IEEE ASIC/SOC Conference, September 2000, Washington, D.C.

DESIGN AND SYNTHESIS OF BUILT-IN SELF-TESTABLE TWO-DIMENSIONAL DISCRETE COSINE TRANSFORM CIRCUITS

Han Bin Kim

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
E-mail: hanbin.kim@eng.sun.com

Dong Sam Ha

Virginia Tech VLSI for Telecommunications (VTVT)
Bradley Dept. of Electrical and Computer Eng.
Virginia Tech, Blacksburg VA 24061
E-mail: ha@vt.edu Web: www.ee.vt.edu/ha

ABSTRACT

We present design of a two-dimensional (2-D) discrete cosine transform (DCT) circuit with built-in self-test (BIST) capability. After modifying an existing fast 2-D DCT algorithm to make it more flexible, we synthesized the data path and the controller using our high-level BIST synthesis tool and incorporated scan design to other modules. Our design achieves high fault coverage at small cost of area overhead and system performance degradation.

I. INTRODUCTION

Built-in self-test (BIST) embeds a test pattern generator (TPG) and a signature register (SR). BIST has become popular for logic circuit testing as well as embedded memory testing.

A high-level BIST synthesis tool generates a data path and a controller in the register-transfer (RT) level with BIST capability by reconfiguring existing registers. A high-level BIST synthesis is effective for testing data intensive circuits, especially DSP (Digital Signal Processing) circuits.

In this paper, we present a design of a two-dimensional (2-D) discrete cosine transform (DCT) circuit with built-in self-test (BIST) capability. The DCT algorithm is widely used to compress images. It is, for example, employed in JPEC and MPEC. Our design procedure is as follows: We modified an existing one-dimensional (1-D) DCT algorithm first and then applied our high-level BIST synthesis tool [1] to generate a data path and a controller with BIST. We designed other modules (such as I/O interface modules and a memory transposition module) manually to implement a 2-D DCT circuit and then incorporated scan design to the modules.

II. BACKGROUND

A. One-Dimensional DCT Algorithm

Since DCT is a separable transformation process, the 2-D DCT can be performed by two separate 1-D DCT processes: a 1-D DCT in the row direction followed by a 1-D DCT in the column direction. Several implementations adopted this approach due to its simple structure and low hardware complexity [2],[3]. However, the approach requires a transposition memory to store and to rearrange the sequence of data for the second 1-D DCT; this degrades of the performance in speed. A different 2-D DCT implementation, presented in [4], eliminates the need for a transposition memory. It improves the performance at the cost of higher circuit complexity.

We used the former approach to implement the 2-D DCT, which performs in three steps: 1-D DCT, and then a memory transposition followed by another 1-D DCT. The formula for the 1-D DCT is given below:

$$X(0) = \sqrt{\frac{1}{N}} \cdot \sum_{n=0}^{N-1} x(n) \tag{1}$$

$$X(k) = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} x(n) \cdot \cos\left(\frac{(2n+1)k\pi}{2N}\right), k=1, 2, \ldots 7 \tag{2}$$

where $x(n)$ represents a pixel value, $X(k)$ represents the transformed value, and N is 8 for the 8×8 block size.

B. Fast DCT Algorithm

Chen et al. proposed a method to improve the original DCT formula described above [5]. The improved one has fewer operations to increase the speed and to reduce the circuit complexity. Chen et al's method is known to be one of the fastest DCT algorithms and is partly employed in our implementation. We describe Chen's algorithm briefly below.

An N-point 1-D DCT algorithm involves $N \times N$ matrix calculations. The $N \times N$ matrix calculations in equations (1) and (2) can be decomposed into the two sets of equations shown in (3) and (4) for N=8 by inspection [2],[5]. The first set contains even indexed terms and the second set the odd indexed terms. The decomposition reduces the number of multiplications from N^2 to $N^2/2$, which reduces 64 multiplications to 32 multiplications for N=8. It is important to note that the decomposition produces the same result as the original formula defined in [1] and [2].

$$\begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix} \qquad (3)$$

$$\begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} x(0)-x(7) \\ x(1)-x(6) \\ x(2)-x(5) \\ x(3)-x(4) \end{bmatrix} \qquad (4)$$

If the two-level multiply-add structure is converted to a multiple-level one by rearranging operations, the number of multiplications is further reduced to 16. Refer to [5] for details of the method.

III. DESIGN OF 2-D DCT

A. Optimization of 1-D DCT Algorithm

The fast DCT algorithm described in the previous section requires a matched inverse DCT (IDCT) algorithm. That is, if data are compressed using the general 1-D DCT algorithm defined in equations (1) and (2), the fast IDCT algorithms cannot be used for decompression. To avoid the incompatibility problem, we modified the algorithm in our design, which increases the number of operations slightly.

The data flow graph shown in Fig. 1 is the modified DCT algorithm employed in our implementation. It has the same structure as Chen's algorithm only for even-indexed terms, which require eight multiplications. The number of multiplications required for odd-indexed terms remains the same (which is 16) as specified in equation (3) and (4). The total number of multiplications necessary for our implementation is 24, whereas it is 16 for Chen's algorithm.

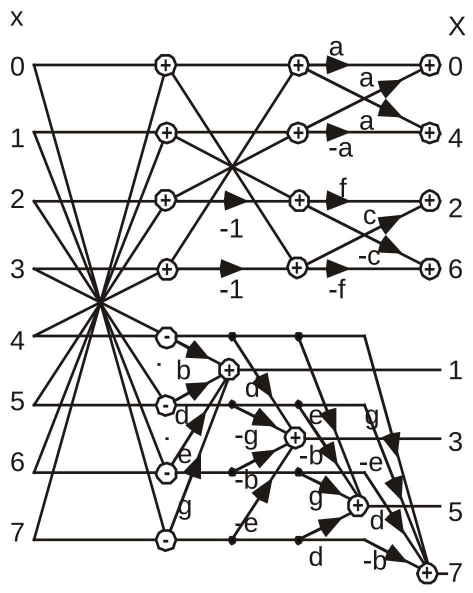

Figure 1: Data flow graph of 1-D DCT algorithm

We applied our high-level BIST synthesis tool called ADVBIST_h to the data flow graph in Fig. 1 and generated a RT-level data path and a controller [1].

B. Structure and Operation of a 2-D DCT Circuit

Fig. 2 shows the block diagram of a 2-D DCT circuit. Each row of an 8×8 image block is loaded serially through DIN port of a serial-to-parallel (ser2par) conversion module, and a 1-D DCT operation is performed for the row. The result is stored in the transposition memory. Upon completely processing the eight rows, 1-D DCT operations are performed column by column in the transposition memory. The result of each column 1-D DCT operation is outputted serially through DOUT port of a parallel-to-serial (par2ser) conversion module.

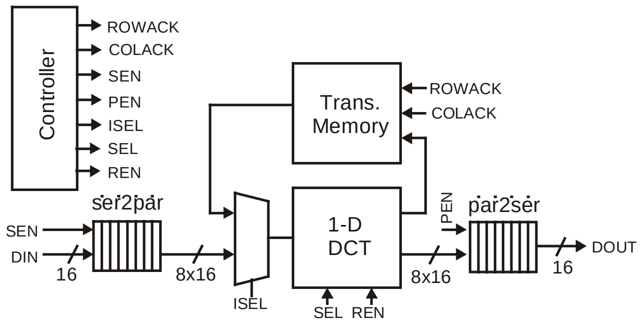

Figure 2: Block diagram of our 2-D DCT circuit

Fig. 3 shows the architecture for our 1-D DCT data path and the controller. The multiplexer-based architecture is employed for the data path, and a finite state machine (FSM) is used for the controller. The system is clocked by single system clock.

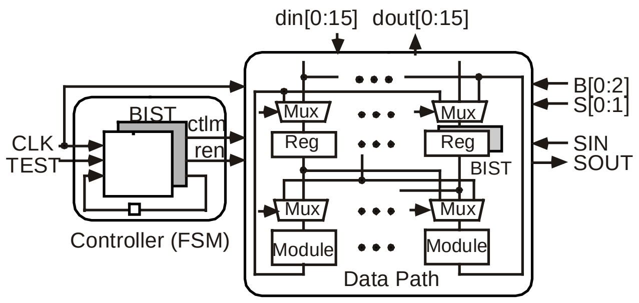

Figure 3: Architecture for our 1-D DCT circuit

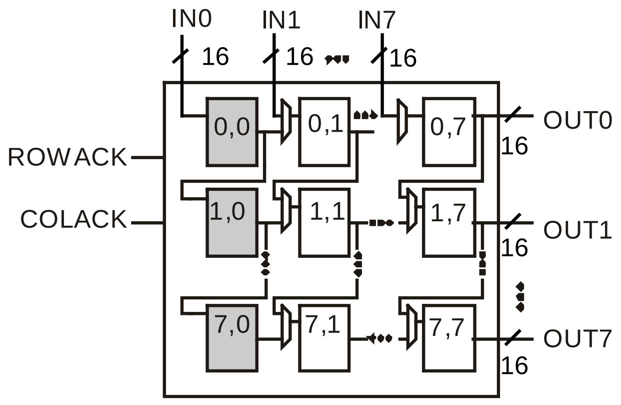

Figure 4: Schematic of the transposition memory

The structure of transposition memory is shown in Fig. 4. A transposition memory consists of an 8×8 register array in which each register is 16-bit long. Data are filled into the array in row-wise from top, and are emptied in column-wise in the right side. Two control signals, ROWACK and COLACK, determine the direction of the data movement.

IV. DESIGN-FOR-TEST TECHNIQUES APPLIED

A. TPGs and SRs

We implemented a TPG as shown in Fig. 5 by modifying a built-in logic block observer (BILBO) [6]. Our TPG has three meaningful functions such that normal operation, scan, and test pattern generation as shown in Table 1. The major difference between our TPG and the original BILBO is that the inputs $Z_{15} – Z_0$ are blocked for our TPG in test pattern generation mode; this is necessary for pseudo-random pattern generation. We used the original BILBO for signature analysis.

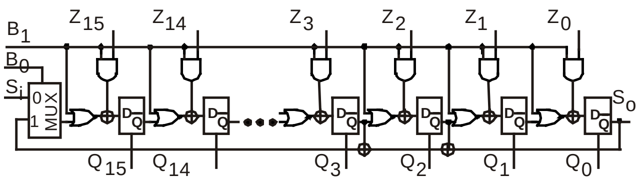

Figure 5: Schematic of modified 16-bit TPG

Table 1: Operating modes of our 16-bit TPG

B0	B1	Modes
1	1	Normal function
0	0	Shift register
1	0	Test pattern generation

B. Testing of the Data Path

We defer description on testing of data path modules to Section V, which also covers experimental results measured from the layout.

C. Testing of the Transposition Memory

We aim to test stuck-at faults of the registers in the transposition memory. Our approach is to march through all-0-patterns to detect stuck-at 1 faults and then all-1-patterns to detect stuck-at 0 faults. After test patterns are scanned into the registers in the rightmost column, a register (i, j) copies from its left neighbor register (i, j-1) on every clock in the memory testing mode; the registers in the rightmost column are copied into scan registers in the data path module.

The operation of the testing process is as follows. First, all-0-patterns are scanned into the registers in the transposition memory module, whose registers are shaded in Fig. 4. Next, eight clocks are applied to the module in the memory testing mode. The patterns march through the array and are captured in the output registers. After switching to the scan mode, the contents of the output registers are scanned out for examination. The scheme achieves 100 % stuck-at fault coverage.

D. Testing of the Controller

The controller circuit consists of a 9-bit counter and an associated combinational logic. The combinational logic receives the content of the counter and generates 66 control signals.

The counter serves as a TPG in our BIST mode. Test responses on the control signals are compacted in space to 32 outputs and then compressed in time using two existing SRs in the data path. Fig. 6 shows the structure of the controller after incorporation of BIST. The XOR gates in the left MISR (Multiple Input Signature Register) compacts test responses in space. The fault coverage reaches 97.0 % for 50 patterns and remains the same after the 50 patterns. Hence, we apply 50 test patterns (generated by the counter) to test the controller.

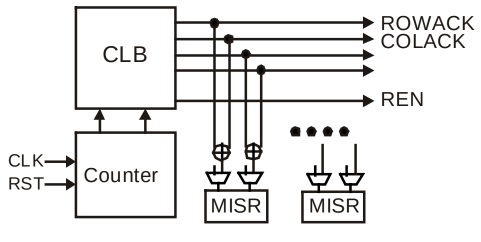

Figure 6: Schematic of the controller

E. Testing of the I/O Conversion Modules.

We apply an all-0-test pattern to primary inputs INS(0:15), which are inputs of the serial-to-parallel converter. Then we apply eight clocks to march the all-0-

pattern through the eight 16-bit registers in the converter. The result is latched into a scan register in the data path module and scanned out for examination. We repeat the same process for the all-1-test pattern. Testing of the parallel-to-serial module is similar. We scan in a test pattern into a scan register in the module and apply 7 clocks. The test response is examined on primary outputs DOUT(0:15). The scheme achieves 100 % stuck-at fault coverage.

F. Overall Test Schedule

The six dedicated control signals, TEST, B[0:2], and S[0:1], are used to control ten different modes of operations. (Refer to Fig. 3.) Four BIST sessions are allocated. to test six arithmetic modules (two adders, two subtractors, and two multipliers) in the data path. Four other test sessions include controller test mode, the transposition memory test mode, serial-to-parallel converter test mode, and parallel-to-serial converter test mode. The remaining two modes are normal mode and the scan mode.

V. EXPERIMENTAL RESULTS

A. Application of High-Level BIST Synthesis

15 control steps were scheduled for the completion of the 1-D DCT operation, and 6 arithmetic modules were assigned for the operations. Our high-level BIST synthesis tool ADVBIST_h was applied in which the number of test sessions was set to 4 [1]. The data path generated by ADVBIST_h was built-in self-testable with four TPGs and two SRs.

B. Testing of Data Path

Fig. 7 shows BIST configurations for testing of the data path. (Only resources that are related to BIST are shown in the figure.) All the BIST registers are connected through a single scan chain to initialize TPGs and to scan out signatures of the test responses captured in SRs. Four BIST test sessions are used for testing six arithmetic modules as described earlier, and the modules and BIST registers are shown in Table 2. For each test session, initial test patterns are scanned in, the target modules are tested, and test responses are scanned out. All related control signals are generated from the controller. The fault coverage of adders and the subtractors reaches 100 % with 40 and 50 test patterns, respectively. The fault coverage of the multipliers reaches 99.0 % for 50 test patterns and increases to 99.65 % for 200 test patterns. We used 150 test patterns for the multipliers; This achieves 99.6 % of fault coverage.

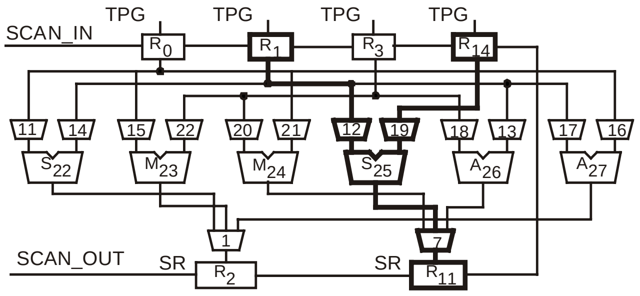

Figure 7: BIST configurations for the testing of DCT data paths

Table 2: Test schedule

Session	Module	TPG	TPG	SR
0	S25	R1	R14	R11
1	S22	R0	R1	R2
2	M23	R0	R3	R2
	M24	R3	R0	R11
3	A26	R3	R1	R11
	A27	R1	R0	R2

C. Logic & Layout Design

The resultant BIST data path and the control unit were described in VHDL and were synthesized to obtain a gate level circuit. The circuit was placed and routed using an HP 0.5 μm CMOS standard cell library with triple metal layers. We also implemented a circuit without BIST in a similar manner to measure the overhead of the BIST design. The characteristics of the two circuits are summarized in Table 3. Fig. 8 shows final layout of 2-D DCT circuit with BIST.

Table 3: Characteristics of two chips

Description	w/o BIST	w/ BIST
Area of the core logic	10.84 mm^2	11.92 mm^2
Max. operable clock rate	32.4 MHz	30.8 MHz
Area overhead	-	9.95 %
Clock speed degradation	-	5.2 %
Equiv. NAND2 gate counts		
of the data path	9926	10956
of the control unit	317	553
of the memory unit	10,030	12451
Total	22,612	27,160

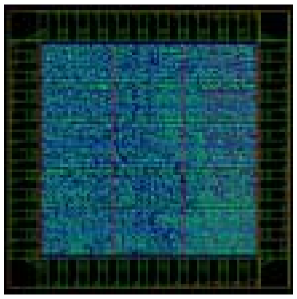

Figure 8: Layout of 2-D DCT circuit with BIST

VI. CONCLUSION

In this paper, we described design of a 2-D DCT circuit. Our high-level BIST synthesis tool was applied to implement the data path and the controller. Scan design was applied to other modules. The area overhead of the DFT including BIST is 9.95%, which is tolerable in industry.

REFERENCES

[1] H.B. Kim and D.S. Ha, “A High-Level BIST Synthesis Method Based on a Region-wise Heuristic for an Integer Linear Programming,” *IEEE International Test Conference*, pp. 903-912, Sept. 1999.

[2] A. Madisetti and A.N. Wilson, Jr., "A 100 MHz 2-D 8x8 DCT/IDCT Processor For HDTV Applications, " *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, pp 158-165, Apr. 1995.

[3] T. Xanthopoulos and A.P. Chandrakasan, “A Low-Power IDCT Macrocell for MPEG-2 MP@ML Exploiting Data Distribution Properties for Minimal Activity,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 693-703, May 1999.

[4] V. Srinivasan and K.J.R. Liu, "VLSI Design of High-Speed Time-Recursive 2-D DCT/IDCT Processor for Video Application", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no 1, pp.87-96, Feb. 1996

[5] W.H. Chen, C.H. Smith, and S.C. Fralick, “A Fast Computational Algorithm for the Discrete Cosine Transform,” *IEEE Trans. Commun*., vol. COM-25, pp. 1004-1009, Sept. 1977.

[6] M. Abramovici, M.A. Breuer and A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE, Jan. 1998.