Exact Error Bound of Cox-Rower Architecture for RNS Arithmetic

Shinichi Kawamura, Tomoko Yonemura, Yuichi Komano, and Hideo Shimizu
Corporate Research and Development Center
Toshiba Corporation
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582 Japan
{shinichi2.kawamura, tomoko.yonemura}@toshiba.co.jp

***Abstract*—Residue Number System (RNS) is a method for representing an integer as an *n*-tuple of its residues with respect to a given base. Since RNS has inherent parallelism, it is actively researched to implement fast public-key cryptography using RNS. This paper derives the exact error bound of approximation on the Cox-Rower architecture which was proposed for RNS modular multiplication. This is the tightest bound ever found and enables us to find new parameter sets for the Cox-Rower architecture, which cannot be found with old bounds.**

Keywords— Residue Number System; error bound; base extension; cryptography

I. INTRODUCTION

Residue Number System (RNS) is one of the methods for representing an integer, where a given integer *x* is represented as an *n*-tuple of its residues divided by the base integers. The base is a set of integers which are pairwise co-prime to each other. If we denote the base as $\mathcal{B} = \{m_1, m_2, \cdots, m_n\}$ and RNS representation of *x* as $(x_1, x_2, \cdots, x_n)$, it holds that $x_i = x \bmod m_i$.

In RNS, addition, subtraction and multiplication are carried out by independent addition, subtraction and multiplication with respect to each base element. Therefore, *n*-folds speeding-up can be achieved by implementing the *n* parallel processing units compared to the case with a single processing unit. This parallelism seems suitable to efficiently implement public-key cryptography which is constructed by several hundred or several thousand bits of integer operations with modular reduction. But a modular reduction in RNS had not been easy to carry out before it was replaced with Montgomery reduction in [1]. After the proposal of RNS Montgomery reduction, promising results have been obtained for RSA scheme [2]-[6], Elliptic Curve Cryptosystem [7]-[13], Pairing-based Cryptosystem [14]-[15], modular inversion [16], Lattice-based Cryptosystem [17], and an architectural study [18].

The Chinese remainder theorem (CRT) plays a crucial role in the implementation of RNS Montgomery reduction. The CRT describes the relationship between RNS representation and radix representation. In [3], an approximation method for computing CRT was proposed and a hardware design suitable to the computation was also proposed as the Cox-Rower architecture, which was investigated or used in many RNS works [4], [6], [9]-[18]. This paper, as well, investigates the same approximation and the Cox-Rower architecture.

At first, Cox-Rower architecture was implemented on ASIC in [4]. After [9], FPGA implementation become popular because recent FPGA has an ability to implement many DSP (Digital Signal Processor) modules on a single FPGA chip and thus, RNS implementation becomes cost-effective especially for small number production. Although error bound used for the design has an acceptable accuracy for most practical implementations, all Cox-Rower implementations so far use inexact error bound. Therefore, it seems hard, not to say impossible, to totally optimize the performance, to eliminate unnecessary margin, to describe comprehensive error property and to facilitate further discovery of new RNS algorithms.

In this paper, we will derive an exact upper bound of approximation error.

This paper organized as follows. In Sect. II, basic notations for RNS and the CRT are introduced. In Sect. III, the exact error bound is derived. Sect. IV concludes this paper.

II. RNS AND CRT

First, we define the following notation

$$\langle x \rangle_m = x \bmod m.$$

The right hand side is usually defined for integers *x* and *m* (*m* > 1). We will extend the definition to a real number *x*, for $m = 1$ as

$$\langle x \rangle_1 = x - \lfloor x \rfloor.$$

Since the symbol $\lfloor x \rfloor$ means the minimum integer that does not exceed *x*, $\langle x \rangle_1$ represents an operation to extract fractional part of *x*. Since $\langle x \rangle_m = x \bmod m = x - m\lfloor x/m \rfloor$ for *m* > 1, we can regard $\langle x \rangle_1$ as a natural extension for $m = 1$.

As an RNS base, we prepare pairwise co-prime n integers to obtain a set $\mathcal{B} = \{m_1, m_2, \cdots, m_n\}$, where, for any i, j $(i \neq j)$, $\gcd(m_i, m_j) = 1$ holds. Then, we define RNS representation of *x* as

$$(x_1, x_2, \cdots, x_n) = \left(\langle x \rangle_{m_1}, \langle x \rangle_{m_2}, \cdots, \langle x \rangle_{m_n}\right).$$

In order to introduce the CRT, we define a few constants by

$$M = \sum_{i=1}^{n} m_i,$$

$$M_i = \frac{M}{m_i}.$$

Finally, when $\gcd(x, m_i) = 1$ holds, a multiplicative inverse of x exists with respect to a base element m_i. We denote it as $\langle x^{-1} \rangle_{m_i}$. By the definition of a multiplicative inverse, it follows that $\langle x \cdot \langle x^{-1} \rangle_{m_i} \rangle_{m_i} = \langle 1 \rangle_{m_i}$.

The CRT is a theorem which specifies the relationship between a radix representation of an integer x and its RNS representation $(x_1, x_2, \cdots, x_n)$. Equation (1) is a way to describe the relationship. Here, we call it as an integer formula.

$$x = \left\langle \sum_{i=1}^{n} \langle x_i M_i^{-1} \rangle_{m_i} M_i \right\rangle_M \tag{1}$$

where, $0 \leq x < M$.

By dividing both sides of (1) by M, we obtain another form of CRT. Notice that suffix of a large bracket changes from M to 1. We call it as a rational formula in contrast to an integer formula (1).

$$\frac{x}{M} = \left\langle \sum_{i=1}^{n} \frac{\langle x_i M_i^{-1} \rangle_{m_i}}{m_i} \right\rangle_1 . \tag{2}$$

We will here denote the main part of right hand side of (2) as $f(x)$.

$$f(x) = \sum_{i=1}^{n} \frac{\langle x_i M_i^{-1} \rangle_{m_i}}{m_i} = \sum_{i=1}^{n} \frac{\xi_i}{m_i} \tag{3}$$

The numerators of (3) were substituted by ξ_i defined by

$$\xi_i = \langle x_i M_i^{-1} \rangle_{m_i}.$$

Note:
In [3], in order to construct an efficient base extension algorithm, it was studied to approximate the integer part of function $f(x)$ based on the equation $k = \lfloor f(x) \rfloor$ (Refer to Appendix A). Once k is obtained, x can be computed from RNS representation by

$$x = \sum_{i=1}^{n} \xi_i M_i - kM.$$

III. Approximation of $f(x)$

$f(x)$ is defined as a summation of ξ_i / m_i. We introduce approximations for both denominators and numerators.

We first take r, a bit length of a base m_i, equal to a bit length of a given ALU (Arithmetic Logic Unit)'s word, and take base elements having a form of a pseudo-Mersenne prime.

$$m_i = 2^r - \mu_i$$

By choosing μ_i as a relatively small integer, m_i can be approximated by 2^r.

Next, we approximate numerator ξ_i with the most significant q-bit of ξ_i which is defined by the function $tru(q, \xi_i)$:

$$tru(q, \xi_i) = \left\lfloor \xi_i 2^{-(r-q)} \right\rfloor \cdot 2^{(r-q)} \tag{4}$$

where, *tru* stands for truncation.

Finally, we define approximate function $\tilde{f}(x)$ of $f(x)$ by

$$\tilde{f}(x) = \sum_{i=1}^{n} \frac{tru(q, \xi_i)}{2^r}. \tag{5}$$

Proposition 1 *Function $f(x)$, its approximate function $\tilde{f}(x)$, and upper bound of error e satisfy the following equation:*

$$\tilde{f}(x) \leq f(x) \leq \tilde{f}(x) + e \tag{6}$$

where,

$$e = n(2^{-q} - 2^{-r}) + \frac{1}{2^r} \sum_{i=1}^{n} \left(1 - \frac{1}{m_i}\right) \mu_i. \tag{7}$$

■

Refer to Appendix B for the proof of the Proposition 1. The first term of (7) originates from the approximation of numerators and the remaining terms from that of denominators. Equation (6) may be rewritten as $f(x) - e \leq \tilde{f}(x) \leq f(x)$.

From proposition 1, we can derive new proposition below regarding base extension, which is an improvement of Theorem 1 of [3] and Theorem 2 of [12]. This provides a condition that the base-extension function extends the base without error (See Appendix A and C).

Proposition 2 *Let e be an error bound defined by* (7), $k = \lfloor f(x) \rfloor$, *and* $\tilde{k} = \lfloor \tilde{f}(x) + \alpha \rfloor$, *where* α *is an offset satisfying* $e \leq \alpha < 1$. *If x is in the range of* $0 \leq x < (1 - \alpha)M$, *then* $\tilde{k} = k$. ■

Table 1. RNS Base design for 512 bit modulus

r (bit)	n	log$M = r*n$ > 512 bit	q_{min} (bit)	e	Note
17	31	527	7	0.281	Same as [12]
16	33	528	7	0.357	New
15	35	525	7	0.497	New
14	37	518	11	0.482	New
13	-	-	-	-	Not applicable

Table 2. RNS Base design for $r = 32$ and $\alpha = 0.5$

Modulus size (bit)	n	q_{min} (bit)	Residual error e_0
160	6	4	2.1×10^{-8}
256	9	5	6.0×10^{-8}
512	17	6	2.8×10^{-7}
1024	33	6	1.3×10^{-6}
2048	65	7	6.6×10^{-6}
4096	129	9	3.0×10^{-5}

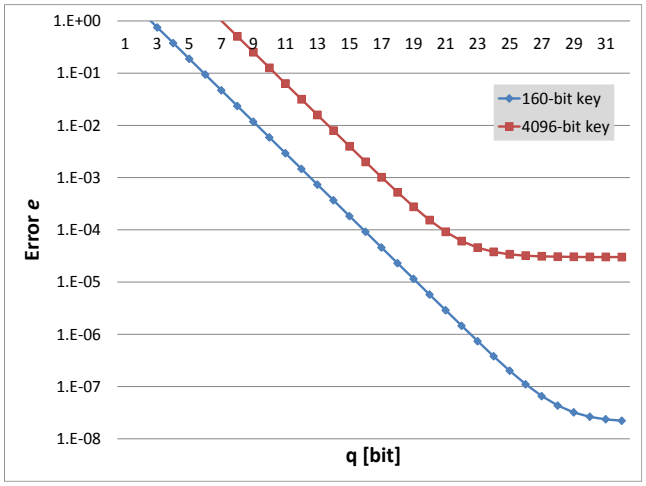

Fig. 1 Error bound e versus q for $r = 32$

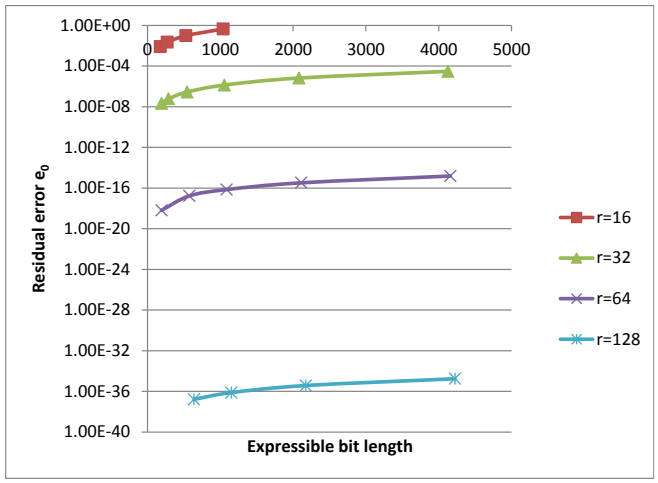

Fig. 2 Residual error versus expressible bit length

Even before (7), we have already known some error bounds which include Max(μ_i) instead of μ_i as a part of their representation, such as the one in [3]. Although not exact, it provides sufficient accuracy for the design of original Cox-Rower architecture. Recently, refinement of parameter design for a new type of Cox-Rower architecture has been proposed based on the closer evaluation of error bound [12]. New algorithm, named double level Montgomery, is different from the original in that it employs Montgomery reduction within a Rower unit instead of the ordinary modular reduction. This shows that refinement of error bound leads to improvement of a scheme. This motivates the derivation of exact upper bound in the Proposition 1, which makes it possible to find much finer parameter design. Indeed, this is the tightest bound ever found.

In Table 1, we show some parameters for Elliptic Curve Cryptosystem computed based on Proposition 1 assuming the double level Montgomery [12] for 512 bit modulus. Table 1 also shows e's value computed by (7). Although no parameter was known so far for $r < 17$, we found new parameters for r = 14-16. Even with (7), no parameter exists for $r \leq 13$.

Table 2 shows the case for the original Cox-Rower architecture. In this case r is fixed to 32 and the offset value α is assumed to be 0.5. Compared to the result for 1024 bit RSA in [3], q is improved from 7 to 6. This means Cox unit can be constructed with 6 bit adder rather than 7 bit adder. The last column of Table 2 shows the residual error e_0 defined by the second term of (7), i.e., $e_0 = 2^{-r}\sum_{i=1}^{n}(1 - 1/m_i)\mu_i$. This is the error that remains even if $q = r$ is selected. For a given offset α, effective range of q can be represented by the following equation.

$$\left\lceil \log_2 \frac{1}{(\alpha - e_0)n^{-1} + 2^{-r}} \right\rceil \leq q \leq r$$

The lower bound can be derived by substituting (7) to the condition of $e \leq \alpha$.

In both tables, two sets of base are generated for each case, since two sets of base are required for RNS Montgomery reduction. For instance, in the first case of Table 1, we generate two bases with n = 31. In searching the base, we took a simple strategy to increment μ_i stating from an initial value and add m_i to the base if it is co-prime with every already selected modulus. Since (7) can be computed once the base is fixed, we can estimate error bound with (7) regardless of strategy for base choice.

Relationship between q and e with $r = 32$ is shown in Fig. 1 for the cases that target key size are 160-bit and 4096-bit. Other cases in Table 2 fall between these two cases. Potential range of q is from 1 to r. Since no feasible parameter exits for $e \geq 1$, we assume a condition $e \leq \alpha = 0.5$ in Table 2. In this case, 4 is the minimum value of q for 160-bit case and $q = 9$ is for 4096-bit case. Both values appear in Table 2. Around $q = 28$, the graph of 160-bit approaches to the floor determined by the residual error e_0, and the graph of 4096-bit case floored around $q = 22$. The graphs have a slope of -1 between minimum q and these floors. Graph for bigger key size always locates above that of smaller key size because the first term of (7) is proportional to base size n. Similarly, the bigger the target key size, the higher the floor becomes. This is due to the second term of (7) is larger for bigger key size. If a bigger (resp. smaller) r is chosen under a fixed key size, location of the graph becomes lower (resp. higher). Figure 2 shows residual error e_0 for various sizes of r and n, though the latter does not explicitly appear on the graph.

IV. Conclusion

This paper has derived the exact error bound of approximation in the CRT process implemented on the Cox-Rower architecture. Although first error bound was proposed in [3] and much closer bound was proposed in [12], this paper has provided the tightest bound ever found. Such a bound will work effectively in the optimization of the Cox-Rower architecture design, like the one discussed in [12]. Actually, we have shown new optimum parameters both for double level Montgomery Cox-Rower architecture and the original one.

Finally, fast computation using RNS is realized by trading off computation time against number of processing units operating in parallel. Therefore, this approach is significantly

advantageous in such cases where coefficient of hardware cost is smaller or expensive hardware is acceptable. Recently, such a case is becoming more realistic since many DSP modules can be implemented on a single FPGA, and in addition, multicore CPU and GPU become commodities. It seems, however, that current multicore CPU and GPU are not designed taking account of RNS. If they are intended for RNS, they will affect software implementation of pub-lick key cryptography enormously. Moreover, the modularity and the regularity of RNS architecture seem to have a great affinity for leading-edge VLSI technologies such as 3D (three dimension) circuit. It can be expected that the exact error bound proposed here will contribute to the development of such RNS oriented computation platforms.

REFERENCES

[1] K. C. Posch and R. Posch, "Modulo reduction in residue number systems," IEEE Trans. on Parallel and Distributed System, Vol.6, No.5, pp.449-454, May 1995.

[2] J. Schwemmlein, K. C. Posch, R. Posch, "RNS-modulo reduction upon a restricted base value set and its applicability to RSA cryptography," Computer & Security, Vol.17, No.7, pp.637-650, 1998.

[3] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-rower architecture for fast parallel Montgomery multiplication," EUROCRYPT2000, LNCS1807, pp.523-538, Springer 2000.

[4] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura, "Implementation of RSA algorithm based on RNS Montgomery multiplication," CHES2001, LNCS2162, pp. 364-376, Springer 2001.

[5] J.-C Bajard and L. Imbert, "A full RNS implementation of RSA," IEEE Trans. on Computer, (Brief contributions) Vol. 53, No.6, pp.769-774, June 2004.

[6] F. Gandino, F. Lamberti, G. Paravati, J.-C. Bajard, and P. Montuschi, "An algorithmic and architectural study of Montgomery exponentiation in RNS," IEEE Trans. on Computers, Vol. 61, No.8, pp.1071-1083, 2012.

[7] D. M. Schinianakis, A. P. Kakarountas, and T. Stouraitis, "A new approach to elliptic curve cryptography: an RNS architecture," Proc. of IEEE MELECON 2006, pp.1241-1245, May 16-19, Benalmadena (Malaga), Spain.

[8] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Souraitis, "An RNS implementation of an Fp elliptic curve point multiplier," IEEE Trans on Circuits and Systems, Vol.56, No.6, pp.1202-1213, June 2009.

[9] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over **F**p," CHES2010, LNCS6225, pp. 48-64, Springer 2010.

[10] S. Antao, J.-C. Bajard, and L. Sousa, "RNS-based elliptic curve point multiplication for massive parallel architectures," Compu. J., 55(5): 629-647, May, 2012.

[11] D. M. Schinianakis and T. Souraitis, "Multifunction residue architectures for cryptography," IEEE Trans on Circuits and Systems, Vol. 61, No.4, pp.1156-1169, April 2014.

[12] J.-C. Bajard and Nabil Merkiche, "Double level Montgomery cox-rower architecture, new bounds," Smart Card Research and Advanced Applications (CARDIS), LNCS 8968, pp.139-153, Springer 2015.

[13] K.Bigou and A. Tisserand, "Single base modular multiplication for efficient hardware RNS implementations of ECC", CHES2015, LNCS9293, pp.123-140, Springer 2015.

[14] R. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, and G. Yao, "FPGA implementation of pairing using residue number system and lazy reduction," CHES2011, LNCS6917, pp. 421-441, Springer 2011.

[15] G. X. Yao, J. Fan, R. C.C. Cheung, and I. Verbauwhede, "Faster pairing coprocessor architecture," Pairing 2012, LNCS 7708, pp.160-176, Springer 2012.

[16] K. Bigou and A. Tisserand, "Improving modular inversion in RNS using the plus-minus methods," CHES 2013, LNCS8086, pp.233-249, Springer 2013.

[17] J.-C. Bajard, J. Eynard, N. Merkiche, and T. Plantard, "RNS arithmetic approach in lattice-based cryptography," 22nd IEEE Symposium on Computer Arithmetic, 2015.

[18] B. Gérard, J.-G. Kammerer, and N. Merkiche, "Contribution to the design of RNS architecture," 22nd IEEE Symposium on Computer Arithmetic, 2015.

APPENDIX

A. A base extension algorithm

Definition

Let $(x_1, x_2, \dots, x_n)$ be an RNS representation of $x \in [0, M-1]$ with respect to a base $\mathcal{B} = \{m_1, m_2, \cdots, m_n\}$. Given a new integer m_{n+1} satisfying $\gcd(M, m_{n+1}) = 1$. Base extension is a procedure to compute $x_{n+1} = \langle x \rangle_{m_{n+1}}$ from $\mathcal{B}$ and $(x_1, x_2, \dots, x_n)$.

Base extension 1 (No approximation)

First, compute $k = \lfloor f(x) \rfloor$, where $f(x)$ is defined by (3). Then, compute

$$x_{n+1} = \left\langle \sum_{i=1}^{n} \langle x_i M_i^{-1} \rangle_{m_i} M_i - kM \right\rangle_{m_{n+1}}.$$

Computing *k* without approximation is inefficient.

Base extension 2 (With approximation)

First, compute $\tilde{k} = \lfloor \tilde{f}(x) + \alpha \rfloor$, where $\tilde{f}(x)$ is defined by (5). Then, compute

$$\tilde{x}_{n+1} = \left\langle \sum_{i=1}^{n} \langle x_i M_i^{-1} \rangle_{m_i} M_i - \tilde{k} M \right\rangle_{m_{n+1}}.$$

$\tilde{k}$ can be computed by *n* times of *q*-bit addition.

Theorem 1 of [3] claimed that for the error bound e' defined in [3]:

If x is in the range of $0 \le x < (1-\alpha)M$ for a given offset α satisfying $e' \le \alpha < 1$, then $\tilde{k} = k$. Thus, $\tilde{x}_{n+1} = x_{n+1}$ is obtained.

Namely, a base extension is correctly carried out under such conditions.

B. Proof of the Proposition 1

First, we expand the denominator of $1/m_i$ as follows:

$$\frac{1}{m_i} = \frac{1}{2^r} \cdot \frac{1}{1 - \mu_i/2^r} = \frac{1}{2^r} \cdot \frac{1}{1-\gamma_i}.$$

We introduce a new variable here.

$$\gamma_i = \mu_i / 2^r$$

By the definition of truncation function, it follows

$$tru(q,\xi_i) \le \xi_i \le tru(q,\xi_i) + 2^{r-q} - 1.$$

By dividing each side by m_i, we obtain

$$\frac{tru(q,\xi_i)}{2^r(1-\gamma_i)} \le \frac{\xi_i}{m_i} \le \frac{tru(q,\xi_i) + 2^{r-q} - 1}{2^r(1-\gamma_i)}$$

$$\frac{tru(q,\xi_i)}{2^r} \le \frac{\xi_i}{m_i} \cdot (1-\gamma_i) \le \frac{tru(q,\xi_i) + 2^{r-q} - 1}{2^r}$$

$$\frac{tru(q,\xi_i)}{2^r} + \frac{\xi_i}{m_i} \cdot \gamma_i \le \frac{\xi_i}{m_i} \le \frac{tru(q,\xi_i)+2^{r-q}-1}{2^r} + \frac{\xi_i}{m_i} \cdot \gamma_i. \quad \text{(A1)}$$

Equation (A2) is derived since the range of ξ_i is $[0, m_i - 1]$.

$$0 \le \frac{\xi_i}{m_i} \cdot \gamma_i \le \left(1 - \frac{1}{m_i}\right)\gamma_i \quad \text{(A2)}$$

Equation（A1）can be modified by taking account of (A2),

$$\frac{tru(q,\xi_i)}{2^r} \le \frac{\xi_i}{m_i} \le \frac{tru(q,\xi_i)}{2^r} + (2^{-q} - 2^{-r}) + \left(1 - \frac{1}{m_i}\right)\gamma_i.$$

Taking summation of each side, we obtain

$$\sum_{i=1}^{n} \frac{tru(q,\xi_i)}{2^r} \le f(x) \le \sum_{i=1}^{n} \frac{tru(q,\xi_i)}{2^r} + n(2^{-q} - 2^{-r}) + \sum_{i=1}^{n} \left(1 - \frac{1}{m_i}\right)\gamma_i.$$

Thus, the final result is obtained as

$$\tilde{f}(x) \le f(x) \le \tilde{f}(x) + n(2^{-q} - 2^{-r}) + \frac{1}{2^r}\sum_{i=1}^{n}\left(1 - \frac{1}{m_i}\right)\mu_i.$$

C. Proof of the Proposition 2

What to prove is $k \le \tilde{f}(x) + \alpha < k + 1$, because this leads to $\lfloor \tilde{f}(x) + \alpha \rfloor = k$.

From the Proposition 1, it holds $f(x) - e \le \tilde{f}(x) \le f(x)$. By adding α to each side, we obtain

$$f(x) + \alpha - e \le \tilde{f}(x) + \alpha \le f(x) + \alpha.$$

Since $e \le \alpha$, the leftmost side follows

$$f(x) + \alpha - e \ge f(x) = \lfloor f(x) \rfloor + \langle f(x) \rangle_1 = k + \frac{x}{M} \ge k.$$

Similarly, the rightmost side follows

$$f(x) + \alpha = \left(k + \frac{x}{M}\right) + \alpha < k + (1-\alpha) + \alpha = k + 1.$$

Inequality is due to the condition $x < (1-\alpha)M$.