Elliptic Curve Factorization Method: Towards Better Exploitation of Reconfigurable Hardware

Special-purpose Hardware for Attacking Cryptographic Systems (SHARCS)
September 2007

Giacomo de Meulenaer, François Gosset,
Guerric Meurice de Dormale*, Jean-Jacques Quisquater

Université catholique de Louvain
UCL/DICE Crypto Group
Belgium

This talk is based on our FCCM'07 paper

Outline

- Basics & motivations
- Previous works & proposal
- Curve operations
- Architecture
- Results
- Conclusion

Cryptography & PKC

- Secure communication over unsecured channel

  ➔ Cryptography

- Why Public Key Cryptography?
  - Digital signature
  - Key agreement
  - Public key encryption

  ➔ RSA

PKC: RSA

- RSA scheme
  - Rivest, Shamir and Adleman (77)
  - Most deployed system
- Security parameters
  - Hard problem: integer factorization
  - Best factorization algorithm: NFS (Pollard 91)
  - Minimum key size: 1024-bit

Why attacking systems?

- Feasibility

  Cost reachable for a given adversary?

  ➔ Security of a given set of parameters

- Forecast

  How long data will remain secure?

- Means
  - **Hardware**-based cost assessment
  - **Cost-effective** algorithms and architectures

Number Field Sieve

- One hardware proposal: SHARK (05)
- Relation collection step

  ➔ factorization of 10^{14} 125-bit numbers

- Elliptic Curve Method (Lenstra 87)
  - Highly regular
  - Not too I/O intensive
  - Many parallel computations

Outline

- Basics & motivations
- ***Previous works & proposal***
- Curve operations
- Architecture
- Results
- Conclusion

Previous works

- Software
  - Loria team: GMP-ECM
- Hardware
  - ASIC & FPGA: Simka *et al.*, *FCCM'05, IEE'05*
  - FPGA: Gaj *et al.*, *CHES'06*
    - → Bit-serial by parallel
    - → General purpose logic

Our proposal

- Today's FPGAs: dedicated multipliers
  - Virtex-4SX: 128 → 512 DSP48s
  - Spartan-3: max 104 Multipliers
  - Spartan-3ADSP: max 126 DSP48s
- High-performance FPGAs
  - Virtex-4SX: best speed × #mult / $
- Our choice

  Use dedicated multipliers of high-perf. FPGAs

Outline

- Basics & motivations
- Previous works & proposal
- ***Curve operations (modular arith.)***
- Architecture
- Results
- Conclusion

Elliptic curve arithmetic (Phase1)

- Scalar multiplication $k \cdot P$, main loop:

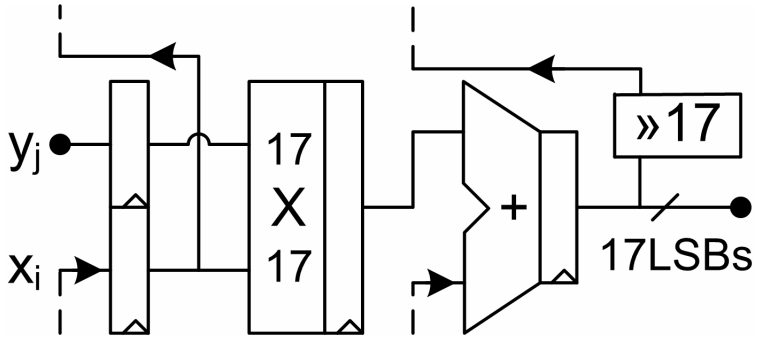
$M1 \leftarrow (x_P - z_P)^2$	$M3 \leftarrow (x_P - z_P) \times (x_Q + z_Q)$
$M2 \leftarrow (x_P + z_P)^2$	$M4 \leftarrow (x_P + z_P) \times (x_Q - z_Q)$
$M5 \leftarrow M1 \times M2$	$M7 \leftarrow (M3 + M4)^2$
$M6 \leftarrow (M2 - M1) \times a_{24}$	$M8 \leftarrow (M3 - M4)^2$
$M9 \leftarrow x_{P-Q} \times M8$	$M10 \leftarrow (M2 - M1) \times (M1 + M6)$

- Highly regular and simple
- Main operation: $(A \pm B) \times (C \pm D)$
- 4 Sqrs and 6 Mults

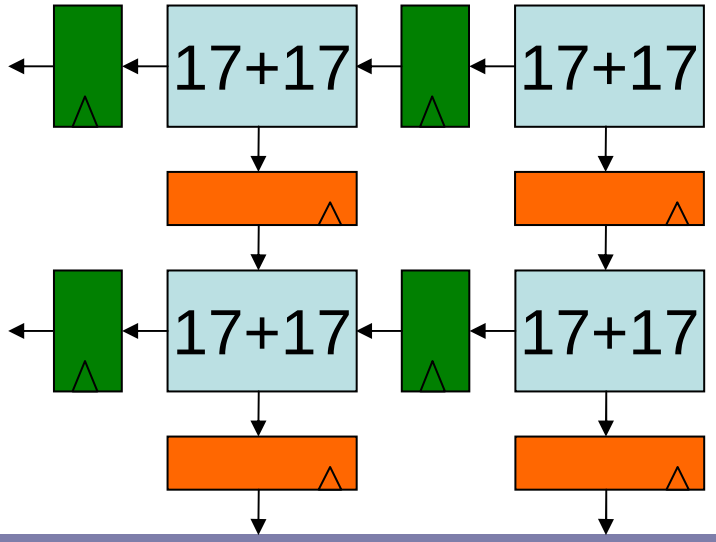
Arithmetic circuits

- Highly pipelined
- Digit size: 17 bits

DSP48:

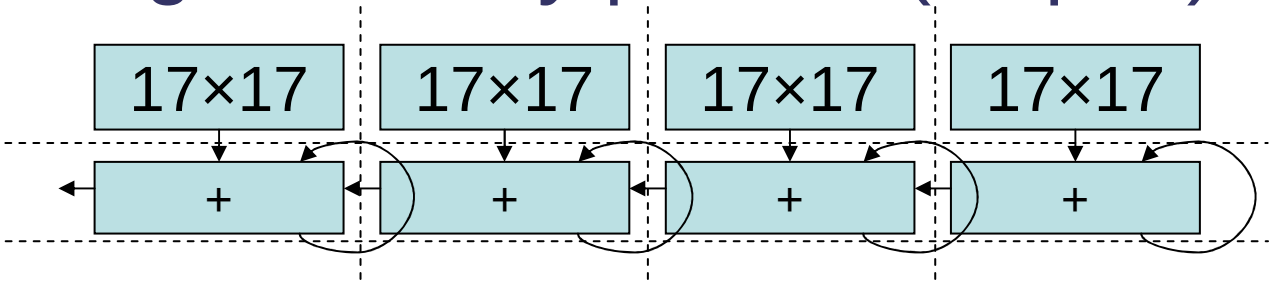


- Pipelined ripple adders
  - Horizontally (carry)
  - Vertically

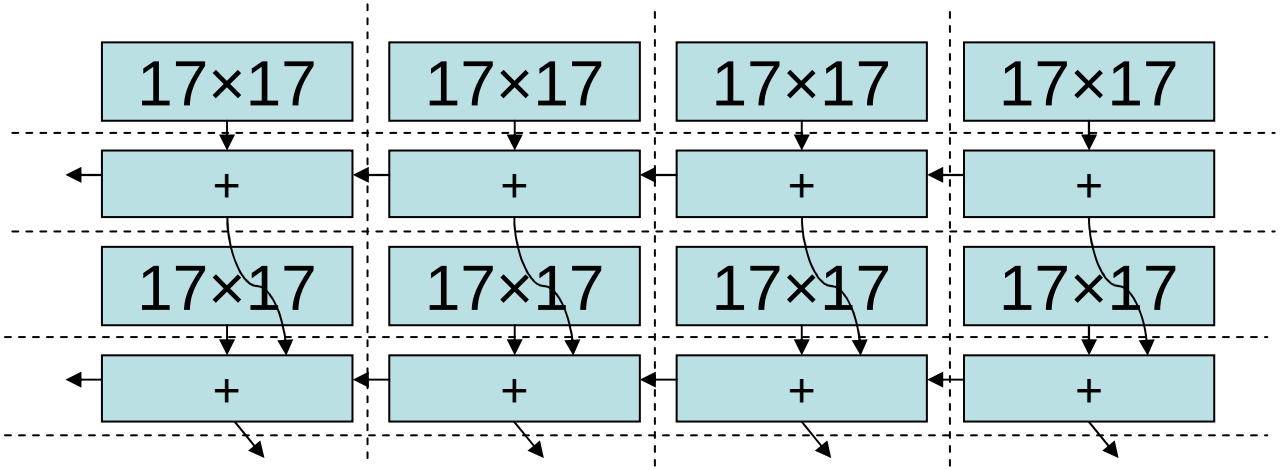


Modular multiplier

- Digit-serial by parallel (looped)



- Parallel (fully unrolled)



Modular multiplier cont'd

- Digit-serial
  - – General circuit
  - + Scalable area: d (=8) Mults
- Parallel → Potentially best AT product
  - + Data-driven, fully specialized
  - + Simple data load/unload
  - – Flexibility: $2d^2$ Mults (V4 SX25 ok)

Montgomery modular mult

- Only trivial divisions
- Work on LSBs
- Montgomery domain & R constant

  In: $x' = x \text{ R mod } n$, $y' = y \text{ R mod } n$

  Out: $x'y' \text{ R}^{-1} \text{ mod } n = xy \text{ R mod } n$

For i=0 **to** 7 **do**
  $u_i \leftarrow (a_0 + x_i \times y_0) \times n' \bmod 2^{17}$
  $A \leftarrow (A + x_i \times y + u_i \times n) / 2^{17}$ $(A < 2n)$
If $A \geq n$ **then**
  $A \leftarrow A - n$ $(A < n)$
Return(A)

Final subtraction:

- Extra computation
- not LSBs anymore

Improved Montgomery mult

For i=0 **to** 6 **do**

$u_i \leftarrow (a_0 + x_i \times y_0) \bmod 2^{17}$

$A \leftarrow (A + x_i \times y + u_i \times ns) / 2^{17}$ $\quad (A < 2^{17}n + n)$

$u_7 \leftarrow (a_0 + x_7 \times y_0) \times n' \bmod 2^{17}$

$A \leftarrow (A + x_7 \times y + u_7 \times n) / 2^{17}$ $\quad (A < 3n)$

If $A \bmod 2 = 1$ **then**

$A \leftarrow A + n$ $\quad (A < 4n)$

Return$(A/2)$ $\quad (A < 2n)$

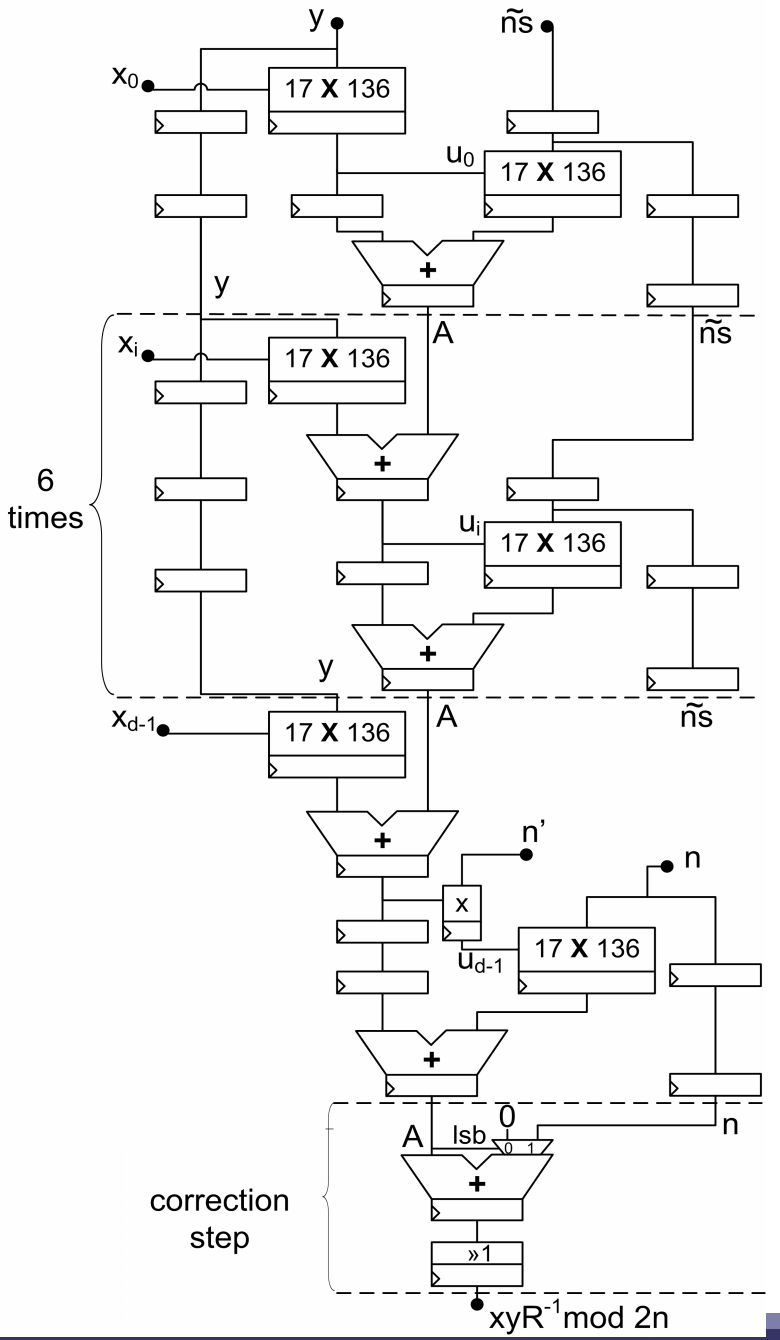
$A \leftarrow (A + x_i \times y) / 2^{17} + u_i \times (ns_{7..1} + 1)$

- Final Subtraction: mod n → mod $2n$
- Save 7 Mults: $ns = n \times n'$
- No increase of the size of IN/OUT

Modular multiplier

- 1st round
- Main rounds
- Last round
- Correction step

- Differences with serial:
  - Last iteration
  - Correction step



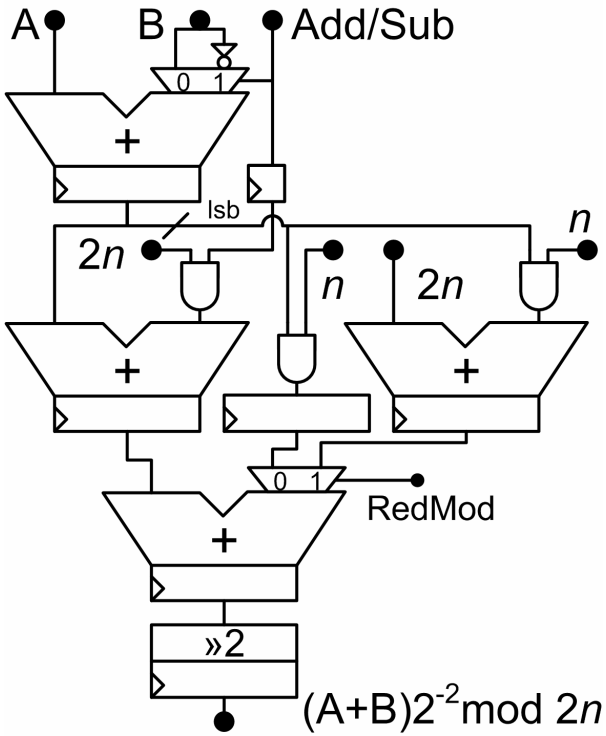
Modular adder/subtracter

A ± B mod 2n, A,B in [0,2n[

No comparison allowed!

- Make the result positive
- Divide it by 4 mod n

- Add: (A + B + qn)/4 q in [0,4[
- Sub: (A – B + 2n + qn)/4
- Result: (A ± B) 2^{-2} mod 2n



Compatibility with Mult

- Result: $(A \pm B)\, 2^{-2} \bmod 2n$

  ➔ pre-multiply inputs by $2^4 \bmod n$

  $(A2^4 \pm B2^4)/4 \times (A'2^4 \pm B'2^4)/4$

  $= C2^2 \times C'2^2 = CC'\, 2^4$ ok!

Outline

- Basics & motivations
- Previous works & proposal
- Curve operations
- ***Architecture***
- Results
- Conclusion

ECM architecture

- Client: FPGAs (VHDL)

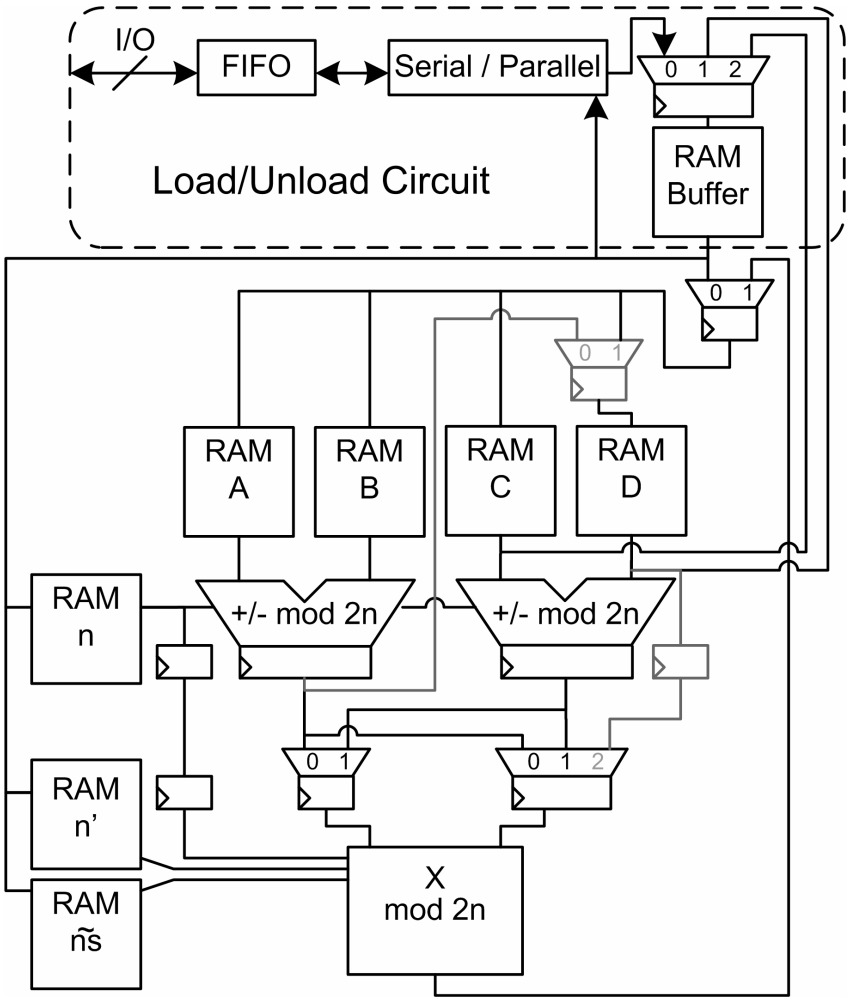
  $(A \pm B) \times (C \pm D)$

  → scalar mult (*k*P)

- Server: PC (C, GMP)

  → precomputations

  (Possible to move them inside the FPGA)



Outline

- Basics & motivations
- Previous works & proposal
- Curve operations
- Architecture
- ***Results***
- Conclusion

Hardware arithmetic results

135-bit Designs	Frequ. [Mhz]	Area [Slices]	Through. [Gbit/s]	DSP48
Mult Mod	220	3405 (33%)	30	128 (100%)
Add/Sub	245	446 (4%)	33	0

- Virtex-4 SX25-10
- No floor planning
- 100% DSP48 utilization
- Routing!

ECM results

Computation, 135-bit	Gaj *et al.* (ches'06)	**Our design**
FPGA	XC3S5000-5	**XC4VSX25-10**
Slices	32,200 (100%)	**6006 (60%) (+128 DSP48)**
bRAMS	28 (30%)	**31 (25%)**
Max frequency	100 MHz	**220 MHz**
# Phase1/sec.	1148	**16,000**
FPGA price (quantity)	$130 (10,000)	**$116 (2500)**

- Gaj *et al.*: extrapolation (linear)

➔ 14-fold improvement

Software results

- 125-bit inputs on a 32-bit data bus
- Our hardware: 16,000 Phase1/s
  - Send: 15 Mb/s (or at best 2Mb/s)
  - Receive: 2 Mb/s
- Pentium4 3.2 GHz WinXP → only 4 FPGAs

→ Precomputations should be done in hardware in the future

Cost assessment

- SHARK in 1 year

  → factorization of 5.5 10^6 numbers/second
- Gaj *et al.* (ches'06)

  $25.8M for 200k Spartan-3 ($130 each)
- Our results

  $1.6M for 13k Virtex-4 ($116 each)
- Virtex-4 based COPACOBANA

  $3.2M for the 225 engines

Conclusion & further works

- ECM on FPGA as support for NFS
- Embedded Mults make the difference
  ➔ 15-fold improvement
- High-perf FPGAs → best Mult throughput/$
- Compare with digit-serial architecture
- Phase2
- Include precomputation in hardware

Questions ?

http://www.dice.ucl.ac.be/crypto