FP7-ICT-2009-4-248613

DIAMOND

Diagnosis, Error Modelling and Correction for Reliable Systems Design

Instrument: Collaborative Project

Thematic Priority: Information and Communication Technologies

Definition of the DIAMOND Platform (Deliverable D4.2)

Due date of deliverable: June 30, 2010
Actual submission date: June 28, 2010

Start date of project: January 1, 2010 Duration: Three years

Organisation name of lead contractor for this deliverable: IBM Haifa Research Lab

Revision 1.3

Project co-funded by the European Commission within the Seventh Framework Programme (2010-2012)		
Dissemination Level		
PU	Public	☒
PP	Restricted to other programme participants (including the Commission Services)	☐
RE	Restricted to a group specified by the consortium (including the Commission Services)	☐
CO	Confidential, only for members of the consortium (including the Commission Services)	☐

Notices

For information, contact Dr. Jaan Raik, e-mail: jaan@pld.ttu.ee.

This document is intended to fulfil the contractual obligations of the DIAMOND project concerning deliverable D4.2 described in contract number 248613.

Table of Revisions

Version	Date	Description and reason	Author	Affected sections
1.0	May 3, 2010	Contents Created	O. Rokhlenko	*Entire document*
1.1	May 4, 2010	Added IBM test cases and HC11 open-source test case	O. Rokhlenko	*IBM test cases, Open-source test cases*
1.2	May 12, 2010	Added OpenSparc S1 and DLX processor architecture	A. Finder, R. Könighofer	*Open source and academic benchmarks*
1.3	June 15, 2010	Enhanced the Introduction section, added Executive summary and some general test	O. Rokhlenko	*Entire document*

Authors, Beneficiary

Oleg Rokhlenko, IBM
Cindy Eisner, IBM
Görschwin Fey, UNIB
Alexander Finder, UNIB
Jaan Raik, TUT
Maksim Jenihhin, TUT
Gunnar Carlsson, Ericsson
Georg Hofferek, TUG
Robert Könighofer, TUG

Executive Summary

This document presents diagnosis and repair benchmarks which will be used for evaluation of the tools developed in the scope of DIAMOND project. These benchmarks include designs contributed by the industrial partners, open source designs and academic examples. The main goal of this deliverable is to detail the main features of the designs that will provide challenges for diagnosis and repair of fault on different levels and the tools supporting it.

List of Abbreviations

ASI - Address Space Identier
ASR - Ancillary State Registers
CISC - Complex Instruction Set Computing
CPU - Central Processing Unit
DLX - A processor architecture by Hennessy and Patterson

FP7 - European Union’s 7th Framework Programme
FPGA - Field Programmable Gate Array
ISA - Instruction Set Architecture
MBIST - Memory Built-In Self Test
MIPS - Million Instructions per second
RISC - Reduced Instruction Set Computing
RTL - Register Transfer Level
SoC - System on Chip
TLB - Translation Lookaside Buffer
UART - Universal Asynchronous Receiver/Transmitter

Table of Contents

1 Introduction

In the following chapters, a number of individual diagnosis and repair benchmarks are described in detail. In each case, there is a general description of the design, including an indication of its size and complexity, and any features of technical interest from a diagnosis viewpoint.

The industrial partners — IBM and Ericsson — chose benchmarks from their current activities. In addition the consortium chose 5 open source designs. All together, these benchmarks cover a wide range of designs and architectures, of sufficient size and complexity to provide challenges for diagnosis and repair of faults on different levels and the tools supporting it.

The following includes design characteristics of the benchmarks chosen for evaluation. The main benchmark features include:

- Analyzing design vulnerability to soft errors under different constraints, such as limited protection budget, strict requirement of perfect integrity of the results and a necessity to handle large scale circuits.
- Testing board and system level operation on the MBIST.
- Collecting realistic examples of design errors and applying diagnosis and correction methods to correct them.
- Validation of specification and implementation level diagnosis and correction methods on hierarchical setup.
- Validation of specification and implementation level diagnosis and correction methods on a peripheral core.
- Validation of specification and implementation level diagnosis and correction methods on CPU core together with communication interfaces.

2 IBM benchmarks

Due to a moderate level of uncertainty regarding the schedule of internal development, IBM chose 3 different test cases which differ in size, complexity and diagnosis features. Among these test cases, one will be selected for the final evaluation.

Also, due to the confidential character of the data, full details are provided in the confidential part of the document [2].

2.1 POWER processor unit

General description

One of the functional units of POWER7 core will be selected for evaluation.

Diagnosis features

In modern designs the budget for latch protection against soft errors is limited. Thus an accurate vulnerability analysis which identifies most vulnerable latches is required. Verification of detection logic is at least as important as protecting vulnerable latches. This is especially true in the design that is part of the CPU used in blade centers. Therefore, the selected unit can become a valuable test-case for DIAMOND tools evaluation.

2.2 Mainframe processor unit

General description

One of the functional units of System z11 microprocessor will be selected for evaluation.

Diagnosis features

As in all prior mainframe designs, hardware reliability is a top concern during the design of the z11 processor. The top priority is preserving absolute integrity of program results, followed by continuous system availability. Therefore, this particular design is a perfect test case of detection logic verification.

2.3 Memory buffer

General description

The SoC is a memory controller for main memory applications.

Diagnosis features

Estimating vulnerability to soft errors and reducing it is one of the key tasks in reliability diagnosis and repair methodology in a memory buffer. Thus this design can serve an excellent candidate for evaluation of vulnerability analysis tools.

3 Ericsson benchmarks

Due to the confidential character of the data, full details are provided in the confidential part of the document [2].

3.1 Ericsson SoC chip and core

General description

The SoC has 2.53Mbits of memory within 79 memory instances. It is coded in VHDL. All the testbenches are coded in VHDL as well. There is one top level testbench and testbenches for each block respectively.

Design features

- It is a real industrial design, containing both data paths and complex control structures.
- All RTL code, testbenches and netlists are available.
- Specifications on functionality and testbenches are available.
- It is fairly large, which may stress tool performance.
- It has a hierarchical structure, which is suitable for adding a test, diagnosis and repair architecture to the design.
- It is implemented with IEEE 1149.1 Boundary Scan, which also could be a basis for the architecture.
- It is implemented with scan technique and memory BIST, which could be used as an infrastructure where additional test, diagnosis and repair instruments could be integrated.

Diagnosis features

The device has an IEEE 1149.1 TAP controller and Boundary Scan cells. Memory self-test (MBIST) is also implemented in the device. The MBIST can be operated on from the IEEE 1149.1 TAP. This opens opportunities for board and system level operation on the MBIST.

4 Open-source & academic benchmarks

4.1 The DLX Processor Architecture

The sections below provide information which is non-confidential and publicly available [4, 5].

General description

The DLX processor architecture is a simple RISC-like load/store architecture designed by Hennessy and Patterson [4], primarily to serve educational needs. This design is particularly suitable for our purposes. On one hand, it is a realistic example, often used in verification literature (cf. e.g. [1]). On the other hand, it is rather easy to understand, also due to the fact that lots of documentation and tutorials are available online. Finally, we think that it is of reasonable size for evaluating prototype tools, as they will be developed within DIAMOND.

The DLX architecture consists of 32 general purpose registers, each being of 32 bits in size. Furthermore, it contains 32 single-precision floating point registers (in IEEE 754 format [3]) that can also be used as 16 double-precision registers. Finally, there is a set of special registers that contain status information. The memory is accessed using 32-bit addresses, in Big Endian mode. Instructions are of size 32 bits and must be aligned.

There are four classes of instructions: (a) load and store operations, (b) arithmetical and logical operations, (c) branches and jumps, and (d) floating-point operations. Load and store operations can be applied to any general-purpose or floating-point register. Only one addressing mode is supported, namely base register plus a 16-bit (signed) offset. Arithmetical and logical operations can be applied to registers only. Operands may be other registers or constants (but no memory locations). There are four jump instructions (jump to fixed label, or to register value; remember return address, or don't) and two branch instructions (branching if a register value is equal, or unequal to zero). All floating point instructions exist in a single-precision and a double-precision version. The double-precision instructions always affect two neighboring floating-point registers.

Diagnosis features

We will implement our own transaction-level description of a DLX-CPU in the programming language C.[1] As it is inevitable in every development process, we expect mistakes to happen during implementation. Using a version control system we will keep all erroneous versions and use them as benchmarks for our diagnosis and repair tools. Thus, we will have more realistic examples of design errors.

4.2 16-bit RISC CPU

General description

The 16-bit RISC CPU benchmark is a simple generic processor that has some features of the MIPS and ARM CPUs. The code contains fetch, decode and execute functions. The latter includes writeback. While there are only 12 instructions defined, they can be easily extended. It is important that the benchmark includes a wide range of addressing modes:

- Perform operation on two registers, place result into a third register
- Load a word from memory into a register
- Store a word from a register into memory
- Load low half of a register with an immediate value
- Load high half of a register with an immediate value
- Copy contents of a register into another one
- Jump to a 16-bit address
- Branch to register + offset
- Branch by offset if register == 0

This example has been made available by Jeffrey A. Meunier from University of Connecticut at wikisource pages `http://freesourcecode.wikispaces.com/cpu_emu_16-1.0.0-cpu.c`.

[1]This implementation will be made publicly available under a suitable open-source licence (to be determined).

Diagnosis features

Similar to the DLX benchmark, this example is applied in validation of specification-level diagnosis and correction methods in the DIAMOND project.

4.3 HC11 core

General description

The HC11 CPU Core is a fully-synthesizable VHDL implementation of the HC11 CPU provided by Green Mountain (GM) Computing Systems Inc. The design is an implementation of the functionality of the Motorola M68HC11 microcontroller's CPU. Its original reference manual can be found at `http://www.freescale.com/files/microcontrollers/doc/ref_manual/M68HC11RM.pdf`.

The M68HC11 is a 8-bit data, 16-bit address CISC microcontroller now produced by Freescale Semiconductor. It has two eight-bit accumulators, A and B, two sixteen-bit index registers, X and Y, a condition code register, a 16-bit stack pointer, and a program counter. In addition, some instructions treat the A and B registers as a combined 16-bit D register. The M68HC11 is optimized for low power consumption and high-performance operation at bus frequencies up to 4 MHz.

Other features include:

- Powerful bit-manipulation instructions
- Six powerful addressing modes (Immediate, Extended, Indexed, Inherent and Relative)
- Power saving STOP and WAIT modes
- Memory mapped I/O and special functions

GM HC11 CPU implements all instructions of M68HC11 with the exception of the divide instructions. GM have synthesized the CPU core for both Xilinx and Altera FPGAs using FPGA Express from Synopsys. The design used 1076 slices and runs at 31MHz on the Xilinx Virtex 400E part. On the Altera APEX 20K100 part, the design ran at 32MHz and used 2142 LEs.

The GM HC11 CPU Core package includes the synthesizable core, projects, self-checking testbenches and a debugger. The package is an open-source and a free download.

Diagnosis features

The HC11 benchmark is selected as an example of a real-world CPU core. In the DIAMOND validation this benchmark is applied either in implementation level diagnosis and correction or alternatively as a core in a hierarchical setup, where a system is represented at the specification level and some cores are provided at the implementation level.

4.4 UART 16550

General description

The package available at OpenCores `http://opencores.org/project,uart16750` contains a synthesizable 16550/16750 UART core implementation. The UART (Universal Asynchronous Receiver/Transmitter) core provides serial communication capabilities, which allow communication with the modem or other external devices, like another computer using a serial cable and RRS232 protocol. The device changes incoming parallel information to serial data which can be sent on a communication line. A second UART can be used to receive the information. The UART performs all the tasks, timing, parity checking, etc. needed for the communication.

The main features of the design are as follows:

- Full synchronous design
- Pin compatible to 16550/16750
- Register compatible to 16550/16750
- Baudrate generator with clock enable
- Supports 5/6/7/8 bit characters
- None/Even/Odd parity bit generation and detection
- Supports 1/1.5/2 stop bit generation
- 16/64 byte FIFO mode
- Receiver FIFO trigger levels 1/4/8/14/16/32/56
- Control lines RTS/CTS/DTR/DSR/DCD/RI/OUT1/OUT2
- All interrupts sources/modes

The core was synthesized on an Altera Cyclone II, connected to x86 standard hardware and then tested with standard OS drivers from: Linux 2.2/2.4/2.6, Windows 2000/XP/Vista, BSD, and DOS.

Diagnosis features

The UART 16550 benchmark is selected as an example of a peripheral core. Similarly to HC11, which is a processor core, this benchmark is applied either in implementation level diagnosis and correction or alternatively as a core in a hierarchical setup, where a system is represented at the specification level and some cores are provided at the implementation level.

4.5 S1 Core

General description

The S1 Core is an open source hardware microprocessor design developed by Simply RISC. The S1 Core is a reduced Verilog implementation of the OpenSPARC T1 from Sun Microsystems. The design is licensed under the GNU General Public License and the development status of the design is stable. While the original design is a complete microprocessor with 8 cores the S1 represents only one 64-bit SPARC v9 core which supports one to four independent execution threads. In addition the design includes

- a wishbone bridge,
- a reset controller,
- and a basic interrupt controller.

The S1 Core specification can be found at `http://www.srisc.com/download/simplyrisc-s1-0.1.pdf`.

Basically, the design follows the SPARC v9 64-bit *Instruction Set Architecture* (ISA) and the ISA is supported by the GCC compiler. For the S1 Core implementation a 64-bit wide data bus and address bus are used. Further the S1 Core includes the following features:

- 16 Kbytes of primary instruction cache (Level 1) with a 32-byte line size
- 8 Kbytes of primary data cache (Level 1) with a 16-byte line size
- 3 Mbytes of secondary (Level 2) cache
- Fully associative instruction and data translation lookaside buffers (TLB)

As mentioned before the core has support for four threads. This support consists of a full register file with eight register windows per thread, with most of the address space identifiers (ASI), ancillary state registers (ASR), and privileged registers replicated per thread. The four threads share the instruction, the data caches, and the TLBs. The S1 Core has a single issue, six stage pipeline. The six stages are

1. Fetch
2. Thread Selection
3. Decode
4. Execute
5. Memory
6. Write Back

The main block of the S1 Core is the *SPARC Core to wishbone master interface bridge* that translates the requests and return packets of the SPARC Core into the wishbone protocol. The wishbone interconnect allows to connect several other cores to the S1.

The features of the bus interface are:

- Standard signal names identified by leading `wbm_` chars,
- No ERR/RTY support,
- 64-bit address bus,
- 64-bit data bus supporting 8, 16, 32 and 64-bit accesses,
- Data transfer ordering is Big Endian,
- Support for single Read/Write cycles.

The full specification of the microarchitecture can be found at [6].

Diagnosis features

Similarly to the previous open core designs in the DIAMOND validation this benchmark is applied either in implementation level diagnosis and correction or alternatively as a core in a hierarchical setup, where a system is represented at the specification level and some cores are provided at the implementation level. The S1 core includes features of the two previous designs. The publicly available implementation level description contains a CPU core together with communication interfaces.

5 References

[1] J. R. Burch and D. L. Dill. Automatic verification of pipelined microprocessor control. In D. L. Dill, editor, *Sixth Conference on Computer Aided Verification (CAV'94)*, pages 68–80. Springer-Verlag, Berlin, 1994. LNCS 818.

[2] DIAMOND Consortium. D4.2 - definition of the diamond platform (confidential). DIAMOND - Diagnosis, Error Modeling and Correction for Reliable System Design, ICT 2009.3.2, 2010.

[3] IEEE Computer Society Standards Committee. Working group of the Microprocessor Standards Subcommittee and American National Standards Institute. *IEEE standard for binary floating-point arithmetic*. ANSI/IEEE Std 754-1985. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1985.

[4] David A. Patterson and John L. Hennessy. *Computer architecture: a quantitative approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[5] Patty M. Sailer, Philip M. Sailer, and David R. Kaeli. *The DLX Instruction Set Architecture Handbook*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[6] Sun Microsystems, Inc. OpenSPARC T1 Microarchitecture Specification. `http://opensparc-t1.sunsource.net/specs/OpenSPARCT1_Micro_Arch.pdf`, 2006.