Hardware Architectures for HECC

Gabriel GALLIN and **Arnaud TISSERAND**

CNRS – Lab-STICC – IRISA
HAH Project

CryptArchi June, 2017

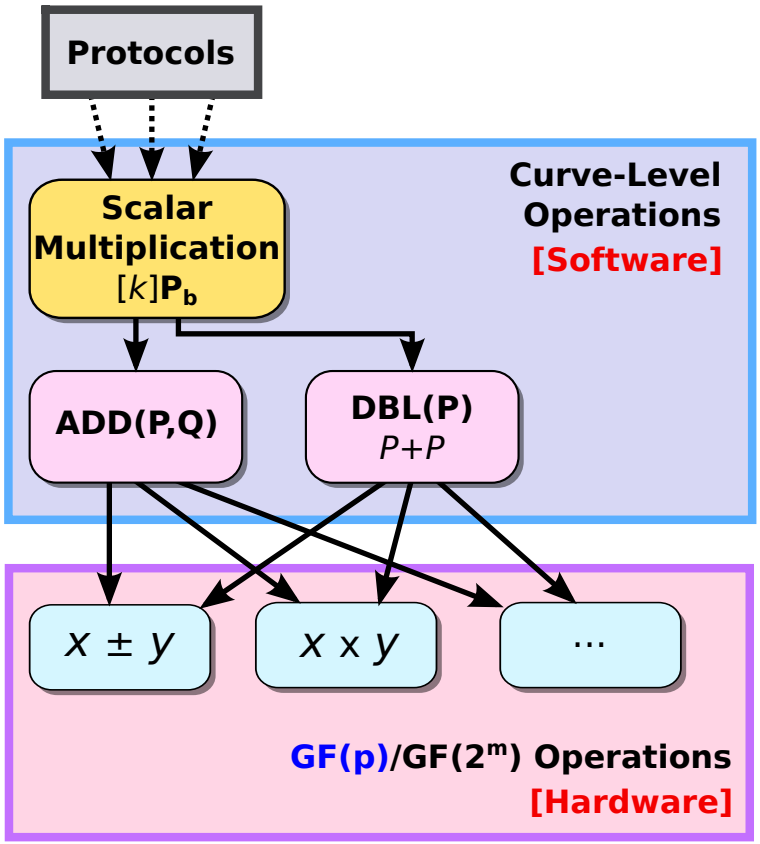
Summary

Summary

Public-Key Cryptography (PKC)

- Provides cryptographic primitives such as digital signature, key exchange and specific encryption schemes
- First PKC standard: RSA
  - $\geq$ 2000-bit keys recommended today
  - Too costly for embedded applications
- Elliptic Curve Cryptography (ECC):
  - Better performances and lower cost than RSA
  - Allows more advanced schemes
- Hyper-Elliptic Curve Cryptography (HECC):
  - Evolution of ECC focusing on larger sets of curves
  - Supposed to have a smaller cost than ECC

Operations Hierarchy in (H)ECC



- ADD and DBL built using $\mathbb{F}_{\mathcal{P}}$ operations
- Modular arithmetic in $\mathbb{F}_{\mathcal{P}}$:
  - $100 \cdots 200$ bits elements for HECC
  - Operations involve modular reduction
  - Choice for $\mathcal{P}$:
    - Generic $\mathcal{P}$: more flexible but slower
    - Specific $\mathcal{P}$ (*e.g.* pseudo-Mersenne): faster but more specific
- Modular multiplication (M) and square (S):
  - Most common and costly operations
  - Efficient dedicated units

Main metric: numbers of M and S in $\mathbb{F}_{\mathcal{P}}$

ECC, HECC, Kummer-HECC

	$\mathbb{F}_{\mathcal{P}}$ elements size	ADD	DBL	source
ECC	ℓ_{ECC}	$12\mathtt{M} + 2\mathtt{S}$	$7\mathtt{M} + 3\mathtt{S}$	[Bernstein and Lange]
HECC	$\ell_{\mathrm{HECC}} \approx \frac{1}{2}\ell_{\mathrm{ECC}}$	$40\mathtt{M} + 4\mathtt{S}$	$38\mathtt{M} + 6\mathtt{S}$	[Lange, 2005]
Kummer	ℓ_{HECC}	$19\mathtt{M} + 12\mathtt{S}$		[Renes et al., 2016]

- ECC:
  - Size of $\mathbb{F}_{\mathcal{P}}$ elements $2\times$ larger
  - Simpler ADD and DBL operations
- HECC:
  - Smaller $\mathbb{F}_{\mathcal{P}}$
  - More operations in $\mathbb{F}_{\mathcal{P}}$ for ADD / DBL
- Kummer-HECC is more efficient than ECC [Renes et al., 2016]:
  - ARM Cortex M0: up to 75% clock cycles reduction for signatures
  - AVR AT-mega: up to 32% cycles reduction for Diffie-Hellman

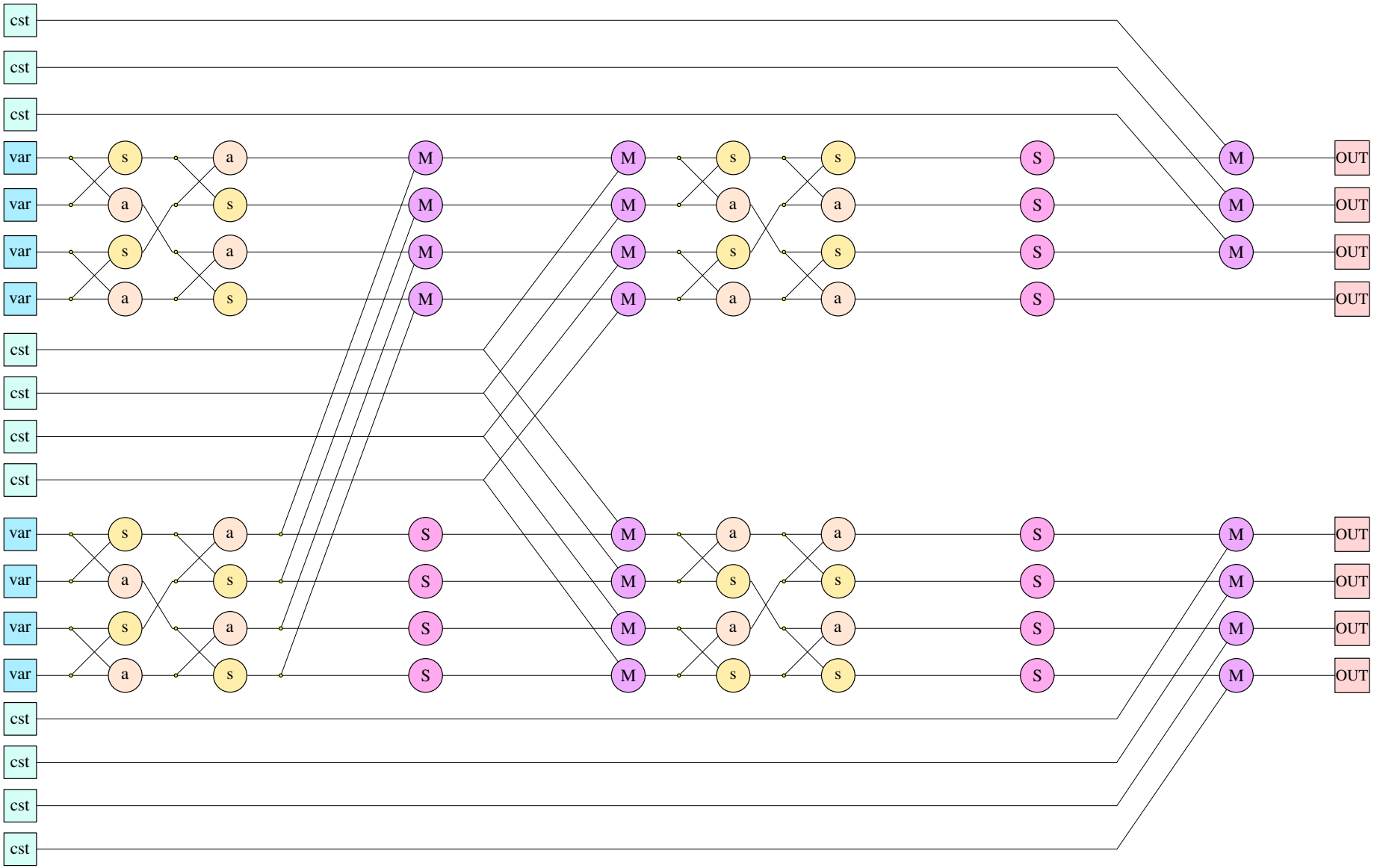
Summary

Curve-Level Operations in Kummer

- No ADD operation but still DBL
- *Differential addition*: $\texttt{xADD}(\pm P, \pm Q, \pm(P-Q)) \rightarrow \pm(P+Q)$
- xADD and DBL can be combined:
  $\texttt{xDBLADD}(\pm P, \pm Q, \pm(P-Q)) \rightarrow (\pm[2]P, \pm(P+Q))$

For details see [Renes et al., 2016], [Gaudry, 2007] and [Bos et al., 2016]

xDBLADD $\mathbb{F}_{\mathcal{P}}$ Operations



Scalar Multiplication

Montgomery ladder based *crypto_scalarmult* [Renes et al., 2016]:

Require: m-bit scalar $k = \sum_{i=0}^{m-1} 2^i k_i$, point P_b, $cst \in \mathbb{F}_{\mathcal{P}}^4$
Ensure: $V_1 = [k]P_b$, $V_2 = [k+1]P_b$

$V_1 \leftarrow cst$
$V_2 \leftarrow P_b$
for $i = m-1$ **downto** 0 **do**
  $(V_1, V_2) \leftarrow \texttt{CSWAP}(k_i, (V_1, V_2))$
  $(V_1, V_2) \leftarrow \texttt{xDBLADD}(V_1, V_2, P_b)$
  $(V_1, V_2) \leftarrow \texttt{CSWAP}(k_i, (V_1, V_2))$
end for
return (V_1, V_2)

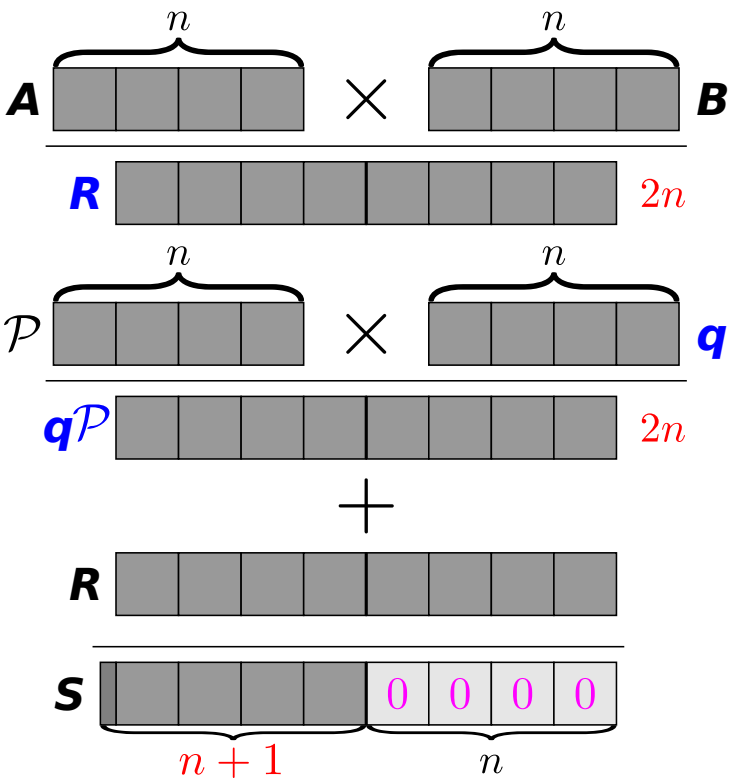
$\texttt{CSWAP}(k_i, (X, Y))$ returns (X, Y) if $k_i = 0$, else (Y, X)

- Constant time, uniform operations (independent from key bits)
- Some parallelism between xDBLADD internal $\mathbb{F}_{\mathcal{P}}$ operations
- CSWAP: very simple but involves secret bits (to be protected)

Summary

Montgomery Modular Multiplication (MMM)

$$R = A \times B \qquad n \times n \rightarrow 2n \text{ bits}$$
$$q = (R \times (-\mathcal{P}^{-1})) \mod (2^n) \qquad n \times n \rightarrow n \text{ bits}$$
$$q\mathcal{P} = q \times \mathcal{P} \qquad n \times n \rightarrow 2n \text{ bits}$$



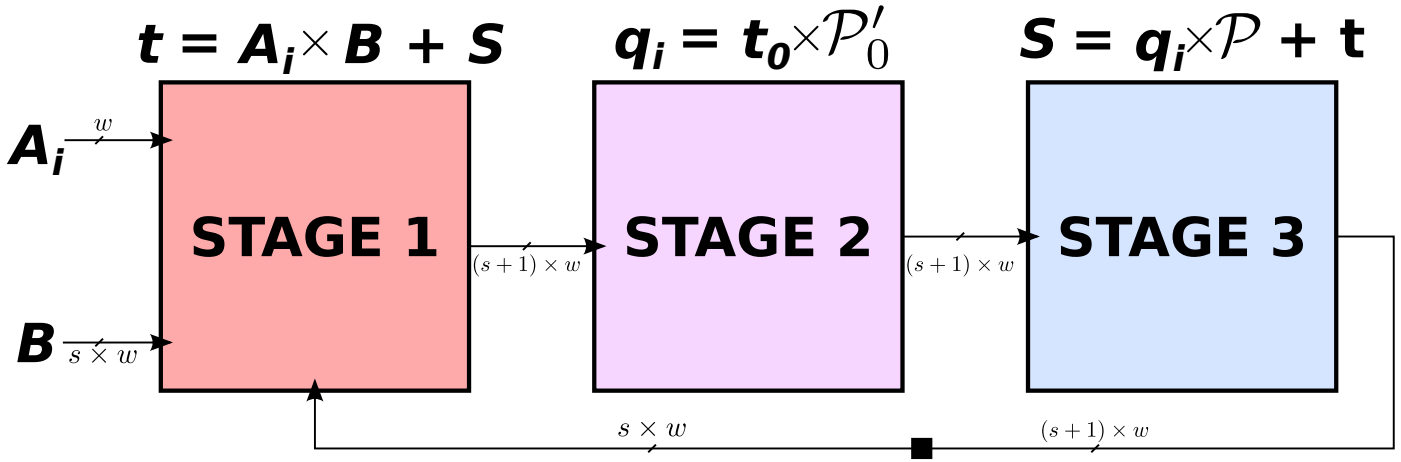
- Objective: $A \times B \bmod \mathcal{P}$
- Proposed in [Montgomery, 1985]
- Variants are actual state-of-the-art for $\mathbb{F}_{\mathcal{P}}$ multiplication (with generic $\mathcal{P}$)
- Final reduction step discards n LSBs

Modular Multiplication: Dependencies Problem

- In practice, MMM is interleaved
  - Operands are split into s words of w bits such that $n = s \times w$
  - Iterations over partial products and reductions on words
  - *Coarsely Integrated Operand Scanning* (CIOS) from [Koç et al., 1996]
- Impact on hardware implementation
  - Dependencies $\rightarrow$ latencies between internal iterations
  - Hardware pipeline in DSP slices cannot be filled efficiently
- Proposed solution: Hyper-Threaded Modular Multiplier (HTMM)
  - Based on simple CIOS algorithm
  - Use idle stages to compute other independent MMMs in parallel

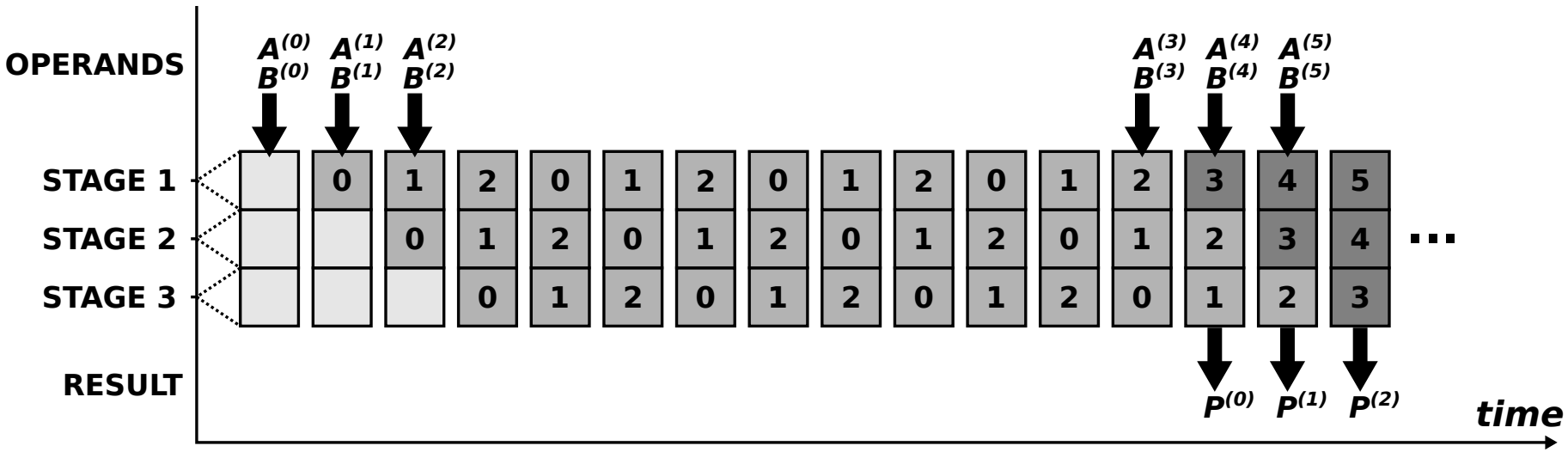
HTMM Internal Architecture

- HTMM architecture: 3 hardware stages
  - Stages are fully pipelined (several clock cycles per stage)
  - 3 to 4 DSP slices in each stage



HTMM Internal Architecture

- HTMM architecture: 3 hardware stages
  - Stages are fully pipelined (several clock cycles per stage)
  - 3 to 4 DSP slices in each stage



HTMM Implementations

- Xilinx FPGAs
  - Virtex 4 XC4VLX100 (V4)
  - Virtex 5 XC5VLX110T (V5)
  - Spartan 6 XC6SLX75 (S6)
- Comparison with fastest MMM implementation in literature
  - Design presented in [Ma et al., 2013]
  - Implemented on the same FPGAs for fair comparison
- 2 versions of HTMM:
  - HTMM_DRAM : operands stored in FPGA slices (LUTs)
  - HTMM_BRAM : operands stored in FPGA BRAMs
- Parameters for HTMM:
  - $\mathcal{P} \rightarrow$ 128 bits
  - $w = 34$ bits, $s = 4$
  - Operands size $n = s \times w = 134$ bits

HTMM Implementations Results

Results for 3 independent multiplications:

Version	FPGA	DSP	BRAM 18K/9K	FF	LUT	Slices	Freq. (MHz)	Nb. cycles	Time (ns)
[Ma et al., 2013]	V4	21	6/0	1311	1201	879	252	65	258
	V5	21	6/0	1310	1027	406	296		220
	S6	21	0/6	1280	1600	540	210		309
HTMM_DRAM	V4	11	0/0	1638	1128	1346	330	79	239
	V5	11	0/0	1616	652	517	400		198
	S6	11	0/0	1631	1344	483	302		261
HTMM_BRAM	V4	11	2/0	615	364	449	328	79	241
	V5	11	2/0	593	371	249	357		221
	S6	11	0/2	587	359	180	304		260

S6: -47% DSPs, -66% BRAMs, -66% slices, -15% duration

For only 1 single `M`, HTMM is less efficient (69 cycles against 25)

Summary

Architectures Exploration for (H)ECC

HECC architectures require different types of units:

- $\mathbb{F}_{\mathcal{P}}$ arithmetic units: add/sub, mul, sqr, inv, ...
- Memories, (secure) registers, ...
- Interconnect, global input/output, ...
- Dedicated (secure) control

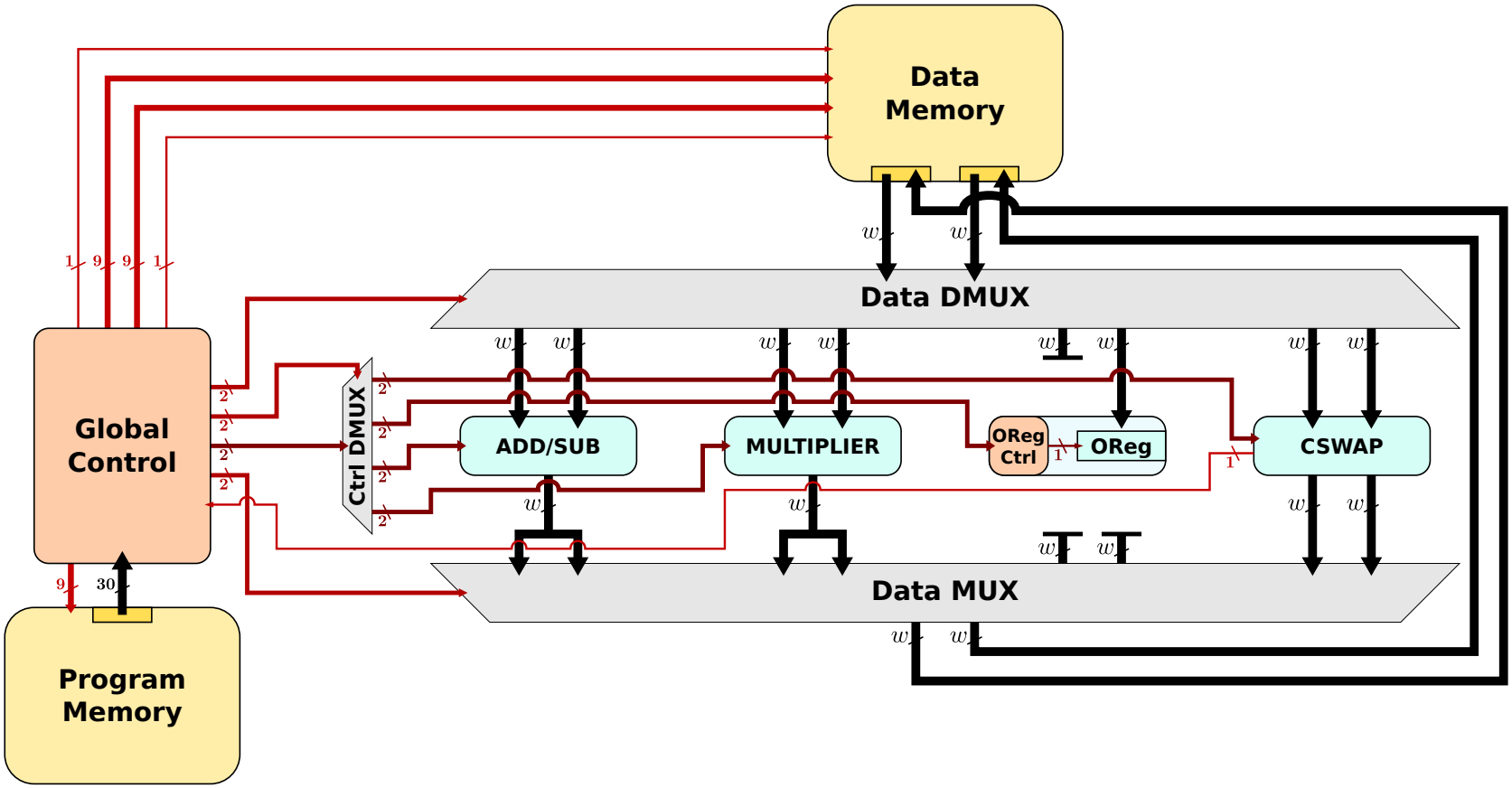
Problems

- Coding a complete accelerator fully in HDL is costly
- Large design space for various architectures types and parameters (nb. units, algorithms, internal communications and control)
- Need for evaluation of various architectures and parameters
- Need for numerical validation and debug

Proposed Design Framework

- Hierarchical description and simulation for HECC architectures at CCABA level (Critical-Cycle Accurate, Bit Accurate)
  - Units inputs/outputs are bit accurate
  - Units inputs/outputs and external control are critical cycle accurate
- Description of various architectures at high-level
  - Composition of units for differents parameters and optimizations
  - Scheduling tool for control and communications (*work in progress*)
- Units described, optimized and validated in HDL
  - Perfectly known behavior $\rightarrow$ no need for cycle accurate simulation
  - Area, latency, ... come from actual FPGA implementation
- Dedicated simulator in Python
  - Fast development and numerical validation
  - Sage (`http://www.sagemath.org/`) interface for HECC support

Typical Architecture Model



Parameters specified at design time:

- Width w and nb. words s for internal communications ($s \times w = n$)
- Types and number of units

Configuration for Implementations

- 128 bits HECC solutions
- $\mathbb{F}_{\mathcal{P}}$ adder-subtractor (AddSub):
  - 4 cycles latency pipeline
  - $8 \cdots 11$ cycles delay depending on w
- $\mathbb{F}_{\mathcal{P}}$ multiplier (HTMM):
  - Hyper-threaded multiplier for 3 sets of operands computed in parallel
  - 5 cycles latency for loading and reading
  - $68 \cdots 71$ cycles delay depending on w
- CSWAP unit:
  - Secure management of key bits
  - $2 \cdots 4$ cycles delay depending on w

Results for Basic Architecture (1 Add/Sub, 1 HTMM)

Version $s \times w$	Clock cycles	Units	DSP	BRAM	FF	LUT	Slices	RAM #lines
4x34	207,383	HTMM	11	2	587	359	180	12
		AddSub	0	0	366	226	80	-
		DATA_MEM	0	1	0	0	0	112
		PRGM_MEM	0	1	0	0	0	208
		CSWAP	0	0	536	290	103	-
2x68	185,615	HTMM	11	2	970	633	315	12
		AddSub	0	0	713	382	148	-
		DATA_MEM	0	2	0	0	0	56
		PRGM_MEM	0	1	0	0	0	234
		CSWAP	0	0	553	297	122	-
1x136	183,051	HTMM	11	2	1066	623	309	12
		AddSub	0	0	784	464	212	-
		DATA_MEM	0	4	0	0	0	26
		PRGM_MEM	0	1	0	0	0	250
		CSWAP	0	0	685	431	155	-

s: number of words, w: size of words

Increasing the Number of Arithmetic Units

Version $s \times w$	Clock cycles	Units	DSP	BRAM	FF	LUT	Slices	RAM #lines
4x34	**203,543**	HTMM **x 2**	**22**	**4**	**1174**	**718**	**360**	12
		ADDSUB **x 2**	0	0	**732**	**452**	**160**	-
		DATA_MEM	0	1	0	0	0	**108**
		PRGM_MEM	0	1	0	0	0	**213**
		CSWAP	0	0	536	290	103	-
2x68	**125,455**	HTMM **x 2**	**22**	**4**	**1940**	**1266**	**630**	12
		ADDSUB **x 2**	0	0	**1426**	**764**	**296**	-
		DATA_MEM	0	4	0	0	0	**50**
		PRGM_MEM	0	1	0	0	0	**211**
		CSWAP	0	0	553	297	122	-
1x136	**115,211**	HTMM **x 2**	**22**	**4**	**2132**	**1246**	**618**	12
		ADDSUB **x 2**	0	0	**1568**	**928**	**424**	-
		DATA_MEM	0	4	0	0	0	**25**
		PRGM_MEM	0	1	0	0	0	**235**
		CSWAP	0	0	685	431	155	-

s: number of words, w: size of words

256b ECC vs 128b HECC (similar theoretical security)

FPGA	Version	DSP	BRAM 18K	Slices	Freq. (MHz)	Nb. cycles	Time (ms)
V4	ECC	37	11	4655	250	109,297	0.44
	HECC_1u	11	7	1413	330	183,051	0.55
	HECC_2u	22	9	2356	330	115,211	0.35
V5	ECC	37	10	1725	291	109,297	0.38
	HECC_1u	11	7	873	360	183,051	0.51
	HECC_2u	22	9	1542	360	115,211	0.32

Gain 1u on V5: -70% DSPs, -30% BRAMs, -49% slices, +30% duration

Gain 2u on V5: -40% DSPs, -10% BRAMs, -10% slices, -15% duration

ECC results from [Ma et al., 2013]

Conclusions and Perspectives

- Kummer based HECC is an efficient alternative to ECC
  - More complex formulas but larger internal parallelism
  - Large exploration space for architectures and arithmetic
- We designed a CCABA modeling and simulator
  - High-level hierarchical description of architectures
  - Units described in HDL, only critical cycles are used
  - Fast validation/debug and evaluation of solutions in exploration space
- Future works
  - Study advanced scheduling algorithms
  - Automating generation of HDL code from high-level description
  - Explore new architectural solutions

This work was partially funded by **HAH** project http://h-a-h.inria.fr/

Thank you for your attention

References I

[Bernstein and Lange] Bernstein, D. J. and Lange, T.
Explicit-formulas database.
http://hyperelliptic.org/EFD/.

[Bos et al., 2016] Bos, J. W., Costello, C., Hisil, H., and Lauter, K. (2016).
Fast cryptography in genus 2.
Journal of Cryptology, 29(1):28–60.

[Cohen et al., 2005] Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., and Vercauteren, F. (2005).
Handbook of Elliptic and Hyperelliptic Curve Cryptography.
Discrete Mathematics and Its Applications. Chapman & Hall/CRC.

[Gaudry, 2007] Gaudry, P. (2007).
Fast genus 2 arithmetic based on theta functions.
Journal of Mathematical Cryptology, 1(3):243–265.

[Hankerson et al., 2004] Hankerson, D., Menezes, A., and Vanstone, S. (2004).
Guide to Elliptic Curve Cryptography.
Springer.

[Koç et al., 1996] Koç, Ç. K., Acar, T., and Kaliski, Jr., B. S. (1996).
Analyzing and comparing Montgomery multiplication algorithms.
Micro, IEEE, 16(3):26–33.

[Lange, 2005] Lange, T. (2005).
Formulae for Arithmetic on Genus 2 Hyperelliptic Curves.
Applicable Algebra in Engineering, Communication and Computing, 15(5):295–328.

References II

[Ma et al., 2013] Ma, Y., Liu, Z., Pan, W., and Jing, J. (2013).
A high-speed elliptic curve cryptographic processor for generic curves over GF(p).
In *Proc. 20th International Workshop on Selected Areas in Cryptography (SAC)*, volume 8282 of *LNCS*, pages 421–437. Springer.

[Montgomery, 1985] Montgomery, P. L. (1985).
Modular multiplication without trial division.
Mathematics of Computation, 44(170):519–521.

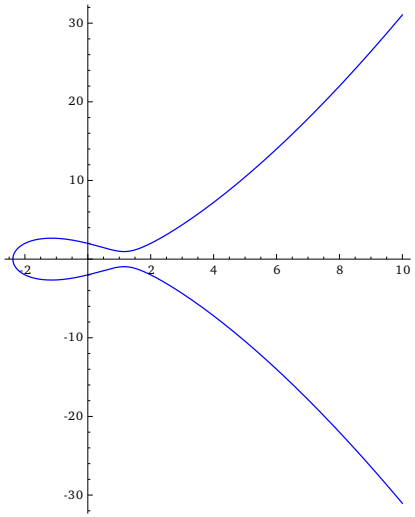
[Montgomery, 1987] Montgomery, P. L. (1987).
Speeding the Pollard and elliptic curve methods of factorization.
Mathematics of Computation, 48(177):243–264.

[Orup, 1995] Orup, H. (1995).
Simplifying quotient determination in high-radix modular multiplication.
In *Proc. 12th Symposium on Computer Arithmetic (ARITH)*, pages 193–199. IEEE Computer Society.

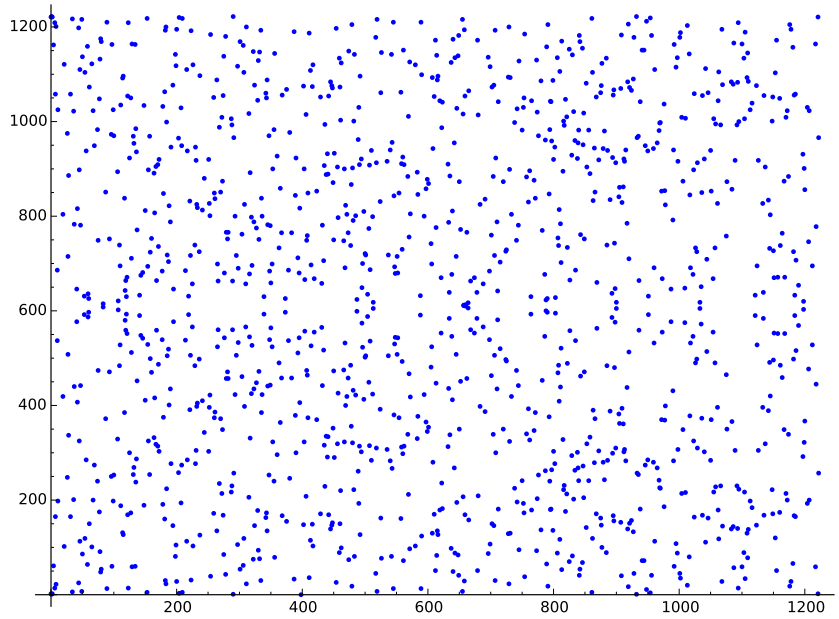
[Renes et al., 2016] Renes, J., Schwabe, P., Smith, B., and Batina, L. (2016).
μKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers.
In *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9813 of *LNCS*, pages 301–320. Springer.

Elliptic and Hyper-Elliptic Curves for Crypto

- Elliptic Curves
  - Equation (Weierstrass) $E/\mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
  - Defined over finite fields $\mathbb{K}$: $\mathbb{F}_{2^m}$, prime finite field $\mathbb{F}_\mathcal{P}$ or $GF(p)$
  - $\mathbb{F}_\mathcal{P}$ elements for coefficients and coordinates: $200 \cdots 400$ bits



Curve over $\mathbb{R}$ (not for crypto)



Curve over $\mathbb{F}_{1223}$

Elliptic and Hyper-Elliptic Curves for Crypto

- Elliptic Curves
  - Equation (Weierstrass) $E/\mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
  - Defined over finite fields $\mathbb{K}$: $\mathbb{F}_{2^m}$, prime finite field $\mathbb{F}_{\mathcal{P}}$ or $GF(p)$
  - $\mathbb{F}_{\mathcal{P}}$ elements for coefficients and coordinates: $200 \cdots 400$ bits
- Hyper-Elliptic Curves
  - **More complex!**
  - Equation $H/\mathbb{K} : y^2 + h(x)y = f(x)$, $deg(h) < g$ and $deg(f) = 2g + 1$
  - g: genus of the curve, $g \leq 2$ in practice for reliable HECC
  - $\mathbb{F}_{\mathcal{P}}$ elements for coefficients and coordinates: $100 \cdots 200$ bits

Elliptic and Hyper-Elliptic Curves for Crypto

- Elliptic Curves
  - Equation (Weierstrass) $E/\mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$
  - Defined over finite fields $\mathbb{K}$: $\mathbb{F}_{2^m}$, prime finite field $\mathbb{F}_\mathcal{P}$ or $GF(p)$
  - $\mathbb{F}_\mathcal{P}$ elements for coefficients and coordinates: $200 \cdots 400$ bits
- Hyper-Elliptic Curves
  - **More complex!**
  - Equation $H/\mathbb{K} : y^2 + h(x)y = f(x)$, $deg(h) < g$ and $deg(f) = 2g + 1$
  - g: genus of the curve, $g \leq 2$ in practice for reliable HECC
  - $\mathbb{F}_\mathcal{P}$ elements for coefficients and coordinates: $100 \cdots 200$ bits
- Kummer surface
  - Not an additive group: no addition law
  - Can be used in HECC using some *(magic)* trick
  - Reduced complexity for curve operations

HTMM Detailed Architecture