|  |  |
|---|---|
| 報告番号 | ※　　　第　　　号 |

# 主　論　文　の　要　旨

| | |
|---|---|
| 論文題目 | Studies on Modular Arithmetic Hardware Algorithms for Public-key Cryptography （公開鍵暗号のための剰余系演算のハードウェアアルゴリズムに関する研究） |
| 氏　　名 | Marcelo Emilio Kaihara |

## 論 文 内 容 の 要 旨

Public-key cryptography plays an important role in digital communication and storage systems. Processing public-key cryptosystems requires huge amount of computation, and, there is therefore, a great demand for developing dedicated hardware to speed up the computations. In this thesis, we focus on modular arithmetic hardware algorithms for public-key cryptosystem since these two operations are the computationally most intensive parts in encryption and decryption processes.

After reviewing major algorithms for computing modular multiplication and division in Chapter 2, we present in Chapter 3, a mixed radix-4/2 algorithm for modular multiplication/division suitable for VLSI implementation. The hardware algorithm is based on the Montgomery multiplication algorithm for modular multiplication and the Extended Binary GCD algorithm for modular division. These two algorithms are combined into the proposed algorithm in order to share hardware components. The new algorithm carries out both calculations using simple operations such as shifts, additions and subtractions. The radix-2 signed-digit representation is used to avoid carry propagation in all additions and subtractions. A modular multiplier/divider based on the algorithm performs an $n$-bit modular multiplication/division in $O(n)$ clock cycles where the length of the clock cycle is constant and independent of $n$. A modular multiplier/divider based on this hardware algorithm has a linear array structure with a bit-slice feature and can be implemented with much smaller hardware than that necessary to implement both multiplier and divider separately.

Chapter 4 presents a hardware algorithm for modular multiplication/division

based on the extended Euclidean algorithm. This hardware algorithm performs modular division, Montgomery multiplication, and ordinary modular multiplication. In order to calculate Montgomery multiplication, we propose a new computation method that consists of processing the multiplier from the most significant digit first. The ordinary modular multiplication is based on the interleaved modular multiplication algorithm. Each of these three operations is carried out through the iteration of simple operations such as shifts and additions/subtractions. In order to avoid carry propagation in all additions and subtractions, the radix-2 signed-digit representation is employed. A modular multiplier/divider based on the algorithm has a linear array structure with a bit-slice feature and carries out $n$-bit modular multiplication/division in $O(n)$ clock cycles, where the length of the clock cycle is constant and independent of $n$. This multiplier/divider can be implemented using a hardware amount only slightly larger than that of the modular divider.

Chapter 5 presents a new fast method for calculating modular multiplication named Bipartite Modular Multiplication. The calculation is performed using a new representation of residue classes modulo $M$ that enables the splitting of the multiplier into two parts. These two parts are then processed separately, in parallel, potentially doubling the calculation speed. The upper part and the lower part of the multiplier are processed using the interleaved modular multiplication algorithm and the Montgomery algorithm respectively. Conversions back and forth between the original integer set and the new residue system can be performed at speeds up to twice that of the Montgomery method without the need for precomputed constants. This new method is suitable for both hardware implementation; and software implementation in a multiprocessor environment.

A fast hardware algorithm for calculating modular multiplication based on this method is presented at the end of this chapter. In this hardware algorithm, the addition of the partial products to the intermediate accumulated product is pipelined in order to reduce the critical path delay. A radix-4 version of the hardware algorithm is then given and its hardware implementation is discussed.

Finally, in Chapter 6 we conclude that taking advantage of similarities and symmetries is a good technique for reducing hardware requirement and for speeding up the calculations.