Kris Gaj
George Mason University

Factoring in Hardware

GEORGE MASON UNIVERSITY
SUNG KYUN KWAN UNIVERSITY

Factoring Team

Mathematicians/ Cryptographers

Software experiments

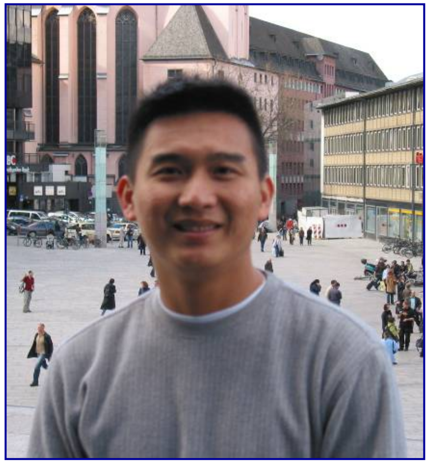
Soonhak Kwon
Associate Professor
Sungkyunkwan University,
Suwon, Korea;
Ph.D in Mathematics,
Johns Hopkins University
Maryland, U.S.A.;
visiting professor at GMU,
Aug. 2005-Sep. 2006

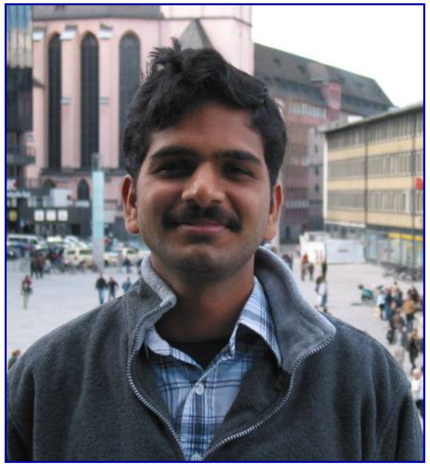
Patrick Baier
D. Phil. in
Mathematics,
Oxford University
Oxford, U.K
Affiliated with George
Washington University

Paul Kohlbrenner
Ph.D student,
ECE Department
George Mason University
Virginia, U.S.A.

Factoring Team

Hardware design





Hoang Le | Ramakrishna Bachimanchi | Khaleeluddin Mohammed

MS in Computer Engineering students
ECE Department
George Mason University
Virginia, U.S.A.

Outline

Motivation

- Factoring records
- Limitations of Factoring in Software

Factoring in Hardware

- Available Hardware Technologies and Platforms
- Previous Work on Hardware Architectures for Number Field Sieve (NFS)

Special Purpose Factoring Methods Required within NFS

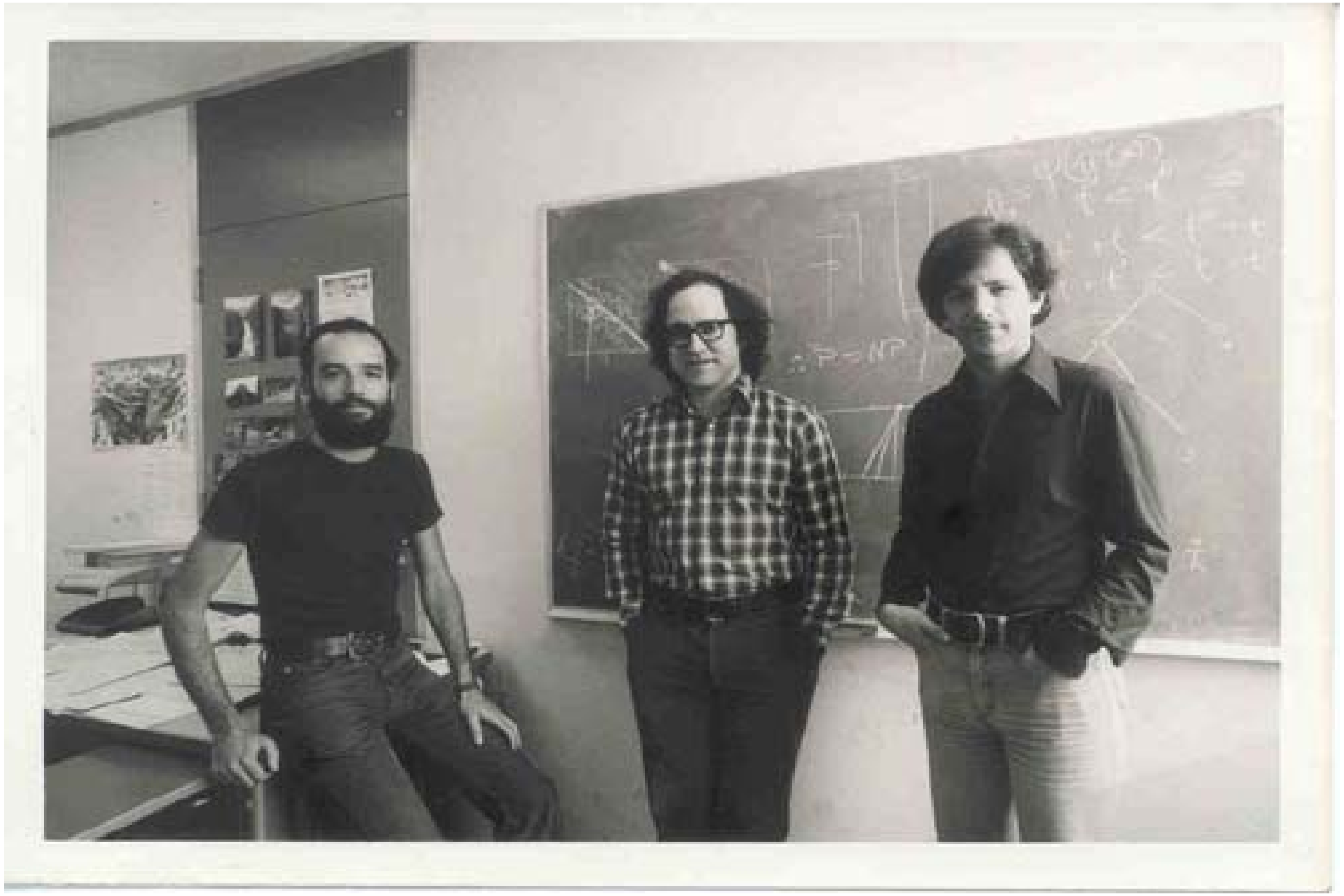
- Pollard’s “rho” Method
- Pollard’s p-1 Method
- Elliptic Curve Method (ECM)

Our Hardware Architectures for “rho”, p-1 and ECM

Optimal Choice of the Hardware Technology and Platform

Future Work, Conclusions and Open Problems

RSA & Factoring in Software



In 1977 by

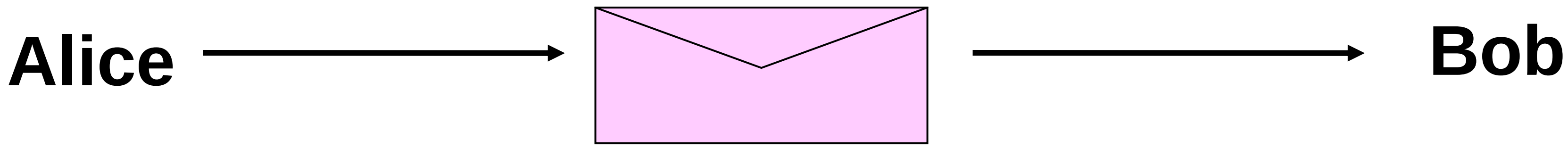
Ron Rivest, Adi Shamir & Leonard Adleman

developed the first public key cryptosystems, they called RSA

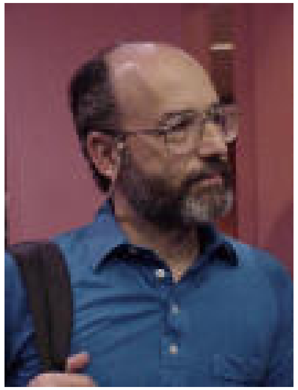
Common Applications of RSA

Secure WWW, SSL

S/MIME, PGP



ACM A.M. Turing Award
2002

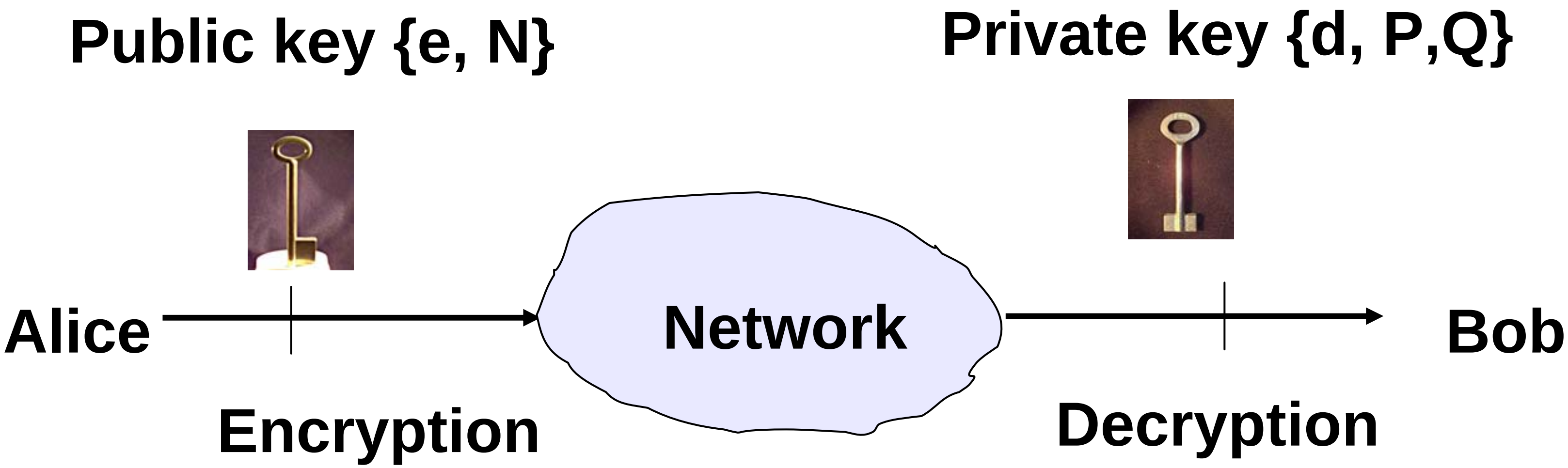


R. Rivest
A. Shamir
L. Adleman

"For Seminal Contributions to the Theory and Practical Applications of Public Key Cryptography"

RSA

Major Public Key Cryptosystem



$$\{ e, N \} \quad \overset{\longleftarrow}{\cancel{\longrightarrow}} \quad \{ d, P, Q \}$$

N = P·Q **P, Q - large prime factors**

$$e \cdot d \equiv 1 \mod ((P-1)(Q-1))$$

RSA challenge published in Scientific American

1977

Ciphertext:

9686 9613 7546 2206 1477 1409 2225 4355
8829 0575 9991 1245 7431 9874 6951 2093
0816 2982 2514 5708 3569 3147 6622 8839
8962 8013 3919 9055 1829 9451 5781 5145

Public key:

N = 114381625757 888867669235779976146
612010218296721242362562561842935
706935245733897830597123563958705
058989075147599290026879543541

e = 9007

129 decimal digits = 429 bits

Award $100

Rivest estimation - 1977

The best known algorithm for factoring a
129-digit number requires:

40 000 trilion years
$= 40 \cdot 10^{15}$ years

assuming the use of a supercomputer
being able to perform

1 multiplication of 129 decimal digit numbers in 1 ns

Rivest's assumption translates to the delay of a single logic gate $\approx$ 10 ps

Estimated age of the universe: 100 bln years = 10^{11} years

Breaking RSA-129

When: August 1993 - 1 April 1994, **8 months**

Who: D. Atkins, M. Graff, A. K. Lenstra, P. Leyland
+ 600 volunteers from the entire world

How: **1600 computers**
from Cray C90, through 16 MHz PC,
to fax machines

Only 0.03% computational power of the Internet

Results of cryptanalysis:

“The magic words are squeamish ossifrage”

An award of $100 donated to Free Software Foundation

Elements affecting the progress in factoring large numbers

- computational power

  1977-1993 increase of about 1500 times

- computer networks

  Internet

- **better algorithms**

Supercomputer Cray

Computer Museum, Mountain View, CA

How to factor for free?

A. Lenstra & M. Manasse, 1989

- **Using the spare time of computers, (otherwise unused)**

- **Program and results sent by e-mail (later using WWW)**

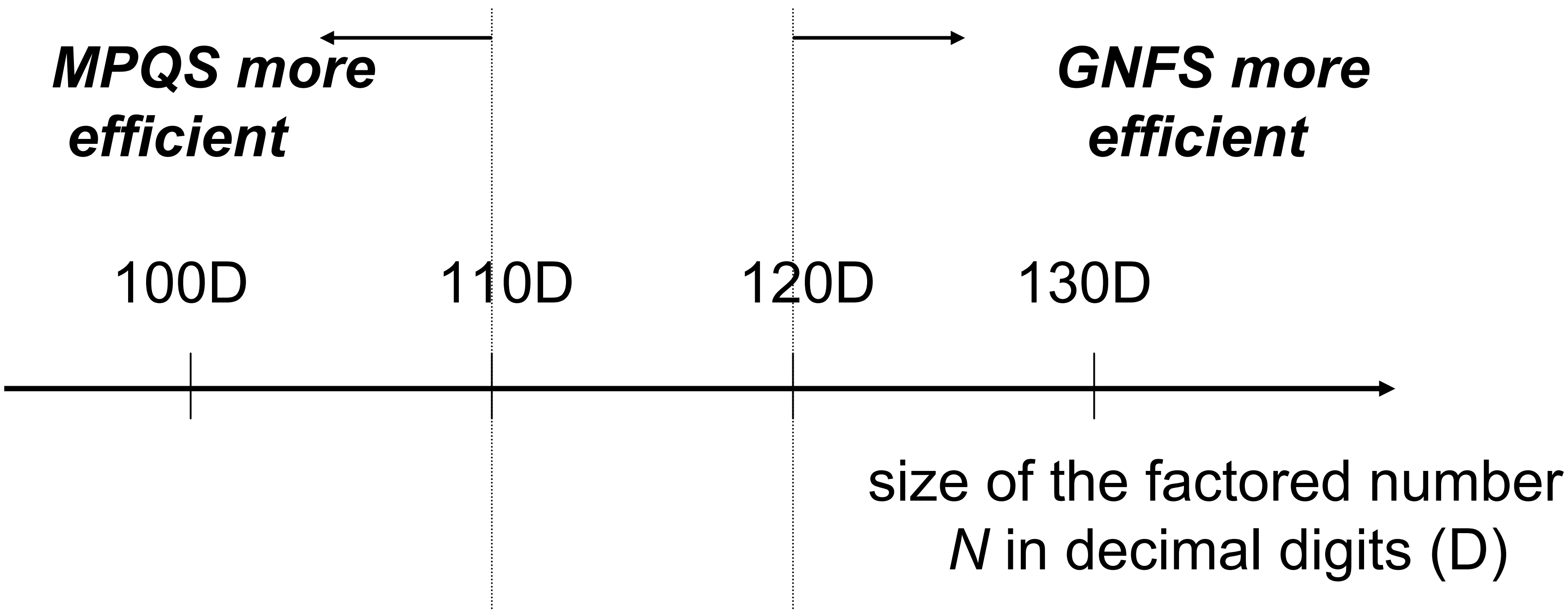
Best known general purpose factoring algorithms

Multiple Polynomial Quadratic Sieve
MPQS

General Number Field Sieve
GNFS

$L_N[1/2, 1] = exp((1 + o(1))\cdot(\ln N)^{1/2}))\cdot(\ln \ln N)^{1/2})$

$L_N[1/3, 1.92] = exp((1.92 + o(1))\cdot(\ln N)^{1/3}))\cdot(\ln \ln N)^{2/3})$



Factoring 512-bit number

512 bits = 155 decimal digits
old standard for key sizes in RSA

17 March - 22 August 1999

Group of Herman te Riele
Centre for Mathematics and Computer Science (CWI), Amsterdam

First stage — Several hundreds of workstations

2 months

Second stage

Cray C916
~2 weeks

Recommendations of RSA Security Inc.

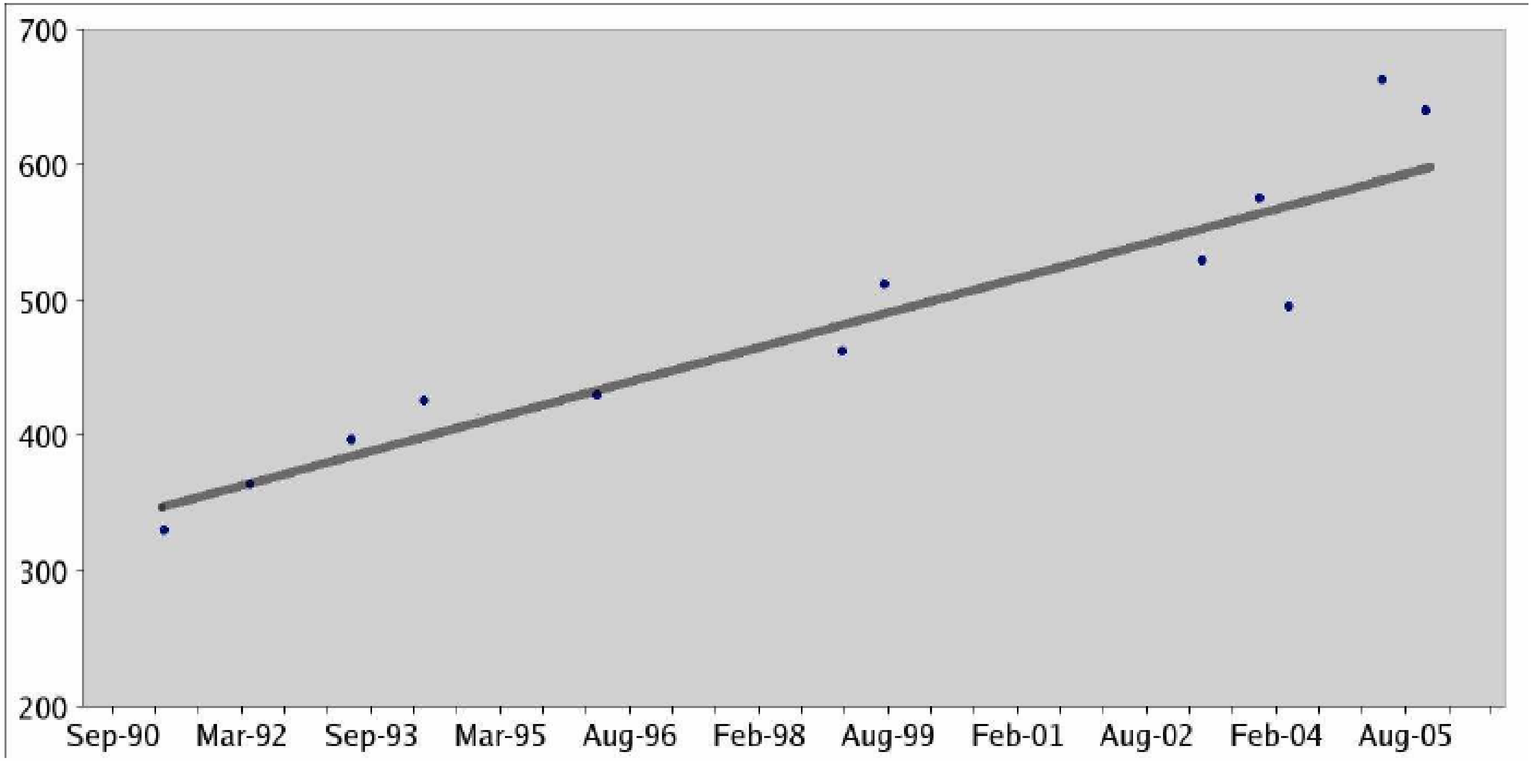
May 6, 2003

Validity period	Minimal RSA key length (bits)	Equivalent symmetric key length (bits)
2003-2010	1024	80
2010-2030	2048	112
2030-	3072	128

Factorization records

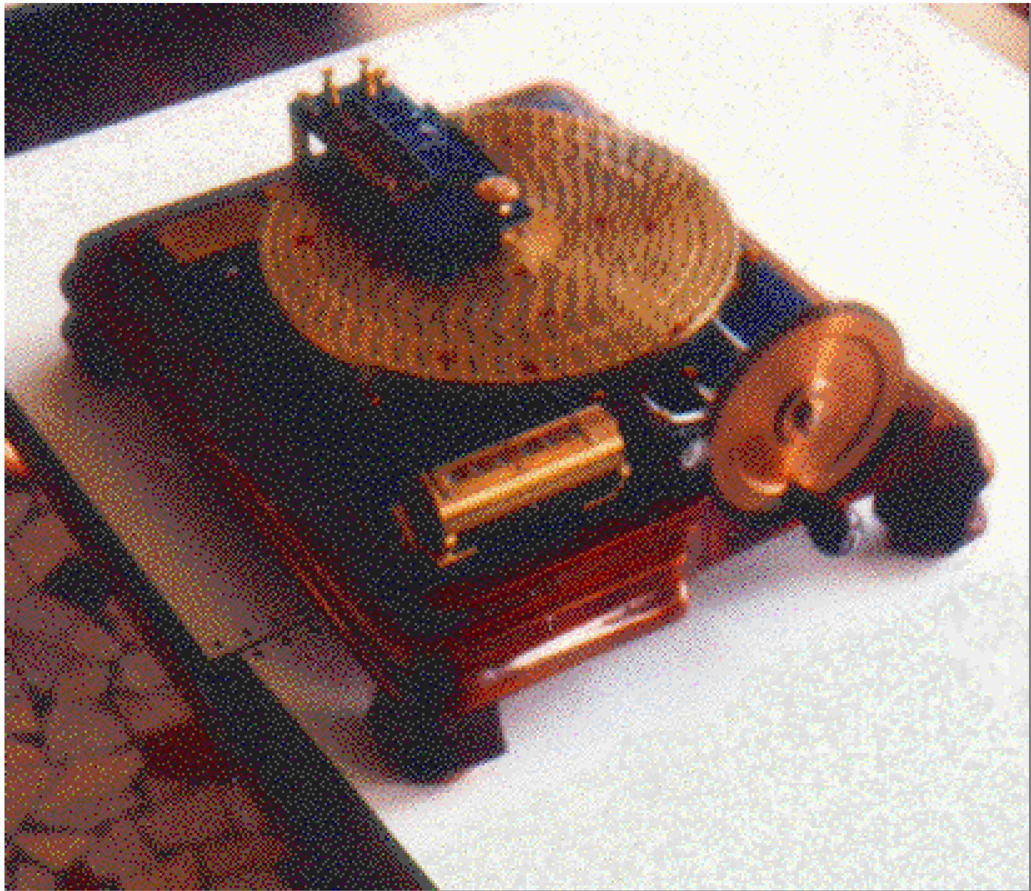
number	decimal digits	date	time (phase 1)	algorithm
C116	116	1990	275 MIPS years	mpqs
RSA-120	120	VI. 1993	830 MIPS years	mpqs
RSA-129	129	IV. 1994	5000 MIPS years	mpqs
RSA-130	130	IV. 1996	1000 MIPS years	gnfs
RSA-140	140	II. 1999	2000 MIPS years	gnfs
RSA-155	155	VIII. 1999	8000 MIPS years	gnfs
C158	158	I. 2002	3.4 Pentium 1GHz CPU years	gnfs
RSA-160	160	III. 2003	2.7 Pentium 1GHz CPU years	gnfs
RSA-576	174	XII. 2003	13.2 Pentium 1GHz CPU years	gnfs
C176	176	V. 2005	48.6 Pentium 1GHz CPU years	gnfs
RSA-200	200	V. 2005	121 Pentium 1GHz CPU years	gnfs

Factorization records



He who has absolute confidence in linear regression will expect a 1024-bit RSA number to be factored on December 17, 2028

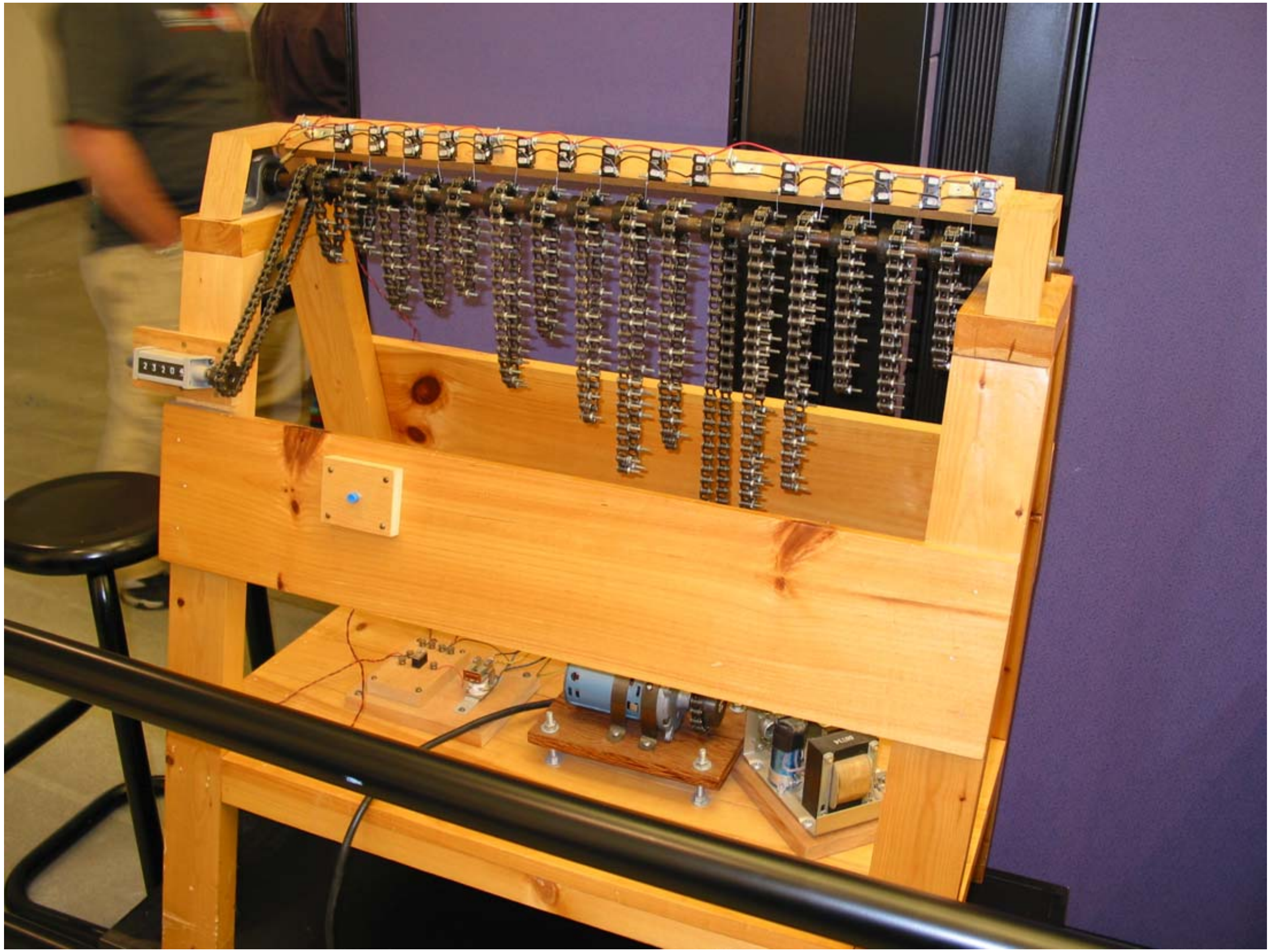
Factoring in Hardware



Machine à Congruences [E. O. Carissan, 1919]

Lehmer Sieve

Bicycle chain sieve [D. H. Lehmer, 1926, U.C. Berkeley]



Computer Museum, Mountain View, CA

Bernstein's Machine

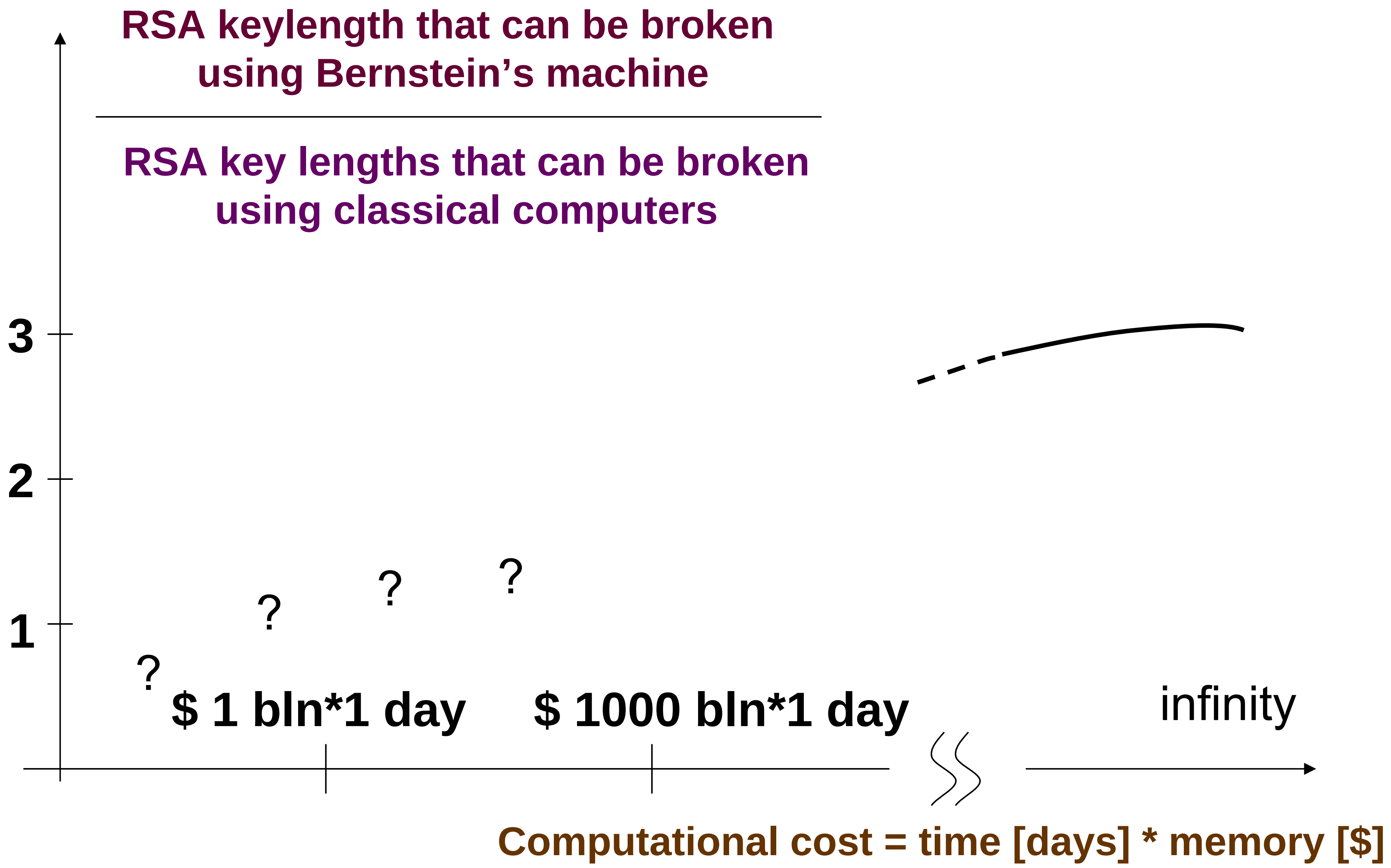
Fall 2001

Daniel Bernstein, professor of mathematics at
University of Illinois in Chicago
proposes hardware architecture capable of
performing factoring with better
asymptotic complexity
than any known software algorithm

D. Bernstein, ***Circuits for Integer Factorization: A Proposal***

http://cr.yp.to/papers.html#nfscircuit

Bernstein's Machine



Workshop Series

SHARCS - Special-purpose Hardware for Attacking Cryptographic Systems

1st edition:	Paris,	Feb.	24-25,	2005
2nd edition:	Cologne,	Apr.	3-4,	2006
3rd edition:	Vienna,	Sep.	9-10,	2007

See

http://www.ruhr-uni-bochum.de/itsc/tanja/SHARCS/

Two competing implementation approaches

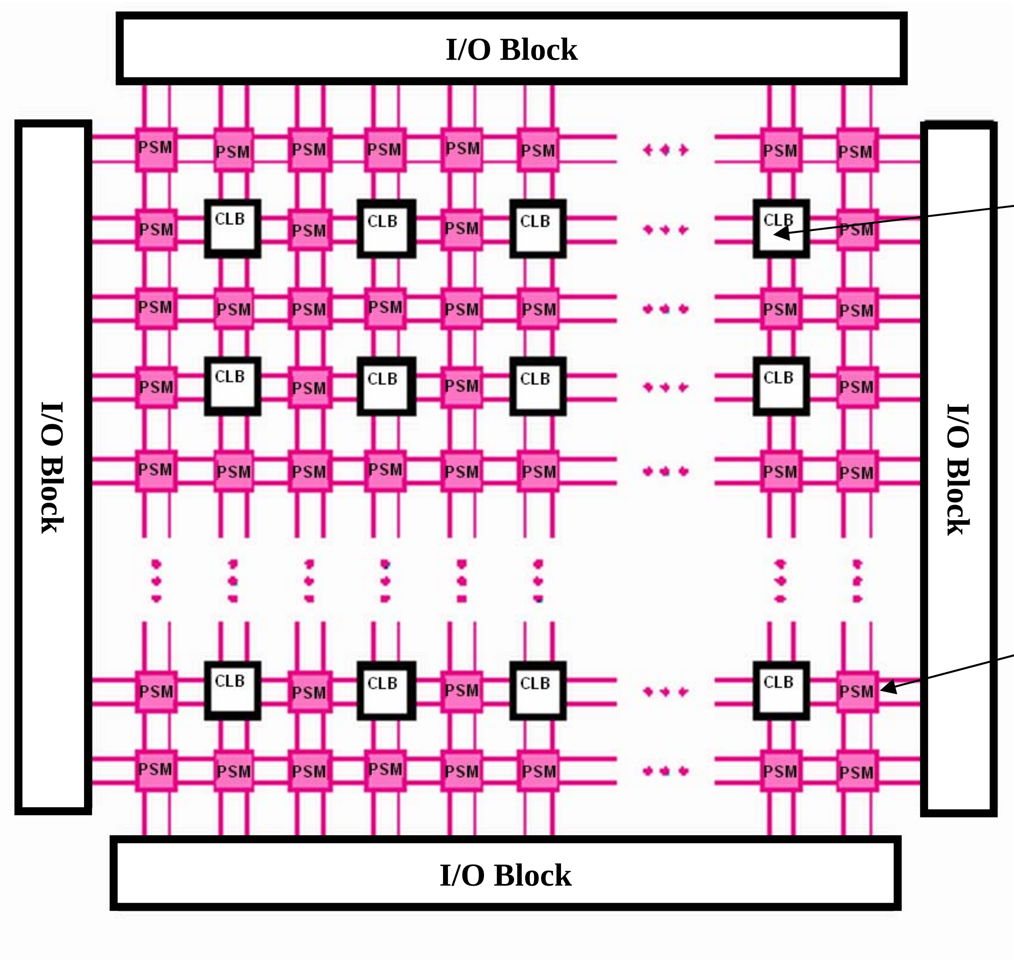
ASIC
Application **S**pecific **I**ntegrated **C**ircuit

- designed all the way from behavioral description to physical layout
- designs must be sent for expensive and time consuming fabrication in semiconductor foundry

FPGA
Field **P**rogrammable **G**ate **A**rray

- no physical layout design; design ends with a bitstream used to configure a device
- bought off the shelf and reconfigured by designers themselves

What is an FPGA Chip ?



General structure of an FPGA

Configurable Logic Blocks (Programmable Functional Units)

Programmable Switch Matrices (Programmable Interconnects)

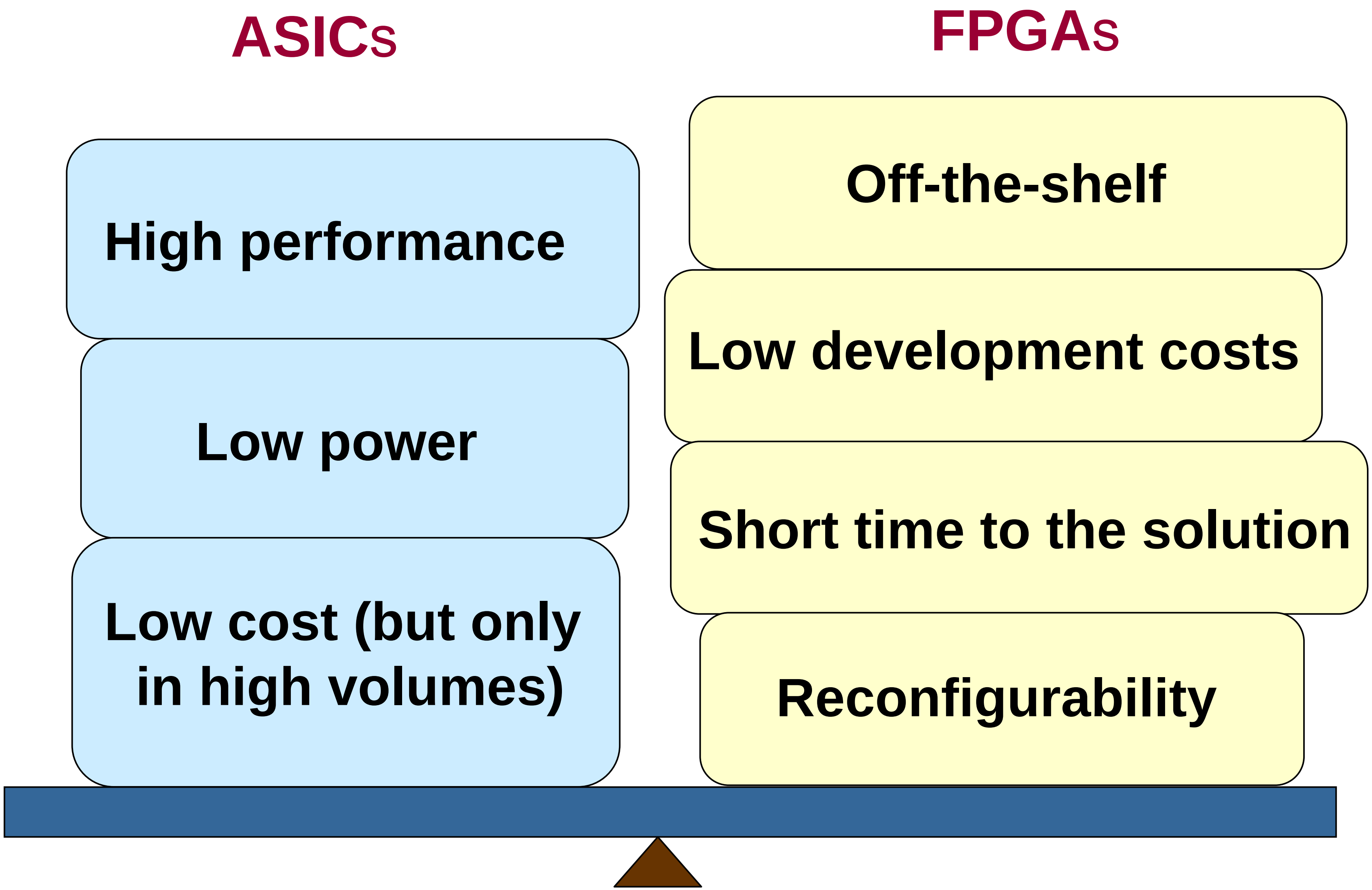
Source: [Brown99]

Families of FPGA Devices

Low-cost	High-performance
Spartan 3 (< $130*)	Virtex II (< $2,700*)
Spartan 3E (< $35*)	Virtex 4 (< $3,000*)

- approximate cost of the largest device per unit for a batch of 10,000 units

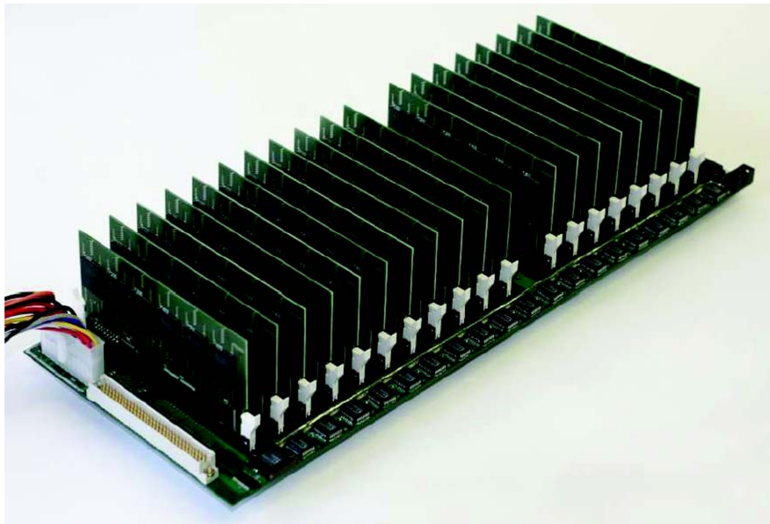
FPGAs vs. ASICs

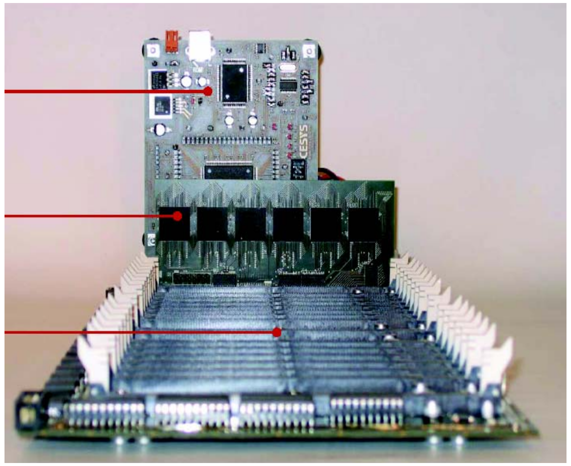


COPACOBANA

Ruhr University, Bochum,
University of Kiel, Germany, 2006

Cost: € 8980



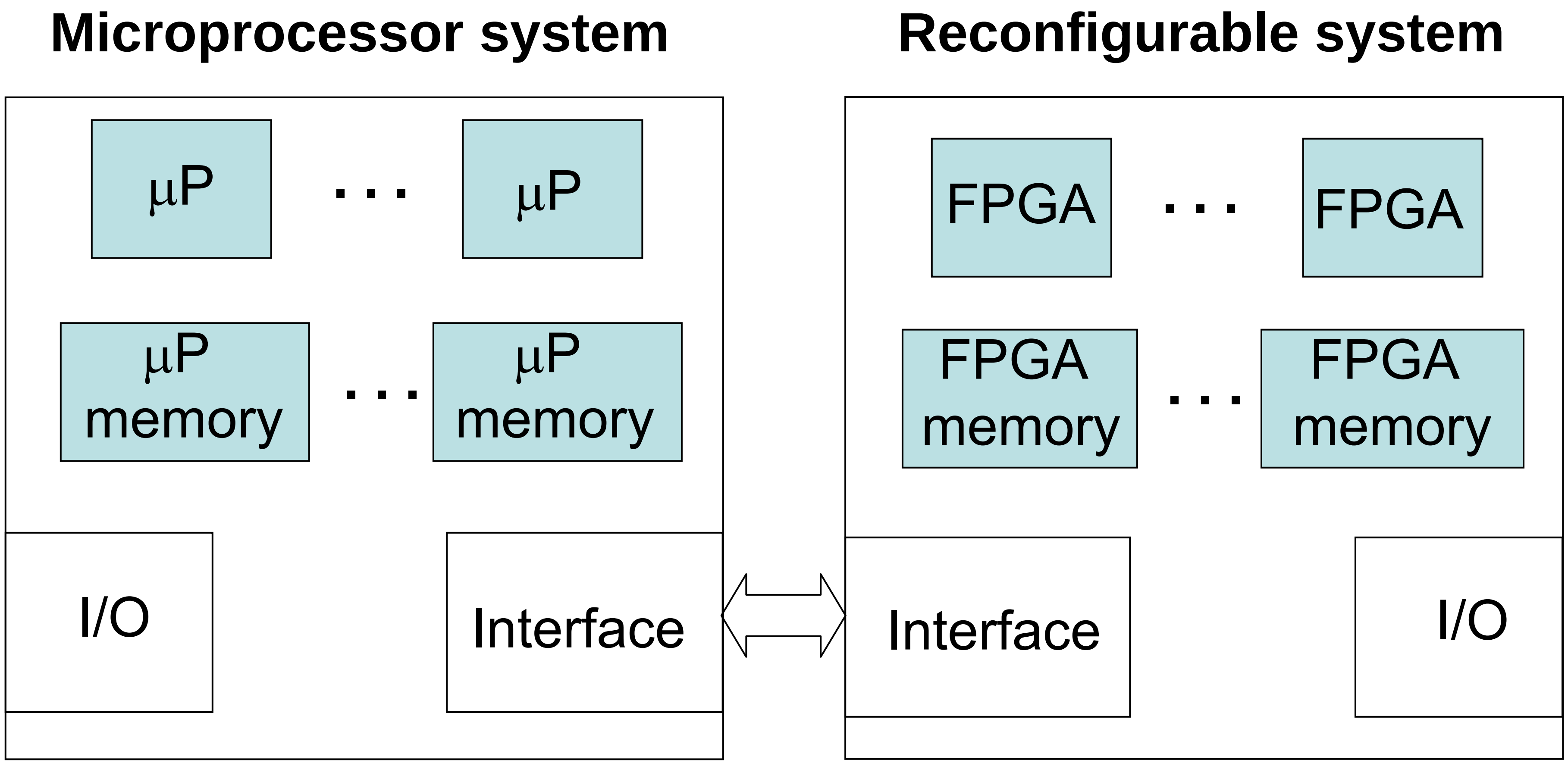


120 Spartan 3 FPGAs
Clock frequency 100 MHz

General-purpose reconfigurable computers

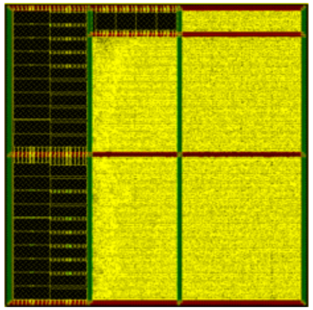
Machine	Released
SRC 6 from SRC Computers	2002
Cray XD1 from from Cray	2005
SGI RASC from SGI	2005
SRC 7 from SRC Computers, Inc,	2006

What is a reconfigurable computer?

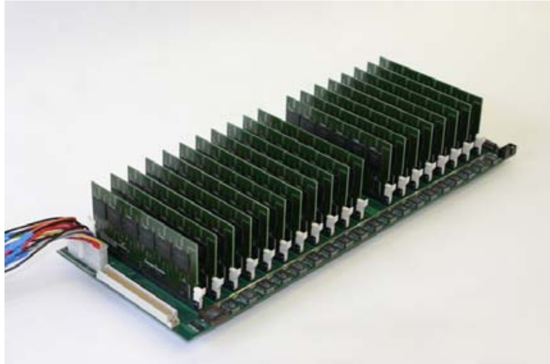


Comparison among technologies

Microprocessors **FPGAs** **ASICs**



SRC



COPACOBANA

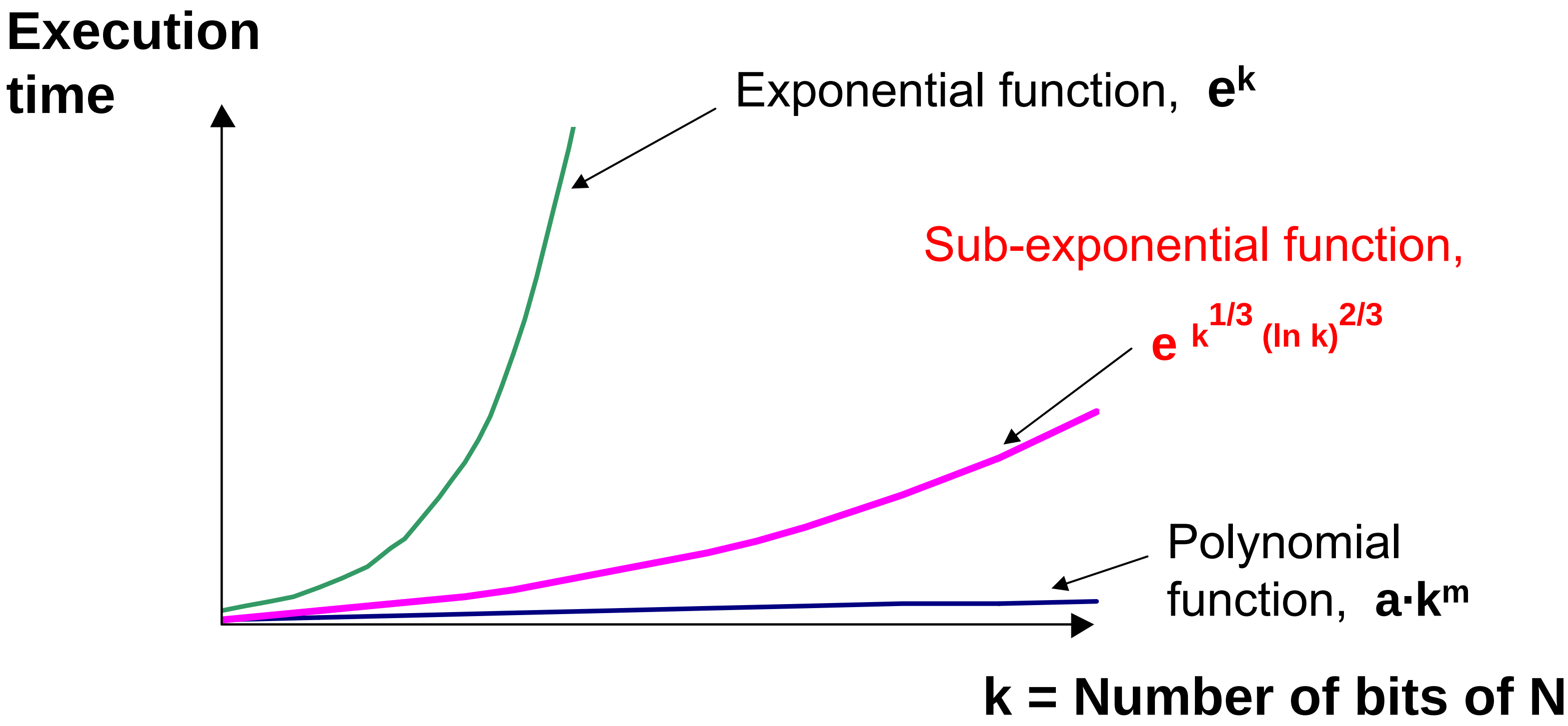
Number Field Sieve
In Hardware

Best Algorithm to Factor Large Numbers

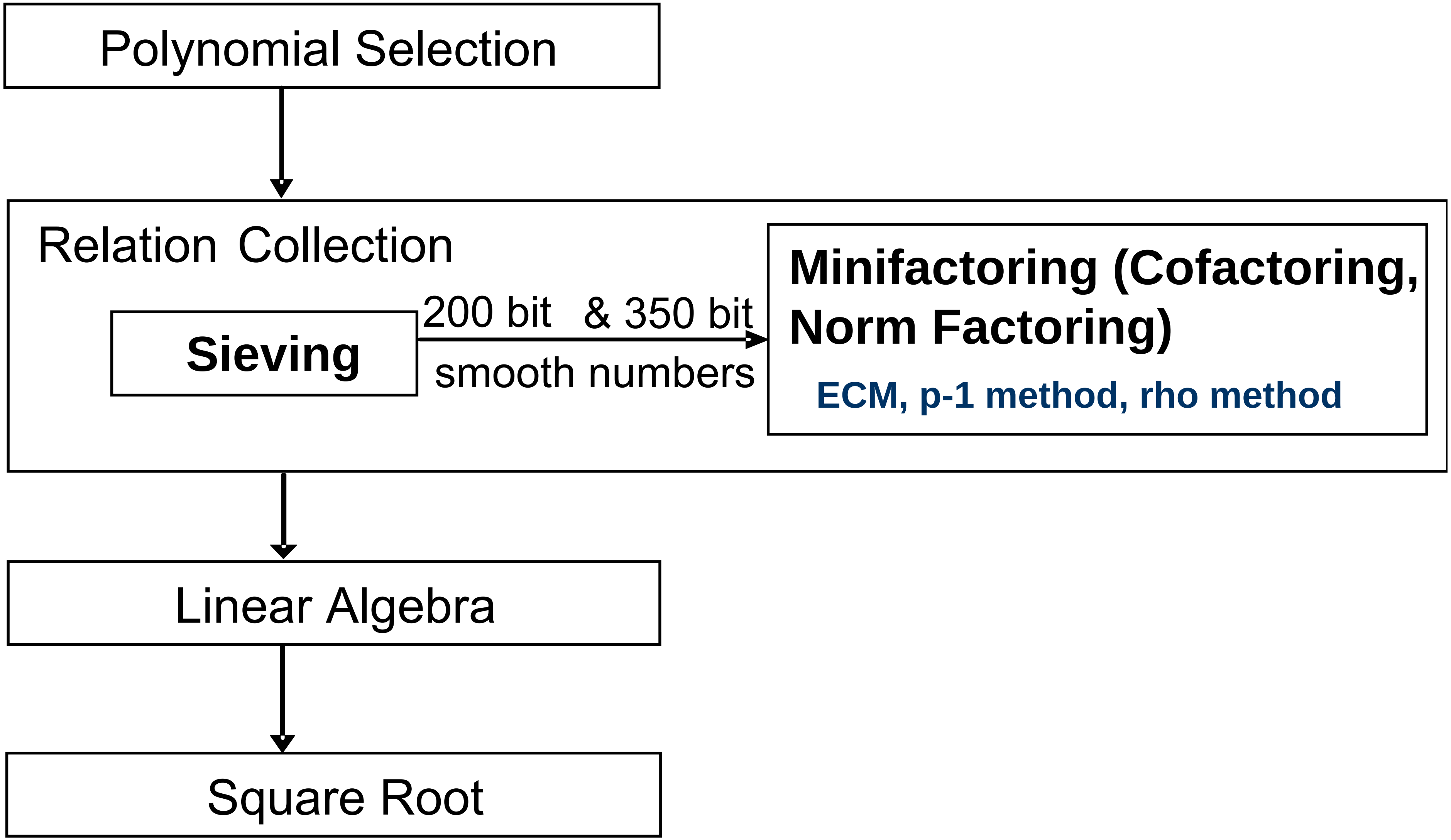
NUMBER FIELD SIEVE

Complexity: Sub-exponential time and memory

N = Number to factor,
k = Number of bits of N



Factoring 1024-bit RSA keys using Number Field Sieve (NFS)



B-smooth numbers

Integer N is B-smooth if and only if
N can be represented in the form

$$N = p_1^{e1} \cdot p_2^{e2} \cdot p_3^{e3} \cdot \ldots . \cdot p_t^{et} ,$$

where

$$\forall_i \quad p_i \le B$$

Relation Collection Step (1)

Given

$$F_1(x,y),\ F_2(x,y) \in Z\,[x,y],$$

two special homogeneous polynomials,
e.g. of degree 5 and 1

1. Find $(a,b) \in Z \times N$, $\gcd(a,b) = 1$
such that $F_1(a,b)$ and $F_2(a,b)$ are smooth.

2. Factor completely $F_1(a,b)$ and $F_2(a,b)$.

Relation Collection Step (2)

Sieving

Finding parameters a and b such that

$F_1(a, b)$, $F_2(a, b)$

are likely to be B_1 and B_2 smooth respectively,
i.e., factor completely using primes smaller than B_1 and B_2 respectively

Minifactoring (Norm factoring / Co-factoring)

Fully factoring numbers $F_1(a, b)$ and $F_2(a, b)$ found by sieving

Theoretical Designs for Sieving (1)

<u>1999-2000</u>

TWINKLE (Shamir, CHES 1999;

Shamir & Lenstra, Eurocrypt 2000)

- based on optoelectronic devices (fast LEDs)
- not even a small prototype built in practice
- not suitable for 1024 bit numbers

<u>2003</u>

TWIRL (Shamir & Tromer, Crypto 2003)

- semiconductor wafer design
- requires fast communication between chips located on the same 30 cm diameter wafer
- difficult to realize using current fabrication technology

Theoretical Designs for Sieving (2)

<u>2003-2004</u>

Mesh Based Sieving / YASD

(Geiselmann & Steinwandt, PKC 2003

Geiselmann & Steinwandt, CT-RSA 2004)

- not suitable for 1024 bit numbers

<u>2005</u>

SHARK (Franke et al., SHARCS & CHES 2005)

- relies on an elaborate butterfly switch
  connecting large number of chips
- difficult to realize using current technology

Theoretical Designs for Sieving (3)

2007

Non-Wafer-Scale Sieving Hardware
(Geiselmann & Steinwandt, Eurocrypt 2007)

- based on moderate size chips (2.2 x 2.2 cm)
- communication among chips seems to be realistic
- 2 to 3.5 times slower than TWIRL
- supports only linear sieving, and not more optimal lattice sieving

Estimated recurring costs with current technology (US$×year)

by Eran Tromer, May 2005

	768-bit	1024-bit
Traditional PC-based	1.3×10^{7}	10^{12}
TWINKLE	8×10^{6}	
TWIRL	5×10^{3}	10×10^{6}
Mesh-based	3×10^{4}	
SHARK		230×10^{6}

But: non-recurring costs, chip size, chip transport networks…

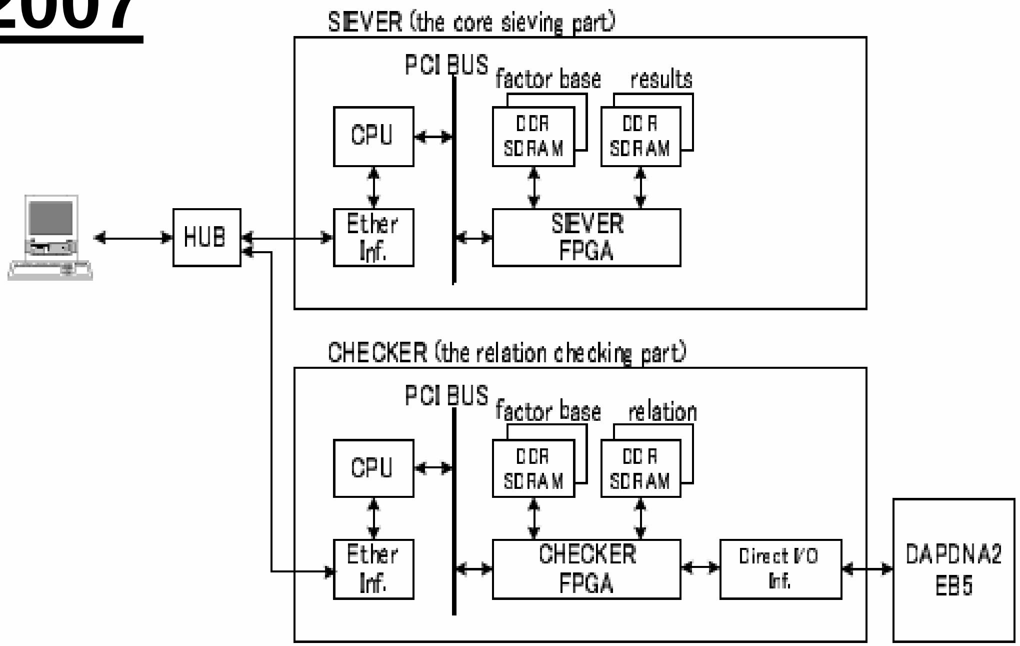
However…

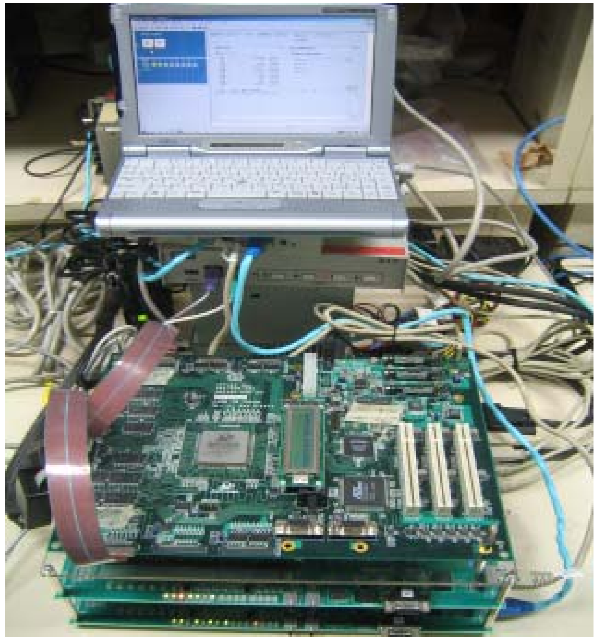
None of the theoretical designs ever built.

Just analytical estimations, no real implementations, no concrete numbers

First Practical Implementation of the Relation Collection Step in Hardware

2007





Japan

Tetsuya Izu and Jun Kogure
and Takeshi Shimoyama (Fujitsu)

CHES 2007 – September 2007

First large number factored using FPGA support

Factored number:

N	=	P	·	Q
423-bits		205 bits		218 bits

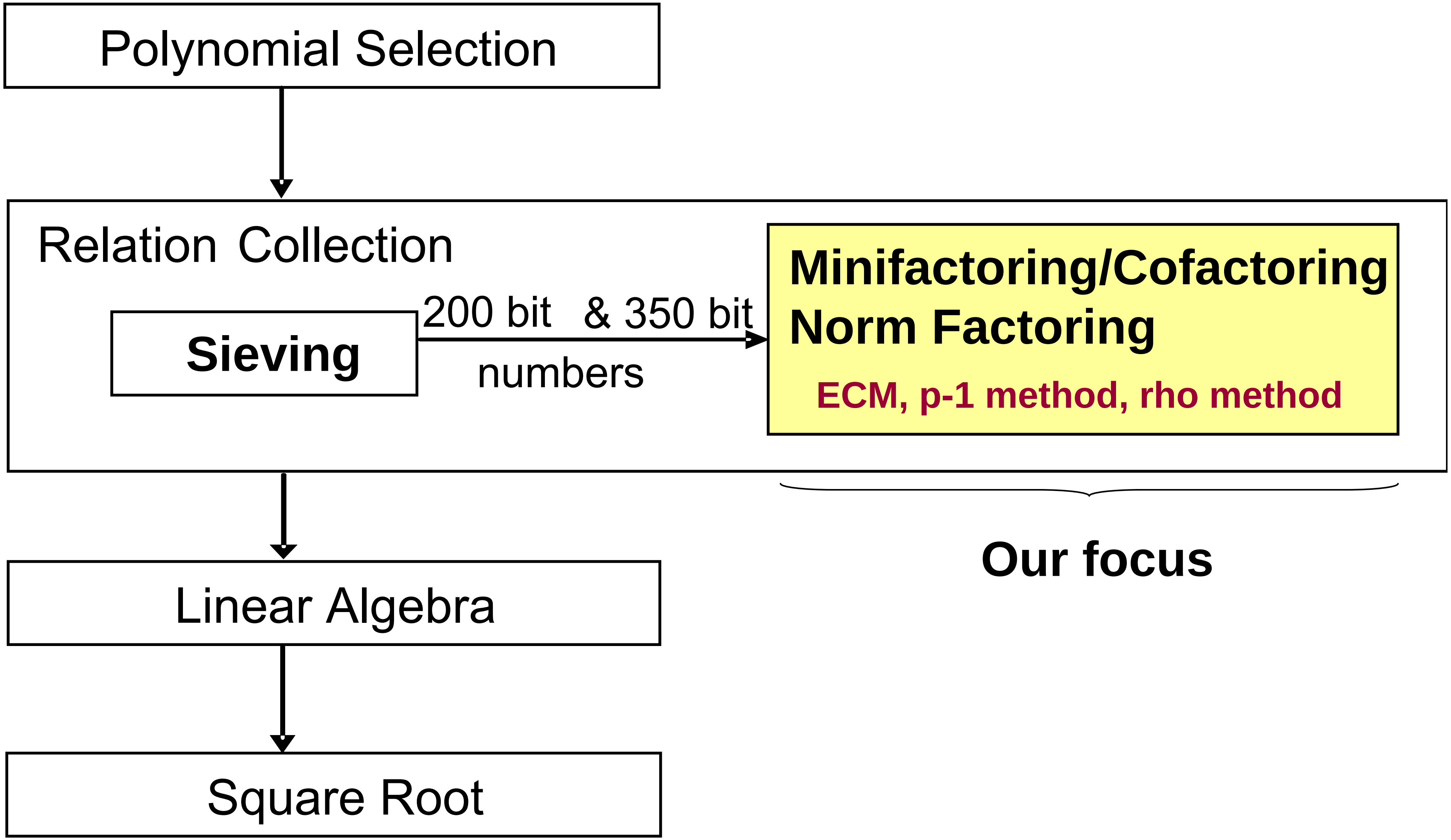
Time of computations:

One month of computations using a PC supported by FPGA boards

Problems:

- **Speed up vs. software not clearly determined**
- **Limited scalability**

Factoring 1024-bit RSA keys using Number Field Sieve (NFS)



ECM in Hardware

Previous Proof-of-Concept Design

Pelzl, Šimka,	SHARCS	Feb 2005
Kleinjung, Franke,	FCCM	Apr 2005
Priplata, Stahlke,	IEE Proc.	Oct 2005
Drutarovský, Fischer,		
Paar		

Rho, P-1 & ECM

Special-purpose factoring algorithms

Trial Division
Pollard's rho Method
Pollard's p-1 Method
Elliptic Curve Method

- Exponential complexity
- Often superior to more advanced (sub-exponential) methods for finding small factors
- Using more advanced methods to find small divisors is not a good use of resources

Pollard's Rho Method

<u>Birthday paradox:</u> If more than 23 "random" people are in a room (or even if they aren't) there is a more than 50% probability that the birthdays of two of them fall on the same day of the year.

Pollard's rho method - Example

$N = 97 \cdot 1889 = 183\,233$

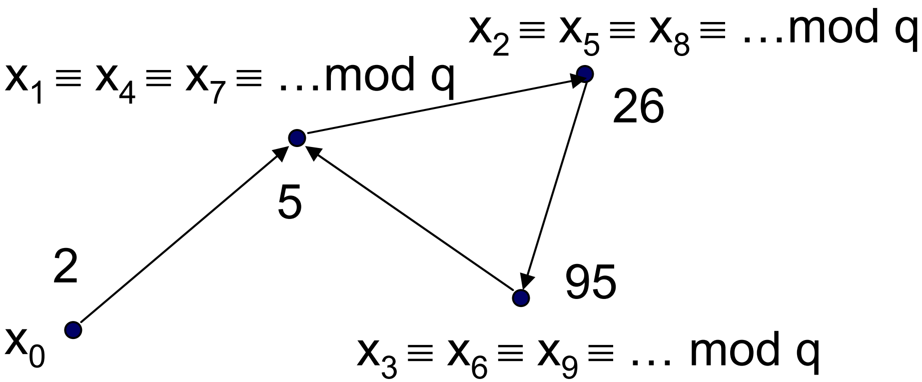
$$x_{i+1} = x_i^2 + 1 \bmod N$$

$$\mathbf{x_0 \to x_1 \to x_2 \to x_3 \to x_4 \to x_5 \to x_6 \to x_7 \to x_8 \to x_9 \ \ldots}$$

$$\mathbf{2 \to 5 \to 26 \to 677 \to 91864 \to 15449 \to 102236 \to 39678 \to 5749 \to 69062 \ \ldots}$$

mod 97:

$$\mathbf{2 \to 5 \to 26 \to 95 \to 5 \to 26 \to 95 \to 5 \to 26 \to 95 \ \ldots}$$



$x_1 \equiv x_4 \bmod q$

$q \mid (x_1 - x_4)$

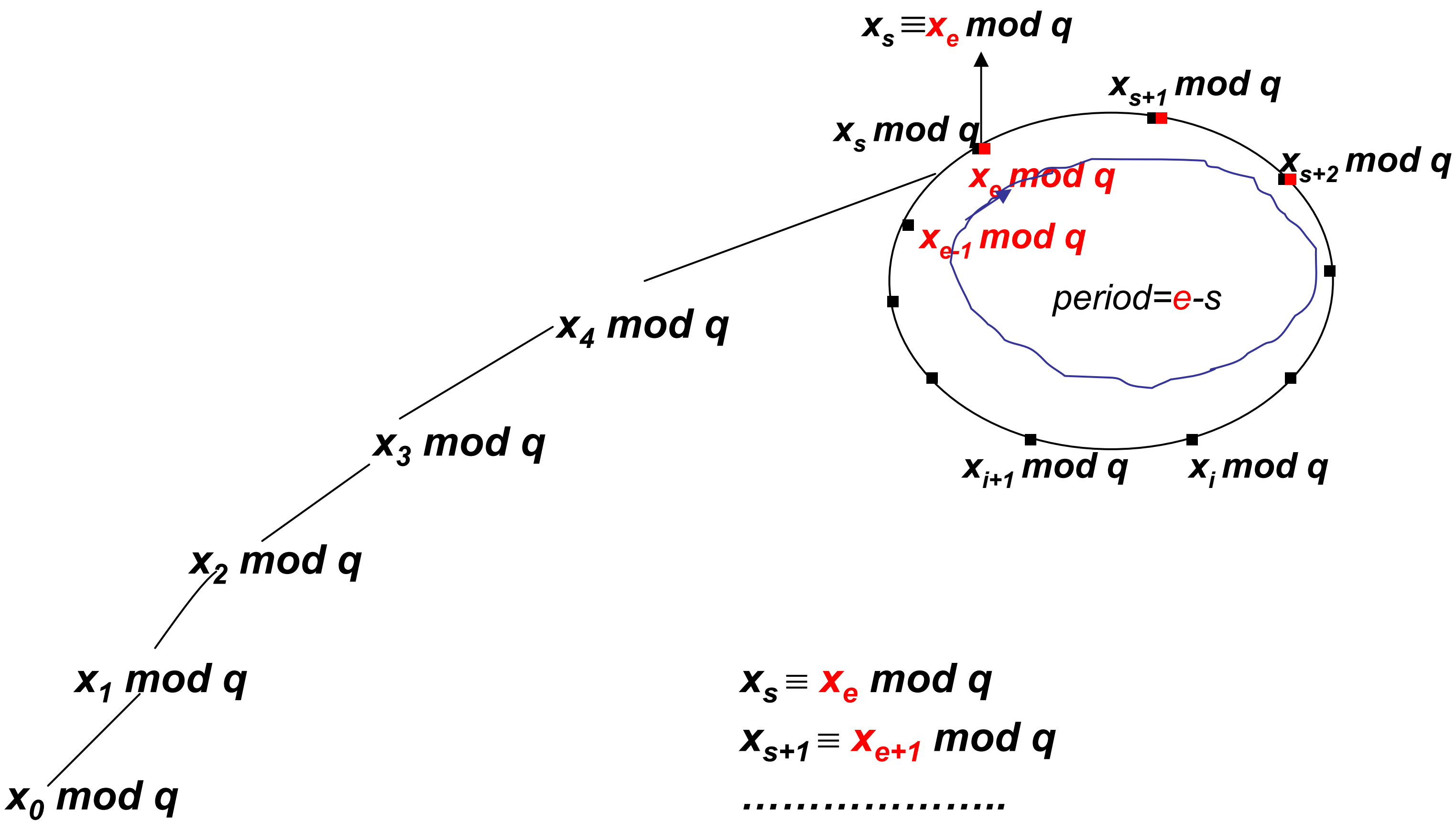
$q \mid N$

$q \mid \gcd(x_1 - x_4, N)$

$q = \gcd(-91\,859,\ 183\,233)$

$= \mathbf{97}$

Pollard's Rho Method



$$x_s \equiv x_e \bmod q$$

$$x_{s+1} \equiv x_{e+1} \bmod q$$

…………………

…………………

$$x_{s+k} \equiv x_{e+k} \bmod q$$

Rho Method - Floyd's Version

$\mathbf{x_1\text{-}x_2}$	$x_1\text{-}x_3$	$x_1\text{-}x_4$	$x_1\text{-}x_5$	$x_1\text{-}x_6$	--				$x_1\text{-}x_i$
$x_2\text{-}x_3$	$\mathbf{x_2\text{-}x_4}$	$x_2\text{-}x_5$	$x_2\text{-}x_6$	$x_2\text{-}x_7$	--				$x_2\text{-}x_i$
$x_3\text{-}x_4$	$x_3\text{-}x_5$	$\mathbf{x_3\text{-}x_6}$	$x_3\text{-}x_7$	$x_3\text{-}x_8$	--				$x_3\text{-}x_i$
$x_4\text{-}x_5$	$x_4\text{-}x_6$	$x_4\text{-}x_7$	$\mathbf{x_4\text{-}x_8}$	$x_4\text{-}x_9$	--				$x_4\text{-}x_i$
$x_5\text{-}x_6$	$x_5\text{-}x_7$	$x_5\text{-}x_8$	$x_5\text{-}x_9$	$\mathbf{x_5\text{-}x_{10}}$	--				$x_5\text{-}x_i$
$x_6\text{-}x_7$	$x_6\text{-}x_8$	$x_6\text{-}x_9$	$x_6\text{-}x_{10}$	$x_6\text{-}x_{11}$	$\mathbf{x_6\text{-}x_{12}}$	---------------------------------------			$x_6\text{-}x_i$
$x_7\text{-}x_8$	$x_7\text{-}x_9$	$x_7\text{-}x_{10}$	$x_7\text{-}x_{11}$	$x_7\text{-}x_{12}$	$x_7\text{-}x_{13}$	$\mathbf{x_7\text{-}x_{14}}$	-------------------------		$x_7\text{-}x_i$
$x_8\text{-}x_9$	$x_8\text{-}x_{10}$	$x_8\text{-}x_{11}$	$x_8\text{-}x_{12}$	$x_8\text{-}x_{13}$	$x_8\text{-}x_{14}$	$x_8\text{-}x_{15}$	$\mathbf{x_8\text{-}x_{16}}$	---------------	$x_8\text{-}x_i$

$x_k\text{-}x_{k+1}$ $x_k\text{-}x_{k+2}$ $x_k\text{-}x_{k+3}$ -- $\mathbf{x_k\text{-}x_{2k}}$ --------- $x_k\text{-}x_i$

Pollard's Rho Algorithm - Floyd's Version

$f(x)=x^2+a$ with $a \notin \{-2,0\}$

 # iterations $t < 100\sqrt{q_{max}}$ (q_{max} is the maximum factor we expect to find using rho method)

We choose random x_0 in the range(0,N-1) and $x_1=f(x_0)$

	x_0	
	$\downarrow$	$d=1$
$x_2 \leftarrow$	x_1	$d=d*(x_2-x_1)$
$\downarrow f(f())$	$\downarrow f()$	
x_4	x_2	$d=d*(x_4-x_2)$
$\downarrow$	$\downarrow$	
x_6	x_3	$d=d*(x_6-x_3)$
x_t	$x_{t/2}$	$d=d*(x_t-x_{t/2})$
$\downarrow$	$\downarrow$	
x_{t+2}	$x_{(t+2)/2}$	$d=d*(x_{t+2}-x_{(t+2)/2})$
x_{2i}	x_i	$d=d*(x_{2i}-x_i)$
$\downarrow$	$\downarrow$	
$x_{2(i+1)}$	x_{i+1}	$d=d*(x_{2i+2}-x_{i+1})$
x_{2t}	x_t	$d=d*(x_{2t}-x_t)$

*$x_{2i+2}=f(f(x_{2i})), x_{i+1}=f(x_i)$

$q=gcd(d,N)$

Can be eliminated

Minimization for area and/or memory

Rho Method - Brent's Version

x_1-x_2 x_1-x_3 x_1-x_4 x_1-x_5 x_1-x_6 -- x_1-x_i

x_2-x_3 x_2-x_4 x_2-x_5 x_2-x_6 x_2-x_7 -- x_2-x_i

x_3-x_4 x_3-x_5 x_3-x_6 x_3-x_7 x_3-x_8 -- x_3-x_i

x_4-x_5 x_4-x_6 x_4-x_7 x_4-x_8 x_4-x_9 -- x_4-x_i

x_5-x_6 x_5-x_7 x_5-x_8 x_5-x_9 x_5-x_{10} -- x_5-x_i

x_6-x_7 x_6-x_8 x_6-x_9 x_6-x_{10} x_6-x_{11} x_6-x_{12} -------------------------------------- x_6-x_i

x_7-x_8 x_7-x_9 x_7-x_{10} x_7-x_{11} x_7-x_{12} x_7-x_{13} x_7-x_{14} ------------------------- x_7-x_i

x_8-x_9 x_8-x_{10} x_8-x_{11} x_8-x_{12} x_8-x_{13} x_8-x_{14} x_8-x_{15} x_8-x_{16} --------------- x_8-x_i

----

x_k-x_{k+1} x_k-x_{k+2} x_k-x_{k+3} ---------------- $x_{2^k}-x_{2^k+2^{k-1}+1}$ -- $x_{2^k}-x_{2^{k+1}}$

Rho Method - Brent's Version
Sequence of Operations

v_2	d	v_1
x_2	$d=1$	x_2
x_3		
x_4	$d=d^*(x_4-x_2)$	x_4
x_5		
x_6		
x_7	$d^*(x_7-x_4)$	
x_8	$d^*(x_8-x_4)$	x_8
x_9		
x_{10}		
x_{11}		
x_{12}		
x_{13}	$d^*(x_{13}-x_8)$	
x_{14}	$d^*(x_{14}-x_8)$	
x_{15}	$d^*(x_{15}-x_8)$	
x_{16}	$d^*(x_{16}-x_8)$	x_{16}

Can be eliminated

Minimization for execution time 24%

p-1 algorithm

Inputs :

N – number to be factored

a – arbitrary integer such that gcd(a, N)=1

B_1 – smoothness bound for Phase1

B_2 – smoothness bound for Phase2

Outputs:

q - factor of N, $1 < q \leq N$

or FAIL

p-1 algorithm – Phase 1

precomputations

1: $k \leftarrow \prod_{p_i} p_i^{e_i}$ such that p_i - consecutive primes $\leq B_1$

e_i - largest exponent such that $p_i^{e_i} \leq B_1$

2: $q_0 \leftarrow a^k \bmod N$ *main computations*

---

postcomputations

3: $q \leftarrow \gcd(q_0 - 1, N)$

4: if $q > 1$

5: return q (factor of N)

6: else

7: go to Phase 2

8: end if

p-1 algorithm – Phase 2

$d \leftarrow 1$

for each prime $p = B_1$ to B_2 do

  $d \leftarrow d \cdot (q_0{}^p - 1) \pmod{N}$ *main computations*

end for

$q \leftarrow \gcd(d, N)$

if $q > 1$ then *postcomputations*

  return q

else

  return FAIL

end if

p-1 Phase 1 – Numerical example

N = 1 740 719 = 1279·1361

$a = 2$

$B_1 = 20$

$k = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 232\,792\,560$

$q_0 = a^k \bmod N = 2^{232\,792\,560} \bmod 1\,740\,719 = 1\,003\,058$

$q = \gcd(1\,003\,058 - 1;\ 1\,740\,719) =$ **1361**

- -

Why did the method work?

$q-1 = 1360 = 2 \cdot 5 \cdot 17 \mid k$

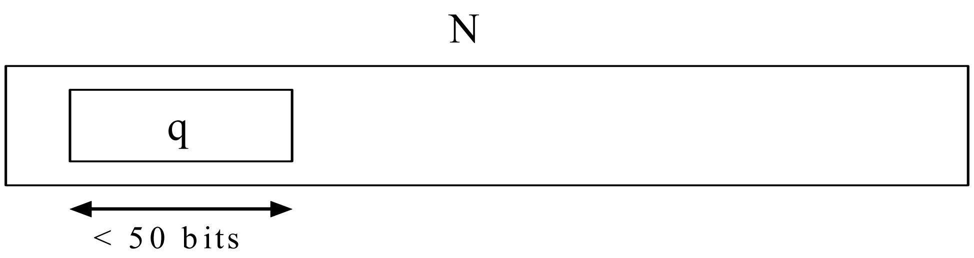
$a^k \bmod q = a^{(q-1) \cdot m} \bmod q = 1$

$q \mid a^k - 1$

What is ECM?

Elliptic Curve Method of Factoring

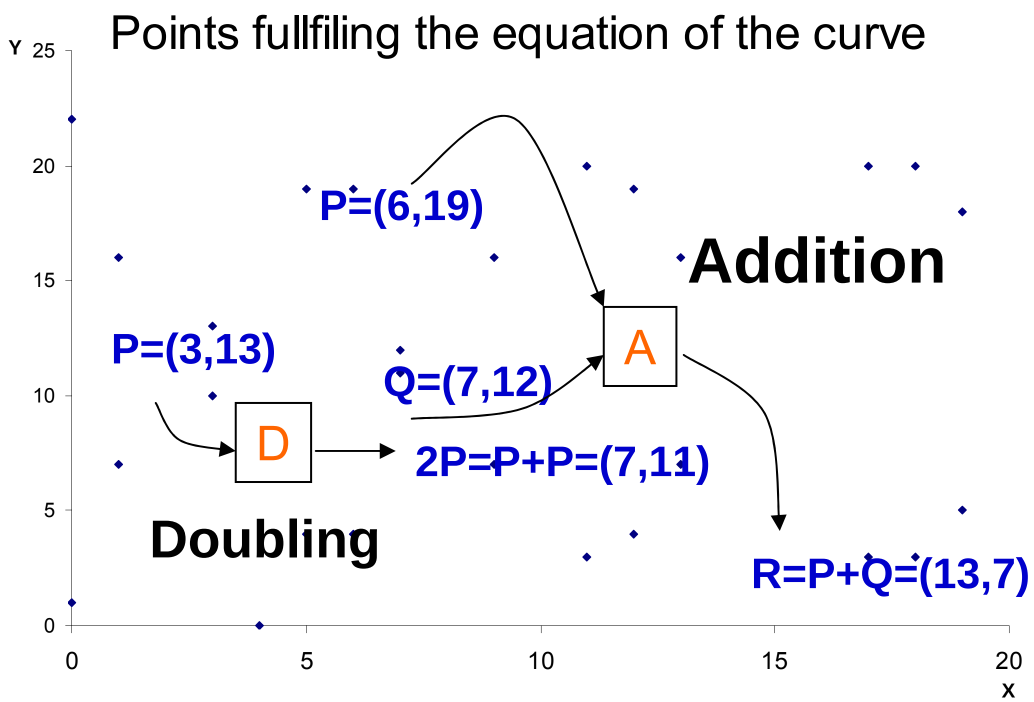
Lenstra	1985	Phase 1
Brent, Montgomery	1986-87	Phase 2



Factoring time depends mainly on the size of factor q

Elliptic Curve

$$Y^2 = X^3 + X + 1 \mod \text{p} \qquad (p = 23)$$



+ special point ϑ (point at infinity) such that:

$$P + \vartheta = \vartheta + P = P$$

$$\exists\, P: \qquad \underbrace{P, 2P, 3P, \ldots\ldots\ldots\ldots, nP = \vartheta}_{\text{all points of the curve}}, (n+1)P = P, 2P$$

Projective vs. Affine coordinates

- affine coordinates $P_a=(X_P, Y_P)$
  - addition and doubling **require inversion**
- projective coordinates $P_p=(x_P, y_P, z_P)$
  - addition and doubling can be done **without inversion**
- projective coordinates for Montgomery form of the curve
  - addition and doubling **do not require y coordinate**

    (y coordinate can be recovered from x and z at the end of a long chain of computations)

$$P_{pM} = (x_P : \; : z_P)$$

$$\vartheta = (0 : \; : 0)$$

Scalar Multiplication

$$Q = k \,.\, P = \underbrace{P + P + P + \text{- - - - - - - - -} + P}_{k\text{- times}}$$

point

number
(scalar)

point

ECM Algorithm

Inputs :

N – number to be factored

E – elliptic curve

P_0 – point of the curve E : initial point

B_1 – smoothness bound for Phase1

B_2 – smoothness bound for Phase2

Outputs:

q - factor of N, $1 < q \leq N$

or FAIL

ECM algorithm – Phase 1

precomputations

1: $k \leftarrow \prod_{p_i} p_i^{e_i}$ such that p_i - consecutive primes $\leq B_1$

e_i - largest exponent such that $p_i^{e_i} \leq B_1$

2: $Q_0 \leftarrow kP_0 = (x_{Q_0} : \; : z_{Q_0})$ *main computations*

---

postcomputations

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

4: if $q > 1$

5: return q (factor of N)

6: else

7: go to Phase 2

8: end if

ECM algorithm – Phase 2

$d \leftarrow 1$

for each prime $p = B_1$ to B_2 do

  $(x_{pQ_0}, y_{pQ_0}, z_{pQ_0}) \leftarrow pQ_0$

  $d \leftarrow d \cdot z_{pQ_0} \pmod{N}$

end for

main computations

$q \leftarrow \gcd(d,\ N)$

if $q > 1$ then

  return q

else

  return FAIL

end if

postcomputations

ECM Phase 1 – Numerical example

$N = 1\,740\,719 = 1279 \cdot 1361$

$E : y^2 = x^3 + 30x + 1 \quad \pmod{1\,740\,719}$

$P_0 = (4 : \; : 1)$

$B_1 = 20$

$k = 2^4 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 232\,792\,560$

$kP_0 = (256\,230 : \quad : 1\,242\,593)$

gcd (1 242 593; 1 740 719) = **1361**

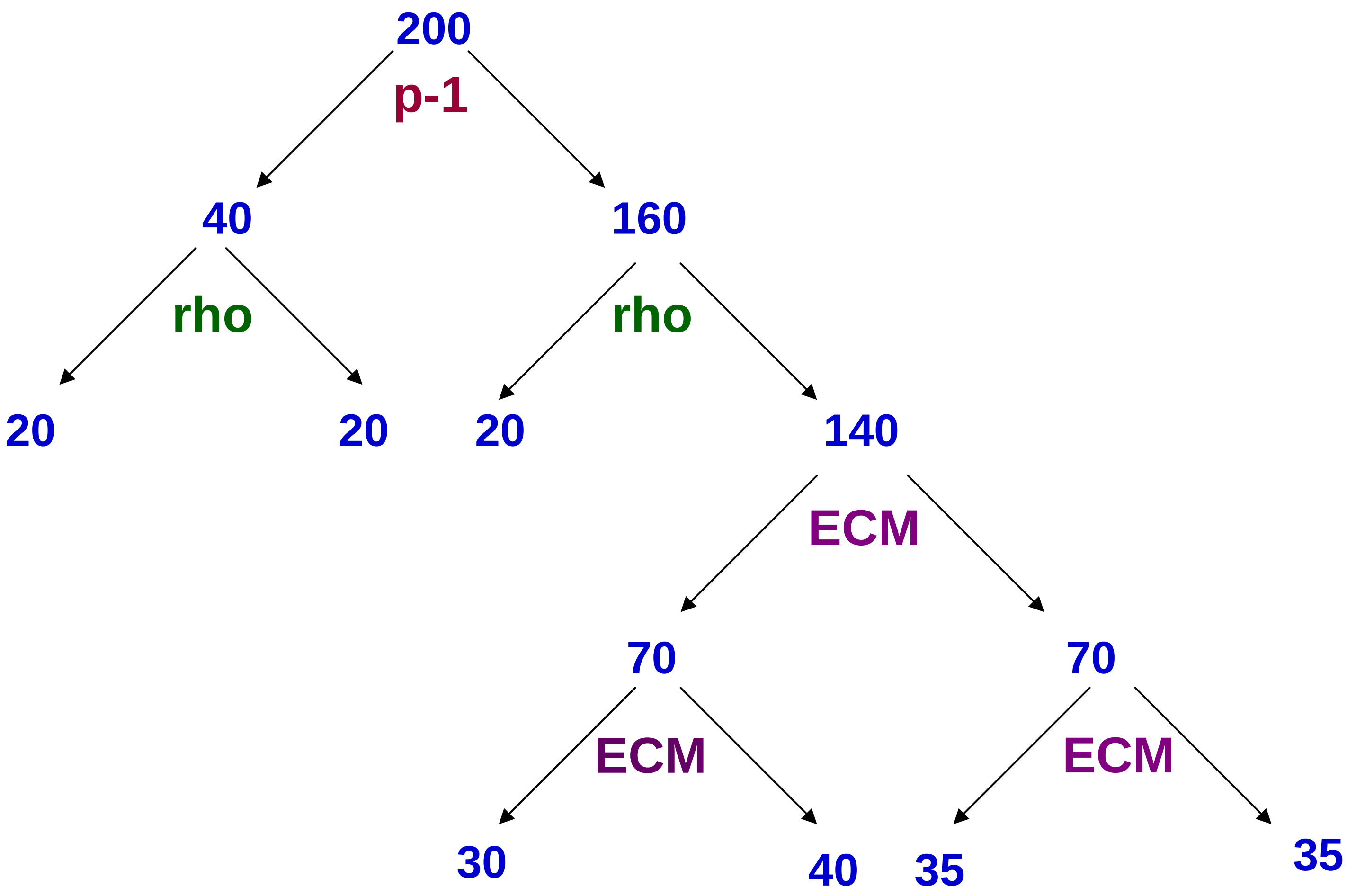
Why does the method work?

$\#E = 1326 = 2 \cdot 3 \cdot 13 \cdot 17 \mid k \quad$ (over GF(1361))

$kP_0 = m \cdot \#E \cdot P_0 = O = (0:1:0) \qquad$ (over GF(1361))

$z_{Q0} = 0 \bmod 1361$

Possible use of three special purpose factoring methods in NFS



Choice of parameters

ECM method

B1 ↗ B2 ↗

$$\frac{\text{Probability of success} \nearrow}{\text{Execution time} \nearrow}$$

Optimized analytically for the maximum ratio ⇒ $\dfrac{\text{Probability of success}}{\text{Execution time}}$

⇓

B1 = 960 B2=57000

---

p-1 method

The same as in ECM

---

rho method

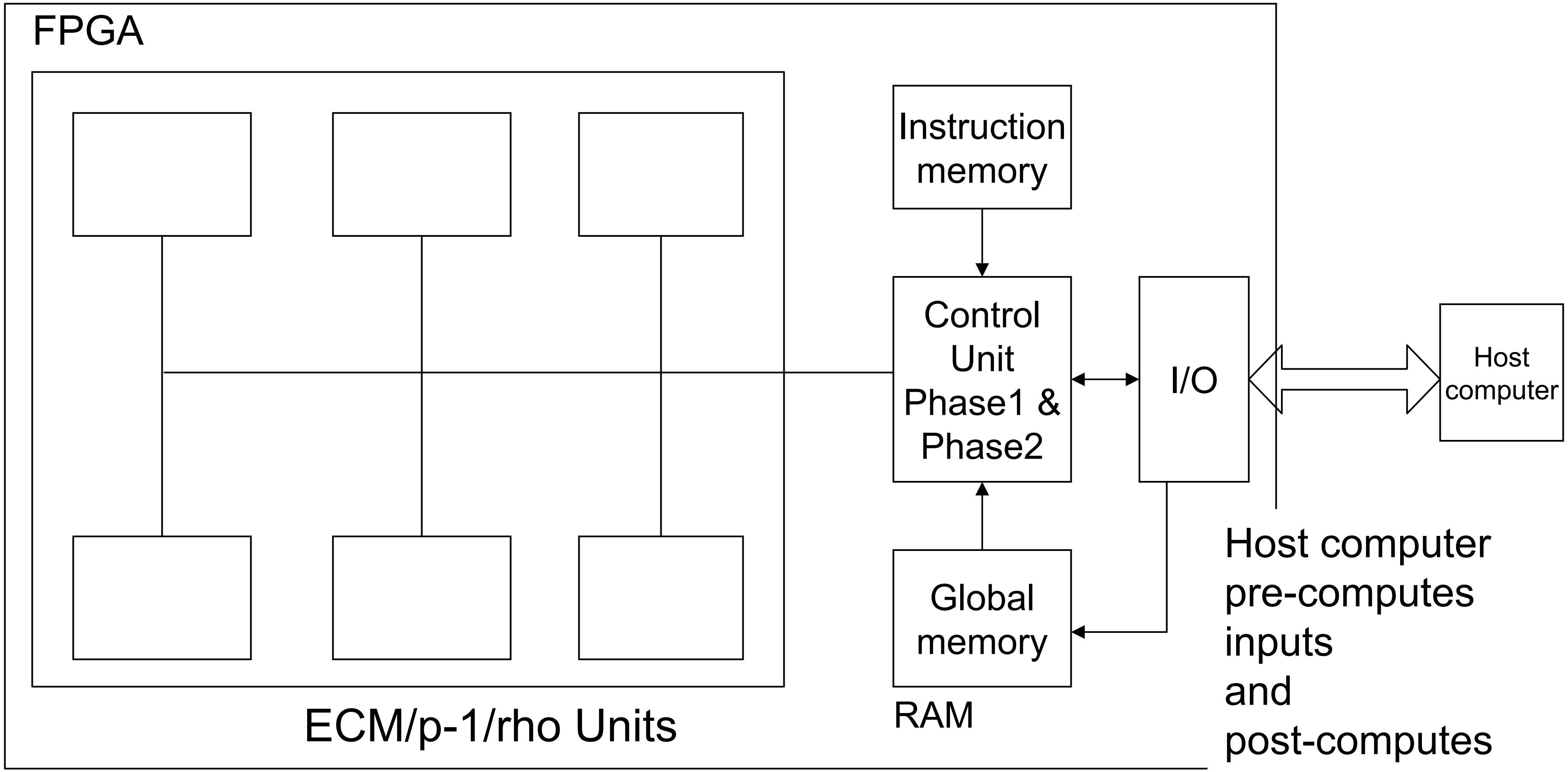
t = 8192

Approximately the same execution time as a single run of ECM.
Likely to find factors up to 2^{26}

Rho, P-1 & ECM in Hardware

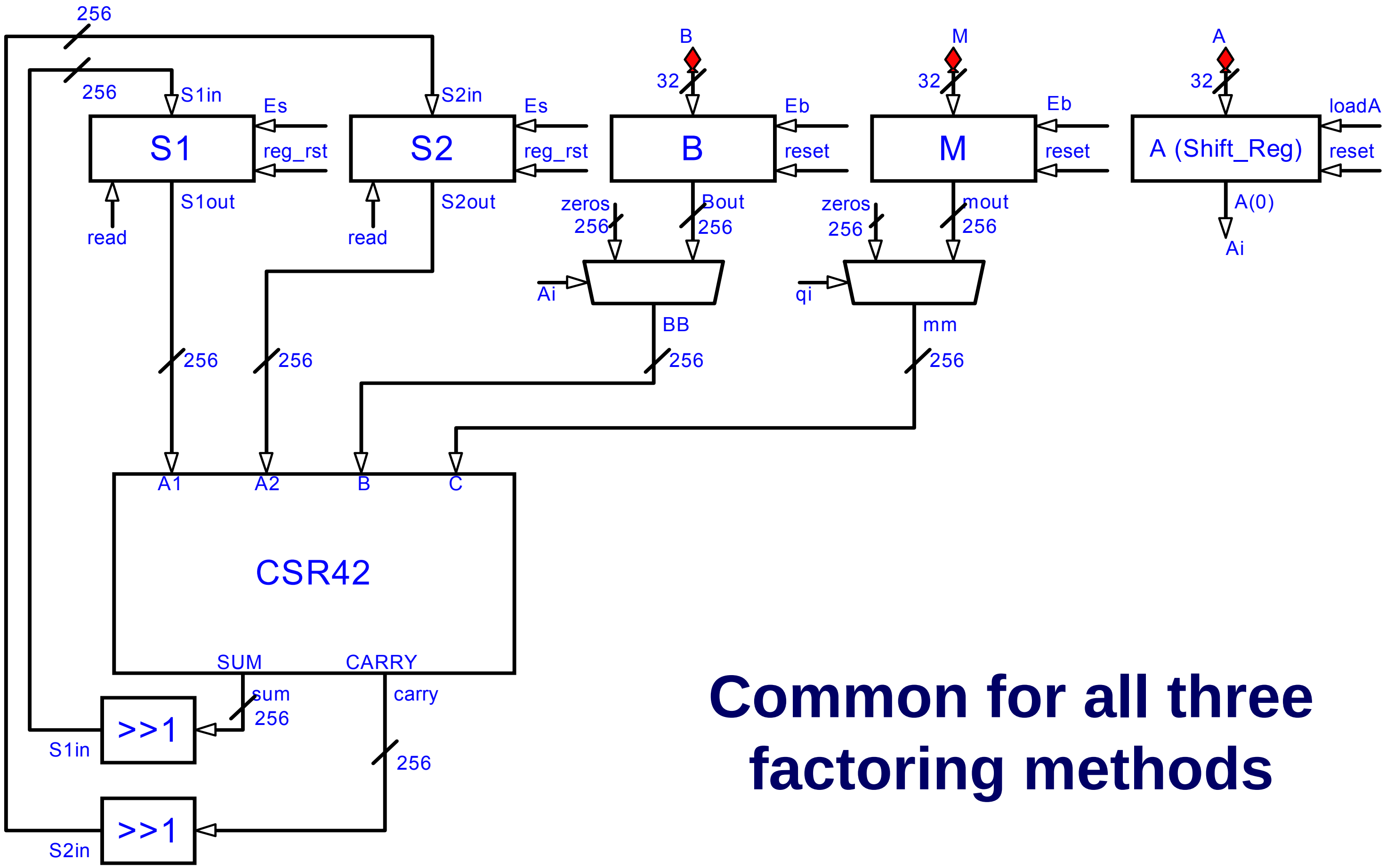
FPGA IMPLEMENTATION

Our architecture : Top-level view



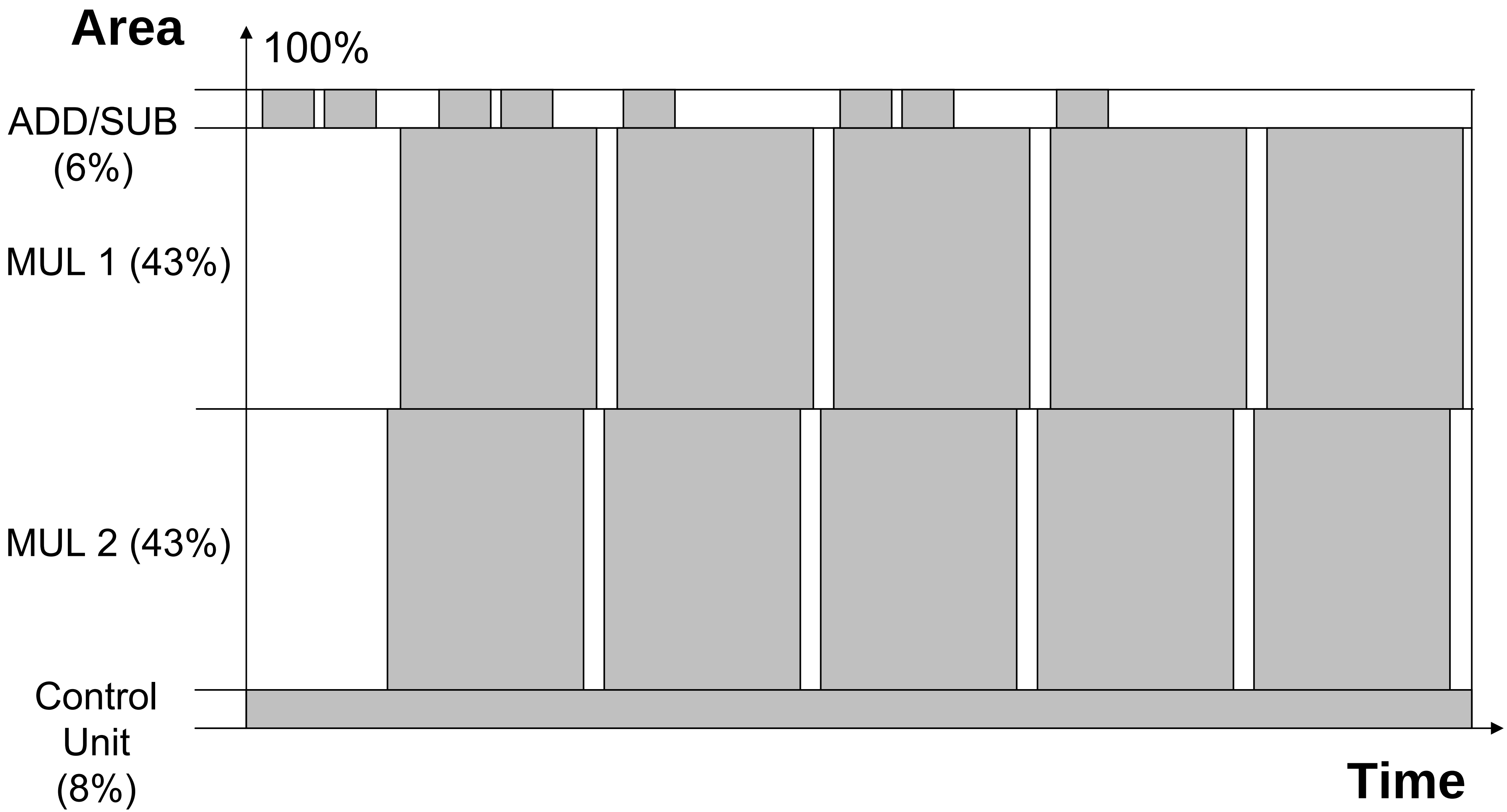
Host computer pre-computes inputs and post-computes final gcds

Basic Building Block – Montgomery Multiplier

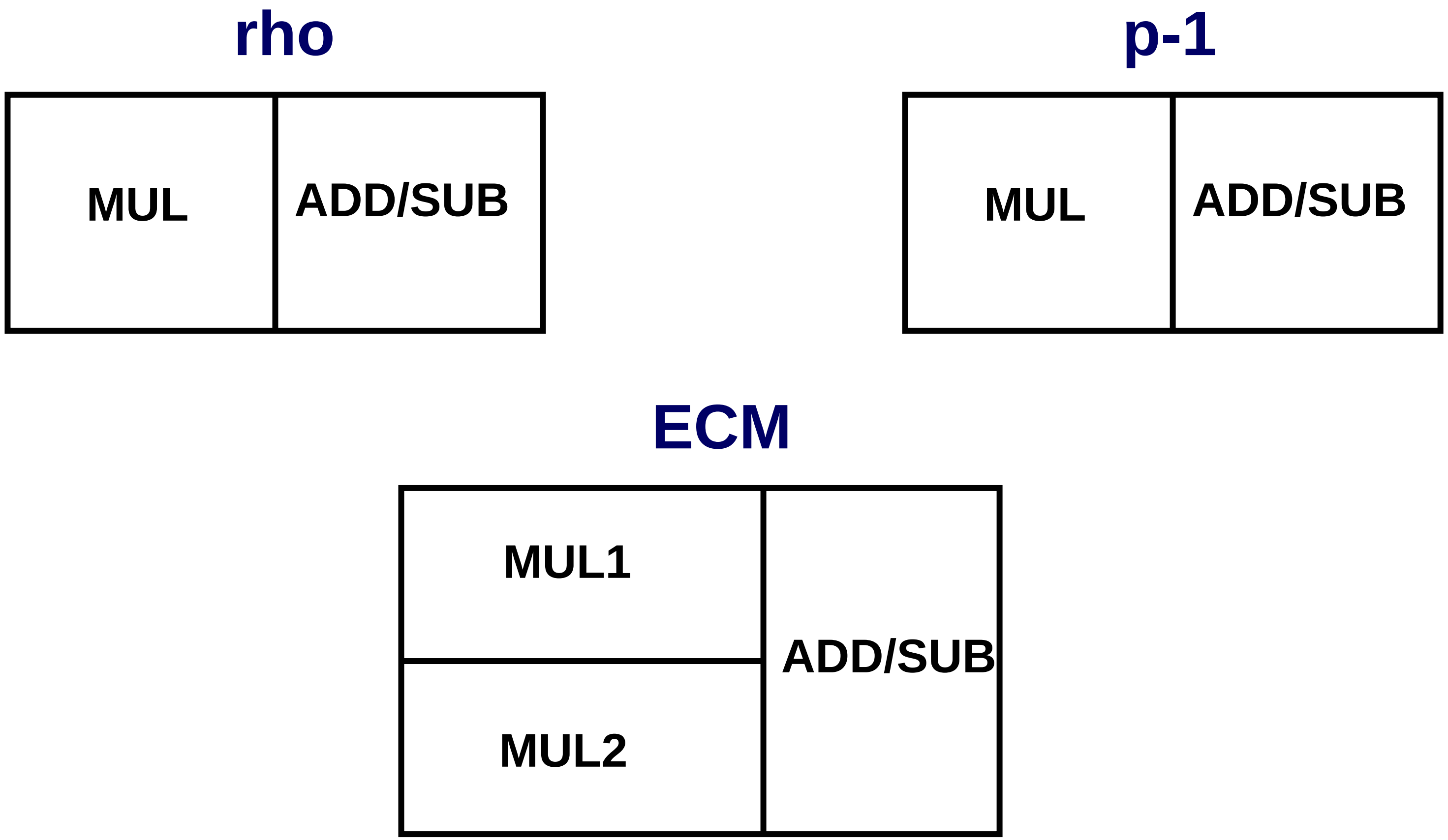


Common for all three factoring methods

Resources utilization in time
ECM Phase 1



Optimum Execution Unit



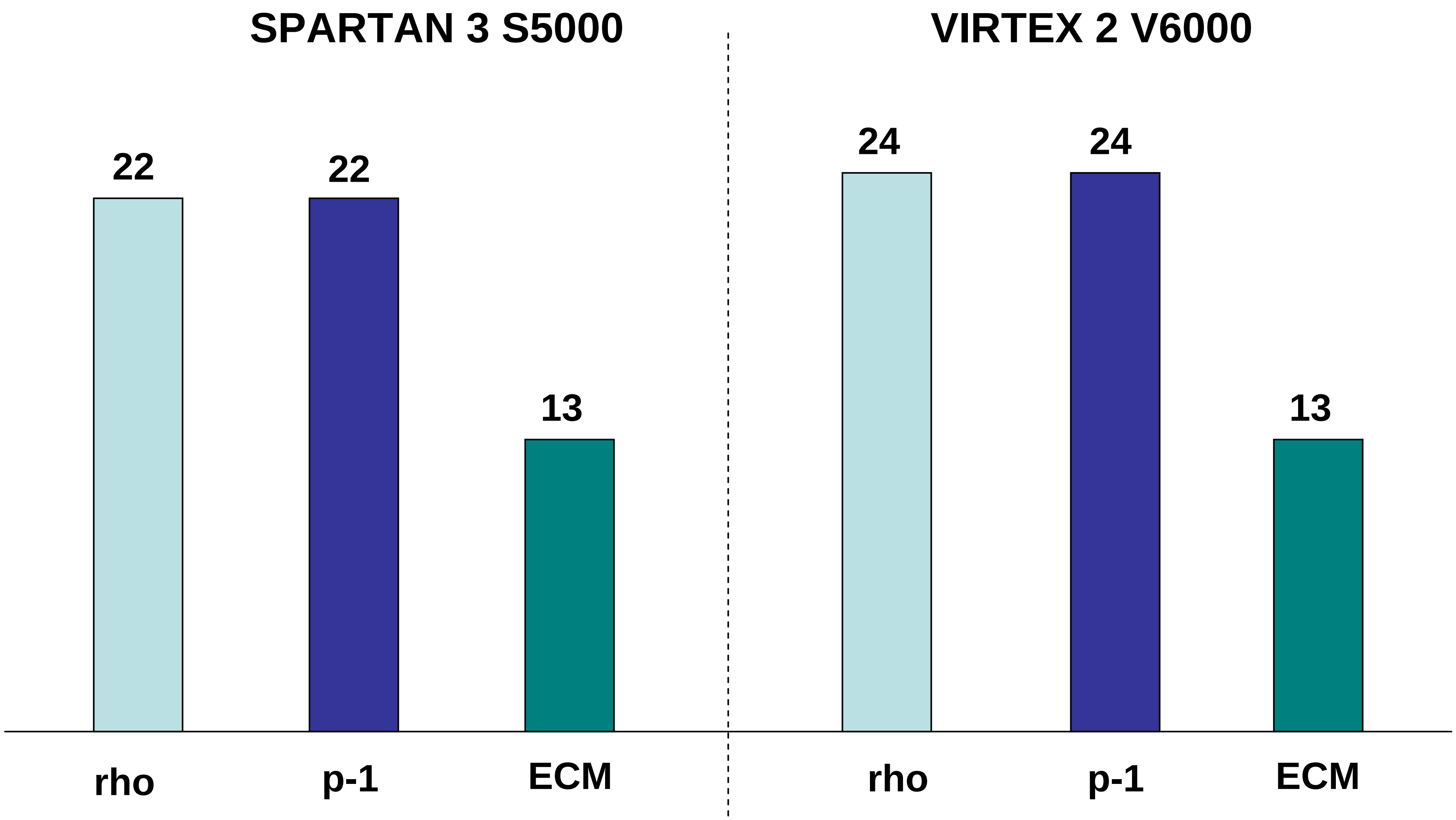
Parameters used in performance testing

N: 198 – bit number

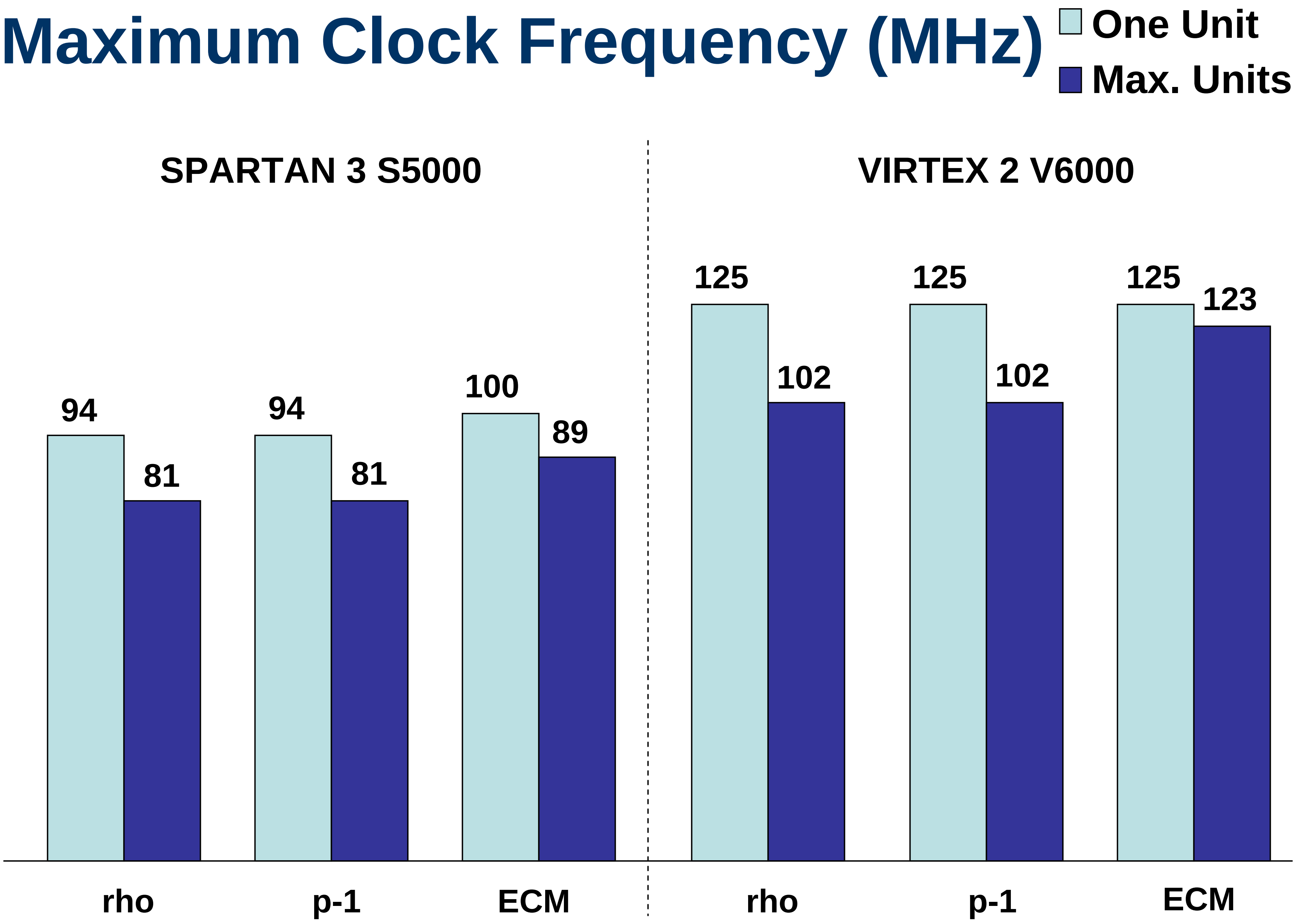
$B_1 = 960$

$B_2 = 57{,}000$

Maximum Number of Execution Units per FPGA Device



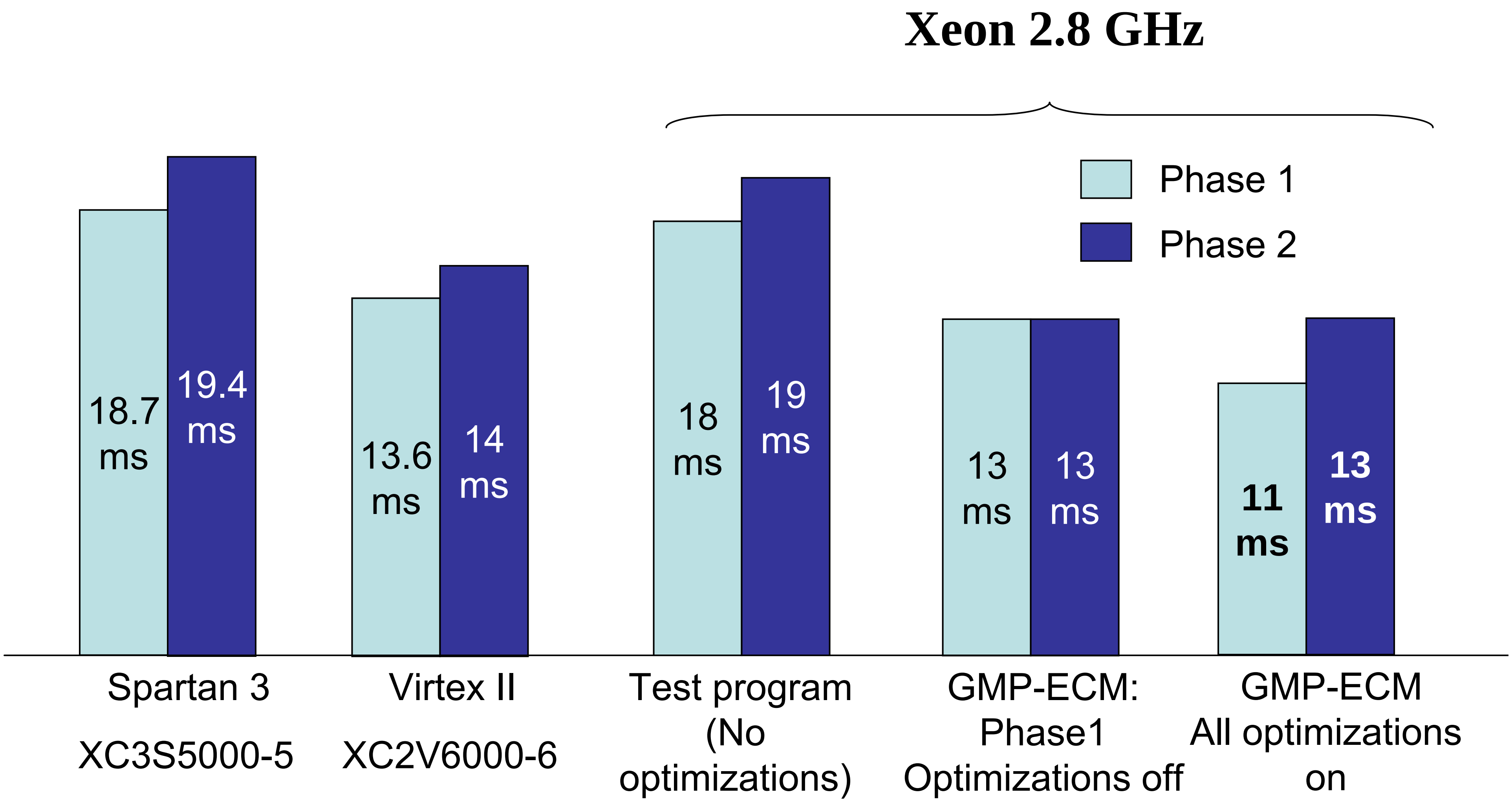
Maximum Clock Frequency (MHz)



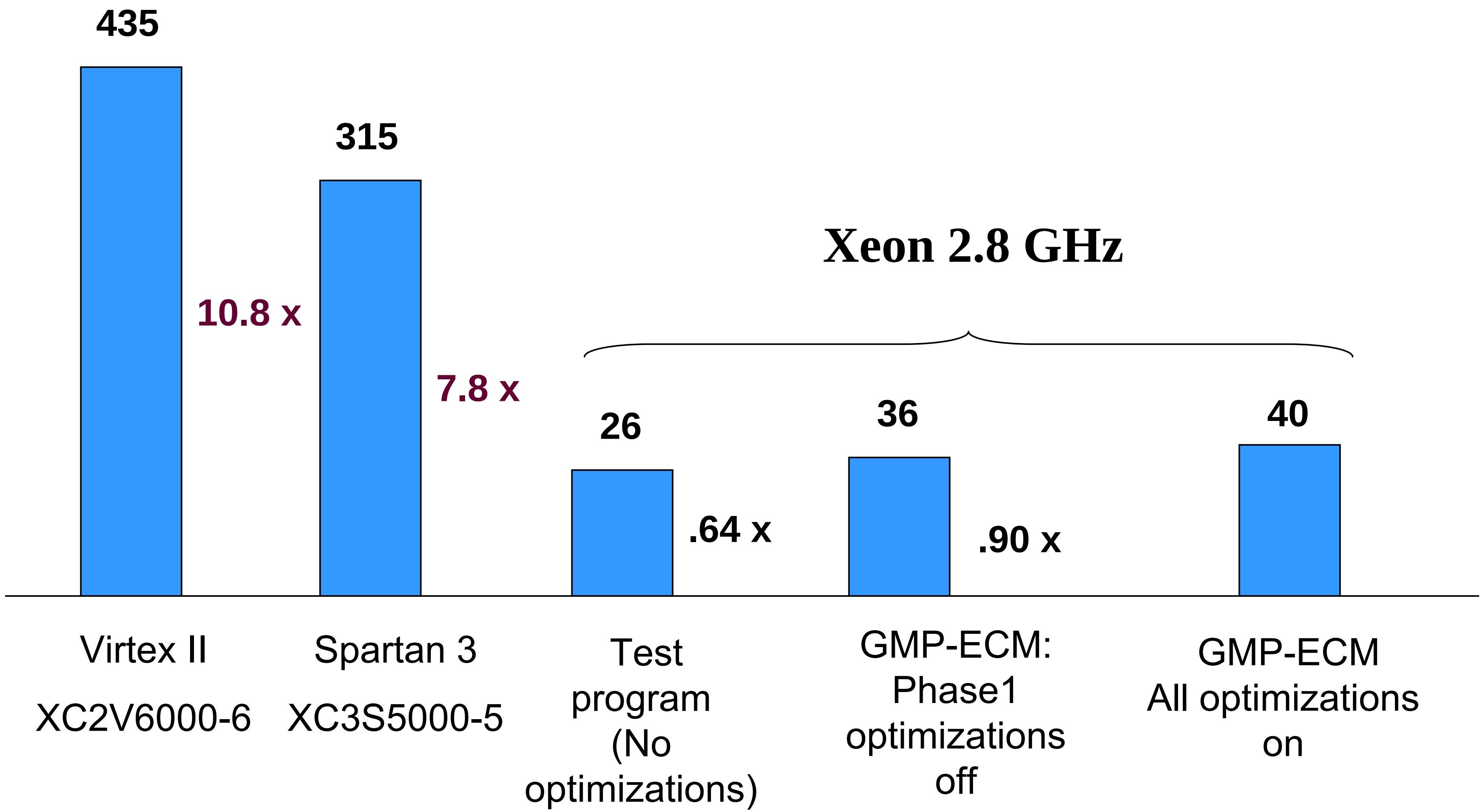
Software Implementation
GMP-ECM running on Intel Xeon 2.8GHz

	Phase 1	Phase 2
Elliptic Curve	Montgomery form: $by^2z = x^3+ax^2z+xz^2$	Weierstrass form: $Y^2 = X^3+AX+B$
Coordinate	Projective	Affine
Optimization Techniques (Reducing Time)	Lucas chain (PRAC algorithm)	Fast polynomial multiplication Montgomery's D_1D_2 method
Optimization Techniques (Incresing Probability)		Brent-Suyama extension
Porting Optimizations To Hardware	Possible with pre-computations in software	Inverter required. Large amounts of memory required

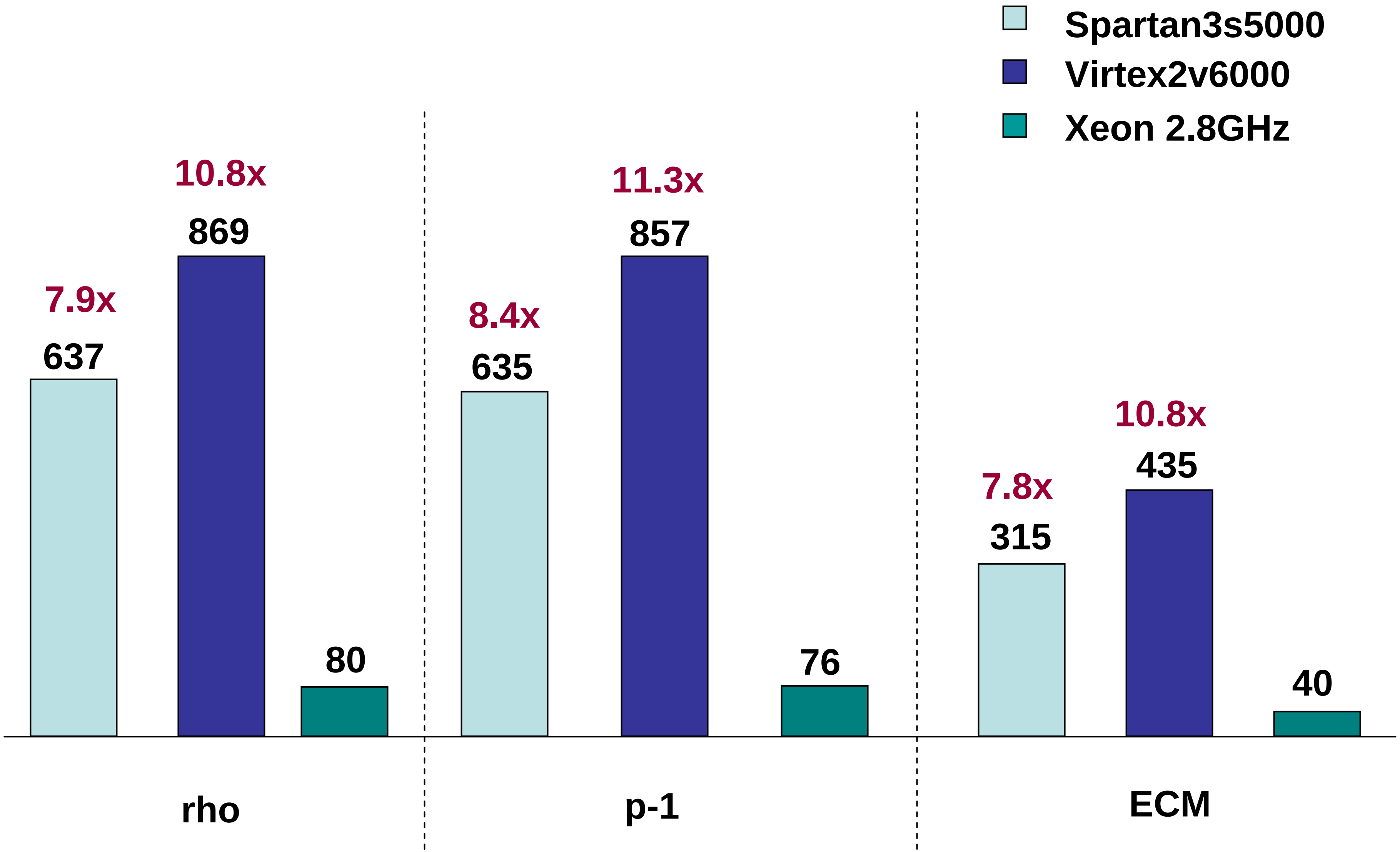
FPGAs vs. Microprocessor ECM Execution Time



FPGAs vs Microprocessors
Phase 1 & Phase 2
ECM computations per second



Factoring Runs per Second



Comparison with the Proof-of-Concept Design by Pelzl and Šimka

Time x Area Product

Assuming the same memory management (i.e., improved memory management in Pelzl/Šimka):

	Improvement
Phase 1	**x 3.7**
Phase 2	**x 6.4**

Effect of using more aggressive parameters of ECM

Our ECM parameters

B1 = 960
B2 = 57000
D = $2 \cdot 3 \cdot 5 \cdot 7 = 210$

Parameters according to Rainer Steinwandt et al.

B1 = **402**
B2 = **9680**
D = $2 \cdot 3^2 \cdot 5 = 90$

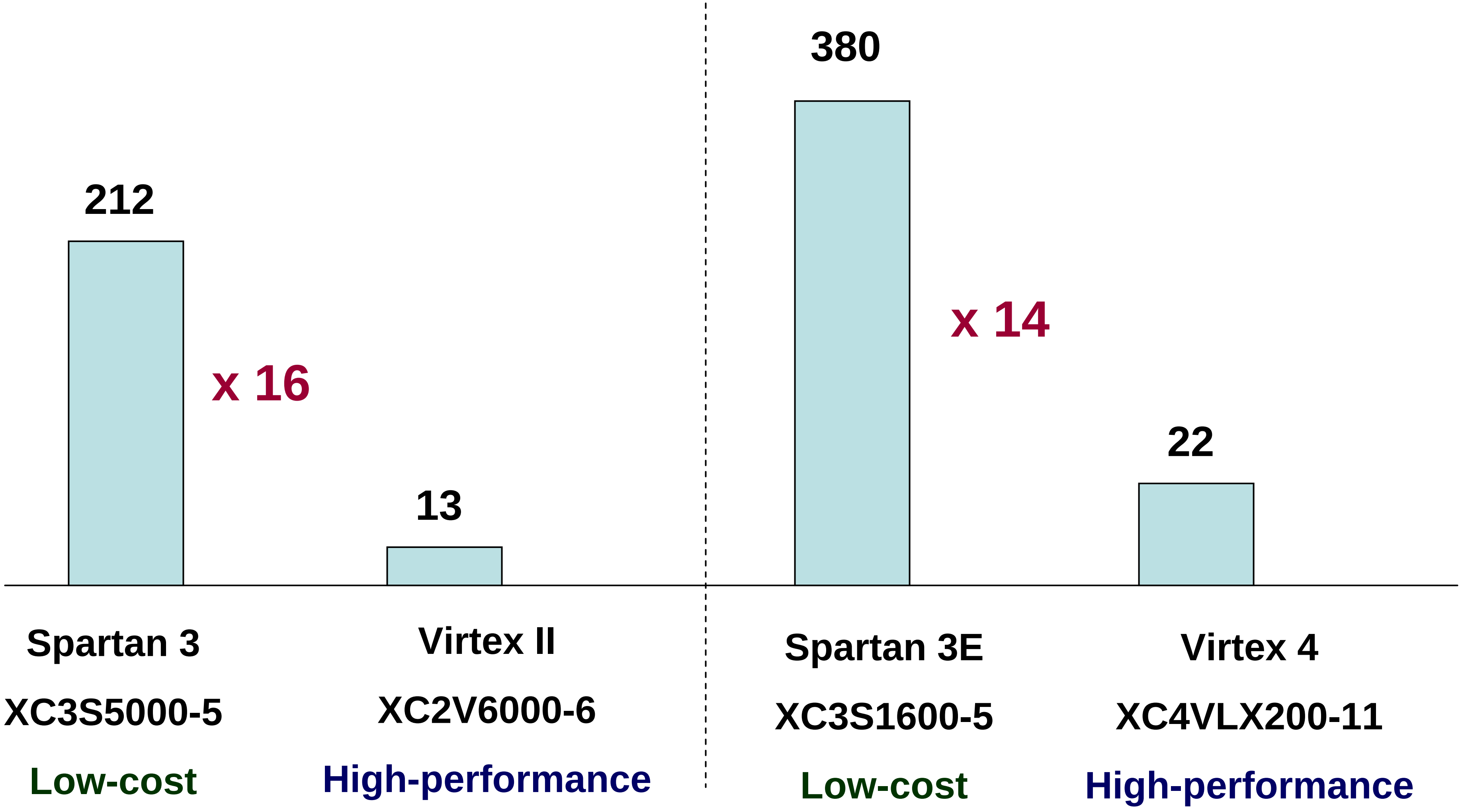
Execution time of Phase 1 & Phase 2 in hardware

36.6 ms → 3.5 x → 10.4 ms

Execution time of Phase 1 & Phase 2 in software (GMP-ECM)

24.8 ms → 2.8 x → 8.8 ms

Performance to cost ratio
Number of Phase 1 & Phase 2 ECM operations per second per $100



ASIC IMPLEMENTATION

ASIC Technology

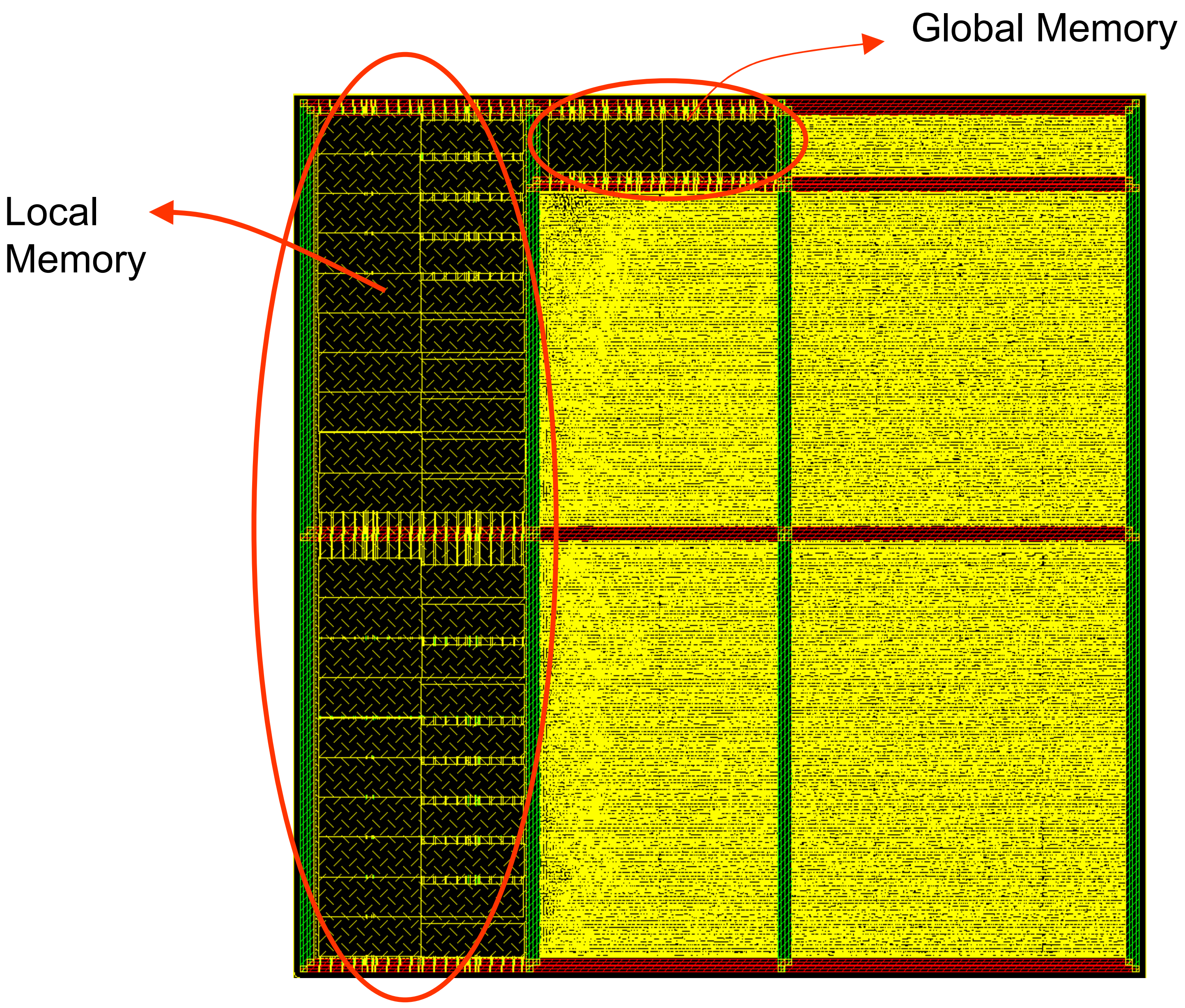
Process:
0.13 μm

Libraries:
cb13fs120_tsmc_max

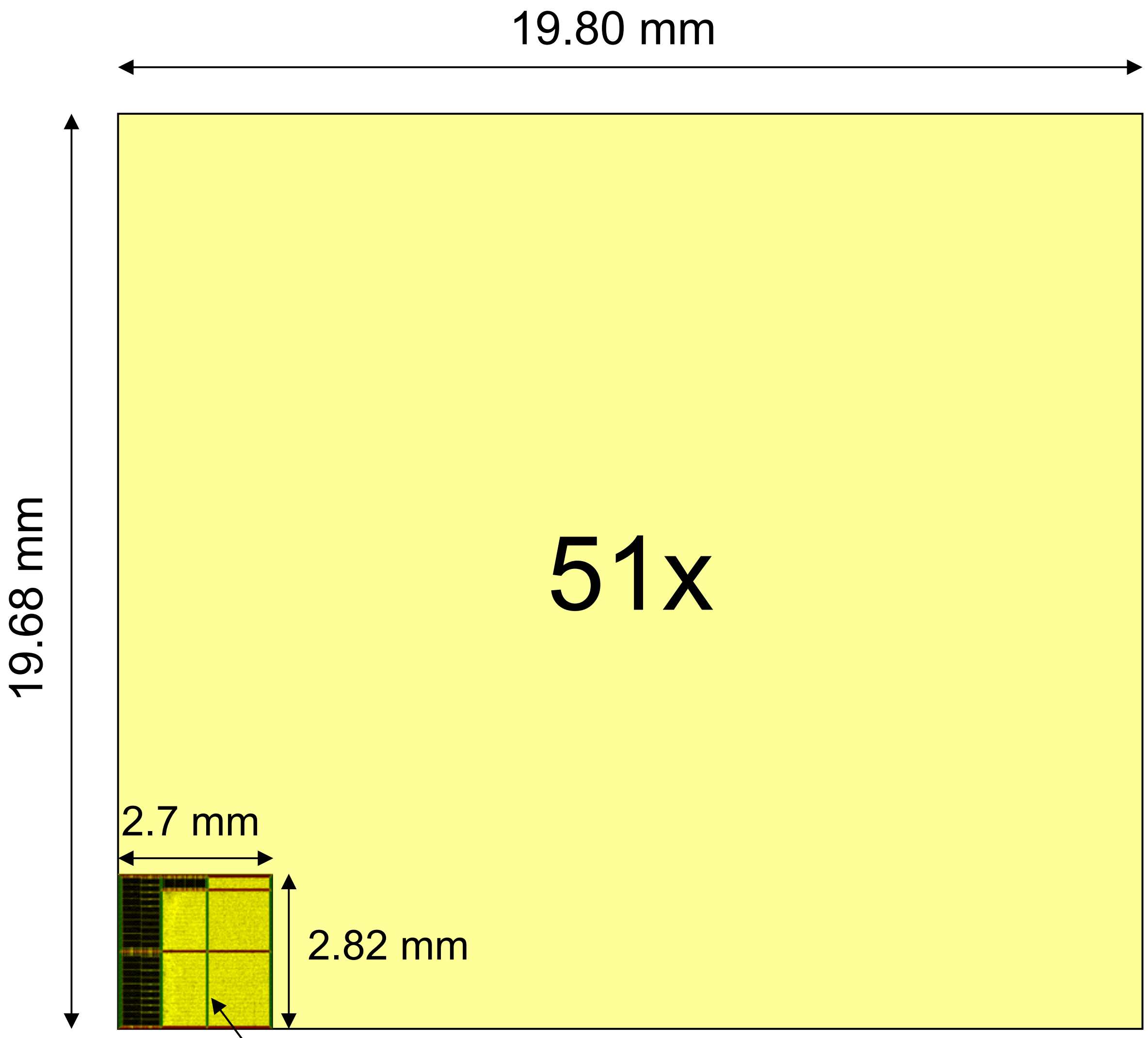
Memory Library:
ram16x128_max, ram32x64_max, ram32x32_max

Tools:
Synopsys Design Analyzer, Astro, Primetime

Rho in an ASIC 130 nm



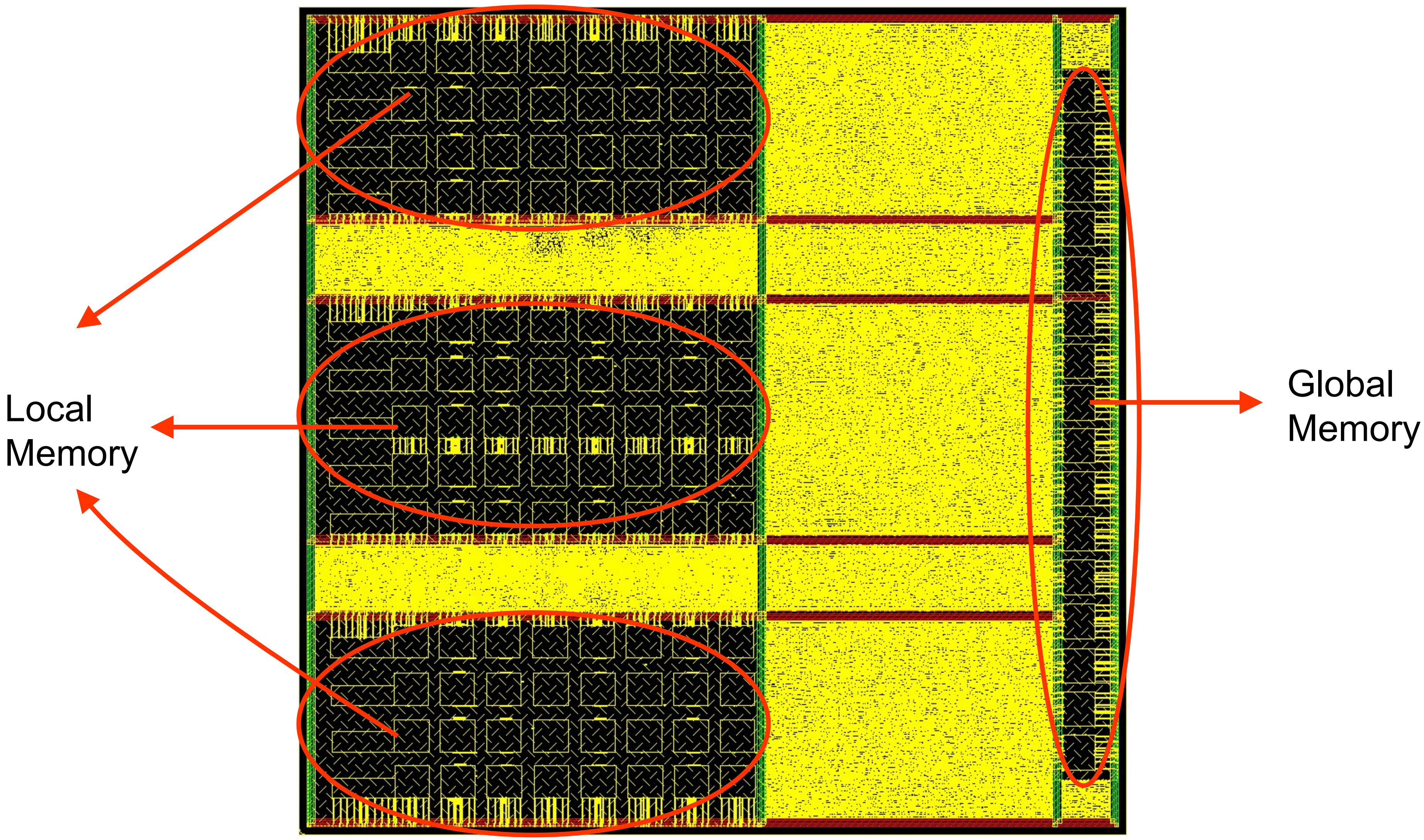
ASIC 130 nm vs. Virtex II 6000 – rho (24 units)



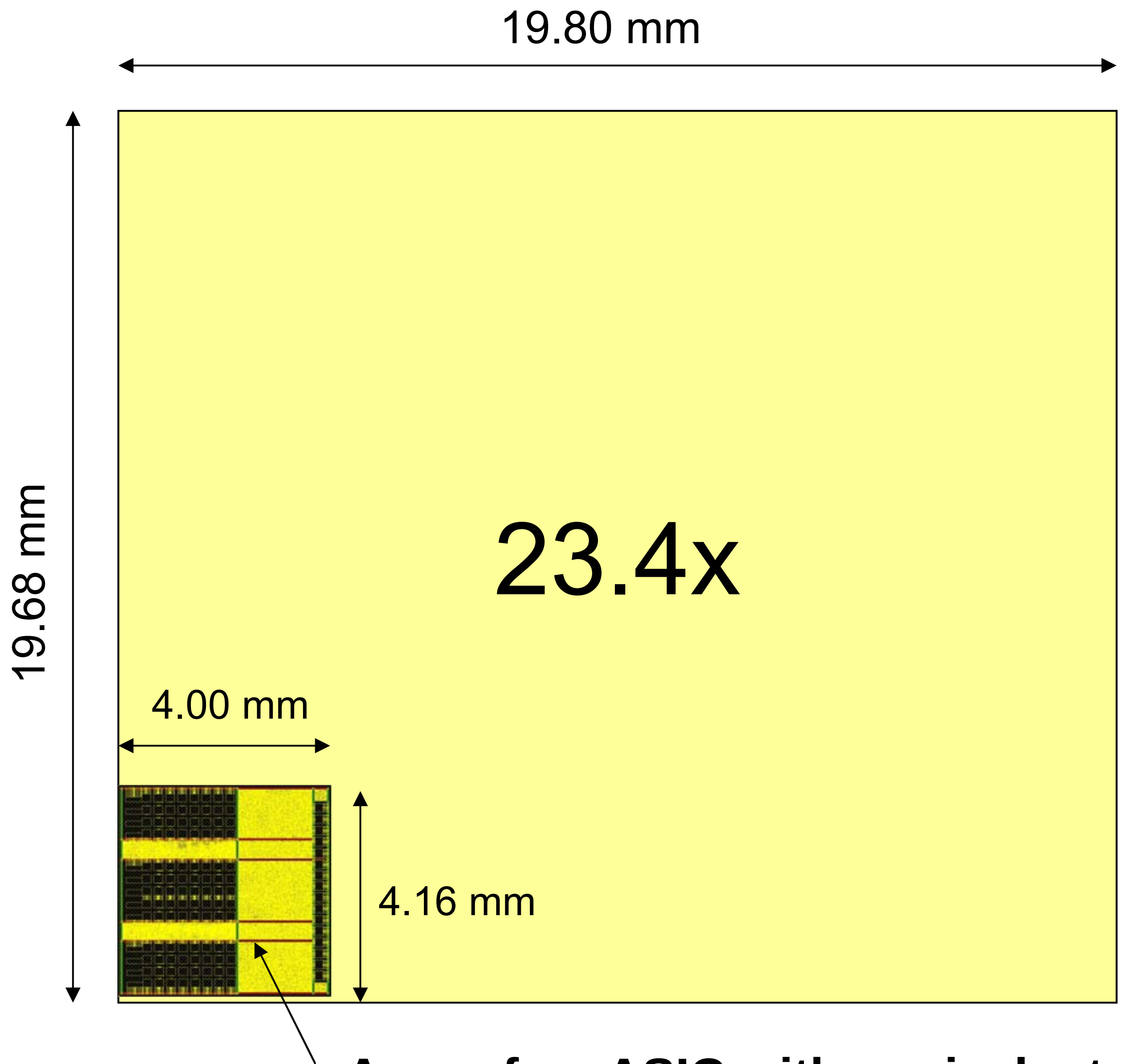
Area of Virtex II 6000
(estimation by R.J. Lim Fong,
MS Thesis, VPI, 2004)

Area of an ASIC with equivalent functionality

ECM in an ASIC 130 nm



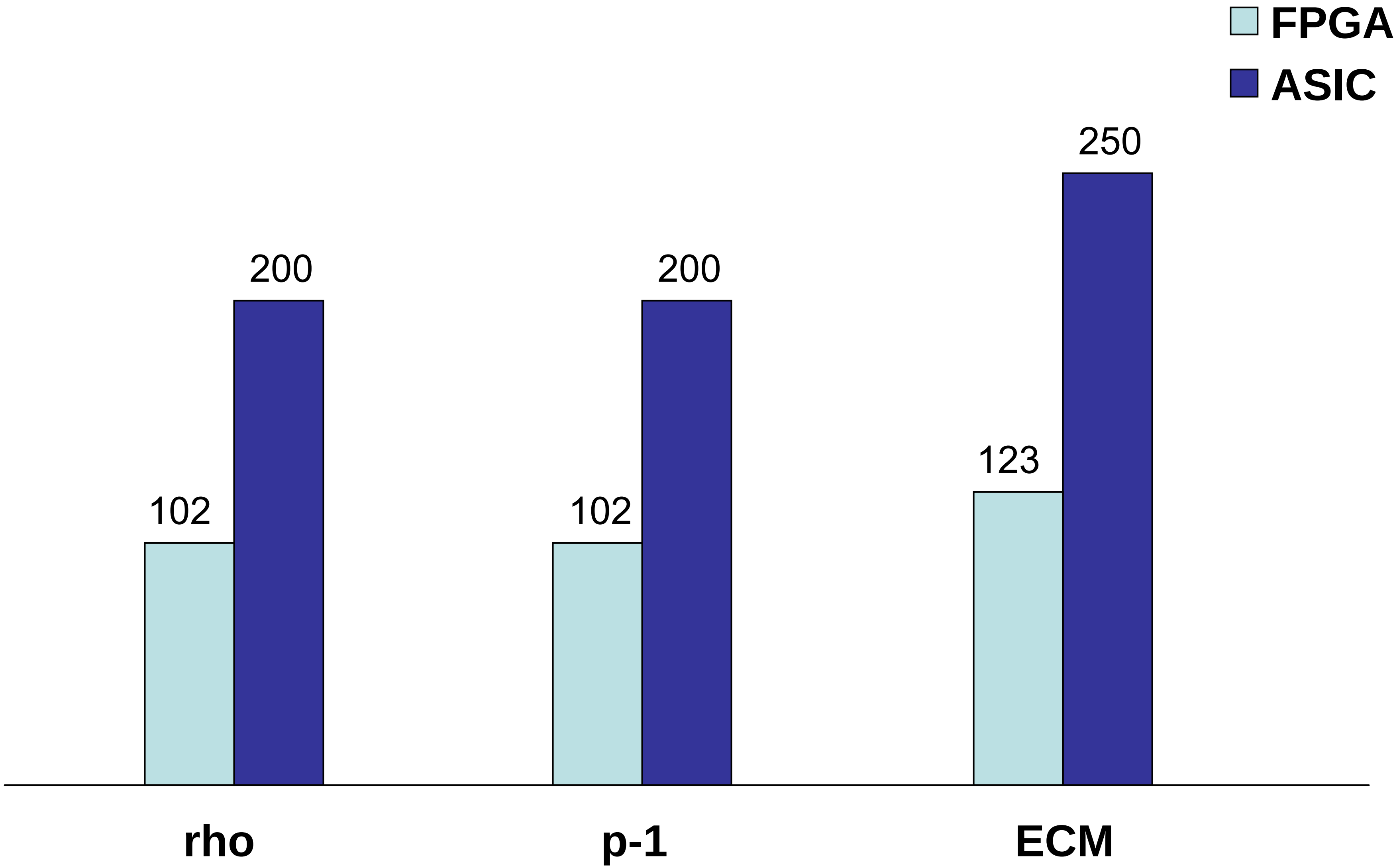
ASIC 130 nm vs. Virtex II 6000 – ECM (13 units)



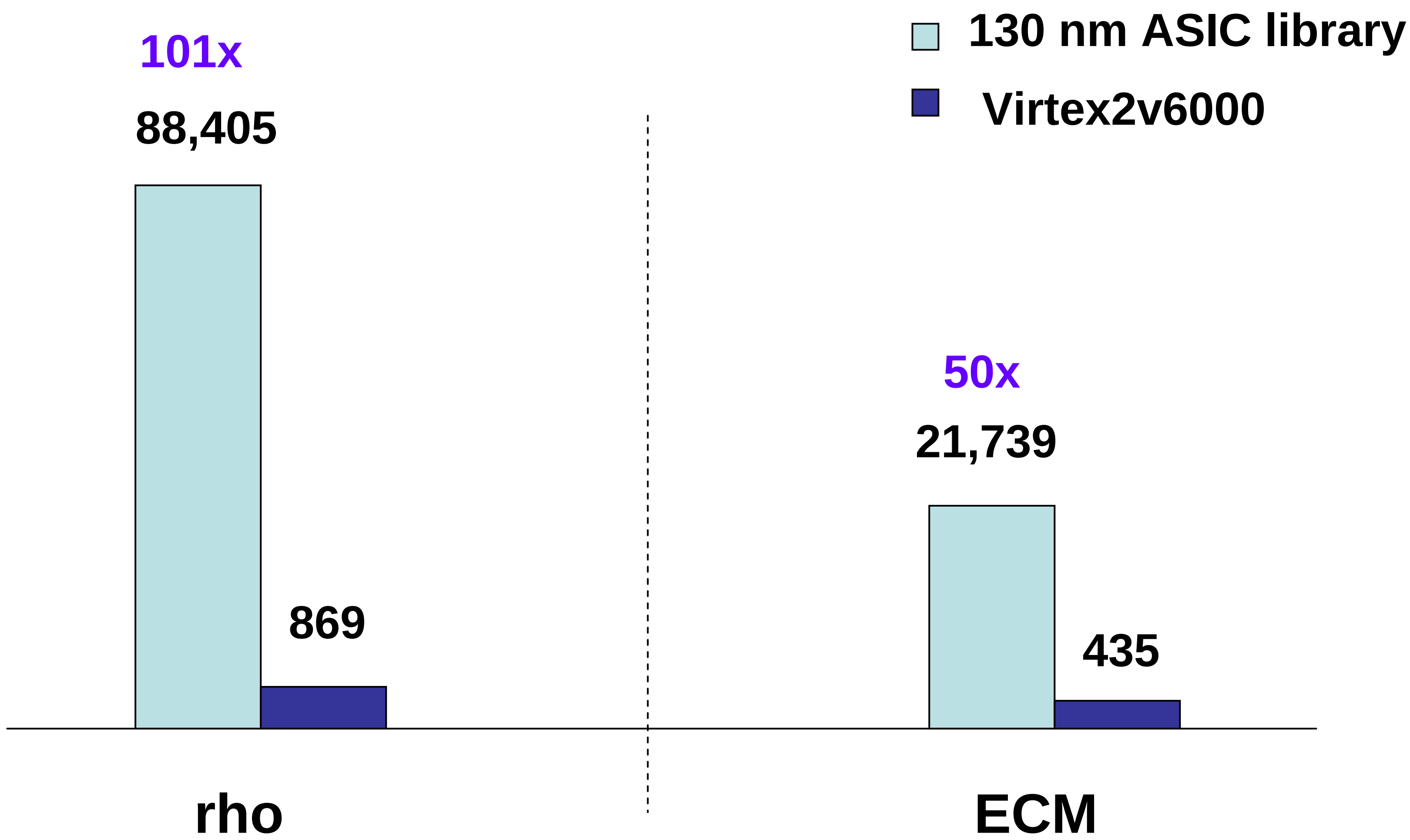
Area of Virtex II 6000
(estimation by R.J. Lim Fong, MS Thesis, VPI, 2004)

Area of an ASIC with equivalent functionality

Clock Frequency in MHz



Number of rho & ECM computations per second using the same chip area



SRC IMPLEMENTATION

SRC 6
reconfigurable computer

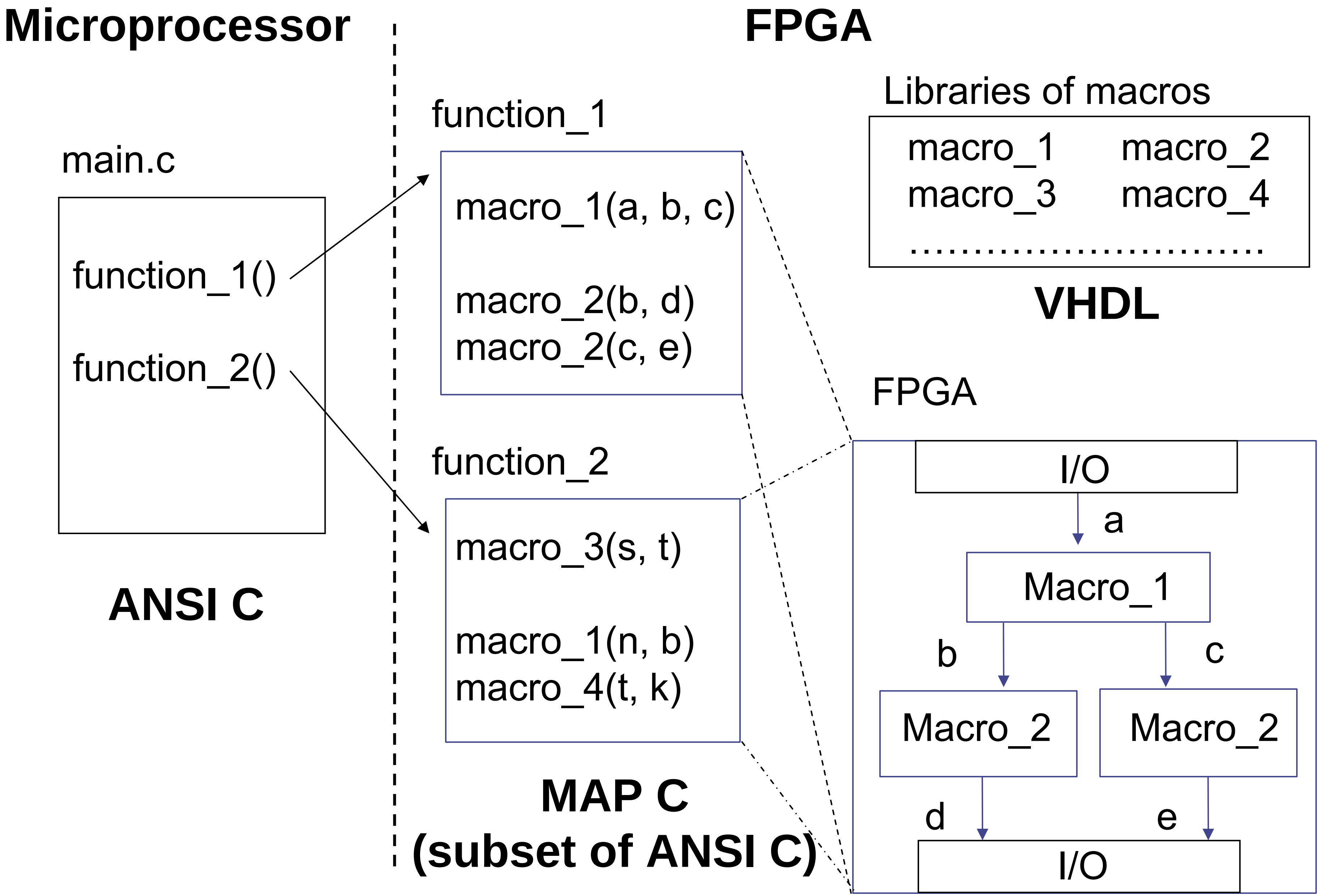
SRC 6 from
SRC Computers

Basic unit:

- 2 x Pentium Xeon, 3 GHz
- 2 x Xilinx Virtex II FPGA
  XC2V6000 running at 100 MHz
- 24 MB of the FPGA-board RAM
- Fast communication interface between the microprocessor board and the FPGA board, 1600 MB/s

Multiple basic units can be connected using Hi-Bar Switch and Global Common Memory

SRC Programming Model



SRC Programming Environment

+ very easy to learn and use
+ standard ANSI C
+ hides implementation details
+ very well integrated environment
+ mature - in production use for over 5 years with constant improvements

- subset of C
- legacy C code requires rewriting
- C limitations in describing HW (paralellism, data types)
- closed environment, limited portability of code to HW platforms other than SRC

SRC Program Partitioning

μP system

C function
for μP

HLL
(High Level
Language)

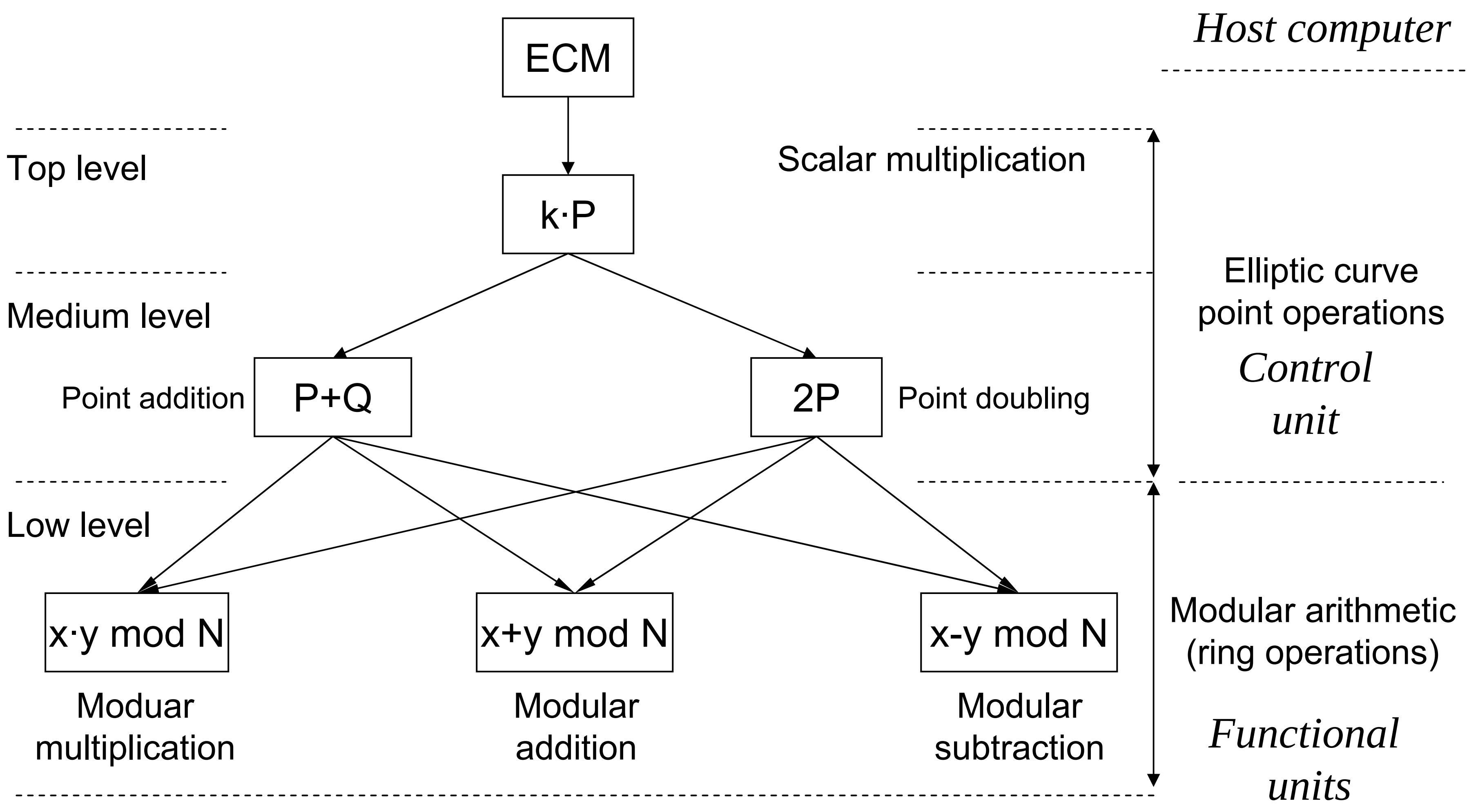
C function
for MAP

FPGA system

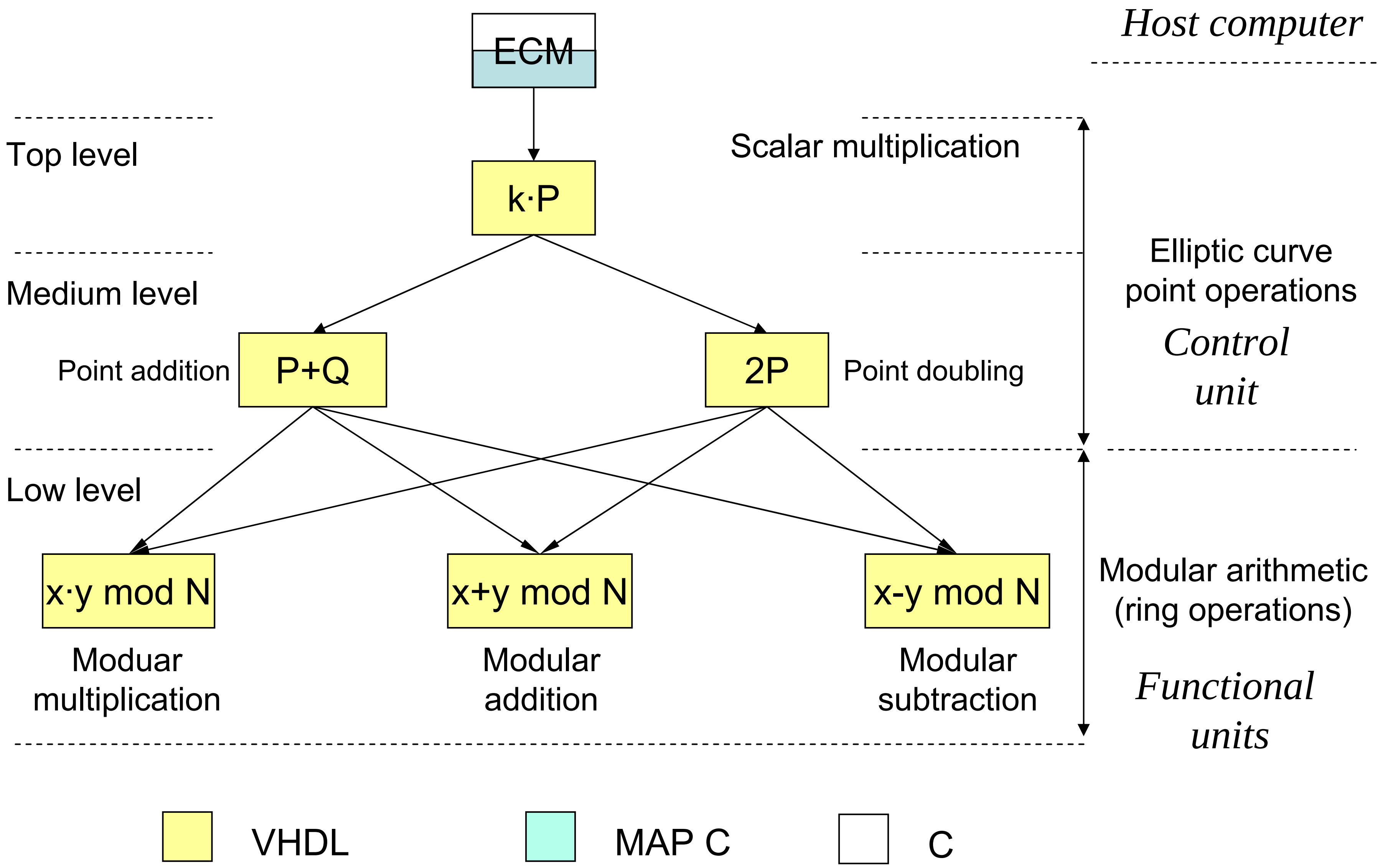
VHDL
macro

HDL
(Hardware
Description
Language)

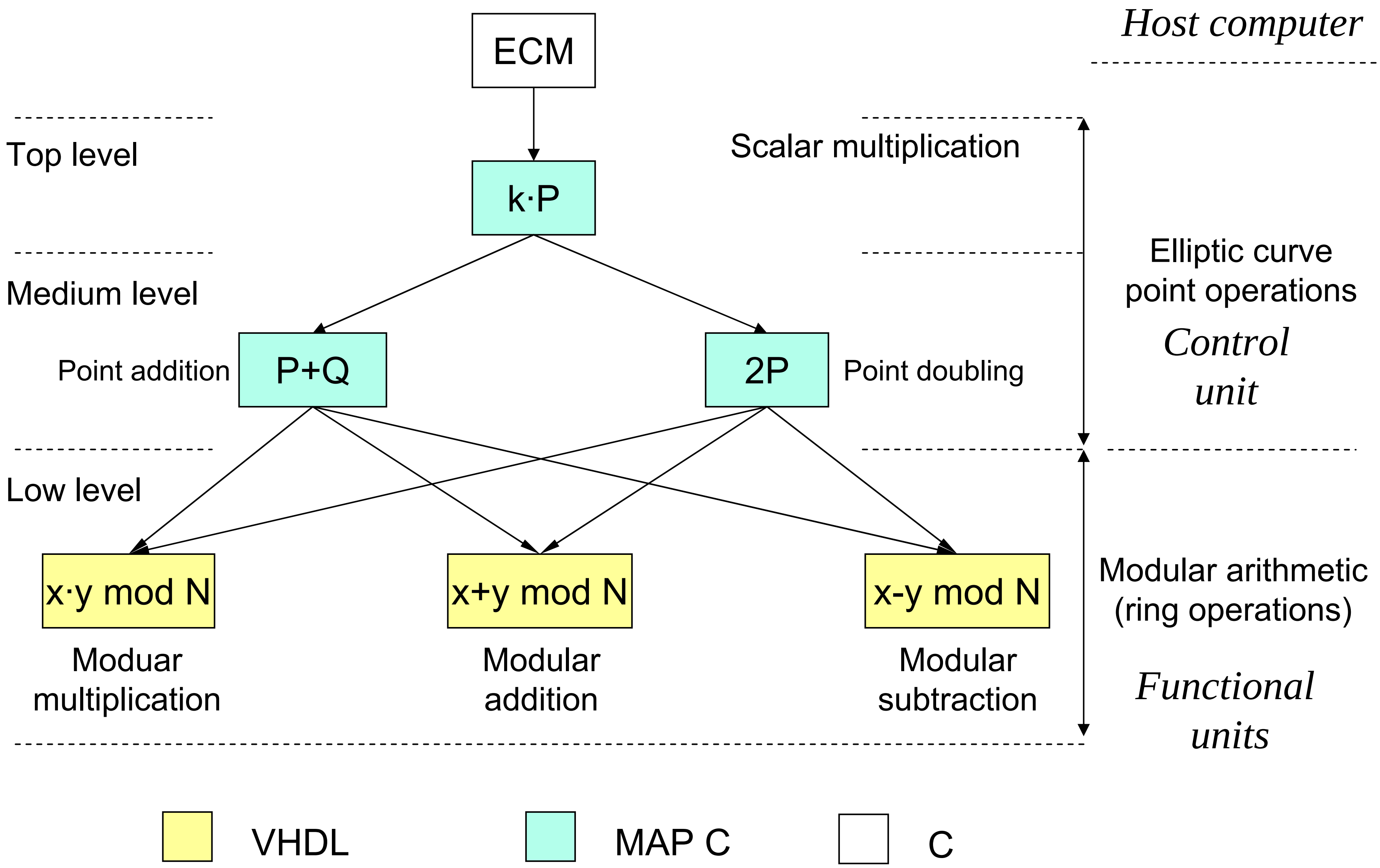
Hierarchy of Elliptic Curve Operations



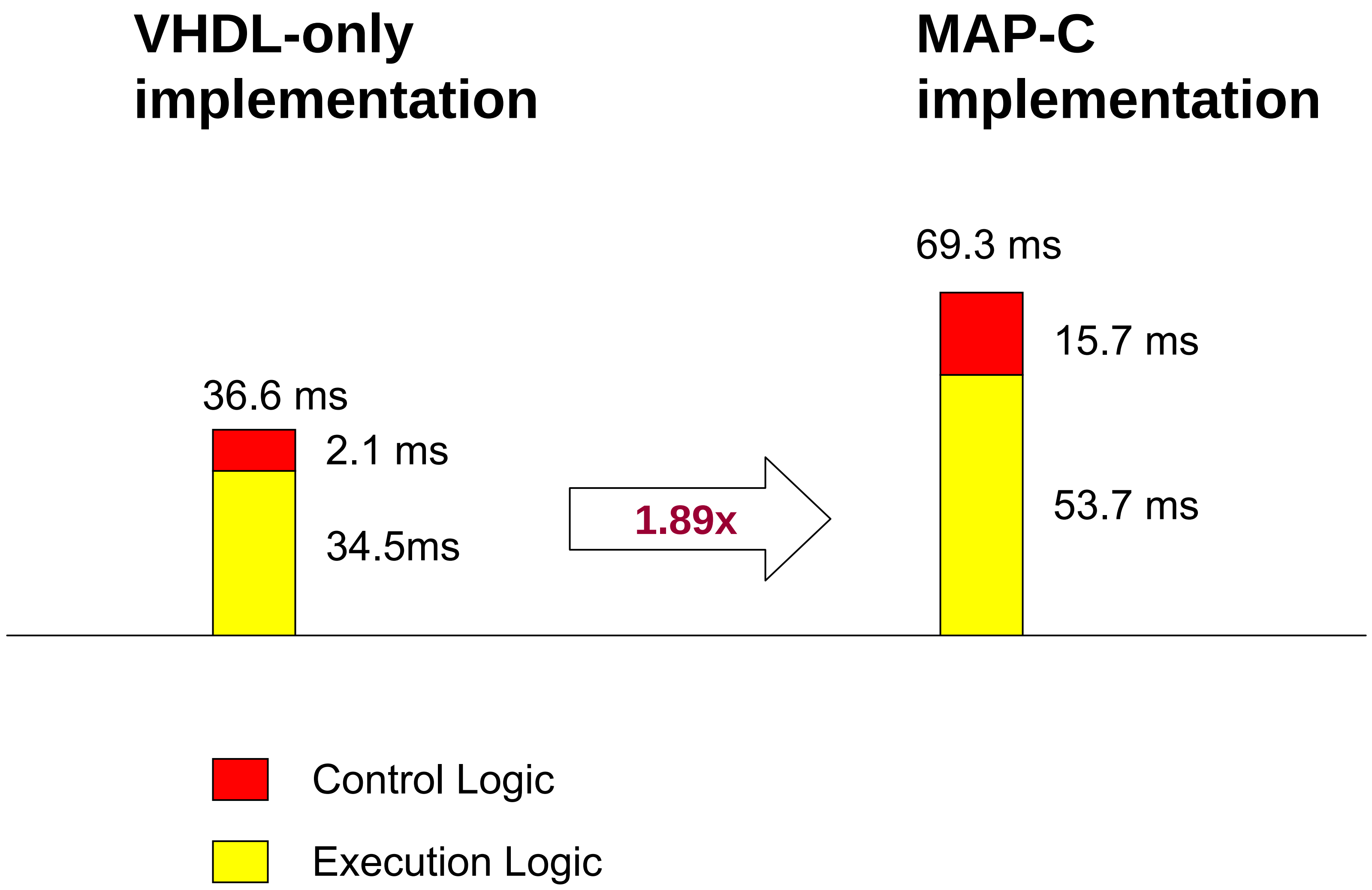
VHDL-only implementation



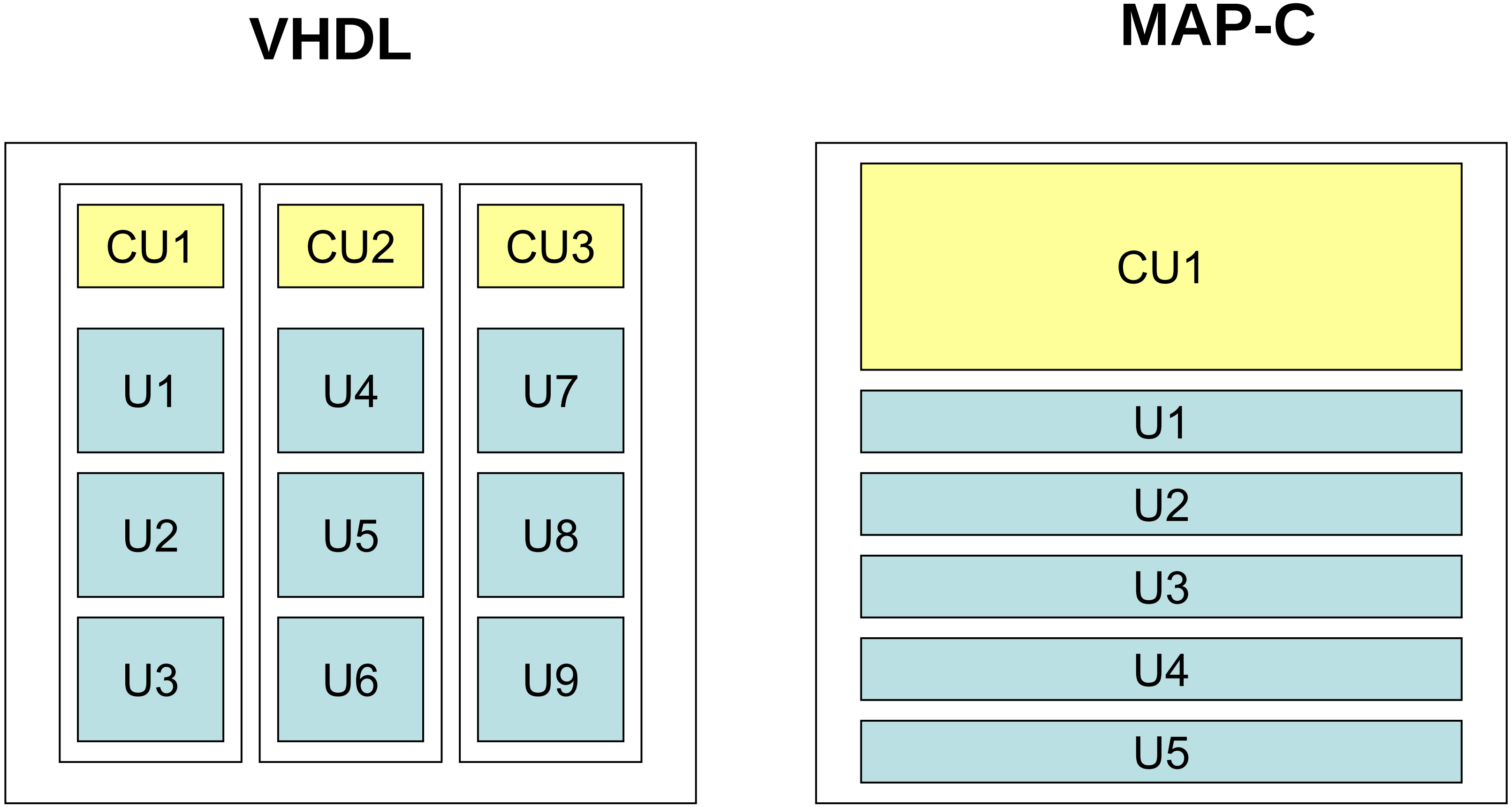
MAP C implementation



MAP C Compiler Performance Penalty



MAP C Compiler Area Penalty



9 Units

5 Units

Conversion of MAP C to Hardware (1)

a = src_0

…………

…………

a = src_1

…………

…………

a = src_2

…………

…………

a = src_3

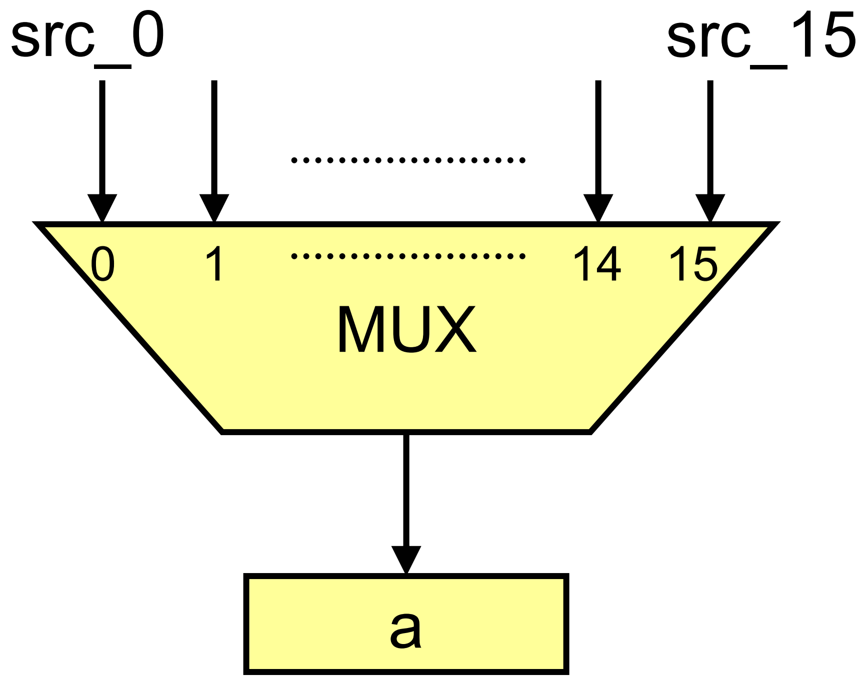
…………

…………

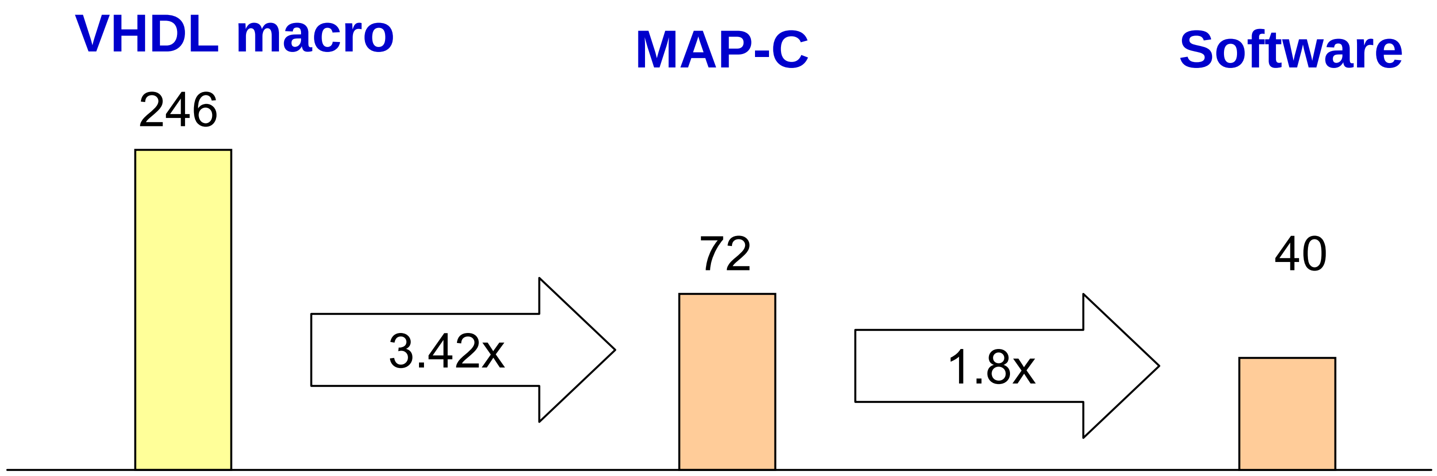
…………

…………

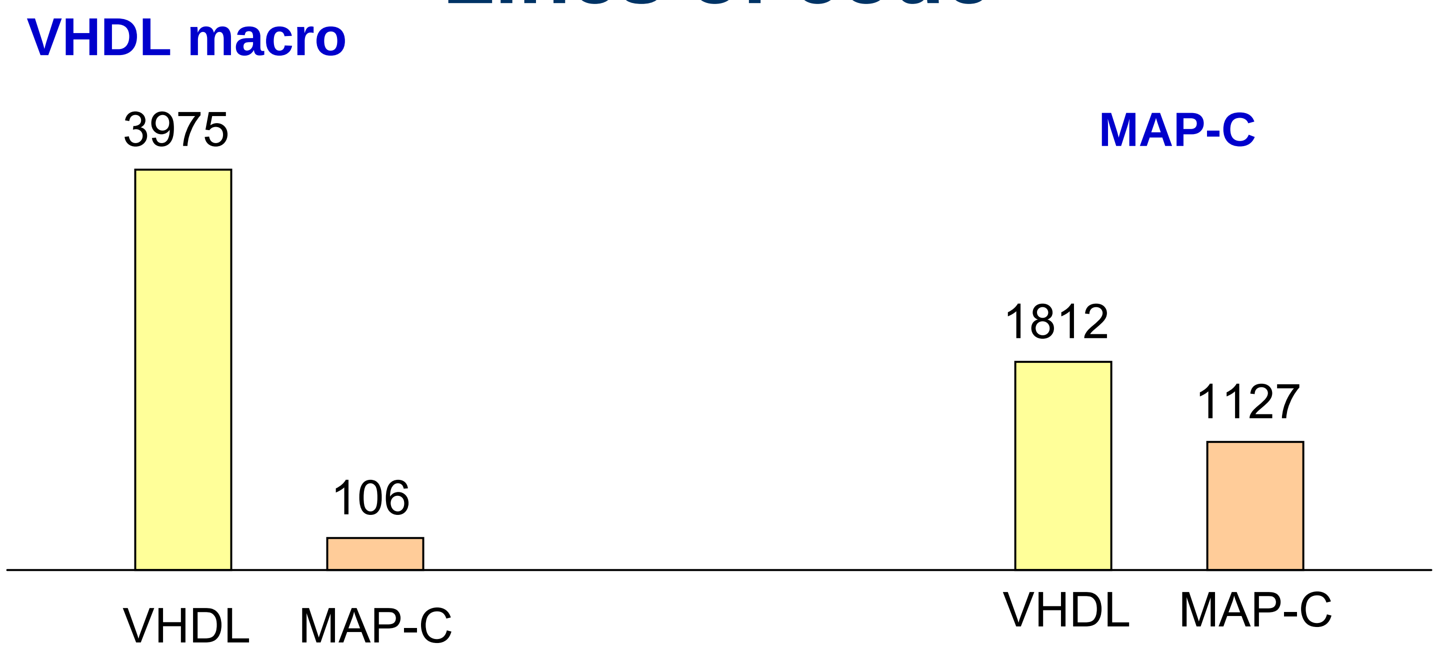
a = src_15



ECM Operations / sec



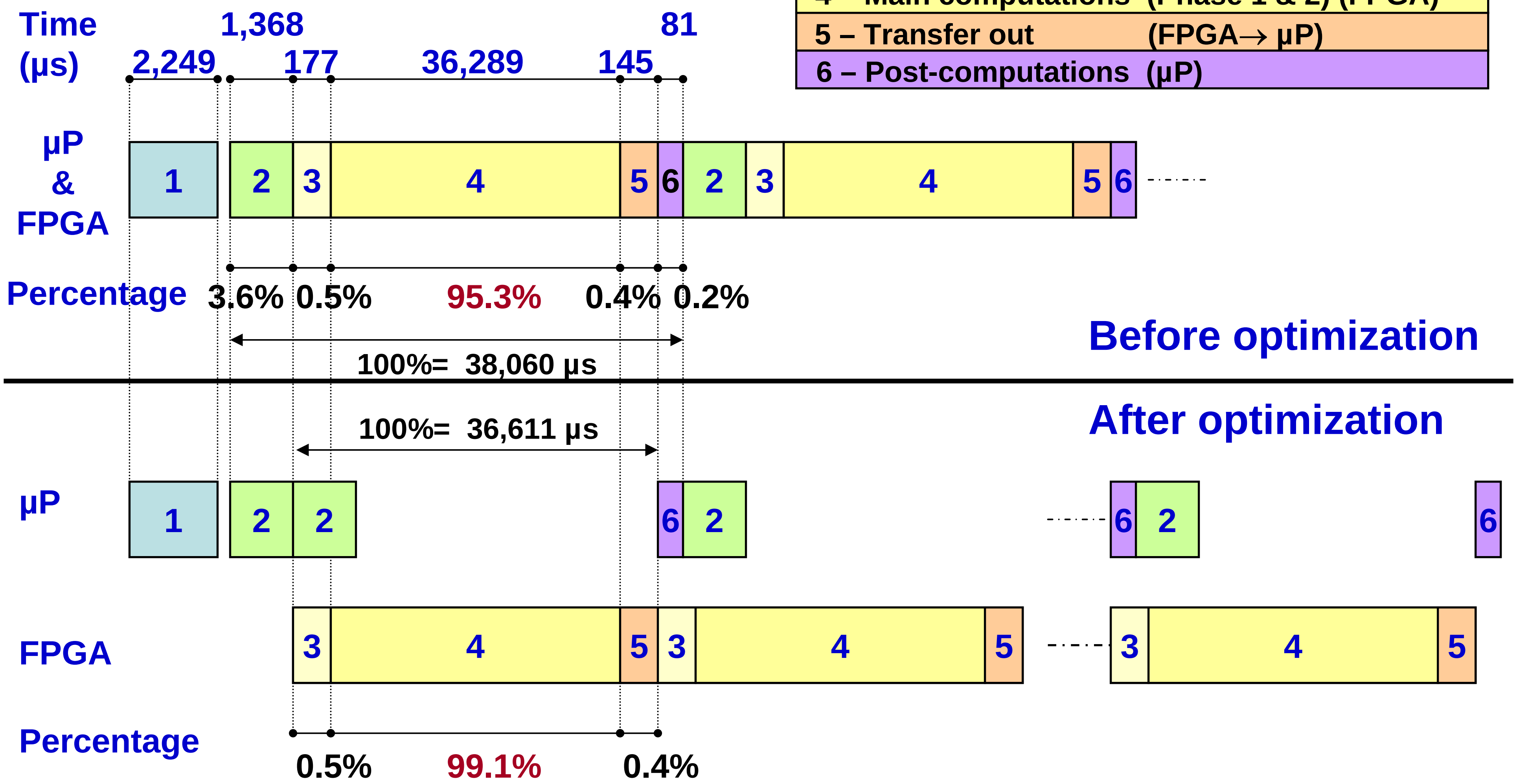
Lines of code



Experimental testing of ECM VHDL-only implementation using SRC 6

Legend:

1 – General pre-computations independent of N	
2 – Pre-computations	(µP)
3 – Transfer in	(µP→FPGA)
4 – Main computations	(Phase 1 & 2) (FPGA)
5 – Transfer out	(FPGA→ µP)
6 – Post-computations	(µP)



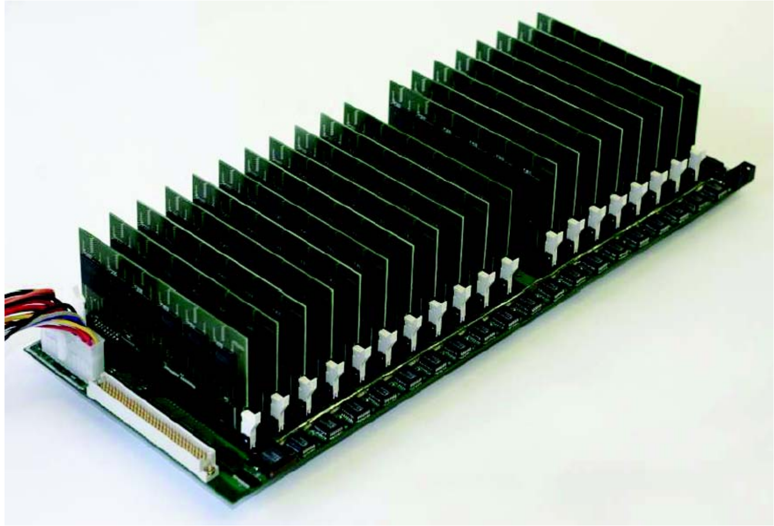
Future Work,
Conclusions
& Open Problems

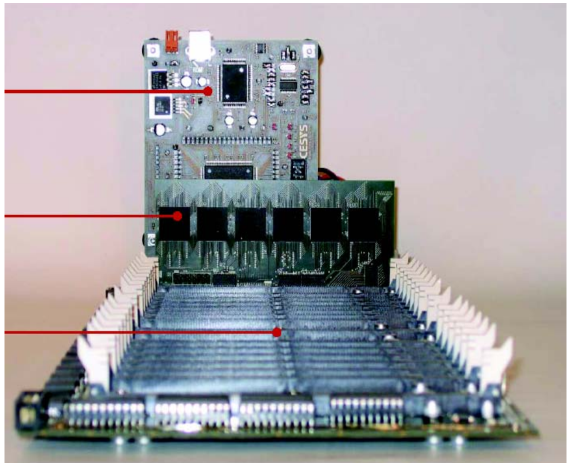
Near term future work

Porting our implementations of

- **rho**
- **p-1**
- **ECM**

to COPACOBANA





120 Spartan 3 FPGAs
Clock frequency 100 MHz

Future work

Integration of a selected software implementation of NFS with our hardware accelerators.

Actual factoring of medium size numbers using integrated software/hardware implementation of NFS.

Prototype hardware implementations of Sieving and Linear Algebra step.

Conclusions

Hardware implementations of ECM, rho and p-1 methods provide a **substantial improvement** vs. optimized software implementations in terms of the **performance to cost ratio**

- low-cost FPGAs vs. microprocessors x 8-10
- ASICs vs. low-cost FPGAs x 50-100

Best environment for prototyping
of hardware implementations of codebreaking machines

- general-purpose reconfigurable computers (e.g., SRC)

Best environment for the final design
of the cost-optimized cipher breaker

- special-purpose machines based on
  - low-cost FPGAs (or ASICs for very high volumes)

Still unsolved mathematical problems…

Optimal choice of parameters for NFS that minimizes total execution time of NFS

by assuring the best possible balance among

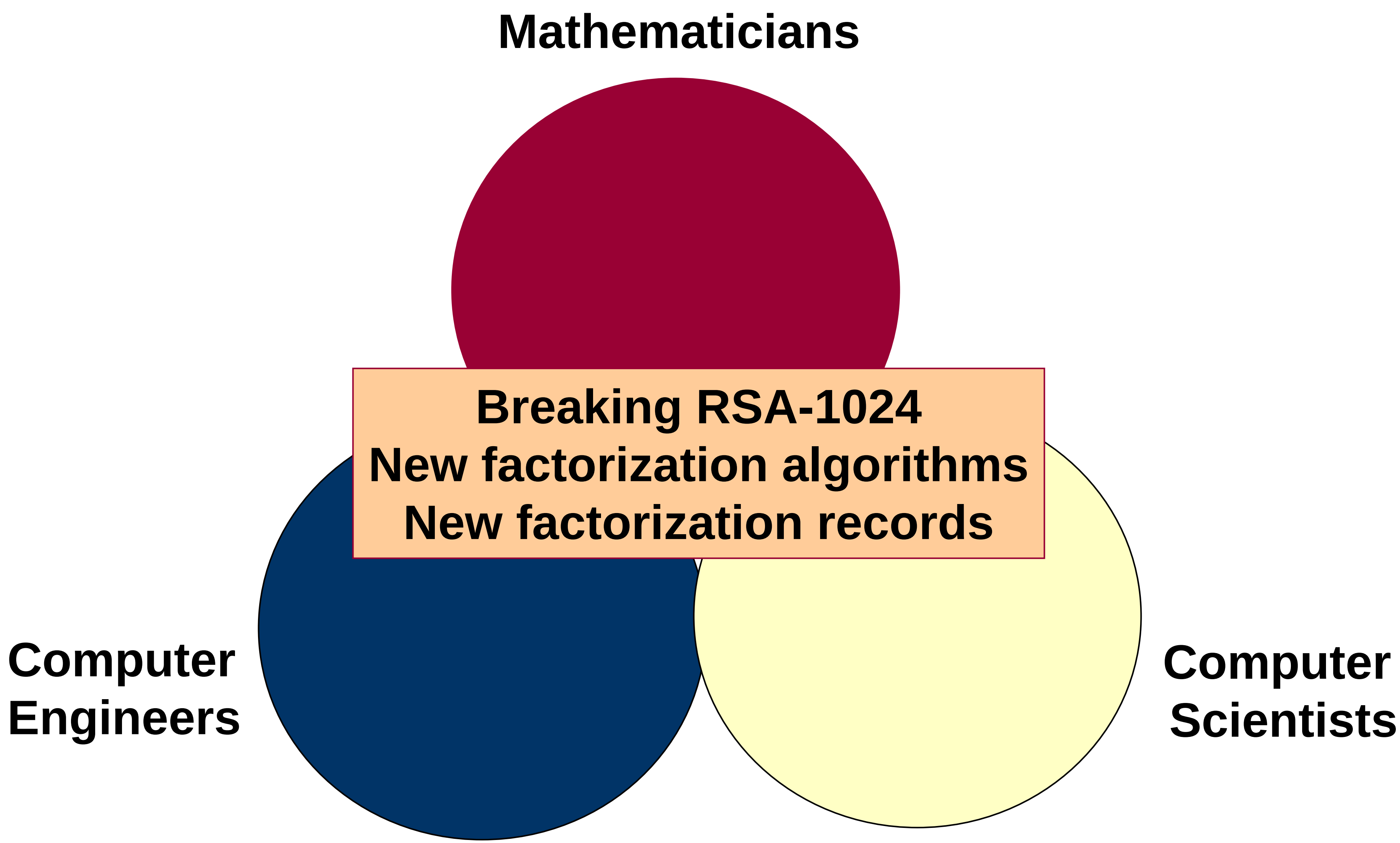
- Rho, p-1, and ECM methods within minifactoring
- Sieving and minifactoring within relation collection step
- Relation collection step and linear algebra steps within NFS

Still unsolved mathematical problems…

New factoring algorithms specifically
targeting hardware

In particular, new factoring algorithms with
complexity better than
General Number Field Sieve

Need for greater synergy



Thank you!

Questions???