Processor Arrays for $GF(2^m)$ Multiplication

Multithreaded/ Multicore Implementations

F. Gebali, PhD, PEng
Electrical & Computer Engineering
University of Victoria
Victoria, Canada

Outline

Dr. Gebali Research Interests

1. Finite field arithmetic for data encryption/decryption
2. Networks-on-Chip and System-on-Chip (SoC)
3. Communication Networks
4. Assistive technology and Apps

Motivation for Field Multiplication

1. Multiplication operation is basic to other operations
2. Applications in coding
3. Applications in public-key cryptosystems
4. Needed in embedded and mobile devices
5. Contradictory performance requirements and restrictions

Finite Fields $GF(2^m)$

Finite-Field Representations

1. Polynomial basis
2. Prime
3. Binary
4. Dual basis
5. Normal basis

Polynomial Basis

1. Finite field $GF(2^m)$ has 2^m elements
2. Elements are represented as polynomials of degree $< m$
3. Polynomial coefficients are 0 or 1
4. Arithmetic is done in $GF(2)$ using AND and XOR gates
5. There is an associated field polynomial of degree m

Finite-Field Arithmetic

1. Earlier designs are heuristic
2. Heuristic solutions have limited design space to explore
3. Formal approaches opens up the design space
4. Convert algorithm into Regular Iterative Algorithm (RIA)

Design Considerations

1. Number of processors (Area)
2. Processor word width (Area, Time)
3. Latecny,
4. Intercprocessor communications
5. Input and output data schemes

Problem Formulation

1. $GF(2^m)$ is defined using the irreducible polynomial:

$$Q(x) = x^m + q_{m-1}x^{m-1} + \cdots + q_2x^2 + q_1x + 1$$

2. NIST defines two trinomials:

$$Q(x) = x^{233} + x^{73} + 1 \qquad \text{and} \qquad Q(x) = x^{409} + x^{87} + 1$$

3. We shall consider the general trinomial

$$Q(x) = x^m + x^k + 1$$

Field Elements

1. Polynomial basis used to represent the field elements:

$$\{ \ 1, \quad \alpha, \quad \alpha^2, \quad \cdots, \quad \alpha^{m-1} \ \}$$

2. α is a root of $Q(x)$:

$$\alpha^m + \alpha^k + 1 = 0 \qquad \text{or} \qquad \alpha^m = \alpha^k + 1$$

Representation of Elements in $GF(2^m)$

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

$$B = \sum_{j=0}^{m-1} b_j \, \alpha^j$$

The product is given by:

$$C = A \times B = \left[\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i \, b_j \, \alpha^{i+j} \right] \bmod Q(\alpha)$$

$$= \sum_{k=0}^{m-1} c_k \, \alpha^k$$

Formal Algorithm Design

Main Steps

1. Convert algorithm to Regular Iterative Algorithm (RIA)
2. Obtain index dependencies of all algorithm variables
3. Obtain the dependence graph or computation domain
4. Determine timing and I/O restrictions
5. Determine scheduling functions
6. Determine projection directions

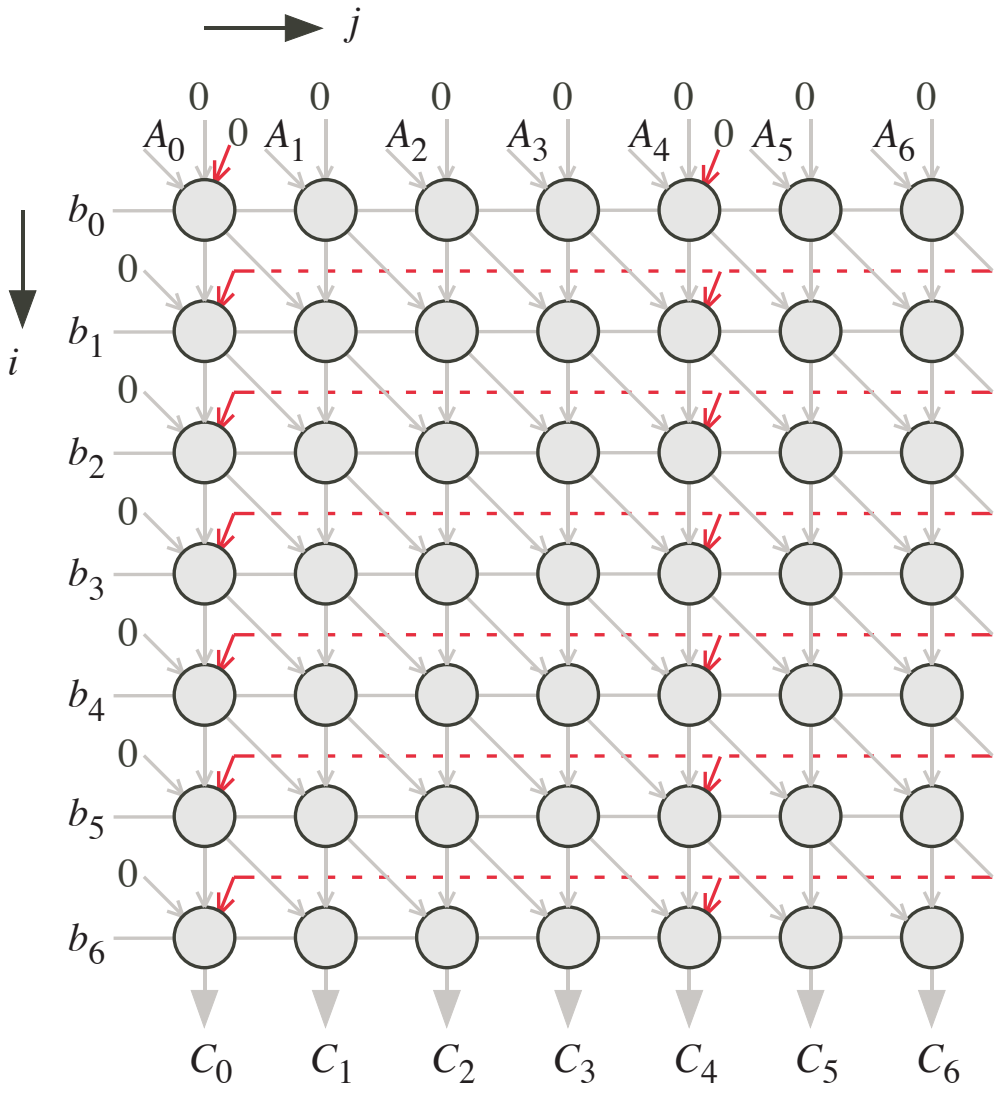
Progressive Multiplier Reduction (PMR)

$$
\begin{aligned}
C &= A \times B = \left[\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i \, b_j \, \alpha^{i+j} \right] \mod Q(\alpha) \\
&= \sum_{i=0}^{m-1} b_i \left[\alpha^i A \mod Q(\alpha) \right]
\end{aligned}
$$

Progressive Multiplier Reduction (PMR) Iterations

$$
\begin{aligned}
a_j^0 &= A_j \\
a_j^i &= a_{j-1}^{i-1} + a_{m-1}^{i-1} \quad \text{when } j = 0,\ k \\
a_j^i &= a_{j-1}^{i-1} + 0 \quad j \neq 0,\ k \\
c_j^i &= c_j^{i-1} + b_i a_j^i \\
C_j &= c_j^{m-1}
\end{aligned}
$$

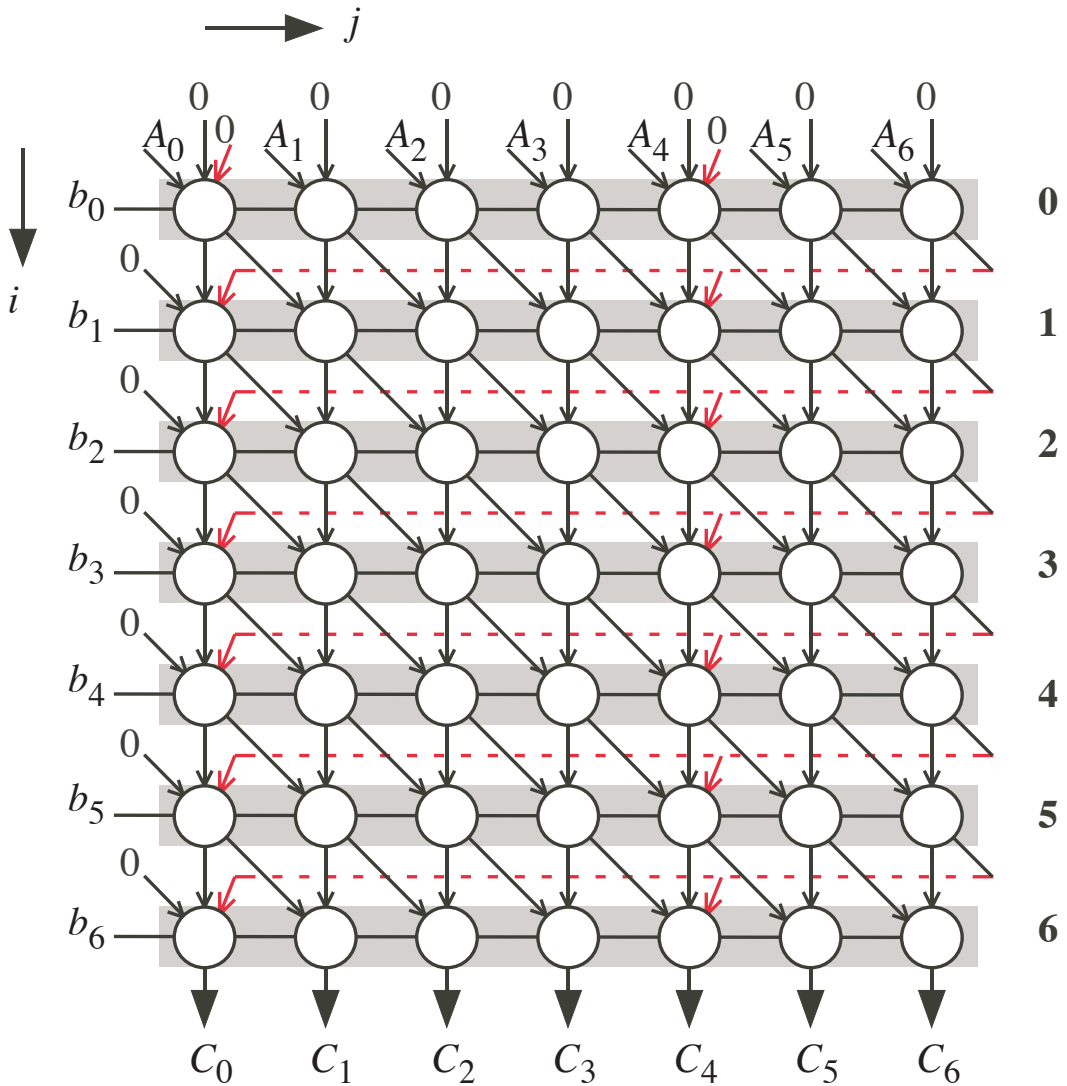
PMR Dependence Graph



PMR Timing: Linear Affine Scheduling

$$
\begin{aligned}
t(\mathbf{p}) &= \mathbf{s_1}\,\mathbf{p} - \gamma_1 = i \\
\mathbf{s_1} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \\
\gamma_1 &= 0
\end{aligned}
$$

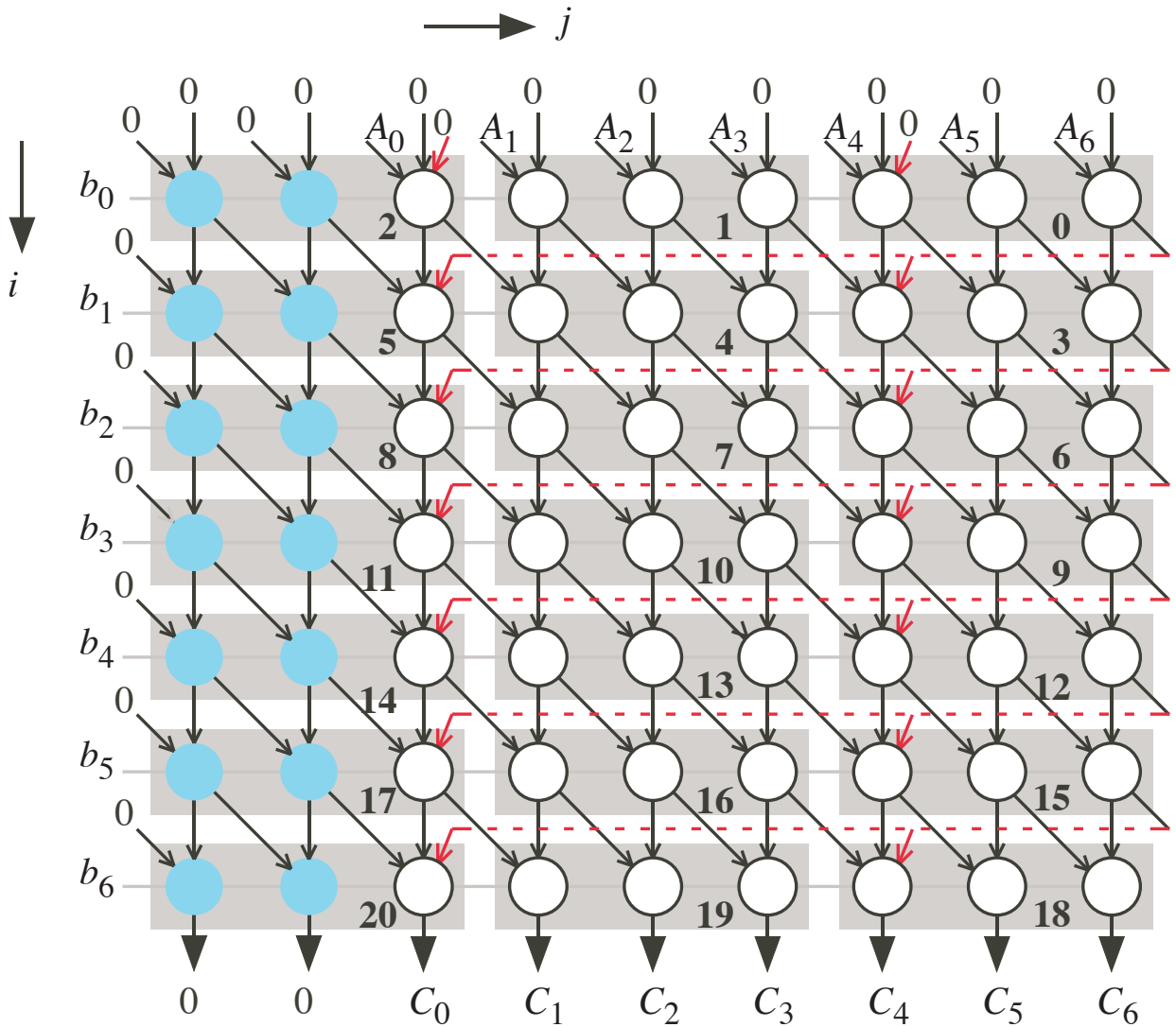
PMR Timing: Linear Affine Scheduling



PMR Timing: Non-Linear Scheduling

$$
\begin{aligned}
t(\mathbf{p}) &= \mathbf{s_2}\ \mathbf{p} - \gamma_2 \\
&= i \left\lceil \frac{m}{T} \right\rceil - \left\lfloor \frac{j + \mu_2}{T} \right\rfloor + \left\lfloor \frac{m}{T} \right\rfloor \\
\mathbf{s_2} &= \left[\left\lceil \tfrac{m}{T} \right\rceil \quad - \left\lfloor \tfrac{\cdot + \mu_2}{T} \right\rfloor \right] \\
\mu_2 &= w \left\lceil \frac{m}{T} \right\rceil - m \\
\gamma_2 &= - \left\lfloor \frac{m}{T} \right\rfloor
\end{aligned}
$$

PMR Timing: Non-Linear Scheduling



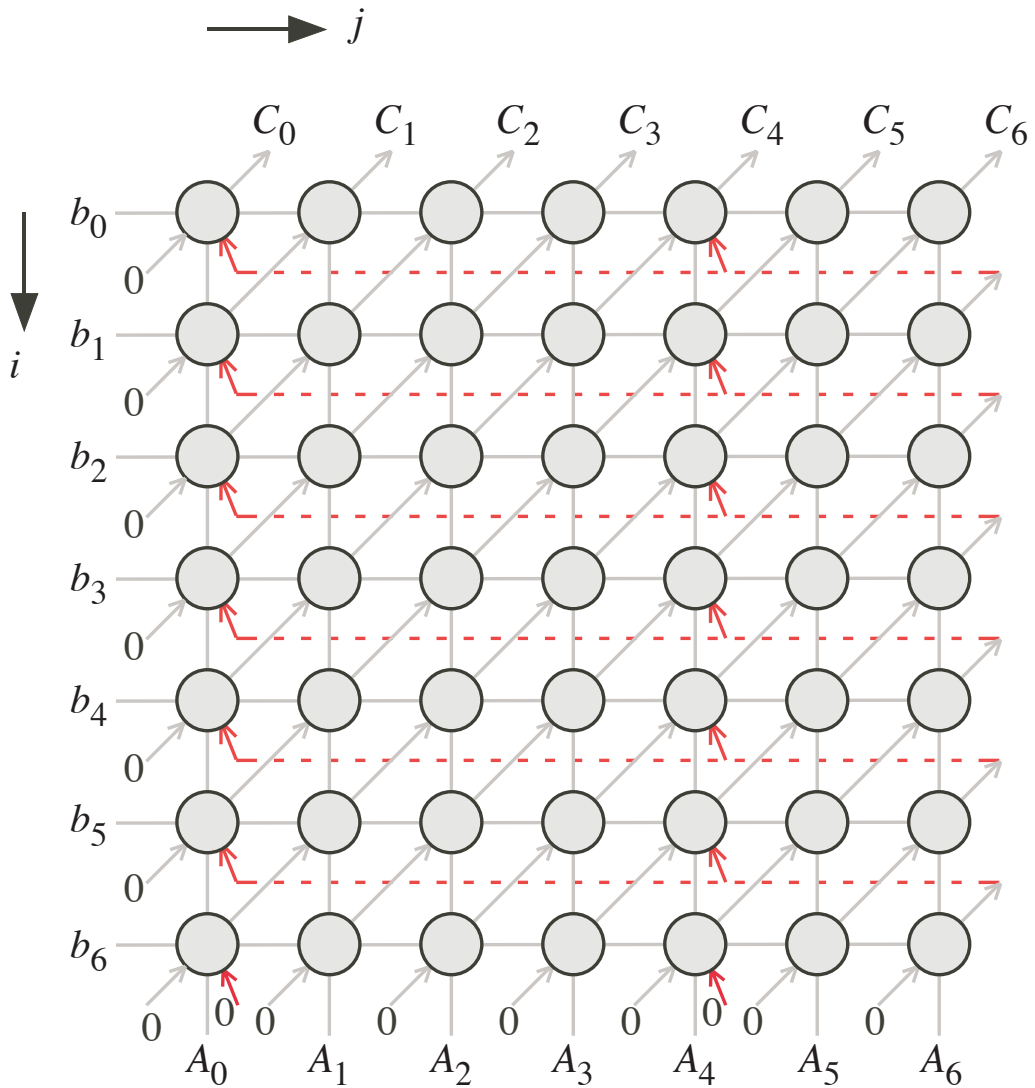
Progressive Product Reduction (PPR)

$$
\begin{aligned}
C &= A \times B = \left[\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i \, b_j \, \alpha^{i+j} \right] \mod Q(\alpha) \\
&= \sum_{i=0}^{m-1} \left[\alpha^i b_i A \mod Q(\alpha) \right]
\end{aligned}
$$

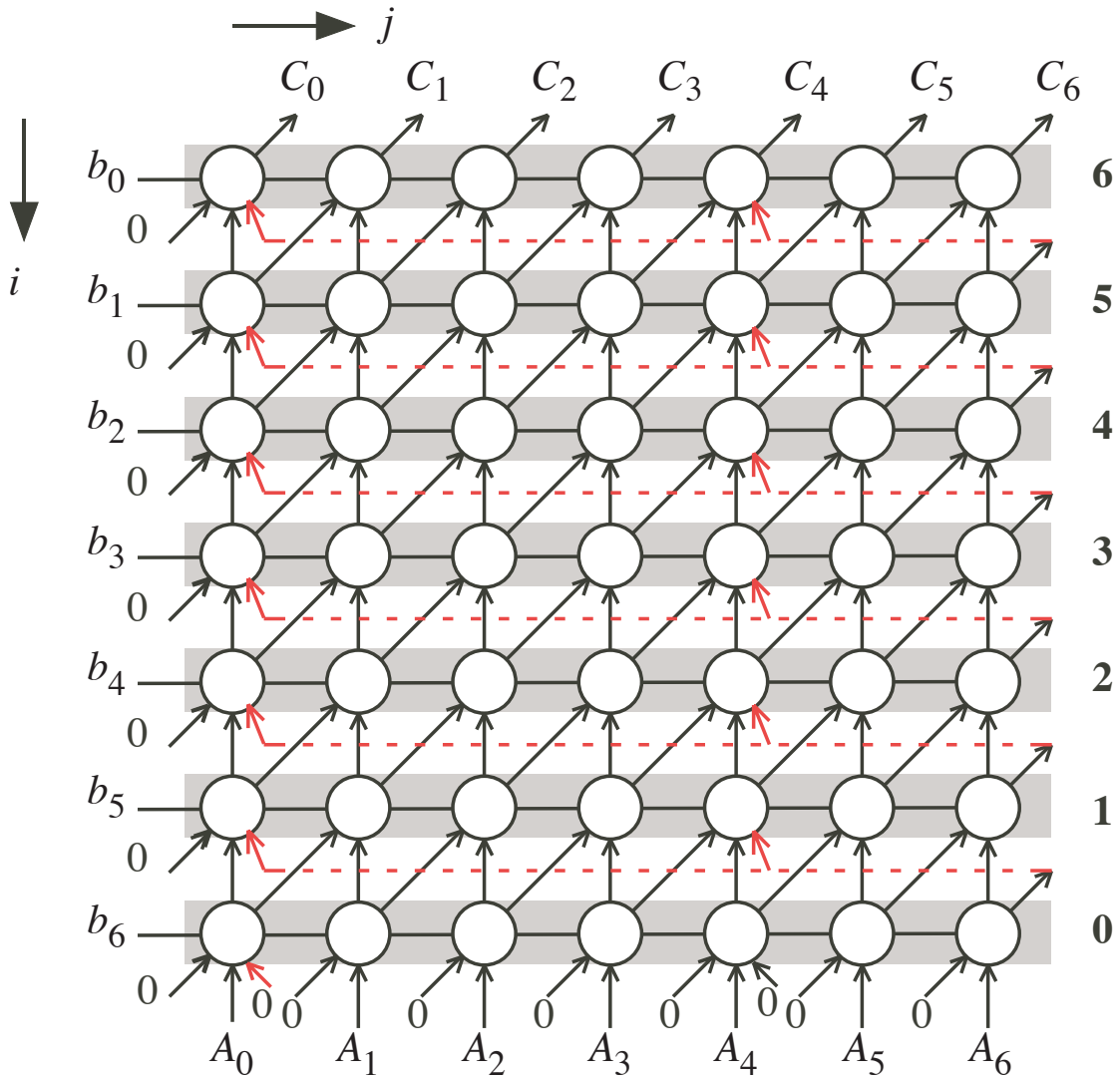
Progressive Product Reduction (PPR) Iterations

$$
\begin{aligned}
c_j^m &= 0 \\
c_j^i &= c_{j-1}^{i+1} + b_i a_j \\
&\quad j \neq 0,\ k \\
c_j^i &= c_{j-1}^{i+1} + b_i a_j + c_{m-1}^{i+1} \\
&\quad \text{for } j = 0,\ k \\
C_j &= c_j^0
\end{aligned}
$$

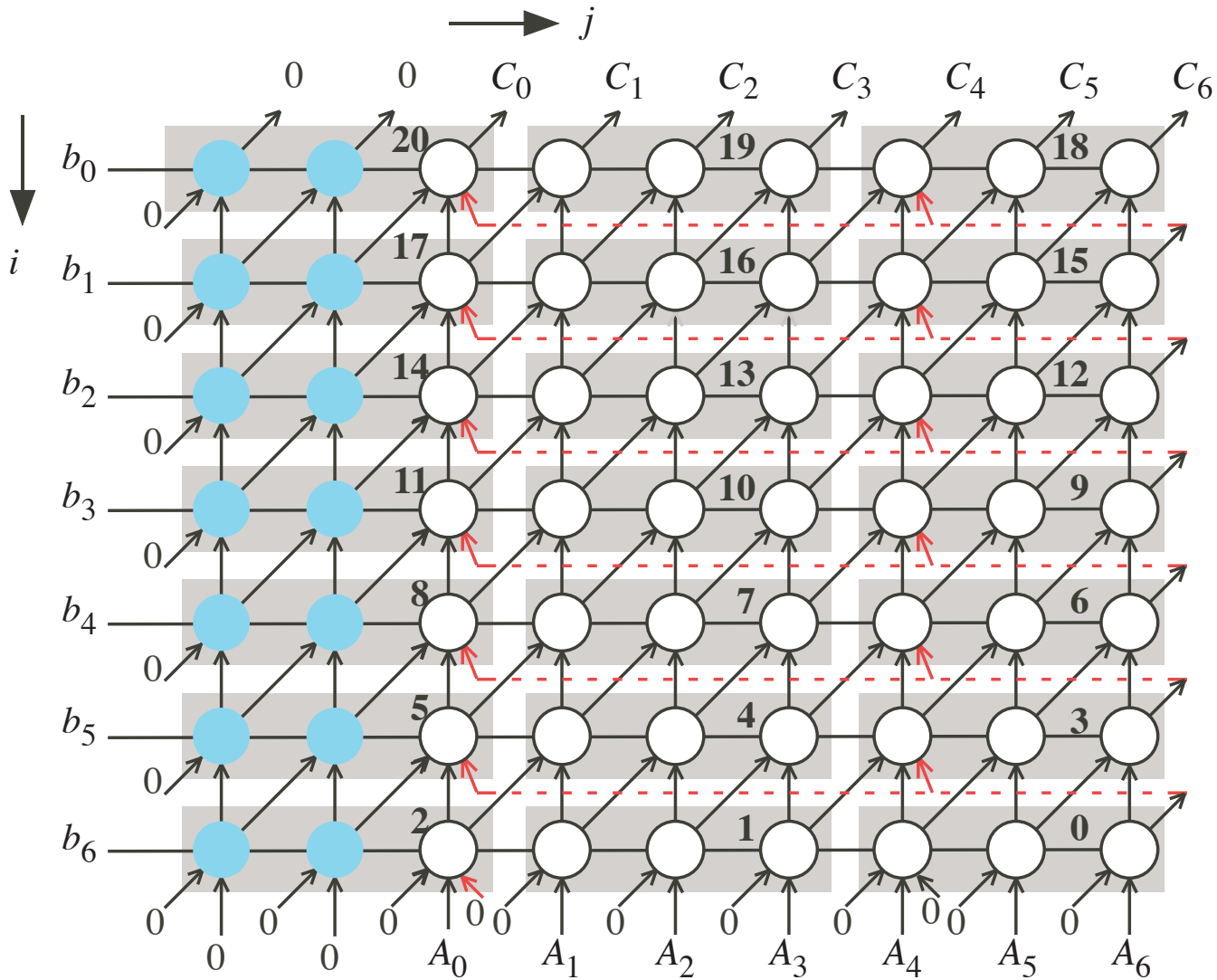
PMR Dependence Graph



PMR Timing: Linear Affine Scheduling



PMR Timing: Non-Linear Scheduling



Projection Function

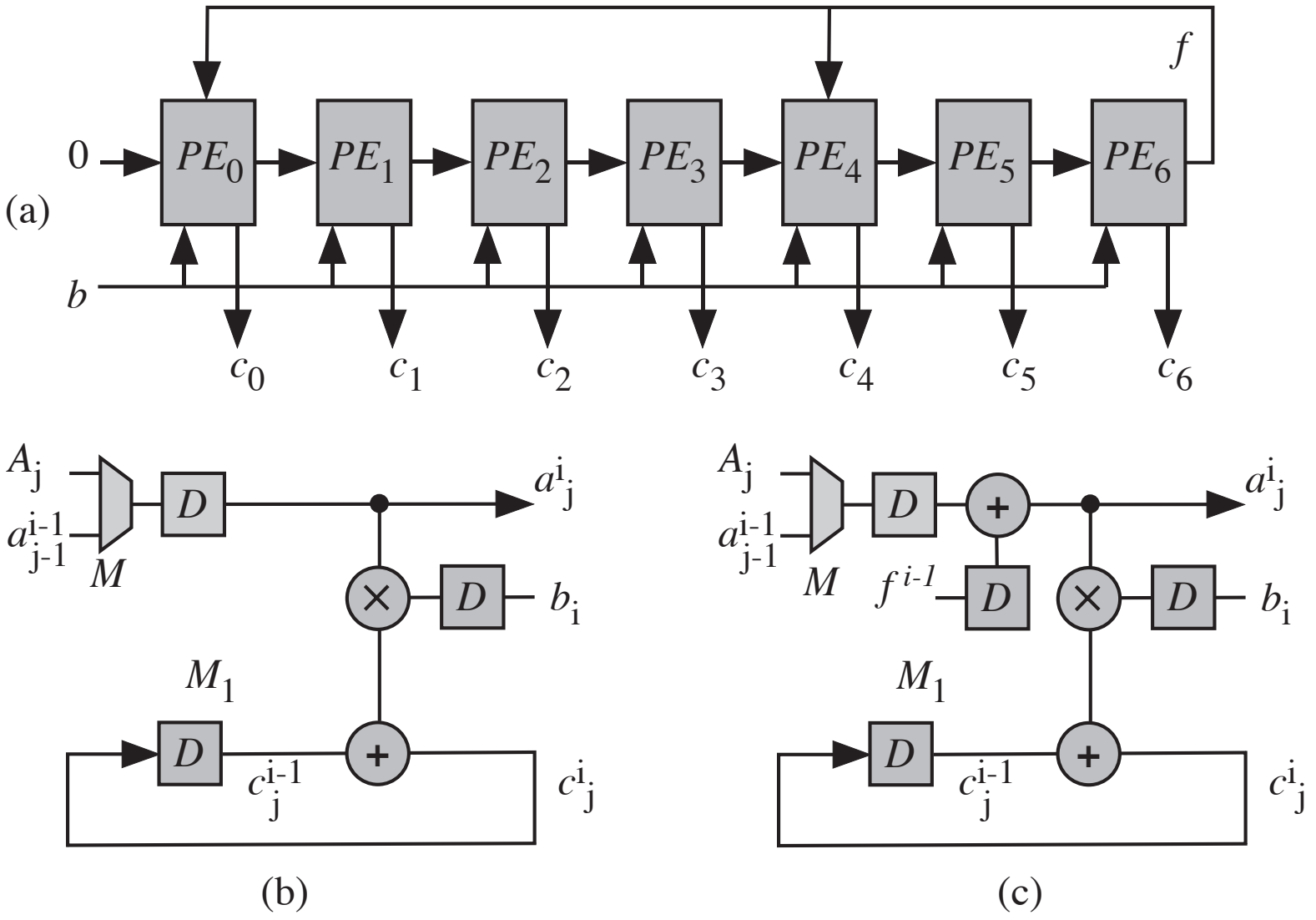
1. Projection operation reduces the dimensionality of the dependance graph
2. Based on the projection direction (**d**)
3. Choice of **d** is related to our choice of **s**
4. The projection operation is based on the projection matrix (**P**) with:

$$\mathbf{Pd} = 0$$

Projection Matrix Examples

$$\begin{aligned} \mathbf{P} &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ \mathbf{P} &= \begin{bmatrix} 0 & \left\lceil \frac{\cdot + \mu}{W} \right\rceil \end{bmatrix} \\ \mu &= W \left\lceil \frac{m}{W} \right\rceil - m \end{aligned}$$

Systolic Array Example



Summary

Summary

1. Brief review of polynomial basis for $GF(2^m)$
2. Design considerations for $GF(2^m)$ arithmetic
3. Considered $GF(2^m)$ with trinomial field polynomial
4. Obtained two techniques for iterative field multiplication: PMR and PPR
5. Showed different algorithm timing strategies
6. Briefly discussed the projection operation

Thank You