A Scalable Platform for Cryptanalysis of Computationally Intensive Algorithms

Mustafa Hakan Solmaz, Hasan Erdoğan, and Rıza Aykaç

Abstract— **A cryptographic co-processor design methodology is introduced in this study. The co-processor is connected to a microprocessor through a bus interface protocol, namely Processor Local Bus (PLB) protocol of IBM. Since the co-processor is designed to be a modular block, it can be instantiated in any number as long as FPGA fabric is empty. The methodology is demonstrated for exhaustive key search of DES algorithm. A Microprocessor system is designed and realized on ML605, a Xilinx Virtex-6 FPGA Evaluation Board, which captures initial performance of 8 billion key searches per second.**

Index Terms— **Cryptanalysis, DES, Brute force attack, Cryptographic peripheral design.**

I. Introduction

Cryptography is the study of secure communication methods in the presence of adversaries. Classical cryptography existed in the ancient ages in the form of substitution or transposition ciphers. In the computer era cryptography turned into a discipline which heavily uses advanced mathematical methods and computer science practices. While cryptography is the study of introducing secure communication methods, there is a parallel discipline, cryptanalysis, which is trying to extract secret information from encrypted messages. Although there seems to be a negative perception about seeking other people's secret information, cryptanalysis is as important as cryptography. In the absence of cryptanalysis we cannot know how secure our cryptosystem is.

Classical cryptanalysis used basic tools for breaking classical cryptosystems. Frequency analysis can be given as an example to these methods. For example "e" is known to be the most frequent letter in English, while "the" is the most frequent word which ends with "e". In a classical cryptosystem which fails to hide these statistics, frequency analysis can be used to break the cryptosystem provided that the encrypted text is long enough. As with cryptography, modern cryptanalysis also changed face with the advent of high speed computers and electronic computation systems.

Among the methods of modern cryptanalysis we will focus on brute force attack and their applications. A brute force attack or exhaustive key search can be described as follows: Suppose that an attacker has one or more plain text - cipher text pairs and the cryptosystem details except the encryption key. Can he/she find the secret key so that he/she can decrypt other cipher texts for which the plain text is not known? The answer to this question is theoretically yes for all symmetric ciphers. A pseudo flowchart can be described as below:

Let the key space be $K = \{k_1, k_2, k_N\}$ plain text p cipher text c and $d_{k_i}(c)$ be the decryption function with key k_i .

For $i = 1,2,.,N$ loop
  Compute $y = d_{ki}(c)$
  If $y = p$ then return k_i .
End loop

For a key length of N it can be shown that the correct key is found in N/2 iterations on the average [11]. It is also possible to run a brute force attack by using encryption function.

Among the famous applications of brute force attacks are the attempts to break DES algorithm. The first special computer was proposed by Diffie and Helman right after the standardization of DES [4]. Later a distributed network of 100.000 PC's was able to reveal key after 22 hours. Electronic Frontiers Foundation built a machine which also succeeded in finding the key. Most recently a special hardware named as Copacobana was designed by SciEngines Corporation. More than one hundred FPGAs are used in Copacobana to execute exhaustive search algorithms like the one defined above [5], [6].

In this paper we will go through the steps of implementing an exhaustive key search of a crypto algorithm on a commercially available off the shelf (COTS) FPGA board. DES algorithm is chosen since it is the most frequently studied algorithm in literature and there are performance measures to compare against. As FPGA platform we chose Virtex-6 ML605 Evaluation Board, a recent board from Xilinx. Although there is a Virtex-6 family of FPGA on this board the design can easily be ported to any other Xilinx FPGA. It can also be ported to other vendors' FPGAs with some more effort.

The outline of the manuscript is as follows: In the next section we give a brief introduction about embedded

Manuscript received March11, 2012.
M. H. Solmaz, H. Erdoğan and R. Aykaç are with Information Security Dept, Aselsan Inc.,P.O.Box 1 ,06172 Yenimahalle, ANKARA / TURKEY (e-mail: hsolmaz,herdogan,raykac@aselsan.com.tr).

processors and custom peripheral concepts in an FPGA. In Section 3 DES algorithm implementation is introduced. Section 4 covers the details of custom peripheral for brute force attack designed for exhaustive key search and its interface to an embedded processor. Hardware and software co-design practices are also presented in this section. Section 5 will give performance results and comparisons to other platforms. Section 6 discusses design practices followed during the study. Section 7 closes with conclusion.

II. EMBEDDED PROCESSORS AND CUSTOM PERIPHERALS

A. *Embedded Processors*

Embedded system design and FPGA design were two different branches in the last decade. With the advances of process technologies FPGAs with thousands of configurable blocks became available and embedded system design with FPGAs has emerged as a new branch. Today FPGA vendors are providing soft or hard core processors in their chips. They also provide tool chains for hardware and software development for hardware and software platforms.

The name of the soft-core processor for Xilinx platforms is Microblaze, where as the hard core which is present in certain families is called PowerPC. Microblaze as an example of soft processor is built by combining blocks of Xilinx slices. The advantage of this approach is that you can generate a microprocessor exactly as you need. You do not have to accommodate unused CPU parts.

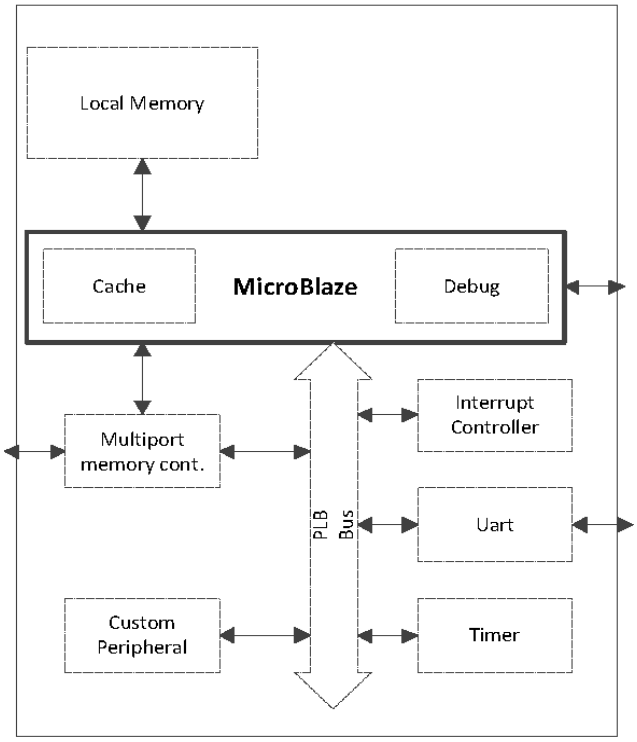

Fig. 1. Microblaze embedded system overview.

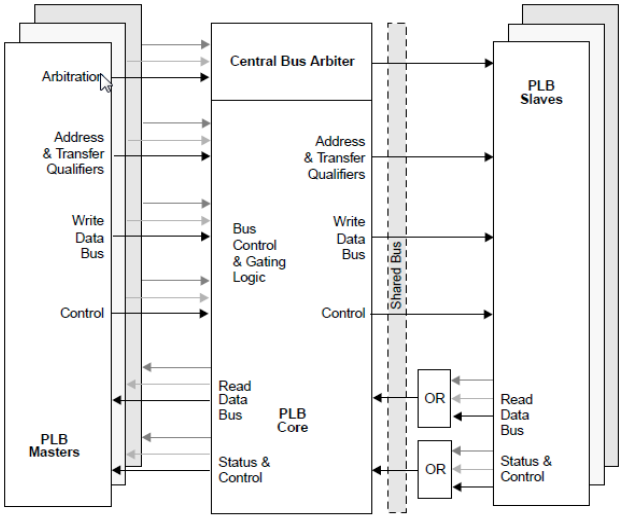

Fig. 2. PLB Overview [1].

Microblaze is a 32-bit Harvard architecture RISC processor. As being Harvard architecture it has a separate program and data bus running concurrently to execute programs or access memory or peripherals. A template for Microblaze system is seen in Fig. 1. Almost every part of this processor system is optional, only the section which contains Microblaze core is mandatory.

In an embedded FPGA design we must partition the design as hardware and software. Time-critic, resource hungry part of the design must be in FPGA fabric. Digital signal processing tasks or cryptographic tasks fall into this category. Interfacing these regions with processor is done through bus protocols, which we will introduce next.

B. *Processor Local Bus*

Processor Local Bus (PLB) is a standard which is introduced by IBM to interface peripherals directly with processor core. As shown in Fig. 1, PLB is the two sided big arrow to which all the peripherals are connected. There are master(s) and slave(s) connected to a PLB bus. The arbitration between masters and slaves is performed on central bus arbitration unit [1],[2] as shown in Fig. 2.

In our system configuration, the processor and custom cryptographic peripheral is connected via PLB. Once configured through PLB interface, our custom peripheral is intended to work at higher speeds.

III. DATA ENCRYPTION STANDARD (DES)

The Data Encryption Standard was selected by NBS (later named as NIST) as official Federal Information Processing Standard (FIPS) in 1976 [8]. Since its key space is too small and subject to several attacks, it is replaced by triple DES, later with AES. An important feature of DES is that it is so widely investigated in cryptology community that it became a tutorial tool for understanding block ciphers and their cryptanalysis.

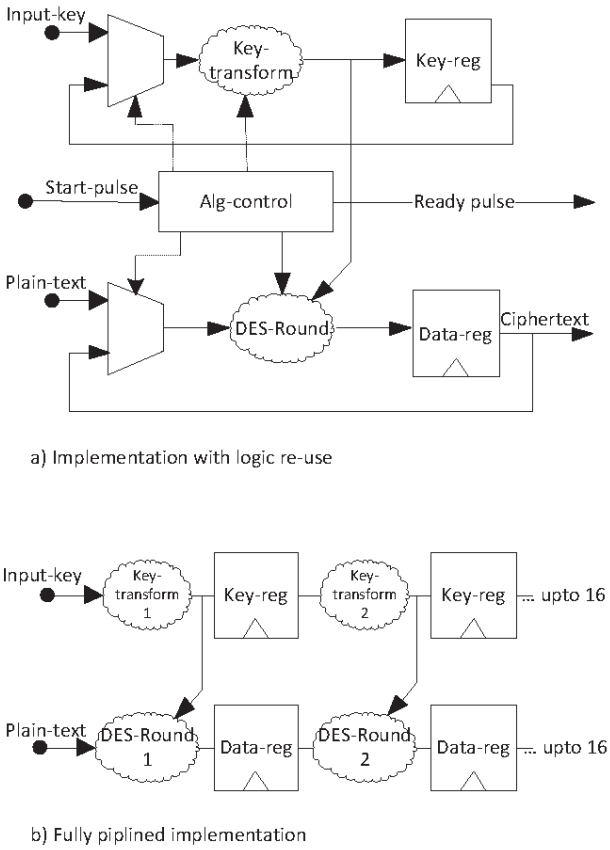

Fig. 3. Logic re-use vs. fully pipelined implementation of DES core.

Leaving the details to [3], [7], and [8], we can say that it contains 16 rounds in each of which a Feistel Network is used. S-boxes and permutations are used for achieving confusion and diffusion properties respectively.

In a straightforward implementation of DES in Virtex-6 FPGA, we obtained an area allocation of 95 slices and synthesis frequency of 350MHz. (Fig.3-a). In this implementation there is a key transform block, a Feistel Network block and a simple control logic which arranges what to do in each cycle. Once an input is given to this architecture it gives output after 16 cycles. Only after 16 cycles a new input can be fed to the module.

Although for practical communication purposes this implementation is fine, for brute force attack a fully pipelined implementation is much better in terms of throughput/area ratio (Fig3-b). Unlike the previous implementation there is not a control block in this implementation. Every key-transform and DES round has its own logic which are separated with register blocks. One can feed new key, plain-text pair into this architecture at every cycle. This architecture yields an output at every clock period after 16 clock cycles when the pipeline is full. We obtained 16 times larger throughput in expense of 10 times FPGA area. This saving is due to control logic absence in fully pipelined architecture.

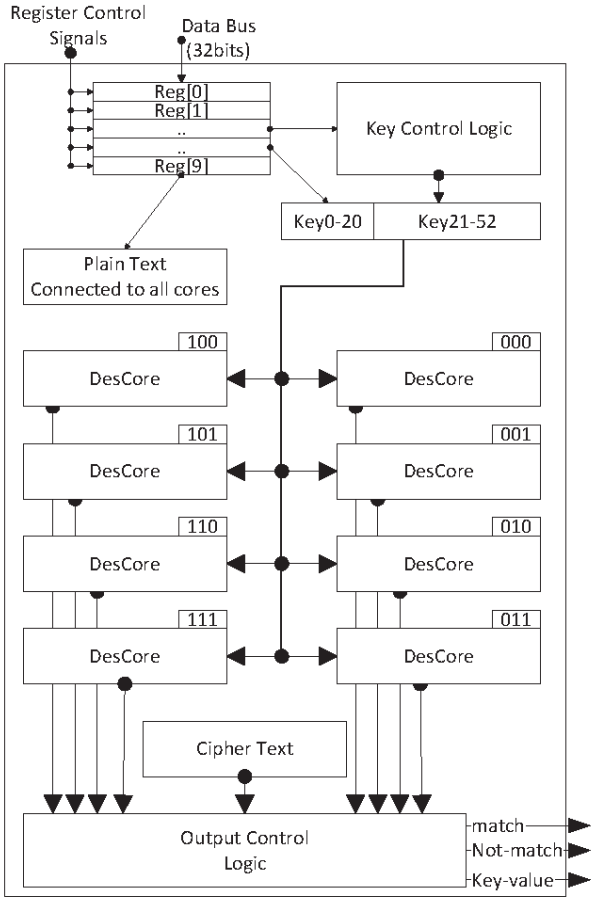

Fig. 4. PLB peripheral designed for brute force attacking to DES algorithm.

IV. Brute Force Attack Peripheral

An exhaustive key search peripheral is designed to include 8 distinct DES cores all of which are designed to run fully pipelined. These cores are managed by key control logic and output control logic blocks as shown in Fig. 4.

Key control logic is responsible to give different key values as an input to the cores. This is achieved by the sharing of the least significant 3 bits of the key separately between each core. Key control logic decides the next 32 bits of the key which starts from zero and distributes the same bits to each core. Page offset is incremented at every cycle until it reaches to 0xFFFFFF value. The key length for DES is 56 bits and the remaining 21 bits, is determined by the processor which controls the brute force attack peripherals connected to it via Processor Local Bus. Processor manages the peripherals to run in the span of all key space formed by mentioned 21 bits.

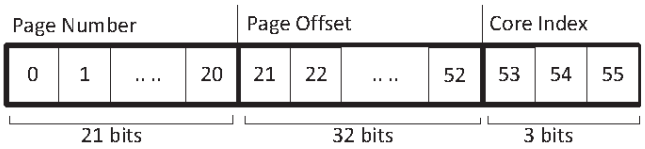

Fig. 5. DES brute force attack key space division.

The most significant 21 bits are called the page number of the keys. The latter 32 bits are called the page offset of the keys. The least significant 3 bits are constant part of the keys, each of which is connected to a DES core (Fig. 5).

The processor or the software side of the system decides the Page number in $[0\ 2^{21}-1]$ range which approximately corresponds to 2 million pages and each brute force attack peripheral spans 2^{35} keys which approximately corresponds to 32 billion.

Output control logic manages the output evaluations of the DES cores. This logic compares the reference cipher text given as an input to the peripheral with the outputs of each core in that peripheral. A core output is the computed cipher text of the plain text with the key determined by the key control logic. This plain text is also given as an input to the peripheral by the processor. Output control logic does at most 2^{35} comparisons for each run with the page number of the key given by the peripheral. If any of the DES cores finds a match between the computed cipher text and reference cipher text, the output control logic returns the key value and a *'match-found'* signal to processor. If none of the cores finds a match, a *'match-not found'* signal is generated. The processor runs the brute force attack peripherals until a match is found. In case of a *'match-found'*, the corresponding key value is read from the peripheral.

In Fig. 6 a "lucky" execution of the brute force attack peripheral core for plain text X"0123456789ABCDEF" and cipher text X"731C4E97EFA7462B" is seen. We know that the correct key for this pair is X"12680000000080". (This is the checksum stripped 56-bit key). For this simulation, brute force attack peripheral is executed with page number X"12680". As will be mentioned later, there are two clocks in the waveform. The slow clock is the interface clock with which the Microblaze processor runs, while the fast clock is for DES cores. Only a pulse-transfer which is seen in the simulation plot is made between clock domains. Once the peripheral starts, the key controller block generates the full key for each pipelined DES core at each cycle. After 16 cycles the comparator module becomes active and outputs from each core is compared against the cipher text. Once X"0000000000000000" is obtained as comparison result the corresponding key is returned as the output of the core. It is seen that for this particular case the correct key, X"12680000000080", is found 16 cycles later than the comparator's activation. Since we try 8 keys per cycle, 0x80/8=16 is the number of trials as expected.

V. Implementation And Performance Results

A. ML605 Development Platform

ML605 is an embedded development kit which has an FPGA from Virtex-6 family, XC6VLX240T-1FFG1156, on it. It has several peripherals including DDR3 memory, tri-speed Ethernet interface, SFP connection, USB to UART bridge, a dot matrix lcd, leds, dip-switches and industry standard FPGA Mezzazine Connector. Among the above we use only USB to

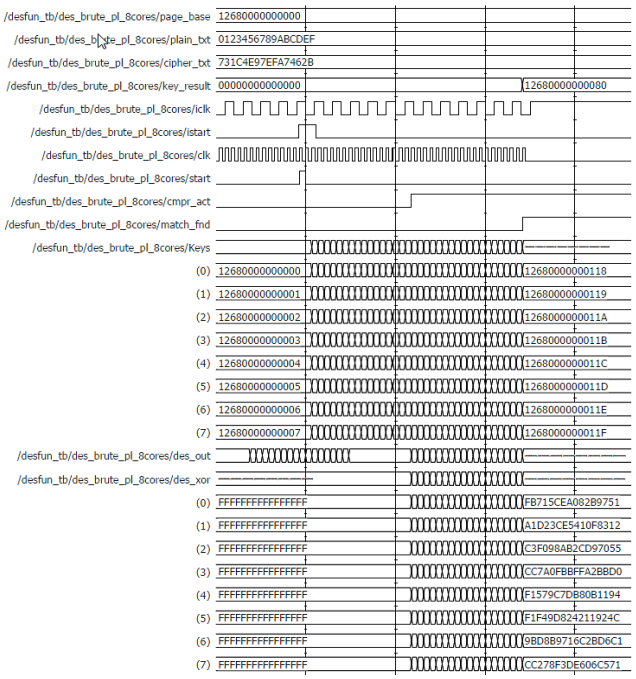

Fig. 6. Simulation result for a short search.

UART bridge and a few leds as peripheral [9]. Our main interest was about the abundant reconfigurable logic which is available on the FPGA.

With Microblaze soft processor, there is a great flexibility to select the combination of peripheral, memory and interface features for our problem solution on a single FPGA.

The ISE Design Suite provided by Xilinx includes the Embedded Development Kit (EDK) and the Software Development Kit (SDK). The EDK includes the tool suite for designing embedded applications with Xilinx platform FPGAs and Microblaze soft processor cores [10]. We developed our system design and wrote some of the HDL codes using EDK. We implemented application software for Microblaze to control brute force attack peripherals with C programming language using SDK.

B. Microprocessor System Overview

A simplified version of the hardware system is seen in Fig. 7. In the center, there is a Microblaze processor as the main control unit. The software of this processor can either run from the internal block memory or from external DDR3 memory which is available on the board. There is a debug interface over which the software can be interrupted, monitored during development phase. All peripherals are connected to Microblaze system through PLB interface. An UART and general purpose IO peripheral is picked from standard library of Xilinx embedded system platform. The rest of the reconfigurable logic area is filled up with custom cryptographic core. In our case we were able to fit 5 DES brute force peripherals into our FPGA with an area utilization of %85.

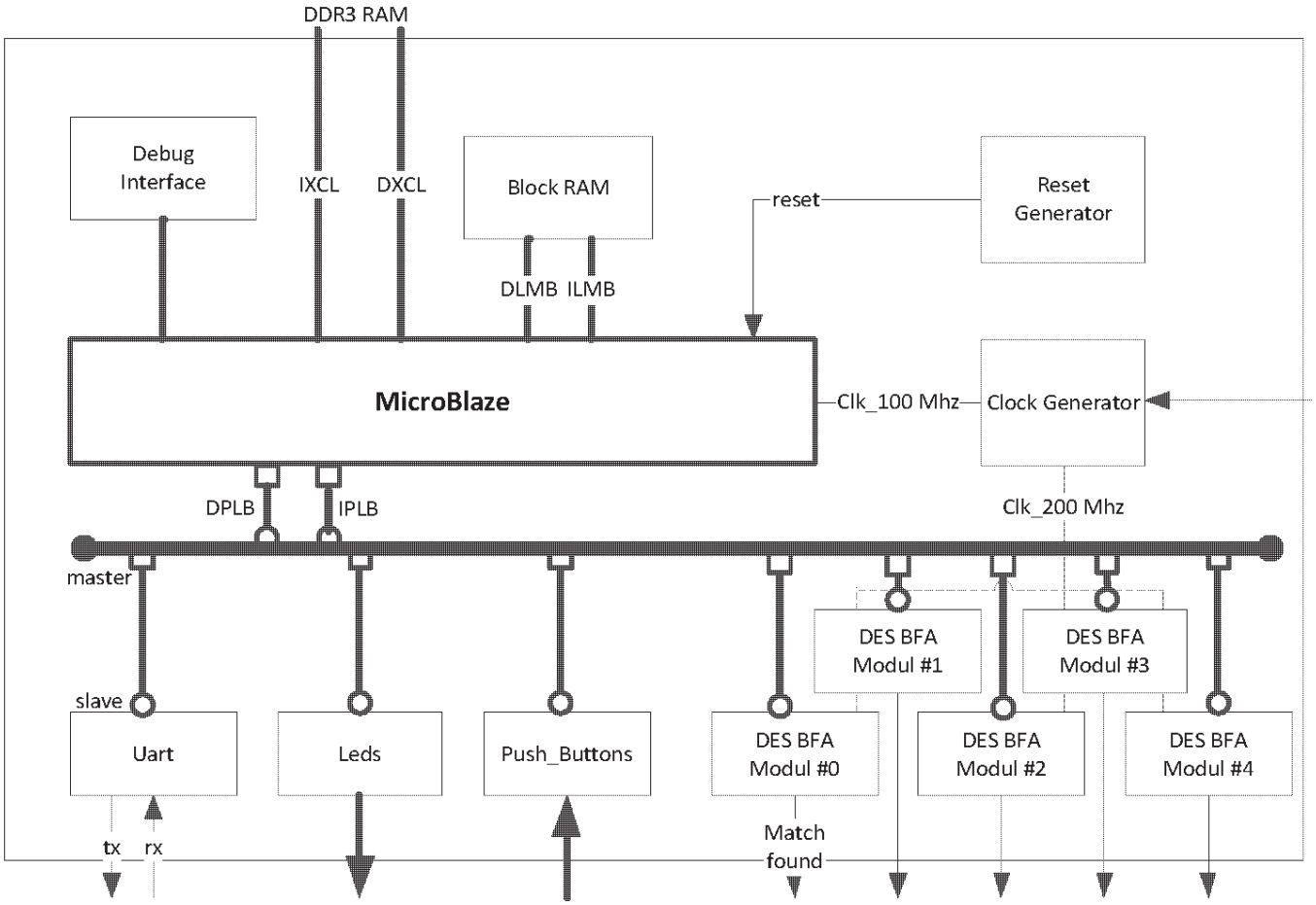

Fig. 7. ML605 Microprocessor system overview.

Synthesis frequency of an FPGA design is a critical parameter in its performance. A high synthesis frequency is desired for high performance. However it is not always possible to obtain a unique high frequency for the whole design. In our case Microblaze seems to be the bottleneck for synthesis frequency which is around 100 MHz. To overcome this limitation we introduced a higher frequency, 200 MHz, which will be the operating frequency of our cryptographic peripherals. DES brute force attack peripherals are designed such that they communicate with the Microblaze system with a slow rate while they execute internally with a faster clock speed.

C. *Performance Results*

We generated the programming file of hardware platform on Virtex-6 FPGA which includes a Microblaze soft processor and 5 brute force attack peripherals. Each peripheral consists of 8 pipelined DES cores running at 200 MHz frequency.

Each brute force attack peripheral finishes the search of the whole page in 21.5 seconds that is 8 cores x 2^{32} keys per 21.5 seconds which yields approximately 1.6 billion keys per second. A system with 5 brute force attack peripherals means that 5 page numbers of the key values are searched simultaneously. This increases the performance of our system to 8 billion keys per seconds. Thus, the correct key of a given open-text, cipher-text pair can be found in 52 days on the average.

To provide a comparison with the Copacobana system, it uses 120 Spartan-3 family FPGAs on 20 different boards with 136MHz working frequency [5]. ML605 has a single Virtex-6 family FPGA with 200MHz working frequency on DES brute force attack cores. Copacobana has a key search performance of approximately 65 billion keys per second [5], whereas our system has 8 billion.

From cost point of view, we can say that ML605 has a comparable performance with respect to Copacobana. In [11] it is calculated that an average search time of less than 9 days is achievable with Copacobana system for € 8980. With our design, parallel execution of 6 boards yields an average duration of 8.6 days with a cost of 6x€1371 = €8226.

Our main interest in this study was to demonstrate a useful methodology for implementing a computationally heavy cryptographic task on an embedded FPGA platform. With this methodology one can bridle huge amounts of configurable logic in a systematic way. The performance can be increased

by focusing on application specific needs after then.

VI. Design Practices

Change of scale in a problem may invalidate a solution which is fine for the previous scale. For example, in a small scale you may design a module, simulate it and observe every behavior of it in simulation and then burn your design into FPGA. For an exhaustive search hardware design, it is no way possible to simulate the whole behavior. It is even not possible to make simulation in most of the Embedded FPGA platforms.

Our approach in this scale is to verify smaller modules in hardware independently. After verifying all small modules, we combined them to get intermediate modules. If a verification problem occurs at a step we only focus on the last integration.

To give an example of above approach, we first designed a single DES core and verified it on hardware. Hardware verification can be done by encapsulating the core with a fixed input and comparison circuit at the output. The module is triggered with a push button and the comparison result is connected to a led which is common to all FPGA boards. If the led blinks after we press the push button on board, we make sure that the module is working as expected (Fig 7). This approach also gives the opportunity to see the speed and area performance of a module independently.

On the other extend we first generate an empty peripheral and see that it can communicate with the processor. After verifying the empty peripheral, we populate it with a verified crypto core.

This approach is also good for team work. For example while one team member is working on software development with an early peripheral, another team member can work on increasing the performance of the peripheral.

VII. Conclusion

In this paper we proposed a contemporary solution for cryptanalysis of computationally intensive algorithms on FPGA platforms. The solution consists of two parts. One is designing a custom peripheral which contains the algorithm; the other part is connecting many instances of this peripheral to a microprocessor system and control the execution of the peripherals through light software. We demonstrated the use of methodology through a well-known application, brute force attacking to DES algorithm. It appeared that we can obtain performance results compared to state-of-the art designs with this methodology.

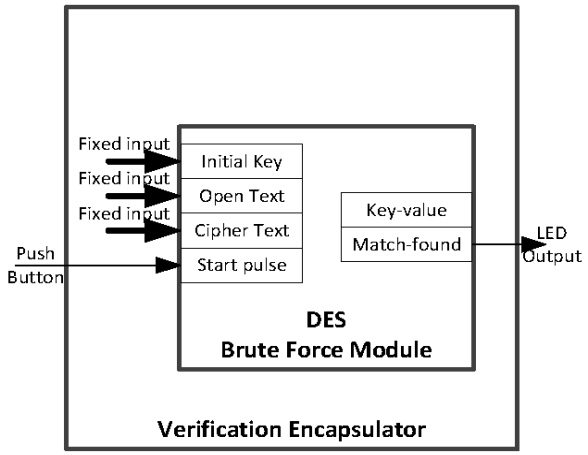

Fig. 7. Independent verification of a design module.

Acknowledgment

We would like to thank Ali Yazıcı and Bikem Temürcü for their support and motivation in this study.

References

[1] IBM Corp., *CoreConnect Bus Architecture Product Brief*, GK10-3116-00, September 1, 1999.
[2] IBM Corp., *128-bit Processor Local Bus Architecture Specifications*, Version 4.6, SA-14-2538-04, July, 2004.
[3] NIST Special Publication 800-67, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, May 2008.
[4] W. Diffie and M. E. Hellman, "Exhaustive cryptanalysis of the NBS Data Encryption Standard". *COMPUTER*, 10(6):74–84, June 1977.
[5] Tim Güneysu, Timo Kasper, Martin Novotny, Christof Paar, and Andy Rupp. "Cryptanalysis with COPACOBANA". *IEEE Transactions on Computers*, 57(11):1498–1513, 2008.
[6] COPACOBANA—A Cost-Optimized Parallel Code Breaker. http://www.copacobana.org/
[7] Christoph Paar and Jan Pelzl. *Understanding Cryptography, A Textbook for Students and Practitioners*, Springer, 2010.
[8] FIPS PUB 46-3 , DATA ENCRYPTION STANDARD (DES) , October 1999
[9] Xilinx, ML605 Hardware User Guide UG534 (v1.6) July 18, 2011 Available at http://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf
[10] Getting Started with the Xilinx Virtex-6 FPGA ML605 Evaluation Kit, UG533 (v1.5) October 20, 2011 Available at http://www.xilinx.com/support/documentation/boards_and_kits/ug533.pdf
[11] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler. "How to Break DES for € 8,980". In SHARCS Workshop. Cologne, Germany, 2006.