A Study on Soft Core Processor Configurations for Embedded Cryptography Applications

Benjamin Kueffler[1], Mohamed El-Hadedy[2]

[1] Department of Electrical and Computer Engineering, California Polytechnic State University, Pomona - bkkueffler@cpp.edu
[2] Department of Electrical and Computer Engineering, California Polytechnic State University, Pomona - mealy@cpp.edu

Abstract

FPGAs have provided a platform for embedded designs to achieve impressive performance with minimal resources by targeting specific applications. This platform is especially desirable for embedded cryptography applications, which are increasingly being needed in resource constrained embedded devices. In this survey, tests are performed on the Xilinx MicroBlaze processor within a 7-Series FPGA. Each of these tests enables hardware features on the processor commonly used in embedded applications. Minimum area, high performance, maximum frequency, and frequency optimized implementations are produced in order to survey trade offs that exist within these various architectures. The application chosen to survey these trade offs is the RSA cryptography algorithm. These tests demonstrate the trade off of power and area for performance, with an emphasis on trade offs for embedded cryptography applications. The result of these tests indicate that targeted hardware has the ability to achieve multiplied performance compared to hardware not optimized around the RSA algorithm.

Index Terms

FPGA, hardware/software interface, MicroBlaze, soft-core processor, cryptography

I. Introduction

SINCE their introduction in 1985, Field Programmable Gate Array (FPGA) designs have gained popularity as a tool to create digital circuits without the cost and complexities associated with application specific integrated circuits (ASIC). Their reprogrammability have provided designers with a way to achieve hardware acceleration while still being able to rapidly change the design. Overhead costs associated with programmable logic kept flexible solutions such as processor cores out of FPGA designs. However, since their inception, FPGAs have seen capacities increase ten thousand times, and have seen performance increase one-hundred fold [1]. This has given architects the ability to implement processors within the FPGA fabric, to provide additional flexibility and computational variety to their designs.

The importance of software alongside hardware in systems is substantial, as the ability for software configuration can optimize the performance driven hardware on the chip in real time. In addition to flexibility, software partitioning can be used to drive power costs of chips down on the order of 90%, by only enabling the highest performing clusters of hardware at a time [2]. Beyond this, the hardware/software interface is useful in providing additional debugging and real time updates by reading the status of particular hardware sectors.

At the intersection of FPGAs and software is the soft processor. Soft processors, such as the Xilinx MicroBlaze, allow for the implementation of software algorithms on reconfigurable logic. This allows for optimization after testing to occur in hardware alongside the traditional optimization of software, due to the reprogrammable nature of the soft processors on FPGAs.

This paper explores the potential of the MicroBlaze embedded processor targeting performance and resource trade offs while running the RSA cryptography algorithm. Related works, technologies, and the background on the subject is found in section II. Section III presents the purpose in greater detail. Section IV explains the hardware architecture of the system. Section V discusses the specific implementations that were used to compare trade offs while running the algorithm. Section VI provides the RSA algorithm implementation as well as test procedure. The results, including maximum frequency, execution time, resource, and power utilization are all included in Section VII. Finally, section VIII concludes the study and offers future areas of study.

II. Background and Related Work

One of the main competitors of soft processors are their hard counterparts. System designers have often chosen an off chip processing solution to incorporate alongside their separate FPGA device. However, in recent years, system on a chip (SoC) devices have grown in popularity, allowing for both programmable logic and hard processor to be present within the same fabric. The Xilinx Zynq family is an example of this, with a single or dual core ARM processor at the disposal of the software designer to communicate directly with the FPGA over the Advanced eXtensible Interface (AXI) bus [3]. These hard processors give a challenge for their soft counterparts by offering faster speeds and higher performance per Watt [4]. However, soft processors offer up the potential for reduced cost and greater flexibility, with the ability to change processor

architectures or features via reprogramming.

The MicroBlaze is a 32/64 bit customizable soft processor that can be used alongside any Xilinx FPGA based system. This soft processor contains the configurability needed to reduce cost to make it preferable to hard processors in many applications. The MicroBlaze is able to be customized in 32 or 64 bit modes, allowing for both large and small data widths to be implemented. In recent years, a floating point unit was added as a customizable option, allowing for more taxing floating point programs to be run on the core. More granularity exists within the microblaze system by allowing optional barrel shifters, multipliers, and divider units to be generated through the GUI. This can provide the option for small embedded applications to forgo these features in order to achieve a very small processor. Such minimal processors can achieve utilization as low as 600 LUTs and 300 registers in Spartan 7 series FPGA devices [5]. More intensive applications may choose to utilize these features, achieving higher performance.

Further customizability exists with the ability to change pipeline architecture. For minimal area, three pipeline stages may be configured. For optimal performance, five pipeline stages may be used. For optimum frequency, 8 pipeline stages can be implemented. This flexibility allows for rapid prototyping, allowing for testing among all of the different configurations to see which one suits the application the strongest. Such reprogramming has even been used during runtime, as SRAM based FPGAs allow for dynamic reconfiguration, potentially allowing for minimal core power during idling periods, while reconfiguring for maximum performance during periods of stress [6]. Furthermore, soft processors allow for a customizable amount of cores to suit the application [4]. Modern ASIC processors contain many cores, but many often go unused. Modern SoC FPGA/Processor solutions contain a couple cores, but often not enough. The soft processor can provide the flexible solution to instantiate just the right amount of cores given the multi-threaded nature of the application. This can have a significant impact on the area and power cost of the system.

Due to the ability for soft core processors to be adapted exactly for the application purpose, they have the potential to exceed the capabilities of their hard processor SoC or ASIC counterparts. As a final advantage, due to their ability to use small amounts of area on the chip, their cost per core can be reduced significantly lower than a hard processor [7]. This can allow for advantages when scaling the production of these devices.

III. Purpose

In order to analyze the effect and realize the advantage of soft processor implementations, six different MicroBlaze configurations will be produced. These implementations will be tested alongside a software algorithm designed to take advantage of certain hardware acceleration that is found in some of the configurations. In order to give an accurate look at the advantages and disadvantages of the configurations, power, area, performance, and performance per Watt will be the items of comparison. The algorithm that is chosen will be RSA encryption and decryption. This algorithm is computationally heavy, and is utilized in many embedded systems, so it acts as a good comparison to be generalized to other applications utilizing the MicroBlaze system.

The system architecture will also attempt to be similar to common embedded configurations. It will use the AXI bus, a common high performance bus utilized in many systems. Additional slaves will be added to the bus to control peripherals, keep time, and allow for serial communication. This common interface allows for the tests to reflect many similar AXI based systems.

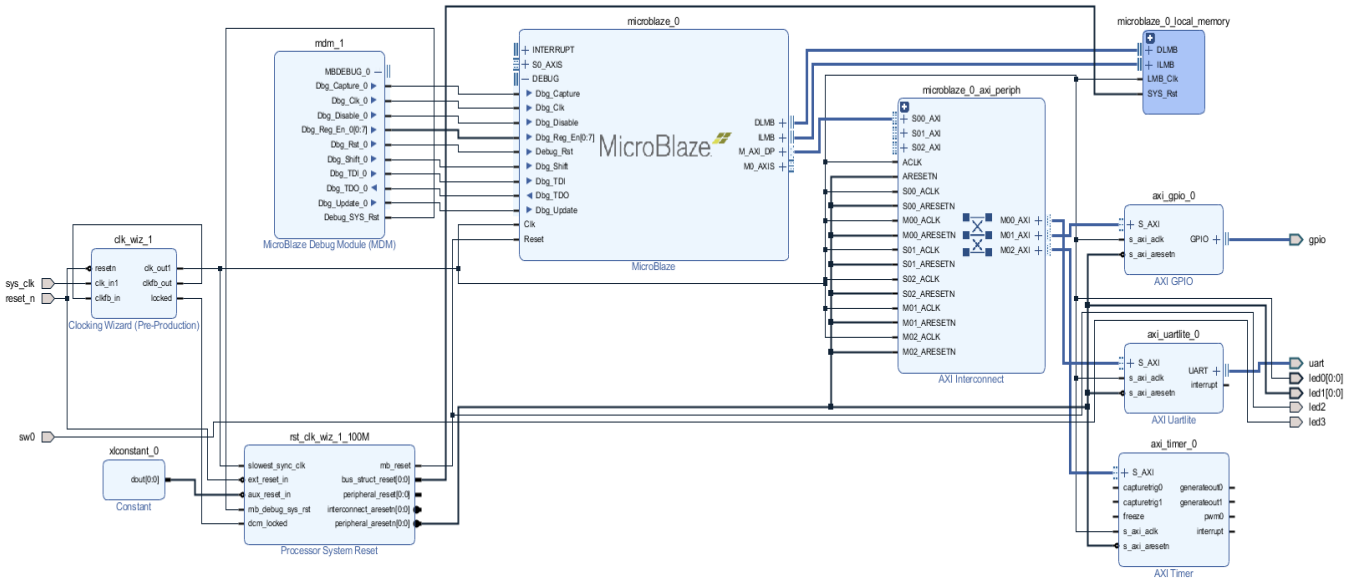

Fig. 1. System Block Diagram

IV. System Architecture

The system architecture consists of a typical FPGA design utilizing a soft processor such as the MicroBlaze. The MicroBlaze system, along with many other designs, utilizes AMBA AXI as the bus. The processor acts a master in order to control peripherals and access memory. The data and instruction memory lies on block RAM acting as a slave on the AXI bus. Additional slaves utilized for the purpose of testing include the AXI Uartlite and AXI timer submodules. The uartlite acts as an interface between AXI and UART and allows tests to occur by loading in files to encrypt, and reading out the result for verification. The AXI timer is enabled by the processor at the start of the algorithm under test, and is critical for giving cycle-accurate readings of the execution time. A clock generator instantiates a phase locked loop (PLL) in order to produce the system's clock, which is set to the maximum achievable frequency in the particular implementation.

V. Implementations

Six distinct implementations of the processor were chosen in order to achieve a variety of tradeoffs for analysis. Note that in each of these implementations, regardless of target, the debug mode feature was included in the MicroBlaze, this increases area by a constant amount among all the tests (about 500 LUTs/Registers).

A. Three Stage Pipeline

This build targets the minimum area achievable. It utilizes a three stage pipeline. This type of processor build is frequently used among FPGA designers working in the embedded field, because it allows easy entry into building a software system with minimum sacrifice of chip space. This build sacrifices performance by giving up additional hardware features, but will remain cheap in terms of power and area.

B. Three Stage Pipeline with Multiplier and Divider

This build augments the normal three stage pipeline by adding a multiplier and divider within the execution stage. This uses up additional space on the chip, but also decreases the maximum achievable frequency to 115 MHz.

C. Five Stage Pipeline

The five stage pipeline represents the most used MicroBlaze pipeline. It is the pipelined used for the maximum performance of the MicroBlaze. In this implementation, the five stage pipeline with no multiplier, divider, or branch target buffer (BTB) is used. This barebones pipeline provides a baseline for the performance of the five stage pipeline.

D. Five Stage Pipeline with Multiplier

This implementation augments the five stage pipeline to include a multiplier unit in the execution stage. This reduces the execution time of the program significantly. Due to the DSP slice resource on Xilinx FPGAs, the multiplier does not add higher resource utilization of the LUTS or flip flops.

E. Five Stage Pipeline with Multiplier and Divider

This build adds a multiplier and a divider unit in the execution stage of the five stage pipeline. This greatly reduces the execution time of the program. Resource utilization goes up a moderate amount from the addition of the divider.

F. Five Stage Pipeline with Multiplier, Divider, and Branch Target Buffer

Uses the same five stage pipeline with the multiplier and divider, but adds an additional branch target buffer. The BTB enhances the branch prediction capabilities of the microblaze by adding an additional cache to store the results of previous branches. In a vacuum, this would increase performance for most designs, but the MicroBlaze implementation of the BTB does reduce the maximum frequency by about 20%. This means that to see a performance increase, the addition of the BTB must make up for the lost maximum frequency.

VI. Software and Test

In order to give an accurate depiction of the tradeoffs of the MicroBlaze system, a software application was produced that could take advantage of additional hardware acceleration. Encrypting and decrypting messages takes significant computational resources, and can be scaled with better hardware. The RSA algorithm was used, which uses prime numbers as a basis for encoding and decoding. One public key is generated for encryption, while one private key is used for decrypting the encrypted message.

Both encryption and decryption require modulo and multiplication to encode and decode. These operations can be accelerated with multiply and divide operations, which require additional hardware to be present in processors. Without this hardware, the performance of the encoding and decoding will be reduced. This tradeoff of area and power in return for performance was the basis for the tests done using this algorithm.

The test consisted of encrypting and decrypting two scripts. One of the scripts consisted of 32kB of latin text, and the other script consisted of 32kB of numbers. Different scripts were used in order to verify that the algorithm was working as intended with all characters and to record any potential variations in performance with a variety of characters. The test data was large in order to give a more accurate result. All the data from the scripts was loaded in over UART into the internal block RAM of the FPGA before the algorithm starting encrypting. This was done in order to remove any performance impact from external memory or serial communication links. The AXI Timer submodule kept a cycle-precise timer that was initialized directly before starting the encryption. After the decryption finished, the timer was stopped. This ensured that no serial communication time for reading out was included in the final time count.

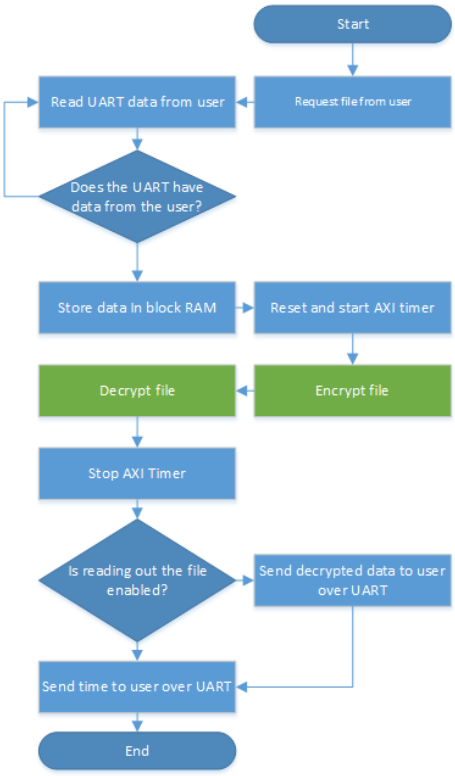

Fig. 2. Software flow diagram. Green denotes software algorithm that is timed during the test

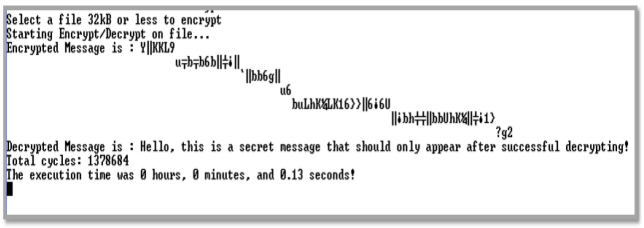

Fig. 3. Example of a small file containing one sentence being encrypted and then subsequently decrypted

VII. Results

All tests and implementations were performed the Xilinx 7 series chip 7s50csga324. The following details the results among different categories.

A. Maximum Frequency

For each implementation, the maximum frequency was found and used for the corresponding test. The five stage pipeline, with its increased pipeline stages, is able to hold 125 MHz even with the addition of multipliers and dividers. However, due to the longer signal paths within the three stage pipeline, the maximum frequency drops when adding multipliers and dividers. In addition to this, the branch target buffer also adds significantly to the critical path of the system. The result is that when implementing the BTB, the frequency must also be reduced by about 20%. Therefore, systems implementing the BTB must be able to take advantage of improved branch prediction in order to make up for the loss in frequency.

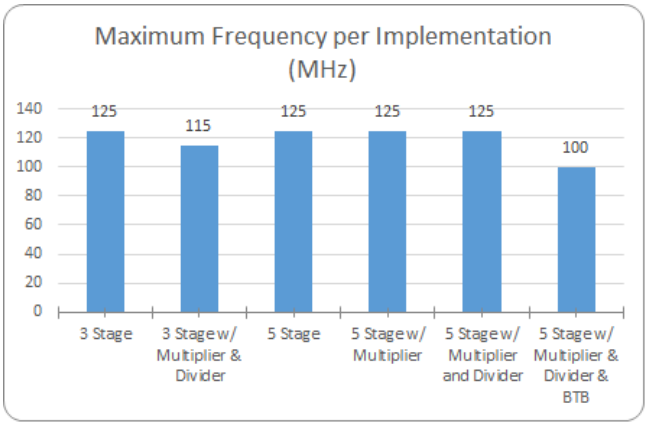

Fig. 4. Maximum Frequency per Implementation(MHz)

B. Execution Time

The execution time represents the total time of both encrypting and decrypting a file. The overall performance of the implementations is given by this execution time. The baseline runs, 3 5 stage pipelines without additional hardware had the longest execution times. This is a result of the extra cycles incurred from producing multiply and divide operations without the extra hardware. As a consequence of not having these resources, more instructions and cycles were needed to accomplish those operations.

Among all tests, the multiplier unit improved performance by about 20%. This is due to the frequency of non-power of two multiplies present in the encryption and decryption algorithm. Even greater performance benefit was seen from the divider unit, which improved performance by 60%. The modulus operator is repeated many times per each character within the cryptographic algorithm, which incurs many instructions without a true divider. The addition of the divider and multiplier drove both the 3 stage and five stage tests down almost 80% from their baseline.

The branch target buffer is also notable, since execution time seemingly goes up slightly with the addition of the BTB. In reality, the BTB itself likely helps performance slightly, but the BTB added an additional critical path which reduced the frequency by 20%. This impact to the clock speed directly affected the design's performance by the same amount, which is why adding the BTB did not see a system level performance increase.

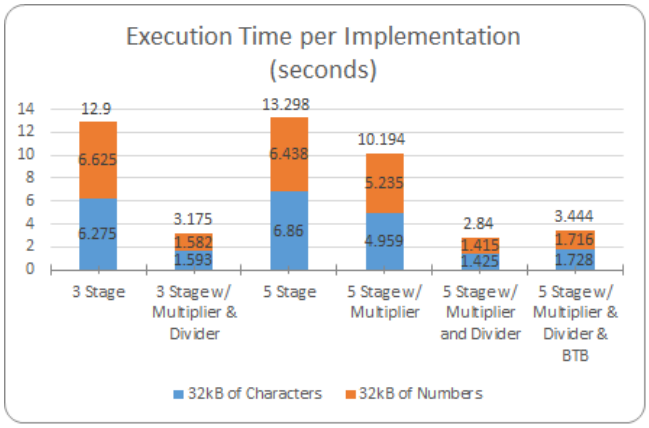

Fig. 5. Execution Time per Implementation (seconds)

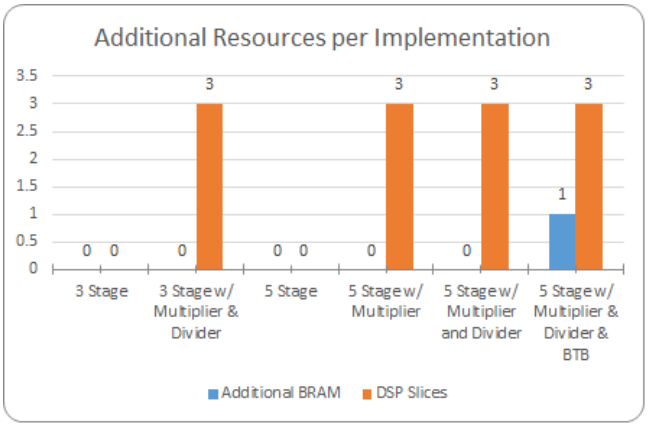

Fig. 7. Additional Resources per Implementation

C. Resource Utilization

Resource utilization sees a significant increase when going from the three stage pipeline to the five stage pipeline. Utilization increases about 33%, which correlates to the addition of 2 extra stages with extra logic and registers added with each stage. The benefit of Xilinx DSP slices can be seen from these results. Adding a multiplier in a pipeline sees virtually no increase in LUT or FF utilization, since the DSP slice performs the entire multiply. The additional benefit of these slices is that they have much smaller critical paths than if they were built out of traditional LUT or FF hardware. The dividers increase utilization by about 6%, but see a very large performance benefit compared to implementations without the divider. Finally, the addition of the BTB does not add significant logic, but does incur one block RAM in order to store the branch target.

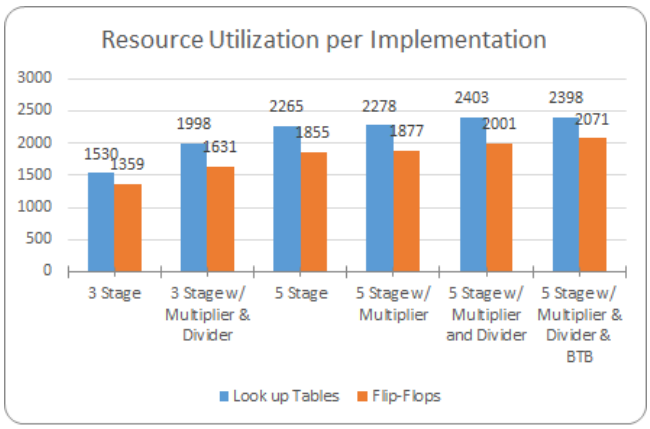

Fig. 6. Resource Utilization per Implementation

D. Power Utilization

Power was compared from two different sources. The Xilinx provided power estimator built into the Vivado framework was used to get a estimation of power for each of the different tests.

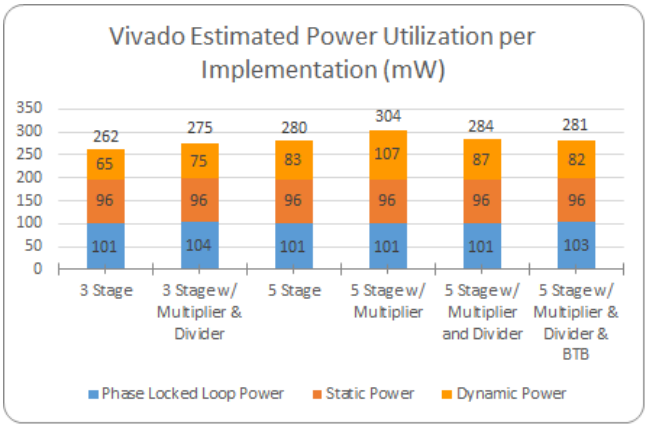

Fig. 8. Vivado estimated utilization per implementation (mW)

In order to capture a more accurate representation, measurements were done during processing which allowed for a read out of the board power. In order to isolate the FPGA device power from these board measurements, a baseline power reading was done with no logic utilization. This established the baseline power draw of the system, allowing for an effective comparison between the estimated power, which only accounts for on-chip power.

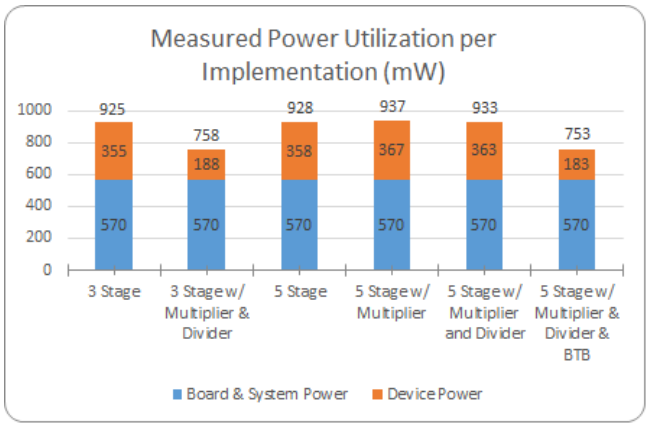

Fig. 9. Measured power utilization per implementation during runtime (mW)

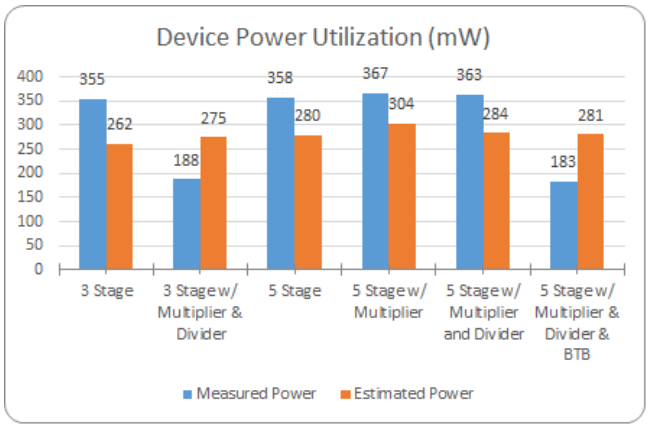

Fig. 10. Measured and estimated power utilization per implementation (mW)

In addition to establishing the power utilization of various implementations, the error between the measured and estimated power can be determined. In particular, estimated power does not appear to accurately gauge the power for differing clock frequencies. This leaves the error rates for the implementations at less than 125 MHz higher than the other implementations. Although device statistics, environment, and toggle estimations were input into the estimation calculations, the estimation error could be improved further by simulating the algorithm and producing a switching activity interchange format (SAIF) file.

As the ability to increase speed and logic increases, power becomes more of a concern for digital designs. In this example with one MicroBlaze on a relatively low end FPGA, these power numbers appear small. However, many systems will realize multi-core architectures or chip architectures with the potential to consume more power. The results shown from this study are important due to this scalability. Although there is significant overhead shown with static power and PLL power, this is with only 5% of the FPGA being used. With 100% utilization of the FPGA, the dynamic power will play a large role in determining the overall power. The less logic and slower the clock speed, the less power will be used. The 5 stage pipeline with BTB now demonstrates some advantage for certain systems. Due to its slower frequency, it was lower in performance even with the additional hardware. However, this also makes it consume less power, giving it a commendable performance per watt.

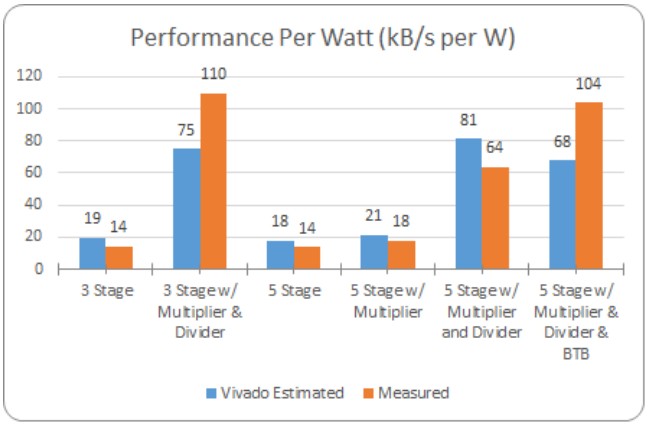

Fig. 11. Performance per Watt (kB/s per W)

VIII. Conclusion

FPGA devices benefit greatly from the use of soft processors. These processors add greater system flexibility over a system without software components. Applications such as cryptography are often found on FPGAs in order to take advantage of the hardware acceleration alongside the traditional processor core. The critical component of these cores is what sacrifices will need to be made to achieve this flexibility. This concern is even greater in the embedded field, where power and speed are both costly.

Clock frequency plays a large role in increasing performance and power utilization. Adding more hardware specific to an application's purpose, such as multipliers for cryptography, can strongly increase performance. The RSA algorithm tested sees a significant increase in performance when the soft processor includes multipliers and dividers in the execution unit. However, a system's design cannot be guaranteed to see greater performance strictly by increasing the hardware; only hardware that directly targets the applications and instructions being processed in the core will see benefit. Adding the branch target buffer in the MicroBlaze core did not add enough benefit to the RSA application to compensate for its clock frequency drawback. Additional power information from these tests indicate that power estimation is not enough to gauge the effectiveness of the performance per watt of the design. For future automated hardware optimizations on reconfigurable devices, measured powered is desired to give accurate feedback on the power efficiency of the system.

The ability to tune these algorithms with reconfigurable hardware is key in optimizing hardware components for the targeted application. Improvements in optimization are a subject of future work, with automated optimizations possible through utilizing partial reconfiguration alongside the chosen

algorithm. Developments in processor architecture are also a subject of interest, architectures such as RISC V allow for greater flexibility with extendable instruction sets desired by FPGA applications. With a greater amount of logic fitting on a single reconfigurable device, soft processor cores will continue to be a benefit to system designers who seek additional configurability and flexibility with their algorithms and designs.

REFERENCES

[1] S. M. Trimberger. "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, Apr. 2015.

[2] Henkel, J, "A low power hardware/software partitioning approach for core-based embedded systems," Design Automation Conference (DAC), 1999.

[3] Xilinx, "Zynq-7000 SoC First Generation Architecture," ds190 datasheet, July 2018.

[4] R. Lysecky and F. Vahid, "A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning," Design, Automation and Test in Europe, Mar. 2005.

[5] Xilinx, "MicroBlaze Processor Reference Guide," MicroBlaze Reference, October 2017.

[6] K. Vipin and S. A. Fahmy, "FPGA Dynamic and Partial Reconfiguration," ACM Computing Surveys, vol. 51, no. 4, pp. 1-39, 2018.

[7] J. G. Tong, I. D. L. Anderson, and M. A. S. Khalid, "Soft-Core Processors for Embedded Systems," 2006 International Conference on Microelectronics, Dec. 2006.