Roberts: Reconfigurable Platform for Benchmarking Real-time Systems

Thomas C.P. Chau, Wayne Luk
Department of Computing
Imperial College London, UK
{c.chau10, w.luk}@imperial.ac.uk

Peter Y.K. Cheung
Department of Electrical and Electronic Engineering
Imperial College London, UK
p.cheung@imperial.ac.uk

ABSTRACT

This paper presents Roberts, a Reconfigurable platfOrm for BEnchmarking Real-Time Systems. Roberts is the first platform which can be customised for a given system-under-test to support benchmarking of real-time properties and energy consumption. The benchmarking takes into account system workload and environmental events, with facilities for generating test vectors conforming to the specification of system-under-test, and with support for on-line monitoring of the response time, output values and energy consumption. The proposed benchmarking platform has been implemented in the DE4 development system to provide cycle-accurate timing measurement at nano-second precision to analyse high performance applications. An evaluation of our approach shows that the platform can be used in analysing the performance of target applications and overheads of other timing facilities, such as the interval timer on processors.

Keywords

FPGA, benchmarking, real-time systems

1. INTRODUCTION

A real-time system must respond to events within explicit deadlines or the system may risk severe consequences. Examples of applications that require real-time response include flight control systems, robotics, assembly lines and nuclear plant control. There are important requirements that real-time systems must meet to support critical applications.

- Timeliness
- Functional correctness

For decades, a proliferation of work is proposed addressing the above issues. Advanced real-time applications require tight response time and measurement precision has increased to microsecond level. For example, high-frequency trading is becoming popular with execution time in the order of microseconds [1].

However, traditional execution time analysis techniques such as profiling are subject to precision problem. For software, there is gprof to analyze where a program spent its execution time on. Some system calls allow obtaining system time for execution time measurement. However, due to factors such as compiler support and interrupt of I/Os, these measurement techniques significantly degrade system performance, having precision at most in milliseconds.

This work was presented in part at the Third International Workshop on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART2012), Naha, Okinawa, Japan, May 31-June 1, 2012.

Despite simulation of system at register-transfer-level can provide cycle-accurate information, simulation does not scale well since technology advances are making real-time systems more complex [2]. Distributed computing and multiprocessing complicate task, resource and communication scheduling. Static design and analysis methods are often infeasible for ensuring correctness of systems within deployed environment.

Running benchmarks on deployed systems provides an additional level of confidence as some defects at both timing and functional aspects cannot be discovered before deployment. It also provides an opportunity to identify changes in performance due to hardware, software or algorithms.

This paper presents Roberts, a real-time system benchmarking platform using FPGA technology. The platform tests the timeliness, functional correctness and energy consumption of the system-under-test (SUT). The measurement of response time is cycle accurate, with precision up to 3.6ns on an FPGA running at 275MHz. Before benchmarking, a test vector generation tool produces test vectors based on timing and functional specification of the real-time system. During benchmarking, the platform sends test vectors to the SUT, captures output values, and detects timing or functional errors.

The benchmarking approach is application driven. Roberts exploits reconfigurability for customising the platform, such that the SUT is not limited to those residing in the same FPGA, but can be an external non-FPGA system with interface to the benchmarking facilities. Moreover, the benchmarking hardware can be detached from the system after benchmarking so that it does not produce performance or resource overhead for the system.

The contributions of this paper include:

- Roberts, a benchmarking platform for deployed real-time system. The platform can be customised for a given system-under-test to support system-level measurement.
- An implementation of Roberts in the DE4 development system, which is capable of cycle-accurate timing measurement at nano-second precision to analyse high performance applications.
- An evaluation of our approach, showing that Roberts

can be used in analysing the performance of processor-based and logic-based applications.

The rest of this paper is organised as follows: Section 2 describes existing work in system testing and performance measurement. Section 3 presents a design approach associated with our platform. Section 4 provides details of the implementation. Section 5 evaluates the effectiveness of the proposed approach and Section 6 concludes this paper.

2. BACKGROUND

There has been much work on testing and verifying system performance. Built-in self-test (BIST) has been well studied and is widely used in testing integrated circuits [3]. The main purpose of BIST is to reduce test duration and complexity of external test equipment. BIST is found in safety-critical systems where comprehensive self-test is performed at power up and a periodic self-test assures that the system is safe within a safety interval [4]. These methods add cost to the system, such as extra circuitry and memory space. The test structure might affect the system performance and the testing hardware itself can also fail. A BIST for logic blocks in FPGAs has been proposed in [5], which allows a test configuration to be programmed for off-line testing.

An on-chip test architecture was presented in [6] to test an SoC implemented on FPGA using run-time traffic patterns. Output data from SUT are stored and uploaded for off-line functional verification. In [7], programmable logic cores are embedded on fixed-function integrated circuits for debugging purpose. Real-time and on-chip profiling for soft processor running on FPGA has been presented in [8] which allows cycle-accurate execution time of specified code regions to be measured. ChipScope Pro [9], SignalTap II [10] and Corus [11] are commercial logic analyzers and toolkits for on-chip FPGA verification and debug.

There are performance counter and internal timer for soft processors on FPGA [12]. They provide a simple way of measuring execution time. However, such methods have several drawbacks, such as increased use of hardware, degraded clock frequency, additional interrupts and longer latency.

Automated test equipment (ATE) is another testing approach which is commonly used in electronic manufacturing industry [13]. ATE is able to test small electronic devices as well as complex systems. It employs a master controller to host test applications and store test results. An interface test adapter (ITA) and prober provide connection between the tester and device under test. ATE has two main problems [14]. First, the device under test could often be at the leading edge of technology where the tester is not able to test due to lack of speed and precision. Second, the tester needs to react to a range of device. Cost is added to support complex interfacing and coordination protocol.

There are benchmarking facilities utilising reconfigurable hardware. Athena [15] is an evaluation tool for automated benchmarking of cryptography algorithms targeting multiple FPGA platforms. GroundHog [16] is a benchmarking suite that measures the power consumption of reconfigurable technology for mobile computing applications. Both approaches target specific application domains, and do not explicitly evaluate applications and systems with real-time requirements. Hartstone [17] adopts Whetstone benchmark programs as synthetic load for testing real-time systems' ability to handle hard real-time applications. However, it does not provide a framework which applies test data to real-time systems and obtains information for evaluation.

3. PLATFORM DESIGN

In this section, we describe an approach for using Roberts as a platform to benchmark timing and functional behaviour of real-time systems. Much of the existing work on run-time testing [6–12] does not focus on benchmarking real-time systems. They require modifications to the architecture under test and do not explicitly verify the real-time requirements.

Our work performs automatic benchmarking at system-level. There are two novel aspects which address the requirements of real-time system as mentioned in Section 1:

- Roberts is a benchmarking platform which focuses on assessing deployed systems in meeting real-time requirements. It evaluates more for the SUT's response time rather than for its throughput. It addresses the drawbacks of static analysis which does not involve environment effects on system performance. Timeliness and functional correctness are explicitly specified and verified in the benchmarking process. Measurement is cycle-accurate at nano-second precision.

- Roberts exploits reconfigurability to customise for different applications and SUTs. The platform hardware is designed as a template, which is a generic description with different parameters to capture different customisation. Given the timing requirement and data format of the SUT, our benchmarking unit only needs customisation of: a) clock frequency, b) I/O connections to the SUT, c) interface to test vector storage.

Figure 1 shows the proposed benchmarking platform. It consists of a host which is usually implemented off-line on a computer and a benchmarking unit on an FPGA. The SUT can be implemented on-chip or off-chip.

The host generates test vectors off-line for the benchmarking unit. During benchmarking, the system driver sends test vectors to the SUT and the monitor observes outputs at run-time. Feedback from the benchmarking unit is visualised on the host for debugging and updating SUT's specification.

3.1 Pre-benchmarking

Figure 2 shows the design flow of Roberts. We first discuss the pre-benchmarking stage, which is an off-line preparation on the host.

The design flow starts from a specification of SUT. It includes timing properties (Table 1) and functional requirements (Table 2) given by user during system and application design. Having the SUT specification, test vectors are generated and the benchmarking unit is customised.

A k-bit test vector is denoted as $X = < T, C, I, O >$, where T, C, I and O represent timing properties, control signals, input data and output data, respectively. The length of I and O determines the number of I/O pins connecting the benchmarking unit and the SUT.

In step A, test vectors of real-time systems should incorporate timing properties T, such as duration and frequency of input events, as well as response time of the system to those events. Control signals C and input data I could occur periodically or at irregular intervals, the minimum of

Table 1: Timing parameters of an SUT specification

Timing properties	Description
Input setup time	The duration that an input value is applied to the SUT.
Response time	Correct output should be generated within this time.
Output hold time	The duration that the output should hold.

Table 2: Functional requirements of an SUT specification

Functional requirements	Description
Functional model	The functional level description of the SUT's behaviour.
Input data	The values needed for processing.
Output data	The expected output value.
I/O connection	The number of input/output pins of the SUT.

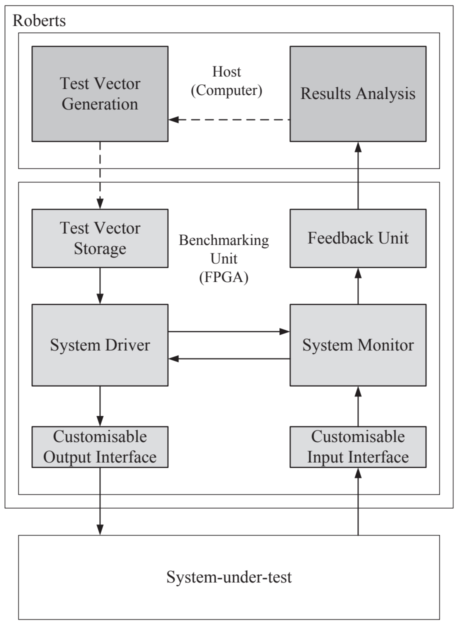

Figure 1: Interaction of Roberts and system-under-test (solid and dotted arrows represent on-line and off-line data transfer respectively)

which is t seconds. The number of input events determines the number of test vectors which is denoted as n.

In step B, functional simulation generates correct outputs O based on the input events. Simulation could be performed by a software test bench, such as one coded in SystemC or Verilog, that emulates the operations performed by the SUT. At this point, there are n test vectors which include golden timing and functional information.

In step C, the benchmarking unit is customised according to the input event interval t and number of test vectors n. The benchmarking unit is provided as a template, which allows parameterisation of frequency, I/O connections and interface to test vectors' storage.

Here is an illustration of the customisation of benchmarking unit. Consider an SUT reading input data in intervals of t seconds. To satisfy the input timing requirement, the benchmarking unit should run at a clock frequency f and send test vector for l cycles. The relationship of f and l is described as follows:

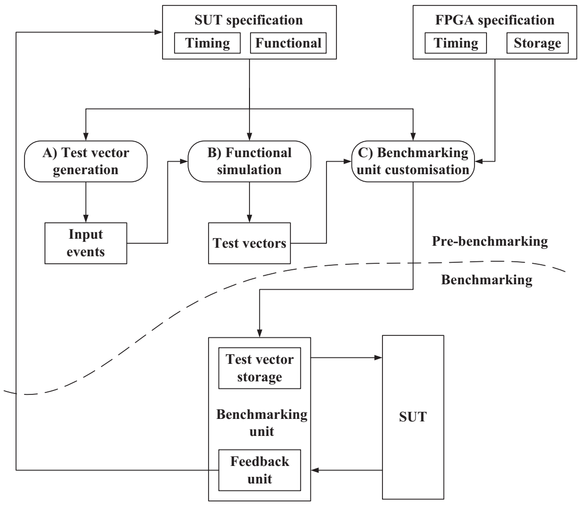

Figure 2: Design flow of the benchmarking platform

$$l = \lfloor t \times f \rfloor \text{ where } l \geq 1 \tag{1}$$

The benchmarking efficiency is bounded by the memory throughput. The throughput requirement p of test vector storage is:

$$p = \frac{k}{t} \tag{2}$$

Our template employs on-chip memory blocks where possible to buffer test vectors and maximise memory throughput. The allocation of memory blocks depends on the test vectors width and supported memory configuration of the FPGA. When the size of test vectors is too large to fit in on-chip memory, interface to off-chip data storage is provided and the on-chip memory remains as data buffer. The size of test vectors, denoted as s is:

$$s = n \times k \tag{3}$$

Figure 3 illustrates the customisation steps (step C in Figure 2) of the benchmarking unit. As an example, consider the following settings:

- SUT specification - The SUT is running at 400MHz. It is pipelined requiring input every 2 clock cycle, i.e.

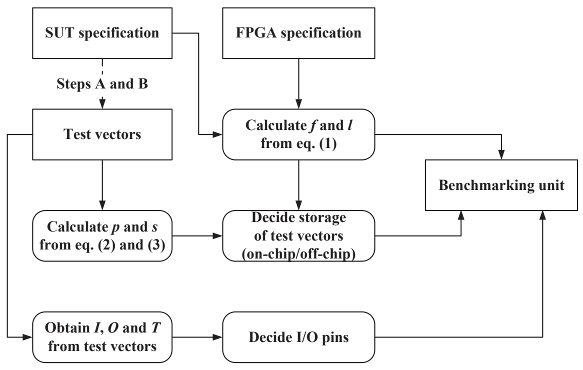
SUT specification
FPGA specification
Steps A and B
Test vectors
Calculate f and l from eq. (1)
Benchmarking unit
Calculate p and s from eq. (2) and (3)
Decide storage of test vectors (on-chip/off-chip)
Obtain I, O and T from test vectors
Decide I/O pins

Figure 3: Benchmarking unit customisation flow (Step C)

$t = 2/(400 * 10^6) = 5ns$. There are 1000 test vectors ($n = 1000$).

- FPGA specification - The maximum clock frequency of the benchmarking platform is 200MHz.
- After steps A and B - Timing information is represented in 32 bits ($T = 32$). Control signals occupy 2 bits ($C = 2$). Input data and output data are 32 bits each ($I = O = 32$). Therefore, $k = 32 + 2 + 32 + 32 = 98$.

Using equation 1, $f = 200$MHz and $l = 1$. According to equation 2 and 3, $p = 2.45$GB/s and $s = 12$KB.

As a result, the benchmarking unit needs to be clocked at 200MHz with one test vector fetched every clock cycle. 64 I/O pins are needed to connect the SUT and benchmarking unit. The total size of test vector is 12KB and the memory throughput should be at least 2.4GB/s. In this example, the size of test vectors is able to put in on-chip memory blocks of the FPGA. Two on-chip memory blocks with depth of 1000 are used, one has 64 bits data width for I/O data, the other has 32 bits data width for timing information.

3.2 Benchmarking

During benchmarking, the benchmarking unit automatically checks functional correctness in different real-time situations. Power consumption is monitored through an ADC connected to a current meter. It communicates with the SUT through an interface customised in step C of the pre-benchmarking stage (Figure 2).

The SUT is subject to the effect of system workload and environmental stimulus. For example, a computer system may be processing several processes simultaneously and affected by surrounding environment such as communication latency. The benchmarking unit reflects the effects of these issues on performance.

The benchmarking unit includes a system driver and a system monitor. The system driver fetches test vectors from the storage and sends test data to the SUT for required input clock cycles. At the same time, golden results, including expected output value and timing properties are sent to the monitor for verification.

The system monitor samples output values of the SUT and verifies the timing and functional correctness. The checked results are fed back and visualised on the host computer. Based on the feedback information, the SUT is debugged.

To improve benchmarking efficiency, the system monitor could send feedback to the system driver which would optimize the selection of test vectors being sent to the SUT. For example, regression testing could dynamically select the set of test vectors covering a particular area of interest. This allows the system to be tested more effectively because benchmarking could involve millions of test vectors. This part will be addressed in the future.

4. IMPLEMENTATION

In this section, the implementation of Roberts is given in detail.

4.1 Test Vector Generation

Extensible markup language (XML) is used for the test vector description which consists of timing and functional properties. The properties can be given explicitly by user or generated randomly in defined ranges.

We use SystemC and Verilog to describe the functional model of SUT and perform functional simulation to produce golden output values. Combining input events and golden output values, a test vector file is generated.

4.2 Benchmarking Unit

The benchmarking unit can be implemented on any FPGA device. The design of I/O pins and interface are easily customised and automated by scripts. For demonstration purpose, we use an Altera DE4 development board with a Stratix IV EP4SGX530 FPGA.

The maximum clock frequency of the benchmarking unit is 275MHz, which is generated by a phase-locked loop (PLL) from a 50MHz clock. An on-chip memory block is instantiated for storing test vectors. The memory block employs dual-port access, so when the size of test vectors scales, off-chip data storage is used and the on-chip memory block acts as a data buffer.

The resource and power overhead of the benchmarking unit is small. When the benchmarking unit is clocked at 167MHz with 162 bits test vectors, it uses 689 ALUTs (0.16%), 309 registers (0.073%) and draws 0.35W power.

The benchmarking unit and SUT are linked through the general purpose I/Os on the FPGA board. To simplify the evaluation setting, both the benchmarking unit and SUT reside on the same FPGA, but this is not a requirement of our approach.

Following timing specification, the system driver fetches test vectors from the on-chip memory, and sends data to the SUT. It also sends golden data and timestamps to a FIFO.

The system monitor is responsible for checking the timing and functional correctness of SUT. First, the monitor checks the FIFO in every clock cycle to see if there is any data waiting for checking. If so, it calculates the elapsed cycle. If the expected output is not obtained from the SUT within the required clock cycle, a deadline is missed and the monitor asserts a response time error.

The Altera's SignalTap II logic analyzer acts as a feedback unit to visualise and record signals captured by the benchmarking unit. It provides flexibility to debug the SUT according to specific system states and data values.

The power and energy consumption of the SUT are derived from the current measurement using an ADC on the DE4 development board. If such interface is not available, an external current meter can be used.

5. EVALUATION

This section illustrates how Roberts benchmarks real-time systems.

5.1 Benchmarking processor-based applications

We benchmark a SUT running on a NIOS II/e soft processor, which is clocked at 150MHz. The SNU real-time benchmark suite [18] is used as test programs running on the soft processor. It contains numeric and DSP algorithms written in C language for execution time analysis. We have selected several of them for benchmarking as listed in Table 3.

Table 3: SNU real-time benchmark programs

Benchmark	Description
sqrt	Square root function implemented by Taylor series.
fft1	FFT using Cooly-Turkey algorithm.
ifft1	Inverse FFT.
minver	Matrix inversion for 3x3 floating-point matrix.
qsort	Non-recursive quick sort on 20 floating-point numbers.

During benchmarking, test vectors are sent to the SUT sequentially in time interval of $100\mu s$, i.e. $t = 100\mu s$. The processor starts processing once all the data are obtained.

A test vector consists of 32 bits input data, 32 bits output data, 96 bits timing properties and 2 bits control signals, i.e. $k = 162$, $T = 96$, $C = 2$ and $I = O = 32$. For each benchmark program, 100 sets of test data are used such that the number of test vectors n varies from 100 to 2000.

Using equation 1, the benchmarking unit has 64 I/O pins. It is set to run at 100MHz and the test vector input rate is 10000 cycles. Using equation 2, the throughput requirement is 202.5KB/s. The maximum amount of test vectors is 40KB following equation 3. Considering the test data size and throughput requirement, all the test vectors are stored in on-chip memory of the FPGA.

Table 4 shows the average processing time of the benchmark programs measured using Roberts. For comparison, we also use a 32-bit timestamp interval timer available in NIOS II.

Both Roberts and the interval timer have measured similar response time. For Roberts clocked at 100MHz, the measurement precision is 10ns. For the interval timer, the measurement precision is 2 clock cycles which is 13.3ns. An offset of about $1\mu s$ is observed because our method has taken two things into account: 1) The communication time between the benchmarking platform and the SUT, such as interrupt latency; 2) The time that the interval timer itself spent on reading timer values.

Table 4: Average processing time of benchmark programs (Time is in μs)

	Roberts	Interval timer
sqrt	142.97	141.57
fft1	2254.37	2253.36
ifft1	2435.44	2434.45
minver	4.99	3.09
qsort	104.19	103.21

In summary, compared with the interval timer, Roberts has the following advantages:

- Not intrusive to the SUT - To read the interval timer's values, explicit read calls are added to the source code. The interval timer creates frequent interrupts which consume processor cycles at the beginning and at the end of measurement sections.
- Measurement of point-to-point latency - The interval timer is only able to measure a selected section of code running inside the system. Roberts takes interrupt latency and non-deterministic environment effects into account.

5.2 Benchmarking an FPGA application

A JPEG encoder IP core [19] is used to demonstrate Roberts in benchmarking high performance data processing application. For an input image, one input data represents the red, green, and blue pixel values and each pixel value is represented in 8 bits, so the input data bus is 24 bits wide. The encoder processes images block by block. Each 8x8 block of data needs to be input to the encoder on 64 consecutive clock cycles. The JPEG bitstream is output through a 32 bits bus.

The encoder is clocked at 167MHz, which is the maximum frequency recommended by the CAD tool. The encoder requires input every clock cycle, so $t = 1/(167*10^6) = 5.99ns$.

Combining I/O data, control and timing values, one test vector is 162 bits wide ($k = 162$), with $T = 96$, $C = 1$, $I = 26$ and $O = 39$.

Using equation 1, the benchmarking unit should run at 167MHz when the test vector input rate is 1 cycle. Equation 2 suggests that the minimum throughput is 3.34GB/s. The largest image has 55K pixels. Combining data and control signals, there are 57K test vectors ($n = 65K$) leading to 1.2MB test data, it is sufficient to store all test vectors in the on-chip memory of FPGA.

If the encoder is used in a camera to compress image or produce motion JPEG video, it is subject to real-time requirement. For example a camera needs to compress an image to JPEG in 0.5ms. We use Roberts to measure the processing time of the JPEG encoder versus different image sizes, as shown in Figure 4.

Roberts can be used to identify additional performance margin of the SUT. When the JPEG encoder is clocked at 167MHz, it fails to process an image in 0.5ms if the image size exceeds 55K pixels. However, when it is clocked at 175MHz, the encoder is able to meet the processing time requirement without any failure. The deployed system can run faster than the maximum frequency reported by static timing analysis of the CAD tool, because the tool is more conservative due to consideration of silicon, environment and data variability.

The dynamic power consumption of the encoder is 86mW based on the ADC readings. Figure 5 shows the energy consumption of the JPEG encoder. Since the power consumption of the encoder is consistent, the energy consumption increases linearly with the processing time.

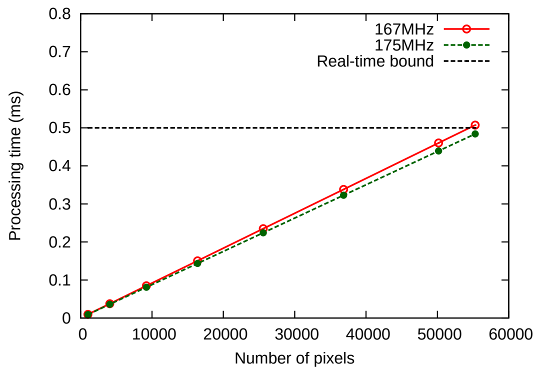

Figure 4: Processing time of JPEG encoder versus difference image sizes

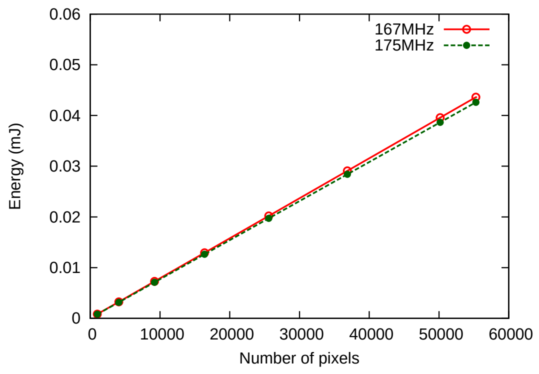

Figure 5: Energy consumption of JPEG encoder versus difference image sizes

6. CONCLUSION

This paper presents a benchmarking platform for real-time systems. The reconfigurability of FPGA allows a customisable benchmarking unit. Cycle-accurate measurement is performed to check timing and functional correctness. The evaluation platform on FPGA demonstrates its ability to capture SUT's point-to-point response time at nano-second resolution.

In the future, it would be of great interest to design a set of benchmark programs that target Roberts for evaluation of real-time systems. Further research includes test vector compression, dynamic test vector selection and extension of our approach to benchmark run-time reconfigurable designs. Lastly, the feedback data from benchmarking would also be useful in run-time frequency scaling [20] and error handling of real-time systems.

Acknowledgment

The support of the Croucher Foundation, UK EPSRC and Altera is gratefully acknowledged. The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreements number 248976, 257906 and 287804.

7. REFERENCES

[1] M. J. Mcgowan, "The rise of computerized high frequency trading: use and controversy," *Duke L. & Tech*, 2010.

[2] J. Stankovic, "Misconceptions about real-time computing: a serious problem for next-generation systems," *Computer*, vol. 21, no. 10, pp. 10–19, 1988.

[3] E. Mccluskey, "Built-in self-test techniques," *IEEE Design & Test of Computers*, vol. 2, no. 2, pp. 21–28, 1985.

[4] N. R. Storey, *Safety Critical Computer Systems.* Addison-Wesley Longman Publishing Co., Inc., 1996.

[5] C. Stroud *et al.*, "Built-in self-test of logic blocks in FPGAs (finally, a free lunch: BIST without overhead!)," in *Proc. IEEE VTS*, 1996, pp. 387–392.

[6] W. Chen and L. Shannon, "An on-chip testbed that emulates runtime traffic and reduces design verification time for FPGA designs," in *Proc. Int. Conf. on FPT*, 2008, pp. 361–364.

[7] B. Quinton and S. Wilton, "Post-silicon debug using programmable logic cores," in *Proc. Int. Conf. on FPT*, 2005, pp. 241–247.

[8] L. Shannon and P. Chow, "Using reconfigurability to achieve real-time profiling for hardware/software codesign," in *Proc. Int. Symp. on FPGA*, 2004, pp. 190–199.

[9] ChipScope Pro and the serial I/O toolkit. [Online]. Available: http://www.xilinx.com/tools/cspro.htm

[10] Design debugging using the SignalTap II logic analyzer. [Online]. Available: http://www.altera.com/literature/hb/qts/qts_qii53009.pdf

[11] Corus. [Online]. Available: http://www.veridae.com/index.php/products/corus-suite.html

[12] AN 391: Profiling Nios II Systems - Altera. [Online]. Available: http://www.altera.com/literature/an/an391.pdf

[13] J. M. Ellis, "Application specific automated test equipment system for testing integrated circuit devices in a native environment," US Patent 6 324 485, 1999.

[14] D. Gizopoulos, *Advances in Electronic Testing: Challenges and Methodologies (Frontiers in Electronic Testing).* Springer-Verlag, 2006.

[15] K. Gaj *et al.*, "ATHENa - Automated Tool for Hardware EvaluatioN: Toward Fair and Comprehensive Benchmarking of Cryptographic Hardware Using FPGAs," in *Proc. Int. Conf. on FPL*, 2010, pp. 414–421.

[16] P. Jamieson *et al.*, "Benchmarking and evaluating reconfigurable architectures targeting the mobile domain," *ACM Trans. on Design Automation of Electronic Systems*, vol. 15, pp. 14:1–14:24, 2010.

[17] N. Weiderman, "Hartstone: synthetic benchmark requirements for hard real-time applications," in *Proc. PIWG*, 1990, pp. 126–136.

[18] SNU real-time benchmarks. [Online]. Available: http://www.cprover.org/goto-cc/examples/snu.html

[19] JPEG Encoder Verilog. [Online]. Available: http://opencores.org/project,jpegencode

[20] J. Bower *et al.*, "Dynamic clock-frequencies for FPGAs," *Microprocessors and Microsystems*, vol. 30, no. 6, pp. 388–397, 2006.