POWER CONSUMPTION BENCHMARKING FOR RECONFIGURABLE PLATFORMS

Teemu Pitkänen (Tampere University of Technology, Finland; teemu.pitkanen@tut.fi);
Peter Jamieson (Miami University, Oxford, OH, USA; jamiespa@muohio.edu);
Tobias Becker (Imperial College, London, UK; tobias.becker04@imperial.ac.uk);
Sami Moisio (Nokia Devices R&D, Tampere, Finland; sami.moisio@nokia.com);
Jarmo Takala (Tampere University of Technology, Finland; jarmo.takala@tut.fi)

ABSTRACT

Software Defined Radios (SDR) wideband mobile terminal must be capable of high-speed data processing while consuming low power and keeping the design and manufacturing costs as low as possible. In order to make SDR nodes mobile, the power consumption is the major design issue. Analysis of power consumption for the various implementations is challenging, since each implementation contains its own benchmarking tools and thus the results are not comparable. In this paper, we use the GroundHog 2009 benchmark suite, designed to be platform independent, to evaluate power dissipation of four modern FPGAs and one microcontroller. We also introduce generic RTL library for the GroundHog design cases and test bench infra-structure to make the toolset usage easy. In addition, we show how much power can be saved by using clock management, available on the one of the FGPA-boards. The power savings range from 38% to 1150%.

1. INTRODUCTION

Software Defined Radios (SDR) call for efficient and highly configurable platforms that allow implementation of the rapidly evolving 3G and 4G digital wireless communication standards. There are various implementation technologies, which can be used for such a purpose depending on the requirements of the application at hand, e.g., Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP), General-Purpose Processors (GPP), Application-Specific Integrated Circuits (ASIC), or a combination of these.

FPGAs are often used as SDR platform, since an SDR wideband mobile terminal must be capable of high-speed data processing, low power consumption and low design and manufacturing cost [1, 17]. Especially in the mobile domain the power consumption or average energy dissipation is a critical as it determines the battery life. Also the peak current has to be considered for thermal aspects. The analysis of the power consumption of different implementation methods is challenging as there is large variation between different implementations even on a one particular target technology, i.e., an FPGA can be designed for low leakage current but a poorly designed configuration may consume more power than a good design on an FPGA with higher leakage current.

For fair measurement of the power consumption for different implementation methods, not only the application but also the input of the system should remain the same. However the input of the system should not be one fixed set of inputs, since one might optimize the measured system for a specific case and the results are not realistic.

Evaluation and benchmarking power consumption of the reconfigurable architectures has been achieved by using existing benchmarks [7], such as MCNC [2], or by modeling power consumption using circuit models and in house designs. Much of the previous work [3, 4, 5], allows power measurements on reconfigurable architectures, but they do not realistically model modern applications on these devices. As for earlier benchmarks, such as MCNC, there are no input stimuli and they are implemented in low level design descriptions.

The GroundHog 2009 (GH09) benchmark suite has been developed to fill the gap between existing benchmarks and requirements of the mobile domain applications in the near and far future [7]. The GH09 includes seven designs; one targeting fine-grained FPGA fabrics and six designs that are specified at a high level, allowing them to target a range of reconfigurable technologies. The designs in the GH09 can be stimulated with synthetically generated input and verified against the golden model output created by a tool included in the suite. The suite can be used as a tool to evaluate the power consumption of the different target technologies and it can also give guidance for selecting a suitable method for low power implementation.

In this paper, we use GH09 benchmarks to evaluate power dissipation of four modern FPGAs targeting the mobile domain and one microcontroller. We introduce basic HDL-implementations and test bench infrastructure for the power measurements for each of the high level designs of the GH09. The basic HDL-designs and test bench infrastructure is made publicly available [19] and hopefully

will made the starting to use the GH09 suite more interesting.

The reminder of this paper is organized as follows: Section 2 describes our benchmark suite in details. Section 3 describes the mechanism to feed the input stimulus to the Device Under Test (DUT) and the verification of the DUT. Section 4 describes the measurements carried out for four low power FPGAs and one microcontroller. Finally, Section 5 concludes the paper.

2. 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 advanced low power modes where power is reduced significantly when the device is idling or performing low throughput tasks. GroundHog 2009 is a benchmarking suite that targets reconfigurable architectures in the mobile computation domain with the goal of providing the means to evaluate current and future technologies for low power and increase 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 are used in current and future mobile devices.
- 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.
- Stimulate the designs with actions a system will likely perform in current mobile devices and future mobile devices.
- 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.
- Prevent system or tool optimizations for a specific benchmark, while still encouraging innovation.

GroundHog 2009 has been created as the first attempt to satisfy these challenges. There are four main elements of the benchmark suite that make-up this innovative framework and address many of the 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.

Fabric analysis is included in the benchmark suite to allow our community to quickly evaluate the power consumption of fine-grained FPGAs.

2.1. GroundHog 2009 Bencmark Suite

GroundHog 2009 (GH09) 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 - Encryption
GH09.B.4 - Data compression
GH09.B.5 - Bridge chip
GH09.B.6 - 2D convolution

Designs starting from GH09.B.1 are described in details in chapters: 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5 and 2.1.6.

GH9.B.0 is an application-independent test case designed to evaluate the basic characteristics of the FGPA fabric, the other cases are functional descriptions of applications, which are representative for mobile systems. The GH09.B.0 is left out of the scope of this evaluation; detailed description of this case can be found [18]. These designs were selected because they are simple, but represent the properties of a range of possible designs. For example, the 2D convolution design evaluates how a technology can efficiently implement arithmetic operations. On the other hand, the data compression design evaluates how a technology can efficiently implement memories that are accessed in pseudo random fashion.

These benchmarks were chosen by studying 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 because 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 connectivity 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 encryption 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

```
<event>
    <time>5500ns</time>
    <resource>
        <name>bus</name>
            <value0>00000011</value0>
    </resource>
</event>
<event>
    <time>1000000ns</time>
    <resource>
        <name>spi</name>
        <value0>10100000000000
                0011111111</value0>
    </resource>
</event>
```

Figure 1- Input stimulus in XML format

tools allow benchmark users to create input stimuli, in XML format, to evaluate their solutions. These stimuli can be created to model continuous throughput inputs as well as intermittent stimuli that more closely model the on/off activity within a mobile device. The stimuli is described as an events and time stamps when event occurs. This is illustrated in the figure 1: the event is described with <event> and <time> describes the time when event is occurs, <name> shows which object the event is concerning. An object type is specified within the test design specifications, i.e. serial, parallel or a block of data. <value> identifies an object inside the object and set the value of the object. The XML file also contains information about the design environment, the minimum acceptable heartbeat of the system (the minimum time between events), and serial data arrival. It can also contain conditions for the design, i.e., when the system interrupts what actions the system excepts from the "master" device.

An included tool also provides a golden model simulator to help benchmark users to 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 behaving correctly. The golden model output is also provided in XML. The reference output specifies the earliest time when the output data could arrive, but the latest arrival time is not specified. The data arrival is left to be specified in the environment variables.

As for the input stimuli the time stamp only defines the time when data is ought to be received, not the time when it should be processed. Detailed description of the designs, stimulus generation and environments can be found in our earlier work [7,8,18].

2.1.1 GH09.B.1- Port expander and keypad controller

The design is used to expand the number of ports of the master device and at the same time to work as a keypad controller. The design has a serial data input and output and 16 general-purpose IO ports, which can be programmed through a serial interface. The design contains two interrupt signals, which is used to notify the master device if a key is pressed on the keypad or a general-purpose IO has changed. This test is designed to test control flow of a low performance system.

2.1.2 GH09-B.2- Glue logic

This design is composed of three simple state machines, and each state machine changes its value according to the commands on a 9-bit bus. The command bus can set each state machine to fast, slow or off modes. This test is designed to test control flow type processing by combining both low and high performance.

2.1.3 GH09.B.3 – Encryption

This design is an encryption algorithm based on Advanced Encryption Standard (AES) [20]. A 128-bit cypher is used with a 128-bit cryptographic key and 128-bit data blocks. Input and output data is divided to 16-bit blocks. Input data transmission is controlled by a simple handshaking protocol and the output is set to be ready to read by a one bit clock high pulse. This test is designed to test data processing of high performance systems.

2.1.4 GH09.B.4 - Data compression

This design performs data compression using the Lempel-Ziv-Welch (LZW) lossless data compression algorithm. The design processes a block of 8-bit inputs and outputs a compressed version of that block. The design uses 4k of symbols in the code book, which lead to 12-bit output word. The input is controlled with a simple 2-bit handshaking protocol. This test is designed to test memory orientated high throughput systems.

2.1.5 GH09.B.5 - Bridge chip

This design implements a protocol conversion between a serial bus and a parallel bus. The bridge is a master controlling the serial bus of a slave device and at the same time the bridge is a slave in the parallel bus. The parallel bus is Wishbone direct bus. This test is designed to test a high throughput data oriented system which transfers serial and parallel data.

2.1.6 GH09.B.6 - 2D convolution

This design is a 2-dimensional convolution core where a constant 5×5 matrix is convolved with a 400×400 data matrix, which is entered from an external 16-bit interface controlled by a simple 2-bit handshaking protocol. Matrices are defined in terms of signed fixed point numbers of the

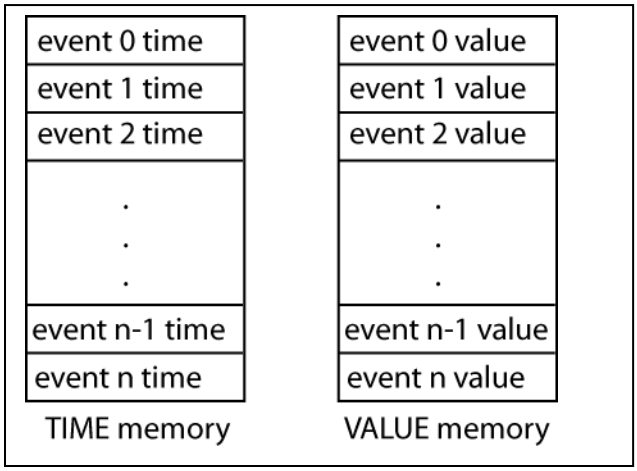

Figure 2Time and Value event memories

form of 8 bits presenting the integer part and eight bits presenting the fractional part. This test is designed to test high throughput, complex arithmetic and data orientated systems

3. TEST BENCH GENERATION

In this section, we describe the principal structure of the test bench for power measurements based on the Groundhog 2009 benchmarks.

The test bench infrastructure is designed to feed the input to the GH09 implementation on a selected device and verify the output of the system. The desired input data is transferred to the memory of the test bench, from which it is read and output to the system. The input to the system can be for example, continuously sampled and the sampling must be done at a frequency higher, preferred multiple values higher, of the system frequency to ensure correct input stimulus. This method consumes a lot of memory in the test bench and since the GH09 stimulus contains the time when certain events should happen starting from the beginning of the stimulus. Now the test bench can calculate time, the time calculation is done in the test bench clock cycles, and sample new data for the system when the time of the event occurs, i.e., the memory needs to store time values and event values. We parsed the XML-stimuli to two vector memories as an event value and event time memory, shown in figure 2. In special cases such as block or serial transmission events an extra memory is created. In the event value and time memory each event is stored to individual lines in the memory; if a special event is triggered the memory contains an extra slot to notify that the next stimulus is going to be loaded from the special memory. The golden model reference memories are generated in a similar manor.

The finite state machine controlling the reading and outputting of the values from the memories contains five

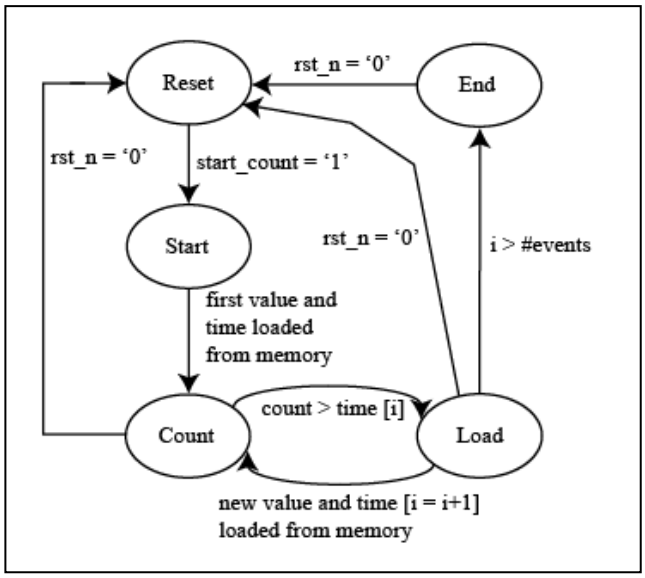

Figure 3 State machine controlling the testbench

states as illustrated by the state diagram shown in Figure 3. The first state "reset" sets the DUT reset in the active state but even in this reset state the clocks are running. When signal "start count" is active, which is externally set either from a pushbutton of the test bench device or a signal from a measurement device as the state machine starts to count cycles and initial values are loaded to the DUT and first values from the time and the value memory are loaded to the test bench. The counting of the time happens in "count" state where the clock cycle counter is increased by one in each clock cycle. When the cycle count is same or greater as the value in the time memory the current event is triggered and state machine goes to "load" state, i.e., the current bit vector loaded from the value memory is fed to the output and new vectors are loaded from the time and value memories. This is repeated until the memories run out of elements and the "stop" signal is activated to inform an external measurement device that end of the measurement is reached, i.e., the state machine goes to "end" state. When low active reset signal ("rst_n") is activated the state machine immediately goes to "reset" state and the system will start over.

The structure of the test bench is presented in figure 4, where stimulus memories are connected to the state machine, presented at fig. 2 and fig 3, and reference memories (golden model) are connected to output verification. When the design case requires an extra memory, i.e. data for the sequential stimulus, the extra memory is connected to the state machine. The output verification indicates whether output of the DUT is correct.
A Phase Locked Loop(PLL) is used to generate the clock for the test bench, which is required to be faster than the clocks fed to the DUT, preferable multiple frequencies. If the PLL is not able to produce the clock frequency that is

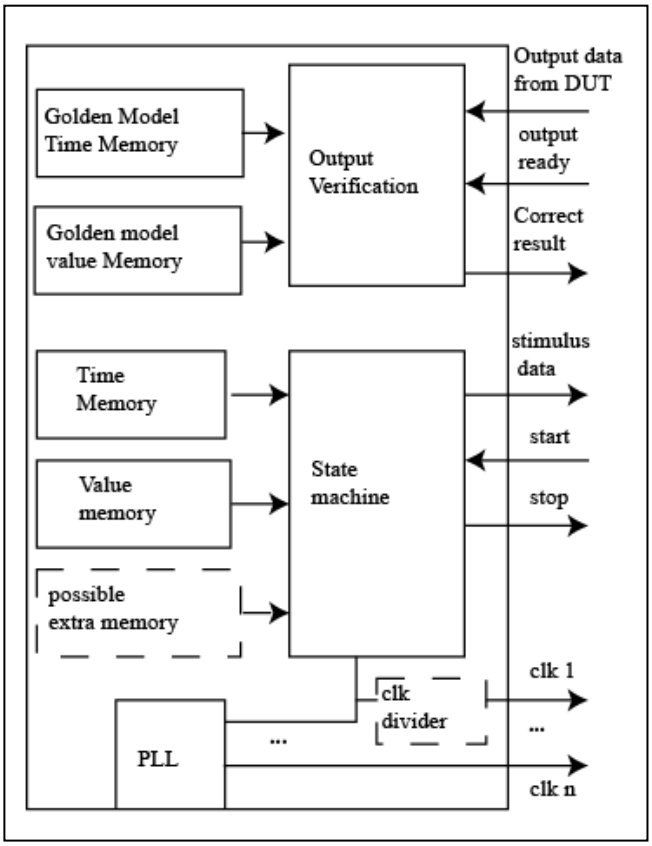

Figure 4 Structure of the testbench

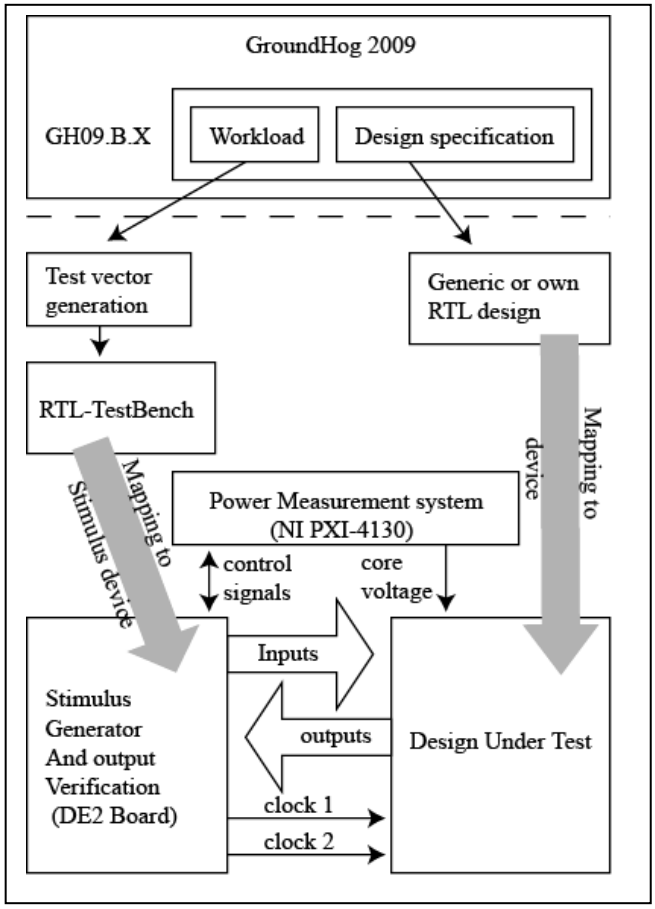

Figure 5 Measurement system

required in the DUT, a clock divider is used in clock generation. Clock divider can be used only with low frequencies, i.e. under 1 MHz, to ensure correct operation.

Figure 4 shows the measurement system used for our power measurements. At the top of the figure is GroundHog 2009 software suite, which outputs is workload and the design specification for selected benchmark. In the middle of the figure is the test vector generation, where the workload, i.e., input stimulus and reference output, is modified to vector format. The input stimuli and output of the golden model are converted to a set of vectors and timestamps, i.e. vector memories, that are then read by an external FPGA board. We use Altera DE2 Development and Education FPGA board[13] as Stimulus generator and output verification device. This FPGA board is connected to the implementation of a design and feeds the input stimuli to the DUT so that power measurements can be made and the output from the design cane be verified against the golden model output. The GH09-suite includes sample environment descriptions, i.e. heart beat of the system and required data rate of serial transmission, which are followed in the test bench creation. These environment descriptions allow designs to target a range of mobile devices.

The shortest time between any consecutive events defines the timing accuracy of the stimulus generation system, if DUT cannot accept new data the test is stopped and an error message is displayed, i.e. the DUT is not able to accept data at the rate specified in the input stimulus. The latest arrival time of the output is not specified in the GH09, so only the correctness of the output is verified, it is left to user to verify that implemented system fulfills the speed requirements according to the environment variables. The design flow from the output of the GH09 suite, i.e., the stimulus in XML-format, to the actual mapping of the test vector generator is automated. The toolset; RTL-test benches, test vector generation, and generic design examples are available at [19].

4. EXPERIMENTS

In our measurements we use four commercially available FPGAs to illustrate how GH09 benchmark toolset can be used to benchmark reconfigurable architectures in terms of power consumption; in addition, we compare benchmark results for a microcontroller. For our experiments, we selected FPGAs, which have been design for low power; SiliconBlue's ICE65L04 and ICE65L08 [9], Actel's AGL600 [10], Lattice MachXO2 1200 [11], and for software implementation we have used a microcontroller

Table 1 Resources available on the FPGAs

FPGA	*LUT/FF*	*Memory* [bits]	*I/O Pins*
AGL600	27.600	108k	235
iCE65L04	7,040	80k	176
iCE65L08	15,378	128k	222
XO2-1200	2,560	64k	108

based system Cypress' PSoC [12]. For the each FPGA circuits, we map the basic RTL implementations of the GH09 benchmarks and measure the power consumption for a particular workload. Unfortunately the available Lattice chip is too small for GH09.B.3 benchmark case. For the microcontroller, we present GH09.B.3 benchmark case. We also present five benchmark cases on the Lattice MachXO2 chip using available advanced methods for low power designing to illustrate how much power can be saved with low power features included in a chip.

The modified stimulus is loaded to Stimulus generator together with the test bench for the case in hand. From the specifications of the selected case, the user can select a design from a basic VHDL RTL library or create their own implementation according to the specifications. The basic implementations are created to help the user to become familiar with the GH09 suite and to have a reference design for each case. The RTL implementation of the case is mapped to the DUT of the selected architecture. We use the National Instruments PXI-1033 chassis [14] with PXI-4130 voltage source [15] and TB-2709 Sampling DAQ [16]. This power measurement system feeds the core voltage to the DUT and measures the core voltage and current drawn by the DUT, the voltage is kept stable by sensing it in order to avoid a voltage drop when more current is required.

Table 1 provides a brief overview of the FPGAs measured; the first column describes the FPGA, and columns two, three and four shows the size of the FPGA, available RAM-memory and the number of the I/O pins. The size of the FPGA is combined number of Look up tables (LUT) and flip flops(FF), this number is not directly comparable, since architectures uses different sized LUTs and FF configurations. The numbers are taken from the datasheets of each vendor.

Following tables, 2-8 present average power dissipation of the GH09 designs with basic RTL, in each case the default parameters for the generation of the workload are used. The parameters effects to the stimuli are described in detail in [8], each table contains average power, resource usage, operational voltage and operating frequency, is some resource is a constant it's notified in the caption of the table.

Table 2 and 3 shows the average power dissipation of GH09.B.1 with three FGPA boards. The table 2 presents

Table 2. Power consumption of 7x8 Keypad of GH09.B.1 with 32 MHz clock frequency and 1.2V supply voltage.

FPGA	*Resource* [%]	P_{AVG} [mW]
AGL600	10	3.54
iCE65L04	28	1.8
XO2-1200	63	1.2

Table 3. Power consumption of GPIO of GH09.B.1 with 32MHz clock frequency and 1.2V supply voltage.

FPGA	*Resource* [%]	P_{AVG} [mW]
AGL600	10	3.55
iCE65L04	28	1.82
XO2-1200	63	1.2

Table 4. Power consumption of GH09.B.2 with 32MHz clock frequency and 1.2V supply voltage.

FPGA	*Resource* [%]	P_{AVG} [mW]
AGL600	1	1.08
iCE65L04	4	1.7
XO2-1200	5	0.79

Table 5. Power consumption of GH09.B.3.

FPGA/ Controller	*Oper. Freq.* [MHz]	V_{CC} [*V*]	*Resource* [%]	P_{AVG} [*mW*]
AGL600	10	1.5	86	19.56
iCE65L08	10	1.2	47	13.13
PSOC-3	32	3.3	-	62.7

power consumption for a 7x8 keypad, where 15 of the 16 available GPIOs are operating as keypad controller. The required response time is 50 ms which is equivalent to 20 key-stokes in a second. In table 3 a workload for the GPIO controller is used where 8 ports are used as an input and 8 ports as an output.

Table 4 present the average power consumption of the GH09.B.2 with three FGPAs. The result shows that with this design and current workload (default workload 0) the power dissipation has variance between devices compared to the earlier case.

Table 5 shows power consumption for the GH09.B.3 with two FGPAs and microcontroller. The AGL600 was unable to run the test with required 10MHz speed with operational voltage of 1.2V, the critical path of the device with 1.5V sufficient. PSOC-3 uses 3.3V operational voltage and the internal clock frequency of the microcontroller is 32MHz, with lower frequency the system is not able to

calculate the AES with the speed workload requires. The test design to the PSOC-3 is written with C. As excepted the microcontroller requires the most power, in this case almost five times more compared to the iCE65L08.

Table 6 presents the power consumption of the GH09.B.4 generic RTL-design. The operational voltage of the AGL600 is required to set to 1.5 V. Here we find significant differences in the power dissipation, which may be explained by the fact that the memories of the AGL600 are less power efficient. The memory consumption of the devices differs since the size of the codebook is set to maximally fit to the device. The codebook width is not defined in the GH09-suite, but the code size has an effect on the compression ratio and the compression results, since the code words are more limited, this is a limitation of the basic implementation, it waste the memory and it can be easily optimized. The memory usage column describes how many bits of RAM are used and how much of the ram available ram blocks are used.

Table 7 provides power dissipation of the GH09.B.5 with three FPGAs. To show the effect of the frequency we added a test with different clock frequency on the XO2 device. The power scales nearly linearly as expected.

Table 8 shows power consumption for the GH09.B.6 with four FGPAs. The operational voltage of the AGL600 is required to set to 1.5 V instead of the 1.2V as the rest of the devices. In here the XO2 takes surprisingly high power, clock tree of the system draws the most of the power.

In the Table 9, we show the power consumption of various power optimized versions of the each GH09 test case with Lattice Mach-XO2. In each design the clock muxes are inserted in the clock tree to control the distribution of the available clocks. Clocks are also internally disabled when specific inputs have no activity. The clock management gives improvements from 38% to 1150%. The improvement of the GH09.B.6 in this particular case when device is performing 2D-convolution operation, it consumes nearly 420 µW and when the clocks are disables power consumption is 70 µW, giving average consumption of 82 µW. The improvement of the GH09.B.1 is also significant, the result is average of the both workloads presented previously, with the 8x7 keypad the savings are bigger, since the input rate of 20 key-stokes in second makes efficient idling possible.

As an additional notice the results provided here is not general comparison between particular FPGAs, but a means to show the differences between different architectures when used with different benchmark cases. Such a comparison could be done with designs optimized individually for specific device make use of the low power features in present in the device. Also the devices contains different amount of features, including for example PLLs and internal oscillators, which consume power in the measurements. Within this study with the basic implementations the extra features is turned off within a device if possible.

Table 6. Power consumption of GH09.B.4 with 10MHz clock frequency.

FPGA	V_{CC} [V]	*Resource* [%]	P_{AVG} [*mW*]	*Memory* [kbits]/[%]
AGL600	1.5	1	9.23	98 / 100
iCE65L04	1.2	10	0.63	65 / 80
iCE65L08	1.2	7	1.19	98 / 100
XO2-1200	1.2	17	0.46	49 / 86

Table 7. Power consumption of GH09.B.5. with 1.2V supply voltage

FPGA	*Oper.Freq* [MHz]	*Resource* [%]	P_{AVG} [mW]
AGL600	10	1	1.76
iCE65L04	10	3	0.49
XO2-1200	10	5	0.31
XO2-1200	32	5	0.91

Table 8. Power consumption of GH09.B.6 with 10MHz clock frequency and 32kbit memory.

FPGA	V_{CC} [*V*]	*Resource* [%]	P_{AVG} [*mW*]	*Memory* [%]
AGL600	1.5	19	5.9	40
iCE65L04	1.2	23	0.59	45
iCE65L08	1.2	11	1.1	28
XO2-1200	1.2	59	1.05	57

Table 9. Power consumption of power optimized designs with Lattice Mach XO2-1200.

Design	*Oper. Freq.* [MHz]	V_{CC} [V]	P_{AVG} [mW]	*Impr.* [%]
GH09.B.1	32	1.2	0.14	~750
GH09.B.2	32	1.2	0.575	37.6
GH09.B.4	10	1.2	0.28	64.3
GH09.B.5	10	1.2	0.09	244.4
GH09.B.6	10	1.2	0.08	~1150

5. CONCLUSION

In this work, we used the GroundHog 2009 benchmarking suite targeting reconfigurable architectures in the mobile domain to measure power dissipation of four modern FPGAs and on microcontroller. We introduced a basic RTL

design library for the suite and created test bench infrastructure to make the GroundHog suite easy to use. We show how easy changes to the basic component can have huge effect on the power dissipation; this can be done with platform supporting power management, such as clock tree manipulation.

The basic RTL-library and test bench infrastructure is available in OpenCores repository located at: http://opencores.org/project,groundhog2009_repository,downloads

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

6. REFERENCES

[1] A. Abidi, "The Path to the Software-Defined Radio Receiver", IEEE Journal of Solid-State circuits, Vol. 42, No. 5, 2007

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

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

[4] 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.

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

[6] T. Becker, P. Jamieson, W. Luk, P.Y.K. Cheung, and T. Rissa., "Towards Benchmarking Energy Efficiency of Reconfigurable Architectures", *18th International Conference on Field Programmable Logic and Applications (FPL'08), 2009* . Heidelberg, Germany

[7] P. Jamieson, T. Becker, W. Luk, P.Y.K. Cheung, T. Rissa and T. Pitkänen. "Benchmarking Reconfigurable Architectures in the Mobile Domain", IEEE Symposium on Field-Programmable Custom Computing Machines, 2009.

[8] P. Jamieson, T. Becker, P.Y.K. Cheung, W. Luk, T. Rissa and T. Pitkänen, "Benchmarking and Evaluating Reconfigurable Architectures in the Mobile Domain", ACM Transactions on Design Automation of Electronic Systems, vol. 15, no. 2, 2010.

[9] SiliconBlue. iCE DiCE: iCE65L04 Ultra Low-Power FPGA Known Good Die, March 2009.

[10] Actel, Igloo Handbook, Jan 2008

[11] Lattice, Mach XO2: The do-it-All PLD for Low Density Applications, 2011

[12] Cypress, PSoC 3: CY8C38 Family Data sheet, May 2011

[13] Altera DE2 Development and Education board user manual, 2006

[14] National Instruments, PXI-1033 Chassis, Datasheet, 2010

[15] National Instruments, PXI-4130 Power SMU, Datasheet, 2010

[16] National Instruments, TB-2709 Sampling DAQ, Datasheet, 2010

[17] K. Akabane, H. Shiba, M. Matsui, M. Umehira and K. Uehara, "Performance Evaluation of Reconfigurable Processor for SDR Mobile Terminals and SDR Base Station using Autonomous Adaptive Control Technology", ICICS, pages 148-152, 2005

[18] T.Becker, P. Jamieson, P.Y.K Cheung and T. Rissa, "Power Characterization for Fine-Grain Reconfigurable Fabrics", International Journal of Reconfigurable Computing, vol. 2010 Article ID 787405, 9 pages, 2010.

[19] OpenCore, project GroundHog 2009, Downloads, http://opencores.org/project,groundhog2009_repository,downloads

[20] "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, November, 2001

.