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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 June 2010 (10.06.2010)

PCT

(10) International Publication Number
WO 2010/064205 A1

(51) International Patent Classification:
H03M 13/41 (2006.01)

(21) International Application Number:
PCT/IB2009/055472

(22) International Filing Date:
3 December 2009 (03.12.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/119,522 3 December 2008 (03.12.2008) US

(71) Applicant *(for all designated States except US)*: **NXP B.V.** [NL/NL]; High Tech Campus 60, NL-5656 AG Eindhoven (NL).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **CHABOT, Xavier** [FR/US]; c/o NXP Semiconductors, IP&L Department, 1109 McKay Drive, MS41, San Jose, California 95131 (US).

(74) Agents: **WILLIAMSON, Paul, L.** et al.; c/o NXP Semiconductors, IP&L Department, Betchworth House, 57-65 Station Road, Redhill Surrey SH1 1DL (GB).

(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, CL, 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, PE, 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, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) **Title**: SYSTEM AND METHOD FOR VITERBI DECODING USING APPLICATION SPECIFIC EXTENSIONS

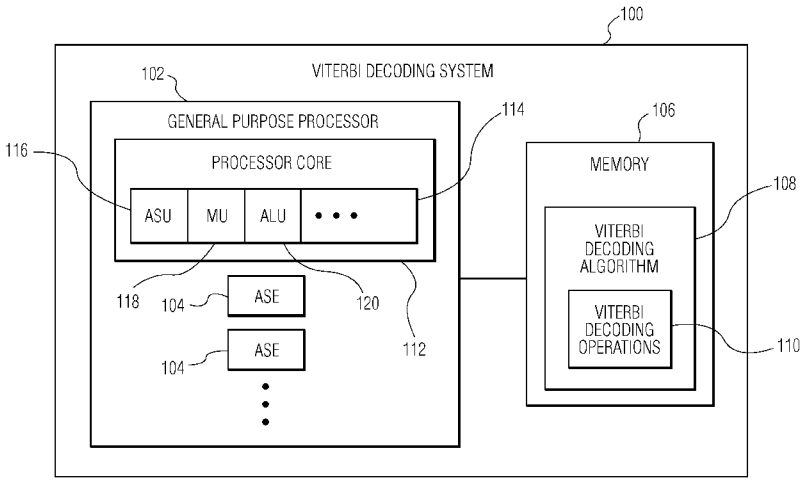

FIG. 1

(57) **Abstract**: A system and method for Viterbi decoding utilizes a general purpose processor with application specific extensions to perform Viterbi decoding operations specified in a Viterbi decoding algorithm stored in memory.

WO 2010/064205 A1

SYSTEM AND METHOD FOR VITERBI DECODING USING APPLICATION SPECIFIC EXTENSIONS

[0001] Embodiments of the invention relate generally to electronic systems and, more particularly, to a system and method for Viterbi decoding using application specific extensions.

[0002] Viterbi decoding is used for decoding convolutional codes and solving estimation problems for a variety of applications such as software digital radios and pattern recognitions. Because Viterbi decoding puts computing burdens on general purpose processors, external hardware may be implemented to perform Viterbi decoding. However, the external hardware puts restrictions on Viterbi decoding software optimization, reduces Viterbi decoding software portability, and increases development costs for interfacing with the general purpose processors and risks associated with system integration.

[0003] Thus, there is a need for a system and method for Viterbi decoding that assists Viterbi decoding software optimization, improves Viterbi decoding software portability, and lowers development costs for interfacing with the general purpose processors and system integration risks.

[0004] A system and method for Viterbi decoding utilizes a general purpose processor with application specific extensions to perform Viterbi decoding operations specified in a Viterbi decoding algorithm stored in memory.

[0005] In an embodiment, a Viterbi decoding system comprises memory and a general purpose processor. The memory is configured to store a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations. The general purpose processor comprises a plurality of application specific extensions, wherein each application specific extension is configured to perform at least one of the Viterbi decoding operations specified in the Viterbi decoding algorithm stored in the memory. The Viterbi decoding system is configured such that all the Viterbi decoding operations specified in the Viterbi decoding algorithm are performed exclusively within the general purpose processor using at least one of the application specific extensions.

[0006] In an embodiment, a method for Viterbi decoding using application specific extensions comprises (a) obtaining a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations and (b) exclusively performing the plurality of Viterbi decoding operations within a general purpose processor using a plurality of application specific extensions in the general purpose processor.

[0007] In an embodiment, a Viterbi decoding system comprises memory and a general purpose processor. The memory is configured to store a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations. The general purpose processor comprises a processor core and a plurality of application specific extensions, wherein the processor core includes a plurality of functional units and each application specific extension is configured to perform one of the Viterbi decoding operations specified in the Viterbi decoding algorithm stored in the memory. The Viterbi decoding system is configured such that all the Viterbi decoding operations specified in the Viterbi decoding algorithm are performed exclusively within the general purpose processor using at least one of the application specific extensions.

[0008] Other aspects and advantages of embodiments of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrated by way of example of the principles of the invention.

[0009] Fig. 1 illustrates a Viterbi decoding system using a general purpose processor with application specific extensions in accordance with an embodiment of the invention.

[0010] Fig. 2 illustrates the Viterbi decoding process of the Viterbi decoding algorithm stored in the memory of Fig. 1 in accordance with an embodiment of the invention.

[0011] Fig. 3 illustrates a part of the Viterbi decoding process using a "BIT_INTERLEAVE_DUAL16" application specific extension in the general purpose processor in accordance with an embodiment of the invention.

[0012] Fig. 4 illustrates a part of the Viterbi decoding process using a "BIT_SHIFT_INTERLEAVE_DUAL16" application specific extension in the general purpose processor in accordance with an embodiment of the invention.

[0013] Fig. 5 illustrates a part of the Viterbi decoding process using a "VNEXTSTATE_LE" application specific extension in the general purpose processor on a little endian add-compare-select pattern in accordance with an embodiment of the invention.

[0014] Fig. 6 illustrates a part of the Viterbi decoding process using the "VNEXTSTATE_LE" application specific extension in the general purpose processor on a little endian traceback pattern in accordance with an embodiment of the invention.

[0015] Fig. 7 illustrates a part of Viterbi decoding using a "VNEXTSTATE_BE" application specific extension in the general purpose processor on a big endian add-compare-select pattern in accordance with an embodiment of the invention.

[0016] Fig. 8 illustrates a part of Viterbi decoding using the "VNEXTSTATE_BE" application specific extension in the general purpose processor on a big endian traceback in accordance with an embodiment of the invention.

[0017] Fig. 9 is a schematic flow chart diagram of a method for Viterbi decoding using application specific extensions in accordance with an embodiment of the invention.

[0018] Throughout the description, similar reference numbers may be used to identify similar elements.

[0019] With reference to Fig. 1, a Viterbi decoding system 100 using a general purpose processor 102 with application specific extensions 104 in accordance with an embodiment of the invention is described. Embodiments of the Viterbi decoding system can be applied to various electronic systems, in particular, pattern recognition systems based on a finite state Markov process and digital communication systems. Example applications of the various electronic systems may include image recognition, speech recognition, musical melody recognition, forward error correction, and software digital radio.

[0020] As shown in Fig. 1, the Viterbi decoding system 100 includes memory 106 and the general purpose processor 102, which is not tied to a particular application. The memory is configured to store a Viterbi decoding algorithm 108 that specifies Viterbi decoding operations 110 to be performed by

the general purpose processor. An exemplary Viterbi decoding algorithm is described in detail below with reference to Fig. 2. The general purpose processor includes applications specific extensions (ASEs) 104 and a processor core 112. Each ASE is configured to perform at least one of the Viterbi decoding operations specified in the Viterbi decoding algorithm stored in the memory. The processor core includes functional units 114, such as an addition/subtraction functional unit (ASU) 116 to perform addition functions and subtraction functions, a multiplication functional unit (MU) 118 to perform multiplication functions, and an arithmetic logic functional unit (ALU) 120 to perform logic functions. All the Viterbi decoding operations specified in the Viterbi decoding algorithm are performed exclusively within the general purpose processor using at least one of the application specific extensions.

[0021] The ASEs 104 may be implemented in hardware and/or software. In some embodiments, each ASE may be a set of processor instructions for the general purpose processor 102, where the set of processor instructions perform a Viterbi decoding operation specified in the Viterbi decoding algorithm 108 stored in the memory 106. In some embodiments, the ASEs may reuse existing functional units 114 in the processor core 112, which will result in more efficient source code that better utilizes processor resources, more flexible and portable software, and less risk than to develop and to integrate more complex hardware in a system-on-chip (SoC).

[0022] Fig. 2 illustrates the Viterbi decoding process of the Viterbi decoding algorithm 108 stored in the memory 106 of Fig. 1 in accordance with an embodiment of the invention. In this embodiment, the Viterbi decoding algorithm includes three processes, a branch metric process, an add-compare-select (ACS) process, and a traceback process. According to the operation sequence of the Viterbi decoding algorithm, the first process is the branch metric process, which pads input values for omitted data and computes branch metric values from each depunctured group of input values. The second process is the ACS process, which loops through a block of input branch metrics and builds a trellis of possible paths. The third process is the traceback process, which starts from a known last value and goes back through the trellis of possible paths to select at each step the most likely output bit. In the illustrated embodiment of Fig. 2, the Viterbi decoding

algorithm processes N*1/R softbits, where N and 1/R are two integers that are greater than zero. For instance, as in the case of Fig. 2, When 1/R is equal to 2, the branch metric process computes branch metric values for the two softbits S_0 and S_1 and outputs the branch metric values S_0+S_1, $S_0-S_{,1}$ $-S_0-S_1$, and $-S_0+S_1$ to the ACS process. The ACS process involves processing the branch metric values from the branch metric process and metric initiation information from a sixteen-bit Path_metric data word and outputs decision bits. The traceback process involves processing the decision data bits from the ACS process and initiation state information and outputting N bits. The number of processing steps of the branch metric process and the ACS process is N and the number of processing steps of the traceback process is N.

[0023] The Viterbi decoding algorithm 108 specifies Viterbi decoding operations for the general purpose processor. In some embodiments, the Viterbi decoding operations specified in the Viterbi decoding algorithm include Viterbi decoding branch metric summing and subtracting operations that may be used in the branch metric process, Viterbi decoding ACS operations that may be used in the ACS process, and Viterbi decoding bit manipulating operations that may be used in the traceback process.

[0024] Each ASE 104 of the general purpose processor 102 performs at least one of the Viterbi decoding operations specified in the Viterbi decoding algorithm 108. Embodiments of the ASEs may perform Viterbi decoding branch metric summing and subtracting operations, Viterbi decoding ACS operations, and Viterbi decoding bit manipulating operations specified in the Viterbi decoding algorithm.

[0025] Embodiments of the ASEs 104 that are configured to perform Viterbi decoding branch metric summing and subtracting operations are first described.

[0026] A "SADDSUBR2_DUAL16 ASE" in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on two sixteen-bit input data blocks A and B, which are packed into a thirty two-bit input data word, to generate two sixteen-bit output data blocks C and D, which are packed in a thirty two-bit output data word, where C= A+B, D=A-B. In some embodiments, the "SADDSUBR2_DUAL16" ASE computes branch metric for code rate R=1/2

Viterbi decoding systems. In some embodiments, the "SADDSUBR2_DUAL16" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating sixteen-bit arithmetic.

[0027] A "SADDSUBR4_QUAD16" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on four sixteen-bit input data blocks A, B, C, and D, which are packed into two thirty two-bit input data words, to generate four sixteen-bit output data blocks E, F, G, and H, which are packed in two thirty two-bit output data words, where E=A+B+C+D, F=A+B+C-D, G=A+B-C+D, and H=A+B-C-D. In some embodiments, the "SADDSUBR4_QUAD16" ASE computes half of the needed branch metrics for code rate R=1/4 Viterbi decoding systems. A "SADDSUBR4N_QUAD16" ASE described below computes the other values. In some embodiments, the "SADDSUBR4_QUAD16" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating sixteen-bit arithmetic.

[0028] The "SADDSUBR4N_QUAD16" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on four sixteen-bit input data blocks A, B, C, and D, which are packed into two thirty two-bit input data words, to generate four sixteen-bit output data blocks E, F, G, and H, which are packed in two thirty two-bit output data words, where E=A-B+C+D, F=A-B+C-D, G=A-B-C+D, and H=A-B-C-D. In some embodiments, the "SADDSUBR4N_QUAD16" ASE computes half of the needed branch metrics for code rate R=1/4 Viterbi decoding systems. The "SADDSUBR4_QUAD16" ASE described above computes the other values. In some embodiments, the "SADDSUBR4N_QUAD16" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating sixteen-bit arithmetic.

[0029] A "SADDSUBR4_QUAD8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on four eight-bit input data blocks A, B, C, and D, which are packed into a thirty two-bit data word, to generate four eight-bit output data blocks E, F, G, and H, which are packed in a thirty two-bit output data word, where E=A+B+C+D, F=A+B+C-D, G=A+B-C+D, and

H=A+B-C-D. In some embodiments, the "SADDSUBR4_QUAD8" ASE computes half of the needed branch metrics for code rate R=1/4 Viterbi decoding systems. A "SADDSUBR4N_QUAD8" ASE described below computes the other values. In some embodiments, the "SADDSUBR4_QUAD8" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating eight-bit arithmetic.

[0030] The "SADDSUBR4N_QUAD8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on four eight-bit input data blocks A, B, C, and D, which are packed into a thirty two-bit data word, to generate four eight-bit output data blocks E, F, G, and H, which are packed in a thirty two-bit output data word, where E=A-B+C+D, F=A-B+C-D, G=A-B-C+D, and H=A-B-C-D. In some embodiments, the "SADDSUBR4N_QUAD16" ASE computes half of the needed branch metrics for code rate R=1/4 Viterbi decoding systems. The "SADDSUBR4_QUAD8" ASE described above computes the other values. In some embodiments, the "SADDSUBR4N_QUAD8" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating eight-bit arithmetic.

[0031] A "SADDSUBR4_OCT8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding branch metric summing and subtracting operation on four eight-bit input data blocks A, B, C, and D, which are packed into a thirty two-bit input data word, to generate eight eight-bit output data blocks E, F, G, H, I, J, K, and L, which are packed in two thirty two-bit output data words, where E=A+B+C+D, F=A+B+C-D, G=A+B-C+D, H=A+B-C-D, I=A-B+C+D, J=A-B+C-D, K=A-B-C+D, and L=A-B-C-D. In some embodiments, the "SADDSUBR4_OCT8" ASE computes half of the needed branch metrics for code rate R=1/4 Viterbi decoding systems. The other branch metrics may be computed by negating some of the elements of the E, F, G, H, I, J, K, and L data blocks. In some embodiments, the "SADDSUBR4_OCT8" ASE may perform the Viterbi decoding branch metric summing and subtracting operation using saturating eight-bit arithmetic.

[0032] Embodiments of the ASEs 104 that are configured to perform Viterbi decoding bit manipulating operations are now described.

[0033] A "BIT_INTERLEAVE_DUAL16" ASE is configured to perform a Viterbi decoding bit interleaving operation on two sixteen-bit input data blocks A and B, which are packed into a thirty two-bit input data word, to generate a thirty two-bit output data words C, where each bit of C is taken in turn from A and B, for example, C[0]=A[0], C[1]=B[0], C[2]=A[1], C[3]=B[1], C[4]=A[2], C[5]=B[2], etc., which can be mathematically expressed as C[I]=B[(I-1)/2] and C[J]=A[J/2], where I is an odd integer from one and thirty one and J is an even integer from zero to thirty. Fig. 3 illustrates a part of the Viterbi decoding process using the "BIT_INTERLEAVE_DUAL16" ASE in accordance with an embodiment of the invention. As shown in Fig. 3, a sequence of dual sixteen-bit integer less or equal logic test (ILEQ) processes the results of dual ACS operations and generates decision bits that are not in order and need to be interleaved. The "BIT_INTERLEAVE_DUAL16" ASE collects the decision bits from the dual sixteen-bit ILEQ and interleaves the decision bits into decision words.

[0034] A "BIT_INTERLEAVE_QUAD8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding bit interleaving operation on four eight-bit input data blocks A, B, C, and D, which are packed into a thirty two bit input data word, to generate a thirty two-bit output data word E, where each bit of E is taken in turn from A, B, C, and D, for example, E[0] = A[0], E[1]=B[0], E[2]=C[0], E[3]=D[0], E[4]=A[1], AND E[5]=B[1]. In some embodiment, a sequence of quad eight-bit ILEQ may generate decision bit that are not in order and need to be interleaved and the "BIT_INTERLEAVE_QUAD8" ASE may collect the decision bits from the ILEQ and interleave the decision bits into decision words.

[0035] A "BIT_SHIFT_INTERLEAVE_DUAL16" ASE is configured to perform a Viterbi decoding bit shift interleaving operation on two sixteen-bit input data blocks A and B packed into a thirty two-bit first input data word and a thirty two-bit second input data word, which includes an integer N that is greater or equal to zero and smaller than thirty one, to generate a thirty two-bit output data word C, where C[N] = ext32b(A[0])<<N, C[N+1]=ext32b(B[0])<<N+1, and all other bits of C are reset to 0, where the ext32b function extends a bit into a thirty two-bit data word. In some embodiments, N is a Viterbi decoding state number.

Fig. 4 illustrates a part of the Viterbi decoding process using the "BIT_SHIFT_INTERLEAVE_DUAL16" ASE in accordance with an embodiment of the invention. As shown in Fig. 4, a sequence of dual sixteen-bit ILEQ processes the results of dual ACS operations and generates decision bits that are not in order and need to be interleaved. The "BIT_SHIFT_INTERLEAVE_DUAL16" ASE shifts the decision bits from the dual sixteen-bit ILEQ and interleaves the decision bits into decision words.

[0036] A "BIT_SHIFT_INTERLEAVE_QUAD8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding bit shift interleaving operation on four eight-bit input data blocks A, B, C, and D packed into a thirty two-bit first input data word, and a thirty two-bit second input data word, which includes an integer N that is greater or equal to zero and smaller than twenty nine, to generate a thirty two-bit output data word E, where E[N] = ext32b(A[0])<<N, E[N+1]=ext32b(B[0])<<N+1, E[N+2]=ext32b(C[0])<<N+2, E[N+3]=ext32b(D[0])<<N+2, and all other bits of E are reset to 0, where the ext32b function extends a bit into a thirty two-bit data word. In some embodiments, N is a Viterbi decoding state number. In some embodiments, the "BIT_SHIFT_INTERLEAVE_QUAD8" ASE shifts the decision bits from quad eight-bit ILEQ and interleaves the decision bits into decision words.

[0037] A "VSTATE2BIT" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding bit manipulating operation on a thirty two-bit unsigned integer input data word A and a thirty two-bit second input data word, which includes an integer N that is greater or equal to zero and smaller than thirty two, to generate a thirty two output data word B, where B= A| 1<<N. In some embodiments, N is a Viterbi decoding state number. In some embodiments, the "VSTATE2BIT" ASE performs the decoding bit manipulating operation at the traceback process to accumulate decoded bits packed into thirty two-bit words. In some embodiments, the "VSTATE2BIT" ASE used to set the N decoded bit in a thirty two-bit unsigned "decodedbits" input data word, using the parity of the current state stored in the "state" variable, if(state&0x1) decodedbits = VSTATE2BIT(decodedbits,N). In other words, if the "state" variable is odd, the Nth bit is set to one. If the "state" variable is even, the Nth bit is unchanged and left to its initial value, which should be zero.

[0038] A "VNEXTSTATE_LE" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding bit manipulating operation on a thirty two-bit unsigned integer first input data word and a thirty two-bit second input data word, which includes an integer N that is greater or equal to zero and smaller than thirty two, to generate a thirty two-bit output data word, where N is the current state for the current step and the output is the most likely state of the previous step (this is part of backtracking, i.e., going backwards through the steps produced by the ACS process, where each step corresponds to one decoded bit). Each state has two possible next states in the previous step.

[0039] Fig. 5 and Fig. 6 illustrate a part of the Viterbi decoding process using the "VNEXTSTATE_LE" application specific extension in the general purpose processor on a little endian ("LE") add-compare-select (ACS) pattern and a "LE" traceback pattern in accordance with an embodiment of the invention. Fig. 5 shows a "LE" ACS pattern with which the ACS process creates the path metric and the decision bits. The traceback process is going backwards through this pattern. As shown in Fig. 5, the "LE" ACS pattern includes the path metric for the state pair <i, i+M/2> used to compute the path metric for the state pair <2i, 2i+1>, where i is a "state" variable and "M" is the total number of possible states. Fig. 6 shows a "LE" traceback pattern, which can be performed by the "VNEXTSTATE_LE" ASE. As shown in Fig. 6, the "LE" traceback pattern goes backwards from <2i, 2i+1> to <i, i+M/2>. In other words, the "LE" traceback pattern uses the decision bit for one state in a <2i, 2i+1> pair. The "LE" traceback pattern sets the "next state" to be an upper state <i> if the decision bit for the current state is zero. The "LE" traceback pattern sets the "next state" to be a lower state <i+M/2> if the decision bit for the current state is set to one.

[0040] In some embodiments, N is a Viterbi decoding state number. In some embodiments, the "VNEXTSTATE_LE" ASE is used for instance to find which state is most likely to be preceding the current state, using the value of the decision bit for that state, nextstate = VNEXTSTATE_LE (decisions, state). As used herein, "next state" means the next state after the current state during the traceback process, which is actually the previous state of the current state. The Viterbi decoding bit manipulating operation performed by the "VNEXTSTATE_LE" ASE may be described by the following C code excerpt,

```
unsigned int VNEXTSTATE_LE(unsigned int decisions, unsigned int state)
{
        If(decision & 1<<state) {
                If(state&1) return (state – 1)/2 + M/2;
                Else return state/2 + M/2;
        } else {
                If (state & 1) return (state -1)/2;
                Else return state/2;
        }
}
```

[0041] Fig. 7 and Fig. 8 illustrate a part of Viterbi decoding using a "VNEXTSTATE_BE" application specific extension in the general purpose processor on a big endian ("BE") add-compare-select pattern and a "BE" traceback in accordance with an embodiment of the invention. Fig. 7 shows a "BE" ACS pattern with which the ACS process has created the path metric and the decision bits. The "BE" ACS pattern takes the path metric for the state pair <2i, 2i+1> and computes the metric for the state pair <i, i+M/2>. Fig. 8 shows a "BE" traceback pattern that can be performed by a "VNEXTSTATE_BE" ASE. The BE traceback pattern goes backwards: if the current state if one of <i, i+M/2>, the "next" state in the traceback is an "upper" state <2i> if the decision bit for the current state is zero, and the next state is an "lower" state <2i+1> if the decision bit for the current state is one. In some embodiments, the "VNEXTSTATE_BE" ASE is used for instance to find which state is most likely to be preceding the current state, using the value of the decision bit for that state, nextstate = VNEXTSTATE_BE (decisions, state), where N is a Viterbi decoding state number. The Viterbi decoding bit manipulating operation performed by the "VNEXTSTATE_BE" ASE may be described by the following C code excerpt,

```
unsigned int VNEXTSTATE_BE(unsigned int decisions, unsigned int state)
{
        If (decision & 1<<state)
        {
                If (state & M/2) return 2* state – M/2 +1;
                Else return 2*state +1;
```

```
} else {
If (state & M/2) return 2* state – M/21;
        Else return 2*state ;
}
}
```

[0042] An "ORI_QUAD32" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding bit manipulating operation on four thirty two-bit input data words A, B, C, D to generate a thirty two-bit output data word E, which is the logical OR combination of the four input data words into, E=A|B|C|D. In some embodiment, the "ORI_QUAD32" ASE is used to combine results from the ASEs performing Viterbi decoding bit interleaving operations and the ASEs Viterbi decoding bit shift interleaving operations described above.

[0043] Embodiments of the ASEs that are configured to perform Viterbi decoding ACS operations are now described.

[0044] A "VACS_DUAL16" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding ACS operation on four sixteen-bit path metric input data blocks packed in two thirty two-bit input data words and also using two branch metric data blocks packed into a thirty two-bit branch metric data word to generate four sixteen-bit path metric output data block packed in two thirty two-bit output data words. The "VACS_DUAL16" ASE performs the Viterbi decoding ACS operation in parallel on the least significant bit (LSB) side of the path metric input data words and branch metric data words and on the most significant bit (MSB) side of the path metric input and branch metric data words. The Viterbi decoding ACS operation performed by the "VACS_DUAL16" ASE may be described by the following C code excerpt,

```
VACS_DUAL16 (&output1,&output2,input1,input2,bmetric)
{
lsb(output1) = max(lsb(input1)+lsb(bmetric),lsb(input2)-lsb(bmetric));
msb(output1) = max(lsb(input1)-lsb(bmetric),lsb(input2)+lsb(bmetric));
lsb(output2) = max(msb(input1)+msb(bmetric),msb(input2)-msb(bmetric));
msb(output2) = max(msb(input1)-msb(bmetric),msb(input2)+msb(bmetric));
}
```

[0045] A "VACS_QUAD8" ASE in accordance with an embodiment of the invention is configured to perform a Viterbi decoding ACS operation on eight eight-bit input data blocks packed in two thirty two-bit input data words using four eight-bit branch metric data blocks packed into a thirty two bit branch metric data word to generate eight eight-bit output data blocks packed in two thirty two-bit output data words. The "VACS_QUAD8" ASE performs the Viterbi decoding ACS operation in parallel on the LSB side of the input and branch metric data words and on the MSB side of the input and branch metric data words.

[0046] A "VDECISION_DUAL16" ASE in accordance with an embodiment of the invention is configured to process four sixteen-bit input data blocks packed into two thirty two-bit path metric input data words and also using two sixteen-bit branch metric data blocks packed into a thirty two-bit branch metric data word and a Viterbi decoding state number N, which is an integer that is greater or equal to zero and smaller than thirty two, to generate four Viterbi decoding decision bits for the four Viterbi decoding states, N, N+1, N+2, and N+3, where each Viterbi decoding decision bit is included in a thirty two-bit Viterbi decoding decision data word. The "VDECISION_DUAL16" ASE performs the Viterbi decoding operation in parallel on the LSB side of the path metric input and branch metric data words and on the MSB side of the path metric input and branch metric data words. The Viterbi decoding operation performed by the "VDECISION_DUAL16" ASE may be described by the following C code excerpt,

```
int VDECISION_DUAL16(input1,input2,bmetric,n)
{
UInt32 tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7, tmp8;
UInt32 tmpdec1, tmpdec2, tmpdec3, tmpdec4;
tmp1 = lsb(input1) + lsb(bmetric);
tmp2 = lsb(input1) - lsb(bmetric);
tmp3 = lsb(input2) - lsb(bmetric);
tmp4 = lsb(input2) + lsb(bmetric);
tmp5 = msb(input1) + msb(bmetric);
tmp6 = msb(input1) - msb(bmetric);
tmp7 = msb(input2) - msb(bmetric);
```

```
tmp8 = msb(input2) + msb(bmetric);
tmpdec1 = tmp1 >= tmp3 ? 1<<n  : 0;
tmpdec2 = tmp2 >= tmp4 ? 1<<n+1 : 0;
tmpdec3 = tmp5 >= tmp7 ? 1<<n+2 : 0;
tmpdec4 = tmp6 >= tmp8 ? 1<<n+3 : 0;
return  tmpdec1|tmpdec2|tmpdec3|tmpdec4;
}
```

[0047] A "VDECISION_QUAD8" ASE in accordance with an embodiment of the invention is configured to process four eight-bit input data blocks using four eight-bit branch metric data blocks packed into a thirty two-bit branch metric data word and a Viterbi state data word to generate eight Viterbi decoding decision bits for the Viterbi decoding states N to N+7. The Viterbi state data word includes a state number N that is an integer greater or equal to zero and smaller than thirty two Each of the eight Viterbi decoding decision bits is included in a thirty two-bit Viterbi decoding decision data word.

[0048] Fig. 9 is a schematic flow chart diagram of a method for Viterbi decoding using application specific extensions in accordance with an embodiment of the invention. At block 900, a Viterbi decoding algorithm is obtained, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations. At block 902, the plurality of Viterbi decoding operations are exclusively performed within a general purpose processor using a plurality of application specific extensions in the general purpose processor.

[0049] Although the operations of the method herein are shown and described in a particular order, the order of the operations of the method may be altered so that certain operations may be performed in an inverse order or so that certain operations may be performed, at least in part, concurrently with other operations. In another embodiment, instructions or sub-operations of distinct operations may be implemented in an intermittent and/or alternating manner.

[0050] In addition, although specific embodiments of the invention that have been described or illustrated include several components described or illustrated herein, other embodiments of the invention may include fewer or more components to implement less or more functionality.

[0051] Furthermore, the invention is not to be limited to the specific forms or arrangements of parts so described and illustrated. The scope of the invention is to be defined by the claims appended hereto and their equivalents.

What is claimed is:

1. A Viterbi decoding system comprising:

memory configured to store a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations; and

a general purpose processor comprising a plurality of application specific extensions, wherein each application specific extension is configured to perform at least one of the Viterbi decoding operations specified in the Viterbi decoding algorithm stored in the memory,

wherein the Viterbi decoding system is configured such that all the Viterbi decoding operations specified in the Viterbi decoding algorithm are performed exclusively within the general purpose processor using at least one of the application specific extensions.

2. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding decision bits generating operation in parallel on least significant bit parts of two path metric input data words and most significant bit parts of the two path metric input data words and also using a branch metric data word to generate four decision bits for four Viterbi decoding states.

3. The Viterbi decoding system of claim 2, wherein each least significant bit part of the two input data words includes sixteen bits and each most significant bit part of the two input data words includes sixteen bits.

4. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding add-compare-select operation in parallel on least significant bit parts of two path metric input data words and a branch metric data word and most significant bit parts of the two path metric input data words and the branch metric data word to generate a first path metric output data word and a second path metric output data word.

5. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding bit interleaving operation on a first input data block and a second input data block to generate an output data word, where each input data block has sixteen bits ranging from bit number zero to bit number fifteen, the first input data block and the second input data block being packed into an input data word that has thirty two bits, the output data word having thirty two bits ranging from bit number zero to bit number thirty one, the number I bit of the output data word being equal to the number (I-1)/2 bit of the second input data block, and the number J bit of the output data word being equal to the number J/2 bit of the first input data block, where I is an odd integer from one and thirty one and J is an even integer from zero and thirty.

6. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extension is configured to perform a Viterbi decoding bit shift interleaving operation on a first input data block and a second input data block using a Viterbi decoding state number N to generate an output data word, where each input data block has sixteen bits ranging from bit number zero to bit number fifteen, the first input data block and the second input data block being packed into a first input data word that has thirty two bits, where N is an integer that is greater than or equal to zero and smaller than thirty one, the Viterbi decoding number N being in a second input data word that has thirty two bits, the output data word having thirty two bits ranging from bit number zero to bit number thirty one, where the bit number N of the output data word is equal to the bit number zero of the first input data block, the bit number N+1of the output data word being equal to the bit number zero of the second input data block, and all other bits of the output data words being reset to zero.

7. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extensions is configured to process a current Viterbi decoding state and a Viterbi decoding decision bits to generate a next Viterbi decoding state, wherein the next Viterbi decoding state is

an upper Viterbi decoding state if the Viterbi decoding decision bit is set to zero or a lower Viterbi decoding state if the Viterbi decoding decision bit is set to one.

8. The Viterbi decoding system of claim 1, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding branch metric calculating operation on a first input data block and a second input data block to generate a first output data block and a second output data block, where each input data block has sixteen bits, the first input data block and the second input data block being packed into an input data word that has thirty two bits, each output data block having sixteen bits, the first output data block and the second output data block being packed into an output data word that has thirty two bits, the first output data block being equal to the addition of the first input data block and the second input data block, and the second output data block being equal to the subtraction of the first input data block and the second input data block.

9. A method for Viterbi decoding using application specific extensions, the method comprising:

obtaining a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations; and

exclusively performing the plurality of Viterbi decoding operations within a general purpose processor using a plurality of application specific extensions in the general purpose processor.

10. The method of claim 9, wherein the exclusively performing includes performing a Viterbi decoding decision bits generating operation in parallel on least significant bit parts of two path metric input data words and most significant bit parts of the two path metric input data words and also using a branch metric data word to generate four decision bits for four Viterbi decoding states using at least one of the application specific extensions in the general purpose processor.

11. The method of claim 9, wherein the exclusively performing includes performing a Viterbi decoding add-compare-select operation in parallel on least

significant bit parts of two path metric input data words and a branch metric data word and most significant bit parts of the two path metric input data words and the branch metric data word to generate a first path metric output data word and a second path metric output data word using at least one of the application specific extensions in the general purpose processor.

12. The method of claim 9, wherein the exclusively performing includes performing a Viterbi decoding bit interleaving operation on a first input data block and a second input data block to generate an output data word using at least one of the application specific extensions in the general purpose processor, where each input data block has sixteen bits ranging from bit number zero to bit number fifteen, the first input data block and the second input data block being packed into an input data word that has thirty two bits, the output data word having thirty two bits ranging from bit number zero to bit number thirty one, the number I bit of the output data word being equal to the number (I/1)/2 bit of the second input data block, and the number J bit of the output data word being equal to the number J/2 bit of the first input data block, where I is an odd integer from one and thirty one and J is an even integer from zero and thirty.

13. The method of claim 9, wherein the exclusively performing includes performing a Viterbi decoding bit shift interleaving operation on a first input data block and a second input data block using a Viterbi decoding state number N to generate an output data word using at least one of the application specific extensions in the general purpose processor, where each input data block has sixteen bits ranging from bit number zero to bit number fifteen, the first input data block and the second input data block being packed into a first input data word that has thirty two bits, N being an integer that is greater than or equal to zero and smaller than thirty one, the Viterbi decoding number N being in a second input data word that has thirty two bits, the output data word having thirty two bits ranging from bit number zero to bit number thirty one, where the bit number N of the output data word is equal to the bit number zero of the first input data block, the bit number N+1 of the output data word being equal to the bit number zero of

the second input data block, and all other bits of the output data words being reset to zero.

14. The method of claim 9, wherein the exclusively performing includes processing a current Viterbi decoding state and a Viterbi decoding decision bits to generate a next Viterbi decoding state using at least one of the application specific extensions in the general purpose processor, wherein the next Viterbi decoding state is an upper Viterbi decoding state if the Viterbi decoding decision bit is set to zero or a lower Viterbi decoding state if the Viterbi decoding decision bit is set to one.

15. The method of claim 9, wherein the exclusively performing includes performing a Viterbi decoding branch metric calculating operation on a first input data block and a second input data block to generate a first output data block and a second output data block using at least one of the application specific extensions in the general purpose processor, where each input data block has sixteen bits, the first input data block and the second input data block being packed into an input data word that has thirty two bits, each output data block having sixteen bits, the first output data block and the second output data block being packed into an output data word that has thirty two bits, the first output data block being equal to the addition of the first input data block and the second input data block, and the second output data block being equal to the subtraction of the first input data block and the second input data block.

16. A Viterbi decoding system comprising:

memory configured to store a Viterbi decoding algorithm, wherein the Viterbi decoding algorithm specifies a plurality of Viterbi decoding operations; and

a general purpose processor comprising a processor core and a plurality of application specific extensions, wherein the processor core includes a plurality of functional units and each application specific extension is configured to perform one of the Viterbi decoding operations specified in the Viterbi decoding algorithm stored in the memory,

wherein the Viterbi decoding system is configured such that all the Viterbi decoding operations specified in the Viterbi decoding algorithm are performed exclusively within the general purpose processor using at least one of the application specific extensions.

17. The Viterbi decoding system of claim 16, wherein the plurality of functional units include an addition/subtraction functional unit configured to perform addition functions and subtraction functions, a multiplication functional unit configured to perform multiplication functions, and an arithmetic logic functional unit configured to perform logic functions.

18. The Viterbi decoding system of claim 16, wherein the plurality of application specific extensions are configured to reuse at least one of the functional units of the processor core.

19. The Viterbi decoding system of claim 16, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding decision bits generating operation in parallel on least significant bit parts of two path metric input data words and most significant bit parts of the two path metric input data words and also using a branch metric data word to generate four decision bits for four Viterbi decoding states.

20. The Viterbi decoding system of claim 16, wherein a particular application specific extension of the plurality of application specific extensions is configured to perform a Viterbi decoding add-compare-select operation in parallel on least significant bit parts of two path metric input data words and a branch metric data word and most significant bit parts of the two path metric input data words and the branch metric data word to generate a first path metric output data word and a second path metric output data word.

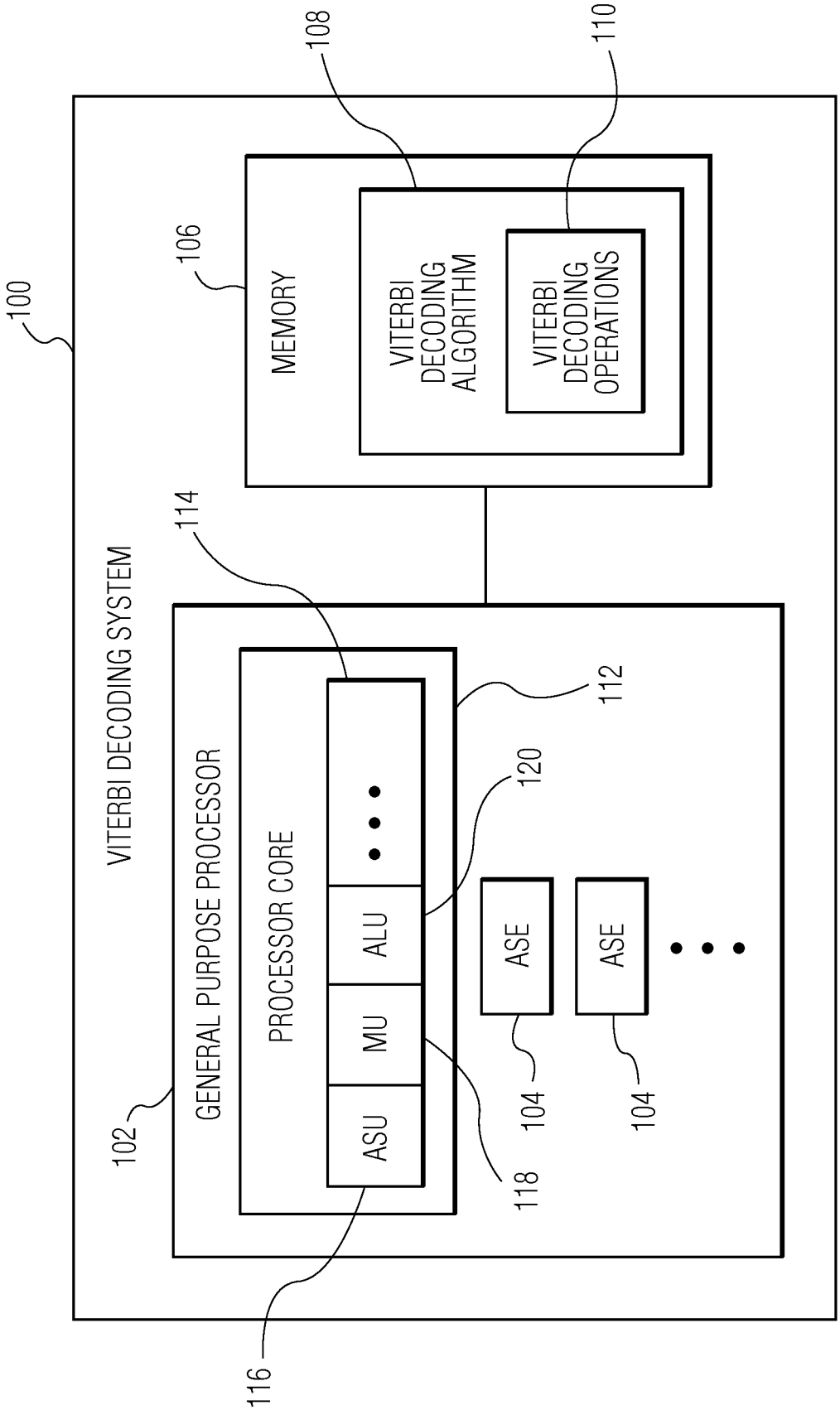
100
VITERBI DECODING SYSTEM
106
MEMORY
108
VITERBI DECODING ALGORITHM
110
VITERBI DECODING OPERATIONS
114
GENERAL PURPOSE PROCESSOR
102
PROCESSOR CORE
112
ASU
MU
ALU
120
ASE
ASE
104
104
118
116

FIG. 1

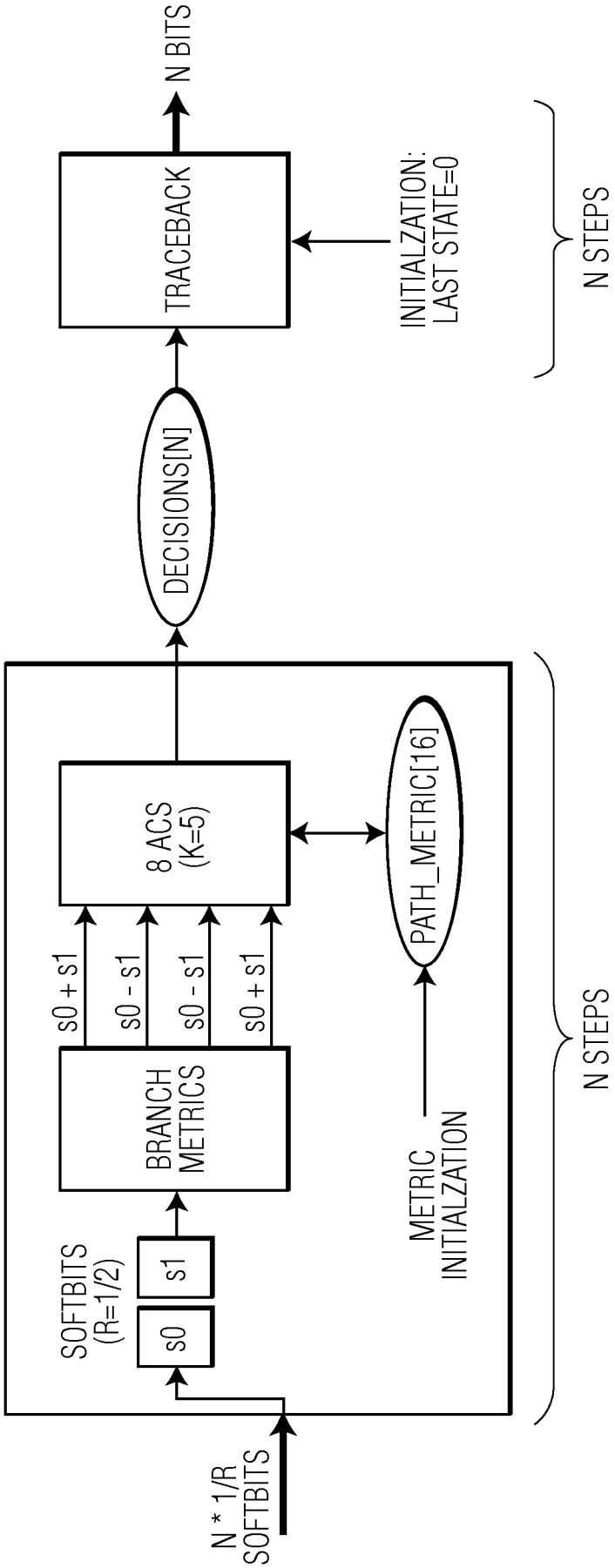
N BITS
TRACEBACK
INITIALZATION:
LAST STATE=0
N STEPS
DECISIONS[N]
8 ACS
(K=5)
PATH_METRIC[16]
s0 + s1
s0 - s1
s0 - s1
s0 + s1
BRANCH
METRICS
SOFTBITS
(R=1/2)
s0
s1
METRIC
INITIALZATION
N STEPS
N * 1/R
SOFTBITS

FIG. 2

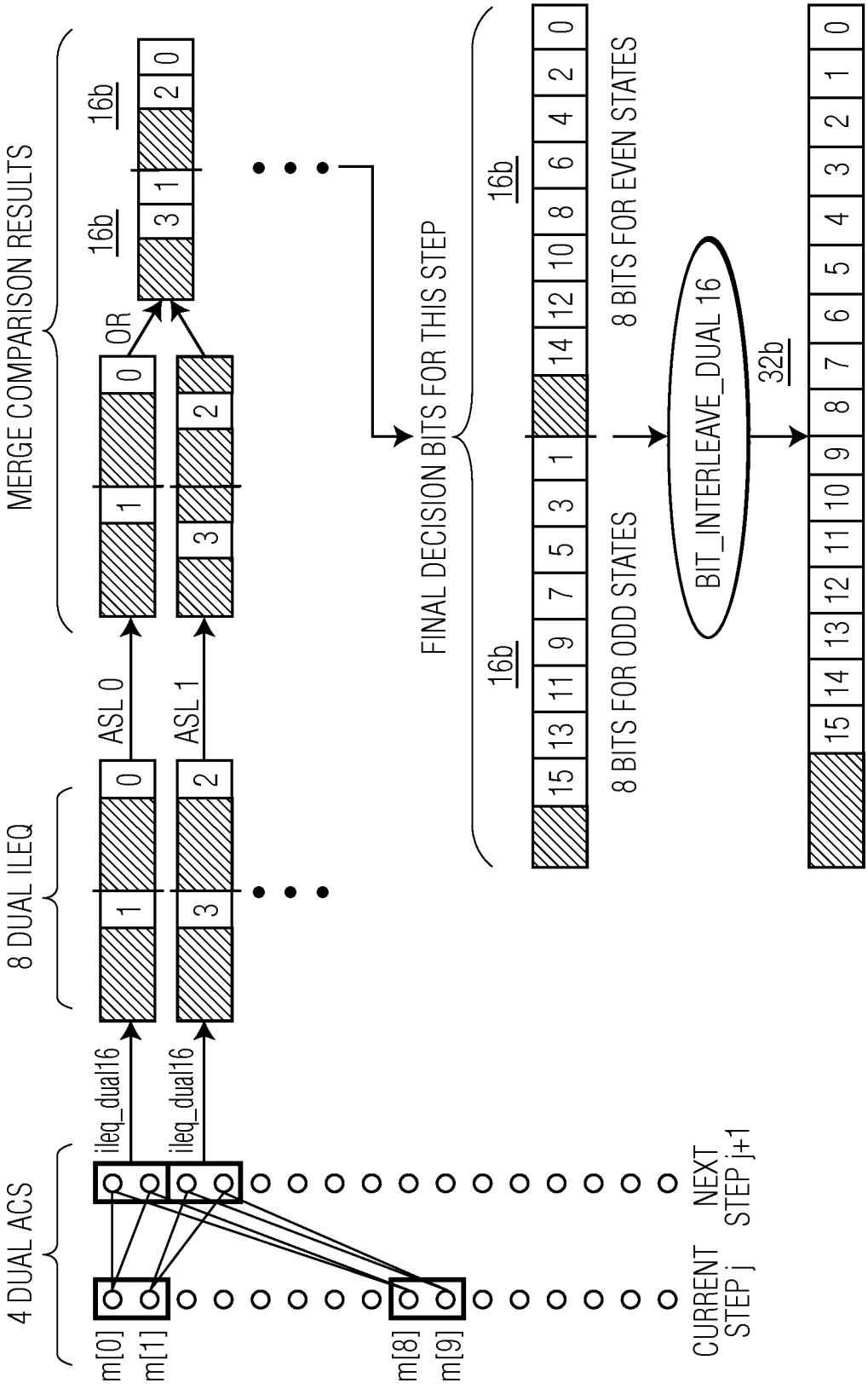
4 DUAL ACS
8 DUAL ILEQ
MERGE COMPARISON RESULTS
ileq_dual16
ileq_dual16
ASL 0
ASL 1
OR
16b
16b
16b
16b
m[0]
m[1]
m[8]
m[9]
CURRENT STEP j
NEXT STEP j+1
FINAL DECISION BITS FOR THIS STEP
8 BITS FOR ODD STATES
8 BITS FOR EVEN STATES
BIT_INTERLEAVE_DUAL 16
32b

FIG. 3

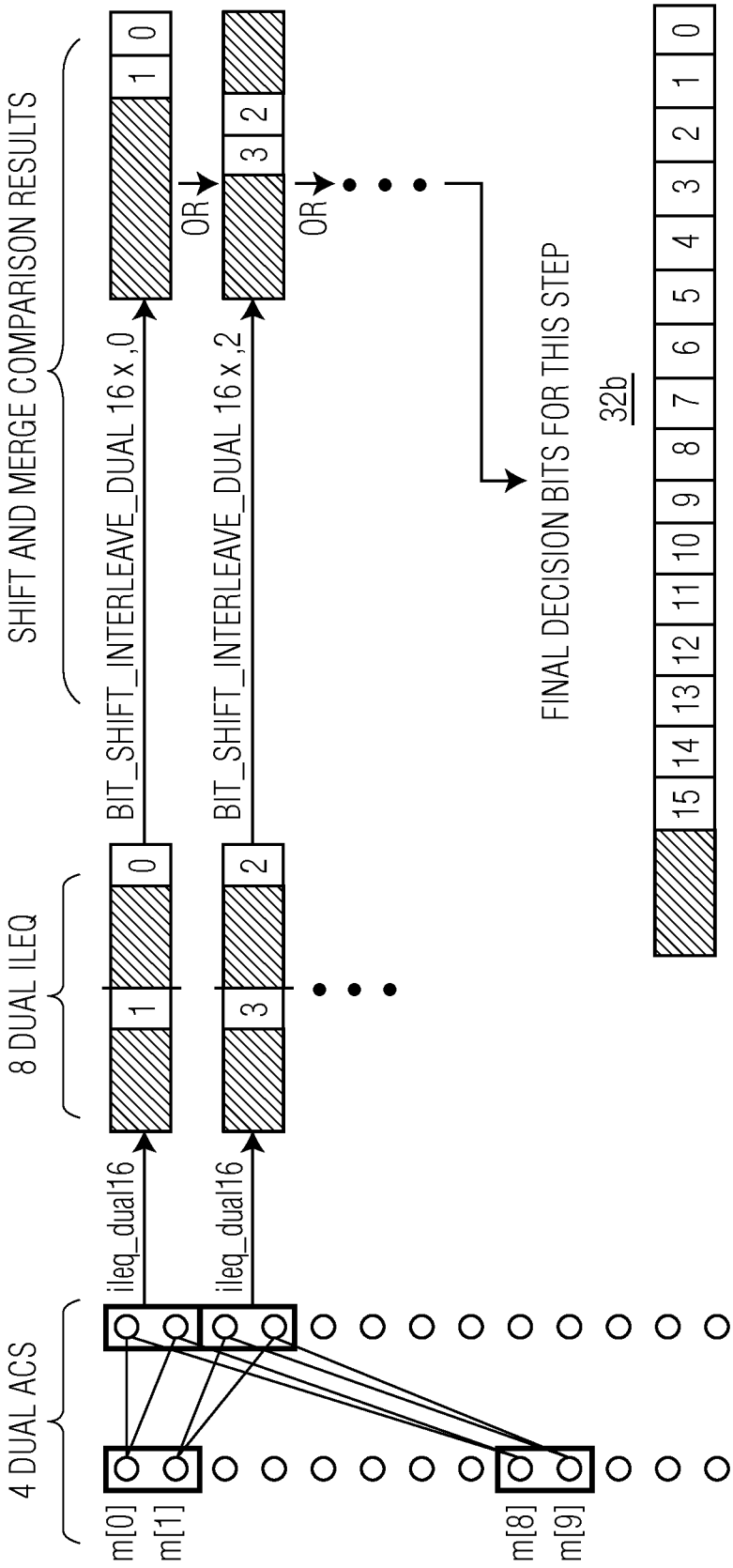
4 DUAL ACS
8 DUAL ILEQ
SHIFT AND MERGE COMPARISON RESULTS
ileq_dual16
ileq_dual16
BIT_SHIFT_INTERLEAVE_DUAL 16 x ,0
BIT_SHIFT_INTERLEAVE_DUAL 16 x ,2
OR
OR
FINAL DECISION BITS FOR THIS STEP
32b
m[0]
m[1]
m[8]
m[9]
CURRENT STEP j
NEXT STEP j+1

FIG. 4

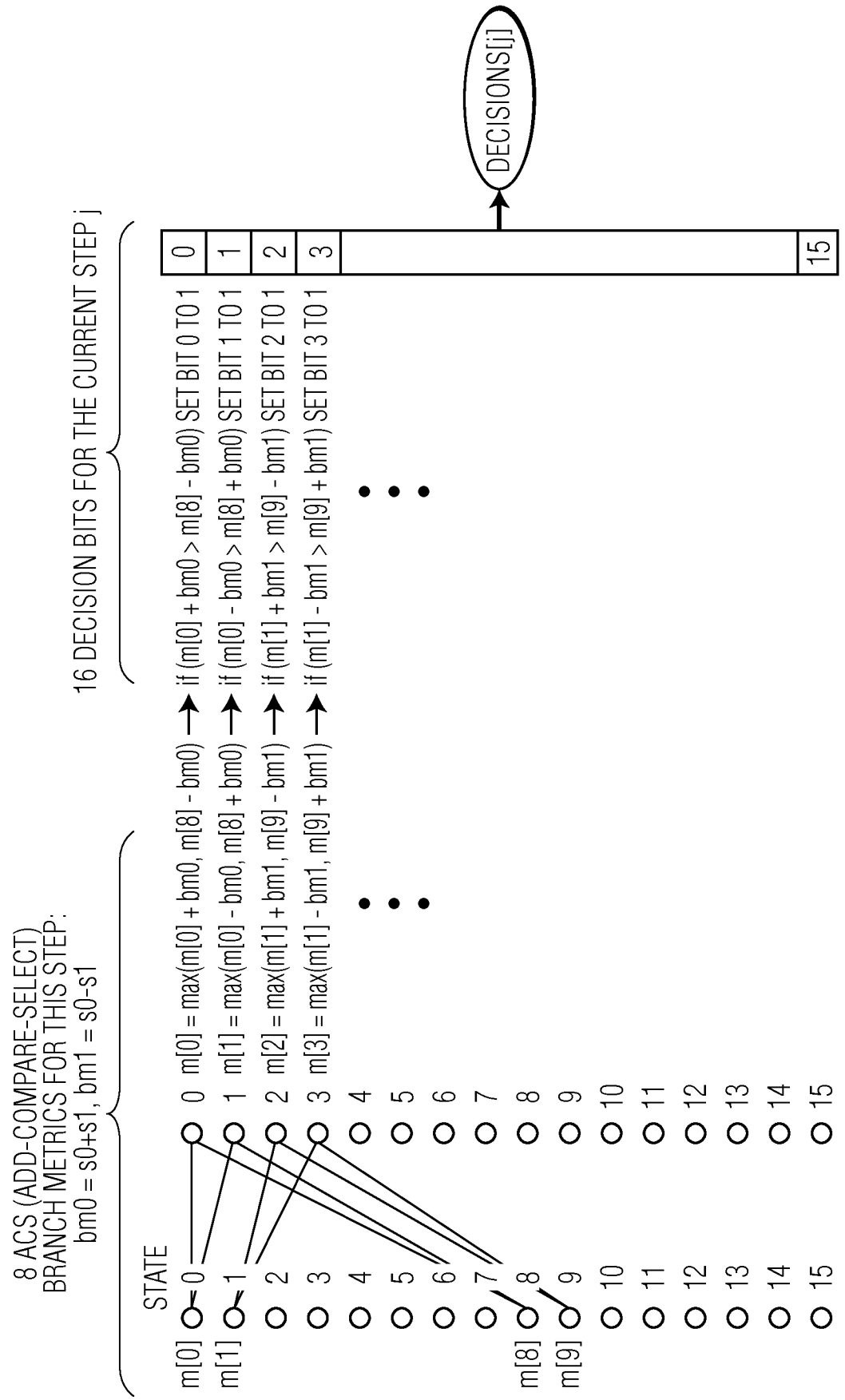
8 ACS (ADD-COMPARE-SELECT)
BRANCH METRICS FOR THIS STEP:
bm0 = s0+s1, bm1 = s0-s1
16 DECISION BITS FOR THE CURRENT STEP j
STATE
m[0]
m[1]
m[8]
m[9]
CURRENT STEP j
NEXT STEP j+1
0 m[0] = max(m[0] + bm0, m[8] - bm0) if (m[0] + bm0 > m[8] - bm0) SET BIT 0 TO 1
1 m[1] = max(m[0] - bm0, m[8] + bm0) if (m[0] - bm0 > m[8] + bm0) SET BIT 1 TO 1
2 m[2] = max(m[1] + bm1, m[9] - bm1) if (m[1] + bm1 > m[9] - bm1) SET BIT 2 TO 1
3 m[3] = max(m[1] - bm1, m[9] + bm1) if (m[1] - bm1 > m[9] + bm1) SET BIT 3 TO 1
0
1
2
3
15
DECISIONS[j]

FIG. 5

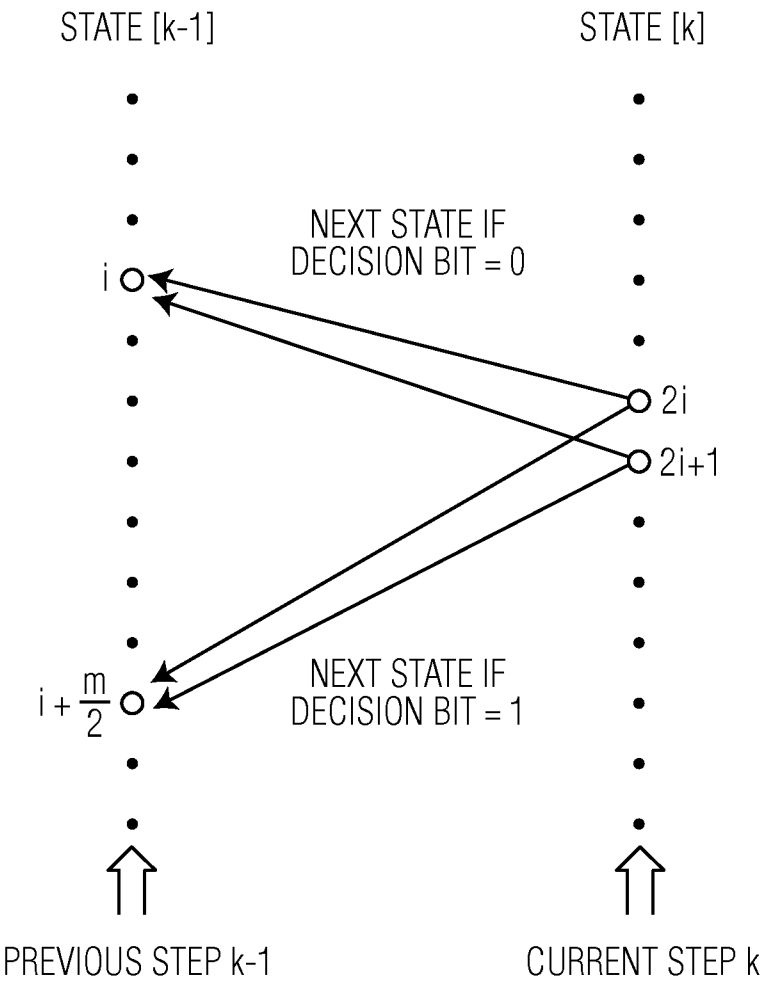
STATE [k-1]
STATE [k]
NEXT STATE IF
DECISION BIT = 0
i
2i
2i+1
i + m/2
NEXT STATE IF
DECISION BIT = 1
PREVIOUS STEP k-1
CURRENT STEP k

FIG. 6

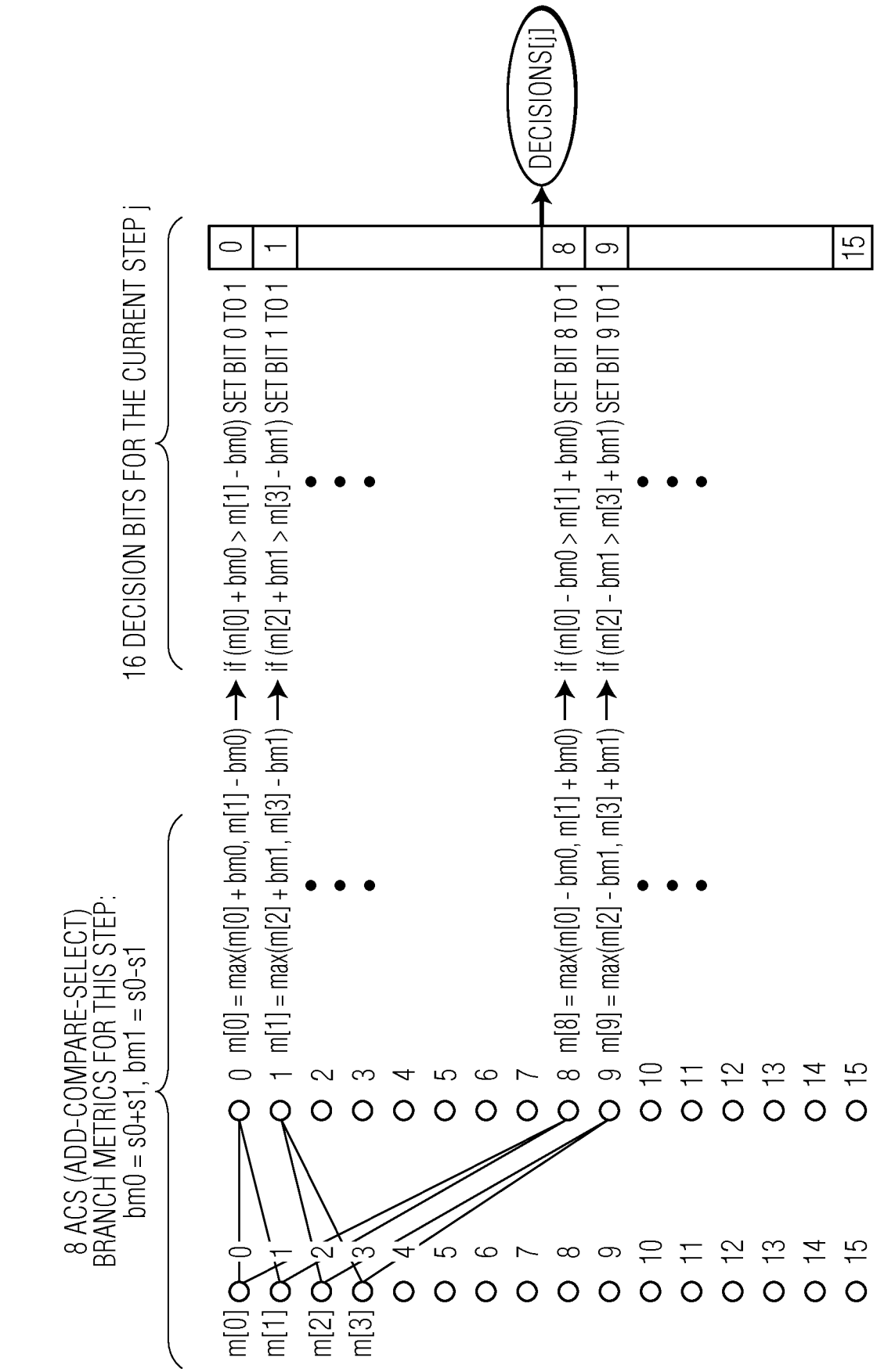
8 ACS (ADD-COMPARE-SELECT)
BRANCH METRICS FOR THIS STEP:
bm0 = s0+s1, bm1 = s0-s1
16 DECISION BITS FOR THE CURRENT STEP j
m[0]
m[1]
m[2]
m[3]
CURRENT STEP j
NEXT STEP j+1
0 m[0] = max(m[0] + bm0, m[1] - bm0) → if (m[0] + bm0 > m[1] - bm0) SET BIT 0 TO 1
1 m[1] = max(m[2] + bm1, m[3] - bm1) → if (m[2] + bm1 > m[3] - bm1) SET BIT 1 TO 1
8 m[8] = max(m[0] - bm0, m[1] + bm0) → if (m[0] - bm0 > m[1] + bm0) SET BIT 8 TO 1
9 m[9] = max(m[2] - bm1, m[3] + bm1) → if (m[2] - bm1 > m[3] + bm1) SET BIT 9 TO 1
DECISIONS[j]
0
1
8
9
15

FIG. 7

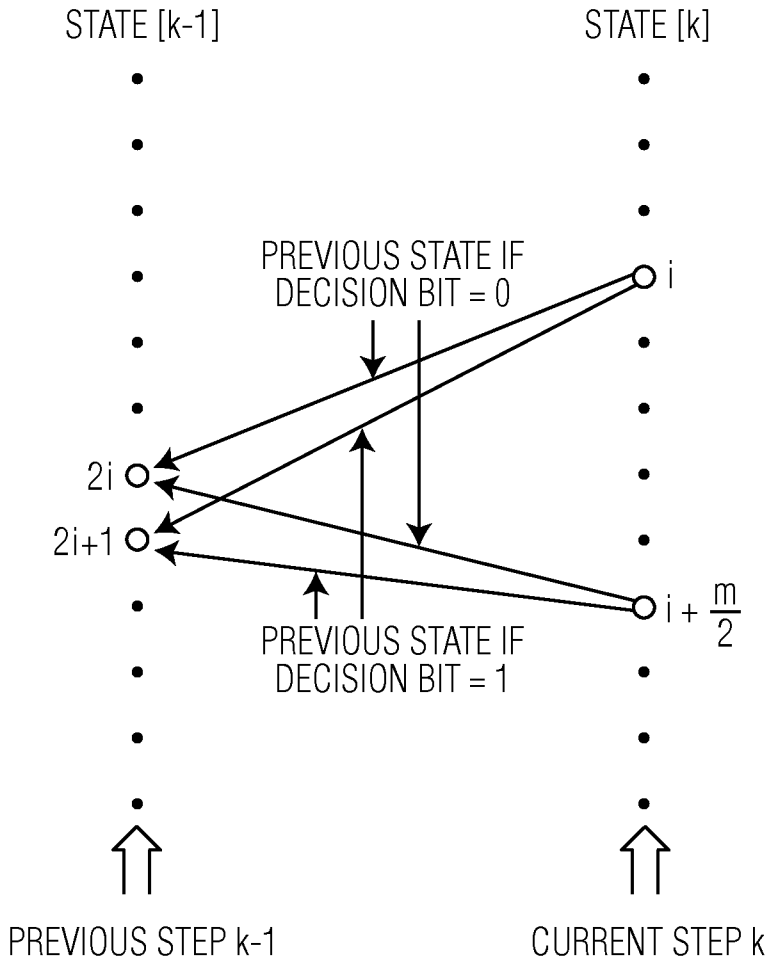
STATE [k-1]
STATE [k]
PREVIOUS STATE IF DECISION BIT = 0
i
2i
2i+1
i + m/2
PREVIOUS STATE IF DECISION BIT = 1
PREVIOUS STEP k-1
CURRENT STEP k

FIG. 8

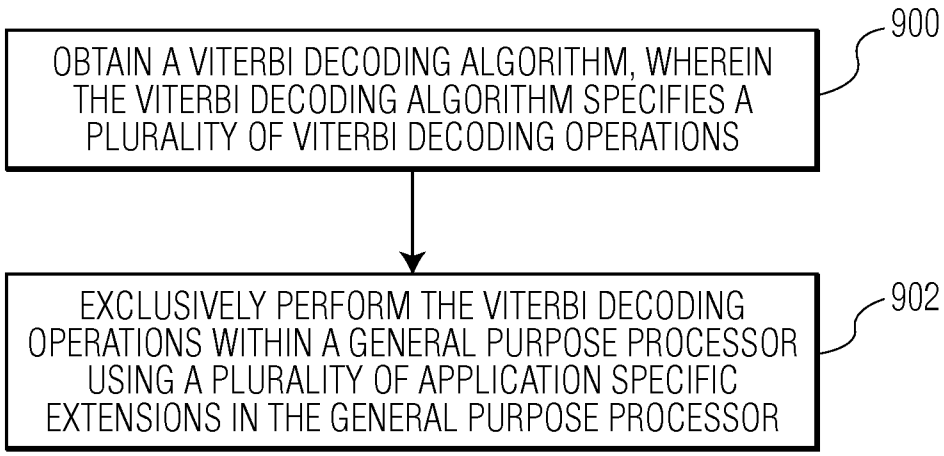
OBTAIN A VITERBI DECODING ALGORITHM, WHEREIN THE VITERBI DECODING ALGORITHM SPECIFIES A PLURALITY OF VITERBI DECODING OPERATIONS
900
EXCLUSIVELY PERFORM THE VITERBI DECODING OPERATIONS WITHIN A GENERAL PURPOSE PROCESSOR USING A PLURALITY OF APPLICATION SPECIFIC EXTENSIONS IN THE GENERAL PURPOSE PROCESSOR
902

FIG. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2009/055472

A. CLASSIFICATION OF SUBJECT MATTER
INV. H03M13/41

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Phil Karn: "fec-3.0.1.tar.bz2" internet article 13 October 2006 (2006-10-13), XP002574333 internet Retrieved from the Internet: URL:http://www.ka9q.net/code/fec/> [retrieved on 2010-03-22] the whole document viterbi*.*, *.c ----- -/--	1,9,16

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 March 2010	06/04/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Rydyger, Kay

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2009/055472

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Intel Corporation: "using the streaming simd extensions 2 (SSE2) to evaluate a hidden markov model with viterbi decoding" internet article 9 September 2000 (2000-09-09), XP002574334 Retrieved from the Internet: URL:http://homepages.cae.wisc.edu/{ece734/mmx/w_hmm_viterbi.pdf> [retrieved on 2010-03-22] the whole document	1,9,16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2009/055472

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☒ Claims Nos.: 2-8, 10-15, 17-20
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
see FURTHER INFORMATION sheet PCT/ISA/210

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

Continuation of Box II.2

Claims Nos.: 2-8, 10-15, 17-20

The dependent claims are directed to particular "application specific extensions" carrying out, when a corresponding programme is run on the processor, functions apparently related to Viterbi processing. Independent claim 1 is directed to a "Viterbi decoding system" comprising a general purpose processor with application specific extensions. Said extensions therefore denote computer/processor hardware. The dependent claims further define the processor extensions to be "configured" so as to carry out particular functions. Leaving aside the lack of clarity as to when a processor or processor extensions is "configured" to provide a certain functionality (a general purpose processor cannot be "configured" other than running a particular computer programme on it), the application as a whole does not disclose the corresponding hardware realisations of said specific extensions. The figures of the application, for example, disclose the general concept of the functionality to be implemented in "application specific extensions", however, given the complexity of such processors, it is not possible to put the invention into practise without undue burden. Accordingly, an international search has not been carried out.

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guideline C-VI, 8.2), should the problems which led to the Article 17(2) declaration be overcome.