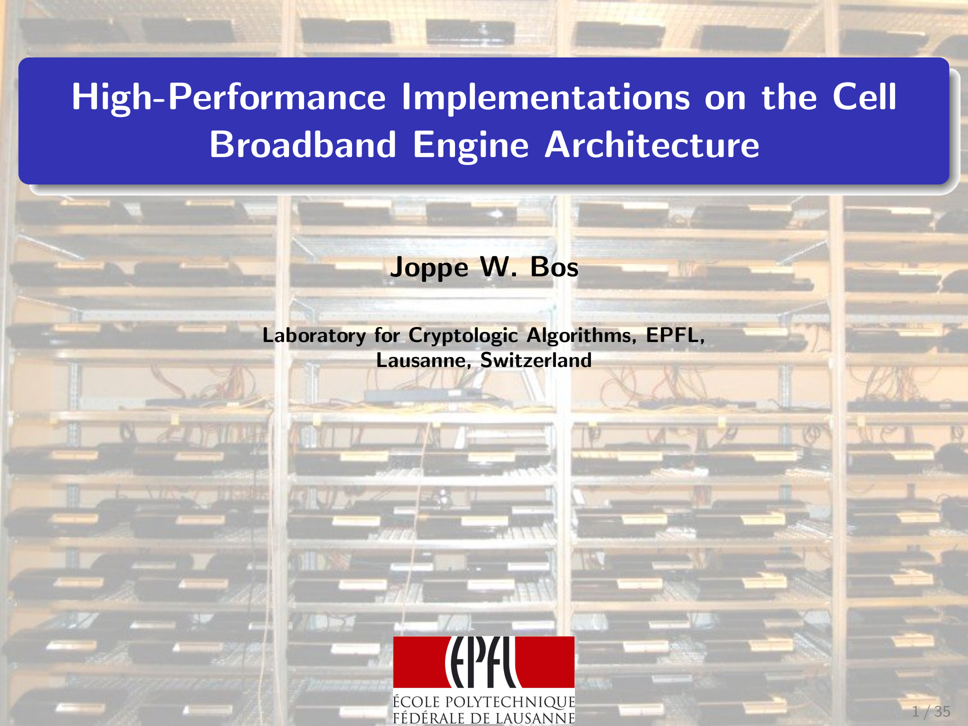


High-Performance Implementations on the Cell Broadband Engine Architecture

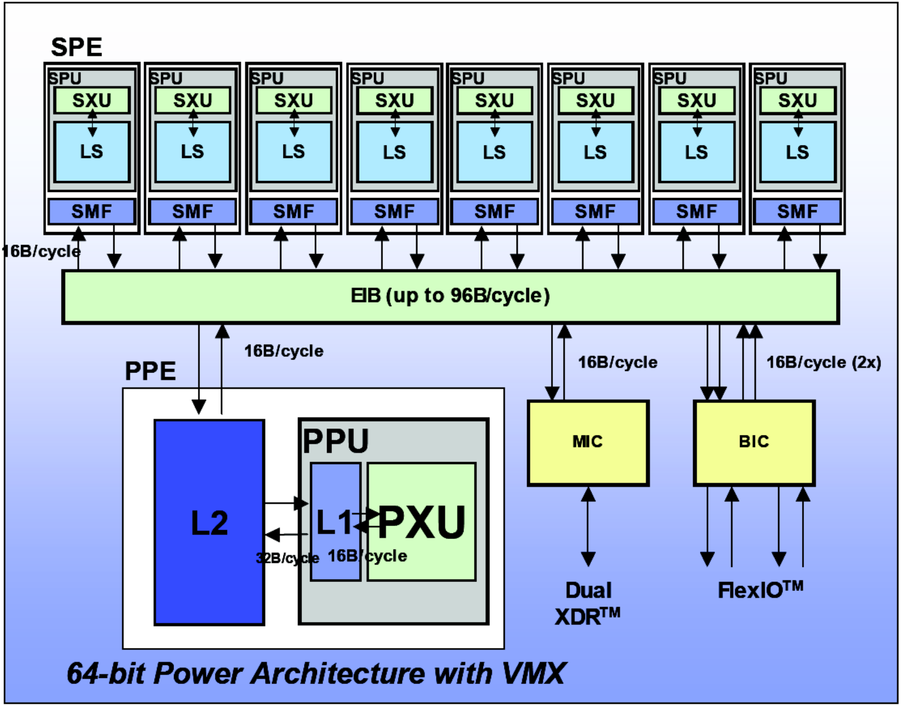
Joppe W. Bos

**Laboratory for Cryptologic Algorithms, EPFL,
Lausanne, Switzerland**

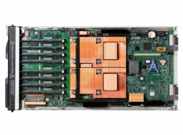
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Outline

- The Cell Broadband Engine Architecture
- Project 1: 112-bit prime field ECDLP
- Project 2: Fast arithmetic modulo a Mersenne number in ECM


SPE
SPU
SXU
LS
SMF
16B/cycle
EIB (up to 96B/cycle)
16B/cycle
PPE
16B/cycle
16B/cycle (2x)
MIC
BIC
L2
PPU
L1
PXU
32B/cycle
16B/cycle
Dual XDR™
FlexIO™
64-bit Power Architecture with VMX

Cell Availability



	PS3	PS3	PCIe	BladeServer
	slim	discontinued		QS22*
Speed	3.2GHz	3.2GHz	2.8GHz	3.2GHz
#SPEs	6	6	8	16
Memory	≈256MB	≈256MB	4GB	≤32GB
Price	$299.99	$100 – $300	≈ $8k	$10k – $14k
Power	250W	280W	210W	230W
Compatibility	PSOne	PSOne, Linux	Linux	Linux

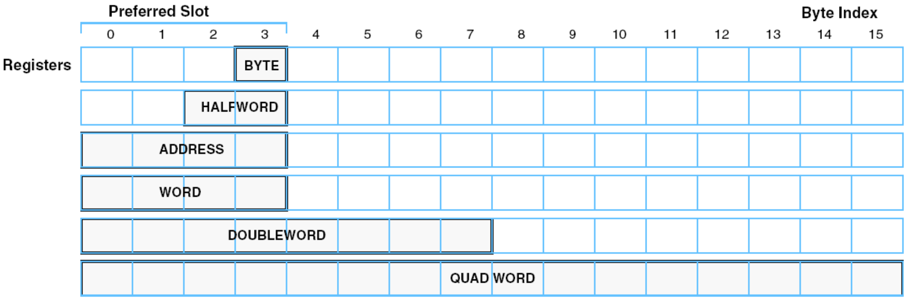
$\star$ IBM PowerXCell 8i processor, offering five times the double precision performance of the previous Cell/B.E. processor.

Cell architecture, the SPEs

The SPEs contain

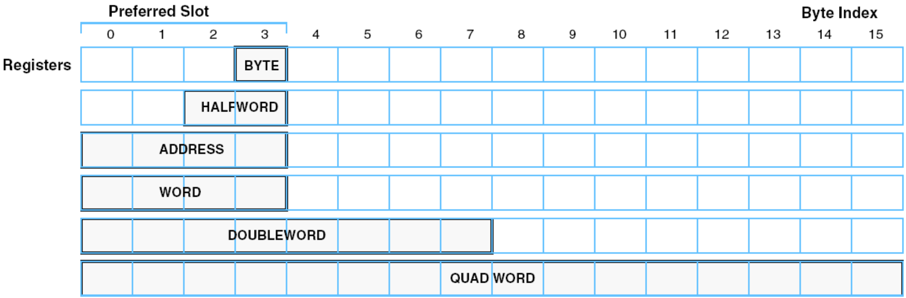
- a Synergistic Processing Unit (SPU)
  - Access to 128 registers of 128-bit
  - SIMD operations
  - Dual pipeline (odd and even)
  - In-order processor
- 256 KB of fast local memory (Local Store)
- Memory Flow Controller (MFC)
  - Direct Memory Access (DMA) controller
  - Handles synchronization operations to the other SPUs and the PPU
  - DMA transfers are independent of the SPU program execution

SPU registers



• Byte:	16 × 8-bit SIMD
• Half-word:	8 × 16-bit SIMD
• Word:	4 × 32-bit SIMD

SPU registers



- Byte: $16 \times$ 8-bit SIMD
- Half-word: $8 \times$ 16-bit SIMD
- Word: $4 \times$ 32-bit SIMD

Theoretical performance of $16 \times 3.2 \cdot 10^9 = 51.2$ billion 8-bit integer operations per second.

Special SPU instructions

All distinct binary operations $f : \{0,1\}^2 \to \{0,1\}$ are present.
Furthermore:

- shuffle bytes
- or across
- average of two vectors
- select bits
- carry/borrow generate
- multiply and add
- add/sub extended
- count leading zeros
- count ones in bytes
- gather lsb
- sum bytes
- multiply and subtract

only $16 \times 16 \to 32$-bit multiplication
but, $16 \times 16 + 32 \to 32$-bit multiply-and-add instruction

Special SPU instructions

All distinct binary operations $f : \{0,1\}^2 \rightarrow \{0,1\}$ are present.
Furthermore:

shuffle bytes
or across
average of two vectors
select bits
carry/borrow generate
multiply and add
add/sub extended
count leading zeros
count ones in bytes
gather lsb
sum bytes
multiply and subtract

only 4-way SIMD $16 \times 16 \rightarrow 32$-bit multiplication
but, 4-way SIMD $16 \times 16 + 32 \rightarrow 32$-bit multiply-and-add instruction

SPU pipelines and latencies

Instruction class	Latency	Pipeline
Load and store	6	Odd
Branch hints	15	Odd
Single-precision floating point	6	Even
Double-precision floating point	13*	Even
Floating point integer	7	Even
Shuffle	4	Odd
Simple fixed-point	2	Even
Word rotate and shift	4	Even

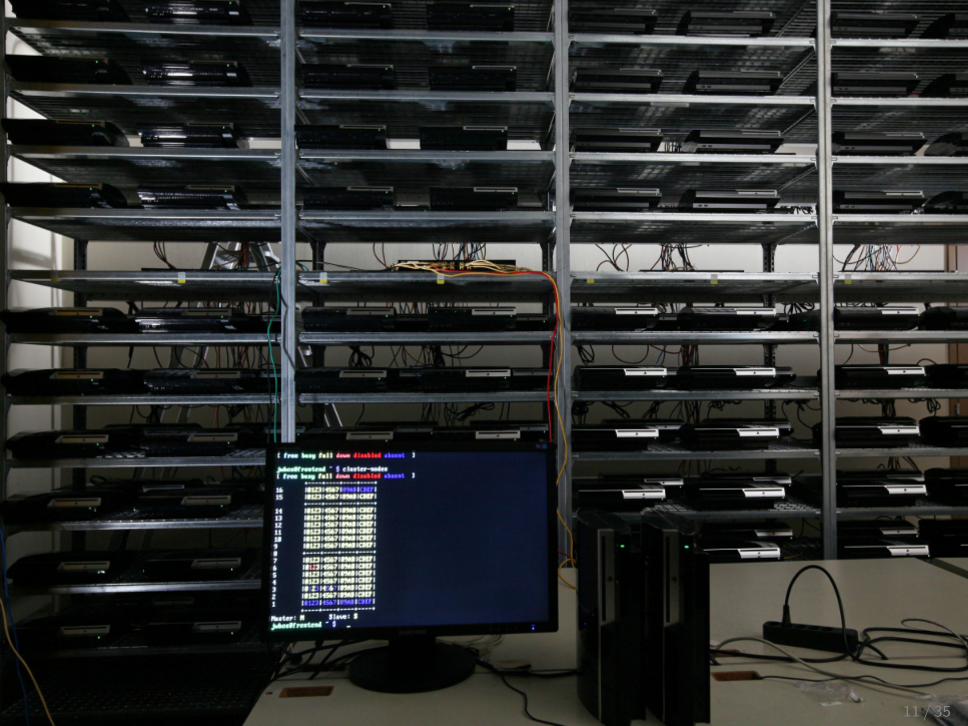
One odd and one even instruction can be dispatched per clock cycle.
Challenge to the programmer (or compiler).

Considerations

- Branching
  - No "smart" dynamic branch prediction
  - Instead "prepare-to-branch" instructions to redirect instruction prefetch to branch targets
- Memory
  - The executable **and** all data should fit in the LS
  - *Or* perform manual DMA requests to the main memory (max. 214 MB)
- Instruction set limitations
  - $16 \times 16 \rightarrow 32$ bit multipliers (4-SIMD)
- Challenge
  - One odd and one even instruction can be dispatched per clock cycle.

LACAL setup

- Physically in the cluster room: 190 PS3s
- 6 × 4 PS3s in the PlayLaB (attached to the cluster)
- 5 PS3 in our offices for programming purposes
- ⇒ 219 PS3s in total.



```
[ free busy full down disabled absent  ]

jubos@frontend ~ $ cluster-nodes
[ free busy full down disabled absent  ]
            +-----+-----+-----+-----+
16          |0123 |4567 |89AB |CDEF |
15          |0123 |4567 |89AB |CDEF |
            +-----+-----+-----+-----+
14          |0123 |4567 |89AB |CDEF |
13          |0123 |4567 |89AB |CDEF |
12          |0123 |4567 |89AB |CDEF |
11          |0123 |4567 |89AB |CDEF |
10          |0123 |4567 |89AB |CDEF |
9           |0123 |4567 |89AB |CDEF |
8           +-----+-----+-----+-----+
7           |0123 |4567 |89AB |CDEF |
6           |0123 |4567 |89AB |CDEF |
5           |0123 |4567 |89AB |CDEF |
4           |0123 |4567 |89AB |CDEF |
3           |0123 |4567 |89AB |CDEF |
2           |0123 |4567 |89AB |CDEF |
1           |0123 |4567 |89AB |CDEF |
            +-----+-----+-----+-----+
Master: M         Slave: S
jubos@frontend ~ $  _
```

Outline

- The Cell Broadband Engine Architecture
- Project 1: 112-bit prime field ECDLP
  - Joppe W. Bos, Thorsten Kleinjung, Arjen K. Lenstra, *On the Use of the Negation Map in the Pollard Rho Method*, Algorithmic Number Theory (ANTS) 2010, volume 6197 of LNCS, pages 67–83, 2010
  - Joppe W. Bos, *High-Performance Modular Multiplication on the Cell Processor*, Arithmetic of Finite Fields (WAIFI) 2010, volume 6087 of LNCS, pages 7-24, 2010
  - Joppe W. Bos, Marcelo E. Kaihara, Peter L. Montgomery, *Pollard rho on the PlayStation 3*, Handouts of SHARCS 2009, pages 35-50
- Project 2: Fast arithmetic modulo a Mersenne number in ECM

The ECDLP

The setting:

- E is an elliptic curve over $\mathbb{F}_p$ with p prime.
- $P \in E(\mathbb{F}_p)$ a point of (prime) order n.
- $Q = k \cdot P \in \langle P \rangle$.

Problem: Given E, p, n, P and Q what is k?

ECDLP Parameters

Certicom Challenge

- Solve the ECDLP for EC over $\mathbb{F}_p$ (p odd prime) and $\mathbb{F}_{2^m}$.
- 109-bit prime challenge solved in November 2002 by Chris Monico
  Required time: 4000-5000 PCs working 24/7 for one year.
- Next challenge is an EC over an 131-bit prime field

The 131-bit challenge requires 2000 times the effort of the 109-bit

ECDLP Parameters

ECC Standards

- Standard for Efficient Cryptography (SEC),
  SEC2: Recommended Elliptic Curve Domain Parameters
  Prime fields bit length: { 112, 128, 160, 192, 224, 256, 384, 521 }
- Wireless Transport Layer Security Specification
  Prime fields bit length: { 112, 160, 224 }
- Digital Signature Standard (FIPS PUB 186-3)
  Prime fields bit length: { 192, 224, 256, 384, 521 }

ECDLP Parameters

ECC Standards

- Standard for Efficient Cryptography (SEC),
  SEC2: Recommended Elliptic Curve Domain Parameters
  Prime fields bit length: { 112, 128, 160, 192, 224, 256, 384, 521 }
- Wireless Transport Layer Security Specification
  Prime fields bit length: { 112, 160, 224 }
- Digital Signature Standard (FIPS PUB 186-3)
  Prime fields bit length: { 192, 224, 256, 384, 521 }

How fast can we solve this 112-bit ECDLP?

How fast can we solve an 112-bit ECDLP?

Pollard rho

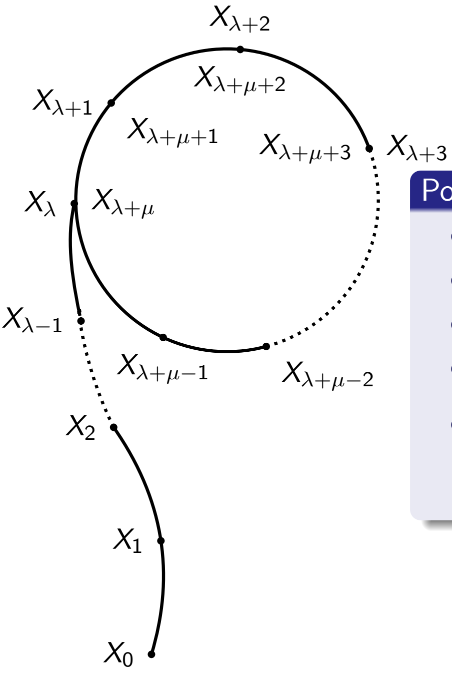
The most efficient algorithm in the literature (for generic curves) is Pollard rho. The underlying idea of this method is to search for two distinct pairs $(c_i, d_i), (c_j, d_j) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ such that

$$c_i \cdot P + d_i \cdot Q = c_j \cdot P + d_j \cdot Q$$

$$(c_i - c_j) \cdot P = (d_j - d_i) \cdot Q = (d_j - d_i)k \cdot P$$

$$k \equiv (c_i - c_j)(d_j - d_i)^{-1} \bmod n$$

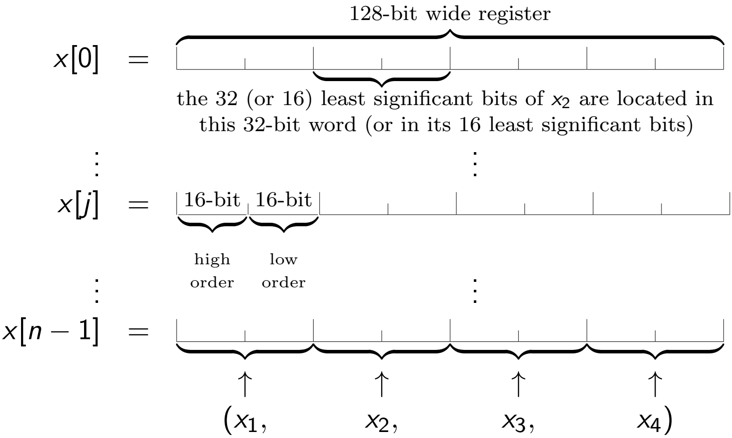
J. M. Pollard. Monte Carlo methods for index computation (mod p).*Mathematics of Computation*, 32:918-924, 1978.



Pollard Rho

- "Walk" through the set $\langle P \rangle$
- $X_i = c_i \cdot P + d_i \cdot Q$
- Iteration function $f : \langle P \rangle \to \langle P \rangle$
- This sequence eventually collides
- Expected number of steps (iterations): $\sqrt{\frac{\pi \cdot |\langle P \rangle|}{2}}$

Integer Representation


128-bit wide register
$x[0]$ =
the 32 (or 16) least significant bits of x_2 are located in this 32-bit word (or in its 16 least significant bits)
$x[j]$ =
16-bit
16-bit
high order
low order
$x[n-1]$ =
$(x_1,$
$x_2,$
$x_3,$
$x_4)$

Implementation Details

- Optimize for high-throughput, not low-latency
  - Interleave two 4-way SIMD streams
- An efficient 4-way SIMD modular inversion algorithm
- Compute on 400 curves in parallel
  - simultaneous inversion (Montgomery)
- Do not use the negation map optimization

Implementation Details

- Optimize for high-throughput, not low-latency
  - Interleave two 4-way SIMD streams
- An efficient 4-way SIMD modular inversion algorithm
- Compute on 400 curves in parallel
  - simultaneous inversion (Montgomery)
- Do not use the negation map optimization

Trade correctness for speed

- When adding points X and Y do not check if $X = Y$.
  Save code size *and* increase performance (no branching).
- Faster modular reduction which might compute the wrong result.

Special Moduli

112-bit target

The 112-bit prime p used in the target curve $E(\mathbb{F}_p)$ is

$$p = \frac{2^{128}-3}{11 \cdot 6949}$$

Let $R = 2^{128}$, use a redundant representation modulo $\widetilde{p} = R - 3 = 11 \cdot 6949 \cdot p$

Note: $x \cdot 2^{128} \equiv x \cdot 3 \bmod \widetilde{p}$

$$\begin{array}{rccl} \mathfrak{R}: & \mathbb{Z}/2^{256}\mathbb{Z} & \to & \mathbb{Z}/2^{256}\mathbb{Z} \\ & x & \mapsto & \left(x \bmod 2^{128}\right) + 3 \cdot \left\lfloor \frac{x}{2^{128}} \right\rfloor \end{array}$$

$$x = x_H \cdot 2^{128} + x_L \equiv x_L + 3 \cdot x_H = \mathfrak{R}(x) \bmod \widetilde{p}$$

Sloppy Reduction

How often does it happen that $\mathfrak{R}(\mathfrak{R}(a \cdot b)) >= R$?

Given $x = x_0 + x_1 R$, $0 \leq x < R^2$, then
$\mathfrak{R}(x) = x_0 + 3x_1 = y = y_0 + y_1 R \leq 4R - 4$ and hence: $y_1 \leq 3$

Sloppy Reduction

How often does it happen that $\mathfrak{R}(\mathfrak{R}(a \cdot b)) >= R$?

Given $x = x_0 + x_1 R$, $0 \leq x < R^2$, then
$\mathfrak{R}(x) = x_0 + 3x_1 = y = y_0 + y_1 R \leq 4R - 4$ and hence: $y_1 \leq 3$

If $y_1 = 3$, then $y_0 + y_1 R = y_0 + 3R \leq 4R - 4$ and thus $y_0 \leq R - 4$.
If $y_1 \leq 2$, then $y_0 \leq R - 1$.

$$\mathfrak{R}(\mathfrak{R}(x)) = \left\{ \begin{array}{c} y_0 + 3y_1 \leq (R-4) + 3 \cdot 3 \\ y_0 + 2y_1 \leq (R-1) + 3 \cdot 2 \end{array} \right\} = R + 5.$$

Rough heuristic approximation: $\frac{6}{R+6}$

Sloppy Reduction

How often does it happen that $\mathfrak{R}(\mathfrak{R}(a \cdot b)) >= R$?

Given $x = x_0 + x_1 R$, $0 \leq x < R^2$, then
$\mathfrak{R}(x) = x_0 + 3x_1 = y = y_0 + y_1 R \leq 4R - 4$ and hence: $y_1 \leq 3$

If $y_1 = 3$, then $y_0 + y_1 R = y_0 + 3R \leq 4R - 4$ and thus $y_0 \leq R - 4$.
If $y_1 \leq 2$, then $y_0 \leq R - 1$.

$$\mathfrak{R}(\mathfrak{R}(x)) = \left\{ \begin{array}{l} y_0 + 3y_1 \leq (R-4) + 3 \cdot 3 \\ y_0 + 2y_1 \leq (R-1) + 3 \cdot 2 \end{array} \right\} = R + 5.$$

Rough heuristic approximation: $\frac{6}{R+6}$
More sophisticated heuristic:

$$\left(\frac{\phi(\tilde{p})}{\tilde{p}}\right) \cdot \sum_{k=1,2} \left(3 - k - k \log\left(\frac{3}{k}\right)\right) \approx \frac{0.99118}{R} < \frac{1}{R}$$

Performance Results

Operation (sloppy modulus $\tilde{p} = 2^{128} - 3$, modulus $p = \frac{\tilde{p}}{11 \cdot 6949}$)	Average # cycles per two interleaved 4-SIMD operations	Average # cycles per operation	Operations per iteration	Average # cycles per iteration
Sloppy multiplication modulo $\tilde{p}$ (multiplication+reduction)	430 (318 + 112)	54 (40 + 14)	6	322
Modular subtraction	40 even, 24 odd: 40 total	5	6	30
Modular inversion	n/a	4941	$\frac{1}{400}$	12
Unique representation mod p	192	24	1	24
Miscellaneous	544	68	1	68
Total				456

Performance Results

Operation (sloppy modulus $\tilde{p} = 2^{128} - 3$, modulus $p = \frac{\tilde{p}}{11\cdot6949}$)	Average # cycles per two interleaved 4-SIMD operations	Average # cycles per operation	Operations per iteration	Average # cycles per iteration
Sloppy multiplication modulo $\tilde{p}$ (multiplication+reduction)	430 (318 + 112)	54 (40 + 14)	6	322
Modular subtraction	40 even, 24 odd: 40 total	5	6	30
Modular inversion	n/a	4941	$\frac{1}{400}$	12
Unique representation mod p	192	24	1	24
Miscellaneous	544	68	1	68
Total				456

Hence, our 214-PS3 cluster:

- computes $9.1 \cdot 10^9 \approx 2^{33}$ iterations per second
- works on $> 0.5M$ curves in parallel

Storage

- Per PS3: one distinguished point (4×16 bytes) per two second
- When storing the data naively: $\approx$ 300GB expected

Comparison

XC3S1000 FPGAs [1]

- FPGA-results of EC over 96- and 128-bit generic prime fields for COPACOBANA [2]
- Can host up to 120 FPGAs (US$ $10,000$)

Our implementation

- Targeted at 112-bit prime curve
- Use 128-bit multiplication + fast reduction modulo $\widetilde{p}$
- For US$ $10,000$ buy 33 PS3s

[1] T. Güneysu, C. Paar, and J. Pelzl. Special-purpose hardware for solving the elliptic curve discrete logarithm problem. *ACM Transactions on Reconfigurable Technology and Systems*, 1(2):1-21, 2008.
[2] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler. Breaking ciphers with COPACOBANA a cost-optimized parallel code breaker. In CHES 2006, volume 4249 of LNCS, pages 101-118, 2006.

Comparison

	96 bits	128 bits
COPACOBANA	$4.0 \cdot 10^7$	$2.1 \cdot 10^7$
+ Moore's law	$7.9 \cdot 10^7$	$4.2 \cdot 10^7$
+ Negation map	$1.1 \cdot 10^8$	$5.9 \cdot 10^7$
PS3	$4.2 \cdot 10^7$	
33 PS3	$1.4 \cdot 10^9$	

33 PS3 / COPACOBANA (96 bits): 12.4 times faster
33 PS3 / COPACOBANA (128 bits): 23.8 times faster

Comparison

	96 bits	128 bits
COPACOBANA	$4.0 \cdot 10^7$	$2.1 \cdot 10^7$
+ Moore's law	$7.9 \cdot 10^7$	$4.2 \cdot 10^7$
+ Negation map	$1.1 \cdot 10^8$	$5.9 \cdot 10^7$
PS3	$4.2 \cdot 10^7$	
33 PS3	$1.4 \cdot 10^9$	

33 PS3 / COPACOBANA	(96 bits):	12.4	times faster
33 PS3 / COPACOBANA	(128 bits):	23.8	times faster

Note

The 33 dual-threaded PPE were not used

The new COPACOBANA has faster FPGAs
(no performance results known yet).

The 112-bit Solution

The point P of prime order n is given in the standard.
The x-coordinate of Q was chosen as $\lfloor (\pi - 3)10^{34} \rfloor$.

The 112-bit Solution

The point P of prime order n is given in the standard.
The x-coordinate of Q was chosen as $\lfloor(\pi - 3)10^{34}\rfloor$.

- Expected #iterations $\sqrt{\frac{\pi \cdot n}{2}} \approx 8.4 \cdot 10^{16}$
- January 13, 2009 – July 8, 2009 (not running continuously)
- When run continuously using the latest version of our code, the same calculation would have taken 3.5 months

$$
\begin{array}{lll}
P = & (188281465057972534892223778713752, & 3419875491033170827167861896082688) \\
Q = & (1415926535897932384626433832795028, & 3846759606494706724286139623885544) \\
n = & 4451685225093714776491891542548933 &
\end{array}
$$

The 112-bit Solution

The point P of prime order n is given in the standard.
The x-coordinate of Q was chosen as $\lfloor(\pi - 3)10^{34}\rfloor$.

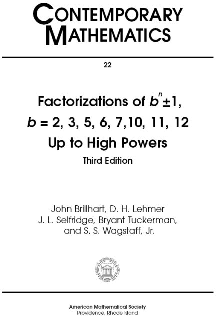
- Expected #iterations $\sqrt{\frac{\pi \cdot n}{2}} \approx 8.4 \cdot 10^{16}$
- January 13, 2009 – July 8, 2009 (not running continuously)
- When run continuously using the latest version of our code, the same calculation would have taken 3.5 months

$$\begin{array}{lrr} P = & (188281465057972534892223778713752, & 3419875491033170827167861896082688) \\ Q = & (1415926535897932384626433832795028, & 3846759606494706724286139623885544) \\ n = & 4451685225093714776491891542548933 & \end{array}$$

$$Q = 312521636014772477161767351856699 \cdot P$$

Outline

- The Cell Broadband Engine Architecture
- Project 1: 112-bit prime field ECDLP
- Project 2: Fast arithmetic modulo a Mersenne number in ECM
  - Joppe W. Bos, Thorsten Kleinjung, Arjen K. Lenstra, Peter L. Montgomery. *Efficient SIMD arithmetic modulo a Mersenne number*, Cryptology ePrint Archive: Report 2010/338, 2010.



CONTEMPORARY
MATHEMATICS

22

Factorizations of $b^n \pm 1$,
b = 2, 3, 5, 6, 7,10, 11, 12
Up to High Powers
Third Edition

John Brillhart, D. H. Lehmer
J. L. Selfridge, Bryant Tuckerman,
and S. S. Wagstaff, Jr.

American Mathematical Society
Providence, Rhode Island

The Elliptic Curve Factorization Method

H. W. Lenstra, *Factoring integers with elliptic curves*, Annals of Mathematics 126 (1987), 649–673.

Goal: factor $n \in \mathbb{Z}$

Pretend that $\mathbb{Z}/n\mathbb{Z}$ is a field, pick a random curve $E_{a,b}(\mathbb{Z}/n\mathbb{Z})$ and a random point $P \in E_{a,b}(\mathbb{Z}/n\mathbb{Z})$. Compute:

$$(x, y, z) := \prod_{q \leq B_1} q^{\left\lfloor \frac{\log B_1}{\log q} \right\rfloor} P,$$

where q is prime, computations are modulo n. If $p = \gcd(n, z) \neq \{1, n\}$ then a non-trivial factor p of n has been found, else repeat.

The Elliptic Curve Factorization Method

H. W. Lenstra, *Factoring integers with elliptic curves*, Annals of Mathematics 126 (1987), 649–673.

Goal: factor $n \in \mathbb{Z}$

Pretend that $\mathbb{Z}/n\mathbb{Z}$ is a field, pick a random curve $E_{a,b}(\mathbb{Z}/n\mathbb{Z})$ and a random point $P \in E_{a,b}(\mathbb{Z}/n\mathbb{Z})$. Compute:

$$(x, y, z) := \prod_{q \leq B_1} q^{\left\lfloor \frac{\log B_1}{\log q} \right\rfloor} P,$$

where q is prime, computations are modulo n. If $p = \gcd(n, z) \neq \{1, n\}$ then a non-trivial factor p of n has been found, else repeat.

The expected time used by ECM to find a factor p of a number n is

$$O(L(p)^{\sqrt{2}+o(1)} M(\log n))$$

where $L(p) = e^{\sqrt{\log p \log \log p}}$ and $M(\log n)$ represents the complexity of multiplication modulo n.

Special Moduli

Moduli of special form allow fast computation

- Proposed in the 1960s in the setting of residue number systems
  R. D. Merrill. *Improving digital computer performance using residue number theory*. Electronic Computers, IEEE Transactions on, EC-13(2):93–101, April 1964
- Speed up fast Fourier transform based multiplications
  R. Crandall and B. Fagin. *Discrete weighted transforms and large-integer arithmetic*. Mathematics of Computation, 62(205):305–324, 1994
- Speeding up elliptic curve cryptography
  The NIST curves
  D. J. Bernstein. *Curve25519: New Diffie-Hellman speed records*. In PKC 2006, volume 3958 of LNCS, pages 207 228, 2006.
- Factorization of *Cunningham numbers*, numbers of the form $b^n \pm 1$ for $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers.
  A. J. C. Cunningham and H. J. Woodall. *Factorizations of* $b^n \pm 1$ *for* $b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers*. Frances Hodgson, London, 1925

Mersenne numbers: $a = 2^M - 1$

We target M in the range $[1000, 1200]$

- Target the finite field arithmetic, the ECM implementation is from **GMP-ECM**
- Optimize for throughput and reasonable latency
- Compute on 4 computations in parallel
- Do not interleave multiple of 4-way SIMD streams
- Instead exploit the parallelism inside the computations
- What prime sizes are doable with ECM? (RSA multi-prime, unbalanced RSA)

Mersenne numbers: $a = 2^M - 1$

We target M in the range $[1000, 1200]$

- Target the finite field arithmetic,
  the ECM implementation is from **GMP-ECM**
- Optimize for throughput and reasonable latency
- Compute on 4 computations in parallel
- Do not interleave multiple of 4-way SIMD streams
- Instead exploit the parallelism inside the computations
- What prime sizes are doable with ECM?
  (RSA multi-prime, unbalanced RSA)

unsigned radix-2^{32}	signed radix-2^{13}
$a = \sum_{j=0}^{38} a_j 2^{32j}$	$a = \sum_{j=0}^{95} a_j 2^{13j}$
$0 \leq a_j < 2^{32}$	$-2^{12} \leq a_j < 2^{12}$

Exploit the fast multiply-and-add instruction on the Cell

Modular Arithmetic: Two Approaches

In- and output are in unsigned radix-2^{32}

1. conversion of inputs a and b to signed radix-2^{13} representation;
2. carry-less calculation of the $2M$-bit product $a \cdot b$ in signed 32-bit radix-2^{13} representation;
3. reduction modulo N and conversion to radix-2^{32} representation of the $2M$-bit product $a \cdot b$, resulting in $c = a \cdot b \bmod N \in \{0, 1, \ldots, N-1\}$.

Additions and subtractions in unsigned radix-2^{32} are faster
The conversion back can absorb the reduction almost for free
The conversions are expensive

Modular Arithmetic: Two Approaches

In- and output are in signed radix-2^{13}

1. conversion of inputs a and b to signed radix-2^{13} representation;
2. carry-less calculation of the $2M$-bit product $a \cdot b$ in signed 32-bit radix-2^{13} representation;
3. reduction modulo N and conversion to radix-2^{32} representation of the $2M$-bit product $a \cdot b$, resulting in $c = a \cdot b \bmod N \in \{0, 1, \ldots, N - 1\}$.

Additions and subtractions in unsigned radix-2^{32} are faster
The conversion back can absorb the reduction almost for free
The conversions are expensive

Example: Conversion to signed radix-2^{13}

Straightforward approach is slow due to lots of data dependencies.

Other approach:
pre-compute (radix-2^{32} representation) $C_0 = 2^{12} \cdot \sum_{j=0}^{95} 2^{13j}$.

1. Calculate the radix-2^{32} representation of $a + C_0$ (carries)
2. extract the radix-2^{13} representation $\sum_{j=0}^{95} \tilde{a}_j 2^{13j} = a + C_0$ using masks and shifts (in parallel)
3. subtract C_0: $a_j = \tilde{a}_j - 2^{12}$, for $j = 0, 1, \ldots, 95$ (in parallel)

Example: Conversion to signed radix-2^{13}

Pack two signed radix-2^{13} digits in one 32-bit word (2x speedup).
Obtain a, regarded as a polynomial

$$P_a(X) = \sum_{j=0}^{95} a_j X^j \in \mathbb{Z}[X]$$

with $P_a(2^{13}) = a$

Multiplication

$$\text{Product polynomial: } P(X) = P_a(X)P_b(X) = \sum_{j=0}^{190} p_j X^j$$

$$\text{with } |p_j| \leq 96 \cdot (2^{12})^2 < 2^{31} \text{ such that } P(2^{13}) = a \cdot b$$

Carry-less product calculation of a and b allows computation modulo 2^{32}

Four levels of Karatsuba multiplication

- 81 independent polynomial multiplications $Q^{(k)}(X) = P_a^{(k)}(X)P_b^{(k)}(X)$ of degree ≤ 5
- Carry-less schoolbook multiplications ($96 \times 96 \rightarrow 192$-bit) $\rightarrow$ factor 2 speedup over regular schoolbook multiplication
- Carry-less additions and subtractions of the $Q^{(k)}(X)$'s result in the polynomial $P(X)$

Performance

SPE effort for 4-way SIMD phase one ECM trials for $N = 2^{1193} - 1$, $B_1 = 3 \cdot 10^9$

operation	number of calls	radix-2^{32}		signed radix-2^{13}	
mod N		cpc	hours	cpc	hours
$a \cdot b$	26 193 284 192	6971	15.89	5666	12.92
a^2	13 358 576 558	4814	5.60	4306	5.00
$\left.\begin{matrix}a+b\\a-b\end{matrix}\right\}$	18 990 126 989	268	0.44	645	1.12
$a+b$	523 868 924	180	0.01		
$a-b$	523 868 924	180	0.01		
		total	21.95		19.05

The PS3-cluster:

24k curves expected to find a 65-digit factor:	$<$ 4 days
110k curves expected to find a 70-digit factor:	two and a half weeks

Comparison

Table : Time to complete 24 phase one ECM trials.

processor	GHz	cores	hours	
			Mersenne	generic
Intel Xeon E5430	2.66	8	23.70	43.13
Intel Core i7 920	2.67	4	46.28	83.52
Intel Core2 Quad Q9550	2.83	4	47.26	85.93
Intel Core2 Quad Q6700	2.66	4	48.80	86.45
AMD Phenom 9500	2.22	4	38.48	65.75
AMD Opteron 1381	2.50	4	33.78	58.46
PlayStation 3	3.19	6	19.20	

Results

M	targeted composite	completed number of trials		result
		phase one	phase two	
1051	c310	23 136	9 186	p63 · c248
1073	c281	24 504	1 460	p66 · p215
1139	c313	49 080	35 490	p68 · p246
1163	c318	50 152	47 768	p73 · p246
1181	c291	25 393	8 808	p73 · p218
1187	c266	15 089	9 860	p63 · p204
1237	c373	71 556	70 809	p70 · c303

Results

M	targeted composite	completed number of trials		result
		phase one	phase two	
961	c254	53 384	1 190	p61 · p193
1051	c310	23 136	9 186	p63 · c248
1073	c281	24 504	1 460	p66 · p215
1139	c313	49 080	35 490	p68 · p246
1163	c318	50 152	47 768	p73 · p246
1181	c291	25 393	8 808	p73 · p218
1187	c266	15 089	9 860	p63 · p204
1237	c373	71 556	70 809	p70 · c303