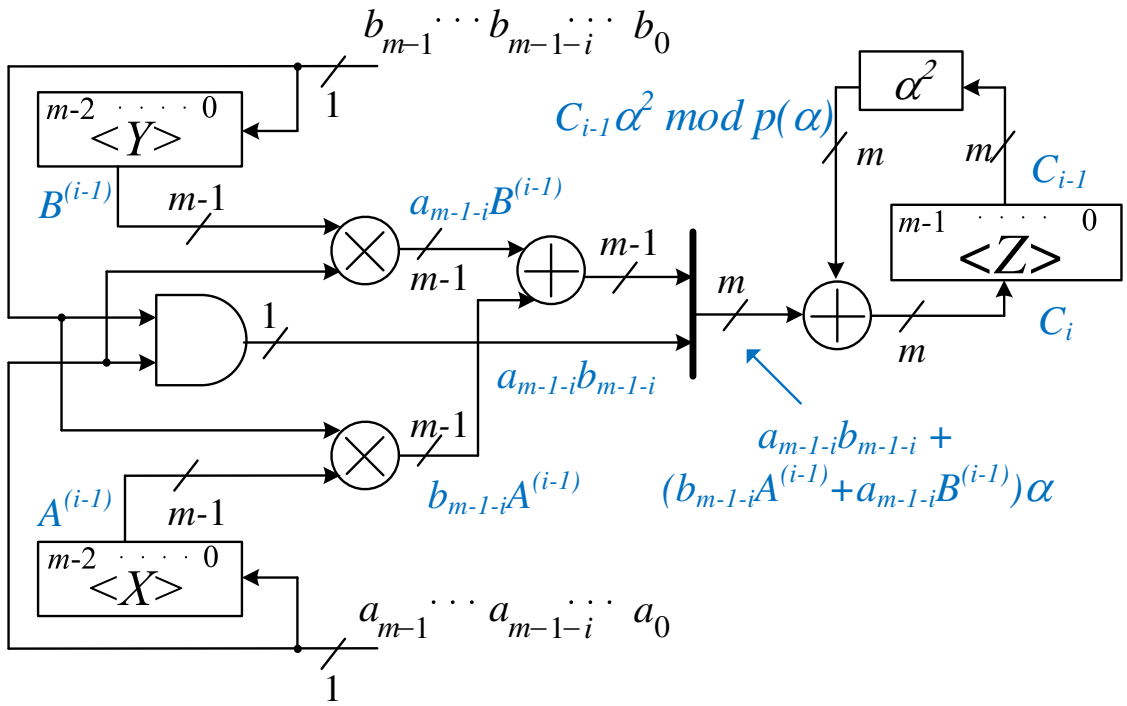
$$C_i = a_{m-1-i}b_{m-1-i} + \left(a_{m-1-i}B^{(i-1)} + b_{m-1-i}A^{(i-1)}\right)\alpha + C_{i-1}\alpha^2 \bmod p(\alpha).$$

- Multiplication by α is just a left shift.
- Only reduction is due to multiply by α^2.
- Three registers X, Y, and Z are initially cleared.
- Inputs are entered to X and Y serially from MSB.
- After m clock cycles Z contains $AB \bmod p(\alpha)$.

Proposed Architecture

$$C_i = a_{m-1-i}b_{m-1-i} + \left(a_{m-1-i}B^{(i-1)} + b_{m-1-i}A^{(i-1)}\right)\alpha + C_{i-1}\alpha^2 \bmod p(\alpha).$$

- Multiplication by α is just a left shift.
- Only reduction is due to multiply by α^2.
- Three registers X, Y, and Z are initially cleared.
- Inputs are entered to X and Y serially from MSB.
- After m clock cycles Z contains $AB \bmod p(\alpha)$.



Outline

Complexities

Proposition 2: The total number of gates and propagation delay (PD) in the proposed PB multiplier over $GF(2^m)$ are as follows:

$$\begin{cases} \#ANDs = 2m-1, & \#FFs = 3m-2, \\ \#XORs = 2m + N_{\alpha^2} - 1, \end{cases}$$

and

$$PD = \max\{T_{\alpha^2} + T_X, T_A + 2T_X\}.$$

- For the five NIST recommended fields for ECDSA, the complexities of multiplication by α^2 are as follows:

m	163	233	283	409	571
N_{α^2}	6	2	6	2	6
T_{α^2}	$2T_X$	T_X	T_X	T_X	T_X

Space Complexity Comparison

Multiplier	FF	AND	XOR	2 : 1 1-bit MUX Functional	2 : 1 1-bit MUX Parallel Loading
MSB BL-SIPO [1]	$2m$	m	$m+\omega-2$	0	m
LSB BL-SIPO [1]	$2m$	m	$m+\omega-2$	0	m
BL-PISO [4]	$3m+t_{\omega-2}-1$	$2m-1$	$(\omega-1)(m-1)+\omega-3+\sum_{i=1}^{\omega-2} t_i$	0	$2m$
PIPO [5]	$5m-1$	$\frac{m^2+m}{2}$	$\frac{m^2+m}{2}$	$4m$	$2m$
MSB BL-FSIPO (this work)	$3m-2$	$2m-1$	$2m+N_{\alpha^2}-1$	0	0

$p(x)=x^m+\sum_{i=1}^{\omega-2} x^{t_i}+1$ is the field polynomial.

- The area complexity of the proposed bit-level multiplier is
  - larger than the one proposed in BL-SIPO [1].
  - close to that of BL-PISO [4].
  - better than that of BL-PIPO [5].

[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989
[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008
[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Time Complexity Comparison

Multiplier	Propagation Delay	Serial Pre-loading Latency	Computation Latency
MSB BL-SIPO [1]	$T_A + T_X$	m	m
LSB BL-SIPO [1]	$T_A + T_X$	m	m
BL-PISO [4]	$T_A + \left(1 + \lceil \log_2(\omega - 1) \rceil + \lceil \log_2(m) \rceil\right) T_X$	m	m
PIPO [5]	$T_A + \lceil \log_2 m \rceil T_X + 2T_M$	m	$2t_{\omega-2} + 1$
MSB BL-FSIPO (this work)	$\max\{T_{\alpha^2} + T_X, T_A + 2T_X\}$	0	m

T_A, T_X, and T_M denote the delay in 2-inputs AND, XOR, and 2-to-1 MUX, respectively.
$p(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$ is the field polynomial.

- Speed of BL-FSIPO is independent of m, similar to BL-SIPO.
- BL-FSIPO has slightly lower speed compared to BL-SIPO.
- BL-FSIPO has higher speed compared to BL-PISO and BL-PIPO.

The proposed BL-FSIPO multiplier offers the highest throughput if inputs are serially loaded one bit at a time.

[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989
[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008
[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Comparisons: $p(x) = x^{233} + x^{74} + 1$

Multiplier	MPD *ns*	GE		Latency		TP/G @ 1 GHz	
		PIL	SIL	PIL	SIL	PIL	SIL
LSB BL-SIPO [1]	0.07	2973	2507	233	466	336	199
MSB BL-SIPO [1]	0.07	2973	2507	233	466	336	199
BL-PISO [4]	0.43	5484	4552	233	466	182	110
PIPO [5]	0.41	95759	94827	149	382	16	6
MSB BL-FSIPO	0.11	4129	4129	233	233	242	242

MPD = maximum propagation delay. **GE** = # NAND equivalence.
PIL = parallel input loading. **SIL** = serial input loading.
TP/G = normalized throughput.
Estimates based on 65nm STMicroelectronics standard CMOS library.

[1] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, 1989
[4] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases," in *Cryptographic Hardware and Embedded Systems - CHES 2008* (E. Oswald and P. Rohatgi, eds.), no. 5154 in Lecture Notes in Computer Science, pp. 300–314, Springer Berlin Heidelberg, Jan 2008
[5] J. Imaña, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, pp. 935–946, May 2011

Conclusion and Future Work

Conclusion:

- We have proposed a new MSB-first bit-level fully serial-in parallel-out PB multiplier.
- The proposed multiplier improves throughput in resource constrained environments with low data-path input.
- To the best of our knowledge, this is the first time that such a BL-FSIPO PB multiplier is proposed.

Future Work:

- Hardware implementations of the proposed architecture
- Digit level extension of an MSD-first DL-FSIPO PB multiplier.
- Construct an architecture for LSD-first DL-FSIPO PB multiplier.

Thank You!