operation, the signature or ciphertext for a given message M is not repeated. This prevents attacks like a probable text attack. Second, we incorporate error-correcting codes and the result is that our scheme provides an error detection and correction capability. Storage requirements for public keys are about 3×10^5 bits, if n is about 300 or 400 bits. In addition, under this scheme, the sender has a very light load, while the receiver bears a heavy computational load.

W. XINMEI

Department of Information Engineering
Xidian University
Xi'an, 710071, People's Republic of China

18th April 1990

References

1 DIFFIE, W., and HELLMAN, M. E.: 'New direction in cryptography', *IEEE Trans.*, 1976, **IT-22**, pp. 644–654
2 RIVEST, R. L., SHAMIR, R., and ADELMAN, L.: 'A method for obtaining digital signatures and public key cryptosystem', *Commun. ACM*, 1978, **21**, pp. 120–126
3 MERKLE, R., and HELLMAN, M.: 'Hiding information and signatures in trapdoor knapsacks', *IEEE Trans.*, 1978, **IT-24**, pp. 520–530
4 MCELIECE, R. J.: 'A public-key cryptosystem based on algebraic coding theory'. DSN Progress Report, 1978, **42–44**, pp. 114–116

CARRY–SAVE ADDERS FOR COMPUTING THE PRODUCT *AB* MODULO *N*

Indexing terms: Digital circuits, Adders, Modular multiplication

The letter describes a new algorithm for modulator multiplication using carry–save adders. The proposed algorithm is based on the sign-estimation technique. A carry–save adder structure consisting of three rows of $n + 3$ simple 1-bit adder cells, and two copies of 3-bit carry look-ahead logic can be used to implement a single step of the algorithm. A completely pipelined array for modular multiplication designed by cascading n carry–save adders performs modulator multiplication at the clock rate.

Introduction: It is possible to compute $P = AB \pmod N$, where $2^{n-1} < N < 2^n$, by first forming AB using binary multiplication algorithms,[1] and then computing the remainder in $AB = qN + r$ where $r < N$. However, this is not an efficient procedure because the second part requires the division of a $2n$-bit number by an n-bit number. An efficient technique that achieves the computation of $P = AB \pmod N$ in n steps was given in Reference 2, where one left shift, one addition, and at most two subtractions are performed at each step. Let A_i represent the ith bit of A. The following algorithm computes $P = AB \pmod N$ by the application of the Horner algorithm:

Algorithm 1
1. Set $P^{(0)} = 0$
2. Repeat Step 2a for $i = 1, 2, 3, \ldots, n$
   2a. $P^{(i)} = 2P^{(i-1)} + A_{n-i}B \pmod N$
3. Halt

In Step 2a, we perform a left shift on partial product $P^{(i-1)}$, and add the value $A_{n-i}B$. We must then reduce the partial product to the range $[0, N)$, i.e., $0 \le P^{(i)} < N$. If the reduction is performed in the ith step, then we have $0 \le B, P^{(i)} < N$. Thus, we obtain $0 \le P^{(i+1)} < 3N$ in the following step. Thus we have to perform up to two subtractions to reduce the partial product. The algorithms given in References 2 and 3 are based on this observation.

Application of sign-estimation technique: To implement the addition operation in Step 2a of the algorithm, we use a word-serial bit-parallel carry–save adder (one-level CSA) which, in one clock cycle, produces two n-bit numbers, C and S, from three n-bit numbers, X, Y, and Z, such that $(C, S) := X + Y + Z$. After the addition, we need to subtract the modulus from the partial product several times until it is reduced to the range $[0, N)$. The difficulty in this approach is computing the sign of the partial product $P^{(i)}$. The carry–save adder does not directly produce $P^{(i)}$; it computes $C^{(i)}$ and $S^{(i)}$ such that $P^{(i)} = C^{(i)} + S^{(i)}$. Thus, we need a technique to compute the sign of $P^{(i)}$, using the bits of $C^{(i)}$ and $S^{(i)}$, without performing an addition operation involving n-bit binary numbers. The sign estimation technique[4] achieves this purpose. We define the function $T(x) = 2^t(2^{-t}x)$, i.e., T replaces the t least significant bits of x with t zeros. It follows that $T(x) \le x < T(x) + 2^t$. By applying this transformation to $C^{(i)}$ and $S^{(i)}$, we obtain

$$T(C^{(i)}) + T(S^{(i)}) \le C^{(i)} + S^{(i)} < T(C^{(i)}) + T(S^{(i)}) + 2^{t+1} \quad (1)$$

A carry–save adder is used to subtract N from $C^{(i)} + S^{(i)}$, which takes $C^{(i)}$, $S^{(i)}$ and $M = -N$ as inputs and produces new $C^{(i)}$ and $S^{(i)}$ values as outputs. Suppose that, after Q subtractions, we obtain $0 \le T(C^{(i)}) + T(S^{(i)})$. If we perform one more subtraction and obtain $T(C^{(i)}) + T(S^{(i)}) < 0$, then $T(C^{(i)}) + T(S^{(i)}) < -2t$, because it must be a multiple of 2^t. Therefore, after Q subtractions, we have $0 \le T(C^{(i)}) + T(S^{(i)}) < N - 2^t$. Thus, the inequality in expr. 1 can be written as

$$0 \le C^{(i)} + S^{(i)} < N - 2^t + 2^{t+1} = N + 2^t \quad (2)$$

In the next step, we compute $C^{(i+1)}$ and $S^{(i+1)}$. Because $C^{(i+1)} + S^{(i+1)} = 2(C^{(i)} + S^{(i)}) + A_{n-i-1}B$, this gives

$$0 \le C^{(i+1)} + S^{(i+1)} < 2(N + 2^t) + N = 3N + 2^{t+1}$$

If we perform three subtractions ($Q = 3$), then $C^{(i+1)} + S^{(i+1)}$ will be less than 2^{t+1}. To satisfy the requirement of eqn. 2, we choose $t + 1 = n$ (since $2^{t+1} = 2^n \le N + 2^t = N + 2^{n-1}$). Thus we can reduce the partial product computed by a carry–save adder using at most three subtraction. This can be achieved by first adding $(2M = -2N)$ to $(C + S)$ and then performing another addition operation, $C + S + M$, depending on the estimated sign of $C + S$. We conclude that when $t = n - 1$ and $T(C^{(i)}) + T(S^{(i)}) \ge 0$, then $0 \le C^{(i+1)} + S^{(i+1)} < 3N + 2^n < 5N$. As all partial products will be in the range $[0, 5N)$, we allocate $n + 3$ bits for $S^{(i)}$ and $C^{(i)}$.

Algorithm 2
1. Set $S^{(0)} = 0$ and $C^{(0)} = 0$.
2. Repeat Step 2a, 2b, and 2c for $i = 1, 2, 3, \ldots, n$
   2a. $(C^{(i)}, S^{(i)}) := 2(C^{(i-1)} + S^{(i-1)}) + A_{n-i}B$
   2b. $(\hat{C}^{(i)}, \hat{S}^{(i)}) := C^{(i)} + S^{(i)} + 2M$.
   If $T(\hat{C}^{(i)}) + T(\hat{S}^{(i)}) \ge 0$ then set $C^{(i)} = \hat{C}^{(i)}$ and $S^{(i)} = \hat{S}^{(i)}$
   2c. $(\hat{C}^{(i)}, \hat{S}^{(i)}) := C^{(i)} + S^{(i)} + M$
   If $T(\hat{C}^{(i)}) + T(\hat{S}^{(i)}) \ge 0$ then set $C^{(i)} = \hat{C}^{(i)}$ and $S^{(i)} = \hat{S}^{(i)}$
3. $(\hat{C}^{(i)}, \hat{S}^{(i)}) := C^{(i)} + S^{(i)} + M$
4. Compute $P := C^{(n)} + S^{(n)}$ and $\hat{P} := \hat{C}^{(n)} + \hat{S}^{(n)}$
5. If $\hat{P} \ge 0$ then set $P = \hat{P}$
6. Halt.

In Algorithm 2, $\hat{C}^{(i)}$, $\hat{S}^{(i)}$ and $\hat{P}$ represent the temporary values of $C^{(i)}$, $S^{(i)}$, and P. We use transformation T to estimate the sign of $C^{(i)} + S^{(i)}$ using the bits of the temporary carry and sum, starting from bit location $t = n - 1$. After the execution of Step 2 for $i = n$, the temporary and primary values of the product are bounded as

$$0 \le P < 2^{t+1} = 2^n < \tfrac{3}{2}N \quad (3)$$

$$-N \le \hat{P} < 2^{t+1} - N = 2^n - N < \tfrac{1}{2}N \quad (4)$$

We compute these values using carry-propagate adders and pick the one in the range $[0, N)$. Since $P = \hat{P} + N$, it follows from eqns. 3 and 4 that $0 \le P < N$, if $\hat{P} < 0$. Otherwise, we pick $\hat{P} > 0$ as the final product, since $\hat{P}$ will be in the range $[0, N)$. There exists no value of $t > 1$ for which the estimation

technique produces the correct sign for all possible values of C and S. Only when $t \leq 1$ can we produce the exact sign, i.e., we have to check all the bits of C and S to find the exact sign of $C + S$. This observation follows from eqn. 2.

Carry–save adder structure: The carry–save adder structure given in Fig. 1 implements Step 2 of Algorithm 2. It consists of three rows of $n + 3$ simple 1-bit adder cells and two copies of 3-bit carry look-ahead logic, as shown in Figs. 1 and 2. The

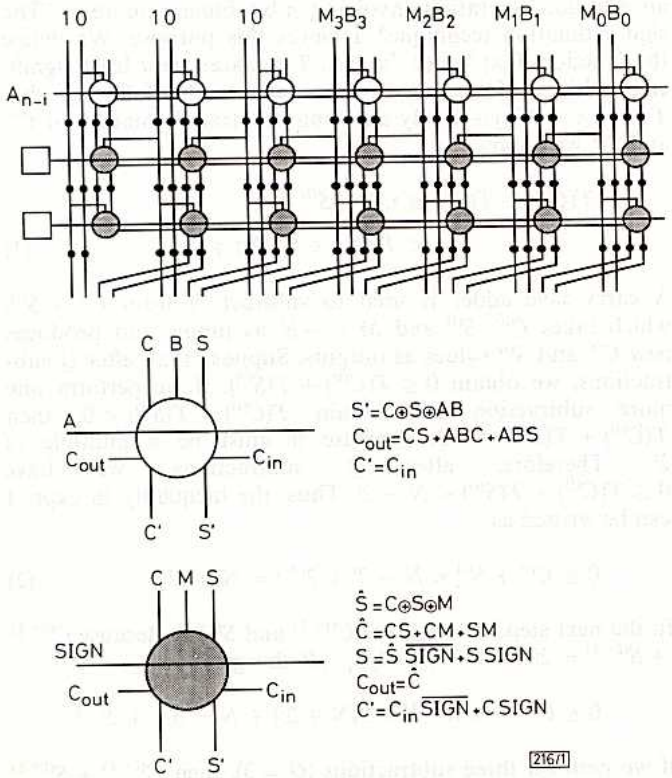

Fig. 1 *One-level carry–save adder structure*

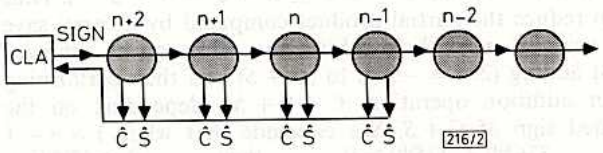

Fig. 2 *Carry look-ahead logic*

black circles in Fig. 1 are 1-bit latches. The carry–save adder structure receives A, B and $M = -N$ (in two's complement form) from the top in a word-serial bit-parallel fashion. The first row of the structure implements Step 2a, and the other two rows implement Steps 2b and 2c, respectively. First, temporary values $\hat{C}^{(i)}$ and $\hat{S}^{(i)}$ are computed. Then the carry look-ahead logic attached to last two rows receives the most significant 4 bits of $\hat{C}^{(i)}$ and $\hat{S}^{(i)}$, and computes the sign using

$$\text{SIGN} = \hat{S}_{n+2} \oplus \hat{C}_{n+2} \oplus [G_{n+1} + G_n P_{n+1} + G_{n-1} P_n P_{n+1}]$$

where $G_i = \hat{C}_i \hat{S}_i$ and $P_i = \hat{C}_i + \hat{S}_i$ for $i = n - 1, n, n + 1$. If the estimated sign is positive (i.e., SIGN = 0), then the temporary values are taken to be primary values for the next cycle.

We note that if a one-level carry–save adder structure is used, then the final pairs (C, S) and $(\hat{C}, \hat{S})$ are produced after $3n$ clock cycles. However, a pipelined array can also be designed by cascading n such structures. If the pipe is full at all times, the data rate will be equal to the clock rate. In the last step of the algorithm, the pairs can be summed separately (and in parallel) to produce the final result. This part of the algorithm (Steps 4 and 5) can be performed using a pair of carry–propagate adders with a triangular array of latches. The resulting array has a latency of $4n$ clock cycles and performs modulator modifications at the clock rate.

C. K. KOÇ
C. Y. HUNG

Department of Electrical Engineering
University of Houston
Houston, TX 77204, USA

20th April 1990

References

1 HWANG, K.: 'Computer arithmetic, principles, architecture, and design' (Wiley, 1979)
2 BLAKLEY, G. R.: 'A computer algorithm for the product AB modulo M', *IEEE Trans.*, 1983, **C-32**, pp. 497–500
3 SLOAN, K. R., JUN.: 'Comments on "A computer algorithm for the product AB modulo M"', *IEEE Trans.*, 1985, **C-34**, pp. 290–292
4 KOÇ, Ç. K., and HUNG, C. Y.: 'Multi-operand modulo addition using carry save adders', *Electron. Lett.*, 1990, **26**, (6), pp. 361–363

COMPARISON OF WDM COUPLER TECHNOLOGIES FOR USE IN ERBIUM DOPED FIBRE AMPLIFIER SYSTEMS

Indexing terms: Optical fibres, Multiplexers and multiplexing

Fused-tapered and interference filter technology are compared for specific application in 1480 nm pumped erbium doped fibre amplifiers as WDM multiplexers for signal and pump light. In particular, the temperature, polarisation and wavelength dependence of the coupling ratio are measured and the implications for amplifier gain variability are discussed.

Introduction: Erbium doped fibre amplifiers could offer considerable potential for optical transmission systems.[1,2] One factor which might influence their applicability to, in particular, transoceanic systems is the stability of the optical gain against variations in temperature and polarisation. In multi-amplifier systems this gain will need to be controlled, possibly by adjusting the pump power. Clearly gain variations need to be minimised to ease the requirements of the control loop. One possible contribution to any gain variations may result from the WDM coupler used to multiplex the signal and pump wavelengths. We examine only 1480 nm semiconductor pumping. The important characteristics of such a coupler include the insertion loss of both signal and pump and the variation of these losses with temperature, polarisation and wavelength. In this letter these characteristics are measured for both a commercially available fibre-tailed interference filter multiplexer[3] and a packaged fused tapered coupler.[4]

Coupler characterisation: Fig. 1 shows the wavelength dependent transmission characteristic of the pump path for the two devices. Excess losses in the fused couplers were ~0·1 dB at the optimum wavelengths. The interference filter coupler exhibited ~0·4 dB loss for both signal and pump port. The higher loss is anticipated for devices using bulk optical components. For the interference filter, both the pump and signal

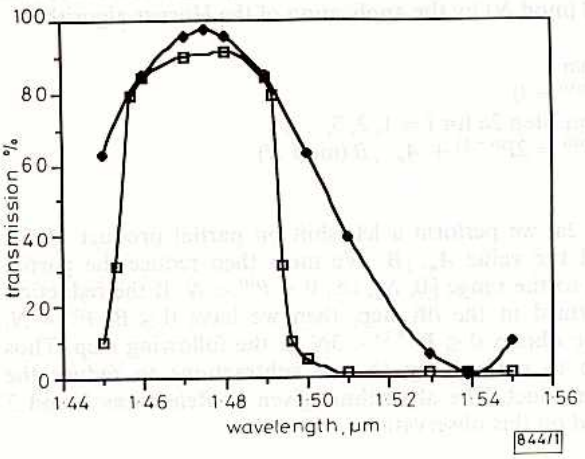

Fig. 1 *Wavelength dependence of pump and signal transmission loss*
⊡ interference filter
● fused coupler