Thomas Wollinger

Hardware Implementation of Hyperelliptic Curve Cryptosystems

Chair for Communication Security
Ruhr-University of Bochum
www.crypto.rub.de
wollinger@crypto.rub.de

Scuola Loretto
Incisa, Italia
www.wollinger.org
thomas@wollinger.org

Acknowledgement

Special thanks to

- HoWon Kim and
- Christof Paar

from U Bochum.

Contents

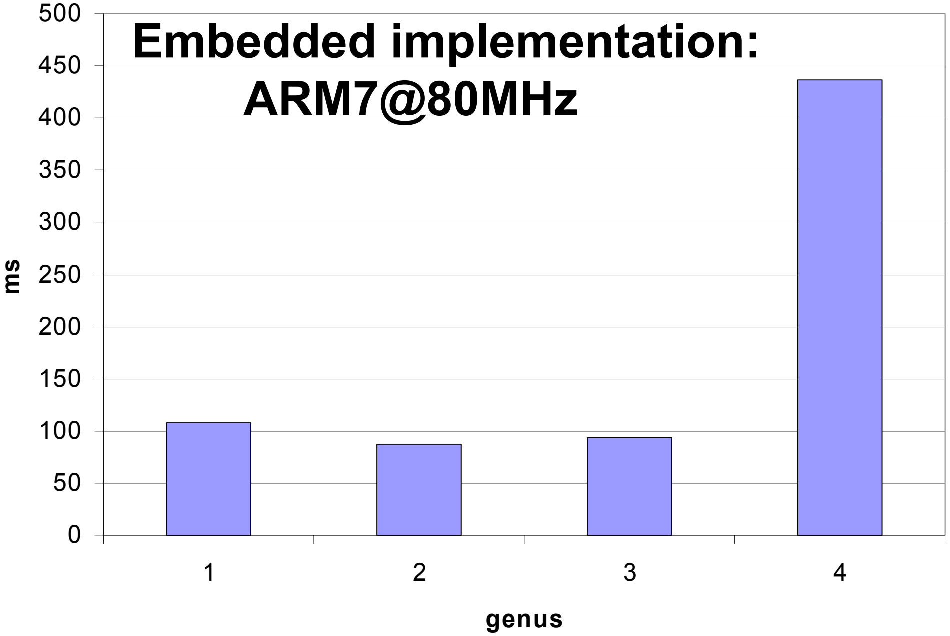
- Motivation
- Hardware: Basic Definitions (nutshell)
- Hyperelliptic Curve Cryptosystems (for engineers)
- HECC on FPGA
- Conclusion

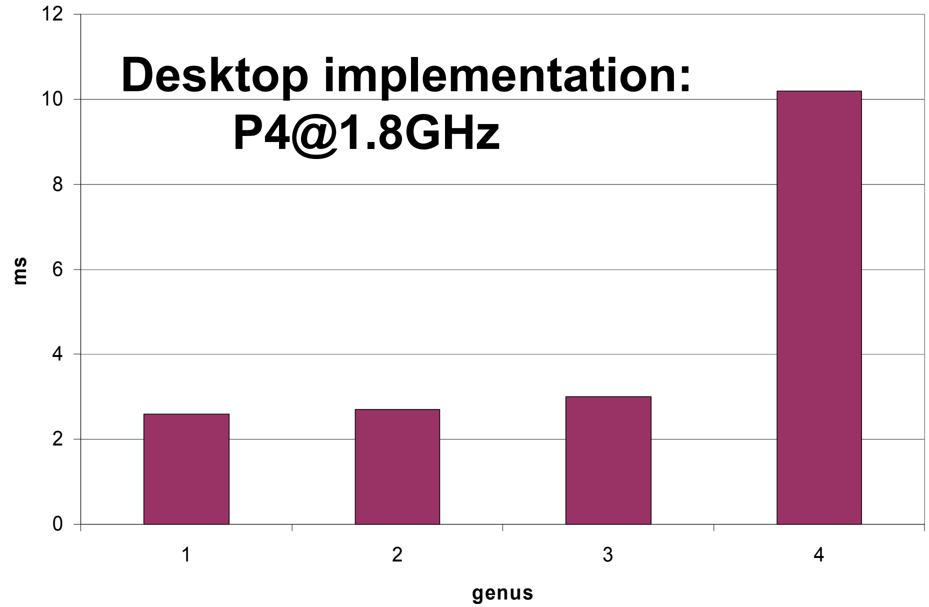
Motivation I

Can HECC Beat ECC in Practice?

HECC in Software

Presentation von C. Paar ECC 2003 (characteristic two):

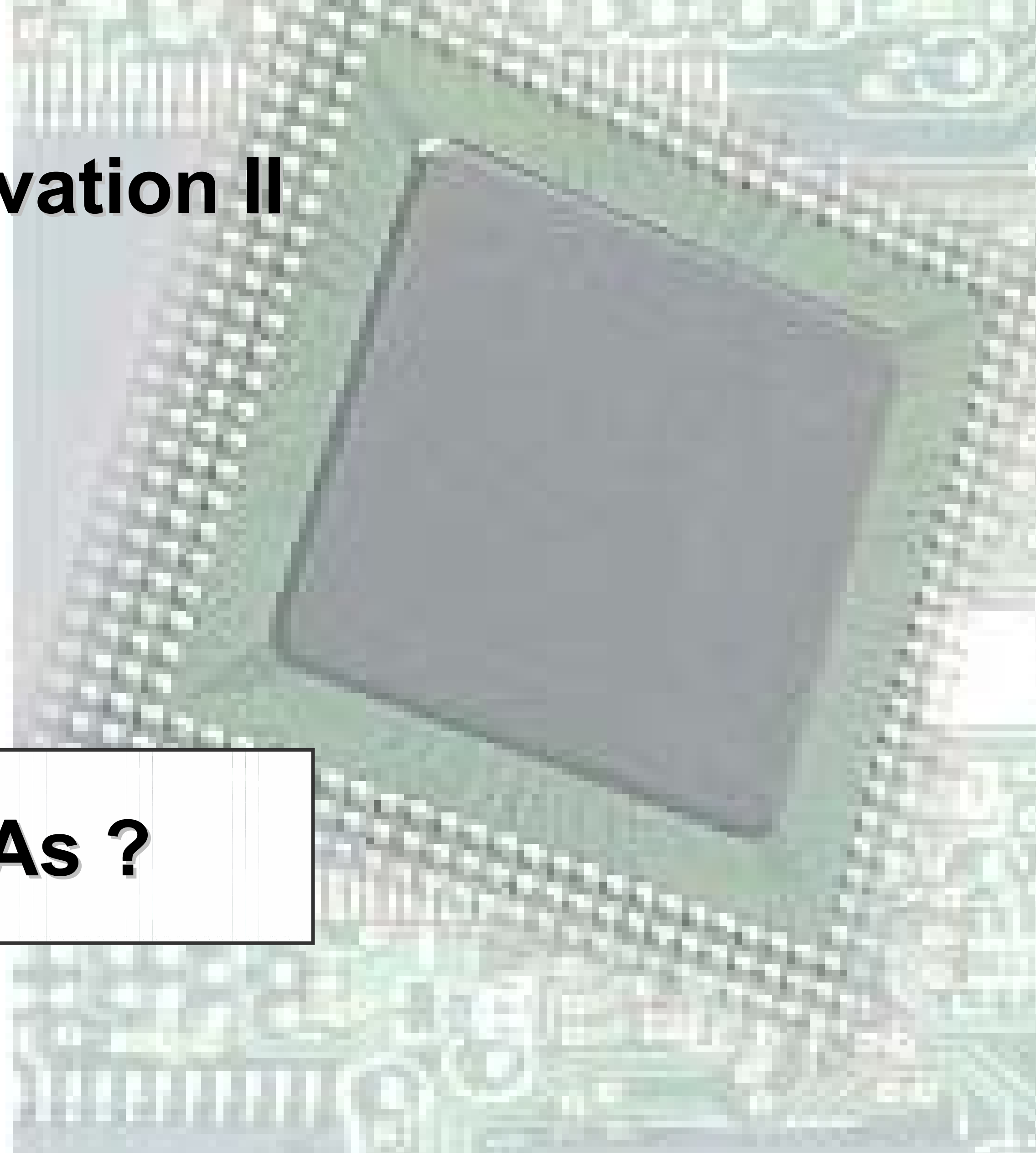




⇒ **SW: ECC, genus-2, and genus-3 HECC have similar performance**

⇒ **How about hardware implemenation?**

Motivation II



Why FPGAs ?

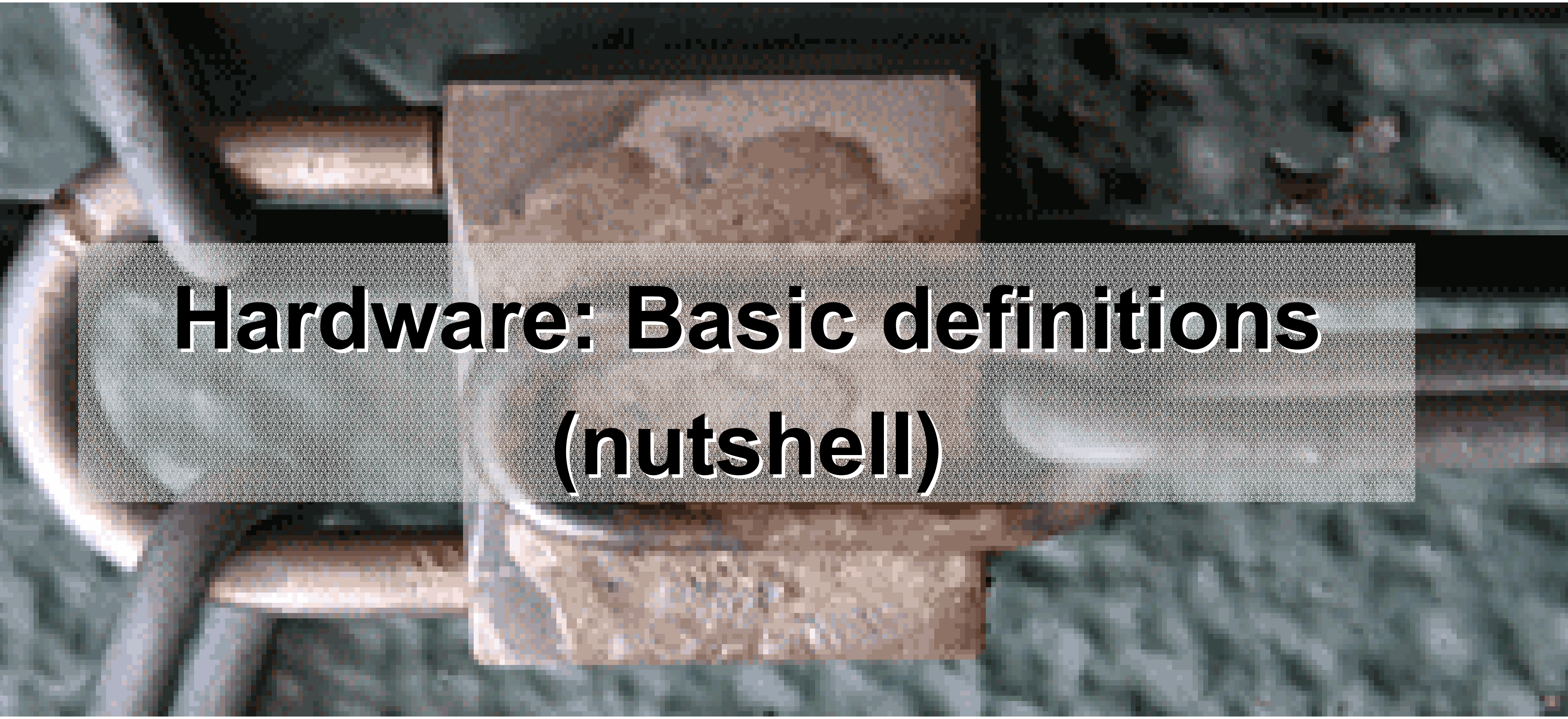
FPGA and Crypto

- **Algorithm agility:** easy exchange of algorithms on the fly (protocols need different algorithms)
- **Algorithm upload:** upload new algorithm (new standard)
- **Architecture efficiency:** allows architectures optimized for specific algorithm (fixed finite field and group formulae)
- **Resource efficiency:** runtime configuration allows exchange of algorithms (private/public key)
- **Algorithm modification:** easily possible to create customized algorithms (use different group formulae)
- **Throughput:** much faster than SW, almost as good as ASICs
- **Cost efficiency relative to ASIC**: especially if there are only a small number of chips needed and development cost are low

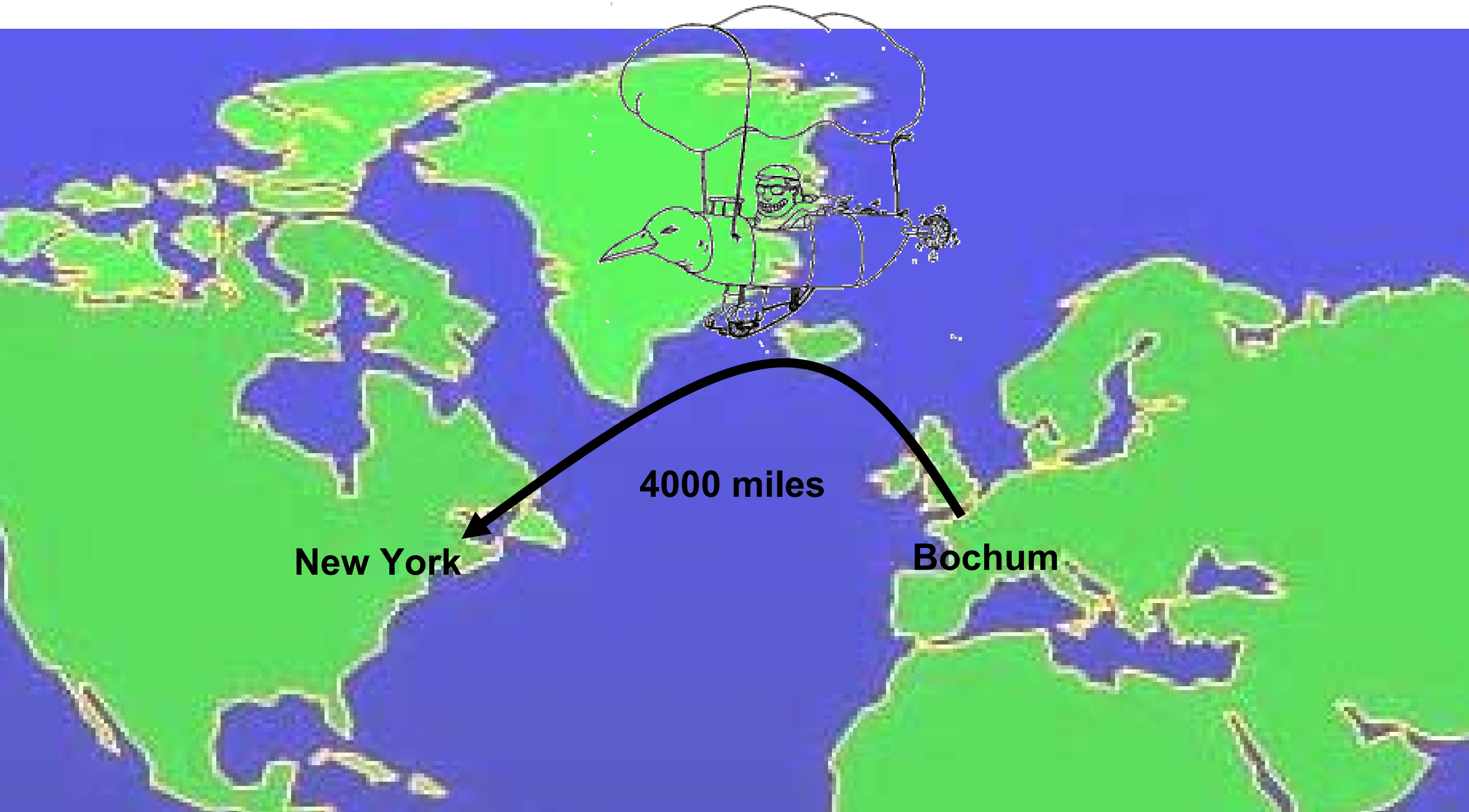
Motivation III

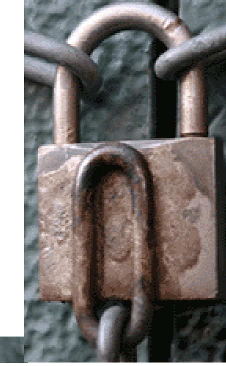
Personal Reasons ?

Finish my PhD ☺


Hardware: Basic definitions
(nutshell)

Throughput vs. Latency





Throughput vs. Latency

Airplane	Passengers	Speed (mph)
Concorde	132	1350
Boeing 747	470	610

- How much faster is the Concorde? **2.2**
- How much larger is the 747's capacity? **3.6**
- What is the throughput for the Concorde in passengers/hr?

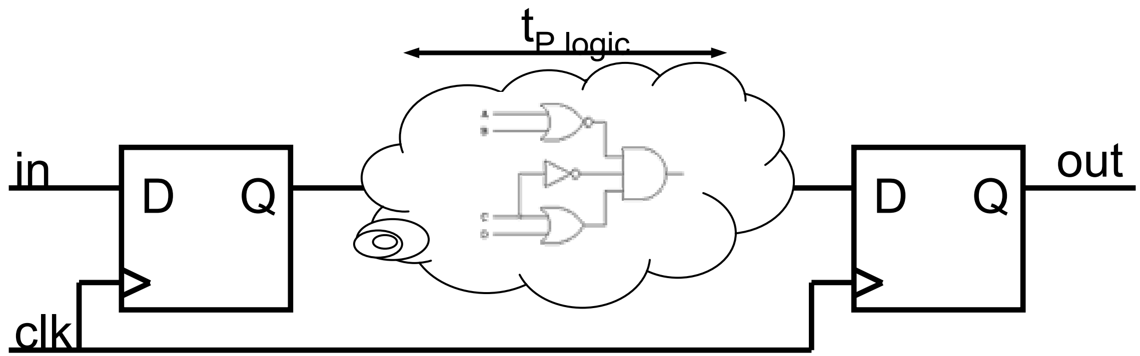
132x1350/4000 = 44.6

- How much larger is the throughput of the 747? **1.6**
- What is the latency of the Concorde [747]?

3 hours [6.5 hours]

Critical Path vs. Clock Frequency

- Critical Path – The Longest Path From Outputs of Registers to Inputs of Registers



$$t_{Critical} = t_{P\ FF} + t_{P\ logic} + t_{S\ FF}$$

- Min. Clock Period = Length of The Critical Path
- Max. Clock Frequency = 1 / Min. Clock Period

How to Speed up Hardware Implementations?

- **parallelism** ↑ (latency ↓)
- **critical path** ↓ (clock frequency ↑)

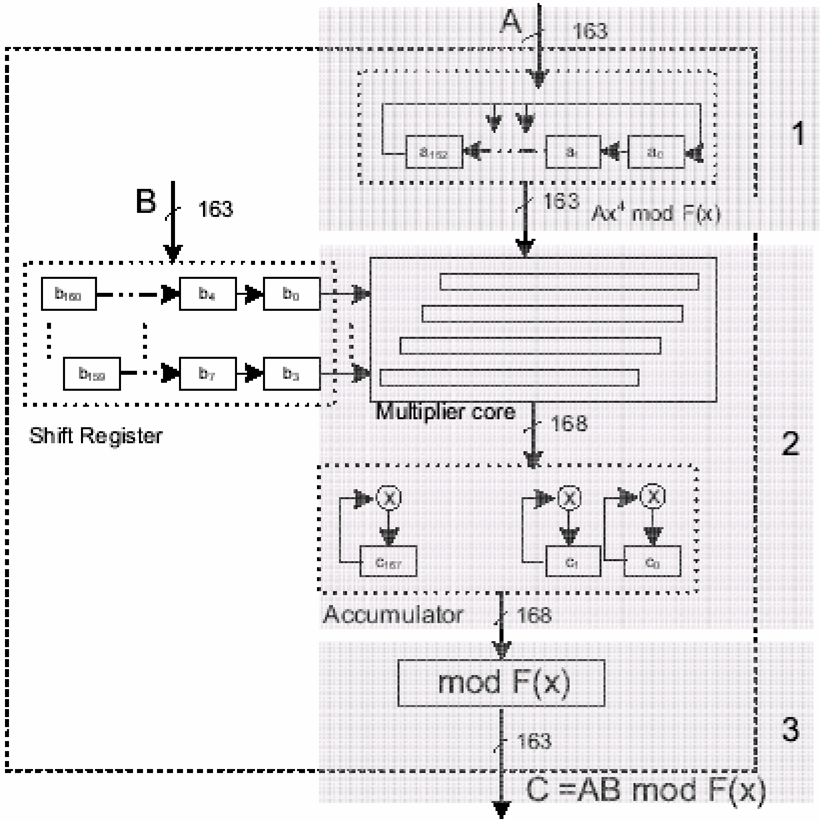
How can we increase the parallelism for the multiplier?

Digit-Serial/Parallel Multiplier

- Given: number of bits that are processed in parallel is defined to be the digit-size *D* and d = ⌈m/D⌉

$$C \equiv AB \bmod p(\alpha)$$

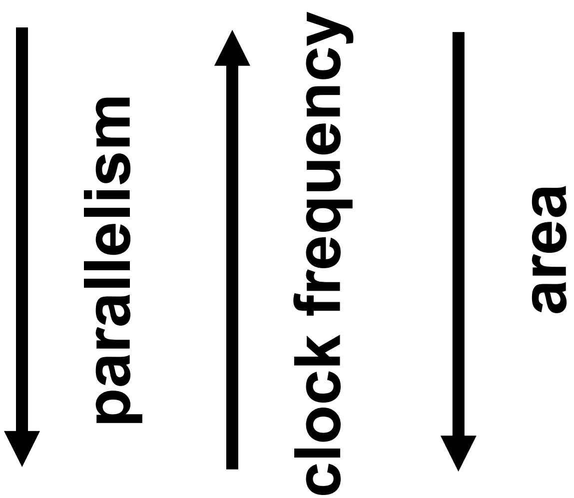
$$= A \sum_{i=0}^{d-1} B_i \alpha^{Di} \bmod p(\alpha)$$



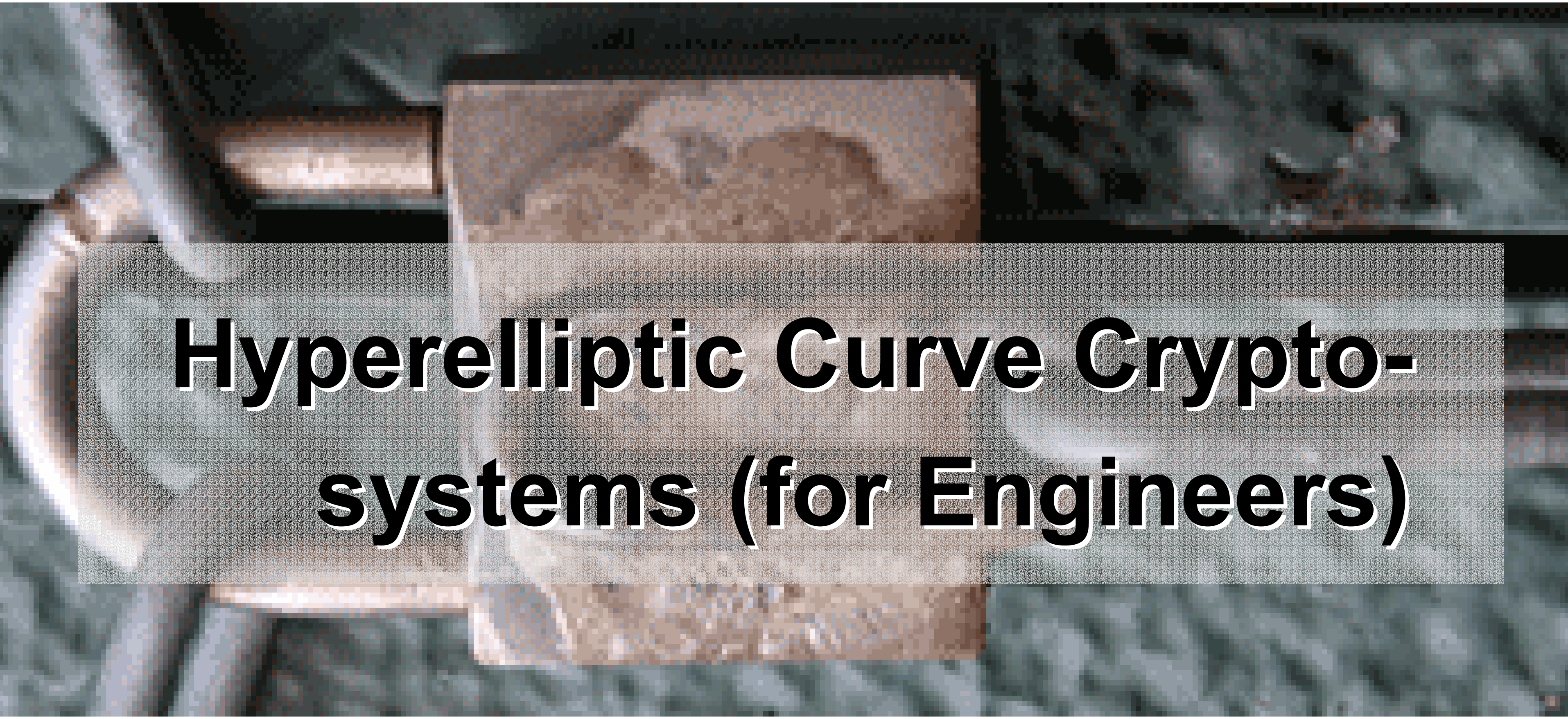
How to Speed up Hardware Implementations?

For example: $GF(2^m)$ multiplier:

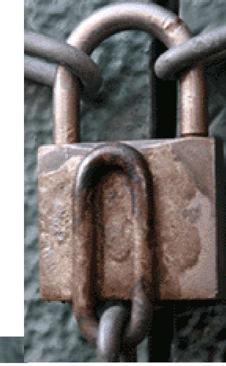
- Bit serial multiplier (D=1)
- Digit-serial/parallel multiplier (e.g. D=8)
- Parallel multiplier (D=m)



⇒ algorithmic/engineering optimum for each HW implementation



Hyperelliptic Curve Crypto-systems (for Engineers)



HEC: The Definition

A *HEC of genus g* over a finite field **F** is given by the set of solutions ***(x,y)*** ∈ F x F to the equation

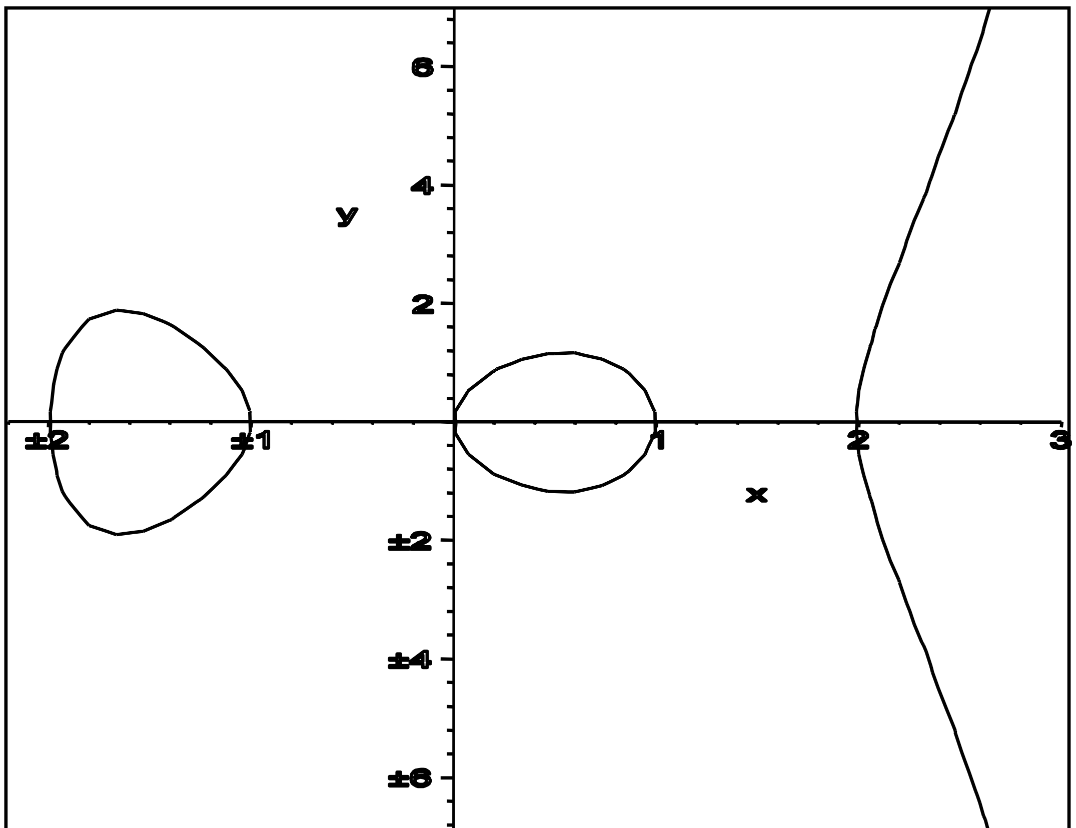
$$C: \; y^2 + h(x)\, y = f(x)$$

where

- ***g*** is the „genus“
- ***h(x)*** is a polynomial of degree **≤ *g*** over *F*
- ***f(x)*** is a monic polynomial of degree ***2g+1*** over *F*
- certain further conditions

An Example: HEC over the Reals

C: $y_2 = x^5 - 5x^3 + 4x + 3$ over **R**

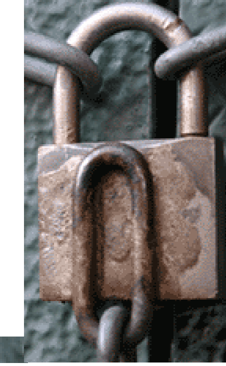


Where is the Group for the DL Problem?

1. Group elements are *not* points on the curve – unlike ECC
2. Group elements are „divisors“ (= formal sum of *g* points):

$$D = f(P_1, \ldots, P_g) = \sum_{i=1}^{g} m_{P_i} P_i$$

3. Abelian group: (reduced) divisors forms „Jacobian“ of the curve $\mathbf{J}_C(\mathbf{F}_q)$



Group cardinality

- HEC of genus g over F_q
- The cardinality of $J_C(F_q)$ is given by Hasse-Weil:

$$\left\lceil (\sqrt{q}-1)^{2g} \right\rceil \leq \left| J_C(F_q) \right| \leq \left\lfloor (\sqrt{q}+1)^{2g} \right\rfloor$$

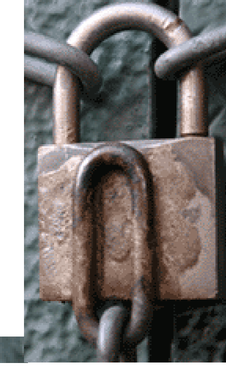
- Major implication: **group size ≈ (field size)^g**
- Don‘t choose genus ≥ 3 (4) because of attacks [Frey/Rück, Gaudry, Theriault, …]

Example: Group size vs. field size

Ex. group size = 2^{160}

- ECC (g=1): field size = 160 bit
- HECC (g=2): field size = 80 bit
- HECC (g=3): field size = 56 bit
- HECC (g=4): field size = 52 bit

Small element size important for HW ⇒ internal buses equal to the field size



HECC: So, Where is the Catch?

- Trade-off: „group operation“ becomes much more complex as genus increases

Genus	*# inv.*	*# mult. & sq. per add/doub*	*rel. field size*[3)]
1[1)] (ECC)	-	16	1
2[2)]	1	24/15	0.5
	-	50/37	
3[2)]	1	71/25	0.35
4[2)]	2	152/85	0.32

1) ECC with projective coordinates GF(p)

2) HEC over even fields and special curve parameters [Lange 2003, Pelzl at al. 2004]

3) Same security [Theriault 2003]

How does a HECC group operation look like (remember "P+P")?

- Example: Adding divisors on HEC of genus 3

Polynomial arithmetic:

Input: $D_1 = div(a_1,b_1)$, $D_2 = div(a_2,b_2)$

Output: $D_3 = D_1 + D_2 = div(a_3,b_3)$

Composition: $d = gcd(a_1,a_2,b_1+b_2+h) = s_1a_1+s_2a_2+s_3(b_1+b_2+h)$

$a'_3 = a_1a_2/d$

$b'_3 = [s_1a_1b_2+s_2a_2b_1+s_3(b_1b_2+f)]/f \mod a'_3$

Reduction: WHILE $deg(a'_k) > g$, DO

$a'_k = f - b'_{k-1} \mod a'_k$

$b'_k = (-h-b'_{k-1}) \mod a'_k$

END WHILE

$a_3 = a'_k$

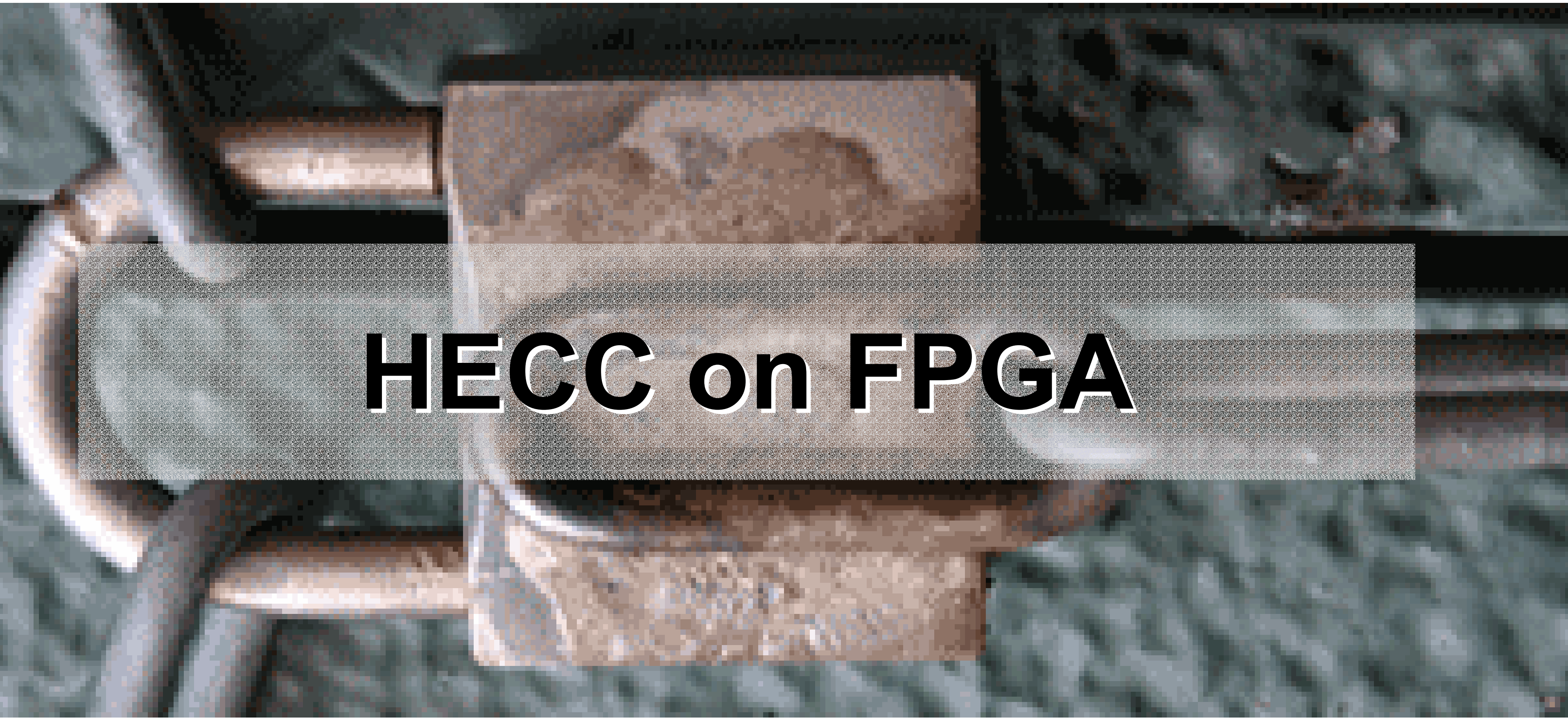
$b_3 = b'_k$

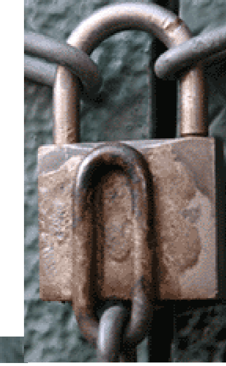
Explicit formulae (field arithmetic only):

```
t1 = a*e;
t2 = b*d;
t3 = b*f;
t4 = c*e;
t5 = a*f;
t6 = c*d;
t7 = sqr(c+f);
t8 = sqr(b+e);
t9 = (a+d)*(t3+t4);
t10= (a+d)*(t5+t6);
r =(f+c+t1+t2)*(t7+t9) + t10*(t5+t6) + t8*(t3+t4);
t11 = (b+e)*(c+f);
inv2 = (t1+t2+c+f)*(a+d)+t8;
inv1 = inv2*d + t10 + t11;
inv0 = inv2*e + d*(t10+t11) + t9 + t7;
t12 = (inv1+inv2)*(k+n+l+o);
t13 = (l+o)*inv1;
t14 = (inv0+inv2)*(k+n+m+p);
t15 = (m+p)*inv0;
t16 = (inv0+inv1)*(l+o+m+p);
t17 = (k+n)*inv2;
rs0 = t15;
rs1 = t13+t15+t16;
rs2 = t13+t14+t15+t17;
rs3 = t12+t13+t17;
rs4 = t17;
t18 = rs3+rs4*d;
s0s = rs0 + f*t18;
s1s = rs1 + rs4*f + e*t18;
s2s = rs2 + rs4*e + d*t18;
w1 = inv(r*s2s);
w2 = r*w1;
w3 = w1*sqr(s2s);
w4 = r*w2;
w5 = sqr(w4);
```

```
s0 = w2*s0s;
s1 = w2*s1s;
s2 = w2*s2s;
z0 = s0*c;
z1 = s1*c+s0*b;
z2 = s0*a+s1*b+c;
z3 = s1*a+s0+b;
z4 = a+s1;
z5 = to_GF2E(1L);
t1 = w4*h2;
t2 = w4*h3;
u3s = d + z4 + s1;
u2s = d*u3s + e + z3 + s0 + t2 + s1*z4;
u1s = d*u2s + e*u3s + f + z2 + t1 + s1*(z3+t2) + s0*z4 + w5;
u0s = d*u1s + e*u2s + f*u3s + z1 + w4*h1 + s1*(z2+t1)
                                        + s0*(z3+t2) + w5*(a+f6);
t1 = u3s+z4;
v0s = w3*(u0s*t1 + z0) + h0 + m;
v1s = w3*(u1s*t1 + u0s + z1) + h1 + l;
v2s = w3*(u2s*t1 + u1s + z2) + h2 + k;
v3s = w3*(u3s*t1 + u2s + z3) + h3;
a3 = f6 + u3s + v3s*(v3s+h3);
b3 = u2s + a3*u3s + f5 + v3s*h2 + v2s*h3;
c3 = u1s + a3*u2s + b3*u3s + f4 +
v2s*(v2s+h2) + v3s*h1 + v1s*h3;
k3 = v2s + (v3s+h3)*a3 + h2;
l3 = v1s + (v3s+h3)*b3 + h1;
m3 = v0s + (v3s+h3)*c3 + h0;
```

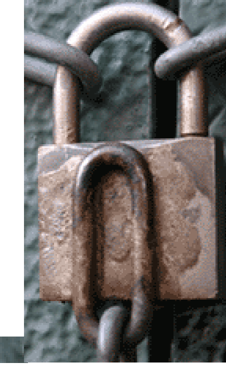
HECC on FPGA





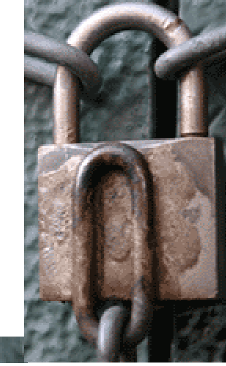
History of HECC on FPGA

- [W. 2001,W. et al. 2002]: discussion of hardware architectures for HECC
- [Boston et al. 2002]: first complete hardware implementation, scalar multiplication - 20.2ms (Cantor, genus 2 HEC, $GF(2^{113})$)
- [Clancy 2002, Clancy 2003]: extention of Boston et al., best scalar multiplication 9ms (Cantor, genus 2, $GF(2^{83})$)
- [Elias et al. 2004]: used inversion-free explicit formulae, best scalar multiplication 2,03 ms (genus 2, $GF(2^{113})$)



Our Contribution

- First FPGA implementation targeting *affine* explicit formulae.
- Comparison of affine and inversionfree HECC coprocessor.
- Investigated different field sizes.
- Examined 3 HECC coprocessor types.
- We used:
  - genus-2 curves
  - characteristic two
  - affine explicit formulae from [Lange 2003, Pelzl et al. 2004]
  - projective explicit formulae: special parameters and modified formulae from [Lange 2003]



Examine Different Multipliers

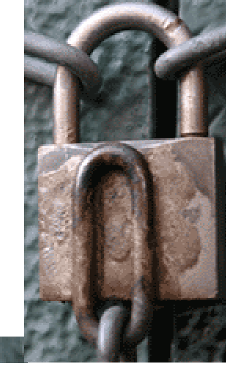
GF(2^{89}) Multiplier

Digit Size [bits]	Area [slices]	f [MHz]	Latency [ns]
1	145	97.5	913
4	239	106.8	215
8	414	110.1	109
16	645	87.4	69
32	1189	71.6	42
45	1616	63.7	31
89	3205	52.2	19

Critical path dominated by control logic (comparator logic and multiplexer)

Critical path dominated by data path

What Multiplier Should We Choose?



Possible answers:

The one with the

– Least area

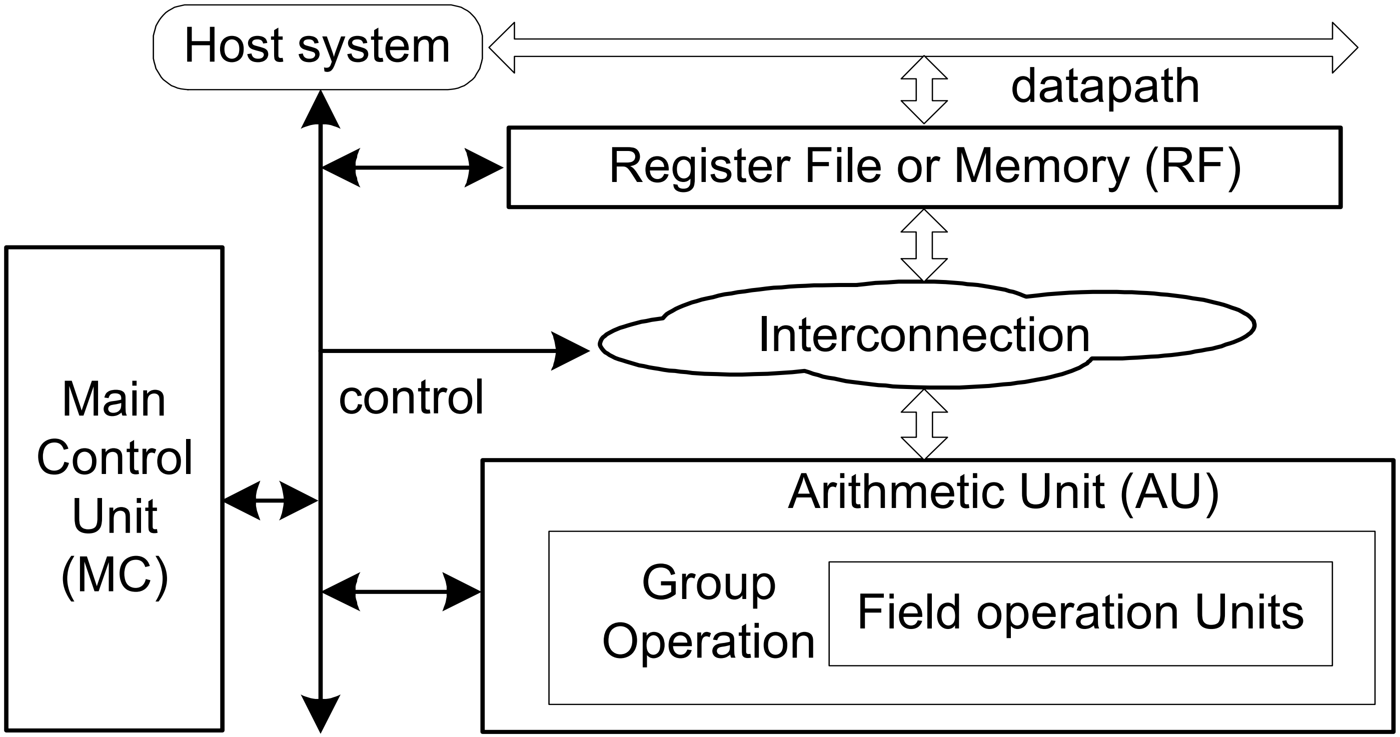
– Highest frequency

– Shortest latency

How about the frequency of the *whole* design (< 63 MHz)?

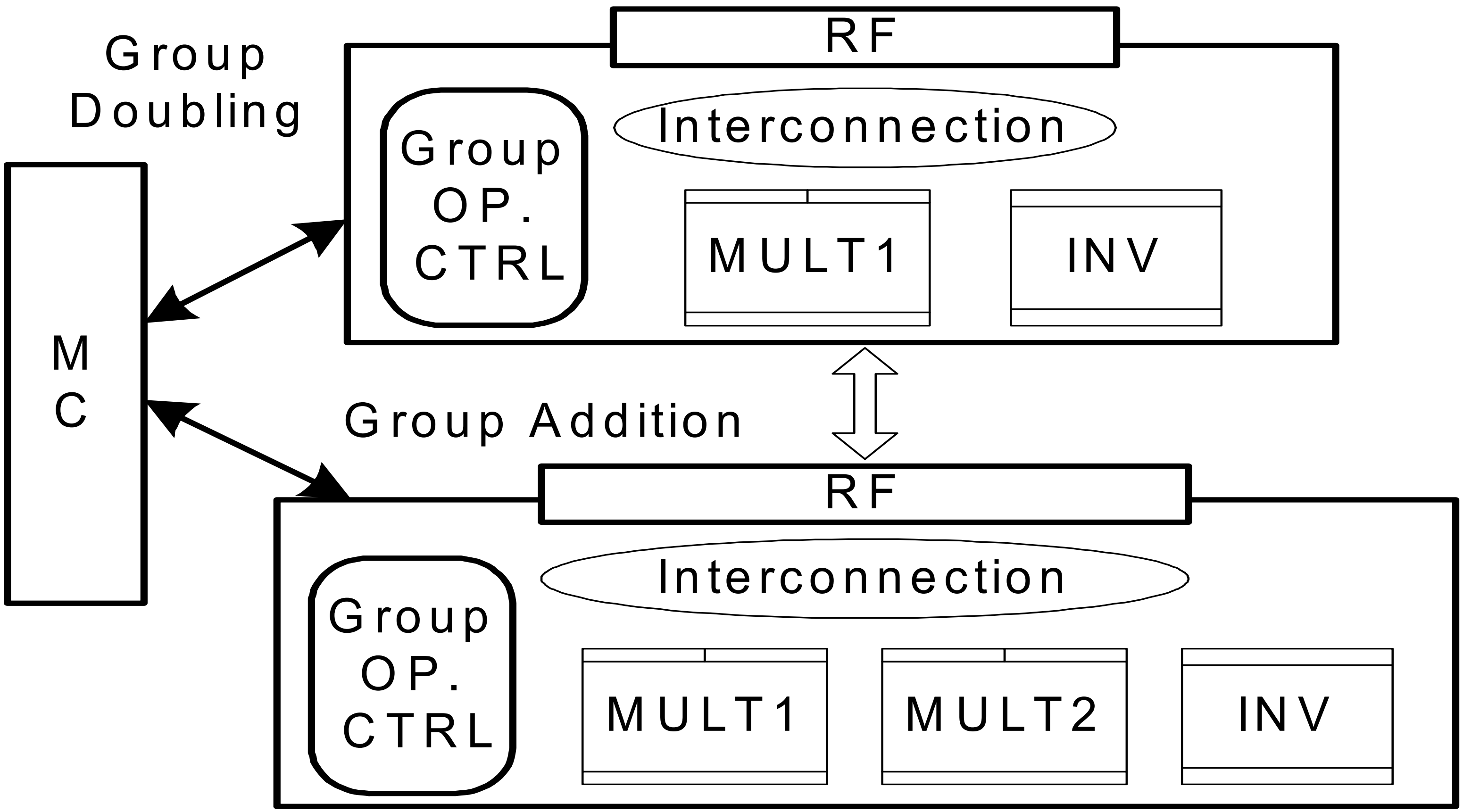
Digit Size [bits]	Area [slices]	f [MHz]	Latency [ns]
16	645	87.4	69
32	1189	71.6	42
45	1616	63.7	31

General HECC Coprocessor Architecture



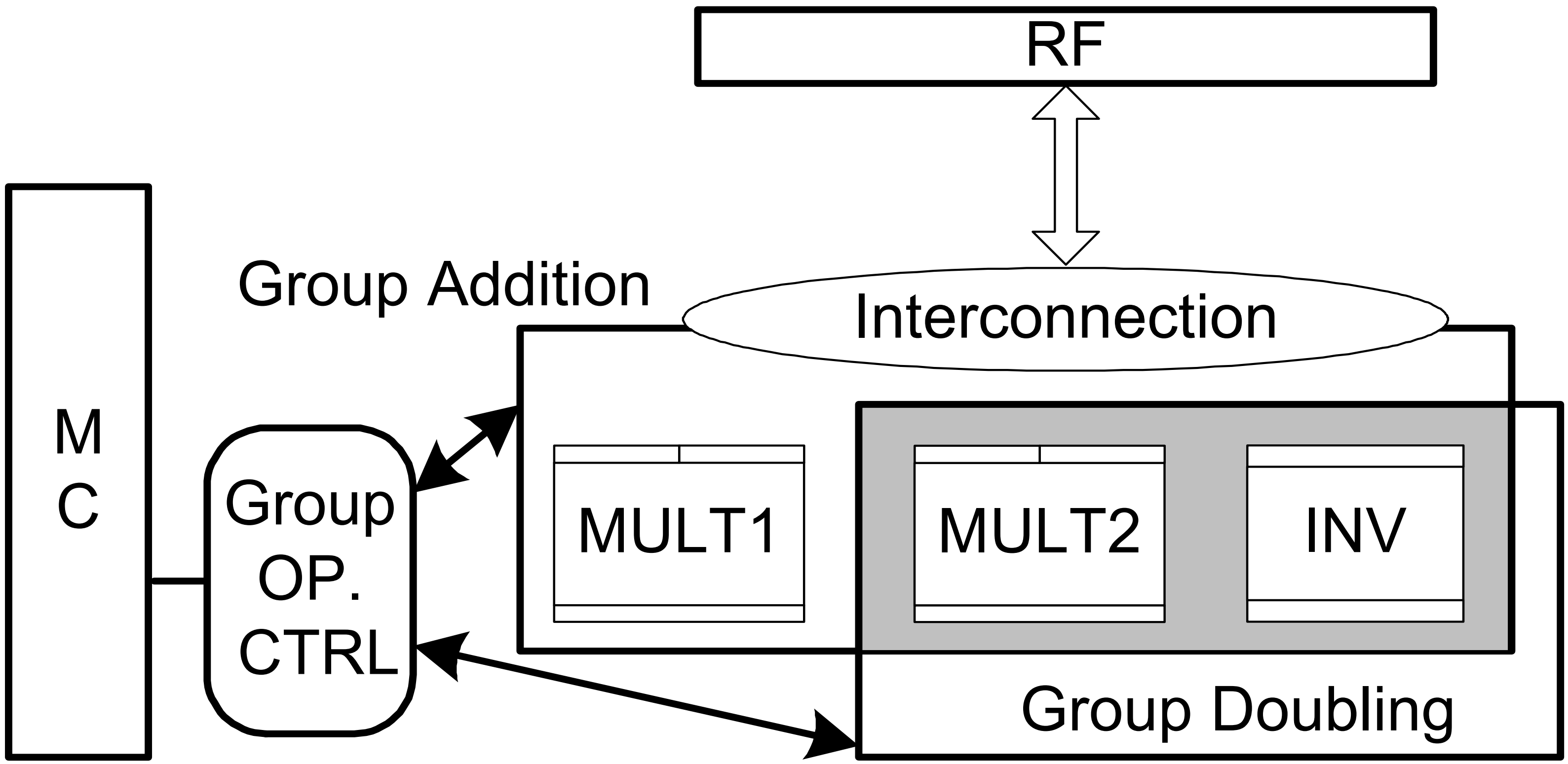
⇒ **Three design option**

Type 1 Design: High Performance



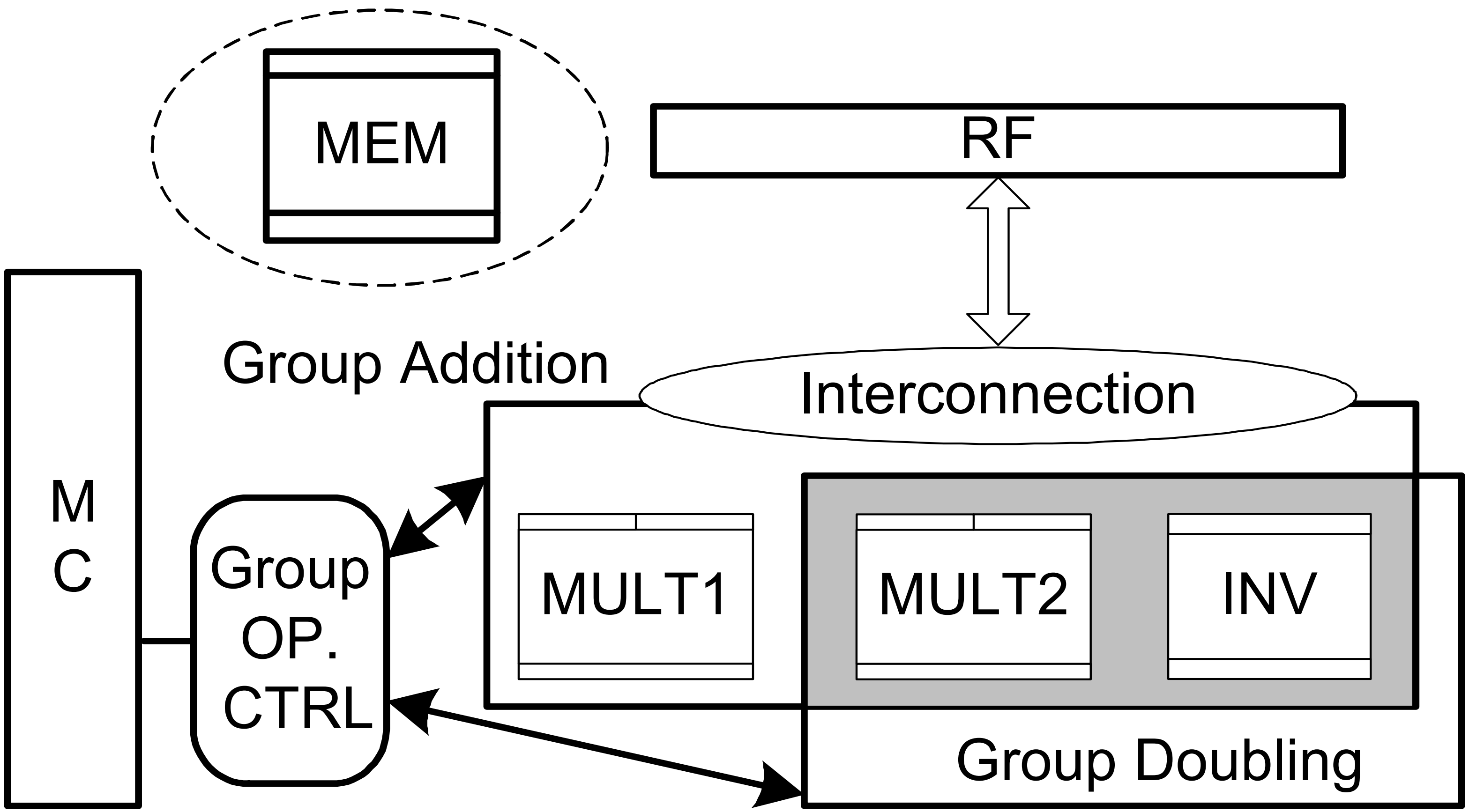
Interconnection: MUX
Scalar mult: right-to-left (parallel)
RF: Add: 13 , Doub: 10

Type 2 Design: Resource Sharing



Interconnection: MUX
Scalar mult: left-to-right
RF: 14

Type 3 Design: Moderate Area



Interconnection: MUX, bus
Scalar mult: left-to-right
Memory: 1,536 bits

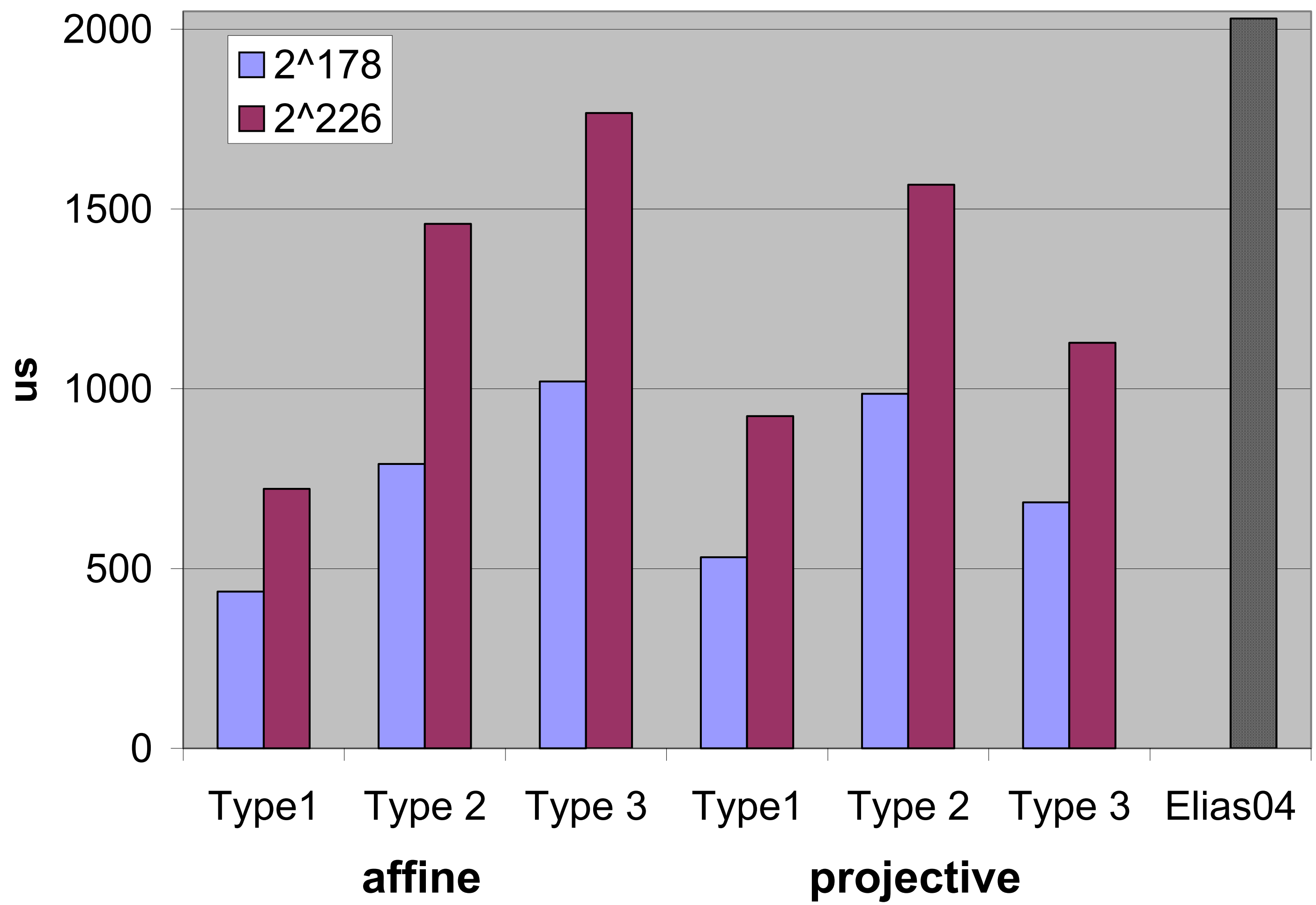
Results

		Size [slice]	f [MHz]	Latency [us]
Type 1		9,950	62.9	436
Type 2	affine	7,096	50.1	791
Type 3		4,995	50.5	1020
Type 1		12,133	36.51	531
Type 2	projective	8,693	27.07	986
Type 3		5,605	48.10	684

- Genus 2 HECC over GF(2^{89}) & group operation introduced in [Lange 2003, Pelzl et al. 2004]
- Xilinx Virtex II FPGA (XC2V4000 ff1517-6)
- Digit-Multiplier D=32

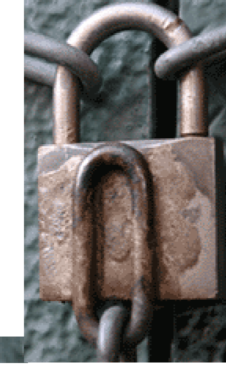
How do our results compare to previous HECC/ECC coprocessors?

Results: Latency

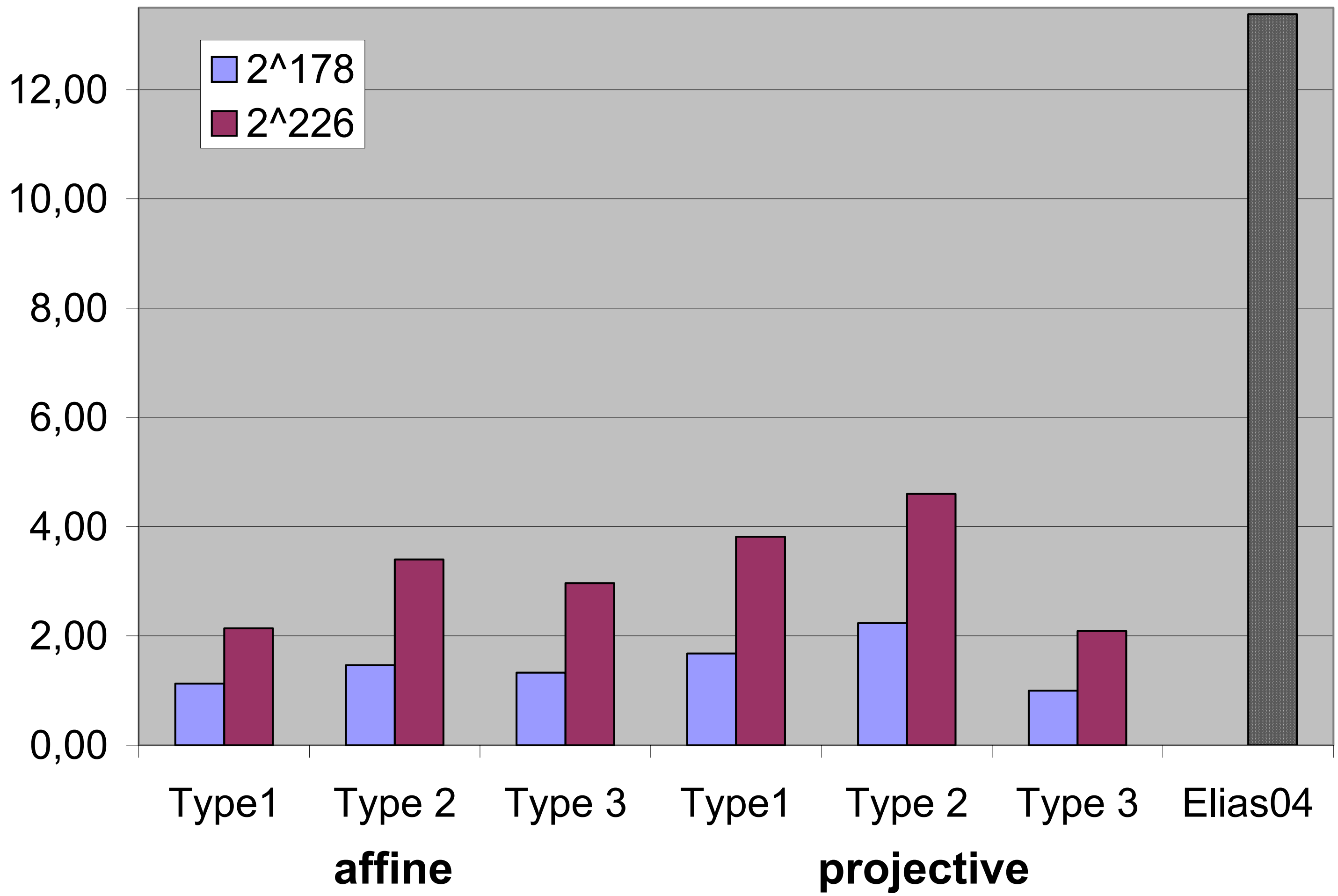


Unlimited HW ⇒ faster cryptosystem ⇒ AT product

Xilinx Virtex II FPGA (XC2V4000ff1517-6)

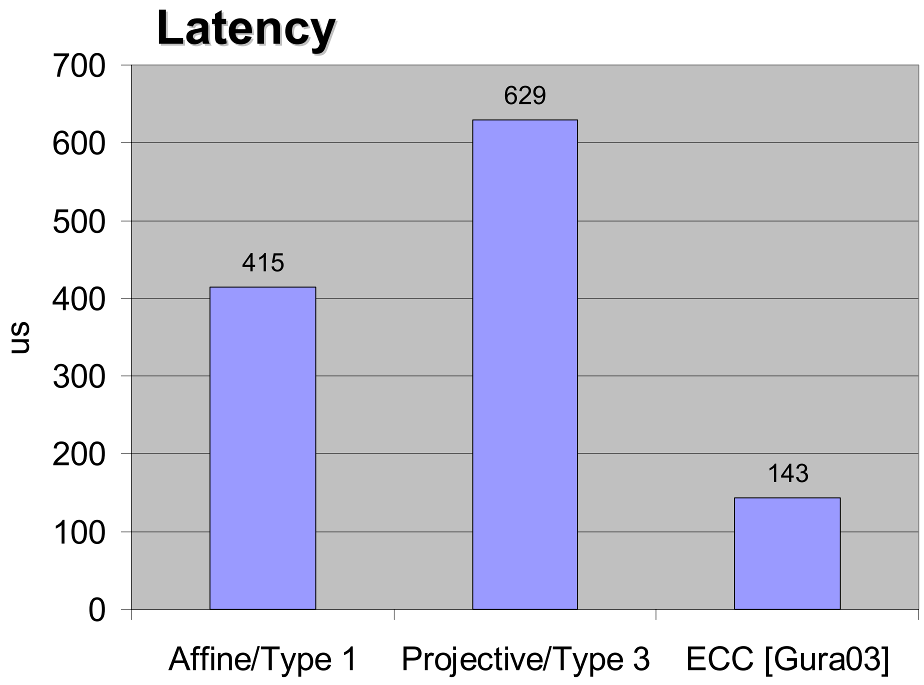


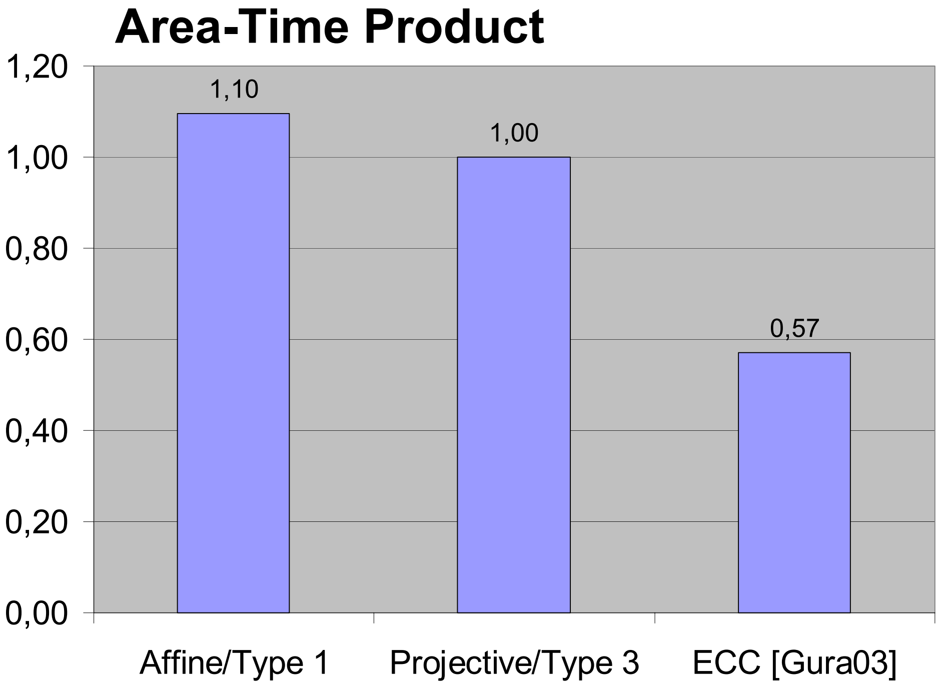
Results: Area-Time Product

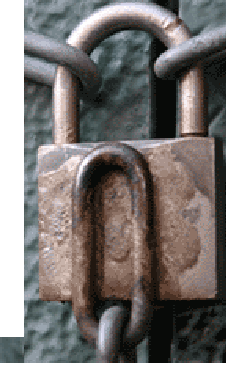


Xilinx Virtex II FPGA (XC2V4000ff1517-6)
Normalized to the best AT product

ECC versus HECC on FPGA (group order ≈2^{160})







Conclusion: HECC on FPGA

- First FPGA implementation using affine explicit formulae
- Comparison between HECC coprocessor using projective and affine coordinates.
- 64% better latency compared to [Elias et al. 2004]
- 72% smaller area compared to [Elias et al. 2004]
- Best AT-product for HECC implementation (13 times better than [Elias et al. 2004])
- More work to be done to improve HECC on FPGA
- further reading: [Wollinger 2004, Kim at al. 2004]

References

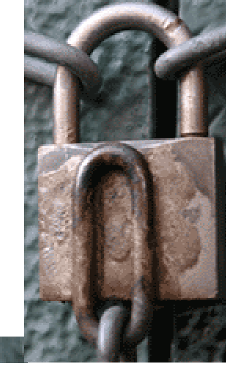
Boston N., Clancy T., Liow Y., Webster J., 2002. *Genus Two Hyperelliptic Curve Coprocessor*. B. S. Kaliski, C . K. Koc, and C. Paar, Ed. Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS 2523, 529 - 539. Springer-Verlag, 2002.

Updated version available at http://www.cs.umd.edu/~clancy/docs/hec-ches2002.pdf.

Clancy T., 2002. *Analysis of FPGA-based Hyperelliptic Curve Cryptosystems.* Master's thesis, University of Illinois Urbana-Champaign.

Clancy T., 2003. *FPGA-Based Hyperelliptic Curve Cryptosystems.* invited paper presented at AMS Central Section Meeting, April 2003.

Elias G., Miri A., Hin Yeap T., 2004. *High-Performance, FPGA-Based Hyperelliptic Curve Cryptosystems.* In The Proceeding of the 22nd Biennial Symposium on Communications.



References

Gura N., Chang S., Eberle H., Sumit G., Gupta V., Finchelstein D., Goupy E., Stebila D., 2001. *An End-to-End Systems Approach to Elliptic Curve Cryptography.* In C. K. Koc and C. Paar, Ed., Cryptographic Hardware and Embedded Systems - CHES 2001, LNCS1965, 351 - 366. Springer-Verlag, Berlin.

Kim H., Wollinger T., Choi Y., Chung K., and Paar C., 2004. *Hyperelliptic Curve Coprocessors on a FPGA.* Workshop on Information Security Applications – WISA. LNCS Springer Verlag, Berlin.

Lange T., 2003. *Formulae for Arithmetic on Genus 2 Hyperelliptic Curves.* September 2003. http://www.ruhr-uni-bochum.de/itsc/tanja/preprints/expl_sub.pdf.

Pelzl J., Wollinger T., Paar C., 2004. High Performance Arithmetic for Special Hyperelliptic Curve Cryptosystems of Genus Two. In *International Conference on Information Technology: Coding and Computing – ITCC 2004*. IEEE Computer Society.

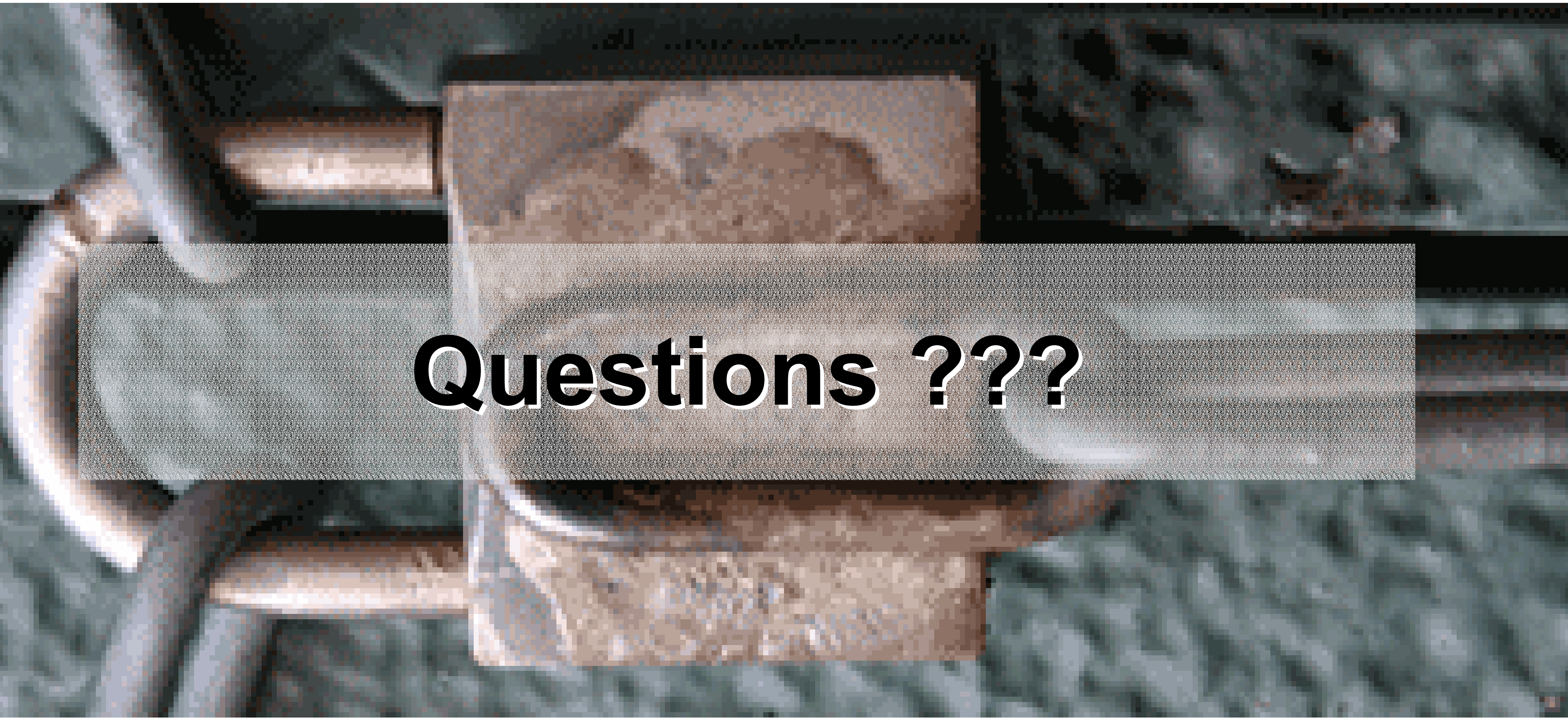
References

Theriault N., 2003. Index calculus attack for hyperelliptic curves of small genus. In G. Goos, J. Hartmanis, and J. van Leeuwen, Ed., *Advances in Cryp- tology - ASIACRYPT '03*. LNCS 2894, 79 – 92. Springer Verlag, Berlin.

Wollinger T., 2001. *Computer Architectures for Cryptosystems Based on Hyperelliptic Curves.* Master's thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 2001.

Wollinger T., Paar C., 2002. *Hardware Architectures proposed for Cryptosystems Based on Hyperelliptic Curves.* In Proceedings of the 9th IEEE International Conference on Electronics, Circuits and Systems – ICECS 2002, volume III, 1159 - 1163.

Wollinger T., 2004. *Software and Hardware Implementation of Hyperelliptic Curve Cryptosystems.* PhD thesis, Department of Electrical Engineering and Information Sciences, Ruhr-Universität Bochum, Bochum, Germany.



Additional information:
http://www.wollinger.org

http://www.crypto.rub.de

thomas@wollinger.org