International Journal of Electronics and Communication Engineering (IJECE)
ISSN 2278-9901
Vol. 2, Issue 4, Sep 2013, 33-40

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

SCALABLE ARCHITECTURE FOR COMPUTATIONALLY INTENSIVE APPLICATIONS

HARSHALI ZODPE[1], SHWETA KUKADE[2], PRAKASH WANI[3] & RAKESH MEHTA[4]

[1,2] Department of Electronics and Telecommunication, Maharashtra Institute of Technology, Pune, Maharashtra, India

[3]Department of Electronics and Telecommunication, College of Engineering, Pune, Maharashtra, India

[4]Mechatronics Test Equipment (I) Pvt. Ltd., Pune, Maharashtra, India

ABSTRACT

High Performance Computing Platforms are being used to address operations with complex computational requirements or with significant processing time requirements or requirement to process significant amount of data. With the advent of low cost Field Programmable Gate Arrays (FPGA's), building hardware with parallel architecture for computationally intensive applications has now become possible. FPGA's offer massive and parallel architectures. This paper presents a FPGA based design of parallel architecture which is scalable for hardware implementation of computationally intensive applications. The aim of this work is to design a reconfigurable parallel and scalable High Performance Computing Platform to accelerate computations. The Cryptanalysis of Advanced Encryption Standard (AES) Algorithm is used as a proof of concept.

KEYWORDS: AES, Cryptanalysis, FPGA's, High Performance Computing, Parallel and Scalable Architecture

INTRODUCTION

In recent years, FPGAs have become very popular as coprocessors in high performance computing systems. FPGAs achieve much better performance than multi-core CPUs for certain types of computations. Also several computationally intensive applications implemented PCs do not utilize all the hardware functionalities provided by the standard CPUs. For efficient utilization, increased speed up and reduced power consumption, building special purpose hardware is a solution for massively parallel computation. The reliance of the world's infrastructure on computer systems, and the consequent pervasiveness of the latter, makes their "security" an issue of great importance. With the advent of high speed electronic communication there is more information than ever to protect. The constant increase of information transmitted electronically has led to an increased reliance on cryptography. The security of symmetric and asymmetric ciphers is usually determined by the size of their key-length. Hence when designing a cryptosystem, the key-length must be chosen according to the assumed computational capabilities of an attacker. Cryptanalysis of modern cryptographic algorithms involves massive and parallel computations. Cryptanalysis is the study of retrieving the plain-text without knowledge of the valid key. In the absence of mathematical breakthroughs to a cryptanalytical problem, a promising way to tackle these computations is to build special-purpose hardware which will provide better cost-performance ratio. the high non-recurring engineering cost for ASIC's had put most projects for building special purpose hardware for cryptanalysis out of the reach for commercial or research institutions and the performance of software programmed processors is dependent on the clock and usually takes larger computation time. However, FPGA's offer advantages over traditional software and hardware implementations of computationally intensive algorithms.

In the proposed work, we present a FPGA based design for customized high performance computing platform for solving computationally intensive operations. The potential for scalable performance in High Performance Computing results in performance improvement by several orders. A system, including all its hardware and software resources, is called

scalable if it can scale up (i.e., improve its resources) to accommodate ever-increasing performance and functionality demand and/ or scale down (i.e., decrease its resources) to reduce cost. The Cryptanalysis of Advanced Encryption Standard (AES) Algorithm using brute force attack is used as a proof of concept. A lot of today's most popular parallel computing systems with focus on cryptanalysis such as the COPACOBANA [1] consists of dedicated hardware especially built to solve a small range of specific problems. In contrast to these systems, our cluster exclusively consists of standardized interfaces and components and is freely scalable and configurable. Thus, it is not limited to a small number of problems. As the focus of this work is to design a special purpose reconfigurable hardware platform, brute force attack is considered for cryptanalysis and not much attention is paid to various cryptanalysis techniques.

HISTORICAL BACKGROUND

Cryptanalysis has a historical background. Research on special-purpose hardware for cryptanalysis has a rich and illustrious history. In 1938, Polish mathematicians led by Marian Rejewski constructed the Bomba Kryptologiczna (or Bomba for short), an electromechanical machine that allowed them to break the German Enigma cipher by exhaustively trying all 17,576 rotor positions. This success was expanded by British cryptographers (most notably Alan Turing and Gordon Welchman), who designed ingenious cipher-breaking machines enabling the Allied forces to read Enigma encrypted messages during World War II [2]. A parallel effort of cryptanalysis of another German cipher, the Lorenz SZ40/42, resulted in the construction of Colossus, one of the world's first programmable computers. Colossus contained 1,500 thermionic valves (vacuum tubes) and was able to process 5,000 characters per second [3]. In the 1980s, Pomerance et al designed a hardware architecture called Quasimodo for factoring large integers using the quadratic sieve algorithm. Quasimodo was actually built but never functioned properly [4]. In 1998, the Electronic Frontier Foundation (EFF) built a parallel key search DES hardware cracker called Deep Crack with an overall budget of 210,000 US$ [5]. Deep Crack consists of about 1,500 custom chips and needs at most nine days to find a 56-bit DES key by "brute force." In the late 1990s, Shamir proposed TWINKLE, an electro-optical device for performing the sieving step of the Number Field Sieve (NFS) algorithm [6]. He estimated that a single chip of the size of a 6-inch GaAs wafer would allow one to factor a 512-bit number in reasonable time and, as a consequence, break 512-bit RSA keys. TWIRL, a successor of TWINKLE, could reduce the total sieving time for 512-bit numbers to less than ten minutes [7]. Even though both TWINKLE and TWIRL are purely hypothetical devices that were never built due to technical issues (e.g. too large chip area) and high cost, they received considerable attention in the cryptographic community and initiated a slew of follow-up research [8, 9]. Recent attempts to implement cryptanalytic devices mainly use FPGAs as underlying hardware platform [10, 11]. In 2006, the Cost Optimal Parallel Code Breaker (COPACOBANA) for DES brute-force attack was built for less than US$ 10,000 [1]. COPACOBANA, hosting 120 low-cost FPGAs, was successful in breaking some symmetric ciphers with a key size of up to 64 bits, e.g. KeeLoq, DES, and A5/1. Thus, with the emergence of newer and powerful devices, continuous efforts are being made to make cryptanalysis faster and better.

AES ALGORITHM

In November 2001 the Rijndael algorithm was chosen as the Advanced Encryption Standard by the National Institute of Standards and Technology (NIST) as the successor of the Data Encryption Standard (DES) [12]. AES is a symmetric-key block cipher. AES operates on 128-bit data blocks and accepts 128-, 192- and 256-bit keys. It is an iterative cipher, which means that both encryption and decryption consist of multiple iterations of the same basic round function. In each round, a different round (or internal) key is being used.

In AES, the number of cipher rounds depends on the size of the key. It is equal to 10, 12 or 14 for 128-, 192- or

256-bit keys, respectively. Based on the internal structure of a round function, AES belongs to the group of SP-network block ciphers. The main transformations employed in this cipher are substitutions and permutations and are applied to all bits of data block in every round. AES encryption round employs consecutively four main operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. For decryption the inverse version of all these transformations is required. These inverse transformations are called InvSubBytes, InvShiftRows, InvMix- Columns, and InvAddRoundKey. Cryptanalysis involves finding plaintext from the known ciphertext without the information of the key. Thus for cryptanalysis we need to perform decryption for all possible keys. The decryption process is as shown in Figure 1.

The input to the decryption block is a 128-bit ciphertext. Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing Nb bytes, where Nb is the block length divided by 32. In inverse byte substitution transformation the inverse S- box is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF(28). In inverse shift rows transformation the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb -shift(r, Nb) bytes, where the shift value shift(r, Nb) depends on the row number. The inverse shift rows transformation proceeds as follows:

$$S'_{r,(c+shift(r,Nb))modNb} = Sr,c \text{ for } 0 < r < 4$$

$$and\ 0 \leq c < Nb$$

In inverse mix column transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by,

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

The above steps are executed in a round loop for round = 1 to Nr-1. The last round has only three steps viz. Inverse byte substitution, Inverse shift rows and Add round key. The output of the decryption block is 128-bit plaintext.

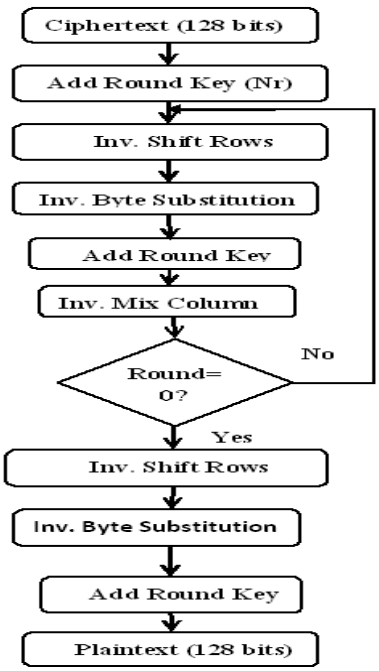

Figure 1: AES Decryption Process

The AES algorithm takes the Cipher Key, ***K***, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of ***Nb*** (***Nr*** + 1) words: the algorithm requires an initial set of ***Nb*** words, and each of the ***Nr*** rounds requires ***Nb*** words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i* in the range 0 < *i* < ***Nb*** (***Nr*** + 1).

In cryptanalysis the captured ciphertext is decrypted with all possible keys. Hence a counter is used to generate the Cipher Key, K and its expansion is performed to generate a key schedule. A known-plaintext attack requires the adversary to have access to (part of) the plaintext corresponding to the captured ciphertext blocks. In the proposed design using brute force technique, the captured ciphertext block is decrypted with all possible keys and the resultant plaintext is compared with the know-plaintext. As shown in Figure 2, the Decryption block decrypts the ciphertext and the resultant plaintext is compared with the known-plaintext. The key, for which the resultant plaintext matches with the known plaintext, is considered to be the correct key. Each Decryption block decrypts the ciphertext with different key. Thus with 'n' nos. of Decryption blocks 'n' different keys can be searched in one clock cycle thereby reducing the time required for key search by a factor of 'n'. The number of Decryption blocks 'n' depends on the available logic resources in an FPGA device and the logic utilization of one Decryption block.

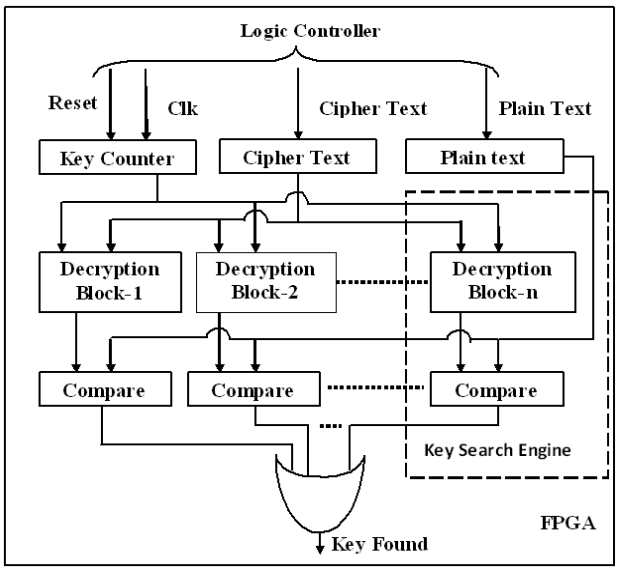

Figure 2: Cryptanalysis

SYSTEM ARCHITECTURE

The proposed system is designed considering the following criteria's: i) Computationally intensive operations which have independent solution space can be parallelized. ii) All concurrent instances have a limited requirement to communicate with each other. iii) The demand for data transfer between host and computation elements is low as computations heavily dominate communication. iv) Most of the algorithms and their corresponding hardware nodes demand for very little local memory, which can be provided by the on-chip RAM of an FGPA. v) Since the cryptanalytical applications demand for plenty of computing power, a hardware architecture that is scalable and uses standard of the shelf commercial components is an ideal solution.

The idea is to instantiate multiple instances of the algorithm simultaneously, so that the solution space is explored at a fast pace and the solution is found at a fraction of time required by a sequential process. As seen in Figure 3, the system is architected as a single master and multiple slave system. It provides a USB interface to the external world and its functions are controlled using a standard computer.

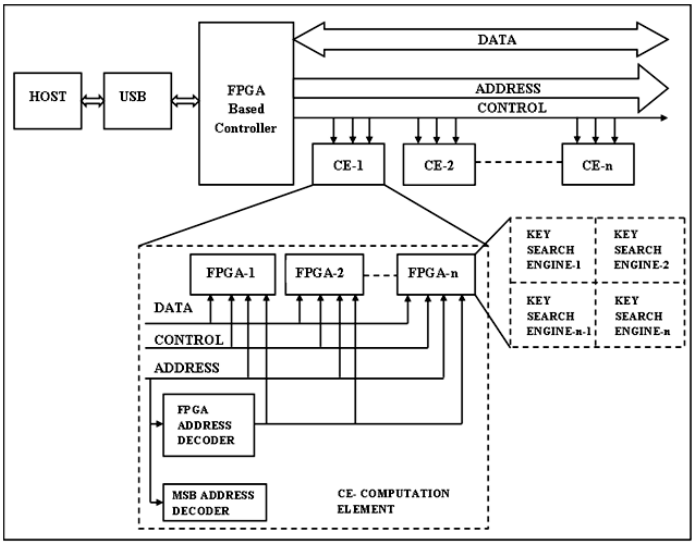

Figure 3: System Block Diagram

The basic hardware components of the system are Controller Card, Computation Elements and host PC. The Controller card and the Computation Elements are interfaced using a Backplane Chassis. A Power Supply Card is used generate the required voltage levels for the Controller Card and the Computation Elements. Virtex 6 FPGA (XC6VLX240T-1FFG1156C) is used as the target device in Controller Card and the Computational Elements. The detailed functionality of each block is explained as follows:

Controller Card

The Controller Card controls the data transfer between the host PC and the Computational Elements. This is acting as a master in the system. On the host side it communicates with the host computer to receive commands, receive Computing Element FPGA configuration files and send back computed results.

On the computing elements side it configures the individual Computing Element FPGAs with the required algorithms, passes Data to FPGAs and polls FPGAs for recent results. The detailed system block diagram for Controller Card is as shown in Figure 4. the Controller Card includes standard interfaces.

This card is mounted in one of the slot of the backplane chassis. The power supply circuit takes the DC voltage from the main Power Supply Card and converts it to the required voltage level i.e. 3.3V. The Clock Circuit provides the system clock. The Reset circuit is the master reset for the system.

The USB interface is used to communicate with the host PC. The DDR2-SDRAM is interfaced and reserved for future use. Switches are provided as handshaking signals. LEDs are interfaced to facilitate visual indications for testing, debugging and functional use.

The configuration files required to program the FPGA are stored in PROM. VPX connector is used for Interconnection Bus interface and Rocket IO protocol is used for transfer of data between controller and slave card. It offers a data rate up to 6.5 Gbps. As computations heavily dominate communication, any other communication protocol with less data rate can also be used.

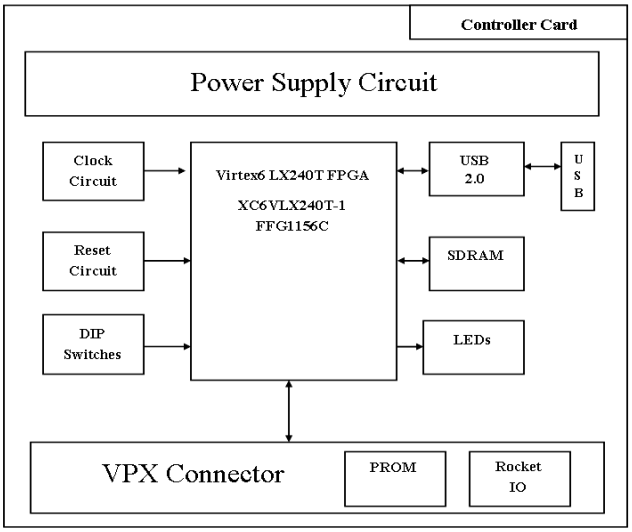

Figure 4: System Block Diagram for Controller Card

Slave Card

Figure 5 gives the detailed architecture of the Slave Card. The Computation Element forms the slave card. Up to 20 slave cards can be connected in the backplane chassis. All these Computational Elements are connected to a common bus and each element has a fixed hardwired address. Each computing element will house six Virtex6 FPGAs. The computing elements are configured with multiple instances of the same algorithm and search for a key in their allocated solution space. The algorithms inside the computational elements are designed such that more than one solution can be searched in a single cycle. Each FPGA is assigned a unique address using address decoder circuit and has the same configuration file.

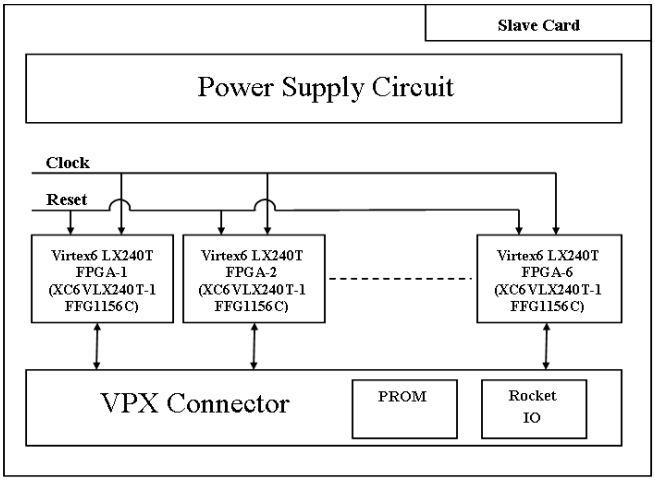

Figure 5: Slave Card

All the FPGAs on all the Slave Cards are connected in a daisy chain and programmed using a sing JTAG connector and configuration file. Also the daisy chain is designed such that even if one of the slave card is removed, the slot will be bypassed and the chain will not break.

Power Supply Card

The power supply card will convert the 230V/50Hz AC mains to +5V - +12V DC using AC – DC converter, to be used as a source for DC – DC converters used for generating required voltages for each slave card. Separate Connectors for power supply for controller card and individual slave cards will be used.

Backplane Chassis

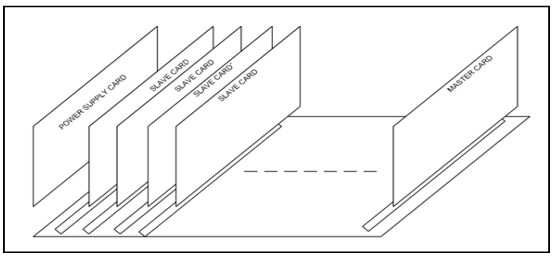

Figure 6: Backplane Chassis

The backplane hosts the controller card and the slave cards. All modules are connected by a 64-bit data bus and a 16-bit address bus. The power supply and configuration bit files are routed to all the FPGAs through the backplane bus. Figure 6 shows the structure of the backplane chassis.

Host PC

The host PC is a latest technology standard computer. It is used to configure and control the FPGA cards. It runs the customized software tools to configure the FPGA cards.

IMPLEMENTATION OF AES CRYPTANALYSIS

We first implemented the design with one instance of the key search engine on Xilinx Xc3s5000-4fg900 device. Two instances of the design could be fit in a single FPGA chip. The results in terms of area and number of keys search per second are compared with [8] and it is found that our design gives better result. The same design is then implemented on Xilinx Xc6vlx240t-1ff1156c device. The device utilization for single instance of the design is as shown in Table 1. As the solution space is independent, we then created multiple instance and instantiated them simultaneously. Eight instances of the design could be fit in a single FPGA chip such that the time required for key search is reduced by eight. The device utilization for the design with eight instances is as shown in Table 1. The results in terms of number of keys search per second are compared with [13]. and it is found that our design with multiple instances gives better result.

Table 1: Comparative Results

	Single Instance of AES	**Single Instance of AES**	**Eight Instances of AES**	**Results from [13]**
Target Device	Xc3s5000-4fg900	Xc6vlx 240t-1ff1156c	Xc6vlx 240t-1ff1156c	Xc3s5000-4fg900
Number of Slice Registers	92%	1%	12%	80.98 %
Number of Slice LUTs	90%	9%	78%	---
Min period	3.84 ns	4.010 ns	4.064 ns	3.79 ns
Max Frequen-cy	260.4 MHz	249.351 MHz	246.06 MHz	263.16 MHz
Keys/Sec/ FPGA	$520x10^6$	$244x10^6$	$1.97x10^9$	$526x10^6$

CONCLUSIONS

In this work we have presented the design for FPGA based scalable architecture for computationally intensive applications which can reconfigured run time to adapt to the required algorithm. To test the scalability and of the hardware, the designs for cryptanalysis of AES algorithm with single and multiple instances are implemented. The experimental results show that, by creating multiple instances of the design in a single FPGA, the time required for key search can be reduced by ‘n’ fold. Also we can scale the hardware architecture with multiple FPGAs in a single board to increase the number of computations per second.

REFERENCES

1. T. Guneysu, T. Kasper, M. Novotn´y, C. Paar, and A. Rupp. Cryptanalysis with COPACOBANA. *IEEE Transactions on Computers*, 57(11):1498–1513, Nov. 2008.
2. J. E. Wilcox. Solving the Enigma: History of the Cryptanalytic Bombe. Center for Cryptologic History, National Security Agency, 2001.
3. Welchman, Gordon (1984), *The Hut Six Story: Breaking the Enigma Codes*, Harmondsworth, England: Penguin Books, pp. 138–145, 295–309.
4. C. B. Pomerance, J. W. Smith, and R. S. Tuler.: Apipeline architecture for factoring large integers with the quadratic sieve algorithm. SIAM Journal on Computing, 17(2):387–403, Apr. 1988.
5. Electronic Frontier Foundation, Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design. O'Reilly & Associates Inc., July 1998.
6. A. Shamir.: Factoring large numbers with the TWINKLE device (Extended abstract). In C¸.K. Koc¸ and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 2–12. Springer Verlag, 1999.
7. A. Shamir and E. Tromer.: Factoring large numbers with the TWIRL device. In D. Boneh, editor, Advances in Cryptology — CRYPTO 2003, volume 2729 of Lecture Notes in Computer Science, pages 1–26. Springer Verlag, 2003.
8. W. Geiselmann, F. Januszewski, H. K¨opfer, J. Pelzl, and R. Steinwandt.: A simpler sieving device: Combining ECM and TWIRL. In M. S. Rhee and B. Lee, editors, Information Security and Cryptology — ICISC 2006, volume 4296 of Lecture Notes in Computer Science, pages 118–135. Springer Verlag, 2007.
9. W. Geiselmann and R. Steinwandt.: Non-wafer-scale sieving hardware for the NFS: Another attempt to cope with 1024-bit. In M. Naor, editor, Advances in Cryptology — EUROCRYPT 2007, volume 4515 of Lecture Notes in Computer Science, pages 466–481. Springer Verlag, 2007.
10. N. Mentens, L. Batina, B. Preneel, and I. M. Verbauwhede.: Time-memory trade-off attack on FPGA platforms: UNIX password cracking. In K. Bertels, J. M. Cardoso, and S. Vassiliadis, editors, Reconfigurable Computing: Architectures and Applications — ARC 2006, volume 3985 of Lecture Notes in Computer Science, pages 323–334. Springer Verlag, 2006. 14
11. G. Meurice de Dormale, P. Bulens, and J.-J. Quisquater.: Collision search for elliptic curve discrete logarithm over GF (2*m*) with FPGA. In P. Paillier and I. Verbauwhede, editors, Cryptographic Hardware and Embedded Systems — CHES 2007, volume 4727 of Lecture Notes in Computer Science, pages 378–393. Springer Verlag, 2007.
12. National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES), November 2001. Federal Information Processing Standards.
13. Andrey Bogdanov, Elif Bilge Kavun, Christof Paar, Christian Rechberger, and Tolga Yalcin.: Better than brute-force—optimized hardware architecture for efficient biclique attacks on AES-128. In SHARCS 2012, workshop without formal proceedings.