Architecture for Motion Estimation Computing Arrays with Built-in Self-Detection/Correction

N. Parijatha

Abstract: Built-In Self-Detection/Correction (BISDC) architecture for Motion Estimation Computing Arrays (MECAs) based on biresidue codes, any single error in each processing element in an MECA can be effectively detected and corrected online using the proposed BISD and Built-In Self-Correction circuits. Performance analysis and evaluation demonstrate that the proposed BISDC architecture performs well in error detection and correction with minor area overhead and timing penalty.

Generally, Motion Estimation Computing Array (MECA) performs up to 50% of computations in the entire video coding system, and is typically considered the computationally most important part of video coding systems. For a commercial chip, a video coding system must introduce Design For Testability (DFT), especially in an MECA. The objective of DFT is to increase the ease with which a device can be tested to guarantee high system reliability. Among these techniques, BIST has an obvious advantage in that expensive test equipment is not needed and tests are low cost. Moreover, BIST can generate test simulations and analyze test responses without outside support, making tests and diagnoses of digital systems quick and effective. However, as the circuit complexity and density increases, the BIST approach must detect the presence of faults and specify their locations for subsequent repair. The extended techniques of BIST are Built-In Self-Diagnosis (BISD) and Built-In Self-Repair (BISR). Although BIST and BISR are utilized in many studies, most studies focused on memory testing.

1. Introduction

The new Joint Video Team (JVT) video coding standard has garnered increased attention recently. Generally, Motion Estimation Computing Array (MECA) performs up to 50% of computations in the entire video coding system, and is typically considered the computationally most important part of video coding systems. Thus, integrating the MECA into a System-On-Chip (SOC) design has become increasingly important for video coding applications.

Although advances in VLSI technology allow integration of a large number of processing elements (PEs) in an MECA into an SOC, this increases the logic-per-pin ratio, thereby significantly decreasing the efficiency of chip logic testing. For a commercial chip, a video coding system must introduce Design For Testability (DFT), especially in an MECA.

N.Parijatha is with Department of ECE, Anurag Group Of Institutions Formerly CVSR College Of Engineering, A.P, India, Email: pooja26.nathi@gmail.com

The objective of DFT is to increase the ease with which a device can be tested to guarantee high system reliability. Many DFT approaches have been developed. These approaches can be divided into three categories: ad hoc (problem oriented), structured, and Built-In Self-Test (BIST). Among these techniques, BIST has an obvious advantage in that expensive test equipment is not needed and tests are low cost.

The Motion Estimation Computing Array is used in Video Encoding applications to calculate the best motion between the current frames and reference frames. The MECA is in decoding application occupies large amount of area and timing penalty. By introducing the concept of Built-In Self Test technique the area overhead is increased in less amount of area.

The Self-Detection and Self Correction Operations diagram is shown in Fig 1.2 and their operations are simply described as follows. First, the input data of Cur. Pixel and Ref.pixel for a specific PE in the MECA are sent to the Test Code Generator (TCG) to generate the corresponding test codes. Second, the test codes from the TCG and output data from the specific PE are detected and verified in Detector And Selector (DAS) circuits to determine whether the specific PE has an error. In other words, the self-detection capability uses the detector circuit in DAS. Third, the selector circuit in DAS delivers the error signal to SAC for error correction. Finally, the error correction data from SAC, or error-free data from the selector circuit in DAS, are passed to the next specific PE for subsequent testing.

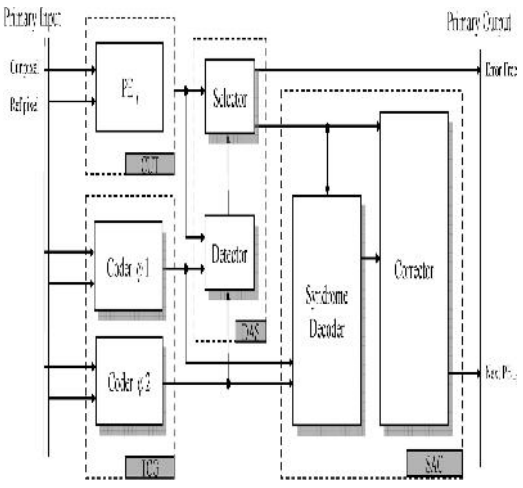

Fig 1.1 Block Diagram of Built-in Self Detection&

Processing element

Processing element calculates the sum of absolute differences between current pixels and reference pixels. The

16 pixels absolute difference is given to adder unit to perform Sum of the Absolute Difference (SAD).

Coder

The following definitions are based on the Biresidue codes, are applied to verify the feasibility of the two coders in the (TCG) Test Code Generator.

Definition1:

$$|N_1+N_2|_\Psi = ||N_1|_\Psi, |N_2|_\Psi|_\Psi$$

Definition2: Let $N_j = n_1+n_2+\ldots\ldots\ldots+n_j$. then

$$|N_j|_\Psi = ||n_1|_\Psi+|n_2|_\Psi+ \ldots\ldots\ldots\ldots+|n_j|_\Psi|_\Psi$$

In the project we are using 2 Coder modules because bi residue Codes are used for calculation of SAD. And the ϕ1 and ϕ2 values are selected by satisfying the conditions $A=2^a-1$ and $B=2^b-1$ such that (Greatest Common Divisor) GCD (a, b) =1.

Detector

Detector module will detect whether there is an error in output of PE i.e. SAD. Here the Error (e) is calculated. The output of PE and theoretically calculated SAD will be subtracted which is given as

$$e = SAD' - SAD$$

Selector

Selector takes the output of the processing element as an input. Another input to the selector is the output of the detector. If the detector block detects any error in the PE output then the selector block will give the PE output to syndrome decoder to detect in which bit position there is an error and also to the corrector block to correct the single bit error.

Syndrome Decoder

This module decodes the syndrome values which specify the error in the bit position of SAD. Syndromes can be expressed as

$$(S_{\Psi 1}, S_{\Psi 2}) = (|N'_j-X|_{\Psi 1}, |N'_j-Y|_{\Psi 2}) = (|e|_{\Psi 1}, |e|_{\Psi 2})$$

Corrector

Input to the corrector module is the output of the selector module which is SAD that needs to be corrected. The bit position which needs correction is specified by the syndrome decoder. The corrector architecture consists of LUT and 12 multiplexers.

2. Introduction to BIST

With recent advances in semiconductor manufacturing technology, the production and usage of Very-Large-Scale Integration (VLSI) circuits has run into a variety of testing challenges during wafer probe, wafer sort, pre-ship screening, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance, repair test, etc. Traditional test techniques that use Automatic Test Pattern Generation (ATPG) software to target single faults for digital circuit testing have become quite expensive and can no longer provide sufficiently high fault coverage for deep submicron or nanometer designs from the chip level to the board and system levels. One approach to alleviate these testing problems is to incorporate Built-In Self Test (BIST) features into a digital circuit at the design stage. With logic BIST, circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides.There are two general categories of BIST techniques for testing random logic:

(1) Online BIST.

(2) Offline BIST.

Online BIST is performed when the functional circuitry is in normal operational Mode. It can be done either concurrently or no concurrently. In concurrent online BIST, testing is conducted simultaneously during normal functional operation. The functional circuitry is usually implemented with coding techniques or with duplication and comparison. When an intermittent or transient error is detected, the system will correct the error on the spot, rollback to its previously stored system states, and repeat the operation, or generate an interrupt signal for repeated failures. In no concurrent online BIST, testing is performed when the functional circuitry is in idle mode. This is often accomplished by executing diagnosis software routines (macrocode) or diagnosis firmware routines (microcode). The test process can be interrupted at any time so that normal operation can resume.

Offline BIST is performed when the functional circuitry is not in normal mode. This technique does not detect any real-time errors but is widely used in the industry.

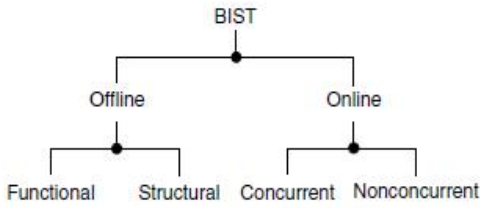

Fig 2.1Types of BIST

Logic BIST techniques for testing the functional circuitry at the system, board, or chip level to ensure product quality.

Functional offline BIST performs a test based on the functional specification of the functional circuitry and often employs a functional or high-level fault model. Normally such a test is implemented as diagnostic software or firmware. Structural offline BIST performs a test based

on the structure of the functional circuitry. There are two general classes of structural offline BIST techniques:

1. External BIST: In which test pattern generation and output response analysis is done by circuitry that is separate from the functional circuitry being tested.

2. Internal BIST: In which the functional storage elements are converted into test pattern generators and output response analyzers. Some external BIST schemes test sequential logic directly by applying test patterns at the inputs and analyzing the responses at its outputs. Such techniques are often used for board-level and system level self-test. The BIST schemes discussed here all assume that the functional storage elements of the circuit are converted into a scan chain or multiple scan chains for combinational circuit testing. Such schemes are much more common than those that involve sequential circuit testing and are the primary focus of this chapter.

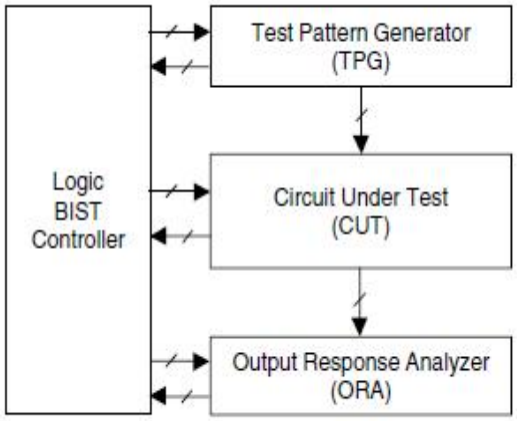

Fig 2.2 Basic structure of BIST

The Test Pattern Generator (TPG) automatically generates test patterns for application to the inputs of the Circuit Under Test (CUT). The Output Response Analyzer (ORA) automatically compacts the output responses of the CUT into a signature. Specific BIST timing control signals, including scan enable signals and clocks, are generated by the logic BIST controller for coordinating the BIST operation among the TPG, CUT, and ORA. The logic BIST controller provides a pass/fail indication once the BIST operation is complete.

It includes comparison logic to compare the final signature with an embedded golden signature, and often comprises diagnostic logic for fault diagnosis. As compaction is commonly used for output response analysis, it is required that all storage elements in the TPG. A typical logic BIST system consist CUT and ORA. These should be initialized to known states prior to self-test, and no unknown (X) values be allowed to propagate from the CUT to the ORA. In other words, the CUT must comply with additional BIST-specific design rules.

There are a number of advantages to using the structural offline BIST technique rather than conventional scan:

1. BIST can be made to effectively test and report the existence of errors on the board or system and provide diagnostic information as required, it is always available to run the test and does not require the presence of an external tester.

2. Because BIST implements most of the tester functions on-chip, the origin of errors can be easily traced back to the chip, some defects are detected without being modeled by software. N-detect, a method for detecting a fault N time's, is done automatically. At-speed testing, which is inherent in BIST, can be used to detect many delay faults.

3. Test costs are reduced due to reduced test time, tester memory requirements, or tester investment costs, as most of the tester functions reside on-chip itself. However, there are also disadvantages associated with this approach. More stringent BIST-specific design rules are required to deal with unknown (X) sources originating from analog blocks, memories, non-scan storage elements, asynchronous set/reset signals, tri-state buses, false paths, and multiple-cycle paths, to name a few. Also, because pseudo-random patterns are mostly used for BIST pattern generation, additional test points (including control points and observation points) may have to be added to improve the circuit's fault coverage.

While BIST-specific design rules are required and the BIST fault coverage may be lower than that using scan, BIST does eliminate the expensive process of software test pattern generation and the huge test data volume necessary to store the output responses for comparison. More importantly, a circuit embedded with BIST circuitry can be easily tested after being integrated into a system. Periodic in-system self-test, even using test patterns with less than perfect fault coverage, can diagnose problems down to the level where the BIST circuitry is embedded. This allows system repair to become trivial and economical.

Test Pattern Generation

For logic BIST applications, in-circuit TPGs constructed from Linear Feedback Shift Registers (LFSRs) are most commonly used to generate test patterns or test sequences for exhaustive testing, pseudo-random testing, and pseudo-exhaustive testing. Exhaustive testing always guarantees 100% single-stuck and multiple stuck fault coverage. This technique requires all possible 2^n test patterns to be applied to an n-input combinational Circuit Under Test (CUT), which can take too long for combinational circuits where n is huge, therefore, pseudo-random testing is often used for generating a subset of the 2^n test patterns and uses fault simulation to calculate the exact fault coverage. In some cases, this might become quite time consuming, if not infeasible. In order to eliminate the need for fault simulation while at the same time maintaining

100% single-stuck fault coverage, we can use pseudo-exhaustive testing to generate 2w or 2k−1 test patterns, where w<k<n, when each output of the n-input combinational CUT at most depends on w inputs. For testing delay faults, hazards must also be taken into consideration.

Standard LFSR

N-stage standard LFSR consists of n- number of D flip-flops and a selected number of exclusive-OR (XOR) gates. Because XOR gates are placed on the external feedback path, the standard LFSR is also referred to as an external-XOR LFSR.

Modular LFSR

Similarly, an n-stage modular LFSR with each XOR gate placed between two adjacent D flip-flops.

Exhaustive Testing

Exhaustive Testing requires applying 2^n exhaustive patterns to an n-input combinational Circuit Under Test (CUT). Any binary counter can be used as an Exhaustive Pattern Generator (EPG) for this purpose, however, because the order of generation of the inputs is not important, it may be more efficient to use an autonomous, maximum-length LFSR that can cycle through all states. To do this, it is necessary to modify the LFSR so that the all-zero state is included. A general procedure for constructing modified (maximum-length) LFSRs that include the all-zero state is given in. These modified LFSRs are called complete LFSRs (CFSRs).

Pseudo-Random Testing

One approach that can reduce test length but sacrifices the circuit fault coverage uses a Pseudo-Random Pattern Generator (PRPG) for generating a pseudorandom sequence of test patterns. Pseudo-Random Testing has the advantage of being applicable to both sequential and combinational circuits, however, there are difficulties in determining the required test length and fault coverage. Schemes to estimate the random test length required to achieve a certain level of fault detection or obtain a certain defect level can be found.

3. Motion Estimation Computing Array (MECA)

Generally, Motion Estimation Computing Array (MECA) performs up to 50% of computations in the entire video coding system. In VCS, Video data needs to be compressed before storage and transmission, complex algorithms are required to eliminate the redundancy, extracting the redundant information.

Motion Estimation (ME) is the process of creating motion vectors to track the motion of objects within video footage. It is an essential part of many compression standards and is a crucial component of the H.264 video compression standard. In particular ME can consist of over 40% of the total computation.

Motion Estimation is the technique of finding a suitable Motion Vector (MV) that best describes the movement of a set of pixels from its original position within one frame to its new positions in the subsequent frame. Encoding just the motion vector for the set of pixels requires significantly less bits than what is required to encode the entire set of pixels, while still retaining enough information to reproduce the original video sequence.

Digital Video Compression

Video compression is achieved on two separate fronts by eliminating spatial redundancies and temporal redundancies from video signals. Removing spatial redundancies involves the task of removing video information that is consistently repeated within certain areas of a single frame. For example a frame shot of a blue sky will have a consistent shade of blue across the entire frame. This information can be compressed through the use of various discreet cosine transformations that map a given image in terms of its light or color intensities.

This paves the way for spatial compression by only capturing the distinct intensities, instead of the spread of intensities over the entire frame. Since compression through removing spatial redundancies does not involve the use of motion estimation. Compression through the removal of temporal redundancies involves compressing information that is repeated over a given sequence of frames. Consequently, the motion estimation process is the process of deriving a suitable Motion Vector (MV) that best describes the spatial movement of objects from one frame to the next.

Motion compensation is a key process in temporal video compression, which eliminates temporal redundancies found in video. Temporal redundancy occurs when an identical set of pixels exist across multiple video frames. This is often caused when an object appears in a set of video footages, which might correspond to several thousands of frames. From frame to frame the object might change its position due to motion resulting from camera pans or zooming (global motion), the active motion of the object itself (translational motion), or a combination of both global and translational motion.

Block Matching Motion Estimation

Several different algorithms derived from various theories, including object-oriented tracking, exist to perform motion estimation. Among them, one of the most popular algorithms is the Block Matching Motion Estimation (BME) algorithm. BME treats a frame as being composed of many individual sub-frame blocks, known as macro Blocks. Motion Vectors are then used to encode the motion of the Macro Blocks through frames of video via a frame by frame matching process.

When a frame is brought into the encoder for compression, it is referred to as the current frame. It is the goal of the BME unit to describe the motion of the Macro Blocks within the current frame relative to a set of reference frames. The reference frames may be previous or future

frames relative to the current frame. Each reference frame is also divided into a set of sub-frame blocks, which are equal to the size of the macro Blocks. These blocks are referred to as reference Blocks.

The BME algorithm will scan several candidate reference Blocks within a reference frame to find the best match to a macro Block. Once the best reference Block is found a motion vector is then calculated to record the spatial displacement of the macro Block relative to the matching reference Block, as shown in Figure 3.1.

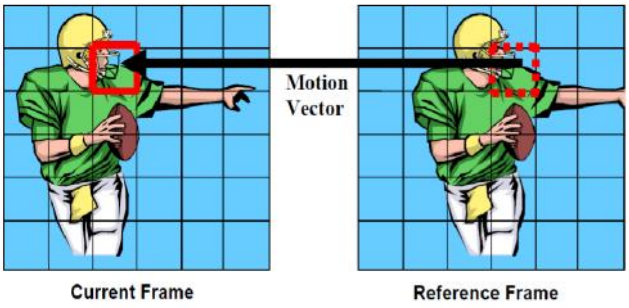

Fig 3.1 Block Matching between Current & reference frames

Search Windows

When searching a reference frame for possible macro Block matches, the entire reference frame is not searched. Instead the search is restricted within a search window. Search windows in most H.264 implementations have a size of 48-pixel (rows) x 63-pixel (columns). In this, we use the same 48x63 search window size. This window consists of a vertical search range of [-16, +16] and a horizontal search range of [-24, +23] pixels as illustrated in Figure 3.3.

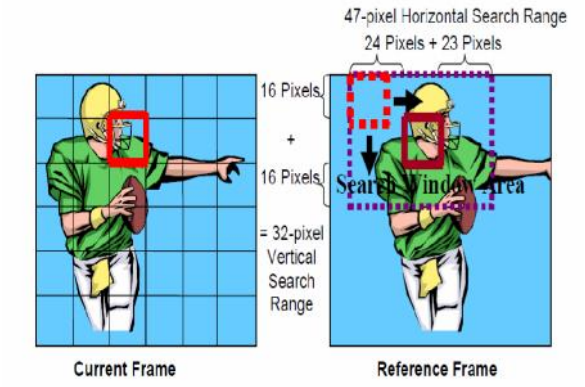

Fig 3.2 Search Window size Definition

In the figure 3.2, the dashed large rectangle in the reference frame represents the 48x63 search window area. The dashed square in the top left corner of the search window represents the first of the 1584 possible candidate 16x16 reference Blocks. Each subsequent reference Block is offset by either one pixel row or one pixel column from its predecessor while the entire search window area is covered by the overlapping candidate reference Blocks. Note that the original 16x16 macro Block is positioned at the centre of the search window. In order to compare it to every candidate reference Block within the search window, the macro Block has a maximum displacement of 24 pixels to the left, 23 pixels to the right, 16 pixels up, and 16 pixels down from its original position – resulting in a horizontal search range of [-24, +23] and a vertical search range of [-16, +16].

4. Modules

PE Array Architecture

PE (Processing Element) is a module that calculates the absolute difference between the pixel of the reference block and the pixel of the current block. Figure 4.1 shows the architecture of PE Array 4x4.To enable the reference data shifting to top, bottom, right or left in PE Array 4x4, each PE is connected to the PE of top, bottom, right or left one. This structure generates SAD 4x1 by accumulating the absolute difference of each PE. Furthermore, SAD 4x4 is generated by accumulating generated SAD 4x1.

However, because long delay will be induced by the multiple arithmetic accumulating modules which generates SAD 4x1 and SAD 4x4, registers are inserted to improve maximum clock frequency traditional PE can shift pixel data in one direction by connecting PEs of the same direction. However, the proposed PE can shift reference pixel data to top, bottom, right or left. The current pixel data is always shifting into the PE Array from top direction. The PE module is designed to be able to shift to top, bottom, right or left.

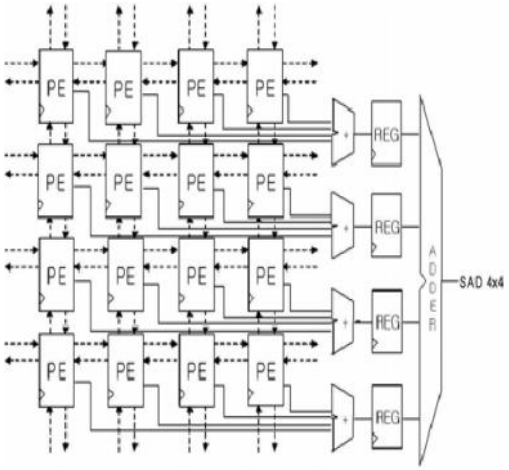

Fig 4.1 PE Array 4x4 architecture

As shown in Fig 4.1 Reference pixel data need input from four directions as well as output to four directions. Therefore, four input ports and four output ports are prepared in each PE. The connection of PEs is shown in Fig 4.2. In this each PE is connected with surrounding PEs "from top" connects to "to bottom", "from bottom" connects to "to top", "from left" connects to "to right", and "from right" connects to "to left". An example when "from top" is selected by the multiplexer.The reference data from the output port "to bottom" of upper PEs is shifted to bottom PEs and the search position is shifted. In this way, each I/O port of PE enables the shift of the reference pixel data by selecting the input of reference pixel data using multiplexer.

SAD Modules

There are 16 SAD (Sum of the Absolute Difference) modules in the architecture, where each one is in charge of the SAD computation of one primitive 4x4 sub-block in parallel, as shown in figure 4.2. In the SAD module, there are 16 absolute difference computing unit processing the 16 pair of pixels in parallel, and then the 16 absolute residues are fed into the adder unit to get one 4x4 SAD.

Adding the 16 absolute residues to obtain one 4x4 SAD is implemented by employing multilevel 3-2 compressors, which can make the SAD modules attain low latency and small area.

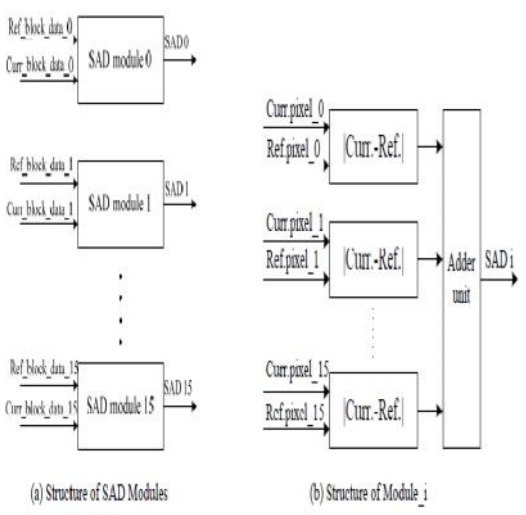

Fig 4.2 Basic Structure of SAD Modules

Processing element calculates the sum of absolute differences between current pixels and reference pixels. The 16 pixels absolute difference is given to adder unit to perform SAD.

Coder

The Coder generates residue code, two coders are used to generate bi residue codes. Bi- residue code is calculated using modulus operator. The modulus values are selected in such a way that it satisfies the following conditions. The block diagram of the coder module is shown in the fig 4.9, the total module is combined with modulus, multiplexer, adder and register. Current pixel and reference pixels are applied to the modulus. The output is the modulus of the respective pixels.

In the project we are using 2 coder modules because bi residue codes are used for calculation of SAD. And the ϕ1 and ϕ2 values are selected by satisfying the conditions $A=2^a-1$ and $B=2^b-1$ such that GCD (a, b) =1 Where A, B are Consider integer N coded as a triple (N, $|N|_A$, $|N|_B$). a=3,b=4 hence A=7 and B=15 respectively.

Multiplexer is used here to a select either feed backed result for addition and it will go as output to the next adder module. Modulus values and the outputs of the both multiplexers are fed to the adder. Adder combines the both inputs and the result is the addition of the inputs. And again the result of the adder will do for modulus and send to the register. Register used to store the value. These will repeat for six clock cycles. The diagram 4.5 shown is for single coder.

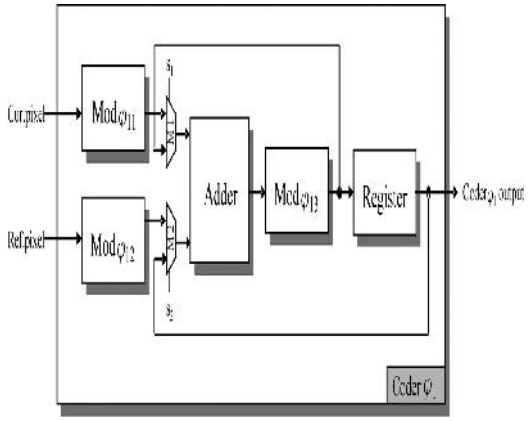

Fig 4.3 Coder Block Diagram

In the above sections a 6 tap FIR filter is implemented in 4 different structures which are Direct form, Transposed form, Symmetric form, Distributed arithmetic based FIR filters. The functionality of these structures was verified and found same for all. The difference in implementation results only performance wise difference but filtering action will be same.

By using arithmetic building blocks, various structures of FIR filters are implemented In VHDL.The ModelSimSE 6.2C simulator has been used to simulate the design at various stages.

Xilinx synthesis tool (Xilinx ISE 9.2i version) has been used to synthesize the design for Spartan 3E family FPGA(XC3S500E).On chip verification is done by Xilinx chip scope pro 9.2i tool.

In the implementation it is assumed that the coefficients are constant and fixed. Especially in Distributed Arithmetic type the complete Look up Table contents are calculated based on the coefficients. This assumption is true and applicable for several practical applications. The present work implements few FIR filter structures on spartan3E FPGAs but analyzing the trade-off between performance and chip area.

As the present implemented structures makes use of only VHDL constructs, hence can be ported on any FPGA family. The work also brings out fundamental design goals in FPGA based design. The Distributed Arithmetic usage and its advantages over conventional arithmetic are clearly discussed.

5. Results

Results of Transmitter Side

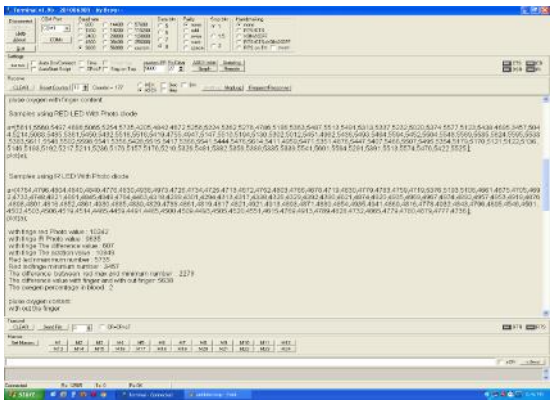

Figure 5.1 Result of transmitter side

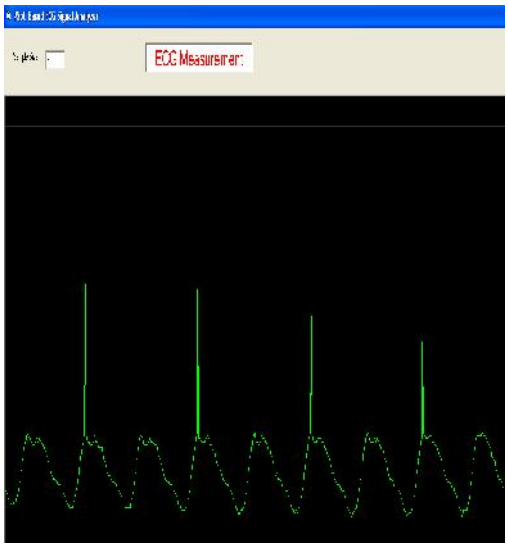

Figure 5.2 ECG Signal

Results of Receiver Side

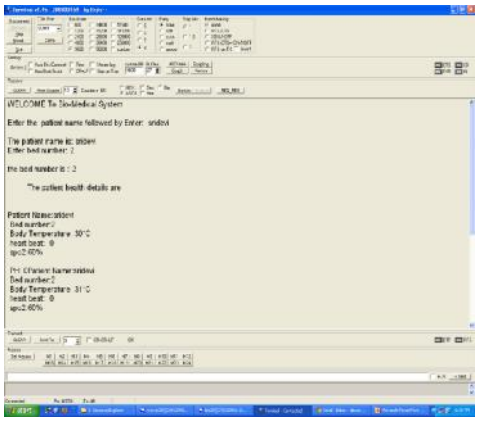

Figure 5.3 Result of Receiver side

6.Conclusion

It implemented a low cost, low power heart rate monitoring and portable biomedical using microcontroller technology.

The modular approach of the signal-conditioning unit provides the advantages of flexibility and high performance. Each biomedical signal can be optimally conditioned without concerning the operations of A/D conversion and wireless transmission. This unit is, therefore, layout in a daughter printed-circuit board. For different biomedical signals, different resistors and capacitors were used without any modification of the common circuitry.

The use of a microcontroller as the building block of the wireless recorder has the benefits of intelligence, compact size, and reliability. By the aid of this highly integrated microcontroller, external components, and hence wirings are kept to a minimum. The intelligence of this device is due to the processor itself, which could handle simple pre- processing tasks. The maximal throughput of the A/D conversion and the data transmission is about 40 kbits per second, thus, limiting the applications to low-frequency signals, such as ECG, EMG, EEG, and so forth. Further advantage of this device is its low-power consumption, which is attractive for portable applications. Moreover, this part was also layout on a motherboard to increase its mechanical strength. The change of different front-end modules is thus speeded up. We hope that the system should be adapted for minimizing the device's size and allow for daily life usage.

7.Future Scope

The input to the MECA is taken in binary format. By Adding the Image to Bit Converter input to MECA is directly in the form of frames, timing required for Motion Estimation will be reduced.

The input to the MECA is 8-bit data. It also can be extended to higher volume of data. But the Calculation time required is also high.

8. References

[1] Z.L.He, C.Y.Tsui, K.K.Chan, and M.L.Liou, "Low-power VLSI design for motion estimation using adaptive pixel truncation," IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 5, pp. 669–678, Aug. 2000.

[2] THAMMAVARAPU R. N. RAO, MEMBER, IEEE ," Bi residue Error-Correcting Codes for Computer Arithmetic" C. G. Peng, D. S. Yu, X. X. Cao, and S. M. Sheng, "Efficient VLSI design and implementation of integer motion estimation for H.264 SDTV encoder," in Proc. IEEE Int. Conf. Solid-State Integr. Circuits, 2006, pp. 2019–2021.

[3] P.Gallagher, V.Chickermane, S.Gregor, and T.S.Pierre,"A building block BIST methodology for SOC designs: A case study," in Proc. Int.Test Conf., Oct. 2001, pp. 111–120.

[4] T. H. Wu, Y. L. Tsai, and S. J. Chang, "An efficient design-for-testability scheme for motion estimation in H.264/AVC," in Proc. Int. Symp. VLSI Des. Autom. Test, Apr. 2007, pp. 25–27.

[5] J.C.Yeh, K.L.Cheng, Y.F.Chou, and C.W.Wu, "Flash memory testing and built-in self-diagnosis with march-like test algorithms,"IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 6, pp. 1101–1113, Jun. 2007.

[6] X. Xiong, Y. L. Wu, and W. B. Jone, "Reliability analysis of self-repairable MEMS accelerometer," in Proc. IEEE Int. Symp., Defect Fault Tolerance VLSI Syst., Oct. 2006, pp. 236–244.

[7] R. J. Higgs and J. F. Humphreys, "Two-error-location for quadratic residue codes," Proc. Inst. Electr. Eng. Commun., vol. 149, no. 3, pp.129–131, Jun. 2002.