On the Power Consumption of security algorithms employed in wireless networks

Dimitrios Meintanis
Technical University of Crete
Dept of Electronic & Computer Engineering
Kounoupidiana, 73100 Chania, Greece
Email: meidanis@mhl.tuc.gr

Ioannis Papaefstathiou
Technical University of Crete
Dept of Electronic & Computer Engineering
Kounoupidiana, 73100 Chania, Greece
Email: ygp@mhl.tuc.gr

***Abstract*—Supporting high levels of security in wireless networks is a challenging issue because of the specific problems this environment poses; the provided security by small mobile systems, such as PDAs and mobile phones, is often restricted by their limited battery power and their limited processing power. Driven by these restrictions, the designer will have to decide whether to implement the wireless network security schemes in software or to add special purpose hardware units to the system, executing those CPU intensive tasks. This paper demonstrates and compares the Hardware and Software implementations of a number of widely used security applications employed in wireless networks. We measured the total energy consumption for each security algorithm when implemented in reconfigurable hardware devices and we compared it with the total energy consumption of the equivalent software applications. We demonstrate, that the hardware implementations on a state-of-the-art FPGA are significantly faster while they consume three orders of magnitude less power when compared with the software implementations executed on a state-of-the-art hard-core CPU which is embedded in the same FPGA device.**

I. Introduction

Wireless communications offer both organizations and users many benefits such as portability and flexibility, increased productivity, and lower installation costs. Wireless technologies cover a broad range of differing capabilities oriented toward different uses and needs. Wireless local area network (WLAN) devices, for instance, allow users to move their laptops from place to place within a certain range without the need for wires and without losing network connectivity. Less wiring means greater flexibility, increased efficiency, and reduced wiring costs.

Securing these networks, especially when low power hand held devices are used, is a challenging procedure. Rapid increases in communication data rates and security levels required, together with slow increases in battery capacities, threaten to widen this battery gap to a point where it will impede the adoption of applications and services that require security.

The security schemes used in a wireless network environment are very CPU intensive tasks (especially for the limited processing power embedded CPUs). Long execution time means high power consumption and so, less battery runtime. Designers may overcome this problem by adding special purpose hardware units to the system, executing those CPU intensive tasks.

Moreover, in the industrial market, designers have a significant incentive to get their products to market quickly so as to maximize revenue and time-in-market. For every week that a product is delayed revenues are lost, the product's market risk is increased and thus the chance of success is heavily reduced.

Using Field Programmable Gate Arrays (FPGAs), designers can develop a custom hardware solution without the nonrecurring engineering charges or the fabrication and assembly time delays, typically associated with Application Specific Integrated Circuits (ASICs). FPGAs offer a low-risk, quick time-to-market solution that industrial designers can easily modify when they need to make changes, fix bugs or create product derivatives at some point in the future.

However, power is one of the main problems for FPGAs. The post-fabrication flexibility provided by these devices is implemented using a large number of pre-fabricated routing tracks and programmable switches. These tracks can be long, and can consume a significant amount of power, every time they switch. In addition, the programmable switches add capacitance to each track; this further increases the power dissipation of FPGAs.

In this paper we demonstrate, for the first time, the power consumption of various state-of-the-art, real-world security modules when they are implemented in the hardware resources of the FPGAs as well as when they are executed in the built-in hard-core CPUs that are embedded in those devices. Those actual measurements have been made on a advanced, widely used FPGA, with two internal CPU cores. It should be noted that those CPUs are actual implemented within the FPGAs in standard CMOS technologies just like if they were stand-alone CPUs (they are not implemented in reconfigurable logic).

Since those real-world measurements cover a set of widely-used security applications, where the accuracy is crucial for the stability and durability of the crypto algorithms, we believe that this paper can act as a reference to anyone wanting to support high levels of security in wireless systems.

The main contributions of this work can be summarised in the following:

- This is the only paper, to the best of our knowledge, comparing the actual power consumption between the

software and hardware implementations of security algorithms that are all executed on the same FPGA fabric, which already uses advanced process technology with reduced power supply.
- The underlying reference case consists of real-world security designs, widely used in wireless networks, while we have actually *measured* the power consumption using a very detailed experimental circuitry .
- Moreover, in this paper, we discuss in detail the variation of software and hardware measured power consumption over specific, largely used, security applications.

In particular, it is, to the best of our knowledge, one of the very few times that the, otherwise considered power-hungry, FPGAs are proved to be very efficient when employed in wireless environments. The rest of this paper is organized as follows. We explore the related work and we describe the power measurement methods and the measuring systems used. Next we present our results and finally we conclude our remarks.

II. Related Work

As far as we know, this is the first attempt to measure and compare the real-world values of power consumption between hardware and software security blocks with exactly the same functionality and running on the exact same reconfigurable system.

Hardware power measurements of large FPGAs have received little attention compared to that of standard cell ASIC, which have been extensively studied in the literature. In particular, for FPGAs, only the power consumption of certain matrix multiplication algorithms have been presented in [1] and of various digital signal processing modules in [2]. Moreover, Lysecky and Vahid [3] have studied the differences between the *performance* achieved when certain tasks are executed in the embedded processing cores of a state-of-the-art-FPGA and when their are executed by dedicated hardware modules.

On the other hand, there is a considerable amount of work done in both industry and academia related to cryptographic algorithms and their performance evaluation. In the literature there are several implementations of the three most widely used cryptographic algorithms (DES, AES and MD5) in either software, or hardware; the hardware approaches are tailored to both ASICs and FPGAs [4],[5], [6], [7], [8].

There is also certain work which has been done on the FPGA power consumption trade-offs. Becker, Huebner and Ullmann [9] discussed the exact power consumption trade-offs between the measured runtime consumption of a mapped application and the measured reconfiguration time consumption of different dynamically reconfigured applications. Nothing had been done though, to the best of our knowledge, on the comparison of software and hardware power consumption.

III. Security algorithms

Our four chosen security algorithms, namely AES [10], DES, 3DES [11] and MD5 hash [12], fully comply with

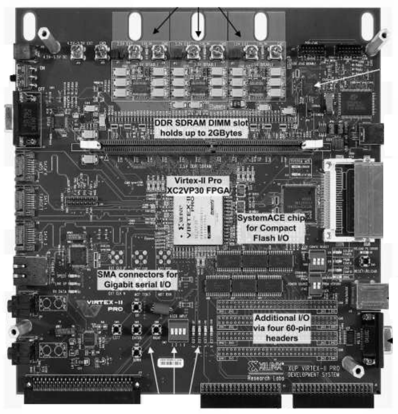
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
VIRTEX-II PRO
XUP VIRTEX-II PRO DEVELOPMENT SYSTEM
Research Labs

Fig. 1. Digilent XUP Board

todays' mobile needs; they are all compact, powerful and widely used in wireless networks.

The hardware security cores used, were not unified. Each module had a different bus architecture and a different input and output logic. Thus we modified their logic by encapsulating a top level Finite State Machine(FSM) to control each module's signals. Our design's top level entity had only two input and four output signals. The inputs are the Clock and the Reset signal and the outputs are four monitoring signals connected to LEDs. Our FSM runs on an endless loop, writing and reading data to and from the cores' internal logic. This endless loop, provide us with stable current consumption throughout the experiment.

In the software part, the code used for each of the security algorithms, was imported and modified by the security core files of the Linux Kernel [13], an operating system very widely used in wireless devices. We have applied all the possible handmade and compilation optimizations and the numbers shown are the best numbers we could achieve.

The equipment used for our experiments is a Digilent XUP Development board [14] (figure 1) with a Xilinx Virtex-II Pro FPGA device.

IV. Software and Hardware Power Measurement

Measuring the FPGA's internal core power consumption, can be a challenging procedure, especially if the board, in which the experiments are carried out on, is not specifically designed for such experiments. In order to measure accurately the power consumption we had to first isolate the board's internal logic voltage, (in most cases 1.2 Volts), and drive it by an external regulated DC power supply.

Then, we insert a shunt resistor of 0.5ohms on the returning path of the internal core voltage source. But since the voltage difference on the resistor's pins is quite small, we have also implemented an amplification circuit as demonstrated in figure 2. This circuit amplifies by a factor of 23 the measured ΔV value. All resistors, had a small tolerance of 1%. A highly advanced mixed signal oscilloscope MSO6052A, of Agilent

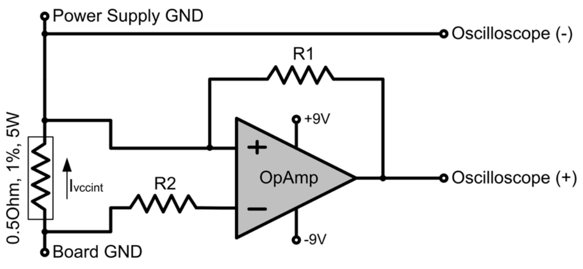

Fig. 2. Operational Amplifier Circuit

Technologies [15], was used for sampling the operational amplifier output.

A. Measuring H/W Power Consumption

Due to the fact that the hardware cores consume the same amount of power throughout the experiment, the measurements were straight forward and extremely accurate.

A manual reset is initiated on the development board and the core enters an endless loop. The current information shown on the osciloscope, remains stable, making it possible for error free current measurement, simply by measuring the voltage level on the display.

This procedure was repeated five times, for all four algorithms, to ensure that the current displayed was the same for all five repetitions. The values demonstrated in the next section are the mean values, while the variance in the power consumption numbers, in the different experiments was insignificant (up to 2%).

Then, the Modelsim SE by Mentor Graphics [16] was used to measure the number of hardware cycles needed for each 32bit data encryption. So, by using the formula $E = Voltage \times Currenttimestime$ we could calculate the true Energy that was used for an 32bit encryption for each algorithm.

B. Measuring Software Power Consumption

Firstly, it should be stressed that the embedded CPU in our experimental set-up is a PowerPC which is well know for its high-performance and relatively low power consumption. In particular is has recently be claimed that ”The PowerPC sets the new standard in terms of performance, low power, security and multi-platform support, making it ideal for next generation 801.11n and WiMAX applications” [17]

Measuring the Software Energy Consumption, was a more complicated procedure since we didn't have the ability to measure the clock cycles needed for each 32bit data encrypted as we can do with the hardware modules.

One initial problem was that no matter what software code the embedded CPU was executing, the power consumption stayed the same. Although we inserted an infinite loop with a noop command in our source code, the power consumption measured associated with the 1.5V supply (i.e. that is the supply connected only to the embedded CPU) was still the same. Thus we could not calculate the actual real processing time of the various software schemes. In order to overcome the above, we have used the power down command of the PowerPC processor; at the end of our program, we altered a special control register, so that the processor would come to its power down state, thus minimizing its power.

Every time we press the reset button, the processor exits the power-down state and starts executing the software loaded in its memory until it enters the power-down state again, at the end of the execution. By measuring this power consumption variation, we were able to calculate the exact power consumed by our device when only the embedded hard-core CPU (i.e. implemented just if it was a stand-alone CPU) was executing the corresponding software as described in the next paragraph.

In particular, we extracted the values from the voltage graph that was captured by our detailed oscilloscope, and inserted them into MatLAB [18]. We divided each measured voltage by 23, which is the amplification factor implied by our amplifier, and then we calculated the integral of the voltage for the time interval between reset and power down, as seen in figure 3.

$$E = \int_{t_{reset}}^{t_{powerdown}} P(t) \cdot dt$$

Fig. 3. Integral Formula

Finally, we divided the total Energy consumed with the total number of 32bit data items that were encrypted. In this manner, we calculated the exact energy that each program consumed for the encryption of a 32bit data stream. It should be noted that in the numbers shown we also include the power consumed in the wake-up process of the CPU. However, we have run each encryption algorithm for 1000 times, without powering off the process, and divided the overall power consumption by 1000 and we realised that the overall wake-up power consumption is less than 8% of the power consumption of each application.

This procedure was repeated three times, for all four algorithms, to ensure that the current displayed was the same for all repetitions. One representative waveform that was acquired by the Oscilloscope is shown in figure 4.

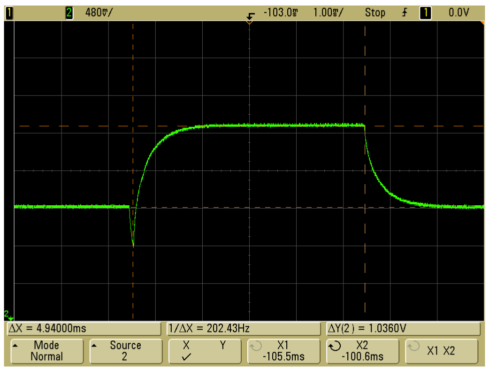

Fig. 4. Oscilloscope Waveform for AES

Data were extracted from the oscilloscope and after importing them to Matlab, we croped everything before the start and after the stop points, so as to get the exact time period that each security application was running. A representative processed waveform is shown in figure 5. The integral of this waveform, represents the actual power that our AES program consumed for encoding 12800 bytes of data.

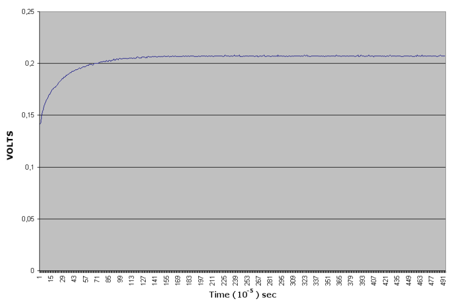

Fig. 5. Edited Waveform for AES

V. Results and Evaluation

After completing both hardware and software power measurements, we extracted and compared the most important results triggered by our experiments.

TABLE II
Time for a 32bit encode in the hardware implementations

	DES	AES	3DES	MD5
Clock Cycles	36	48	38	32

First we had to calculate the total execution time for each hardware algorithm. Although in software it is quite straight forward (we just measure the time from the waveform), in hardware it slightly needed more complicated calculations; since hardware run in an endless loop, we had to calculate the clock cycles needed for each algorithm to complete a 32bit encode.

By using ModelSim, we were able to count exactly the number of cycles needed by each algorithm. In these cycles, we added 24 cycles; this is the number of cycles needed for the embedded CPU to write to and read from its internal bus. This number the ”worst case” cycle time for each read or write transaction. The total cycles needed by each security core are demonstrated in table II.

In all our experiments, the Hardware modules as well as the embedded CPU run at 100MHz. The actual *measured* results are shown in table I As those results clearly demonstrate, although software, in general, is believed in certain cases in wireless systems, to be more power efficient than FPGAs, this is not the case when the very CPU-intensive security schemes are to be implemented. As clearly demonstrated in figure 6, software consumes almost 1000 times more energy than hardware. It should be noted that these values are the real measured data in out state-of-the-art FPGA device. When the PowerPC clock rate was increased the overall energy consumption of the software approach was increased accordingly.

Hardware seems to win over software regarding the performance as well. In figure 7 the difference between in time

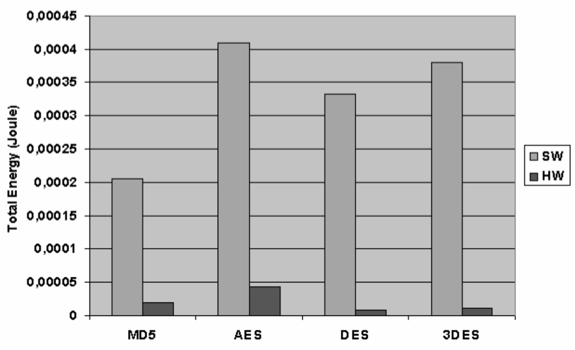

Fig. 6. SW/HW Energy Consume per 32bit Data Enc

needed for a 32bit word encode between the software and the hardware approaches is shown. Software is one thousand times slower than hardware for encoding the same amount of data. Real measured data are displayed in table I.

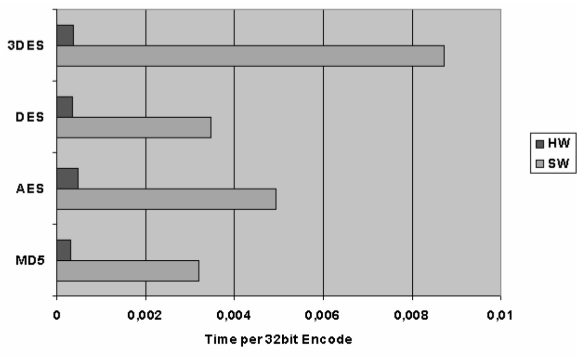

Fig. 7. SW/HW Time consume per 32bit Data Enc

As we may deduce from our measurements, in all those real-world security applications, the hardware cores are almost 1000 times faster and consume three orders of magnitude less energy than the software implementations. Therefore, this work clearly demonstrates that one of the optimal choices for the implementation of the security schemes, in wireless environments, is the state-of-the-art reconfigurable devices which also offer much higher flexibility than ASICs.

Based on the presented results we also claim that the reconfigurable hardware devices can be used in both high-end wireless devices supporting high levels of security such as high-speed wireless network cards and gateways and wireless video processing devices as well as in low-end wireless systems such as battery operated sensor network nodes, wearable devices etc.

VI. Conclusions

Guaranteeing security in wireless networks is challenging because of the specific problems this environment poses; the designer has to decide whether to implement the wireless network security algorithms on software or to add special purpose hardware units to the system, executing those CPU

TABLE I
TRUE MEASURED VALUES

	DES	AES	3DES	MD5
S/W Time (Seconds)	0,0000347	0,0000123	0,0000872	0,000008
H/W Time (Seconds)	0,00000036	0,00000048	0,00000038	0,00000032
S/W Energy (Joules)	0,000958883	0,000208086	0,000435148	0,000160518
H/W Energy (Joules)	$0,918 \times 10^{-07}$	$4,392 \times 10^{-07}$	$0,114 \times 10^{-07}$	$1,824 \times 10^{-07}$

intensive tasks. Our measurements on a state-of-the-art reconfigurable device show that, for the majority of the most widely used security algorithms in wireless networks, the FPGA-based hardware implementations are almost 1000 times faster than the corresponding software application. More importantly, with regards to the power consumption, the real-world measured values demonstrated that the reconfigurable hardware approach consumes three orders of magnitude less energy than the corresponding software implementations.

We strongly believe that these results would lead numerous designers of wireless devices to consider, probably for the first time, to adopt FPGA-based hardware solutions for their implementations of security schemes, instead of the easily implemented but slow and power-hungry software implementations.

ACKNOWLEDGMENTS

This paper is part of the 03ED375 research project, implemented within the framework of the ”Reinforcement Programme of Human Research Manpower” (P.E.N.E.D) and co-financed by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.U.-European Social Fund)

REFERENCES

[1] Scrofano, R. Seonil Choi Prasanna, V.K., “Energy efficiency of FPGAs and programmable processors for matrix multiplication,” *IEEE FPT 2002*, 2002.

[2] P. Waldeck and N. Bergmann, “Evaluating software and hardware implementations of signal-processing tasks in an FPGA,” *IEEE FPT 2004*, 2004.

[3] F. V. Roman Lysecky, “A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning,” *IEEE DATE 2005*, pp. 18–23, 2005.

[4] C. Patterson, “High performance DES encryption in Virtex FPGAs using JBits,” *In Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00)*, 2000.

[5] T. Schaffer, A. Glaser, S. Rao and P. Franzon, “A Flip-Chip Implementation of the Data Encryption Standard (DES),” *IEEE Multi-Chip Module Conference (MCMC'97)*, p. 13, 1997.

[6] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, “Efficient Uses of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis,” *IEEE Transactions on Computers*, p. 52(4):473.482, Apr 2003.

[7] A. Pfitzmann and R. Amann, “More efficient software implementations of (generalized) DES,” *Computers and Security*, p. 12(5):477.500, Aug 1993.

[8] J. Deepakumara, Howard M. Heys and R. Venkatesan, “FPGA Impementation of MD5 Hash Algorithm,” *Electrical and Computer Engineering, 2001*, pp. 919–924, 2001.

[9] J. Becker, M. Huebner, M. Ullmann, “Power Estimation and Power Measurement of Xilinx Virtex FPGAs:Trade-offs and Limitations,” 2003.

[10] Hemanth Satyanarayana. (2004, December) AES128. [Online]. Available: http://www.opencores.org/

[11] ASICS.WS. (2001, September) DES/Triple DES IP Cores. [Online]. Available: http://www.asics.ws/

[12] Universidad Rey Juan Carlos. (2004, August) SystemC/Verilog MD5. [Online]. Available: http://www.escet.urjc.es/ jmartine

[13] The Linux Kernel Archives. [Online]. Available: http://www.kernel.org/

[14] Digilent Inc. XUP, Virtex-II Pro Development System. [Online]. Available: http://www.digilentinc.com/

[15] Agilent Techoligies. [Online]. Available: http://www.home.agilent.com/

[16] Mentor Graphics. Modelsim Special Edition 6.2. [Online]. Available: http://www.mentor.com/

[17] AMCC. (2008, January) AMCCs PowerPC 405EX Embedded Processor Named Product of the Year by Electronic Products Magazine. [Online]. Available: http://www.power.org/

[18] The MathWorks. MATLAB and Simulink for Technical Computing. [Online]. Available: http://www.mathworks.com