$j = 5$. However, increasing j means also decreasing the space in the memory. From the plots it is observed that there is a great improvement in results when j is increased from 1 to 3. However, there is not much difference when j is increased from 3 to 5. We reach a point of diminishing returns at $j = 3$, and the use of too many past values of the gradient vector does not improve the convergence further. In this section, all the simulation studies are the average of 100 independent trials as well.

IV. Conclusion

In this paper, new adaptive filtering algorithms for impulsive noise environments are introduced. These algorithms are obtained as a generalization of the NLMP algorithm by using FLOS. Under the same assumptions of the LMS convergence, it is shown that the new class of algorithms converge for small enough adaptation step size. The computational complexity of the proposed algorithms is in the same order with that of the NLMP algorithm. It is observed that the new algorithms demonstrate superior performance compared to the previous methods. Also, it is demonstrated that speedup in convergence can be achieved by using a "momentum" version of the update with tolerable increase in the memory requirements.

References

[1] B. Mandelbrot and J. W. Van Ness, "Fractional Brownian motions, fractional noises, and applications," *SIAM Rev.*, vol. 10, pp. 422–437, 1968.
[2] S. S. Pillai and M. Harisankar, "Simulated performance of a DS spread spectrum system in impulsive atmospheric noise," *IEEE Trans. Electromag. Compat.*, vol. EMC-29, pp. 80–82, 1987.
[3] M. Bouvet and S. C. Schwartz, "Comparison of adaptive and robust receivers for signal detection in ambient underwater noise," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 621–626, 1989.
[4] D. Middleton, "Statistical physical models of electromagnetic interference," *IEEE Tran. Electromag. Compat.*, vol. EMC-19, pp. 106–127, 1977.
[5] M. Shao and C. L. Nikias, "Signal processing with fractional lower order moments: Stable processes and their applications," *Proc. IEEE*, vol. 81, pp. 986–1009, July 1993.
[6] S. A. Kassam, *Signal Detection in Non-Gaussian Noise*. New York: Springer-Verlag, 1988.
[7] O. Arıkan, A. E. Çetin, and E. Erzin, "Adaptive filtering for non-Gaussian stable processes," *IEEE Trans. Signal Processing Lett.*, vol. 1, pp. 1–3, Nov. 1994.
[8] A. L. Peressini, F. E. Sullivan, and J. J. Uhl, *The Mathematics of Nonlinear Programming*. New York: Springer-Verlag, 1988.
[9] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction, and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[10] S. C. Douglas, "A family of normalized LMS algorithms," *IEEE Signal Processing Lett.*, vol. 1, pp. 49–51, Mar. 1994.
[11] J. Nagumo and A. Noda, "A learning method for system identification," *IEEE Trans. Automat. Contr.*, vol. AC-12, pp. 282–287, June 1967.
[12] Y. Hosoya, "Discrete time stable processes and their certain properties," *Ann. Prob.*, vol. 6, no. 1, pp. 94–105, 1978.
[13] J. R. Treichler, C. R. Johnson, and M. G. Larimore, *Theory and Design of Adaptive Filters*. New York: Wiley, 1986.

A New Division Algorithm Based on Lookahead of Partial-Remainder (LAPR) for High-Speed/Low-Power Coding Applications

Hyung-Joon Kwon and Kwyro Lee

***Abstract*—A new polynomial division algorithm in finite field GF(2^m) based on the lookahead of partial-remainder (LAPR) is proposed. Since our algorithm is based on partial division on group basis and lookahead technique exploiting the linearity in finite field arithmetic, it is possible to completely eliminate polynomial multiplications leading to highly increased throughput per unit time. The inherent regularity and feedforward nature of our algorithm make it possible to be fully pipelined. When pipelined, its throughput is one quotient and one remainder per clock cycle, regardless of the degree of dividend polynomial, which is orders of magnitude faster than the conventional architecture using linear feedback shift register. An area-efficient sequential architecture based on LAPR is also presented. Although the throughput rate of sequential architecture is lower than that of the pipelined one, it is still higher than that of any division architecture ever reported. Those will be shown to be efficient, regular, and easily expandable, and hence, naturally suitable for very large scale integration implementation. In systems requiring modest speed, the high-speed nature of our proposed architecture can be traded for low-power consumption by reducing clock rate. We verified the general validity of the division algorithm based on LAPR by mathematical manipulation and simulation. The superiority of our proposed architecture compared with other reported ones is demonstrated with regard to its throughput, latency delays, and power.**

***Index Terms*— Author, please supply index terms. E-mail keywords@ieee.org for info.**

I. Introduction

Division in the finite field $GF(2^m)$ is the most important building block in coding systems such as BCH (Bose–Chaudhuri–Hocquenghem) and RS (Reed–Solomon) codes for applications to communications, optical disks, portable equipment, and control and computer systems, since these coding systems are based on long polynomial divisions. The conventional architecture for division in finite fields uses linear feedback shift register (LFSR), which consists of k stage feedback shift register, where k is the degree of the divisor polynomial. A diagram for such a division architecture is shown in Fig. 1. The quantities m_0, m_1, $\cdots$, m_k are the coefficients of the divisor polynomial. However, as the high-speed requirement for real-time audio or video coding capability and the low-power requirement for portable equipment increase, this division architecture using LFSR has shown several limitations as described below.

In some very high-speed applications, the presence of a global feedback signal imposes severe constraints on the switching speed. The fact that the input to all k stages is the feedback signal forces all k stages to be synchronous, necessitating the use of a global clock. The need to distribute the global clock and the feedback signal to all stages of the architecture can seriously restrict the maximum switching speed achievable in a practical implementation. To remedy this, an alternative configuration was suggested by Tong [1], where the feedback path is pipelined such that the feedback signal goes through one delay unit before it is fed back to each shift register

Manuscript received July 12, 1996; revised December 27, 1997. This paper was recommended by Associate Editor B. W. Lee.

The authors are with Korea Advanced Institute of Science and Technology, Yusung-Gu, Taejun 305–71, Korea.

Publisher Item Identifier S 1057-7130(99)01820-0.

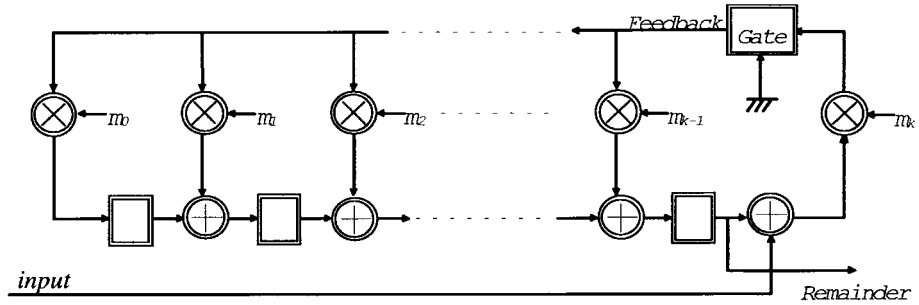

Fig. 1. Conventional division architecture using LFSR (linear feedback shift register).

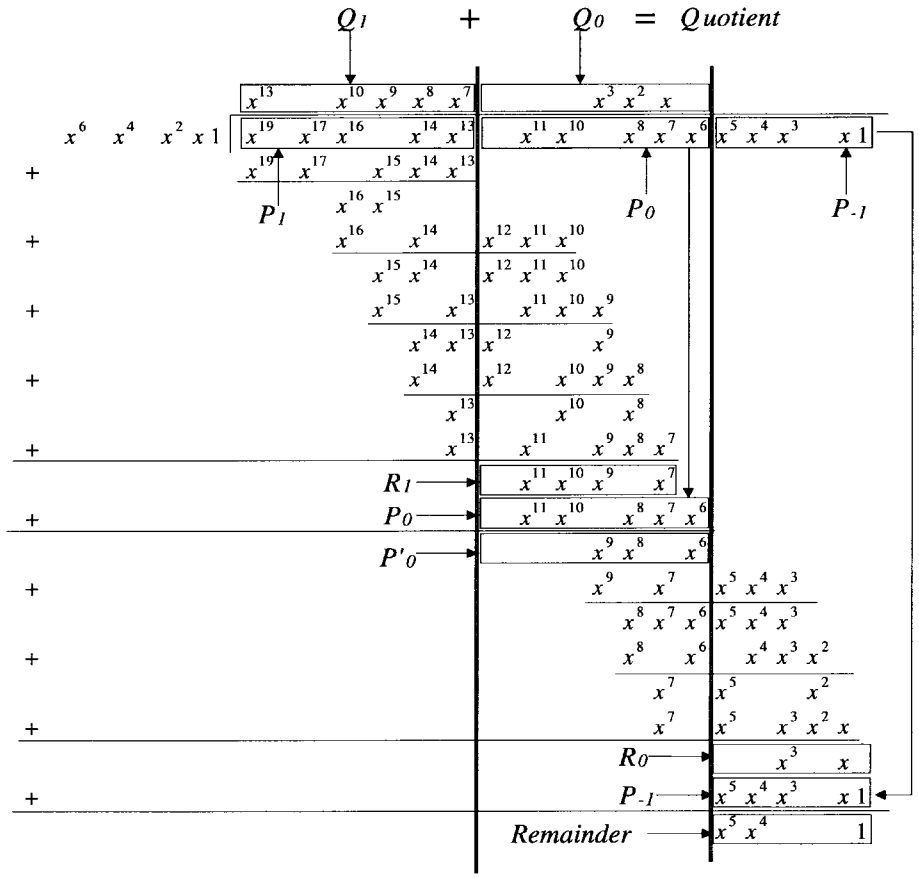

Fig. 2. One example for the division process based on partial-division. Here, the dividend polynomial is: $P(x) = x^{19}+x^{17}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^8+x^7+x^6+x^5+x^4+x^3+x+1$ and the divisor polynomial is: $M(x) = x^6+x^4+x^2+x+1$.

stage. While this can be one of the viable solutions for the problem at hand, the resulting output stream is interleaved by 2 : 1 [1]. This interleaving may not be desirable in certain applications, especially in those where latency delays are critical.

For the serial architectures shown in Fig. 1, the throughput and latency delays are limited by the degree of the dividend polynomial and the number of bits per symbol. For a dividend polynomial which has degree n, the remainder is produced after $(n+1)$ clock cycles for the binary field and after multiples of $(n+1)$ clock cycles for the nonbinary finite field depending on the needed clock cycles for multiplying two field symbols. In designing decoders such as BCH and RS, the delay in division (syndrome generation) has been known to be the bottleneck to achieve high speed [2]–[5].

As for the chip area, the architectures shown in Fig. 1 require k multipliers. Since a circuit for multiplication between two field symbols is complex, area consumption is serious. Although many multiplier circuits have been reported [4]–[7], they still require large area. The division architecture using LFSR often uses lookup tables to perform multiplication between two field symbols [8]. In this case, due to the large ROM size the area is excessive and it becomes infeasible to implement, as the number of bits per symbol and the degree of the divisor polynomial increase. An area-efficient division architecture using Berlekamp's bit-serial multiplier algorithm has been proposed and applied to implement the RS encoder [9], but it has the drawback that one remainder is produced after $(n+1)t$ clock cycles, where t is the number of bits per symbol.

Lastly, in division architecture using LFSR, the fact that the complete LFSR and serial buffer registers should be clocked for every clock cycle without concerning the change of contents cannot avoid useless power consumption [10]. Noting the generally accepted equation $P = cv^2f$, the power consumption in CMOS digital systems is proportional to the total capacitance c, the square of the power

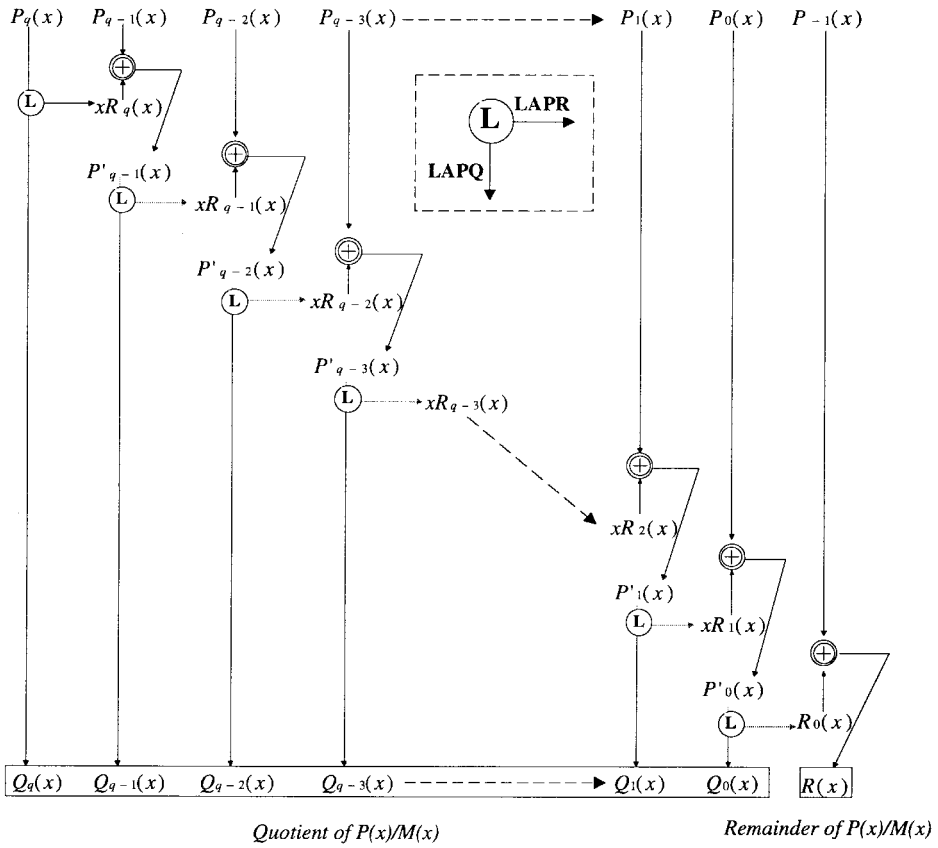

Fig. 3. The schematic diagram of new division algorithm based on LAPR for hardware implementation.

TABLE I
ALL THE NECESSARY INFORMATION FOR LOOKAHEAD CIRCUIT WHEN THE DIVISOR POLYNOMIAL IS: $M(x) = x^6 + x^4 + x^2 + x + 1$. $^*Q_s(x)$ IS NOT NECESSARY, UNLESS THE APPLICATION TO APPLY REQUIRES QUOTIENT EXPLICITLY

Input	*S(x)/x*^{6} INS[6] ~ INS[0]				
	1000000	1010010	$Q_6(x)$	011110	$R_5(x)$
	0100000	0101001	$Q_5(x)$	001111	$R_5(x)$
	0010000	0010100	$Q_4(x)$	101100	$R_4(x)$
	0001000	0001010	$Q_3(x)$	010110	$R_3(x)$
	0000100	0000101	$Q_2(x)$	001011	$R_2(x)$
	0000010	0000010	$Q_1(x)$	101110	$R_1(x)$
	0000001	0000001	$Q_0(x)$	010111	$R_0(x)$
Output of S(x)/M(x)		OUTQ[6] ~ OUTQ[0]		OUTR[5] ~ OUTR[0]	
		+	$Q_S(x)$	+	$R_S(x)$

supply voltage v, and its clock frequency f [11]. Therefore, to reduce power consumption, an efficient strategy which can be taken at the architecture level is to increase the throughput per unit time and to lower the clock frequency as much as possible [11]. When the clock frequency is lower, the timing constraint becomes less critical. This allows the supply voltage to scale down appreciably, leading to the drastic power reduction. One example is the recently proposed parallel duplication of the LFSR architecture to reduce power consumption [10]. In a parallel architecture, the throughput is doubled, which allows the clock frequency to be reduced by one half. However, this inevitably doubles the chip area which increases the total capacitance, leading to no gain in power dissipation unless the supply voltage is properly scaled.

In this paper, we propose a new division algorithm based on LAPR. Since our algorithm is based on partial-division on group basis and lookahead technique, exploiting the linearity in finite field arithmetic, it does not suffer from any of the limitations mentioned above. Furthermore, by employing the lookahead technique, it is possible to completely eliminate polynomial multiplications, leading to the highly increased throughput per unit time. Section II describes the new division algorithm based on LAPR. An efficient strategy for the lookahead of partial-remainder will be discussed in detail. In Section III, we introduce very large scale integration (VLSI) architecture using this new algorithm based on LAPR. Both pipelined and area-efficient sequential architecture will be presented. In Section IV, we compare performance with division architectures reported before. Finally, we draw our conclusion in Section V.

II. NEW DIVISION ALGORITHM BASED ON LAPR

A. Basic Concept of Division Algorithm Based on LAPR

The conventional Euclidean polynomial division process in finite field GF(2^m) can be interpreted as the successive elimination of elements in the dividend polynomial by subtracting (subtraction and addition are identical operations in GF(2^m) [12]) the selected quotient multiple of the divisor polynomial until the degree of the remainder polynomial is reduced to less than that of the divisor polynomial. On the other hand, our division process is the successive group elimination in the dividend polynomial by employing lookahead technique until the last group is reached.

First, we partitioned the long arbitrary dividend polynomial with degree n into several orthogonal groups using the degree of the fixed divisor polynomial, k. The number of elements in each group except the last one is $(k + 1)$. The number of elements in the last group is k, which is the same as the number of elements in the remainder polynomial. The number of groups in the dividend polynomial is $(q + 2)$, where q is the maximum number that satisfies $n \geq q(k + 1) + k$.

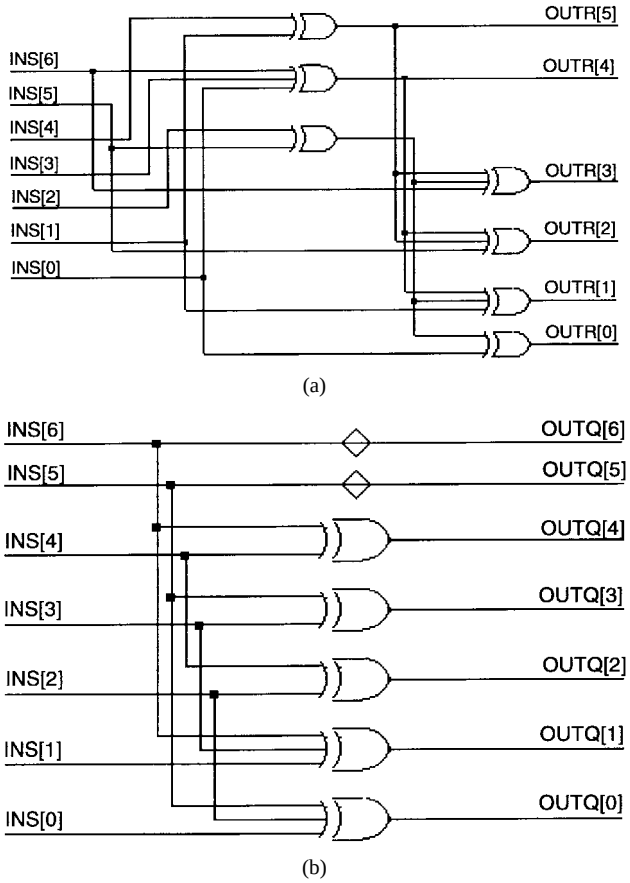

Fig. 4. Lookahead circuit when the divisor polynomial is: $M(x) = x^6 + x^4 + x^2 + x + 1$. Since $M(x)$ is one of the two minimal polynomials of (63, 51) DEC BCH code, where (63, 51) BCH code is now being used in AMPS (advanced mobile phone service), this can be used in (63, 51) BCH decoder for syndrome generation directly. (a) A circuit for lookahead of partial-remainder. (b) A circuit for lookahead of partial-quotient. However, this is not necessary for the syndrome generation in (63, 51) BCH decoder.

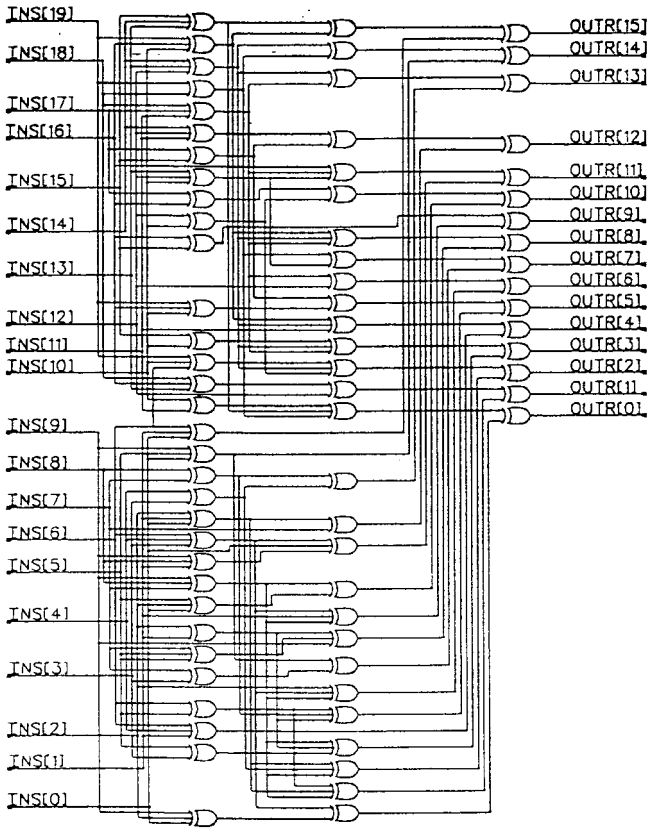

Fig. 5. A circuit for the lookahead of partial-remainder when the divisor polynomial is: $M(x) = x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}$. Where $M(x)$ is the generator polynomial of two symbol correction (15, 11) RS code.

Let us assume that we perform successive elimination of all of the elements in the first group using the same method as the conventional division process. During this time, the division process for other groups is suspended. The quotient and remainder polynomials when all of the elements in the first group are eliminated are regarded as the partial-quotient and partial-remainder, respectively, which are obtained from the partial-division. Note that the degree of the partial-remainder from each group is the same as that of the next group.

Because addition in finite field GF(2^m) is modular-2 based, adding two polynomials with the same degree does not produce carry. The produced partial-remainder can then be added to the next group, then forming a new group with the same degree as the next group. Eliminating this new group and forming a new group by adding the produced partial-remainder to the next group is repeated until the last group is reached. The sum of the last partial-remainder produced and the last group is the overall remainder. The overall quotient can easily be obtained by summing all the partial-quotients produced from each group.

Fig. 2 shows one simple example of this division process where all the coefficients are in GF(2). Both the first (P_1) and the second group (P_0) have seven elements and the last group (P_{-1}) has six. After partial-division is performed to eliminate all of the elements in the first group (P_1), a partial-remainder (R_1) and a partial-quotient (Q_1) are produced. The partial-remainder (R_1) is then added to the second group (P_0) forming a new group (P_0'). One more partial-division for the newly formed group (P_0') produces the next partial-quotient (Q_0) and partial-remainder (R_0). The overall remainder is then obtained by simply adding the lastly produced partial-remainder (R_0) and the last group (P_{-1}).

In this new division process based on partial-division, as it is, no gain in latency delays is obtained since successive elimination is performed to all the elements in the groups. However, if lookahead technique is employed, we can make the latency delays be the number of groups ($q + 2$). We will show that the exact partial-quotient and partial-remainder can directly be obtained by looking only at the group elements in Section II-C. As a result, multiplication between constants and divisor polynomials is completely eliminated. The next subsection describes the mathematical validity of new division algorithm based on partial-division.

B. The Mathematical Validity of New Division Algorithm Based on Partial-Division

Our division process starts from the definition of $P(x)$ as the long arbitrary dividend polynomial of degree n and $M(x)$ as the fixed divisor polynomial of degree k, as follows:

$$P(x) = p_n x^n + p_{n-1}x^{n-1} + p_{n-2}x^{n-2} + p_{n-3}x^{n-3} + \cdots + p_3x^3 + p_2x^2 + p_1x + p_0 \quad (1)$$

$$M(x) = m_k x^k + m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_3x^3 + m_2x^2 + m_1x + m_0. \quad (2)$$

If we define q as the maximum number that satisfies $n \geq q(k + 1) + k$, then the elements in the dividend polynomial can be grouped

into $q+2$ nonoverlapping groups as follows:

$$P(x) = P_q(x)x^{q(k+1)} + P_{q-1}(x)x^{(q-1)(k+1)} + \cdots + P_1(x)x^{(k+1)} + P_0(x)x + P_{-1}(x). \quad (3)$$

All of the groups $P_j(x)$ for $q \geq j \geq 0$ can be expressed as

$$P_j(x) = \Big(p_{(j+1)(k+1)+k-1}x^k + p_{(j+1)(k+1)+k-2}x^{k-1} + p_{(j+1)(k+1)+k-3}x^{k-2} + \cdots + p_{j(k+1)+k+2}x^2 + p_{j(k+1)+k+1}x + p_{j(k+1)+k}\Big)x^k \quad (4)$$

and the last group $P_{-1}(x)$ is expressed as

$$P_{-1}(x) = p_{k-1}x^{k-1} + p_{k-2}x^{k-2} + \cdots + p_3x^3 + p_2x^2 + p_1x + p_0. \quad (5)$$

Note that the number of elements in $P_j(x)$ for $q \geq j \geq 0$ is $k+1$ and the degree of $P_{-1}(x)$ is 1 less than that of divisor polynomial. To proceed with the dividing process $P(x)$ by $M(x)$, let us define $Q_q(x)$ and $R_q(x)$ as the quotient and the remainder from dividing $P_q(x)$ by $M(x)$. Then $P_q(x)$ becomes

$$P_q(x) = M(x)Q_q(x) + R_q(x), \quad \text{where degree of } R_q(x) \leq k-1. \quad (6)$$

Using (6), the first term in (3) can be expressed as

$$P_q(x)x^{q(k+1)} = M(x)Q_q(x)x^{q(k+1)} + x^{k+1}R_q(x)x^{(q-1)(k+1)}. \quad (7)$$

Note that in (7), $Q_q(x)x^{q(k+1)}$ and $x^{k+1}R_q(x)x^{(q-1)(k+1)}$ can be interpreted as the partial-quotient and the partial-remainder when all of the elements in the first group are eliminated and the division process for the next group is suspended. Inserting (7) into (3) leads to

$$P(x) = M(x)Q_q(x)x^{q(k+1)} + \Big(x^{k+1}R_q(x) + P_{q-1}(x)\Big)x^{(q-1)(k+1)} + P_{q-2}(x)x^{(q-2)(k+1)} + \cdots + P_2(x)x^{2k+2} + P_1(x)x^{k+1} + P_0(x) + P_{-1}(x). \quad (8)$$

Now, we define $P'_{q-1}(x)$ as a new group

$$P'_{q-1}(x) = xR_q(x)x^k + P_{q-1}(x). \quad (9)$$

Here, we notice the degree of the first term in (9) is the same as the degree of the second term. Then (8) becomes

$$P(x) = M(x)Q_q(x)x^{q(k+1)} + P'_{q-1}(x)x^{(q-1)(k+1)} + P_{q-2}(x)x^{(q-2)(k+1)} + \cdots + P_2(x)x^{2(k+1)} + P_1(x)x^{k+1} + P_0(x) + P_{-1}(x). \quad (10)$$

Similarly, if one defines Q_{q-1}, $R_{q-1}(x)$ as the quotient and remainder from $P'_{q-1}(x)/M(x)$, then

$$P'_{q-1}(x) = M(x)Q_{q-1}(x) + R_{q-1}(x), \quad \text{where degree of } R_{q-1}(x) \leq k-1. \quad (11)$$

Inserting (11) into (10) again gives

$$P(x) = M(x)Q_q(x)x^{q(k+1)} + M(x)Q_{q-1}(x)x^{(q-1)(k+1)} + \Big(x^{k+1}R_{q-1}(x) + P_{q-1}(x)\Big)x^{(q-2)(k+1)} + \cdots + P_1(x)x^{k+1} + P_0(x) + P_{-1}(x). \quad (12)$$

Repeating the same procedure finally leads to

$$P(x) = M(x)Q_q(x)x^{q(k+1)} + M(x)Q_{q-1}(x)x^{(q-1)(k+1)} + M(x)Q_{q-2}(x)x^{(q-2)(k+1)} + \cdots + M(x)Q_1(x)x^{k+1} + M(x)Q_0(x) + R_0(x) + P_{-1}(x). \quad (13)$$

If one lets $Q(x)$ and $R(x)$ be the overall quotient and the remainder resulting from $P(x)/M(x)$, then from (13) we obtain

$$Q(x) = \sum_{i=0}^{q} Q_i(x)x^{i(k+1)} \quad (14)$$

and

$$R(x) = R_0(x) + P_{-1}(x). \quad (15)$$

Note here that the degree of both $R_0(x)$ and $P_{-1}(x)$ is less than or equal to $(k-1)$, thus the degree of resulting $R(x)$ is also less than or equal to $(k-1)$ as expected.

Fig. 3 shows this division algorithm schematically for hardware implementation. $P_j(x)$ for $-1 \leq j \leq q$ is the group in the dividend polynomial which is obtained by the systematic rule mentioned before. $Q_q(x)$ and $R_q(x)$ are the quotient and the remainder resulting from $P_q(x)/M(x)$, respectively. $P'_j(x)$ is the sum of partial-remainder resulting from previous group and $P_j(x)$. $Q_j(x)$ and $R_j(x)$ are the quotient and the remainder produced from $P'_j(x)/M(x)$ for $j = (q-1)$ down to zero. The overall quotient is the weighted sum of all the partial-quotients $Q_j(x)$ for $0 \leq j \leq q$ and the overall remainder is the simple sum of $R_0(x)$ and last group $P_{-1}(x)$.

C. Efficient Strategy for Lookahead of Partial-Remainder

The issue now is how to find $Q_j(x)$ and $R_j(x)$ for $j = q$ down to zero in a simple and efficient way. We introduce $S(x)$, which has the same format as $P'_j(x)$ for $0 \leq j \leq q-1$

$$S(x) = \Big(s_kx^k + s_{k-1}x^{k-1} + s_{k-2}x^{k-2} + \cdots + s_3x^3 + s_2x^2 + s_1x + s_0\Big)x^k. \quad (16)$$

Let us define $Q_s(x)$ and $R_s(x)$ to be the quotient and the remainder, respectively, by dividing $S(x)$ by $M(x)$. If we store $Q_s(x)$ and $R_s(x)$ from $S(x)/M(x)$ for every possible combination of s_l for $0 \leq l \leq k$ into a table, then each of $Q_j(x)$ and $R_j(x)$ for $0 \leq j \leq q$ can be obtained by referencing this table. To do this, in binary field, $(k+1)$ bits $Q_s(x)$ and k bits $R_s(x)$ have to be stored for every possible combination of s_l for $0 \leq l \leq k$. However, since there are 2^{k+1} possible combinations of s_l for $0 \leq l \leq k$, even for the moderate values of k, storing all $Q_s(x)$ and $R_s(x)$ into one table is a formidable task. However, exploiting the linearity in galois field arithmetic [12], the number of necessary $Q_s(x)$ and $R_s(x)$ can be drastically reduced from 2^{k+1} to $(k+1)$ in binary field. That is, $S(x)/M(x)$ is the same as the linear sum of each element in $S(x)$ divided by $M(x)$

$$S(x)/M(x) = s_kx^{2k}\Big/M(x) + s_{k-1}x^{2k-1}\Big/M(x) + \cdots + s_2x^{k+2}\Big/M(x) + s_1x^{k+1}\Big/M(x) + s_0x^k\Big/M(x). \quad (17)$$

If we define $Q_{s_l}(x)$ and $R_{s_l}(x)$ as the quotient and the remainder from dividing s_lx^{l+k} by $M(x)$ for $0 \leq l \leq k$, respectively, then s_lx^{l+k} can be expressed as follows:

$$s_lx^{l+k} = Q_{s_l}(x)M(x) + R_{s_l}(x), \quad \text{where degree of } R_{s_l}(x) \leq k-1. \quad (18)$$

TABLE II
ALL THE NECESSARY INFORMATION FOR LOOKAHEAD OF PARTIAL-REMAINDER WHEN THE DIVISOR POLYNOMIAL IS: $M(x) = x^4+\alpha^{13}x^3+\alpha^6x^2+\alpha^3x+\alpha^{10}$. HERE, ALL THE COEFFICIENTS ARE IN GF(2^4)

Input	*S(x)/x⁴* INS[19] ~ INS[0]	
	1000 0000 0000 0000 0000	1100 0001 1011 0101
	0**1**00 0000 0000 0000 0000	0110 1001 1100 1011
	00**1**0 0000 0000 0000 0000	0011 1101 0110 1100
	000**1** 0000 0000 0000 0000	1000 1111 0011 0110
	0000 **1**000 0000 0000 0000	1101 0010 0010 1001
	0000 0**1**00 0000 0000 0000	1111 0001 0001 1101
	0000 00**1**0 0000 0000 0000	1110 1001 1001 1111
	0000 000**1** 0000 0000 0000	0111 1101 1101 1110
	0000 0000 **1**000 0000 0000	0011 1001 0101 1001
	0000 0000 0**1**00 0000 0000	1000 1101 1011 1101
	0000 0000 00**1**0 0000 0000	0100 1111 1100 1111
	0000 0000 000**1** 0000 0000	0010 1110 0110 1110
	0000 0000 0000 **1**000 0000	0011 0111 1110 1110
	0000 0000 0000 0**1**00 0000	1000 1010 0111 0111
	0000 0000 0000 00**1**0 0000	0100 0101 1010 1010
	0000 0000 0000 000**1** 0000	0010 1011 0101 0101
	0000 0000 0000 0000 **1**000	0010 1010 1100 1101
	0000 0000 0000 0000 0**1**00	0001 0101 0110 1111
	0000 0000 0000 0000 00**1**0	1001 1011 0011 1110
	0000 0000 0000 0000 000**1**	1101 1100 1000 0111
Output of S(x)/M(x)		OUTR[15] ~ OUTR[0] *Rs(x)*

Using (18), the quotient $Q_s(x)$ and the remainder $R_s(x)$ resulting from $S(x)/M(x)$ become

$$Q_s(x) = Q_{s_k}(x) + Q_{s_{k-1}}(x) + \cdots + Q_{s_2}(x) + Q_{s_1}(x) + Q_{s_0}(x) \tag{19}$$

$$R_s(x) = R_{s_k}(x) + R_{s_{k-1}}(x) + \cdots + R_{s_2}(x) + R_{s_1}(x) + R_{s_0}(x). \tag{20}$$

As shown in (19) and (20), if we know the $(k+1)$ number of $Q_{s_l}(x)$ and $R_{s_l}(x)$ for $0 \le l \le k$ resulting from $s_l x^{l+k}/M(x)$, $Q_s(x)$, and $R_s(x)$ can be obtained as a linear sum of $Q_{s_l}(x)$ and $R_{s_l}(x)$ for $0 \le l \le k$, respectively.

In a nonbinary galois field GF(2^t), since one symbol consists of t sequence of bit, symbol s_l in (16) for $0 \le l \le k$ can be expressed as t sequence of binary bit as $(s_{l,t-1}, s_{l,t-2}, \cdots, s_{l,1}, s_{l,0})$. The linearity is also satisfied in every bit in t tuple [12] so that (19) and (20) can be extended to symbol elements in a straightforward way. Therefore, if we know $(k+1)t$ number of $Q_{s_{l,j}}(x)$ and $R_{s_{l,j}}(x)$ for $0 \le l \le k$ and $0 \le j \le t-1$, $Q_s(x)$ and $R_s(x)$ can be obtained as a linear sum of $Q_{s_{l,j}}(x)$ and $R_{s_{l,j}}(x)$ for $0 \le l \le k$ and $0 \le j \le t-1$, respectively.

We take two specific examples for lookahead circuits to help understand the above generalized theorem.

Example 1: When the divisor polynomial is $M(x) = x^6 + x^4 + x^2 + x + 1$.

In this case, since the degree of $S(x)$ is 6, (16) can be expressed as follows:

$$S(x) = (s_6x^6 + s_5x^5 + s_4x^4 + s_3x^3 + s_2x^2 + s_1x + s_0)x^6. \tag{21}$$

Based on (16)–(20), all of the necessary information to form the lookahead circuits can be listed as shown in Table I. By superposing the result in Table I column by column, we can obtain the following logic expression for the partial-remainder: $r_5 = s_4 + s_1$, $r_4 = s_6+s_3+s_0$, $r_3 = s_6+s_5+s_4+s_2+s_1$, $r_2 = s_6+s_5+s_4+s_3+s_1+s_0$, $r_1 = s_6 + s_5 + s_3 + s_2 + s_1 + s_0$, $r_0 = s_5 + s_2 + s_0$. Each s_j represents the coefficients of $S(x)$ in (21) and each r_j means the coefficients of $R_s(x)$, which are expressed as $R_s(x) = r_5x^5 + r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0$. One can see the number of input bits required to generate each bit of partial-remainder is not 7 but $6 \sim 2$. Moreover, the actual number of EXOR's to produce each bit forming a partial-remainder may be substantially less because of the redundancy. After the gate minimization process, an optimized circuit for LAPR shown in Fig. 4(a) is obtained. A logic expression for the partial-quotient would be $q_6 = s_6$, $q_5 = s_5$, $q_4 = s_6 + s_4$, $q_3 = s_5 + s_3$, $q_2 = s_4 + s_2$, $q_1 = s_6 + s_3 + s_1$, $q_0 = s_5 + s_2 + s_0$ where each q_j means the coefficients of $Q_s(x)$ which are expressed as $Q_s(x) = q_6x^6+q_5x^5+q_4x^4+q_3x^3+q_2x^2+q_1x+q_0$. Fig. 4(b) shows an optimized circuit for LAPQ, but the circuit for LAPQ is inherently less complex than for LAPR. Those lookahead circuits shown in Fig. 4 can generate 6 bits partial-remainder and 7 bits partial-quotient, respectively, from 22^7 possible combinations of seven input bits.

Example 2: When the divisor polynomial is $M(x) = x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}$.

Here, all of the coefficients in the divisor polynomial are in GF(2^4) generated from a primitive polynomial $p(\alpha) = \alpha^4 + \alpha + 1$ [12]. The degree of the divisor polynomial is 4, hence 4 symbol partial-remainder is obtained as a combination of 5 input symbols. In other words, since one symbol consists of 4 bits, 16 bits partial-remainder have to be obtained as a combination of 20 input bits. Because the linearity is also satisfied in symbol elements, all of the necessary information to form the lookahead circuits can be listed as shown in Table II. Readers can easily derive the logic expression for the partial-remainder the same way as in Example 1. After performing gate minimization, an optimized circuit for LAPR is obtained with reduced gate count as shown in Fig. 5. By balancing the addition of partial remainder in a tree structure, critical path does not exceed three EXOR's. Those lookahead circuits can generate 16 bits partial-remainder from 2^{20} possible combinations of 20 input bits.

III. DIVISION ARCHITECTURE BASED ON LAPR

By exploiting the regularity and feedforward nature of our algorithm, we make it fully pipelined, leading its throughput to be one remainder and one quotient per clock cycle regardless of the degree of dividend polynomial. Fig. 6 shows the block diagram of the pipelined division architecture based on LAPR. Here, the block `FIRST` is the register for the first group $P_q(x)$. The q identical blocks `INT` are intermediate group registers, which form new intermediate groups $P'_j(x)$ by adding the partial-remainder from the previous group and the input $P_j(x)$ as shown in (9) mentioned before. The block `LAST` is the remainder register. Simply adding the partial-remainder from $P'_0(x)$ and $P_{-1}(x)$ forms the overall remainder as shown in (15). There are $(q+1)$ identical blocks named `LOOK-AHEADR` and `LOOK-AHEADQ` that generate the partial-remainder and the partial-quotient, respectively, which are obtained by dividing $P'_j(x)$ by $M(x)$, as discussed in the previous section. Overall quotient can be easily obtained by adding the all of the partial-quotients. Fig. 7 shows the operation diagram of the pipelined architecture. Each group in the dividend polynomial is inserted one by one sequentially to its own specific stage from first to last. Each group in the next dividend polynomial can be inserted to its own specific stage as soon as the group of the present dividend polynomial of that stage is processed. After $(q+2)$ cycles, where $(q+2)$ is the number of groups in the dividend polynomial, one remainder and one quotient are produced so that the latency delays of this pipelined architecture are $(q+2)$. After $(q+2)$ cycles, all of the blocks in Fig. 6 operate continuously and simultaneously, leading throughput to be one remainder and one quotient per clock cycle, regardless of the degree of the dividend polynomial.

An area-efficient sequential architecture based on LAPR is shown in Fig. 8. It consists of the block `INT` as the group register and the block `LOOK-AHEADR`, `LOOK-AHEADQ` which generates partial-remainder and partial-quotient from each group, respectively. The blocks used in the pipelined architecture can also be used in this sequential architecture without any modification. Because it uses a single cell recursively to perform successive steps of the division

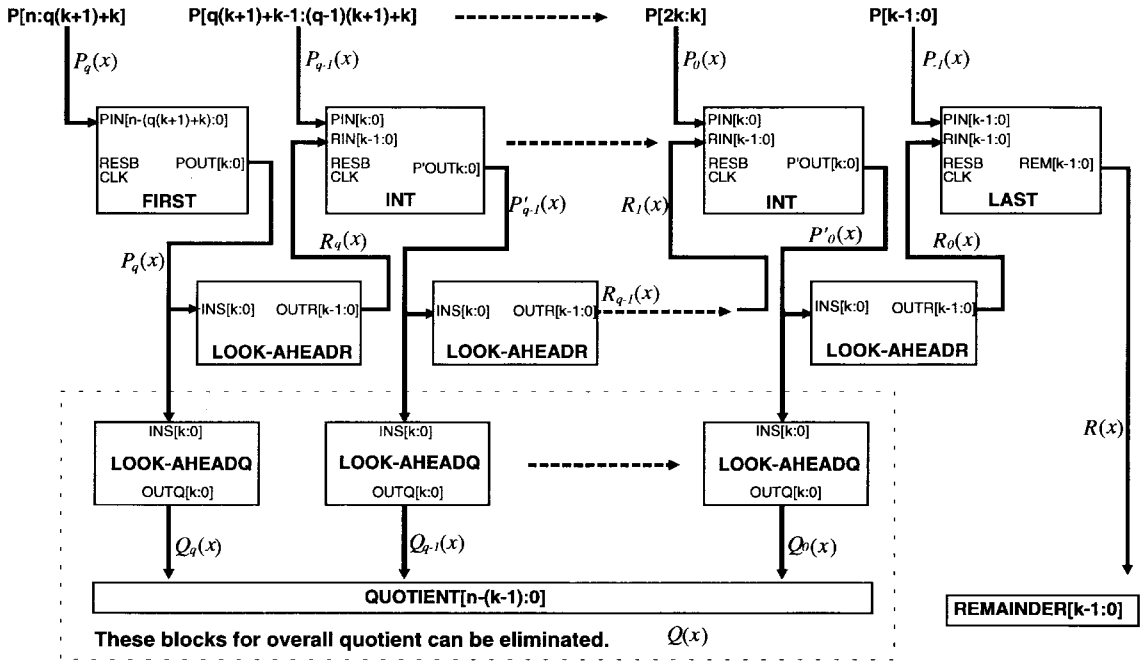

Fig. 6. The pipelined division architecture based on LAPR.

TABLE III
THE COMPARISONS OF PERFORMANCE

Architecture (clock = 30MHz)	Throughput (div/sec)	Delays (cycle)	Needed clock rates (10M div/sec)	Power reduction factor (100K div/sec)	Multipliers used
1) In case of 2 error correction (63,51) BCH code : Error correction code of AMPS.					
Divisor polynomial : $M(x) = x^{12}+x^{10}+x^8+x^5+x^4+x^3+1$					
Degree of divisor : 6, Degree of dividend : 62, The finite field used : GF(2)					
Pipelined(LAPR)	**30M**	**1**	**10 MHz**	**300**	-
Sequential(LAPR)	**5M**	**6**	**60 MHz**	**50**	-
Serial(LFSR)	476K	63	630 MHz	4.76	-
Serial(LFSR,PIPELINE)	238K	126	1.26 GHz	2.38	-
2) In case of 2 symbol correction (32,28) RS code, encoder : Error correction code for Compact Disk[13]					
Divisor polynomial : $M(x) = (x+\alpha^0)(x+\alpha^1)(x+\alpha^2)(x+\alpha^3)$					
Degree of divisor : 4 , Degree of dividend : 31, The finite field used : GF(2^8)					
Pipelined(LAPR)	**30M**	**1**	**10 MHz**	**300**	**Not used**
Sequential(LAPR)	**4.3M**	**7**	**70 MHz**	**43**	**Not used**
Serial(LFSR)	937K	32	320 MHz	9.37	ROM
Serial(LFSR,PIPELINE)	468K	64	640 MHz	4.68	ROM
Bit Serial(SR)	119K	256	2.56 GHz	1.19	Berlekamp

process based on LAPR, its throughput is one remainder and one quotient per $(q+2)$ clock cycles. Although this is slower than the pipelined one shown in Fig. 6, it is still faster than any other division architecture reported before [1], [3], [9].

It should be noted here that in many linear block coding applications such as BCH and RS code, the quotient is not needed. Since our division algorithm based on LAPR does not require partial-quotients for the division process, all of the blocks used to produce the overall quotient in Figs. 6 and 8 can completely be eliminated, unless the application requires the quotient explicitly.

IV. THE COMPARISON OF PERFORMANCE

In this section, we compare the proposed pipelined and sequential architectures based on LAPR with currently available division architectures which were mentioned in Section I, with regard to their throughput, delays and power reduction factor.

Table III shows the comparison result for the two different applications:

1) (63, 51) BCH encoder in GF (2);
2) (32, 28) RS encoder in GF(2^8).

The (63, 51) BCH code and (32, 28) RS code in GF(2^8) are now being used in AMPS cellular phone and CD error correction coding, respectively [13]. The throughput when using the same clock frequency is listed in the second column. The speed enhancement of pipelined architecture based on LAPR over the conventional one using LFSR are 63, 32. The corresponding enhancements of the sequential architecture based on LAPR are 10.5, 4.6, respectively. The clock frequency needed to obtain identical throughput is calculated from the original clock frequency (30 MHz) and the required throughput (10 M div/s). It indicates that other architectures except pipelined and sequential ones based on LAPR cannot meet the required throughput rate without a special clocking scheme, since clock speeds for "low-cost-silicon" implementation are limited to 100–200 MHz.

Noting the generally accepted equation $P = cv^2f$, the power consumption in a CMOS digital system is proportional to the clock

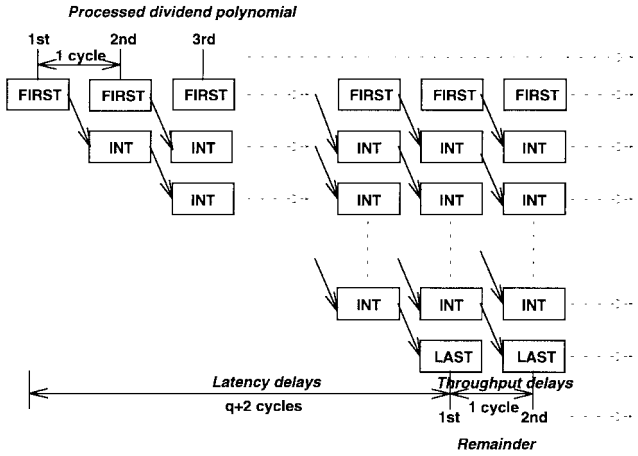

Fig. 7. The schematic operation diagram of pipelined division architecture shown in Fig. 6. Here, for simple illustration, we assumed each lookahead circuit is included in its group register block.

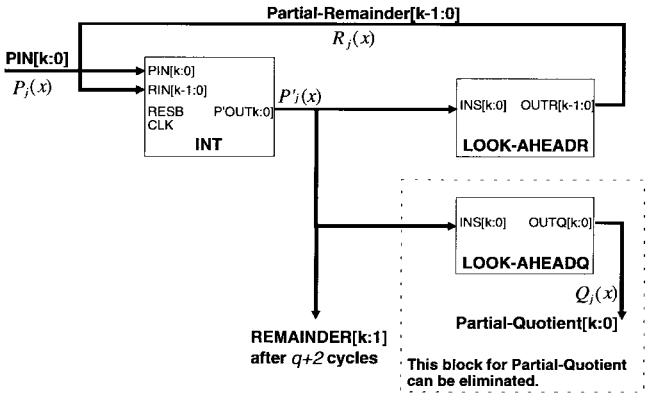

Fig. 8. An area efficient sequential division architecture based on LAPR.

frequency [11]. Hence, the high throughput rate implies that the main clock speed can be reduced by some factor depending on the required modest throughput. In other words, the power consumption can be approximately reduced by the same factor by reducing the clock rate. To show how much power reduction is possible, the power reduction factor when the same amount of throughput (100 K div/s) is expected is shown in the fifth column. It is the ratio of the original clock frequency to the needed clock frequency to obtain the required throughput. Although the total capacitance also has to be considered, knowing that the speed enhancements are orders of magnitude without much increasing area, the net power dissipation will not be increased much by total capacitance. Power reduction that can be obtained by voltage scale-down is not considered here. However, it should be noted that as the needed clock rate is lower, the timing constraint becomes less critical, which allows the supply voltage to be scaled down appreciably, leading to additional drastic power reduction [11].

V. Conclusions

We proposed a new division algorithm based on LAPR, which is very efficient and can be fully pipelined. The efficiency and pipelining obtained from partial-division on group basis and lookahead technique, exploiting the linearity in finite field arithmetic, make our division architecture produce highly increased throughput per unit time without increasing area much.

When division algorithm based on LAPR is applied to implement (63, 51) BCH encoder, which has an arbitrary dividend polynomial with degree 62 and the fixed divisor polynomial (generator polynomial) with degree 12, the pipelined architecture and the sequential one based on LAPR are 63 and 10.5 times faster than the conventional serial one using LFSR. The corresponding speed enhancements of the (32, 28) RS encoder in GF(2^8) are 32, 4.6, respectively. Additionally, this high speed obtained from division algorithm base on LAPR can be traded for low power consumption by reducing clock rate and proper scale down of supply voltage.

Since the proposed division algorithm based on LAPR is efficient, regular, and easily expandable, it can be used directly and efficiently as a decoder building block in VLSI implementation of various coding systems, such as very high-speed and/or very low-power BCH/RS encoders and syndrome generators.

References

[1] G. Seroussi, "A systolic Reed–Solomon encoder," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1217–1220, July 1991.
[2] H. M. Shao and T. K. Truong, "A VLSI design of a pipeline Reed–Solomon decoder," *IEEE Trans. Comput.*, vol. C-34, pp. 393–403, May 1985.
[3] H. M. Shao and I. S. Reed, "On the VLSI design of a pipeline Reed–Solomon decoder using systolic arrays," *IEEE Trans. Comput.*, vol. 37, pp. 1273–1279, Oct. 1988.
[4] A. Yamagishi and H. Imai, "A construction method for decoders of BCH codes Using ROM's," *Trans. IECE Japan*, vol. J-63D, pp. 1034–1041, Dec. 1980.
[5] H. Okano and H. Imai, "A construction method of high-speed decoders using ROM's for Bose–Chaudhuri–Hocquenghem and Reed–Solomon codes," *IEEE Trans. Comput.*, vol. C-36, pp. 1165–1171, Oct. 1987.
[6] J. L. Massey and J. K. Omura, "Computational method and apparatus for fine field arithmetic," U.S. Patent application pending.
[7] C. C. Wang *et al.*, "VLSI architecture for computing multiplications and inverses in GF(2^m)," *IEEE Trans. Comput.*, vol. C-34, pp. 393–403, Aug. 1985.
[8] A. V. Curiger, H. Bonnenberg, and H. Kaeslin, "Regular VLSI architectures for multiplication modulo ($2^n + 1$)," *IEEE J. Solid-State Circuits*, vol. 26, pp. 990–994, July 1991.
[9] I.-S. Hsu and I. S. Reed, "The VLSI implementation of a Reed–Solomon encoder using Berlekamp's bit-serial multiplier algorithm, *IEEE Trans. Comput.*, vol. C-33, no. 10, pp. 906–911, Oct. 1984.
[10] M. Lowy, "Low power spread spectrum code generator based on parallel shift register implementation," in *Proc. '94 IEEE Symp. Low Power Electronics*, Aug. 1994, pp. 22–23.
[11] A. D. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, pp. 473–483, Apr. 1992.
[12] R. E. Blahut, *Theory and Practice of Error Control Codes.* Reading, MA: Addison-Wesley, 1984.
[13] O. Kawamae, T. Takeuchi, and I. Kimura, "A high speed signal processing for quadruple speed CD-ROM," *IEEE Trans. Consumer Electron.*, vol. 40, pp. 679–685, Aug. 1994.