Hardware implementations of ECC

The University of Electro-Communications

Kazuo SAKIYAMA

Introduction

- Public-key Cryptography (PKC)
  - The most famous PKC is RSA and ECC
  - Used for key agreement (Diffie-Hellman), digital signatures, public-key encryption
- Implementation of public-key cryptosystem
  - Slower encryption/decryption (~Mbps) than symmetric key cryptography (AES: ~Gbps)
  - RSA: modular exponentiation, M^e
  - ECC: point multiplication, kP on elliptic curve

Point operations on elliptic curve

- Point Addition

  $R = P + Q$

- Point Doubling

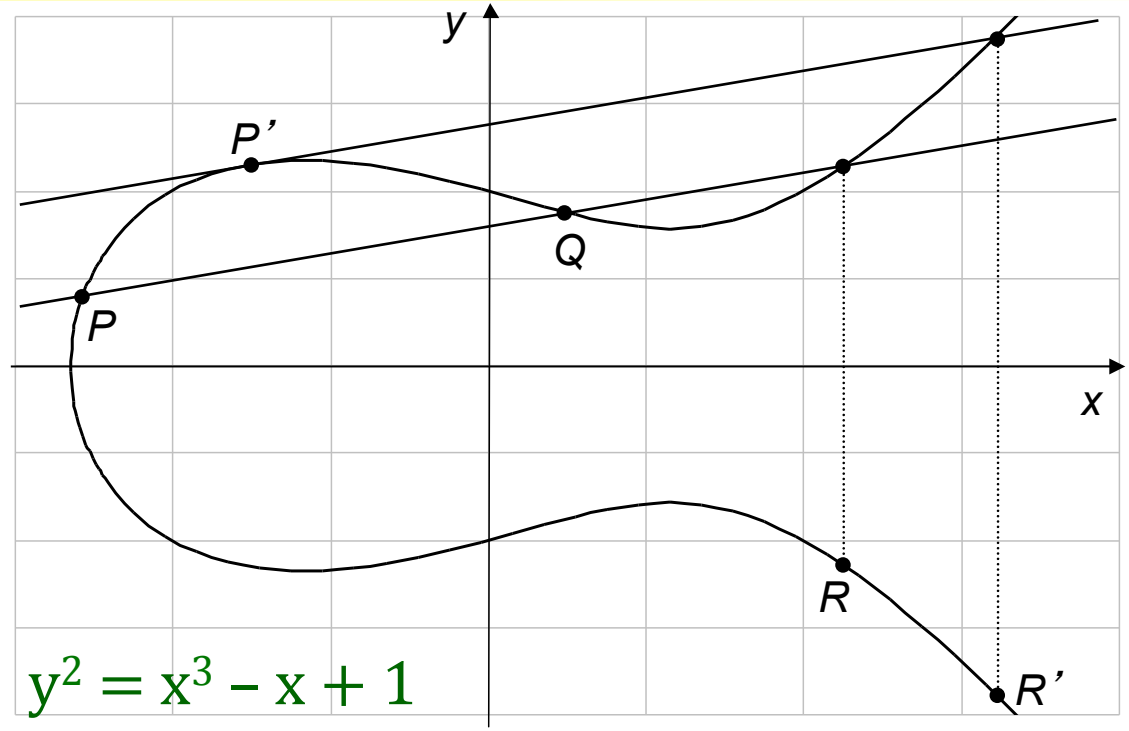
  $R' = 2P'$

  Tangent line

- Scalar Mult.

  kP

  Binary method, etc.



Example of point operations

- Weighted projective coordinate
- All operations are over GF(p)

Require: $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$.
Ensure: $Q = Q + P$.

1: $t_1 = Z_1 Z_1$;
2: $t_2 = X_2 t_1$;
3: $t_3 = Z_2 Z_2$;
4: $t_4 = X_1 t_3 + t_2$;
5: $t_2 = X_1 t_3 - t_2$;
6: $t_5 = t_1 Z_1$;
7: $t_6 = Y_2 t_5$;
8: $t_1 = t_3 Z_2$;
9: $t_3 = t_1 Y_1 + Y_2$;
10: $Y_2 = t_1 Y_1 - Y_2$;
11: $t_5 = t_2 t_2$;
12: $t_1 = t_4 t_5$;
13: $X_2 = Y_2 Y_2 - t_1$;
14: $t_4 = -2X_2 + t_1$;
15: $t_1 = t_5 t_2$;
16: $t_1 = t_3 t_1$;
17: $t_3 = t_4 Y_2 - t_1$;
18: $Y_2 = t_3/2$;
19: $Z_2 = t_2 Z_2$;
20: $Z_1 = Z_1 Z_2$;
21: Return Q;

Require: $P = (X_1, Y_1, 1)$, $Q = (X_2, Y_2, Z_2)$.
Ensure: $Q = Q + P$.

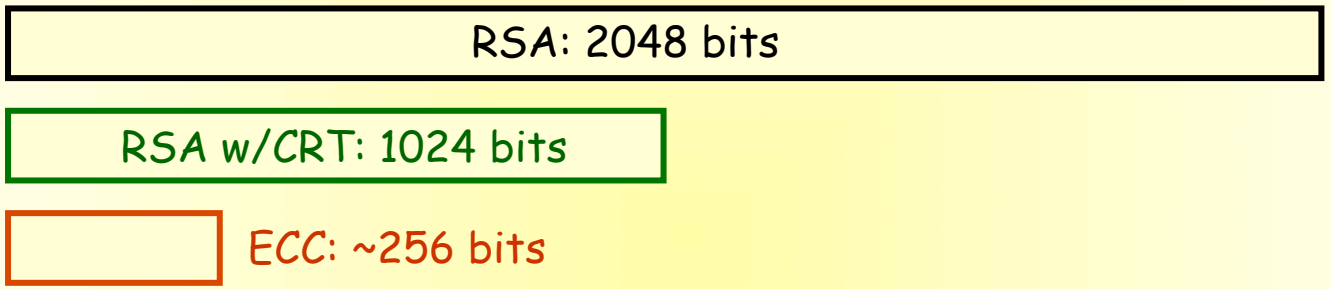
1: $t_3 = Z_2 Z_2$;
2: $t_4 = X_1 t_3 + X_2$;
3: $t_2 = X_1 t_3 - X_2$;
4: $t_1 = t_3 Z_2$;
5: $t_3 = t_1 Y_1 + Y_2$;
6: $Y_2 = t_1 Y_1 - Y_2$;
7: $t_5 = t_2 t_2$;
8: $t_1 = t_4 t_5$;
9: $X_2 = Y_2 Y_2 - t_1$;
10: $t_4 = -2X_2 + t_1$;
11: $t_1 = t_5 t_2$;
12: $t_1 = t_3 t_1$;
13: $t_3 = t_4 Y_2 - t_1$;
14: $Y_2 = t_3/2$;
15: $Z_2 = t_2 Z_2$;
16: Return Q;

Require: $Q = (X_2, Y_2, Z_2)$.
Ensure: $Q = 2Q$.

1: $t_1 = X_2 X_2$;
2: $t_1 = 3t_1$;
3: $t_2 = Z_2 Z_2$;
4: $t_2 = t_2 t_2$;
5: $t_2 = at_2 + t_1$;
6: $t_1 = 2Y_2$;
7: $Z_2 = Z_2 t_1$;
8: $t_3 = t_1 t_1$;
9: $t_4 = X_2 t_3$;
10: $X_2 = 2t_4$;
11: $X_2 = t_2 t_2 - X_2$;
12: $t_1 = t_1 t_3$;
13: $t_1 = t_1 Y_2$;
14: $t_3 = t_4 - X_2$;
15: $Y_2 = t_2 t_3 - t_1$;
16: Return Q;

Data Size in Modular Operations

- Typical key lengths of public-key cryptography



- RSA: $M^e \bmod n$
- ECC: kP over GF(p) and $GF(2^m)$

of Modular Multiplications

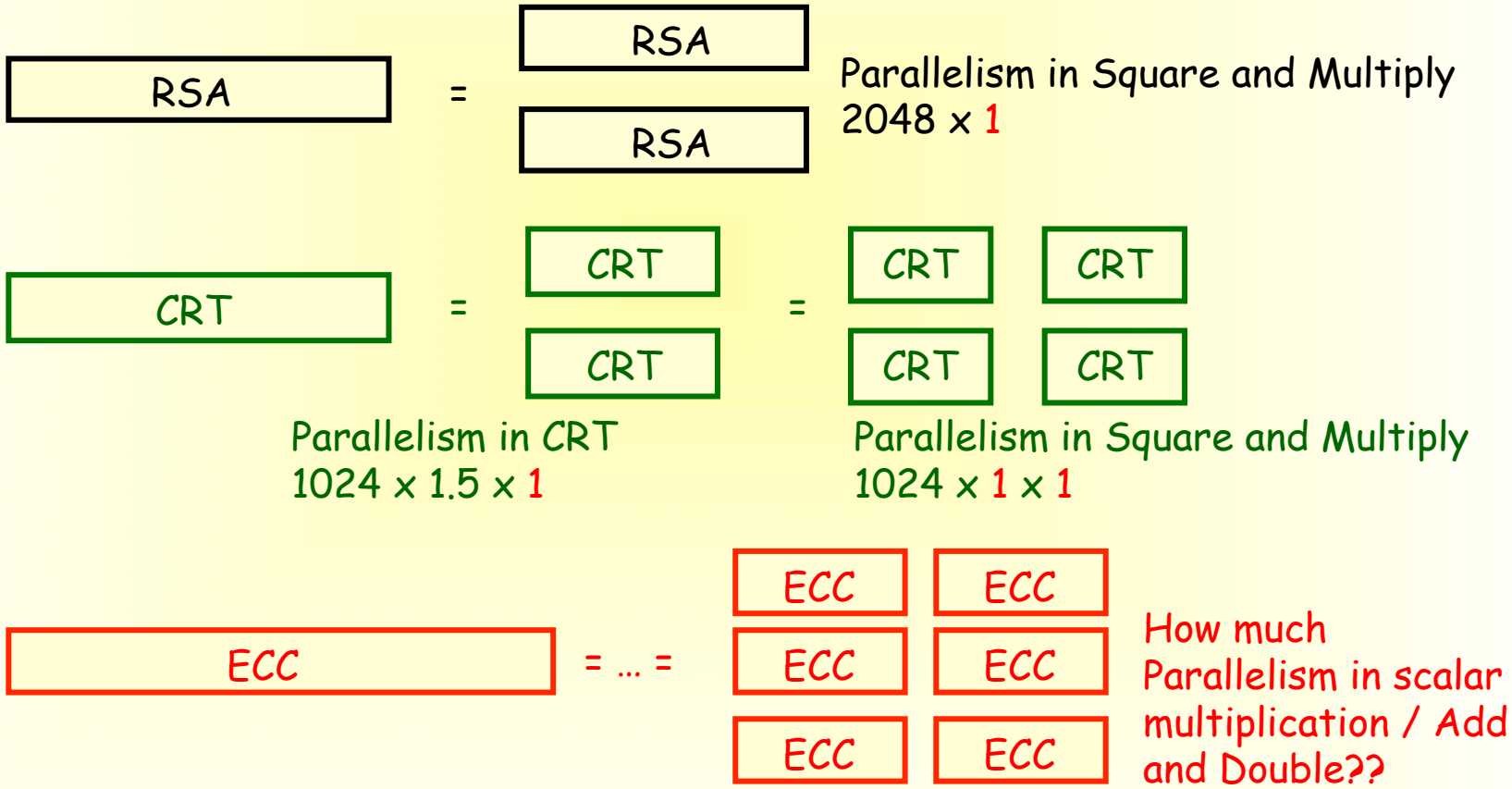
- ECC needs more modular multiplications

RSA	(2048-bit Modular Mult.) x 2048 x 1.5
CRT	(1024-bit Modular Mult.) x 1024 x 1.5 x 2
ECC	(256-bit Modular Mult.) x 160 x 20 x 1.5

- RSA: $M^e \bmod n$
- ECC: kP over GF(p) and $GF(2^m)$

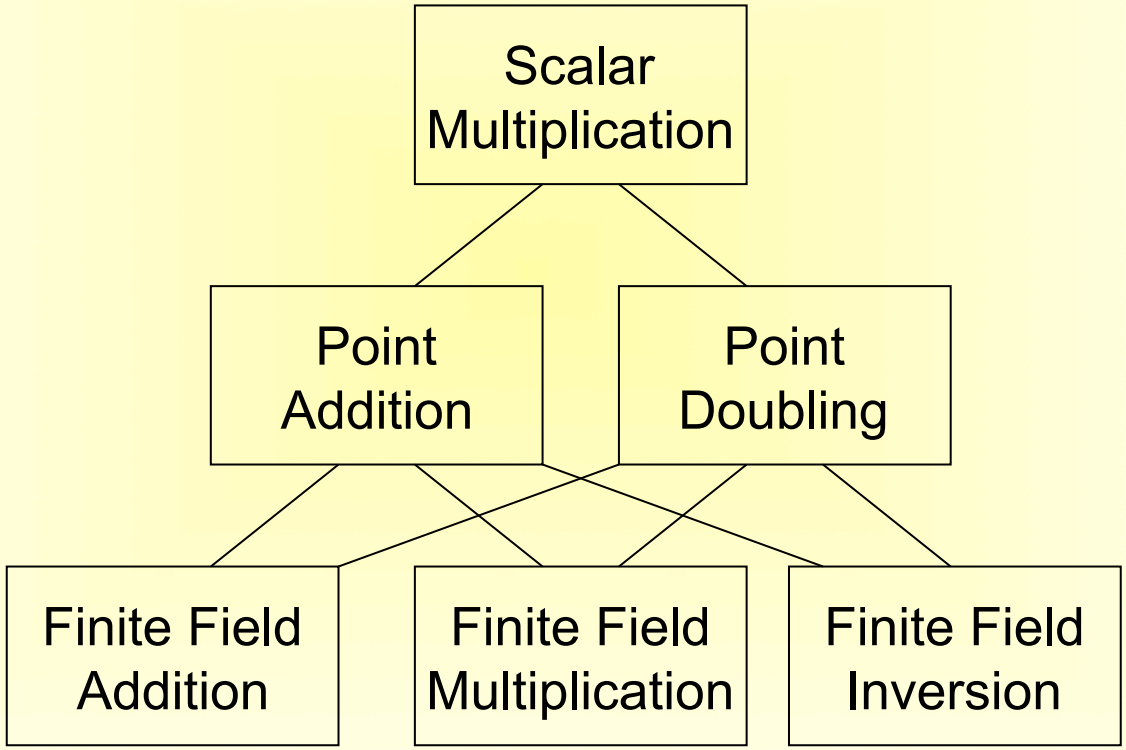
Parallelism in high-speed implementation

- Degree of parallelism



Basic hierarchy for ECC

- Optimization possible in multi-layer



Modular Multiplication

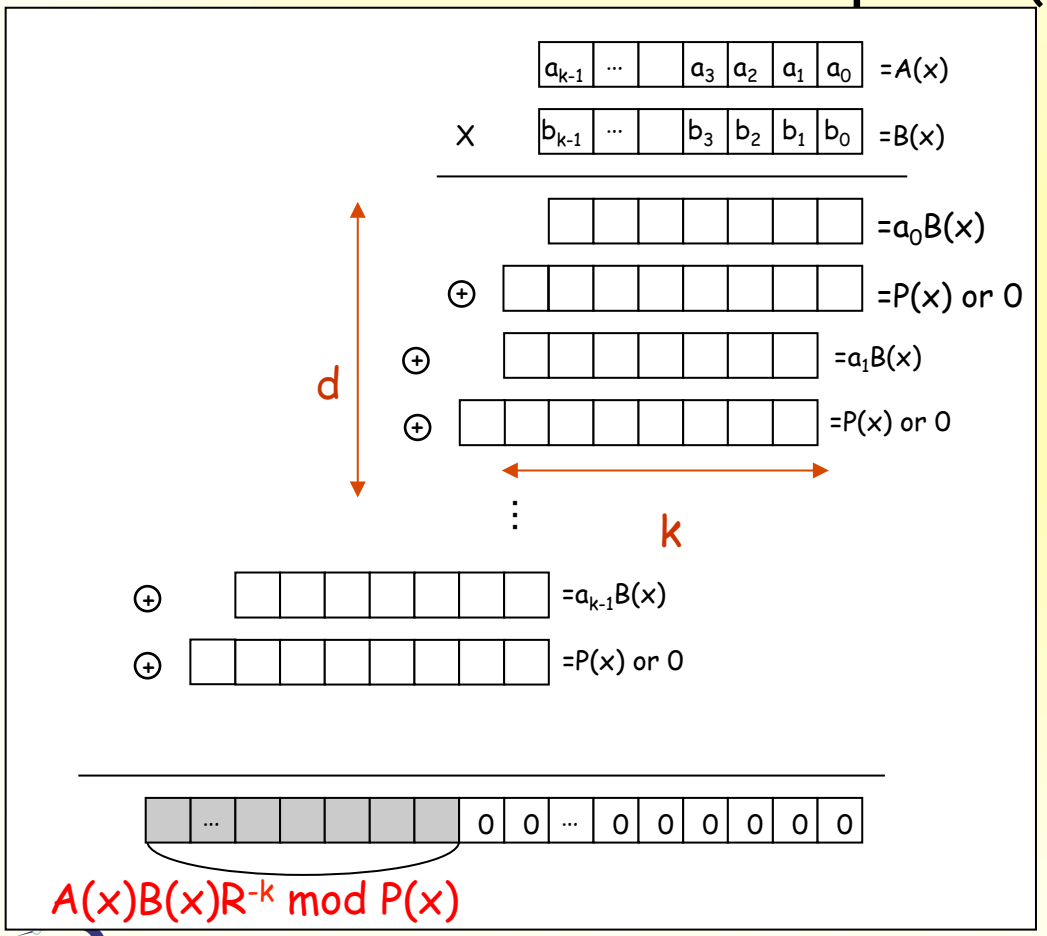
- = Multiplication + Reduction
  - Barrett Reduction (1986): Focus on MSBs
  - Montgomery Reduction (1985): Focus on LSBs

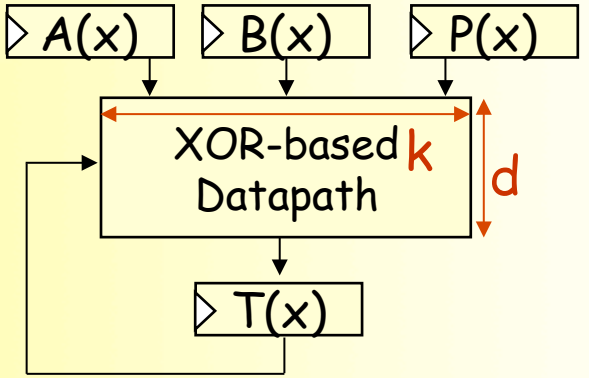
- Based on n-bit multiplier (n = 8, 16, 32, 64)
  - E.g., CIOS [IEEE Micro KKoc and Kaliski Jr 1996].

- Based on bit-serial multiplier
  - Multiplicand x n-bit [CHES Großschädl 2001]

Modular Reduction from LSB (binary field)

- LSB-first bit-serial multiplier (binary fields)



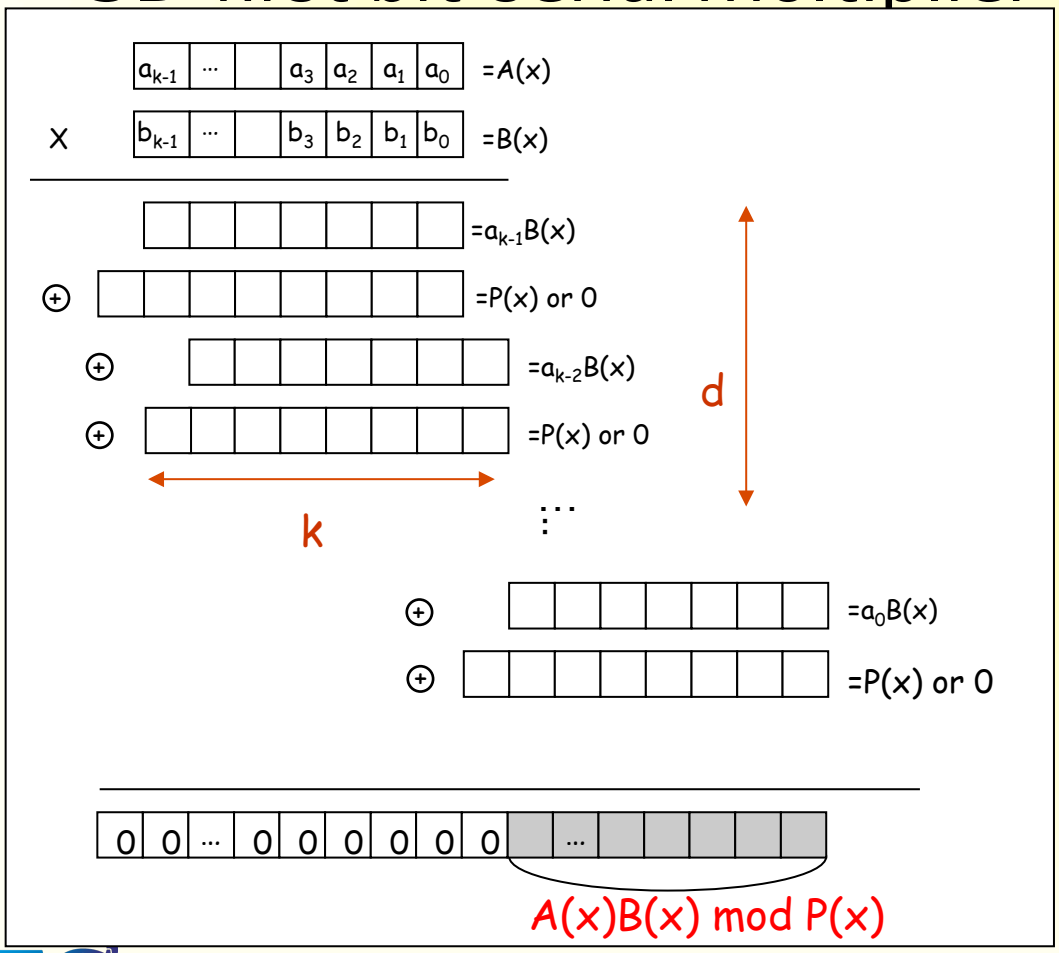


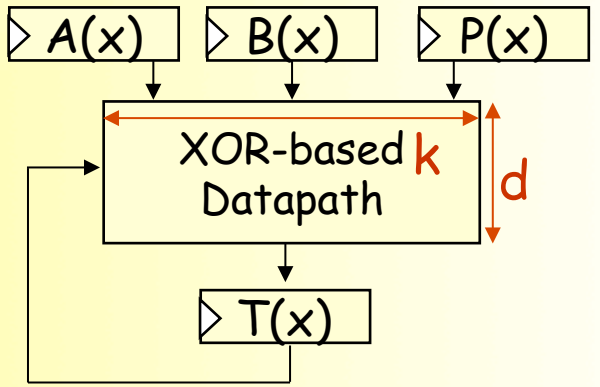
Montgomery Multiplier:
$A(x)B(x)R^{-1}$ mod P(x) $(R = x^k)$

- Delay is determined by **d**
- Loop : k/d

Modular Reduction from MSB (binary field)

- MSB-first bit-serial multiplier (binary field)



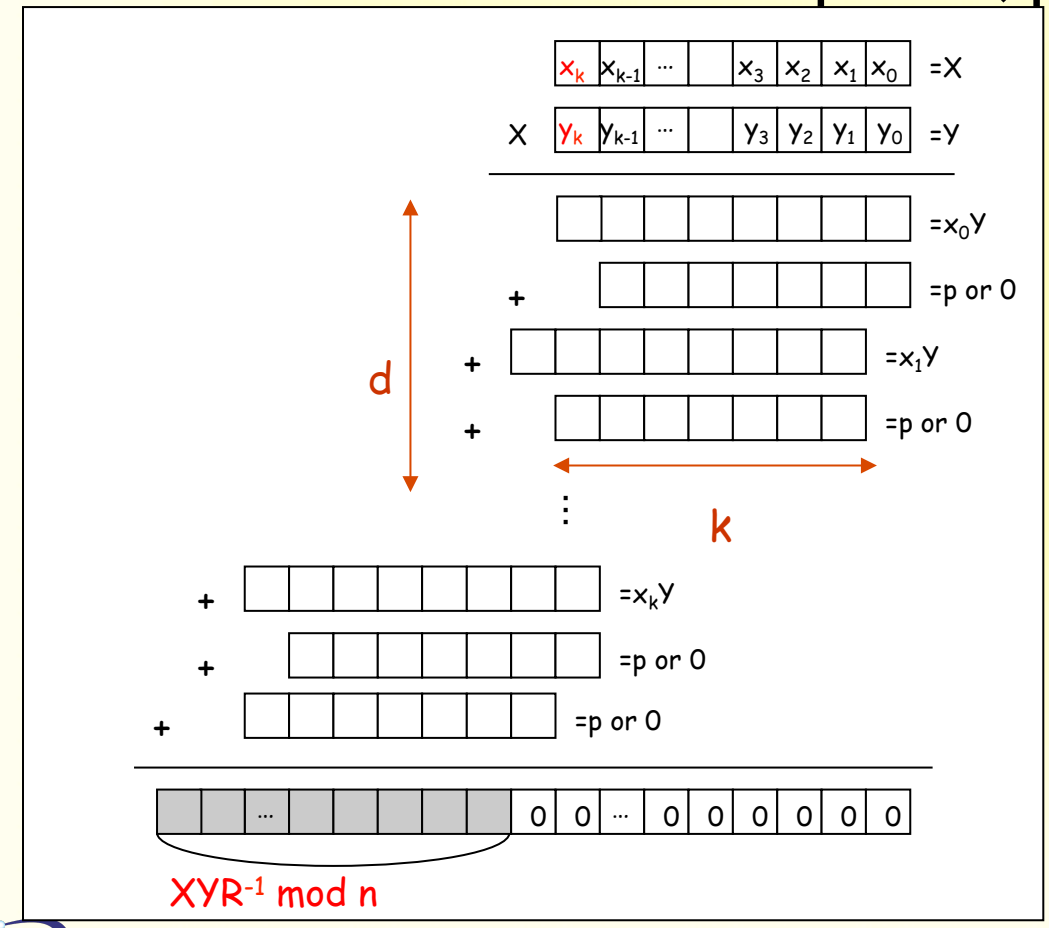


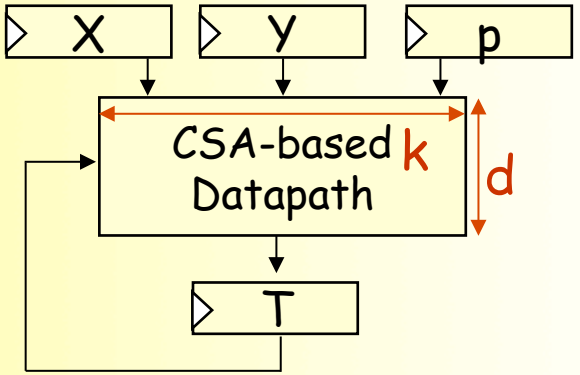
A(x)B(x) mod P(x)

- Delay is determined by **d**
- Loop : k/d

Modular Reduction from LSB (prime field)

- LSB-first bit-serial multiplier (prime field)





Montgomery Multiplier:
XYR^{-1} mod p (R = 2^{k+2})

- Delay is determined by **d**
- Loop : (k+2)/d

Final reduction for prime field p

- $XYR^{-1} \bmod p$
- Intermediate value, $T = (T + x_i y + mp)/2 < 2p$

Input: k-bit integers X, Y with $0 \leq X, Y < p$, $R = 2^k$.
Output: $XYR^{-1} \bmod p$.

1: $T = (t_k, \ldots t_0)_2 \leftarrow 0$
2: **for** i from 0 to $k-1$ **do**
3: $\quad m \leftarrow t_0 \oplus x_i \cdot y_0$
4: $\quad T \leftarrow (T + x_i Y + mp)/2$
5: **end for**
6: **if** $T \geq p$ **then**
7: $\quad T \leftarrow T - p$
8: **end if**
9: return T

Final reduction: $T = T - p < p$

Fixed number of additions of p

- $XYR^{-1} \bmod p$
- Intermediate value, $T = (T + x_i y + mp)/2 < 3p$

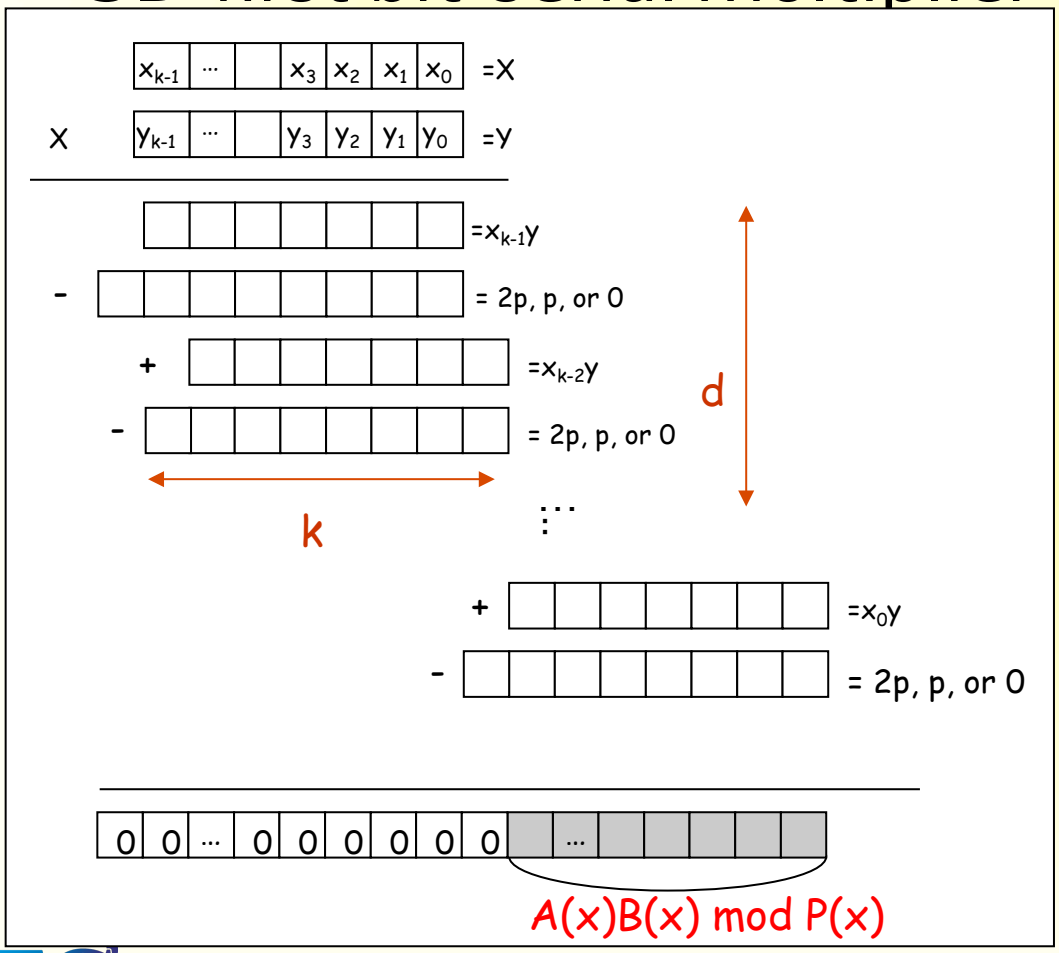
Input: $(k+1)$-bit integers, X, Y with $0 \leq X, Y < 2p$, $R = 2^{k+2}$.
Output: $XYR^{-1} \bmod p$.

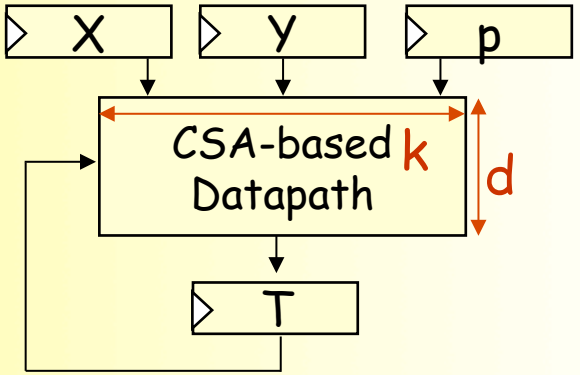
1: $T = (t_{k+1}, \ldots t_0)_2 \leftarrow 0$
2: **for** i from 0 to k **do**
3: $\quad m \leftarrow t_0 \oplus x_i \cdot y_0$
4: $\quad T \leftarrow (T + x_i Y + mp)/2$
5: **end for**
6: $T \leftarrow (T + mp)/2$
7: **return** T

$T = (T + mp)/2 < 2p$

Modular Reduction from MSB (prime field)

- MSB-first bit-serial multiplier (prime field)





X·Y mod p

- Delay is determined by **d**
- Loop : (k+2)/d

Division in MSB-first modular multiplication

- XY mod p
- Intermediate value, $T = (2T + x_i y) < 3p$

Input: k-bit integers X, Y with $0 \leq X, Y < p$.
Output: $XY \bmod p$.
1: $T = (t_{k+1}, \ldots t_0)_2 \leftarrow 0$
2: **for** i from $k-1$ to 0 **do**
3: $\quad T \leftarrow (2T + x_i Y)$
4: $\quad q \leftarrow \lfloor T/p \rfloor$
5: $\quad T \leftarrow (T - qp)$
6: **end for**
7: return T

- Division can be escaped with precomputation

[Barrett Reduction; IEEE ToC, Knezevic, Vercauteren, Verbauwhede, 2010]

Barrett Reduction

- $T/p = T/2^{k-1} * 2^{k-1}/p$
- Step4 : $q = T/2^{k-1}$ ---shift operation (wiring)
- Step5 : $T = (T - q * 2^{k-1}/p)$ ---precomputation

Input: k-bit integers X, Y with $0 \leq X, Y < p$.
Output: $XY \bmod p$.
1: $T = (t_{k+1}, \ldots t_0)_2 \leftarrow 0$
2: **for** i from $k-1$ to 0 **do**
3: $\quad T \leftarrow (2T + x_i Y)$
4: $\quad q \leftarrow \lfloor T/p \rfloor$
5: $\quad T \leftarrow (T - qp)$
6: **end for**
7: return T

- But q = 0, 1, or 2 …

Parallelism in Modular Multiplication

- Both MSB/LSB Modular Multiplication

  [MELECON, Batina et. al., 2004; CHES, Kaihara and Takagi, 2005]

- Combine MSB-first and LSB-first MM
- $A(x)B(x)R^{-1} \bmod P(x)$

$$= (A_1(x)R + A_0)B(x)R^{-1} \bmod P(x)$$

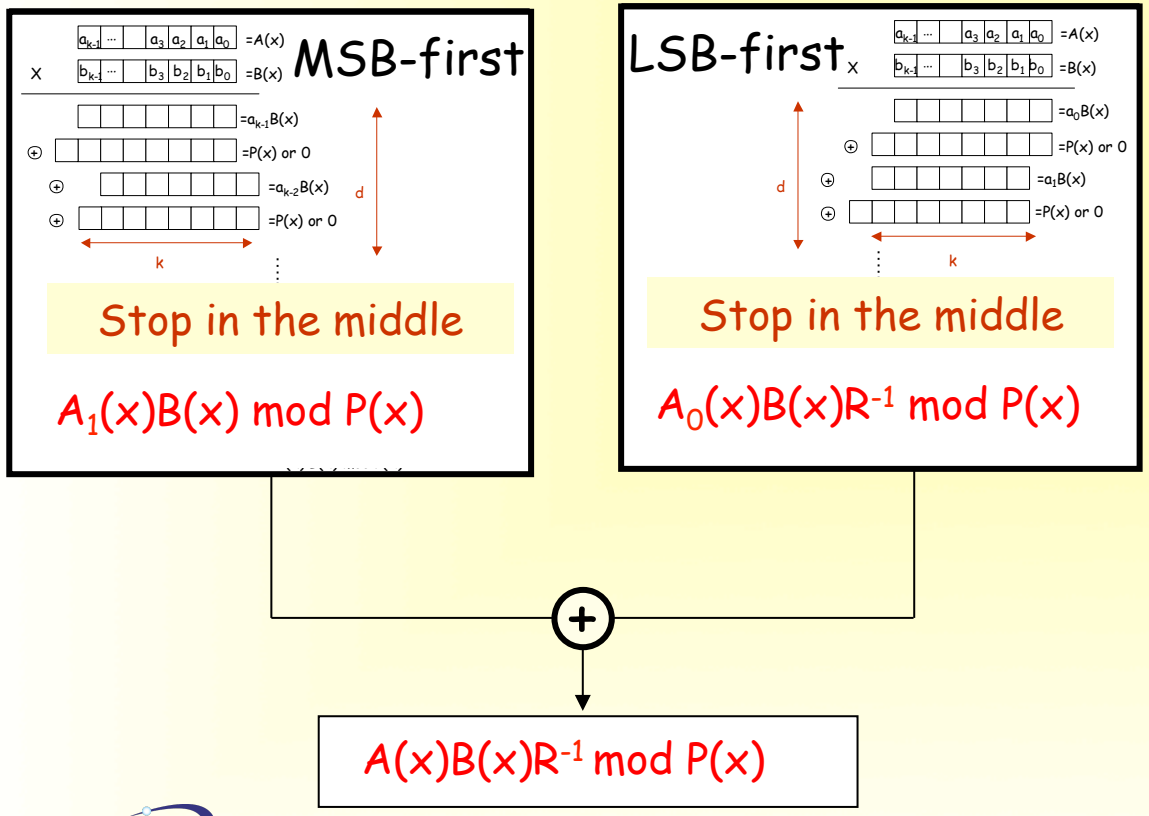
$$= A_1(x)B(x) \bmod P(x) \quad \text{MSB-first}$$

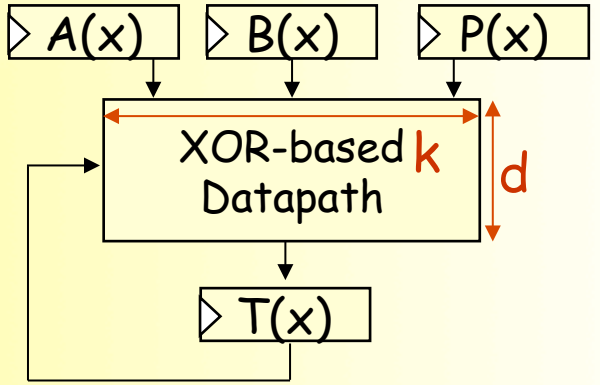
$$+ A_0(x)B(x)R^{-1} \bmod P(x) \quad \text{LSB-first}$$

- $R = x^{k/2}$

Datapath of Bipartite Modular Multiplication

- Bipartite bit-serial multiplier (binary field)





$A(x)B(x)R^{-1} \bmod P(x)$

- Parallelized (x2)
- Loop: $k/2d$

Great Merits in Bipartite Modular Multiplication

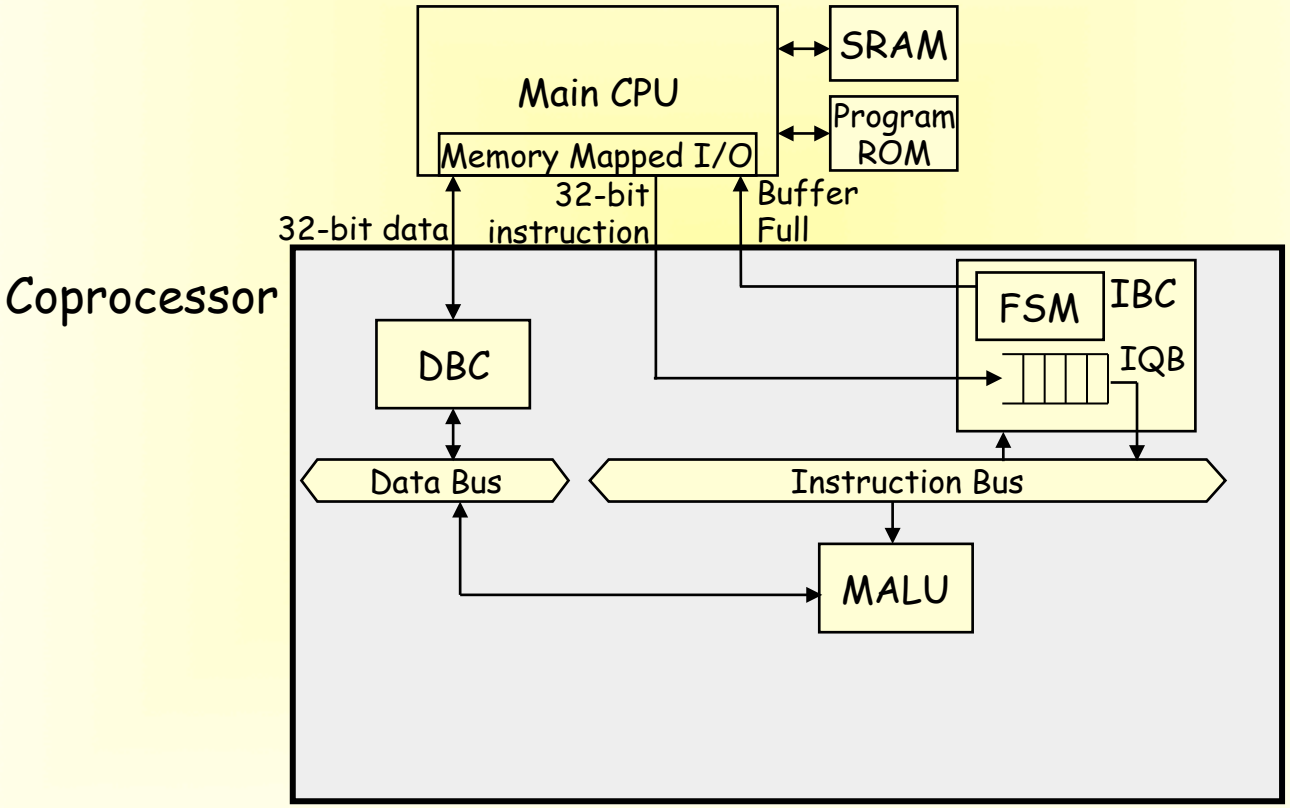
- Parallelism in Modular Multiplication
- Applicable to prime field (original proposal)
  - Need to take care (final) reductions
- Almost double performance, less than double area (F/Fs are shared)

Architecture for ECC coprocessor

- HW/SW co-design
  - CPU + coprocessor for ECC
- Speed performance depends on architecture
  - Trade-off between cost and performance
  - Local dedicated memory -> high performance
- Instruction-set extension, ASIC, etc.

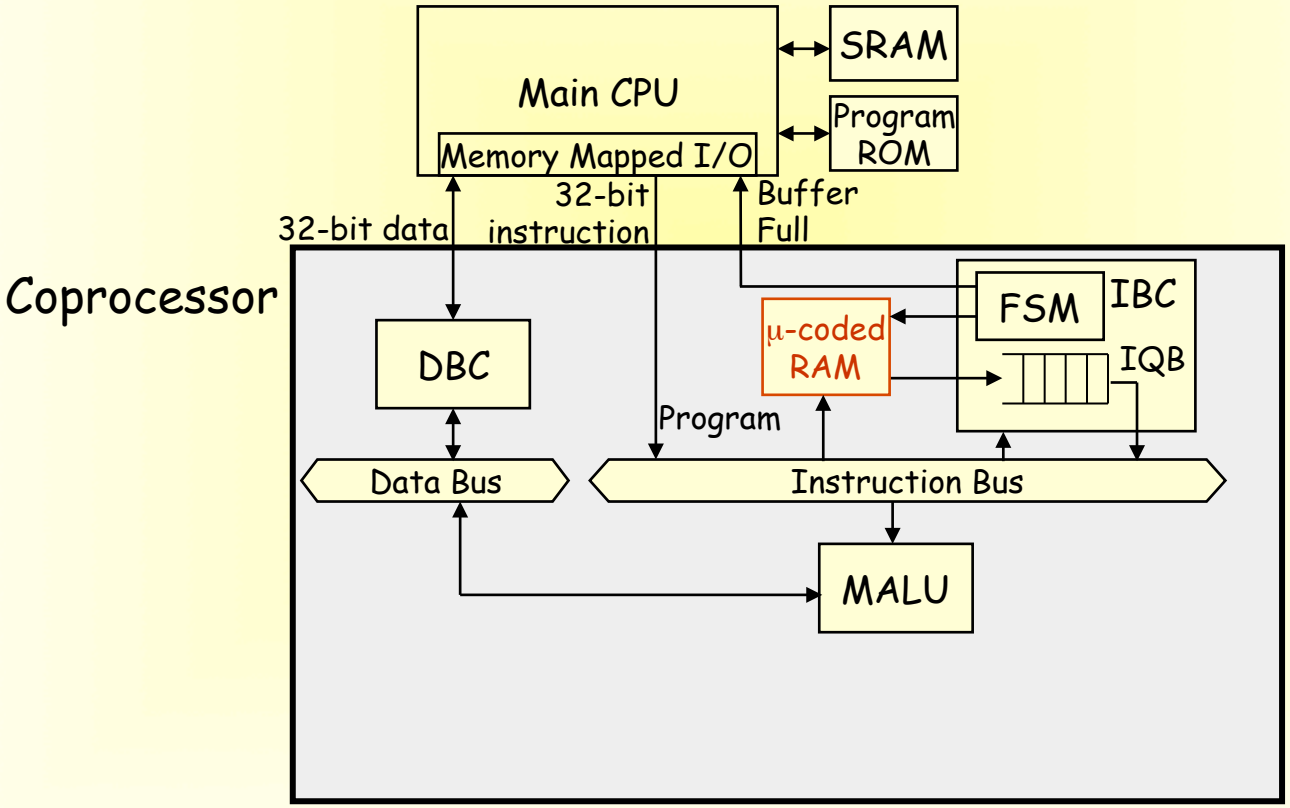
ECC coprocessor (I)

TYPE I: Smallest implementation (baseline)



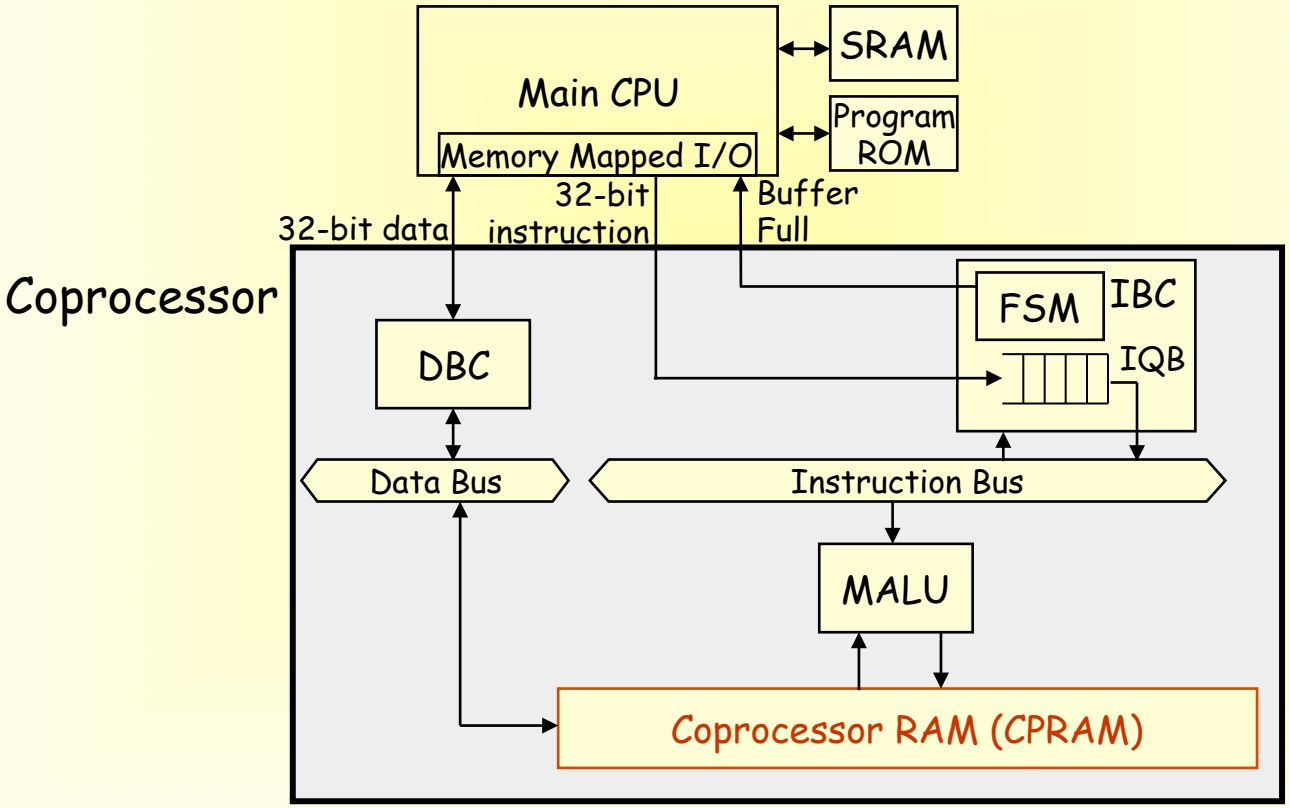
ECC coprocessor (II)

TYPE II: TYPE I + μ-coded RAM



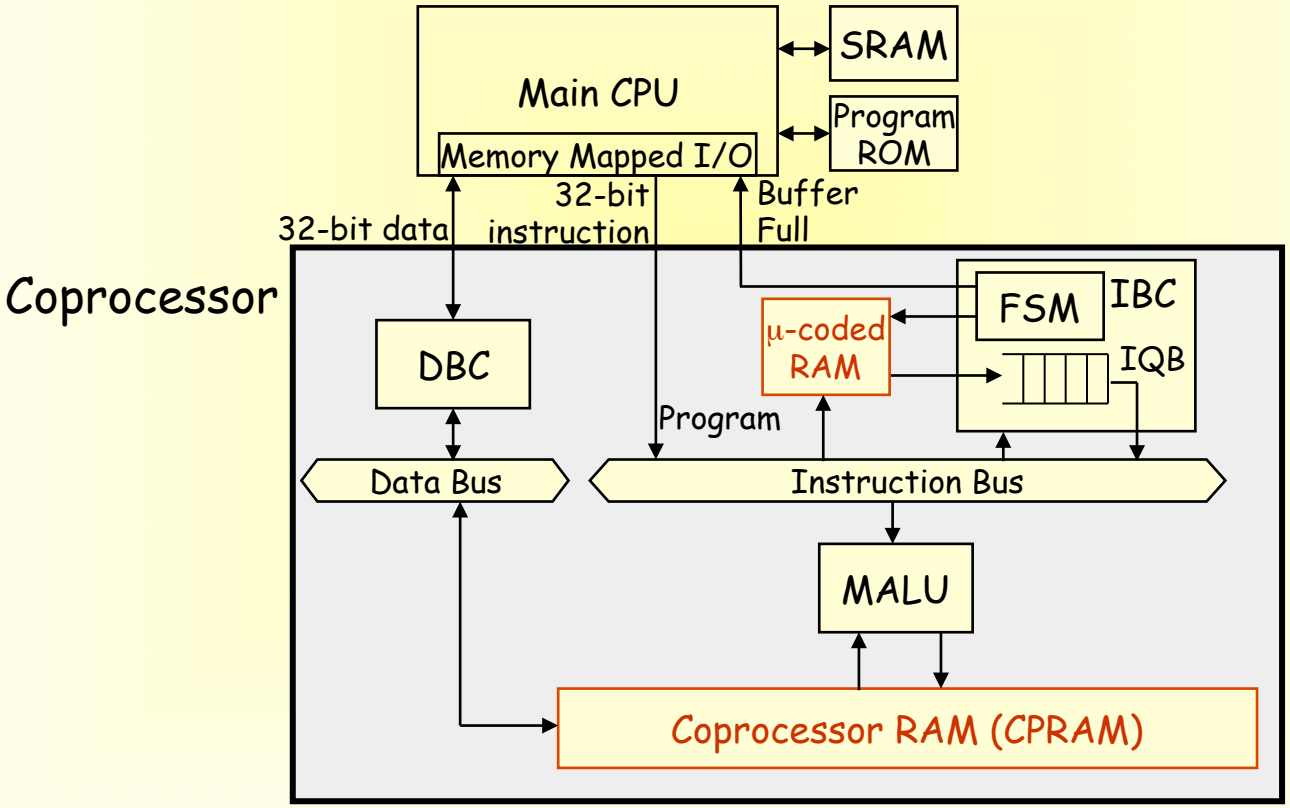
ECC coprocessor (III)

TYPE III: TYPE I + CPRAM



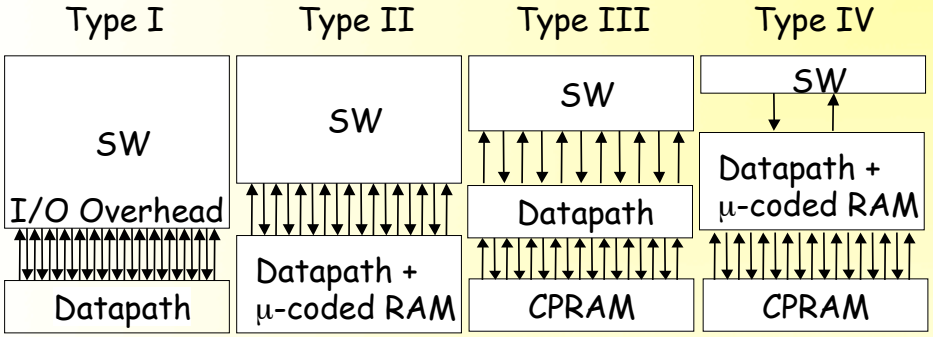
ECC coprocessor (IV)

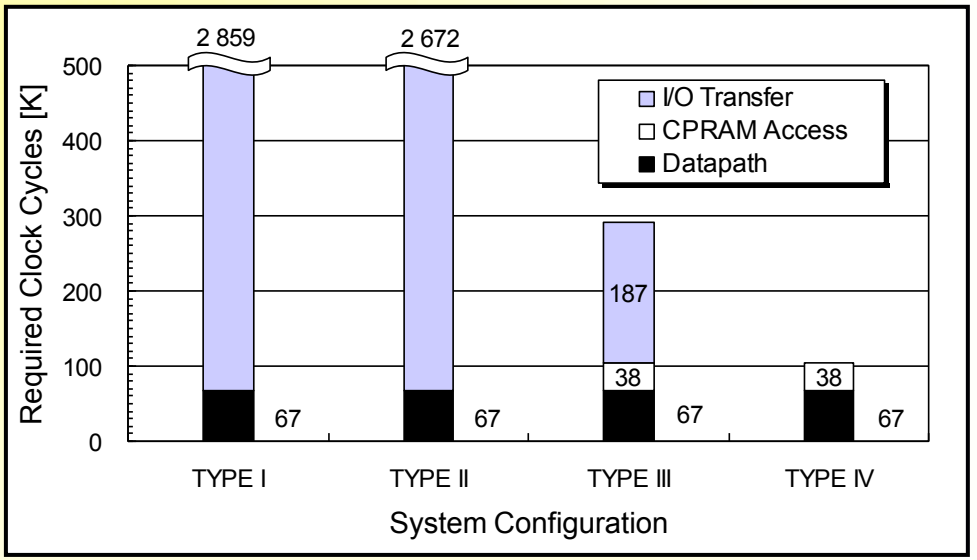
TYPE IV: TYPE I + CPRAM & μ-code RAM



Profiling ECC coprocessor

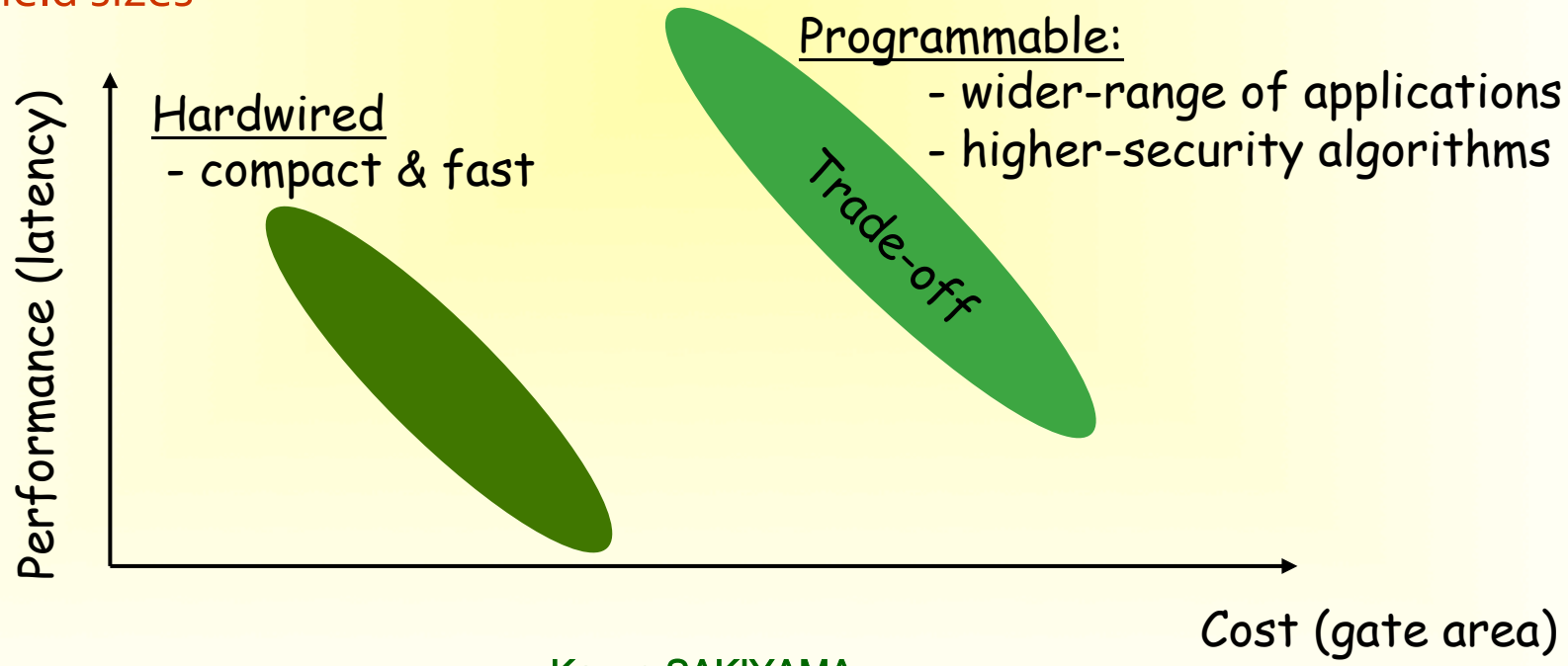
- Performance for HECC over GF(2^{83})
  - The number of cycles is reduced by CPRAM
  - Micro-coded RAM can avoid instruction stalls





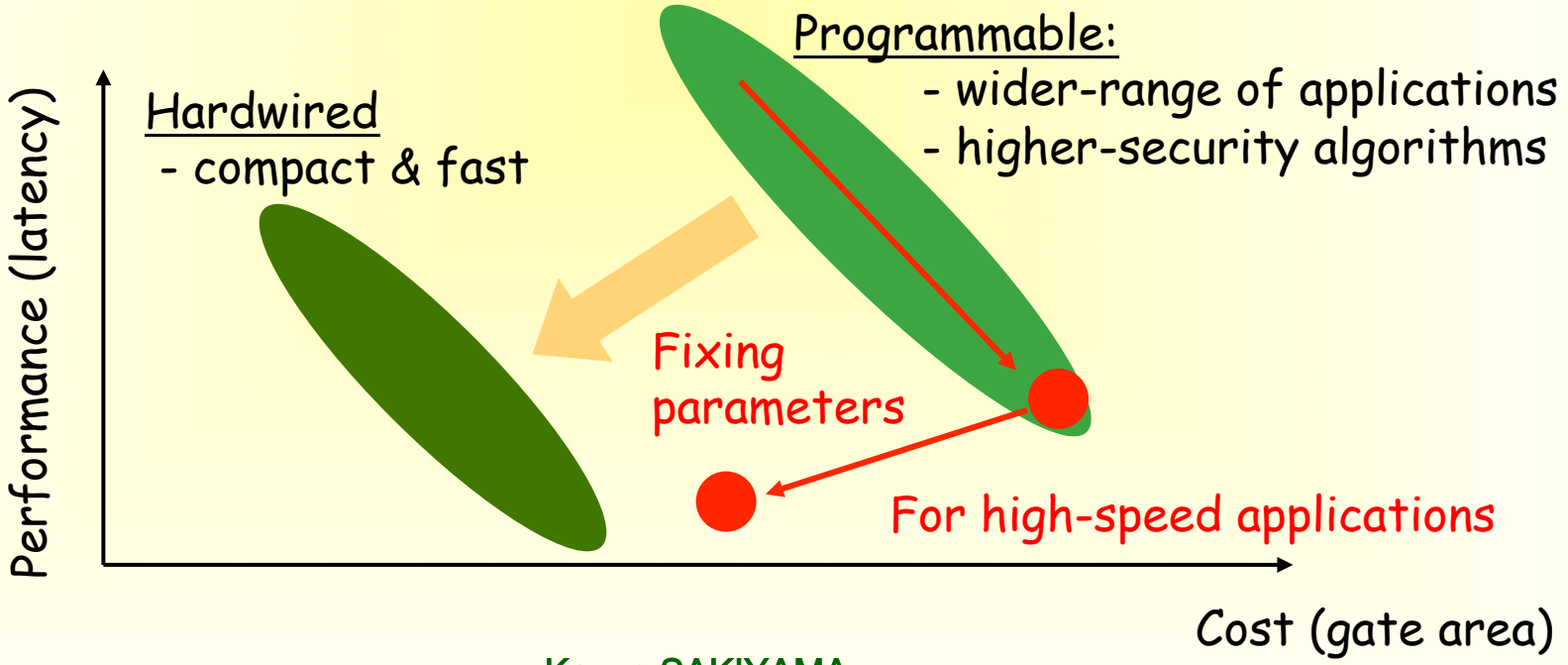
Cost, Performance and Security Trade-offs

- Programmable VS hardwired parameters
  - Curve parameters
  - Computational sequence: Coordinates, operation form.
  - Prime p, irreducible polynomial P(x)
  - Field sizes



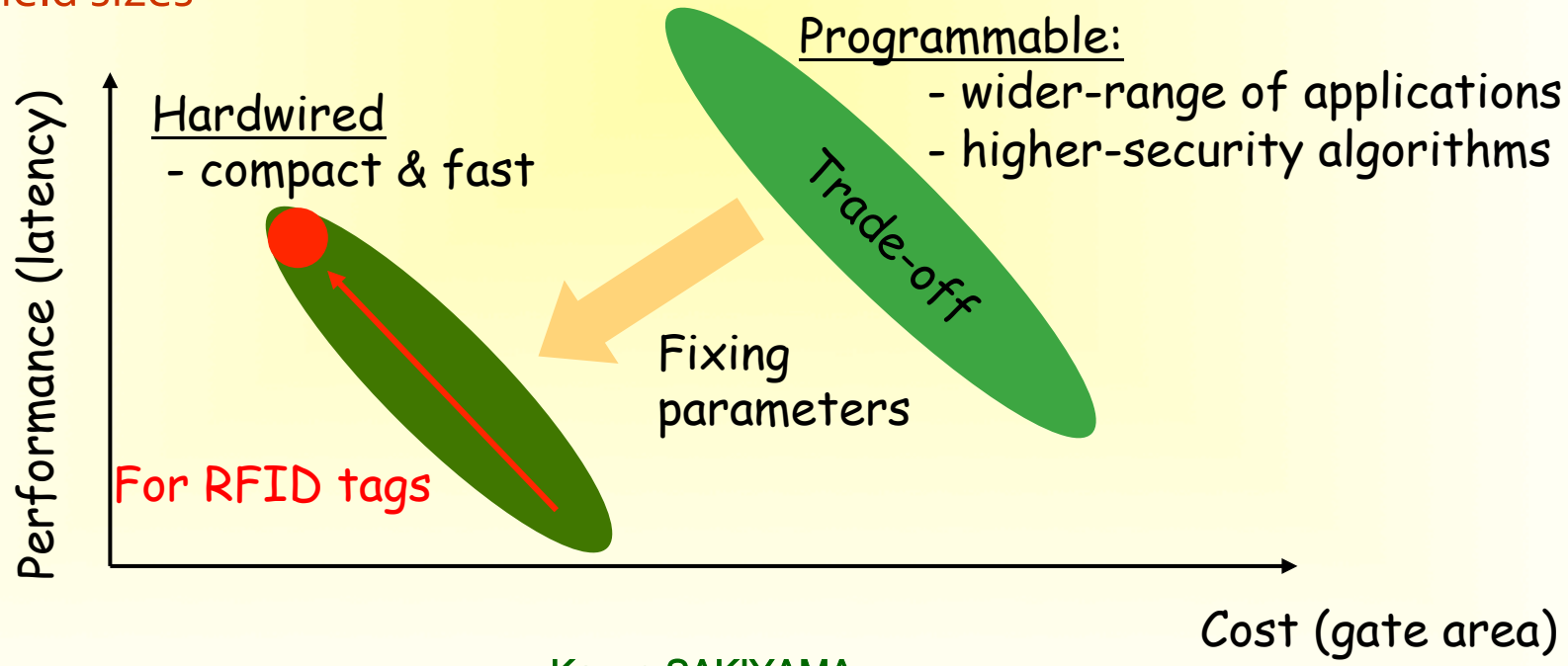
Cost, Performance and Security Trade-offs

- Programmable VS hardwired parameters
  - Curve parameters
  - Computational sequence: Coordinates, operation form.
  - Prime p, irreducible polynomial P(x)
  - Field sizes



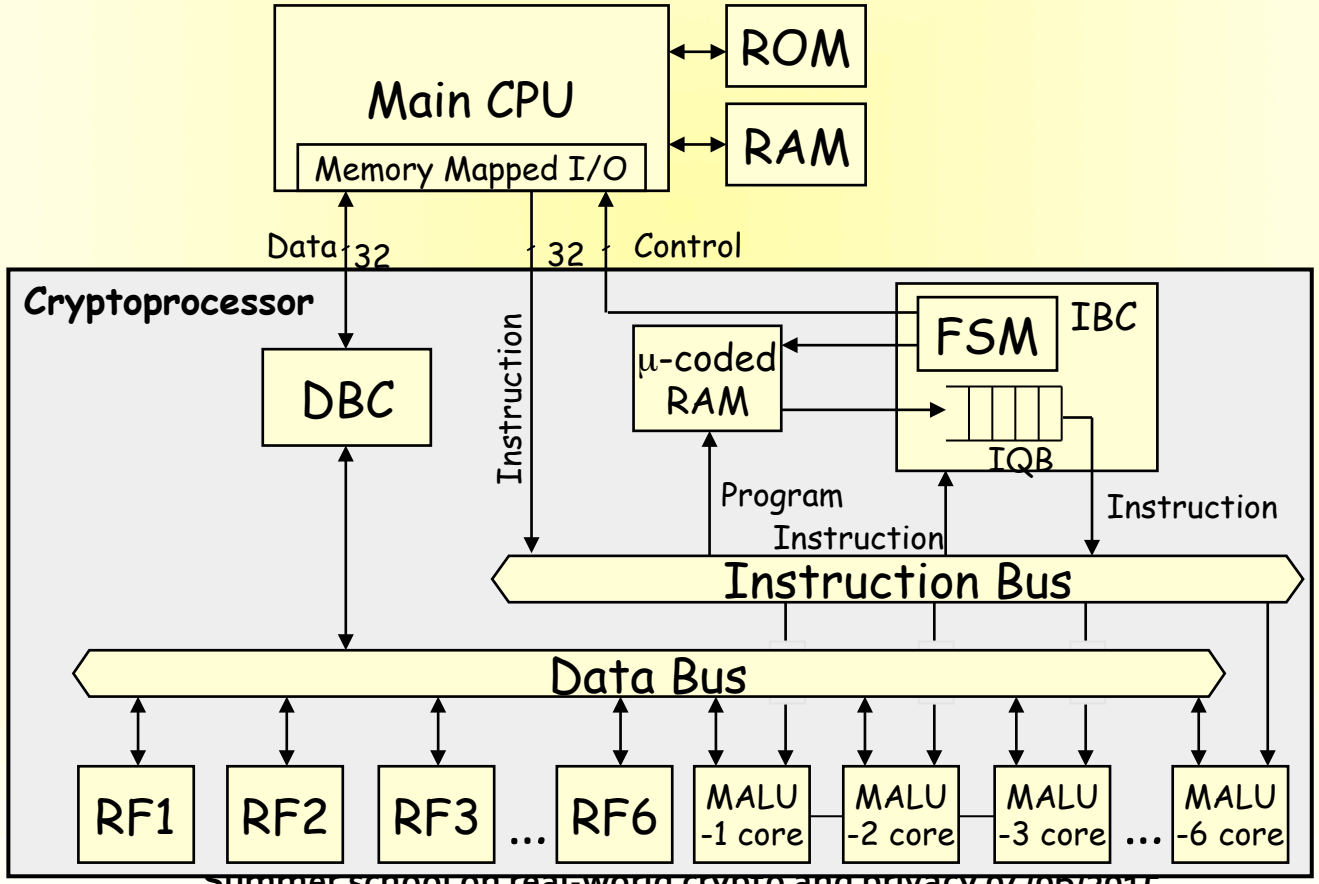
Cost, Performance and Security Trade-offs

- Programmable VS hardwired parameters
  - Curve parameters
  - Computational sequence: Coordinates, operation form.
  - Prime p, irreducible polynomial P(x)
  - Field sizes



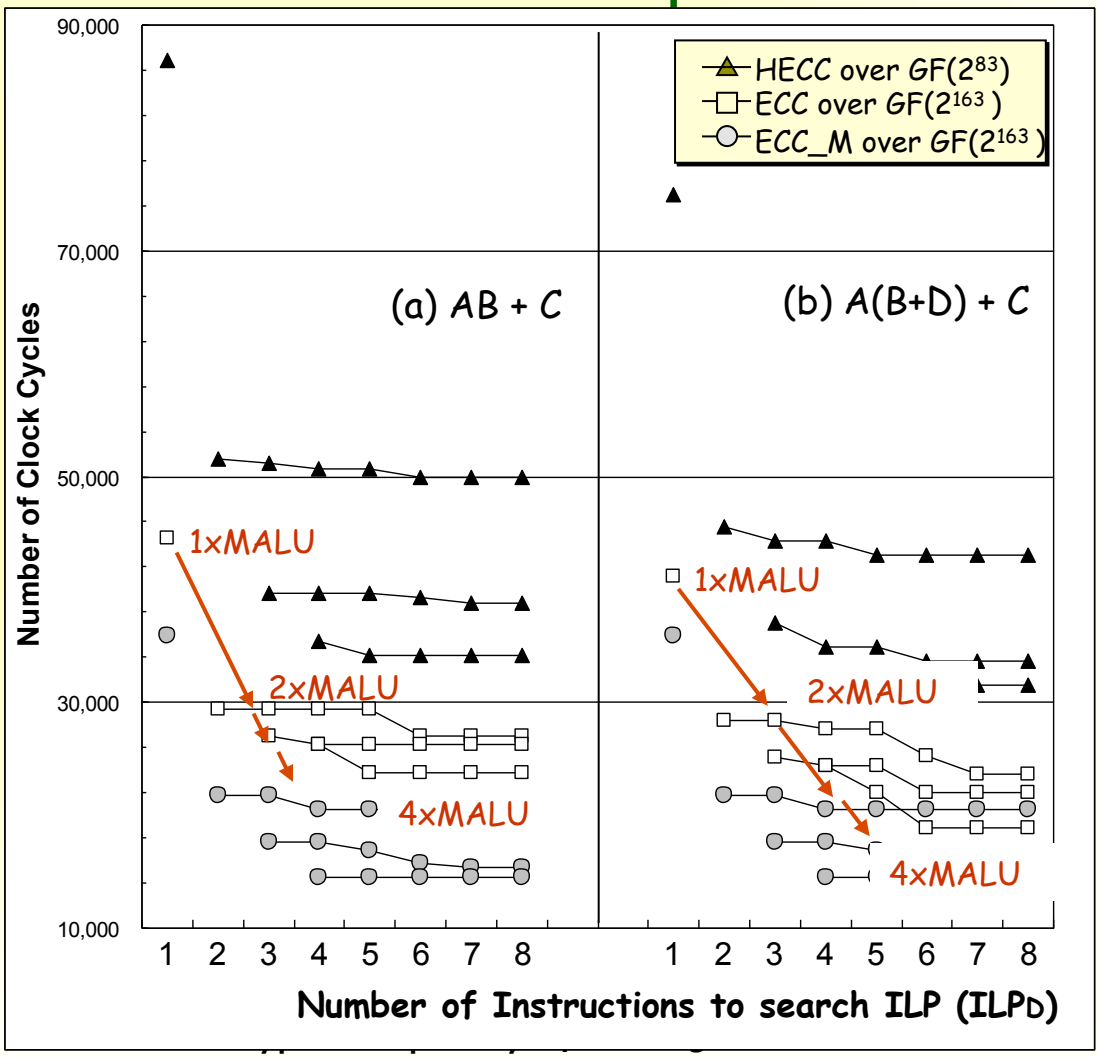
High-speed ECC & HECC

- Programmable high-speed coprocessor
  - ECC / HECC over GF(2^m)
  - Six MALU cores (MALU)



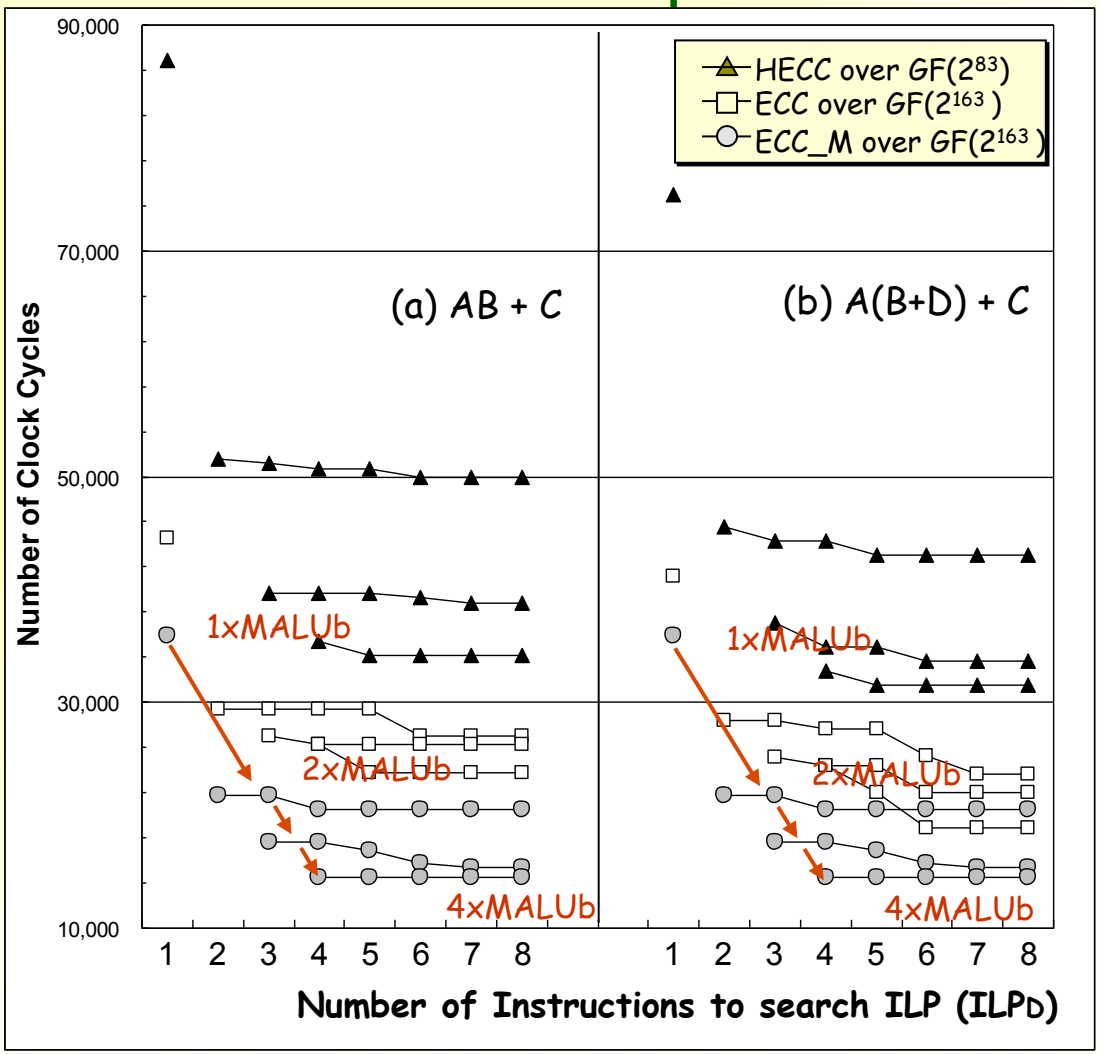
Parallelism in scalar multiplication

- How many instructions executed in parallel?
  - Operation form
    AB + C and
    A(B+C) + D
  - Check the data dependencies for ECC (and HECC)



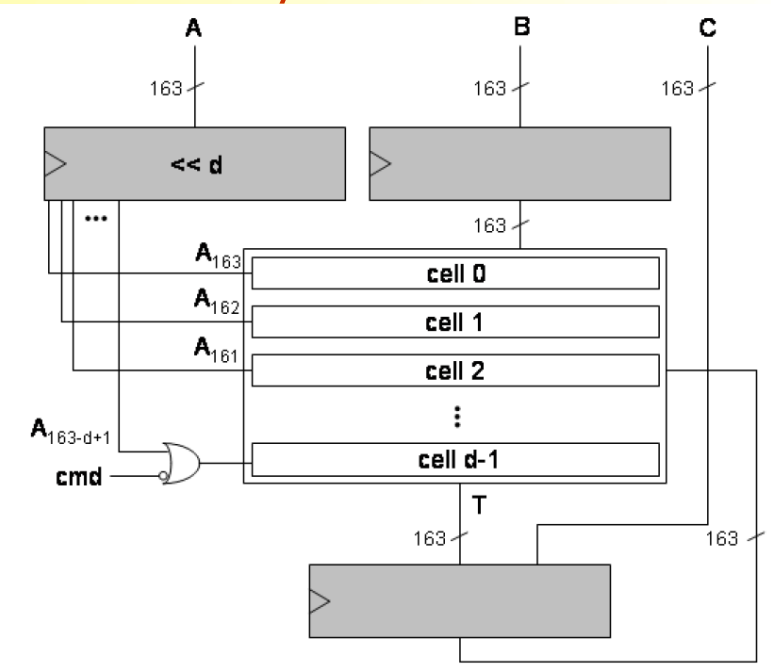
Parallelism in scalar multiplication

- How many instructions executed in parallel?
  - Operation form
    AB + C and
    A(B+C) + D
  - Check the data dependencies for ECC (and HECC)



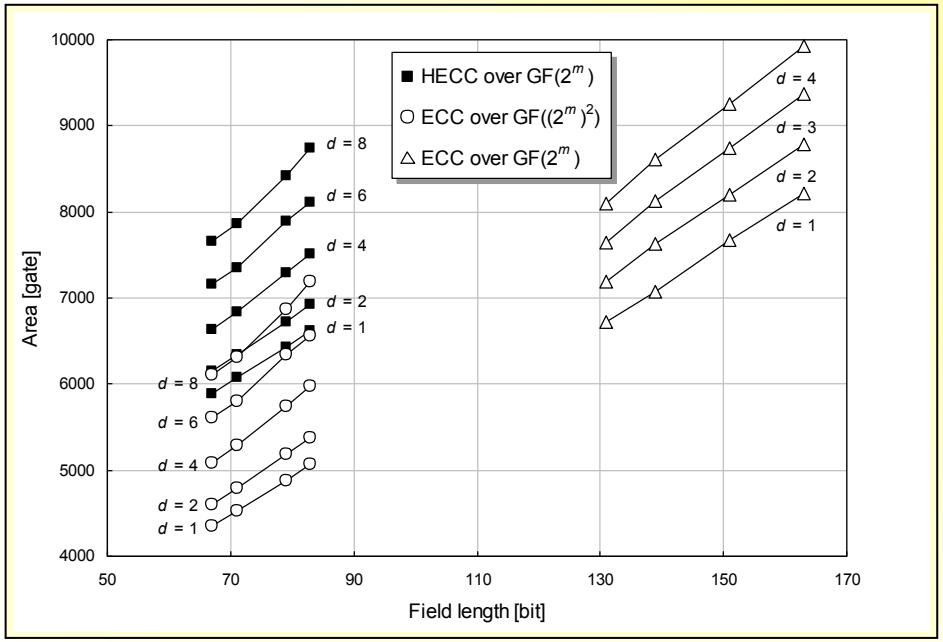
Lightweight ECC For RFID

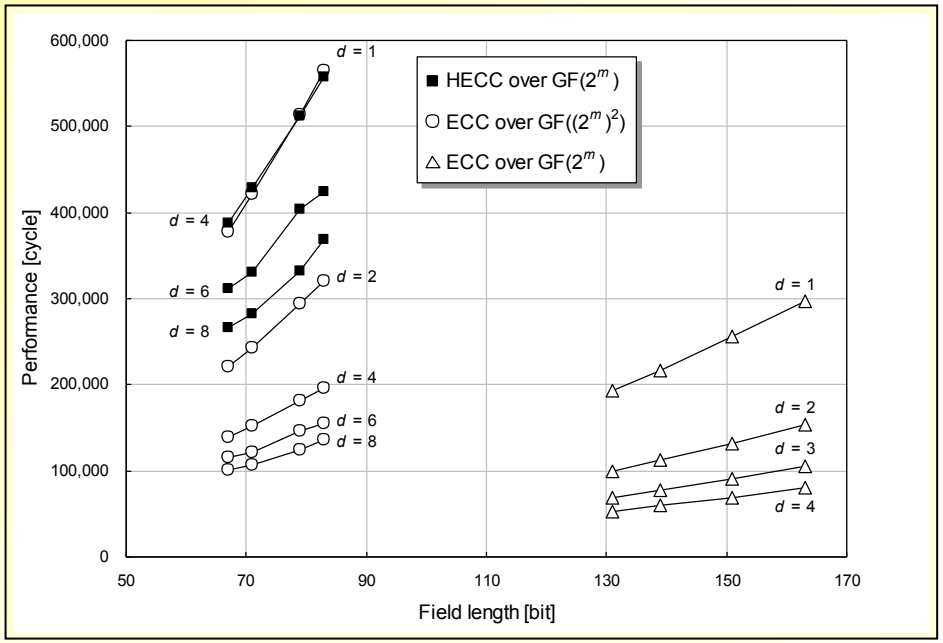
- Low-power MALU for RFID tags
  - Based on MALU
  - Cost and performance for ECC and HECC
  - Power estimation with 0.13-mm CMOS library
- Hardware sharing in MALU
  - Modular addition and multiplication use the same hardware (cell)



Low-power ECC (HECC) for RFID

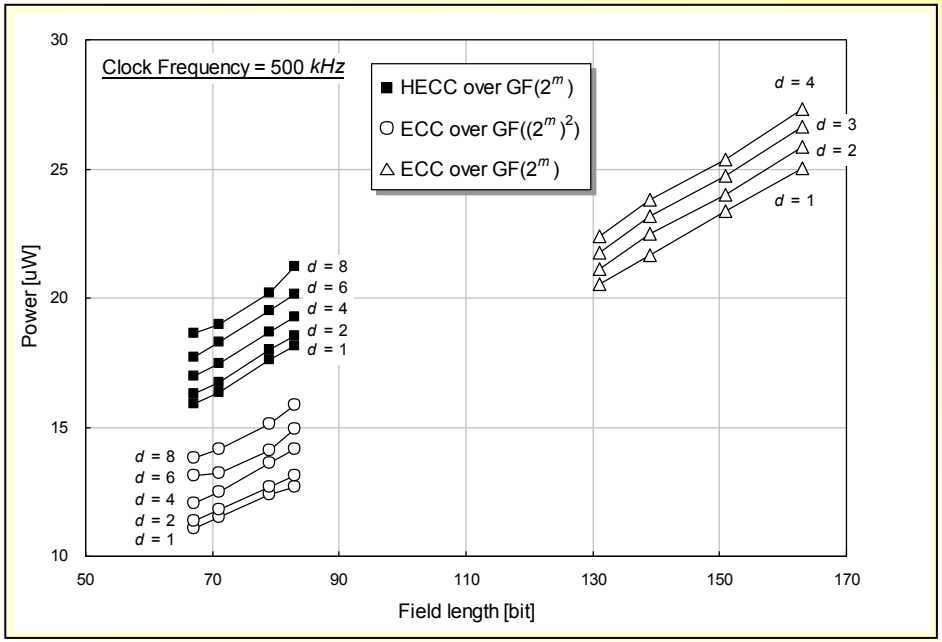
- Synthesis results (area & performance)
  - MALU + hardwired controller
  - ECC shows better performance (106 ms @ 500kHz, d = 4)
  - ECC over a composite field is smaller

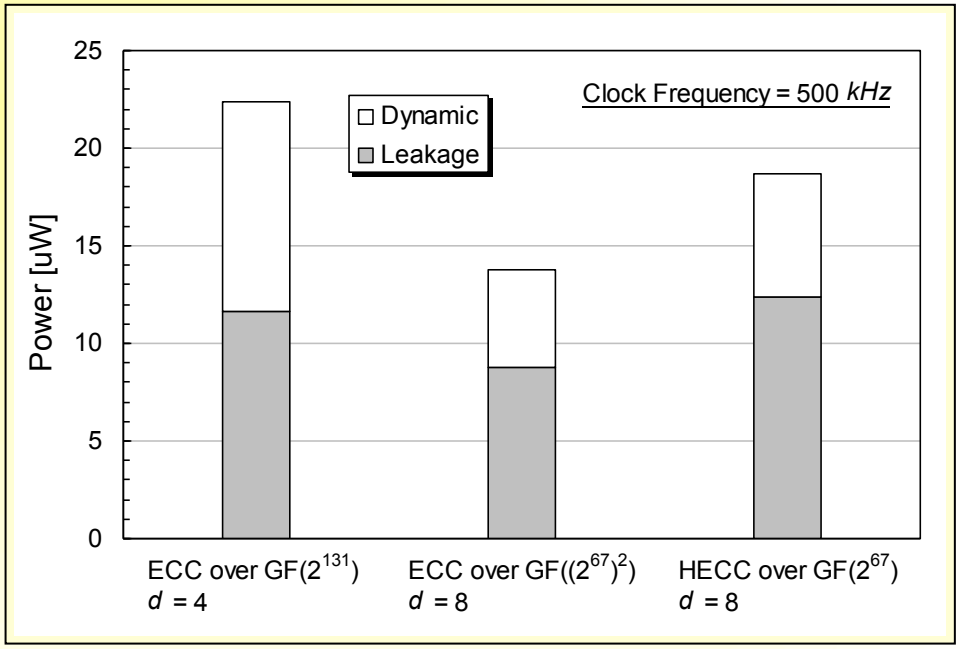




Low-power ECC (HECC) for RFID

- Synthesis results (power)
  - Results only for coprocessor (average)
  - Peak power evaluation is essential
  - RF interface will consume more power…





ECC coprocessor & Side-channel attacks

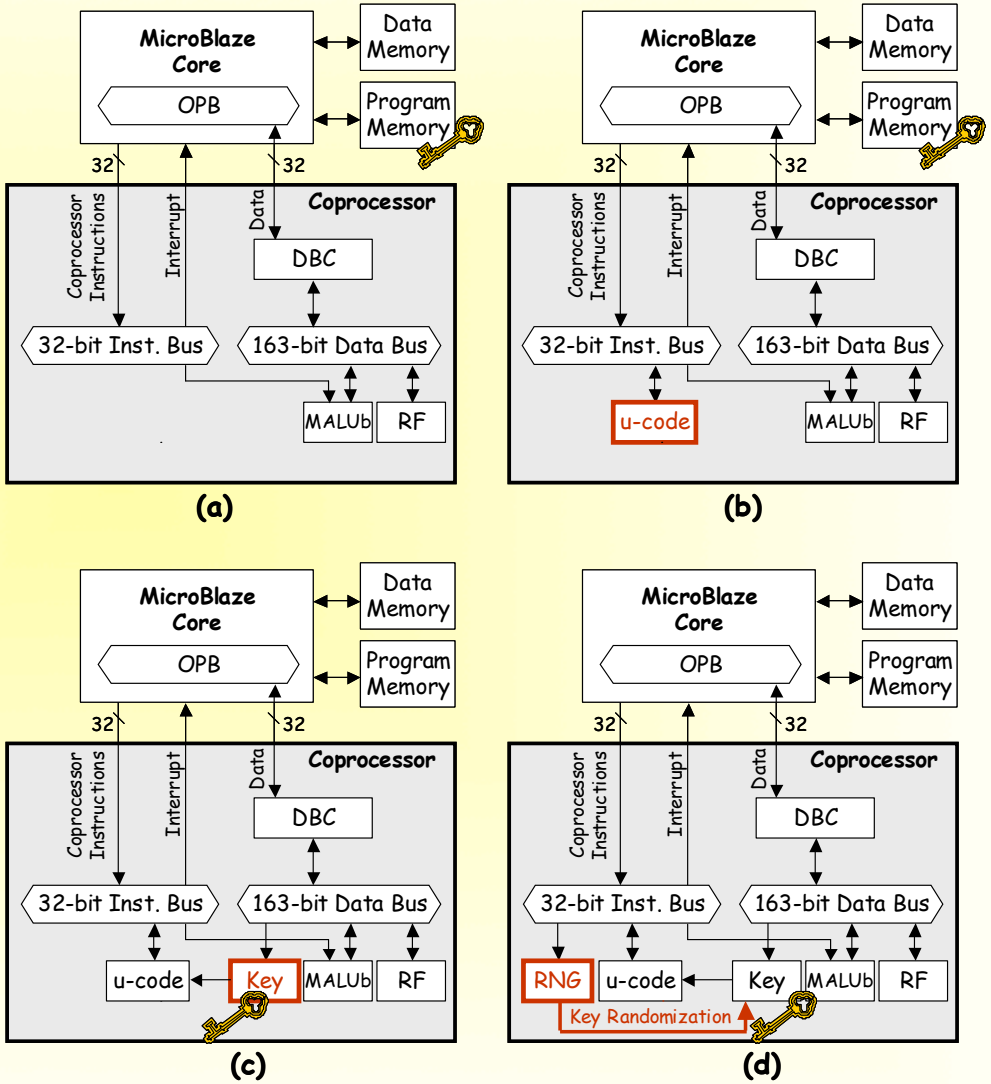
Four architectures

(a) TYPE A:
Baseline

(b) TYPE B:
+ μ-coded ROM

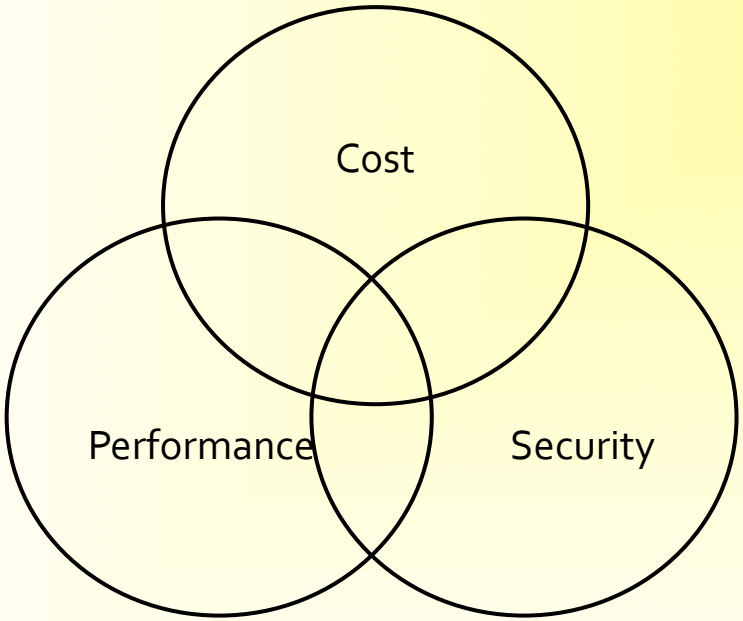
(c) TYPE C:
+ HW key register

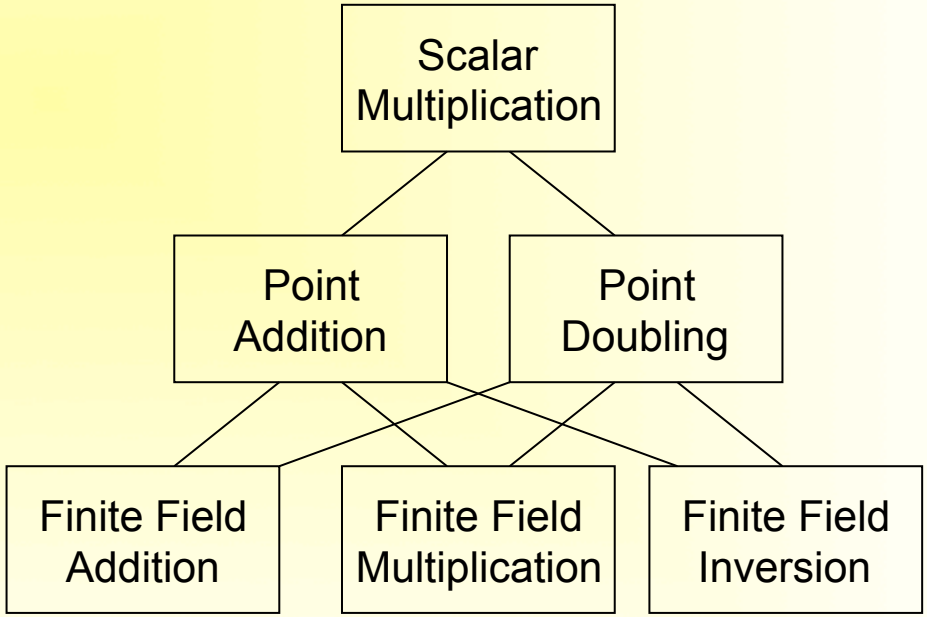
(d) TYPE D:
+ RND for key randomization



Conclusions

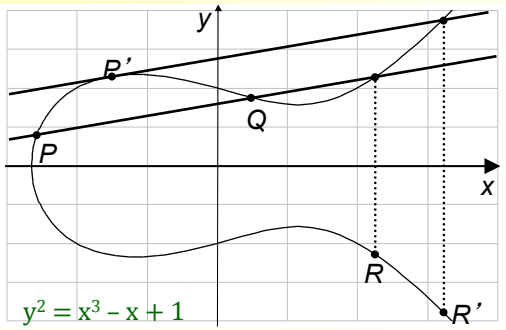
- Trade-offs in ECC hardware
  - Parallel processing
  - Operation forms: AB + C or A(B+D) + C
  - Cost, performance, security





Thank you for your attentions!

Questions?



Kazuo SAKIYAMA