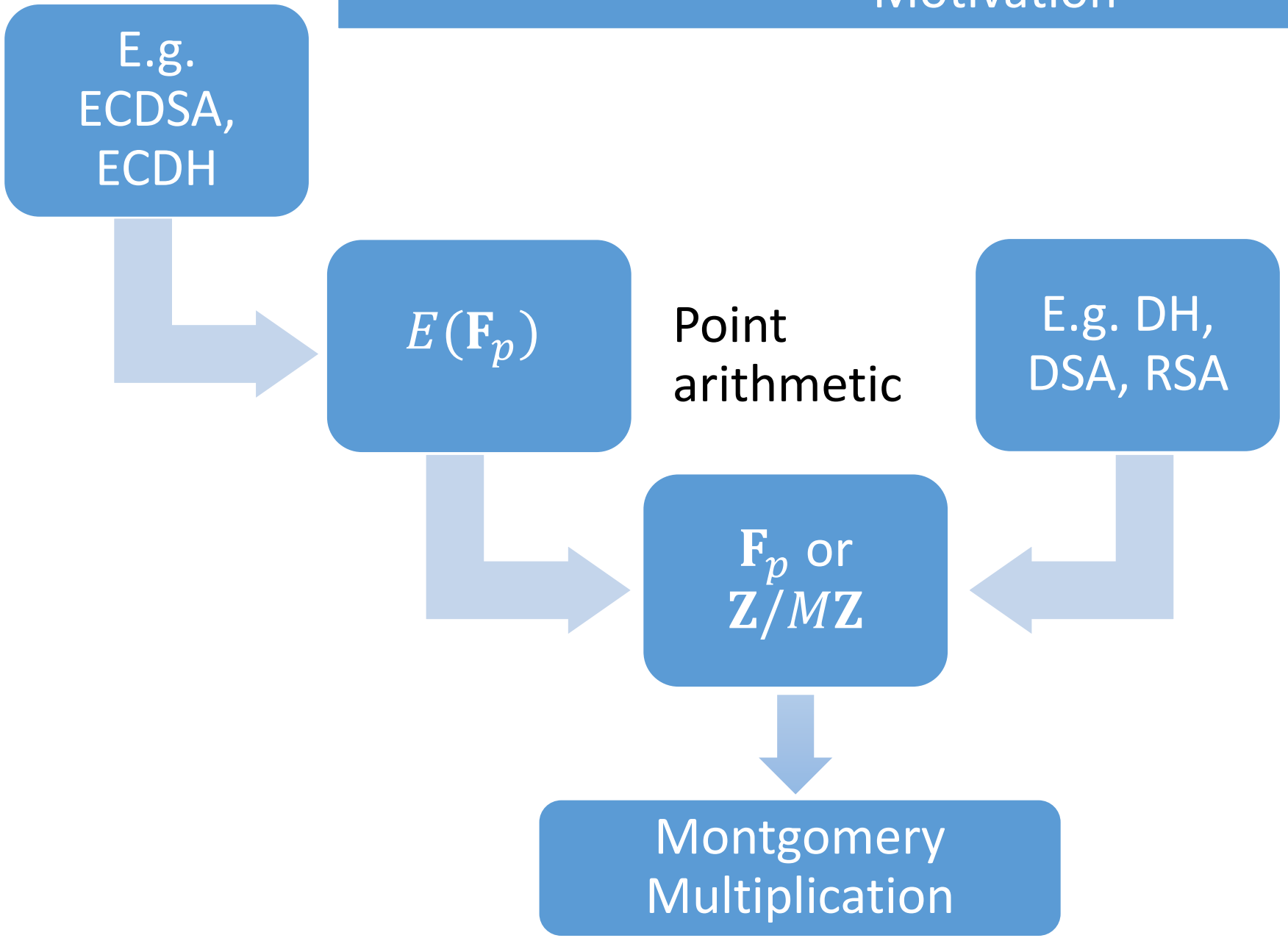
Montgomery Multiplication Using Vector Instructions

Joppe W. Bos, Peter L. Montgomery, Daniel Shumow, and Gregory M. Zaverucha

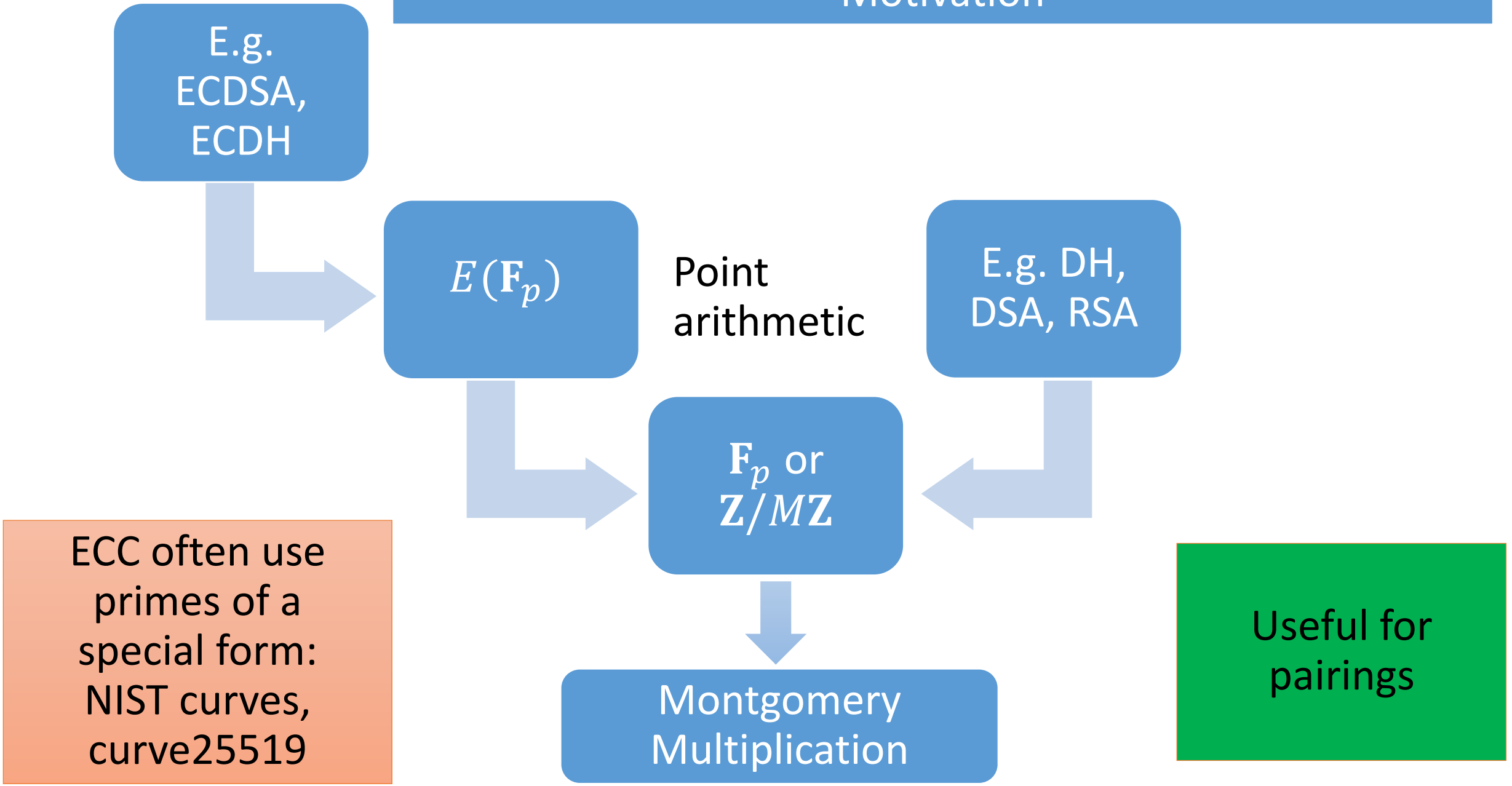
SAC 2013

Microsoft® Research

Motivation


E.g.
ECDSA,
ECDH
$E(\mathbf{F}_p)$
Point
arithmetic
E.g. DH,
DSA, RSA
$\mathbf{F}_p$ or
$\mathbf{Z}/M\mathbf{Z}$
Montgomery
Multiplication

Motivation


E.g. ECDSA, ECDH
$E(\mathbf{F}_p)$
Point arithmetic
E.g. DH, DSA, RSA
$\mathbf{F}_p$ or $\mathbf{Z}/M\mathbf{Z}$
Montgomery Multiplication
ECC often use primes of a special form: NIST curves, curve25519
Useful for pairings

Modular Multiplication

Compute $C = A \times B \pmod{M}$

$R = A \times B$ write $R = q \times M + C$ such that $0 \leq C < M$

Cost: One multiplication + one division with remainder

Modular Multiplication

Compute $C = A \times B \pmod{M}$
$R = A \times B$ write $R = q \times M + C$ such that $0 \leq C < M$
Cost: One multiplication + one division with remainder

Montgomery (Math. Comp. 1985) observed that we can avoid the expensive division when *M* is odd

$$\frac{A}{2} \bmod M = \begin{cases} \frac{A}{2} & \text{if } A \text{ is even} \\ \frac{A+M}{2} & \text{if } A \text{ is odd} \end{cases}$$

$$\mathrm{A} + \mathrm{M} \times \left((\mathrm{A} \times -M^{-1}) \bmod 2^{32}\right) \equiv 0 \pmod{2^{32}},$$

precompute $\mu = -M^{-1} \bmod 2^{32}$

Interleaved Montgomery Multiplication

Input: $A = \sum_{i=0}^{n-1} a_i, B, M, \mu = -M^{-1} \bmod 2^{32}$

Output: $C = AB2^{-32n} \bmod M$

$C = 0$

for $i = 0$ to $n - 1$ do

$C = C + a_i B$	$(1 \times n)$ limbs
$q = \mu C \bmod 2^{32}$	(1×1) limb
$C = (C + qM)/\ 2^{32}$	$(1 \times n)$ limbs

If $C \geq M$ then

$C = C\ - M$

Interleaved Montgomery Multiplication

Input: $A = \sum_{i=0}^{n-1} a_i, B, M, \mu = -M^{-1} \bmod 2^{32}$

Output: $C = AB2^{-32n} \bmod M$

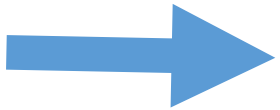
$C = 0$

for $i = 0$ to $n - 1$ do

$C = C + a_i B$	$(1 \times n)$ limbs
$q = \mu C \bmod 2^{32}$	(1×1) limb
$C = (C + qM)/\ 2^{32}$	$(1 \times n)$ limbs

If $C \geq M$ then

$C = C - M$



$q = (c_0 + a_i b_0)\mu \bmod 2^{32}$

$C = (C + a_i B + qM)/\ 2^{32}$

$2 \times (1 \times 1)$ limb

$2 \times (1 \times n)$ limbs

At the cost of *one extra* (1×1) limb multiplication the two $(1 \times n)$ limbs multiplications become *independent*.

Interleaved Montgomery Multiplication

Input: $A = \sum_{i=0}^{n-1} a_i$, B, M, $\mu = -M^{-1} \bmod 2^{32}$

Output: $C = AB2^{-32n} \bmod M$

$C = 0$

for $i = 0$ to $n - 1$ do

$C = C + a_i B$	$(1 \times n)$ limbs
$q = \mu C \bmod 2^{32}$	(1×1) limb
$C = (C + qM)/\ 2^{32}$	$(1 \times n)$ limbs

If $C \geq M$ then

$C = C - M$

Idea

Flip the sign of μ : $\mu = +M^{-1} \bmod 2^{32}$

$2 \times (1 \times 1)$ limb

$$q = (c_0 + a_i b_0)\mu \bmod 2^{32}$$

$$C = (C + a_i B + qM)/\ 2^{32}$$

$2 \times (1 \times n)$ limbs

At the cost of *one extra* (1×1) limb multiplication the two $(1 \times n)$ limbs multiplications become *independent*.

2-way SIMD Interleaved Montgomery Multiplication

Computation 1	**Computation 2**
$d_i = 0$ for $0 \le i < n$	$e_i = 0$ for $0 \le i < n$
for $j = 0$ to $n-1$ **do**	**for** $j = 0$ to $n-1$ **do**
	$q \leftarrow ((\mu \cdot b_0) \cdot a_j + \mu \cdot (d_0 - e_0)) \bmod 2^{32}$
$t_0 \leftarrow a_j \cdot b_0 + d_0$	$t_1 \leftarrow q \cdot m_0 + e_0$ // *Note that* $t_0 \equiv t_1 \pmod{2^{32}}$
$t_0 \leftarrow \left\lfloor \frac{t_0}{2^{32}} \right\rfloor$	$t_1 \leftarrow \left\lfloor \frac{t_1}{2^{32}} \right\rfloor$
for $i = 1$ to $n-1$ **do**	**for** $i = 1$ to $n-1$ **do**
$p_0 \leftarrow a_j \cdot b_i + t_0 + d_i$	$p_1 \leftarrow q \cdot m_i + t_1 + e_i$
$t_0 \leftarrow \left\lfloor \frac{p_0}{2^{32}} \right\rfloor$	$t_1 \leftarrow \left\lfloor \frac{p_1}{2^{32}} \right\rfloor$
$d_{i-1} \leftarrow p_0 \bmod 2^{32}$	$e_{i-1} \leftarrow p_1 \bmod 2^{32}$
$d_{n-1} \leftarrow t_0$	$e_{n-1} \leftarrow t_1$

$$C \leftarrow D - E \quad // \text{ where } D = \sum_{i=0}^{n-1} d_i 2^{32i},\ E = \sum_{i=0}^{n-1} e_i 2^{32i}$$

if $C < 0$ **do** $C \leftarrow C + M$

2-way SIMD Interleaved Montgomery Multiplication

Computation 1	Computation 2
$d_i = 0$ for $0 \le i < n$	$e_i = 0$ for $0 \le i < n$
for $j = 0$ to $n-1$ **do**	**for** $j = 0$ to $n-1$ **do**
	$q \leftarrow ((\mu \cdot b_0) \cdot a_j + \mu \cdot (d_0 - e_0)) \bmod 2^{32}$
$t_0 \leftarrow a_j \cdot b_0 + d_0$	$t_1 \leftarrow q \cdot m_0 + e_0$ // *Note that* $t_0 = t_1 \pmod{2^{32}}$
$t_0 \leftarrow \left\lfloor \frac{t_0}{2^{32}} \right\rfloor$	$t_1 \leftarrow \left\lfloor \frac{t_1}{2^{32}} \right\rfloor$
for $i = 1$ to $n-1$ **do**	**for** $i = 1$ to $n-1$ **do**
$p_0 \leftarrow a_j \cdot b_i + t_0 + d_i$	$p_1 \leftarrow q \cdot m_i + t_1 + e_i$
$t_0 \leftarrow \left\lfloor \frac{p_0}{2^{32}} \right\rfloor$	$t_1 \leftarrow \left\lfloor \frac{p_1}{2^{32}} \right\rfloor$
$d_{i-1} \leftarrow p_0 \bmod 2^{32}$	$e_{i-1} \leftarrow p_1 \bmod 2^{32}$
$d_{n-1} \leftarrow t_0$	$e_{n-1} \leftarrow t_1$

$C \leftarrow D - E$ // *where* $D = \sum_{i=0}^{n-1} d_i 2^{32i}$, $E = \sum_{i=0}^{n-1} e_i 2^{32i}$

if $C < 0$ **do** $C \leftarrow C + M$

Non-SIMD part

$$C = \sum_i d_i 2^{32i} - \sum_i e_i 2^{32i}$$

$$\begin{aligned} q &= \Big((\mu b_0) a_j + \mu(d_0 - e_0)\Big) \bmod 2^{32} \\ &= \Big((\mu b_0) a_j + \mu c_0\Big) \bmod 2^{32} \\ &= (c_0 + a_j b_0)\mu \bmod 2^{32} \end{aligned}$$

Expected Performance Speedup

Sequential Montgomery Multiplication

Long Muls: $2n^2$ Short Muls: n

2-way SIMD Montgomery Multiplication

Long Muls: n^2 Short Muls: $2n$

Instruction	Classical		2-way SIMD
	32-bit	64-bit	32-bit
add			n
sub			n
short mul	n	n/2	2n
muladd	2n	n	
muladdadd	2n(n-1)	n(n/2-1)	
SIMD muladd			n
SIMD muladdadd			n(n-1)

Expected Performance Speedup

Sequential Montgomery Multiplication

Long Muls: $2n^2$ Short Muls: n

2-way SIMD Montgomery Multiplication

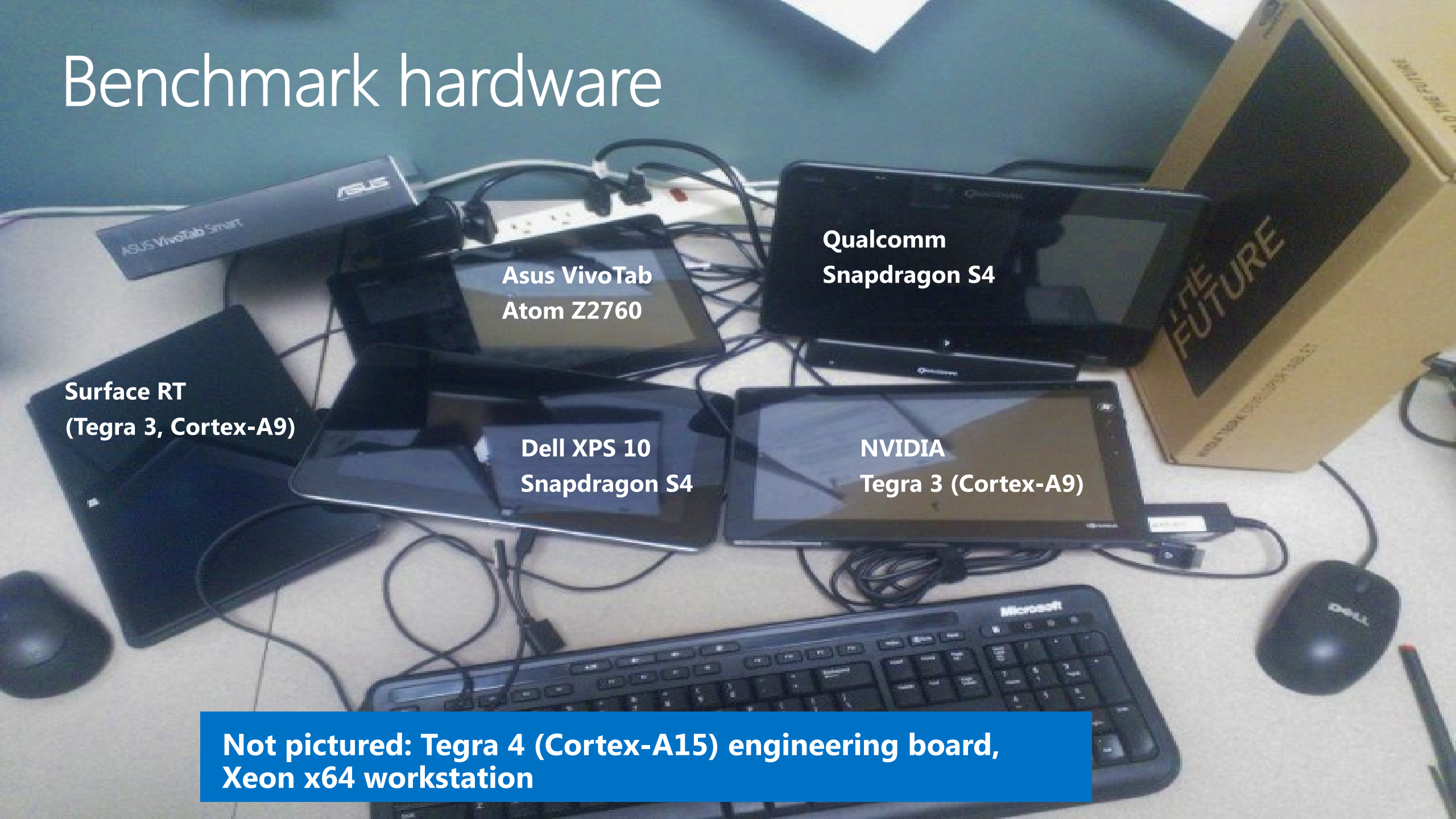
Long Muls: n^2 Short Muls: $2n$

Instruction	Classical		2-way SIMD
	32-bit	64-bit	32-bit
add			n
sub			n
short mul	n	n/2	2n
muladd	2n	n	
muladdadd	2n(n-1)	n(n/2-1)	
SIMD muladd			n
SIMD muladdadd			n(n-1)

Based on #multiplications only we expect:

- 32-bit 2-way SIMD to be at most 2x as fast as 32-bit sequential
- 32-bit 2-way SIMD to be approximately 2x as slow as 64-bit sequential

Benchmark hardware



Performance Results – x86

	Intel Xeon E31230 (3.2 GHz) - PC			Intel Atom Z2760 (1.8 GHz) - Tablet		
RSA	**Classic**	**SIMD**	**Ratio**	**Classic**	**SIMD**	**Ratio**
enc 2048	181,412	414,787	0.44	2,583,643	1,601,878	1.61
dec 2048	4,928,633	12,211,700	0.40	80,204,317	52,000,367	1.54

Performance Results - ARM

	Dell XPS 10 tablet (1.8 GHz) Snapdragon S4			NVIDIA Tegra 4 (1.9 GHz) (dev board, Cortex-A15)			NVIDIA Tegra 3 T30 (1.4 GHz) (dev board, Cortex-A9)		
RSA	**Classic**	**SIMD**	**Ratio**	**Classic**	**SIMD**	**Ratio**	**Classic**	**SIMD**	**Ratio**
enc 2048	1,087,318	710,910	1.53	725,336	712,542	1.02	872,468	1,358,955	0.64
dec 2048	34,769,147	21,478,047	1.62	23,177,617	22,812,040	1.02	27,547,434	47,205,919	0.58

Performance Results

Compare to results from:
eBACS: ECRYPT Benchmarking of Cryptographic Systems and OpenSSL

	Snapdragon S4 (1.8 GHz) vs Snapdragon S3 (1.78 GHz)		Intel Atom Z2760 (1.8 GHz) - Tablet	
RSA	**Classic**	**OpenSSL**	**Classic**	**OpenSSL**
enc 2048	1,087,318	609,593	2,583,643	2,323,800
dec 2048	34,769,147	39,746,105	80,204,317	75,871,800

Can we do (asymptotically) better?

What about faster multiplication methods (Karatsuba)?

- Incompatible with interleaved Montgomery multiplication
- Possible gain ([A]) on 32-bit platform for 1024-bit Montgomery multiplication

Following the analysis from [A] (one level Karatsuba) for 32-bit platforms

Sequential Karatsuba montmul
versus
Sequential interleaved montmul

→ Sequential Karatsuba reduces muls by 1.14x
Sequential Karatsuba reduces adds by 1.18x

Sequential Karatsuba montmul
versus
SIMD interleaved montmul

→ SIMD interleaved reduces muls by 1.70x
SIMD interleaved reduces adds by 1.67x

[A] J. Großschädl, R. M. Avanzi, E. Savas, and S. Tillich. Energy-efficient software implementation of long integer modular arithmetic. CHES 2005

Can we do (asymptotically) better?

What about SIMD Karatsuba montmul versus SIMD interleaved montmul?

- SIMD Karatsuba, but how to calculate SIMD reduction?
- This approach is used in GMP
- GMP is not a crypto lib

	GMP	SIMD	GMP	SIMD
	RSA-2048 enc	RSA-2048 enc	RSA-2048 dec	RSA-2048 dec
Atom Z2760	2,184,436	1,601,878	37,070,875	52,000,367
Intel Xeon E3-1230 (32-bit mode)	695,861	414,787	11,929,868	12,211,700

Can we do (asymptotically) better?

What about SIMD Karatsuba montmul versus SIMD interleaved montmul?

- SIMD Karatsuba, but how to calculate SIMD reduction?
- This approach is used in GMP
- GMP is not a crypto lib

	GMP	SIMD	GMP	SIMD
	RSA-2048 enc	RSA-2048 enc	RSA-2048 dec	RSA-2048 dec
Atom Z2760	2,184,436	1,601,878	37,070,875	52,000,367
Intel Xeon E3-1230 (32-bit mode)	695,861	414,787	11,929,868	12,211,700

Modular Squaring

- Time(Montgomery squaring) $\approx 0.80 \times$ Time(Montgomery Multiplication) [A]
- SIMD Montgomery squaring?
- We didn't use this optimization

[A] J. Großschädl, R. M. Avanzi, E. Savas, and S. Tillich. Energy-efficient software implementation of long integer modular arithmetic. CHES 2005

Conclusions

- ✓ Current vector instructions can be used to enhance the performance of Montgomery multiplication on modern embedded devices
  Examples: 32-bit x86 (SSE) and ARM (NEON) platforms
- ✓ Faster RSA-2048 on **some** tablets: performance on ARM differs significantly
- ✓ If future instruction set(s) support $64 \times 64 \rightarrow 128$-bit 2-way SIMD multipliers: enhance interleaved Montgomery multiplication performance

Future work

- ❖ Investigate SIMD Karatsuba + SIMD (?) Montgomery reduction
- ❖ Investigate SIMD Montgomery squaring