4 BROWER, A.E., and VERHOEFF, T.: 'An updated table of minimum-distance bounds for binary linear codes', *IEEE Trans. Inf. Theory*, 1993, **IT-39**, pp. 662–677
5 FORNEY, G.D.: 'Coset codes - Part I: Introduction and geometrical classification', *IEEE Trans. Inf. Theory*, 1988, **IT-34**, pp. 1123–1151
6 CLARK, G.C., and CAIN, J.B.: 'Error correction coding for digital communications' (Plenum Press, 1081)

Hybrid method for modular exponentiation with precomputation

C.-Y. Chen, C.-C. Chang and W.-P. Yang

Indexing terms: Cryptography, Digital arithmetic

The authors present a new hybrid method for perfonming modular exponentiation using a hybrid ternary–quinary number system. A recent result concerning performing modular exponentiation with precomputation was presented by Dimitrov and Cooklev: their average number of modular multiplications is $0.3381n$, where n is the length of the modulus, while the authors' proposed method only needs $0.3246n$ modular multiplications. Furthermore, compared to Dimitrov and Cooklev's approach, the authors' method reduces the amount of storage by 56.8% if the modulus is a 512 bit number.

Introduction: Modular exponentiation is one of the most important computational problems. A typical example of its applications is in implementing the ElGamal cryptosystem [1]. In that cryptosystem, a fixed element g of a group Z_P, where P is a large prime, is repeatedly raised to many different powers. In a binary number system (BNS), performing the modular exponentiation $g^x \bmod P$ requires on average $1.5n$ modular multiplications, where n is the length of the modulus. Because g is fixed, the values $g^{2^i} \bmod P$ for $i = 0, 1, ..., n-1$ can be computed in advance and loaded in a table for later look-up. Thus, the average number of modular multiplications in this case is $0.5n$. In 1995, Dimitrov and Cooklev [2] presented a new hybrid binary–ternary number system (HBTNS). If the exponent x is represented in such a hybrid binary–ternary form and the precomputed values $g^{2^i3^j} \bmod P$, where $2^i3^j<P$, are stored, then the average number of modular multiplications is reduced to $0.3381n$. However, it needs more precomputed values than BNS. We estimate that this method requires at most $n + (n+1)(\log_3 P - (n/2)\log_3 2)$ stored values. For a 512 bit modulus, it requires 83370 stored values.

In this Letter we present a new hybrid ternary–quinary number system (HTQNS) to represent the exponent. According to this representation, the values $g^{3^i5^j} \bmod P$, where $3^i5^j< P$, should be precomputed and stored. Now, we only focus on nonzero digits, say 1 and 2, in the representation of exponents. We use the technique of [3] to quickly compute $g^x \bmod P$ within $0.3246n$ modular multiplications on average. This result is somewhat faster than that of [2]. On the other hand, the number of stored values in our method is at most $t + (t+1)\log_5 P - (t(t+1)/2)\log_5 3$, where $t = \lfloor \log_3 P \rfloor$. If n is a 512-bit number, then our method requires at most 36027 stored values. Obviously, compared to [2], our method reduces the amount of storage by 56.8%.

Hybrid method:

(1)

Hybrid ternary–quinary number system: The following algorithm is to compute the HTQNS representation of the exponent x:

Input: x
Output: two arrays digit[] and base[]
Step 1: Set $i = 0$.
Step 2: If $x = 0$ then stop and output.
Step 3: If $x \bmod 5 = 0$, set base[i] = 5, digit[i]=0, and $x = x/5$, and goto Step 5.
Step 4: Set $y = x \bmod 3$, digit[i] = y, base[i] = 3, and $x = \lfloor x/3 \rfloor$.
Step 5: Set $i = i + 1$ and goto Step 2.

Take $x = 47$, for example. We have base[] = [3, 5, 3, 3] and digit[] = [2, 0, 0, 1].

Assume the quotient is of the form $15k + t$ at the sth stage. Then, the form of the previous quotient will be one of the following forms:

$\{15k' + i | i = 5*(t \bmod 3)$ or $i = 3*(t \bmod 5) + u$ for $u = 0, 1, 2$ and $i \bmod 5 \neq 0\}$. Let P_s^t, $t \in \{0, 1, 2, .., 14\}$, be the probability of the quotient of the form $15k + t$ at the sth stage. We have a recurrent relation

$$p_s^t = \frac{1}{5} p_{s-1}^{5(t \bmod 3)} + \frac{1}{3} \sum_{\substack{u=0 \\ (u+3t) \bmod 5 \neq 0}}^{2} p_{s-1}^{3(t \bmod 5)+u}$$

That is to say, we have the following recurrent relations:

$$p_s^0 = \frac{1}{5} p_{s-1}^0 + \frac{1}{3} p_{s-1}^1 + \frac{1}{3} p_{s-1}^2$$

$$p_s^1 = \frac{1}{5} p_{s-1}^5 + \frac{1}{3} p_{s-1}^3 + \frac{1}{3} p_{s-1}^4$$

$$\vdots$$

and

$$p_s^{14} = \frac{1}{5} p_{s-1}^{10} + \frac{1}{3} p_{s-1}^{12} + \frac{1}{3} p_{s-1}^{13} + \frac{1}{3} p_{s-1}^{14}$$

Obviously, let $P_0^0 = p_0^1 = ... = p_0^{14} = 1/15$. The asymptotic solutions for the above recurrent relations are $p_\infty^0 = p_\infty^5 = p_\infty^{10} = 0.056$, $p_\infty^1 = p_\infty^6 = p_\infty^{11} = 0.058$, $p_\infty^2 = p_\infty^7 = p_\infty^{12} = 0.076$, $p_\infty^3 = p_\infty^8 = p_\infty^{13} = 0.058$ and $p_\infty^4 = p_\infty^9 = p_\infty^{14} = 0.084$. Therefore, the proportion of the quinary digits is $P_Q = p_\infty^0 = p_\infty^5 = p_\infty^{10} = 0.168$ and the proportion of the ternary digits is $P_T = 1-p_Q = 1-0.168 = 0.832$. The average base of the proposed number system can be evaluated by $b = 3^{p_T}5^{p_Q} \simeq 3.268$. So, the average length of the digits of x in HTQNS is about $(\log_b 2)n \simeq 0.585n$. In addition, the proportions of zeros, ones and twos are $p_0 = p_\infty^0 + p_\infty^3 + p_\infty^5 + p_\infty^6 + p_\infty^9 + p_\infty^{10} + p_\infty^{12} = 0.446$, $p_1 = p_\infty^1 + p_\infty^4 + p_\infty^7 + p_\infty^{13} = 0.277$ and $p_2 = p_\infty^2 + p_\infty^8 + p_\infty^{11} + p_\infty^{14} = 0.277$, respectively.

(2)

Hybrid algorithm for computing modular exponentiation: Assume that the exponent x in HTQNS is represented as base[] = [b_{m-1}, b_{m-2}, ..., b_0] and digit[] = [d_{m-1}, d_{m-2}, ..., d_0]. Before describing the algorithm for performing modular exponentiation we introduce the theorem of [3].

Theorem 1: Suppose $x = \Sigma_{i=0}^{m-1} a_i x_i$, where $0 \leq a_i < h$. If $g^{x_i} \bmod P$ is precomputed for each $0 \leq i < m$ and if $m + h \geq 2$, then $g^x \bmod P$ can be computed with $m + h - 2$ modular multiplications.

According to this theorem, we give the following algorithm to compute $g^x \bmod P$.

Input: g, P, digit[] and base[]
Output: $g^x \bmod P$
Step 1: Set $b = 1$, $a = 1$, and $d = 2$.
Step 2.. If $d = 0$ then stop and output a.
Step 3. Set $it = 0$, $iq = 0$ and $i = 0$.
Step 4: If base[i] = 5 then set $iq = iq + 1$ and go to Step 7.
Step 5: If digit[i] = d then set $b = b*F[it][iq]$. /* $F[it][iq] = g^{3^{it}5^{iq}} \bmod P$. */
Step 6: Set $it = it + 1$.
Step 7: Set $i = i + 1$. If $i = m$, then if $d = 2$ then $a = b$ else $a = a * b$, set $d = d - 1$ and go to Step 2.
Step 8: Go to Step 4.

How many modular multiplications are needed in the above algorithm? We shall count only those multiplications where both multiplicants are unequal to 1. Let N_d be the number of d_is which are equal to d. So, when $d = 2$, it needs $(N_2 - 1)$ multiplications. When $d = 1$, it needs $(N_1 + 1)$ multiplications. Because on average $N_1 = mp_1 = (0.585n)(0.277) = 0.162n$ and $N_2 = mp_2 = (0.585n)(0.277) = 0.162n$, the average number of modular multiplications in the algorithm is $0.324n$.

Now, we count the number of stored values. Because $3^i5^j < P$, the number of these is and js will be

$$\lfloor \log_3 P \rfloor + \sum_{k=0}^{\lfloor \log_3 P \rfloor} \left\lfloor \log_5 \frac{P}{3^k} \right\rfloor \leq t + (t+1)\log_5 P - \frac{t(t+1)}{2} \log_5 3$$

where $t = \lfloor \log_3 P \rfloor$. If $P \simeq 2^{512}$, then at most 36027 stored values are needed.

Future work: This point leads us to wonder whether an optimum HTQNS representation exists such that time and space is further reduced. For example, take two bases k and h, where $k > h$ and $h \neq 2$. We can construct the following recurrent relations:

$$p_s^t = \frac{1}{k} p_{s-1}^{k(t \bmod h)} + \frac{1}{h} \sum_{\substack{u=0 \\ (u+ht) \bmod k \neq 0}}^{h-1} p_{s-1}^{h(t \bmod k)+u}, \quad t = 0, 1, 2, \ldots, kh-1$$

These asymptotic solutions will estimate the average number of modular multiplications. Conversely, the amount of storage is at most $t + \Sigma_{j=0}^{t} \lfloor \log_k (P/h^j) \rfloor$, where $t = \lfloor \log_h P \rfloor$. In our view, searching proper bases k and h such that both time and space are reduced is an open problem.

2 January 1996
Electronics Letters Online No: 19960345

C.-Y. Chen and W.-P. Yang (*Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 300, Republic of China*)

C.-C. Chang (*Institute of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan 621, Republic of China*)

References

1 BRICKELL, E.F., GORDON, D.M., McCURLEY, K.S., and WILSON, D.B.: 'Fast exponentation with precomputation'. Proceedings of Eurocrypt' 92 (Springer-Verlag, 1993), pp. 200–207
2 DIMITROV, V., and COOKLEV, T.: 'Two algorithms for modular exponentiation using nonstandard arithmetics', *IEICE Trans. Fundam.*, 1995, **E78-A**, (1), pp. 82–87
3 EL-GAMAL, T.: 'A public key cryptosystem and signature scheme based on discrete logarithms', *IEEE Trans. Inf. Theory*, 1985, **IT-31**, (4), pp. 469–472

Analogue squarer and multiplier based on MOS square-law characteristic

S.-I. Liu and D.-J. Wei

Indexing terms: Multiplying circuits, CMOS analogue integrated circuits, MOSFET

A simple CMOS squarer based on MOS square-law characteristic is presented. This circuit was fabricated in a 0.8μm single-poly double-metal *n*-well CMOS process. Experimental results show that the nonlinearity of the squarer can be kept below 2% across the entire differential input voltage range of ±1V. The total harmonic distortion is < 2% with the input range up to ±0.8V. Moreover, a four-quadrant multiplier can be also realised using the proposed squarers. The proposed circuits are expected to be useful in analogue signal-processing applications.

Introduction: Many CMOS analogue signal-processing building blocks which exploited the square-law model of the MOS transistors have been developed in the literature [1 – 6], e.g. CMOS multipliers, transconductors and resistors based on the square-algebraic identity can be easily realised since the squaring function can be obtained from the well known square-law model of the MOS transistors operated in saturation [1 – 6]. In this Letter, we propose a simple CMOS squarer which can also be used for realising a four-quadrant multiplier. Experimental results are given to verify the theoretical analysis.

Circuit description: The proposed CMOS squarer and its symbol are shown in Fig. 1. Assume that all the devices in Fig. 1 are biased in the saturation region without the body effect. Let the transconductance parameter and the threshold voltage of M_1 to M_8 be equal to K and V_T, respectively. I_B is the DC current. By describing the source currents of M_1 to M_4, the following relations in Fig. 1 can be obtained i.e.

$$I_1 + I_2 = I_2 + I_3 = I_3 + I_4 = I_B \tag{1}$$

where I_i is the source current of the PMOS transistor M_i (for $i = 1$ to 4). From eqn. 1 and after some algebraic calculations, the voltage v_S can be expressed as

$$v_S = \frac{v_M + v_N}{2} \tag{2}$$

The voltage v_S is the averaging value of the voltages v_M and v_N [2, 5]. From the transistors, M_5 to M_8, it is also found that

$$v_M = \frac{v_1 + V_{DD}}{2} \tag{3}$$

and

$$v_N = \frac{v_2 + V_{DD}}{2} \tag{4}$$

Assume that the aspect ratio of M_9 is twice that of M_5 (and M_7). According to the proposed squarer in Fig. 1, the output voltage V_o can be expressed as

$$\begin{aligned} \frac{V_o}{R} &= I_5 + I_7 - I_9 \\ &= K(V_{DD} - v_M - V_T)^2 + K(V_{DD} - v_N - V_T)^2 \\ &\quad - 2K(V_{DD} - v_S - V_T)^2 \end{aligned} \tag{5}$$

where I_i is the source current of the PMOS transistor M_i (for $i = 5$, 7 and 9). Substituting eqns. 2 – 4 into eqn. 5, the output voltage V_o in Fig. 1 can be obtained as

$$V_o = \frac{K}{8} R(v_1 - v_2)^2 \tag{6}$$

A simple CMOS squarer can be realised. For proper operation, it is required that all the devices operated in the saturation i.e.

$$\max\left(v_1, v_2, \frac{v_1 + v_2}{2}\right) < V_{DD} - 2V_T \tag{7}$$

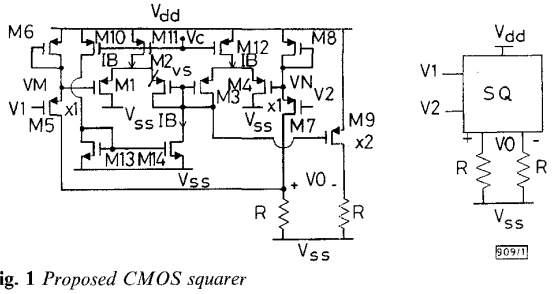

Fig. 1 *Proposed CMOS squarer*

Moreover, based on the square-difference identity ($[a+b]^2 - [a-b]^2 = 4ab$), one squarer with input voltages v_1 and v_2 and the other with input voltages v_1 and $-v_2$ can be used to realise a four-quadrant multiplier as shown in Fig. 2. The output voltage V_o of the multiplier in Fig. 2 can be expressed as

$$V_o = \frac{K}{2} R v_1 v_2 \tag{8}$$

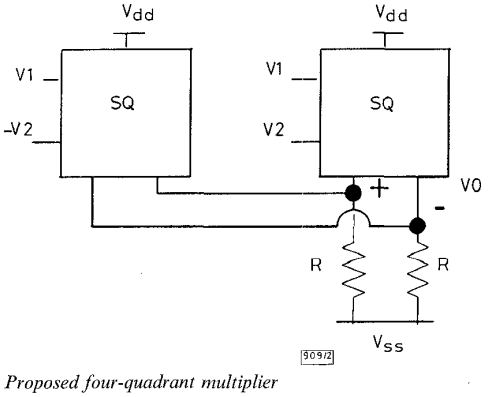

Fig. 2 *Proposed four-quadrant multiplier*