Single Base Modular Multiplication for Efficient Hardware RNS Implementations of ECC

Karim Bigou and Arnaud Tisserand

CNRS, IRISA, INRIA Centre Rennes - Bretagne Atlantique and Univ. Rennes 1

CHES 2015, Sept. 13 – 16

Context

Design efficient hardware implementations of asymmetric cryptosystems using fast arithmetic techniques:

- RSA [RSA78]
- Discrete Logarithm Cryptosystems: Diffie-Hellman [DH76] (DH), ElGamal [Elg85]
- Elliptic Curve Cryptography (ECC) [Mil85] [Kob87]

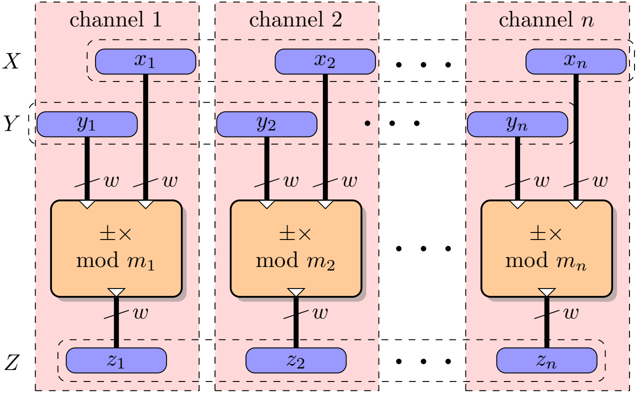
The **residue number system** (RNS) is a representation which enables fast computations for cryptosystems requiring large integers or $\mathbb{F}_P$ elements

Residue Number System (RNS) [SV55] [Gar59]

X a large integer of ℓ bits ($\ell \approx 160$–4096) is represented by:

$\overrightarrow{X} = (x_1, \ldots, x_n) = (X \bmod m_1, \ldots, X \bmod m_n)$

RNS base $\mathcal{B} = (m_1, \ldots, m_n)$, n pairwise co-primes of w bits, $n \times w \geqslant \ell$



RNS relies on the Chinese remainder theorem (CRT)

EMM $=$ w-bit elementary modular multiplication in one channel

RNS Properties

Pros:

- **Carry free** between channels
  - each channel is independant
- **Fast parallel** $+, -, \times$ and some exact divisions
  - computations over all channels can be performed in parallel
  - an RNS multiplication requires n `EMM`s
- **Flexibility** for hardware implementations
  - the number of hardware channels and logical channels can be different
  - various area/time trade-offs and multi-size support
- **Non-positional** number system
  - randomization of internal computations (SCA countermeasures)

Cons:

- **Non-positional** number system
  - comparison, modular reduction and division are much harder
  - **modular reduction** : RNS version of Montgomery reduction `MR`

Montgomery and Pseudo-Mersenne Reductions in RNS

Classical binary positional representation:

- in practice, standards use special primes to perform faster reduction: the pseudo-Mersenne primes
- $P = 2^\ell - c$ where $c < 2^{\ell/2}$ has a small Hamming weight: fast reduction using $2^\ell \equiv c \bmod P$

In RNS, no equivalent to pseudo-Mersenne number in state-of-the-art

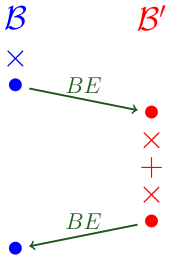
Approaches in RNS literature to speed up modular arithmetic:

- reduce the number of `MR` (*e.g.* [BDE13, BT13]):
  - for instance computing pattern of the form $AB + CD \bmod P$
- improves `MR` in specific context (*e.g.* [Gui10, GLP+12, BT14]):
  - for example RSA or ECC
- choose carefully some parameters of the representation to reduce the internal computation cost of `MR`s [BKP09, BM14, YFCV14]

RNS Montgomery Reduction ($\mathtt{MR}$) [PP95]

Input: $\overrightarrow{X}$, $\overrightarrow{X}'$ with $X < \alpha P^2 < PM$ and $2P < M'$
Output: $(\overrightarrow{\omega}, \overrightarrow{\omega}')$ with $\omega \equiv X \times M^{-1} \bmod P$
$0 \leqslant \omega < 2P$

$\overrightarrow{Q} \longleftarrow \overrightarrow{X} \times (-\overrightarrow{P}^{-1})$ (in base $\mathcal{B}$)
$\overrightarrow{Q}' \longleftarrow \mathtt{BE}(\overrightarrow{Q}, \mathcal{B}, \mathcal{B}')$ ($n \times n$ $\mathtt{EMMs}$)
$\overrightarrow{S}' \longleftarrow \overrightarrow{X}' + \overrightarrow{Q}' \times \overrightarrow{P}'$ (in base $\mathcal{B}'$)
$\overrightarrow{\omega}' \longleftarrow \overrightarrow{S}' \times \overrightarrow{M}^{-1}$ (in base $\mathcal{B}'$)
$\overrightarrow{\omega} \longleftarrow \mathtt{BE}(\overrightarrow{\omega}', \mathcal{B}', \mathcal{B})$ ($n \times n$ $\mathtt{EMMs}$)



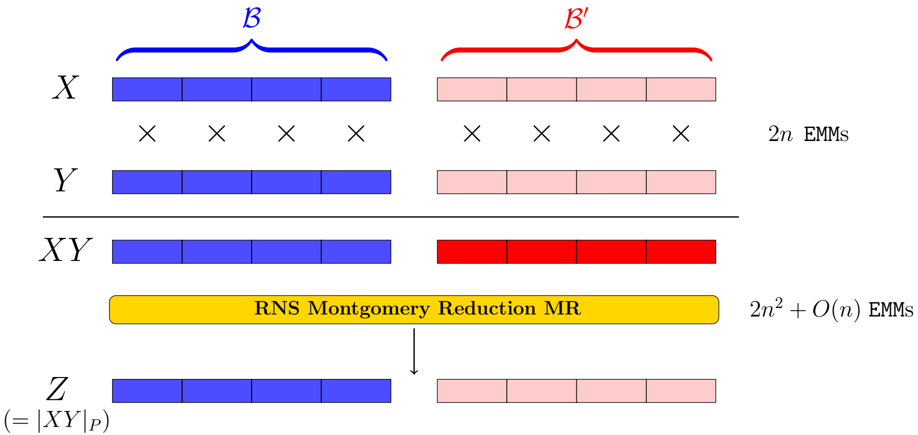
where $M = \prod_{i=1}^{n} m_i$

$\mathtt{BE}$: base extension (*i.e.* conversion)

$\mathtt{MR}$ cost: $2\,n^2 + O(n)$ $\mathtt{EMMs}$

Note: $\mathtt{MM}$ = 1 RNS mult. + $\mathtt{MR}$

Size of Elements Using MM



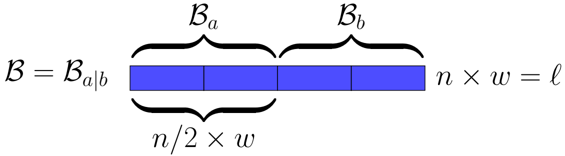
A New RNS Modular Multiplication

First Step: Changing the Representation

We split field elements in 2 parts of the same size

How?

- using **half-bases** :



Using $M_a = \prod_{i=1}^{n_a} m_{a,i}$, we split $\overrightarrow{X}$ into $(\overrightarrow{K_x}, \overrightarrow{R_x})$ such that:

$$\overrightarrow{X} = \overrightarrow{K_x}\,\overrightarrow{M_a} + \overrightarrow{R_x}$$

K_x and R_x are $\ell/2$ bits long

$\mathbb{F}_P$ elements are now represented by (K, R) :
we add a little positional information

We call `Split` the function to get $(\overrightarrow{K_x}, \overrightarrow{R_x})$ from $\overrightarrow{X}$

Decomposition with Split Algorithm

Input: $\overrightarrow{X_{a|b}}$

Precomp.: $\overrightarrow{(M_a^{-1})_b}$

Output: $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$ with $\overrightarrow{X_{a|b}} = \overrightarrow{(K_x)_{a|b}} \times \overrightarrow{(M_a)_{a|b}} + \overrightarrow{(R_x)_{a|b}}$

$\overrightarrow{(R_x)_b} \leftarrow \boxed{\texttt{BE}}\left(\overrightarrow{(R_x)_a}, \mathcal{B}_a, \mathcal{B}_b\right)$ $\quad (\frac{n}{2} \times \frac{n}{2})$ EMMs

$\overrightarrow{(K_x)_b} \leftarrow \left(\overrightarrow{X_b} - \overrightarrow{(R_x)_b}\right) \times \overrightarrow{(M_a^{-1})_b}$

if $\overrightarrow{(K_x)_b} = \overrightarrow{-1}$ **then**

$\quad \overrightarrow{(K_x)_b} \leftarrow \overrightarrow{0}$ $\quad$ */*with Kawamura BE correction [KKSS00] */*

$\quad \overrightarrow{(R_x)_b} \leftarrow \overrightarrow{(R_x)_b} - \overrightarrow{(M_a)_b}$

$\overrightarrow{(K_x)_a} \leftarrow \boxed{\texttt{BE}}\left(\overrightarrow{(K_x)_b}, \mathcal{B}_b, \mathcal{B}_a\right)$ $\quad (\frac{n}{2} \times \frac{n}{2})$ EMMs

return $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$

Note: the cost of Split is dominated by the 2 BEs on half bases :

$$\frac{n^2}{2} + O(n) \text{ when } n_a = n_b = n/2$$

A New Choice for P

Second step: we propose the form $P = M_a^2 - c$ with P prime and c small

Some remarks

- $P = M_a^2 - 1$ is never prime
- in practice, we choose $P = M_a^2 - 2$ with M_a odd *i.e.* $M_a^2 \equiv 2 \bmod P$
- One can find a lot of P for a given size (probabilistic primality tests using `isprime` from Maple, for instance generating 10 000 P of 512 bits in 15 s)
- P is an equivalent for RNS to pseudo-Mersenne numbers for the radix 2 standard representation (for instance $P = 2^{521} - 1$)

Our Single Base Modular Multiplication `SBMM` combines:

- $P = M_a^2 - 2$
- (K_x, R_x) representation
- `Split` function

SBMM Algorithm

Parameters: $\mathcal{B}_a$ such that $M_a^2 = P + 2$ and $\mathcal{B}_b$ such that $M_b > 6M_a$

Input: $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$, $\overrightarrow{(K_y)_{a|b}}$, $\overrightarrow{(R_y)_{a|b}}$ with K_x, R_x, K_y, $R_y < M_a$

Output: $\overrightarrow{(K_z)_{a|b}}$, $\overrightarrow{(R_z)_{a|b}}$ with $K_z < 5M_a$ and $R_z < 6M_a$

$$\overrightarrow{U_{a|b}} \leftarrow \overrightarrow{2K_xK_y + R_xR_y}$$

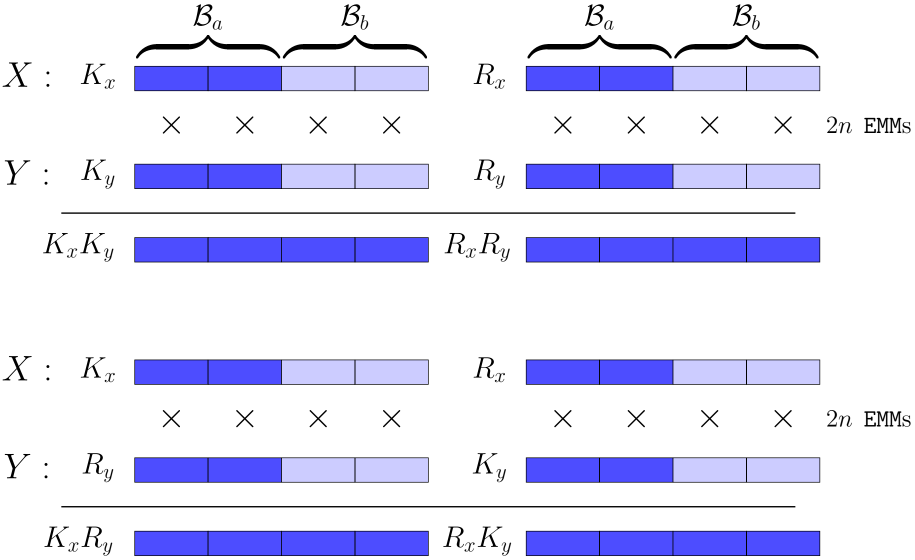
$$\overrightarrow{V_{a|b}} \leftarrow \overrightarrow{K_xR_y + R_xK_y}$$

$$\left.\begin{array}{l} \left(\overrightarrow{(K_u)_{a|b}}, \overrightarrow{(R_u)_{a|b}}\right) \leftarrow \texttt{Split}(\overrightarrow{U_{a|b}}) \\ \left(\overrightarrow{(K_v)_{a|b}}, \overrightarrow{(R_v)_{a|b}}\right) \leftarrow \texttt{Split}(\overrightarrow{V_{a|b}}) \end{array}\right\} \text{ in parallel}$$

$$\left(\overrightarrow{(K_z)_{a|b}}, \overrightarrow{(R_z)_{a|b}}\right) \leftarrow \left(\overrightarrow{(K_u + R_v)_{a|b}}, \overrightarrow{(2 \cdot K_v + R_u)_{a|b}}\right)$$

return $\left(\overrightarrow{(K_z)_{a|b}}, \overrightarrow{(R_z)_{a|b}}\right)$

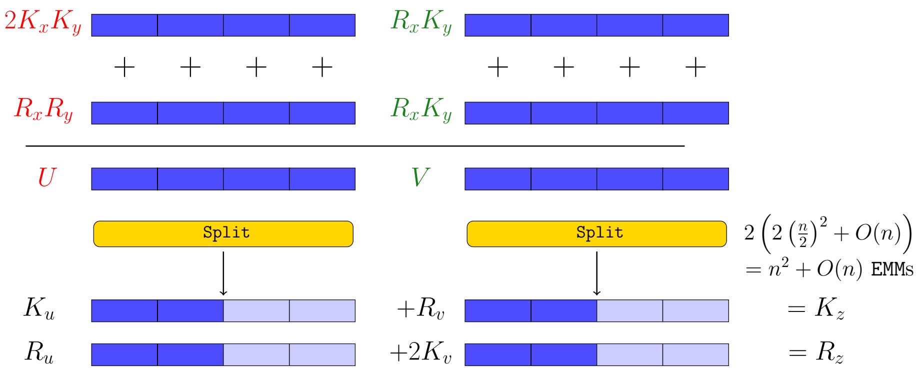
SBMM Principle 1/2



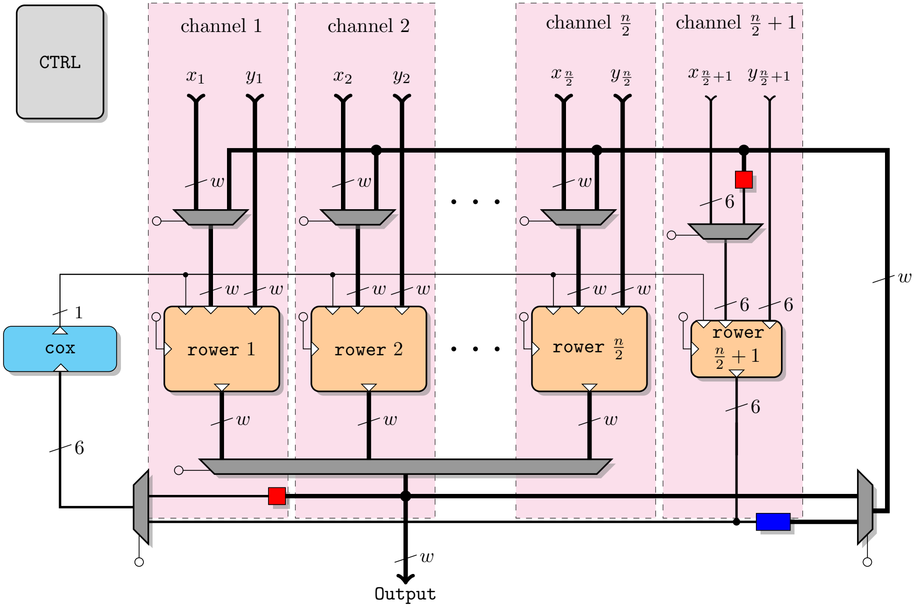
$$XY \equiv \boxed{2}\, K_x K_y + (K_x R_y + K_y R_x) M_a + R_x R_y \equiv U + V M_a \bmod P$$

SBMM Principle 2/2

$$XY \equiv U + V M_a \equiv (K_u + R_v) M_a + (R_u + \boxed{2} K_v) \equiv \boxed{K_z}\ M_a + \boxed{R_z} \bmod P$$



SBMM Architecture with $n/2$ Rowers



Cost of the Algorithms

The output of the algorithm has a few additional bits compared to inputs:

- we use a small extra modulo m_γ to handle them
- in practice $m_\gamma = 2^6$ can be chosen

Algo.	MM [GLP+12]	SBMM	SBMM + Compress
EMM	$\mathbf{2n^2} + 4n$	$\mathbf{n^2} + 5n$	$(\mathbf{n^2} + 7n)$ EMM $+ (n+2)$ GMM
Precomp. EMW	$\mathbf{2n^2} + 10n$	$\mathbf{\frac{n^2}{2}} + 3n$	$\mathbf{\frac{n^2}{2}} + 4n + 2$

EMM is a w-bit modular multiplication
GMM is a one multiplication modulo m_γ (6 bits in practice)
EMW is a w-bit word stored as a precomputation

SBMM is the first RNS modular multiplication algorithm on a single base (two half-bases $= n$ moduli)

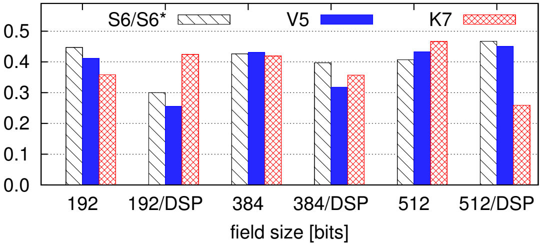
Implementations

FPGA implementations:

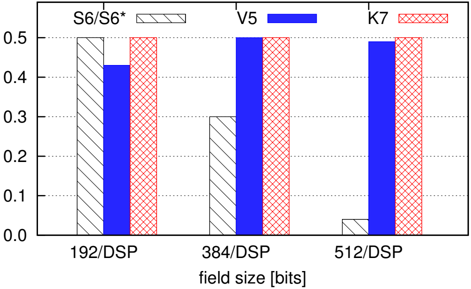
- MM and SBMM have been implemented
- n Rowers (=HW channels) for MM and $n/2$ Rowers for SBMM
  - MM architecture very close to the one in [Gui10]
- 3 field lengths implemented: 192, 384 and 512 bits
- $w = 16$ bits for 192 and 32 for 384 and 512
- on various FPGAs
  - high performance Virtex 5 (LX220)
  - low cost Spartan 6 (LX45/LX100)
  - recent mid-range Kintex 7 (70T)
- 2 configurations: with and without DSP blocks

FPGA Implementation Results (1/2)

Reduction in Slices compared to `MM`: mainly around 40%

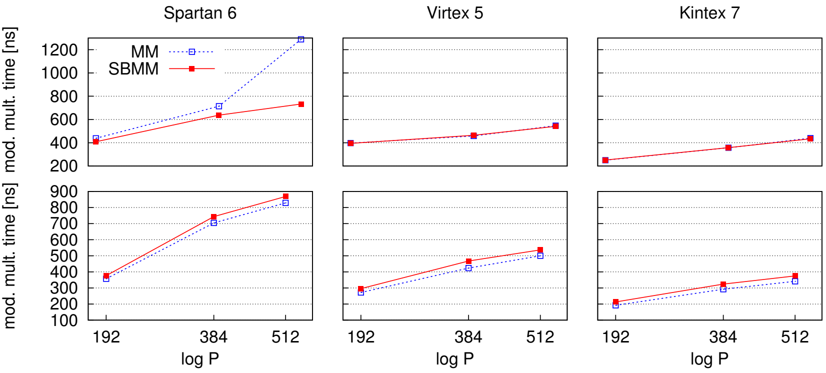


Reduction in DSP blocks 50% for most values



FPGA Implementation Results (2/2)

Timing results for a single modular multiplication with (bottom) and without (top) DSP blocks



Timing overhead always less than 10%

Conclusion

Theoretical conclusions:

- only 1 base : $\#$ moduli / 2
- $\#$ `EMMs` / 2
- $\#$ precomputations / 4
- It works only for special primes P (it is the same for standard primes)

Implementation conclusions:

- the area is almost divided by 2 for a small time overhead ($< 10\,\%$)
- the architecture is still flexible

Further implementation works:

- faster architecture for `SBMM` (factor 2 expected)
- integration in a full RNS ECC cryptosystem
- compatibility with the countermeasures based on RNS

Thank you for your attention

References I

[BDE13] J.-C. Bajard, S. Duquesne, and M. D. Ercegovac.
Combining leak-resistant arithmetic for elliptic curves defined over Fp and RNS representation.
Publications Mathématiques UFR Sciences Techniques Besançon, pages 67–87, 2013.

[BKP09] J.-C. Bajard, M. Kaihara, and T. Plantard.
Selected RNS bases for modular multiplication.
In *Proc. 19th Symposium on Computer Arithmetic (ARITH)*, pages 25–32. IEEE, June 2009.

[BM14] J.-C. Bajard and N. Merkiche.
Double level Montgomery Cox-Rower architecture, new bounds.
In *Proc. 13th Smart Card Research and Advanced Application Conference (CARDIS)*, LNCS. Springer, November 2014.

[BT13] K. Bigou and A. Tisserand.
Improving modular inversion in RNS using the plus-minus method.
In *Proc. 15th Cryptographic Hardware and Embedded Systems (CHES)*, volume 8086 of *LNCS*, pages 233–249. Springer, August 2013.

[BT14] K. Bigou and A. Tisserand.
RNS modular multiplication through reduced base extensions.
In *Proc. 25th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 57–62. IEEE, June 2014.

References II

[DH76] W. Diffie and M. E. Hellman.
New directions in cryptography.
IEEE Transactions on Information Theory, 22(6):644–654, November 1976.

[Elg85] T. Elgamal.
A public key cryptosystem and a signature scheme based on discrete logarithms.
IEEE Transactions on Information Theory, 31(4):469–472, July 1985.

[Gar59] H. L. Garner.
The residue number system.
IRE Transactions on Electronic Computers, EC-8(2):140–147, June 1959.

[GLP$^+$12] F. Gandino, F. Lamberti, G. Paravati, J.-C. Bajard, and P. Montuschi.
An algorithmic and architectural study on Montgomery exponentiation in RNS.
IEEE Transactions on Computers, 61(8):1071–1083, August 2012.

[Gui10] N. Guillermin.
A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$.
In *Proc. 12th Cryptographic Hardware and Embedded Systems (CHES)*, volume 6225 of *LNCS*, pages 48–64. Springer, August 2010.

References III

[JY02] M. Joye and S.-M. Yen.
The Montgomery powering ladder.
In *Proc. 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2523 of *LNCS*, pages 291–302. Springer, August 2002.

[KKSS00] S. Kawamura, M. Koike, F. Sano, and A. Shimbo.
Cox-Rower architecture for fast parallel Montgomery multiplication.
In *Proc. 19th International Conference on the Theory and Application of Cryptographic (EUROCRYPT)*, volume 1807 of *LNCS*, pages 523–538. Springer, May 2000.

[Kob87] N. Koblitz.
Elliptic curve cryptosystems.
Mathematics of computation, 48(177):203–209, 1987.

[Mil85] V. Miller.
Use of elliptic curves in cryptography.
In *Proc. 5th International Cryptology Conference (CRYPTO)*, volume 218 of *LNCS*, pages 417–426. Springer, 1985.

[PP95] K. C. Posch and R. Posch.
Modulo reduction in residue number systems.
IEEE Transactions on Parallel and Distributed Systems, 6(5):449–454, May 1995.

References IV

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Communications of the ACM, 21(2):120–126, February 1978.

[SV55] A. Svoboda and M. Valach.
Operátorové obvody (operator circuits in czech).
Stroje na Zpracování Informací (Information Processing Machines), 3:247–296, 1955.

[YFCV14] G. Yao, J. Fan, R. Cheung, and I. Verbauwhede.
Novel RNS parameter selection for fast modular multiplication.
IEEE Transactions on Computers, 63(8):2099–2105, Aug 2014.

FPGA Implementation Results of State-of-Art MM and SBMM Algorithms with DSP Blocks and BRAMs

Algo.	FPGA	ℓ	Slices(FF/LUT)	DSP/BRAM	#cycles	Freq.(MHz)	time(ns)
MM	S6	192	1733(2780/5149)	36/0	50	140	357
MM	S6	384	3668(6267/11748)	58/0	50	71	704
MM	S6	512	5457(8617/18366)	58/0	58	70	828
SBMM	S6	192	1214(1908/3674)	18/0	58	154	376
SBMM	S6	384	2213(3887/6709)	41/0	58	78	743
SBMM	S6	512	2912(5074/8746)	56/0	66	76	868
MM	V5	192	1941(2957/6053)	26/0	50	184	271
MM	V5	384	3304(5692/10455)	84/12	50	118	423
MM	V5	512	6180(7557/15240)	112/16	58	116	500
SBMM	V5	192	1447(1973/4682)	15/0	58	196	295
SBMM	V5	384	2256(3818/8415)	42/6	58	124	467
SBMM	V5	512	3400(4960/10877)	57/8	66	123	536
MM	K7	192	1732(2759/5075)	36/0	50	260	192
MM	K7	384	3278(5884/9841)	84/0	50	171	292
MM	K7	512	4186(7814/13021)	112/0	58	170	341
SBMM	K7	192	999(1867/3599)	18/0	58	272	213
SBMM	K7	384	2111(3889/6691)	41/0	58	179	324
SBMM	K7	512	3104(5076/8757)	56/0	66	176	375

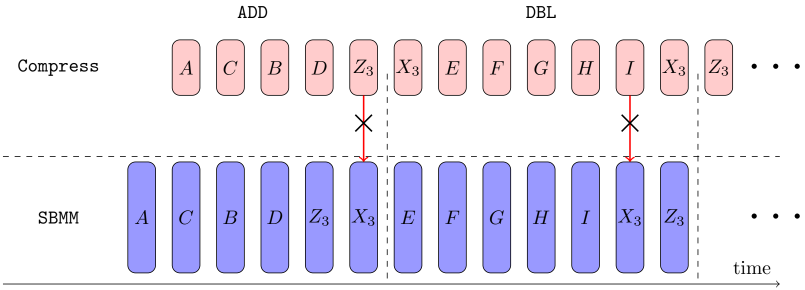
FPGA Implementation Results of State-of-Art MM and SBMM Algorithms without DSP Blocks and BRAMs

Algo.	FPGA	ℓ	Slices(FF/LUT)	#cycles	Freq.(MHz)	time(ns)
MM	S6	192	3238(4288/10525)	50	114	438
MM	S6*	384	7968(8868/27323)	50	70	714
MM	S6*	512	10381(11750/35751)	58	45	1288
SBMM	S6	192	1793(2539/6085)	58	142	408
SBMM	S6*	384	4577(5302/15160)	58	91	637
SBMM	S6*	512	6163(6875/20147)	66	90	733
MM	V5	192	3358(3991/11136)	50	126	396
MM	V5	384	8675(7624/29719)	50	109	458
MM	V5	512	11401(10109/39257)	58	106	547
SBMM	V5	192	1980(2444/6888)	58	147	394
SBMM	V5	384	4942(4696/16672)	58	125	464
SBMM	V5	512	6466(6186/22411)	66	122	540
MM	K7	192	3109(4060/10568)	50	200	250
MM	K7	384	7241(7631/27377)	50	140	357
MM	K7	512	9202(10102/35696)	58	132	439
SBMM	K7	192	1999(2494/6368)	58	231	251
SBMM	K7	384	4208(4649/15137)	58	162	358
SBMM	K7	512	4922(6146/19269)	66	152	434

Formulas for $y^2 = x^3 + ax + b$ with RNS optimizations [BDE13] and (X, Z) coordinates [JY02]

Point Operation	$\mathbf{P_1} + \mathbf{P_2}$ (ADD)	$2\ \mathbf{P_1}$ (DBL)
Formulas	$A = Z_1X_2 + Z_2X_1$	$E = Z_1^2$
	$B = 2X_1X_2$	$F = 2X_1Z_1$
	$C = 2Z_1Z_2$	$G = X_1^2$
	$D = aA + bC$	$H = -4bE$
	$Z_3 = A^2 - BC$	$I = aE$
	$X_3 = BA + CD + 2X_GZ_3$	$X_3 = FH + (G - I)^2$
		$Z_3 = 2F(G + I) - EH$

Parallel Execution Flow Using SBMM and Compress



Compress function

Input: $\overrightarrow{K_{a|b|m_\gamma}}$ and $\overrightarrow{R_{a|b|m_\gamma}}$ with $K,\ R < (m_\gamma - 1)M_a$

Precomp.: $\left|M_a^{-1}\right|_{m_\gamma}$

Output: $\overrightarrow{(K_c)_{a|b|m_\gamma}}$, $\overrightarrow{(R_c)_{a|b|m_\gamma}}$ with $K_c < 3M_a$ and $R_c < 3M_a$

$|R_k|_{m_\gamma} \leftarrow \texttt{BE}\left(\overrightarrow{K_a}, \mathcal{B}_a, m_\gamma\right)$ /* $\overrightarrow{(R_k)_a} = \overrightarrow{K_a}$ */

$K_k \leftarrow \left|(K - R_k)M_a^{-1}\right|_{m_\gamma}$

$\overrightarrow{(R_k)_b} \leftarrow \overrightarrow{K_b} - \overrightarrow{(K_k)_b} \times \overrightarrow{(M_a)_b}$

$|R_r|_{m_\gamma} \leftarrow \texttt{BE}\left(\overrightarrow{R_a}, \mathcal{B}_a, m_\gamma\right)$ /* $\overrightarrow{(R_r)_a} = \overrightarrow{R_a}$ */

$K_r \leftarrow \left|(R - R_r)M_a^{-1}\right|_{m_\gamma}$

$\overrightarrow{(R_r)_b} \leftarrow \overrightarrow{R_b} - \overrightarrow{(K_r)_b} \times \overrightarrow{(M_a)_b}$

return $\overrightarrow{(K_r + R_k)_{a|b|m_\gamma}}$, $\overrightarrow{(R_r + 2K_k)_{a|b|m_\gamma}}$