(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

PCT

(43) International Publication Date
11 June 2009 (11.06.2009)

(10) International Publication Number
WO 2009/072105 A2

(51) International Patent Classification:
G06F 12/02 (2006.01)

(21) International Application Number:
PCT/IL2008/001241

(22) International Filing Date:
17 September 2008 (17.09.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/996,782	5 December 2007 (05.12.2007)	US
60/996,948	12 December 2007 (12.12.2007)	US
61/006,805	31 January 2008 (31.01.2008)	US
61/064,853	31 March 2008 (31.03.2008)	US
61/071,465	30 April 2008 (30.04.2008)	US
61/071,468	30 April 2008 (30.04.2008)	US
61/071,487	1 May 2008 (01.05.2008)	US
61/129,608	8 July 2008 (08.07.2008)	US

(71) Applicant *(for all designated States except US)*: **DENSBITS TECHNOLOGIES LTD.** [IL/IL]; Building 30, FIr 3, Matam Industrial Park, P.O.Box 15111, 31015 Haifa (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **WEINGARTEN, Hanan** [IL/IL]; Hayarden 2, 46377 Herzliya (IL). **STERIN, Eli** [IL/IL]; 16/1 Hahermon Street, 20692 Yoqneam-ilit (IL). **KANTER, Ofir, Avraham** [IL/TL]; 10/1 HaHermon Street, 20692 Yoqneam-ilit (IL). **KATZ, Michael** [IL/IL]; 75 Moriah Ave., 34616 Haifa (IL).

(74) Agent: **PEARL COHEN ZEDEK LATZER;** 5 Shenkar Street, P.O. Box 12704, Herzlia 46733 (IL).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

(54) Title: A LOW POWER CHIEN-SEARCH BASED BCH/RS DECODING SYSTEM FOR FLASH MEMORY, MOBILE COMMUNICATIONS DEVICES AND OTHER APPLICATIONS

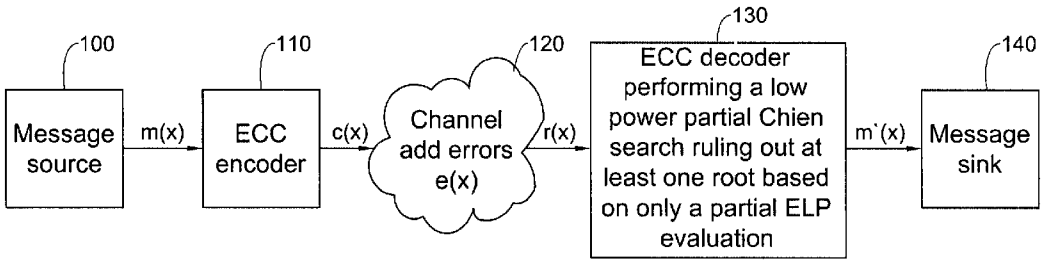


FIG. 1A

(57) Abstract: A low power Chien searching method employing Chien search circuitry comprising at least two hardware components that compute at least two corresponding bits comprising a Chien search output, the method comprising activating only a subset of the hardware components thereby to compute only a subset of the bits of the Chien search output; and activating hardware components other than those in the subset of hardware components, to compute additional bits of the Chien search output other than the bits in the subset of bits, only if a criterion on the subset of the bits of the Chien search output is satisfied.

WO 2009/072105 A2

A LOW POWER CHIEN-SEARCH BASED BCH/RS DECODING SYSTEM FOR FLASH MEMORY, MOBILE COMMUNICATIONS DEVICES AND OTHER APPLICATIONS

REFERENCE TO CO-PENDING APPLICATIONS

Priority is claimed from the following co-pending applications: US Provisional Application No. 60/996,948, filed December 12, 2007 and entitled "Low Power BCH/RS Decoding: a Low Power Chien-Search Implementation", US Provisional Application No. 61/071,487, filed May 1, 2008 and entitled "Chien-Search System Employing a Clock-Gating Scheme to Save Power for Error Correction Decoder and other Applications", US Provisional Application No. 61/071,468, filed April 30, 2008 and entitled "A Low Power Chien-Search Based BCH/RS Recoding System for Flash Memory, Mobile Communications Devices and Other Applications", US Provisional Application No. 60/996,782, filed December 5, 2007 and entitled "Systems and Methods for Using a Training Sequence in Flash Memory", US Provisional Application No. 61/064,853, filed March 31, 2008 and entitled "Flash Memory Device with Physical Cell Value Deterioration Accommodation and Methods Useful in Conjunction Therewith", US Provisional Application No. 61/006,805, filed January 31, 2008 and entitled "A Method for Extending the Life of Flash Devices", US Provisional Application No. 61/071,465, filed April 30, 2008 and entitled "Systems and Methods for Temporarily Retiring Memory Portions" and US Provisional Application No. 61/129,608, filed July 8, 2008 and entitled "A Method for Acquiring and Tracking Detection Thresholds in Flash Devices".

Other co-pending applications include: US Provisional Application No. 60/960,207, filed September 20, 2007 and entitled "Systems and Methods for Coupling Detection in Flash Memory", US Provisional Application No. 61/071,467, filed April 30, 2008 and entitled "Improved Systems and Methods for Determining Logical Values of Coupled Flash Memory Cells", US Provisional Application No. 60/960,943, filed October 22, 2007 and entitled "Systems and methods to reduce errors in Solid State Disks and Large Flash Devices" and US Provisional Application No. 61/071,469, filed April 30, 2008 and entitled "Systems and Methods for Averaging Error Rates in Non-

Volatile Devices and Storage Systems", US Provisional Application No. 60/996,027, filed October 25, 2007 and entitled "Systems and Methods for Coping with Variable Bit Error Rates in Flash Devices", US Provisional Application No. 61/071,466, filed April 30, 2008 and entitled "Systems and Methods for Multiple Coding Rates in Flash Devices", US Provisional Application No. 61/006,120, filed December 19, 2007 and entitled "Systems and Methods for Coping with Multi Stage Decoding in Flash Devices", US Provisional Application No. 61/071 ,464, filed April 30, 2008 and entitled "A Decoder Operative to Effect A Plurality of Decoding Stages Upon Flash Memory Data and Methods Useful in Conjunction Therewith". US Provisional Application No. 61/006,385, filed January 10, 2008 and entitled "A System for Error Correction Encoder and Decoder Using the Lee Metric and Adapted to Work on Multi-Level Physical Media", US Provisional Application No. 61/064,995, filed April 8, 2008 and entitled "Systems and Methods for Error Correction and Decoding on Multi-Level Physical Media", US Provisional Application No. 61/006,806, filed January 31, 2008 and entitled "Systems and Methods for using a Erasure Coding in Flash memory", US Provisional Application No. 61/071,486, filed May 1, 2008 and entitled "Systems and Methods for Handling Immediate Data Errors in Flash Memory", US Provisional Application No. 61/006,078, filed December 18, 2007 and entitled "Systems and Methods for Multi Rate Coding in Multi Level Flash Devices", US Provisional Application No. 61/064,923, filed April 30, 2008 and entitled "Apparatus For Coding At A Plurality Of Rates In Multi-Level Flash Memory Systems, And Methods Useful In Conjunction Therewith", US Provisional Application No. 61/064,760, filed March 25, 2008 and entitled "Hardware efficient implementation of rounding in fixed-point arithmetic", US Provisional Application No. 61/071,404, filed April 28, 2008 and entitled "Apparatus and Methods for Hardware-Efficient Unbiased Rounding". US Provisional Application No. 61/136,234, filed August 20, 2008 and entitled "A Method Of Reprogramming A Non-Volatile Memory Device Without Performing An Erase Operation", US Provisional Application No. 61/129,414, filed June 25, 2008 and entitled "Improved Programming Speed in Flash Devices Using Adaptive Programming", and several other co-pending patent applications being filed concurrently (same day).

FIELD OF THE INVENTION

The present invention relates generally to low power error correction systems and more particularly to Chien search apparatus.

BACKGROUND OF THE INVENTION

The term "Chien search" is used herein to refer to any typically recursive method or apparatus for determining roots of polynomials defined over a finite field. The term is also used herein to refer to any method or apparatus used for finding the roots of error-locator polynomials encountered in decoding, e.g. Reed-Solomon codes and BCH codes in various applications such as but not limited to flash memory and other data storage applications, and data communications applications.

According to Wikipedia, in conventional Chien searches:

"We denote the polynomial (over the finite field GF(q)) whose roots we wish to determine as (formula I): $\Lambda(x) = A0 + A_1 x + X_2 X^2 H\text{-} \cdots \text{-h } XtX^t$

Conceptually, we may evaluate $\Lambda(\beta)$ for each non-zero Pin GF(q). Those resulting in 0 are roots of the polynomial.

The Chien search is based on two observations:

Each non-zero Hmay be expressed as α^{i_β} for some i_β, where Q' is the primitive element of GF(q). Therefore, the powers α^i for $0 \leq i < (N \sim 1)$ cover the entire field (excluding the zero element).

The following relationship exists (formula II):

$$\begin{aligned} A(\alpha^i) &= A_0 + A_1(Cf^i) + X_2(a\psi + \cdots + \lambda t\{a?)^1 \\ &\triangleq 7{<}tf + 7i,i + 72;i + \cdots + 7_{t,i} \end{aligned}$$

$$\begin{aligned} \Lambda(\alpha^{i+1}) &= A_0 + A_1(\alpha^{i+1}) + \dot{A}_2'(\dot{Q}^{i+1})^2 + \ldots + \backslash t(a^{i+i} y \\ &= A_0 + A_1(QV + X_2(Ot)^2 \alpha^2 + \cdots + Xt[O^*)^* a^t \\ &= 7o,i + 7i,_* \alpha + 72,i\,\alpha^2 + \cdots + 7_{t,i}\,\alpha^t \\ &\triangleq 7o,i+i + 7i,t+i + 7'2,i+i + \cdots + 7'M+i \end{aligned}$$

$\gamma_{j.i+1} = \gamma_{j.i} \alpha^{j}$

In this way, we may start at i = Owith $\gamma_{j.0} = \lambda_j$, and iterate through each value of i up to $(N - 1)$ If at any stage the resultant summation is zero, i.e.

$$\sum_{j=0}^{t} \gamma_{j.i} = 0,$$

then $\Lambda(Q^{i})$ = $\bar{U}_a l_{so?\ so}$ α_{ii} $_{s\ a\ roo}$ t. J_n this way, we check every element in the field.

When implemented in hardware, this approach significantly reduces the complexity, as all multiplications consist of one variable and one constant, rather than two variables as in the brute-force approach."

A Chien search therefore may comprise the following steps:

a. Receive a polynomial, A(x) = $\lambda_0 + \lambda_1 X + X_2 X^2 + \cdots - \backslash - XiX^t$

defined over the finite field GF(q)) whose roots are to be determined, where the roots are the set of non-zero β in GF(q). for which $\Lambda(\beta) = 0$

Repeat the following steps b - d for all non-zero β in GF(q)

b. Express β as α^{i} for some $i\beta$, where α is the primitive element of GF(q).

c. Define each $\Lambda(Q^{i})$ as the sum of a set of terms $\{7i.i|0 \leq j \leq t\}$

d. Start at i = Owith $\gamma_{j.0}$ = Λ/, and iterate through each value of i up to i = $(N - 1)$ where the iteration comprises deriving successive sets of terms using (formulae III):

$$\gamma_{j.i+1} = \gamma_{j.i} \alpha^{i}$$

$$\sum_{j=0}^{t} lj.i = 0,$$

However, if at any stage the resultant summation is zero, i.e. stop and output that αi is a root.

Generally, any alpha^i for which the above error locator polynomial is zero, is termed a root. The above polynomial is encountered when decoding error correction

code using Reed-Solomon code or BCH code. The alpha's are all primitive elements in a finite field over which the above polynomial is defined. The index of the power of the root indicates locations of errors. In BCH, each error is a flipped bit. In Reed-Solomon, each error is a symbol in which at least one bit is wrong. In other words, if α^n is a root of the Error Locator Polynomial (ELP) then if binary BCH code is being used, an error has occurred in bit n of the data being read or received. If non-binary BCH code, or RS code, is used, the fact that α^n is a root of the Error Locator Polynomial (ELP) implies that an error has occurred in symbol n of the received or read data.

The state of the art is believed to be represented by the following prior art documents inter alia;

a. United States Patents 6,954,892; 6,990624; 7,1 13,968; Published US Application 2007245220.

b. **Error Correction Coding Mathematical Methods and Algorithms.** Todd K. Moon, A JOHN WILEY & SONS, INC., 2005.

c. **Introduction to** Coding **Theory,** Ron M. Roth, Cambridge University Press, 2006.

d. **Algebraic** Codes **for Data Transmission,** Richard E. Blahut. Cambridge University Press, 2003.

e. **Introduction to Error** Correcting Codes , Michael Purser, Artech House Inc, 1995.

f. "High throughput and low-power architectures for Reed Solomon Decoder", by Akash Kumar (a.kumar at tue.nl, Eindhoven University of Technology) and Sergei Sawitzki (Sergei. sawitzki at philips.com).

g. "Low power decoding of BCH codes", by Yuejian Wu, Nortel Networks, Ottawa, Ont, Canada, in Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on Circuits and Systems, published 23-26 May 2004, Volume: 2, page(s): II- 369-72 Vol.2.

h. "Small area parallel Chien search architectures for long BCH codes", Yanni Chen; Parhi, K.K. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on. Volume 12, Issue 5, May 2004 Page(s): 545 - 549. Digital Object Identifier 10.1 109/TVLSI.2004.826203 .

The following terms may be construed either in accordance with any definition thereof appearing in the prior art literature or in accordance with the specification, or as follows:

Block = a set of flash memory device cells which must, due to physical limitations of the flash memory device, be erased together. Also termed erase sector, erase block.

Cell: A component of flash memory that stores one bit of information (in single-level cell devices) or n bits of information (in a multi-level device having 2 exp n levels). Typically, each cell comprises a floating-gate transistor. n may or may not be an integer. "Multi-level" means that the physical levels in the cell are, to an acceptable level of certainty, statistically partitionable into multiple distinguishable regions, plus a region corresponding to zero, such that digital values each comprising multiple bits can be represented by the cell. In contrast, in single-level cells, the physical levels in the cell are assumed to be statistically partitionable into only two regions, one corresponding to zero and one other, non-zero region, such that only one bit can be represented by a single-level cell.

Charge level: the measured voltage of a cell which reflects its electric charge.

Cycling: Repeatedly writing new data into flash memory cells and repeatedly erasing the cells between each two writing operations.

Decision regions: Regions extending between adjacent decision levels, e.g. if decision levels are 0, 2 and 4 volts respectively, the decision regions are under 0 V, 0 V - 2 V, 2V - 4 V, and over 4 V.

Demapping: basic cell-level reading function in which a digital n-tuple originally received from an outside application is derived from a physical value representing a physical state in the cell having a predetermined correspondence to the digital n-tuple.

Digital value or "logical value": n-tuple of bits represented by a cell in flash memory capable of generating 2 exp n distinguishable levels of a typically continuous physical value such as charge , where n may or may not be an integer.

Erase cycle: The relatively slow process of erasing a block of cells (erase sector), each block typically comprising more than one page, or, in certain non-flash memory devices, of erasing a single cell or the duration of so doing. An advantage of erasing cells collectively in blocks as in flash memory, rather than individually, is enhanced programming speed: Many cells and typically even many pages of cells are erased in a single erase cycle.

Erase-write cycle: The process of erasing a block of cells (erase sector), each block typically comprising a plurality of pages, and subsequently writing new data into at least some of them. The terms "program" and "write" are used herein generally interchangeably.

Flash memory: Non-volatile computer memory including cells that are erased block by block, each block typically comprising more than one page, but are written into and read from, page by page. Includes NOR-type flash memory, NAND-type flash memory, and PRAM, e.g. Samsung PRAM, inter alia, and flash memory devices with any suitable number of levels per cell, such as but not limited to 2, 4, or (as in the embodiment illustrated herein) 8.

Mapping: basic cell-level writing function in which incoming digital n-tuple is mapped to a program level by inducing a program level in the cell, having a predetermined correspondence to the incoming logical value.

Page = A portion, typically 512 or 2048 or 4096 bytes in size, of a flash memory e.g. a NAND or NOR flash memory device. Writing can be performed page by page, as opposed to erasing which can be performed only erase sector by erase sector. A few bytes, typically 16 - 32 for every 512 data bytes are associated with each page (typically 16, 64 or 128 per page), for storage of error correction information. A typical block may include 32 512-byte pages or 64 2048-byte pages.

Precise read, soft read: Cell threshold voltages are read at a precision (number of bits) greater than the number of Mapping levels (2^n). The terms precise read or soft read are interchangeable. In contrast, in "hard read", cell threshold voltages are read at a precision (number of bits) smaller than the number of Mapping levels (2^n where n = number of bits per cell).

Present level, Charge level: The amount of charge in the cell. The amount of charge currently existing in a cell, at the present time, as opposed to "program level", the amount of charge originally induced in the cell (i.e. at the end of programming).

Program: same as "write".

Program level (programmed level, programming level): amount of charge originally induced in a cell to represent a given logical value, as opposed to "present level".

Reprogrammability (Np): An aspect of flash memory quality. This is typically operationalized by a reprogrammability parameter, also termed herein "Np", denoting the number of times that a flash memory can be re-programmed (number of erase-write cycles that the device can withstand) before the level of errors is so high as to make an unacceptably high proportion of those errors irrecoverable given a predetermined amount of memory devoted to redundancy. Typically recoverability is investigated following a conventional aging simulation process which simulates or approximates the data degradation effect that a predetermined time period e.g. a 10 year period has on the flash memory device, in an attempt to accommodate for a period of up to 10 years between writing of data in flash memory and reading of the data therefrom.

Resolution: Number of levels in each cell, which in turn determines the number of bits the cell can store; typically a cell with 2^n levels stores n bits. Low resolution (partitioning the window, W, of physical values a cell can assume into a small rather than large number of levels per cell) provides high reliability.

Retention: of original physical levels induced in the cells; retention is typically below 100% resulting in deterioration of original physical levels into present levels.

Retention time: The amount of time that data has been stored in a flash device, typically without, or substantially without, voltage having been supplied to the flash device i.e. the time which elapses between programming of a page and reading of the same page.

Symbol: Logical value

Threshold level: the voltage (e.g.) against which the charge level of a cell is measured. For example, a cell may be said to store a particular digital n-tuple D if the charge level or other physical level of the cell falls between two threshold values T.

Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions, utilizing terms such as, "processing", "computing", "selecting", "ranking", "grading", "calculating", "determining", "generating", "reassessing", "classifying", "generating", "producing", "stereo-matching", "registering", "detecting", "associating", "superimposing", "obtaining" or the like, refer to the action and/or processes of a computer or computing system, or processor or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within the computing system's registers and/or memories into other data similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices.

The disclosures of all publications and patent documents mentioned in the specification, and of the publications and patent documents cited therein directly or indirectly, are hereby incorporated by reference.

SUMMARY OF THE INVENTION

BCH and RS (Reed-Solomon) are among the most widely used cyclic error correcting codes. They are used in various practical fields such as storage and communication. When these coding schemes are used in mobile applications, power consumption is a major design constraint which sometimes even affects the actual viability of the applicability of the schemes to the mobile applications.

At least the decoding functionality of the above codes typically employs a Chien search. An objective of certain embodiments of the present invention is to provide low power Chien search apparatus useful for mobile applications, memory applications including flash memory applications, and other applications.

There is thus provided, in accordance with at least one embodiment of the present invention, a low power Chien searching method employing Chien search circuitry comprising at least two hardware components that compute at least two corresponding bits comprising a Chien search output, the method comprising activating only a subset of the hardware components thereby to compute only a subset of the bits of the Chien search output; and, only if a criterion on the subset of the bits of the Chien search output is satisfied, activating hardware components other than those in the subset of hardware components, to compute additional bits of the Chien search output other than the bits in the subset of bits.

Further in accordance with at least one embodiment of the present invention, activating-only-if comprises activating all hardware components outside of the subset of the plurality of hardware components, if the Chien criterion is not satisfied by the subset of bits.

Still further in accordance with at least one embodiment of the present invention, the hardware components evaluate at least one error locator polynomial.

Additionally in accordance with at least one embodiment of the present invention, the method also comprises using the roots of the error locator polynomial to

determine locations of errors in a recovered version of sequence of externally provided bits.

Further in accordance with at least one embodiment of the present invention, the sequence of externally provided bits comprises data provided by a host and the recovered version comprises a representation of the data stored in flash memory.

Additionally in accordance with at least one embodiment of the present invention, the data stored in flash memory comprises data encoded in accordance with a Reed-Solomon decoding algorithm.

Still further in accordance with at least one embodiment of the present invention, the data stored in flash memory comprises data encoded in accordance with a BCH decoding algorithm.

Additionally in accordance with at least one embodiment of the present invention, the method further comprises correcting the errors to reproduce the data provided by the host.

Also provided, in accordance with at least one embodiment of the present invention, is a method for correcting a plurality of errors occurring at a corresponding plurality of locations within a recovered version of data provided by a host, the recovered version having been stored in memory, the method comprising constructing a polynomial characterized in that roots thereof indicate locations of the errors in the recovered version of the data; and at least once, determining whether a value of the polynomial equals zero, wherein the value of the polynomial comprises a summation of a sequence of at least two bits, wherein at least once, the determining comprises determining whether each bit in only a subsequence of the sequence of bits equals zero; and subsequently determining whether at least some of the bits in the sequence of bits, other than in the subsequence of bits, equal zero, only if all bits in the subsequence equal zero.

Additionally in accordance with at least one embodiment of the present invention, the method further comprises correcting the errors to reproduce the data provided by the host.

Also provided, in accordance with at least one embodiment of the present invention, is an apparatus for finding roots of a polynomial defined over a finite field, the roots configured to represent location of errors within a recovered version of data, the apparatus comprising polynomial root finding apparatus operatively configured to find roots of a polynomial which is a weighted sum of powers of a variable, the weighted sum being defined by a variable and by a sequence of coefficients by which the powers of the variable are respectively multiplied, the polynomial having a value given an individual sequence of coefficients and given an individual value for the variable, the polynomial root finding apparatus comprising polynomial value determination apparatus operative to determine, for at least one given individual sequence of coefficients and individual value for the variable, whether the value of the polynomial, given the individual sequence of coefficients and the individual value for the variable, equals zero, wherein the value of the polynomial comprises a sequence of at least two bits, the polynomial value determination apparatus comprising partial polynomial value determination apparatus operative to determine whether each bit in only a subsequence of the sequence of bits equals zero; and selectively activatable complementary polynomial value determination apparatus operative to determine whether at least some of the bits in the sequence of bits other than in the subsequence equal zero, only if all bits in the subsequence equal zero.

Further in accordance with at least one embodiment of the present invention, the partial polynomial value determination apparatus comprises a multiplier which is always active and the selectable activatable apparatus is activated only if an Error Locator Polynomial evaluation of the bits in the subsequence is equal to 1.

Further in accordance with at least one embodiment of the present invention, the apparatus also comprising a register upstream of the selectable activatable apparatus.

Still further in accordance with at least one embodiment of the present invention, the bits included in the subsequence of bits comprise the first, lower bits in the sequence.

Additionally in accordance with at least one embodiment of the present invention, the bits included in the subsequence of bits are non-consecutive in the sequence.

Further in accordance with at least one embodiment of the present invention, the hardware components are operative for finding roots of a polynomial which is a weighted sum of powers of a variable, the weighted sum being defined by a variable and by a sequence of coefficients by which the powers of the variable are respectively multiplied, the polynomial having a value given an individual sequence of coefficients and given an individual value for the variable, the value comprising the Chicn search output, the finding including determining, for at least one given individual sequence of coefficients and individual value for the variable, whether the value of the polynomial, given the individual sequence of coefficients and the individual value for the variable. equals zero, wherein the value of the polynomial comprises a sequence of at least two bits.

Further in accordance with at least one embodiment of the present invention, the criterion is whether each of the bits in the subset of bits equals zero.

Still further in accordance with at least one embodiment of the present invention, the subsequently determining comprises determining whether all of the bits in the sequence of bits, other than in the subsequence of bits, equal zero, only if all bits in the subsequence equal zero.

Additionally in accordance with at least one embodiment of the present invention, the subsequently determining comprises determining whether only some of the bits in the sequence of bits, other than in the subsequence of bits, equal zero, only if all bits in the subsequence equal zero; and subsequently determining whether at least some of the bits in the sequence of bits, other than the some bits and the bits in the subsequence of bits, equal zero, only if all of the some bits equal zero and all of the bits in the subsequence equal zero.

Also provided, in accordance with at least one embodiment of the present invention, is a low power Chien searching system employing Chien search circuitry comprising at least two hardware components that compute at least two corresponding bits comprising a Chien search output, the system comprising subset activation apparatus operative to activate only a subset of the hardware components thereby to compute only a subset of the bits of the Chien search output; and polynomial evaluation completion activation apparatus operative, only if a criterion on the subset of the bits of

the Chien search output is satisfied, to activate hardware components other than those in the subset of hardware components, to compute additional bits of the Chien search output other than the bits in the subset of bits.

Further in accordance with at least one embodiment of the present invention, the subset of bits comprises a number of bits c which minimizes the power used by the hardware components to perform the activating step and the only-if-activating step.

Also provided, in accordance with at least one embodiment of the present invention, is a method for saving power consumed by hardware components, the hardware components operatively configured to perform a Chien search, the method comprising providing the hardware components, and initiating the Chien search utilizing only a selective subset of the hardware components whereby power consumption is lower compared with power consumed in a Chien search utilizing all of the hardware components.

Further in accordance with at least one embodiment of the present invention, the Chien search is utilized to determine locations of errors in a recovered version of sequence of externally provided bits.

Still further in accordance with at least one embodiment of the present invention, the sequence of externally provided bits comprises data provided by a host and the recovered version comprises a representation of the data stored in flash memory.

Further provided, in accordance with at least one embodiment of the present invention, is an error correction decoder comprising an error locator polynomial generator operative to generate at least one error locator polynomial; and an error locator polynomial evaluator operative to rule out at least one root of the error locator polynomial based on only a partial evaluation thereof.

Any suitable processor, display and input means may be used to process, display, store and accept information, including computer programs, in accordance with some or all of the teachings of the present invention, such as but not limited to a conventional personal computer processor, workstation or other programmable device or computer or electronic computing device, either general-purpose or specifically constructed, for processing; a display screen and/or printer and/or speaker for

displaying; machine-readable memory such as optical disks, CDROMs, magnetic-optical discs or other discs; RAMs, ROMs, EPROMs, EEPROMs, magnetic or optical or other cards, for storing, and keyboard or mouse for accepting. The term "process" as used above is intended to include any type of computation or manipulation or transformation of data represented as physical, e.g. electronic, phenomena which may occur or reside e.g. within registers and /or memories of a computer.

The above devices may communicate via any conventional wired or wireless digital communication means, e.g. via a wired or cellular telephone network or a computer network such as the Internet.

The apparatus of the present invention may include, according to certain embodiments of the invention, machine readable memory containing or otherwise storing a program of instructions which, when executed by the machine, implements some or all of the apparatus, methods, features and functionalities of the invention shown and described herein. Alternatively or in addition, the apparatus of the present invention may include, according to certain embodiments of the invention, a program as above which may be written in any conventional programming language, and optionally a machine for executing the program such as but not limited to a general purpose computer which may optionally be configured or activated in accordance with the teachings of the present invention.

Any trademark occurring in the text or drawings is the property of its owner and occurs herein merely to explain or illustrate one example of how an embodiment of the invention may be implemented.

Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions, utilizing terms such as, "processing", "computing", "estimating", "selecting", "ranking", "grading", "calculating", "determining", "generating", "reassessing", "classifying", "generating", "producing", "stereo-matching", "registering", "detecting", "associating", "superimposing", "obtaining" or the like, refer to the action and/or processes of a computer or computing system, or processor or similar electronic computing device, that manipulate and/or transform data represented as physical, such as electronic, quantities within the computing system's registers and/or memories, into other data

similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Certain embodiments of the present invention are illustrated in the following drawings:

Fig. IA is a simplified functional block diagram of an encoding/decoding system using a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, the system being constructed and operative in accordance with certain embodiments of the present invention;

Fig. IB is a simplified functional block diagram of the decoder of Fig. IA, which uses a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, which is constructed and operative in accordance with certain embodiments of the present invention;.

Fig. 2 is a simplified functional block diagram of flash memory apparatus comprising, e.g. in an internal microcontroller, the encoding/decoding system of Fig. IA and particularly the decoder of Fig. IB, all operative in accordance with certain embodiments of the present invention;

Fig. 3 is a functional block diagram illustration of an "in series" prior art alternative for blocks 220 and 230 of Fig. IB , in which polynomial evaluation proceeds for each _i in series for a field over GF(2q);

Fig. 4 is a prior art functional block diagram illustration of an "in parallel" prior art alternative for blocks 220 and 230 of Fig. IB, having registers whose initial contents are respective Λ values as shown, in which polynomial evaluation proceeds for N different values of _i in parallel for a field over GF(2q);

Fig. 5A is a simplified functional block diagram illustration of an "in parallel" implementation of blocks 220 and 230 of Fig. IB, having registers whose initial contents are as shown, which effects a two-mode low power partial Chien search characterized in that searching is only sometimes performed at high precision, and at other times is performed only at low precision, all in accordance with certain embodiments of the present invention;

Fig. 5B is a diagram showing the inputs and outputs of an individual one of the multipliers of Fig. 5A, according to a two-precision mode embodiment of the apparatus of Fig. 5A in which multiplication is effected either at full-precision or with partial precision;

Fig. 6 is a table defining various power parameters used in power computations described herein;

Figs. 7A - 7E are simplified electronic diagrams of five alternative implementations of an individual one of the multipliers of Fig. 5A, constructed and operative in accordance with the two-precision mode embodiment of Fig. 5B;

Fig. 8A is a simplified functional block diagram illustration of an "in parallel" implementation of blocks 220 and 230 of Fig. IB, having registers whose initial contents are as shown, which effects a three-mode low power partial Chien search characterized in that searching is only sometimes performed at high precision, and at other times is performed at medium or low precision, all in accordance with certain embodiments of the present invention.

Fig. 8B is a diagram showing the inputs and outputs of an individual one of the multipliers of Fig. 8A, according to a three-precision mode embodiment of the apparatus of Fig. 8A in which multiplication is effected either at full-precision, medium-precision or minimal-precision. It is appreciated that apparatus constructed and operative in accordance with certain embodiments of the present invention can have any suitable number of precision modes, of which 2-mode and 3-mode alternatives are specifically shown; Fig. 8C is a diagram of tap control logic for the tap_enable signal of Fig. 8A according to an embodiment of the present invention;

Fig. 9 is a simplified electronic diagram of one implementation of an individual one of the multipliers of Fig. 5A, constructed and operative in accordance with the three-precision mode embodiment of Fig. 8B but otherwise similar to the multiplier of Fig. 7C, it being appreciated that alternatively, any of the two-precision mode implementations of Figs. 7A, 7B and 7D can be adapted to the three-precision mode embodiment of Fig. 12, mutatis mutandis;.

Fig. 10 is a prior art simplified flowchart illustration of a method of operation for the "in parallel" prior art apparatus of Fig. 4;

Fig. 11 is a simplified flowchart illustration of a method of operation for the apparatus of Fig. 5A, according to the two-precision mode embodiment thereof;

Fig. 12 is a simplified flowchart illustration of a method of operation for the apparatus of Fig. 5A, according to the three-precision mode embodiment thereof; and

Fig. 13 is a simplified flowchart illustration of a method for performing the two-stage root check step of Fig. 12, according to certain embodiments of the present invention.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

Reference is now made to Fig. IA which is a simplified functional block diagram of an encoding/decoding system using a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, the system being constructed and operative in accordance with certain embodiments of the present invention. In Fig. 1A, a message source 100 provides a message m(x) which it may be desired to transmit or to store. e.g. in flash memory, to Error Correction Coding (ECC) apparatus 110. The ECC apparatus 110 may comprise BCH or Reed-Solomon cyclic error correction coding apparatus and is typically operative for computing and for adding, to the message m(x), redundancy bits, thereby to generate a codeword c(x) of a known codebook such as BCH or Reed-Solomon with known parameters. The channel 120, which may comprise any medium through which the message is conveyed from transmitter 100 to receiver 130, or may comprise the storage medium, adds errors e(x) to the codeword c(x).

The errors may stem from various physical processes such as thermal noise, deterioration of storage medium over time and, especially after many read/write operations, inaccuracies in the transmitter or receiver hardware. Each error occurs at a particular location within the message, which is assumed to comprise a sequence of bits or of symbols. In the former case, binary BCH code is typically used for encoding and decoding, whereas in the latter case, non-binary BCH code, or RS code is used. hi the first, binary, instance, n is used in the foregoing discussion to indicate a bit of the data being read or received in which an error has occurred. In the second, non-binary, instance, n is used in the foregoing discussion to indicate a symbol of the data being read or received in which an error has occurred.

r(x) = c(x)+e(x) is the received data which is typically received by an error correcting decoder 130, also termed herein the "receiver". The receiver 130, using the redundancy that was added to the message and the known codebook, is operative to substantially reconstruct the original message m"(x) and convey it to the intended target, message sink 140. According to certain embodiments of the present invention, the decoder 130 is operative to perform a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, e.g. as described and illustrated below.

Reference is now made to Fig. IB which is a simplified functional block diagram of the decoder 130 of Fig. IA. As shown, the decoder 130 uses a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, and is constructed and operative in accordance with certain embodiments of the present invention.

As described above, the encoder 110 can be described in terms of a generation matrix G, thus the encoding process performed by encoder 110 comprises a matrix multiplication c = mG. As described above, c is the transmitted codeword and m is the message to be transmitted or, for data storage applications, the data to be stored. The decoder of Fig. IB is operative to perform syndrome decoding (functionality 200), such that there exists a parity check matrix H which has the following property: $GH^T=0$. It follows that $cH^T= mGH^T=0$ (formula IV). As described above, the received vector r comprises the transmitted codeword c and the errors added in the channel 120 i.e. r = c + e. The "receiver" 130 (which in flash memory applications, may be implemented within microcontroller 244 of Fig. 2) computes the syndrome vector s using the parity check matrix. Specifically (formula V):

$$s = rH^T = cH^T + eH^T = mGH^T + eH^T = 0 + eH^T = eH^T, \text{ or in short } s = eH^T.$$

Another functionality in a conventional decoder is that which generates an Error Locator Polynomial (functionality 210 in Fig. IB). Due to the special form of the BCH and RS codes and of the parity check matrix H the set of equations $s = eH^T$ may be solved directly by exhaustive search in the decoder 130 to find the error vector e and correctly decode the received message r(x), however, the exhaustive search is computationally unattractive. Therefore, typically an Error Locator Polynomial (ELP) is introduced, the roots of which correspond to a one to one mapping of the error locations as described above and as is known in the art.

If $A = \Lambda_1 \alpha^{1+} A_2 \alpha^{2'}+...+ A_J \alpha^{jl}$ equals A_0 at some clock n, this implies, as described above and as known in the art, that α^l is a root of the Error Locator Polynomial (ELP). This in turn implies, if binary BCH code is being used, that an error has occurred in bit n of the data being read or received. If non-binary BCH code, or RS code, is used, the fact that α^1 is a root of the Error Locator Polynomial (ELP) implies that an error has occurred in symbol n of the received or read data. Known algorithms

for deriving the Error Locator Polynomial (ELP) from the syndromes include the Berlekamp-Massey and the Euclidean algorithms as described e.g. in "Error Correction Coding Mathematical Methods and Algorithms", Todd K. Moon, John Wiley & Sons, Inc., 2005. It is assumed that the ELP is normalized that the first monomial (of X^0), A_0, is normalized to 1. In the case of BCH this is indeed the case whereas in the case of RS, multiplying the ELP by the inverse of the monomial of X^0 ensures this is the case.

The Error Locator Polynomial A generated by unit 210 can be written as follows (formula VI):

$$\Lambda(x) = \Lambda_0 + \Lambda_1 X + \Lambda_2 x^2 + \ldots + A_j X^J$$

where J is the number of errors in the received vector. In the assumed, i.e. worst, case, j = J = the maximum number of errors that the error correction algorithm is able to correct per (page or block of) n symbols. The n symbols comprise n bits if an BCH algorithm is used or n symbols if a RS algorithm is used. The symbols A_j and λ_j are used interchangeably. In the formulae and drawings herein, J and t are used interchangeably, e.g. j = 1, ... J or j = 1, ... t.

Once the Error Locator Polynomial (ELP) has been generated by functionality 210, Error Locator Polynomial evaluation functionality 220 then evaluates the Error Locator Polynomial for all the elements of the field over which the Error Locator Polynomial is defined. The elements in the field that zero the Error Locator Polynomial (ELP) are the error locations. Computations are typically performed in the $GF(q^m)$ field which is a finite field.

Denoting α as a primitive element, all the field elements can be conventionally generated from consecutive powers of α i.e. α°, α',...., α^{q^m}.

Errors are then corrected at each error location identified, by unit 230 of Fig. IB. If the code is binary, correction comprises a simple flip of the bit. If the code is non-binary, error value computation typically uses Forney's algorithm. It is appreciated that typically, Error Locator Polynomial (ELP) evaluation described above uses a Chien-Search to find or search for all the roots of $\Lambda(x)$. To do this, x is typically evaluated for all powers of α i.e. x =1, α, α^2, α^3,....., α^{q^m}. This can, for example, be effected by the

conventional ELP-evaluation-by-Chien-search apparatus depicted in prior art Fig. 3 or alternatively it may be effected by a low power partial Chien search operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, e.g. as described in detail below with reference to Figs. 5A —9, 11 — 12.

Fig. 2 is a simplified functional block diagram of a flash memory apparatus comprising, e.g. in an internal microcontroller 244, the encoding/decoding system of Fig. 1A and particularly the decoder of Fig. 1B, all operative in accordance with certain embodiments of the present invention. As shown, the flash memory apparatus of Fig. 2 typically interacts with a host 240 and typically includes the microcontroller 244 as well as one or more erase sectors 246 each comprising one or more pages 248 each including cells 249. The microcontroller 244 effects erasing of, writing on and reading from the erase sector/s 246, by suitably controlling erasing circuitry 250, writing circuitry 252 and reading circuitry 254, respectively. According to certain embodiments of the present invention, microcontroller 244 includes an error correction code decoder operative to receive data from the reading circuitry 254, to decode the data, including performing a low-power Chien search for error locations, including a partial Chien search, operative to rule out at least one root of an error locator polynomial based on only a partial evaluation of the polynomial, e.g. as described in detail below with reference to Figs. 5A - 9, 11 - 12, and to provide the data thus decoded to the host 240 which therefore constitutes both source 100 and sink 140 of Fig. IA. in memory applications.

In flash memory applications, the channel 120 generally represents the deterioration in the data stored in memory over time and due to repeated cycling, and the encoding and decoding (functionalities 110 and 130 in Fig. IA) are performed within one or more suitable controllers e.g. the microcontroller 244 of Fig. 2 which is external to the flash memory device 245 or an external controller operatively associated with the host 240 and external to device 245.

As described above, evaluation of an error locator polynomials for all elements in the field over which the polynomial is defined, may be performed using a conventional Chien search. This is shown in Fig. 3 which is a functional block diagram illustration of an "in series" prior art alternative for blocks 220 and 230 of Fig. IB. In

Fig. 3, Reg_l to Reg_J are J registers 300 which are initiated prior to the beginning of operation to hold $\Lambda_1 \ldots \Lambda_J$, i.e. the coefficients of the Error Locator Polynomial (ELP) where J is the error correction capability of the designed code. The symbol j does not denote a constant over multiple operations of the circuit, but rather varies and denotes the number of errors in the currently decoded data block. The elk signal 310 in Fig. 3 denotes the clock signal that clocks the Reg_l ...Reg_J registers 300. Const_l ....Const _J in Fig. 3 are successive powers of the primitive element in the field α.

In each successive clock of elk signal 310, the contents of each register Reg_l …-Reg_J (which are initially respective Λ values as shown) are multiplied by the respective constants Const_l…..Const_J and latched into respective ones of the Reg_l….Reg_J registers as shown. An adder 320 in Fig. 3 adds the partial sums of the Error Locator Polynomial (ELP) to produce sum A which is the evaluation of the Error Locator Polynomial (ELP) for $x=\alpha^i$ at the i'th clock cycle. IfA equals A_0 at some clock n, this implies, as described above and as known in the art, that α^{-i} is a root of the Error Locator Polynomial (ELP). This in turn implies, if binary BCH code is being used, that an error has occurred in bit n of the data being read or received. If non-binary BCH code, or RS code, is used, the fact that α^{-i} is a root of the Error Locator Polynomial (ELP) implies that an error has occurred in symbol n of the received or read data. Once the algorithm iterates over all elements in the field, and all the errors are identified, the decoding process is complete.

As described above, the Chien-Search algorithm performs evaluation of the Error Locator Polynomial (ELP) for all the elements in the field in order to find Error Locator Polynomial s roots which are the error locations. That is, for each $x=\alpha^\circ$, α^1,...., $\alpha^{q^{m-2}}$ A(x) is evaluated, thereby to obtain an element in $GF(q^m)$. If that element is equal to 0, an error is declared to have occurred at the respective location. Typically, only the sum of the last J monomials of A(x) (sum = $A_i \alpha^i + A_2 \alpha^{2i}+...+ A_j \alpha^{ji}$) is evaluated and the sum is compared to -1; if the sum is -1 then the evaluation of $\Lambda(x)$ at that point is 0. Each element in $GF(q^m)$ may be defined by m sub-elements over GF(q). According to certain embodiments of the invention, e.g. as may be appreciated with reference to steps 1030, 1124, 1224 and 1330 described below, the power consumption of the Chien Search is reduced by computing the result for only the first c<m sub-

elements and comparing these sub-elements to -1, thus sometimes saving the power needed to compute the other m-c sub-elements, where c is any number less than m. It is appreciated that in the GF(2^q) field, -1=1.

Even though the first c sub-elements over GF(q) may be -1, it is not necessarily the case that the rest of the c-m sub-elements will be -1. Therefore, some false alarms may occur. However, as explained in further detail below, it is not generally necessary to reevaluate this polynomial for all m sub-elements every time the first m sub-elements are detected to be -1.

In Fig. 3, the Error Locator Polynomial (ELP) is evaluated iteratively, using intermediate results of previous computations, as stored in the registers Reg_l ...Reg_J (which hold multiples of Λ_i, A_2, ..., Λ_J). Given the current intermediate results the first c sub-elements of the polynomial for the next step can be evaluated as described in detail hereinbelow. However, if it is desired for all intermediate results to contain all m sub-elements, then, for the next computation intermediate results with m sub-elements each are still needed. However, as described below, this potential complication may be mitigated, yielding a significant power saving, for some multiplication implementations.

Fig. 4 is a prior art functional block diagram illustration of an "in parallel" prior art alternative for blocks 220 and 230 of Fig. IB.

According to certain embodiments of the present invention, the prior art apparatus of Fig. 3 may be modified, as shown in Fig. 4, by adding additional hardware to parallelize the computations to occur at N evaluation points in parallel, where N, the parallelization factor, is a suitable integer such as 3, 4 or 5 (N = 3 in the illustrated embodiment). Parallelization may be accomplished by adding N-I additional multipliers per register (for each of J registers, in the illustrated embodiment) and the operational clock frequency is reduced by a factor of N. At each cycle the registers (Reg_l ...Reg_J) are updated by multiplying them by the same amount which would have been updated following N clock cycles in the apparatus of Fig. 3. Thus, for every Nth polynomial evaluation, all intermediate results are still computed with full precision, i.e., all m sub-elements are computed for all multiplications. For example, for RS codes over GF(2^m) there are m sub-elements, each comprising a bit. For RS codes

over GF(3^m) there are m sub elements, each comprising a symbol having 3 possible values: 0, 1 and 2.

However, these intermediate results can now be relied upon to evaluate the first c sub-elements of the Error Locator Polynomial (ELP) for the other N-I evaluation points, as described below, enabling a significant power saving and circumventing the complication described above. The constants are powers (exponents) of the prime element in the field, alpha. Assuming the field described above, alpha = 2. however this is not intended to be limiting.

In the embodiment illustrated in Fig. 4, N=3. Thus, in Fig. 4, N-I = 2 additional multipliers were added for each of the J registers. Specifically, multiplier array 330 of Fig. 3 is augmented with additional multiplier arrays 340 and 350. The multipliers in array 330 are termed herein mult_l to mult_J. The multipliers in array 340 are termed herein mult_l.l to mult_J.l. The multipliers in array 350 are termed herein mult_1.2 to mult_J.2. In the illustrated example, AO, A1 and A2 represent N = 3 different evaluations of the Error Locator Polynomial (ELP) for N = 3 consecutive elements of the field and N-I = 2 more adders 322 and 324 are provided to sum-up the additional two evaluations of the Error Locator Polynomial (ELP), A1 and A2 respectively. In addition the constants Const_l ....Const_.ï are updated to account for 3 consecutive polynomial evaluations for each elk cycle and the clock frequency is reduced by a factor of N = 3.

It is appreciated that the power consumption of the apparatus of Fig. 4 is less than that of the apparatus of Fig. 3 even if all multipliers conventionally use full precision multiplications for all multipliers. This is because the clock frequency to the registers has been reduced and the power consumption at the registers is proportional to the elk frequency.

Fig. 5A is a simplified functional block diagram illustration of an "in parallel" implementation of blocks 220 and 230 of Fig. IB, which effects a two-mode low power partial Chien search characterized in that searching is only sometimes performed at high precision, and at other times is performed only at low precision, all in accordance with certain embodiments of the present invention.

Prior art Fig. 4 is general in that the field over which the multiplication operations are defined is not specified. In Fig. 5A, in contrast, the field over which the multiplication operations are defined is GF 2^15 such that each full-precision multiplication result comprises 15 bits (J = 15). More importantly, whereas in Fig. 4 each multiplier has only one mode of operation (full precision e.g. all 15 bits if the field is GF 2^15), in Fig. 5A, according to certain embodiments of the present invention, multipliers 350 and 348 each have two modes of operation, a full-precision mode of operation in which all 15 bits of the multiplication result, A2, are computed and a limited precision mode of operation in which only 3 bits of the multiplication result, Al, are computed. The expanded modes of operation of multipliers 350 and 348 are enabled by special signals mutl_enable_2 and mult_enable_l respectively.

It is appreciated that in the embodiment of Fig. 5A the power of the multipliers in arrays 348 and 350 is reduced, relative to the embodiment of Fig. *4*, because the multiplier arrays 348 and 350 initially compute only c out of m sub-elements of the multiplications. An enable signal is used, typically for each array, which enables or disables the computation of the rest of the m-c sub-elements. In the illustrated embodiment, the enable signal for multiplier array 348 is termed mult_enable_l , and the enable signal for multiplier array 350 is termed mult_enable_2. In the event that the computation of m-c sub-elements is disabled, toggling of the appropriate circuitry is inhibited and the power consumption is reduced.

As an example, consider a code over the GF(2^{15}) field for an application in which the multipliers are constructed to produce (during normal operation) only the 3 LSB bits of A2, the full 15 bit result. If the element being evaluated happens to be a root of the Error Locator Polynomial (ELP), the results are 15'b000_0000_0000_0001 (summation over all the taps). The 3 lower bits of the summation are equal to 3'b0_01. Only when this result is encountered further evaluation of the rest of the bits is effected. Computing only the first 3 bits of the multiplication result A2 consumes approximately 1/5 of the power of computing the full 15 bit result.

If there is no root for the current computed position, the event that the computed 3 bits summation is equal to 3'b0_01 occurs, on average, only once in 8 positions, such that the full amount of power is expended only, at most, once every 8 clock cycles. If further evaluation is needed, an additional clock cycle may be employed to perform this

computation by delaying the advance of the tap's registers and setting a tap enable signal high to allow full bit computation. The new delay of the Chien Search computation is then 9/8 of the previous delay.

Multiplier enable generation logic blocks 550 and 560 generate the multiplier enable signals for A1 and A2 summation results respectively. These logic units generate mult_enable_1 and mult_enable_2 signals respectively, which enables full bit multiplication in each of the multipliers in arrays 348 and 350 respectively. mult_enable_i is set to logic 1 when A1 is equal to 3'b001 . mult_enable_2 is set to logic 1 when A2 is equal to 3'b001. Tap control logic 570 generates the tap_enable signal which stops the advance of the tap registers during the clock that is used for the full bit multiplication and summation following a suspected hit: A1 = A2 = 3'b001.

Certain methods of operation of the prior art system of Fig. 4, in which each multiplier has only one operating mode, and of the system of Fig. 5A, in which each multipliers 348 and 350 has two operating modes, are described below with reference to the flowcharts of Figs. 10 and 11 respectively.

Fig. 5B is a diagram showing the inputs and outputs of an individual one of the multipliers of Fig. 5A, according to a two-precision mode embodiment of the apparatus of Fig. 5A in which multiplication is effected either at full-precision or with partial precision. Generally, as shown, the operation of each of the multipliers in 330 and 340 in Fig. 5A varies as a function of the value of mult_enable_2 or mult_enable_1 respectively. It is appreciated that partial precision need not comprise 3-bit precision specifically and that any number of bits which is less than the full number of bits may be computed at the partial precision stage.

Fig. 6 is a table of power parameters useful in understanding power computations described below both to demonstrate the general utility of certain embodiments of the present invention and to best select the size/s of the subset/s of bits computed in precision modes other than full-precision mode.

As an example, consider Chien Search apparatus which, like that of Fig. 5A, is adapted for a code defined over the GF(2^{15}) field. If the higher - say —10 bits of a term (monomial) in the summation A_1 are not computed when a partial bit

multiplication occurs using an AND gate with the enable signal, no switching occurs and ~2/3 of the power is saved.

A formula (GF(q^m)) for the total power consumed by the multipliers of Fig. 5A is now presented. P_{reg} denotes the total power drawn by the registers at the original (high) frequency. P_{mult} denotes the power drawn by the multiplier array in Fig 3 at the original frequency and P_{COmb} is the power attributed to the combinatorial logic in Fig 3. It is assumed that the computation of c out of m sub-elements takes c/m of the power drawn by the full multiplier. The power drawn by the circuit in Fig. 4, without sub-element computations, is equal to $P_{total}=P_{reg}/N+P_{comb}$ where P_{COmb} is roughly equal to P_{mult}.

In the apparatus of Fig. 5A, the power consumed, assuming selection of the c first bits, is approximated by the following expression (Formula VII):

$$P(c,N) = \frac{P_{reg}}{N} + \frac{P_{mult}}{N} + \frac{(N-1)}{N}\cdot P_{\cdots\cdots}\cdot\frac{c}{m} + \frac{(N-1)}{N}\cdot P_{\cdots\cdots}\cdot\frac{1}{q^c}$$
$$= \frac{P_{reg}}{N} + \frac{P_{mult}}{N}\cdot\left(1+(N-1)\cdot\left(\frac{c}{m}+\frac{1}{q^c}\right)\right)$$

For example, when m=15, q=2, and it is desired to minimize the power consumed, P(c,N), the term $\left(\frac{c}{m}+\frac{1}{q^c}\right)=\left(\frac{c}{15}+\frac{1}{2^c}\right)$ may be minimized, regardless of N. One can easily check by enumerating over c=1..15 that this term is minimized when c=3 which in total saves roughly ½ of the power. In general, the higher N is, the less power may be consumed. However, N is usually limited by hardware restrictions such as but not limited to gate-count. Therefore, given N the function P(c,N) is minimized over c.

The power computations are rough and depend highly on the code construction, the constants, the field and its generator polynomial and on the process, cell-library and frequency of operation. The above computations, while based on a specific application and having many assumptions built-in e.g. as set out in Fig. 6, serve to show the utility of certain embodiments of the present invention.

One possible implementation of computing a multiplication of two elements and obtaining the result for just some of sub-elements is now described. As the arithmetic is performed over a Galois-field, the multiplication operation of elements z and y in the

field can be expressed as follows: z(x)*y(x) mod p(x) where p(x) is the generator polynomial of this field and z(x) and y(x) are the polynomial representatives of the elements z and y. Thus, the following polynomial notation may be employed (formula VIII):

$$y(x)\text{-}\sum(x) \bmod p\{x)$$

$$= \left[\sum_{\iota=0}^{M-1} .y,^{*'} \right] - \sum_{\iota=0}^{Bl-1} z,x' \Big]_{J}^{\Lambda} \backslash \bmod p(x)$$

$$= \sum_{l=0}^{2m-2}\left(\sum_{k=0}^{l} z_k y_{l-k}\right)x^l \bmod p(x)$$

$$= \sum_{l=0}^{2m-2}\left[\sum_{k=0}^{l} z_k y_{l-k}\right]\left[x^l \bmod p(x)\right] = \sum_{l=0}^{2m-2} A_l V_l(x)$$

where $A_1 = \sum_{k=0}^{l} z_k y,_{-k}$ is a sub-element (i.e. an element in the field GF(q)) and $V_/(x) = [x^{\prime} \bmod p(x)]$ is a polynomial whose elements are in GF(q). The multiplication $A_1V_1(X)$ multiplies each of the elements in $V_1(x)$ by A_1 in the field GF(q). Thus, the final expression in the above development of polynomial notation shows that if y is constant, the multiplication per sub-element may be computed independently of the results from the other elements. Thus, to obtain the first c < m sub-elements of the above multiplication, only c/m of the power is required on average.

If the error locator polynomial is defined over a field GF(2^m), computation of A_1 (e.g. A_1, ...A_(N-1)) may be effected by a set of XOR operations on the bits of z. Thus, following a multiplication by a constant (say $y=\alpha^4$), each bit in the multiplication result, on average, is generated by m/2 XOR operations. Computing only c bits of the result would require only c*m/2 XOR operations. In contrast, the full multiplication result employs m*m/2 XORs such that the system of the present invention is seen to carry out less XOR operations and hence, to consume less power.

Figs. 7A - 7E are simplified electronic diagrams of five alternative implementations of an individual one of the multipliers of Fig. 5A, constructed and operative in accordance with the two-precision mode embodiment of Fig. 5B.

In Fig. 7A, each multiplier of Fig. 5A includes a first multiplying device 414 which produces 15 bits and an additional small multiplier 416 that produces only a 3 bit

result in parallel to the multiplier 414. The larger multipliers 414 remains gated off during normal operation and is switched into action solely in the cycles where a full bit result is to be obtained because the partial result has passed the criterion i.e. equals "1". This embodiment may be less practical in certain applications due to added gate-count. In Fig. 7A, as shown, there is a 15 bit result if mult_enable = 1. However, only first 3 bits of the result are computed if mult_enable =0.

In the embodiment of Fig 7B, the multiplier includes first and second multiplying devices 424 and 426 of which the first is the larger. The second multiplying device 426 is always active and computes the first 3 bits. A second multiplying device 424 computes the upper 12 bits and is activated only if the Error Locator Polynomial evaluation of the first 3 bits is equal to 1. It should be noted that the enable line may be driven by a control state machine which detects that the first 3 bits of the Error Locator Polynomial evaluation are different from 1 and then in the next clock performs the same computation but this time the first, larger multiplying device 424 is enabled. Another variation is to close the loop immediately after the comparator without going through the state machine. This time, everything will happen within the same clock.

The embodiment of Fig. 7C is similar to the embodiment of Fig. 7B except that flip-flop enable registers 430 and 432 are added, as shown, upstream of the 12 bit multiplier. This assures that the inputs to the 12 bit multiplier change only once, every time the enable line goes high, rather than twice: once when the enable line goes high and once when the enable line goes low.

The embodiment of Fig 7D is similar to that of Fig. 7B except that the first 3 bits always computed in the first stage, by the small multiplier 448 do not necessarily comprise the lowest 3 bits and instead may comprise any 3 bits from among the 15 bits.

Three possible modifications are now described with reference to Figs. 7B —7D respectively. These can be used separately or in any suitable combination. To obtain the optimal set of c-sub-elements computed in Fig 7D, the following computations may be performed:

Denote the power consumed by the multipliers in, say, Fig. 5A, for c sub-elements by P_{mult}. The term "sub-element" is used herein to denote a sub-set of the bits

of an element of a finite field over which polynomials being Chien-searched are defined. The multipliers are divided into N groups of size J, where the multiplication results of each group, 350 and 348, are added up to give a c-sub-element evaluation of the Error Locator Polynomial (ELP). The power of each such group may be optimized over all choices of c-sub-elements out of m. Also, if the first sub-element is not computed, then the comparison condition (step 1030 in Fig. 10) should be 0 instead of 1.

Thus the choice of the condition may depend on the choice of the c-sub-elements which are computed, and thus may also depend on the group of multipliers. $P_{mult,c}$ is then the average power consumed by each group at the original frequency, averaged over all groups. The power drawn as a function of the selected subset of elements is then a target function which is to be minimized over all possible subsets of elements (one of which is the lowest c bits as in Fig. 7A). This target function to be minimized is given by (Formula IX):

$$P(c,N) = \frac{P_{reg}}{N} + \frac{P_{mult,m}}{N} + \frac{(N-1)}{N}\cdot P_{mult,c} + \frac{(N-1)}{N}\cdot P_{mult,m}\cdot\frac{1}{q^c}$$
$$= \frac{P_{reg}}{N} + \frac{P_{mull,m}}{N}\cdot\left(1+\frac{N-I}{q^c}\right) + \frac{P_{mull,c}}{N}(N-I)$$

One example is where $P_{mult,c}$ does not depend on the c-sub-elements chosen for the computation. In this case, replacing the apparatus of Fig. 7D by the apparatus of Fig. 7B would give equivalent results.

One additional clock cycle is activated each time the c-sub-element evaluation of the polynomial gives a suspect result. This could be avoided if the rest of the m-c sub-elements are computed on the same clock. This may be achieved by connecting the Error Locator Polynomial (ELP) evaluation condition checker (blocks 550 and 560 in Fig. 5A) directly to the enable signal of the multiplier as shown in Fig. 5B and not through the tap control logic (block 570 in Fig 5A). This enables the multiplier to compute the other (m-c) sub-elements. However, this creates a very long logic route and may limit the clock frequency at which the system might work.

According to yet another embodiment, as shown in Fig. 7E, a latch 450 is added before the "large" multiplier 452. The latch 450 is operative to latch the data to the (m-

c) sub-elements which are only rarely computed. Whenever the enable signal goes high, the latch 450 latches in the new data, enabling the computation of the m-c sub-elements for the new data. Thus, power is consumed only when the enable signal goes high rather than both when it goes high and again later when it goes back to low.

Fig. 8A is a simplified functional block diagram illustration of an "in parallel" implementation of blocks 220 and 230 of Fig. IB, which effects a three-mode low power partial Chien search characterized in that searching is only sometimes performed at high precision, and at other times is performed at medium or low precision, all in accordance with certain embodiments of the present invention. It is appreciated that an apparatus constructed and operative in accordance with multi-precision mode embodiments of the present invention can have any suitable number of precision modes, of which 2-mode and 3-mode alternatives are specifically shown merely by way of example.

Fig. 8B is a diagram showing the inputs and outputs of an individual one of the multipliers of Fig. 8A. As shown, multiplication is effected either at full-precision, medium-precision or minimal-precision. Fig. 8C is a diagram of tap control logic for the tap_enable signal of Fig. 8A according to an embodiment of the present invention.

Fig. 9 is a simplified electronic diagram of one implementation of an individual one of the multipliers of Fig. 5A, constructed and operative in accordance with the three-precision mode embodiment of Fig. 8B but otherwise similar to the multiplier of Fig. *1C,* it being appreciated that alternatively, any of the two-precision mode implementations of Figs. 7A, 7B and 7D can be adapted to the three-precision mode embodiment of Fig. 12, mutatis mutandis. In the embodiment of Fig. 9, each multiplier has 3 precision modes rather than 2. In the illustrated embodiment, 2, 7 and 15 bits of the multiplication result are computed when the multiplier works in minimal mode, medium mode and full-mode precision respectively. The Chien search process then includes an additional Summation step as described below in detail with reference to the flow-chart of Fig. 12.

To optimally select the number of sub-elements computed at each step in Fig. 9, the following formula (Formula X), which estimates the power consumed, may be used.

Preg and Pmul are defined as before and c1 and c2 are the number of sub-elements computed at the first precision and second precision levels respectively:

$$P(c_1,c_2,N)=\frac{P_{reg}}{N}+\frac{P_{mult}}{N}+\frac{(N-1)}{N}\cdot P_{mult}\cdot\frac{c_1}{m}+\frac{(N-1)}{N}\cdot P_{mult}\cdot\frac{c_2}{m}\frac{1}{q^{c_1}}+\frac{(N-1)}{N}\cdot P_{mult}\cdot\frac{1}{q^{c_2}}$$
$$=\frac{P_{reg}}{N}+\frac{P_{mult}}{N}\cdot\left(1+(N-1)\cdot\left(\frac{c}{m}+\frac{c_2}{m}\frac{1}{q^{c_1}}+\frac{1}{q^{c_2}}\right)\right)$$

As an example, consider the case of q=2 and m=15. To minimize the power consumption $P(c_1,c_2,N)$ the following may be minimized: $\left(\frac{c}{m}+\frac{c_2}{m}\frac{1}{q^{c_1}}+\frac{1}{q^{c_2}}\right)$. Enumeration over the values of c_1 and C_2 between 1 and 15 yields that computing 2 bits at the first step and 7 bits at the second step yields the lowest power consumption.

Fig. 10 is a prior art simplified flowchart illustration of a method of operation for the "in parallel" prior art apparatus of Fig. 4. Fig. 11 is a simplified flowchart illustration of a method of operation for the apparatus of Fig. 5A, operative in accordance with a two-precision mode embodiment of the present invention. Fig. 12 is a simplified flowchart illustration of a method of operation for the apparatus of Fig. 8A, operative in accordance with a three-precision mode embodiment of the present invention. Fig. 13 is a simplified flowchart illustration of a method for performing the 2-stage root check step 1230 of Fig. 12, according to certain embodiments of the present invention.

It is appreciated that in each iteration, N error locator polynomials are evaluated. Since there are n/N (due to parallelization) iterations, a total of n polynomials are evaluated per codeword.

It is appreciated that in the illustrated embodiments, the number of precision modes in the embodiment of Figs. 8A - 8C, 9, 12 - 13 happens to be 3, the number of bits computed in the lowest precision mode in the embodiment of Figs. 5A - 7E, 11 happens to be 3, and the number N of evaluation points in the parallelized apparatus of Fig. 3 also happens to be 3. It is appreciated that these selected values are not intended to be limiting nor need they be equal: alternatively, for example, the number of precision modes may be 4, the number of bits computed in the lowest precision mode in

the embodiment of Figs. 5A - 7E, 11 may be 2, and the number N of evaluation points in the parallelized apparatus of Fig. 3 may be 5.

It is appreciated that the GF(2^{15}) field is used herein merely as an example and that any other field parameters may be employed, such as GF(2^A8) or GF(7^A5) or GF(7^A8).

Certain operations are described herein as occurring in the microcontroller internal to a flash memory device. Such description is intended to include operations which may be performed by hardware which may be associated with the microcontroller such as peripheral hardware on a chip on which the microcontroller may reside. It is also appreciated that some or all of these operations, in any embodiment, may alternatively be performed by the external, host-flash memory device interface controller including operations which may be performed by hardware which may be associated with the interface controller such as peripheral hardware on a chip on which the interface controller may reside. Finally it is appreciated that the internal and external controllers may each physically reside on a single hardware device, or alternatively on several operatively associated hardware devices.

Certain operations are described herein as occurring in the microcontroller internal to a flash memory device. Such description is intended to include operations which may be performed by hardware which may be associated with the microcontroller such as peripheral hardware on a chip on which the microcontroller may reside. It is also appreciated that some or all of these operations, in any embodiment, may alternatively be performed by the external, host-flash memory device interface controller including operations which may be performed by hardware which may be associated with the interface controller such as peripheral hardware on a chip on which the interface controller may reside. Finally it is appreciated that the internal and external controllers may each physically reside on a single hardware device, or alternatively on several operatively associated hardware devices.

Any data described as being stored at a specific location in memory may alternatively be stored elsewhere, in conjunction with an indication of the location in memory with which the data is associated. For example, instead of storing page- or erase-sector-specific information within a specific page or erase sector, the same may be

stored within the flash memory device's internal microcontroller or within a microcontroller interfacing between the flash memory device and the host, and an indication may be stored of the specific page or erase sector associated with the cells.

It is appreciated that the teachings of the present invention can, for example, be implemented by suitably modifying, or interfacing externally with, flash controlling apparatus. The flash controlling apparatus controls a flash memory array and may comprise either a controller external to the flash array or a microcontroller on-board the flash array or otherwise incorporated therewithin. Examples of flash memory arrays include Samsung's K9XXG08UXM series, Hynix's HY27UK08BGFM Series, Micron's MT29F64G08TAAWP or other arrays such as but not limited to NOR or phase change memory. Examples of controllers which are external to the flash array they control include STMicroelectrocincs's ST7265x microcontroller family, STMicroelectrocincs's ST72681 microcontroller, and SMSCs USB97C242, Traspan Technologies' TS-481 1, Chipsbank CBM2090/CBM1 190. Examples of commercial IP software for Flash file systems are: Denali's Spectra™ NAND Flash File System, Aarsan's NAND Flash Controller IP Core and Arasan's NAND Flash File System. It is appreciated that the flash controller apparatus need not be NAND-type and can alternatively, for example, be NOR-type or phase change memory-type.

Flash controlling apparatus, whether external or internal to the controlled flash array, typically includes the following components: a Memory Management/File system, a NAND interface (or other flash memory array interface), a Host Interface (USB, SD or other), error correction circuitry (ECC) typically comprising an Encoder and matching decoder, and a control system managing all of the above.

The present invention may for example interface with or modify, as per any of the embodiments described herein, one, some or all of the above components and particularly with the ECC component.

It is appreciated that software components of the present invention including programs and data may, if desired, be implemented in ROM (read only memory) form including CD-ROMs, EPROMs and EEPROMs, or may be stored in any other suitable computer-readable medium such as but not limited to disks of various kinds, cards of

various kinds and RAMs. Components described herein as software may, alternatively, be implemented wholly or partly in hardware, if desired, using conventional techniques.

Included in the scope of the present invention, inter alia, are electromagnetic signals carrying computer-readable instructions for performing any or all of the steps of any of the methods shown and described herein, in any suitable order; machine-readable instructions for performing any or all of the steps of any of the methods shown and described herein, in any suitable order; program storage devices readable by machine, tangibly embodying a program of instructions executable by the machine to perform any or all of the steps of any of the methods shown and described herein, in any suitable order; a computer program product comprising a computer useable medium having computer readable program code having embodied therein, and/or including computer readable program code for performing, any or all of the steps of any of the methods shown and described herein, in any suitable order; any technical effects brought about by any or all of the steps of any of the methods shown and described herein, when performed in any suitable order; any suitable apparatus or device or combination of such, programmed to perform, alone or in combination, any or all of the steps of any of the methods shown and described herein, in any suitable order; information storage devices or physical records, such as disks or hard drives, causing a computer or other device to be configured so as to carry out any or all of the steps of any of the methods shown and described herein, in any suitable order; a program pre-stored e.g. in memory or on an information network such as the Internet, before or after being downloaded, which embodies any or all of the steps of any of the methods shown and described herein, in any suitable order, and the method of uploading or downloading such, and a system including server/s and/or client/s for using such; and hardware which performs any or all of the steps of any of the methods shown and described herein, in any suitable order, either alone or in conjunction with software.

Features of the present invention which are described in the context of separate embodiments may also be provided in combination in a single embodiment. Conversely, features of the invention, including method steps, which are described for brevity in the context of a single embodiment or in a certain order may be provided separately or in any suitable subcombination or in a different order. "e.g." is used herein in the sense of a specific example which is not intended to be limiting.

CLAIMS

1. A low power Chien searching method employing Chien search circuitry comprising at least two hardware components that compute at least two corresponding bits comprising a Chien search output, the method comprising:

activating only a subset of the hardware components thereby to compute only a subset of the bits of the Chien search output; and

only if a criterion on the subset of the bits of the Chien search output is satisfied, activating hardware components other than those in said subset of hardware components, to compute additional bits of said Chien search output other than the bits in said subset of bits.

2. A method according to claim 1 wherein said activating-only-if comprises activating all hardware components outside of said subset of said plurality of hardware components, if the Chien criterion is not satisfied by said subset of bits.

3. A method according to claim 1 wherein said hardware components evaluate at least one error locator polynomial.

4. A method according to claim 3 and also comprising using the roots of the error locator polynomial to determine locations of errors in a recovered version of sequence of externally provided bits.

5. A method according to claim 4 wherein said sequence of externally provided bits comprises data provided by a host and said recovered version comprises a representation of said data stored in flash memory.

6. A method according to claim 4 wherein said data stored in flash memory comprises data encoded in accordance with a Reed-Solomon decoding algorithm.

7. A method according to claim 4 wherein said data stored in flash memory comprises data encoded in accordance with a BCH decoding algorithm.

8. A method according to claim 5 and further comprising correcting said errors to reproduce said data provided by the host.

9. A method for correcting a plurality of errors occurring at a corresponding plurality of locations within a recovered version of data provided by a host, said recovered version having been stored in memory, the method comprising:

constructing a polynomial characterized in that roots thereof indicate locations of the errors in the recovered version of the data; and

at least once, determining whether a value of the polynomial equals zero, wherein the value of the polynomial comprises a summation of a sequence of at least two bits, wherein at least once, said determining comprises determining whether each bit in only a subsequence of said sequence of bits equals zero; and subsequently determining whether at least some of the bits in said sequence of bits, other than in said subsequence of bits, equal zero, only if all bits in said subsequence equal zero.

10. A method according to claim 9 and further comprising correcting said errors to reproduce said data provided by the host.

11. Apparatus for finding roots of a polynomial defined over a finite field, said roots configured to represent location of errors within a recovered version of data, the apparatus comprising:

polynomial root finding apparatus operatively configured to find roots of a polynomial which is a weighted sum of powers of a variable, said weighted sum being defined by a variable and by a sequence of coefficients by which said powers of said variable are respectively multiplied, the polynomial having a value given an individual sequence of coefficients and given an individual value for said variable,

said polynomial root finding apparatus comprising polynomial value determination apparatus operative to determine, for at least one given individual sequence of coefficients and individual value for said variable, whether the value of the polynomial, given said individual sequence of coefficients and said individual value for said variable, equals zero, wherein the value of the polynomial comprises a sequence of at least two bits;

said polynomial value determination apparatus comprising:

partial polynomial value determination apparatus operative to determine whether each bit in only a subsequence of said sequence of bits equals zero; and

selectively activatable complementary polynomial value determination apparatus operative to determine whether at least some of the bits in said sequence of bits other than in said subsequence equal zero, only if all bits in said subsequence equal zero.

12. Apparatus according to claim 11 wherein said partial polynomial value determination apparatus comprises a multiplier which is always active and said selectable activatable apparatus is activated only if an Error Locator Polynomial evaluation of the bits in said subsequence is equal to 1.

13. Apparatus according to claim 12 and also comprising a register upstream of the selectable activatable apparatus.

14. Apparatus according to claim 11 wherein the bits included in said subsequence of bits comprise the first, lower bits in said sequence.

15. Apparatus according to claim 11 wherein the bits included in said subsequence of bits are non-consecutive in said sequence.

16. A method according to claim 1 wherein said hardware components are operative for finding roots of a polynomial which is a weighted sum of powers of a variable, said weighted sum being defined by a variable and by a sequence of coefficients by which said powers of said variable are respectively multiplied, the polynomial having a value given an individual sequence of coefficients and given an individual value for said variable, said value comprising said Chien search output, said finding including determining, for at least one given individual sequence of coefficients and individual value for said variable, whether the value of the polynomial, given said individual sequence of coefficients and said individual value for said variable, equals zero, wherein the value of the polynomial comprises a sequence of at least two bits.

17. A method according to claim 16 wherein said criterion is whether each of the bits in said subset of bits equals zero.

18. A method according to claim 9 wherein said subsequently determining comprises determining whether all of the bits in said sequence of bits, other than in said subsequence of bits, equal zero, only if all bits in said subsequence equal zero.

19. A method according to claim 9 wherein said subsequently determining comprises:

determining whether only some of the bits in said sequence of bits, other than in said subsequence of bits, equal zero, only if all bits in said subsequence equal zero; and

subsequently determining whether at least some of the bits in said sequence of bits, other than said some bits and said bits in said subsequence of bits, equal zero, only if all of said some bits equal zero and all of said bits in said subsequence equal zero.

20. A low power Chien searching system employing Chien search circuitry comprising at least two hardware components that compute at least two corresponding bits comprising a Chien search output, the system comprising:

subset activation apparatus operative to activate only a subset of the hardware components thereby to compute only a subset of the bits of the Chien search output; and

polynomial evaluation completion activation apparatus operative, only if a criterion on the subset of the bits of the Chien search output is satisfied, to activate hardware components other than those in said subset of hardware components, to compute additional bits of said Chien search output other than the bits in said subset of bits.

21. A method according to claim 1 wherein said subset of bits comprises a number of bits c which minimizes the power used by said hardware components to perform said activating step and said only-if-activating step.

22. A method for saving power consumed by hardware components, said hardware components operatively configured to perform a Chien search, said method comprising:

providing said hardware components; and

initiating said Chien search utilizing only a selective subset of the hardware components whereby power consumption is lower compared with power consumed in a Chien search utilizing all of said hardware components.

23. A method according to claim 22 wherein said Chien search is utilized to determine locations of errors in a recovered version of sequence of externally provided bits.

24. A method according to claim 23 wherein said sequence of externally provided bits comprises data provided by a host and said recovered version comprises a representation of said data stored in flash memory.

25. An error correction decoder comprising:

an error locator polynomial generator operative to generate at least one error locator polynomial; and

an error locator polynomial evaluator operative to rule out at least one root of the error locator polynomial based on only a partial evaluation thereof.

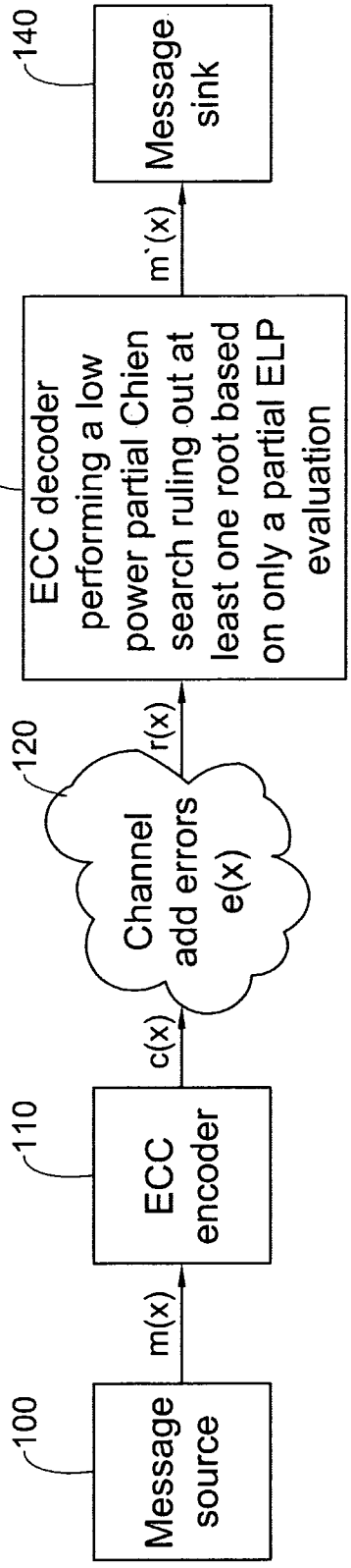

100
Message source
m(x)
110
ECC encoder
c(x)
120
Channel add errors e(x)
r(x)
130
ECC decoder performing a low power partial Chien search ruling out at least one root based on only a partial ELP evaluation
m`(x)
140
Message sink

FIG. 1A

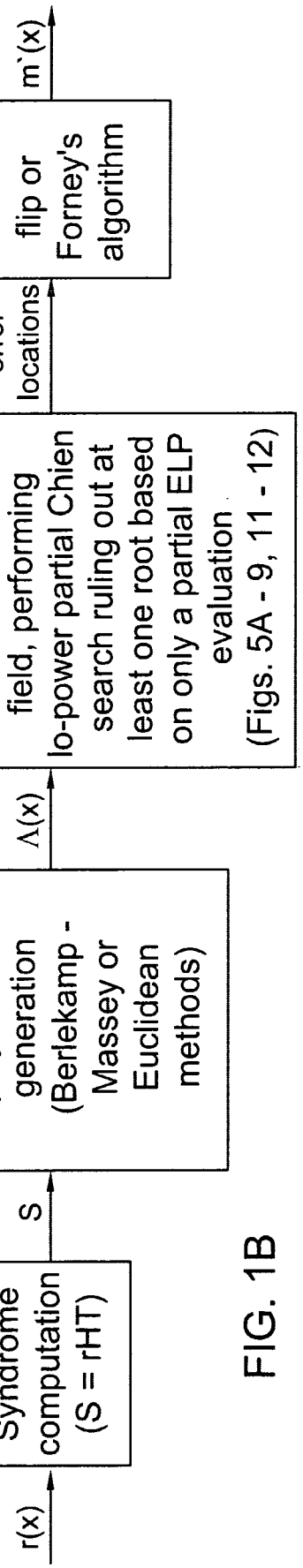
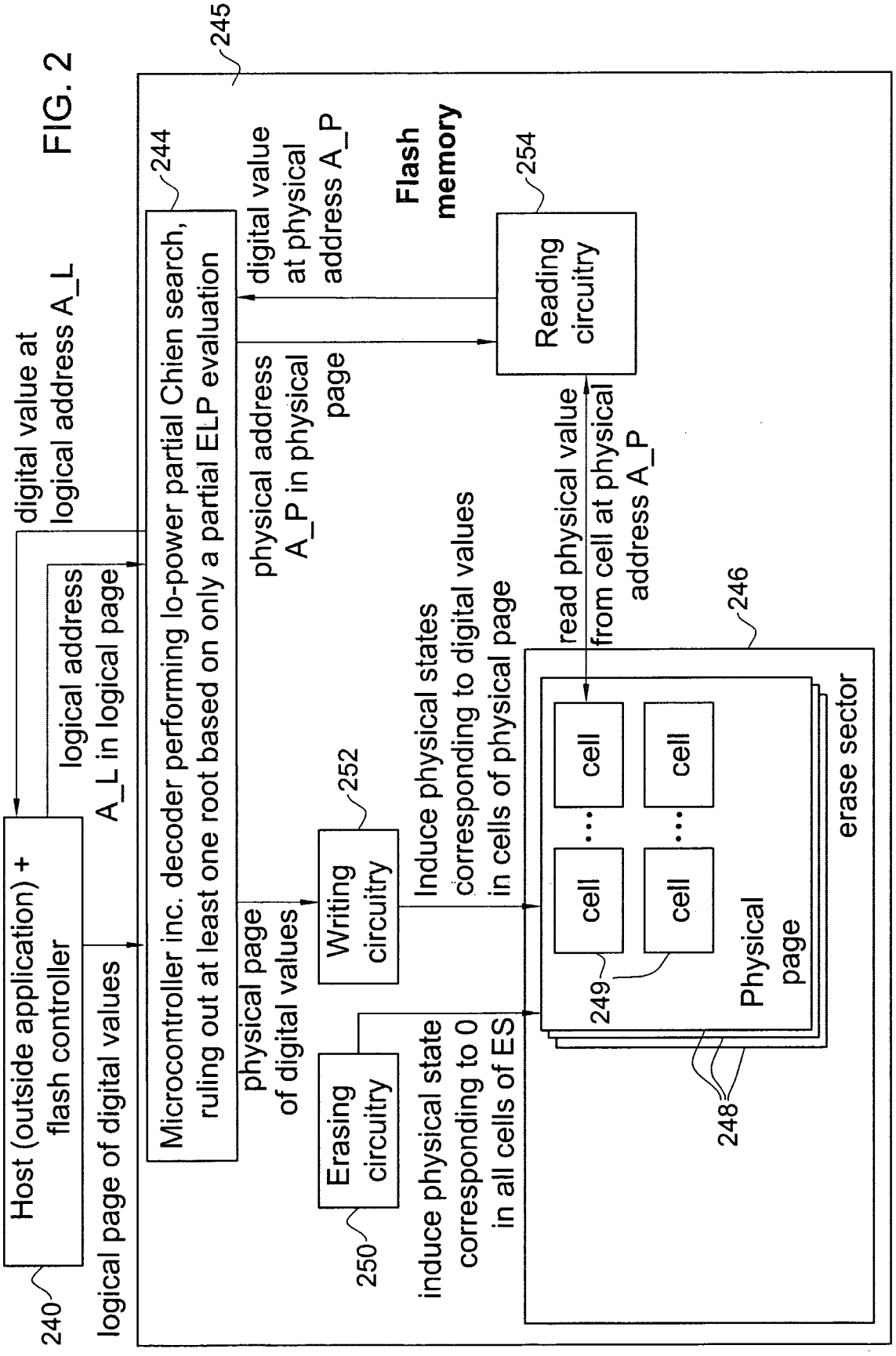

r(x)
200
Syndrome computation (S = rHT)
S
210
Error locator polynomial generation (Berlekamp - Massey or Euclidean methods)
Λ(x)
220
Error locator polynomial evaluation for all elements in field, performing lo-power partial Chien search ruling out at least one root based on only a partial ELP evaluation (Figs. 5A - 9, 11 - 12)
error locations
230
Error correction: flip or Forney's algorithm
m`(x)

FIG. 1B

FIG. 2


245
Flash memory
240
Host (outside application) + flash controller
digital value at logical address A_L
logical address A_L in logical page
logical page of digital values
244
Microcontroller inc. decoder performing lo-power partial Chien search, ruling out at least one root based on only a partial ELP evaluation
digital value at physical address A_P
physical address A_P in physical page
physical page of digital values
252
Writing circuitry
Induce physical states corresponding to digital values in cells of physical page
250
Erasing circuitry
induce physical state corresponding to 0 in all cells of ES
254
Reading circuitry
read physical value from cell at physical address A_P
246
cell
249
Physical page
248
erase sector

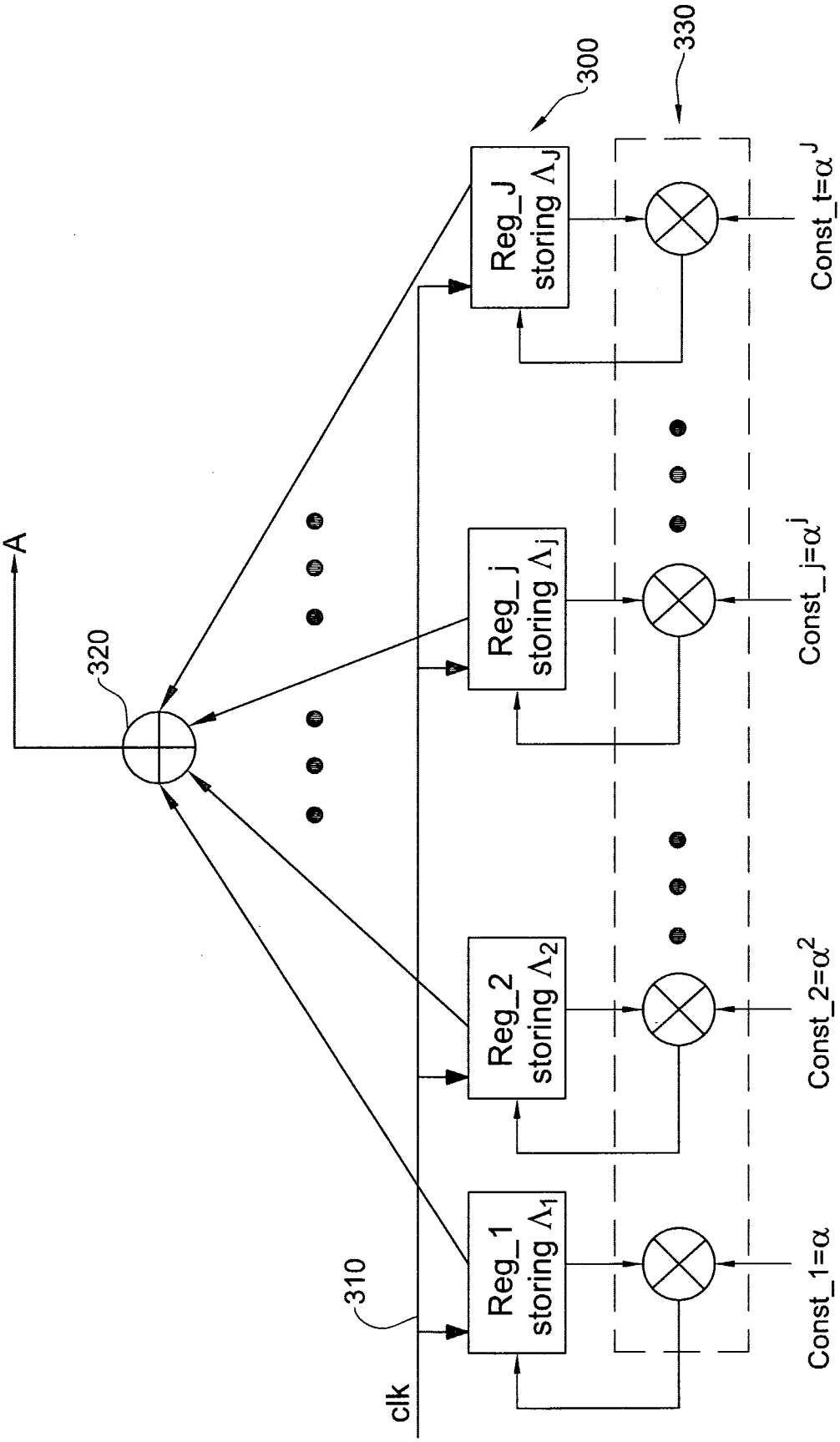


FIG. 3 (PRIOR ART)

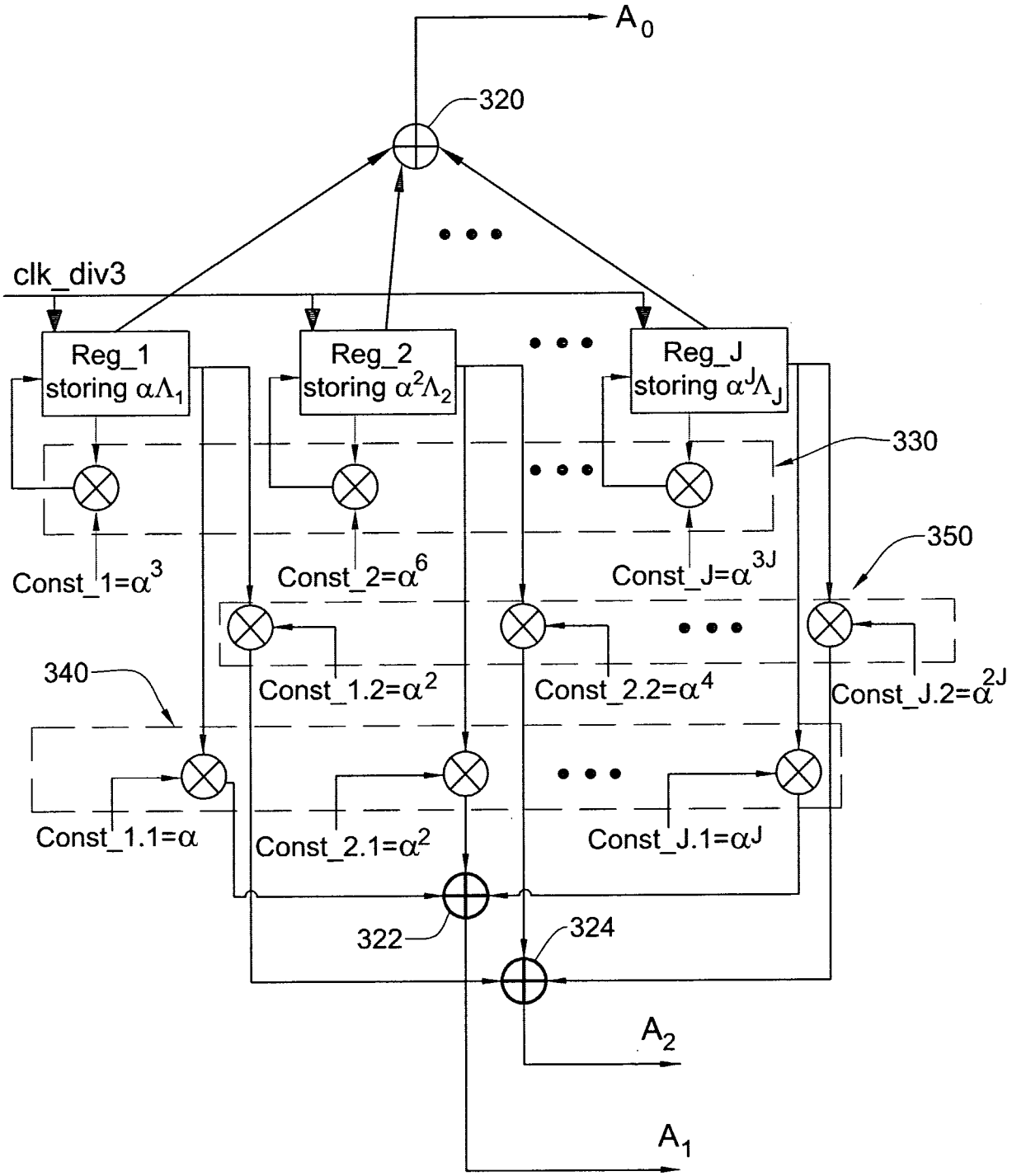
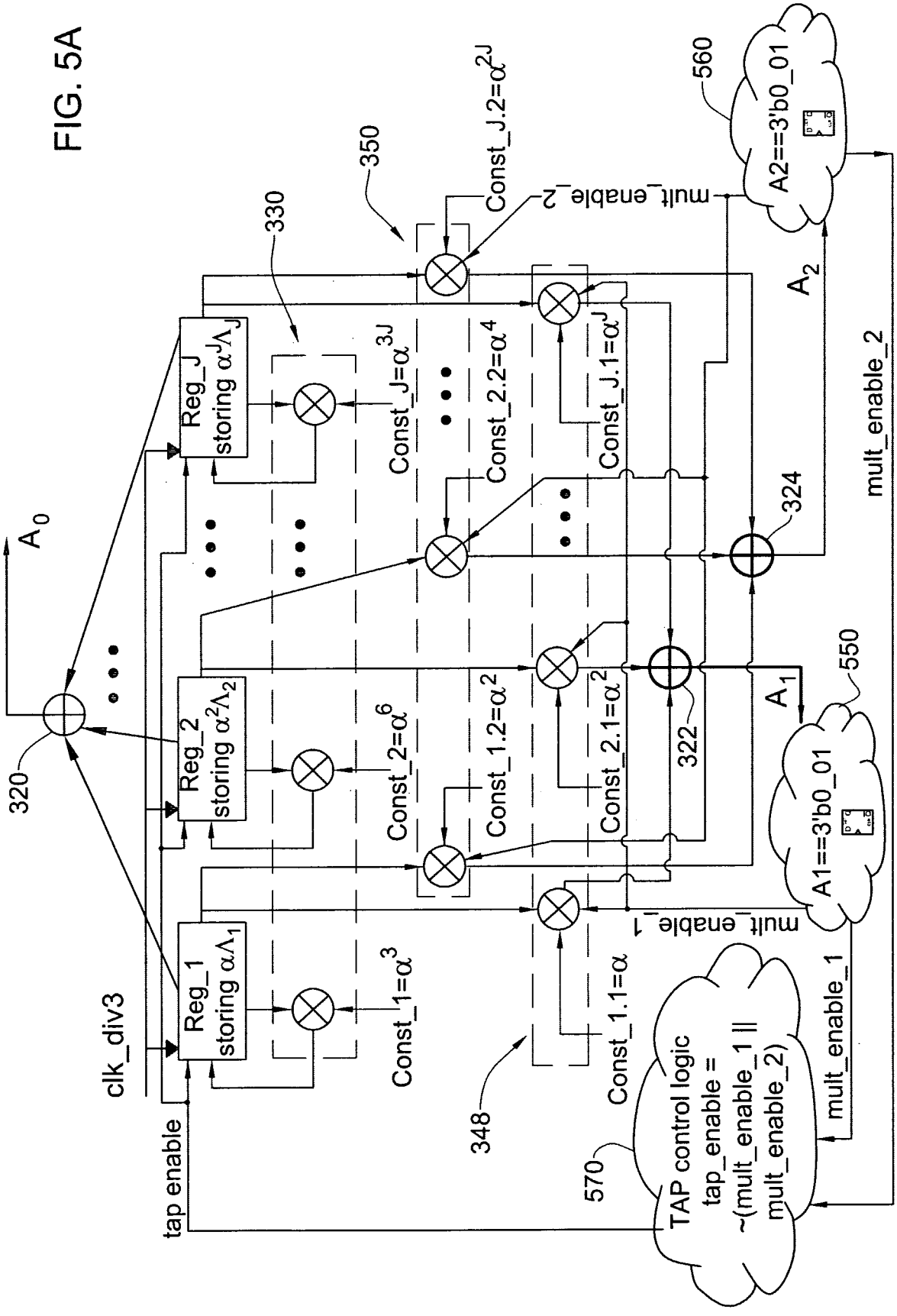


FIG. 4 (PRIOR ART)

FIG. 5A


A0
320
clk_div3
tap enable
Reg_1 storing αΛ1
Reg_2 storing α2Λ2
Reg_J storing αJΛJ
330
Const_1=α3
Const_2=α6
Const_J=α3J
348
Const_1.1=α
Const_1.2=α2
Const_2.1=α2
Const_J.1=αJ
Const_2.2=α4
Const_J.2=α2J
350
mult_enable_1
mult_enable_2
322
324
A1
A2
550
560
A1==3'b0_01
A2==3'b0_01
570
TAP control logic
tap_enable =
~(mult_enable_1 ||
mult_enable_2)

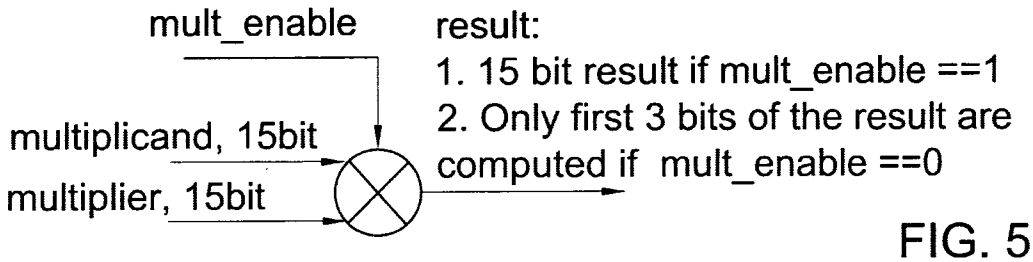


FIG. 5B

Parameter	Description
P_{reg}	Power drawn by the register of each tap.
P_{mult}	Power drawn by a full precision multiplier (all m sub-elements)
P_{comb}	Power drawn by the combinatorial logic.
$1/q^c$	In 1 out of every q^c cycles on average the polynomial evaluation of c sub-elements equals to 1 even though the evaluation of all sub-elements of the polynomial might not be 1, involving recomputation of all sub-elements

FIG. 6

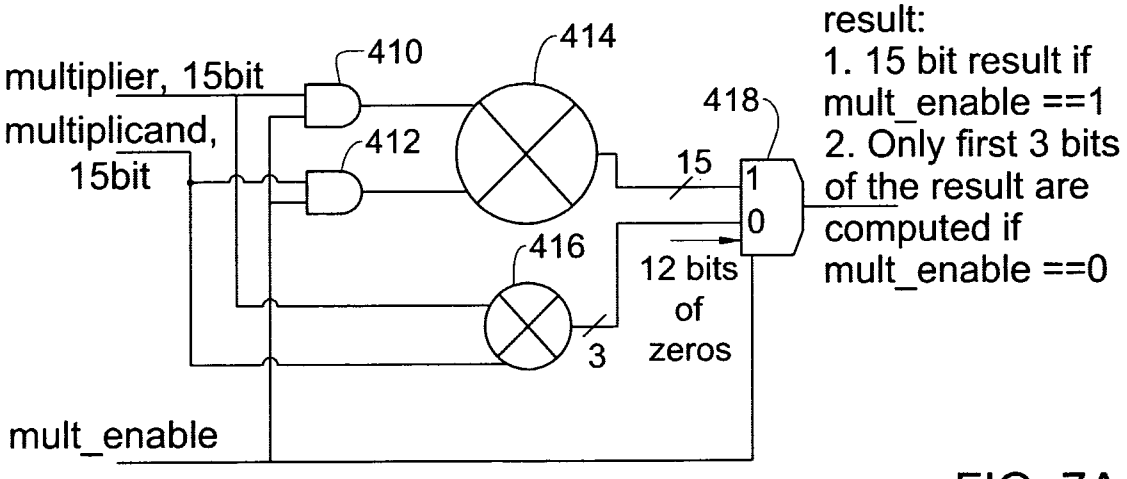


FIG. 7A

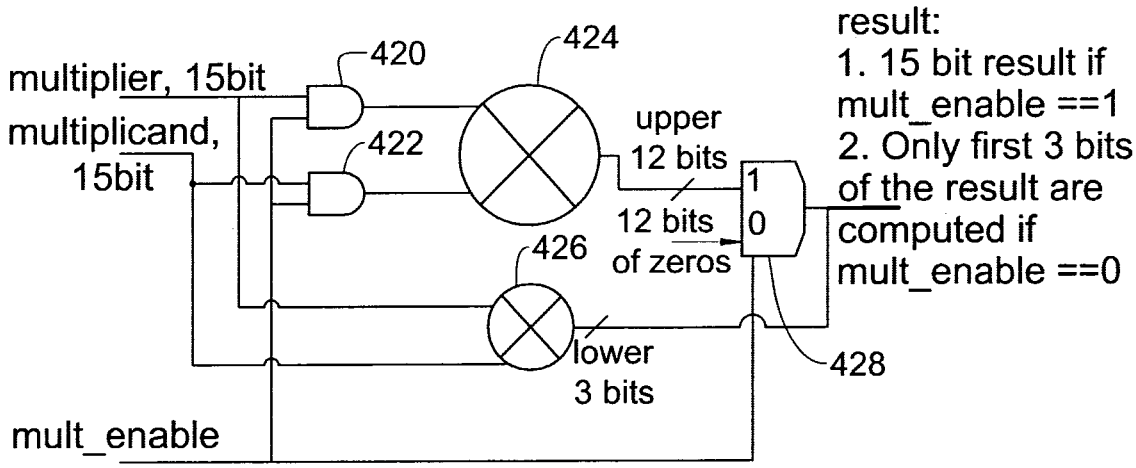

multiplier, 15bit
multiplicand, 15bit
mult_enable
420
422
424
426
428
upper 12 bits
12 bits of zeros
lower 3 bits
1
0
result:
1. 15 bit result if mult_enable ==1
2. Only first 3 bits of the result are computed if mult_enable ==0

FIG. 7B

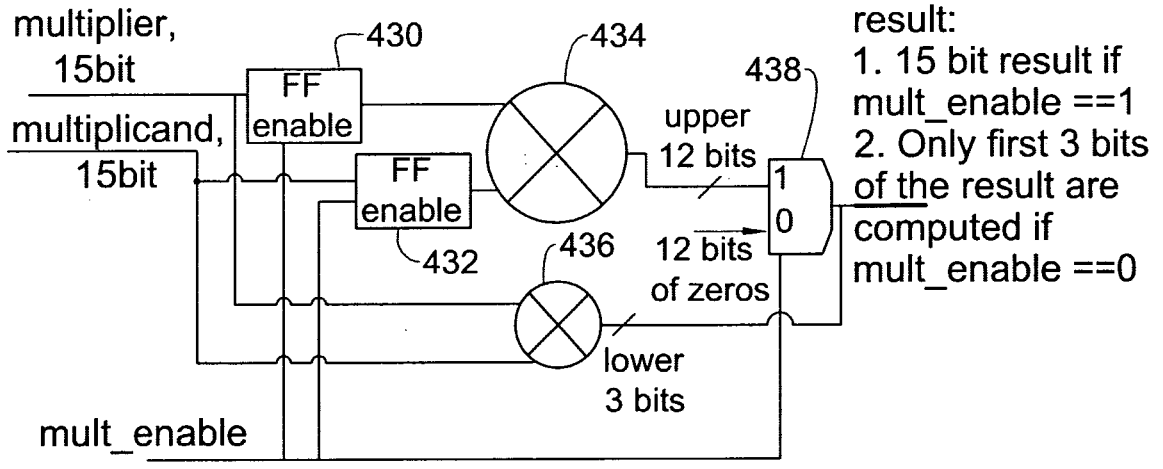

multiplier, 15bit
multiplicand, 15bit
mult_enable
FF enable
FF enable
430
432
434
436
438
upper 12 bits
12 bits of zeros
lower 3 bits
1
0
result:
1. 15 bit result if mult_enable ==1
2. Only first 3 bits of the result are computed if mult_enable ==0

FIG. 7C

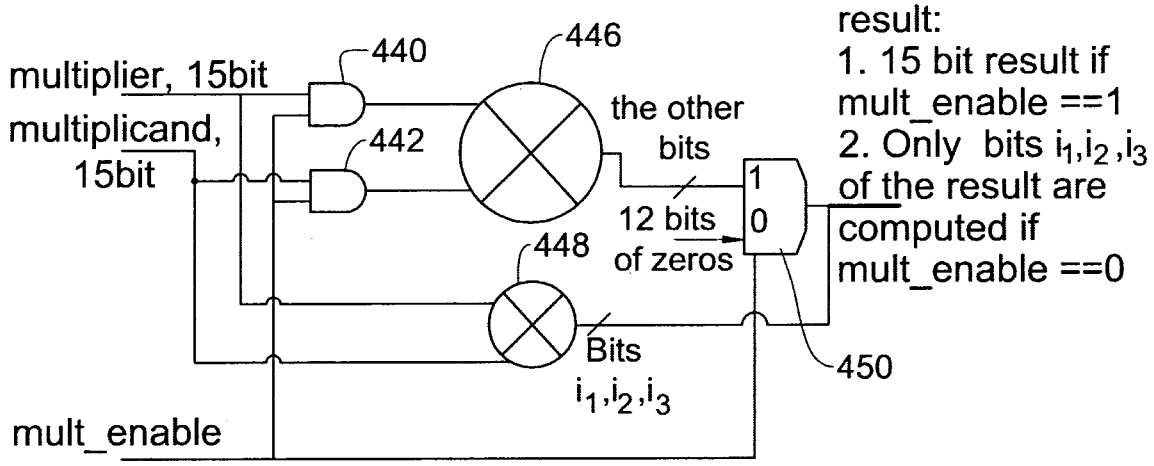

multiplier, 15bit
multiplicand, 15bit
mult_enable
440
442
446
448
450
the other bits
12 bits of zeros
Bits i_1,i_2,i_3
1
0
result:
1. 15 bit result if mult_enable ==1
2. Only bits i_1,i_2,i_3 of the result are computed if mult_enable ==0

FIG. 7D

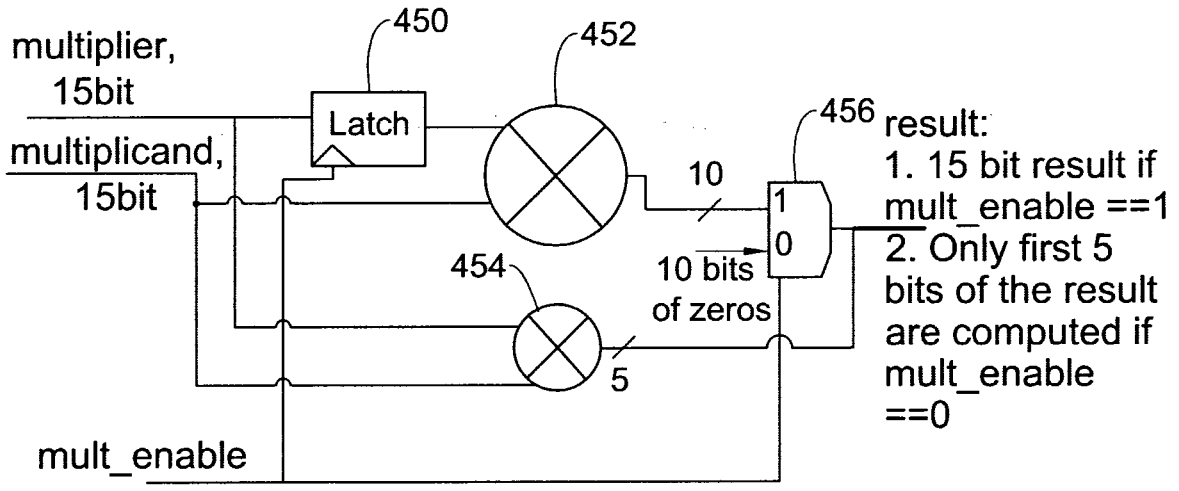
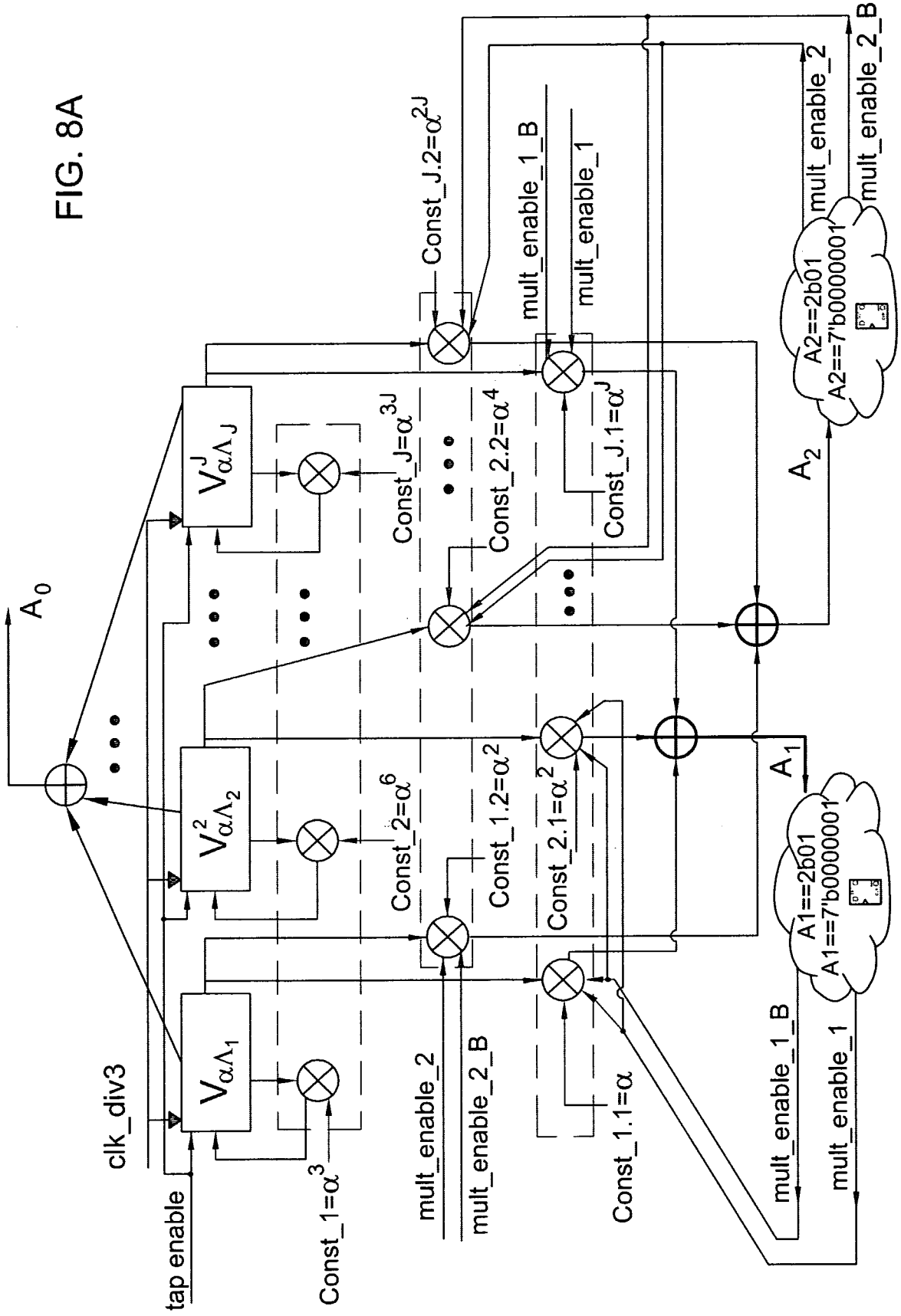

multiplier, 15bit
multiplicand, 15bit
mult_enable
Latch
450
452
454
456
10
10 bits of zeros
5
1
0
result:
1. 15 bit result if mult_enable ==1
2. Only first 5 bits of the result are computed if mult_enable ==0

FIG. 7E

FIG. 8A


A0
clk_div3
tap enable
Const_1=α^3
mult_enable_2
mult_enable_2_B
Const_2=α^6
Const_1.2=α^2
Const_2.1=α^2
Const_1.1=α
mult_enable_1_B
mult_enable_1
A1==2'b01
A1==7'b00000001
A1
Const_J=α^3J
Const_2.2=α^4
Const_J.1=α^J
Const_J.2=α^2J
mult_enable_1_B
mult_enable_1
A2==2'b01
A2==7'b00000001
A2
mult_enable_2
mult_enable_2_B

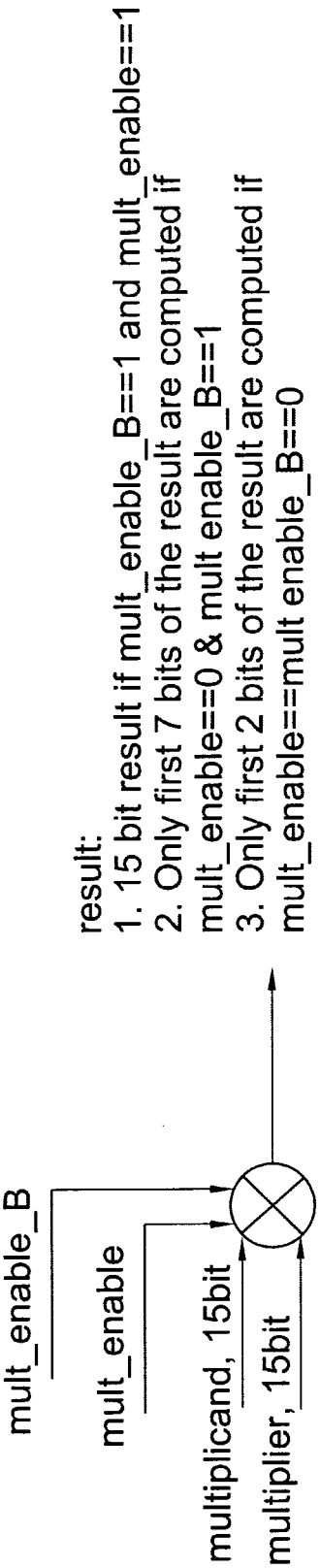

mult_enable_B
mult_enable
multiplicand, 15bit
multiplier, 15bit
result:
1. 15 bit result if mult_enable_B==1 and mult_enable==1
2. Only first 7 bits of the result are computed if
mult_enable==0 & mult enable_B==1
3. Only first 2 bits of the result are computed if
mult_enable==mult enable_B==0

FIG. 8B

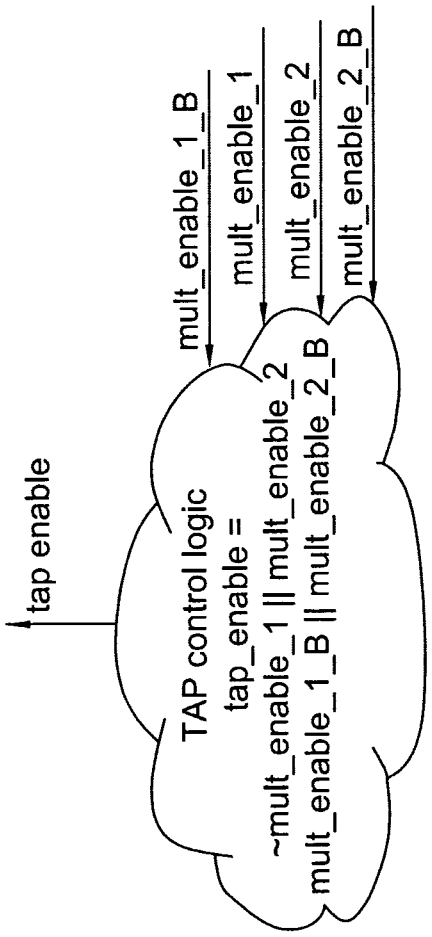

tap enable
TAP control logic
tap_enable =
~mult_enable_1 || mult_enable_2
mult_enable_1_B || mult_enable_2_B
mult_enable_1_B
mult_enable_1
mult_enable_2
mult_enable_2_B

FIG. 8C

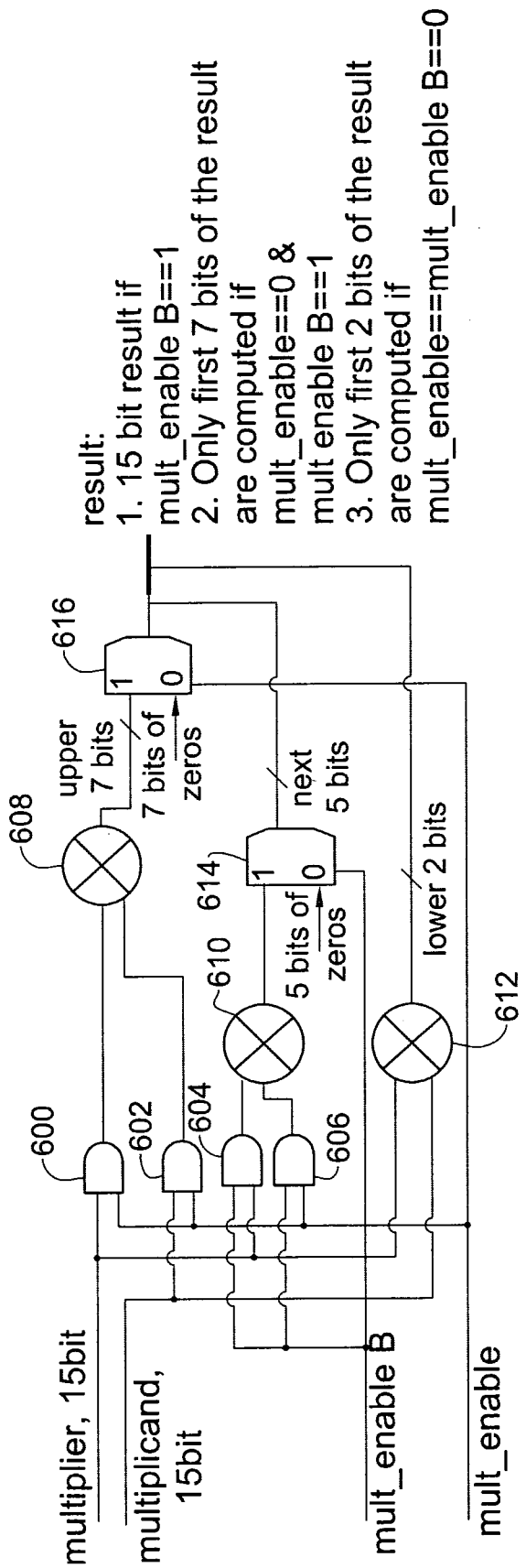

result:
1. 15 bit result if
mult_enable B==1
2. Only first 7 bits of the result
are computed if
mult_enable==0 &
mult enable B==1
3. Only first 2 bits of the result
are computed if
mult_enable==mult_enable B==0
616
608
upper
7 bits
7 bits of
zeros
614
next
5 bits
610
5 bits of
zeros
612
lower 2 bits
600
602
604
606
multiplier, 15bit
multiplicand,
15bit
mult_enable B
mult_enable

FIG. 9

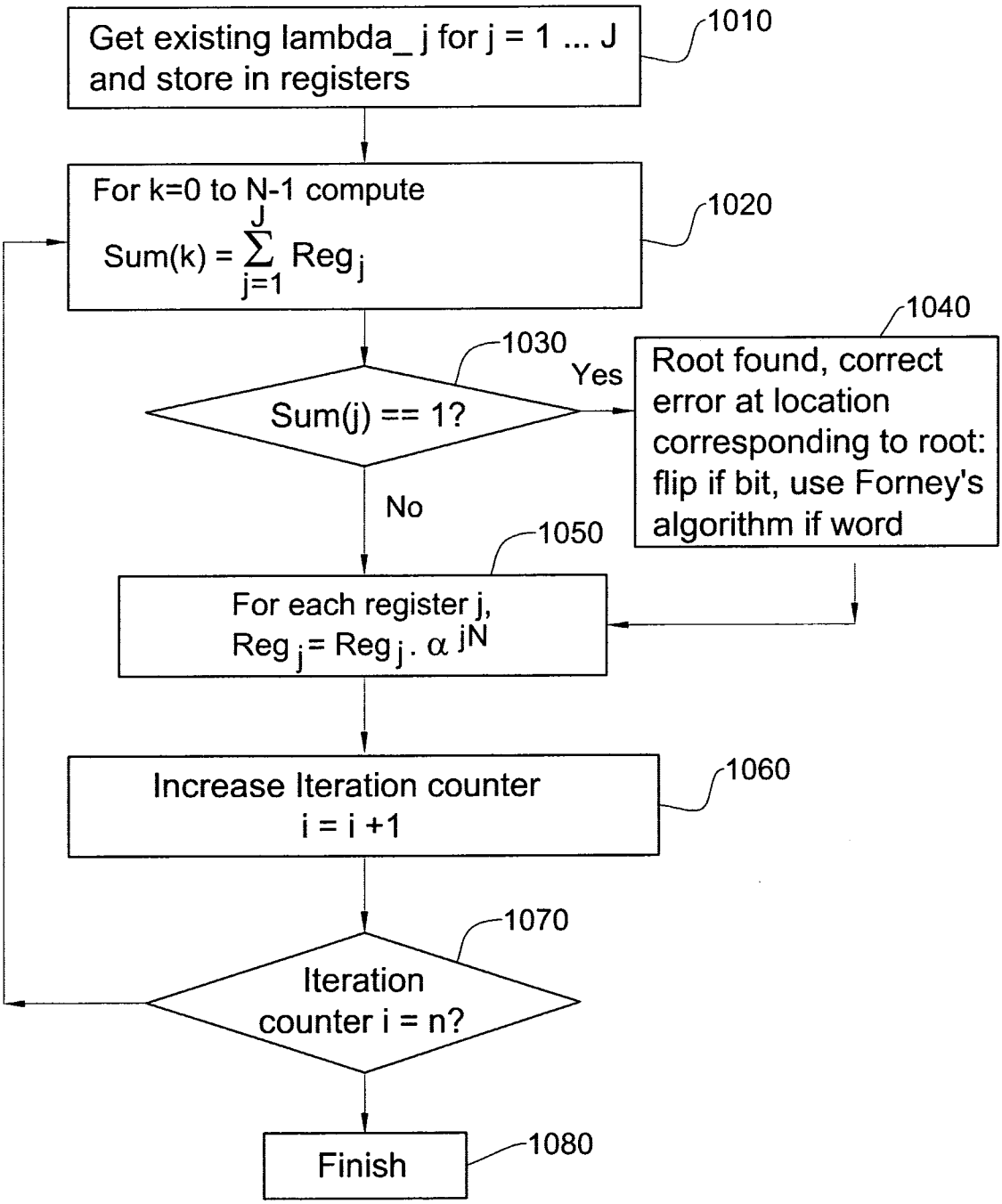

Get existing lambda_ j for j = 1 ... J and store in registers
1010
For k=0 to N-1 compute
Sum(k) = $\sum_{j=1}^{J} Reg_j$
1020
1030
Sum(j) == 1?
Yes
1040
Root found, correct error at location corresponding to root: flip if bit, use Forney's algorithm if word
No
1050
For each register j,
$Reg_j = Reg_j . \alpha^{jN}$
Increase Iteration counter
i = i +1
1060
1070
Iteration counter i = n?
Finish
1080

FIG. 10

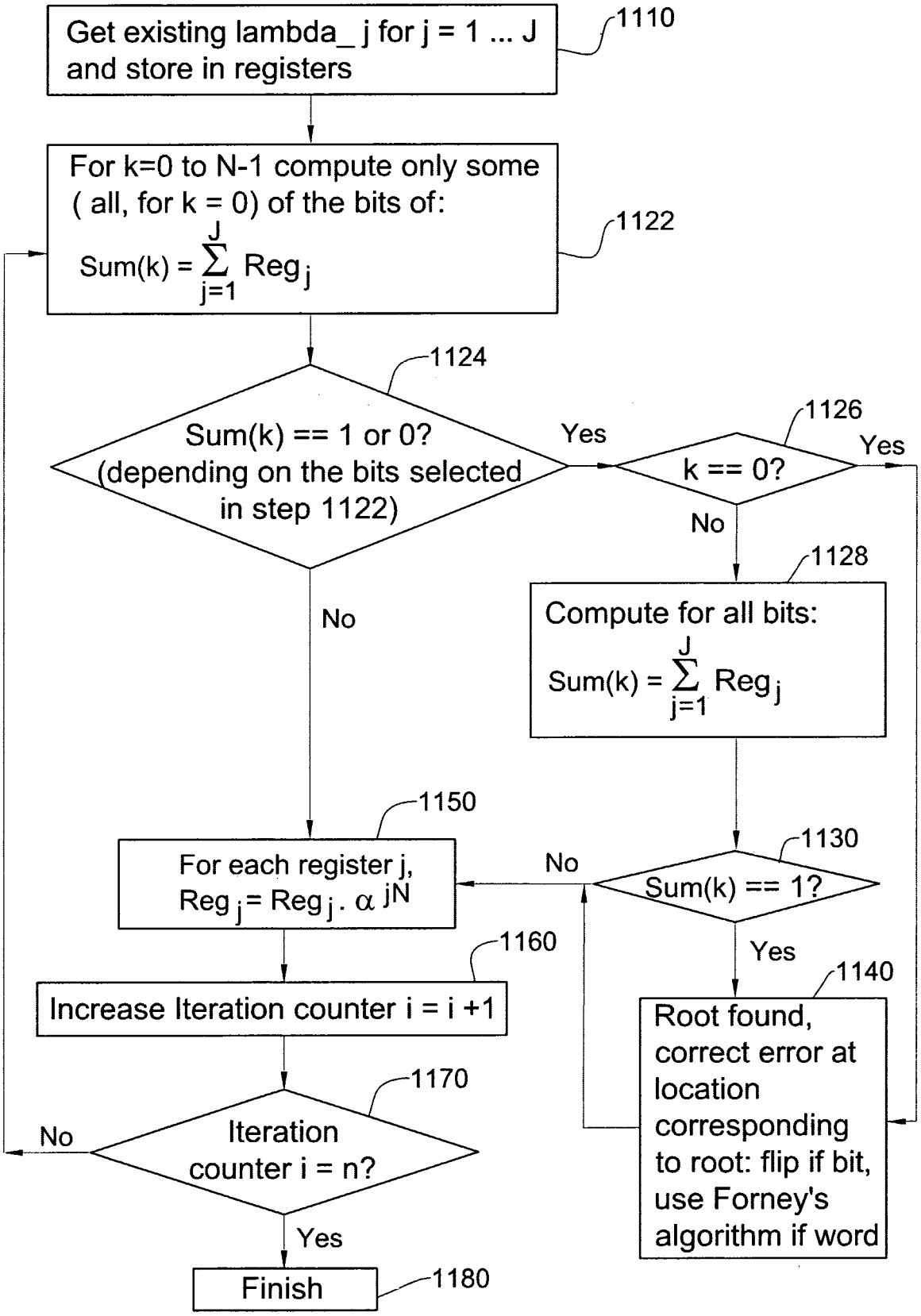

Get existing lambda_ j for j = 1 ... J and store in registers
1110
For k=0 to N-1 compute only some (all, for k = 0) of the bits of:
Sum(k) = $\sum_{j=1}^{J}$ Reg$_j$
1122
1124
Sum(k) == 1 or 0? (depending on the bits selected in step 1122)
Yes
1126
k == 0?
Yes
No
1128
Compute for all bits:
Sum(k) = $\sum_{j=1}^{J}$ Reg$_j$
No
1150
For each register j, Reg$_j$ = Reg$_j$. α^{jN}
No
1130
Sum(k) == 1?
Yes
1160
Increase Iteration counter i = i +1
1140
Root found, correct error at location corresponding to root: flip if bit, use Forney's algorithm if word
1170
No
Iteration counter i = n?
Yes
Finish
1180

FIG. 11

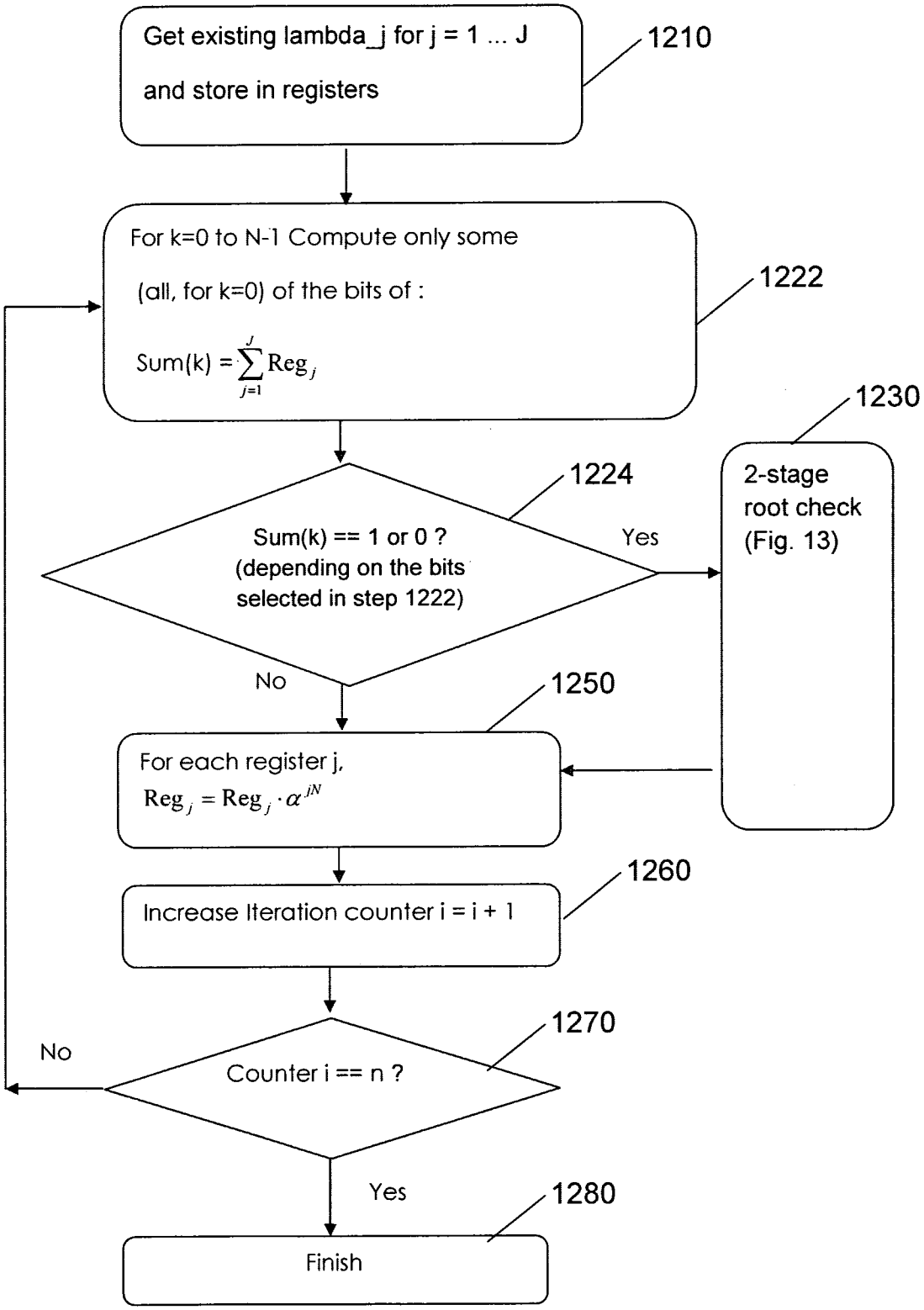

Get existing lambda_j for j = 1 ... J
and store in registers
1210
For k=0 to N-1 Compute only some
(all, for k=0) of the bits of :
$Sum(k) = \sum_{j=1}^{J} Reg_j$
1222
1230
2-stage
root check
(Fig. 13)
1224
Sum(k) == 1 or 0 ?
(depending on the bits
selected in step 1222)
Yes
No
1250
For each register j,
$Reg_j = Reg_j \cdot \alpha^{jN}$
1260
Increase Iteration counter i = i + 1
1270
Counter i == n ?
No
Yes
1280
Finish

FIG. 12

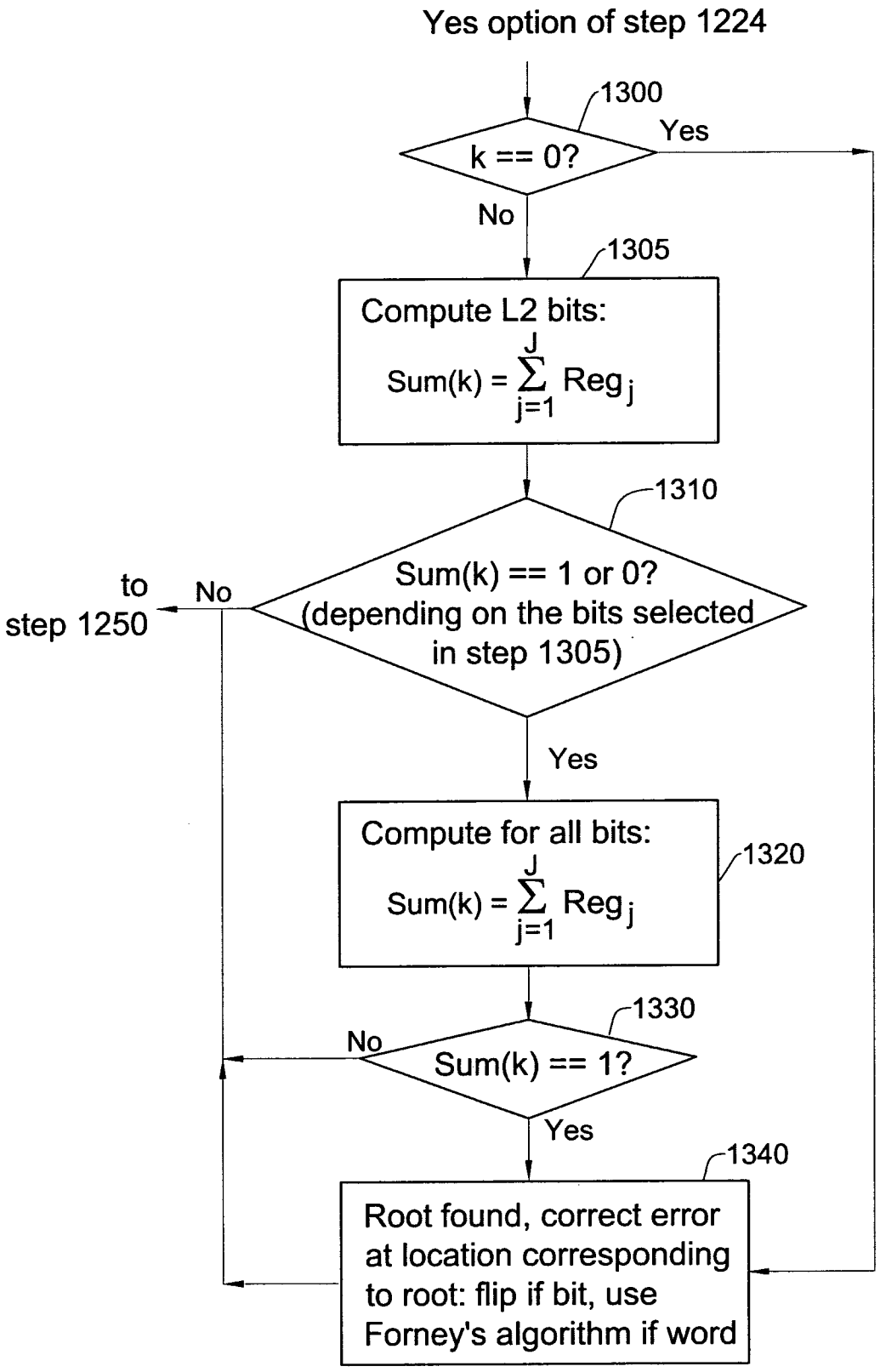

Yes option of step 1224
1300
k == 0?
Yes
No
1305
Compute L2 bits:
Sum(k) = $\sum_{j=1}^{J}$ Reg$_j$
1310
Sum(k) == 1 or 0?
(depending on the bits selected in step 1305)
No
to step 1250
Yes
Compute for all bits:
Sum(k) = $\sum_{j=1}^{J}$ Reg$_j$
1320
1330
Sum(k) == 1?
No
Yes
1340
Root found, correct error at location corresponding to root: flip if bit, use Forney's algorithm if word

FIG. 13