Motion estimation using MSD-first processing

C.-L. Su and C.-W. Jen

Abstract: A new most-significant-digit-first (MSD-first) motion estimation (ME) algorithm based on online arithmetic is proposed in the paper, together with its associated architecture. High-performance ME architectures are crucial for high-quality video applications with high resolution and frame rate, such as HDTV, DVD-video recorders and digital camcorders. The proposed algorithm decomposes the summation of absolute difference (SAD) and the comparison operations of full search block matching (FSBM) into digit level. The digit-level comparisons are interleaved into the separated SAD operations to distinguish the motion vector (MV) as early as possible. It operates in the MSD-first manner and eliminates redundant operations for less significant digits, while still extracting the exact MV of the selected search algorithm. The authors also demonstrate the online arithmetic designs for SAD and comparison, which are two primitive operations in the proposed algorithm. The proposed method saves 47.4% to 64.3% SAD computations of FSBM. In the implementation, a 4×4 array processor of the proposed ME using online arithmetic has a 2.84 ns critical path and 1510 gates with 0.35 μm 1P4M CMOS cell library. It supports 83 mega 4×4 block matching per second and also reduces the ME operations of other fast search algorithms.

1 Introduction

Recently, many compression techniques have been developed and have been adopted for various video standards, such as H.26x and MPEG-1, -2, -4 [1]. Block-based motion estimation and compensation (ME/MC) remove the temporal redundancy in video sequences. For ME, full search block matching (FSBM) [1] gives the optimum motion vector (MV), but the required computation is huge. For example, a video stream with 720×576 frame size and 25 Hz frame rate requires approximately 9000 million operations per second (MOPS) [2]. Various fast search algorithms exist to conquer this problem and can be classified into three categories. One searches partial candidate blocks only within a predefined search window [3–7]. Another reduces the block matching computations within a block i.e. partial pixels are required for computing [9–11]. The third category provides the spatio-temporal predictive technology to attain good results for low-bit rate, low-power applications [12–14]. However, all these fast algorithms guarantee a less optimum MV only, which affects the compression ratio. Several VLSI architectures have been proposed for various search algorithms to speed up their searching time, to reduce the chip size, or to lower the I/O bandwidth requirements [15–17].

Finding the MV relies on the differentiation of the best matching block instead of the exact summation of absolute difference (SAD) value of each candidate. It is reasonable to calculate each SAD at digit level, with processing from the most significant digit (MSD) to least significant digit (LSD) until differentiation. The candidates which are impossible for the best-matched block can be recognised during this processing. The proposed algorithm discards the remaining digit-level SAD operations of these recognised candidates and benefits from the computation reduction. We have shown that the discarded SAD operations are redundant [18] and do not affect the extraction of the exact MV. We also design novel MSD-first comparators and SAD components for the proposed algorithm.

As we know, most-significant-bit-first (MSB-first) processing is inherently suitable for comparison, division, and square-root operations because of the intrinsic algorithmic sequence, which is called online arithmetic. The carry/borrow propagation problem in conventional number systems prevents the MSB-first operations in SAD calculation except the comparisons. Therefore, redundant number systems are adopted here to solve this problem because a number can be represented in more than one way [19]. Among various redundant number representations, Avizienis' binary signed-digit (SD) system [20] is frequently used to implement online arithmetic.

For the past four decades, signed-digit number systems have successfully speeded up arithmetic operations [21–23]. Digit-level pipelining block processing has also been introduced to release the feedback loop bottleneck, such as IIR filters [24], decision-feedback equalisers (DFEs) [25] and coordinate rotation digital computers (CORDICs) [26]. Although there are many advantages in signed-digit systems, the MSD-first realisation is still restricted by overheads of control units and arithmetic components.

In this paper, we use the MSD-first strategy on FSBM. We first simulate the algorithm efficiency of MSD-first FSBM and show that our proposed algorithm can remove 47.4% to 64.3% of redundant SAD operations. The relationship between the algorithm performance and the video characteristics is discussed with cross simulations. Novel SAD and comparator architectures for online arithmetic are proposed for implementation of our MSD-first array processor. We use MSD throughout this paper because binary is a subset of digits.

IEE Proceedings online no. 20030332

doi:10.1049/ip-cds:20030332

Paper first received 5th November 2001 and in revised form 19th June 2002

The authors are with the Department of Electronics Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 30050, Republic of China

2 MSD-first motion estimation

Block-based ME partitions a frame into $N \times N$ blocks (current block, I_c), and compares each block with its reference blocks (I_r) in the previous frame. The offset of the best-matched reference block is denoted by the displacement vector $\boldsymbol{v}$ for each block. Let the searching area be bounded with a displacement p in all directions, where the number of reference blocks is $q = [2p+1]^2$. SAD is the most popular matching criterion, which is given by

$$s[m,n] = \sum_{i=1}^{N}\sum_{j=1}^{N} |I_c(i,j) - I_r(i+m, j+n)|, \quad (1)$$
$$+ p \geqq m,n \geqq -p$$

$$u = \min_{(m,n)}\{s[m,n]\} \quad (2)$$

$$\boldsymbol{v} = (m,n)|_u \quad (3)$$

The SAD of I_c and I_r, i.e. $s[m,n]$, is estimated by summing up the absolute difference between each pixel and its corresponding pixel of the reference block. The MV, $(m,n)|_u$, is the displacement of the best matched reference block which has the minimum SAD value, u, within the searching area. Fig. 1*a* illustrates the simplified block-based ME.

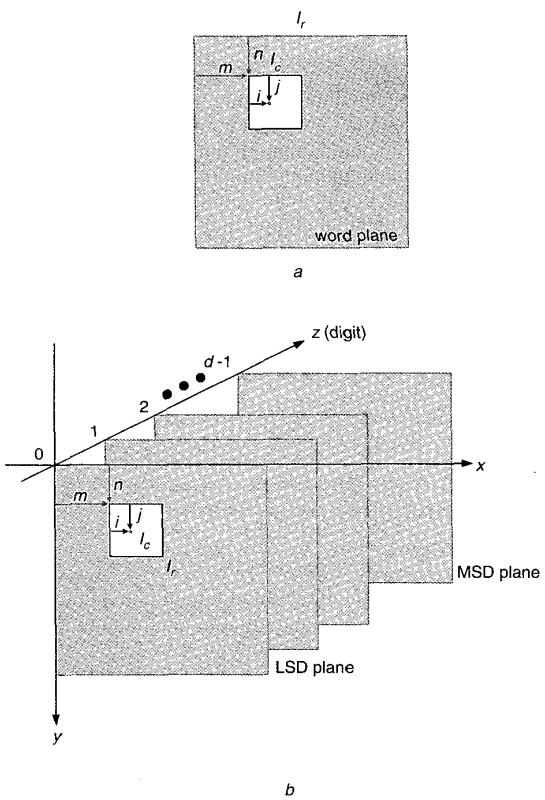

Fig. 1 *Block-matching diagram of motion estimation*
a Word-level representation
b Digit-level representation

2.1 MSD-first ME system

In this paper, we propose the MSD-first ME that removes the redundant SAD and comparison operations of FSBM computations but it is not restricted to FSBM. The proposed MSD-first ME is based on the online arithmetic SAD and comparator, whose operand- and result-flows start from the MSD at digit level. In the following, we apply online arithmetic to the SAD operation first. Then, the online arithmetic comparator finds the minimum SAD value, which corresponds to the best-matched displacement vector. Fig. 2 illustrates a simplified digit-level MSD-first ME system with four-block (2×2) frame and three-bit (2^2,2^1,2^0) pixels. Each current block has two reference blocks in the previous frame only. The inputs of SAD start from the MSD of I_c and I_r. All blocks are represented with three independent digit planes first.

The proposed MSD-first SAD manipulates in signed-digit representations faster and more easily than in the two's complement SAD. It accumulates sequences of the digit-level subtraction outputs. The MSD-first comparator then receives the SAD results of the candidate blocks digit-by-digit. Once the MV is distinguished, the remaining SAD operations are discarded. In the succeeding subsections, we will discuss the word-level comparison and detail our digit-level comparison to show the concept and the benefit of the MSD-first ME.

2.2 Digit-level SAD with word-level comparison

To illustrate how the redundant operations can be removed, we first present the MSD-first ME with word-level comparison and show the computation unit for the following analysis. In Fig. 1*b*, the ME depicted in Fig. 1*a* is redrawn in digit planes. The current and reference blocks with d digits are shown in d digit planes, where the $d-1$ planes denote the MSD plane. Fig. 3*a*, which is derived from Fig. 1*b*, unfolds the digit planes to represent the centre-biased displacement vector with a search range $[-2, 2]$ for simplicity. The circles in (m,n) plane represent the search points, where the indices m and n are the same as those in Fig. 1*b*. In FSBM, there are 25 candidate blocks for each current block to be matched. For four-digit luminance pixels, the SAD block matching can be divided into digit-3 ($z=3$, MSD), digit-2 ($z=2$), digit-1 ($z=1$), and digit-0 ($z=0$, LSD) as shown in Fig. 3*a*. The execution sequence starts with the MSD plane ($z=3$). The number listed in each circle indicates the searching order. Thus, 100 (25 candidate blocks $\times$ 4 digit planes) digit-level SAD operations are required. One digit-level SAD block matching needs $N \times N$-digit SAD operations, which is denoted by one digit SAD. In other words, the 100-digit SADs are equally divided into four digit planes. The additional operation of the succeeding digit SADs (except MSD) for a displacement vector (m,n) is doubling (multiply by 2; assume the radix is 2) and addition. The word-level comparison starts after LSD-plane SAD to find the best matching block. Fig. 3*a* summarises the MSD-first SAD operations with word-level comparison.

2.3 Digit-level SAD with digit-level comparison

Whenever a candidate is sure not to be the best matching block, the remaining SAD operations of its less significant digit planes can be eliminated to reduce the computations. Here, we decompose the word-level comparisons into digit-level comparisons and perform each digit-level comparison after the digit SAD block matching.

2.3.1 Normal mode: The normal-mode MSD-first ME processes the search points from the top left corner as the number shown in Fig. 3*b*. If some reference blocks are removed from the candidates during the lth digit plane ($z=l$) processing, these blocks will be discarded in all less significant digit planes ($z<l$). The grey circles in Fig. 3*b*

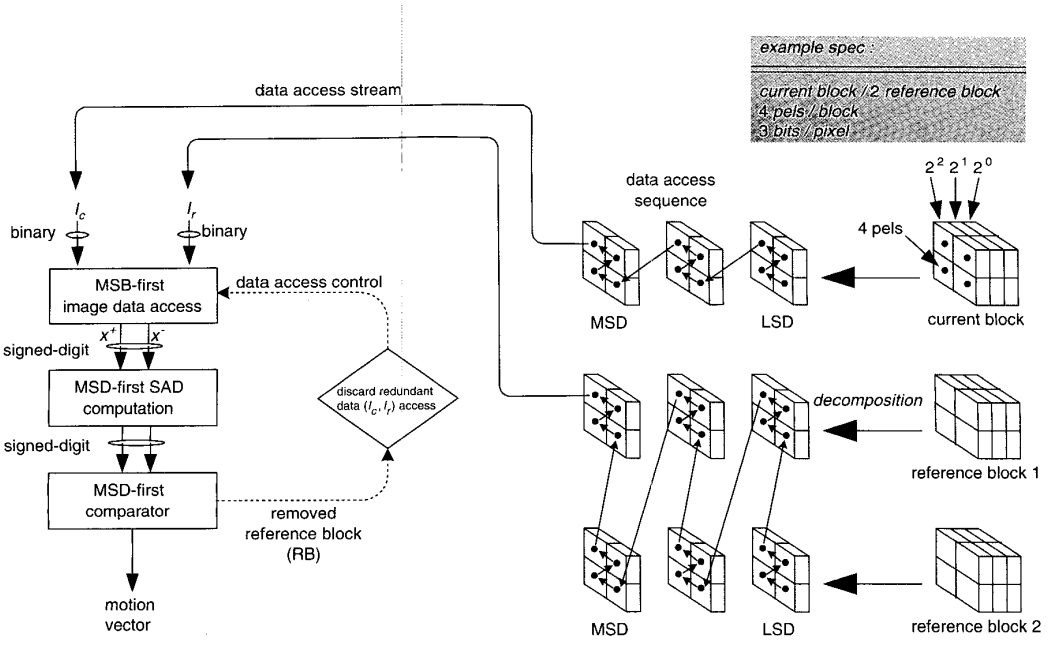

Fig. 2 *Digit-level MSD-first ME system*

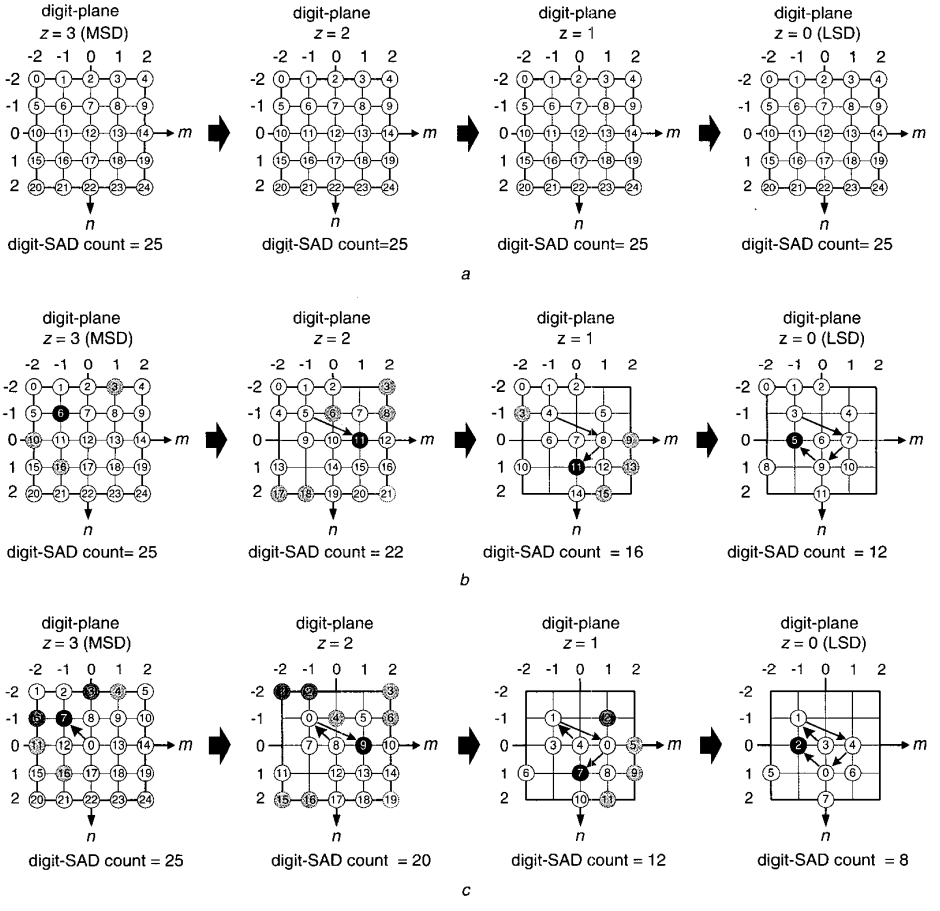

Fig. 3 *Digit-level block matching diagram*
a Word comparison
b Normal mode
c Prediction mode

indicate these removed search points. For example, the displacement vectors (1,−2), (−2,0), and (−1,1) in Fig. 3*b* are removed in the MSD plane, so these vectors in the succeeding planes ($z<3$) are empty. Because three candidate block operations are removed, only 22 digit SADs remain in the less significant planes. The MSD-first SAD operates in the same way for the succeeding less significant planes and thus, it needs gradually decreasing operations for each plane. In the example of Fig. 3*b*, the number of total digit SAD operations of normal mode is 75 (25 + 22 + 16 + 12).

2.3.2 Prediction mode: The candidate with minimum SAD in each digit plane, which is shown in black circles in Fig. 3*b*, is not necessary the best candidate due to the properties of the redundant number systems. However, it will be the actual MV as the digit-level SAD progresses in most situations. The candidate with temporal minimum SAD can guide our proposed MSD-first ME to discard candidates earlier, and hence the black circles in higher digit planes are chosen as the starting search points in their succeeding digit planes. This searching order is named 'prediction mode MSD-first ME' as shown in Fig. 3*c*. The starting point of MSD plane is an exception. It always begins with central position because no reference information exists and general video sequences have most possible MV with (0,0).

In most situations, the black circle in the next lower digit plane still has less SAD than that on the top left corner. This modified search sequence will discard more candidates, which are distinguished as deeper grey circles in Fig. 3*c*. The deeper grey circles of the MSD plane are located on (0,−2) and (−2,−1) in the example. They all occur in the path of top left corner to the first search point, the circle with number zero, because this path with less temporal SAD removes more candidates. Since the search points of next lower digit plane ($z=2$) can reduce down to 20 circles (digit SADs), fewer than that (circle = 22) of normal mode. We add the number of each digit SAD operations of prediction mode, which is 65 (25 + 20 + 12 + 8) in this example.

In brief, our proposed MSD-first ME decomposes the SADs into digit-level SADs and interleaves the comparisons after these digit SADs to discard unnecessary operations early. The computations can be reduced more with the prediction mode, which gives a less temporal minimum SAD. However the actual reduction of operations depends on the characteristics of the video sequence. The simulation of various video sequences is available in the next Section, which demonstrates the effectiveness of our proposed algorithm.

3 Simulation results and analysis

We have implemented our proposed MSD-first ME in C to simulate the digit-level operations. The image size of the test video sequences is 176 by 144 (QCIF). Each test video stream has more than 150 frames. The search range for full-scale motion estimation is ±15 pixels and the images have eight-bit luminance pixels. Two video sequences are chosen 'Suzie', which is composed of simpler pictures and most macroblocks have similar and small MV values, and 'Trevor', which, in contrast, has six reporters with different directions and larger MV values. Fig. 4 shows the first frame of each of the sequences respectively.

3.1 Algorithm simulation results

We have simulated our MSD-first ME of both test sequences respectively with word comparison, digit comparison in normal mode and in prediction mode. Based on

a *b*

Fig. 4 *'Suzie' and 'Trevor' images*
a 'Suzie'
b 'Trevor'

the aforementioned specification, one macroblock matching has 961 (31 × 31) candidate blocks. We first obtain the varying number of removed search points of these sequences and an algorithm is constructed with the three-nest loop structure. Referring to the MSD-first SAD with word comparison in Fig. 3*a*, the first C-program processes the entire digit SAD block matching for 7688 ($m \times n \times z = 31 \times 31 \times 8$) iterations. The second program is modified from the first one, and includes extra digit-level comparisons to remove impossible candidates in normal mode. The variable, Remnant_Candidate_Block, indicates the number of remnant candidate blocks relative to the iteration index as plotted in Fig. 5. The altitude of the

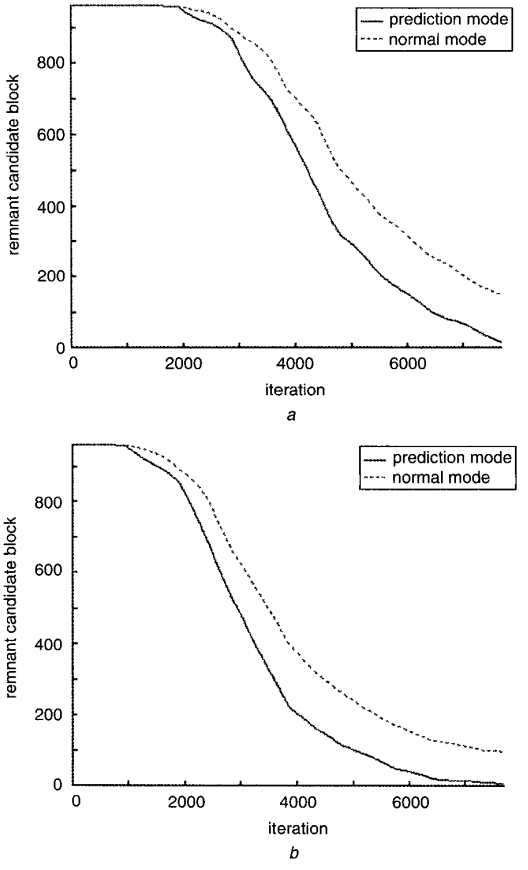

Fig. 5 *MSD-first ME algorithm simulation: comparison of normal mode and prediction mode*
a 'Suzie'
b 'Trevor'

vertical axis is 961, which represents the number of candidates in the video sequences. The width of horizontal axis is 7688, which stands for the iteration count of the total digit SADs. The tick of the horizontal axis is 961/step, which equals to one digit plane ($m \times n$). The number of removed candidates increases as the digit SAD iteration continues.

In Fig. 5, the steeply dropping curve represents higher efficiency of our algorithm, which means the candidates are discarded more early. The algorithm efficiency is represented in the reduction rate of the digit SAD operations. Our simulation illustrates the relationship between the algorithm efficiency and the characteristics of the video sequences, which is denoted by the average remnant blocks of the individual sequences. The prediction-mode ME algorithm always has a curve below that of the normal mode, and thus it always discards the candidates more rapidly than the normal mode. So we can conclude that the prediction mode is always more efficient. In both our examples in Fig. 3*b* and Fig. 3*c*, more than one candidate remains on the LSD plane to be distinguished as the best matching block in the final comparison. The distinction of the exact MV may occur earlier (i.e., on the first few digit-planes instead of the LSD) for some specific video sequences.

Then, we estimate the required hardware operations for digit SADs, which correspond to the number of circles in Fig. 3. Fig. 6 shows the number of digit SADs (vertical axis)

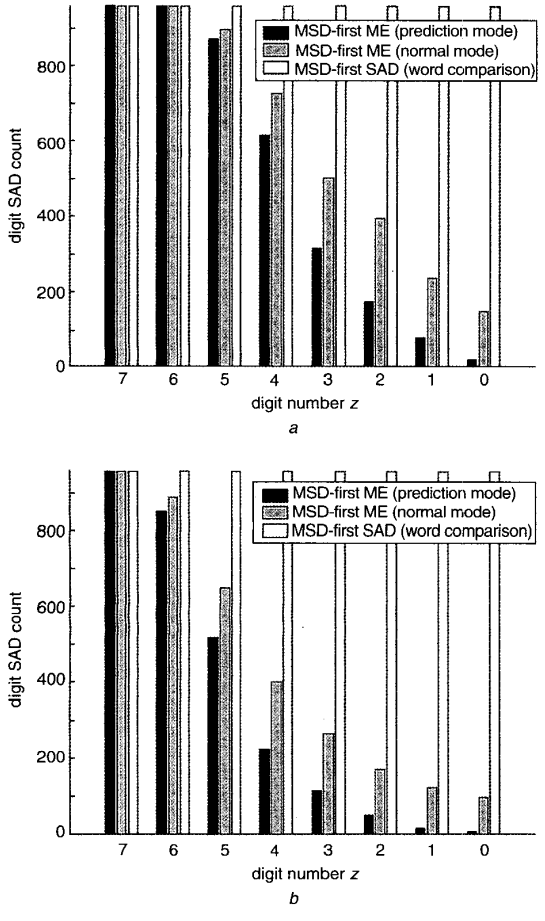

Fig. 6 *Hardware simulation for digit SAD operation count*
a 'Suzie'
b 'Trevor'

of each algorithm (word-level, digit-level normal-mode, and digital-level prediction mode) in each digit plane (horizontal axis). The hardware operations of three algorithms are the sum of digit SADs in eight digit planes of Fig. 6. Table 1 summarises the required hardware operations for different ME algorithms and video sequences. For clarity, the digit SAD of the MSD-first ME is normalised with the word-level comparison. The digit-level MSD-first ME algorithm in prediction-mode reduces the amount of SAD operations to 47.4% ~ 64.3%.

Table 1: MSD-first ME hardware requirement

Percentage of digit-SAD number	Trevor	Suzie
MSD-first SAD operation (word comparison)	100%	100%
MSD-first ME (normal mode)	44.5%	61.4%
MSD-first ME (prediction mode)	35.7%	52.6%

3.2 Algorithm property analysis

The simulation results show the proposed algorithm performs more efficiently for 'Trevor' than 'Suzie'. We are going to explain it in this subsection and analyse what characteristics in the video sequences our algorithm prefers. The algorithm performs digit-level comparisons on the SAD results to discard impossible candidates from the MSD plane. This is illustrated more clearly in Fig. 3. Each circle (candidate block) has a digit-level SAD value. If the distribution of these SAD values is more scattered, the algorithm is able to remove more candidates. Fig. 7 shows the standard deviation of the digit-level SAD, which represents the distribution of individual SAD in each digit plane. 'Trevor' always has higher deviation and hence larger difference for recognition of impossible candidates, and therefore has better algorithm efficiency.

The relationship between block displacement (m,n) and the average SADs of two video clips are depicted in Figs. 8*a* and 8*b*. 'Trevor' has larger average SAD representing the fact that most displacement vectors (m,n) have larger SAD value. This is also indicative of video sequences with

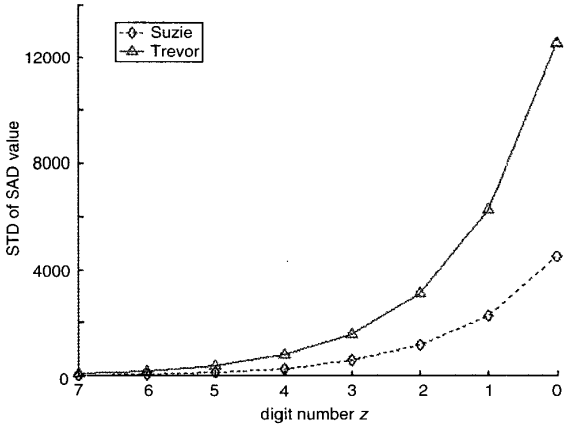

Fig. 7 *Standard deviation of digit SAD against digit number*

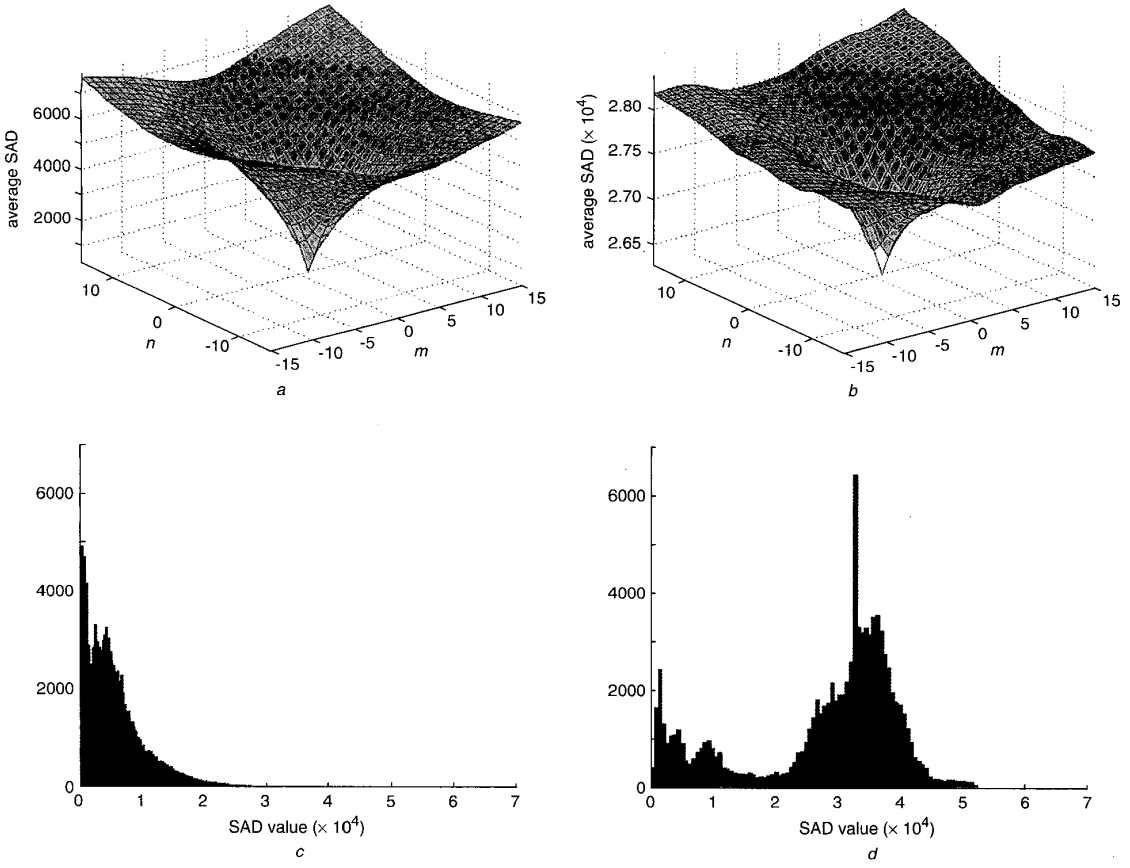

Fig. 8 *SAD value distribution*
a Block displacement against average SAD value for 'Suzie'
Average minimum SAD = 332; average total SAD = 5629
b Block displacement against average SAD value for 'Trevor'
Average minimum SAD = 27686; average total SAD = 27934
c SAD histogram for 'Suzie'
d SAD histogram for 'Trevor'

complex texture and randomly moving objects. For example, the larger number of objects in 'Trevor' can be regarded as complex texture and each reporter moves toward different directions. Our proposed algorithm works best with large minimum SAD value. This is usually due to image rotation, skew (displacement in fraction steps), or zoom in/out. For example, there are image skew (due to lower resolution) and object rotation in 'Trevor'. Figs. 8*c* and 8*d* show the SAD histograms of video sequences that illustrate the SAD distribution. In brief, our proposed

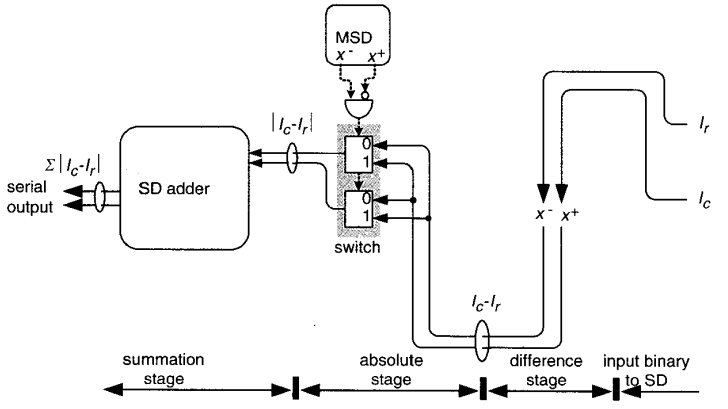

Fig. 9 *Online arithmetic SAD*

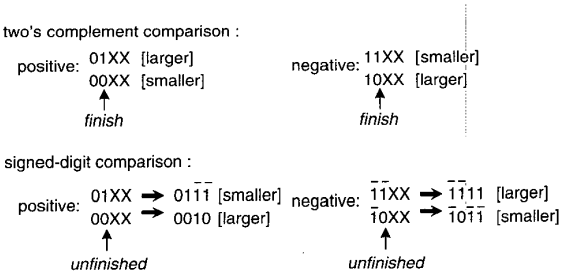

Fig. 10 *Comparison example*

MSD-first ME algorithm is more efficient for video sequences with more complex and rapidly changing images.

4 Implementation of ME

We implement the proposed MSD-first ME algorithm based on the radix-2 Avizienis' signed-digit number, which is one of the most popular redundant number systems. It is represented in two unsigned binary numbers, one is positive and the other is negative, i.e. $X=X^+-X^-$. Each signed digit can be individually represented as $x_i=x_i^+-x_i^-$, where $x_i^+, x_i^- \in \{0,1\}$. Therefore x_i belongs to $\{-1,0,1\}$. All of our digit-level designs use this number system. In the following subsections, we will detail the key components, online SAD and comparator.

4.1 MSD-first SAD component

Equation (1) can be rewritten into the signed-digit format.

$$\sum_{i=1}^{N}\sum_{j=1}^{N}|I_c(i,j)-I_r(i+m,j+n)|,$$
$$+\ p \geq m,n \geq -p; I_c, I_r \geq 0$$
$$=\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\sum_{l=0}^{d-1}I_c^l(i,j)\times 2^l-\sum_{l=0}^{d-1}I_r^l(i+m,j+n)\times 2^l\right|,$$
$$I_c^l, I_r^l \in \{0,1\}$$
$$=\sum_{l=d-1}^{0}\sum_{i=1}^{N}\sum_{j=1}^{N}|I_c^l(i,j)-I_r^l(i+m,j+n)|\times 2^l \qquad (4)$$

The I_c and I_r are decomposed into digit-level components and (4) can be partitioned into three primitive operations for hardware implementation: (i) difference (I_c-I_r) (ii) absolute; and (iii) summation.

4.1.1 Difference: The luminance of the video sequence is always unsigned binary, and the bit-level I_c and I_r belong to $\{0,1\}$. Therefore, (I_c-I_r) can be represented in a signed-digit number just by assembling the two signals with the same weight and binding them together (i.e. $x=x^+-x^-$). Thus the difference is free from computation.

4.1.2 Absolute: Signed-digit negation does nothing but a switching process, i.e. $(x^+ \Leftrightarrow x^-)$. Thus, signed-digit absolute only requires a sign checker and a MUX.

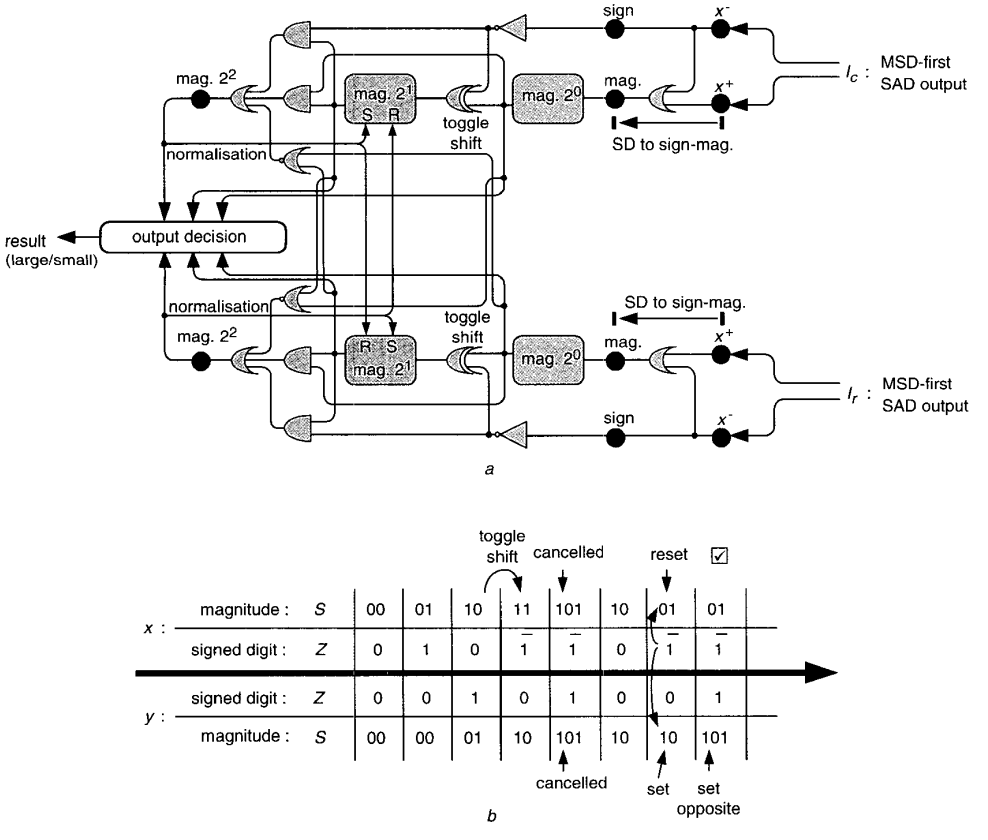

Fig. 11 *MSD-first comparator*
a SAD SD comparator gate level design
b MSD-first comparison example

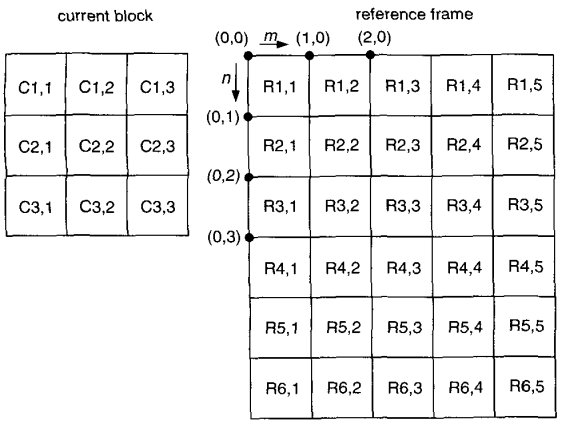

Fig. 12 *Simplified ME example*

4.1.3 Summation: The digits of the same weight (2^i) are clustered and accumulated. Since it is $N \times N$ two nest summations, the maximum result may be a $\lceil 2 \times log_2 N \rceil$ digit value. The digit SAD processes input digits from the MSD. Fig. 9 shows our hardware design for the online arithmetic SAD.

4.2 MSD-first comparator

The comparison is problematic in the redundant number systems because multiple representations exist for a single value. What follows is a simple example to illustrate this problem in Fig. 10. In the two's complement representation, we compare the two numbers digit-by-digit from the MSD. The comparison is completed after the processing of the second digit. The distinction is drawn as soon as the first different digit appears. However, we cannot figure out at the same position because of the redundant representation as the above signed-digit case. Thus, the MSD-first

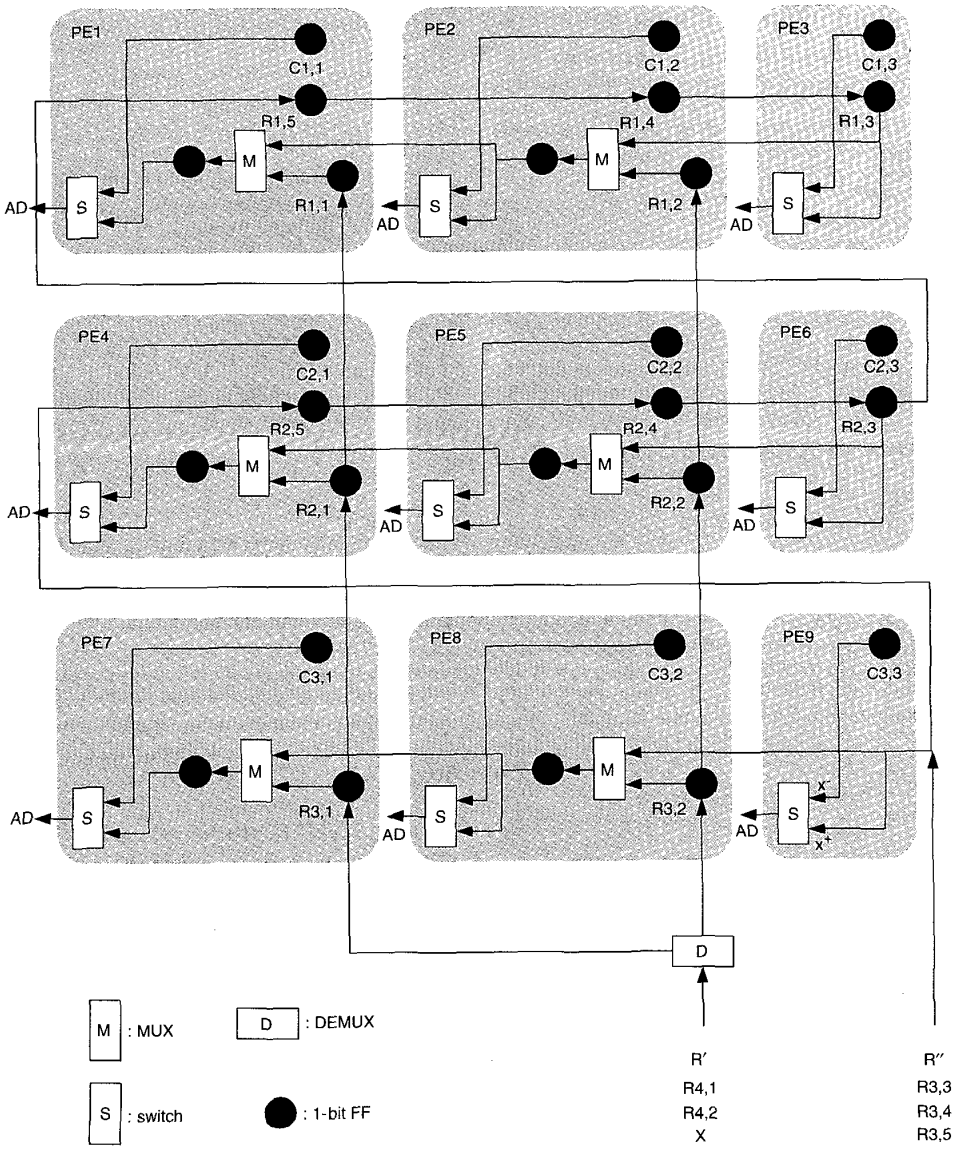

Fig. 13 *Array processor for MSD-first ME*

comparison is the major challenge in redundant number systems.

To conquer this problem, we have shown that the signed-digit comparator calculates the difference of the two numbers being compared till the difference is equal to or large than two [21]. We start the calculation of difference by first converting each number into its signed-magnitude representation. To reduce the internal registers of the comparator, the two signed-magnitude numbers to be compared are further translated into two numbers with an identical relative magnitude by discarding a bias value. Fig. 11*a* shows the hardware design of the MSD-first comparator. The comparator design is compact with only a three-gate delay in the critical path. Fig. 11*b* illustrates an MSD-first comparison. The signed-digit-Z is the input digit and the magnitude-S is the content of registers (Mag. 2^1, Mag. 2^0) in Fig. 11*a*. After eight input digits, the comparator detects the input y is the larger number. The hardware complexity of our comparator design shown in Fig. 11*a* is constant and independent of the wordlength. Detailed description of our MSD-first comparator design is available in [21].

4.3 Data generation unit

This implementation needs the off-chip frame memory with reference data. The data generation unit supports fast skip while the successive digit SAD is unnecessary. It also explores the data reuse for reducing the off-chip bandwidth. Fig. 12 shows an illustrative example of 3×3 block matching with 12 (3×4) candidates in row-major order. The proposed architecture shown in Fig. 13 consists of buffers organised in a 3×3 array, and each black circle represents a 1-bit flip-flop. The C (or R) index below the black circle depicts the initial data to perform the first digit SAD. The switch is identical to that of the MSD-first SAD in Fig 9.

If k candidates are skipped, which equals to p and q steps vertically and horizontally, respectively (i.e. $k = Np + q$, N is the block size), it is simply accomplished by shifting R′ and R″ up by p steps and shifting R′ and R″ left and right respectively by q steps simultaneously. Thus, each block matching need access the reference block at most once to reduce the memory bandwidth. The buffer size is under $4N^2$ bits, which is smaller than the previous digit-parallel architectures. The datapath is regular with simple control circuitry, which is suitable for VLSI implementation.

4.4 Performance evaluation and comparisons

Here, we have implemented and made comparisons among the proposed architecture and the previous solutions [16].

Table 2: MSD-first ME building block simulation

Technology	0.35 μm 1P4M 3.3 V TSMC
Mux	area = 2.345 gate (1 gate = 4 trans.)
	power = 0.413 mV
	data arrival time = 0.41 ns
SD adder	area = 14.7 gate (1 gate = 4 trans.)
	net switching power = 2.4712 mV
	data arrival time = 1.76 ns
SD comparator	area = 19.7 gate (1 gate = 4 trans.)
	dynamic power = 0.677 mV
	data arrival time = 1.43 ns

TSMC = Taiwan Semiconductor Manufacturing Company

Table 3: Comparison of different ME systems

	MSD-first ME	Bit-parallel ME
Combinational area	755.66	2591
Non-combinational area	752	6389
Interconnection area	2.6136	11
Total area	1510 gates	8992 gates
Data arrive time	2.84 ns (= 352 MHz)	10.2 ns (= 98 MHz)
Equalisation SAD count	83 Mega SAD	98 Mega SAD
SAD/gate-sec.	0.054966	0.010898
(normalisation)	(5.04)	(1)

Our implementation use 0.35 μm cell library [27] with 1P4M (one-poly–four-metal) layers. The synthesis and simulation results are listed in Table 2. The single SAD and comparison elements for the proposed algorithm require only 75.685 gates, including the output registers. A complete ME system is composed of these computation elements and the data generation unit. Table 3 summarises the performance and gate count of an illustrative ME system with 16 candidate blocks of size 4×4 on 8-bit pixels, and the reference design available in [16]. Our online arithmetic-based MSD-first ME operates at 352 MHz and the proposed algorithm least skips 47.4% digit SAD operations in FSBM. The array processor has an extremely high throughput of equivalent 82 MHz SAD operations (i.e. wordlength × algorithm efficiency × clock rate). By contrast, the digit-parallel approach in [16] supports 98 MHz SAD operations with 4.95 times more gates. In summary, the final architecture provides 5.04 times computation power to perform SAD per silicon gate than that in [16].

5 Conclusions

In this paper, a new MSD-first ME algorithm with its implementation using online arithmetic has been proposed. We decomposed the SAD and comparison of ME into digit-level operations and interleaved these operations to remove redundant operations, while extracting the exact MV. Complete software simulation and efficiency analysis are available, which demonstrate, the effectiveness of our proposed algorithm. Our simulation results show that on average, the proposed algorithm can save 47.4% to 64.3% of SAD computations of FSBM.

Novel SAD and comparator designs are also proposed to support high throughput. The synthesised MSD-first array processor has a 2.84 ns cycle time and 1510 gates with 0.35 μm 1P4M cell library, which can support 83 MHz SAD for FSBM. Moreover, our proposed algorithm can be easily combined with existing fast search algorithms to further improve their efficiency.

6 References

1 RAO, K.R., and HWANG, J.J.: 'Techniques and standard for image, video and audio coding' (Prentice Hall Inc, 1996)

2 KOMAREK, T., and PIRSCH, P.: 'Array architectures for block matching algorithms', *IEEE Trans. Circuits Syst. Video Technol.*, 1989, **36**, (10), pp. 1301–1308

3 JAIN, R.J., and JAIN, A.K.: 'Displacement measurement and its application in interframe image coding', *IEEE Trans. Commun.*, 1981, **29**, pp. 1799–1808

4 KOGA, T., LINUMA, K., HIRANO, A., IIJIMA, Y., and ISHIGURO, T.: 'Motion compensated interframe coding for video conference'. Proceedings of NTC, Nov. 1981, pp. C9.6.1–5

5 GHANBARI, M.: 'The cross-search algorithm for motion estimation', *IEEE Trans. Commun.*, 1990, **38**, pp. 950–953
6 CHRISTOPOULOS, V., and CORNELIS, J.: 'A center-base adaptive search algorithm for block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 2000, **10**, pp. 423–426
7 SU, L.K., and MERSEREAU, R.M.: 'Motion estimation methods for overlapped block motion compensation', *IEEE Trans. Image Process.*, 2000, **9**, pp. 1509–1521
8 BIERLING, M.: 'Displacement estimation by hierarchical block matching', *Proc. SPIE –Int. Soc. Opt. Eng.*, 1988, **1001**, pp. 942–951
9 LIU, B., and ZACCARING, A.: 'New fast algorithms for the estimation of block motion vectors', *IEEE Trans. Circuits Syst. Video Technol.*, 1993, **3**, pp. 148–157
10 CHAN, Y.L., and SIU, W.C.: 'New adaptive pixel decimation for block motion vector estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, pp. 113–118
11 WANG, Y., WANG, Y., and KURODA, H.: 'A globally adaptive pixel-decimation algorithm for block-motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 2000, **10**, pp. 1006–1011
12 CHALIDABHONGSE, J., and KUO, C.-C.: 'Fast motion vector estimation using multiresolution-spatio-temporal correlations', *IEEE Trans. Circuits Syst. Video Technol.*, 1997, **7**, pp. 477–488
13 KOSSENTINI, F., LEE, Y.W., SMITH, M., and WARD, R.K.: 'Predictive RD optimized motion estimation for very low bit-rate video coding', *IEEE J. Sel. Areas Commun.*, 1997, **15**, pp. 1752–1763
14 CHIMIENTI, A., FANUCCI, L., LOCATELLI, R., and SAPONARA, : 'VLSI architecture for a low-power video codec system', *Microelectron. J.*, 2002, **33**, (5), pp. 417–427
15 HE, Z.L., TSUI, C.T., CHAN, K.K., and LIOU, M.L.: 'Low-power VLSI design for motion estimation using adaptive pixel truncation', *IEEE Trans. Circuits Syst. Video Technol.*, 2000, **10**, pp. 669–678
16 LAI, Y.K., and CHEN, L.G.: 'A data-interlacing architecture with two-dimensional data-reuse for full-search block-matching algorithm', *IEEE Trans. Circuits Syst. Video Technol.*, 1998, **8**, pp. 124–127
17 KOMAREK, L.K., and PIRSCH, P.: 'Array architectures for block matching algorithms', *IEEE Trans. Circuits Syst.*, 1989, **36**, pp. 124–127
18 SU, C.L., and JEN, C.W.: 'Motion estimation using online arithmetic'. Proceedings of IEEE International Symp. on Circuits and systems, 2000, Vol. 1, pp. 683–686
19 PARHI, K.K.: 'VLSI digital signal processing systems: design and implementation' (Wiley, 1999)
20 AVIZIENIS, A.: 'Signed-digit number representations for fast parallel arithmetic', *IRE Trans. Electron. Comp.*, 1961, **EC-10**, (9), pp. 389–400
21 HARATA, Y., NAKAMURA, Y., NAGASE, H., TAKIGAWA, M., and TAKAGI, N.: 'A high-speed multiplier using a redundant binary adder tree', *IEEE J. Solid-State Circuits*, 1987, **SC-22**, (1), pp. 28–33
22 JAIN, S.K., SONG, L., and PARHI, K.K.: 'Efficient semisystolic architectures for finite-field arithmetic', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1998, **6**, pp. 101–113
23 ERCEGOVAC, M.D.: 'Online arithmetic: an overview', *Proc. SPIE - Int. Soc. Opt. Eng.*, 1984, **495**, pp. 86–93
24 McQUILLAN, S.E., and McCANNY, J.V.: 'A systematic methodology for the design of high performance recursive digital filters', *IEEE Trans. Comput.*, 1995, **44**, (8), pp. 971–982
25 KOPPENHÖFER, B.: 'A novel architecture for a decision-feedback equalizer using extended signed-digit feedback'. Proceedings of IEEE International Conf. on Application-specific array processors, 1993, pp. 490–501
26 OSORIO, R.R., ANTELO, E., BRUGUERA, E. J.D., VILLALBA, J., and ZAPATA, E.L.: 'Digit online large radix CORDIC rotator'. Proceedings of IEEE International Conf. on Application-specific array processors, 1995, pp. 246–257
27 'Passport 0.35 micron, 3.3 volt, Optimum Silicon SC Library, CB 35OS142', Avant! Corporation, Mar. 1998