Benchmarking Reconfigurable Architectures in the Mobile Domain

P. Jamieson, T. Becker, W. Luk
Department of Computing, Imperial College

P. Y. K. Cheung
Department of EEE, Imperial College

T. Rissa
Nokia Devices R&D

T. Pitkänen
Tampere University of Technology

Abstract

In this paper, we introduce GroundHog 2009 benchmarking suite that can be used to evaluate the power consumption of reconfigurable technology implementing applications targeting the mobile computing domain. This benchmark suite includes seven designs; one design targets fine-grained FPGA fabrics, and six designs are specified at a high level, which allows them to target a range of reconfigurable technologies. Each of the six designs can be stimulated with synthetically generated input stimuli created by a tool included in the suite. Additionally, another tool can help verify the correctness of each implemented design. Finally, we use our benchmark suite to evaluate the power consumption of two modern FPGAs targeting the mobile domain.

1 Introduction

Reconfigurable architectures would benefit the production of mobile computation devices by allowing late design changes, simplifying the logistics of creating these devices around the world, and by reducing the creation of dedicated ASICs for applications in these devices. At present, there are proposed applications of this technology in the mobile domain [10, 14] since reconfigurable architectures can meet the speed requirements and provide possible power and performance benefits. However, we are not aware of widespread commercial adoption of reconfigurable architectures for mobile applications, perhaps due to the absence of appropriate benchmarks specifically developed for such architectures and applications.

GroundHog 2009 is a benchmark suite that has been created to help motivate optimisations in and measure power consumption of reconfigurable architectures for the mobile domain. This benchmark suite includes 7 designs in which one design provides a worst case fabric analysis of fine-grain FPGAs and the other 6 are more general mobile applications. In addition to these designs, GroundHog 2009 includes two tools. The first tool helps create input stimuli for test-benches to evaluate each of the six applications when mapped to a target architecture. The second tool helps verify the correct operation of each implemented design. Moreover, these tools can be extended to cover new benchmarks.

The challenges with creating such a benchmark that we address in Section 3.1 include:

- To provide design descriptions that allow targeting of multiple architectures.
- To include input stimuli that represent execution instances of the design used in a mobile computation environment. This includes workload scenarios that are not simply a full throughput mode, but include instances when the input stimuli are dormant and the architecture can enter low power modes.
- To select designs that represent potential applications in a mobile device.
- To provide a methodology for measuring and reporting results.

In this paper, we describe GroundHog 2009 Benchmark suite based on these described challenges. Our approach has made an extensive study of previous benchmark suites [4] for computational devices. We use ideas from these previous efforts and create GroundHog 2009 using a combination of these techniques and some of our own ideas.

GroundHog 2009 does not include all parts of what traditionally constitutes a benchmark suite, and instead, allows these aspects to be defined externally. This paradigm shift allows GroundHog 2009 to have the flexibility to target both existing and future architectures and systems, which satisfies our ultimate goal of motivating and facilitating power improvements in reconfigurable architectures so that they will be adopted in the mobile market.

To demonstrate the benefits of this benchmark suite, we experimentally show how two benchmarks from the

GroundHog 2009 can be mapped to existing FPGAs and measured for power consumption for our defined input stimuli. From these results, we illuminate where simple optimisations may be made in the future, and we show how two different architectures can be preliminarily compared.

The remainder of this paper is organised as follow: Section 2 reviews related work. Section 3 describes our benchmark suite in detail; this description includes the contents of the benchmark suite, how designs are specified, how input stimuli are created, and how the environment for a system is described. Section 4 describes the concept of relationships in benchmarking, and how GroundHog 2009 benchmarking suite leverages this concept to make the suite flexible for a wide variety of possibilities. Section 5 describes the measurements we have made for two existing low power FPGAs, and finally, Section 6 concludes the paper.

2 Related Work

Evaluating and benchmarking reconfigurable architectures for power has been achieved by using existing benchmarks, such as MCNC [21], or by modelling power consumption using circuit models and in house designs. We, briefly, discuss this previous work.

One area of research involves reconfiguration as a power reduction technique. For example, Liang *et. al.* [11], Noguera *et. al.* [13], and Burleson *et. al.* [7] build specific instances of an application on a reconfigurable architecture and optimise these implementations for power consumption. In related work, Shang *et. al.* [16] show the dynamic power consumption of a Xilinx Virtex-II FPGA [20] using an internal benchmark. This internal benchmark includes input stimuli, which they use to calculate the switching activity of a real design. Tuan *et. al.* present a low-power FPGA core with several optimisations such as voltage scaling, leakage reduction of configuration memory cells and power gating of tiles with preservation of state and configuration [19].

The MCNC benchmark suite provides a range of simple test circuits and is often used in power-aware research on reconfigurable architectures. Poon *et. al.* [15] use the MCNC benchmark and add power models of a common FPGA architecture exploration tool, VPR [6]. They use the MCNC benchmark suite to find a transition density signal model to estimate the activity within each logic cell of an FPGA. Anderson *et. al.* [3] also use the MCNC benchmarks in their work to estimate power consumption in FPGAs. Gayasen *et. al.* propose a scheme with two programmable supply voltages where the higher voltage is used for critical path logic and the lower voltage for non-critical parts [8]. Using MCNC circuits, they achieve an average power saving of 61%. More recently, Tinmaung *et. al.* [18] optimise for power on FPGAs during logic synthesis and use the MCNC benchmarks to perform measurements of their optimisations.

Much of the previous work allows for power measurements on reconfigurable architectures, but they do not realistically model modern applications on these devices. This is especially true for the mobile computation domain where there is increasing computation demands and limited advances in battery capacity. Energy or average power are relevant in the context of battery capacity, while peak power has to be considered for thermal aspects. In addition, there are no benchmark suites in existence that contain a set of designs that would likely be implemented on reconfigurable architectures in the mobile domain both in the near and far future. As for benchmarks, such as MCNC, there are no input stimuli and they are implemented in low level design descriptions. For this reason, we have created GroundHog 2009 to fill this gap.

3 GroundHog Benchmarks and Tools

At present, most researchers agree that it will be challenging for reconfigurable architectures to be included in mass-market mobile devices even with the benefits of flexibility of design. The limiting factor for this adoption is power consumption, cost, and lack of power mode support (where power modes allow a device to go into low-power states when not used).

GroundHog 2009 is a benchmarking suite that is meant to target reconfigurable architectures in the mobile computation domain with the goal of providing the means to measure innovation in this field so that someday reconfigurable architectures are adopted. There are a number of challenges in creating this benchmark to meet the following goals:

- Collecting realistic (open access) designs that would be used in mobile devices and future mobile devices (discussed in Section 3.2).
- Allow the benchmarks to be mapped to the wide range of reconfigurable architectures, which include FPGAs, CPLDs, coarse-grain architectures, multi-core systems, and even, microprocessors (discussed in Section 3.2).
- Stimulate the designs with actions a system will likely perform in present mobile devices and future mobile devices (discussed in Section 3.3).
- Create a methodology in which the wide variety of technologies in mobile devices can be described so that architectures can be designed to target these specific instances (discussed in Section 3.4 and 4).
- Prevent system or tool optimisations for a specific benchmark, while still encouraging innovation (discussed in Section 4).

We have created GroundHog 2009 as a first attempt to satisfy these challenges. There are four main elements of the benchmark suite that, we believe, makeup this innovative framework and satisfy many of the earlier challenges described. They are:

1. Provide high-level design descriptions
2. Provide synthetically generated, parametrisable input stimuli
3. Allow the environment to be uniquely specified
4. Allow early baseline fabric analysis of fine-grained FPGAs

In this paper, we do not focus on item four, and we direct the reader to our previous work [5]. However, note that fabric analysis is included in the benchmark suite to allow our community to quickly evaluate the power state of FPGAs as they represent the most mature technology that potentially would be included in mobile systems. The remaining three items are described in Sections 3.2 to 3.4. First, we provide a description of what is contained in GroundHog 2009 benchmark suite.

3.1 GroundHog 2009 Benchmark Suite

GroundHog 2009 consists of seven designs and accompanying infrastructure that allows a benchmark user to create input stimuli for these designs and to verify their implementations against a golden model.

The seven designs are:

- GH09.B.0 - Fabric analysis
- GH09.B.1 - Port expander and keypad controller
- GH09.B.2 - Glue logic
- GH09.B.3 - AES encryption cypher
- GH09.B.4 - Data compression using Lempel-Ziv
- GH09.B.5 - Bridge chip
- GH09.B.6 - 2D convolution

Excluding GH09.B.0, these are designs that could be implemented on reconfigurable architectures as part of a mobile system. These designs were selected because they are simple, but represent design qualities of a range of possible designs. For example, the 2D convolution design exercises a technology's ability to implement arithmetic operations, and the data compression design exercises a technologies memory capabilities with pseudo random access patterns. In Section 3.2, we describe more details on how these designs are specified.

In addition to the designs, we have also included software tools to aid the benchmark users in building a measurement framework for their implementations. The tools allow benchmark users to create input stimuli to evaluate their solutions. This stimuli can be created to model classic throughput like inputs as well as intermittent stimuli that more closely models the on/off activity within a mobile device, and this is discussed in Section 3.3. We have made these tools open-source so that it can be modified to output the input stimuli in a format that can be leveraged to fit into the benchmark user's measurement framework. For example, in our setup (described in Section 5.1 the input stimuli are converted to a set of vectors and timestamps that are then read by an external FPGA board. This FPGA board is hooked up to the implementation of a design and feeds the input stimuli to the design so that power measurements can be made.

An included tool also provides a golden model simulator to help benchmark users verify correctness of their implementations. In this way, benchmark users can look at the software emulation of each of the six designs and analyse the behaviour of their implementation for a given input stimuli. This helps in both understanding the expected behaviour of a benchmark design and verifying if the implemented version on a reconfigurable architecture is performing correctly.

Finally, GroundHog 2009 includes sample environment descriptions. These environment descriptions allow solutions to target a range of mobile devices and this is discussed in more detail in Section 3.4.

3.2 Design Descriptions

The designs in GroundHog 2009 are meant to target a wide variety of reconfigurable architectures. These architectures include devices such as FPGAs, CPLDs, coarse-grain architectures, multi-core systems, and microprocessors. This broad range of targets makes it difficult to describe designs in a form that is mapable to all these devices. For example, a design written in a hardware description language (HDL) does not map well to processors, and similarly, a design written in a sequential language, such as C, does not map well to hardware devices.

In addition to the challenge of mapping designs to a range of devices, the choice of a specific design language can result in design decisions. For example, if we chose to use an HDL to describe our designs, then design decisions are prematurely made that may map well to an FPGA but not necessarily to a coarse-grain architecture. If a design uses a multiply-accumulator (MAC) then should it be described in terms of a high-level multiply-accumulate (which doesn't exist as a primitive in Verilog or VHDL), or is the MAC better described as a combination of adder and multi-

plier? It is not clear what is the most appropriate design for such a structure for a range of architectures, and therefore, choosing a design language conflicts with our overall goal of motivating innovation.

For the above reasons, designs within the GroundHog 2009 benchmark suite are described in a high-level format, which is a similar approach taken by SLALOM [9] benchmark suites. Our high-level format design description includes a description of the design, a block diagram of the logical view of the application, and a detailed description of the application in the form of algorithmic descriptions, protocol descriptions, signal descriptions, citations to standardised descriptions, written descriptions, or a mixture of all five. For the sake of space, we have not included an example here, but design specifications can be viewed by downloading the benchmark suite.

The benefit of the high-level approach is benchmark users can map the designs to any target architecture, but to achieve this, the benchmark user needs to make a synthesisable version of each design. For this reason, the six designs in GroundHog 2009 were picked based on how common and simple these designs are.

The six designs have also been chosen as representative designs for mobile applications based on a set of characteristics that includes:

- Bit-width - The width of the operations varying between bit-level and word-level operations.
- Processing flow - This is the type of flow in the computation where data dependency between present and past data results in a varying delay in output is classified *control flow as opposed to* data flow.
- Memory usage - This characterises a design based on how it uses memory. This can either be *simple state and shift registers or more* complex random memory accesses.
- Arithmetic complexity - This defines a design based on the computation structures used where a design that uses operations such as division, multiplication, and more complex math functions would be considered complex.
- Performance requirements - The expected speed at which the outputs need to be generated at.

Table 1 shows each of the six benchmarks classified based on the design characteristics. Column 1 contains the benchmark name, and Columns 2 through 6 contain each of the design characteristics. From this characterisation we can see that the benchmarks have been chosen to differ in at least one characteristic from each other. This does not cover the complete set of possibilities, which may be covered in later releases of the benchmark.

3.3 Input Stimuli

One of the missing aspects of existing benchmarks suite, especially those that are to be used in benchmarking circuit level designs, is the inclusion of input stimuli. Input stimuli are not needed for all experiments, but when targeting power measurements in the mobile domain, input stimuli are necessary for two reasons. The first reason is, though there are existing methods to estimate power consumption, the most realistic method is to measure power while a device is executing real input stimuli. The second reason is that applications within a mobile system will execute at varying rates. This means that an application ranges from executing in full throttle mode to not executing at all, and for this reason, power modes are used in mobile devices, which puts parts of the system in different power consuming states.

GroundHog 2009 benchmark suite includes a tool to synthetically generate input stimuli. Based on a set of parameters, we can generate a time-line of input stimuli for a particular design, and we call these time-line of events, workloads.

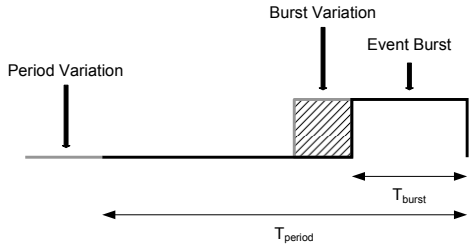

Figure 1. Parameters for the pulse wave

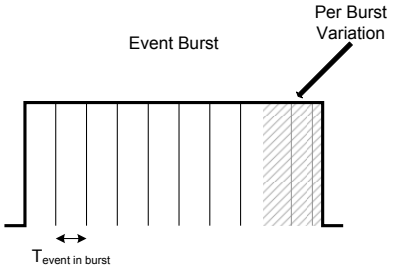

Figure 2. Parameters for events in a burst

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 1 and Figure 2, where an event burst T_{burst} is the burst time wave in which events happen every $T_{event\_in\_burst}$ time units. Each burst occurs within a period defined by the parameter T_{period}.

Table 1. The characteristics of designs in our benchmark suite.

Design Name	Bit-Width	Processing Flow	Memory Usage	Arithmetic Complexity	Performance Requirements
GH09.B.1 - Port expander and keypad controller	90% bit-level	Control	Simple	Simple	Low
GH09.B.2 - Glue logic	90% bit-level	Control	Simple	Simple	Low and High
GH09.B.3 - AES encryption cypher	90% bit-level	Data	Simple	Simple	High
GH09.B.4 - Data compression using Lempel-Ziv	90% bit-level	Control	Random Access	Simple	High
GH09.B.5 - Bridge chip	bits and bytes	Data	Simple	Simple	High
GH09.B.6 - 2d convolution	90% byte-level	Data	Simple	Complex	High

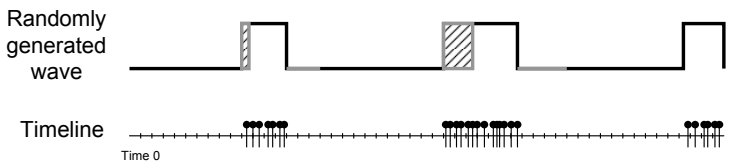

Figure 3. Example of what a time-line of events would look like for a parametrised wave

Given these wave parameters, and parameters for random deviations in the pulse wave, total time of workload execution, and a random model for what input events happen, we can generate a sequence of events. Figure 3 represents a view of a synthetically generated wave and events on a time-line. Based on this method of synthetically generating workloads it is possible to create workloads representing a range of design execution instances.

Workloads are created within the provided tool. Events on the time-line represent input actions that are as simple as an input signal changing to as complex as a packet of information being sent on a port. We output the workloads to XML, but benchmark users are expected to use the tool to create their own test-benches to stimulate their target architecture. The drawback with this approach is that the benchmark user must convert high-level descriptions of the input stimuli into a form usable by their measurement setup.

To help benchmark users in this process, we provide a synthetic workload generator as an open source tool. This means that the workloads can be modified and outputted in a form that compatible with their chosen setup. For example, if the benchmark user is targeting an FPGA platform, then the workload generation tool can be modified to output test vectors directly or via another system to the FPGA under test.

3.4 Environment Description

GroundHog 2009 benchmark suite is meant to push innovation of reconfigurable architectures in the mobile domain. The problem is this mobile domain includes a vast array of devices where each device is built under a range of constraints and technologies. For this reason, our benchmark suite includes a concept called environment specification. This specification in the form of an environment description file (EDF) allows institutions, external to the benchmark specifiers and benchmark users, to describe mobile devices. The goal of this description is to allow institutions to define the constraints under which reconfigurable architectures can be benchmarked to satisfy the institution's needs.

In GroundHog 2009 there are two example EDFs that describe a minimum set of parameters needed for a simple benchmarking environment. These parameters include:

- 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/time. However, we have not made this specification since the implementation could be asynchronous.
- minimum_sampling_speed this parameter defines the sampling rate of the device. This is also known as the heartbeat of a device that is present to receive messages from the system. This can be thought of as the sampling rate when the device goes into power saving modes such as standby mode or other power saving mode.
- minimum_arrival_rate_on_serial_interfaces this parameter defines the rate of general serial interfaces.
- minimum_arrival_rate_on_parallel_interfaces this parameter defines the rate of general parallel interfaces.

GroundHog 2009 includes EDF files that describe a 32MHz clock and 32KHz heartbeat clock, and a 100MHz and 32KHz heartbeat clock for synchronous devices, which are typical clocks for mobile devices. These files are distributed in XML form, and the tools included with the suite can read this EDF as it is used in creating workloads. There is a range

of possibilities for other items included in an EDF. For example, the EDF may include items such as voltage rails, off-chip resources and their interfaces, temperature constraints, etc.

4 Relationships from Benchmarking

In the previous Section, we described the EDF files used to facilitate the description of constraints for a mobile device. We feel this is necessary so that we can benchmark a wide range of possibilities. This approach contrasts with traditional processor benchmarking suites. The difference is that in processor benchmarks, the tool flow (compiler) and benchmarking environment include a restricted range of choices.

The benefit of decoupling the environment from the benchmark is that instead of providing a benchmark suite that is simply used to compare solution A to solution B under conditions C, we facilitate a broader comparison. For example, solution A may be better than solution B under conditions C, which is representative of one particular type of mobile device, but solution B may be better than solution A under conditions D. Also, this approach avoids the scenario where competitors use our benchmarks for marketing claims that their devices are better. The reality, at present, is that reconfigurable architectures are roughly an order of magnitude away in power consumption and support for power modes from being adopted in mainstream mobile devices, and these simple technology comparisons are irrelevant at present.

GroundHog 2009 benchmark suite has an additional layer of decoupling. Within the benchmark suite, there is no specification of what rules need to be specifically followed when using the suite. The reason for this is that these rules, much like the environment, may only apply in certain scenarios, and this would restrict the potential for innovation. For example, a scenario might exists where solution A needs to be benchmarked for a device, but on that device it will not be used for designs GH09.B.3, GH09.B.4, or GH09.B.6. Under this scenario, the benchmark suite only needs to be used for the designs it intends to cover, and the unused designs could be left out.

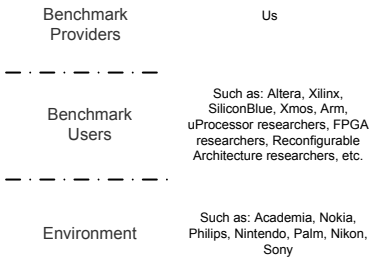

Figure 4. Relationships in benchmarking

Figure 4 illustrates how relationships exist with a benchmark suite. Decoupling benchmark providers from the more relevant relationship between benchmark users and environment providers is a better solution for our given situation and goals. This, we believe, is one of the strengths of GroundHog 2009 benchmark suite.

5 Using the Benchmarks - Examples

In this Section, we illustrate how GroundHog 2009 benchmark suite can be used to benchmark reconfigurable architectures in terms of power consumption. Our target architectures are SiliconBlue's iCE65L04 FPGA and Actel's Igloo AGL600 FPGA. For both of these target architectures, we will map HDL versions of the designs GH09.B.1 and GH09.B.2 to these FPGAs and measure power consumption for a particular workload.

5.1 Architectures and experimental setup

The two FPGAs to be measured as systems under test (SUT) are Actel's AGL600 [1] and SiliconBlue's iCE65L04 [17]. Both of these devices are on the leading edge of low power-consuming FPGA architectures for mobile applications.

Table 2. Resources available on the FPGAs

FPGA	System Gates	RAM bits	I/O Pins
AGL600	600k	108k	235
iCE65L04	200k	80k	176

Table 2 provides a brief overview of these two FPGAs. In column one, the FPGAs are listed. Columns two, three, and four show the number of gates, RAM bits, and I/O pins. In terms of gates per FPGA chip, this number is very hard to compare between different manufacturers. The main point to draw is both of these architectures are small FPGAs.

The GroundHog 2009 designs GH09.B.1, a port expander and keypad controller, and GH09.B.2, a glue logic design consisting of a state machine and three adders, are implemented in HDL design and mapped to both of these SUTs using provided tool flows corresponding to the FPGA. Figure 5 shows how the FPGAs (SUTs) are included in our measurement system. In this figure, a National Instruments PXI-4130 [12] is the measuring instrument that supplies the core voltage to a SUT and measures and records the current supplied to these devices. Additionally, a DE2 board with an Altera Cyclone FPGA [2] is used as a stimulus generator, which sends the events in the workload to stimulate the SUT. The stimuli generator reads the timestamp of the events in a workload and generates corresponding vectors.

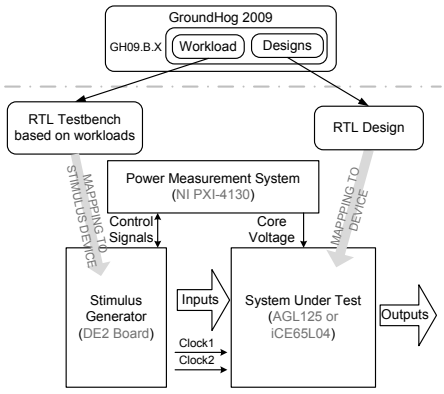

Figure 5. The measurement system

In the measurements below, we are measuring the core FPGA power consumption, which does not include I/O power consumption. We have chosen this power measurement to alleviate some of the complications with board power consumption due to the boards the FPGAs are mounted on and I/O pin loads, which impact the power consumption on the I/O pins. GroundHog 2009 can be used for other types of power measurements.

5.1.1 GH09.B.1 Power Measurements

As described above, GH09.B.1 design is a port expander and keypad controller. This design includes registers that determine how the device is to operate including modes for up to a 7x8 keypad controller, for 15 general purpose input and output pins (GPIOs), and for a mixture of uses of the two. For our experiments, we measure the power consumption for two workloads where workload one stimulates the device operating as a 7x8 keypad controller, and workload two stimulates the device operating with 8 input pins and 8 output pins.

For the 7x8 keypad mode, the system is operated for two system clocks (32 MHz and 150 kHz) for the Actel FPGA and 32 MHz for the SiliconBlue FPGA. The keypad response time is 50ms (which allows 20 key-presses a second). The design when mapped to the Actel Igloo FPGA uses 10% of the FPGA resources and on the SiliconBlue iCE FPGA it uses 28% of the FPGA resources.

Table 3. Power consumption of 7x8 keypad

FPGA	Operation Frequency	$V_{CC,avg}(V)$	$I_{avg}(mA)$	$P_{avg}(mW)$
AGL600	32MHz	1.5	3.751	5.628
AGL600	32MHz	1.2	2.951	3.541
AGL600	150KHz	1.5	0.091	0.137
AGL600	150KHz	1.2	0.059	0.071
iCE65L04	32MHz	1.5	1.949	2.924

Table 3 shows the power measurements for the 7x8 keypad controller design for a workload that randomly sends key-presses based on the workload parameters $T_{event\_in\_burst} = 25ms$, $T_{burst} = 500ms$, and $T_{period} = 1000ms$. Column 1 and 2 show the FPGA and operating frequency of the SUT. Column's 3, 4, and 5 show the voltage, average current, and average power consumption over 5 minutes.

Table 4. Power consumption of GPIOs

FPGA	Operation Frequency	$V_{CC,avg}(V)$	$I_{avg}(mA)$	$P_{avg}(mW)$
AGL600	32MHz	1.5	3.756	5.635
AGL600	32MHz	1.2	2.954	3.546
AGL600	150KHz	1.5	0.094	0.142
AGL600	150KHz	1.2	0.061	0.073
iCE65L04	32MHz	1.5	1.958	2.937

Table 4 shows the power measurements for the design with 8 input and 8 output GPIO pins for a workload that randomly updates the output pins or generates an input value based on the workload parameters $T_{event\_in\_burst} = 1ms$, $T_{burst} = 10ms$, and $T_{period} = 1000ms$. This table has the same structure as Table 3.

These results show that for both architectures there is a slight increase in power consumption for workload two compared to workload one. This is due to a slight increase of activity of the SUT on the serial parallel interface for this scenario.

5.1.2 GH09.B.2 Power Measurements

GH09.B.2 is a glue logic design consisting of a state machine and three adders. Each adder is either incrementing with the slow clock, incrementing with the fast clock, or remaining constant in an idle state. These states are controlled by the state machine which in turn is controlled by external signals. Within the design the three adders are a 4-bit, 8-bit, and 12-bit adder.

Table 5. Power consumption for GH09.B.2

FPGA	Chip Utilisation	Operation Frequency	$V_{CC,avg}$ (V)	I_{avg} (mA)	P_{avg} (mW)
AGL600	6%	32MHz	1.2	1.79	2.15
iCE65L04	4%	32MHz	1.2	1.46	1.75

Table 5 has the same structure as the previous two tables and shows the power measurements for the glue logic design with random state changes defined by the workload parameters $T_{event\_in\_burst} = 1ms$, $T_{burst} = 10ms$, and $T_{period} = 1000ms$. These results show that SiliconBlue's low power FPGA is better than Actel's in this case, but note that this comparison is preliminary and many design factors have not been considered for a fair comparison. For example, the AGL600 has an active phase locked loop compared

to no phase locked loop on the iCE65L04. The goal of this comparison is to illustrate our benchmarks, and we have not made an attempt to perform a fair comparison.

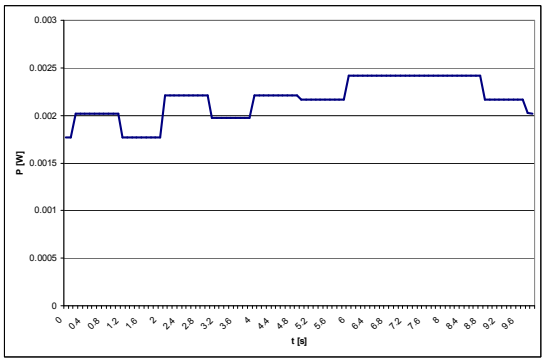

Figure 6. Power measurements of GH09.B.2

Figure 6 shows the power measurements for the first 9 seconds of this design on the Actel Igloo FPGA. The x-axis shows time in seconds and the y-axis shows the power consumption in watts. In this figure, we can see how the power consumption increases as more or fewer adders are incrementing at the slower and higher clock frequencies due to state changes. These step transitions suggest there is room for improvement of the entire design's power consumption. For example, at the points where part of the device (a portion of the adders) is in an idle state, a smart design maybe able to reduce the power consumption of the overall chip. The challenge, however, is that it is unknown when the idle parts of the design will be reactivated, and some sort of quick power on recovery solution needs to be created.

6 Conclusion

In this paper, we introduce the GroundHog 2009 Benchmarking suite for reconfigurable architectures in the mobile domain. We illustrate the composition of this benchmark suite and describe how it differs from existing available benchmarks, emphasising how the relationships in benchmarking can be leveraged to create a flexible benchmarking suite. Finally, we use this suite to measure the power consumption of two designs from the suite (including input stimuli) and show how these results can be used to compare devices and identify potential power optimisations.

The benchmark can be found at `http://cc.doc.ic.ac.uk/projects/GROUNDHOG/`.

References

[1] Actel. *Igloo Handbook*, Jan 2008.

[2] Altera. *Cyclone II Device Handbook*, 2007.

[3] J. Anderson and F. Najm. Power estimation techniques for FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(10):1015–1027, October 2004.

[4] T. Becker, P. Jamieson, W. Luk, P. Cheung, and T. Rissa. Towards Benchmarking Energy Efficiency of Reconfigurable Architectures. In *FPL*, pages 691–694, 2008.

[5] T. Becker, P. Jamieson, W. Luk, P. Cheung, and T. Rissa. Power characterisation for the fabric in fine-grain reconfigurable architectures. In *SPL*, pages 691–696, 2009.

[6] V. Betz and J. Rose. Directional Bias and Non-Uniformity in FPGA Global Routing Architectures. In *14th IEEE/ACM Int'l Conference on CAD*, pages 652–659, 1996.

[7] W. Burleson, R. Tessier, D. Goeckel, S. Swaminathan, P. Jain, J. Euh, S. Venkatraman, and V. Thyagarajan. Dynamically parameterized algorithms and architectures to exploit signal variations for improved performance and reduced power. In *International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[8] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. Irwin, and T. Tuan. A dual-Vdd low power FPGA architecture. In *FPL*, pages 145–157, 2004.

[9] J. Gustafson, D. Rover, S. Elbert, and M. Carter. The design of a scalable, fixed-time computer benchmark. *J. Parallel Distrib. Comput.*, 12(4):388–401, 1991.

[10] P. Havinga, L. Smit, G. Smit, M. Bos, and P. Heysters. Energy management for dynamically reconfigurable heterogeneous mobile systems. In *IPDPS*, pages 840–852, 2001.

[11] J. Liang, R. Tessier, and D. Goeckel. A dynamically-reconfigurable, power-efficient turbo decoder. In *FCCM*, pages 91–100, Washington, DC, USA, 2004.

[12] National Instruments. *NI PXI-4130 - 20V 2A Source Measure Unit*, 2008.

[13] J. Noguera and I. Kennedy. Power reduction in network equipment through adaptive partial reconfiguration. In *FPL*, pages 240–245, 2007.

[14] C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, L. Thiele, and G. Trster. The case for reconfigurable hardware in wearable computing. *Personal and Ubiquitous Computing*, 7(5):299–308, 2003.

[15] K. Poon, A. Yan, and S. Wilton. A Flexible Power Model for FPGAs. In *FPL*, pages 312–321, 2002.

[16] L. Shang, A. S. Kaviani, and K. Bathala. Dynamic power consumption in Virtex™-II FPGA family. In *FPGA*, pages 157–164, 2002.

[17] SiliconBlue. *iCE DiCE: iCE65L04 Ultra Low-Power FPGA Known Good Die*, Sep 2008.

[18] K. O. Tinmaung, D. Howland, and R. Tessier. Power-aware FPGA logic synthesis using binary decision diagrams. In *FPGA*, pages 148–155, 2007.

[19] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger. A 90nm low-power FPGA for battery-powered applications. In *FPGA*, pages 3–11, 2006.

[20] Xilinx. *Virtex-II Pro Platform FPGAs: Functional Description*, Oct 2003.

[21] S. Yang. Logic Synthesis and Optimization Benchmarks, Version 3.0. 1991.