

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
2 October 2008 (02.10.2008)

PCT

(10) International Publication Number
WO 2008/118286 A2

(51) International Patent Classification:
G06F 17/17 (2006.01)

(74) Agents: IANDIORIO, Joseph, S. et al.; Iandiorio Teska & Coleman, 260 Bear Hill Road, Waltham, MA 02454 (US).

(21) International Application Number:
PCT/US2008/003153

(22) International Filing Date: 11 March 2008 (11.03.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/728,427 26 March 2007 (26.03.2007) US

(71) Applicant (for all designated States except US): **ANALOG DEVICES, INC.** [US/US]; Three Technology Way, Norwood, MA 02062 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **WILSON, James** [US/US]; 8 Shepherd Street, Foxboro, MA 02035 (US). **STEIN, Yosef** [IL/US]; 4 Turning Mill Road, Sharon, MA 02067 (US). **KABLOTSKY, Joshua** [US/US]; 99 Nickles Lane, Carlisle, MA 01741 (US). **PRENDERGAST, Colm, J.** [US/US]; 43 Bigelow Street, Cambridge, MA 02139 (US). **YUKNA, Gregory, M.** [US/US]; 234 Plain Street, Norton, MA 02766 (US). **MAYER, Christopher, M.** [US/US]; 1 Haven Street, Dover, MA 02030 (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, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

(54) Title: IMPROVED METHOD AND APPARATUS FOR ACCELERATING PROCESSING OF ADAPTIVE FINITE IMPULSE RESPONSE FILTERING

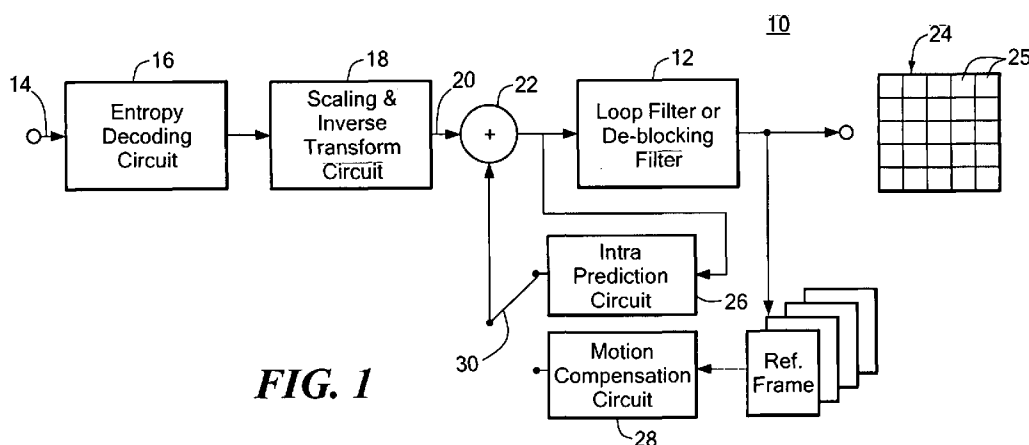


FIG. 1

(57) Abstract: Finite impulse response filtering is achieved by broadcasting to at least one compute unit an instruction having a plurality of data samples, a conditional field associated with each compute unit, and a set of operator values for operating on each data sample; providing a function of each the data sample in accordance with an associated set of operator values identified by the conditional field; and combining the functions to obtain an intermediate finite impulse response of the data samples.

IMPROVED METHOD AND APPARATUS FOR ACCELERATING PROCESSING OF ADAPTIVE FINITE IMPULSE RESPONSE FILTERING

FIELD OF THE INVENTION

This invention relates to an improved method and apparatus architecture in the instruction domain for accelerating processing and more particularly to such an improved instruction architecture especially useful with processors with multiple compute units and in parallel processing of multiple data samples.

BACKGROUND OF THE INVENTION

Video compression involves encoding/decoding of pixel information in 16x16 pixels macroblocks. The new emerging standards like (MPEG4, H.264, and Windows Media) provide a flexible tiling structure in a macroblock. It allows the use of 16 x 16, 16 x 8, 8 x 16, 8 x 8, 8 x 4, 4 x 8, and 4 x 4 sub-macroblock sizes. A Finite Impulse Response (FIR) filter (de-blocking filter) is applied to every decoded macroblock edge to reduce blocking distortion resulting from the prediction and residual difference coding stages of the decoding process. The filter is applied on both 4 x 4 block and 16 x 16 macroblock boundaries, in which three pixels on either side of the boundary may be updated using a five-tap filter. The filter coefficients set or "strength" are governed by a content adaptive non-linear filtering scheme. This is done in a number of ways. Windows Media Video decoder (wmv) uses one protocol involving the boundary strength across block boundaries. H.264 or MPEG-4 part 10 uses pixel gradient across block

boundaries.

The de-blocking filter has two benefits: block edges are smoothed, improving the appearance of decoded images (particularly at higher compression ratios). And in the encoder the filtered macroblock is used for motion-compensated prediction of further frames, resulting in a smaller residual after prediction.

The 2D adaptive filter is applied to both vertical and horizontal edges of 4x4 sub-macroblocks in a macroblock, in the following order vertical first and then horizontal. Each filtering operation may affect up to three pixels on either side of the boundary. In 4x4 pixel sub-macroblocks there are 4 pixels on either side of a vertical or horizontal boundary in adjacent blocks p and q (p_0, p_1, p_2, p_3 and q_0, q_1, q_2, q_3). Depending on the coding modes of neighboring blocks and the gradient of image samples across the boundary, several outcomes are possible, ranging from (a) no pixels are filtered to (b) $p_0, p_1, p_2, q_0, q_1, q_2$ are filtered to produce output pixels P_0, P_1, P_2, Q_0, Q_1 and Q_2 .

The choice of filtering outcome depends on the boundary block strength parameter and on the gradient of image samples across the boundary. The boundary strength parameter **Bs** is chosen according to the following rules:

p or q is (intra coded and boundary is a macroblock boundary)	Bs=4 (strongest filtering)	$P_0, P_1, P_2, Q_0, Q_1, Q_2$
p or q is intra coded and boundary is not a macroblock boundary	Bs=3	P_0, P_1, Q_0, Q_1
neither p or q is intra coded; p or q contain coded coefficients	Bs=2	P_0, P_1, Q_0, Q_1

neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames or a different number of reference frames or different motion vector values	Bs=1	P ₀ , P ₁ , Q ₀ , Q ₁
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	Bs=0 (no filtering)	

The filter sample level decision ($a_p = [1,0]$ for the left side of the filter, and $a_q = [1,0]$ for the right side of the filter) depends on the pixel gradient across block boundaries. The purpose of that decision is to “switch off” the filter when there is a significant change (gradient) across the block boundary or to filter very strongly when there is a very small change (gradient) across the block boundary which is likely to be due to image blocking effect. For example, if the pixel gradient across an edge is below a certain slice threshold ($a_p/a_q = 1$) then a five tap filter (a strong filter) is applied to filter P₀, if not ($a_p/a_q = 0$) then a three tap filter (a weak filter) is applied. In a single compute unit processors the selection between which of the filters to apply is done using If/else, jump instructions. The sequencer must jump over the second filter instruction stream if the first one is selected or jump over the first one if the second one is selected. These jump (If/else) instructions are acceptable in a single compute unit processors but not in a multi-compute unit processors such as a single instruction multiple data (SIMD) processors.

Since an SIMD processor can solve similar problems in parallel on different sets of local data it can be characterized as $\underline{n}$ times faster than a single compute unit processor where $\underline{n}$ is the number of compute units in the SIMD. However, this benefit only is available for sequential types of problems such as FIR, FFT, and DTC, IDCT, etc. The

need for SIMD type processing for non-sequential instruction streams is increasing as image size increases.

However, in such multiple compute unit processors where a single sequencer broadcasts a single instruction stream which drives each of the compute units on different local data sets, e.g. the pixel gradient at block boundaries, the conduct of each compute unit may be different, jump/not jump; and to where – depending upon the effect of the common instruction on the individualized local data, and the sequencer cannot take a decision if to jump/not jump that satisfies all the compute units. Therefore, the high speed and efficiency of SIMD processors has not been applied to the family of non-sequential instructions e.g. conditional (if/else, jump) type of problems.

In the current generation of vector SIMD processors this problem can be solved by deriving from a sequence of instructions a generic instruction having an index section and compute section and broadcasting that generic instruction to the multiple compute units, where the index section is applied to localized data stored in each compute unit to select one of a plurality of stored local parameter sets and applying in each compute unit the selected parameters to the local data according to the compute section to produce each compute unit's localized solution to the generic instruction.

BRIEF SUMMARY OF THE INVENTION

It is therefore an object of this invention to provide a multiple compute unit processor and method for accelerating processing of finite impulse response filtering.

It is a further object of this invention to provide such a multiple compute unit

processor and method which increases computing speed by nearly $\underline{n}$ times where $\underline{n}$ is the number of compute units.

It is a further object of this invention to provide such a multiple compute unit processor and method which avoids jumps which interrupt the operation of deep pipeline processors.

It is a further object of this invention to provide such a multiple compute unit processor and method which can reduce the data memory needed to perform the filtering.

It is a further object of this invention to provide such a multiple compute unit processor and method which is flexible enough to solve all conditional and non-conditional finite impulse response filtering.

This invention results from the realization that an improved finite impulse response filtering can be achieved by broadcasting to at least one compute unit an instruction having a plurality of data samples, a conditional field associated with each compute unit, and a set of operator values for operating on each data sample; providing a function of each data sample in accordance with an associated set of operator values identified by the conditional field and combining said functions to obtain an intermediate finite impulse response of said data samples. The conditional field may identify a set of shift values and the compute unit may include a conditional shift unit responsive to the conditional field for providing a shifted value of the data samples in accordance with a one of the set of shift values identified by the conditional field. Or the conditional field may identify a set of coefficient values and the compute unit may include a conditional multiplier unit responsive to the conditional field for providing a multiplied value of the

data sample in accordance with a one of the set of coefficient values identified by the conditional field.

The subject invention, however, in other embodiments, need not achieve all these objectives and the claims hereof should not be limited to structures or methods capable of achieving these objectives.

This invention features a programmable processor for performing finite impulse response filtering including a sequencer configured to generate an instruction having at least two data samples, a conditional field and a set of operator values for operating on each data sample; and a compute unit responsive to the conditional field for providing a function of each the data sample in accordance with an associated set of operator values identified by the conditional field and for combining the functions to obtain an intermediate finite impulse response of the data samples.

In a preferred embodiment the compute unit may include an accumulator for combining the functions to obtain an intermediate finite impulse response of the data samples. The conditional field may identify a set of shift values and the compute unit may include a conditional shift unit responsive to the conditional field for providing a shifted value of the data samples in accordance with a one of the set of shift values identified by the conditional field. The conditional shift unit may include a shift circuit associated with each shift value and a selector circuit responsive to the conditional field for selecting the output of one of the shift circuits. The conditional shift unit may include a shift circuit and a selector circuit responsive to the conditional field to apply one of the shift values to the shift circuit. The compute unit may include an arithmetic logic unit

(ALU) for accumulating the shifted values of the data samples. The shift values for each data sample may be the same and the conditional field may be bypassed. The processor may include a plurality of compute units, each including a sequencer configured to generate an instruction having at least one data sample, a conditional field and a set of shift values for each data sample; and a compute unit including a conditional shift unit responsive to the conditional field for providing a shifted value of the data sample in accordance with a one of the set of shift values identified by the conditional field. The instruction may include a plurality of data samples, a conditional field associated with each compute unit; and a set of shift values for each data sample. The shift values may include a nulling value for selectively setting the shifted data sample value to zero. One of the shift values may include a zero value for leaving the data sample unchanged. The processor may be a single instruction multiple data (SIMD) processor. The processor may include a loop filter for performing finite impulse response filtering. The data samples may include image pixels arranged in sub-macroblock tiles. Each compute unit may receive the image pixels in at least one column of a sub-macroblock tile. Each compute unit may receive the image pixels in a pair of columns of a sub-macroblock tile. Each compute unit may be associated with a different pair of the columns and responds to a respective pair of the conditional fields to independently select one of the shifted data samples values. The conditional fields may be the same. The data samples may be contained in a data register of the compute unit. The conditional field may identify a set of coefficient values and the compute unit includes a conditional multiplier unit responsive to the conditional field for providing a multiplied value of the data sample in

accordance with a one of the set of coefficient values identified by the conditional field. The conditional multiplier unit may include a multiplier circuit associated with each coefficient value and a selector circuit responsive to the conditional field for selecting the output of one of the multiplier circuits. The conditional multiplier unit may include a multiplier circuit and a selector circuit responsive to the conditional field to apply one of the coefficient values to the multiplier circuit. The compute unit may include an arithmetic logic unit (ALU) for accumulating the multiplied values of the data samples. The multiplied values for each data sample may be the same and the conditional field may be bypassed. The processor may include a plurality of compute units, each including a sequencer configured to generate an instruction having at least one data sample, a conditional field and a set of coefficient values for each data sample; and a compute unit including a conditional multiplier unit responsive to the conditional field for providing a multiplied value of the data sample in accordance with a one of the set of coefficient values identified by the conditional field; the instruction may include a plurality of data samples, a conditional field associated with each compute unit; and a set of coefficient values for each data sample. One of the shift values may include a nulling value for selectively setting the multiplied data sample value to zero. One of the multiplied values may include a multiplication identity value for leaving the data sample unchanged. The identity value may be a zero.

This invention also features a method of performing finite impulse response filtering including broadcasting to at least one compute unit an instruction having a plurality of data samples, a conditional field associated with each compute unit, and a set

of operator values for operating on each data sample; providing a function of each the data sample in accordance with an associated set of operator values identified by the conditional field; and combining the functions to obtain an intermediate finite impulse response of the data samples.

In a preferred embodiment the conditional field may identify a set of shift values and a shifted value of the data sample may be provided in accordance with a one of the set of shifted values identified by the conditional field. The data sample may be shifted in accordance with each of the shift values and one of the shifted sample values may be selected as identified by the conditional field. The data sample may be shifted in accordance with a one of the shift values identified by the conditional field. The shifted values of the data samples may be accumulated. The shift values for each data sample may be the same and the conditional field may be bypassed. One of the shift values may include a nulling value for selectively setting the shifted data sample value to zero. The shift values may include a zero value for leaving the data sample unshifted. The data samples may include image pixels arranged in sub-macroblock tiles. Each compute unit may receive the image pixels in at least one column of a sub-macroblock tile. Each compute unit may receive the image pixels in a pair of columns of a sub-macroblock tile. Each compute unit may be associated with a different pair of the columns and may respond to a respective pair of the conditional fields to independently select one of the functions of the data sample values. The conditional fields may be the same. The conditional field may identify a set of coefficient values and a multiplied value of the data sample may be provided in accordance with a one of the set of coefficient values identified by the conditional field. The data

sample may be multiplied in accordance with each of the coefficient values and one of the multiplied sample values may be selected as identified by the conditional field. The data sample may be multiplied in accordance with a one of the coefficient values identified by the conditional field. The multiplied values of the data samples may be accumulated. The coefficient values for each data sample may be the same and the conditional field may be bypassed. One of the coefficient values may include a nulling value for selectively setting the multiplied data sample value to zero. One of the coefficient values may include a multiplication identity value for leaving the data sample unchanged. The identity value may be zero.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects, features and advantages will occur to those skilled in the art from the following description of a preferred embodiment and the accompanying drawings, in which:

Fig. 1 is a block diagram of a video decoder with a loop filter employing this invention;

Fig. 2 is a schematic illustration of an image with macroblocks and sub-macroblocks superimposed;

Fig. 3A–G are examples of sub-macroblocks tiling structure in a macroblock;

Fig. 4 is a schematic diagram of a 16x16 pixel macroblock with sixteen 4x4 pixel sub-macroblocks;

Fig. 5 is a schematic block diagram of a portion of a compute unit including a

conditional shift unit and accumulator for processing a single data sample with a sequencer and instruction according to this invention;

Fig. 5A is a block diagram of an alternative arrangement for provision of a data sample;

Fig. 6 is a view similar to Fig. 5 of an alternative construction for a conditional shift unit;

Fig. 7 is a view similar to Fig. 5 with two conditional shift units for processing two data samples simultaneously;

Fig. 7A is a view similar to Fig. 7 with two conditional multiplier units for processing two data samples simultaneously;

Fig. 8 is a schematic block diagram of a processor with multiple compute units and a sequencer;

Fig. 9 is an illustration of a transposition of the pixel data samples to accommodate the natural data flow of the processor;

Fig. 10 is a diagram of the method of this invention employing shifting; and

Fig. 11 is a diagram of the method of this invention employing multiplication.

DISCLOSURE OF THE PREFERRED EMBODIMENT

Aside from the preferred embodiment or embodiments disclosed below, this invention is capable of other embodiments and of being practiced or being carried out in various ways. Thus, it is to be understood that the invention is not limited in its application to the details of construction and the arrangements of components set forth in

the following description or illustrated in the drawings. If only one embodiment is described herein, the claims hereof are not to be limited to that embodiment. Moreover, the claims hereof are not to be read restrictively unless there is clear and convincing evidence manifesting a certain exclusion, restriction, or disclaimer.

The preferred embodiment disclosed herein is described in the environment of a video system wherein an image is compressed and encoded in 16x16 pixel macroblocks and then streamed to a decoder. The invention resides in a loop filter or de-blocking filter which is used in both the encoder and the decoder of such systems.

There is shown in Fig. 1 a video decoder 10 in a receiver of such a system which uses a loop filter or de-blocking filter 12. In operation the compressed bit stream representing the image made of 16x16 pixel macroblocks is delivered to the input 14 of entropy decoding circuit 16. The decoded coefficients are submitted to scaling and inverse transform circuit 18 whose outputs are the residual macroblock data for the current macroblock. This is supplied on line 20 to summing circuit 22. The output of summing circuit 22 comprising the reconstructed macroblock is submitted to loop filter or de-blocking filter 12. The output of filter 12 is the reconstructed image 24 in the form of 16x16 pixel tiled macroblocks 25 which have been reassembled and have had their boundaries filtered to restore the picture to the desired quality. The output of loop filter 12 is also used to reconstruct the reference frames. The intra prediction circuit 26 uses unfiltered previous decoded macroblocks to predict current macroblock data. The motion compensation circuit 28 uses out of order predicted (P) and bidirectional predicted (B) reference frames to reconstruct current macroblock data. Depending on the macroblock

type (intra, inter) switch 30 position is set and the intra predicted macroblock 26 or the motion compensated macroblock 28 is added in summing circuit 22 to the residual macroblock data 20 to generate the current reconstructed macroblock. In the remainder of this particular embodiment the discussion will be focused on operation with the switch 30 in the intra prediction position.

An example of such an image, Fig. 2, shows that while many of the macroblocks 25, in the areas where there is not a lot of detail, are kept in single unitary 16x16 pixel macroblocks: in areas where the color, tonality and other features change, the macroblock may be broken into one or more sub-macroblocks, such as shown in macroblocks 25-1, 25-2 and 25-3, for example. The decision of whether to further sub-divide the macroblocks and how to further sub-divide them into sub-macroblocks is dictated by the encoder and the targeted bit rate. For example, in non-real time encoding applications such as DVD's the encoder can run all possible macroblock partitions and select the one that needs the minimum number of bits to encode that macroblock. On the other hand in real time encoding the encoder can't run all possible macroblock partitions but the encoder still seeks for the first (sub-optimal) macroblock partitions that satisfies the desired bit rate.

A sampling of the various combinations is shown in Fig. 3 A-G, where it can be seen: Fig. 3A shows a unitary macroblock of 16x16 pixels; Fig. 3B shows a macroblock composed of two 8x16 sub macroblocks; Fig. 3C shows an macroblock composed of two 16x8 sub macroblocks; Fig. 3D shows an macroblock composed of four 8x8 sub-macroblocks. The macroblock in Fig. 3E includes one 8x8 sub-macroblock, two 4x8

sub-macroblocks, four 4x4 sub-macroblocks and two 8x4 sub-macroblocks. In Fig. 3F, the macroblock includes one 8x16 sub-macroblock, two 4x8 sub-macroblocks and two 8x4 sub-macroblocks. And in Fig. 3G, the macroblock includes one 8x8 sub-macroblock, two 4x8 sub-macroblocks and one 16x8 sub-macroblock.

The actual coding and decoding of images using the macroblock and sub-macroblock procedure involves both luma and chroma pixel information. This embodiment of the invention is explained only with respect to the luma information but applies equally as well to the chroma information.

There is shown in Fig. 4, a typical macroblock 25 composed of 16 sub-macroblock 40 of 4x4 pixels or 16 pixel size. Sub-macroblock 40p includes four rows 42, 44, 46, and 48 of four pixels each. The first row 42 has the pixels named, p_{01} , p_{02} , p_{03} , p_{04} ; the corresponding row 50 in sub-macroblock 40q has its four pixels identified as q_{01} , q_{02} , q_{03} , q_{04} . The remaining rows 44, 46, 48, and 51, 53, 55 are named in a similar fashion. By way of example assume that the border 52 between sub-macroblocks 40p and 40q is the border or boundary to be filtered in this example. In the boundary level there are actually four filter “strengths” that can be used to filter the pixels on either side of that boundary in each row which are identified as indicated in the background as $B_s = 1, 2, 3, 4$, $B_s = 0$ means no filtering. Filter strength $B_s = 4$ is the highest and it involves three out of the four pixels in each row $p_0 - p_2$ and $q_0 - q_2$. The lowest strength $B_s = 1, 2$ and 3 effect only p_0 , p_1 , q_0 and q_1 . The particular filter strength is governed by a content adaptive non-linear filtering scheme which is well defined by the coded specification. The filter is “stronger” at places where is likely to be significant blocking distortion, such

as the boundary of an intra coded macroblock or a boundary between blocks that contain coded coefficients. For example, in the H.264 codec referred to in the Background, *supra*, in the sample level the filter strength ap/aq will be adapted by a function of the pixel gradient across block boundaries where ap is the pixel gradient across the p pixels and aq is the pixel gradient across q pixels. In other codecs, such as, Windows Media Video (.wmv) and MPEG-4 this is a function of boundary strength across block boundaries, where the filter strength will be adapted by comparing a running sum of “ Ψ ” function across the filtered edge against a threshold. Ψ function is defined as:

$$\Psi = \begin{cases} 1 & \text{if } |\text{Pixel}_i - \text{Pixel}_{i+1}| < \text{threshold (where } i = 0 \text{ to filter length)} \\ 0 & \text{else} \end{cases}$$

In this example the explanation will be with reference to the H.264 codec specification. In H.264 the pixel gradient function across block boundaries are defined as:

$$\begin{cases} ap = |p_2 - p_0| < \text{Beta} & (1) \\ aq = |q_2 - q_0| < \text{Beta} & (2) \end{cases}$$

Where Beta is a slice threshold set by H.264 encoder to adapt the amount of the reconstructed image filtering .

Assuming the filter at the boundary 52 is to be $B_s = 4$, the processor that executes the adaptive finite impulse response (FIR) loop filter or de-blocking filter has two choices: if ap equals 1 then the processor must carry out the three filters to update P_0 , P_1 and P_2 as shown in equations (3), (4) and (5).

$$P_0 = (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1)/8 \quad (3)$$

$$P_1 = (p_2 + p_1 + p_0 + q_0)/4 \quad (4)$$

$$P_2 = (2p_3 + 3p_2 + p_1 + p_0 + q_0)/8 \quad (5)$$

If a_p is not 1, that is, if it equals 0 then only one filter need be carried out to update P_0 as shown in equation (6) and leave $P_1 = p_1$ and $P_2 = p_2$ (unchanged).

$$P_0 = (2p_1 + p_0 + q_1)/4 \quad (6)$$

An identical set of equations depending on $a_q = [0,1]$ would be used to process $Q_0 - Q_3$ only in those equations the P's and Q's would be reversed and the equations would be being solved for Q_0, Q_1, Q_2 and equation (6) would be solved for Q_0 .

Thus, if both filter powers were to be $B_s = 4$ and if both pixel gradients a_p and a_q were equal to one the filtering for this row 42, Fig. 4, could be carried out by parallel processing in two identical compute units. However, this cannot be assured for while the $p_0 - p_3$ filtering may be by $a_p = 1$, the $q_0 - q_3$ filtering may be $a_q = 0$. If they were both 1 then both compute units could in parallel move through operations (3), (4), and (5). Or if a_p and a_q both equal 0 the two compute units could in parallel move through operation (6). But if they are different one compute unit must perform operations as shown in (3), (4), and (5), while the other is simply doing the one operation of (6). That is, they are no longer parallel processing in the true sense. In one case the operations would involve (3), (4), and (5) and then jump over operation (6), whereas in the other case, operation (3), (4), and (5) would be jumped over to go directly to operation (6). These, if, else, or jump, or non-sequential type of instructions are not workable in multiple compute unit

processors with deep pipeline stages. The jumps break the flow and require many cycles to clear the pipeline of old instructions and fill it up with the new instructions.

As disclosed in U. S. Patent Application No. 11/045,888, filed January 28, 2005, entitled METHOD AND APPARATUS FOR ACCELERATING PROCESSING OF A NON-SEQUENTIAL INSTRUCTION STREAM ON A PROCESSOR WITH MULTIPLE COMPUTE UNITS by Stein et al. herein incorporated in its entirety by this reference, even though different operations are to be performed, parallel processing can take place in two or more compute units. This is done by converting the non sequential instructions, such as shown in equations (3), (4), (5) and (6) to a more generalized generic instruction that carries both operations within it where the values are represented as shift values. There is broadcast to the SIMD array an instruction stream having a plurality of data samples, a conditional field associated with each compute unit and a set of shift values, to add or subtract in each of the local compute unit's accumulator a shifted value of the input data samples in accordance with one of the set of shift values identified by the local conditional field. For example, the non-sequential instructions represented by the equations (3) and (6) for P_0 can be generalized as follows. For $a_p = 1$ equation (3) can be rewritten as

$$P_0 = \frac{p2}{8} + \frac{2p1}{8} + \frac{2p0}{8} + \frac{2q0}{8} + \frac{q1}{8} \quad (7)$$

and for $a_p = 0$ equation (6) can be rewritten as

$$P_0 = \frac{2p1}{4} + \frac{p0}{4} + \frac{q1}{4} \quad (8)$$

Equation (7) can then be generalized to:

$$\frac{2p_0}{8} + \frac{2p_1}{8} + \frac{p_2}{8} + \frac{2q_0}{8} + \frac{q_1}{8} \quad (9)$$

and equation (8) can be generalized to:

$$\frac{p_0}{4} + \frac{2p_1}{4} + \dots + \frac{q_1}{4} \quad (10)$$

and it can be seen that equation (9) and equation (10) are in the same form except that equation (10) for P_0 and $ap = 0$ has no p_2 or q_0 term. The generic instruction stream can be represented as:

$$p_0 \gg 2 + p_1 \gg 2 + p_2 \gg 3 + q_0 \gg 2 + q_1 \gg 3 \quad (11)$$

And equation (10) can be written as:

$$p_0 \gg 2 + p_1 \gg 1 + p_2 \gg 8 + q_0 \gg 8 + q_1 \gg 2 \quad (12)$$

Where $p_i \gg 8$ is equivalent to mulling the contribution of an 8 bit pixel i to the filter output.

The generic instruction can be represented as:

$$p_0 \gg (ap ? 2:2) + p_1 \gg (ap ? 2:1) + p_2 \gg (ap ? 3:8) + q_0 \gg (ap ? 2:8) + q_1 \gg (ap ? 3:2) \quad (13)$$

where all the terms p_0, p_1, p_2, q_0, q_1 , in both equations (11) and (12) are represented with accompanying shift values associated with each data sample and a conditional field for each conditional shift unit to accumulate a shifted value of the data sample in accordance with one of shift values identified by the conditional field. There may be a number of conditional shift units associated with each compute unit and there may be a number of compute units.

In accordance with this invention the finite impulse response filtering can be

achieved by broadcasting to at least one compute unit an instruction having a plurality of data samples, a conditional field associated with each compute unit, and a set of operator values for operating on each data sample. A function of each data sample is provided in accordance with an associated set of operator values identified by the conditional field. The functions are then combined to obtain an intermediate finite impulse response of the data samples.

In one embodiment the conditional field identifies a set of shift values and the compute unit includes a conditional shift unit responsive to the conditional field for providing a shifted value of the data sample in accordance with a one of the set of shift values identified by the conditional field and in another embodiment the conditional field identifies a set of coefficient values and the compute unit includes a conditional multiplier unit responsive to the conditional field for providing a multiplied value of the data sample in accordance with a one of the set of coefficient values identified by the conditional field. One of the coefficient values may include a multiplication identity value for leaving the data sample unchanged. Or one of the coefficient values may include a multiplication identity value of zero for nulling or setting the multiplied data sample to zero.

There is shown in Fig. 5 a processor 99 implementing the shifting embodiment according to this invention including a conditional shift circuit 100 in a compute unit 102 and sequencer 106. Sequencer 106 provides an instruction 104 to operate conditional shift circuit 100. Conditional shift circuit 100 includes shifters 108, 110, mux 112 and there is an accumulator 114. Instruction 104 carries a data sample 116, shift value 1 (SV1) 118,

shift value 2 (SV2) 120 and a conditional field 122. In operation data sample 116, for example, $p0_1$, is delivered to both shifters 108 and 110 simultaneously. Each shifter 108, 110 shifts the $p0_1$ value by an amount identified by the shift values SV1, SV2, respectively. In this example, SV1 will function to execute the $ap=1$ option while SV2 will function to execute the $ap=0$ option. Both shifted values of $p0_1$ are delivered to mux 112 which selects one of them for delivery to accumulator 114 in accordance with the command CF from conditional field 122.

Variously the shifter values SV1, SV2 may be the same, bypassing the conditional field function, or they may be different. They may contain a nulling value for setting the shifted value of $p0_1$ to zero or they may contain a zero for leaving the data sample unchanged. In this way the data sample $p0_1$ may be processed quickly as to both the ap options with a minimum of hardware and machine cycles.

While thus far the instruction provided by the sequencer includes both conditional fields and shift/multiplier values and data samples this is not a necessary limitation of the invention. For, as shown in Fig. 5A instruction 104-1 may contain all but the data samples and the data samples may be provided from one or more data registers 107-1, 107-2, 107-3...107n.

Instruction 104 may be executed in processor 99a, Fig. 6, by an alternative construction of a conditional shift unit 100a in which the conditional field CF is applied to the mux 112a to command shifter 108a to shift by either SV1 or SV2 to execute $ap=1$ or $ap=0$, for example.

The benefits of the inventions are even more realizable as shown in processor

99b, Fig. 7, processing a pair of pixels in each compute unit using multiple conditional shift units where, for example, two conditional shift units 100b, 100bb respond to instruction 104b which contains two data samples p_{01} 116b and p_{02} 116bb, two shift values associated with each: $SV1p_{01}$, $SV2p_{01}$, $SV1p_{02}$, $SV2p_{02}$, and a conditional field CFp_{01} , and CFp_{02} associated with each conditional shift circuit 100b and 100bb all respectively. In operation data sample 116b, p_{01} , shift values 118b, 120b, $SV1p_{01}$, $SV2p_{01}$, and conditional field 122b, CEp_0 , are applied to conditional shift unit 100b and data sample 116bb, p_{02} , shift values 118bb, 120bb, $SV1p_{02}$, $SV2p_{02}$ and conditional field 122bb, CFp_{02} are applied to conditional shift circuit 100bb. The shifted value whether it be zero, null, positive or negative of p_{01} and p_{02} are then delivered by muxes 112b, 112bb to be added or subtracted from the previous result by accumulator 114b.

The invention is not limited to having a separate condition field for each pixel of the pixel pair: they both can share the same one. It is equally applicable to a number of conditional shift/multiplier units, three, four or many more. There may be one or more conditional shift/multiplier units in each compute unit and there may be a plurality of compute units as shown in Fig. 8 where a processor 99c according to this invention includes a number M of compute units 102' together having a total of N conditional shift/multiplier units 100'. Instruction 104' commensurately has N data samples 116', N sets of coefficient/shift values 118' and 120', $SV1$, $SV2$ and N conditional fields 122' CF. Processor 99, a,b,c may be e.g., a loop filter, or SIMD processor.

An embodiment employing multipliers instead of shifters is shown in Fig. 7A where in instruction 104c the shift values have been replaced by coefficient values 118c,

120c, 118cc, 120cc and compute unit 102c in place of conditional shift circuit employs, conditional multiplier units 108c, 110c, 108cc, 110cc.

While the natural order of processors is suited to the arrangement of pixels *p* and *q* as shown in Fig. 4 for filtering horizontal edge 52, the invention applies as well to filtering a vertical edge 52', Fig. 9 by effecting the transposition from the array 150 to array 152 to convert the configuration 154 to the more naturally ordered configuration 156 like that shown in Fig. 4 as taught in U. S. Patent Application No. 11/045,888, filed January 28, 2005, entitled METHOD AND APPARATUS FOR ACCELERATING PROCESSING OF A NON-SEQUENTIAL INSTRUCTION STREAM ON A PROCESSOR WITH MULTIPLE COMPUTE UNITS by Stein et al. previously cited herein and incorporated by reference.

The invention includes the method including broadcasting an instruction with data sample(s), set(s) of associated shift values and conditional field(s), 200, Fig. 10, and obtaining the shifted value of the data sample(s) per one of the shift value(s) identified by the associated conditional field 202 and then accumulating the result 204. The conditional shift may be effected either by shifting each data sample by all associated shift values and then selecting the shifted value of each data sample as identified by the associated conditional field 206 or identifying with the conditional field the selected shift value and shifting the data sample only by that shift value, 208.

The invention also includes the method including broadcasting an instruction with data sample(s), set(s) of associated coefficient values and conditional field(s), 200a, Fig. 11, and obtaining the multiplied value of the data sample(s) per one of the coefficient

value(s) identified by the associated conditional field 202a and then accumulating the result 204a. The conditional multiplication may be effected either by multiplying each data sample by all associated coefficient values and then selecting the multiplied value of each data sample as identified by the associated conditional field 206a or identifying with the conditional field the selected coefficient value and multiplying the data sample only by that coefficient value, 208a. One of the coefficient values may include a multiplication identity value for leaving the data sample unchanged. Or one of the coefficient values may include a multiplication identity value of zero for nulling or setting the multiplied data sample to zero.

Although specific features of the invention are shown in some drawings and not in others, this is for convenience only as each feature may be combined with any or all of the other features in accordance with the invention. The words “including”, “comprising”, “having”, and “with” as used herein are to be interpreted broadly and comprehensively and are not limited to any physical interconnection. Moreover, any embodiments disclosed in the subject application are not to be taken as the only possible embodiments.

In addition, any amendment presented during the prosecution of the patent application for this patent is not a disclaimer of any claim element presented in the application as filed: those skilled in the art cannot reasonably be expected to draft a claim that would literally encompass all possible equivalents, many equivalents will be unforeseeable at the time of the amendment and are beyond a fair interpretation of what is to be surrendered (if anything), the rationale underlying the amendment may bear no

more than a tangential relation to many equivalents, and/or there are many other reasons the applicant can not be expected to describe certain insubstantial substitutes for any claim element amended.

Other embodiments will occur to those skilled in the art and are within the following claims.

What is claimed is:

CLAIMS

1. A programmable processor for performing finite impulse response filtering comprising:

a sequencer configured to generate an instruction having at least two data samples, a conditional field and a set of operator values for operating on each data sample; and

a compute unit responsive to said conditional field for providing a function of each said data sample in accordance with an associated set of operator values identified by said conditional field and for combining said functions to obtain an intermediate finite impulse response of said data samples.

2. The programmable processor for performing finite impulse response filtering of claim 1 in which said compute unit includes an accumulator for combining said functions to obtain an intermediate finite impulse response of said data samples.

3. The programmable processor for performing finite impulse response filtering of claim 1 in which said conditional field identifies a set of shift values and said compute unit includes a conditional shift unit responsive to said conditional field for providing a shifted value of said data samples in accordance with a one of said set of shift values identified by said conditional field.

4. The programmable processor for performing finite impulse response filtering of claim 3 in which said conditional shift unit includes a shift circuit associated with each shift value and a selector circuit responsive to said conditional field for selecting the output of one of said shift circuits.

5. The programmable processor for performing finite impulse response filtering of claim 3 in which said conditional shift unit includes a shift circuit and a selector circuit responsive to said conditional field to apply one of said shift values to said shift circuit.

6. The programmable processor for performing finite impulse response filtering of claim 3 in which said compute unit includes an arithmetic logic unit (ALU) for accumulating the shifted values of said data samples.

7. The programmable processor for performing finite impulse response filtering of claim 3 in which said shift values for each data sample are the same and said conditional field is bypassed.

8. The programmable processor for performing finite impulse response filtering of claim 3 in which said processor includes a plurality of compute units, each including a sequencer configured to generate an instruction having at least one data

sample, a conditional field and a set of shift values for each data sample; and a compute unit including a conditional shift unit responsive to said conditional field for providing a shifted value of said data sample in accordance with a one of said set of shift values identified by said conditional field; said instruction including a plurality of data samples, a conditional field associated with each compute unit; and a set of shift values for each data sample.

9. The programmable processor for performing finite impulse response filtering of claim 3 in which said one of said shift values includes a nulling value for selectively setting the shifted data sample value to zero.

10. The programmable processor for performing finite impulse response filtering of claim 3 in which said one of said shift values includes a zero value for leaving the data sample unchanged.

11. The programmable processor for performing finite impulse response filtering of claim 1 in which said processor is a single instruction multiple data (SIMD) processor.

12. The programmable processor for performing finite impulse response filtering of claim 1 in which said processor includes a loop filter for performing finite impulse response filtering.

13. The programmable processor for performing finite impulse response filtering of claim 1 in which said data samples include image pixels arranged in sub-macroblock tiles.

14. The programmable processor for performing finite impulse response filtering of claim 13 in which each said compute unit receives the image pixels in at least one column of a sub-macroblock tile.

15. The programmable processor for performing finite impulse response filtering of claim 14 in which each said compute unit receives the image pixels in a pair of columns of a sub-macroblock tile.

16. The programmable processor for performing finite impulse response filtering of claim 14 in which each said compute unit is associated with a different pair of said columns and responds to a respective pair of said conditional fields to independently select one of the shifted data samples values.

17. The programmable processor for performing finite impulse response filtering of claim 16 in which said conditional fields are the same.

18. The programmable processor for performing finite impulse response

filtering of claim 1 in which said data samples are contained in a data register of said compute unit.

19. The programmable processor for performing finite impulse response filtering of claim 1 in which said conditional field identifies a set of coefficient values and said compute unit includes a conditional multiplier unit responsive to said conditional field for providing a multiplied value of said data sample in accordance with a one of said set of coefficient values identified by said conditional field.

20. The programmable processor for performing finite impulse response filtering of claim 19 in which said conditional multiplier unit includes a multiplier circuit associated with each coefficient value and a selector circuit responsive to said conditional field for selecting the output of one of said multiplier circuits.

21. The programmable processor for performing finite impulse response filtering of claim 19 in which said conditional multiplier unit includes a multiplier circuit and a selector circuit responsive to said conditional field to apply one of said coefficient values to said multiplier circuit.

22. The programmable processor for performing finite impulse response filtering of claim 19 in which said compute unit includes an arithmetic logic unit (ALU) for accumulating the multiplied values of said data samples.

23. The programmable processor for performing finite impulse response filtering of claim 19 in which said multiplied values for each data sample are the same and said conditional field is bypassed.

24. The programmable processor for performing finite impulse response filtering of claim 19 in which said processor includes a plurality of compute units, each including a sequencer configured to generate an instruction having at least one data sample, a conditional field and a set of coefficient values for each data sample; and a compute unit including a conditional multiplier unit responsive to said conditional field for providing a multiplied value of said data sample in accordance with a one of said set of coefficient values identified by said conditional field; said instruction including a plurality of data samples, a conditional field associated with each compute unit; and a set of coefficient values for each data sample.

25. The programmable processor for performing finite impulse response filtering of claim 19 in which said one of said shift values includes a nulling value for selectively setting the multiplied data sample value to zero.

26. The programmable processor for performing finite impulse response filtering of claim 19 in which said one of said multiplied values includes a multiplication identity value for leaving the data sample unchanged.

27. A method of performing finite impulse response filtering comprising:

broadcasting to at least one compute unit an instruction having a plurality of data samples, a conditional field associated with each compute unit, and a set of operator values for operating on each data sample;

providing a function of each said data sample in accordance with an associated set of operator values identified by said conditional field; and

combining said functions to obtain an intermediate finite impulse response of said data samples.

28. The method of performing finite impulse response filtering of claim 27 in which said conditional field identifies a set of shift values and a shifted value of said data sample is provided in accordance with a one of said set of shifted values identified by said conditional field.

29. The method of performing finite impulse response filtering of claim 28 in which the data sample is shifted in accordance with each of said shift values and one of said shifted sample values is selected as identified by said conditional field.

30. The method of performing finite impulse response filtering of claim 28 in which the data sample is shifted in accordance with a one of the shift values identified by said conditional field.

31. The method of performing finite impulse response filtering of claim 28 further including accumulating the shifted values of said data samples.

32. The method of performing finite impulse response filtering of claim 28 in which said shift values for each data sample are the same and said conditional field is bypassed.

33. The method of performing finite impulse response filtering of claim 28 in which said one of said shift values includes a nulling value for selectively setting the shifted data sample value to zero.

34. The method of performing finite impulse response filtering of claim 28 in which said one of said shift values includes a zero value for leaving the data sample unshifted.

35. The method of performing finite impulse response filtering of claim 27 in which said data samples include image pixels arranged in sub-macroblock tiles.

36. The method of performing finite impulse response filtering of claim 35 in which each said compute unit receives the image pixels in at least one column of a sub-macroblock tile.

37. The method of performing finite impulse response filtering of claim 36 in which each said compute unit receives the image pixels in a pair of columns of a sub-macroblock tile.

38. The method of performing finite impulse response filtering of claim 37 in which each said compute unit is associated with a different pair of said columns and responds to a respective pair of said conditional fields to independently select one of the functions of the data sample values.

39. The method of performing finite impulse response filtering of claim 38 in which said conditional fields are the same.

40. The method of performing finite impulse response filtering of claim 27 in which said conditional field identifies a set of coefficient values and a multiplied value of said data sample is provided in accordance with a one of said set of coefficient values identified by said conditional field.

41. The method of performing finite impulse response filtering of claim 40 in which the data sample is multiplied in accordance with each of said coefficient values and one of said multiplied sample values is selected as identified by said conditional field.

42. The method of performing finite impulse response filtering of claim 40 in which the data sample is multiplied in accordance with a one of the coefficient values identified by said conditional field.

43. The method of performing finite impulse response filtering of claim 40 further including accumulating the multiplied values of said data samples.

44. The method of performing finite impulse response filtering of claim 40 in which said coefficient values for each data sample are the same and said conditional field is bypassed.

45. The method of performing finite impulse response filtering of claim 40 in which said one of said coefficient values includes a nulling value for selectively setting the multiplied data sample value to zero.

46. The method of performing finite impulse response filtering of claim 40 in which said one of said coefficient values includes a multiplication identity value for leaving the data sample unchanged.

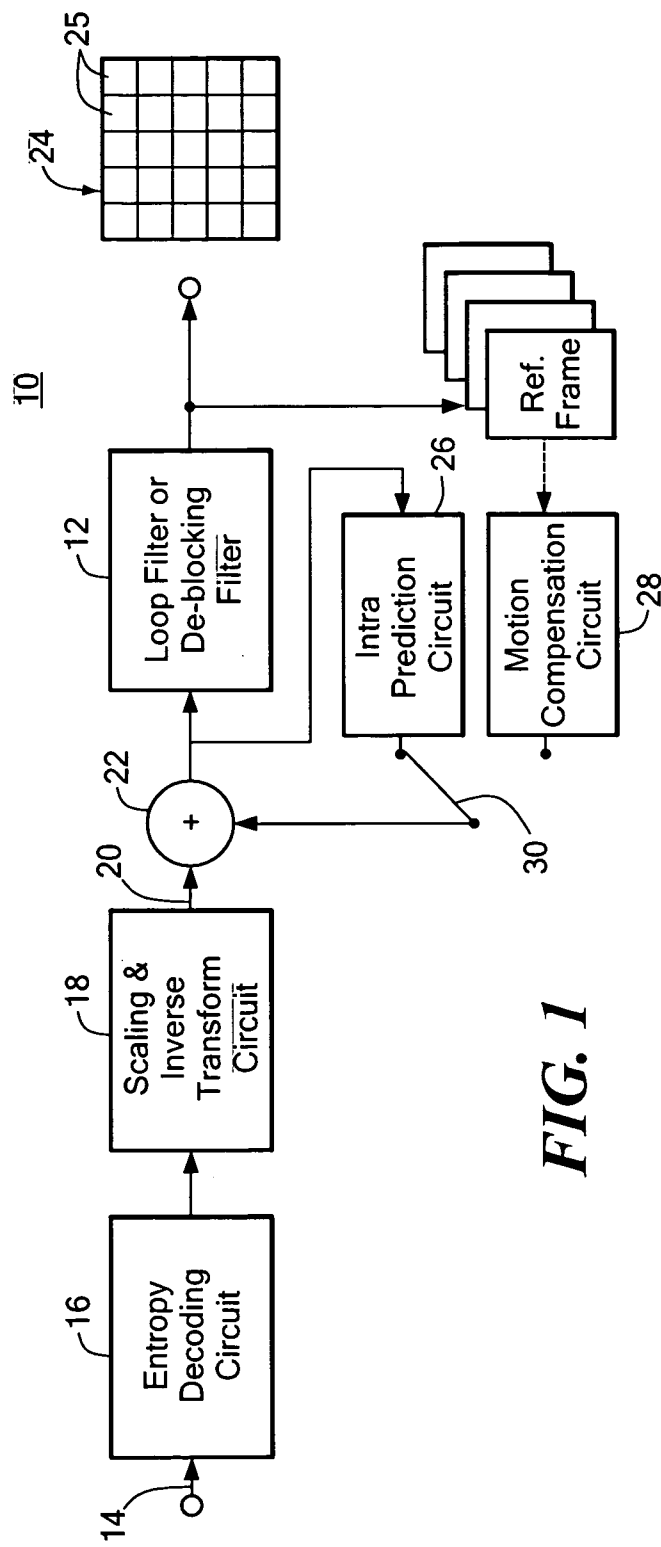
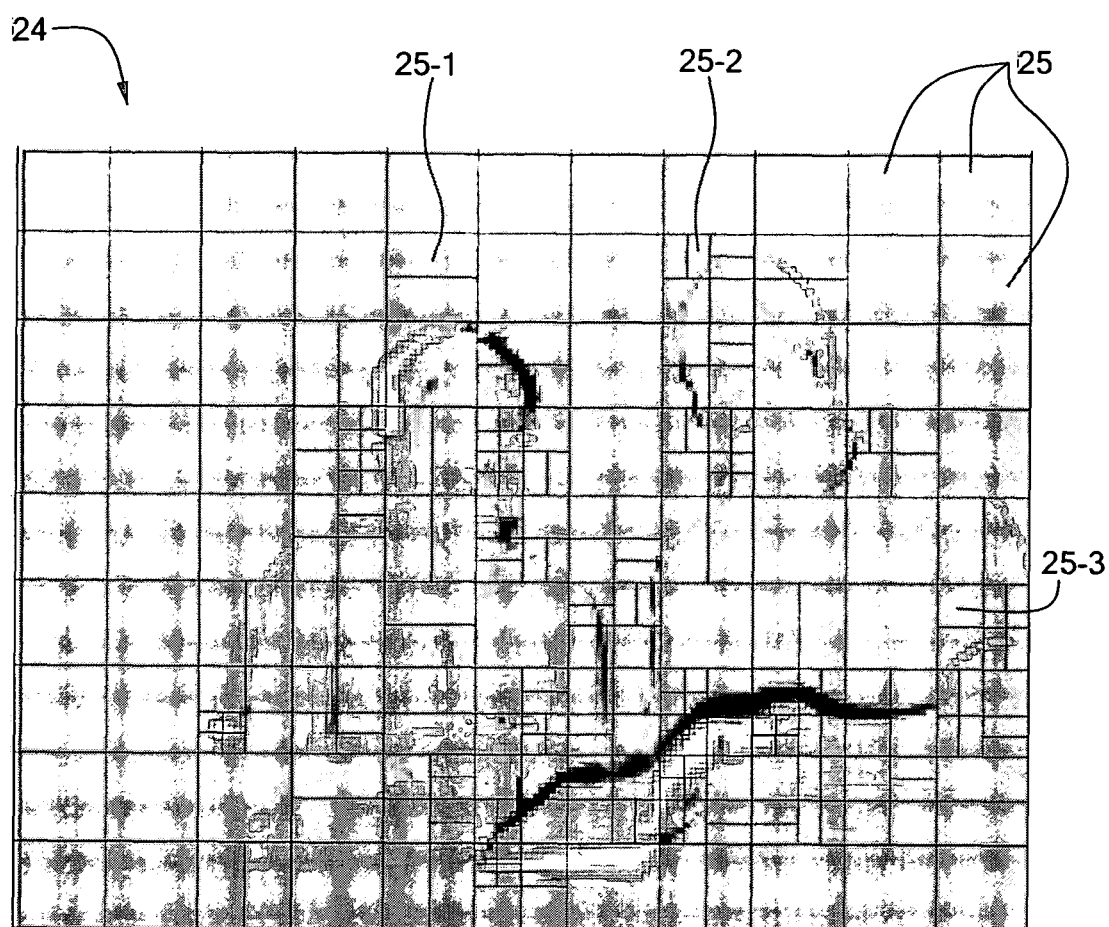


FIG. 1

2/13

**FIG. 2**

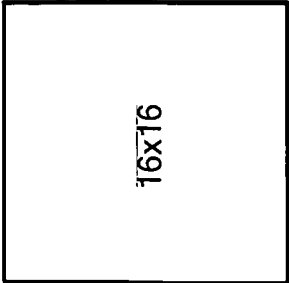


FIG. 3A

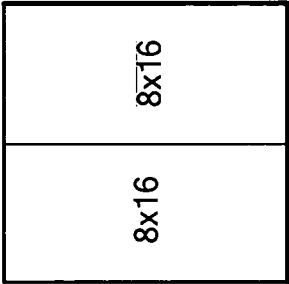


FIG. 3B

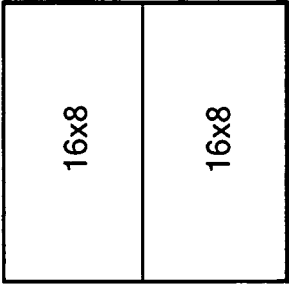


FIG. 3C

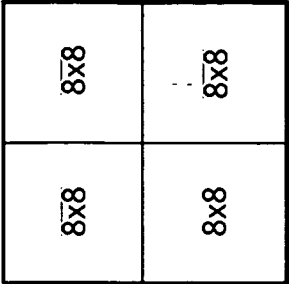


FIG. 3D

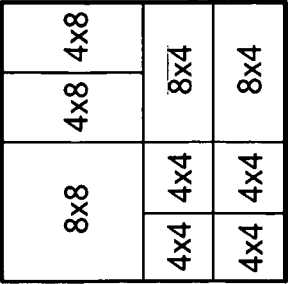


FIG. 3E

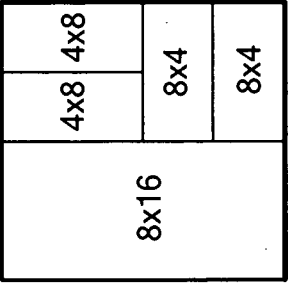


FIG. 3F

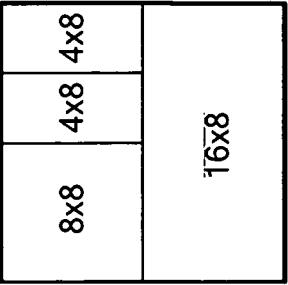


FIG. 3G

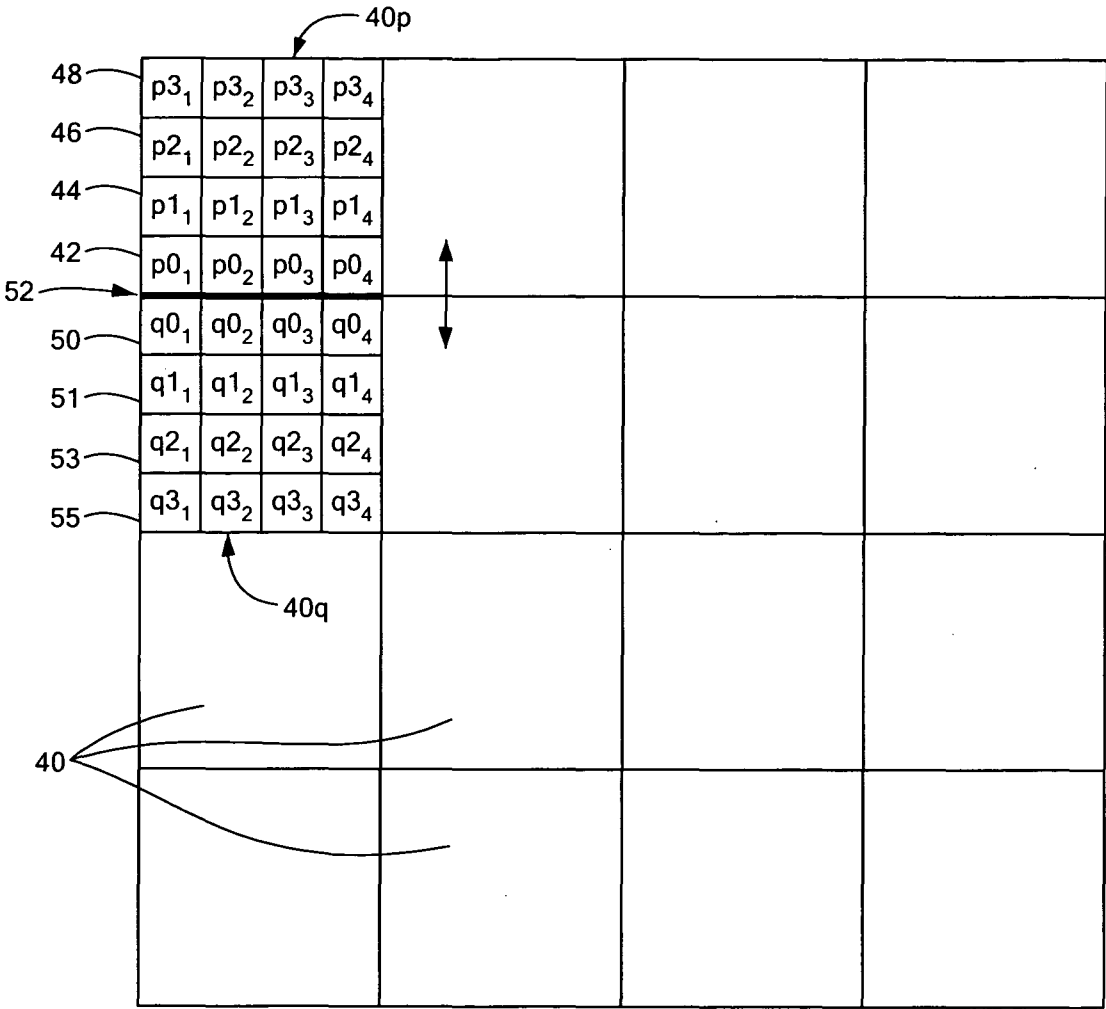


FIG. 4

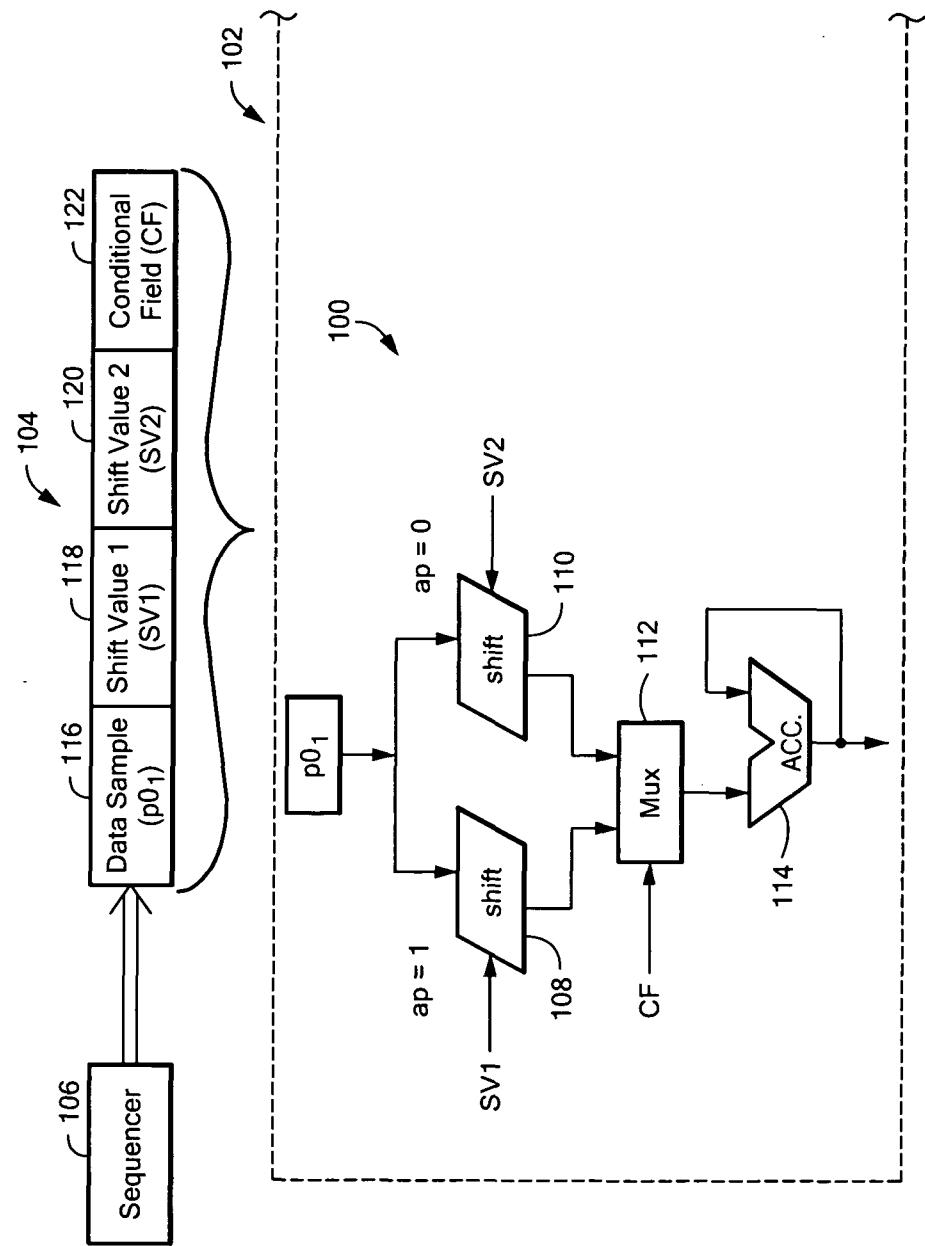


FIG. 5

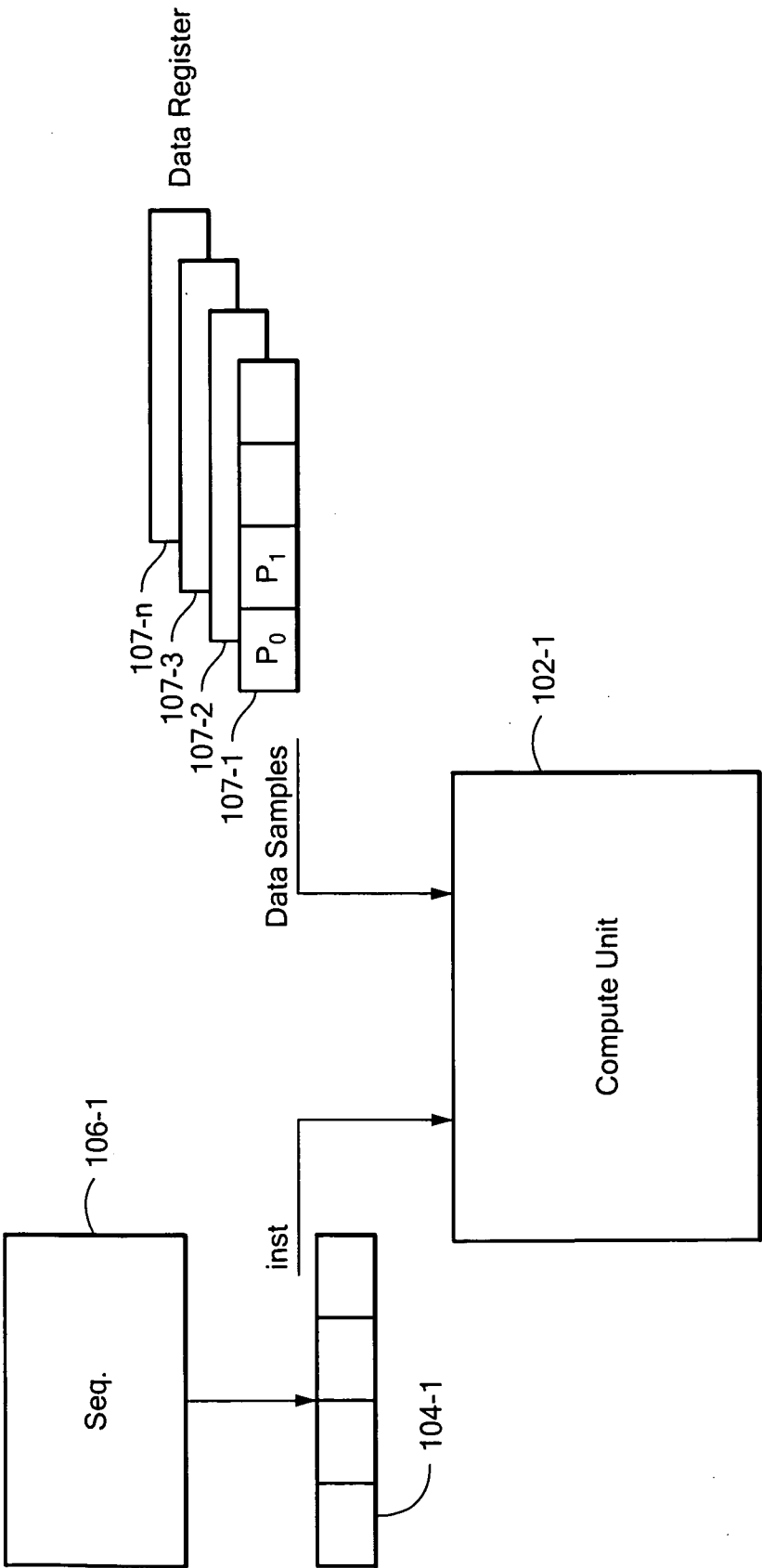


FIG. 5A

7/13

99a

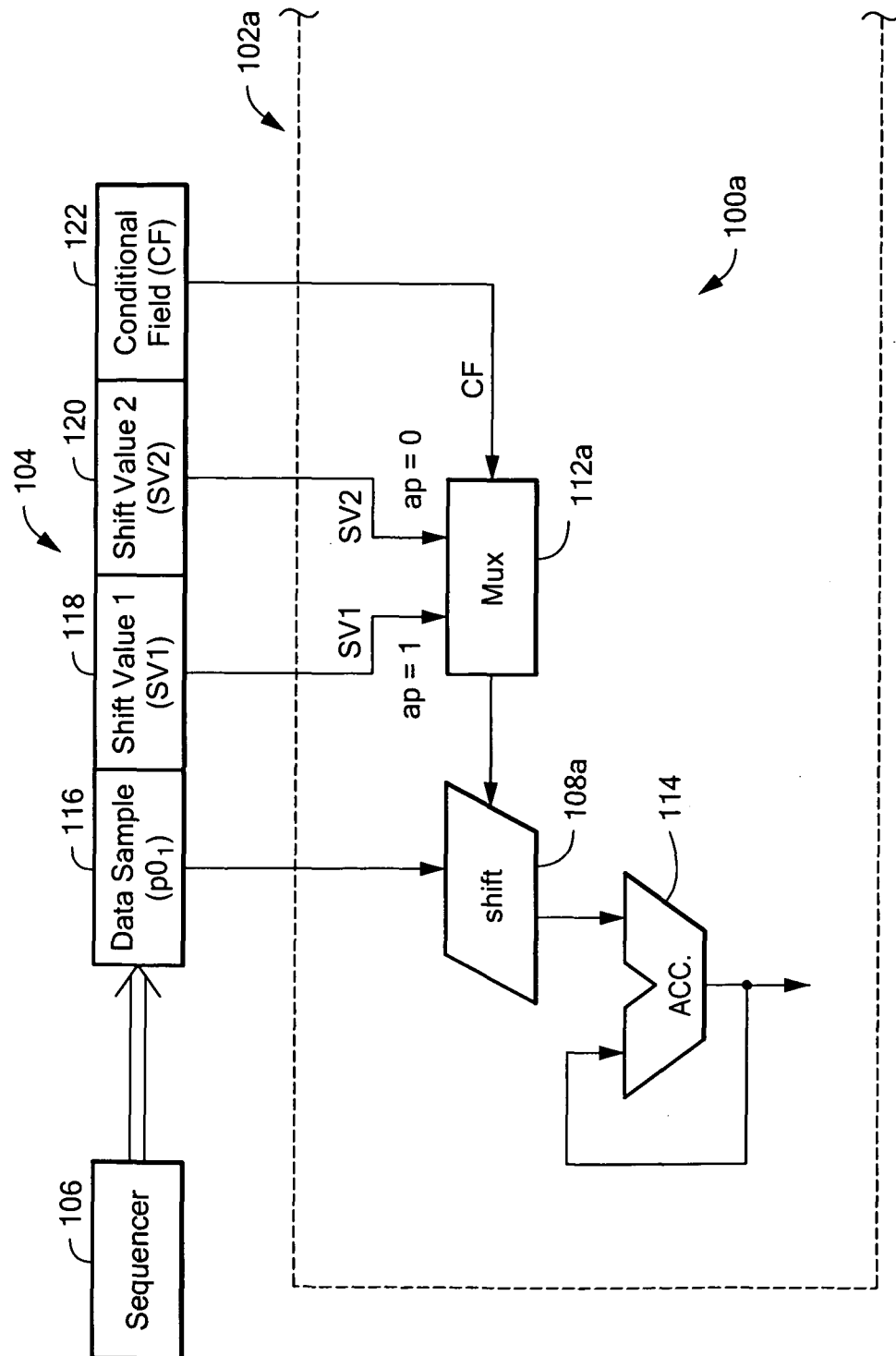


FIG. 6

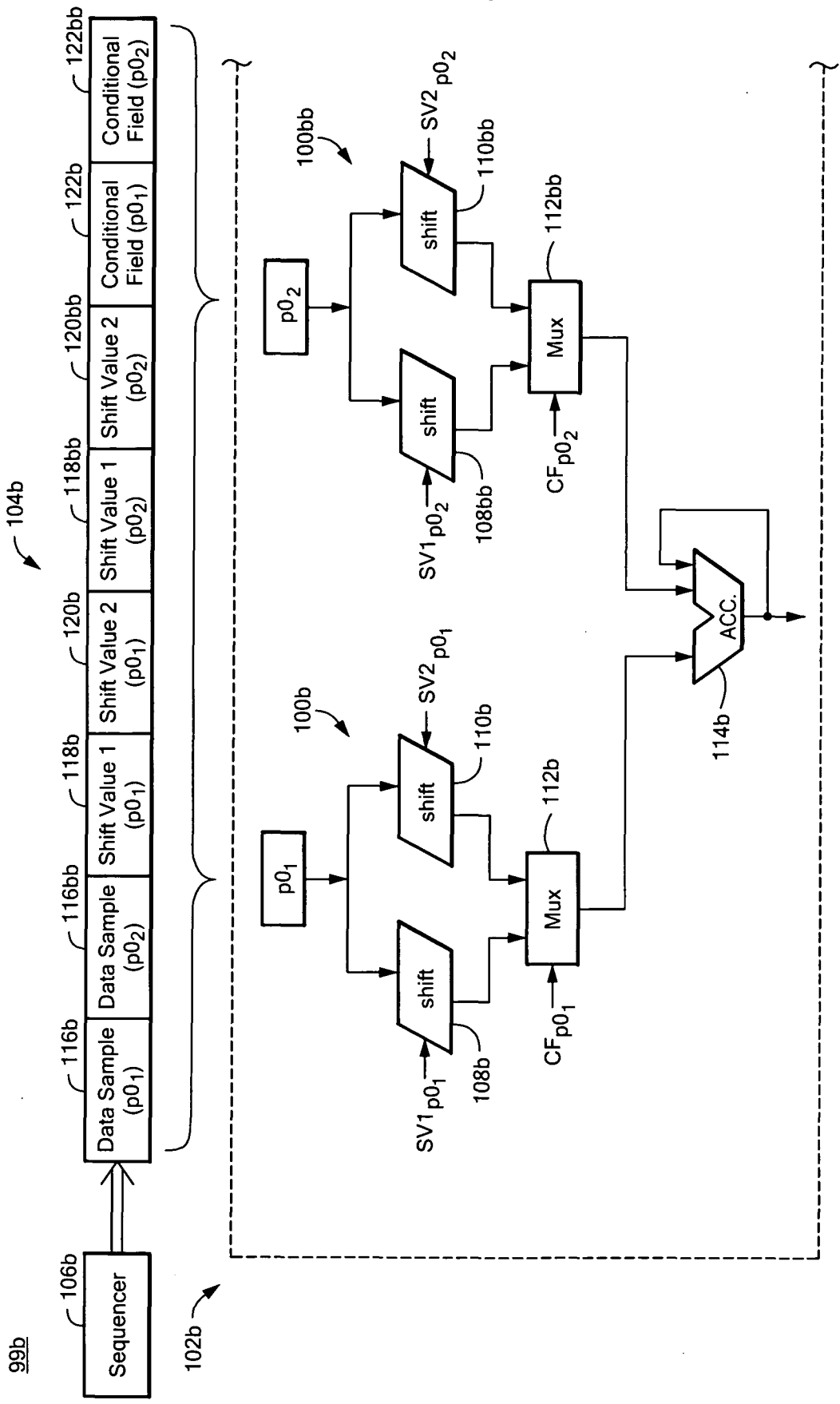


FIG. 7

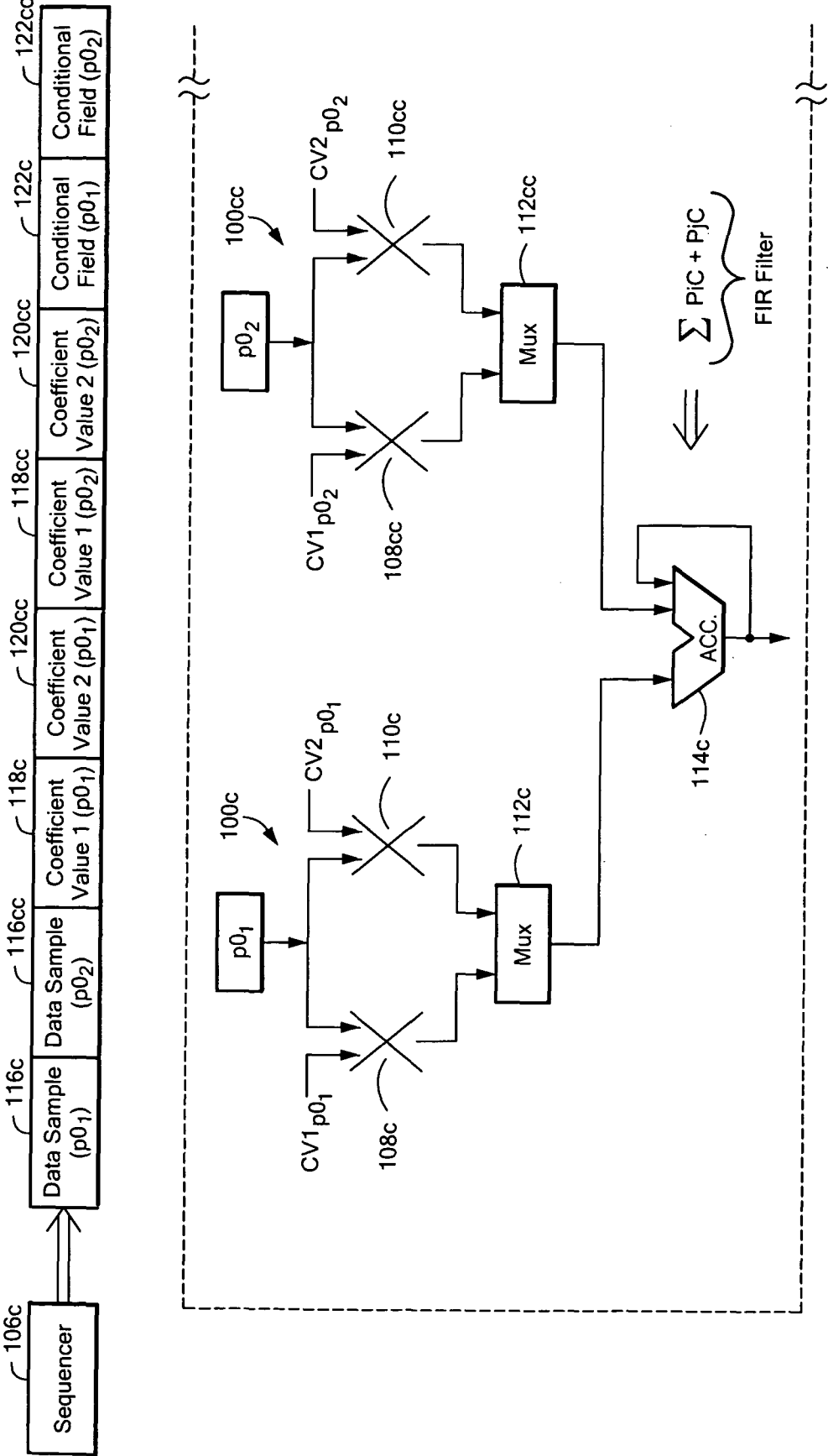


FIG. 7A

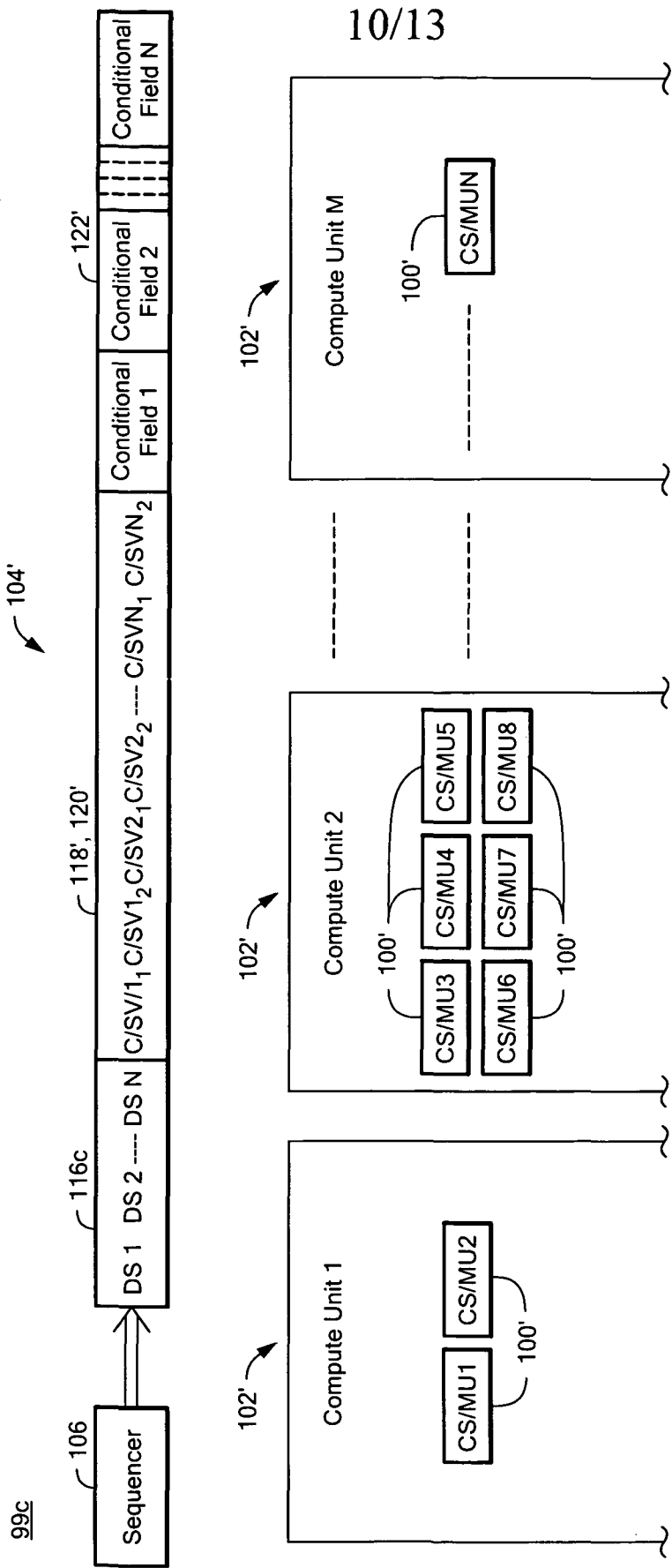


FIG. 8

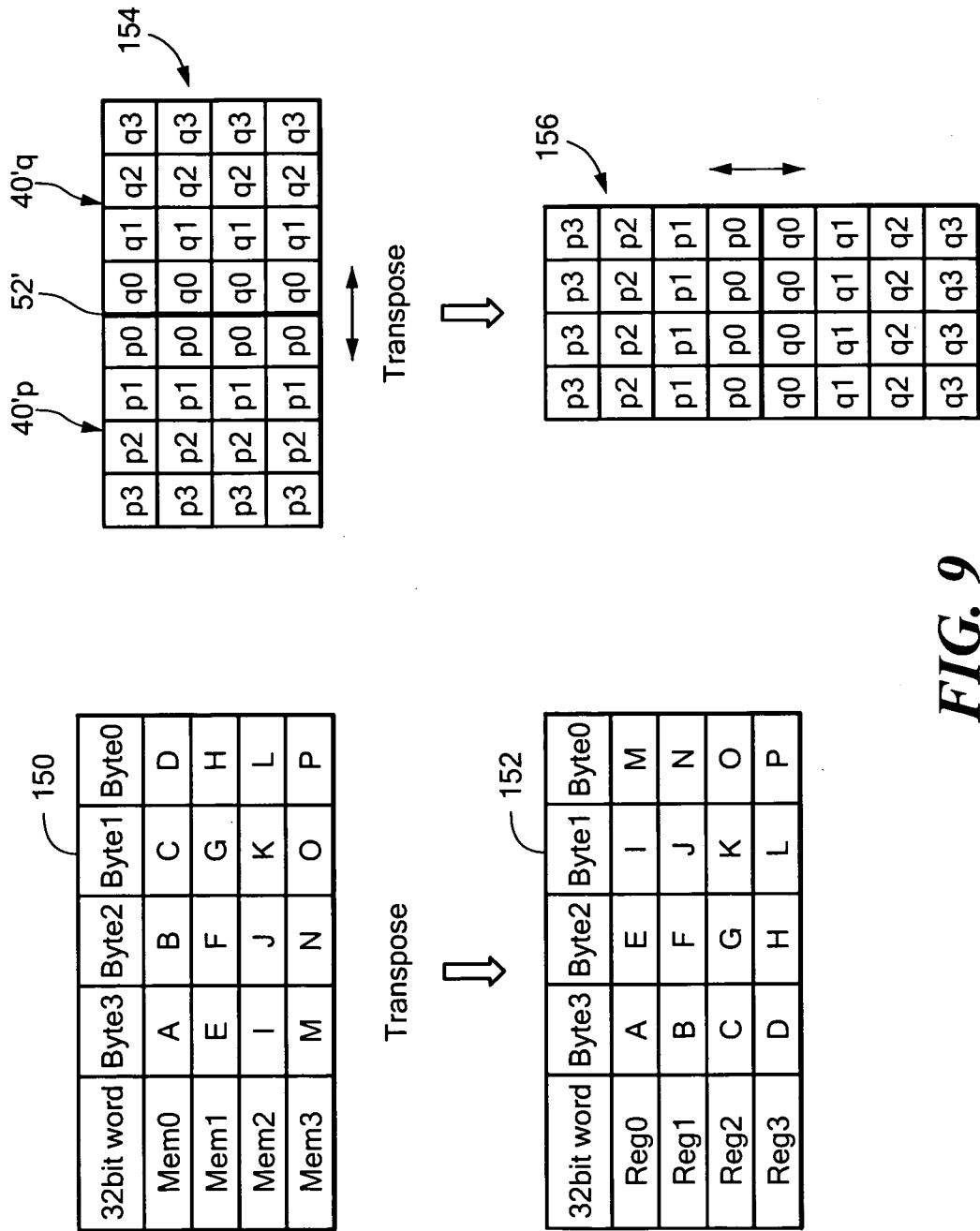
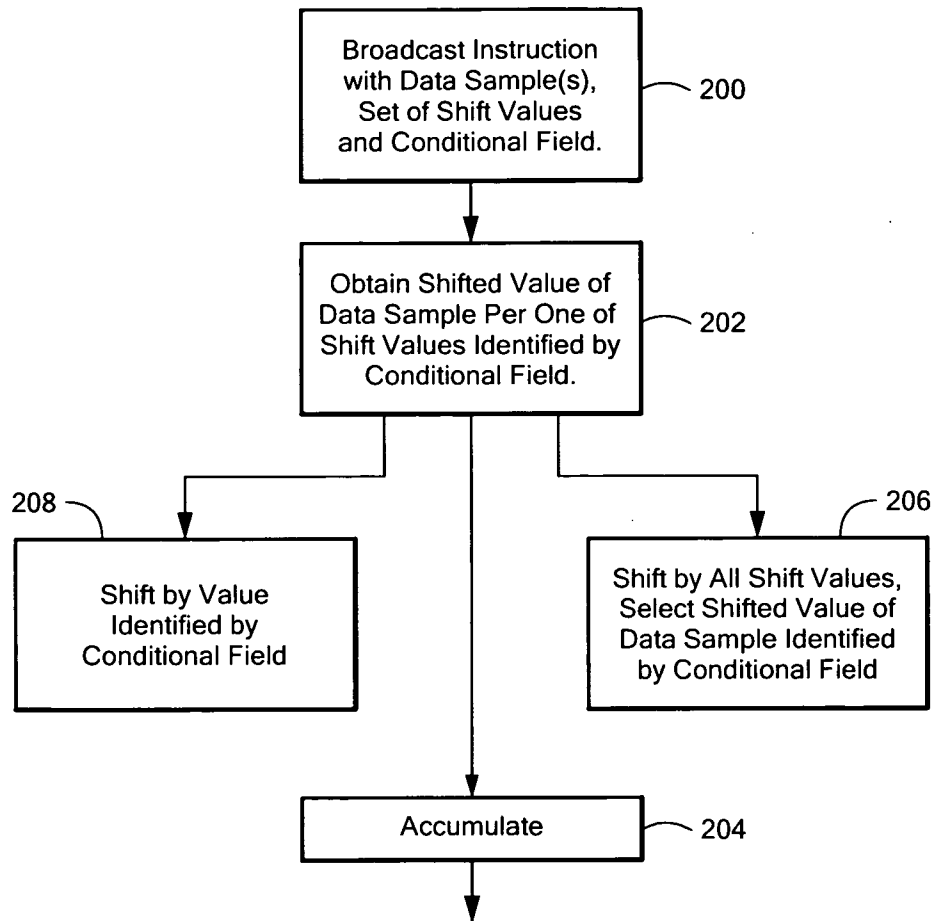
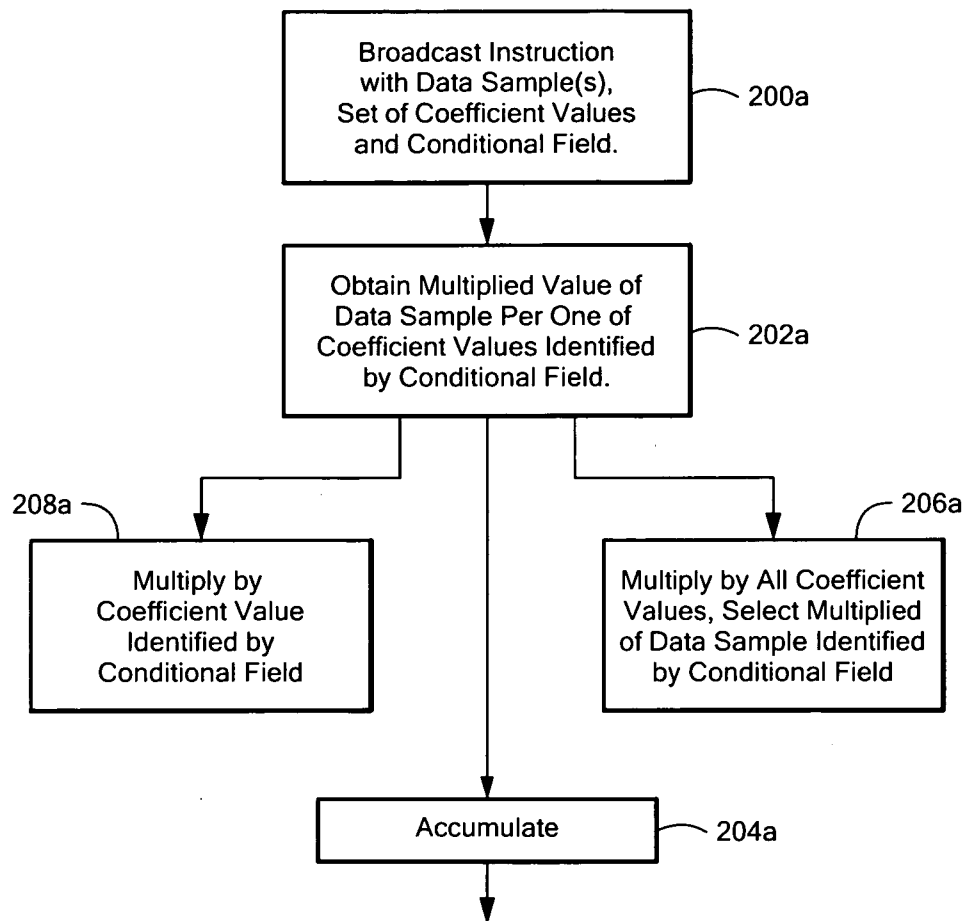


FIG. 9

12/13

**FIG. 10**

13/13

**FIG. 11**