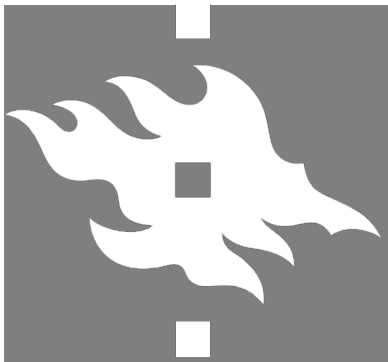


THE STATE-OF-THE-ART OF HARDWARE IMPLEMENTATIONS OF ELLIPTIC CURVE CRYPTOGRAPHY

Kimmo Järvinen

Department of Computer Science
University of Helsinki
kimmo.u.jarvinen@helsinki.fi

ECRYPT-CSA Workshop on Hardware Benchmarking

Bochum, Germany, June 7, 2017

INTRODUCTION

- ECC has become very popular because of high performance and short key sizes
- Huge numbers of HW implementations of ECC are available in the literature (We focus mainly on FPGAs)
- We discuss (the difficulties of) benchmarking ECC HW implementations and survey their state-of-the-art

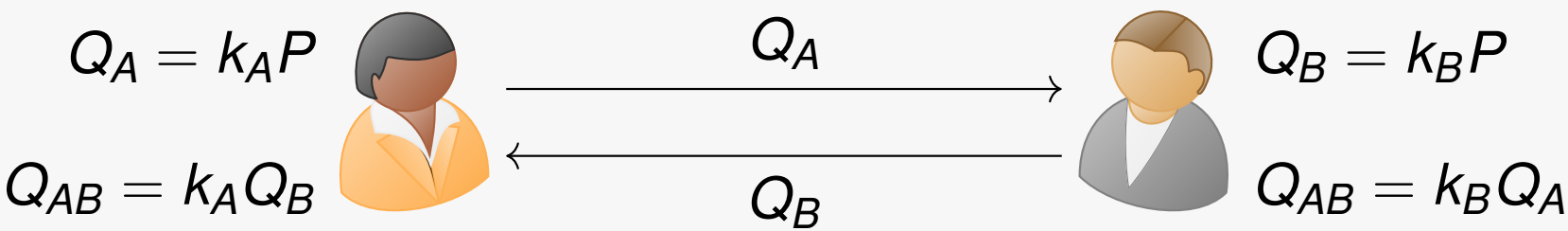
OUTLINE

- **Background on ECC**
  We present preliminaries of ECC
- **ECC Implementations for Different Use Cases**
  We discuss what kind of challenges different use cases bring for designing ECC implementations
- **General Discussion on Benchmarking ECC HW**
  We discuss benchmarking of ECC HW and the related difficulties
- **Benchmarking ECC Implementations**
  We survey specific state-of-the-art ECC implementations and benchmark them against each others

BACKGROUND ON ECC

ELLIPTIC CURVE CRYPTOGRAPHY

- **Elliptic Curve Discrete Logarithm Problem**
  Security is based on the difficulty of solving the ECDLP:
  Given two points P and $Q = kP$, find the integer k
- **Elliptic Curve Diffie-Hellman**



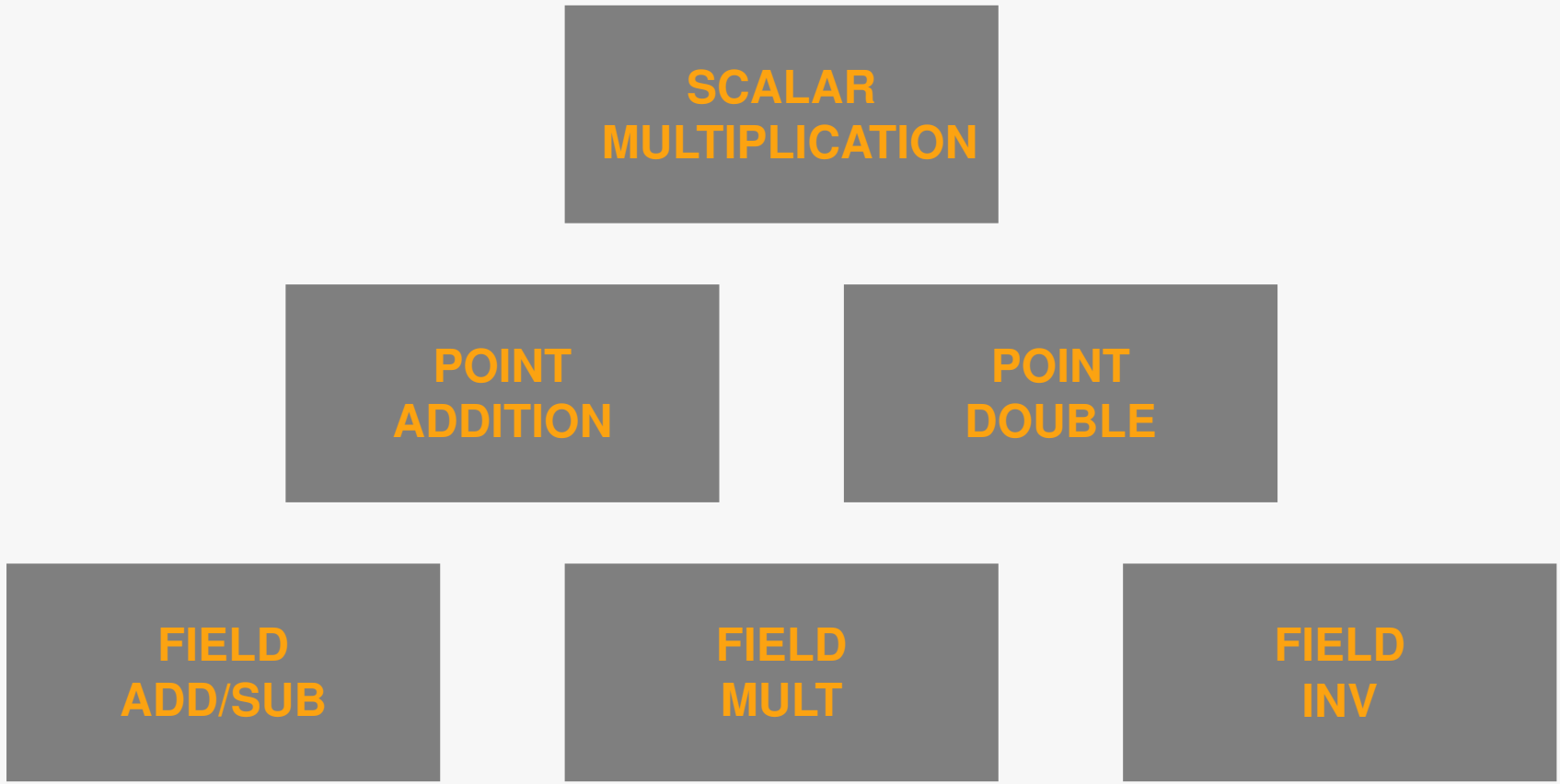
SCALAR MULTIPLICATION

- Efficient and secure computation of scalar multiplication essential for all elliptic curve cryptosystems
- Points on the curve form an additive Abelian group
- Scalar multiplication carried out with a series of
  (a) Point additions $P_3 = P_1 + P_2$ and
  (b) Point doublings $P_3 = P_1 + P_1 = 2P_1$
- Point operations computed with operations in $\mathbb{F}_q$. E.g., for $y^2 = x^3 + ax + b$, $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$ with $x_1 \neq x_2$:

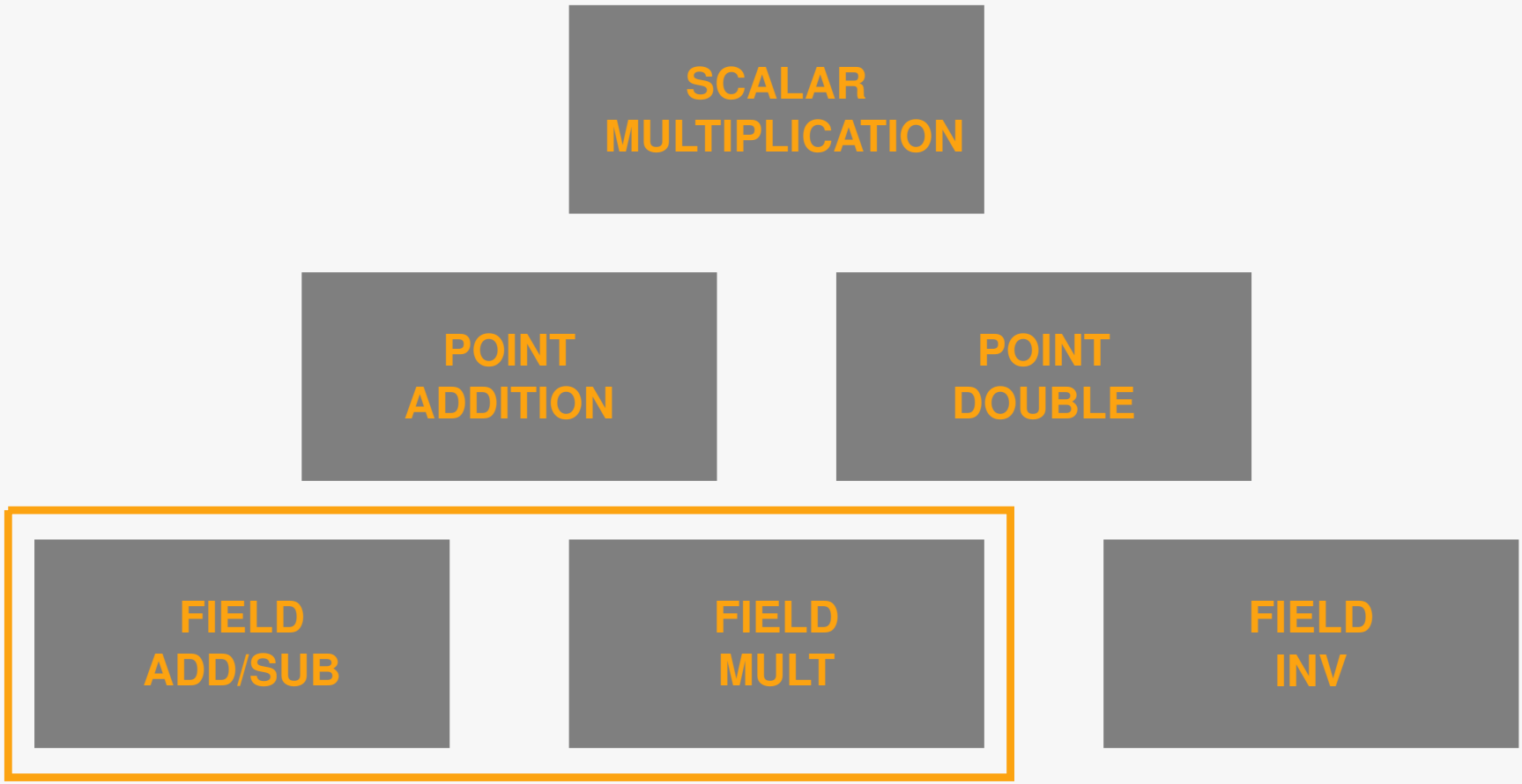
$$x_3 = \lambda^2 - x_1 - x_2, \; y_3 = \lambda(x_1 - x_3) - y_1 \quad \text{where } \lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

- Projective coordinates (X, Y, Z) to avoid inversions

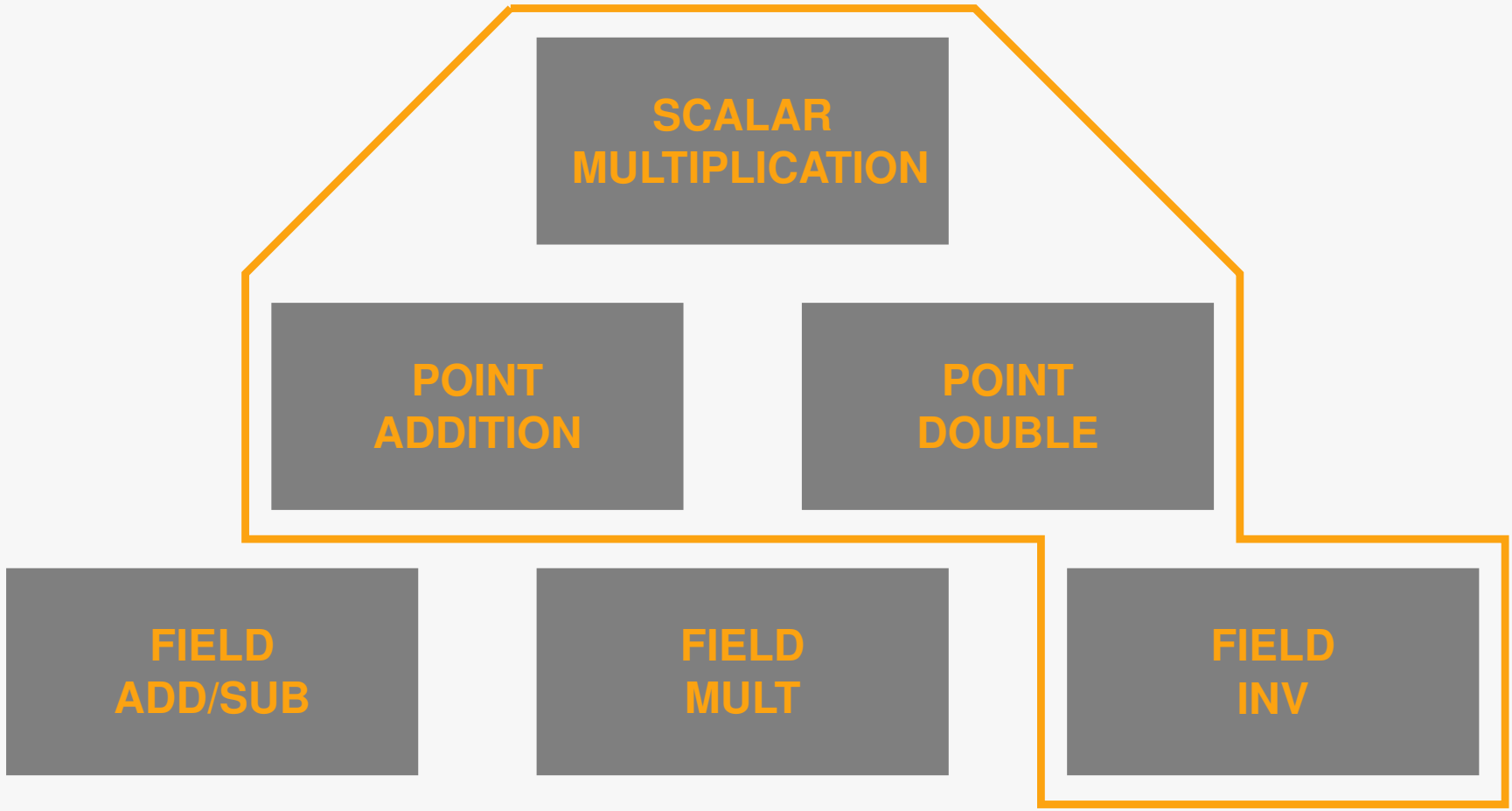
ECC HIERARCHY



ECC HIERARCHY



ECC HIERARCHY



FIELD ARITHMETIC
Multiplication

- **Field Multiplication**
  Critical operation that typically requires the most attention.
  One computes $c = a \times b$ in $\mathbb{F}_p$ by computing (1) $c' = a \times b$ over $\mathbb{Z}$ and (2) $c = c' \bmod p$
- **Prime vs. Binary Fields**
  (a) Binary fields do not have carry propagation and lead to very efficient multipliers in HW
  (b) Prime fields typically benefit less from HW; however, hardwired multipliers in modern FPGAs can be used

FIELD ARITHMETIC
Multiplication

- **Integer Multiplication**
  Large multiplications (e.g., 256 $\times$ 256-bit) typically require multiprecision algorithms even in HW
  (a) Operand-scanning vs. product-scanning vs. hybrid-scanning
  (b) Karatsuba algorithms
  (c) Squaring saves some partial multiplications because $a_i b_j = a_j b_i$ if $a = b$

FIELD ARITHMETIC
Multiplication

- **Modular Reduction**
  The type of prime greatly affects the implementation strategy and efficiency
  (a) Mersenne primes $2^k - 1$ would be the best because reduction is an addition $c'_L + c'_H$ but they are rare: $2^{127} - 1$, $2^{521} - 1$
  (b) Generalized Mersenne primes used for the NIST curves; e.g., $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ that leads to additions/subtractions with full words
  (c) Pseudo Mersenne primes $2^k - \gamma$ compute the reduction via $c'_L + \gamma c'_H$; e.g., Curve25519 uses $2^{255} - 19$
  (d) Barrett reduction, Montgomery domain, etc.

FIELD ARITHMETIC
Inversion

- **Inversion**: Extended Euclidean Algorithm (EEA) vs. Fermat's Little Theorem (FLT)
- FLT computes $a^{-1} = a^{q-2}$ in $\mathbb{F}_q$ via a series of squarings and multiplications
- FLT reuses the multiplier and requires only control logic
- FLT is inherently constant time
- EEA can be faster if implemented with a dedicated unit

POINT OPERATIONS

- Algorithms for point addition and doubling
- Series of field operations
- Explicit-Formulas Database
- Relevant things:
  - Number of operations (multiplications and squarings)
  - Parallelism
  - Number of registers
  - Atomicity or completeness
  - etc.

SCALAR MULTIPLICATION

Input: Integer $k = \sum_{i=0}^{\ell-1} k_i 2^i$, point P
Output: Point $Q = kP$
$Q \leftarrow \mathcal{O}$
for $i = \ell - 1$ **to** 0 **do**
    $Q \leftarrow 2Q$
    if $k_i = 1$ **then** $Q \leftarrow Q + P$

Structure of Scalar Multiplication:

- Preprocessing: precomputations with P, preprocessing of k
- Main for-loop: A series of point operations
- Coordinate conversion (inversion)

ECC IMPLEMENTATIONS FOR DIFFERENT USE CASES

WHY DO WE NEED HARDWARE?

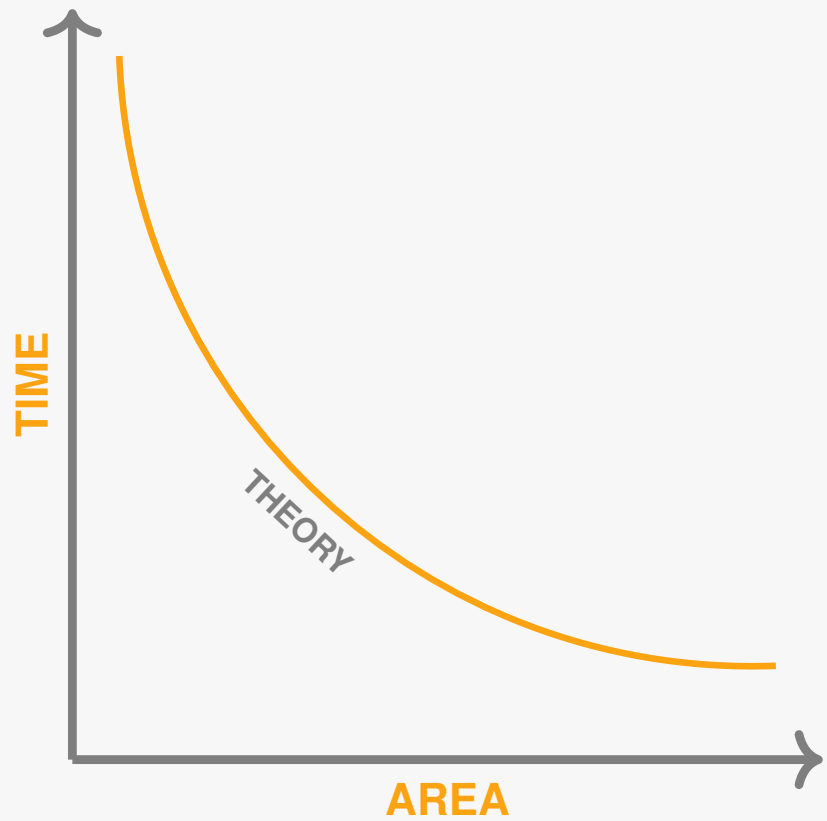
- **Fast Processing Speeds**
  HW provides very high throughput and/or low latency and can free resources from the main processor
- **Minimal Resource Usage**
  HW is required if resources (e.g., chip area, power, energy, etc.) are extremely scarce
- **Implementation Security**
  HW maximizes implementation security

LOW LATENCY

- **Optimization Goal**: Compute a scalar multiplication as fast as possible (time from input to output)
- The traditional optimization goal; vast majority of published ECC implementations fall into this category
- Use fast multipliers, utilize parallelism in point operations, use precomputations, etc.

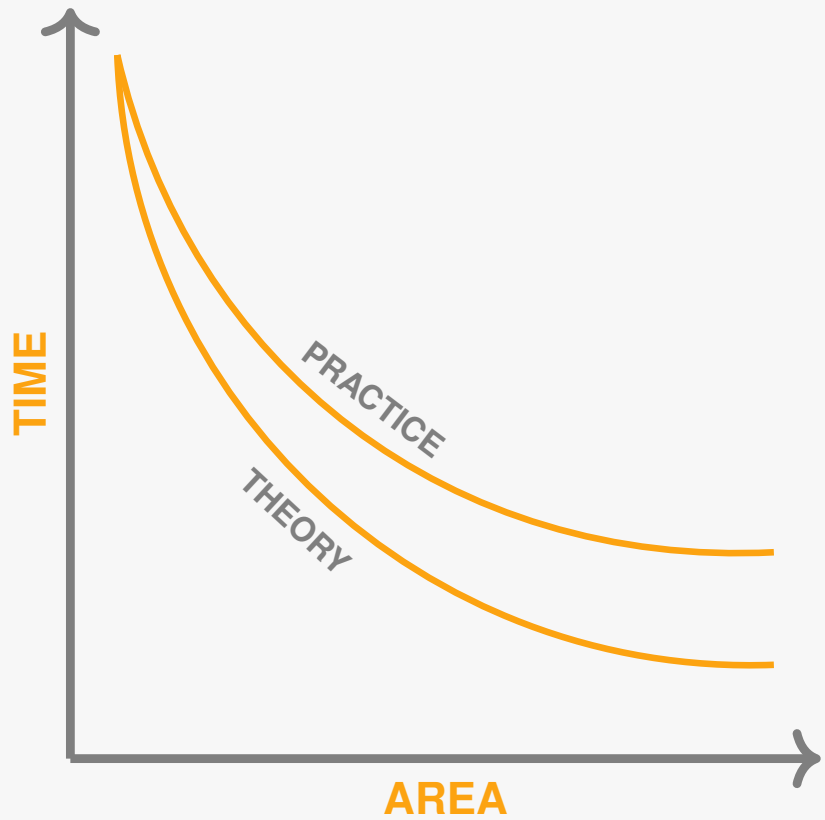
LOW LATENCY
Field Operations

- The latency of field multiplication dominates
  $\Rightarrow$ Use a faster multiplier
- Designing a fast, e.g., 256-bit multiplier is difficult
- In theory, using more area gives a faster multiplier
- Small subproducts over several clock cycles and deep pipelines are often better in practice



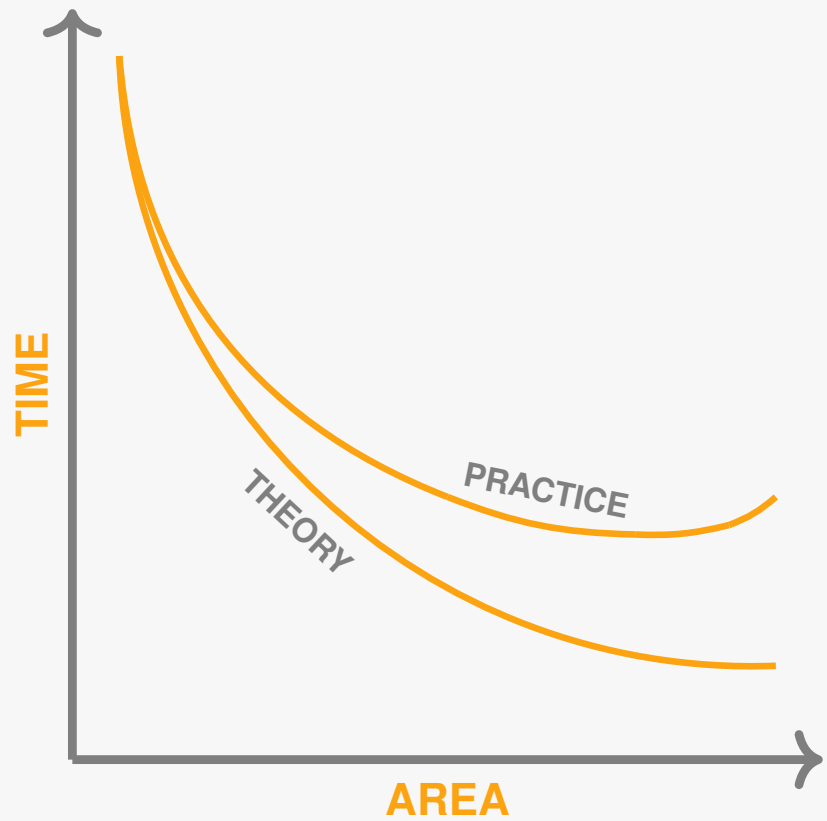
LOW LATENCY
Field Operations

- The latency of field multiplication dominates
  $\Rightarrow$ Use a faster multiplier
- Designing a fast, e.g., 256-bit multiplier is difficult
- In theory, using more area gives a faster multiplier
- Small subproducts over several clock cycles and deep pipelines are often better in practice



LOW LATENCY
Field Operations

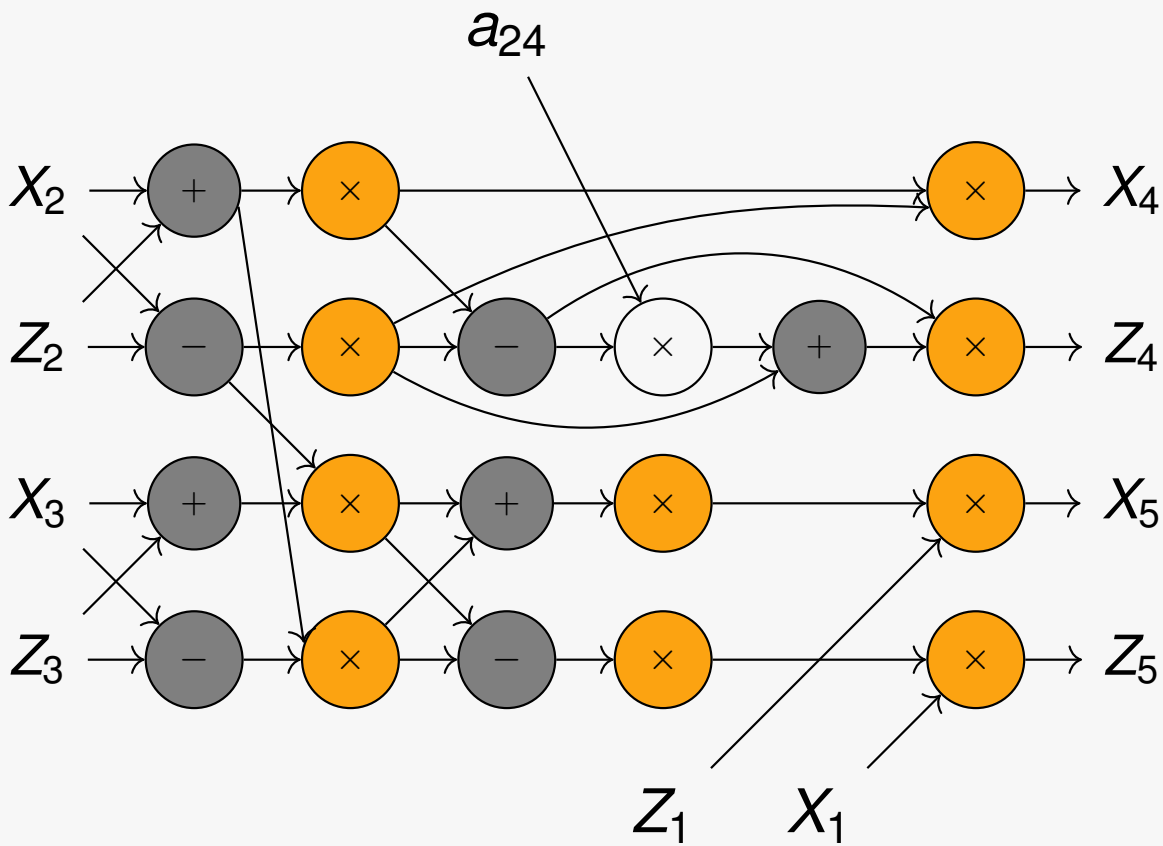
- The latency of field multiplication dominates
  $\Rightarrow$ Use a faster multiplier
- Designing a fast, e.g., 256-bit multiplier is difficult
- In theory, using more area gives a faster multiplier
- Small subproducts over several clock cycles and deep pipelines are often better in practice



LOW LATENCY
Point Operations

- Independent field operations in point operations can be computed in parallel (or in a pipeline)
- Identify the number of parallel arithmetic blocks from the point operation formulas (e.g., Explicit Formula Database)
- Memory access may become a problem

LOW LATENCY
Point Operations



Montgomery (1987): Differential addition and doubling

`https://hyperelliptic.org/EFD/g1p/auto-montgom-xz.html#ladder-ladd-1987-m-3`

LOW LATENCY
Scalar Multiplication

- Minimize the critical path
- Precomputations (window)
  - Precompute multiples of P; e.g., $-(2^w-1)P, \ldots, -3P, -P, P, 3P, \ldots, (2^w-1)P$
  - Convert the integer k appropriately
  - Reduces the number of point additions; fixed P allows reducing the number of point doublings also
  - Also constant-time alternatives exist
- Fast endomorphisms
  - **Koblitz curves:** Frobenius map (x^2, y^2) replaces doublings
  - **GLV/GLS curves:** $\Psi(P) = \lambda P \quad \Rightarrow \quad kP = k_1 P + k_2 \Psi(P)$ when $k = k_1 + k_2 \lambda$

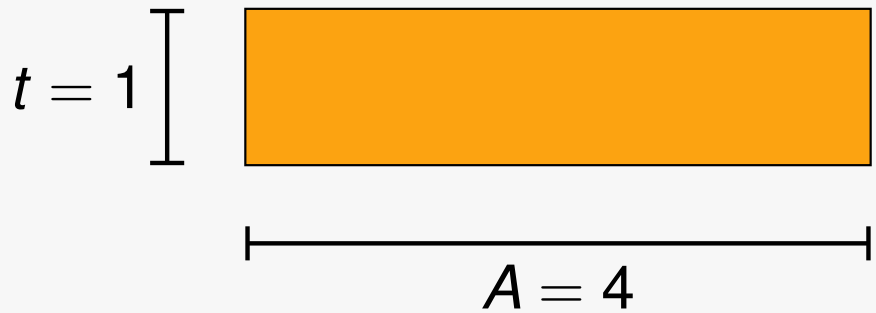
HIGH THROUGHPUT

- **Optimization Goal**: Compute as many scalar multiplications as possible in certain time (operations per second)
- Simply making t, latency of one scalar multiplication, smaller is not feasible (or even possible)
- Typically more efficient to increase N, the number of concurrent scalar multiplications, with parallelism and pipelining

$$T = \frac{N}{t}$$

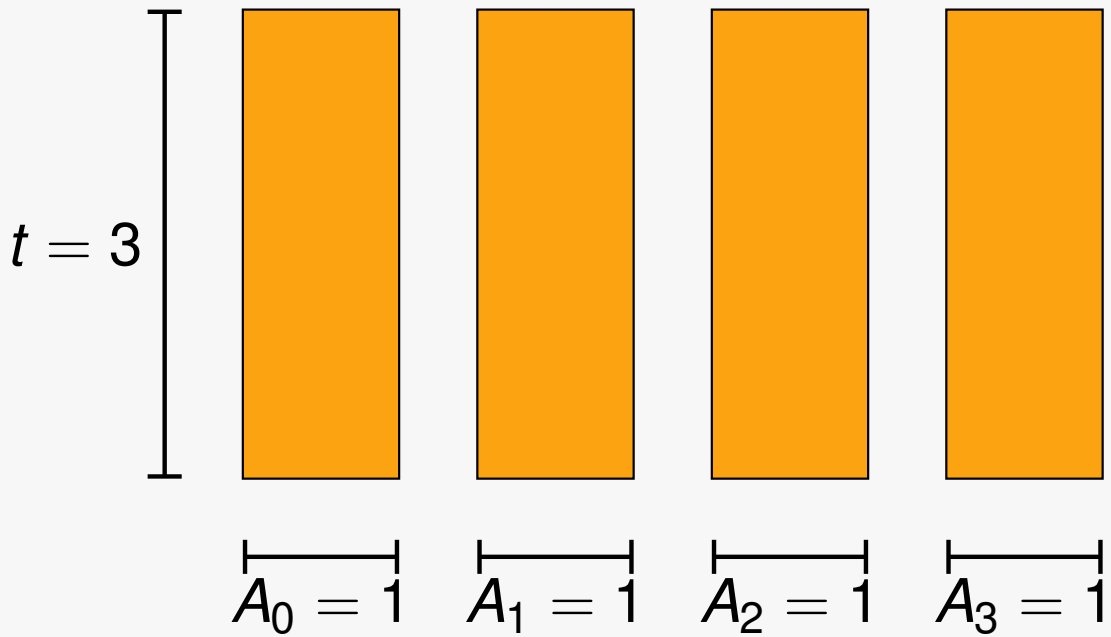
HIGH THROUGHPUT

$$T = \frac{N}{t} = \frac{1}{1} = 1$$



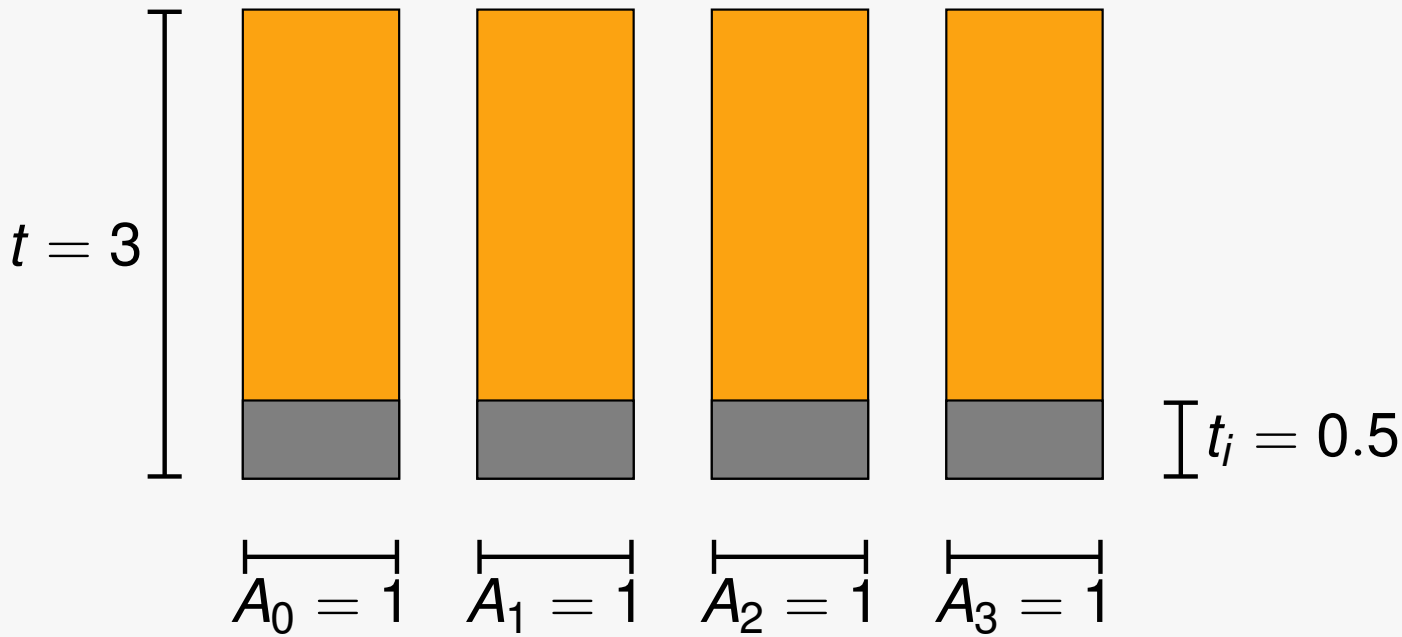
HIGH THROUGHPUT

$$T = \frac{N}{t} = \frac{4}{3} = 1.33$$



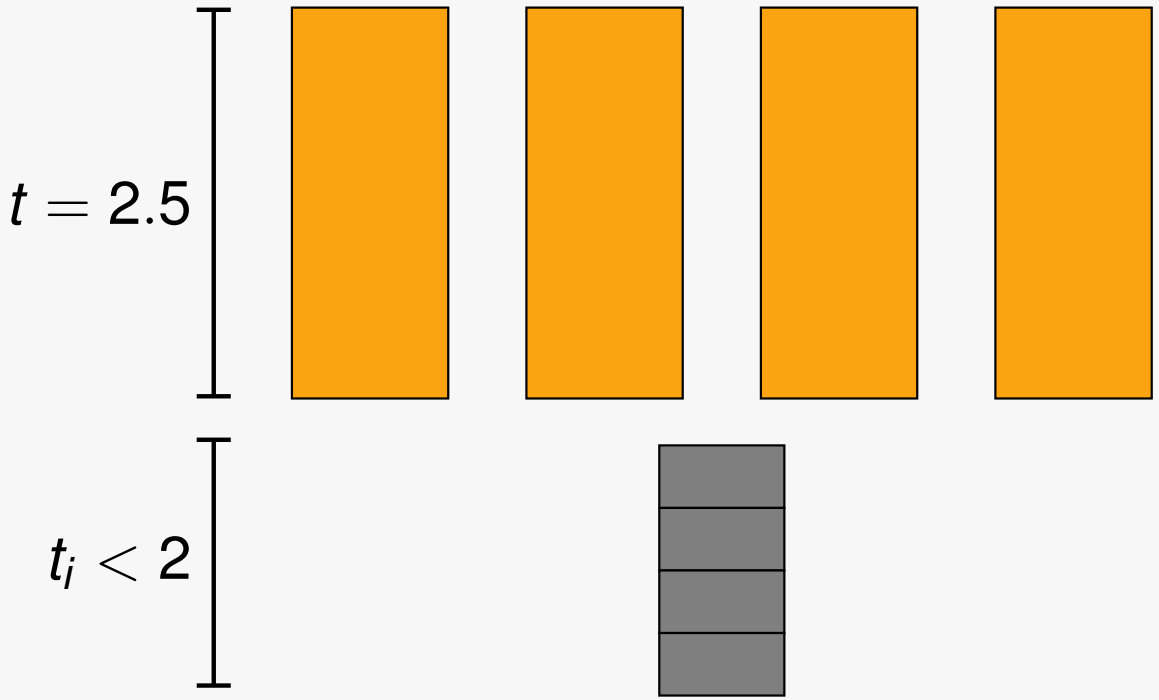
HIGH THROUGHPUT

$$T = \frac{N}{t} = \frac{4}{3} = 1.33$$



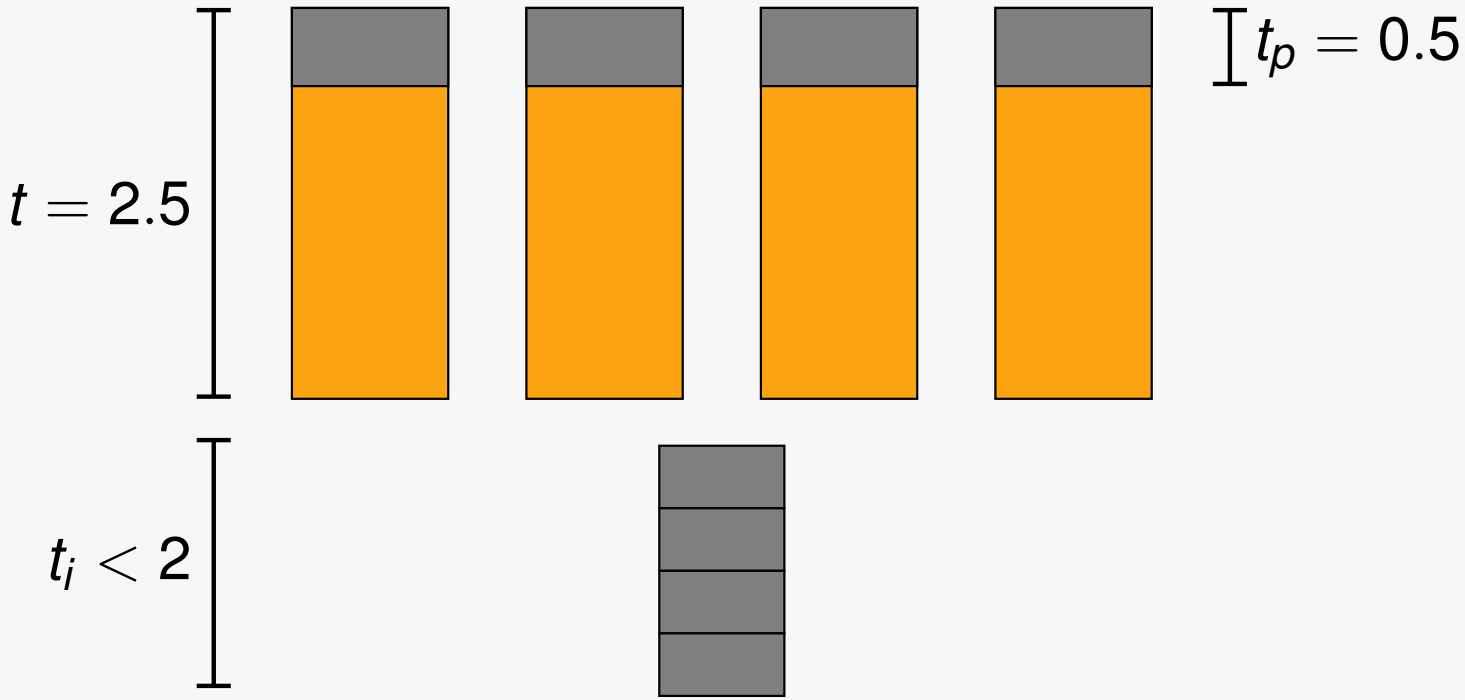
HIGH THROUGHPUT

$$T = \frac{N}{t} = \frac{4}{2.5} = 1.6$$

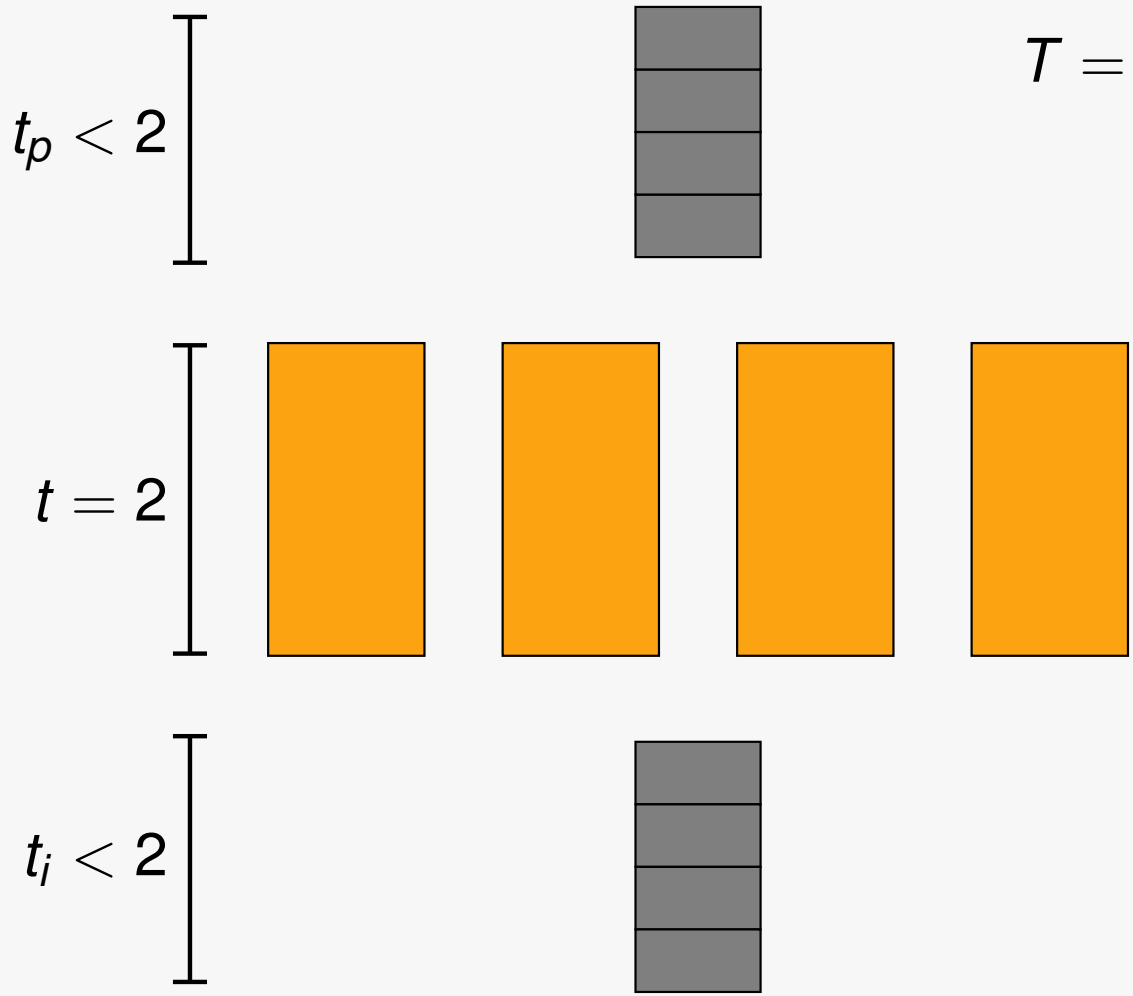


HIGH THROUGHPUT

$$T = \frac{N}{t} = \frac{4}{2.5} = 1.6$$



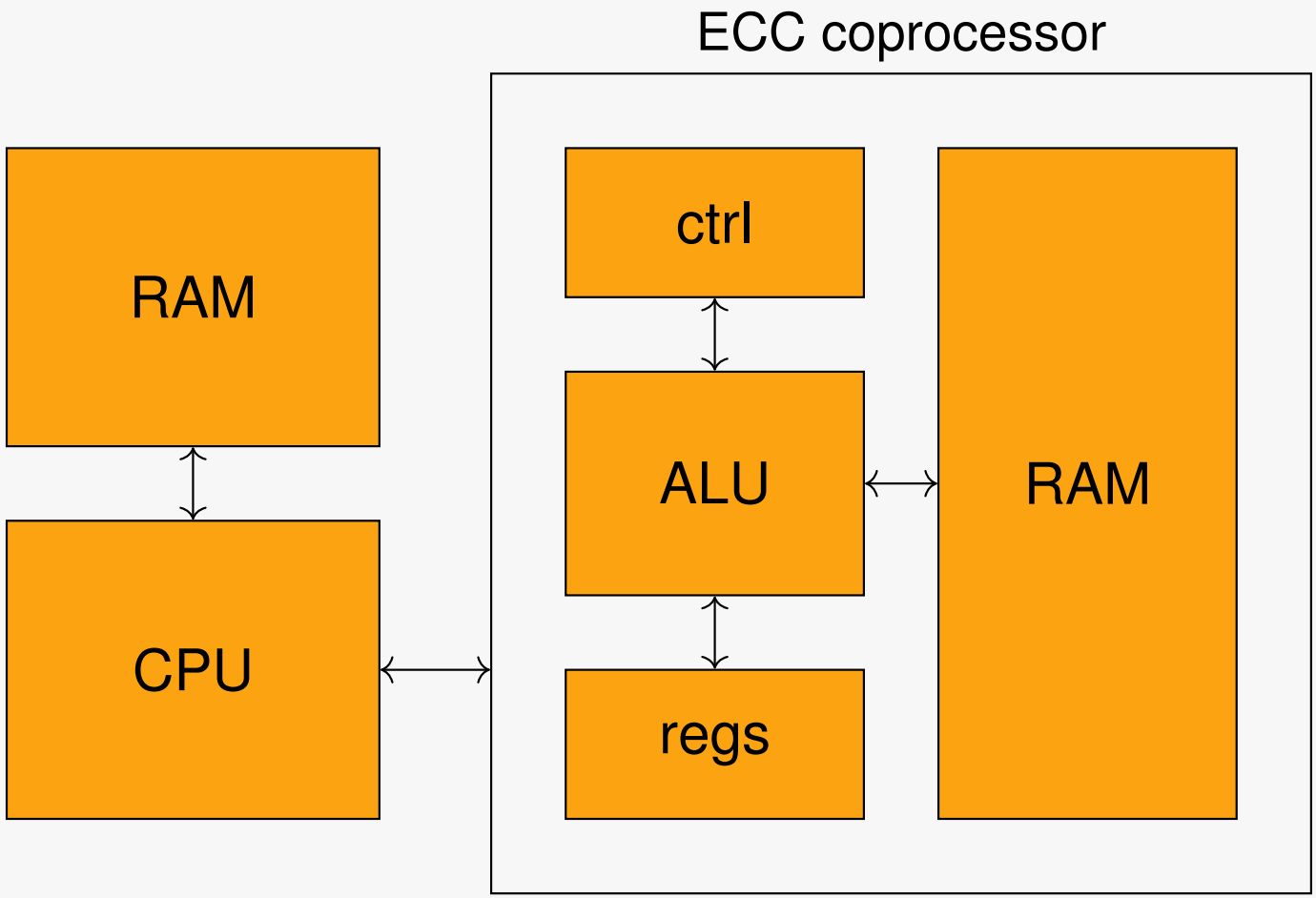
HIGH THROUGHPUT



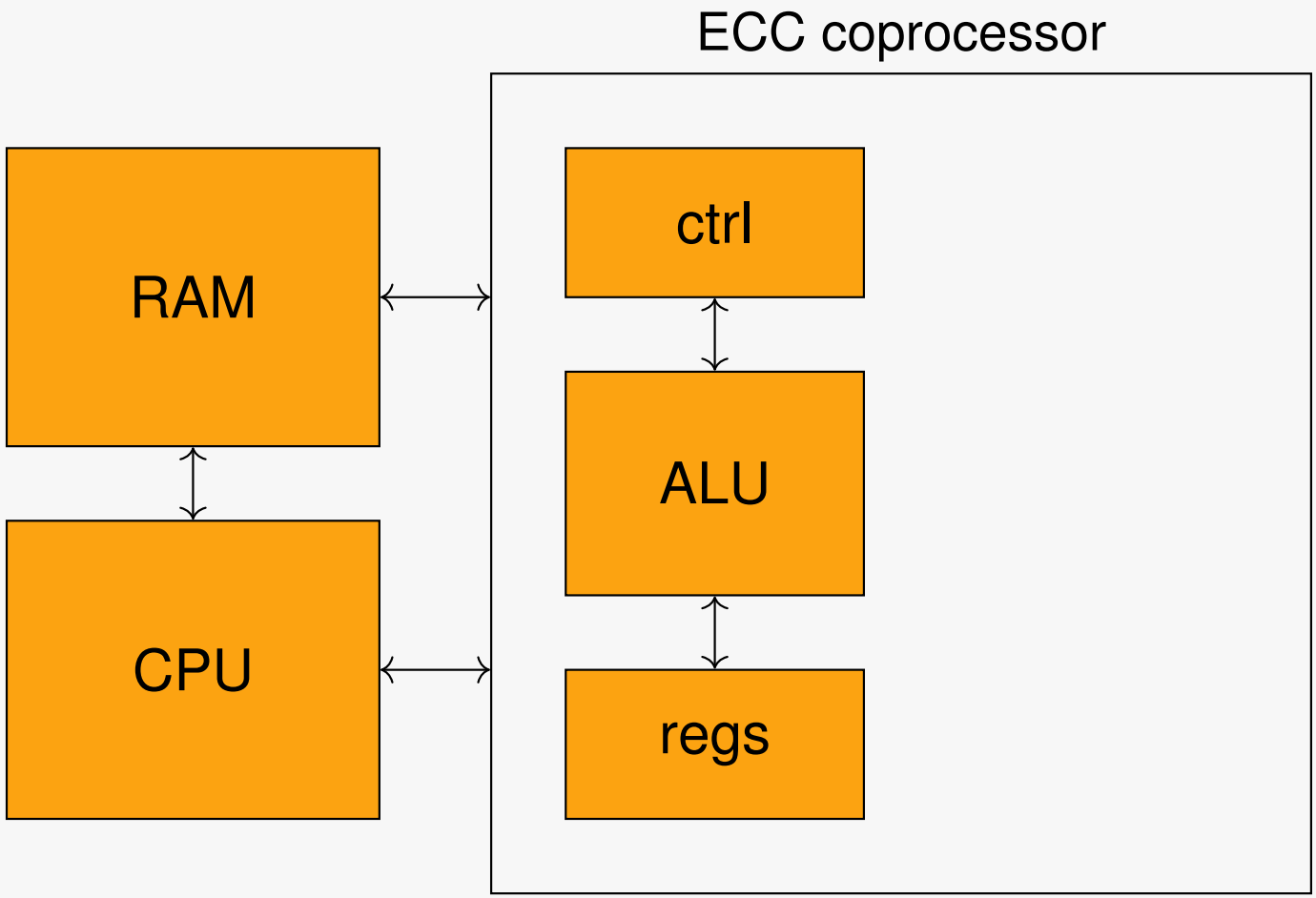
LIGHTWEIGHT ECC

- **Optimization goal:** Minimize the circuit area (or power)
- Stripped down microcontroller that contains only what is needed to implement field arithmetic
- Small datapath width (8-bit or 16-bit)
- Memory/registers and control logic dominate
- Usually the simplest algorithms are the best (e.g., Montgomery ladder)
- . . . but even rather complex algorithms have been used efficiently (e.g., zero-free representations for Koblitz curves)

LIGHTWEIGHT ECC



LIGHTWEIGHT ECC



GENERAL DISCUSSION ON BENCHMARKING ECC HW

DIFFICULTY OF BENCHMARKING

- **Different Curves**
  How to compare results obtained for different curves? For example: Curve25519 vs. NIST K-283
- **Different Platforms**
  How to compare implementations on different platforms? For example: ASIC vs. FPGA, Xilinx vs. Intel (Altera), Spartan-3 vs. Virtex-5, Virtex-4 vs. Virtex-7
- **Different Design Decisions**
  How to compare similar designs with slightly different design decisions or optimization goals? For example: LUTs vs. DSPs vs. BRAMs or support for one or many curves, proof-of-concept of a research idea vs. complete design

DIFFERENT FPGAs

- **Virtex-≤ 4**
  Slice contains two 4-to-1-bit LUTs and two flip-flops
- **Virtex-≥ 5**
  Slice contains four 6/5-to-1/2-bit LUTs and four flip-flops (eight in Virtex-6/7)
- In newer families slices can be configured also as a RAM or shift registers
- There is no objective way to compare slice counts or performance values between different FPGAs generations

BLOCKRAMS vs. REGISTERS

BlockRAMs:

- Plenty of memory available without using logic resources
- Limited reads/writes in clock cycle
- Limited width often leads to waste of memory resources

Registers:

- Allows fast parallel access
- Implementing registers using flip-flops of a logic block (slice) uses also the attached logic
- More straightforward mapping to ASIC

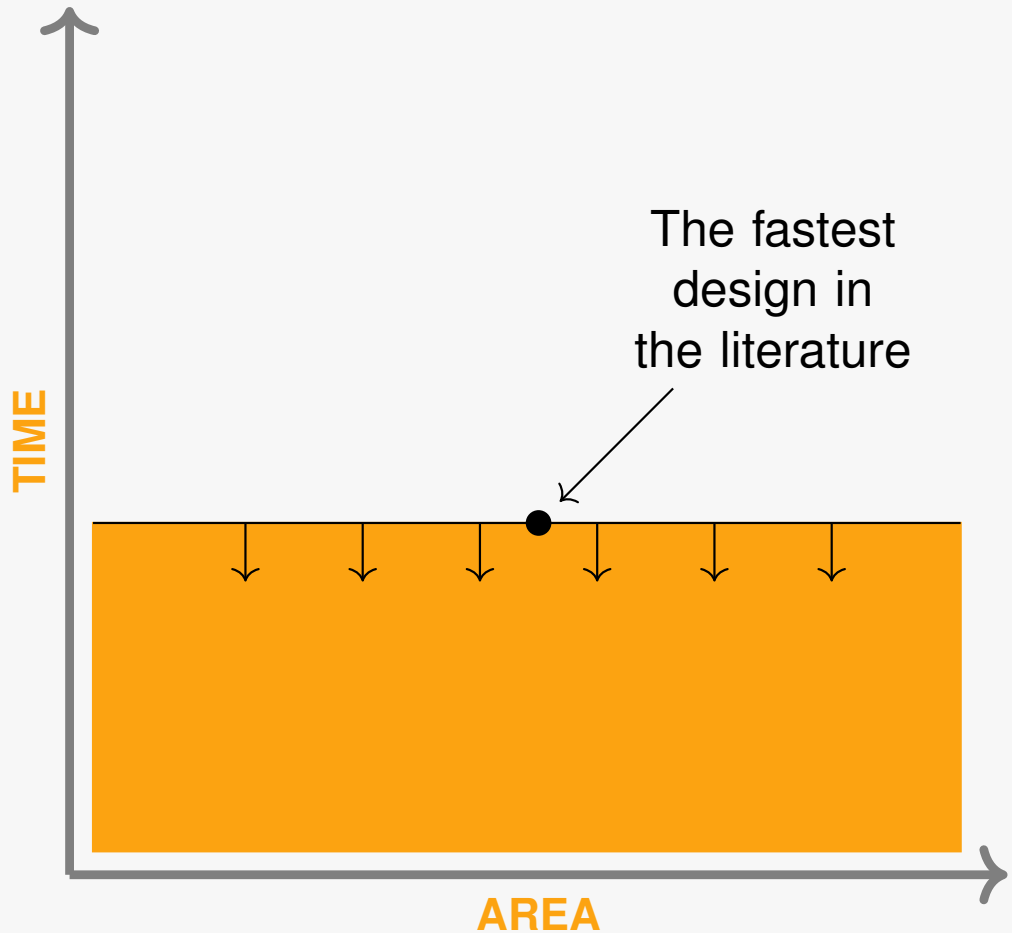
How to fairly compare BlockRAMs and registers?

HOW TO VALUE SECURITY?

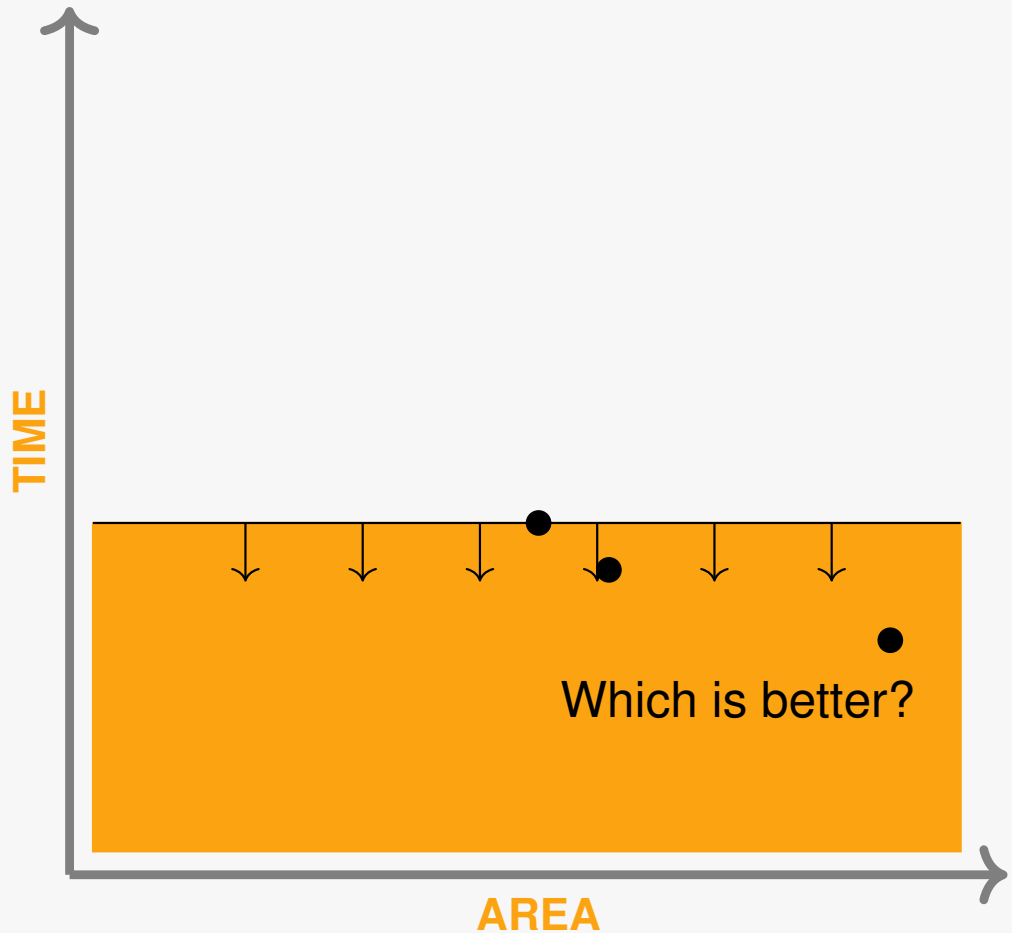
- It is hard to design fast or low-resource ECC...
- ...but it is much harder to do it by implementing countermeasures against side-channel attacks
- Constant time is required by most applications (unfortunately not many implementations are constant time...)
- SPA protection, for example, via Montgomery ladder
- DPA countermeasures are not necessarily needed if k is a nonce (e.g., ECDSA)

How to fairly compare designs with different side-channel countermeasures?

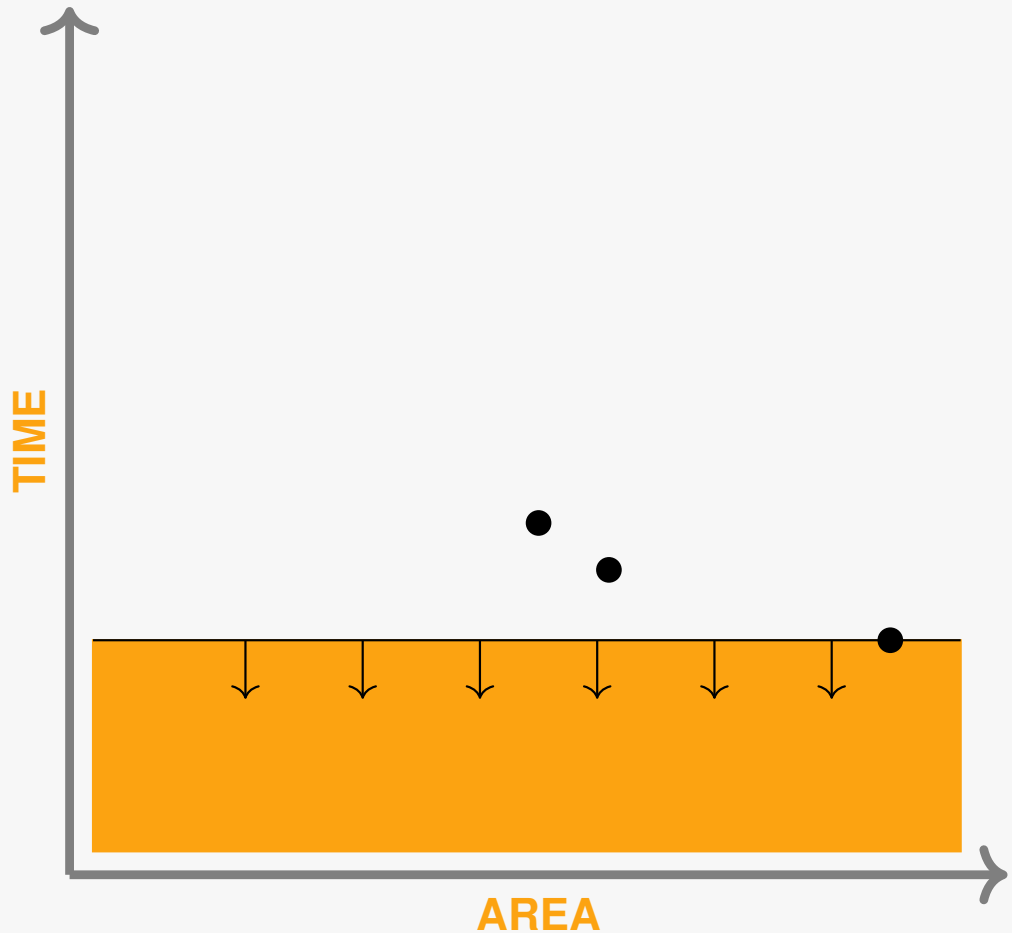
ACADEMIC BENCHMARKING
. . . and Its Problems



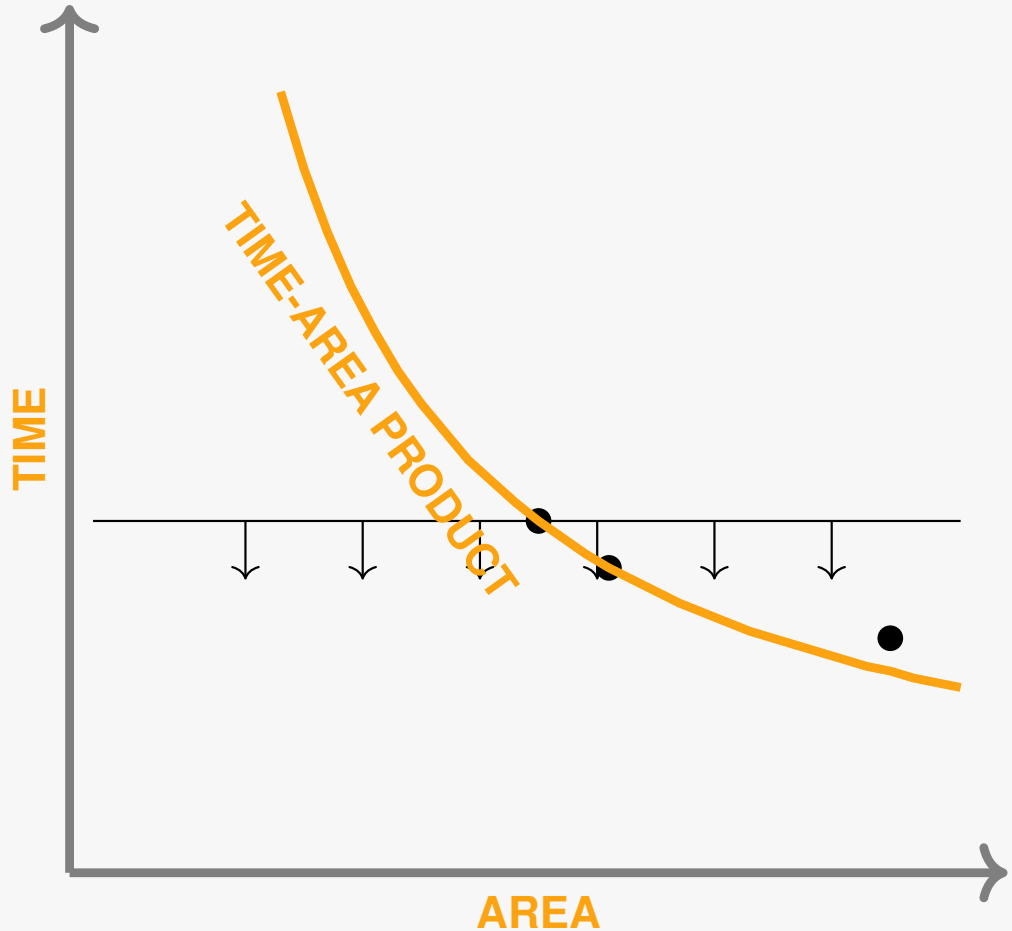
ACADEMIC BENCHMARKING
. . . and Its Problems



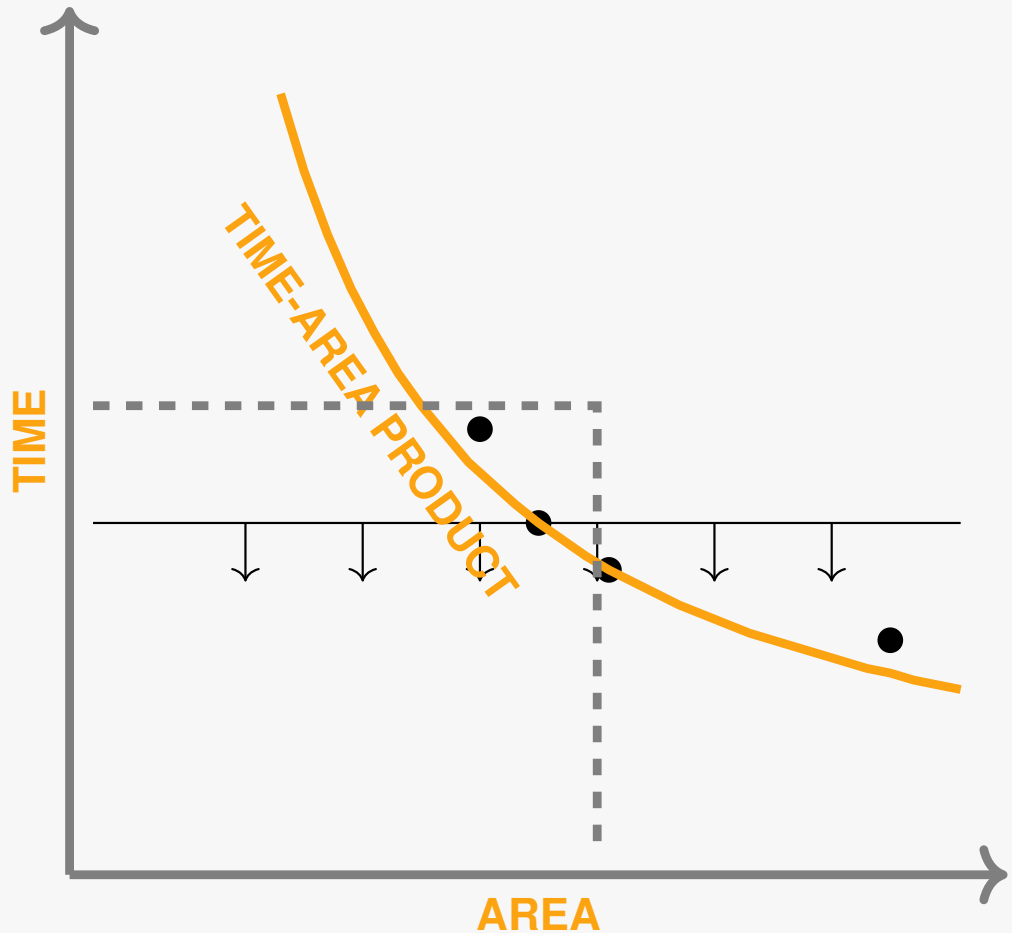
ACADEMIC BENCHMARKING
. . . and Its Problems



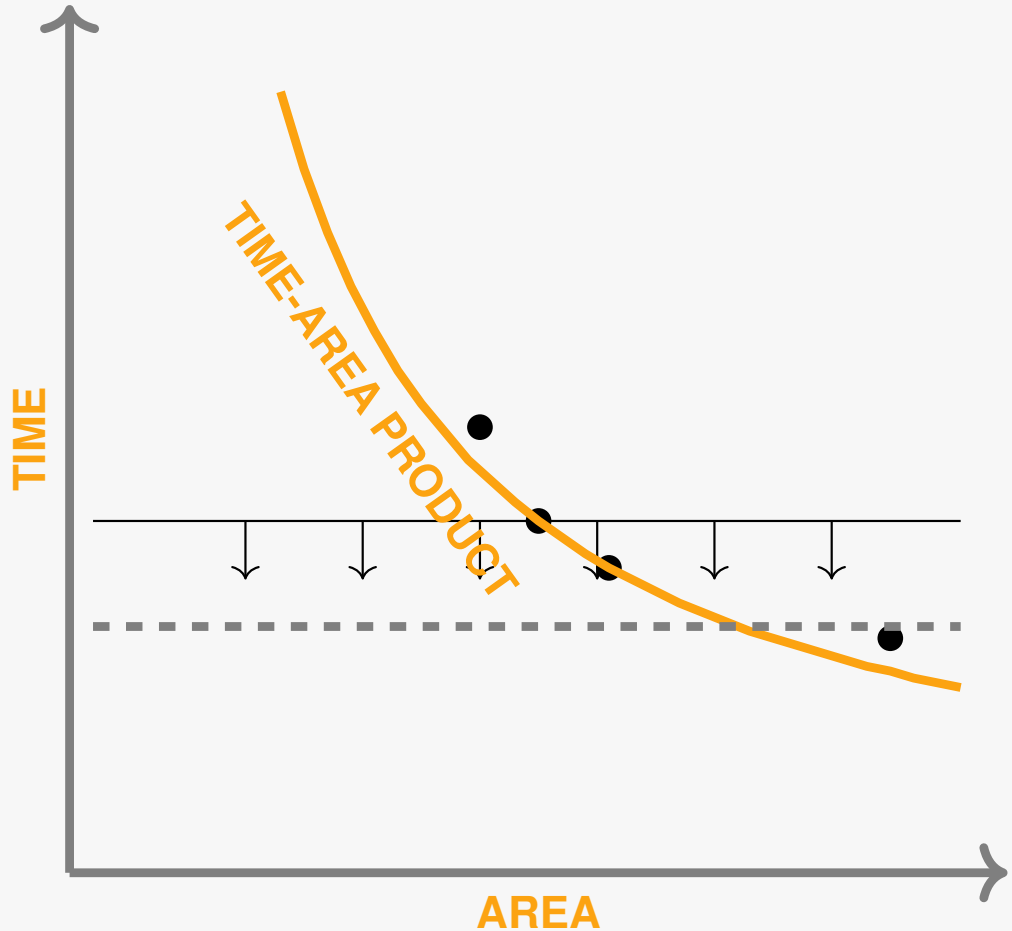
ACADEMIC BENCHMARKING
. . . and Its Problems



ACADEMIC BENCHMARKING
. . . and Its Problems



ACADEMIC BENCHMARKING
. . . and Its Problems



BENCHMARKING ECC IMPLEMENTATIONS

IMPLEMENTATIONS
Prime Curves

Work	Curve	FPGA	Resources	μs
[Gün08]	NIST-P256	Virtex-4	1715 + 32 DSP + 11 BRAMs	495
[Jär16]	Four$\mathbb{Q}$	Zynq-7020	1691 + 27 DSP + 10 BRAM	157
[Kop16]	Curve25519	Zynq-7030	8639 + 260 DSP	118
[Loi15]	NIST-P256	Virtex-5	1980 + 7 DSP + 2 BRAM	3951
[Ma13]	Any (P256)	Virtex-5	1725 + 37 DSP + 10 BRAM	376
[Roy14]	NIST-P256	Virtex-5	4505 + 16 DSP	570
[Sas14]	Curve25519	Zynq-7020	1029 + 20 DSP + 2 BRAM	397

IMPLEMENTATIONS
Binary Curves

Work	Curve	FPGA	Resources	μs	ops
[Aza12]	GLS-254	Virtex-4	12043	16.85	59300
[Aza14a]	Edw.-233	Virtex-4	29252	36.3	27500
[Göv16]	GLS-254	Virtex-5	1552	223	4500
[Göv16]	GLS-254	Virtex-4	3985	317	3200
[Jär11]	NIST-K163	Stratix II	14280 + 25M4K	11.71	235600
[Loi13]	NIST-K233	Virtex-4	2431	603	1700
[Sin15]	NIST-B163	Virtex-5	3513	9.5	105300
[Sut13]	NIST-B233	Virtex-5	6487	19.89	50300

. . . and many, many, many more.

IMPLEMENTATIONS
Low Resources

Work	Curve	Platform	GE	Clocks	*ms*
[Aza14b]	K-163	CMOS-65	11,571	106,700	8
[Bat06]	B-163	CMOS-130	9,926	95,159	190
[Boc08]	B-163	CMOS-220	12,876	–	95
[Hei09]	B-163	CMOS-180	13,250	296,299	279
[Kum06]	B-163	CMOS-350	16,207	376,864	28
[Lee08]	B-163	CMOS-130	12,506	275,816	244
[Pes14]	P-160	CMOS-130	12,448	139,930	140
[Sin15]	K-283	CMOS-130	4,323	1,566,000	98
[Wen11]	B-163	CMOS-130	8,958	286,000	2860
[Wen13]	B-163	CMOS-130	4,114	467,370	467

OTHER PUBLIC-KEY CRYPTOSYSTEMS

Work	Cryptosyst.	FPGA	Resources	μs
[Suz07]	RSA-1024 (80)	Virtex-4	3937 + 17 DSP	1710
[Suz07]	RSA-2048 (112)	Virtex-4	3937 + 17 DSP	12600
[Koz17]	SIDH-512 (128)	Virtex-7	5298 + 64 DSP + 33 BRAM	45000
[Sin14]	RLWE (128)	Virtex-6	1349 + 1 DSP + 2 BRAM	20.1
[Sas14]	Curve25519	Zynq	1029 + 20 DSP + 2 BRAM	397

WHERE ARE WE NOW?

- **Low Latency:**
  - 118 μs on Curve25519 in Zynq-7030 by Koppermann et al.
  - 157 μs on Four$\mathbb{Q}$ in Zynq-7020 by Järvinen et al.
  - Around 10 μs on binary curves (163/233) by many authors
- **High Throughput:**
  - 64730 mults/s on Four$\mathbb{Q}$ in Zynq-7020 by Järvinen et al.,
  - 32304 mults/s on Curve25519 in Zynq-7020 by Sasdrich & Güneysu
  - Several hundreds of thousands on binary curves (even 1,700,000 mults/s on K-163 in Stratix IV in 2011)
- **Low Resources:**
  - Full ECC protocols (ECDSA on P-160) including hash and memory with 12,448 GE by Pessl & Hutter
  - Without memory, only 4,323 GE for K-283 by Sinha Roy et al.

CONCLUSION

- Fair comparison of ECC HW implementations is difficult because there are so many variables
- Publishing source codes would make fair benchmarking easier, but
  (a) code is often written for specific platform (e.g., FPGA)
  (b) the difficulties of different optimization goals, features, etc. still prevail
- Fix as many variables (FPGA family, device, optimization goals, etc.) as possible to have a fair comparison

THANK YOU!
QUESTIONS?

REFERENCES

Aza12 Azarderakhsh, R., Karabina, K.: A new double point multiplication method and its implementation on binary elliptic curves with endomorphisms. Technical report CACR 2012–24, University of Waterloo, Centre for Applied Cryptographic Research (2012)

Aza14a Azarderakhsh, R., Reyhani-Masoleh, A.: Parallel and High-Speed Computations of Elliptic Curve Cryptography Using Hybrid-Double Multipliers, IEEE Transactions on Parallel and Distributed Systems (Volume: 26, Issue: 6, June 1 2015)

Aza14b Azarderakhsh, R., Järvinen, K.U., Mozaffari-Kermani, M.: Efficient algorithm and architecture for elliptic curve cryptography for extremely constrained secure applications. IEEE Trans. Circ. Syst. I–Regul. Pap. 61(4), 1144–1155 (2014)

Bat06 Batina, L., Mentens, N., Sakiyama, K., Preneel, B., Verbauwhede, I.: Low-cost elliptic curve cryptography for wireless sensor networks. In: Buttyan, L., Gligor, V.D., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 6–17. Springer, Heidelberg (2006)

Boc08 Bock, H., Braun, M., Dichtl, M., Hess, E., Heyszl, J., Kargl, W., Koroschetz, H., Meyer, B., Seuschek, H.: A milestone towards RFID products offering asymmetric authentication based on elliptic curve cryptography. In: Proceedings of the 4th Workshop on RFID Security — RFIDSec 2008 (2008)

REFERENCES

Gün08 Güneysu, T., Paar, C.: Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 62–78. Springer, Heidelberg (2008)

Göv16 Gövem B., Järvinen K., Aerts K., Verbauwhede I., Mentens N. (2016) A Fast and Compact FPGA Implementation of Elliptic Curve Cryptography Using Lambda Coordinates. In: Pointcheval D., Nitaj A., Rachidi T. (eds) Progress in Cryptology – AFRICACRYPT 2016. AFRICACRYPT 2016. Lecture Notes in Computer Science, vol 9646. Springer, Cham

Hei09 Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID – a proof in silicon. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)

Jär16 Järvinen K., Miele A., Azarderakhsh R., Longa P. (2016) Four$\mathbb{Q}$ on FPGA: New Hardware Speed Records for Elliptic Curve Cryptography over Large Prime Characteristic Fields. In: Gierlichs B., Poschmann A. (eds) Cryptographic Hardware and Embedded Systems – CHES 2016. CHES 2016. Lecture Notes in Computer Science, vol 9813. Springer, Berlin, Heidelberg

REFERENCES

Koz17 Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M., Jao, D.: Post-Quantum Cryptography on FPGA Based on Isogenies on Elliptic Curves, IEEE Trans. Circuits and Syst. I 64(1), 86-99, 2017.

Lee08 Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic-curve-based security processor for RFID. IEEE Trans. Comput. 57(11), 1514–1527 (2008)

Loi13 Loi, K.C., Ko, S.B.: High performance scalable elliptic curve cryptosystem processor for Koblitz curves. Microprocess. Microsyst. 37(4), 394–406 (2013)

Loi15 Loi, K.C.C., Ko, S.B.: Scalable elliptic curve cryptosystem FPGA processor for NIST prime curves. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 23(11), 2753–2756 (2015)

Ma13 Ma, Y., Liu, Z., Pan, W., Jing, J.: A high-speed elliptic curve cryptographic processor for generic curves over GF(p). In: Lange, T., Lauter, K., Lisonek, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 421–437. Springer, Heidelberg (2014)

Pes14 Pessl, P., Hutter, M.: Curved tags — a low-resource ECDSA implementation tailored for RFID. In: Sadeghi, A.-R., Saxena, N. (eds.) RFIDSec 2014. LNCS, vol. 8651, pp. 156–172. Springer, Heidelberg (2014)

REFERENCES

Roy14 Roy, D.B., Mukhopadhyay, D., Izumi, M., Takahashi, J.: Tile before multiplication: an efficient strategy to optimize DSP multiplier for accelerating prime field ECC for NIST curves. In: Proceedings of the 51st Annual Design Automation Conference–DAC 2014, pp. 177: 1–177: 6. ACM (2014)

Sas14 Pascal Sasdrich, Tim Güneysu: Efficient Elliptic-Curve Cryptography Using Curve25519 on Reconfigurable Devices, ARC 2014

Sin14 Sinha Roy, S., Vercauteren, F., Mentens, N., Chen, D.D., Verbauwhede, I.: Compact Ring-LWE Cryptoprocessor, CHES 2014, LNCS 9731, 371-391, 2014.

Sin15 Sinha Roy, S., Rebeiro, C., Mukhopadhyay, D.: Theoretical modeling of elliptic curve scalar multiplier on LUT-based FPGAs for area and speed. IEEE Trans. VLSI Syst. 21(5), 901–909 (2013)

Sut13 Sutter, G.D., Deschamps, J., Imana, J.L.: Efficient elliptic curve point multiplication using digit-serial binary field operations. IEEE Trans. Industr. Electron. 60(1), 217–225 (2013)

Suz07 Suzuki, D.: How to Maximize the Potential of FPGA Resources for Modular Exponentiation. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 272–288. Springer, Heidelberg (2007)

REFERENCES

Wen11 Wenger, E., Hutter, M.: A hardware processor supporting elliptic curve cryptography for less than 9 kGEs. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 182–198. Springer, Heidelberg (2011)

Wen13 Wenger, E.: Hardware architectures for MSP430-based wireless sensor nodes performing elliptic curve cryptography. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 290–306. Springer, Heidelberg (2013)