Hardware Arithmetic Units and Cryptoprocessors for Hyperelliptic Curve Cryptography

Gabriel GALLIN

CNRS – IRISA – Univ. Rennes 1

November 29$^{\text{th}}$, 2018

Ph.D. supervised by Arnaud TISSERAND, CNRS – Lab-STICC

Cybersecurity Challenges

- Digital systems are widely used in many applications
  - economy: credit cards, online payments, ...
  - medical: medical files, e-Health devices, ...
  - Internet of Things (IoT): self-driving cars, smart homes, ...
  - communications: telephony, emails, social networks, ...
  - ...

- Strong needs for efficent digital security
  - fast for user convinience
  - reduced power consumption for battery-based systems
  - small circuit area for embedded systems
  - resistant to attacks: theoretical, logical and physical

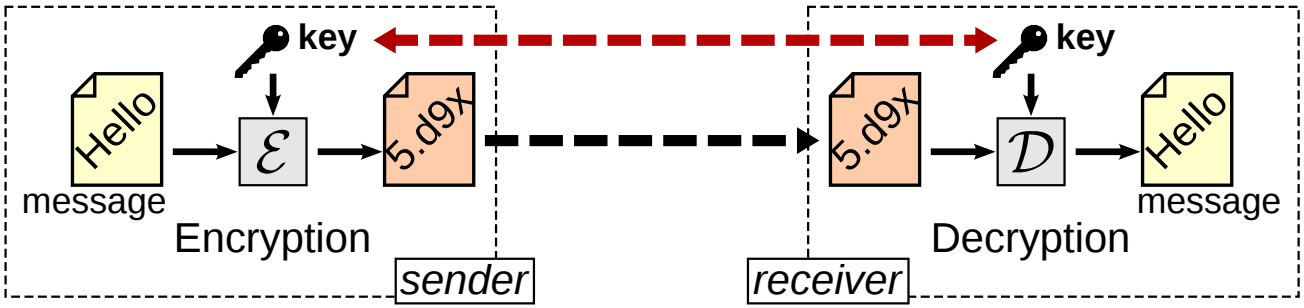
Example: Simplified Payment with Credit Cards

Cryptographic primitives:

- **authentication**: asserts identity of user, credit card and bank
- **integrity**: ensures exchanged data are complete and unmodified
- **confidentiality**: asserts secrecy of exchanded data

Overview on Cryptography: Symmetric Cryptography

- Also called secret-key cryptography
- Encryption and decryption with shared secret key



- Very efficient and wildely used to ensure confidentiality
- Problems with symmetric cryptography
  - secret key must be shared between sender and receiver
  - communications with several parties $\rightarrow$ many keys to manage

Overview on Cryptography: Asymmetric Cryptography

- Also known as public-key cryptography (PKC)
  - uses a pair of private key and public key
  - extensively used for digital signatures and key exchanges
  - more expensive than symmetric cryptography
- First PKC: RSA proposed by Rivest, Shamir and Adleman in 1978
  - huge commercial success and still widely used
  - large keys (> 2000 bits recommended) and very costly for embedded applications
- **E**lliptic **C**urve **C**ryptography by Miller in 1985 and Koblitz in 1987
  - 200 to 500 bits keys recommended: better performances than RSA
  - current PKC standard for various secured applications
    e.g. french passports or secured Internet browsing

Hyper-Elliptic Curve Cryptography

- HECC proposed by Koblitz in 1988
  - size of internal values divided by 2 but more arithmetic operations
  - before late 2000s, HECC was less efficient than ECC
- New HECC cryptosystem proposed by Gaudry [1] in 2007
  - requires less arithmetic operations
  - more efficient than ECC in theory
  - size of internal values is around 128 bits (equiv. to ECC 256b)
- μKummer proposed by Renes *et al.* [6] in 2016
  - software implementation of Gaudry's HECC on microcontrollers
  - -75% and -35% time for digital signature and key exchange
- Very few recent hardware implementations of recent HECC cryptosystems

HAH Project

- **H**ardware and **A**rithmetic for **H**ECC

- 3-year labex project (2014-2017) involving
  - IRISA / Lab-STICC funded by labex CominLabs and Britanny region
  - IRMAR lab. for mathematics funded by labex Lebesgue

HAH Project: Objectives

- Propose new units for basic arithmetic operations in HECC
  - modular arithmetic for 128–300-bit operands
  - design small circuits with high frequencies and low computation time
- Design new hardware cryptoprocessors for HECC
  - implement best state-of-the-art HECC cryptosystems
  - explore various performance *vs.* cost tradeoffs
  - confirm efficiency of HECC *vs.* ECC in hardware
- Robust against physical attacks: SPA (Simple Power Analysis)
- Flexible designs to support different curves and parameters

Summary

Modular Operations in HECC

- HECC requires to compute arithmetic operations $(\pm, \times)$ in $\mathrm{GF}(P)$
  - operands and results $\in \{0, 1, ..., P-1\}$
  - P is a 100–300-bit prime

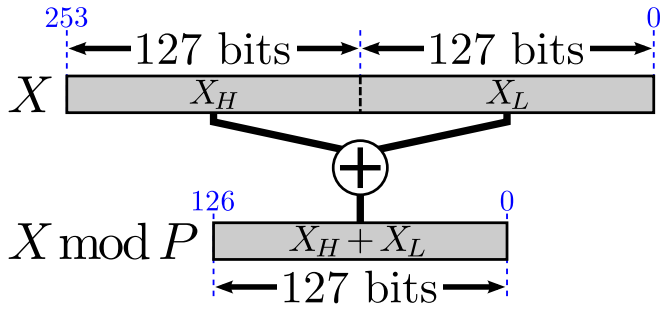
- Most frequent and costly operation: modular multiplication (MM)
  e.g. 75% of overall computation time in μKummer [6]

- Example: multiplications modulo small $P = 23$

$$2 \times 10 = 20 \qquad 2 \times 10 \bmod 23 = 20$$
$$9 \times 18 = 162 \qquad 9 \times 18 \bmod 23 = 1$$
$$4 \times 10 = 40 \qquad 4 \times 10 \bmod 23 = 17$$
$$19 \times 17 = 323 \qquad 19 \times 17 \bmod 23 = 1$$

Modular Reduction

- Fast reduction modulo specific primes with specific structures
  - *e.g.* Mersenne prime $P = 2^{127} - 1$ * used in μKummer:



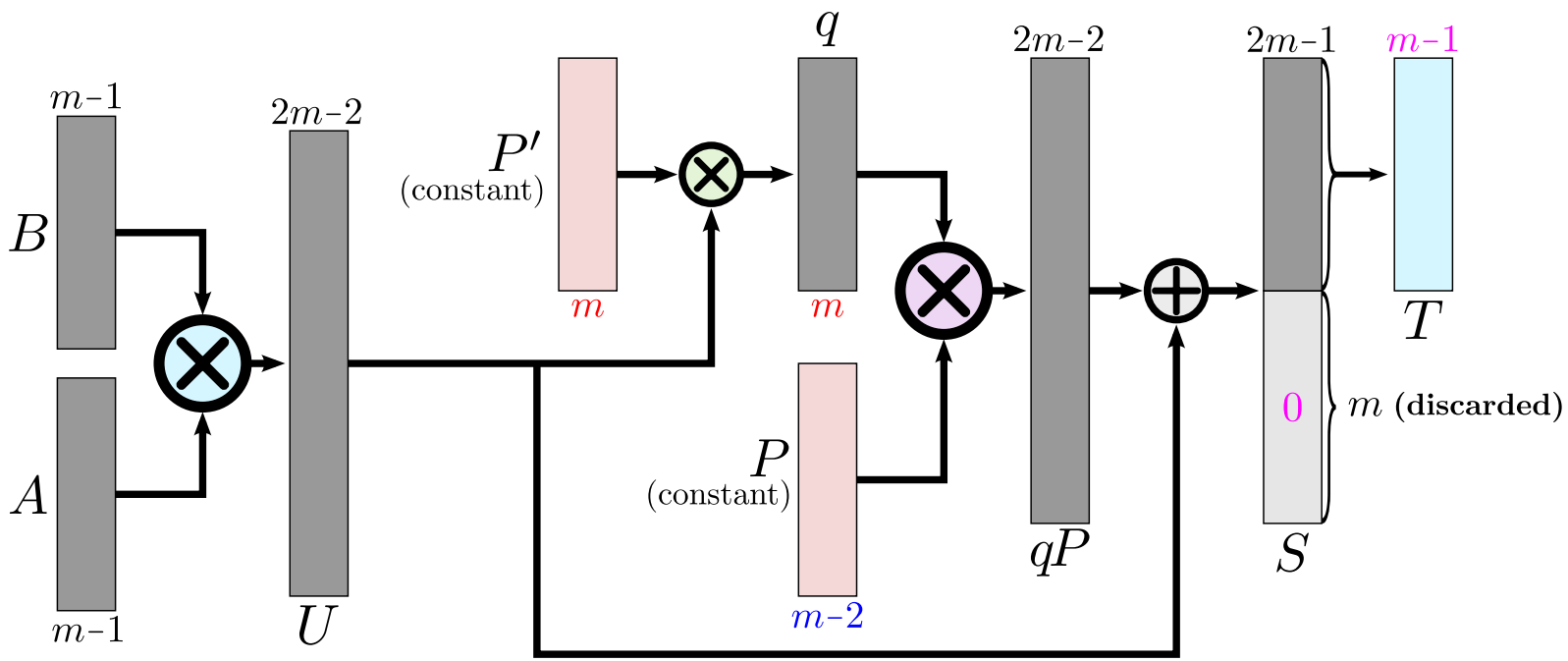
  - limited to very few primes: **not possible with flexibility constraints**

- Reduction modulo generic primes
  - more complex but **supports all primes** of a given max. size
  - several efficient algorithms for operations modulo generic P

*$2^{127} - 1 = (11111111111111111111111...11111111111111111111111)_2$

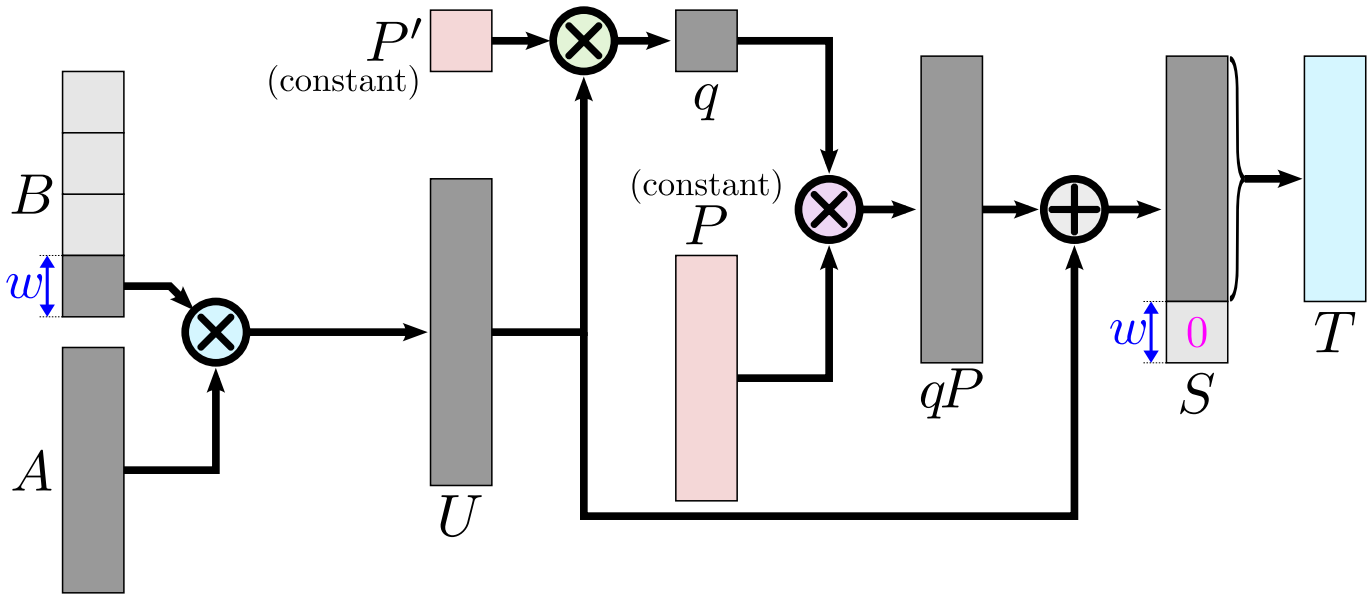
Modular Multiplication: Montgomery's Algorithm

- **M**ontgomery **M**odular **M**ultiplication proposed in 1985 [5]
  - best MM algorithm for generic primes P
  - max. size of P: $m-2$ bits



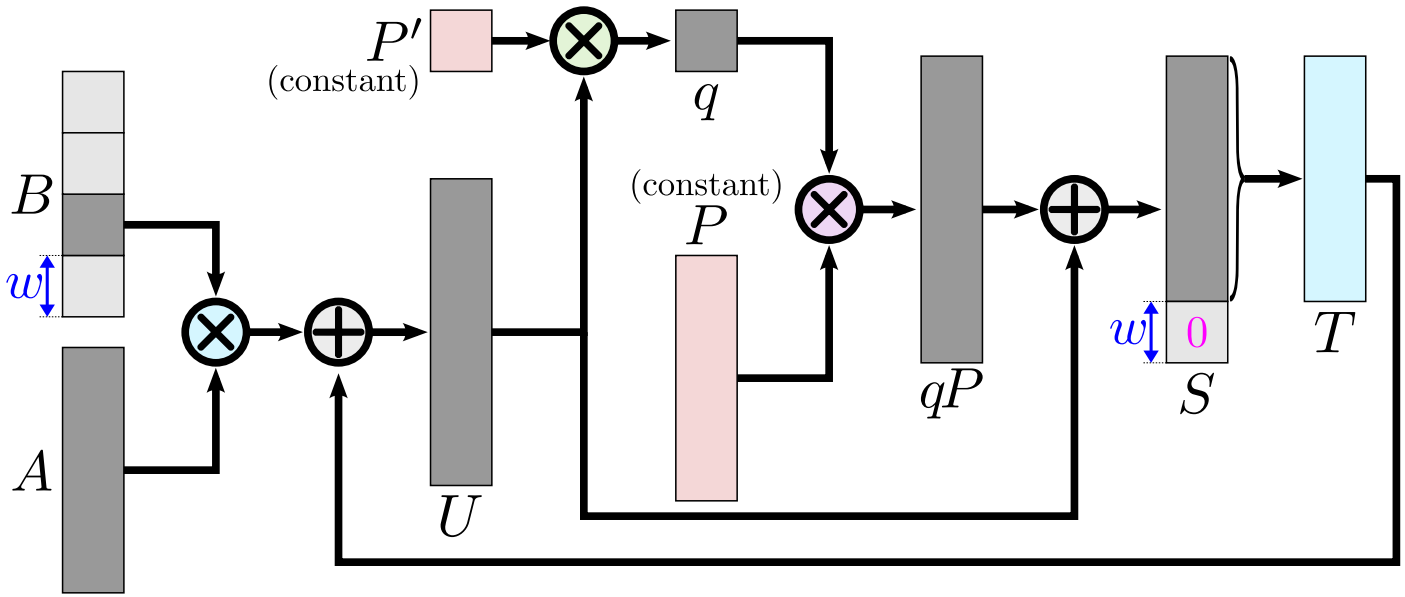
Interleaved MMM

- MMM operands are split into s words of w bits $(s \times w = m)$
  - CIOS (*Coarsely Integrated Operand Scanning*) from Koc *et al.* [2]
  - iterations over small partial products with partial reduction steps
  - strong dependencies between iterations



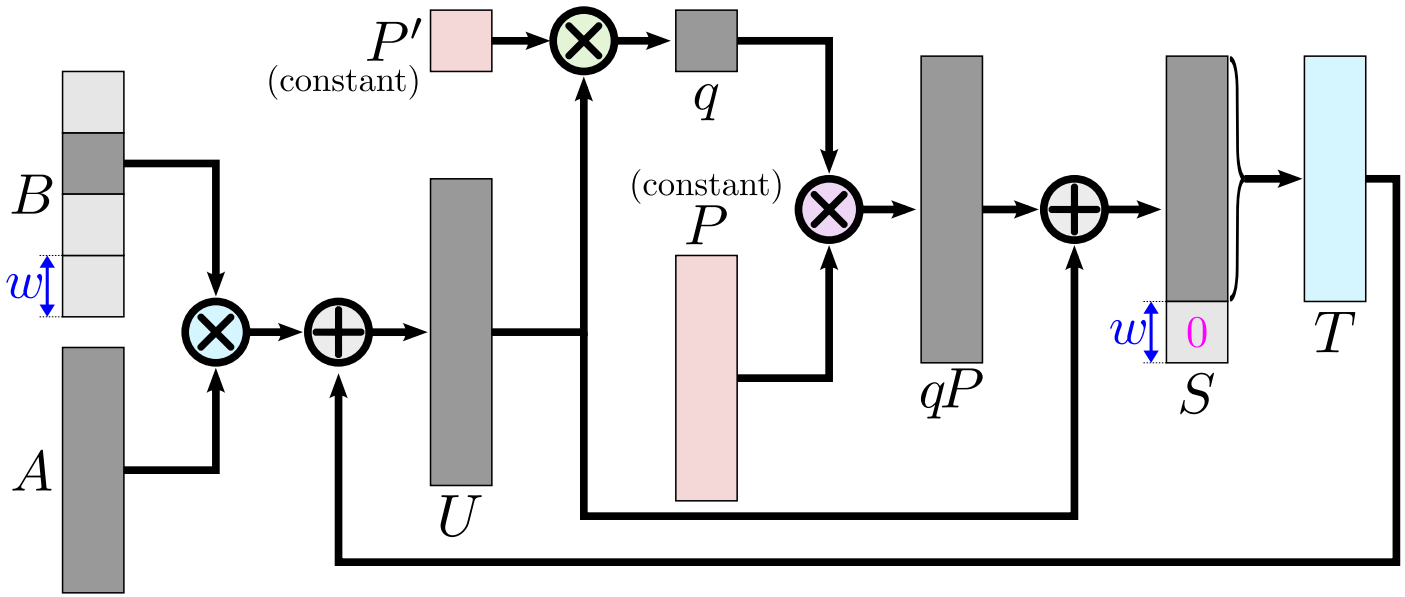
Interleaved MMM

- MMM operands are split into s words of w bits $(s \times w = m)$
  - CIOS (*Coarsely Integrated Operand Scanning*) from Koc *et al.* [2]
  - iterations over small partial products with partial reduction steps
  - strong dependencies between iterations



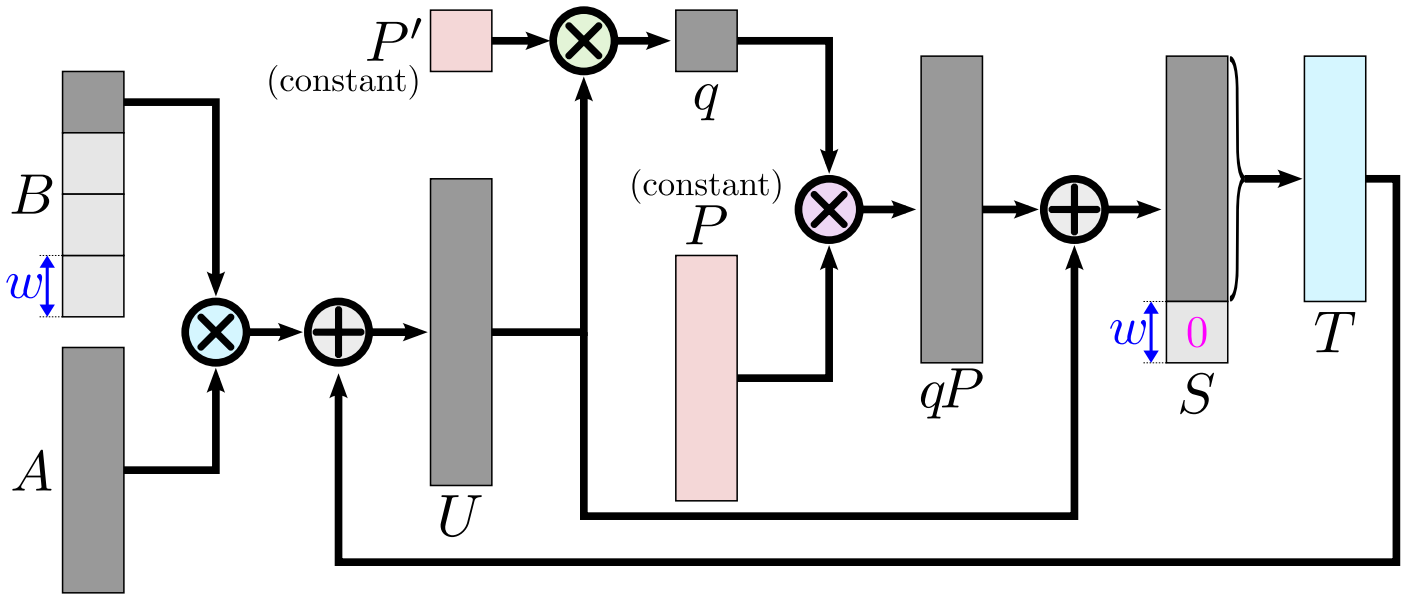
Interleaved MMM

- MMM operands are split into s words of w bits $(s \times w = m)$
  - CIOS (*Coarsely Integrated Operand Scanning*) from Koc *et al.* [2]
  - iterations over small partial products with partial reduction steps
  - strong dependencies between iterations



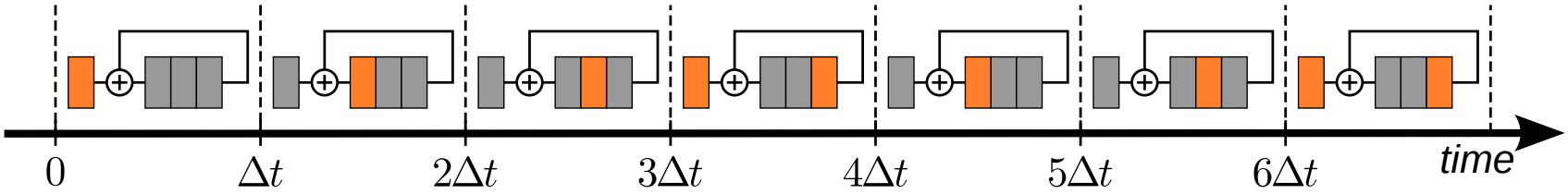
Interleaved MMM

- MMM operands are split into s words of w bits $(s \times w = m)$
  - CIOS (*Coarsely Integrated Operand Scanning*) from Koc *et al.* [2]
  - iterations over small partial products with partial reduction steps
  - strong dependencies between iterations

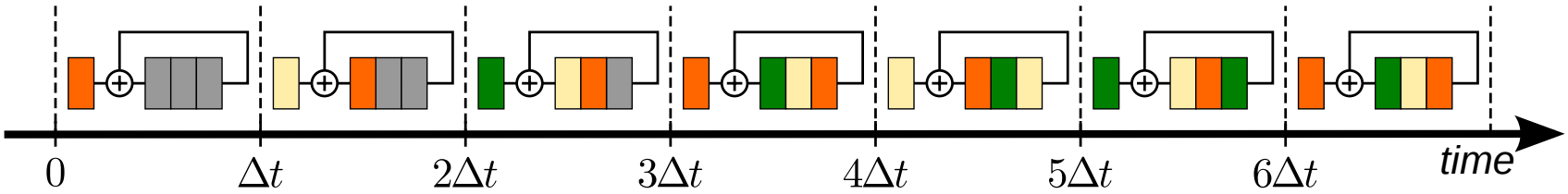


Hyper-Threading: Principle

- Dependencies in CIOS $\rightarrow$ idle stages in the pipeline



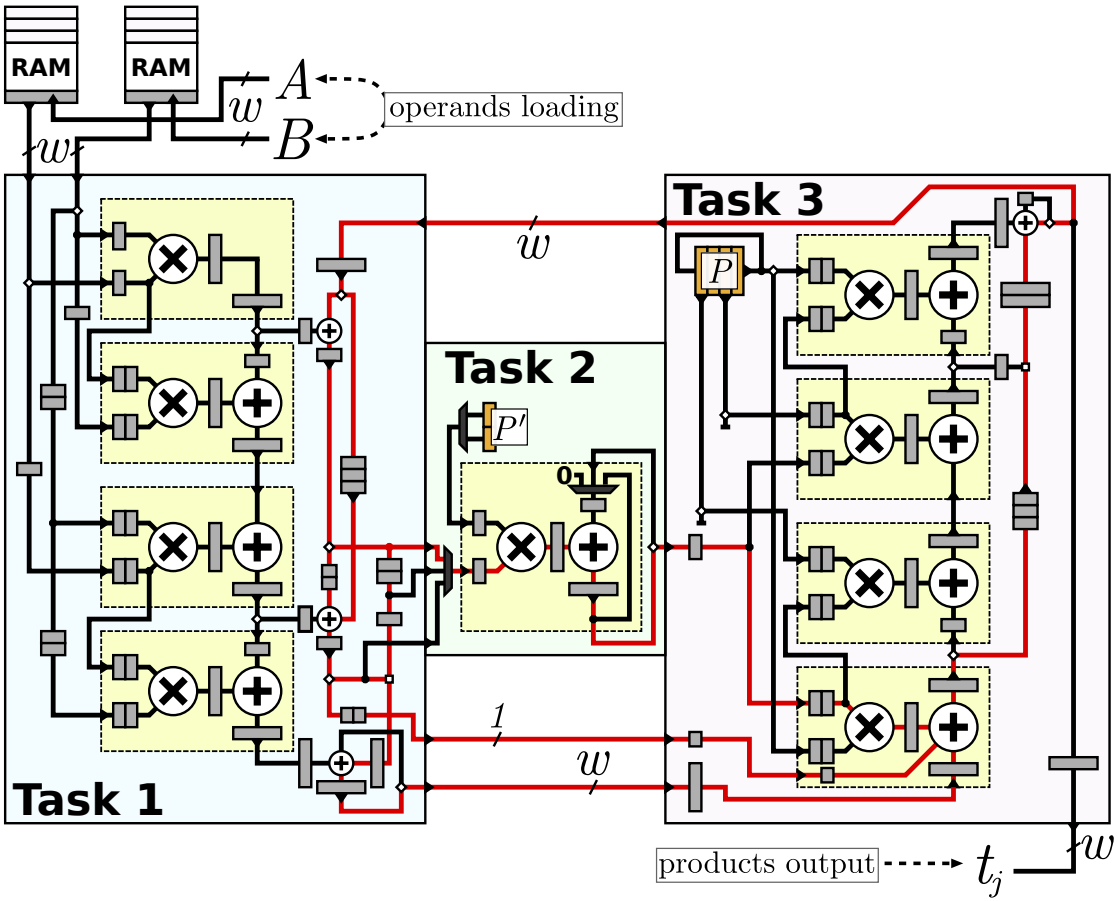
- Our solution: fill idle pipeline stages with independent MMMs

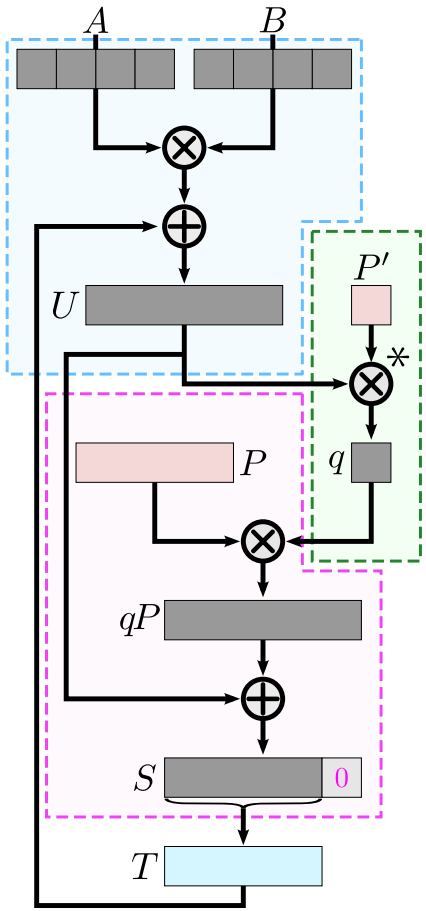


- **H**yper-**T**hreaded **M**odular **M**ultiplier
  - HTMM: physical unit computing σ independent MMMs concurrently
  - hardware ressources are shared among σ Logical Multipliers (LMs)

HTMM Architecture

- Based on 3 pipelined blocks (1 for each partial product in CIOS)
- Width of internal words fixed to $w = 34$ bits $\rightarrow$ only 9 DSP slices
- 3 to 4 stages in DSP slices to reach high frequencies





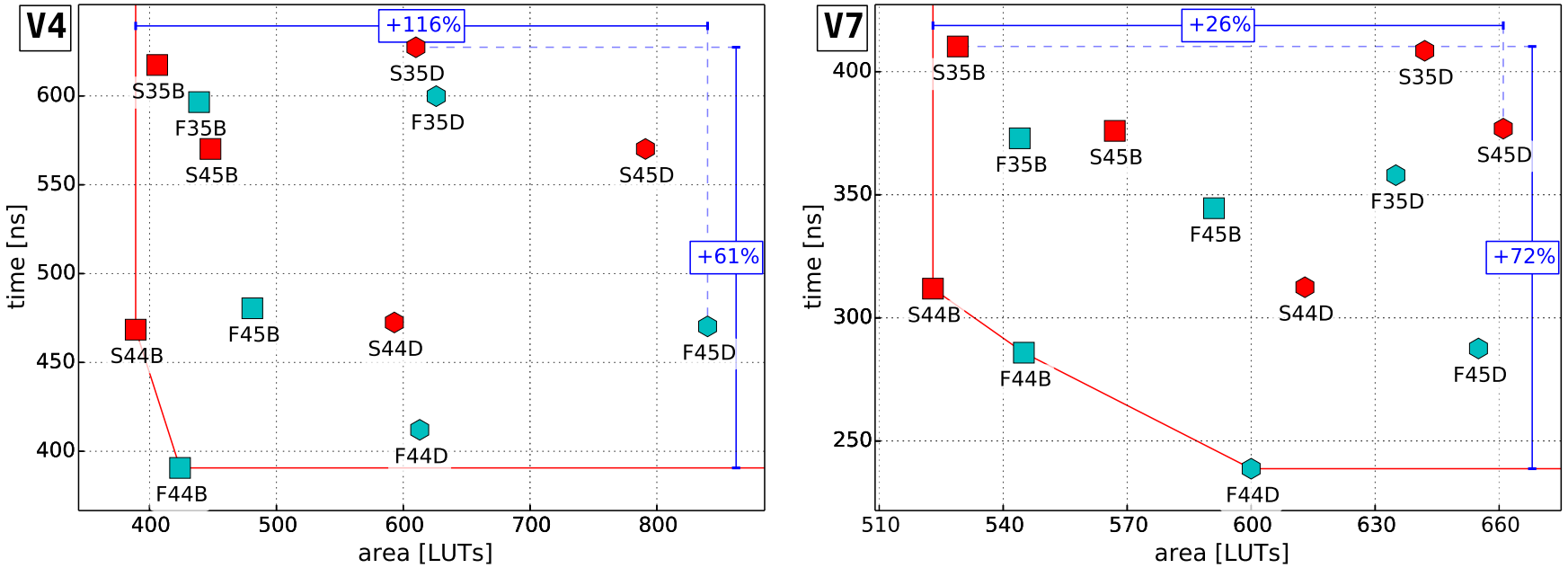
Tools for Architectures Exploration

- Many HTMM parameters to explore: size of P (*e.g.* 128 or 256 bits), w, number of LMs, configurations of memories and DSP slices, algorithmic optimizations, ...
- We designed a software HTMM generator
  - allows fast generation of VHDL codes for many HTMM specifications
  - and optimized for various FPGAs (*e.g.* pipeline config. in DSP slices)
  - available as open-source [1]
- HTMM generator also offers support for some third-party softwares
  - Xilinx tools for implementation, simulation and evaluation
  - Sage mathematics software [2] for numerical validation of HTMM

[1] HTMM generator available at `https://sourcesup.renater.fr/htmm/`

[2] available as open-source at `http://www.sagemath.org/`

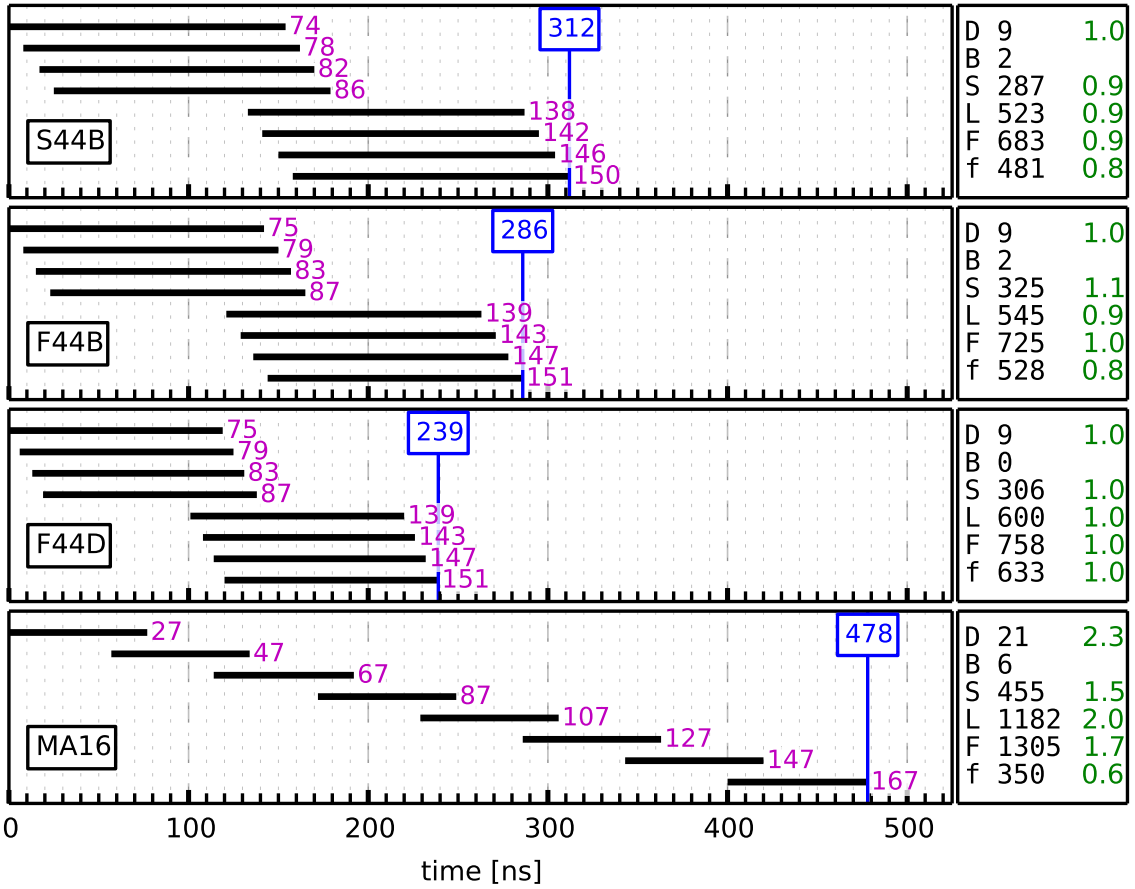
Exploration of 128 bits HTMMs on Virtex-4 and Virtex-7



- Wide exploration space of solutions for time *vs.* area tradeoffs
- Not a lot a "best" solutions (on Pareto fronts)
- Tradeoffs and "best" solutions depend on FPGA

Comparison with 128b MMM from Ma *et al.* [4]

MA16: reimplementation of multiplier from [4] for 128 bits on Virtex-7



- HTMM is smaller and faster than MA16
- HTMM reaches max. frequencies of DSP slices / BRAMs

Summary

Hyperelliptic Curves and Operations for Cryptography

- Hyperelliptic curve: points with coordinates verifing a given equation
  - for HECC, points coordinates are in $\mathrm{GF}(P)$
  - only secure curves with good properties for crypto are used in HECC
- Main curve operation: **scalar multiplication** $[k]\mathcal{P}$
  - corresponds to adding k times a point $\mathcal{P}$ of curve to itself
  - involves many arithmetic operations on coordinates $\rightarrow$ very costly
    e.g. ~ 8000 MMs for 256-bit k
- $\mathcal{P}$ is public but k **is the private key**
  - the value of k must remain secret during computations of $[k]\mathcal{P}$
  - need robust algorithms and implementations to protect k against physical attacks, *e.g.* **SPA** (Simple Power Analysis)

Scalar Multiplication Algorithms (for μKummer)

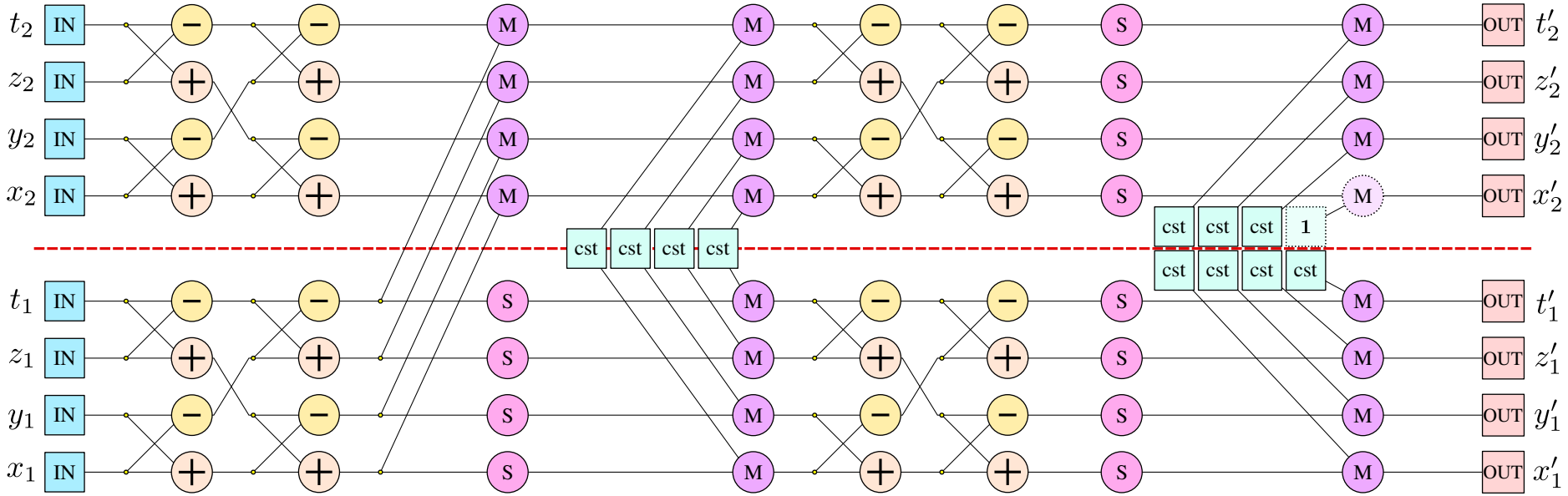
Require: n_k-bit scalar $k = \sum_{i=0}^{n_k-1} 2^i k_i$, point $\mathcal{P}$, $cst \in \mathrm{GF}(P)^4$
Ensure: $\mathcal{V}_1 = [k]\mathcal{P}$, $\mathcal{V}_2 = [k+1]\mathcal{P}$

$\mathcal{V}_1 \leftarrow cst$
$\mathcal{V}_2 \leftarrow \mathcal{P}$
for $i = n_k - 1$ **downto** 0 **do**
  $(\mathcal{V}_1, \mathcal{V}_2) \leftarrow \texttt{CSWAP}(k_i, (\mathcal{V}_1, \mathcal{V}_2))$
  $(\mathcal{V}_1, \mathcal{V}_2) \leftarrow \texttt{xDBLADD}(\mathcal{V}_1, \mathcal{V}_2, \mathcal{P})$
  $(\mathcal{V}_1, \mathcal{V}_2) \leftarrow \texttt{CSWAP}(k_i, (\mathcal{V}_1, \mathcal{V}_2))$
end for
return $(\mathcal{V}_1, \mathcal{V}_2)$

$\texttt{CSWAP}(k_i, (\mathcal{P}_1, \mathcal{P}_2))$ returns $(\mathcal{P}_1, \mathcal{P}_2)$ if $k_i = 0$, else $(\mathcal{P}_2, \mathcal{P}_1)$

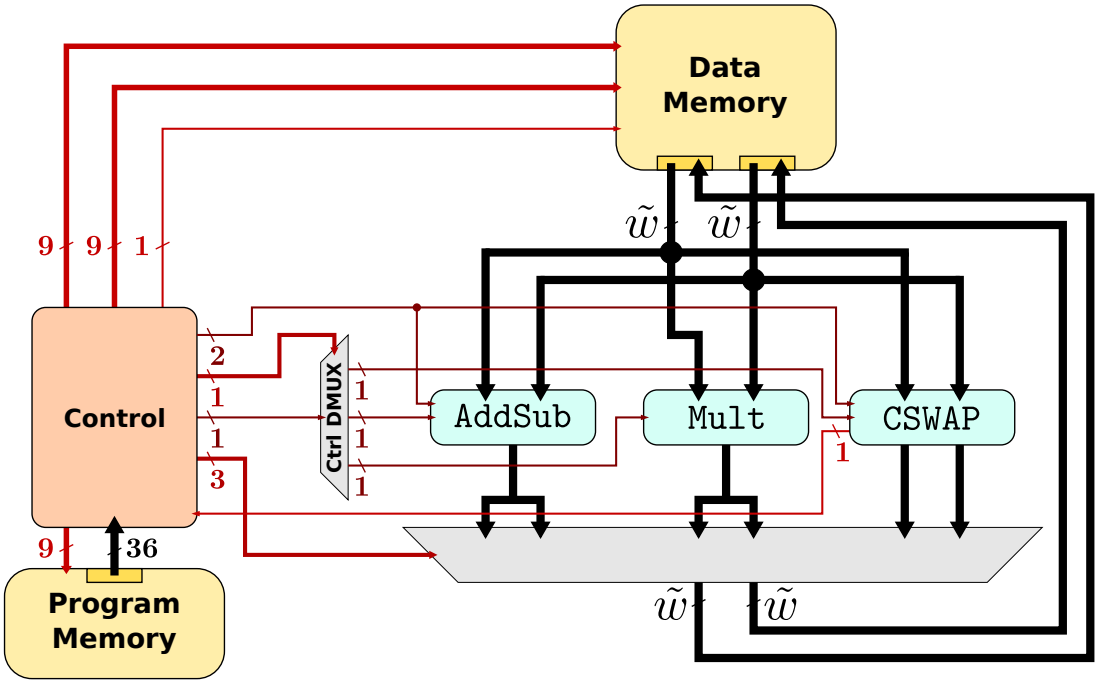
- Constant time and uniform operations (independent from k_i bits)
- CSWAP: very simple but involves secret bits: must be protected

xDBLADD Operation



- Complex operation based on 32 MM (M/S) and 32 modular add/sub
- Regular patterns of 8 independant operations → internal parallelism

Basic Cryptoprocessor Architecture



areas not to scale

- arithmetic units
- data memory
- interconnect
- program memory
- central control unit

- Various architecture parameters: number of units, width $\tilde{w}$, architecture topology, ...
- Full description of many cryptoprocessors in VHDL is not feasible
  - time consuming and validation requires heavy simulations

Units Library in VHDL

- Available units
  - $\mathrm{GF}(P)$ adders and subtractors (with various $\tilde{w}$)
  - HTMMs
  - data memories (with various $\tilde{w}$)

- **Fully described**, **implemented** and **validated in VHDL**
  - behavior is known exactly at each clock cycle (CABA[3])
  - hardware area cost for each unit is perfectly known for various FPGAs

- Implementation results form a small database

[3] Cycle-Accurate Bit-Accurate

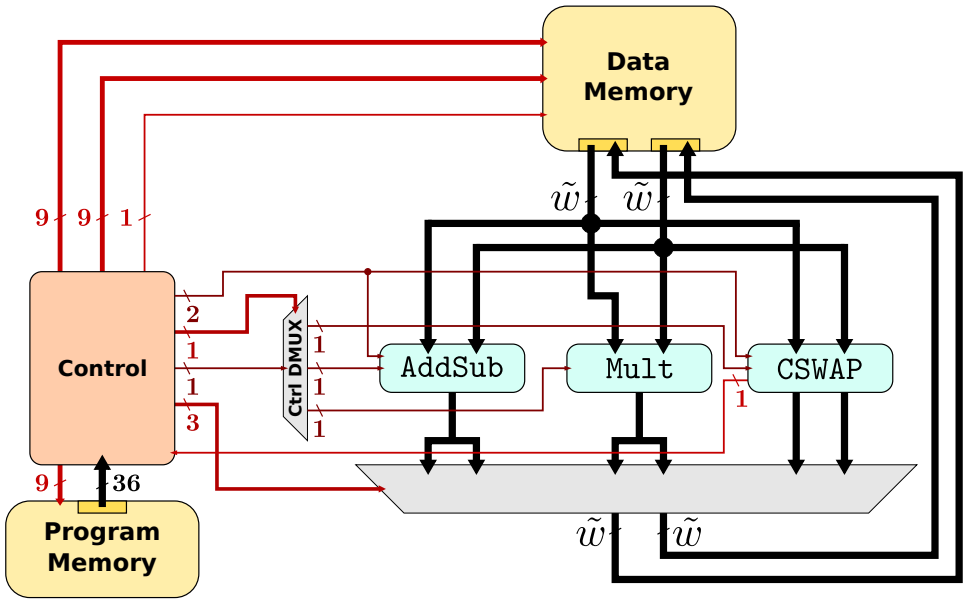
CCABA Exploration Tool

- High-level architectures modeled in **CCABA**
  - CCABA: Critical CABA[4]
  - only critical cycles and signals at architecture level are CABA
    e.g. units I/Os and control
- CCABA model is close to TLM[5] adapted for asymmetric crypto. applications
- **CCABA simulator** for **fast validation** of architectures models
- **Exploration tool** for **fast evaluation** of many architectures
  - performances in clock cycles known exactly from CCABA simulations
  - accurate area estimation based on units library database results

[4] Cycle-Accurate Bit-Accurate
[5] Transaction Level Modeling

Architectures Implemention and Validation in VHDL

- Most interesting architectures have been fully described in VHDL
  - **A2**: small architecture with 1 Mem, 1 AddSub, 1 Mult

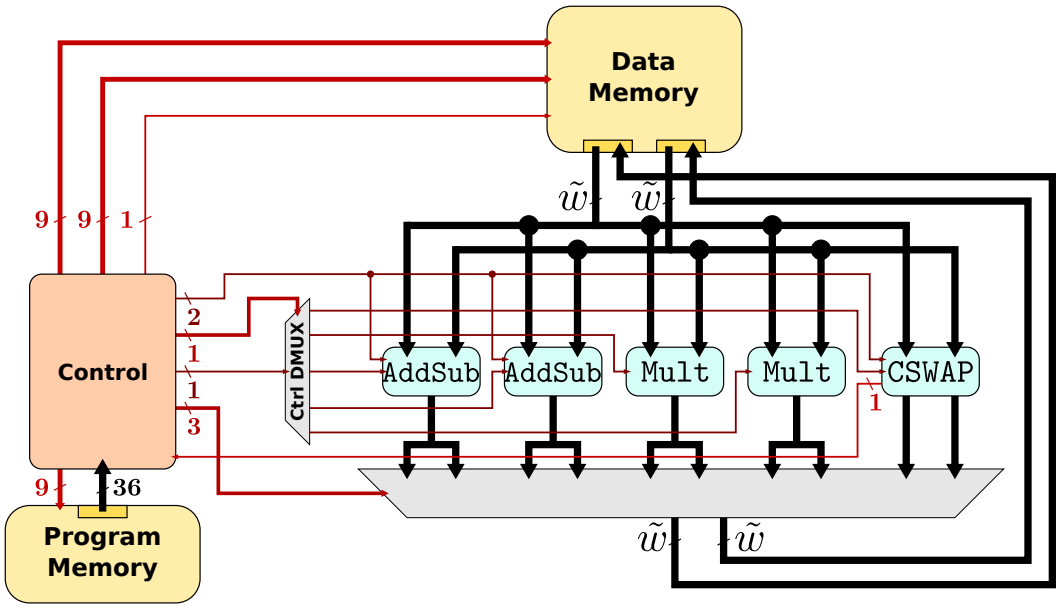


areas not to scale

- Different versions of memories/interconnect with $\tilde{w} \in \{34, 68, 136\}$
  - complete VHDL description of control for each $\tilde{w}$
  - implemented and validated on Virtex-4/5, Spartan-6 and Zynq-7

Architectures Implemention and Validation in VHDL

- Most interesting architectures have been fully described in VHDL
  - A2: small architecture with 1 Mem, 1 AddSub, 1 Mult
  - **A3**: big architecture with 1 Mem, 2 AddSub, 2 Mult

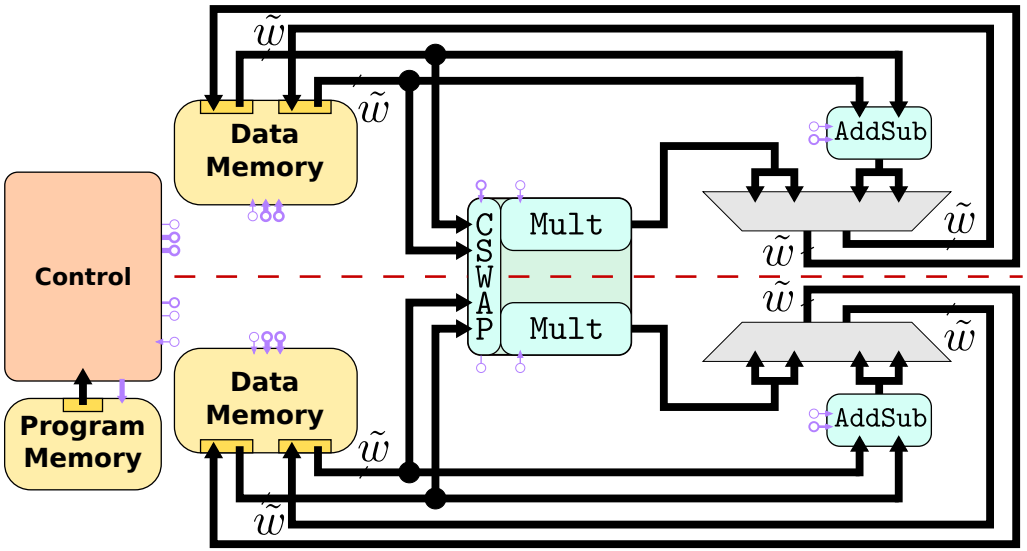


areas not to scale

- Different versions of memories/interconnect with $\tilde{w} \in \{34, 68, 136\}$
  - complete VHDL description of control for each $\tilde{w}$
  - implemented and validated on Virtex-4/5, Spartan-6 and Zynq-7

Architectures Implemention and Validation in VHDL

- Most interesting architectures have been fully described in VHDL
  - A2: small architecture with 1 Mem, 1 AddSub, 1 Mult
  - A3: big architecture with 1 Mem, 2 AddSub, 2 Mult
  - **A4**: big clustered architecture with 2 Mem, 2 AddSub, 2 Mult



areas not to scale

- Different versions of memories/interconnect with $\tilde{w} \in \{34, 68, 136\}$
  - complete VHDL description of control for each $\tilde{w}$
  - implemented and validated on Virtex-4/5, Spartan-6 and Zynq-7

Implementation Results for Best Cryptoprocessors

FPGA	archi.	$\tilde{w}$ bits	LUT	FF	logic slices	DSP slices	BRAM	freq. MHz	time $[k]\mathcal{P}$ ms
Virtex-4	A2	34	863	1689	1081	9	4	327	0.54
	A4	34	1699	3255	2447	18	7	328	0.39
	A3	136	3959	5251	3492	18	9	290	0.37
Virtex-5	A2	34	783	1653	558	9	4	386	0.45
	A4	34	1413	3182	1019	18	7	378	0.34
	A3	136	2658	5170	1657	18	9	356	0.30
Spartan-6	A2	34	911	1619	382	9	4	298	0.59
	A4	34	1565	3120	809	18	7	276	0.46
	A3	136	3128	5040	1182	18	9	238	0.45
Zynq-7	A2	34	855	1619	463	9	4	347	0.50
	A4	34	1475	3020	747	18	7	360	0.36
	A3	136	3147	5033	1143	18	9	322	0.33

- $\tilde{w} = 68$ bits is not interesting in our architectures
- No best solution but various interesting time *vs.* area tradoffs

Comparisons with Best State of the Art Cryptoprocessors

- Ma13: **ECC** processor with generic primes by Ma *et al.* (2013) [4]
- Kop18a: μKummer-based **HECC** processor with very specific prime by Koppermann *et al.* (2018) [3]

FPGA	archi.	$\tilde{w}$ bits	LUT	FF	logic slices	DSP slices	BRAM	freq. MHz	time $[k]\mathcal{P}$ ms
Virtex-5	A2	34	783	1653	558	9	4	386	0.45
	A4	34	1370	2953	1013	18	7	358	0.40
	A3	136	2737	4978	1594	18	9	348	0.34
	Ma13	**336**	**4177**	**4792**	**1725**	**37**	**10**	**291**	**0.38**
Zynq-7	A2	34	855	1619	463	9	4	347	0.50
	A4	34	1475	3020	747	18	7	360	0.39
	A3	136	3147	5033	1143	18	9	322	0.37
	Kop18a	**127**	**8764**	**6852**	**2657**	**49**	-	**139**	**0.08**

Summary

Conclusion and Perspectives

- HTMM: flexible operators for Montgomery modular multiplication
  - finely pipelined to compute several MMMs at the same time
  - 128-bit HTMM is 2 $\times$ faster and smaller than best state of the art
  - HTMM generator available online as open-source
- Flexible HECC cryptoprocessors and exploration tools
  - TLM-inspired CCABA model and tools to explore architectures
  - evaluation of architectures parameters impact on time *vs.* area tradeoffs
  - prime P and curve parameters can be modified at run time
- HECC is more efficient than ECC in hardware

Perspectives

- Evaluate robustness of accelerators against physical attacks
- Explore other types of architectures (*e.g.* data-flow)

Ph.D. contributions I

Main contributions:

[GT18] G. Gallin and A. Tisserand.
Generation of hyper-threaded GF(P) multipliers for flexible curve based cryptography on FPGAs.
submitted to IEEE Transactions on Computers (under major revision), 2018.

[GCT17] G. Gallin, T. O. Celik, and A. Tisserand.
Architecture level optimizations for Kummer based HECC on FPGAs.
In *Proc. 18th International Conference on Cryptology in India (Indocrypt)*, December 2017.

[GT17a] G. Gallin and A. Tisserand.
Hyper-threaded multiplier for HECC.
In *Proc. IEEE Asilomar Conference on Signals, Systems and Computers*, October 2017.

Other conferences and workshops:

[GT17b] G. Gallin and A. Tisserand.
Architecture level optimizations for Kummer based HECC on FPGAs.
15th International Workshop on cryptographic architectures embedded in logic devices (CryptArchi), June 2017.

[GVT15a] G. Gallin, N. Veyrat-Charvillon, and A. Tisserand.
Experimental comparison of crypto-processors architectures for elliptic and hyper-elliptic curves cryptography.
13th International Workshop on cryptographic architectures embedded in logic devices (CryptArchi), June 2015.

[GVT15b] G. Gallin, N. Veyrat-Charvillon, and A. Tisserand.
Comparaison expérimentale d'architectures de crypto-processeurs pour courbes elliptiques et hyper-elliptiques.
In *Proc. Conférence nationale d'informatique en Parallélisme, Architecture et Système (Compas)*, June 2015.
best paper award for computer architecture track

Ph.D. contributions II

Other talks and posters:

[Gal17] G. Gallin.
Architectures matérielles pour la cryptographie sur courbes hyper-elliptiques.
Séminaire sécurité des systèmes électroniques embarqués DGA – IRISA, December 2017. [inv. talk]

[GT17b] G. Gallin and A. Tisserand.
Finite field multiplier architectures for hyper-elliptic curve cryptography.
12ème Colloque national du GDR SOC2, June 2017. [poster]

[GT17c] G. Gallin and A. Tisserand.
Hardware architectures exploration for hyper-elliptic curve cryptography.
6ème Colloque national Crypto'Puces, June 2017. [talk]

[GT16] G. Gallin and A. Tisserand.
Hardware and arithmetic for hyperelliptic curves cryptography.
Colloque annuel international du labex CominLabs, November 2016. [poster]

[GT15] G. Gallin and A. Tisserand.
Comparaison expérimentale d'architectures de crypto-processeurs pour courbes elliptiques et hyper-elliptiques.
Journées nationales Codage et Cryptographie (JC2), October 2015. [talk]

[GVT15c] G. Gallin, N. Veyrat-Charvillon, and A. Tisserand.
Hardware and arithmetic for hyperelliptic curves cryptography.
Rencontres nationales Arithmétiques de l'Informatique Mathématique (RAIM), 2015. [poster]

[GVT15d] G. Gallin, N. Veyrat-Charvillon, and A. Tisserand.
Hardware and arithmetic for hyperelliptic curves cryptography.
Colloque annuel international du labex CominLabs, March 2015. [poster]

This work is funded by **H-A-H** project

Thank you for your attention

References

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

[2] C. K. Koc, T. Acar, and B. S. Kaliski, Jr.
Analyzing and comparing Montgomery multiplication algorithms.
IEEE Micro, 16(3):26–33, June 1996.

[3] P. Koppermann, F. De Santis, J. Heyszl, and G. Sigl.
Fast FPGA implementations of Diffie-Hellman on the Kummer surface of a genus-2 curve.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1):1–17, 2018.

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

[5] P. L. Montgomery.
Modular multiplication without trial division.
Math. of Comp., 44(170):519–521, April 1985.

[6] J. Renes, P. Schwabe, B. Smith, and L. Batina.
μKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers.
In *Proc. 18th International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9813, pages 301–320, August 2016.