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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 July 2009 (09.07.2009)

PCT

(10) International Publication Number
WO 2009/085489 A2

(51) International Patent Classification:
G06F 9/46 (2006.01) *G06F 11/00* (2006.01)
G06F 13/10 (2006.01)

(21) International Application Number:
PCT/US2008/084571

(22) International Filing Date:
24 November 2008 (24.11.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/963,147 21 December 2007 (21.12.2007) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **GOPAL, Vindoh** [IN/US]; 15 West End Ave, Westborough, Massachusetts 01581 (US). **OZTURK, Erdinc** [TR/US]; 120 Elm Street, Apt. 1, Worcester, Massachusetts 01609 (US). **WOLRICH, Gilbert** [US/US]; 4 Cider Mill Road, Framingham, Massachusetts 01701 (US). **FEGHALI, Wajdi** [CA/US]; 199 Massachusetts Ave, Boston, Massachusetts 02115 (US).

(74) Agents: **VINCENT, Lester J.** et al.; Blakely Sokoloff Taylor & Zafman, 1279 Oakmead Parkway, Sunnyvale, California 94085 (US).

(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,

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR EFFICIENT PROGRAMMABLE CYCLIC REDUNDANCY CHECK (CRC)

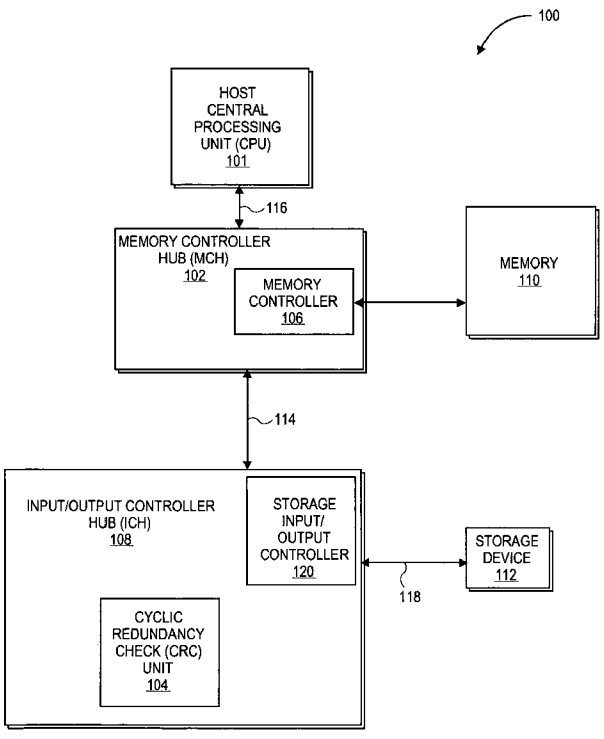

FIG. 1

(57) Abstract: A method and apparatus to optimize each of the plurality of reduction stages in a Cyclic Redundancy Check (CRC) circuit to produce a residue for a block of data decreases area used to perform the reduction while maintaining the same delay through the plurality of stages of the reduction logic. A hybrid mix of Karatsuba algorithm, classical multiplications and serial division in various stages in the CRC reduction circuit results in about a twenty percent reduction in area on the average with no decrease in critical path delay.

WO 2009/085489 A2

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*

METHOD AND APPARATUS FOR EFFICIENT PROGRAMMABLE CYCLIC REDUNDANCY CHECK (CRC)

FIELD

This disclosure relates to error checking and in particular to use of Cyclic Redundancy Check for error checking.

BACKGROUND

A polynomial is a mathematical expression of one or more algebraic terms, for example, "$a + bx + cx^2$", each of which consists of a constant (a, b or c) multiplied by one or more variables (x) raised to a nonnegative integral power. A fixed size remainder of binary division of a data block by an n-bit polynomial may be used to verify that there were no transmission errors when transmitting the data block from a source to a destination. An n-bit polynomial applied to a data block of arbitrary length detects any single error burst that is less than or equal to n bits.

The fixed size remainder is computed for a data block at a source and is transmitted with the data block. The n-bit polynomial is applied to the received data block at the destination to re-compute the fixed size remainder and the recomputed fixed size remainder is compared with the fixed size remainder transmitted with the data block to verify that there were no transmission errors.

A Cyclic Redundancy Check (CRC) is a term that is given to both a fixed size remainder (a residue of binary division of an arbitrary length data block by a fixed size polynomial) and the function to produce the fixed size remainder. There are a plurality of n-bit polynomials that are used to compute a CRC. Most commonly used CRCs use the Galois finite field GF(2) having a finite field of two elements, 0 and 1.

BRIEF DESCRIPTION OF THE DRAWINGS

Features of embodiments of the claimed subject matter will become apparent as the following detailed description proceeds, and upon reference to the drawings, in which like numerals depict like parts, and in which:

Fig. 1 illustrates an embodiment of a plurality of reduction stages used to compute a residue from a 64-bit input and a 32-bit CRC;

Fig. 2 illustrates an embodiment of the CRC circuit shown in Fig. 1.

Fig. 3 illustrates an embodiment of the first three reduction stages shown in Fig. 2;

Fig. 4 illustrates an embodiment of a 8 x 32 multiplier that applies a one level application of Karatsuba (KA);

Fig. 5 illustrates an embodiment of a 16 x 32 multiplier that applies a one level application of Karatsuba (KA);

Fig. 6 illustrates an embodiment of a 16 x 32 multiplier that applies a two level application of Karatsuba (KA); and

Fig. 7 is a flowgraph illustrating an embodiment of a method for selecting an optimal multiplication algorithm to be used a reduction stage used to compute a CRC residue.

Although the following Detailed Description will proceed with reference being made to illustrative embodiments of the claimed subject matter, many alternatives, modifications, and variations thereof will be apparent to those skilled in the art. Accordingly, it is intended that the claimed subject matter be viewed broadly, and be defined only as set forth in the accompanying claims.

DETAILED DESCRIPTION

Fig. 1 is a block diagram of a system 100 that includes an embodiment of a Cyclic Redundancy Check (CRC) unit 104 that computes a fixed size remainder (a residue of binary division of an arbitrary length data block by a fixed size polynomial) according to the principles of the present invention.

The system 100 includes a processor 101, a Memory Controller Hub (MCH) 102 and an Input/Output (I/O) Controller Hub (ICH) 104. The MCH 102 includes a memory controller 106 that controls communication between the processor 101 and memory 110. The processor 101 and MCH 102 communicate over a system bus 116.

The processor 101 may be any one of a plurality of processors such as a single core Intel® Pentium IV ® processor, a single core Intel Celeron processor, an Intel® XScale processor or a multi-core processor such as Intel® Pentium D, Intel® Xeon® processor, or Intel® Core® Duo processor or any other type of processor.

The memory 110 may be Dynamic Random Access Memory (DRAM), Static Random Access Memory (SRAM), Synchronized Dynamic Random Access Memory (SDRAM), Double Data Rate 2 (DDR2) RAM or Rambus Dynamic Random Access Memory (RDRAM) or any other type of memory.

The ICH 104 may be coupled to the MCH 102 using a high speed chip-to-chip interconnect 114 such as Direct Media Interface (DMI). DMI supports 2 Gigabit/second

concurrent transfer rates via two unidirectional lanes. The ICH 104 includes the CRC unit 104. The ICH 104 may also include a storage I/O controller 120 for controlling communication with at least one storage device 112 coupled to the ICH 104. The storage device 112 may be, for example, a disk drive, Digital Video Disk (DVD) drive, Compact Disk (CD) drive, Redundant Array of Independent Disks (RAID), tape drive or other storage device. The ICH 104 may communicate with the storage device 112 over a storage protocol interconnect 118 using a serial storage protocol such as, Serial Attached Small Computer System Interface (SAS) or Serial Advanced Technology Attachment (SATA).

Fig. 2 illustrates an embodiment of the CRC circuit shown in Fig. 1. Computing a CRC requires calculating the remainder resulting from the division of the input data by a generator value. In the embodiment shown in Fig. 2, a CRC is computed for a 32-bit segment 220 of a data stream. The 32-bit segment 220 is shifted by 32-bits 124 and an XOR operation 128 is performed with the 32-bit segment 202 and any pre-existing residue (remainder) 222 which is also shifted by 32-bits 226. The XORed result (input data) and the k-bits of the respective pre-computed polynomials $g_i(x)$ are used to compute the CRC residue in stages 206a-206f Successive stages 206a-206e reduce input data by i-bits until a residue value is output by stage 206f. The residue is fed back 222 for use in processing the next 32-bit segment 220 of the data stream. The residue remaining after the final message portion 220 is applied is the CRC residue determined for the data stream as a whole. The CRC residue can either be appended to the data stream or compared with a received CRC value in the data stream to determine whether data corruption likely occurred.

Fig. 3 illustrates an embodiment of the first three reduction stages 206a, 206b, 206c shown in Fig. 2. Pre-computed (fixed value) polynomials g16, g8 and g4 are derived from the 32-bit CRC and are each stored in a respective one of three storage elements 302a, 302b, 302c. The pre-computed polynomials are multiples of the 32-bit CRC polynomial over Galois Field (GF)(2). The storage elements 302a,..,302c may be registers or memory locations in a memory such as Dynamic Random Access Memory (DRAM), flash memory or any other type of memory.

The pre-computed polynomials g16, g8 and g4 are used to successively reduce a block of data (data segment) into smaller intermediate values in order to produce a CRC residue. The form of the pre-computed polynomials allows the plurality of

stages to perform many of the calculations in parallel by processing mutually exclusive regions of the block of data in parallel. Performing calculations in parallel reduces the time to compute the CRC residue. For example, for a 32-bit input data block, g16 reduces the 16 Most Significant Bits of the 32-bit input data block, g8 reduces the next 8 MSBs and g4 reduces the next 4 MSBs. These reductions may be referred to as the various stages of the reduction.

Referring to Fig. 3, in the embodiment shown for a 64-bit input data segment, the first reduction stage 206a receives a 64-bit input data segment and reduces the input data segment using the g16 polynomial. First, a 16 x 32 multiply operation is performed with the 16 Most Significant Bits (MSB)s of the input data segment and the 32-bit polynomial g16. Next, an Exclusive OR (XOR) operation is performed on the 48-bit result of the multiply operation and the 48 Least Significant Bits (LSB)s of the input data segment to compute a 48-bit output from the first reduction stage 206a.

The second reduction stage 104-2 receives a 48-bit input data segment from the first stage 206a and reduces the input data segment using the g8 polynomial. First, an 8 x 32 multiply operation is performed using the 8 MSBs of the input data segment and the 32-bit g8 polynomial. Next, an XOR operation is performed using the 40-bit result of the multiply operation and the 40 Least Significant Bits (LSB)s of the input data segment to compute a 40-bit output that is forwarded to the third reduction stage 206c.

The third reduction stage 206c receives a 40-bit input from the second stage 206b and reduces the input data segment using the g4 polynomial. First, a 4 x 32 multiply operation is performed with the 4 MSBs of the 40-bit input data segment and the 32-bit g4 polynomial. Then, the third reduction stage performs an XOR operation with the 36-bit result of the multiply operation and the 36 Least Significant Bits (LSB)s of the 40-bit input to provide a 36 bit output that is forwarded to a next reduction stage 206d (Fig. 2) to continue to compute the residue.

The use of storage elements 302a,..,302c to store pre-computed polynomials allows the 32-bit CRC to be selectable (programmable) through selection of the appropriate pre-computed polynomials that are loaded into the storage elements.

As shown in Fig. 3, each reduction stage 206a,..., 206 operates on operands with sizes (number of bits) that are asymmetric, that is, the multiplier and multiplicand have a different number of bits and the multiplier is decreased by half in each successive

reduction stage (16, 8, 4). The sizes of the operands (multiplier, multiplicand) become more asymmetric with each successive stage. For example, the sizes of the input operands (multiplicand, multiplier) to the multiply operation in the various stages are 32:16 (first stage), 32:8 (second stage) and 32:4 (third stage).

The multiply operations in the reduction stages 206a-206c may be performed by a classical carry-less multiplication of the form i*32, however this is suboptimal. In an embodiment of the present invention, each of the plurality of reduction stages to produce a residue for a block of data is optimized to decrease area used to perform the reduction while maintaining the same delay through the plurality of reduction stages 206a,..206f.

The schoolbook method to multiply two polynomials is to multiply each term of a first polynomial by each term of a second polynomial. For example, a first polynomial of degree 1 with two terms $a_1x + a_0$ may be multiplied by a second polynomial of degree 1 with two terms $b_1x + b_0$ by performing four multiply operations and three addition operations to produce a polynomial of degree 2 with three terms as shown below:

$$(a_1x + a_0)(b_1x + b_0) = a_1 b_1x^2 + (a_0b_1x + a_1b_0x) + a_1 b_1$$

The number of multiply operations and Arithmetic Logical Unit (ALU) operations increases with the number of terms in the polynomials. For example, using the schoolbook method, the number of multiply operations to multiply two polynomials each having n terms is n^2 and the number of additions is $(n - 1)^2$.

The Karatsuba algorithm (KA) reduces the number of multiply operations compared to the schoolbook method by multiplying two two-term polynomials ($A(x) = (a_1x + a_{0)}$ and $B(x) = (b_1x + b_0)$), each having two coefficients ((a_1,$a_{0)}$ and (b_1 b_0)), using three scalar multiplications instead of four multiplications as shown below:

$$C(x) = (a_1x + a_0)(b_1x + b_0) = a_1 b_1x^2 + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1 b_1) + a_0b_0$$

Thus, four additions and three multiplications are required to compute the result C(x) of multiplying two two-term polynomials using the Karatsuba algorithm. The KA algorithm relies on the ability to perform shift operations faster than a standard multiplication operation.

The KA algorithm is typically applied to symmetric multiplications, where the operands (multiplier and multiplicand) are the same size. However, the multiply operations in reduction stages 206a,.., 206c are asymmetric. Furthermore, a naïve application of KA increases the critical path significantly while reducing area. In

addition, there may be a number of levels of application of KA that typically range between 2 and 4. The number of levels is limited by the smaller of the operand sizes. For example, one or two levels of application of KA are optimal for computing the product of a 16-bit multiplier by a 32-bit multiplicand and one level of application of KA is optimal for computing the product of an 8-bit multiplier by a 32-bit multiplicand. A classical multiplier is optimal for computing a product of a 4-bit multiplier and a 32-bit multiplicand. In an embodiment of the present invention, the type of multiplication technique, multi-stage KA, single-stage KA or classical multiplier is selected for each stage based on the operand size.

Fig. 4 illustrates an embodiment of an 8 x 32 multiplier that applies a one level application of Karatsuba (KA). The 8 x 32 multiplier may be used in the second reduction stage 206b shown in Fig. 3. As discussed earlier, the input data segment is reduced by 8-bits in the second reduction stage 206b. To perform the reduction by 8-bits, the multiplier applies a one level application of KA to perform a multiply operation on a 32-bit multiplicand (represented by A in Fig. 4) and an 8-bit multiplier (represented by B in Fig. 4).

The Karatsuba algorithm (KA) reduces the number of multiply operations compared to the schoolbook method by multiplying two two-term polynomials (A(x) = ($a_1x + a_0$) and B(x) = ($b_1x + b_0$)), each having two coefficients ((a_1,a_0) and (b_1 b_0)), using three scalar multiplications instead of four multiplications as shown below:

$$C(x) = (a_1x + a_0)(b_1x + b_0) = a_1 b_1x^2 + ((a_0 + a_1) (b_0 + b_1) - a_0b_0 - a_1 b_1) + a_0b_0$$

Thus, four additions and three multiplications (that is, (1) a_1b_1, (2) a_0b_0, and (3) (($a_0 + a_1$) ($b_0 + b_1$)) are required to compute the result C(x) of multiplying two two-term polynomials using the Karatsuba algorithm.

As shown in Fig. 4, the 32-bit multiplicand A is subdivided into eight 4-bit elements labeled a7-a0 and the 8-bit multiplier B is subdivided into two 4-bit elements labeled b1 and b0. The product C is A x B which is computed by applying KA to the following groups of elements of A and B:

(1) b1, b0 and a1, a0;

(2) b1, b0 and a2, a3;

(3) b1, b0 and a5, a4; and

(4) b1, b0 and a7, a6 in the order shown in Fig. 4.

The following four products (P0-P3) are computed by applying KA to elements in A and B.

$P0 = (a_1x + a_0)(b_1x + b_0) = a_1 b_1x^2 + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1 b_1) + a_0b_0$

$P1 = ((a_3x + a_2)(b_1x + b_0) = a_3 b_1x^2 + ((a_2 + a_3)(b_0 + b_1) - a_2b_0 - a_3 b_1) + a_2b_0)2^8$

$P2 = ((a_5x + a_4)(b_1x + b_0) = a_5 b_1x^2 + ((a_4 + a_5)(b_0 + b_1) - a_4b_0 - a_1 b_5) + a_4b_0)2^{16}$

$P3 = ((a_7x + a_6)(b_1x + b_0) = a_7 b_1x^2 + ((a_6 + a_7)(b_0 + b_1) - a_6b_0 - a_7 b_1) + a_6b_0)2^{24}$

As shown, P1 is shifted by eight bits, P2 is shifted by 16-bits and P3 is shifted by 24 bits. The products are then XORed to provide a 40-bit result C(x).

$C(x) = P0 + P1 + P2 + P3.$

The worst case path for each product P0-P3 is the term that includes both multiplication and addition, for example, $((a_6 + a_7)(b_0 + b_1)$ when computing P3 or $((a_2 + a_3)(b_0 + b_1)$ when computing P1.

With a straightforward construction of an XOR tree after the core multipliers for the Karatsuba algorithm, the critical path delay of the multiplication circuit is the delay of the XOR operations before the core multipliers, delay of the core multiplier and the delay of the XOR tree. This may be reduced through the use of the asymmetric property of the Karatsuba Multiplier. For example, there are a different number of multiplication levels for computing different bits of the 40-bit result C(39:0). As shown in Fig. 4,the critical path is asymmetric with the lower order four bits C(3:0) of the result have one level of computation, C(7:0) and C(39:36) have four levels of computation, C(35:8) have five levels of computation. Also, as shown in Fig. 3, the Most Significant Bits (MSB)s of the input to each stage, for example, input[63:48] in the first level and input[47:0] in the next level are in the critical path because they are input to a multiplier with multiple levels of computation whereas the Least Significant Bits (LSBs) are only input to accumulators.

Thus, a critical path may be reduced for a particular bit or bits of the product by bypassing a multiplier. For example, any of product bits C[35:8] may be computed through a redundant set of XOR gates instead of through the Karatsuba multiplier, that is, the Karatsuba multiplier may be bypassed for these particular bits. The set of XOR gates may be used to compute $C[35] = a_{35}* b_0 + a_{34}* b_1 + a_{33}* b_0 + a_{35}* b_2 \ldots a_0* b_{35}$ to reduce this critical path.

Fig. 5 illustrates an embodiment of a 16 x 32 multiplier that applies a one level application of Karatsuba (KA). The 16 x 32 multiplier may be used in the first

reduction stage 206a shown in Fig. 3. As discussed earlier, the input data segment is reduced by 16-bits in the first reduction stage 206a. To perform the reduction by 16-bits, the multiplier applies a one level application of KA to perform a multiply operation on a 32-bit multiplicand (represented by A in Fig. 5) and an 16-bit multiplier (represented by B in Fig. 5).

As shown in Fig. 5, the 32-bit multiplicand A is subdivided into four 8-bit elements labeled a4-a0 and the 16-bit multiplier B is subdivided into two 8-bit elements labeled b1 and b0. The product C is A x B which is computed by applying KA to the following groups of elements of A and B:

(1) b1, b0 and a1, a0; and

(2) b1, b0 and a2, a3;in the order shown in Fig. 4.

The following two products (P0-P1) are computed by applying KA to elements in A and B.

$P0 = (a_1x + a_0)(b_1x + b_0) = a_1 b_1x^2 + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1 b_1) + a_0b_0$

$P1 = ((a_3x + a_2)(b_1x + b_0) = a_3 b_1x^2 + ((a_2 + a_3)(b_0 + b_1) - a_2b_0 - a_3 b_1) + a_2b_0)2^{16}$

As shown, P1 is shifted by sixteen bits. The products are then XORed to provide a 48-bit result C(x).

$C(x) = P0 + P1 + P2 + P3.$

The worst case path for each product P0-P1 is the term that includes both multiplication and addition, for example, $((a_2 + a_3)(b_0 + b_1)$ when computing P1. As discussed in conjunction with Fig. 4, the critical path delay may be reduced through the use of the asymmetric property of the Karatsuba Multiplier.

Fig. 6 illustrates an embodiment of a 16 x 32 multiplier that applies a two level application of Karatsuba (KA). The 16 x 32 multiplier may be used in the first reduction stage 206a shown in Fig. 3. As discussed earlier, the input data segment is reduced by 16-bits in the first reduction stage 206a. To perform the reduction by 16-bits, the multiplier applies a two level application of KA to perform a multiply operation on a 32-bit multiplicand (represented by A in Fig. 6) and an 16-bit multiplier (represented by B in Fig. 6).

As shown in Fig. 6, the 32-bit multiplicand A is subdivided into eight 4-bit elements labeled a7-a0 and the 16-bit multiplier B is subdivided into four 4-bit elements labeled b3-b0. The product C is A*B which is computed by applying a two level KA 600 to the following groups of elements of A and B to compute (b3: b0) *

(a3:a0) which is a symmetric multiplication, that is, each operand has the same number of bits:

(1) b1, b0 and a1, a0; and

(2) b1, b0 and a2, a3.

Then applying a two level KA 602 to the following groups of elements of A and B to compute (b3: b0) * (a7:a4) which is also symmetric multiplication, that is, each operand has the same number of bits:

(1) b1, b0 and a4, a5; and

(2) b1, b0 and a6, a7.

Both two level KAs 600, 602 are performed in the same manner with different groups of elements of A and B. Thus, only one of the two level KAs, two level KA 600 will be described here.

A first level KA is applied to elements b3:b0 in B and elements a3:a0 in A to generate first level KA elements 604-1, 604-2, 604-3 and 604-4 as shown in Fig. 6. A second level KA is then applied to each of these first level KA elements 604-1, 604-2, 604-3 and 604-4. These second level KAs are labeled 606, 608, 612 and 614 in Fig. 6 for ease of reference. A KA is applied to elements a3, b3, a2, b2 and a separate KA is applied to elements a1, b1, a0, b0 generated by the first level KA in second level KA labeled 606. A KA is applied to first level KA element 604-2 in second level KA labeled 608. A KA is applied to first level KA element 604-3 in second level KA labeled 610. A KA is applied to first level KA element 604-4 in second level KA labeled 612.

The worst case path is the term that includes both multiplication and addition, for example, (a3+a2+a1+a0).(b3+b2+b1+ b0). As discussed earlier in conjunction with Fig. 4, the critical path delay may be reduced through the use of the asymmetric property of the Karatsuba Multiplier.

The area savings are even greater as the number of bits in the CRC polynomial is increased, for example, a 64 bit CRC polynomial instead of a 32 bit CRC polynomial. The reduction in area is dependent on the size of the CRC reduction circuit.

Fig. 7 is a flowgraph illustrating an embodiment of a method for selecting an optimal multiplication algorithm to be used a reduction stage that is used to compute a CRC residue.

At block 700, if the size of the mulitiplicand for the reduction stage is greater than four, processing continues with block 702. If not, processing continues with block 704.

At block 702, if the size of the multiplicand is greater than eight, processing continues with block 706. If not, processing continues with block 708.

At block 704, a classical multiplication technique is selected to compute the product in the reduction stage.

At block 706, a one level application of Karatsuba (KA) is selected to compute the product in the reduction stage.

At block 708, either a one level application of Karatsuba or a two-level application of Karatsuba is selected to compute the product in the reduction stage.

It will be apparent to those of ordinary skill in the art that methods involved in embodiments of the present invention may be embodied in a computer program product that includes a computer usable medium. For example, such a computer usable medium may consist of a read only memory device, such as a Compact Disk Read Only Memory (CD ROM) disk or conventional ROM devices, or a computer diskette, having a computer readable program code stored thereon.

While embodiments of the invention have been particularly shown and described with references to embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of embodiments of the invention encompassed by the appended claims.

CLAIMS

1. An apparatus comprising:

a plurality of reduction stages, each of the plurality of the reduction stages to compute a product of a multiplier and a multiplicand having different sizes, the product to be computed using an optimal multiplication algorithm for the respective sizes of the multiplicand and the multiplier.

2. The apparatus of claim 1, wherein the multiplication algorithm applies a one level application of Karatsuba (KA).

3. The apparatus of claim 1, wherein the multiplicand is 32 and the multiplier is 16.

4. The apparatus of claim 1, wherein the multiplicand is 32 and the multiplier is 8.

5. The apparatus of claim 1, wherein the multiplication algorithm applies a two level application of Karatsuba (KA).

6. The apparatus of claim 5, wherein the multiplicand is 32 and the multiplier is 16.

7. The apparatus of claim 1, wherein the multiplicand is 32, the multiplier is less than or equal to 4 and the product is computed using a classical multiplication technique.

8. A method comprising:

providing a plurality of reduction stages, each of the plurality of the reduction stages to compute a product of a multiplier and a multiplicand having different sizes; and

selecting an optimal multiplication algorithm for each reduction stage to compute the product dependent on respective sizes of the multiplicand and the multiplier.

9. The method of claim 8, wherein the multiplication algorithm applies a one level application of Karatsuba (KA).

10. The method of claim 8, wherein the multiplicand is 32 and the multiplier is 16.

11. The method of claim 8, wherein the multiplicand is 32 and the multiplier is 8.

12. The method of claim 8, wherein the multiplication algorithm applies a two level application of Karatsuba (KA).

13. The method of claim 12, wherein the multiplicand is 32 and the multiplier is 16.

14. The method of claim 8, wherein the multiplicand is 32, the multiplier is less than or equal to 4 and the product is computed using a classical multiplication technique.

15. A system comprising:

a dynamic random access memory; and

a cyclic redundancy check unit coupled to the dynamic random access memory, the cyclic redundancy check unit comprising:

a plurality of reduction stages, each of the plurality of the reduction stages to compute a product of a multiplier and a multiplicand having different sizes, the product to be computed using an optimal multiplication algorithm for the respective sizes of the multiplicand and the multiplier.

16. The system of claim 15, wherein the multiplication algorithm applies a one level application of Karatsuba (KA).

17. The system of claim 15, wherein the multiplicand is 32 and the multiplier is 16.

18. The system of claim 15, wherein the multiplicand is 32 and the multiplier is 8.

19. The system of claim 15, wherein the multiplication algorithm applies a two level application of Karatsuba (KA).

20. The system of claim 19, wherein the multiplicand is 32 and the multiplier is 16.

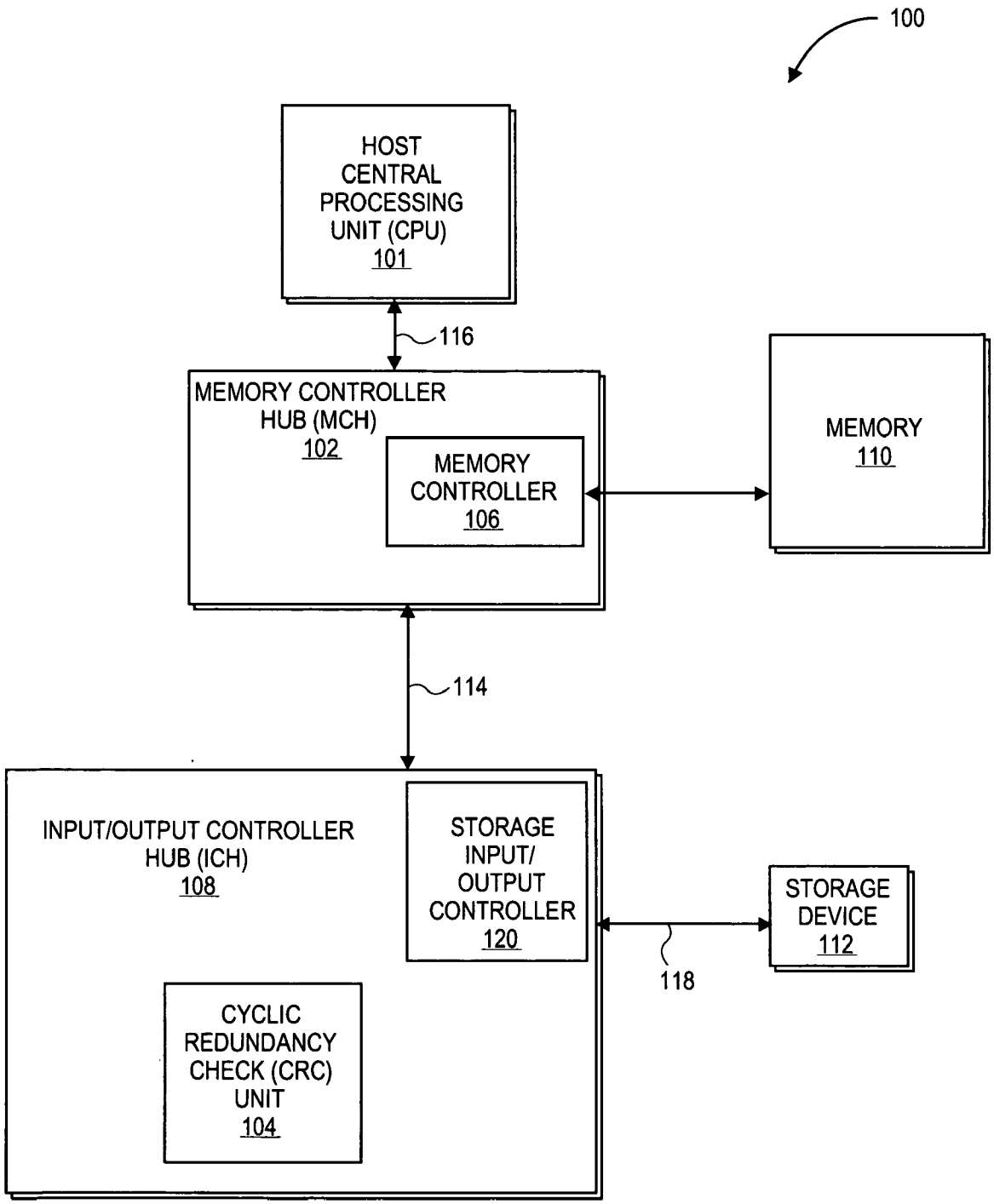
100
HOST CENTRAL PROCESSING UNIT (CPU) 101
116
MEMORY CONTROLLER HUB (MCH) 102
MEMORY CONTROLLER 106
MEMORY 110
114
INPUT/OUTPUT CONTROLLER HUB (ICH) 108
STORAGE INPUT/ OUTPUT CONTROLLER 120
STORAGE DEVICE 112
118
CYCLIC REDUNDANCY CHECK (CRC) UNIT 104

FIG. 1

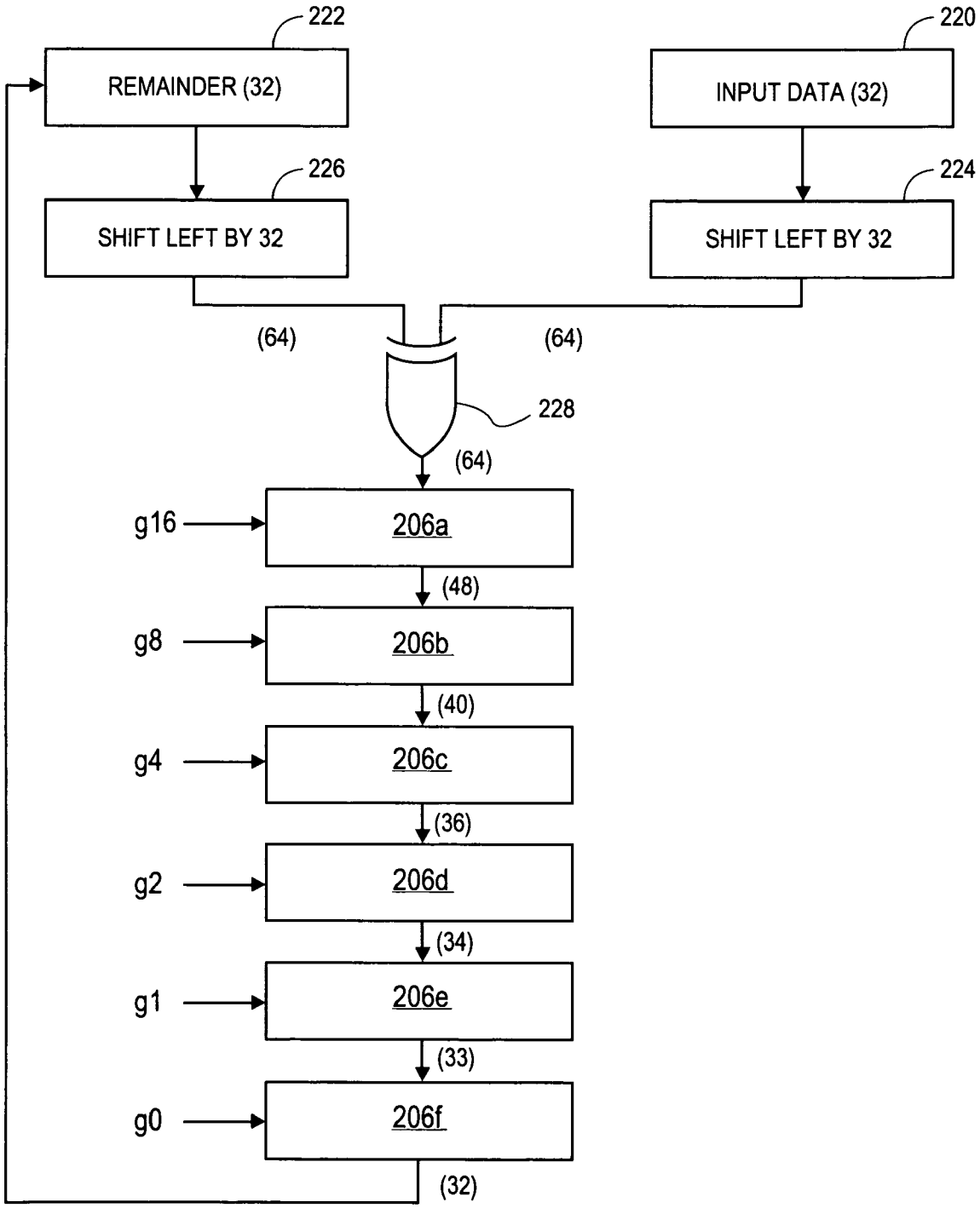
222
REMAINDER (32)
220
INPUT DATA (32)
226
SHIFT LEFT BY 32
224
SHIFT LEFT BY 32
(64)
(64)
228
(64)
g16
206a
(48)
g8
206b
(40)
g4
206c
(36)
g2
206d
(34)
g1
206e
(33)
g0
206f
(32)

FIG. 2

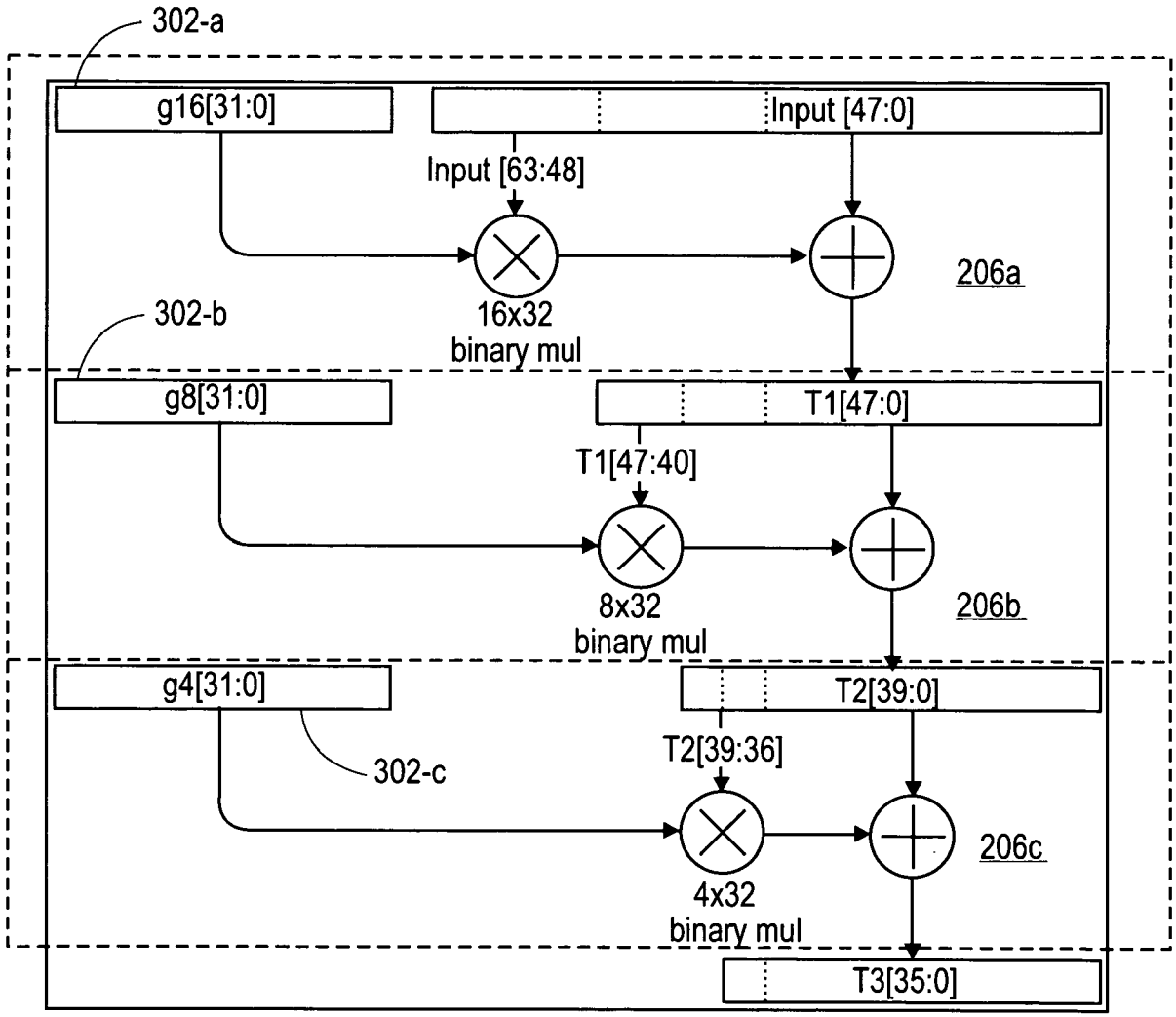
302-a
g16[31:0]
Input [47:0]
Input [63:48]
16x32
binary mul
206a
302-b
g8[31:0]
T1[47:0]
T1[47:40]
8x32
binary mul
206b
g4[31:0]
T2[39:0]
T2[39:36]
302-c
4x32
binary mul
206c
T3[35:0]

FIG. 3

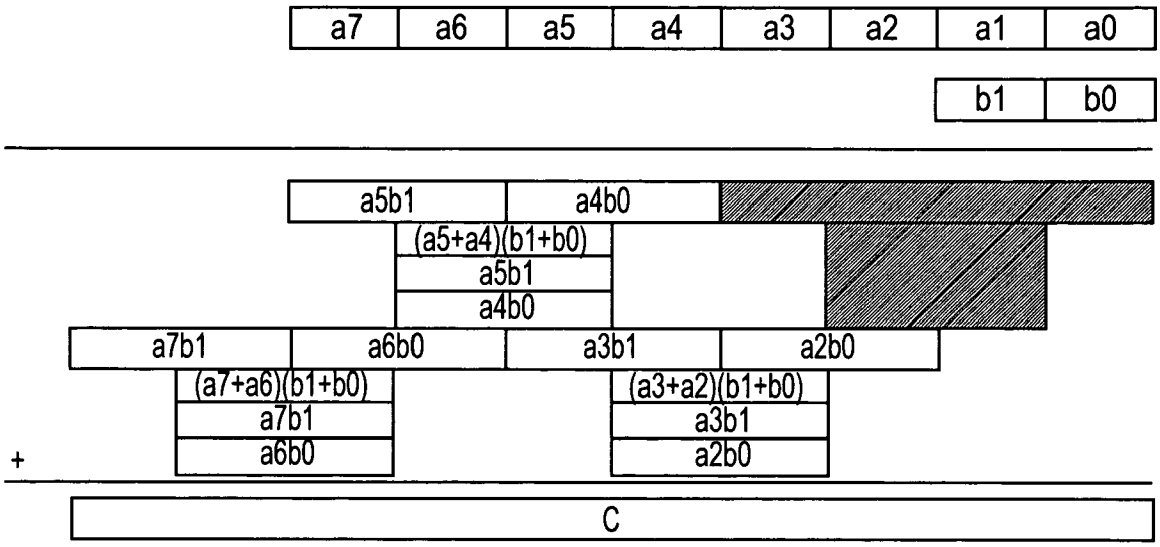
a7
a6
a5
a4
a3
a2
a1
a0
b1
b0
a5b1
a4b0
(a5+a4)(b1+b0)
a5b1
a4b0
a7b1
a6b0
a3b1
a2b0
(a7+a6)(b1+b0)
a7b1
a6b0
(a3+a2)(b1+b0)
a3b1
a2b0
+
C

FIG. 4

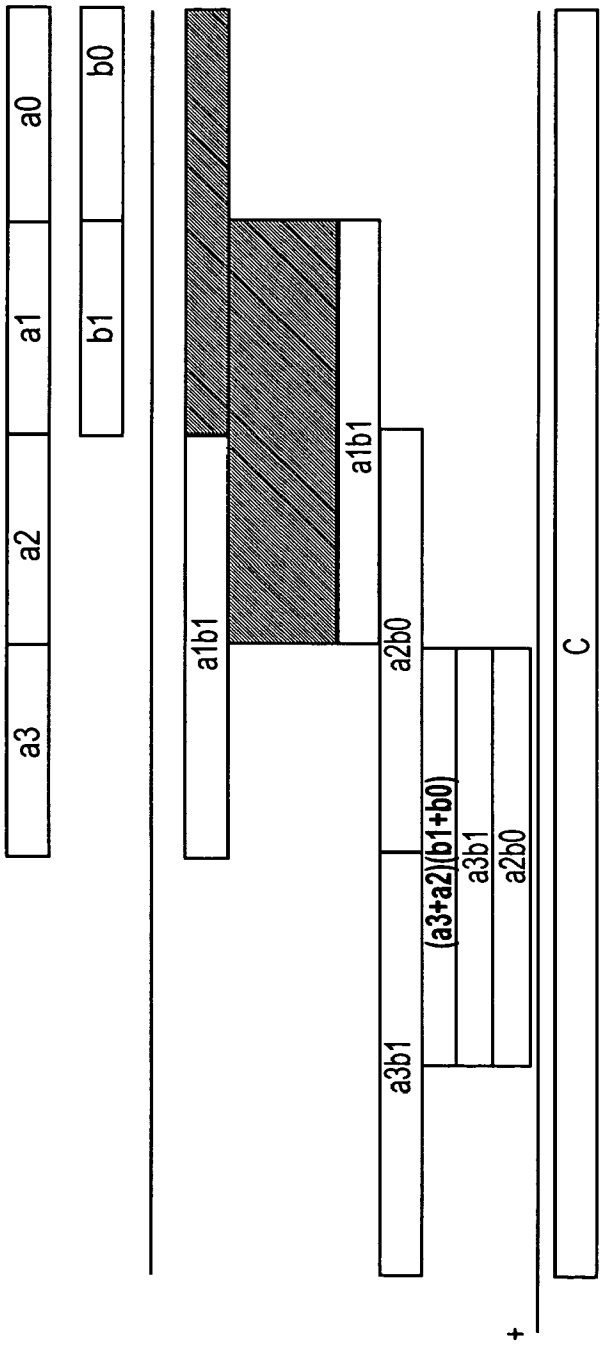

FIG. 5

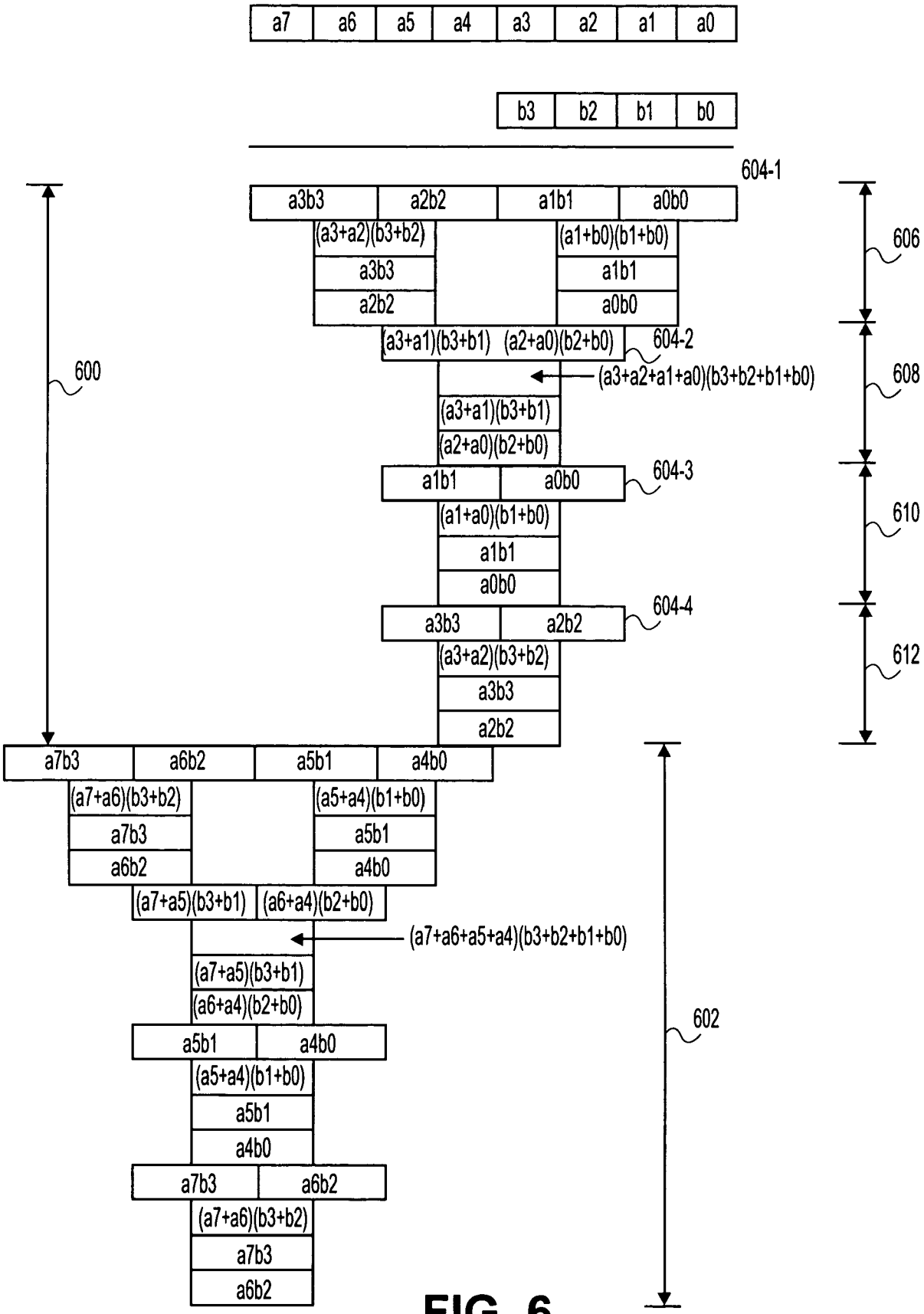
a7 a6 a5 a4 a3 a2 a1 a0
b3 b2 b1 b0
604-1
a3b3 a2b2 a1b1 a0b0
(a3+a2)(b3+b2) (a1+b0)(b1+b0)
a3b3 a1b1
a2b2 a0b0
(a3+a1)(b3+b1) (a2+a0)(b2+b0)
604-2
(a3+a2+a1+a0)(b3+b2+b1+b0)
(a3+a1)(b3+b1)
(a2+a0)(b2+b0)
a1b1 a0b0
604-3
(a1+a0)(b1+b0)
a1b1
a0b0
a3b3 a2b2
604-4
(a3+a2)(b3+b2)
a3b3
a2b2
a7b3 a6b2 a5b1 a4b0
(a7+a6)(b3+b2) (a5+a4)(b1+b0)
a7b3 a5b1
a6b2 a4b0
(a7+a5)(b3+b1) (a6+a4)(b2+b0)
(a7+a6+a5+a4)(b3+b2+b1+b0)
(a7+a5)(b3+b1)
(a6+a4)(b2+b0)
a5b1 a4b0
(a5+a4)(b1+b0)
a5b1
a4b0
a7b3 a6b2
(a7+a6)(b3+b2)
a7b3
a6b2
600
602
606
608
610
612

FIG. 6

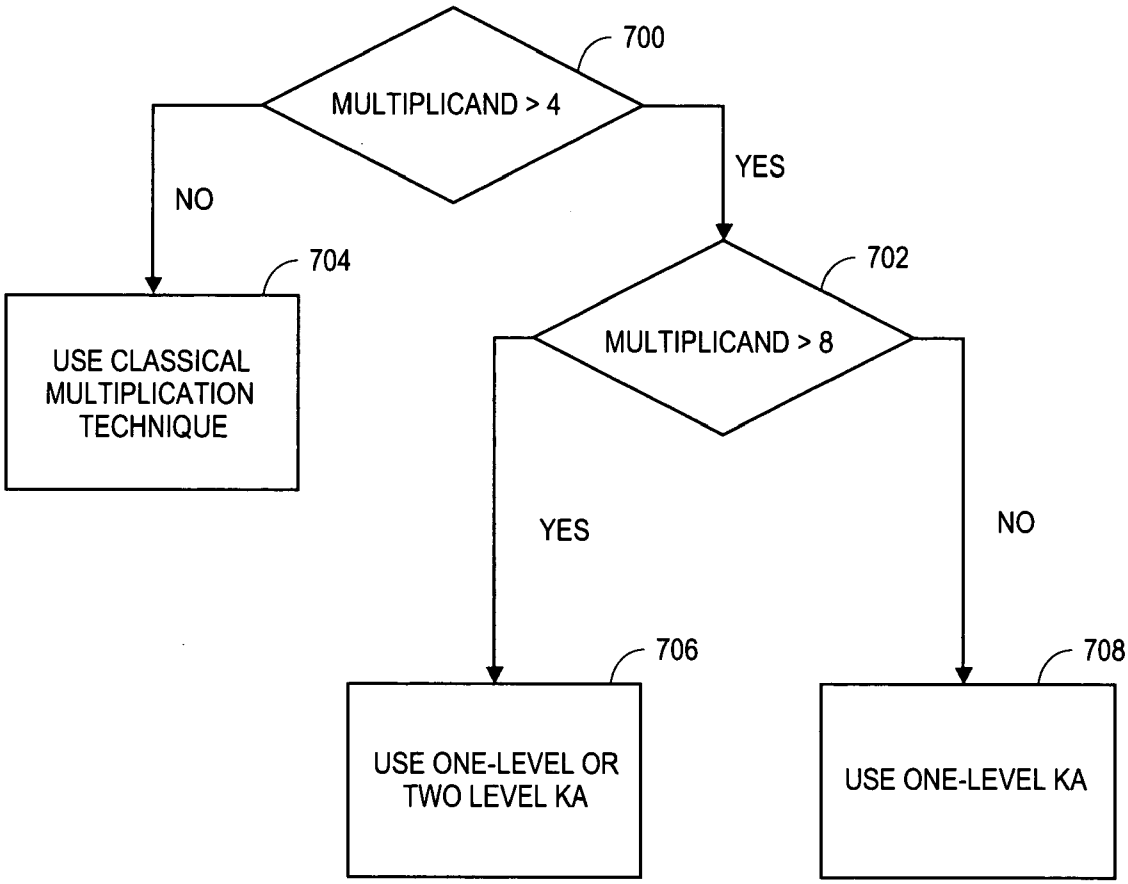
700
MULTIPLICAND > 4
NO
YES
704
USE CLASSICAL MULTIPLICATION TECHNIQUE
702
MULTIPLICAND > 8
YES
NO
706
USE ONE-LEVEL OR TWO LEVEL KA
708
USE ONE-LEVEL KA

FIG. 7