Modulo *M* multiplication-addition: algorithms and FPGA implementation

J.-L. Beuchat and J.-M. Muller

Variants of a modular multiplication algorithm originally due to Koç and Hung, that are especially suited for FPGA implementation, and that allow to compute $(XY + W)$ modulo M, where there is no need to know M at design-time, are presented. Some results obtained on a Xilinx Virtex-E FPGA are shown.

Introduction: Modular multiplication is a key step in many cryptographic applications. It is also important for efficiently implementing residue number system (RNS) arithmetic. Various solutions have been suggested for implementing modular multiplication $\langle XY\rangle_M = XY \bmod M$, where M is an n-bit number, among them being use of Montgomery's multiplication algorithm [1], or interleaving the multiplication with the residual computation. This Letter is devoted to the second class of methods. More precisely, we are interested in multiplication algorithms that derive from Horner's algorithm:

$$\langle XY\rangle_M = \langle(\cdots((x_{r-1}Y)\cdot 2 + x_{r-2}Y)\cdot 2 + \cdots)\cdot 2 + x_0Y\rangle_M$$

where r is the number of digits of X. These algorithms (in radix 2) use a recurrence of the form:

$$Q[r-i] = \langle 2Q[r-i-1] + x_{r-i}Y\rangle_M \qquad (1)$$

where $Q[r] = 0$. Such recurrences can easily be modified to allow the computation of $\langle XY + W\rangle_M$. From (1), we get:

$$Q[r-i] = \langle 2Q[r-i+1] + x_{r-i}Y + w_{r-i}\rangle_M \qquad (2)$$

If we add some 'redundancy' to the representation of the residue classes modulo M (i.e. if $Q[r-i]$ belongs to a set of more than M elements), then determining which small multiple of M must be added to or subtracted from $2Q[r-i-1] + x_{r-i}Y$ to keep a bounded value only requires examining a few most significant bits of $2Q[r-i-1] + x_{r-i}Y$. Depending on the target technology, the $Q[j]$'s can be represented in redundant (e.g. carry-save) form (typically, for an ASIC implementation), or in conventional non-redundant binary form for an FPGA implementation. Different variants have been suggested:

(i) Koç and Hung [2] implement (1) in radix 2 with carry-save adders. They perform up to three subtractions at each step to keep the $Q[j]$'s bounded. The operands are in non-redundant form, and the product is obtained in redundant form. To perform modular exponentiation with their method, one needs to insert conversions that make a pipeline implementation inefficient.
(ii) Takagi and Yajima [3] suggest a radix-2 and a radix-4 implementation of the recurrence. The $Q[j]$'s are represented in a redundant signed digit system.
(iii) Jeong and Burleson [4] suggest two radix-2 carry-save implementations of a recurrence, later on improved by Kim and Sobelman [5].

We start from a non-redundant version of Kim and Sobelman's recurrence. We deal with FPGA implementation of modular multiply and accumulate operations. FPGAs are arrays of logic cells (CLBs). Our main target is the Xilinx Virtex-E family of FPGAs. On such circuits, very efficient ripple carry adders (RCAs) are available, so that using redundant addition is no longer interesting. The proof of the theorems, along with a more detailed presentation of our results, can be found in [6].

Radix-2 modular multiplication-addition: Based on Kim and Sobelman's work [5], the first modulo M multiplication-addition algorithm studied in this Letter consists in computing:

$$\begin{cases} T[r-i] = 2P[r-i+1] + x_{r-i}Y + w_{r-i} \\ P[r-i] = \langle T[r-i]\rangle_{2^n} + \varphi(T[r-i] \operatorname{div} 2^n) \end{cases} \qquad (3)$$

for $i = 1$ to r, with $P[r] = 0$ and $\varphi(k) = \langle 2^n \cdot k\rangle_M$. If M is known at design time, the function φ can be stored in a small table, the size of which depends only on the maximal value of $T[r-i] \operatorname{div} 2^n$. Since $0 \le \langle T[r-i]\rangle_{2^n} \le 2^n - 1$ and $0 \le \varphi(T[r-i] \operatorname{div} 2^n) \le M - 1$, we deduce that $P[r-i]$ is an $(n+1)$-bit number, the range of which is:

$$0 \le P[r-i] \le 2^n + M - 2 \le 2^{n+1} - 3$$

Assume now that $0 \le Y \le M - 1$. Substituting $P[r-i+1] = 2^{n+1} - 3$, $w_{r-i} = 1$ and $M = 2^n - 1$ into (3) yields an upper bound of $2^{n+2} + 2^n - 6$ for $T[r-i]$. Thus $T[r-i] \operatorname{div} 2^n$ is a 3-bit number less than or equal to 4, and each bit of $\varphi(T[r-i] \operatorname{div} 2^n)$ can be computed by means of a 3-input table. An in-depth study of the recurrence defined by (3) allows to further reduce the size of the table. Theorem 1 states that $T[r-i]$ is an $(n+2)$-bit number and describes how to compute $\langle XY + W\rangle_M$ from $P[0]$.

Theorem 1: Assume that $X \in \mathbb{N}$, $Y \in \{0, \ldots, M-1\}$ and $W \in \mathbb{N}$. Then, the maxima of $T[r-i]$ and $P[r-i]$ are, respectively, defined by $T_{\max} = 2^{n+2} - 3 - \langle n\rangle_2$ and $P_{\max} = (2^{n+2} + 2^n - 5 - 2\cdot\langle n\rangle_2)/3$. $T_{\max}$ and $P_{\max}$ are reached at $M = [2^{n+1}/3] + 1$. Furthermore, the algorithm returns an $(n+1)$-bit number $P[0] = \langle XY + W\rangle_M + \lambda M$, where λ satisfies:

$$0 \le \lambda \le \begin{cases} 2 & \text{if } 2^{n-1} + 1 \le M \le 2^{n-1} + 2^{n-2} - 1 \\ 1 & \text{if } 2^{n-1} + 2^{n-2} \le M \le 2^n - 1 \end{cases}$$

Fig. 1 describes a possible implementation of an iteration stage on a Virtex-E FPGA. In this example, the operator performs the multiplication-addition modulo M_1 or M_2. A *Select* signal allows to chose among these two moduli known at design time. A first RCA carries out the sum of $(2P[r-i+1] + w_{r-i})$ and $x_{r-i}Y$. Each bit of φ is then computed by means of a 3-input table addressed by the most two significant bits of $T[r-i]$ and the *Select* signal. These tables are embedded in the look-up tables (LUTs) of the second RCA which adds $\langle T[r-i]\rangle_{2^n}$ and $\varphi(T[r-i] \operatorname{div} 2^n)$. This iteration stage fits into a single CLB column, which includes two separate carry chains and involves only local routing.

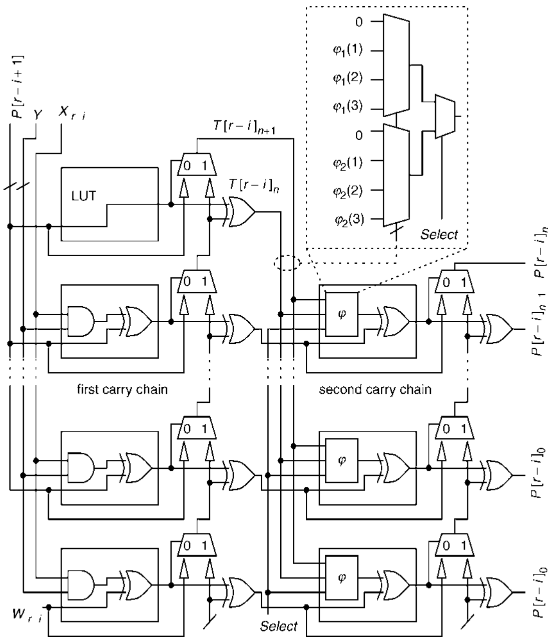

Fig. 1 *Implementation of iteration stage on Virtex-E FPGA*

This first algorithm has a drawback in the sense that $P[0]$ is not a valid input ($P_{\max} > M - 1$). The right-to-left and left-to-right modular exponentiation algorithms require for instance the computation of $\langle P[0]^2\rangle_M$, and a modulo M correction is needed between two consecutive steps. We therefore define a modified iteration:

$$\begin{cases} T'[r-i] = 2P'[r-i+1] + x_{r-i}Y + w_{r-i} \\ P'[r-i] = \langle T'[r-i]\rangle_{2^{n-1}} + \psi(T'[r-i] \operatorname{div} 2^{n-1}) \end{cases} \qquad (4)$$

where $P'[r] = 0$ and $\psi(k) = \langle 2^{n-1}\cdot k\rangle_M$. Theorem 2 guarantees that $P'[0]$ is a valid input and that modular exponentiation does not require intermediate modulo M corrections.

Theorem 2: Assume that X and W belong to N and that Y is an $(n+1)$-bit integer satisfying:

$$0 \le Y \le Y_{\max} < (2^{n+2} + 11 - 4\cdot\langle n\rangle_2)/3$$

The maximal values of $T'[r-i]$ and $P'[r-i]$ are, respectively, defined by $T'_{\max} = 2^{n+2} - 1$ and $P'_{\max} = (2^{n+2} - 7 + 2\cdot\langle n\rangle_2)/3$. *$T'_{\max}$ and*

$P'_{\max}$ are reached at $M = \lfloor (2^{n+1} + 2^{n-1})/3 \rfloor + 1$. Furthermore, $P'[0]$ is either $\langle XY + W \rangle_M$ or $\langle XY + W \rangle_M + M$.

We deduce from Theorem 2 that the computation of each bit of ψ requires the most three significant bits of $T'[r-i]$. If the operator handles a single modulus known at design time, it is again possible to implement an iteration stage in a single CLB column on a Virtex-E device.

Until now, we have assumed that M was a constant and that the values of φ or ψ were pre-computed and included in the VHDL or Verilog code. It is however possible to build the table on-the-fly by means of a subtracter and an adder, and to store the values of φ or ψ in, respectively, three or six registers.

Theorem 3: Define $\alpha(M) = 2M - 2^n$. The functions φ and ψ can be computed recursively on $\mathbb{N}^*$ as follows:

$$\varphi(k) = \begin{cases} \varphi(k-1) - \alpha(M) & \text{if } \varphi(k-1) - \alpha(M) \geq 0 \\ \varphi(k-1) - \alpha(M) + M & \text{otherwise} \end{cases}$$

and

$$\psi(k) = \begin{cases} \psi(k-2) - \alpha(M) & \text{if } \psi(k-2) - \alpha(M) \geq 0 \\ \psi(k-2) - \alpha(M) + M & \text{otherwise} \end{cases}$$

with $\varphi(0) = \psi(0) = 0$ and $\psi(-1) = M - 2^{n-1}$.

Consider the first algorithm described in this Letter and assume that $\varphi(1)$ is determined in a preprocessing step; $\varphi(2)$ and $P[r-1]$ are then computed in parallel. Since $0 \leq Y < M$, we deduce from (3) that $0 \leq T[r-1] \leq M$ and the first iteration only involves $\varphi(0) = 0$. It is obvious that $T[r-1] \operatorname{div} 2^n = 0$ and $0 \leq P[r-1] \leq M$. Consequently, $0 \leq T[r-2] \leq 3M \leq 2^{n+1} + 2^n - 3$ and the second iteration requires $\varphi(0)$, $\varphi(1)$ and $\varphi(2)$ which are already stored in the table. The evaluation of $\varphi(3)$ is again performed in parallel. A similar scheme can be applied to the second algorithm where $\psi(2)$ and $\psi(3)$ are computed from $\psi(0) = 0$ and $\psi(1) = 2^{n-1}$ in a preprocessing step. Then, two values are computed at each step.

Implementation results and conclusions: A modulo M multiplication-addition operator consists for instance of a single iteration stage, an $(n+1)$-bit register to store $P[r-i]$ and a small control unit responsible for the initialisation ($P[r] = 0$). Table 1 summarises some place and route results. Experiment 1 involves an iteration stage which performs a modulo M addition according to (2). At the price of a larger area, we shorten the critical path by computing φ on-the-fly and implementing the algorithm defined by (3) (experiment 2). When the modulus is a constant, we take advantage of the architecture illustrated by Fig. 1 and reduce both area and delay (experiment 3).

Table 1: Area and delay of some modulo M multiplication-addition operators on XCV300E-7 FPGA (each CLB contains two slices)

		$n=8$	$n=16$	$n=24$	$n=32$
1	Area [slices]	23	39	56	73
	Delay [ns]	16	20	23	24
2	Area [slices]	40	72	105	139
	Delay [ns]	12	15	16	16
3	Area [slices]	12	20	30	38
	Delay [ns]	10	12	14	16

Our results show that fast modular multiplication can be achieved on an FPGA, and that the modulus can be an input to the operator: there is no need to know it at design time.

© IEE 2004 *17 February 2004*
Electronics Letters online no: 20040463
doi: 10.1049/el:20040463

J.-L. Beuchat and J.-M. Muller (*Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, 46, Allée d'Italie, F-69364 Lyon Cedex 07, France*)

E-mail: jean-luc.beuchat@ens-lyon.fr

References

1 Montgomery, P.: 'Modular multiplication without trial division', *Math. Comput.*, 1985, **44**, (170), pp. 519–521

2 Koç, C.K., and Hung, C.Y.: 'Carry-save adders for computing the product AB modulo N', *Electron. Lett.*, 1990, **26**, (13), pp. 899–900

3 Takagi, N., and Yajima, S.: 'Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem', *IEEE Trans. Comput.*, 1992, **41**, (7), pp. 887–891

4 Jeong, Y.-J., and Burleson, W.P.: 'VLSI array algorithms and architectures for RSA modular multiplication', *IEEE Trans. Very Large Scale Integr. Syst.*, 1997, **5**, (2), pp. 211–217

5 Kim, S., and Sobelman, G.E.: 'Digit-serial modular multiplication using skew-tolerant domino CMOS'. Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, 2001, (IEEE Computer Society, Arlington, VA, USA), Vol. 2, pp. 1173–1176

6 Beuchat, J.-L., and Muller, J.-M.: 'Opérateurs itératifs de multiplication-addition modulaire pour FPGA', Tech. 2003–40, Laboratoire de l'Informatique du Parallélisme, August 2003. Available at http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2003/RR2003-40.ps.gz