Proceedings of the 5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems, Hangzhou, China, April 16-18, 2006 (pp34-38)

Testable Design Techniques for Variable Block Size Motion Estimator Used in H.264/AVC

Po-Yu Yeh, Bo-Yuan Yeh, In-Yi Cheng, Sy-Yen Kuo, and Shyue-Kung Lu***

Dep. of Electrical Engineering, National Taiwan University, Taipei, Taiwan
Dep. of Information Management, National Taipei University of Science and Technology*
Dep. of Electronic Engineering, Fu Jen Catholic University, Taipei, Taiwan**

Abstract: -In this paper, testable design techniques for variable block size motion estimators used in H. 264/AVC are proposed. The whole motion estimator can be viewed as an iterative logic array (ILA) consisting of basic cells (modules). Design-for-testability techniques are applied for the cell (module) function such that the M-testability conditions proposed in previous works can be met for the motion estimation array. The goal of the DFT techniques is to make the cell (module) function bijective. The M-testability conditions guarantee 100% single-cell (module)-fault testability with a minimum number of test patterns. The hardware overhead and the number of test patterns are 4.22% and 128, respectively.

Index Terms—variable block size, M-testable, design for testability, motion estimation, H.264

1 Introduction

Motion estimation (ME) techniques are used to remove temporal redundancy for video coding systems [1, 2, 3]. The computation complexity of ME is from 50% to 90% for a typical video coding system. Moreover, in H.264/AVC, variable block size motion estimation is adopted to further increase the coding efficiency. Motion estimation is the most important part of video compression coding algorithm. It can reduce most redundant information of video sequence, but ME will lead to huge computations due to a large number of block matching and SADs (Sum of Absolutely Difference). The ME of H.264/AVC is much complex than that of MPEG1/2/4. However, in average, The H.264/AVC compressed video bitstream is only 50% of that compressed by MPEG-4 at the same quality. H. 264/AVC divides one 16 × 16 macro block (MB) into kinds of sub-macro-blocks (sub-MBs), result in more precise prediction and higher compression efficiency. There are 41 SADs (Sum of Absolute Difference) produced in a single 16 × 16 MB-matching, unlike previous standards, only the last one (the SAD of whole MB) should be calculated. Therefore, the H.264/AVC ME circuit is divided into several small and piecemeal circuits. This will complicate the complexity of DFT (design for testability) circuits.

In general, high-speed implementation of the ME algorithms is required. There are several architectures [1-3] proposed for motion estimation. One of the most popular architectures is the iterative logic arrays due to their locality and regularity. The computation parallelism of each processing element can achieve very high throughput.

However, integrating a large number of processing elements on a single chip results in the increase in the logic-per-pin ratio, which drastically reduces the controllability and observability of the logic on the chip. Consequently, testing such highly complex and dense circuits becomes very difficult and expensive [4]. Therefore, in this paper, testable design techniques for variable block size motion estimators are proposed. The previous M-testability [5] conditions can be applied for the motion estimation array if some design-for-testability techniques are used to make the basic cell (module) function bijective. The M-testability conditions guarantee 100% single-cell (module)-fault testability with a minimum number of test patterns. The hardware overhead and the number of test patterns are 4.22% and 128, respectively.

The organization of this paper is described as follows. Section 2 introduces the all-binary ME algorithm and its architecture. Section 3 reviews the M-testability conditions and some definitions are given. Section 4 proposes the testable ME designs. Analyzed results are shown in Section 5. Finally, some conclusions are given in Section 6.

2 All-Binary ME Algorithm

Fig. 1 shows the motion estimation between successive frames. According to the all-binary ME method [6], the features (*eFrame*) of each frame are extracted. This method will lead to huge reduction of arithmetic logics without almost negligible PSNR drop. The ME algorithm is based on the block matching of 16 × 16 macroblocks (MBs). Therefore, the eFrame should be divided into several eMBs with the size of 16 × 16.

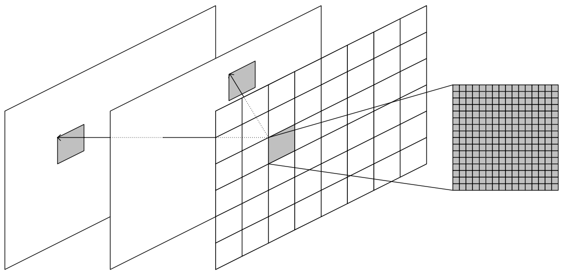

Fig. 1: The extracted eFrame which contains only 1-bit information per pixel from the original frame. The eFrame is divided into 16 × 16 eMBs.

The block matching is performed by calculating the *absolute difference* between the current and reference eMBs *(AD-eMB)*. The AD-eMB can be obtained by simply performing 16 × 16 bitwise XOR operations (Fig. 2). In order to increase the compression efficiency, the sub-block with variable block sizes (VBS) is adopted in H.264/AVC standard. There are 41 VBS blocks for each eMB (Fig. 3). We can use the absolute differences of the smallest sub-blocks to derive the SADs of larger sub-blocks. According to the VBS blocks, H.264/AVC encoder can choose the lowest SAD to achieve the highest compression ration.

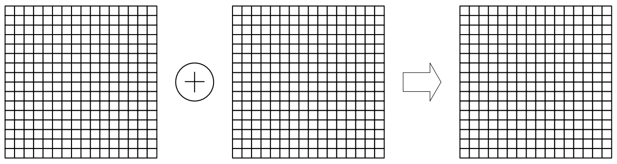

Fig. 2: The derivation of the AD-eMB.

The architecture for the variable block size ME is shown in Fig. 4. In this figure, *p+1* parallel *block matching modules* (BMM) are used to perform motion estimation. Each BMM also includes one 16 × 16-bit *current eMB register buffer (Cur-eMB)* and one *16* × (*16+p*)-bit *reference eMB register buffer (Ref-eMB)*.

A single BMM performs block matching for a 16 × 16 eMB and output 41 SADs. The number of BMMs can be increased until the sufficient throughput is achieved. The internal architecture of a BMM is shown in Fig. 5. It consists of 16 *SAD4×4 modules* and 25 *AccSAD modules* for the calculation of the SADs of 16 4 × 4 sub-blocks and 25 accumulative SADs, respectively.

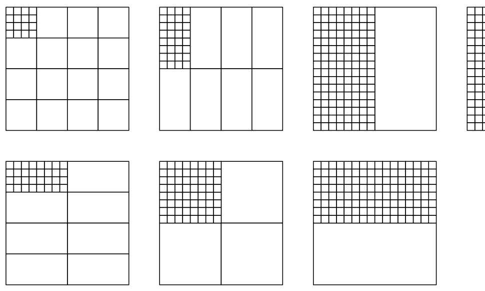

Fig. 3: There are 41 VBS blocks in one eMB.

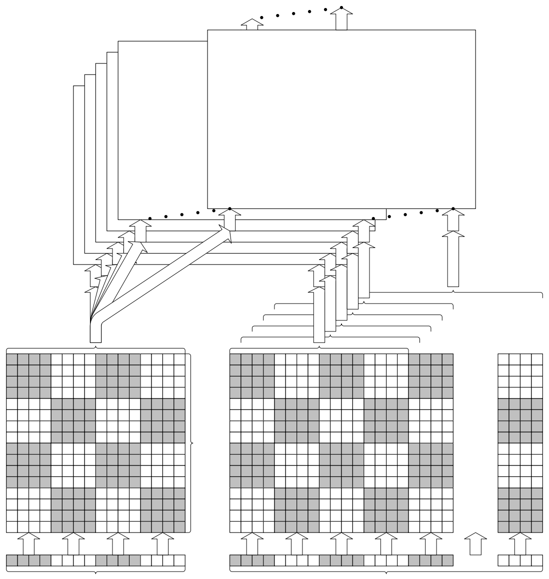

Fig. 4: The parallel architecture of H.264/AVC motion estimation.

3. Review of M-Testability and Definitions

In order to ease our discussion, some definitions are defined first. They are also used in [5].

Definition: A *cell* in an ILA with function f is a combinational machine (Σ, Δ, f), where $f\!: \Sigma \rightarrow \Delta$ is the cell function, and $\Sigma = \Delta = \{0, 1\}^w$, w denotes the word length of a cell. An ILA is an array of cells.

Definition: We say that the function f of a cell is *bijective* when $\forall \theta_1 \neq \theta_2, f(\theta_1) \neq f(\theta_2), \theta_1, \theta_2 \in \Sigma$.

Reference eFrame#2 Reference eFrame#1 Current eFrame

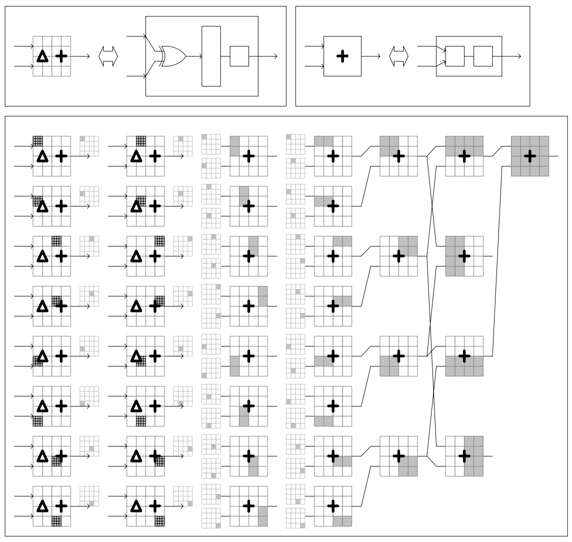

Fig. 5: Internal architecture of a BMM

Definition: We say that an ILA is *M-testable* if it can be tested with 2^w test patterns regardless of the array size.

M-testability conditions can be found in [5]. Thanks to the array architecture and DFT techniques. The M-testable techniques can also be applied to the motion estimator.

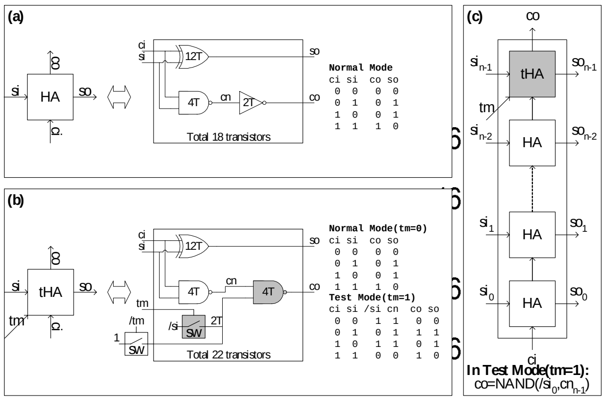

Fig. 6: (a) 1-bit half adder; (b) Testable HA (tHA), and (c) Testable n-bits HA (tnHA).

Fig. 6(a) shows an original HA and its input/output mapping is not bijective. In Fig. 6(b), four transistors are added, and then we can see that the tHA's input and output are 1-to-1 mapped (bijective) and all the gates are fully tested in test mode *(tm = 1)*. The normal function is kept in the normal mode (tm = 0). Although the bijective property is achieved, the hardware overhead is too high to be acceptable *(4/18 = 22.22%)*. In Fig. 6(c), there are ***n*** HAs in the *tnHA* cell, but the highest HA is replaced with the *tHA* cell. Notice that the ***!si₀*** signal is connected from the inside-XOR of the lowest HA. We can all prove the bijective property of *tnHA*. *TnFA, TWFS* and *TWFA* cells can be defined analogously.

4. Testable ME Architectures

4.1 Testing Register Buffers (tRegBuf)

Basically, the Cur-eMB and Ref-eMB register buffers are parallel shifters for storing the extracted features. The inputs and outputs of the register buffers are 1-to-1 mapped in nature (Fig. 4). Therefore, we can first test the tRegBuf by shifting in the test patterns line by line to the bottom lines of all eMBs and then get the output patterns line by line from the top lines of the register buffers. Each current and reference line has ***16+(16+p) = 32+p*** DFFs, and the number of exhaustive input patterns is $\boldsymbol{2^{32+p}}$. This number is too large for practical applications. Fortunately, each vertical line is independent to the other vertical lines. Therefore, we can separate each ***(32+p)***-bit line into several *n*-bit sub-lines. The number of test patterns will be reduced greatly to $\boldsymbol{2^n}$. Obviously, test patterns could be generated by an *n*-bit counter and the output responses are predictable.

3.2 Testable SAD4x4$_{HA}$ Cells

Fig. of a SAD4x4$_{HA}$. It mainly consists of HA cells and XOR gates The inputs c_0~c_{15} and r_0~r_{15} are connected to Cur-eMB and Ref-eMB registers, respectively. The XOR gates compute the 4 × 4 ADs between the two 4 × 4 sub-blocks of Cur-eMB and Ref-eMB. The HA array sums up the 4 × 4 ADs. Each column of the HA array can be regarded as an incremental stage. However, the original circuit needs $\boldsymbol{2^{32}}$ input test patterns. For reducing the test patterns and testability, only four n-bits HA stages should be replaced by *tnHA* cells (*n = 1, 2, 3* and *4*) due to their carry outs that cannot be neglected in the next stage. We can predict that the modified circuit needs at lease five independent inputs due to the five inputs of the last eight stages. We can control the two buses $\{c_{15}...c_0\}$ and $\{r_{15}...r_0\}$ at all-zeros or all-ones independently (there would be $\boldsymbol{2 \times 2 = 4}$ cases), and then multiplexed with $\{ti_4...ti_0\}$ as shown in **Fig.** 7(b).

Lemma: The mapping from $\{ti_4...ti_0\}$ to $\{to_4...to_0\}$ is bijective as shown in **Fig.** 7(b).

Proof: The proof is omitted for simplicity.

Even though each stage of the SAD4x4$_{HA}$ cell has different wordlength, the bijective property can be preserved from the first stage to the 15th stage. That is similar to an ILA. Although the number of SAD4x4$_{HA}$ cells increases rapidly, we can test all SAD4x4$_{HA}$ cells by cascading all of them since the function of the SAD4x4$_{HA}$ cell is bijective.

There will be $\boldsymbol{2^5}$ patterns for each case, so we need a total of $\boldsymbol{2^5 \times 4 = 2^7}$ test patterns for fully testing the ***16*(n+1)*** cells) is quite

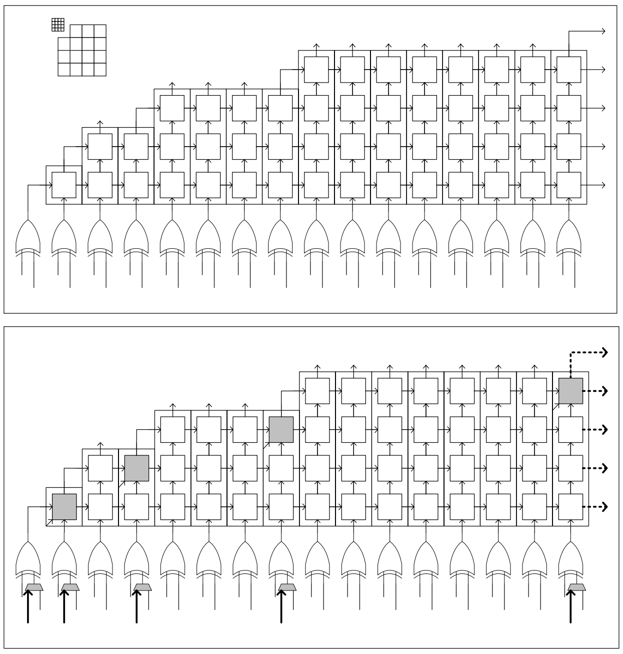
Fig. 7: (a) The original SAD4x4$_{HA}$ cell, and (b) the testable SAD4x4$_{HA}$ cell.

simple because we can just connect each cell's outputs *{to$_4$…to$_0$}* to the next cell's inputs *{ti$_4$…ti$_0$}*.

4.3 Testable AccSAD Cells

In Fig. 5, 25 wFA cells of the original AccSAD part have unbalanced number of input/output SADs and the testability is hard to achieve due to lots of independent and small area logics. Before proposing the testable AccSAD$_{Module}$ network, we should introduce the bijective tAccSAD$_{Module}$ cell first. **Fig.** 8(a) and (b) describe the normal mode and test mode of tAccSAD$_{Module}$ cell, respectively. There are four ***w***-bit twFA and one ***w***-bit twFS cells in the tAccSAD$_{Module}$ cell. The subtraction operation is almost the same as the addition operation in hardware implementation except the extra inverters and the bijective property is also kept. When we input the four ***w***-bit SADs *{Ia, Ib, Ic, Id}* of 4 × 4 sub-blocks in normal mode, the tAccSAD$_{Module}$ cell will output the five ***w***-bit accumulative SADs *{Oa, Ob, Oc, Od, Oe}*. We will show that the mapping from *{Tia, Tib}* to *{Toa, Tob}* is still bijective in test mode.

Oservation: The mapping from *{Tia, Tib}* to *{Toa, Tob}* is bijective.

Proof: The proof is omitted for simplicity.

In fact, if we replace twFS cells with twFA, the tAccSAD$_{Module}$ cell is still bijective. This can be proved by similar manner.

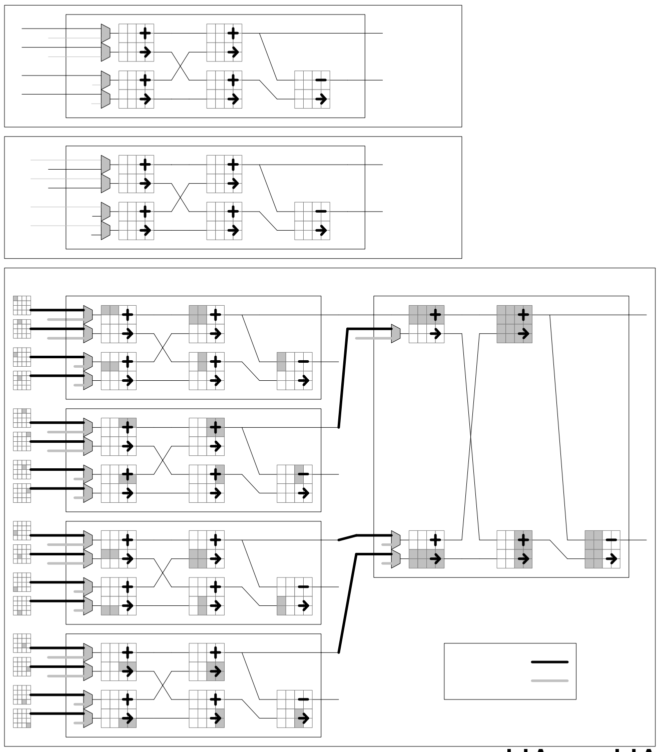
Fig. 8: (a) tAccSAD$_{Module}$ (normal mode); (b) tAccSAD$_{Module}$ (test mode); and (c) Butterfly-like tAccSAD$_{Module}$ network (tAccSAD$_{Module}$-NW).

In Fig. 8(c), the wFA cells are replaced by five tAccSAD$_{Module}$ cells and the bijective property of tAccSAD$_{Module}$ cell is also preserved. In normal mode, the sixteen input SADs of 4 × 4-subblocks will be fed to the first stage and then output eight SADs of 4 × 8-subblocks (as the illustrated gray squares on twFA and twFS cells in the **Fig.** (b), and each square indicates one 4 × 4 sub-block). The second stage will calculate the four SADs for 8 × 8 sub-blocks and four SADs for 8 × 4 subblocks. The third stage will perform four subtractions for the 4 SADs of 8 × 4 subblocks between 8x8 and 8x4 subblocks. The 4th, 5th and 6th stages will calculate the spared SADs for larger VBS blocks in the same manner.

In test mode, all the tAccSAD$_{Module}$ cells of ***p+1*** BMMs are cascaded one after one (total cascaded cells ***c = 5*(p+1)***). According to the bijective property of tAccSAD$_{Module}$ cell, exhaustive input test patterns ($\mathbf{2^{2w}}$ test patterns) can be propagated to the next cell and then all cells can be tested simultaneously. Fortunately, the output responses can be predicted according the input test patterns. If the input test pattern of the first cell is *{A$_0$, B$_0$}*, and assume the expression *{A$_k$, B$_k$}* denotes the output of the ***k*th** cell, where ***k = 1~c***:

$$
\begin{aligned}
\square\{A_1, B_1\} &= \{ (3A_0+2B_0)_{mod}2^w,\ (2A_0+B_0)_{mod}2^w\} \\
\square\{A_2, B_2\} &= \{ (3A_1+2B_1)_{mod}2^w,\ (2A_1+B_1)_{mod}2^w\} \\
&= \{ (13A_0+8B_0)_{mod}2^w,\ (8A_0+5B_0)_{mod}2^w\} \\
\square\{A_4, B_4\} &= \{ (13A_2+8B_2)_{mod}2^w,\ (8A_2+5B_2)_{mod}2^w\} \\
&= \{(233A_0+144B_0)_{mod}2^w,\ (144A_0+89B_0)_{mod}2^w\} \\
&\dots\dots \\
\square\{A_c, B_c\} &= \{ (pA_0+qB_0)_{mod}2^w,\ (rA_0+sB_0)_{mod}2^w\}
\end{aligned}
$$

Therefore, we can calculate the final output constant $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{s}$ in advance with about $\log_2(\boldsymbol{c})$ iterations. Note that $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{s}$ are constants and the hardware implementation can be done by just arithmetic shift and addition operations.

4 Analyzed Results and Comparisons

Table 1 shows the required test patterns for ach of the proposed testable designs. The most important feature is that the test patterns can be generated by a binary counter. Table 1 shows the number of test patterns for the proposed testable designs. Table 2 compares the hardware overhead and the number of test patterns for each design technique. From this table, we can see that the proposed approaches are better than the traditional DFT and ATPG tools.

Table 1: The number of test patterns

@p=8	tRegBuf	tSAD4x4$_{HA}$	tSAD4x4$_{Hybrid}$	tAccSAD$_{Module}$-NW	tAccSAD$_{n\text{-bits}}$-NW
Test Patterns (TP)	2^n = 256@n=8	2^7 = 128	2^8 = 256	2^{2w} 65536@w=8	2^{2n+1}+(w/n)*2^3 528@(w,n)=(8,4) 131088@(w,n)=(16,8)

Table 2: Comparisons with the traditional ATPG method

@p=8	tME-I	tME-II	tME-III	tME-IV	ATPG
Registers	tRegBuf	tRegBuf	tRegBuf	tRegBuf	N/A
SAD4x4-par	tSAD4x4$_{Hybrid}$	tSAD4x4$_{Hybrid}$	tSAD4x4$_{HA}$	tSAD4x4$_{HA}$	N/A
AccSAD-par	tAccSAD$_{n\text{-bits}}$-NW	tAccSAD$_{Module}$-NW	tAccSAD$_{n\text{-bits}}$-NW	tAccSAD$_{Module}$-NW	N/A
Max. TP	528@(w,n)=(8,4)	65536@w=8	528@(w,n)=(8,4)	65536@w=8	>>100k@w=8
Total HO	5.14%@(w,n)=(8,4)	7.54%	5.50%@(w,n)=(8,4)	4.22%	15%~20%

6. Conclusions

In this paper, testable design techniques are proposed for all-binary variable block size motion estimators. As compared with traditional DFT and ATPG tools, the proposed techniques have lower hardware overhead. Moreover, the required test patterns can be greatly reduced. The hardware overhead and the number of test patterns are 4.22% and 128, respectively.

References

[1] C. M. Ou, C. F. Le, and W. J. Hwang, "An efficient VLSI architecture for H.264 variable block size motion estimation," *IEEE Trans. Consumer Electronics*, vol. 51, no. 4, pp. 1291-1299, Nov. 2005.

[2] L. Deng, W. Gao, M. Z. Hu and Z. Z. Ji, "An efficient hardware implementation for motion estimation of AVC standard," *IEEE Trans. Consumer Electronics*, vol. 51, no. 4, pp. 1360-1366, Nov. 2005.

[3] C. C. Yeh, S. Y. Chien, Y. W. Huang, T. C. Chen, T. C. Wang, and L. G. Chen, "Analysis and Architecture Design of Variable Block Size Motion Estimation for H.264/AVC," *IEEE Trans. Circuits and Systems,* vol. pp, no. 99, 2005. (to appear)

[4] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational logic circuits," *IEEE Trans. Computers,* vol. C-31, no. 4, pp. 555-560, June 1982.

[5] C. W. Wu and P. R. Cappello, "Easily Testable Iterative Logic Arrays," *IEEE Trans. Comput.,* vol. 39*,* no. 5, pp. 640-652, May 1990.

[6] J. H. Luo, C. N. Wang and T. Chiang; "A novel all-binary motion estimation (ABME) with optimized hardware architectures," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 8, pp. 700-712, Aug. 2002.

[7] H. Chung and A. Ortega, "Analysis and Testing for Error Tolerant Motion Estimation," in *Proc. DFT 2005*, pp. 514-522, Oct. 2005.

[8] D. Li., and M. Hu, "Built-in self-test design of motion estimation computing array," in *Proc. NEWCAS*, pp. 349-352, June 2004.

[9] W.P. Marnane and W. R. Moore, "Testing a motion estimator array," in Proc. Application-Specific Array Processors. Pp. 734-745, Sept. 1990