

GroundHog 2009 – Energy Benchmarking for reconfigurable architectures on mobile computing devices

Executive Summary

GroundHog 2009 benchmark suite has been created to motivate and evaluate the energy consumption of reconfigurable architectures in the domain of mobile computing. For this reason a number of concepts have been introduced to make this possible that did not exist with current publicly available benchmarks in this domain.

The details contained in this document, additional specifications, and infrastructure tools are meant to assist in benchmarking reconfigurable devices in the mobile domain. This document and the additional materials should be used by individuals who understand how to create and map designs to reconfigurable architectures, and who have the ability to measure the power consumption of these devices when stimulated by inputs. The infrastructure tool is written in C, includes external libraries (that may need to be installed), and is tested in a Linux environment.

The key goals of this benchmark suite are:

- Allow mobile device manufacturers to evaluate reconfigurable architectures for their desired purposes.
- Motivate researchers to improve architectures and tools in the mobile device power domain and allow comparison of these innovations.
- Create designs that are simple, but representative of a range of designs targeting mobile devices. To achieve this, describe these designs in a technology independent format.
- Avoid potential drawbacks and argument over the benchmarks and implementations of the benchmarks where they are simply used to compare A vs B. Instead, we provide a defined relationship with reconfigurable architecture vendors and mobile device manufacturers, from which benchmarking is dealt with. The academic community will have access to these benchmarks and use them in a similar manner to pre-existing benchmark suites.

Figure 1 shows a view of the benchmark suite, and how this suite is used by benchmark users including the environment which defines the constraints on devices. The details provided in the GroundHog Benchmark suite provide the means for benchmarking these devices. The benchmark user, however, is expected to implement the designs on their devices, create a test bench to stimulate their designs based on provided workloads, and measure the energy consumption of their system. We have made an attempt to make GroundHog benchmark suite a guideline for this process and many of the details are left for agreement between the benchmark implementers and the mobile device manufacturers who define the environment in which reconfigurable technologies will operate in.

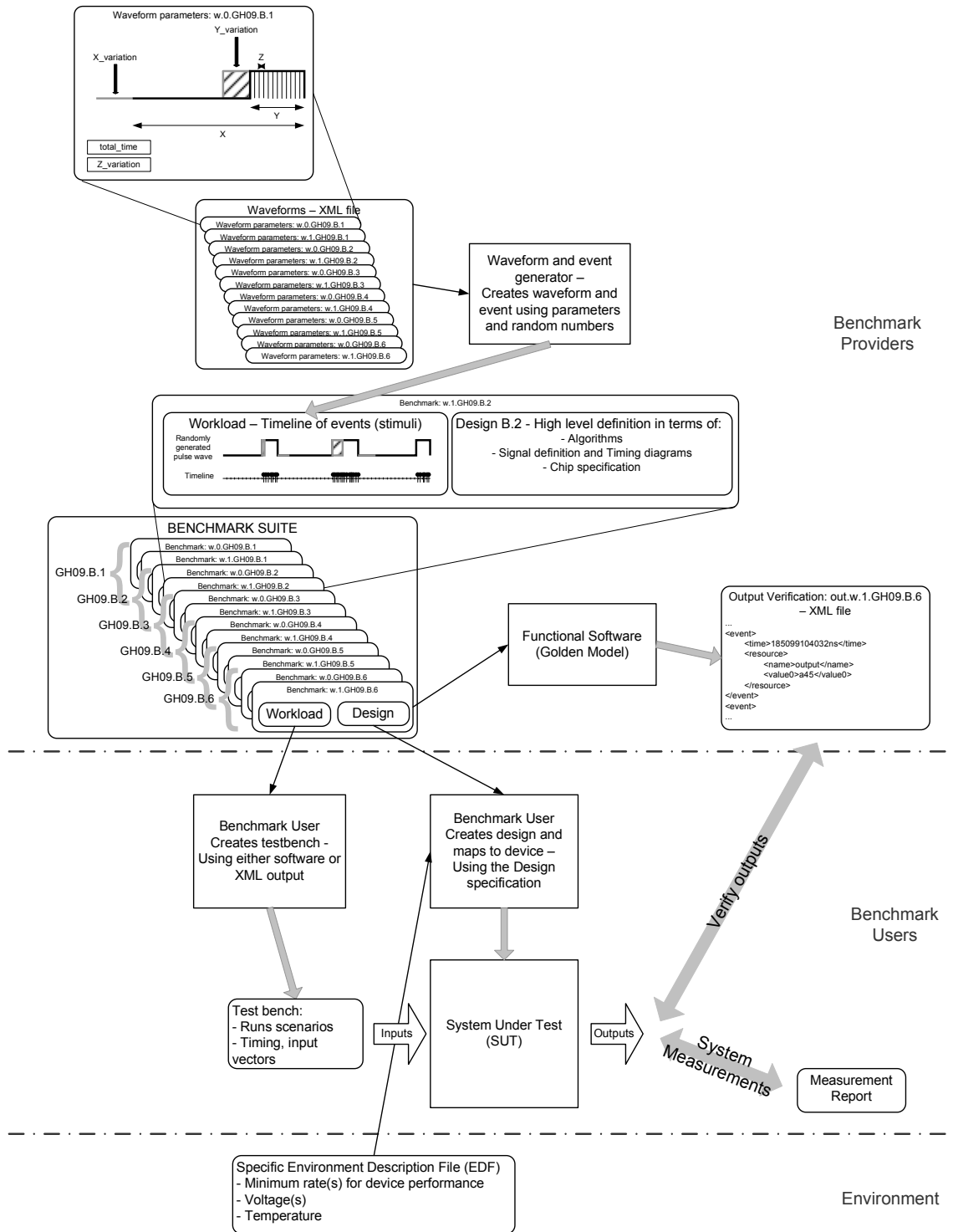


Figure 1 – In this figure the groundhog benchmark suite is shown in some detail, and we illustrate how there is a distinct separation between the benchmark, users, and environment.

Introduction

GroundHog 2009 benchmark suite has been created to motivate and evaluate reconfigurable architectures in the mobile computing domain. This document describes the elements of the benchmark suite and how they help in evaluating solutions targeting the mobile computing domain emphasizing the energy measurement of these solutions.

Benchmarking Philosophy

GroundHog 2009 does not simply allow the comparison of architecture A vs. architecture B for a set of benchmarks. The reasons for this are based on some of the key goals that this benchmark suite was created with. They are:

- Provide mobile device manufacturers with a standardized set of simple benchmarks to help them evaluate architectures and designs in terms of energy consumption. To do this, the architectures need to be constrained to the manufacturer's specifications.
- Motivate researchers to improve reconfigurable architectures and tools in the mobile device power domain and allow comparison of these innovations. Provide some freedom for this innovation in terms of constraints on their solutions.
- Allow the designs to be implemented on a range of target systems such as FPGAs, course grain reconfigurable architectures, and DSP processors. For this reason, the designs in the suite are simple, and represent of a range of designs that potentially could target mobile computation. These designs are described in a high-level technology independent format.

Our philosophy is to avoid some previous drawbacks of benchmarks where they are simply used to compare A vs B. Instead, we focus on providing a defined relationship with the people creating the system under test (SUT) and the people evaluating the SUT. This relationship is needed since we have focused on a broad range of potential architectures and target mobile devices, and we do not want to restrict the environment for innovation.

For the cases when a specific environment needs to be constrained, we allow constraints to be created between the system designer and the system evaluator. This means that the results generated from measuring the energy consumption of benchmarks on a particular device are relevant to the environment under which they are designed, and A to B comparisons is dependent on these defined constraints.

Definitions and Concepts

The following concepts and definitions will be used throughout our benchmark:

- Design – A design is a description of the problem to be solved by a computation device.
- System Under Test (SUT) – This is the architecture/device that implements the set of designs in the benchmark suite. The SUT takes in inputs and generates outputs, and it will be measured in terms of energy consumption.
- Workload – A description of what inputs will be sent to a SUT for a given design.
- Benchmark – The combination of a design and a workload. Note that there are multiple workloads for each design, and for this reason we don't call a design a benchmark (as is more traditional).
- Benchmark suite providers – This describes the creators of the benchmark suite and this document.
- Benchmark users – This describes the people that implement a design on a device which becomes the SUT, implement the system to send input stimuli, and benchmark the SUT.
- Benchmark implementers – Same as benchmark user.
- Environment description – This is a description of the environment under which the SUT is created. This can be used by mobile device manufacturers to describe the constraints under which the SUT must be used/designed. The environment can also specify some common constraints under which academics would like to design their SUT and compare their results.
- Environment description file (EDF)– This is an XML file containing the environment description.
- Environment specifiers – These are the people that provide the environment description. This concept allows for a relationship between the benchmark users and the environment specifiers that are completely independent of the benchmark suite providers.
- Energy consumption – This is the amount of energy that a SUT uses during the execution of a benchmark.

What is included in the GroundHog 2009 Benchmark Suite?

This benchmark suite is provided by the benchmark suite providers and includes the following:

- A fabric analysis benchmark for quickly evaluating the energy consumption of the logic fabric of a fine-grained FPGA.
 - This benchmark is called GH09.B.0 and is described further in the document “GH09.B.0.fabric_analysis_spec.pdf”
- Six high-level design descriptions and associated workloads that represent a range of applications and input stimuli that might be used within the mobile computing domain. The six benchmarks are:
 - GH09.B.1.portkey - Port Expander and Keypad Controller
 - GH09.B.2.glue - Glue logic consisting of state machines that control counters
 - GH09.B.3.crypto - Cryptography using a simple AES encryption algorithm
 - GH09.B.4.compress - Data compression using a simple Lziv algorithm
 - GH09.B.5.bridge – A bridge chip that converts requests from masters on a parallel bus for a slave on a serial bus
 - GH09.B.6.2dcon – A generalized 2D convolution algorithm
- A software tool that creates the workloads and outputs these workloads in an XML file. These workloads in either XML form or using internal data structures can be used to create a test bench for the SUT.
- A software tool that simulates the behaviour of a benchmark, and outputs an XML file that details the output response of the SUT for the respective workload. This tool is used to verify the correct behaviour of the SUT.
- Documentation (including this document) that explain this benchmark suite and the included tools.

What is a benchmark user expected to do?

As already described, the GroundHog 2009 benchmark suite does not provide synthesizable or compile ready designs. This means that benchmark users will need to create and map the six designs onto the SUT that they will measure energy consumption of. During this creation process, the benchmark user is expected to use the environment description (provided either externally or by themselves) as the set of constraints on their system. For example, the environment description file might specify that the main clock speed that the SUT will be used is 100 MHz.

In addition to this mapping process, the workloads need to be fed into the SUT. The benchmark users, therefore, have to build some test system that uses the information in a workload and converts it into a form that can be inputted to their SUT.

The benchmark user is expected to verify that their SUT is performing correctly. Once the SUT is correctly executing a specified design with its respective workload and constrained by the environment description, then the SUT can be measured for energy consumption over the execution of a complete workload.

The reality is that the benchmark user is expected to do many tasks to get measurements from their chosen device. We have made an effort to specify designs that are simple, and these designs should be easily developed into synthesizable and compile ready designs. Over time, it is expected that there will be a repository of these designs that will provide for good examples and possibly complete solutions for design specifications.

Structure of this distribution

We briefly describe the directory structure of the GroundHog benchmark suite distribution:

- GROUNDHOG_DISTRIBUTION
 - groundhog_09_meta_document.pdf – this file
 - workload_description.pdf – a file describing the xml tags in a workload
 - ENVIRONMENT_FILES – contains sample environment files
 - 32MHz_environment_sample.xml
 - 100MHz_environment_sample.xml
 - BENCHMARK – this is the top level for the benchmark details.
 - GH09.B.0
 - Design spec
 - HDL – container for the HDL files
 - GH09.B.1.portkey
 - GH09.B.1.portkey_spec_sheet.pdf
 - GH09.B.1.portkey_workload_desc.pdf
 - GH09.B.2.glue
 - GH09.B.2.glue_spec_sheet.pdf
 - GH09.B.2.glue_workload_desc.pdf
 - GH09.B.3.crypto
 - RELATED_DOCUMENTS
 - fips-197.pdf
 - GH09.B.3.crypto_spec_sheet.pdf
 - GH09.B.3.crypto_workload_desc.pdf
 - GH09.B.4.compress

- GH09.B.4.compress_spec_sheet.pdf
 - GH09.B.4.compress_workload_desc.pdf
- GH09.B.5.bridge
 - RELATED_DOCUMENTS
 - wishbone_spec_b3.pdf
 - GH09.B.5.bridge_spec_sheet.pdf
 - GH09.B.5.bridge_workload_desc.pdf
- GH09.B.6.2dcon
 - GH09.B.6.2dcon_spec_sheet.pdf
 - GH09.B.6.2dcon_workload_desc.pdf
- BENCHMARK_INFRASTRUTURE – this is the top level container for all the tools included in the benchmark distribution.
 - GROUNDHOG_INFRASTRUCTURE_TOOL – details of directory not expanded please see README.txt and further documentation

Methods and Setup

In this section, we describe more detailed description of what is included in the benchmark suite and how to use the benchmark suite. Note that this description is a guideline for using the benchmark suite. We do not take the stand that there are explicit rules that must be satisfied to create a proper benchmark report. Instead, the relationship between the benchmark user and environment specifiers (for example, an FPGA manufacturer and a cell phone manufacturer) will determine specific rules that must be adhered to when benchmarking a SUT. Instead, we provide an infrastructure and basic guidelines on how to perform benchmarking.

We expect that the benchmark suite will be used by benchmark users while maintaining scientific rigour. This means that assumptions and design decisions should be reported in addition to the measurements of their respective SUT.

How to create a benchmark?

A benchmark is an implementation of a design on a SUT executed with a workload. This definition of benchmark as a combination of workload and synthesizable design differs from traditional notions of benchmarks. In this section, we will describe what a design description includes, how to interpret these design specifications, what a workload is, and how to use a workload to create a test bench.

Design specifications

Each of the six design specification, GH09.B.1 – GH09.B.6, is described in a similar fashion. We model specifications from chip specifications where the document attempts to describe how a chip operates. Unfortunately, the six benchmarks have different characteristics from one another, and we can not use one common method of describing the behaviour for all the specifications. For this reason, each specification includes a mixture of signal descriptions, algorithmic descriptions, citations to standardized descriptions (which are also included in the distribution), and written descriptions.

Each design specification includes the following sections:

- General description – this section gives a high-level view of what the design is.
- Features – this section describes key features of the design including items such as bit-widths, algorithmic standards, and protocol standards.
- Block diagram – this is a high-level view of the design implemented as a self contained block in a system. The inputs and outputs described here are a representation of needed inputs and outputs to implement a logical view of the design.

- Details – this section describes details about the design.

Using this design specification, the benchmark users can implement the design on their chosen device. The question, however, remains in what detail must the implementation meet the specification.

In terms of operational constraints such as following a standard that defines algorithmic behaviour, it is expected that the design will follow these standards exactly. For example, GH09.B.3.crypto is meant to specify the AES-128 cipher. The implementation needs to follow this standard strictly for the defined bit-widths as specified.

The input and output standards described in the design specifications are logical views of transferring signals in and out of the SUT. This logical view aligns with the workloads, but when these designs are implemented, it is expected that more layers of input output may be layered on top of the logical view. For example, modern FPGAs include high-speed serial transmission pins. A design implemented on these devices may use this capability to transmit data, which then should obey the protocols described within the design specifications.

The design specifications do not include all signals that may be needed to implement the design in a SUT. For example, all workloads include a reset event, but no design specification mentions the need for a reset signal. It is expected that additional signals will be included in implementations of the designs. Additional signals may be introduced by the relationship between the benchmark user and the environment specifiers, or it is possible additional signals may be introduced to allow for some sort of optimization (for example a communication port with a supervisor sensor). In all cases, these details should be reported, and any energy measurements need to include energy consumed by these signals.

Workloads

Workloads are descriptions of the input stimuli sent to a SUT implementing a specific design specification. As described earlier the combination of a workload and an implemented design specification forms a benchmark.

Workloads are built based on events where an event is a time and associated action(s). Therefore, a workload is a series of times and associated actions. The times and associated actions in a workload are sorted sequentially in time starting from the earliest event and sequentially continuing to the last event.

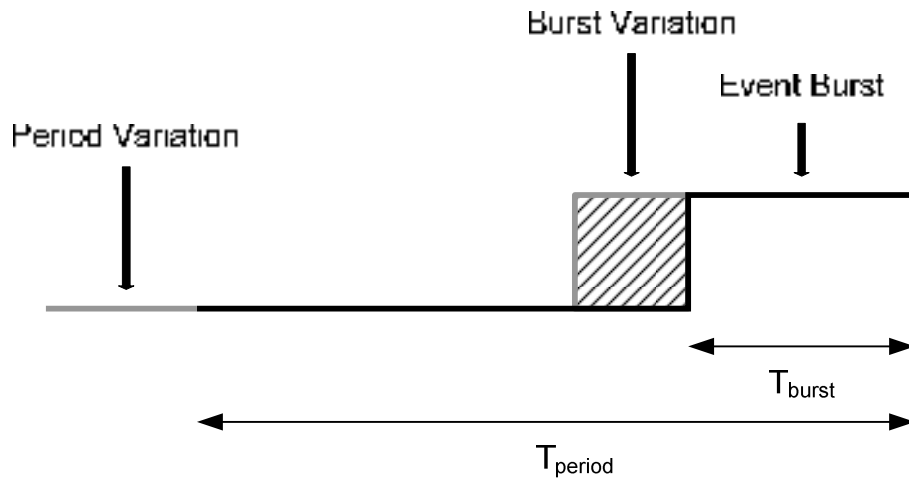


Figure 2 – A view of one period of a randomly generated pulse wave and parameters describing that wave.

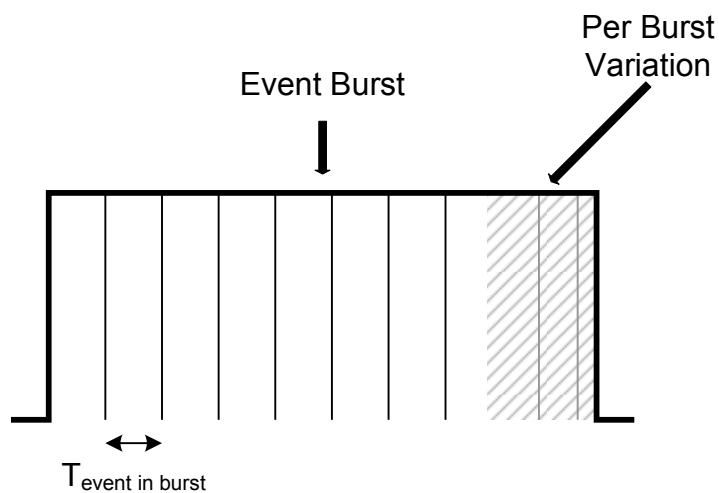


Figure 3 – A zoomed in view of the event burst where the vertical lines represent an event.

The workloads within the GroundHog 2009 benchmark suite are created synthetically based on parameters that describe a pulse wave. These parameters are shown in Figure 2 and Figure 3, where an event burst (T_{burst}) is the time in a pulse wave in which events happen every $T_{\text{event_burst}}$ time units. Each burst occurs at a period defined by the parameter T_{period} .

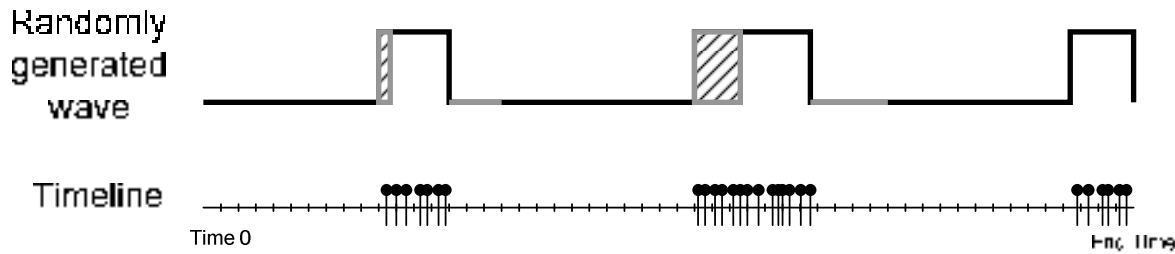


Figure 4 – An example of a randomly generated wave and what a workload timeline might look like showing where the events happen.

Given these wave parameters, parameters for random time deviation in the wave, total time of workload execution, and a random model for what input events happen, we generate a sequence of events that represent the workload. Figure 4 represents a view of a synthetically generated wave and events on a timeline.

For the GroundHog benchmark suite, we have created a set of workloads defined by a default waveform file. This default set of waveform parameters is replicated in a workload description file that accompanies each design description.

An infrastructure tool is provided in this benchmark release that takes the default waveform file (that includes the defined waveforms) and can create an XML output file representing the events. These output files can be restrictively large, and for this reason, we have supplied the tool as open source so that the test bench developers can use this workload generation tool itself to create their test benches. In either case, the XML file or internal data structures representing a workload need to be converted into a form that will feed input stimuli to the SUT.

The test bench designer needs to understand the events in the workload and what they mean. Workload events can either be *macro events* that represent a combination of stimuli (either sequential or parallel) or events can be *simple signal events* where the event explicitly names a signal and a corresponding value. The events associated with a workload are explained in more depth in the associated workload description files included in the directories containing the design specification.

Time for each event is defined as the point on the timeline at which an event starts to happen. The duration of the event is dependent on if this is a *macro event* or not. In most cases the benchmark users will be able to determine how time and event are related. In those cases where this is not true, we make an attempt in the workload descriptions to explain what happens.

Note that the workloads are generated based on arrival rate information that is provided from the environment description. This means that the event times are aligned to arrival rates defined within this file. This means that these details are provided by the environment specifiers.

How to measure a benchmark?

As described above, a benchmark is a design implemented on a chosen device and that SUT then stimulated by a respective workload as a test bench. It is expected that each benchmark will then be measured in terms of energy consumption. The question then is what is involved in this measurement.

The complexities of measuring power for the wide variety of devices available mean that once again these details are left to be decided by the relationship between the benchmark users and the environment specifiers. There are, however, a few guidelines to follow when measuring the SUT.

The following is a guideline of what needs to be done:

1. It is expected that the correct execution of a workload will be verified.
 - a. To help achieve this we have included a golden functional model in our benchmark tools that provides an XML output with the expected outputs of a SUT for a given workload input. This output file is a guideline of expected outputs with annotated time estimation. This file, however, does not consider all timing conditions and scenarios, and in some cases ordering of events is arbitrarily chosen. For these reasons, this tool is to be used as a guideline of the expected outputs.
2. Any measurements for a SUT should provide additional details about design criteria, optimizations, and system details. In other words, details of the system should be openly provided between the benchmark users and the environment specifiers. In the case of academia, this allows for the reproduction and verification of the measured results.
3. Benchmark measurements for energy consumption should be done for the complete execution of the workload and include a graph of power consumption throughout that execution. If, for some reason, a benchmark is not executed it should be reported with a reason. The workloads have been created to stress designs in some cases, and therefore, it is expected that SUTs will not be able to meet the demands of all the workloads.
4. Depending on the focus of the measurements, it is not expected that all designs specs and workloads will be measured. For example, a device manufacturer and environment specifier may not care about the cryptography design (GH09.B.3.crypto). For this reason, there is no need for these measurements.

Environment Details

The environment in our benchmark suite setup describes the constraints under which designs operate on the chosen devices. This environment allows a disconnection between the benchmark suite and the benchmark implementations. This is done so that relationships between the benchmark implementers and environment specifiers can be independently defined between these two groups.

To help implement this relationship we have created an Environment Description File. This is, currently, an XML file that provides a description of the environment in an electronically readable format. We have, however, not provided an in depth specification for this file. Instead, the environment description file included with this release is called “32MHz_environment_example.xml” and “100MHz_environment_example.xml”.

The “32MHz_environment_example.xml” file is replicated below:

```
<environment>
  <minimum_operating_speed>32ns</minimum_operating_speed>
  <!-- can be interpreted as fast clock (Hz) ... 32MHz -->
  <minimum_sampling_speed>31250ns</minimum_sampling_speed>
  <!-- can be interpreted as slow clock (Hz) (the heartbeat) .. 32 KHz -->
  <minimum_arrival_rate_on_serial_interfaces>32000ns</minimum_arrival_rate_on_serial_interfaces>
  <!-- can relate to the serial clock -->
  <minimum_arrival_rate_on_parallel_interfaces>32000ns</minimum_arrival_rate_on_parallel_interfaces>
  <!-- can relate to the parallel clock -->
</environment>
```

The basic xml tags that are needed within the environment file to work with the workload generation tool are defined as:

- `minimum_operating_speed` – this parameter is defined in terms of a time and specifies the operating speed of the device. If interpreted as a clock speed, then the frequency is $1/\text{time}$. However, we have not made this specification since the implementation could be asynchronous or in another format.
- `minimum_sampling_speed` – this parameter defines the sampling rate of the device. This is the heartbeat of a device. This can be thought of as the sampling rate when the device goes into standby mode or other power saving mode.
- `minimum_arrival_rate_on_serial_interfaces` – this parameter defines the rate of any serial interfaces.
- `minimum_arrival_rate_on_parallel_interfaces` – this parameter defines the rate of any parallel interfaces.

These parameters are defined by the environment and are passed to the provided tool infrastructure. This information is used and included in the workload. Depending on the relationship between benchmark users and the environment specifiers, it is possible that these

parameters will be overridden and additional parameters are specified. These additional parameters can include items such as external resources available to be used, I/O restrictions, available voltage rails, etc. In other words, the EDF files included with this benchmark suite are a basic guideline of what an EDF could include.

Note that these additional details if included in the EDF are not supported by the supplied infrastructure tools, and if some of the new factors affect the tools then changes will need to be made in the tools by external users.

Infrastructure Tools Details

The infrastructure tools provided with this benchmark suite are bundled in one piece of software. This software is open source to allow individuals to customize this tool for their specific implementation details. The software is documented separately. Please refer to the README.txt files in the GROUNDHOG_INFRASTRUCTURE_TOOL directory for more details.

At present this software tool does two things:

1. One tool creates workloads based on provided parameters. These can be outputted to XML or used internally as a data structure. Due to the large XML files that can be generated, it is recommended that the internal data structures in the tool are used to create test benches as opposed to parsing XML files externally.
2. One tool Uses workloads to simulate the behaviour of a benchmark and output an XML file that can be used as a golden model to verify a SUT running a particular workload.

These tools have been developed in a Linux environment. We have made an effort to describe how to build these tools.

Reporting Results

As specified throughout this document, the GroundHog benchmark suite has been created to help stimulate and evaluate solutions for reconfigurable devices for the mobile domain. There is no need to make strict regulations on what details should be reported for the benchmarking since we expect this benchmark suite to be used in a variety of scenarios.

In general, we expect that reports will attempt to describe and evaluate the SUT in as rigorous a form necessary for proper evaluation by the concerned communities.