

Finite Field Multiplication Using Reordered Normal Basis Multiplier

October 17, 2011

Fayez Gebali
ECE Dept.
University of Victoria
Canada



Turki Al-Somani
Elec. Eng. Dept.
Om Al-Qura University
Saudi Arabia



Outline

- 1 Motivation
- 2 Optimal Normal Basis
- 3 Iterative Algorithm Parallelization
- 4 Implementation
- 5 Summary & Conclusions

Operation Hierarchy in Elliptic Curve Cryptography

Field Arithmetic: $+$, \times , squaring and inversion



Point Addition & Doubling ($P_1 + P_2, 2P$)



Point Scalar Multiplication (kP)

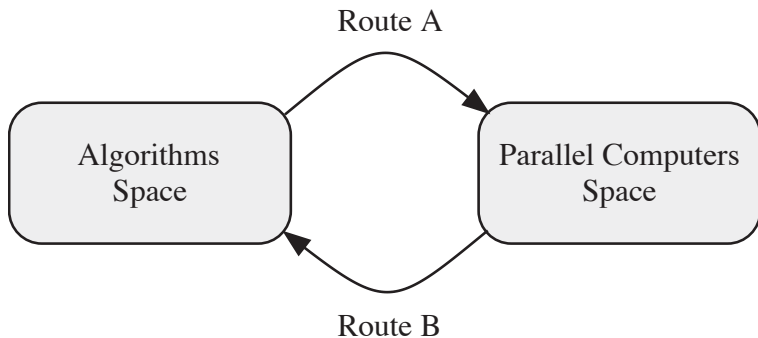


Elliptic Curve Cryptography (ECC)

Goals of this Presentation

- 1 Introduce optimal normal basis (ONB) for $GF(2^m)$
- 2 Discuss ONB multiplication $C = A \times B$
- 3 Present regular iterative algorithm parallelization method
 - 1 Task scheduling
 - 2 Task projection (to threads or processors)
- 4 Design space exploration
- 5 Performance evaluation modeling

Problem Definition



Optimal Normal Basis (ONB)

Optimal Normal Basis for $GF(2^m)$

- 1 Use the normal element $\beta = \gamma + \gamma^{-1}$
- 2 γ is the primitive $(2m + 1)$ first root of unity:

$$\gamma^{2m+1} = 1 \quad \text{and} \quad \gamma^j \neq 1 \quad 1 \leq j < 2m + 1$$

- 3 Optimal normal basis: $\beta_1, \beta_2, \dots, \beta_3, \dots, \beta_m$
- 4 $\beta_j = \gamma^j + \gamma^{-j}$
- 5 Elements $A \in GF(2^m)$ can be represented in the new basis as:

$$A = \sum_{j=1}^m a_j \beta_j$$

Optimal Normal Basis Multiplication $C = A \times B$

$$\begin{aligned} C &= \left[\sum_{i=1}^m a_i (\gamma^i + \gamma^{-i}) \right] \left[\sum_{j=1}^m b_j (\gamma^j + \gamma^{-j}) \right] \\ &= X + Y + Z \end{aligned}$$

Optimal Normal Basis Multiplication $C = A \times B = X + Y + Z$

$$X = \sum_{\substack{1 \leq i, j \leq m \\ i \neq m}} a_i b_j \left[\gamma^{i-j} + \gamma^{-(i-j)} \right]$$

$$Y = \sum_{i=1}^m \sum_{j=1}^{m-i} a_i b_j \left[\gamma^{i+j} + \gamma^{-(i+j)} \right]$$

$$Z = \sum_{i=1}^m \sum_{j=m-i+1}^m a_i b_j \left[\gamma^{i+j} + \gamma^{-(i+j)} \right]$$

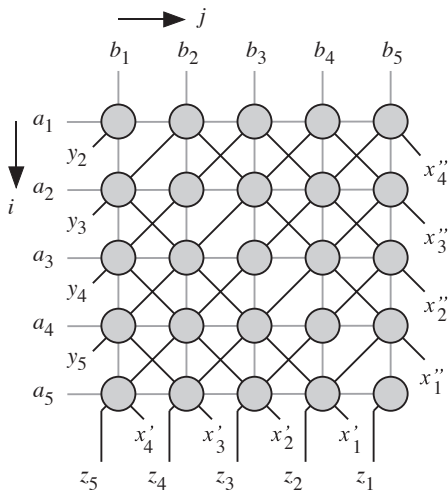
Algorithm Parallelization

Iterative Algorithm Parallelization

- 1 The equations describe a regular iterative algorithm (RIA)
- 2 Inputs are A and B
- 3 Outputs are X , Y and Z
- 4 We study inputs and outputs independently
- 5 We study how each variable depends on the indices i and j

Dependence Graph (Case $m = 5$)

2-D Space

Circles \equiv taskslines \equiv variables

Parallel Hardware/Software Implementations

We need to identify two functions to do¹:

- 1 **Scheduling** data and tasks
- 2 **Projection** of tasks to threads or processors

¹F. Gebali, *Algorithms & Parallel Computing*, John Wiley, 2011.

Task Scheduling

Data & Task Scheduling

- 1 Assume a task at location

$$\mathbf{p} = [p_1 \quad p_2]^t$$

- 2 \mathbf{s} is scheduling vector

- 3 Use an affine scheduling function

$$t(\mathbf{p}) = \mathbf{sp} - s$$

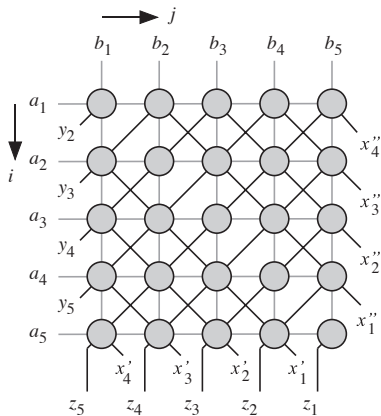
- 4

$$\mathbf{s} = [s_1 \quad s_2]$$

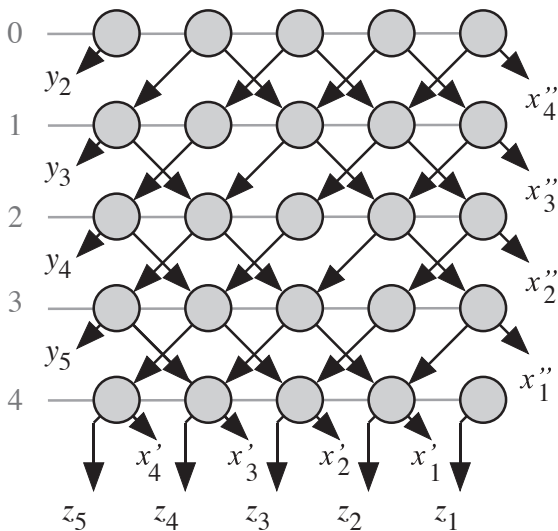
- 5 Design exploration by choice of proper \mathbf{s}

Choice of Scheduling Vectors

- 1 $\mathbf{s}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$ ↘
- 2 $\mathbf{s}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}$ ↓
- 3 $\mathbf{s}_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ →
- 4 $\mathbf{s}_4 = \begin{bmatrix} 1 & -1 \end{bmatrix}$ ↙
- 5 $\mathbf{s}_5 = \begin{bmatrix} -1 & 1 \end{bmatrix}$ ↗

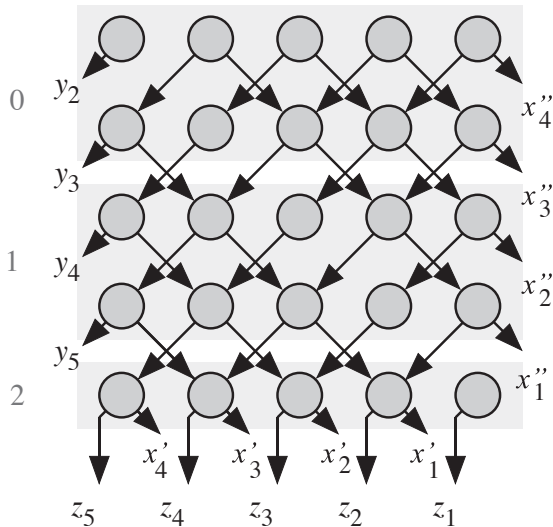


Task & Data Scheduling for $s_2 = [1 \ 0]$ ↓



Nonlinear Scheduling ($\mathbf{s} = [1 \ 0]$ and $\alpha = 2$)

$$t(\mathbf{p}) = \left\lfloor \frac{\mathbf{sp} - \mathbf{s}}{\alpha} \right\rfloor$$



Task Projection

Task Projection: Allocation of Tasks to Threads/Processors

- 1 Assume a task at location

$$\mathbf{p} = [p_1 \quad p_2]^t$$

- 2 Use an affine projection function

$$\bar{\mathbf{p}} = \mathbf{P}\mathbf{p}$$

- 3 \mathbf{P} is **projection matrix**

$$\mathbf{P} = [P_1 \quad P_2]$$

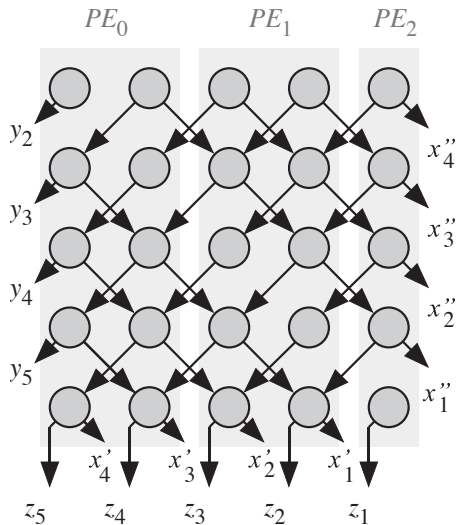
- 4 We use projection direction \mathbf{d} such that $\mathbf{P}\mathbf{d} = 0$
- 5 Design exploration by choice of **proper \mathbf{d}**

Design Space Exploration

Scheduling vectors	Possible Projection Directions		
$\mathbf{s}_1 = [1 \quad 1]$	$\mathbf{d}_{11} = [1 \quad 1]^t$	$\mathbf{d}_{12} = [1 \quad 0]^t$	$\mathbf{d}_{13} = [0 \quad 1]^t$
$\mathbf{s}_2 = [1 \quad 0]$	$\mathbf{d}_{21} = [1 \quad 0]^t$	$\mathbf{d}_{22} = [1 \quad -1]^t$	$\mathbf{d}_{23} = [1 \quad 1]^t$
$\mathbf{s}_3 = [0 \quad 1]$	$\mathbf{d}_{31} = [0 \quad 1]^t$	$\mathbf{d}_{32} = [-1 \quad 1]^t$	$\mathbf{d}_{33} = [1 \quad 1]^t$
$\mathbf{s}_4 = [1 \quad -1]$	$\mathbf{d}_{41} = [1 \quad -1]^t$	$\mathbf{d}_{42} = [0 \quad -1]^t$	$\mathbf{d}_{43} = [1 \quad 0]^t$
$\mathbf{s}_5 = [-1 \quad 1]$	$\mathbf{d}_{51} = [-1 \quad 1]^t$	$\mathbf{d}_{52} = [-1 \quad 0]^t$	$\mathbf{d}_{53} = [0 \quad 1]^t$

Nonlinear Projection ($\mathbf{d} = [1 \ 0]$ and $\beta = 2$)

$$\bar{\mathbf{p}} = \left\lfloor \frac{\mathbf{P}\mathbf{p}}{\beta} \right\rfloor$$



Design Space Exploration: Processor Design Options

Option	Bits Processed in parallel (ρ)	Time Step Duration (τ)	Parallelism Level
1	1	$\alpha\beta$	Full-serial
2	α	β	Serial-parallel
3	β	α	Serial-parallel
4	$\alpha\beta$	1	Full-parallel

Power consumption of PE and Processor Array in mW for $m = 32$

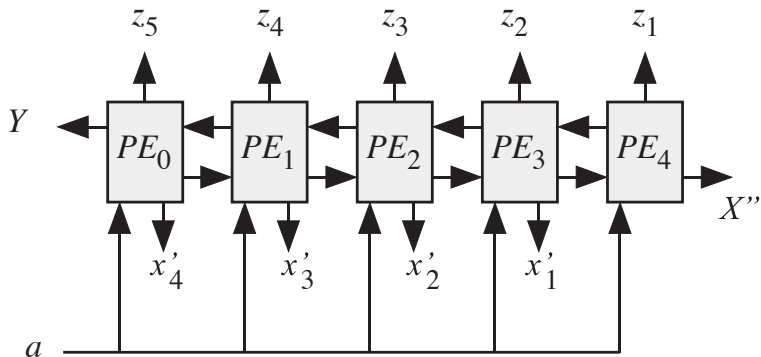
β	P(PE)	P(Array)
1	18.28	584.96
2	19.37	309.92
4	21.45	171.60
8	26.42	105.68
16	36.79	73.58
32	57.1	57.1

$$P(\text{PE}) = P_0 + \beta P_1$$
$$P_0 = 16.675 \text{ mW}, P_1 = 1.26 \text{ mW}$$

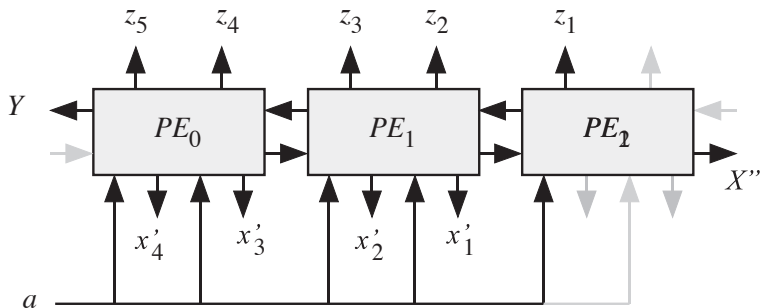
Advantages of Nonlinear Scheduling and Projection

- 1 Control number of iterations
- 2 Control total workload per iteration
- 3 Control processor/thread workload
- 4 Control number of processors/threads

Linear Projection Processor Array: $m = 5$, s_2 and d_{21}



Nonlinear Projection Processor Array: $m = 5, \beta = 3, \mathbf{s}_2$ and \mathbf{d}_{21}



Summary & Conclusions

Summary & Conclusions

- 1 Multiplication algorithm for normal basis multiplier
- 2 Algorithm implementation \implies scheduling & projection
- 3 Technique for implementing iterative algorithms
- 4 Nonlinear techniques allow for control of performance

THANK YOU