Measuring the Gap Between FPGAs and ASICs

Ian Kuon and Jonathan Rose
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON

{ikuon,jayar}@eecg.utoronto.ca

ABSTRACT

This paper presents experimental measurements of the differences between a 90nm CMOS FPGA and 90nm CMOS Standard Cell ASICs in terms of logic density, circuit speed and power consumption. We are motivated to make these measurements to enable system designers to make better informed choices between these two media and to give insight to FPGA makers on the deficiencies to attack and thereby improve FPGAs. In the paper, we describe the methodology by which the measurements were obtained and we show that, for circuits containing only combinational logic and flip-flops, the ratio of silicon area required to implement them in FPGAs and ASICs is on average 40. Modern FPGAs also contain "hard" blocks such as multiplier/accumulators and block memories and we find that these blocks reduce this average area gap significantly to as little as 21. The ratio of critical path delay, from FPGA to ASIC, is roughly 3 to 4, with less influence from block memory and hard multipliers. The dynamic power consumption ratio is approximately 12 times and, with hard blocks, this gap generally becomes smaller.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Types and Design Styles

General Terms

Design, Performance, Measurement

Keywords

FPGA, ASIC, Area Comparison, Delay Comparison, Power Comparison

1. INTRODUCTION

We were motivated to measure the area, performance and power consumption gap between field-programmable gate arrays (FPGAs) and standard cell application-specific integrated circuits (ASICs) for the following reasons:

1. In the early stages of system design, when system architects choose their implementation medium, they often choose between FPGAs and ASICs. Such decisions are based on the differences in cost (which is related to area), performance and power consumption between these implementation media but to date there have been few attempts to quantify these differences. A system architect can use these measurements to assess whether implementation in an FPGA is feasible. These measurements can also be useful for those building ASICs that *contain* programmable logic, by quantifying the impact of leaving part of a design to be implemented in the programmable fabric.
2. FPGA makers seeking to improve FPGAs can gain insight by quantitative measurements of these metrics, particularly when it comes to understanding the benefit of less programmable (but more efficient) hard heterogeneous blocks such as block memory [3, 17, 28] multipliers/accumulators [3, 17, 28] and multiplexers [28] that modern FPGAs often employ.

In this paper we focus on a comparison between a 90 nm CMOS SRAM-programmable FPGA and a 90 nm CMOS standard cell technology. We chose an SRAM-based FPGA because that approach by far dominates the market, and it was necessary to limit the scope of comparison in order to make this work tractable. Similarly, standard cells [8, 21] are currently the dominant choice in ASIC implementations versus pure gate arrays and the newer "structured ASIC" platforms [18, 19].

We present these measurements knowing that some of the methodology used will be controversial. We will carefully describe the comparison process so that readers can form their own opinions of the validity of the result. As always, the set of benchmarks we use are highly influential on the results, and indeed any given FPGA vs. ASIC comparison can vary significantly based on the application, as our results show. Since we perform measurements using a large set of designs, it was not feasible to individually optimize each design and it is likely that manual optimizations or greater tuning of the tools could yield improved results for any individual design; however, this is true for both the ASIC and FPGA platforms. We believe our results are more meaningful than past comparisons because we do consider a range of benchmarks instead of focusing on just a single design as has been done in most past analyses.

This paper is organized as follows: Section 2 describes previous work on measuring the gap between FPGAs and ASICs. Section 3 details the experimental methodology we

FPGA'06, February 22–24, 2006, Monterey, California, USA.

use in this work. The approach is a fundamentally empirical one in which the same circuits are implemented through two computer-aided design (CAD) flows which are described in Sections 4 and 5. Section 6 gives a precise definition of the comparison metrics. Section 7 presents the comparison results, and Section 8 concludes the paper.

2. PAST FPGA TO ASIC COMPARISONS

There have been a small number of past attempts to quantify the gap between FPGAs and ASICs which we will review here.

One of the earliest statements quantifying the gap between FPGAs and pre-fabricated media was by Brown *et al.* [4]. That work reported the logic density gap between FPGAs and Mask-programmable Gate Arrays (MPGAs) to be between 8 to 12 times, and the circuit performance gap to be approximately a factor of 3. The basis for these numbers was a cursory comparison of the largest available gate counts in each technology, and the anecdotal reports of the approximate operating frequencies in the two technologies at the time. While the latter may have been reasonable, the former suffered from optimistic gate counting in FPGAs.

In this paper we are seeking to measure the gap against standard cell implementations, rather than the less common MPGA. MPGAs have lower density relative to standard cells, which are on the order of 33% to 62% smaller and 9% to 13% faster than MPGA implementations [15]. Aside from the reliance on anecdotal evidence, the analysis in [4] is dated since it does not include the impact of hard dedicated circuit structures such as multipliers and block memories that are now common [3, 28]. In this work, we address this issue by explicitly considering the incremental impact of such blocks.

More recently, a detailed comparison of FPGA and ASIC implementations was performed by Zuchowski *et al.* [30]. They found that the delay of an FPGA lookup table (LUT) was approximately 12 to 14 times the delay of an ASIC gate. Their work found that this ratio has remained relatively constant across CMOS process generations from 0.25 µm to 90 nm. ASIC gate density was found to be approximately 45 times greater than that possible in FPGAs when measured in terms of kilo-gates per square micron. Finally, the dynamic power consumption of a LUT was found to be over 500 times greater than the power of an ASIC gate. Both the density and the power consumption exhibited variability across process generations but the cause of such variability was unclear. The main issue with this work is that it also depends on the number of gates that can be implemented by a LUT. In our work, we remove this issue by instead focusing on the area, speed and power consumption of application circuits.

Wilton *et al.* [27] also examined the area and delay penalty of using programmable logic. The approach taken for the analysis was to replace part of a non-programmable design with programmable logic. They examined the area and delay of the programmable implementation relative to the non-programmable circuitry it replaced. This was only performed for a single module in the design consisting of the next state logic for a chip testing interface. They estimated that when the same logic is implemented on an FPGA fabric and directly in standard cells, the FPGA implementation is 88 times larger. They measured the delay ratio of FPGAs to ASICs to be 2.0 times. Our work improves on this by comparing more circuits and using an actual commercial FPGA for the comparison.

Compton and Hauck [11] have also measured the area differences between FPGA and standard cell designs. They implemented multiple circuits from eight different application domains, including areas such as radar and image processing, on the Xilinx Virtex-II FPGA, in standard cells on a 0.18 µm CMOS process from TSMC, and on a custom configurable platform. Since the Xilinx Virtex-II is designed in 0.15 µm CMOS technology, the area results are scaled up to allow direct comparison with 0.18 µm CMOS. Using this approach, they found that the FPGA implementation is only 7.2 times larger on average than a standard cell implementation. The authors believe that one of the key factors in narrowing this gap is the availability of heterogeneous blocks such as memory and multipliers in modern FPGAs and, in our work, we quantify these claims.

While the present work aims to measure the gap between FPGAs and ASICs, it is noteworthy that the area, speed and power penalty of FPGAs is even larger when compared to the best possible custom implementation using full-custom design. It has been observed that full-custom designs tend to be 3 to 8 times faster than comparable standard cell ASIC designs [8]. In terms of area, a full-custom design methodology has been found to achieve 14.5 times greater density than a standard cell ASIC methodology [12]. Finally, the power consumption of standard cell designs has been observed as being between 3 to 10 times greater than full-custom designs [7, 9].

3. NEW FPGA TO ASIC COMPARISON

The measurements of the gaps between between FPGAs and ASICs described in the previous section were generally based on simple estimates or single-point comparisons. To provide a more definitive measurement, our approach is to implement a range of benchmark circuits in FPGAs and standard cells with both designed using the same IC fabrication process geometry. The two implementations are then compared in terms of silicon area, maximum operating frequency and power consumption.

This comparison was performed using 90 nm CMOS technologies to implement a large set of benchmarks. We selected the Altera Stratix II [3] FPGA based on the availability of specific device data [10]. This device is fabricated using TSMC's Nexsys 90 nm process [1]. The IC process we use for the standard cells is STMicroelectronic's CMOS090 Design Platform [22]. This platform offers standard cell libraries optimized for speed or density and both high-V_T and standard-V_T versions are available. While the TSMC and STMicroelectronics processes are not identical, we believe they are sufficiently similar to allow them to be compared in this work. The results from both platforms will assume a nominal supply voltage of 1.2 V.

3.1 Benchmark Selection

The selection of benchmarks can significantly impact the results and, therefore, before considering how these benchmarks are implemented, we describe how the benchmarks were initially selected. We considered a variety of benchmarks (coded in either Verilog or VHDL) from a range of sources including publicly available designs from Opencores (`http://www.opencores.org/`) and designs developed for projects at the University of Toronto.

There were two critical factors that had to be considered in benchmark selection. The first was ensuring that the Verilog or VHDL RTL was synthesized similarly by the different tools used for FPGA and ASIC implementation. We did not have access to a single synthesis tool that could adequately target both platforms. Therefore, we had to ensure that the results from the two synthesis tools were sufficiently similar. To check this, we compared the number of registers inferred by the two synthesis processes, which we describe in Sections 4 and 5.1. We rejected any design in which the register counts deviated by more than 5%. Some differences in the register count are expected because different implementations are appropriate on the different platforms. For example, FPGA designs tend to use one-hot encodings for state machines because of the low incremental cost for flip-flops.

The other issue impacting benchmark selection was ensuring that the designs can make use of the block memories and dedicated multipliers on the Stratix II. This is important because one of the aims of our work is analyzing the improvements possible when these hard dedicated blocks are used. However, not all designs will use such features which made it important to ensure that the set of benchmarks include both cases when these hard structures are used and not used.

Based on these two factors, the set of benchmarks in Table 1 were selected for use in this work. To provide an indication of the size of the benchmarks, the table also lists the number of Altera Stratix II ALUTs, 9x9 multipliers and memory bits used by each design. The ALUT is slightly more powerful than the traditional 4-input LUT-based logic block [3]. The 9x9 multipliers are the smallest possible division of the Stratix II's DSP block. These basic blocks can be combined to form larger multipliers (four can be used to make an 18x18 multiplier and eight are needed to make a 36x36 multiplier). While all the benchmarks are relatively modest in size, we believe that the circuits are sufficiently large to give us an accurate measure of the gap between FPGAs and ASICs.

Table 1: Benchmark Summary

Design	ALUTs	Total 9x9 Multipliers	Memory Bits
booth	68	0	0
rs_encoder	703	0	0
cordic18	2 105	0	0
cordic8	455	0	0
des_area	595	0	0
des_perf	2 604	0	0
fir_restruct	673	0	0
mac1	1 885	0	0
aes192	1 456	0	0
fir3	84	4	0
diffeq	192	24	0
diffeq2	288	24	0
molecular	8 965	128	0
rs_decoder1	706	13	0
rs_decoder2	946	9	0
atm	16 544	0	3 204
aes	809	0	32 768
aes_inv	943	0	34 176
ethernet	2 122	0	9 216
serialproc	680	0	2 880
fir24	1 235	50	96
pipe5proc	837	8	2 304
raytracer	16 346	171	54 758

4. FPGA CAD FLOW

The Altera Quartus II v5.0SP1 FPGA software was used for all stages of the CAD flow. Synthesis is performed using Quartus II Integrated Synthesis (QIS). Quartus II was configured to perform `balanced` optimization which optimizes the speed of timing critical portions of the design and area for the remainder of the design. When large memories were required, they were coded by explicit instantiation in the RTL using an appropriate configuration of Altera's `altsyncram` design library function. To enable further optimization of the design, QIS was left in its default configuration in which it automatically instantiates ROMs, RAMs and DSP blocks (the latter of which contains hard multiply-accumulate circuits) when needed. All other options were also left at their default setting.

Placement and routing with the Quartus II "fitter" was performed using the "Standard Fit" effort level. This is the highest effort level and the tool attempts to obtain the best possible timing results irrespective of any timing constraints [2]. We rely on this high effort level to produce the fastest design possible, since we do not constrain the design with any timing constraints. Similar results were obtained when the design was constrained to an unattainable 1 GHz. After placement and routing, the Quartus Timing Analyzer was used for static timing analysis.

In this flow, we allow the fitter to select the specific Stratix II device used; however, we restrict the selection process to use only the fastest speed grade parts. FPGA manufacturers test the speed of their parts after manufacturing and then bin the parts into typically three different speed grades which capture different portions of the manufacturing range of the process. An ASIC's delay is based on the worst case temperature, voltage and process since ASIC parts generally are not speed binned; therefore, using the fastest speed grade devices could arguably favour the FPGA. To address this issue, we will also present results using the slowest speed grade parts. Device selection is also important because it affects the available resources. For industrial FPGA designs, generally the smallest (and cheapest) part would be selected. As will be described later, our FPGA to ASIC comparison optimistically ignores the issue of device size granularity.

It is also important to note that the final operating frequency of the design can vary depending on the random seed given to the placement tool. Therefore, we repeated the entire FPGA CAD flow five times using five different seeds. Any results we report are derived from the placement that resulted in the fastest operating frequency.

5. ASIC CAD FLOW

The standard cell CAD flow is significantly more complicated than the relatively push-button approach for FPGAs. The flow was built around tools from Cadence and Synopsys that were provided by CMC Microsystems (`http://www.cmc.ca`). We relied on vendor documentation, tutorials created by CMC Microsystems and tool demonstration

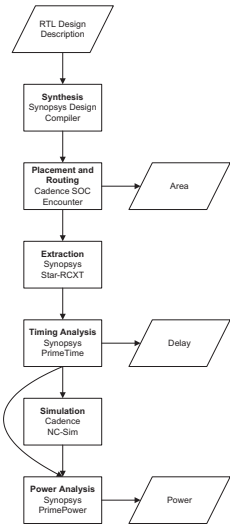

Figure 1: ASIC CAD Flow

sessions provided by the vendors to determine how best to use these tools. Figure 1 illustrates the steps in the CAD flow and this section describes each in greater detail.

5.1 ASIC Synthesis

In the ASIC flow, synthesis was performed using Synopsys Design Compiler V-2004.06-SP1. A common compile script was used for all the benchmarks. The approach for the compilation was a top-down approach [23] in which all the modules starting from the top-level module down are compiled together. This preserves the design hierarchy. It is a reasonable approach because individually the benchmarks have relatively modest sizes and therefore, neither CPU time nor memory size is a significant issue during compilation. The script starts by first analyzing the hardware description language (HDL) source files for the benchmark and then elaborating and linking the top level module. Next the constraints for the compilation are applied.

As a starting point, the clocks in the design are initially constrained to operate at 2 GHz. This unrealistic constraint ensures that the compilation attempts to achieve the fastest clock frequency possible for each of the circuits. To ensure the area of the design remains reasonable, the maximum area is constrained to 0. This is a standard approach [23] for ensuring area optimization despite the fact that it too is an unreasonable constraint.

The ST 90 nm process design kit available to us includes four different standard cell libraries. Two of the libraries are designed for area efficiency but there are relatively few cells in these libraries. The other two libraries are optimized for speed. In each of the two cases, area or speed optimization, one of the libraries uses a low leakage high-V_T implementation while the other library uses higher performing standard-V_T transistors. In Synopsys Design Compiler, we set all four libraries as the target libraries which means that it can select cells from any of these libraries as it sees fit.

After setting the optimization constraints and target cells, the design is compiled using Design Compiler's `high-effort` compilation setting. After this full compilation is complete, a high-effort incremental mapping is performed. This compilation either maintains or improves performance by performing gate-level optimizations [24].

For any modern design, Design for Testability (DFT) techniques are necessary to enable testing for manufacturing defects. In standard cell ASICs, it is customary to use scan chains to facilitate testing [26]. These scan chains require that all sequential cells in a design are replaced with their scan-equivalent implementation and, therefore, in all the compilations performed with Design Compiler we make use of its Test Ready Compile option which performs this replacement automatically. For the FPGA-based implementation, testing is performed by the manufacturer and the inherent programmability of the FPGA generally means no extra circuitry is required.

Once these two compilations have been performed, a reasonable operating frequency for the clocks in the design should be available. The desired clock period for each clock in the design is then adjusted from the unrealistic 0.5 ns constraint to the critical path delay that was obtained in the compilation. A final high effort compilation is performed using these realistic clock constraints. For this final compilation, we enable `sequential area recovery` optimizations which allows Design Compiler to remap sequential elements that are not on the critical path in order to save area. Following this compilation, scan-chains are inserted to connect the scan-enabled flip flops. Finally, once the scan chains have been inserted, the final netlist and the associated constraints are saved for use by the placement and routing tools.

In cases where the benchmark circuits required memories, the appropriate memory cores were generated by STMicroelectronics' memory compilers. CMC Microsystems and Circuits Multi-Projets (CMP) (`http://cmp.imag.fr`) coordinated the generation of these memory cores with STMicroelectronics. We chose to use low power 1.2 V memories, which resulted in memories that were significantly slower than regular memories. (We were not able to obtain higher speed memories in time for this work) Within this low power class of memories, we selected compilers for higher speed over higher density or further reduced power consumption. We also opted to make the memories as square as possible. The models provided for the memories did not exactly match our voltage and temperature analysis conditions and, to account for this, we scaled the delay and power measurements using scaling factors determined by performing HSPICE simulations.

In all of the compilations, no effort was made to optimize the power consumption of the design. This is likely atypical for modern designs but we believe it ensures a fair comparison with the FPGA implementation. With the current FPGA CAD tools, power optimization is not an option and, therefore, using tools such as Synopsys Power Compiler to optimize the standard cell designs would demonstrate an excessively large disparity between the approaches.

5.2 ASIC Placement and Routing

The netlist and constraints produced by synthesis were placed and routed with Cadence SOC Encounter GPS v4.1.5. The flow was adapted from the one described in the Encounter Design Flow Guide and Tutorial [5] and will be described below.

The sizes of our benchmark designs allow us to avoid the hierarchical chip floor-planning steps required for large designs. Instead, we implement each design as an individual block and we do not perform any design partitioning. We

found this approach to be reasonable in terms of run time and memory size for our benchmarks.

The first step was to create a floorplan; a key decision here was to set the target row utilization to 85% and the target aspect ratio to 1.0. Row utilization is the percentage of the area required for the standard cells relative to the total row area allocated for placement. A high row utilization minimizes wasted area but makes routing more difficult. We selected a target of 85% to minimize any routing problems. This is intentionally below higher utilizations of > 85 % that make placement and routing more challenging [29]. We encountered difficulty placing and routing the circuits with the large memory macro blocks; therefore, the target row utilization in those benchmarks was reduced to 75%. After the floorplanning with these constraints, placement is performed. This placement is timing-driven using the worst-case timing models. After placement, scan chain reordering is performed to reduce the wirelength required for the scan chain.

Next, the placement is refined using a built-in congestion optimization command and Encounter's `optDesign` macro command. This macro command performs optimizations such as buffer additions, gate resizing and netlist restructuring. After these optimizations, the clock tree is inserted. Setup and hold time violations are then corrected using the new true clock tree delay information. Once the violations are fixed, filler cells are added to the placement in preparation for routing.

Routing is performed using Encounter's Nanoroute engine. We allow the router to use the seven metal layers available in the STMicroelectronics process. After routing completes, we add any metal fill required to satisfy metal density requirements. A detailed extraction is then performed. This extraction is not of the same quality as the sign-off extraction but is sufficient for guiding the timing-driven optimizations. The extracted information is used to perform post-routing optimizations that are focused on improving the critical paths. These optimizations include drive strength adjustments. After these in-place optimizations, routing is again performed and the output is again checked for any connectivity or design rule violations. The final netlist is then saved in various forms as required for the subsequent steps in the CAD flow.

5.3 Extraction and Timing Analysis

With our current tool and technology kit setup, the RC information we provide to SOC Encounter GPS is not suitable for the final timing and power analysis. Therefore, after the final placement and routing is complete, a final sign-off quality extraction is performed using Synopsys StarRCXT V-2004.06. This final RC extraction is saved for use in final timing and power analysis by Synopsys PrimeTime SI version X-2005.06 and Synopsys PrimePower version V-2004.06SP1 respectively.

6. COMPARISON METRICS AND MEASUREMENT METHOD

Once the designs were implemented using both the ASIC and FPGA approaches and we were confident that the implementations were directly comparable, the area, delay and power of the designs were compared. In this section, we give a precise definition of each metric and the method used for measurement.

6.1 Area

Determining the area of the standard cell implementation is straightforward as it is simply the final core area of the placed and routed design. For the FPGA, the area is calculated using the actual silicon area of each of the resources *used* by the design. This means that we take the final number of Altera Stratix II resources including the basic logic LABs, the M512, M4K, MRAM memories and DSP blocks and multiply each by the silicon area of that specific block[10]. This includes the area for the routing surrounding each of the blocks. The entire area of a block is used regardless of whether only a portion of the block is used. For example, if only a single memory bit were used in one of the large 589 824-bit MRAM blocks we would include the entire area of the MRAM block. We recognize that this approach may be considered optimistic for a few reasons. First, it ignores the fact that FPGAs unlike ASICs are not available in arbitrary sizes. A designer is forced to select one particular discrete size even if it is larger than required for the design. While this is an important factor, our goal is to focus on the cost of programmable fabric itself; therefore, we believe, it is acceptable to ignore any area wasted due to the discrete nature of FPGA device families. Related to this, is the fact that we are also handling the heterogeneity of the FPGA optimistically. With commercial FPGAs, a designer is forced to tolerate fixed ratios of logic, memories and multipliers. Again, since our focus is on the cost of programmability itself, we consider it acceptable to ignore the impact of the fixed heterogeneous block ratios.

For both implementation media, we do not consider the impact of any input or output cells. As well, to avoid disclosing any proprietary information, no absolute areas will be reported in this work; instead, we will only report the ratio of the FPGA area to the ASIC area.

6.2 Speed

Static timing analysis was used to measure the critical path of the each design. This timing analysis determines the maximum clock frequencies for each design. In the case of the eth_top benchmark which contains multiple clocks, we compare the geometric average of all the clocks in each implementation. Timing analysis for the FPGA was performed using the timing analysis integrated in Quartus II. For the standard cell implementation, Synopsys PrimeTime SI (which accounts for delay due to cross-talk) was used with the worst-case timing models.

6.3 Power

Power has become one of the most important issues separating FPGA and ASIC designs but it is one of the most challenging metrics to compare. In this section, we first describe how we measure the static and dynamic components of a design's power consumption. The two contributions are separated both to simplify the analysis and because we are only able to report meaningful results for the dynamic power consumption comparison. In an attempt to ensure a fair and useful comparison, we adjusted the measurements of the static power and we describe our adjustments later in this section so as to explain the limited static power consumption results we are able to report.

6.3.1 Dynamic and Static Power Measurement

The preferred measurement approach, particularly for dynamic power measurements, is to stimulate the post-placed

and routed design with designer-created testbench vectors. For the present work, we take this approach when an appropriate testbench is available for a benchmark, and the result tables will indicate if this was possible. Useful testbenches are generally not available and, in those cases, we use a less accurate approach that relies on arbitrary settings of toggle rates and static probabilities for the nets in the designs.

All the power measurements were taken at a junction temperature of 25 °C using typical silicon. Both the FPGA *and* ASIC implementations are simulated at the *same* operating frequency to allow us to directly compare the dynamic power consumptions. The operating frequency for all the designs was kept constant at 33 MHz. This frequency was selected since it was a valid operating frequency for all the benchmark designs on both platforms. We now describe the process used to generate the power consumption estimates for the FPGA and ASIC designs.

For the FPGA implementation, the placed and routed design is exported as a netlist along with the appropriate delay annotations from Quartus II. If there are test bench vectors available for the benchmark, then digital simulation is performed using Mentor Modelsim 6.0c, which creates a Value Change Dump (VCD) file containing the switching activities on all circuit nodes. The Quartus Power Analyzer reads this file and determines the static and dynamic power consumption. The activities are computed with glitch filtering enabled so that transitions that do not fully propagate through the routing network are ignored. Since we are only focused on the programmable fabric in this investigation, only core power (supplied by `VCCINT`) reported by the power analyzer is considered. The power analyzer breaks this power consumption down into static and dynamic components.

For the standard cell design, simulation of the placed and routed netlist with back-annotated timing is performed using Cadence NC-Sim 5.40. This also produces a VCD file capturing the states and transitions for all circuit nodes in the design. This file, along with the parasitic information extracted by Star-RCXT, is used to perform power analysis with the Synopsys PrimePower tool, version V-2004.06SP1. For this dynamic analysis, PrimePower automatically handles glitches by scaling the power when the interval between toggles is less than the rise and fall delays of the net. PrimePower also divides the power consumption into the static and dynamic components.

For most designs, proper testbenches were not available. In such cases, power measurements were taken by assuming all the nets in all designs toggle at the same frequency and that all the nets have the same static probability. While this is not realistic, it provides a rough estimate of the power consumption differences between implementations. When this approach is used for measurements, it is noted. We chose this approach over statistical vectorless estimation techniques that use toggle rates and static probabilities at input nodes to estimate the toggle rates and probabilities throughout the design because the two power estimation tools produced significantly different activity estimates.

6.3.2 Dynamic and Static Power Comparison Methodology

We believe that the ASIC and FPGA dynamic power consumption measurements can directly be compared but the static power consumption requires adjustment before a reliable comparison is possible. This adjustment is necessary because many of the benchmarks do not fully utilize a specific FPGA device. To account for this, the static power consumption reported by the Quartus Power Analyzer is scaled by the fraction of the core FPGA area that the circuit uses. This decision is arguable as any purchaser of an FPGA is necessarily limited to specific devices and, therefore, would indeed incur the extra static power consumption. However, this device quantization effect obscures the underlying properties that we seek to measure, and changes depending on an FPGA vendor's decision on how many devices to put in an FPGA family. We also anticipate that future generations of FPGAs will allow the power shutdown of unused portions of the devices.

To be clear, we give a hypothetical example of the fractional static power calculation: if a circuit used 1 LAB and 1 MRAM block occupying a hypothetical area of 101 μm^2 on an FPGA that contained a total of 10 LABs and 2 MRAM blocks occupying an area of 210 μm^2, we would multiply the reported static power consumption by $101/210 = 0.48$ to obtain the static power consumption used for comparison purposes. This approach assumes the leakage power is approximately proportional to the total transistor width of a design which is reasonable based on [14] and that the area of a design is a linear function of the total transistor width.

It is important to note that these measurements compare the power consumption gap as opposed to *energy* consumption gap. An analysis of the energy consumption gap would have to reflect the slower operating frequencies of the FPGA. The slower frequency means that more time or more parallelism would be required to perform the same amount of work as the ASIC design. To simplify the analysis in this work, only the power consumption gap will be considered.

7. RESULTS

The measurement methodology described above was applied to each of the benchmarks listed in Table 1, and the metrics were compared. In the following sections, the area, delay and power gap between FPGAs and ASICs will be reported and discussed.

7.1 Area

The area gap between FPGAs and ASICs for the benchmark circuits is summarized in Table 2. The gap is reported as the factor by which the area of the FPGA implementation is larger than the ASIC implementation. As described previously, this gap is sensitive to the benchmarks' use of heterogeneous blocks (memory and multipliers) and the results in the table are categorized in four ways: Those benchmarks that used only the basic logic fabric of clusters of LUTs are labelled "Logic Only." Those that used logic clusters and hard DSP blocks containing multiplier-accumulators are labelled "Logic and DSP." Those that used clusters and memory blocks are labelled "Logic and Memory," and finally those that used all three are labelled "Logic, DSP and Memory". We implemented the benchmarks that contained multiplication operations with and without the hard DSP blocks so results for these benchmarks appear in two columns, and allow the direct measurement of the benefit of these blocks.

First, consider those circuits that only use the basic logic LUT clusters: the area required to implement these circuits in FPGAs compared to standard cell ASICs is on average a factor of 40 times larger, with the different designs covering a range from 23 to 55 times. This is significantly larger than

the area gap suggested by [4], which used extant gate counts as its source. It is much closer to the numbers suggested by [30].

We can confirm the plausibility of this larger number based on our recent experience in designing and building complete FPGAs [16, 20]. As part of this work, we created a design similar to the Xilinx Virtex-E, a relatively modern commercial architecture. If we consider such a design, only the lookup tables and flip-flops perform the basic logic operations that would also be necessary in a standard cell design. The FPGA however also requires additional circuitry to enable programmable connections between these lookup tables and flip-flops. This excess circuitry is the fundamental reason for the area gap. Using our model of the Virtex-E, we calculated that the LUT and flip-flop only take up 3.4 % of the total area for a Virtex-E cluster and its neighbouring routing. The absolute area in the standard cell design required to implement the functionality implemented by the LUT and flip-flop will be similar to area for the FPGA's LUT and flip-flop. This suggests the area gap should be at least 100%/3.4% = 29. This is similar to our experimental measurement.

The hard heterogeneous blocks do significantly reduce this area gap. As shown in Table 2, the benchmarks that make use of the hard multiplier-accumulators and logic clusters are on average only 28 times larger than an ASIC. When hard memories are used, the average of 37 times larger is slightly lower than the average for regular logic and when both multiplier-accumulators and memories are used, we find the average is 21 times. Comparing the area gap between the benchmarks that make use of the hard multiplier-accumulator blocks and those same benchmarks when the hard blocks are not used best demonstrates the significant reduction in FPGA area when such hard blocks are available. In all but one case the area gap is significantly reduced[1]. This reduced area gap was expected because these heterogeneous blocks are fundamentally similar to an ASIC implementation with the only difference being that the FPGA implementation requires a programmable interface to the outside blocks and routing.

These results demonstrate the importance of the introduction of these heterogeneous blocks in improving the competitiveness of FPGAs. It is important to recall that for these heterogeneous blocks, the analysis is somewhat optimistic for the FPGAs. As described earlier, we only consider the area of blocks that are used, and we do not consider the fixed ratio of logic to heterogeneous blocks that a user is forced to tolerate and pay for.

It is noteworthy that significant variability in the area gap is observed in the benchmarks that make use of the heterogeneous blocks. One contributor to this variability is the varying amounts of heterogeneous content. Our classification system is binary in that a benchmark either makes use of a hard structure or it does not but this fails to recognize the varying amounts of heterogeneity in the benchmarks. To address this, we can consider the fraction of a design's area that is used by heterogeneous blocks. If we consider only the benchmarks that employ DSP blocks, we find that the percentage of the total area which is used by DSP blocks exhibits a correlation of -0.87 with the area gap measurement. This relatively strong inverse correlation corresponds with our expectations since as the DSP area content is increased the design becomes more like a standard cell design thereby resulting in a reduced area gap.

[1]The area gap of the rs_decoder1 increases when the multiplier-accumulator blocks are used. This is atypical and it appears to occur because the 5 bit by 5 bit multiplications in the benchmark are more efficiently implemented in regular logic instead of the Stratix II's 9x9 multiplier blocks.

Table 2: Area Ratio (FPGA/ASIC)

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	33			
rs_encoder	36			
cordic18	26			
cordic8	29			
des_area	43			
des_perf	23			
fir_restruct	34			
mac1	50			
aes192	49			
fir3	45	20		
diffeq	44	13		
diffeq2	43	15		
molecular	55	45		
rs_decoder1	55	61		
rs_decoder2	48	43		
atm			93	
aes			27	
aes_inv			21	
ethernet			34	
serialproc			42	
fir24				9.8
pipe5proc				25
raytracer				36
Geomean	40	28	37	21

7.2 Speed

The speed gap for the benchmarks used in this work is given in Table 3. The table reports the ratio between the FPGA's critical path delay relative to the ASIC for each of the benchmark circuits. As was done for the area comparison, the results are categorized according to the types of heterogeneous blocks that were used on the FPGA.

Table 3 shows that, for circuits with logic only, the average FPGA circuit is 3.2 times slower than the ASIC implementation. This generally confirms the earlier estimates from [4], which were based on anecdotal evidence of circa-1991 maximum operating speeds of the two approaches. However, these results deviate substantially from those reported in [30], which is based on an apples-to-oranges LUT-to-gate comparison.

For circuits that make use of the hard DSP multiplier-accumulator blocks, the average circuit was 3.4 times slower in the FPGA than in an ASIC, and in general the use of the hard block actually slowed down the design as can be seen by comparing the second and third column of Table 3. This result is surprising since one would expect the faster hard multipliers to result in faster overall circuits. We examined each of the circuits that did not benefit from the hard multipliers to determine the reason this occurred. For the

Table 3: Critical Path Delay Ratio (FPGA/ASIC) - Fastest Speed Grade

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	4.8			
rs_encoder	3.5			
cordic18	3.6			
cordic8	1.8			
des_area	1.8			
des_perf	2.8			
fir_restruct	3.5			
mac1	3.5			
aes192	4.0			
fir3	3.9	3.4		
diffeq	4.0	4.1		
diffeq2	3.9	4.0		
molecular	4.4	4.5		
rs_decoder1	2.2	2.7		
rs_decoder2	2.0	2.2		
atm			2.7	
aes			3.7	
aes_inv			4.0	
ethernet			1.6	
serialproc			1.0	
fir24				2.5
pipe5proc				2.5
raytracer				1.4
Geomean	3.2	3.4	2.3	2.1

Table 4: Critical Path Delay Ratio (FPGA/ASIC) - Slowest Speed Grade

Name	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	6.6			
rs_encoder	4.7			
cordic18	4.9			
cordic8	2.5			
des_area	2.6			
des_perf	3.8			
fir_restruct	5.0			
mac1	4.6			
aes192	5.4			
fir3	5.4	4.6		
diffeq	5.4	5.5		
diffeq2	5.2	5.4		
molecular	6.0	6.1		
rs_decoder1	3.0	3.6		
rs_decoder2	2.7	3.0		
atm			3.6	
aes			4.9	
aes_inv			5.5	
ethernet			2.2	
serialproc			1.4	
fir24				3.3
pipe5proc				3.5
raytracer				2.0
Geomean	4.3	4.5	3.1	2.8

molecular benchmark, the delays with and without the DSP blocks were similar because there are more multipliers in the benchmark than there are DSP blocks. As a result, even when DSP blocks are used the critical path on the FPGA is through a multiplier implemented using regular logic blocks. For the rs_decoder1 and rs_decoder2 benchmarks, only small 5x5 bit and 8x8 bit multiplications are performed and the DSP blocks which are based on 9x9 bit multipliers do not significantly speed up such small multiplications. In such cases where the speed improvement is minor, the extra routing that can be necessary to accommodate the fixed positions of the hard multiplier blocks can eliminate the speed advantage of the hard multipliers. Finally, the diffeq and diffeq2 benchmarks perform slower when the DSP blocks are used because the 32x32 bit multiplications performed in the benchmarks are not able to fully take advantage of the hard multipliers which were designed for 36x36 bit multiplication. As well, those two benchmarks contain two unpipelined stages of multiplication and it appears that implementation in the regular logic clusters is efficient in such a case. We believe that with a larger set of benchmark circuits we would have encountered more benchmarks that could benefit from the use of the hard multipliers, particularly if any designs were more tailored to the DSP block's functionality. However, as these results demonstrated, the major benefit of these hard DSP blocks is not the performance improvement, if any, but rather the significant improvement in area efficiency.

For the circuits that make use of the block memory the FPGA-based designs are on average 2.3 times slower and for the few circuits using both memory and multipliers the FPGA is on average 2.1 times slower. The use of memory blocks does appear to offer a performance advantage; however, this effect is exaggerated because of the slow low power memories used for the standard cell design as described in Section 5.1. We believe that, if higher speed memories were used instead for the ASIC, the performance advantage of the block memories would be relatively minor since, based on gate delays, the speed can be improved by over 20% [25, 6]. Therefore, our conclusion for the memory blocks is the same as for the DSP blocks, which is that the primary benefit from such blocks is improved area efficiency.

As described earlier, the FPGA delay measurements assume the fastest speed grade part is used. Comparing to the fastest speed grade is useful for understanding the best case disparity between FPGAs and ASICs but it is not entirely fair. ASICs are generally designed for the worst case process and it may be fairer to compare the ASIC performance to that of the slowest FPGA speed grade. Table 4 presents this comparison. For logic only circuits, the ASIC performance is now 4.3 times greater than the FPGA. When the circuits make use of the DSP blocks the gap is 4.5 times and when memory blocks are used the performance difference is 3.1 times. For the circuits that use both the memory and the multipliers, the average is 2.8 times. As expected, the slower speed grade parts cause a larger performance gap between ASICs and FPGAs.

7.3 Power Consumption

In Table 5, we list the ratio of FPGA dynamic power consumption to ASIC power consumption for the benchmark

circuits. Again, we categorize the results based on which hard FPGA blocks were used. As described earlier, two approaches are used for power consumption measurements and the table indicates which method was used. "Sim" means that the simulation-based method (with full simulation vectors) was used and "Const" indicates that a constant toggle rate and static probability was applied to all nets in the design. Static power results are not presented for reasons that will be described later.

The results indicate that on average FPGAs consume 12 times more dynamic power than ASICs when the circuits contain only logic. If we consider the subset of designs for which the simulation-based power measurements were used we observe that the results are on par with the results from the constant toggle rate method. We are more confident in the results when this technique is used. However, the results using the constant toggle rate approach are relatively similar and the simulation-based outcome is within the range of the results seen with the constant toggle rate method.

When we consider designs that include hard blocks such as DSP blocks and memory blocks, we observe that the gap is 12, 9.2 and 9.0 times for the cases when multipliers, memories and both memories and multipliers are used, respectively. The area savings that these hard blocks enabled suggested that some power savings should occur because a smaller area difference implies fewer excess transistors which in turn means that the capacitive load on the signals in the design will be less. With a lower load, dynamic power consumption is reduced and we observe this in general. In particular, we note that the circuits that use DSP blocks consume equal or less power when the area efficient DSP blocks are used as compared to when those same circuits are implemented without the DSP blocks. The one exception is again rs_decoder1 which suffered from an inefficient use of the DSP blocks.

In addition to the dynamic power, we measured the static power consumption of the designs for both the FPGA and the ASIC implementations; however, as will be described, we were unable to draw any useful conclusions. We performed these measurements for both typical silicon at 25 °C and worst-case silicon at 85 °C. To account for the fact that the provided worst case standard cell libraries were characterized for a higher temperature, the standard cell results were scaled by a factor determined from HSPICE simulations of a small sample of cells. We did not need to scale the results for typical silicon. The results we observed for these two cases deviated significantly. For logic only designs, on average the FPGA-based implementations consumed 87 times more static power than the equivalent ASIC when measured for typical conditions and typical silicon but this difference was only 5.4 times under worst case conditions for worst case silicon.

The usefulness of either of these results is unclear. Designers are generally most concerned about worst-case conditions which makes the typical-case measurements uninformative and potentially subject to error since more time is spent ensuring the accuracy of the worst-case models. The worst-case results measured in this work suffer from error introduced by our temperature scaling. As well, static power, which is predominantly due to sub-threshold leakage for current technologies[13], is very process dependent and this makes it difficult to ensure a fair comparison given the available information. In particular, we do not know the confidence level of either worst-case leakage estimate. These estimates are influenced by a variety of factors including the maturity of a process and, therefore, a comparison of leakage estimates from two different foundries, as we attempt to do here, may reflect the underlying differences between the foundries and not the differences between FPGAs and ASICs that we seek to measure. Another issue that makes comparison difficult is that, if static power is a concern for either FPGAs or ASICs, manufacturers may opt to test the power consumption and eliminate any parts which exceed a fixed limit. Both business and technical factors could impact those fixed limits. Given all these factors, to perform a comparison in which we could be confident, we would need to perform HSPICE simulations using identical process models. We did not have these same concerns about dynamic power because process and temperature variations have significantly less impact on dynamic power.

Despite our inability to reliably measure the absolute static power consumption gap, we did find that, as expected, the static power gap and the area gap are somewhat correlated. (The correlation coefficient of the area gap to the static power gap was 0.80 and 0.81 for the typical and worst case measurements respectively.) This was expected because transistor width is generally proportional to the static power consumption [14] and the area gap partially reflects the difference in total transistor width between an FPGA and an ASIC. This relationship is important because it demonstrates that hard blocks such as multipliers and block memories, which reduced the area gap, reduce the static power consumption gap as well.

Table 5: Dynamic Power Consumption Ratio (FPGA/ASIC)

Name	Method	Logic Only	Logic & DSP	Logic & Memory	Logic, Memory & DSP
booth	Sim	16			
rs_encoder	Sim	7.2			
cordic18	Const	6.3			
cordic8	Const	6.0			
des_area	Const	26			
des_perf	Const	9.3			
fir_restruct	Const	9.0			
mac1	Const	18			
aes192	Sim	12			
fir3	Const	12	7.4		
diffeq	Const	15	12		
diffeq2	Const	16	12		
molecular	Const	15	15		
rs_decoder1	Const	13	16		
rs_decoder2	Const	11	11		
atm	Const			11	
aes	Sim			4.0	
aes_inv	Sim			3.9	
ethernet	Const			15	
serialproc	Const			24	
fir24	Const				5.2
pipe5proc	Const				12
raytracer	Const				12
Geomean		12	12	9.2	9.0

8. CONCLUSION

This paper has presented empirical measurements quantifying the gap between FPGAs and ASICs. We observed that for circuits implemented entirely using LUTs and flip-flops (logic-only), an FPGA is on average 40 times larger and 3.2 times slower than a standard cell implementation. An FPGA also consumes 12 times more dynamic power than an equivalent ASIC on average. We confirmed that the use of hard multipliers and dedicated memories enable a substantial reduction in area and power consumption but these blocks have a relatively minor impact on the delay differences between ASICs and FPGAs.

9. ACKNOWLEDGEMENTS

We are indebted to Jaro Pristupa for the extensive support he provided for both the technology kits and the numerous CAD tools required for this work. This comparison would not have been possible without the area measurements of the Stratix II provided by Richard Cliff from Altera and the technology files and memory cores provided by CMC Microsystems and Circuits Multi-Projets. Paul Chow, Peter Jamieson, Alex Rodionov, and Peter Yiannacouras provided some of the benchmarks we used in this work. Ian Kuon received financial support from NSERC and this research project was also supported by a NSERC Discovery Grant.

10. REFERENCES

[1] Altera Corporation. Partnership with TSMC yields first silicon success on Altera's 90-nm, low-k products, June 2004. `http://www.altera.com/corporate/news_room/releases/releases_archive/2004/products/nr-tsmc_partnership.html`.

[2] Altera Corporation. *Quartus II Development Software Handbook*, 5.0 edition, May 2005.

[3] Altera Corporation. *Stratix II Device Handbook*, 3.0 edition, May 2005.

[4] S. D. Brown, R. Francis, J. Rose, and Z. Vranesic. *Field-programmable gate arrays*. Kluwer Academic Publishers, 1992.

[5] Cadence. *Encounter Design Flow Guide and Tutorial, Product Version 3.3.1*, February 2004.

[6] Cadence Design Systems. TSMC Standard Cell Libraries, 2003. Available online at `http://www.cadence.com/partners/tsmc/SC_Brochure_9.pdf`.

[7] A. Chang and W. J. Dally. Explaining the gap between ASIC and custom power: a custom perspective. In *DAC '05*, pages 281–284, New York, NY, USA, 2005. ACM Press.

[8] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC & Custom Tools and Techniques for High-Performance ASIC Design*. Kluwer Academic Publishers, 2002.

[9] D. G. Chinnery and K. Keutzer. Closing the power gap between ASIC and custom: an ASIC perspective. In *DAC '05*, pages 275–280, New York, NY, USA, 2005. ACM Press.

[10] R. Cliff. Altera Corporation. Private Communication.

[11] K. Compton and S. Hauck. Automatic design of area-efficient configurable ASIC cores. *IEEE Transactions on Computers*, submitted.

[12] W. J. Dally and A. Chang. The role of custom design in ASIC chips. In *DAC '00*, pages 643–647, 2000.

[13] V. De and S. Borkar. Technology and design challenges for low power and high performance. In *ISLPED '99*, pages 163–168, New York, NY, USA, 1999. ACM Press.

[14] W. Jiang, V. Tiwari, E. de la Iglesia, and A. Sinha. Topological analysis for leakage prediction of digital circuits. In *ASP-DAC '02*, page 39, Washington, DC, USA, 2002. IEEE Computer Society.

[15] H. S. Jones Jr., P. R. Nagle, and H. T. Nguyen. A comparison of standard cell and gate array implementations in a common CAD system. In *IEEE 1986 CICC*, pages 228–232, 1986.

[16] I. Kuon, A. Egier, and J. Rose. Design, layout and verification of an FPGA using automated tools. In *FPGA '05*, pages 215–226, New York, NY, USA, 2005. ACM Press.

[17] Lattice Semiconductor Corporation. *LatticeECP/EC Family Data Sheet*, May 2005. Version 01.6.

[18] LSI Logic. RapidChip Platform ASIC, 2005. `http://www.lsilogic.com/products/rapidchip_platform_asic/index.html`.

[19] NEC Electronics. ISSP (Structured ASIC), 2005. `http://www.necel.com/issp/english/`.

[20] K. Padalia, R. Fung, M. Bourgeault, A. Egier, and J. Rose. Automatic transistor and physical design of FPGA tiles from an architectural specification. In *FPGA '03*, pages 164–172, New York, NY, USA, 2003. ACM Press.

[21] M. J. S. Smith. *Application-Specific Integrated Circuits*. Addison-Wesley, 1997.

[22] STMicroelectronics. 90nm CMOS090 Design Platform, 2005. `http://www.st.com/stonline/prodpres/dedicate/soc/asic/90plat.htm`.

[23] Synopsys. *Design Compiler Reference Manual: Constraints and Timing*, version v-2004.06 edition, June 2004.

[24] Synopsys. *Design Compiler User Guide*, version v-2004.06 edition, June 2004.

[25] Toshiba Corporation. 90nm (Ldrawn=70nm) CMOS ASIC TC300 Family, BCE0012A, 2003. Available online at `http://www.semicon.toshiba.co.jp/eng/prd/asic/doc/pdf/bce0012a.pdf`.

[26] N. H. E. Weste and D. Harris. *CMOS VLSI Design A Circuits and Systems Perspective*. Pearson Addison-Wesley, 2005.

[27] S. J. Wilton, N. Kafafi, J. C. H. Wu, K. A. Bozman, V. Aken'Ova, and R. Saleh. Design considerations for soft embedded programmable logic cores. *IEEE JSSC*, 40(2):485–497, February 2005.

[28] Xilinx. *Virtex-4 Family Overview*, 1.4 edition, June 2005.

[29] X. Yang, B.-K. Choi, and M. Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. *IEEE Trans. Computer-Aided Design*, 22(4):410–419, April 2003.

[30] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel. A hybrid ASIC and FPGA architecture. In *ICCAD '02*, pages 187–194, November 2002.