Benchmarking and Evaluating Reconfigurable Architectures Targeting the Mobile Domain

PETER JAMIESON
Miami University
TOBIAS BECKER, PETER Y. K. CHEUNG, and WAYNE LUK
Imperial College
TERO RISSA
Nokia Devices R&D
and
TEEMU PITKÄNEN
Tampere University of Technology

We present the GroundHog 2009 benchmarking suite that evaluates the power consumption of reconfigurable technology for applications targeting the mobile computing domain. This benchmark suite includes seven designs; one design targets fine-grained FPGA fabrics allowing for quick state-of-the-art evaluation, and six designs are specified at a high level allowing them to target a range of existing and future reconfigurable technologies. Each of the six designs can be stimulated with the help of synthetically generated input stimuli created by an open-source tool included in the downloadable suite. Another tool is included to help verify the correctness of each implemented design. To demonstrate the potential of this benchmark suite, we evaluate the power consumption of two modern industrial FPGAs targeting the mobile domain. Also, we show how an academic FPGA framework, VPR 5.0, that has been updated for power estimates can be used to estimates the power consumption of different FPGA architectures and an open-source CAD flow mapping to these architectures.

Categories and Subject Descriptors: B.7.1 [**Intergrated Circuits**]: Types and Design Styles—*Gate arrays*; B.8.0 [**Performance and Reliability**]: General

General Terms: Measurement, Experimentation

Additional Key Words and Phrases: Benchmarking, FPGAs, benchmark, mobile, power

Authors' addresses: P. Jamieson, Department of Electrical and Computer Engineering, Miami University, Oxford, OH, 45056; email; jamieson@eecg.toronto.edu; T. Becker, P. Y. K. Cheung, W. Luk, Imperial College, London SW7 2AZ, UK; T. Rissa, Nokia Devices R&D; T. Pitkänen, Tampere University of Technology, P.O. Box 527, FI-33101 Tampere, Finland.

DOI 10.1145/1698759.1698764 http://doi.acm.org/10.1145/1698759.1698764

ACM Reference Format:
Jamieson, P., Becker, T., Cheung, P. Y. K., Luk, W., Rissa, T., and Pitkänen, T. 2010. Benchmarking and evaluating reconfigurable architectures targeting the mobile domain. ACM Trans. Des. Autom. Electron. Syst. 15, 2, Article 14 (February 2010), 24 pages.
DOI = 10.1145/1698759.1698764 http://doi.acm.org/10.1145/1698759.1698764

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 demonstrated applications of reconfigurable technology in the mobile domain [Havinga et al. 2001; Plessl et al. 2003] where these designs can meet the speed requirements and provide possible power and performance benefits over other approaches. 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 optimizations in and measure the power consumption of reconfigurable architectures targeting the mobile domain. This benchmark suite includes seven designs in which one design provides a worst-case fabric analysis of fine-grain FPGAs and the other six 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 future benchmarks added to next versions of the benchmark suite.

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 that would benefit from a reconfigurable technology based on the need for hardware acceleration, pin expansion, or hardware implementation.
—to provide a methodology for measuring and reporting results.

In this article, we describe the GroundHog 2009 benchmark suite based on these described challenges. Our approach has made an extensive study of previous benchmark suites [Becker et al. 2008] 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 by what we call relevant relationships. 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 commercial FPGAs and measured for power consumption for our defined input stimuli. From these results, we illuminate where simple optimizations may be made in the future, and we show how two different architectures can be preliminarily compared. We also map two of the benchmarks to an academic evaluation framework, VPR 5.0 [Luu et al. 2009]. Using an updated version of VPR 5.0 that includes power estimation, we can analyze a design mapped to different FPGA architectures to determine the best power consuming architecture.

One goal of this benchmark suite is that it will motivate the design of new reconfigurable architectures targeting the mobile domain, and we claim that the designs within this suite can target a range of reconfigurable technologies. However, in this work, we only demonstrate the application of some of the benchmarks on fine-grain FPGAs. The reason for this is that we believe these fine-grain FPGAs are the most likely technologies to move into this market at present based on conversations with mobile device manufactures and the availability of industrial products, including Actel's Igloo FPGAs [Actel 2008] and SiliconBlue's iCE FPGAs [SiliconBlue 2008].

This work was originally presented at FCCM'09 [Jamieson et al. 2009]. In this version, we have focused on updating our measurements on commercial FPGAs based on discussions with the vendors. Also, we have focused on illustrating how these benchmarks can be used in an academic setting to evaluate ideas as they apply to FPGAs in the mobile domain. To achieve this we have made some effort in updating the VPR 5.0 framework to support power estimation.

The remainder of this article is organized as follows: 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 the 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. Section 6 illustrates how GroundHog 2009 benchmarks can be mapped into VPR 5.0 and evaluated in terms of energy and power consumption, and finally, Section 7 concludes.

2. RELATED WORK

Evaluating and benchmarking reconfigurable architectures for power has been achieved by using existing benchmarks, such as MCNC [Yang 1991], or by

modeling 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. [2004], Noguera and Kennedy [2007], and Burleson et al. [2001] build specific instances of an application on a reconfigurable architecture and optimize these implementations for power consumption. In related work, Shang et al. [2002] show the dynamic power consumption of a Xilinx Virtex-II FPGA [Xilinx 2003] 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. [2006] present a low-power FPGA core with several optimizations such as voltage scaling, leakage reduction of configuration memory cells, and power gating of tiles with preservation of state and configuration.

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. [2002] use the MCNC benchmark and add power models of a common FPGA architecture exploration tool, VPR [Betz and Rose 1996]. 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 and Najm [2004] also use the MCNC benchmarks in their work to estimate power consumption in FPGAs. Gayasen et al. [2004] propose a scheme with two programmable supply voltages where the higher voltage is used for critical path logic and the lower voltage for noncritical parts. Using MCNC circuits, they achieve an average power saving of 61%. More recently, Tinmaung et al. [2007] optimize for power on FPGAs during logic synthesis and use the MCNC benchmarks to perform measurements of their optimizations.

Reducing FPGA power consumption is a goal for all the FPGA vendors including some recent power optimizations in both Altera's StratixIII FPGA [Altera 2006] and Xilinx's Virtex 5 [Xilinx 2006]. Only a few vendors are targeting their devices to the low-power mobile domain. Actel's Igloo FPGAs [Actel 2008] and SiliconBlue's iCE FPGAs [SiliconBlue 2008] are low-cost FPGAs (approximately 1 to 2 USD) that are the leading edge in low-power consuming FPGAs targeting a handheld market.

Finally, a number of low-power CAD algorithms and FPGA architectural optimizations have been proposed from a broad range of researchers (too many to list here). Lamoureux's thesis work Lamoureux [2007] presents a range of tools and optimization techniques focused on reducing power consumption. These include algorithms in technology mapping, clustering, placement, and routing that use parts of the Poon power estimation framework to analyze how to best map a design to an FPGA to minimize power consumption. Lamoureux [2007] also investigates architecture features to minimize power in the clock tree and reduce power consumption due to glitching.

Much of the previous work allows for power measurements on reconfigurable architectures and power optimizations, but they do not realistically model modern applications on these devices. This is especially true for the mobile computation domain where there are 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 factors for this adoption are 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, multicore 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 Sections 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 Sections 3.4 and 4).
—prevent system or tool optimizations 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, make-up this innovative framework and satisfy many of the earlier challenges described. They are:

(1) providing high-level design descriptions;
(2) providing synthetically generated, parametrizable input stimuli;
(3) allowing the environment to be uniquely specified; and
(4) allowing early baseline fabric analysis of fine-grained FPGAs.

In this work, we do not focus on item four, and we direct the reader to previous work [Becker et al. 2009]. However, note that fabric analysis is included in the benchmark suite to allow our community to quickly evaluate the power consumption 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 the 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 how a technology can efficiently implement arithmetic operations. On the other hand, the data compression design exercises how a technology can efficiently implement memories that are accessed pseudorandomly. In Section 3.2, we describe more details on how these designs are specified.

These benchmarks were chosen by looking at existing mobile phones and questioning what might a reconfigurable device be useful for and what are common functions within a communicating mobile device. Both the port expander and bridge chip are included as these designs address a common problem within mobile devices by expanding the number of pins. This is the case since the baseband processor is pin limited. Similarly, glue logic implements pin expansion as well as other types of custom fixes potentially needed for a mobile device. The 2D convolution design was included as a common algorithm used in the both digital radio transmission and reception, and it is also used in many DSP applications including graphics processing. Both AES and data compression are algorithms that might be used as hardware acceleration for applications loaded onto the phone. They might also be included as parts of the radio transmission depending on the mobile device. For example, many military applications require encryption hardware on the transmission path.

In addition to the designs, we have also included open-source 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. These stimuli can be created to model classic throughput like inputs as well as intermittent stimuli that more closely model 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 they 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 analyze the behavior of their implementation for a given input stimulus. This helps in both understanding the expected behavior of a benchmark design and verifying whether 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, multicore systems, and microprocessors. This broad range of targets makes it difficult to describe designs in a form that is mappable 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 multiplier? 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 first mentioned; reasons, designs within the GroundHog 2009 benchmark suite are described in a high-level format, which is a similar approach taken by the SLALOM [Gustafson et al. 1991] 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 standardized descriptions, written descriptions, or a

Table I. The Characteristics of Designs in Our Benchmark Suite

Design Name	Bit-Width	Processing Flow	Memory Usage	Arithmetic Complexity	Performance Requirements
GH09.B.1 - Port expand and keypad controller	90% bit-level	Control	Simple	Simple	Low
GH09.B.2 - Glue logic	90% bit-level	Control	Simple	Simple	Low / High
GH09.B.3 - AES encryption cypher	90% bit-level	Data	Simple	Simple	High
GH09.B.4 - Compression using Lempel-Ziv	90% bit-level	Control	Random	Simple Access	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

mixture of all these. 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 that benchmark users can map the designs to any target architecture, but to achieve this, the benchmark user needs to make a synthesizable 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 to differ based on a set of characteristics that includes the following.

—*Bit-Width*. This is the width of the operations varying between bit-level and word-level operations.

—*Processing Flow*. This defines how computation is performed for a design. We classify a design as *control* if there are data dependencies between present and past data results, because these dependencies result in varying delays. Designs without past/present data dependencies are classified as *data* flow.

—*Memory Usage*. This characterizes a design based on how it uses memory. This can either comprise *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*. This is the expected speed at which the outputs need to be generated.

Table I 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 characterization we can see that the benchmarks have been chosen to differ from each other in at least one characteristic. This does not cover the complete set of possibilities which may be covered in future versions of the benchmark suite.

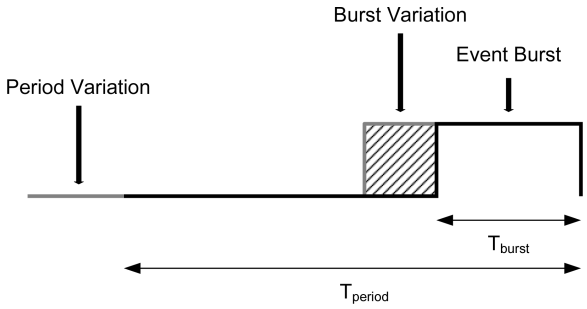

Fig. 1. Parameters for the pulse wave.

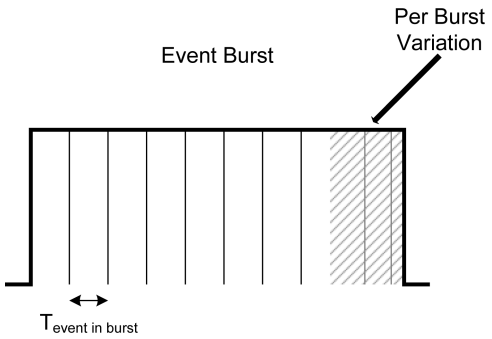

Fig. 2. Parameters for events in a burst.

3.3 Input Stimuli

One of the missing aspects of existing benchmark suites, 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.

The 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-lines of events and workloads.

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}.

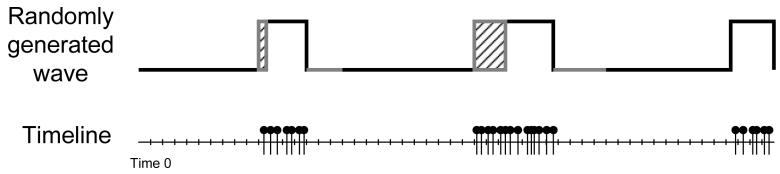

Fig. 3. Example of what a timeline of events would look like for a parametrized wave.

Given these wave parameters, and parameters for random variations 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. 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 is 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. It is also possible to map the workload generation tool to a soft-processor on an FPGA and then the processor will generate input vectors internally.

3.4 Environment Description

The 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 is in the form of what we call an Environment Description File (EDF), and allows institutions, external to the benchmark specifiers and benchmark users, to describe mobile frameworks. 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 the following.

—*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,

```
<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 realte to the serial clock -->
        <minimum_arrival_rate_on_parallel_interfaces>32000ns</minimum_arrival_rate_on_paralle_interfaces>
        <!-- can relate to the parallel clock -->
</environment>
```

Fig. 4. One of the sample EDFs included with GroundHog 2009.

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.

Figure 4 shows one EDF file included in the distribution that describes a 32MHz clock and 32KHz heartbeat clock for a synchronous device, which are typical for mobile devices. These files are distributed in XML form, and the tools included with the suite can read this EDF to create the workloads. The EDFs included only contain rates of operation. There is a huge range of possibilities of other items that could be 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 and system benchmarking suites. The difference is that in these 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 reconfigurable architecture vendors 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

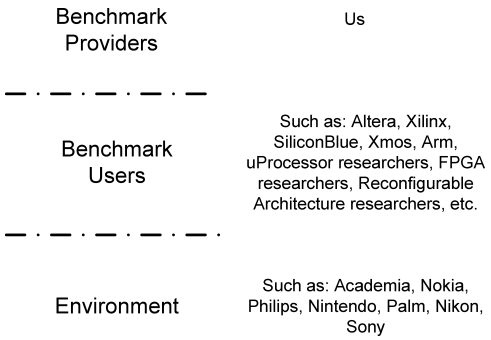

Fig. 5. Relationships in benchmarking.

power consumption and support for power modes from being adopted in mainstream mobile devices, and these simple technology comparisons are irrelevant at present.

The 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 exist where solution *A* needs to be benchmarked for a device, but on that device it will never 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 5 illustrates how relevant relationships exist for a benchmark environment. Decoupling benchmark providers (us) 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 the GroundHog 2009 benchmark suite.

5. USING THE BENCHMARKS ON COMMERCIAL FPGAS

In this section, we illustrate how the 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. These results are meant to show the viability of our setup, and for reasons explained shortly the values presented are not a fair comparison of the two FPGA architectures. The reason that only two of the benchmarks are used is that they are, presently, the only low-level implementations of the designs. Based on contributions by the research community we hope to have low-level implementations of all of the benchmarks, but for this work, we only show how the benchmark can be used in practice.

Table II. Resources Available on the FPGAs

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

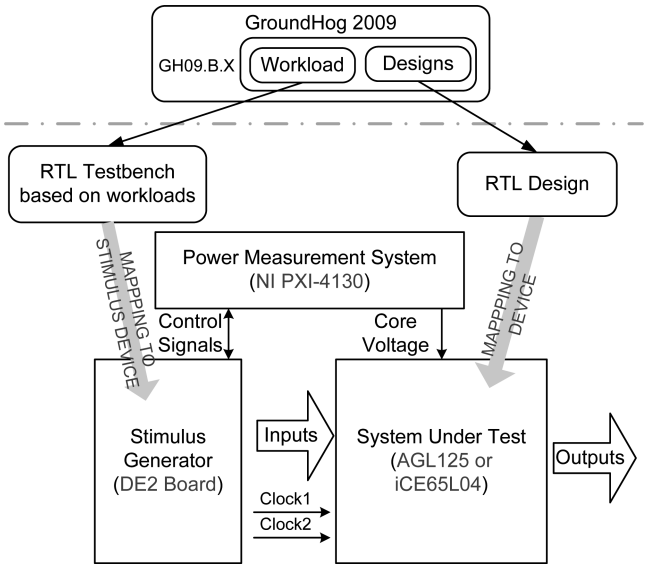

Fig. 6. The measurement system.

5.1 Architectures and Experimental Setup

The two FPGAs to be measured as Systems Under Test (SUT) are Actel's AGL600 [Actel 2008] and SiliconBlue's iCE65L04 [SiliconBlue 2008]. Both of these devices are on the leading edge of low-power-consuming FPGA architectures for mobile applications.

Table II provides a brief overview of these two FPGAs. In column one, the FPGAs are listed. Columns two, three, and four show the number of system gates, RAM bits, and I/O pins. In terms of system gates per FPGA chip, this number is very hard to compare between different manufacturers, and these numbers are taken from each vendor's documentation. The main point to draw is that both of these architectures are small FPGAs, and not to compare these two devices based on the number of system gates.

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 6 shows how the FPGAs (SUTs) are included in our measurement system. In this figure, a National Instruments PXI-4130 [National Instruments 2008] 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 [Altera 2007] is used as a stimulus generator, which sends the

Table III. Power Consumption of 7×8 Keypad

FPGA	Operation Frequency	$V_{CC,\ avg}(V)$	$I_{avg}(mA)$	$P_{avg}(mW)$
AGL600	32MHz	1.5	3.75	5.62
AGL600	32MHz	1.2	2.95	3.54
AGL600	150KHz	1.5	0.09	0.13
AGL600	150KHz	1.2	0.06	0.07
iCE65L04	32MHz	1.2	1.50	1.80
iCE65L04	32MHz	1.0	1.33	1.33

Table IV. Power Consumption of GPIOs

FPGA	Operation Frequency	$V_{CC,\ avg}(V)$	$I_{avg}(mA)$	$P_{avg}(mW)$
AGL600	32MHz	1.5	3.76	5.64
AGL600	32MHz	1.2	2.95	3.55
AGL600	150KHz	1.5	0.09	0.14
AGL600	150KHz	1.2	0.06	0.07
iCE65L04	32MHz	1.2	1.52	1.82
iCE65L04	32MHz	1.0	1.34	1.34

events in the workload to stimulate the SUT. The stimuli generator reads the timestamp of the events in a workload and generates corresponding vectors.

In the measurements that follow, 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.

5.1.1 *GH09.B.1 Power Measurements.* As described earlier, the GH09.B.1 design consists of a port expander and keypad controller. This design includes registers that determine how the device is to operate including modes for up to a 7×8 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 7×8 keypad controller, and workload two stimulates the device operating with 8 input pins and 8 output pins.

For the 7×8 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 III shows the power measurements for the 7×8 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 $T_{period} = 1000ms$. Columns 1 and 2 show the FPGA and operating frequency of the SUT. Columns 3, 4, and 5 show the voltage, average current, and average power consumption over 5 minutes.

Table IV 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

Table V. 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	1.42	1.70
iCE65L04	4%	32MHz	1.0	1.17	1.17

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 III.

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 V 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 the no-phase locked loop on the iCE65L04. The goal of this comparison is to illustrate how the benchmarks can be mapped and measured for power consumption, and we have not made an attempt to perform a fair comparison.

Figure 7 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. USING GROUNDHOG 2009 IN AN ACADEMIC FRAMEWORK

Our previous setup and measurements on commercial FPGAs is useful to compare different existing commercial FPGAs, but these measurements provide very little insight or a framework to evaluate architectural changes and the

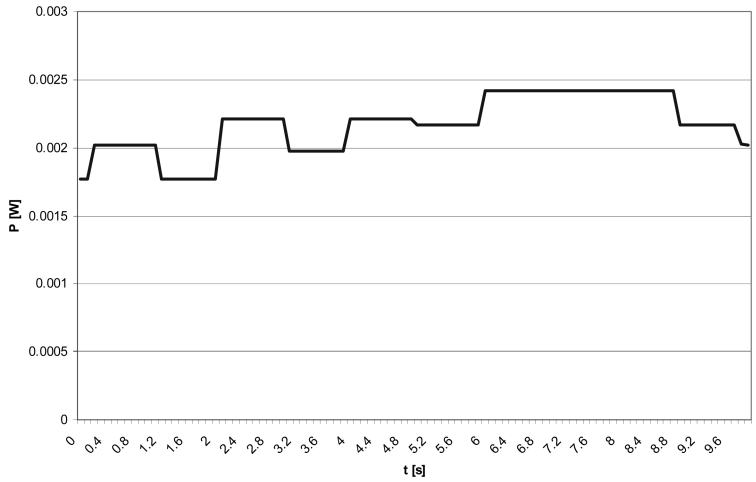

Fig. 7. Power measurements of GH09.B.2.

CAD flow that maps designs to these devices. In this section, we briefly introduce the VPR 5.0 framework, which we have updated for power estimation, and we show how both GroundHog designs can be mapped to a range of FPGA architectures and evaluated in terms of power consumption.

First, we describe the VPR 5.0 framework for power estimation. This framework includes details of the CAD flow that maps designs to the architecture and how an architecture is described. Once we have described the framework we perform two experiments. In the first experiment, we map GH09.B.1 (port expander and keypad controller) to different FPGA architectures with varied cluster sizes and the different routing architectures. In the second experiment, we map GH09.B.2 to a specific architecture and investigate the energy consumption in terms of details of the design and the architecture.

6.1 VPR 5.0 for Power Estimation

In our experiments, we use an island-style FPGA that consists of an array of logic blocks and I/O cells that are interconnected using programmable routing [Betz et al. 1999]. The programmable routing consists of wire segments that connect, via programmable switches, to both logic blocks, and other wire segments. These connection points are called either switch blocks or connection blocks, where a switch block joins wires and a connection block joins a logic block to wires. The logic blocks, also called clusters, consist of a number of (BLEs), which themselves are commonly a combination of a LUT and flip-flop [Singh and Marek-Sadowska 2002].

Most modern FPGAs within the traditional domain use what is called unidirectional routing architectures, where the wires making up the programmable routing only propagate signals in one direction. This routing architecture choice is made based on work by both Lemieux and Lewis [2004] and Lewis et al. [2005] that shows that unidirectional architectures result in faster and more

area-efficient designs mapped to these architectures compared to an FPGA with bidirectional routing (in which wires can propagate signals in either direction). In the first experiment we will evaluate FPGAs with both types of routing architecture.

Each FPGA architecture is described using many parameters that define its connectivity and makeup, including the number of (BLEs) per cluster (N), the input size of a LUT (K) in a BLE, the number of routing tracks per channel (W), the input connectivity to the BLEs in a soft logic cluster (F_{Cin}), the output connectivity from the BLEs to the routing tracks (F_{Cout}), logical wire length in terms of the number of clusters spanned (L), and the switch block flexibility connecting routing tracks with each other (F_s). In addition to these parameters, an architecture is also described in terms of its routing architecture and the transistor sizings for the programmable routing and clusters.

These architectural details are inputted into VPR 5.0 [Luu et al. 2009], which is a tool that can estimate the speed and area consumption of a design mapped to an FPGA. VPR creates an FPGA (based on the parameters) onto which it places and routes a design that has been mapped into clusters by previous CAD flow stages. The point of such a tool is that it allows us to experiment with different architectures and the CAD flow that maps a design to these architectures to find the best implementations in terms of speed and area consumption (and power for our updated tool).

Figure 8 shows the entire CAD flow including VPR's placer and router. In this figure, for each stage of the CAD flow we list the academic tool used in this work in parentheses. High-level synthesis of the GroundHog benchmarks is done via OdinII (an improved version of Odin [Jamieson and Rose 2005]). This tool converts a Verilog design into a netlist of logic. For logic optimization and technology mapping we use ABC [Mishchenko et al. 2007]. For clustering we use T-VPACK [Marquardt et al. 1999], and for both placement and routing we use VPR 5.0 placement and routing algorithms [Luu et al. 2009] originally created by Betz et al. [1999].

As mentioned earlier, VPR 5.0 can estimate speed and area, but in this work we are concerned with power consumption. Poon et al. [2002] created a power estimation framework for VPR 4.3, and for this work we have updated this framework for VPR 5.0.

Poon et al.'s [2002] original power estimation framework first estimates switching activity on the nets connecting the clusters (where a net is comprised of wires making up a connection between one output and a number of inputs). Next, these switching activation estimates, the FPGA architecture, and the placed-and-routed design are used to compute the dynamic and static power consumption of that design on a particular FPGA architecture. Our updated framework uses this estimation framework as a starting point, making the needed changes to support the basic estimation as well as a new feature in VPR 5.0, unidirectional routing.

Activity estimation of a design is done using ACE 1.0, a tool developed by Poon et al. [2002]. This activation estimation tool uses transition density models, assuming all primary inputs have a transition density of 0.5. This approach does not fit well with the principles of GroundHog 2009, where we emphasize

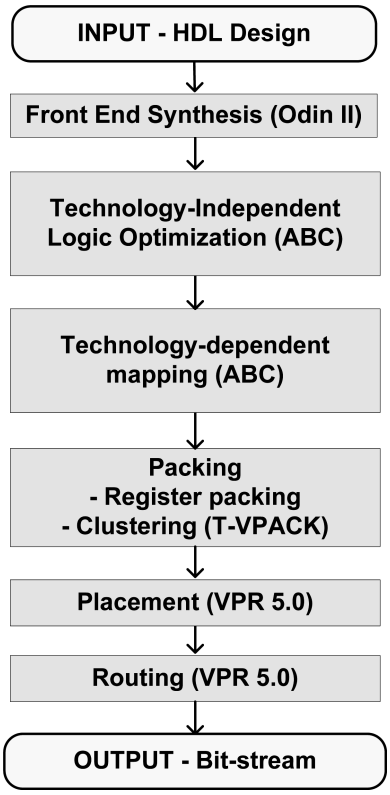

Fig. 8. CAD flow to map a design to an FPGA.

the need for realistic input stimuli. To improve this estimation, future efforts will need to use the workloads generated by the GroundHog framework to to activity estimation, but we leave this as future work.

The power models created by Poon et al. [2002] are in 180nm CMOS technology, and we use this technology point in our results. This choice allows us to see trends in power and energy consumption, but we cannot experiment with the power consumption of modern FPGAs.

For our experiments we use the following architectural parameters:

(1) $F_{cin} = 0.15$
(2) $F_{cout} = 0.10$
(3) $F_s = 3$
(4) $L = 1$

Note that for routing parameters F_{cin}, F_{cout}, and F_s a decimal number in these columns represents fractional connectivity that depends on the channel width, W, and integer values represents absolute connectivity. For example, if W is 20 then each output of the cluster for Basic will connect to 2 of the tracks in the channel. The architectures have routing parameters F_{cin}, F_{cout}, and F_s selected

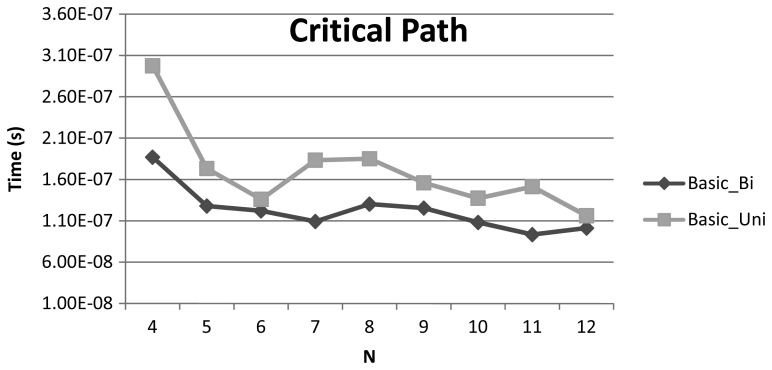

Fig. 9. Critical path delay as N increases.

based on the ranges given in previous work by Lemieux and Lewis [2001]. For the rest of the parameters in the architecture, we use architecture files created by Kuon and Rose [2008]. We used the architecture file for an $N = 10$, $K = 4$, and transistors sized based on Area-Delay for 180nm technology.

For each benchmark and for each given architecture, we map the benchmark to the architecture and find both the minimum array size and the minimum channel width, W, using a binary search. The channel width is then increased by 20% to alleviate any routing pressure. This approach is commonly used in architectural experiments using VPR. The critical path delay, area consumption, and power consumption results are geometrically averaged for all the benchmarks, and these averaged results are used in our analysis.

6.2 Evaluating GH09.B.1 for Cluster Size and Routing Architecture

In our first experiment, we map the port expander and keypad controller to FPGA architectures where cluster size (N) varies from 4 to 12, and we use architectures with bidirectional and unidirectional routing. For this experiment we will show graphs of the critical path delay, area, and energy consumption per clock cycle as N increases. In general, as N increases, the logic block absorbs more of the design and intracluster routing is traded off for intercluster routing. Intercluster routing is expensive in terms of speed, area consumption, and power consumption, so as N increases we expect to see improved results. However, as the cluster increases in size, there is a point where the intracluster routing begins to dominate area and power consumption, and this is a traditional trade off between intercluster and intracluster routing.

Figure 9 shows the critical path delay of GH09.B.1 on the two FPGA routing architectures as N increases. We can see that the critical path delay decreases as more of the design logic is absorbed into the cluster. Figure 10 shows the area of GH09.B.1 on the two FPGA routing architectures as N increases, and we see a similar trend where area goes down as more logic is absorbed into the cluster and the programmable routing area decreases. In terms of comparing the two routing architectures, bidirectional is faster and unidirectional is smaller, but these observations are only true for this specific benchmark and for the given

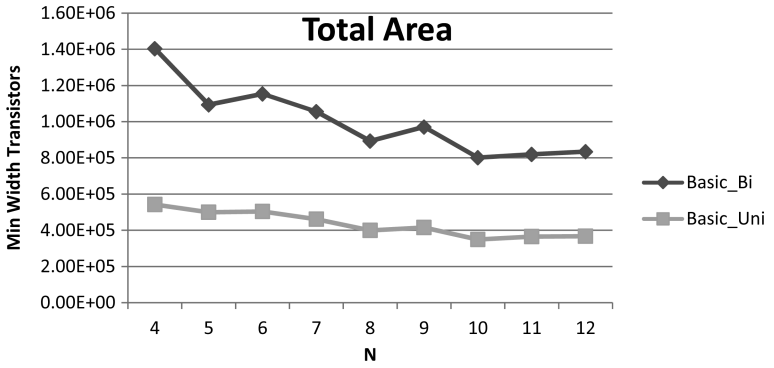

Fig. 10. FPGA area as N increases.

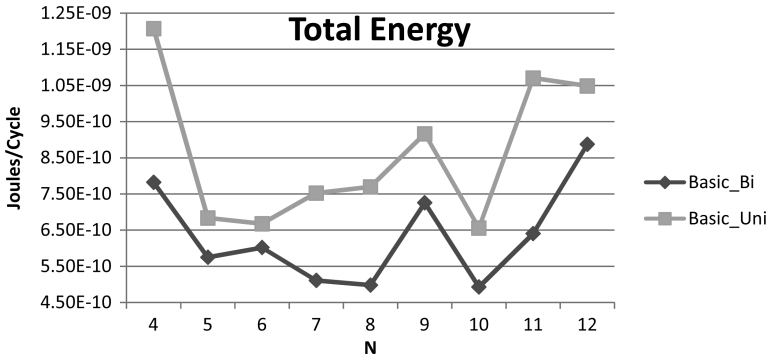

Fig. 11. Total energy consumed per clock cycle as N increases.

routing architecture and architectural parameters. A more detailed experiment would need to be done to make a more conclusive statement on the effects of the two routing architectures on area and speed for this design.

Figure 11 shows the energy consumed per clock cycle as N increases. We use energy per clock cycle as a metric instead of power consumption since energy measurements remove the impact of switching speed, and we can compare both routing architectures in terms of energy consumption regardless of the operating frequency. In the total energy graph, we can see that the energy consumption is somewhat erratic for increasing values of N, but in general values of N in the range of 5 to 10 seem to result in an architecture that consumes less energy. Also, for the given architecture parameters, the bidirectional routing architecture is more energy efficient.

To understand the the internal energy consumption, we can break down the results into energy consumption of the clock, clusters, and routing. Figure 12 shows a breakdown of energy for the routing, logic, and clock of our design mapped to a unidirectional routing architecture as N increases. Clock energy, in our estimation framework, is a function of the size of the FPGA, so as area decreases the clock power also decreases. In terms of the energy trade-off between routing and cluster energy consumption, we can see in general that

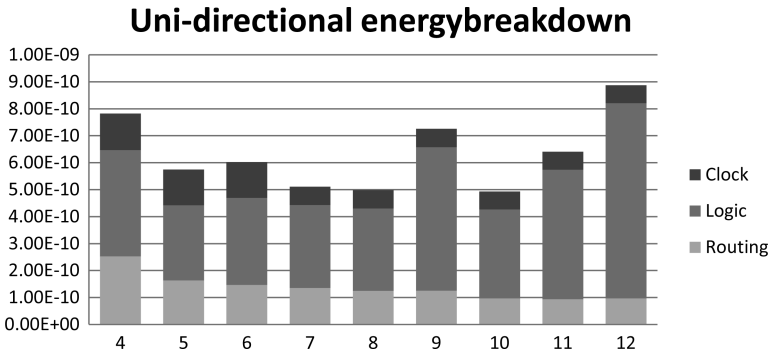

Fig. 12. Total energy consumed as N increases.

Table VI. FPGA Architectural Parameters and Transistor Sizes for This Experiment

Routing	F_{cin}	F_{cout}	F_s	L	N	K	Routing Buffer Size	Routing Multiplexer Size
unidirectional	0.15	0.10	3	1	10	4	6.2	2.363

routing energy is a small contributor to total energy, and as N increases the intercluster routing decreases. The logic energy consumption is the dominating factor, and increases as N grows.

Note that FPGAs normally consume a high amount of energy in the programmable routing, but our results don't show this. The reason is that, in our experiments, programmable routing is sized for a minimum-width transistor in all the programmable switches and buffers. This is not normally the case for real FPGAs, and an FPGA would be transistor sized for some optimization criteria such as Area-Delay or Delay-Energy. For our experiments, we are not focused on looking for an optimal FPGA architecture and instead are demonstrating the feasibility of running such experiments.

Also from Figure 12, we can see that when N is equal to 9 the energy consumed per clock cycle spikes and doesn't follow trends. These results are not surprising given that the graph is generated for only one benchmark and one random placement seed. Many of the CAD algorithms that map the design to the FPGA are meta-algorithms that use random seeds, and normally, we would run multiple executions to smooth out possible random effects. Again, we are not interested on a complete study here, and are showing what is possible with the GroundHog 2009 benchmark suite and VPR 5.0.

6.3 Analysis of GH09.B.2 Power Consumption

In the last experiment, we showed that a GroundHog 2009 benchmark can be mapped into an academic framework to measure speed, area, and power consumption to evaluate different architectures. In this section, we will show some instances where our framework can be used to evaluate the internal elements of a design's power consumption mapped to an FPGA.

For this experiment, we map GH09.B.2 (glue logic) to an FPGA architecture with the parameters listed in Table VI. The result of is the design is mapped to a 4x4 array, and there are 49 clusters and 102 nets connecting the clusters to

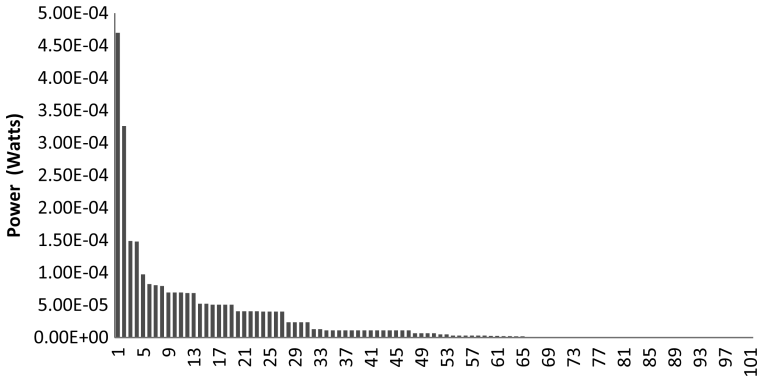

Fig. 13. Histogram of the power consumed for each routing net in GH09.B.2.

each other and primary inputs and outputs. The minimum channel width found by VPR is 18, which was increased to 22 (for the 20% increase). The critical path delay is 4.25e-09 seconds and this path travels through 5 clusters. The routing consumed 25% of the power (0.00285 Watts), and the clusters consumed 42% of the power (0.00491 Watts). Finally, 13% of the total power consumption was due to leakage power.

Figure 13 shows a histogram of power consumption of each routing net in GH09.B.2. When we look at these internal details of the power consumed in the device we see that 41 of the 102 intercluster routing nets consume no power. The reason for this is that the activation estimation tool assumes random inputs, and some of the nets have an activation that becomes so small that they do not consume any significant amount of power. The highest power consuming routing net is part of the critical path, which isn't surprising, and from the details, we know that this critical path is based on the third input pin of the input bus in the GH09.B.2. This bus is used to control the states of the finite state machine. By looking at the Verilog design, it is clear that this bus signal is one of the signals that is used in many comparisons. This is the case based on the choice of encoding schemes in our finite state machine, and with this information we might consider another encoding scheme that would reduce the impact of this signal on power consumption.

More importantly, this brief experiment shows how the academic framework can be used to examine the details of power consumption, and can lead into insights on how to reduce overall power consumption for a design mapped to an FPGA.

7. CONCLUSION

In this work, we introduce the GroundHog 2009 benchmarking suite targeting reconfigurable architectures in the mobile domain. We describe 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.

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 commercial devices and identify potential power optimizations. Also, we demonstrate how the benchmarks can be used within an academic framework to try new architecture designs, CAD algorithms, and evaluate a benchmark's power consumption.

The GroundHog 2009 benchmark suite and all publications related to this work can be found at:

`http://cc.doc.ic.ac.uk/projects/GROUNDHOG/.`

In addition to the basic description of the benchmarks, we have also created an online repository on OpenCores at:

`http://www.opencores.org/?do=project&who=groundhog2009_repository.`

The idea behind this repository is to allow individuals to post and download existing synthesizable solutions for each of the benchmarks, and all the files used in this work are available there. Hopefully, over time there will be more synthesizable instances of these designs to download that are not only synthesizable on FPGAs, but other architectures such as microprocessors.

It is our hope to establish a community of researchers in this area. The contributions needed are synthesizable versions of the benchmarks for a range of architectures, and new research into reconfigurable architectures targeting this domain.

REFERENCES

ACTEL. 2008. *Igloo Handbook*. Actel.

ALTERA. 2006. *Stratix III Device Handbook*. Altera.

ALTERA. 2007. *Cyclone II Device Handbook*. Altera.

ANDERSON, J. AND NAJM, F. 2004. Power estimation techniques for FPGAs. *IEEE Trans. Very Large Scale Integr. Syst. 12,* 10, 1015–1027.

BECKER, T., JAMIESON, P., LUK, W., CHEUNG, P., AND RISSA, T. 2008. Towards benchmarking energy efficiency of reconfigurable architectures. In *Proceedings of the International Conference on Field-Programmable Logic and Applications*. 691–694.

BECKER, T., JAMIESON, P., LUK, W., CHEUNG, P., AND RISSA, T. 2009. Power characterisation for the fabric in fine-grain reconfigurable architectures. In *Proceedings of the Southern Programmabla Logic Conference*.

BETZ, V. AND ROSE, J. 1996. Directional bias and non-uniformity in FPGA global routing architectures. In *Proceedings of the 14th IEEE/ACM International Conference on CAD*. 652–659.

BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic.

BURLESON, W., TESSIER, R., GOECKEL, D., SWAMINATHAN, S., JAIN, P., EUH, J., VENKATRAMAN, S., AND THYAGARAJAN, V. 2001. Dynamically parameterized algorithms and architectures to exploit signal variations for improved performance and reduced power. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

GAYASEN, A., LEE, K., VIJAYKRISHNAN, N., KANDEMIR, M., IRWIN, M., AND TUAN, T. 2004. A dual-Vdd low power FPGA architecture. In *Proceedings of the International Conference on Field Programmable Logic and Application*. 145–157.

GUSTAFSON, J., ROVER, D., ELBERT, S., AND CARTER, M. 1991. The design of a scalable, fixed-time computer benchmark. *J. Parallel Distrib. Comput. 12,* 4, 388–401.

HAVINGA, P., SMIT, L., SMIT, G., BOS, M., AND HEYSTERS, P. 2001. Energy management for dynamically reconfigurable heterogeneous mobile systems. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS)*. 840–852.

JAMIESON, P., BECKER, T., LUK, W., CHEUNG, P., AND RISSA, T. 2009. Benchmarking reconfigurable architectures in the mobile domain. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*.

JAMIESON, P. AND ROSE, J. 2005. A Verilog RTL synthesis tool for heterogeneous FPGAs. In *Proceedings of the International Conference on Field-Programmable Logic and Applications*. 305–310.

KUON, I. AND ROSE, J. 2008. Automated transistor sizing for FPGA architecture exploration. In *Proceedings of the IEEE-ACM Design Automation Conference (DAC'08)*. 792–795.

LAMOUREUX, J. 2007. On the interaction between power-aware computer-aided design algorithms for field-programmable gate arrays. Ph.D. thesis, University of British Columbia.

LEMIEUX, G. AND LEWIS, D. 2001. Using sparse crossbars within LUT clusters. In *Proceedings of the ACM/SIGDA International Symposium on FPGAs*. 59–68.

LEMIEUX, G. AND LEWIS, D. 2004. Directional and single-driver wires in FPGA interconnect. In *Proceedings of the IEEE International Conference on Field-Programmable Technology*. 41–48.

LEWIS, D., AHMED, E., BAECKLER, G., BETZ, V., BOURGEAULT, M., CASHMAN, D., GALLOWAY, D., HUTTON, M., LANE, C., LEE, A., LEVENTIS, P., MARQUARDT, S., MCCLINTOCK, C., PADALIA, K., PEDERSEN, B., POWELL, G., RATCHEV, B., REDDY, S., SCHLEICHER, J., STEVENS, K., YUAN, R., CLIFF, R., AND ROSE, J. 2005. The Stratix II logic and routing architecture. In *Proceedings of the ACM/SIGDA International Symposium on FPGAs*. 14–20.

LIANG, J., TESSIER, R., AND GOECKEL, D. 2004. A dynamically-reconfigurable, power-efficient turbo decoder. In *Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)*. 91–100.

LUU, J., KUON, I., JAMIESON, P., CAMPBELL, T., YE, A., FANG, W. M., AND ROSE, J. 2009. VPR 5.0: FPGA CAD and architecture xploration tools with single-driver routing, heterogeneity and process scaling. In *Proceedings of the ACM/SIGDA International Symposium on FPGAs*.

MARQUARDT, A., BETZ, V., AND ROSE, J. 1999. Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density. In *Proceedings of the ACM/SIGDA International Symposium on FPGAs*. 37–46.

MISHCHENKO, A., CHATTERJEE, S., AND BRAYTON, R. K. 2007. Improvements to technology mapping for LUT-based FPGAs. *IEEE Trans. CAD 26,* 2, 240–253.

NATIONAL INSTRUMENTS. 2008. *NI PXI-4130 - 20V 2A Source Measure Unit*. National Instruments.

NOGUERA, J. AND KENNEDY, I. 2007. Power reduction in network equipment through adaptive partial reconfiguration. In *Proceedings of the International Conference on Field Programmable Logic and Application*. 240–245.

PLESSL, C., ENZLER, R., WALDER, H., BEUTEL, J., PLATZNER, M., THIELE, L., AND TRESTER, G. 2003. The case for reconfigurable hardware in wearable computing. *Personal Ubiq. Comput. 7,* 5, 299–308.

POON, K., YAN, A., AND WILTON, S. 2002. A flexible power model for FPGAs. In *Proceedings of the Field-Programmable Logic and Applications*. 312–321.

SHANG, L., KAVIANI, A. S., AND BATHALA, K. 2002. Dynamic power consumption in Virtex-II FPGA family. In *Proceedings of the ACM/SIGDA 10th International Symposium on Field-Programmable Gate Arrays (FPGA'02)*. 157–164.

SILICONBLUE 2008. *iCE DiCE: iCE65L04 Ultra Low-Power FPGA Known Good Die*. SiliconBlue.

SINGH, A. AND MAREK-SADOWSKA, M. 2002. Efficient circuit clustering for area and power reduction in FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on FPGAs*. 59–66.

TINMAUNG, K. O., HOWLAND, D., AND TESSIER, R. 2007. Power-Aware FPGA logic synthesis using binary decision diagrams. In *Proceedings of the ACM/SIGDA 15th International Symposium on Field Programmable Gate Arrays (FPGA'07)* . 148–155.

TUAN, I., KAO, S., RAHMAN, A., DAS, S., AND TRIMBERGER, S. 2006. A 90nm low-power FPGA for battery-powered applications In *Proceedings of the International Symposium on Field Programmable Gate Arrays*. 3-11.

XILINX. 2003. *Virtex-II Pro Platform FPGAs: Functional Description*. Xilinx.

XILINX. 2006. *Virtex-5 Family Overview*. Xilinx.

YANG, S. 1991. Logic synthesis and optimization benchmarks, version 3.0. Tech. rep. Microelectronics Centre of North Carolina. Research Triangle Park, NC.

Received June 2009; revised October 2009; accepted December 2009