Efficient Algorithms and Architectures for Double Point Multiplication on Elliptic Curves

Reza Azarderakhsh: rxaeec@rit.edu

and Koray Karabina: kkarabina@fau.edu

Double Point Multiplication

- Has applications for
  - Signature verification
  - Post-quantum cryptography
  - Efficient single point multiplication

- There are only few laddering algorithms for the computation of double point multiplication.
  - They have never been investigated for hardware implementations.

Binary Elliptic Curves

- $f(\sigma)$ is a monic irreducible polynomial over $\mathbb{F}_2[\sigma]$
- $\mathbb{F}_{2^n} = \mathbb{F}_2[\sigma]/\langle f(\sigma)\rangle$ is a finite field with 2^n elements
- $E_{a,b}(\mathbb{F}_{2^n}) = \{(x,y)\colon\ y^2 + xy = x^3 + ax^2 + b, x, y \in \mathbb{F}_{2^n}\} \cup \infty$
- $E_{a,b}(\mathbb{F}_{2^n}, +)$ is an abelian group
- $\left|E_{a,b}(\mathbb{F}_{2^n})\right| = 2^n + 1 + t,\ |t| \leq 2\sqrt{2^n}$
- $kP = P + P + \cdots + P$, the sum of k points
- $P + \infty = P,\ P + (-P) = \infty.\ P = (x,y) \Rightarrow -P = (x, x+y)$

Scalar Point Multiplication

- Let $\langle P \rangle$ be a prime order subgroup of $E(\mathbb{F}_{2^n})$
- Diffie-Hellman key exchange:

  Single point multiplication (SPM):

  – E, P: public, k: secret random, Compute: kP

- Cramer-Shoup encryption, key generation:

  Double point multiplication (DPM):

  – E, P, Q: public, a, b: secret random, Compute: $aP + bQ$

Efficiency and Security

Ideal scenario:

- Efficient algorithms that perform DPM and SPM so that protocols are faster
- Suitable curves so that DPM can be used to further speed up SPM:
  - Choose a curve and λ so that $\mathrm{P} \rightarrow \lambda P$ is super efficient
  - Write $k = k_1 + k_2\lambda$ for much smaller k_i
  - Compute $kP = (k_1 + k_2\lambda)P = k_1P + k_2(\lambda P)$
- Methods to perform SPM via DPM:
  1. Straus-Shamir trick and interleaving techniques
  2. Differential addition chains

Efficiency and Security (2)

Recipe:

- Choose your curve and parameters so that
  - Discrete Logarithm Problem is intractable: $P, kP \rightarrow k$
- Choose your algorithms so that
  - The leakage of side channel information is minimum

Bad News: The most efficient variants of DPM algorithms (Straus-Shamir, interleaving, differential addition chain) are not side-channel friendly

Good News: There are *secure* variants of DPM algorithms

Double Point Multiplication Laddering Algorithms

- We investigate three main laddering algorithms suitable for hardware implementations.

Algorithm	Cost per-bit	Regular	Differential addition chain	Parallelizable
JT [14]	$0.5A + 1D$	Yes	No	No
DJB [6]	$2A + 1D$	Yes	Yes	Yes
AK [3]	$1.4A + 1.4D$	Yes	Yes	Yes

Details in the next slides.

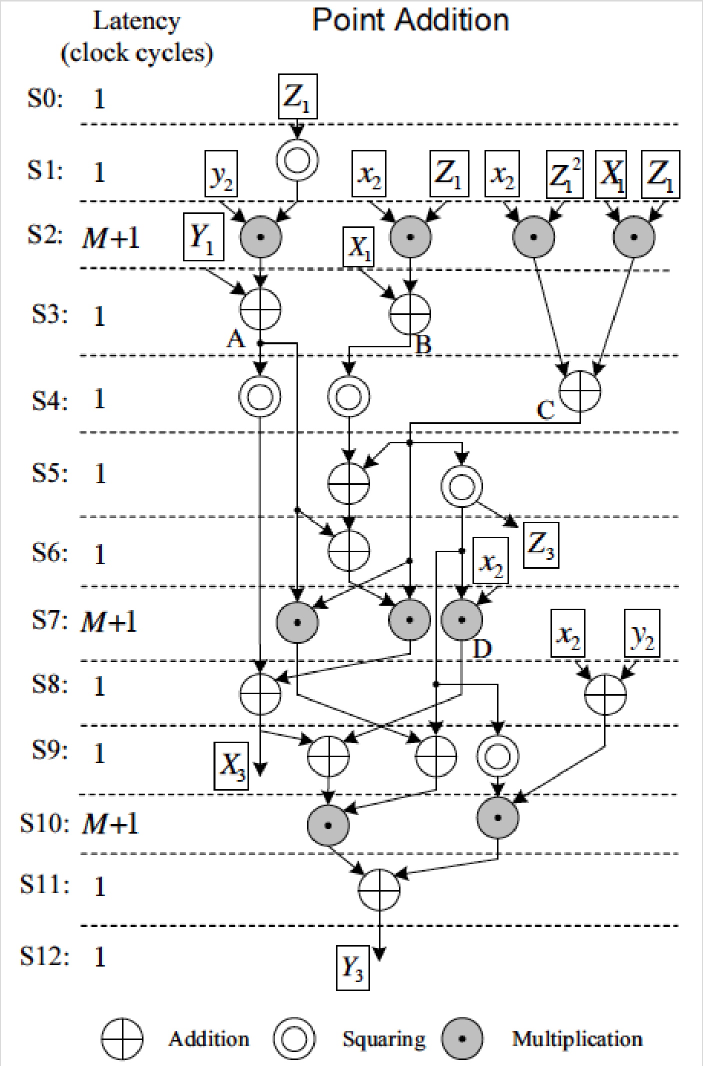
Joye and Tunstall (JT) Algorithm

- JT, 2009 proposes a *regular* recoding of scalars
- One of their method is to use a signed-digit recoding method with the digit set
- If are -bit integers, then requires
  1. About iterations
  2. The current state is updated to , where
  3. Per-bit cost is
- Toy example:

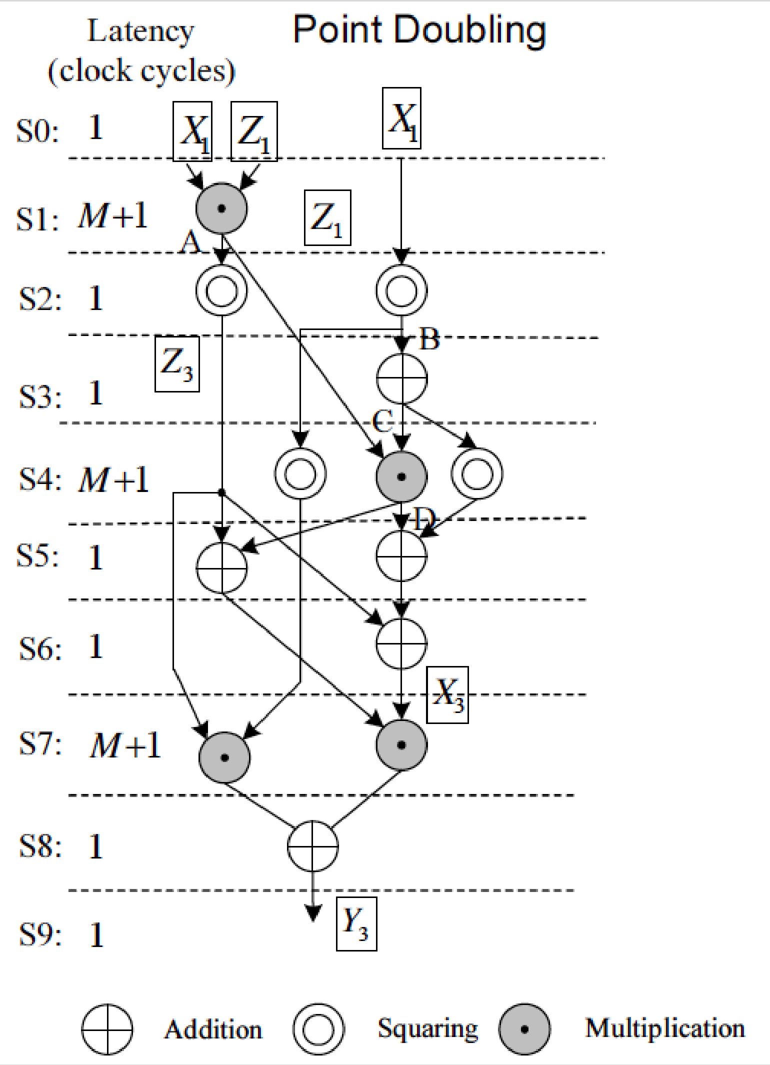
a	1	1	-3	3
b	1	1	3	1
Point	$P+Q$	$5P+5Q$	$17P+23Q$	$71P+93Q$

Parallel Computation of PA and PD for JT

$13M + 4S + 9A$



$5M + 4S + 5A$



We use **Explicit** point addition and doubling formulae

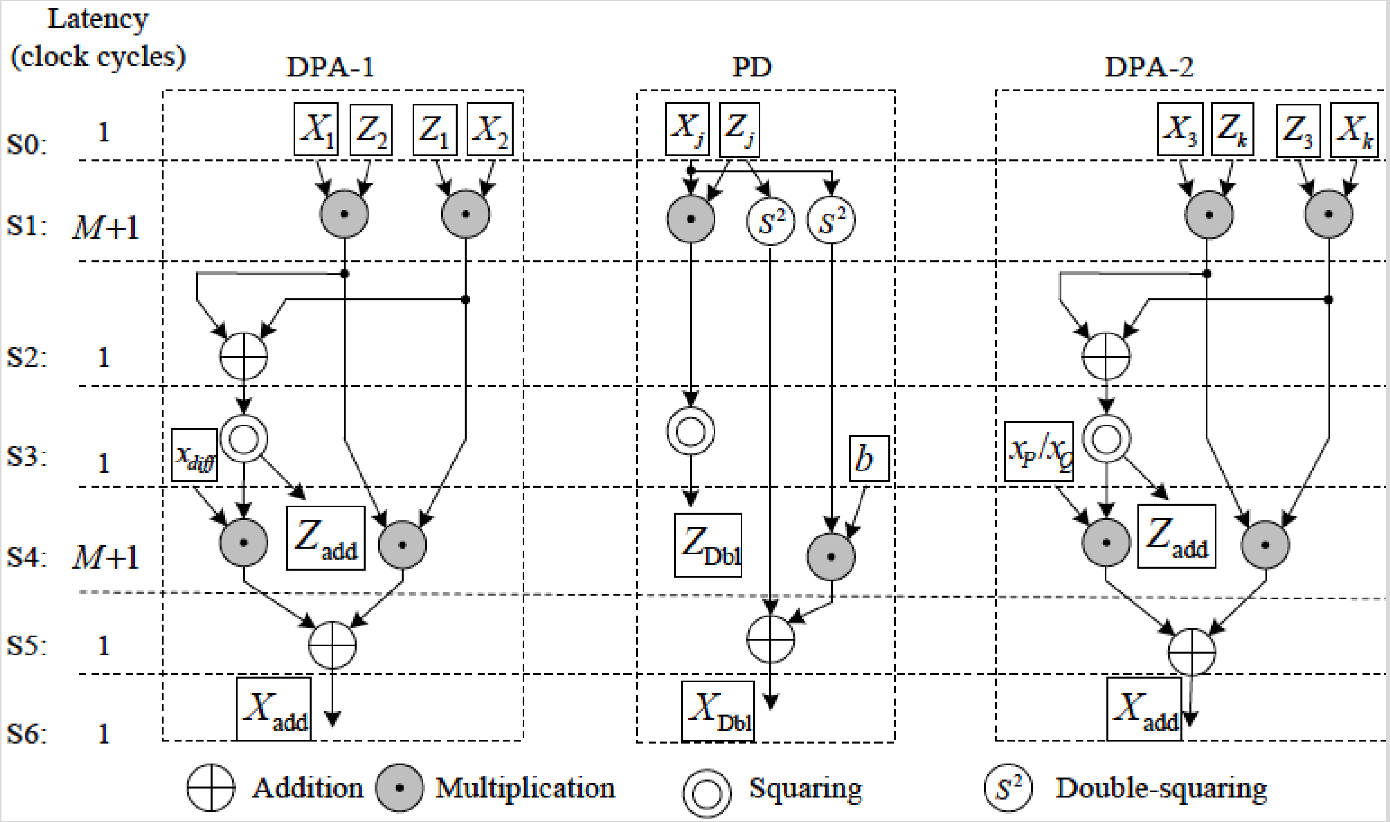
D. J. Bernstein (DJB) Algorithm

- DJB, 2006 proposes a *new binary chain*
- DJB has a *uniform* structure
- Explicit description is provided in our paper
- Per-bit cost is $2A + 1D$

Example: $a = 71, b = 93$

$\boldsymbol{k}$	**CS**	**DS**	$\boldsymbol{v_{k+1}(1)}$	$\boldsymbol{v_{k+1}(2)}$	$\boldsymbol{v_{k+1}(3)}$
0			$P + Q$	$2P + 2Q$	$P + 2Q$
1	$(1,1)$	$(-1,-1),(0,-1)$	$3P + 3Q$	$2P + 2Q$	$2P + 3Q$
2	$(3,1)$	$(1,1),(1,0)$	$5P + 5Q$	$4P + 6Q$	$5P + 6Q$
3	$(2,2)$	$(1,-1),(-1,0)$	$9P + 11Q$	$8P + 12Q$	$9P + 12Q$
4	$(3,1)$	$(1,-1),(0,-1)$	$17P + 23Q$	$18P + 24Q$	$18P + 23Q$
5	$(3,2)$	$(-1,-1),(0,1)$	$35P + 47Q$	$36P + 46Q$	$36P + 47Q$
6	$(3,1)$	$(-1,1),(-1,0)$	$\boldsymbol{71P + 93Q}$	$72P + 94Q$	$71P + 94Q$

Parallel Computation of PA & PD for DJB



Two PA unit are employed in parallel

We use mixed differential point addition and doubling formulae based on Lopez-Dahab with the cost of:

$$6\mathrm{M} + 5\mathrm{S} + 3\mathrm{A}$$

Azarderakhsh and Karabina (AK) Algorithm

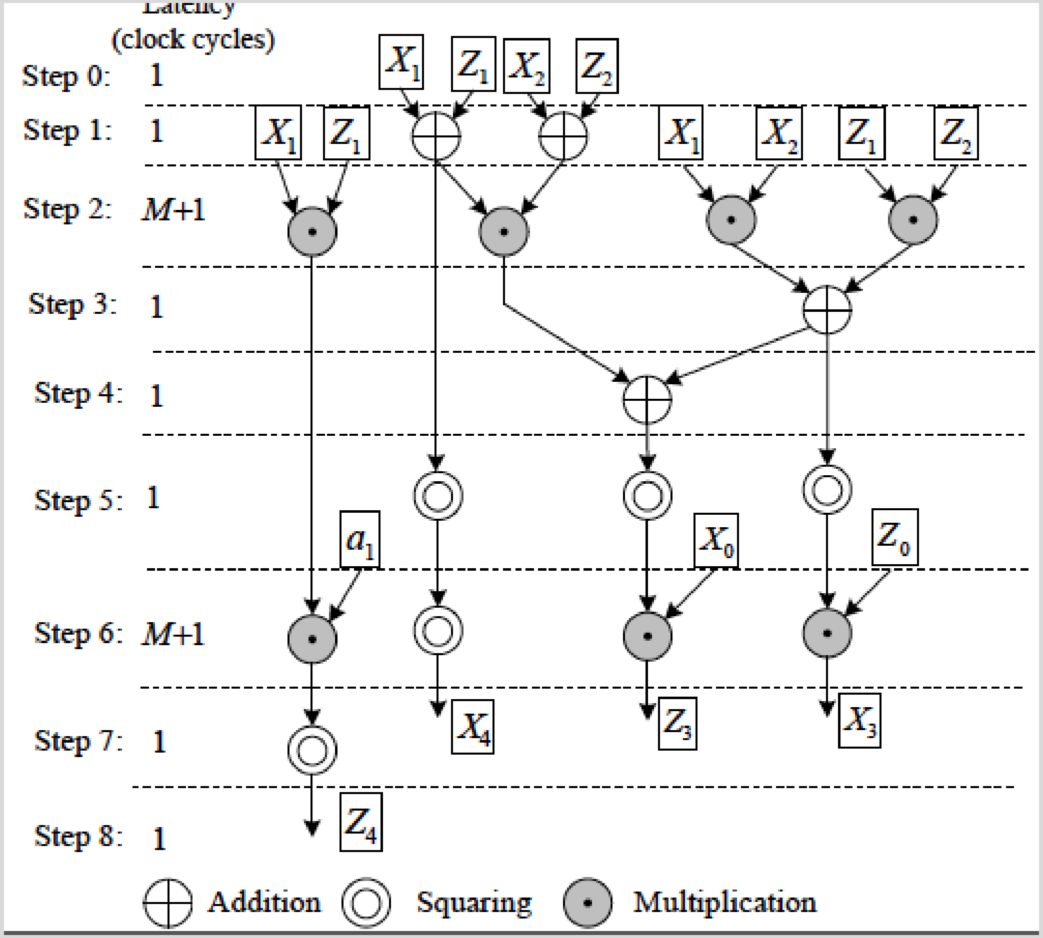
- AK, 2014: based on differential addition chains
- AK has a *uniform* structure
- Explicit description is provided in our paper
- Per-bit cost is $1.4A + 1.4D$

Example: $a = 71, b = 93$

d	e	u	v	Δ	R_u	R_v	R_Δ
71	93	$(1,0)$	$(0,1)$	$(1,-1)$	P	Q	$P - Q$
71	11	$(1,1)$	$(0,2)$	$(1,-1)$	$P + Q$	$2Q$	$P - Q$
30	11	$(2,2)$	$(1,3)$	$(1,-1)$	$2P + 2Q$	$P + 3Q$	$P - Q$
15	11	$(4,4)$	$(1,3)$	$(3,1)$	$4P + 4Q$	$P + 3Q$	$3P + Q$
2	11	$(8,8)$	$(5,7)$	$(3,1)$	$8P + 8Q$	$5P + 7Q$	$3P + Q$
1	11	$(16,16)$	$(5,7)$	$(11,9)$	$16P + 16Q$	$5P + 7Q$	$11P + 9Q$
1	5	$(21,23)$	$(10,14)$	$(11,9)$	$21P + 23Q$	$10P + 14Q$	$11P + 9Q$
1	2	$(31,37)$	$(20,28)$	$(11,9)$	$31P + 37Q$	$20P + 28Q$	$11P + 9Q$
1	1	$(31,37)$	$(40,56)$	$(-9,19)$	$31P + 37Q$	$40P + 56Q$	$-9P + 19Q$

Result $= R_u + R_v = 71P + 93Q$

Parallel Computation of PA & PD for AK



Cost: $6M + 5S + 1D + 4A$

We use **Projective** differential point addition and doubling based on the equation proposed by Stam.

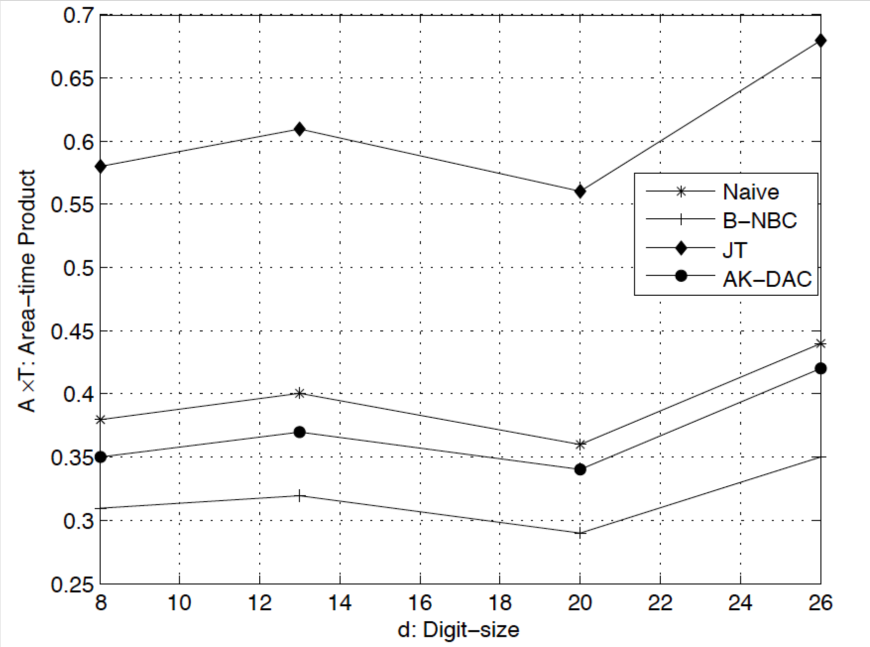
Implementation Results on FPGA

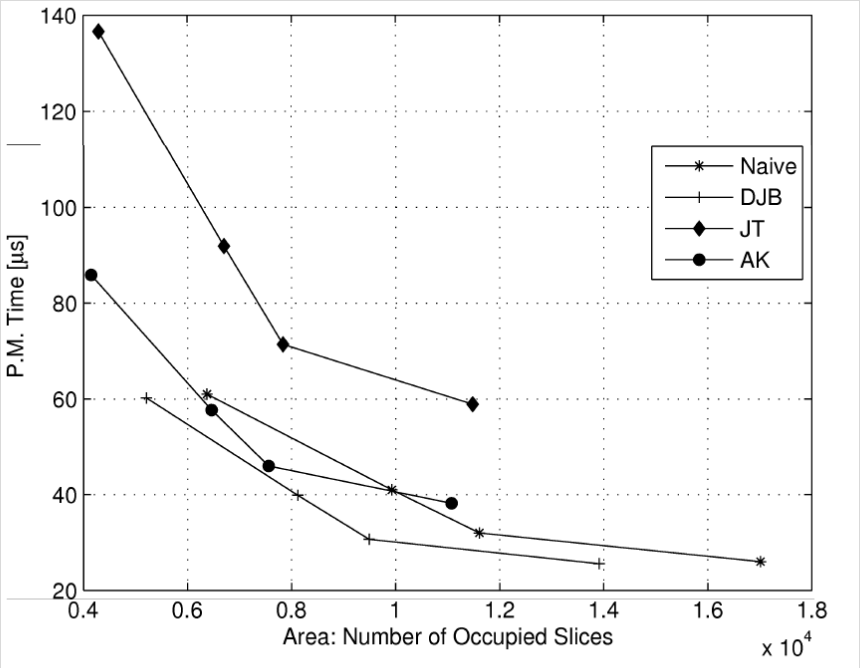
Naive Method 6 Mults. (Section 2.1)						
d	q	Latency [# Clock cycles]	CPD [ns]	Time [μs]	Area [# Slices]	AT Area × Time
7	34	17,937	3.40	60.9	6,218	0.38
13	18	10,305	3.93	40.4	9,693	0.39
18	13	7,920	3.97	31.4	11,335	0.35
26	9	6012	4.31	25.9	16,612	0.43
JT 4 Mults. (Section 2.2) [12]						
7	34	40,057	3.42	136.9	4,196	0.57
13	18	23,145	3.98	92.1	6,541	0.60
18	13	17,860	4.01	71.6	7,649	0.54
26	9	13,632	4.33	59.1	11,210	0.66

DJB 5 Mults. (Section 2.3) [5]				
Latency [# Clock cycles]	CPD [ns]	Time [μs]	Area [# Slices]	AT Area × Time
17,828	3.38	60.2	5,207	0.31
10,244	3.90	39.9	8,117	0.32
7,874	3.91	30.7	9,492	0.29
5,978	4.29	25.7	13,911	0.35
AK 4 Mults. (Section 2.4) [3]				
25,437	3.38	85.9	4,146	0.35
14,884	3.88	57.7	6,462	0.37
11,586	3.97	45.9	7,557	0.34
8,947	4.28	38.2	11,075	0.42

Area-Time Comparison

Comparison of implementation results of different double point multiplication algorithms on FPGA





Conclusion

- Double point multiplication has several applications.
- Efficient algorithms and architectures for double point multiplication are proposed.
- AK requires **smaller area**
- DJB provides the **fastest computations**

- Future work: Implementations on resource constrained environments