

An Improved Architecture of a Hardware Accelerator for Factoring Integers with Elliptic Curve Method

michal.andrzejczak@wat.edu.pl

Faculty of Cybernetics
Military University of Technology in Warsaw

CryptArchi, 2018

- 1 Introduction to ECM
- 2 Initial design
- 3 Architecture analysis and improvement
- 4 Results

- Proposed by Lenstra in 80's
- Original work called "Phase one"
- Part of General Number Field Sieve
- Used for factoring numbers around 200 bits long
- low memory requirements, many computations on elliptic curves
- many improvements and extensions
- easily parallelisable
- efficient in hardware

Algorithm 1 ECM algorithm, phase 1

Require: a complex number N , random point P_0 on random elliptic curve E , integer bound B_1

Ensure: a factor of N or *fail*

1: $k \leftarrow \prod_{p \leq B_1} p^{\log_p B_1}$

2: $Q_0 \leftarrow [k]P_0$;

3: $q \leftarrow \gcd(z_{Q_0}, N)$;

4: **if** $q > 1$ **then**

5: **return** q

6: **else**

7: **return** *fail*

8: **end if**

$$\mathcal{O}(N) = e^{\sqrt{\log p \log \log p}^{\sqrt{2}+o(1)}} M(\log N)$$

Where:

- $M(\log n)$ is the complexity of point multiplication mod N
- p is the smallest factor of N

Global view of initial design

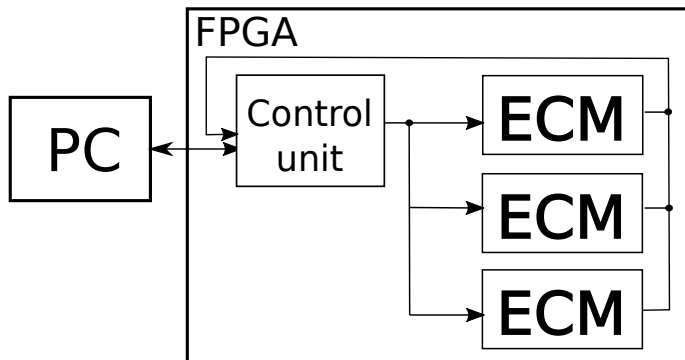


Figure: Global view of proposed solution

ECM level view

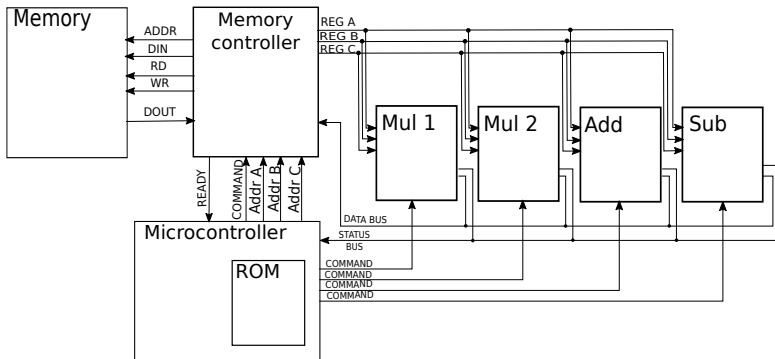


Figure: ECM level view

- execute program stored in ROM memory
- control all arithmetic units
- can execute up to 4 parallel instructions at time
- implements several state machines

Table: One step of scalar multiplication in case of $z_{P-Q} = 1$

Adder	Subtractor	Multiplier 1	Multiplier 2
$a_1 = x_P + z_P$	$s_1 = x_P - z_P$		
$a_2 = x_Q + z_Q$	$s_2 = x_Q - z_Q$	$m_1 = s_1^2$	$m_2 = a_1^2$
	$s_3 = m_2 - m_1$	$m_3 = s_1 \cdot a_2$	$m_4 = s_2 \cdot a_1$
$a_3 = m_3 + m_4$	$s_4 = m_3 - m_4$	$x_{2P} = m_1 \cdot m_2$	$m_6 = s_3 \cdot a_{24}$
$a_4 = m_1 + m_6$		$x_{P+Q} = a_3^2$	$m_8 = s_4^2$
		$z_{P+Q} = m_8 \cdot x_{P-Q}$	$z_{2P} = s_3 \cdot a_4$

Instruction set executed by controller:

- **ADD** - instruction used for addition
- **SUB** - used for subtraction
- **MULA**- multiplication by first unit
- **MULB**- multiplication by second unit
- **LOADN**- modulus read from memory

- concatenate data from memory and write to registers
- prevent from parallel data override
- implements semaphore table for full memory address space
- operates with scalable memory size

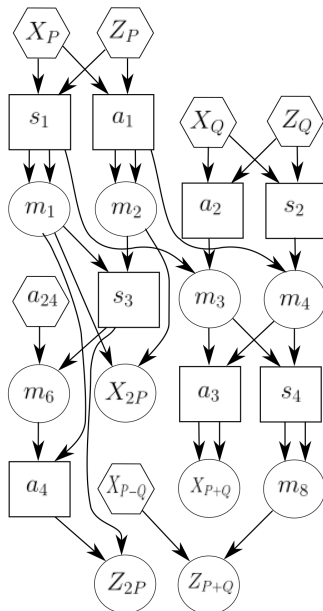
- Utilizes only logic elements
- Executes in n clock cycles
- Modular multiplication in Montgomery domain
- Based on *"An Optimized Hardware Architecture for the Montgomery Multiplication Algorithm"* by K. Gaj, M. Huang, S. Kwon, T. A. El-Ghazawi

Author:	[1]	[1]	[2]	[2]	[3]	Initial 1	Initial 2
Device:	S35000	V4LX200	V4SX35	XC4VSX35	XCV4SX25	SGX530	5CSEMA
Number length:	198 - bit	198 - bit %	202 - bit	134 - bit	135 - bit	192 - bit	192 - bit
Max. number of modules	13	24	24	24	1	96	10
Max. clock freq.	80 MHz	104 MHz	200 MHz	200 MHz	220 MHz	150 MHz	88 MHz
Clock cycles in phase 1	1 666 500	1 666 500	1 473 596	797 288	13 750	2 101 400	2 101 400
Time for phase 1	21 ms	16 ms	7,37 ms	3,99 ms	63 ms	14 ms	25 ms
Curves/sec:	624	1 448	3 240	6000	16 000	6822	418

- 1 *"Area-time efficient implementation of the elliptic curve method of factoring in reconfigurable hardware for application in the number field sieve"* by K. Gaj et al.
- 2 *"Optimized Implementation of the Elliptic Curve Factorization Method on a Highly Parallelized Hardware Cluster"* by R. Zimmermann
- 3 *"Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware"* by G. de Meulenaer et al.

- Reducing memory communication overhead
- Replacing modular multiplication modules with faster, DSP-based algorithms

Data dependency path in initial design



Data dependency path in initial design

Memory bottlenecks

- The same data loaded multiple times
- The same data loaded for different arithmetic unit
- Temporary results unnecessarily stored in memory

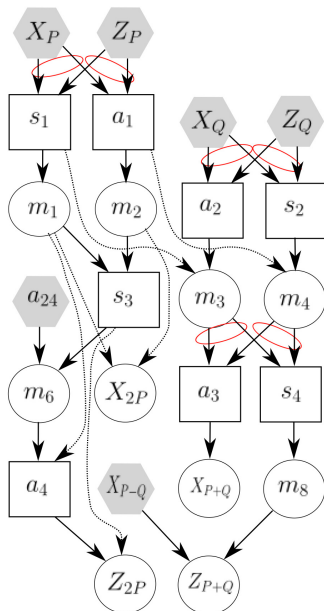
Data dependency path

Proposed improvements

- The same data loaded multiple times
-> **expanding instruction set** (opt 1)
- The same data loaded for different arithmetic unit
-> **expanding instruction set** (opt 1)
- Temporary results unnecessarily stored in memory
-> **adding additional cache registers (4)** (opt 2)

Name	Description
RLOAD	Load data from one register to another one. Can be used to load data for two registers at once
MLOAD	Memory load to one or two registers
MULA	Start multiplication in unit A
MULB	Start multiplication in unit B
ADD	Start addition
SUB	Start subtraction
WAITFOR	Waits for end of computation in selected module

Data dependency path improvements



Replacing modular multiplication unit




- New unit utilizes logic and DSP blocks for multiplication
- 3 DSP per multiplier, 6 DSP per ECM unit
- Executes in 54 clock cycles for 192 - bit numbers
- Nearly 4x faster than previously used
- Small routing overhead for small number of DSP blocks per multiplier
- Based on "*Fast Implementation of Public-Key Cryptography on a DSP*" by K. Itoh, M. Takenaka, N. Torii, S. Temma, Y. Kurihara

Comparison of improvements performance

Parameter	Initial	opt1	opt2	opt3	Stratix IV opt3
Logic:	30 060	31,394	30 982	25 566	372 951
Logic usage(%):	97 %	98 %	97 %	80 %	92 %
DSP:	-	-	-	66 (92%)	654 (63 %)
Max. clock freq.	88 MHz	88 MHz	85 MHz	90,1 MHz	151 MHz
Num. of Units	10	10	9	11	109
Clock cycles per bit	1580	1374	1215	481	481
Clock cycles in phase 1:	2 101 400	1 827 420	1 615 950	639 730	639 730
Curves/sec:	418	480	473	1546	25 557
Improvement factor:	1	1,14	1,13	3,69	3,72

Thank you!

For Further Reading I

-  K. Gaj et al., “Area-time efficient implementation of the elliptic curve method of factoring in reconfigurable hardware for application in the number field sieve,” *IEEE Transactions on Computers*, vol. 59, pp. 1264-1280, 2010/9
-  R. Zimmermann, “Optimized Implementation of the Elliptic Curve Factorization Method on a Highly Parallelized Hardware Cluster”, Master Thesis, TU Braunschweig, 2009 r.
-  G. de Meulenaer, F. Gosset, G. de Dormale, J. Quisquater, “Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware”, [15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007)], Napa, CA, 2007, pp. 197-206.

For Further Reading II

-  K. Gaj, M. Huang, S. Kwon, T. A. El-Ghazawi, “An Optimized Hardware Architecture for the Montgomery Multiplication Algorithm,” *Public Key Cryptography*, , 2008
-  M. Andrzejczak, “Koprocesor kryptograficzny wspierający faktoryzację liczb metoda krzywych eliptycznych,” [Konferencja młodych naukowców wiwat 2017, Falenty, Polska, 2017], in press.
-  K. Itoh, M. Takenaka, N. Torii, S. Temma, Y. Kurihara “Fast Implementation of Public-Key Cryptography on a DSP”, [Cryptographic Hardware and Embedded System], 2002.