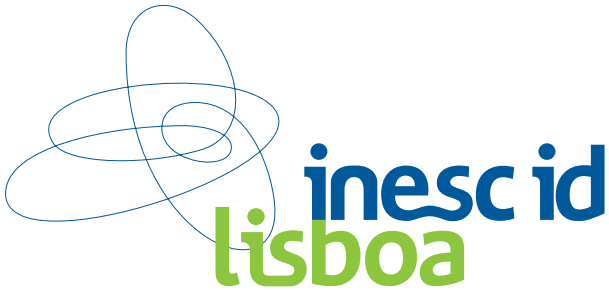

Insertion and Improvement of Testability Mechanisms on a Specialized Multimedia IP Core

Technical Report No. 32/2009

Authors: Nuno Sebastião
Nuno Roma
Paulo Flores

Lisbon
May 2009

Abstract

The set of custom validation mechanisms and test structures that were integrated in an ASIC implementation of a specialized multimedia IP core are presented in this paper. The dedicated test structures include a custom memory BIST controllers, which are capable of providing the addresses of all the faulty memory positions, a set of two scan chains and a JTAG compliant interface. Moreover, dedicated hardware structures were also proposed and included in the original IP core design, in order to provide extra test control points that significantly improve the controllability on specific locations of the circuit. An AT91SAM9263-EK board was used as an ATE equipment which required the development of dedicated software to support the STIL format for test vector generation. The obtained results demonstrated that the original circuit test coverage may be increased by a factor of 11.6, due to the use of scan chains and additional test control points placed at specific locations within the IP core. Moreover, the corresponding test pattern generation time was reduced by a factor of 1560 when these test structures were used. Despite the complexity of the circuit, the obtained results also show that the entire ASIC can be tested in less than 70ms (using a 100MHz clock).

Keywords:

- IP Core Test;
- Scan Chains;
- BIST;
- JTAG;

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last few years there has been a growing trend to design very complex processing systems by integrating pre-designed IP cores which implement, in a particularly efficient way, certain specific and critical parts of the main system. These IP cores can either be used to implement specific and dedicated processing structures that are integrated with other larger scale modules in the form of co-processors, or used as autonomous processing architectures, by following a System-on-Chip (SoC) approach.

One of such modules that has deserved a particular attention in the scope of digital video coding is the motion estimator. The task of this block is often regarded as one of the most important operations in video coding to exploit temporal redundancies in sequences of images, and it usually involves most of the computational cost of these systems [1]. As a consequence, real-time Motion Estimation (ME) is usually only achievable by adopting specialized and highly optimized VLSI structures. One example of a particularly efficient IP core that implements an Application Specific Instruction Set Processor (ASIP) for data-adaptive ME has been recently proposed [2]. This core includes efficient units for data processing and a set of embedded memories for local caching of the image pixels and of the program code to be executed. For an effective validation of this IP core regarding the performance, resource usage and power consumption, we have implemented it in an ASIC. Note that this type of implementation is the most suitable to be integrated in mass-production consumer products, such as mobile devices.

Nowadays, it is widely accepted that test structures need to be considered and included in the design of most ASICs, in order to account for eventual defects and errors that may occur during the fabrication. Due to the increased complexity of VLSI circuits, it is often not possible to effectively test the circuit without dedicated hardware structures included in the chip, especially in circuits that include memories and complex state machines. In the particular case of the considered ME processor, a subset of the testing methodologies which are frequently adopted in synchronous sequential circuits was

adopted and applied [3]. Such techniques make use of memory elements in the circuit to build dedicated test structures that provide an improved ability to control and observe the logic values at the circuit's internal nodes. Moreover, additional dedicated hardware for test had to be included in specific locations of the circuit which presented a reduced controllability. This required the knowledge of the IP core architecture and a detailed analysis of the fault coverage results.

To further improve the quality of the test and to provide the ability to test the circuit outside its production environment, a dedicated Built-In Self Test (BIST) was designed and implemented. This type of structures allow to test the circuit without special external equipment and without the need to remove the circuit from the target system. Moreover, the considered BIST was designed to allow the test of some critical parts of the circuit at nominal clock speed, while some other test techniques often only allow the circuit to be tested at lower clock frequencies.

Among the several elements that may exist in an circuit, embedded memories have the most complex fault models [4]. Nevertheless, they can be efficiently tested by adopting a BIST approach. As a consequence, a custom memory BIST controller architecture and the respective march test were developed, not only to test the embedded SRAMs, but also to allow the listing of all faulty addresses.

This report is organized as follows: In chapter 2, the architecture of the implemented ME processor is presented. In chapter 3 it is presented the set of test structures that were added to the core processor in order to improve its testability after fabrication. These structures include a BIST controller, to test the embedded memories, a set of test control points, a set of scan chains and a JTAG interface. Chapter 4 presents the testing platform that was used to validate the fabricated circuit. In chapter 5, before the conclusions, the experimental results of the implemented circuit are presented and an analysis, from a test prespective, is performed.

Chapter 2

Motion estimator architecture

The programmable and specialized architecture for ME proposed in [2] was tailored to efficiently program and implement a broad class of powerful, fast and/or adaptive ME search algorithms. The offered flexibility is attained by adopting a simple and efficient micro-architecture, illustrated in Fig. 2.1, whose modular structure is composed by optimized units that support a minimum and specialized instruction set. This data-path is also developed around a specialized arithmetic unit that efficiently computes the Sum of Absolute Differences (SAD) similarity function. Furthermore, a quite simple and hard-wired control unit is used to generate all the required control signals [2].

The Instruction Set Architecture (ISA) of the implemented ASIP was designed to meet the requirements of most ME algorithms, including some recent approaches that adopt irregular and random search patterns, such as the data-adaptive ones. Such ISA is based on a register-register architecture and provides a quite reduced number of different instructions (eight), that focus on the set of operations that are widely used in most ME algorithms [2]. The instructions directly operate the values stored in a register file composed by 24 General Purpose Registers (GPRs) and 8 Special Purpose Registers (SPRs), capable of storing one 16-bit word each.

The processor data-path, depicted in Fig. 2.1, includes two specialized units to increase the efficiency of the most complex and specific operations: an Address Generation Unit (AGU) and a SAD Unit (SADU). The `LD` operation is efficiently executed by the AGU, which is capable of fetching all the pixels of either the reference macroblock (MB) or of the corresponding search area (SA). The `SAD16` graphics instruction is efficiently implemented by the SADU. The processor includes two local scratch-pad SRAM memories for caching the pixels that are used during the computation of the Sum of Absolute Differences (the SA and the MB pixels). Both memories are dual-port 8-bit word SRAMs but with different capacities: 2048 words for the SA memory and 512 words for the MB memory. The processor also includes a program memory that holds the code to be ex-

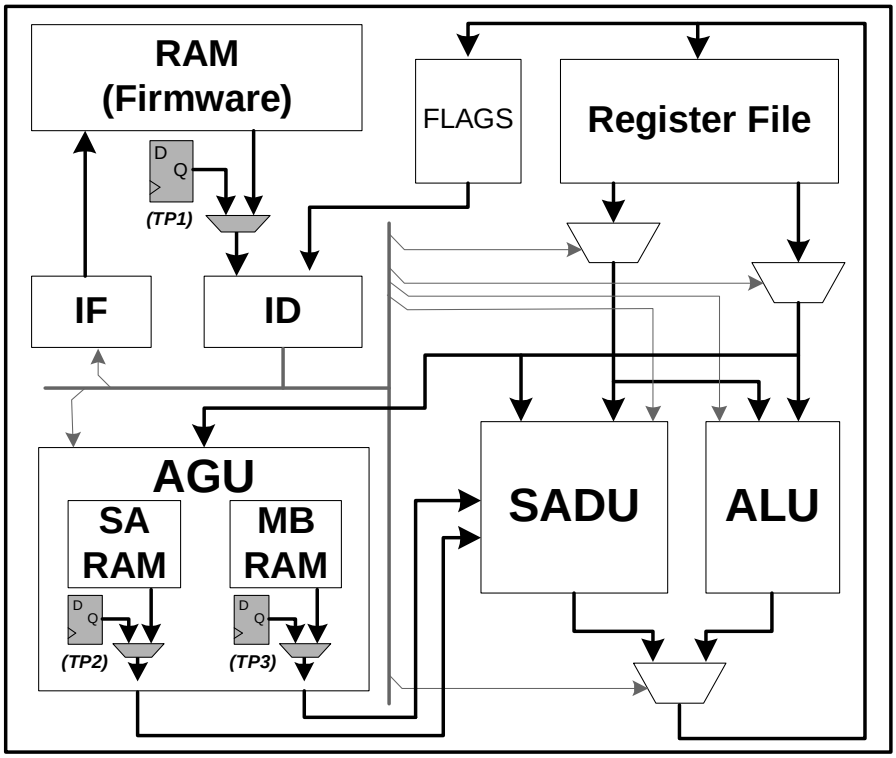

Figure 2.1: Architecture of the ASIP proposed in [2] with the included test control points.

ecuted (firmware). This memory is a single-port SRAM with 1024 16-bit byte-writable words. More detailed information about this processor architecture, including the description of its functional interface, can be found in [2].

Chapter 3

IP Core test structures

Although dedicated test structures are usually necessary in most of todays VLSI circuits, the added cost of using them must always be considered, since they increase the circuit area, may introduce a negative impact in the circuit's timing and require additional circuit pins. As a consequence, the specific test structures and mechanisms that were adopted in the circuit had to be carefully selected, by taking into account the particular characteristics of the considered ME processor, in particular, the circuit's datapath and the several memory devices that are tightly coupled with the processor structure.

The following sections describe the methodologies that were used to improve the processor's testability, namely the implementation of the scan chains and the generation of the respective test patterns, the design of the custom memory BIST controller and of

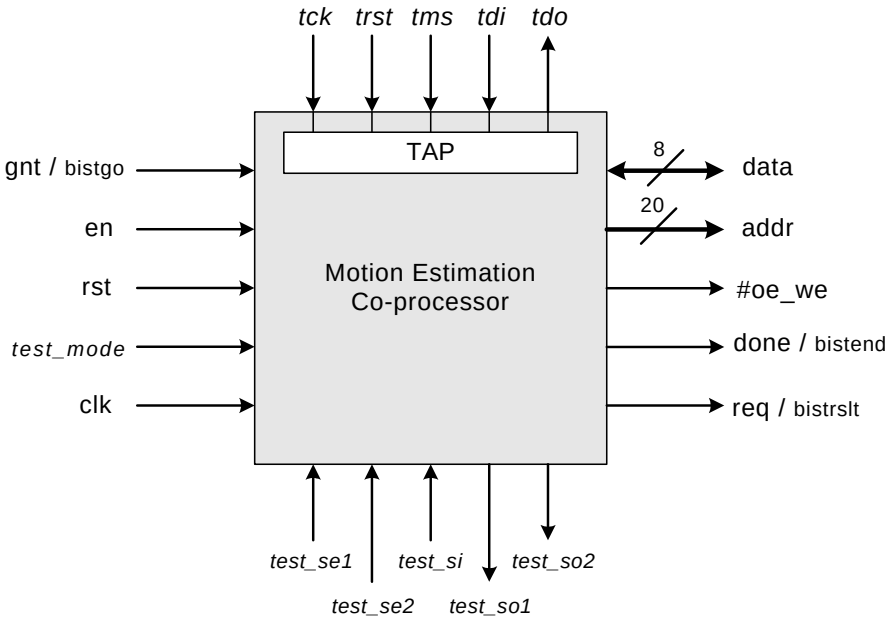

Figure 3.1: ASIC interface, including the test dedicated I/Os.

the JTAG controller. In Fig. 3.1 it is represented the external interface of the implemented multimedia IP core, which includes the dedicated I/O ports that were introduced for testing purposes (represented in *italic*). The set of adopted methodologies that were applied to improve the testability of this ASIC, namely, the implementation of the scan chains and the generation of the respective test patterns, the design of the custom memory BIST controller and of the JTAG controller are presented in the following subsections.

3.1 Scan chains and test pattern generation

It is widely known that the effort to control and observe an internal circuit node in a sequential circuit is significantly greater than in pure combinational circuits, due to the presence of memory elements. One common strategy to simplify the delivery of the test patterns is to rearrange, at test time, the connection between these memory elements in order to obtain a scan chain. To implement it, the ordinary flip-flops that were used in the design have to be replaced by scan-type flip-flops, which provide additional inputs that enable them to either function in normal mode or in test mode [3].

The addition of these scan chains and of the corresponding scan flip-flops increases the circuit area, the power consumption and may still have impact on circuit timing, due to the additional logic elements. As a consequence, different scan styles may be adopted [3], differing in the cost of the penalty incurred in each of these factors and in the complexity of generating the corresponding test control signals. The selection of the particular scan style adopted in the implemented circuit was not only dependent on previously mentioned constraints [3], but was also limited by the availability of the required scan flip-flops in the adopted standard cell library [5]. Since only multiplexed scan flip-flops are available, the *multiplexed flip-flop scan style* was the only alternative to implement the scan chains in this circuit.

Moreover, due to the presence of unregistered outputs in the memory blocks, the original circuit had to be modified in order to improve its fault coverage. Such modifications considered the inclusion of additional hardware (flip-flops and multiplexers) in order to provide a direct control of the memory output signals (TP1, TP2 and TP3 in Fig. 2.1). By including these internal test control points, the controllability of these signals, which are directly connected to combinational logic (e.g. instruction decoder), is greatly increased, therefore achieving a higher fault coverage.

The implemented circuit considered the inclusion of two distinct scan chains. One scan chain allows the extraction of the faulty addresses from the memory BIST controllers (described in section 3.2). It includes the address registers of the three memory BIST

controllers and is composed of 30 flip-flops. The other scan chain includes the remaining flip-flops in the design (752 flip-flops) and the test dedicated flip-flops, used to control the memory output signals (64 flip-flops). These two chains are controlled with the following signals: the *test_mode* signal, which drives the control inputs of the flip-flops (set and reset) into a non-active state and that must be set high during the entire test procedure; the *test_se1* and *test_se2* signals, that control how the circuit's flip-flops of the first and second scan chains, respectively, are connected; the *test_si* signal, that is used to input data to both scan chains; and the *test_so1* and *test_so2* signals, the outputs of the first and of the second scan chains, respectively.

The generation of the test patterns was carried out by an Automatic Test Pattern Generation (ATPG) tool considering the Single Stuck-at Fault (SSF) model. The adoption of the SSF model on the considered ME circuit arised from its wide range of applicability, its technology independence and the wide support among the industry. In the considered case, the used ATPG tool was the Synopsys TetraMAX.

3.2 Memory test

To test the fabrication of the three processor memories, the following march test was defined. {⇑(w55h);⇑(r55h,wAAh);⇑(rAAh,w55h);⇓(r55h,wAAh);⇓(rAAh;w55h);
⇑(r55h); ⇑(w00h);⇑(r00h,wFFh);⇓(rFFh;w00h);⇑(r00h)}
Each ascending (⇑) or descending(⇓) march element comprises a finite sequence of read(r) or write(w) operations, that are repeated in each memory cell in the corresponding ascending or descending address order. The faults that may exist are detected in the read operations, when the obtained values are compared against the values defined in the march element. Since the memory devices included in the considered processor are byte oriented, the values to be read or written are represented as bytes in hexadecimal base. As an example, the ⇑(*r55h,wAAh*) march element means: in ascending address order, read from the addressed memory position the byte value 0x55 and write, to the same memory position, the 0xAA value. When the memory positions are larger than a byte, the pattern is simply replicated to the other bytes.

The selection of this particular march test arises from its ability to detect all transition faults, all stuck-at faults and all address decoding faults. Additionaly, this test is also capable of detecting some coupling faults and state coupling faults [6, 7].

To guarantee that the implemented march test allows at-speed memory testing and also that it provides more information from the test procedure than a simple good/defect response, a custom dedicated memory BIST controller was designed. The test that is

implemented by this controller is conducted by comparing the values that are read from the memory with those expected to be stored at the various memory cells. In case of a mismatch, the BIST controller will signal the detection of a fault and will stop its operation. Then, by using a proper shift register, the controller has the capability to serially shift out the address of the failing cell. Furthermore, the controller also has the capability to resume the test sequence (from the failed address), in order to complete the remaining test procedure. In prototyping environments, this feature provides the advantage of returning more information than a mere good/defect test result. The designer can then use this information to circumvent such fault and still allow the operation of a partially defective circuit. As an example, if a program memory cell is defective, the designer may write an assembly code that avoids that particular address, thus making it possible to use the remaining circuit.

Since the program memory of the implemented ASIC provides a byte-write capability (individual write to each byte of the 16 bit words), the developed BIST controller also provides support to test this feature. Furthermore, it also provides the capability to test the address decoding logic of both ports of the SA and MB memories (dual-port SRAMs).

The architecture of the implemented memory BIST controller is shown in Fig. 3.2. This controller is composed by one comparator for error detection (with one of its inputs registered), an up/down counter for sequential address generation and a shift register for byte-write enable signal generation. The controller's state machine comprises 31 states and is responsible for implementing the defined march test. The controller interface to the outer circuitry includes three input control signals and two output result signals. The set of input control signals include the enable (*bisten*), the reset (*bistrst*) and the start/resume test sequence (*bistgo*). The set of output result signals indicate a detected fault (*bistrslt*) and the end-of-test sequence (*bistend*).

While the test procedure is being performed, the controller's enable signal (*bisten*) is high. Nevertheless, to actually start the test sequence, the *bistgo* signal must be asserted high during one clock cycle. The *bistrslt* signal indicates the test result (logic value 0 if no error was detected; logic value 1 if an error was detected), while the *bistend* signal indicates the end of the test sequence. If no error is detected, the *bistend* signal is set high and the *bistrslt* signal will remain low. In the event of an error is detected during the test sequence, the *bistrslt* signal will be set to high, while the *bistend* signal remains low, and the controller will enter into a pause state. At this state, the controller will wait for the activation of the *bistgo* signal, indicating that the result has been read and the memory address has been shifted out (if desired by the user) through the previously described scan chain. The test sequence may then be resumed. The controller also

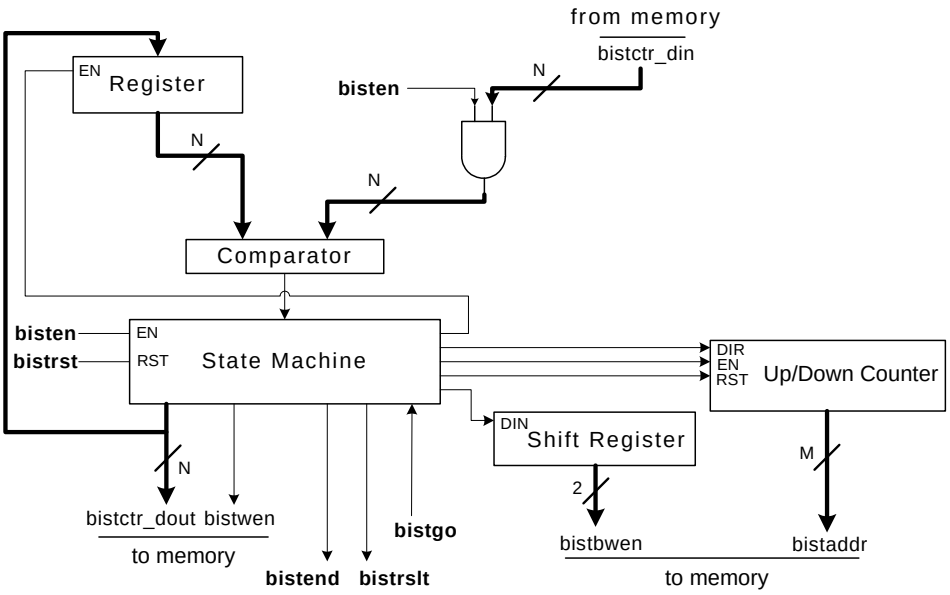

Figure 3.2: Architecture of the memory BIST controller.

includes additional output signals to address the memory (*bistaddr*), to set the memory data inputs (*bistctr_dout*) and to control the memory write enable signals (*bistbwen* and *bistwen*). The *bistctr_din* input of the BIST controller is driven by the memory data output.

Considering that the three memory blocks that are embedded in the processor architecture have distinct configurations, it was decided to implement a dedicated BIST controller for each block. Such approach was adopted not only because there was enough space available in the die, but also because it allowed to reduce the timing penalty, the routing congestion and the individual complexity of each BIST controller.

To reduce the number of required pins to operate the test structures, several control inputs (except for the enable signal) and outputs of the three BIST controllers were multiplexed. Moreover, since the memory test is only triggered on-demand and with the rest of the circuit inactive, the control inputs for the memory BIST controllers were also multiplexed with the circuit's functional inputs. This is the case of the *bistgo*, *bistrslt* and *bistend* signals, which share the I/O pin with the processor's *GNT*, *REQ* and *DONE* signals, respectively. The *bisten* signal of each BIST controller is asserted using the Joint Test Action Group (JTAG) compatible interface, described in section 3.3. This strategy reduces the number of required pins, from 15 (if each of the controllers had independent control pins) to none (when using pin sharing and the JTAG interface), and simplifies the operation of the memory BIST controllers.

3.3 Embedded JTAG controller

To provide the processor with a standard test interface, as well as to allow the test of the implemented processor's interconnections at board level, an IEEE 1149.1 [8] (JTAG) compliant interface was included. This interface comprises a Test Access Port (TAP) controller and a boundary scan register.

As described in the IEEE 1149.1 standard [8], the TAP includes the following connections: *Test Clock Input (tck), Test Mode Select (tms), Test Data Input (tdi)* and *Test Data Output (tdo).* Since a power-up reset of the test logic structures is not performed inside the chip, a *Test Reset (trst)* connection is also made available. A detailed description of the TAP module and of its inputs and outputs can be found in [8].

In the considered ME processor, the TAP controller was implemented using a 4 bit width instruction register, thus providing the possibility to encode up to 16 instructions. According to the IEEE standard [8], the implementation of the BYPASS, EXTEST and SAMPLE/PRELOAD instructions is mandatory. Additionally, this standard only specifies the encoding of the BYPASS instruction (0xF for the defined instruction width) while all of the remaining instructions have implementation-specific encoding. Other optional instructions, defined in the standard, were also implemented, such as the HIGHZ and the IDCODE. The HIGHZ instruction is quite useful in the scope of the implemented ASIC, since this processor has a three-state bidirectional bus that may be connected to a shared system bus. Therefore, this instruction allows the output drivers of the circuit to be placed at high impedance, allowing the testing of other devices also connected to the same system bus. The IDCODE instruction allows the device to be identified in a larger system and to check the current version of the circuit. Besides these pre-defined instructions, the presented design also implements three additional user-specified instructions that enable the memory BIST controllers. These instructions are the SELECTSAMEM, the SELECTMBMEM, and the SELECTINSTMEM, which assert the signals that are required to enable the memory BIST controllers for the search area, the macroblock and the program memories, respectively. Therefore, with these instructions it is possible to activate the memory test without the need for additional package pins, as it was described in section 3.2.

Chapter 4

Video coding and test platforms

To validate the manufactured processor after fabrication, the implemented ASIP was included in a prototyping platform which implements a complete H.264 video encoding system, as shown in Fig. 4.1. This same platform was also used to implement the testing system.

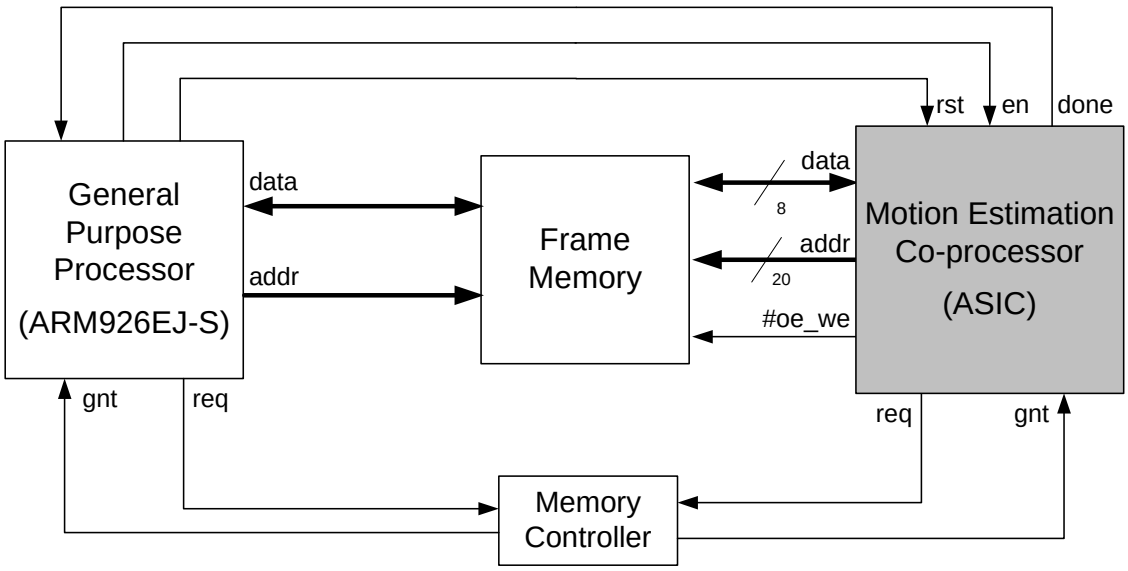

Figure 4.1: Video encoding system.

To implement this prototyping video encoder and test platform, an AT91SAM9263-EK evaluation kit [9], from ATMEL, was adopted. This development board includes an AT91SAM9263 micro-controller, based on the ARM926EJ-S processor, capable of running up to 240MHz. This board has also an extensive set of peripherals for control, communication and data storage purposes. Figure 4.2 shows the development board with the fabricated ASIC (on the center of the picture) connected in an interface board.

When used as a video encoding platform, the ARM926EJ-S processor executes all the

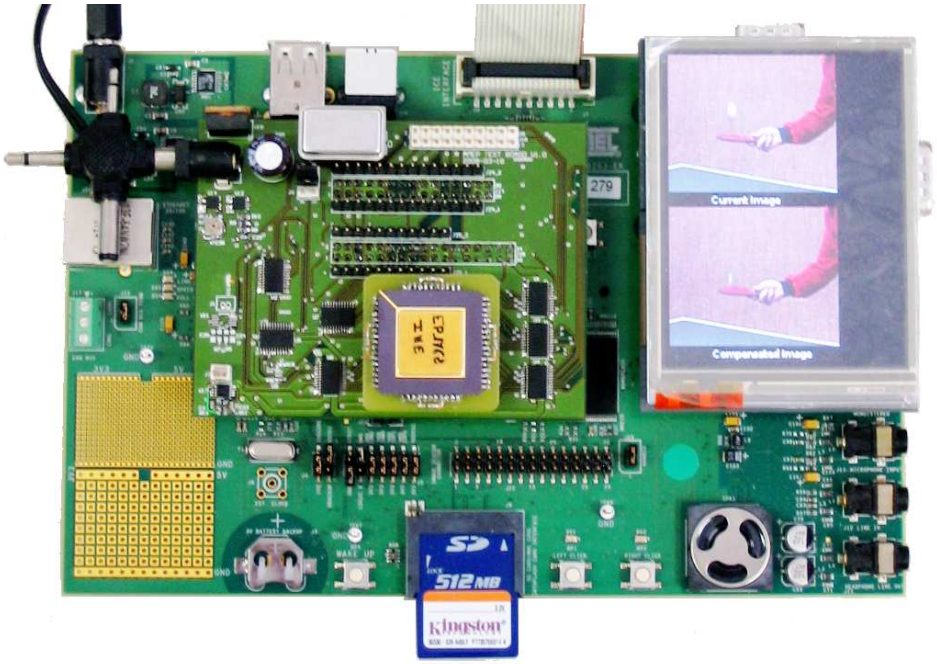

Figure 4.2: Development board.

video encoder operations, except for the ones concerning the ME task. The ME operations are executed by the manufactured ASIC, acting as a specialized co-processor of the main processing unit of the video encoder. This ME co-processor computes, in parallel with the other video coding operations, the several motion vectors that are required by the encoder to implement the video temporal prediction mechanism.

On the other hand, when used as a testing platform, the ARM926EJ-S processor executes the procedures of an Automatic Test Equipment (ATE). It implements the several test sequences, by driving the circuit inputs and by validating its responses. The AT91SAM9263 microcontroller is also able to drive the JTAG interface of the implemented processor, making it possible to validate the correct implementation of the TAP controller and to allow the access to the memory BIST functions.

Since the patterns generated by the ATPG tool (Synopsys TetraMAX) were exported in the Standard Test Interface Language (STIL) format, a custom interpreter was also developed in order to adapt the STIL format and to generate the appropriate code to be used by the implemented test platform.

In fact, the STIL format test vector data is represented in such a way that the file size is minimized. However, the interpretation of this format at runtime imposes a significant computational overhead on the ARM processor. Therefore, the test vector data provided by the STIL file is pre-processed and converted, by the interpreter, to another format

that requires a reduced computational overhead, at runtime, thus minimizing the time required to apply the test vectors.

This STIL format interpreter was developed using the PERL language and generates two files to be used by the test platform: a file containing the C source code to be executed by the ARM processor, including the timing information present in the STIL file; and a data file, to be loaded by the program, that contains the test vector data including the inputs of the circuit and the expected output values.

Chapter 5

Experimental results

The implemented ME ASIC, whose final layout is presented in Fig. 5.1, was manufactured under the *mini*@SIC program from EUROPRACTICE, using a StdCell library from Faraday Technology, based on a 0.18μm CMOS process from UMC, with 1 poly and 6 metal layers (UMC L180 1P6M MM/RFCMOS) [5]. Table 5.1 presents the obtained implementation results. These results show that the processor IP core implementation (including the test structures but excluding the program and the two pixel local memories) requires 42k equivalent logic gates (but 112k equivalent logic gates are required to implement the whole processor) and may be operated at a maximum frequency of 140MHz. This allows the computation of motion vectors in real-time, for most current video coding applications.

The results related to the test structures in the implemented ASIC are summarized in Table 5.2. It can be seen that the fault coverage that would be obtained if no test

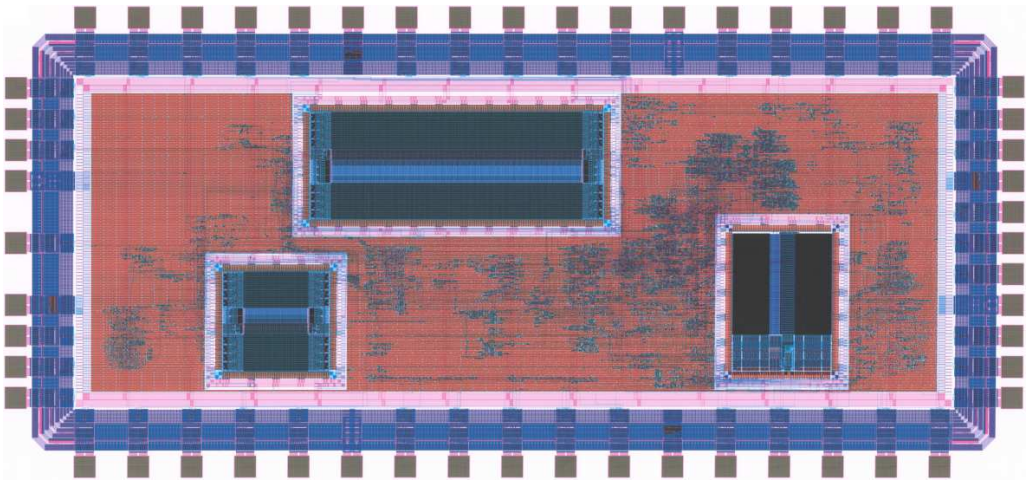

Figure 5.1: Implemented prototype of the ME ASIP in an ASIC based on the UMC 0.18μm CMOS process.

Table 5.1: Implementation results of the motion estimator using the UMC 0.18μm CMOS ASIC.

Silicon Area / Equivalent Logic Gates	
IP Core	0.25 mm^2 / 26 kGates
Memory Test Controllers	0.007 mm^2 / 0.7 kGates
Scan Chains Overhead	0.003 mm^2 / 0.3 kGates
Test Control Points	0.012 mm^2 / 1.2 kGates
JTAG TAP Controller	0.14 mm^2 / 14 kGates
Memories	0.68 mm^2 / 70 kGates

structures were included in the design is of only 8% for the core logic (not including the memories nor the TAP controller). Moreover, it is interesting to note that by applying a simple naive approach, that merely replaces the original IP core flip-flops with scan flip-flops, the ATPG tool achieved a fault coverage of 82% and generated 912 test patterns. In constrast, by considering the inclusion of the additional test control points (TP1, TP2 and TP3 as described in section 3.1 and illustrated in Fig. 2.1) in order to provide a direct control of the memories output signals, the ATPG tool increased the number of test patterns to 1286, leading to a significantly more satisfactory fault coverage of 93%. Furthermore, the time required by TetraMAX to generate the test patterns was just 10 minutes when the scan chains with the proposed test control points were included. This represents a significant improvement over the 18 hours required to generate them when these test control points were not used. If no test structures were included, it would take 260 hours to generate the test patterns.

The generated 1286 test patterns require approximately 6 million ATE clock cycles to be delivered to the processor. This represents a total testing time of about 60ms for the core logic, when using an ATE clock period of 10ns. The memory BIST controllers

Table 5.2: Test results of the implemented ASIC.

	w/o test structs	**w/ test structs w/o test points**	**w/ test structs w/ test points**
# test patterns	-	912	1286
Test coverage	8%	82%	93%
Test pattern generation time	260 hours	18 hours	10 minutes

require approximately 645μs to test the three memory blocks using the same clock period (about 85μs for the MB memory, 330μs for the SA memory and 230μs for the program memory).

The implementation areas, presented in Table 5.1, also show that the additional logic required by the scan chains (1.2% area increase) and by the memory BIST controllers (2.8% area increase) has a relatively small cost in terms of hardware resources. When considering the JTAG controller, the area increase is quite significant, although the relative impact may be mitigated in larger designs. Nevertheless, the JTAG TAP controller was still considered for implementation in this circuit, allowing the board level interconnection test procedure when included in the final system. Moreover, the implementation of this controller avoided the usage of dedicated circuit pins to control the internal memory BIST controllers, which also reduced the circuit implementation cost.

Chapter 6

Conclusions

This report presented a set of custom validation mechanisms and test structures that were integrated in the ASIC implementation of a specialized IP core for adaptive motion estimation. Some required changes to the original IP core design were also presented in order to significantly increase its testability. To provide the processor with a standard test interface, which allows the test at board level, a JTAG controller was also included. The circuit with the proposed test structures was implemented using a StdCell library based on a 0.18μm CMOS process. According to the obtained results, the inclusion of scan chains and of additional test control points at specific locations on this specialized circuit increased the test coverage of the ASIC by a factor of 11.6 and reduced the corresponding time to generate the test patterns by a factor of 1560, without significantly increasing the required silicon area. As a consequence, the improvements that were obtained by considering the described test structures entirely justify their inclusion on this ASIC. Moreover, they also allowed to conduct the test of the circuit and its validation using the development platform as an automatic test equipment.

References

[1] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Acad. Publish., Jun. 1997.

[2] T. Dias, S. Momcilovic, N. Roma, and L. Sousa, "Adaptive motion estimator for autonomous video devices," *EURASIP J. on Embedded Systems*, 2007.

[3] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. COMPUTER SCIENCE PRESS, 1990.

[4] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, 2000.

[5] *Faraday ASIC Cell Library FSA0A_C 0.18µm Standard Cell (v1.0)*, Faraday Techn. Corp., August 2004.

[6] A. J. van de Goor and I. B. S. Tlili, "March tests for word-oriented memories," in *DATE '98: Proceedings of the conference on Design, automation and test in Europe*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 501–509.

[7] A. van de Goor and S. Hamdioui, "Fault models and tests for two-port memories," *VLSI Test Symposium, 1998. Proceedings. 16th IEEE*, pp. 401–410, Apr 1998.

[8] *IEEE 1149.1-2001 - Standard Test Access Port and Boundary-Scan Architecture*, IEEE, June 2001.

[9] *AT91SAM9263-EK Evaluation Board - User Guide*, ATMEL Corporation, March 2007.