Efficient hybrid tree/linear array architectures for block-matching motion estimation algorithms

M.-J. Chen
L.-G. Chen
K.-N. Cheng
M.C. Chen

Indexing terms: Motion estimation, Videoconferencing, Video coding

Abstract: Execution latency and I/O bandwidth play essential roles in determining the effectiveness and the cost of a parallel hardware implementation for block-matching motion estimation algorithms. Unfortunately, almost all traditional architecture designs, e.g. the two-dimensional mesh-connected systolic array architecture (2DMCSA), and the tree-type structure (TTS), fail to take these two factors into account simultaneously. As a result, they suffer from either large execution latency or huge input bandwidth requirements. The authors propose a family of tree/linear architectures, which efficiently optimise the total implementation cost by combining the merits of the 2DMCSA and the TTS. Moreover, to facilitate hardware designs, the authors present the tree-cut techniques and the on-chip buffer design method to meet computational demands of various video compression applications. Since the proposed architectures are capable of executing the exhaustive search and the fast search block-matching algorithms, they offer relatively flexible and cost-effective hardware solutions for a wide range of video coding systems, including CD-ROM, portable visual communications systems and high-definition TV.

1 Introduction

During recent years the vigorous growth of personal computers together with the rapid progress of digital communications technologies have spurred the evolution in transforming information into a digital format. A digital representation of signals has advantages over its analogue counterpart in terms of noise immunity and signal processing feasibility. Consequently, it provides considerable leverage of signal quality for storage, transmission, and duplication. However, the obvious disadvantage of such approach is that the size of data becomes several orders larger. To economically utilise limited communications networks and storage resources, it is necessary for digital image/video applications to employ data compression techniques. Several video coding standards, such as ITU CCITT H.261 [1], H.263 [2] standards, and MPEG [3–6] standards, have been developed to provide a platform for interportability. Because the removal of temporal redundancy between successive image frames relies heavily on the use of a block-matching motion estimation technique, the design of a block-matching motion estimation algorithm and its corresponding hardware structure considerably affect the performance and the cost of a video coding system.

Among various block-matching algorithms, the 2-D full search (or exhaustive search) method (2DFS) dominates most parallel hardware implementations [7–11] because of a high degree of regularity in its memory accesses and computations. However, a massive computational effort is usually needed even for the applications with a moderate frame rate and a medium frame size. For instance, it requires over 778 million subtract/accumulate operations and 1,557 million data accesses per second to perform the 2DFS algorithm with a search range –8 to 7 and a block size of 16×16 for a standard videoconferencing image sequence (352 lines $\times$ 288 pixels/line $\times$ 30 frames/sec). For the applications where power consumption and processing speed are critical, it is clear that trading-off estimation performance with computation complexity is necessary.

In the past, various fast search algorithms [13–18] have been proposed to alleviate the computational burden imposed by the 2DFS method. Basically the majority of these fast search algorithms reduce the number of searched positions largely based on the assumption that the mean-absolute-error (MAE) between the current-frame block and the previous-frame search candidate increases monotonically as the search position moves away from the best match position. Consequently, compared to the 2DFS, they usually place more serious demands on the supporting memory systems since rather flexible data access patterns are needed. Moreover, since the distance between each consecutive search position often varies with the search step, e.g. the three-step hierarchical algorithm, it is difficult to take advantage of interblock data dependency to reduce input bandwidth.

IEE Proceedings online no. 19960541
Paper first received 6th November 1995 and in revised form 9th April 1996
M.-J. Chen, L.-G. Chen and K.-N. Cheng are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 10764, Republic of China
M.C. Chen is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90024, USA

Recently, we presented the one-dimensional full search algorithm (1DFS) [19]. The method distinguishes itself from other fast algorithms by the fact that it maintains the regularity of the 2DFS scheme while an effective tradeoff between computational complexity and estimation accuracy is achieved. This attractive feature makes it possible to optimise input bandwidth without resorting to a sophisticated control scheme for the scheduling of memory access activities. Intuitively, parallel architectures for the 2DFS can be modified slightly to implement the 1DFS algorithm. Unfortunately, the 1DFS method, like most other fast search algorithms, also falls into the hierarchical search category. Thus, low execution latency is desirable since the *n*th step search process cannot start before the search result of the (*n*–1)th step search process is readily available. Although using block-level interleaving [12] can eliminate the waiting cycle for a massive pipeline structure, the higher the execution latency that an architecture possesses, the more the segmented image blocks are interleaved. As a result the hardware overheads in arranging and buffering input data, generally dictated by the execution latency of an architecture, can become exceedingly high if conventional systolic array architectures are used.

For parallel implementations of block-matching algorithms most current designs can be broadly classified as the 2DMCSA [9] and the TTS [12]. The 2DMCSA exploits data dependency between each consecutive search position by properly inserting data-skew registers into the processing element array based on a predetermined time schedule and a systematic projection method [20]. Though a minimum amount of input bandwidth is achieved, it leads to huge execution latency since the final results can only be obtained after the computations travel through a long propagation path. On the other hand, the TTS conducts the computations of each displaced candidate in an as-soon-as-possible (ASAP) fashion such that the execution latency is greatly optimised. Since no-data skew is allowed, image pixels associated with a displaced candidate have to be accessed simultaneously to ensure high utilisation of processing elements (PEs). However, the input pin-count increases rather rapidly as more PEs are integrated into a chip [Note 1], thereby making the structure impractical whenever the target applications demand large numbers of PEs. To keep input bandwidth under a reasonable number while minimising execution latency, our effort is to investigate various possibilities in combining the appealing attributes of these two architectures.

2 2DFS and 1DFS block-matching algorithms

For the block-matching motion estimation a current-frame image from a video sequence is divided into blocks with a size of $N \times N$. Then, for a maximum motion displacement of p pels, each segmented block is compared with the displaced candidate blocks in the previous frame within a $(N+2p) \times (N+2p)$ square-shaped search window. A two-dimensional motion vector is selected for each block such that the distortion resulting from a block-matching motion compensation scheme is minimised.

Note 1: Although data sharing between two consecutive search blocks can be used to reduce input bandwidth, in practice, the scheme is rarely adopted owing to the fact that a quite large hardware overhead is necessary to handle the situation when the search process proceeds to a new row or column of search candidates. Consequently, we assume that no interblock data dependency is exploited for the TTS in our discussion.

The straightforward search scheme is the 2DFS where the distortion corresponding to all $(2p+1)^2$ candidates within a search area are calculated and compared. The optimal motion vector is chosen by selecting the search position that gives the lowest distortion. Usually, the MAE criterion, defined by eqn. 1, is favoured against the mean-square-error criterion owing to its simplicity and computational efficiency for implementations.

$$MAE(i,j) = \frac{1}{N^2}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}|X_k(m,n) - X_{k-1}(m+i,n+j)| \tag{1}$$

where $X_k(m, n)$ is the pixel value at the position (m, n) in the frame k. Instead of examining all candidates within a two-dimensional search window, the 1DFS method adopts a coarse-to-fine search procedure. Centred at (0, 0), the 1DFS first searches through all displaced candidates along one co-ordinate, e.g. the horizontal dimension, to find the position having the minimum distortion. Based on this result the 1DFS in a similar manner manages to locate the best match position in another co-ordinate, e.g. the vertical dimension. The best match from the first step becomes the initial displacement for the current block. Centred at this initial displacement, the search procedure repeats with a halved search range. Then the motion vector is determined. To illustrate clearly the 1DFS algorithm, a graphical representation of the search procedures is shown in Fig. 1. As can be seen, the 1DFS preserves the natural dataflow of the 2DFS; therefore it provides an opportunity in exploiting inter-block data dependency to reduce input bandwidth as compared to other fast search schemes. Since the 1DFS and the 2DFS methods are similar in their search characteristics, it is possible to design parallel architectures that can implement them efficiently.

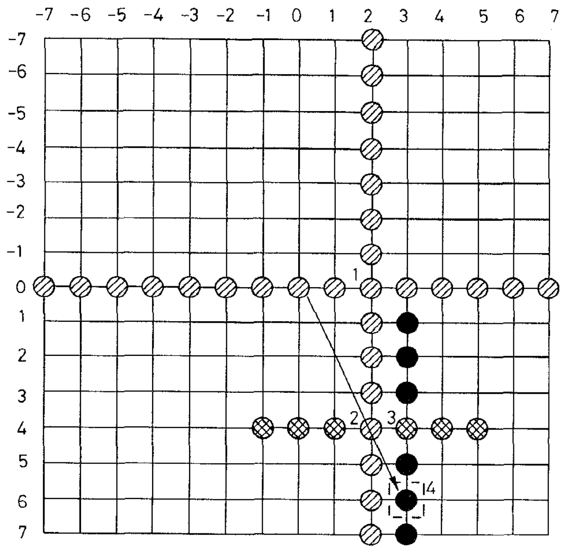


Fig. 1 *Illustration of one-dimensional full search algorithm*
Search range –7 to +7
Motion vector = (3, 6)

3 Two-dimensional mesh-connected systolic array structures and tree-type architectures

The 2DMCSA is originally derived from projecting the four-dimensional dataflow graph (DG) onto a two-dimensional plane. With an appropriate time schedule, the computations can be executed in a massive pipelined parallel structure without any idle cycle while only a small number of input is supplied. To illustrate the design procedure, Fig. 2 shows the three-dimensional DG of the 2DFS algorithm with the time schedule vector (m, i, j) equal to (2, 1, 1). Based on the three-dimensional DG, the two-dimensional systolic architecture ($N = 4$ and $p = 2$) of Fig. 3 is derived by the m- and j-plane projection, respectively. For brevity, here we assume that all pixels of a 4 × 4 current-frame block is preloaded into the individual PE. According to the time scheduling, these PEs perform the operations of absolute pixel difference between the current block pixels and the search window pixels concurrently. Then, these absolute differences are accumulated with the partial sums provided by the adjacent (upper) PEs. After four clock cycles, the partial sum corresponding to each column of the displaced candidates begins to shift to the linear adder array. The MAE of the displaced candidate is obtained at the end of the adder array with eight cycles of delay. These MAE values then enter the motion vector selection unit (M), and the corresponding motion vector is determined.

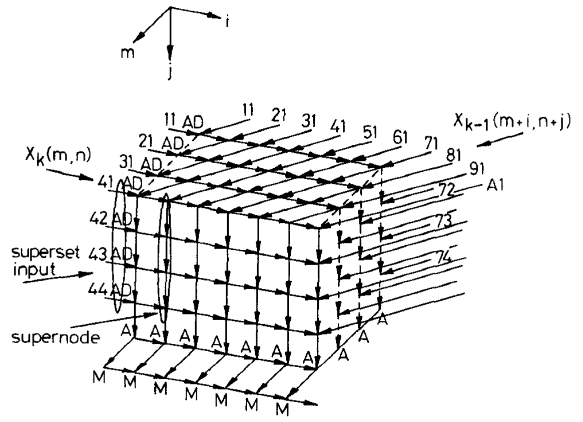


Fig. 2 *3-D dataflow graph of 2DFS algorithm with time schedule vector of (2, 1, 1)*

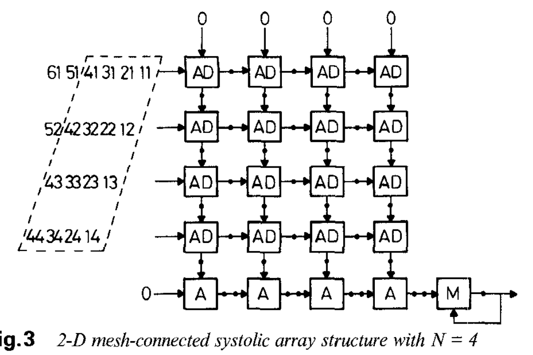


Fig. 3 *2-D mesh-connected systolic array structure with $N = 4$*

The tree-type array architecture decomposes the computations into two sections, where the subtract/absolute section (D) calculates the absolute pixels differences and the accumulate section (A) computes the sums of the absolute differences to obtain the MAE. As shown in Fig. 4, the two-dimensional subtract/absolute DG of the 2DMCSA is recorded into a one-dimensional linear array, whereas the two-dimensional accumulate section is transformed into a tree-type array in order to minimise the number of accumulation stages. The major difference between this approach and the 2DMCSA is the need to access N^2 search area data in parallel. Because the absolute differences corresponding to a reference block are calculated simultaneously within a PE array and they are accumulated in a tree manner, the theoretical shortest propagation path is acheived while the maximal input bandwidth requirement is incurred.

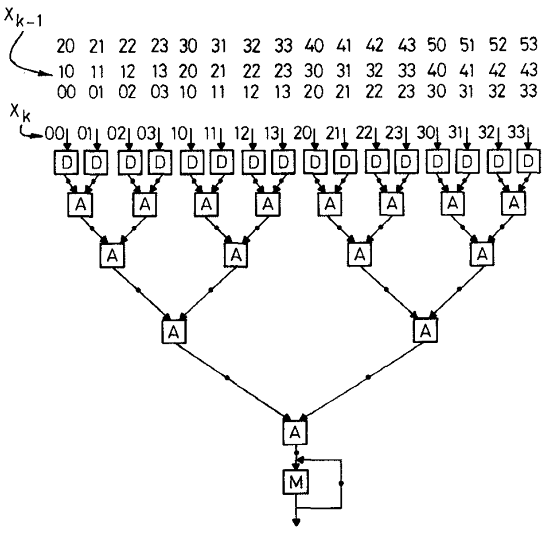


Fig. 4 *Tree-type array architecture with $N = 4$*

4 Hybrid tree/linear architectures

It is clear that the tree-style computation has an advantage in optimising execution latency. On the other hand, the systolic dataflow projection offers a powerful means in maximising on-chip data reusage. Therefore a hybrid architecture that adopts these desired characteristics in the design can potentially provide better tradeoffs with respect to the implementations targeting at the 2DFS and 1DFS algorithms.

The hybrid tree/linear architecture can be viewed as a combination of the tree architecture and the systolic architecture. With respect to the design of the hybrid structure, the tree technique is restricted to the computations corresponding to each M columns (M is smaller than or equal to N), whereas the systolic mapping method is applied to the accumulation process of the partial sums produced by these sub-trees. Consequently, the hybrid structure has a one-dimensional MN-PE subtree array designated for calculating the absolute differences and a linear adder array responsible for accumulating the partial sums to get the final result. To make our design procedure clear, each subtree and its associated inputs can be considered as a supernode and superset input, respectively. If we represent the 2DFS DG using supernodes and superset inputs, the original four-dimensional DG can be transformed into a three-dimensional DG. Again, by projecting the DG onto m-plane, a one-dimensional systolic array, as shown in Fig. 5, consisting of subtrees and parallel pel inputs is obtained.

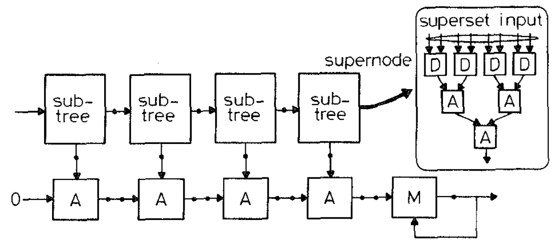


Fig.5 *Hybrid tree/linear architecture with N = 4 and M = 1*

Basically, the TTS and the 2DMCSA are the two special cases of the hybrid structure, where M is equal to N for the TTS while the 2DMCSA degenerates the adder subtree to a single adder. Since both latency and input bandwidth can have strong impacts on the cost of a single-chip implementation, the hybrid structure is capable of efficiently controlling these two factors by choosing an appropriate size of the subtrees.

Table 1 compares number of adders, execution latency, input pin count, number of processing cycles, and flexibility of various architectures. As expected, the TTS has the shortest execution latency, which is on the logarithmic order of the PE number. But it also demands the most in the aspect of input bandwidth. However, the structure has the most flexibility since there exists the least amount of constraint on how the data should proceed within the PE array. Consequently, a wide variety of block-matching algorithms, including the 1DFS, the three-step hierarchical search, and the 2DFS, can be implemented. In contrast, the only algorithm that is suitable for the 2DMCSA is the 2DFS because the large control overhead incurred by its huge execution latency can easily prevent any hierarchical fast algorithms from being efficiently implemented. The hybrid architecture, on the other hand, tradeoffs latency with input bandwidth. By varying M, the hybrid structure can control execution latency and input bandwidth so that a single-chip design with reasonable I/O pin count and control overhead can be realised.

4.1 Tree-cut techniques

Since a significant amount of inherent parallelism is available in the block-matching algorithms, parallel processing techniques can be extensively employed to speedup the estimation process. The cost of the implementation, however, depends strongly on the number of PEs integrated into a single chip. Therefore for most applications where a moderate computational power is needed, the use of hardware timesharing provides a superior solution. The choices of the PE reduction are made between the number of the subtrees and the number of PEs in the subtrees. In general, execution latency is mainly determined by the number of the subtrees, while input pin count relies heavily on the number of the PEs in the subtree. Because input bandwidth usually creates a more serious problem for a single-chip implementation, our discussions here are restricted to the tree-cut techniques for reducing the number of PEs within each subtree. Nevertheless, the choice between the two factors is not exclusive. In fact, a better system's cost evaluation should take both two factors into consideration.

In the following, the hardware timesharing with a factor of two is assumed in our examples to show the various hardware impacts from two different scheduling techniques. Furthermore, we use the name of degenerated subtree for the subtree under a PEs reduction to distinguish it from the original subtree.

4.1.1 Direct form:

For the direct form approach, the absolute differences of each original subtree are decomposed into two independent parts, i.e. the even part and the odd part, as defined in eqn. 2 according to the n index.

$$MAE(i,j) = \frac{1}{N^2}\sum_{m=0}^{N-1}(SUM_{even}(m) + SUM_{odd}(m)) \quad (2)$$

$$SUM_{even}(m) = \sum_{n=0}^{\frac{N}{2}-1}|X_k(m,2n) - X_{k-1}(m+i,2n+j)|$$

$$SUM_{odd}(m) = \sum_{n=0}^{\frac{N}{2}-1}|X_k(m,2n) - X_{k-1}(m+i,2n+1+j)|$$

The computations of these even and odd partial sums are then interleaved within the degenerated subtrees. Since it takes two cycles to obtain the partial sums of the original subtrees, an accumulator and a recursive loop have to be appended to the degenerated subtree's outputs such that the even and odd partial sums can be accumulated before they are shifted to the linear adder array. Moreover, the switches that control connections between the accumulators and the linear adder array are essential to ensure a proper function. Fig. 6 shows the block diagram of the structure with a direct form scheduling and a block size of 4 × 4, where the overheads for the hardware interleaving are four additional accumulators, registers and switches.

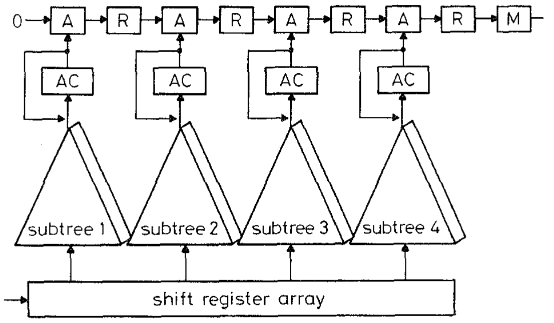


Fig.6 *Block diagram of 1/2-cut hybrid structure with direct form scheduling*

4.1.2 Modified form:

Compared to the direct form approach, the modified form scheme schedules the computations in a way that the odd partial sums and

Table 1: Comparisons of various block-matching architectures

Architecture type	Area (adder)	Latency (cycle)	Cycle required (2DFS)	Input (word)	Flexibility (BMA)
2DMCSA	$2N^2 + N + 1$	$3N + 1$	$(2p + N)(2p + 1)$	N	2DFS
TTS	$2N^2$	$2\log N + 2$	$(2p + 1)^2$	N^2	all
Hybrid	$2N^2 + M$	$\log NM + 2\frac{N}{M} + 2$	$(2p + 2\frac{N}{M})(2p + 1)$	NM	1DFS & 2DFS

the even partial sums are calculated independently before they are finally accumulated at the output of the linear array as depicted by eqn. 3. Because the accumulations between the odd and even partial sums are moved to the output of the linear array, the hardware overheads are minimised to only one accumulator, recursive loop, and switch, as shown in Fig. 7.

$$MAE(i,j) = \frac{1}{N^2}\sum_{m=0}^{N-1} SUM_{even}(m) + \frac{1}{N^2}\sum_{m=0}^{N-1} SUM_{odd}(m)$$
$$SUM_{even}(m) = \sum_{n=0}^{\frac{N}{2}-1} |X_k(m,2n) - X_{k-1}(m+i, 2n+j)|$$
$$SUM_{odd}(m) = \sum_{n=0}^{\frac{N}{2}-1} |X_k(m,2n) - X_{k-1}(m+i, 2n+1+j)| \quad (3)$$

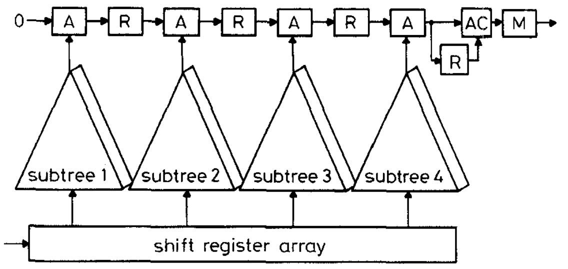


Fig. 7 *Block diagram of 1/2-cut hybrid structure with modified form scheduling*

5 Memory interleaving and on-chip buffer design

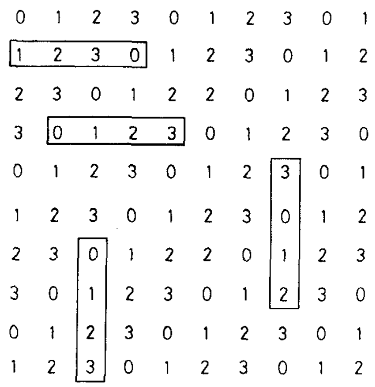
The use of parallel processing significantly enhances the computational capability of a single chip. The challenge, however, lies in the design of the memory system that can sustain a continuous flow of data from memory to processing array without incurring large overheads. Two important schemes, namely: memory interleaving, and on-chip buffering, are employed in the system for the external and internal data accesses, respectively.

5.1 Memory interleaving

The key to ensure that the computational power of the hybrid structure can be fully utilised is to meet the bandwidth requirements imposed by the processing element array. Provided each memory module can perform at most one read operation per clock cycle, the number of memory modules has to match the number of PEs in the subtree, i.e. the number of input ports, to keep all the PEs busy. In addition, due to the rather random characteristics in the data access of the 1DFS algorithm, the distribution of the image pixels has to be done in a way that the adjacent N image pixels can be read simultaneously on each clock cycle, where N represents the number of input ports. One way to achieve the goal is to allocate each N horizontal or vertical adjacent image pixels to different memory modules. Let $X_{k-1}(i, j)$ represent the previous-frame pixel value at index (i, j) and let $MM(i, j)$ denote the memory module that $X_{k-1}(i, j)$ resides. Then the assignment of $X_{t-1}(i, j)$ to N memory modules can be formalised as follows:

$$MM(i,j) = (i+j) \text{ modulo } N \quad (4)$$

To exemplify this memory interleaving scheme, Fig. 8 shows the pel distribution for a block size of 4 × 4, where the numbers indicate the memory modules that the image pixels are allocated. As can be seen, every four consecutive pels always reside on four independent memory modules both horizontally and vertically. Thus, it can provide an efficient data access of either N rows or columns of data simultaneously. In this way, the horizontal search and the vertical search of the 1DFS can be executed successfully.



0	1	2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	2	0	1	2	3
3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	2	0	1	2	3
3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0	1	2

Fig. 8 *Image pel distribution for memory interleaving*

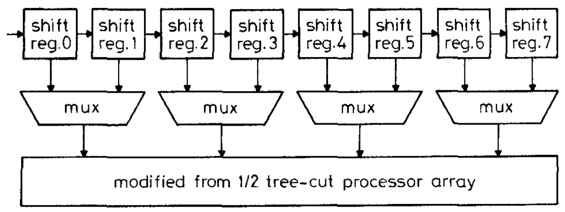


Fig. 9 *Block diagram of shift register array for modified form 1/2 tree-cut array*

5.2 On-chip buffer design

The on-chip buffer maximises the data reusage, thereby relieving the heavy input and memory bandwidth demands placed by the parallel processing elements. To realise the desired dataflow of the hybrid structure, a parallel-in-parallel-out shift register array is used. Unlike the 2DMCSA, each shift register array is composed of N words of shift registers, as shown in Fig. 9. At each clock cycle, new N parallel input data are sequentially shifted into the shift register array and are shifted out after N clock cycles. For the 1/2-cut hybrid structure with either the direct form or the modified form scheduling, the shift register based on-chip buffer is modified to adjust to the changes of the desired dataflow regarding the hardware time-multiplexing. Referring to Fig. 9, the N-word shift registers are split into two $N/2$-word shift registers, where the one holds the odd part inputs and the other is responsible for the even part inputs. By toggling the multiplexers, the inputs of the degenerated array are selected from one of the two shift register arrays.

6 Single chip design

The 1/2-cut hybrid structure with the modified form scheduling has been designed under 0.8µm CMOS technology using Magic. Its functionality has been verified by Verilog. Fig. 10 shows the chip layout, which requires 97 active signal pads and contains 16 8-PE degenerated subtrees in a die size of 12.0 × 4.3 mil^2. According to SPICE simulations, it is functional up to 30MHz, or 33.3ns per cycle. It is sufficient to meet the

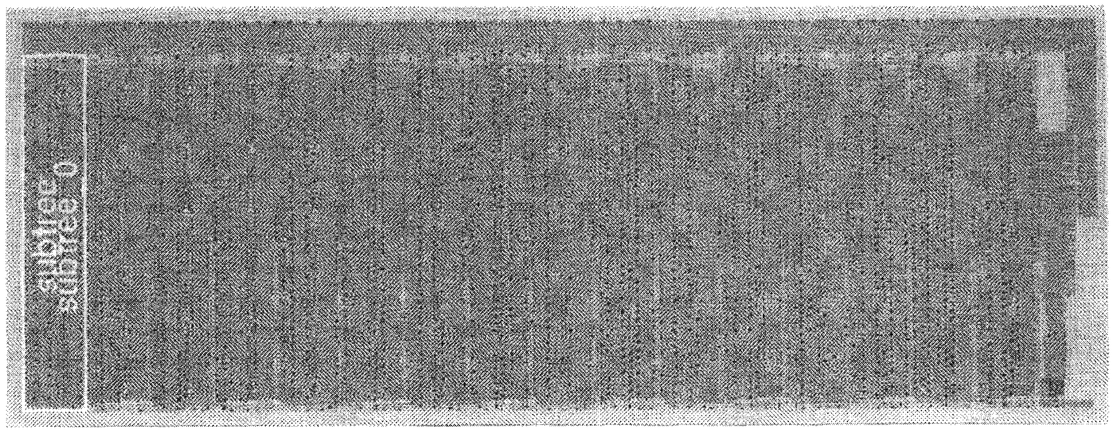


Fig. 10 *Chip layout of 1/2-cut hybrid tree/linear array architecture*

specifications for the real-time processing of the 2DFS algorithm for videoconferencing and videophony applications, and the 1DFS algorithm for the NTSC and HDTV applications. The characteristics of the chip are summarised in Table 2.

Table 2: Characteristics of motion estimation chip

Technology	0.8µm CMOS
Chip size	12.0 × 4.3mm
Chip clock	30MHz
Number of pads	97

7 Conclusions

We have proposed a family of hybrid tree/linear architectures and their corresponding memory structures for the 1DFS and 2DFS block-matching motion estimation algorithms. Targeting at the joint optimisation of execution latency and input bandwidth, we have shown that by combining the tree technique and the systolic mapping method the proposed architectures potentially provided a superior design in reducing implementation costs in terms of the memory system's complexity, I/O pin count, and the control overhead in buffering/arranging input data. To allow a flexible internal and external data access we have designed an effective memory structure using the memory interleaving and the on-chip buffer schemes. Furthermore, we presented the tree-cut techniques that enabled an efficient time-sharing of the hardware and simultaneously reduced input bandwidth. Finally, a single-chip circuit implementation using the 1/2-cut technique demonstrated the versatility and effectiveness of our proposed hybrid architectures.

8 References

1 CCITT Study Group XV: 'Draft revision of recommendation H.261 – video codec for audio visual services at px64kbps'. Temporary document 5-E, July 1990

2 ITU-T Recommendation H.263: 'Video coding for low bitrate communications'. (Draft), July 1995

3 LEGALL, D.J.: 'MPEG: A video compression standard for multimedia applications', *Commun. ACM*, 1991, **34**, pp. 46–58

4 ISO/IEC 1172-2: 'Information technology—coding of moving picture and associated audio for digital storage media at up to about 1.5 Mbit/s: part 2 video'. August 1993

5 ARAVIND, R.: 'Image and video coding standards', *AT&T Tech. J.*, 1993, **72**, pp. 67–89.

6 SCHAFER, R., and SIKORA, T.: 'Digital video coding standards and their role in video communications', *Proc. IEEE*, 1995, **83**, pp. 907–924

7 YANG, K.-M., SUN, M.-T., and WU, L.: 'A family of VLSI designs for the motion compensation block-matching algorithm', *IEEE Trans.*, 1989, **CAS-36**, pp. 1317–1325

8 VOS, L., and STEGHERR, M.: 'Parameterizable VLSI architectures for the full-search block-matching algorithm', *IEEE Trans.*, 1989, **CAS-36**, pp. 1309–1316

9 KOMAREK, T., and PIRSCH, P.: 'Array architectures for block-matching algorithms', *IEEE Trans.*, 1989, **CAS-36**, pp. 1301–1308

10 RUETZ, P.A.: 'A high-performance full-motion video compression chip set', *IEEE Trans. Circuits Syst. Video Technol.*, 1992, **2**, pp. 111–122

11 ISHIHARA, K.: 'A half-pel precision MPEG2 motion-estimation processor with concurrent three-vector search'. ISSCC digest of technical papers, 1995, pp. 286–287

12 JEHNG, Y.-S., CHEN, L.-G., and CHIUEH, T.-Z.: 'An efficient and simple VLSI tree architecture for motion estimation algorithms', *IEEE Trans. Circuits Syst. Video Technol.*, 1993, **41**, pp. 889–899.

13 JAIN, J.R., and JAIN, A.K.: 'Displacement estimation and its application in interframe image coding', *IEEE Trans.*, 1981, **COM-29**, pp. 1799–1808

14 KOGA, J.: 'Motion compensated interframe coding for video conferencing'. Proceedings of National Telecommunications conference, 1981, pp. G5.3.1–5.3.5

15 SRINIVASAN, R., and RAO, K.R.: 'Motion predictive interframe coding', *IEEE Trans.*, 1985, **COM-33**, pp. 1011–1015

16 LI, R., and LIOU, M.L.: 'A new three-step search algorithm for block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 1994, **4**, pp. 438–442

17 JONG, H.-M., CHEN, L.-G., and CHIUEH, T.-D.: 'Accuracy improvement cost reduction of 3-step search block matching algorithm for video coding', *IEEE Trans. Circuits Syst. Video Technol.*, 1994, **4**, pp. 88–90

18 CHEN, M.C., and WILLSON, A.N.: 'A high accuracy predictive logarithmic motion estimation algorithm for video coding'. Proceedings of international symposium on *Circuits and systems*, May 1995, Vol. 1, pp. 617–620

19 CHEN, M.-J., and CHEN, L.-G.: 'One-dimensional full search motion estimation algorithm for video coding', *IEEE Trans. Circuits Syst. Video Technol.*, 1994, **4**, pp. 504–509

20 KUNG, S.Y.: 'VLSI array processors' (Prentice Hall, Englewood Cliffs, NJ, 1988)