jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 3, PP. 241–246
Manuscript received February 21, 2012; revised March, 2012. DOI: 10.2478/v10177-012-0033-3

Dedicated Digital Hardware for DVB-CSA Encryption

Przemysław M. Szecówka and Patryk W. Marucha

***Abstract*—DVB-CSA (Digital Video Broadcast – Common Scrambling Algorithm) is encryption method commonly used to protect the paid channels of digital television. The paper presents a study of its implementation in specialized digital hardware. The algorithm was successfully converted to logic architecture, coded in hardware description language (VHDL), verified and synthesized for programmable logic device (FPGA). For Xlinx Spartan 3 implementation, the expected throughput may be estimated to 100 Mbps in pipelined mode.**

***Keywords*—DVB-CSA, Digital Video Broadcast, Common Scrambling Algorithm, encryption, hardware, VHDL, FPGA.**

I. Introduction

DIGITAL Video Broadcast since its introduction has continued successful expansion to the television market [1]. Nowadays it seems obvious that it will become the common standard, replacing all the traditional analogue systems as well as all competitive digital solutions. The key feature of digital broadcast is transmission of various elements of several channels in the form of bitstream. This bitstream is organized in packets of fixed length. On the receiver side the DVB consists in continuous hunting for packets marked by specific identifiers. These packets may carry some elements of the broadcast directly or shall be concatenated with several others to built up video/audio stream. It may be noted that the mechanisms of DVB transmission are inspired by the protocols well known from the internet technology. Specific proof of this association is quite common practice of utilizing internet routers for local transmission of digital TV. The cable TV operators extensively use internet dedicated equipment because of lower price and higher flexibility.

One of the key advantages of digital broadcast, besides the guaranteed accuracy of received entertainment, is the natural possibility to encrypt the transmission [2], [3]. This encryption secures the income of all the industry involved in production of television entertainment. Thanks to encryption the TV broadcast may be distributed freely but the real use of it is restricted to these end-users or local distributors who paid for the key. The commonly recognized standard of encryption is DVB-CSA (Digital Video Broadcast – Common Scrambling Algorithm). As it will be shown, the cipher algorithm seems to be (again) inspired by internet technology. DVB-CSA looks like some AES (Advanced Encryption Standard) cipher, e.g. Rijndael [4], [5], somewhat reinforced for protection against other kind of attacks. Specific policy of paid TV protection is protection of the DVB-CSA algorithm itself. It is forbidden to develop and sell software providing decryption without a license. The official documentation of cipher is not freely available either. The authors of this project suffered a lot from misunderstandings of rough specifications, appearing in the context of cipher reliability investigation [6]–[8]. This protection however does not cover prospective hardware implementation of DVB-CSA, at this stage anyway. This paper presents a study of digital architecture dedicated for DVB-CSA encryption. The design was implemented in VHDL and verified together with complementary decryption part (to be published elsewhere). The authors are pretty sure that this is not the first attempt to implement DVB-CSA in dedicated hardware, although none scientific paper on this topic was found.

P. M. Szecówka is with Faculty of Microsystem Electronics and Photonics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland (e-mail: przemyslaw.szecowka@pwr.wroc.pl).

P. W. Marucha is with Aduma S.A., Klecińska 125, 54-413 Wrocław, Poland (e-mail: patrykmarucha@gmail.com).

II. DVB-CSA Encryption Overview

DVB Transport Stream consists of packets carrying the specific data. The packet has a fixed length and consists of 4-byte header and 184 bytes of the payload. The first byte is the start pattern (01000111), used for the synchronization, i.e. detecting the start of packet in the bitstream. Another specific field is the 13-bit packet identifier (PID) which defines the purpose of the data transported in the payload section (e.g. this is the video stream for the program X). The 0x1FFF value of PID denotes the empty packets, which are sometimes inserted between the others. Usually it happens when the number of programs delivered is much below the capacity of the stream, which must flow with the defined speed regardless of the broadcast real timing

DVB-CSA encryption is a combination of stream and block ciphers. Both parts use the same 64-bit key. There are also two 64-bit initialization vectors, one delivered from outside and the other generated internally. In general the encryption may be applied to data structures not exceeding 184 bytes. These structures are divided to 8-byte blocks. For the residual data, smaller than 8-byte, only stream part of algorithm is applied. Seems like for standard DVB packets, where only the 184-bit (i.e. 23 bytes) payload is encrypted, there is no problem with indivisibility. The authors however designed more universal architecture, able to process data blocks of any size. The whole process is outlined in Fig. 1.

The encryption starts from block part. At the beginning the last 8-byte block of data BNn is XORed with initialization vector WI. The result is sent to the input of block cipher BS. The result of block encryption is stored in memory P and

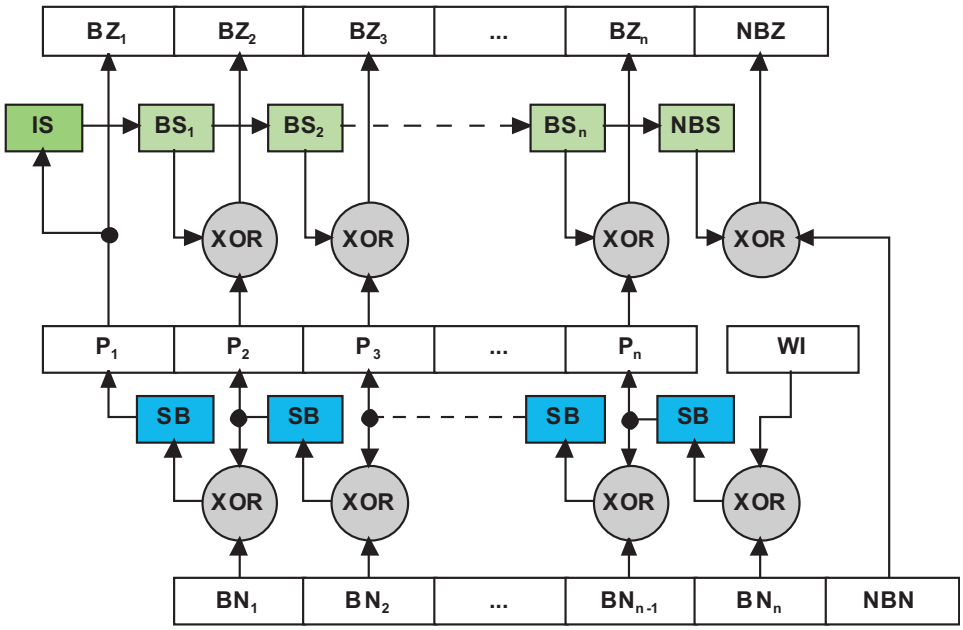

Fig. 1. DVB-CSA encryption overview. BN – input data , SB – block cipher operation, P – memory, BS – stream cipher output, BZ – encrypted output, WI – initialization vector for block cipher, IS – initializatin vector for stream cipher.

simultaneously XORed with the pre-last 8-byte block. The result of XOR is again block ciphered, sent to memory and simultaneously XORed with predecessing block of data etc. All the remaining input data is treated in this manner. The first 8-byte block of data BN1 is processed as the last one and has specific purpose. It is sent to the output directly, without stream encryption and simultaneously it is used as initialization vector IS for the stream cipher applied for the rest of block-encrypted data.

After initialization phase, the stream cipher generates consecutive bits of data which are XORed with consecutive bits of block encryption output, stored in memory blocks P. The final stage of DVB-CSA process is stream encryption of the last portion of original input data which was too small to form 8-byte block and hence omitted in block encryption, as stated above.

A. Block Encryption

Block encryption process consists of 2 phases initialization and processing. Initialization consists in extension of 64-bit key to 448 bits. This extension relies on permutation function, which moves each bit of a key to another place, strictly defined in algorithm specification. Permutation is performed 6 times, the results are XORed in appropriate way and eventually concatenated to form the extended 448 bit key.

Processing phase of block encryption is shown in Fig. 2. The central element is register H, coupled with combinatorial logic. The register contains 8 blocks of 8-bit data. The key parts of combinatorial logic are XOR with the extended key Kr, s-box T, permutation U and series of other XOR operations. The s-box table converts 8-bit vectors to other ones, according to a look-up table defined in specification. Permutation U is defined below.

U [0..7] : = S[6], S[2], S[5], S[4], S[0], S[3], S[7], S[1]

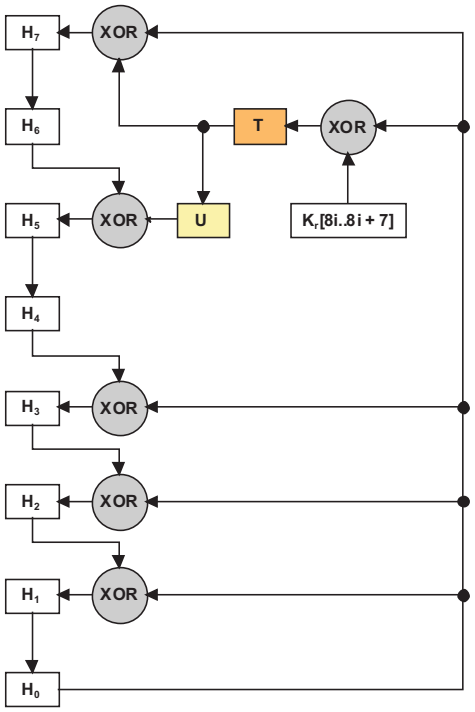

Fig. 2. Schematic of DVB-CSA block encryption; processing phase.

Block encryption starts from presetting the register H with initialization vector. Then after 56 iterations (updates of H) the register contains the final block-encrypted data.

B. Stream Encrytpion

Stream encryption consists of 2 phases initialization and generation. Initialization phase is shown in Fig. 3. The data is stored in 2 registers – A and B. Each of them contains

ten 4-bit nibbles. At the beginning the first 8 nibbles of each register are preset with 64-bit key (32 bits for each register). The last 2 nibbles of A and B are reset. Then all the data is shifted 32 times, with combinatorial logic T1-T5 involved. T1 transform operates on the first 8-byte block of data encrypted with block cipher. In consecutive iterations consecutive bytes are processed in such way that 4 clock cycles are devoted to each byte. Depending on parity of iteration counter the div and mod results of division by 16 are directed to the outputs IA and IB and swapped respectively. T2 transform is a combination of the s-box transforms applied to seven 5-bit vectors constructed from permutations of selected bits of register A. The 2-bit outputs from all s-boxes are concatenated to form 14-bit result of T2, which is directed to T3, T5 and XORs. These 5-to-2 s-boxes are special, different from 8-to-8 s-box described above. T3 is relatively simple operation applied to register B. Selected bits of B are coupled to form four 4-bit combinations. Each 4-bit combination is then XORed and this way 4-bit result is obtained. T4 block is a combination of muxes, adder and registers, providing 1 clock cycle delay. In consecutive iterations E is a copy of F, F is either copy of E or sum of E, Z and c, depending on q, where c is previous value of carry output of the adder, stored in register. T5 is a simple shift of bits obtained from XOR. The whole initialization phase takes 32 clock cycles.

Generation phase is shown in Fig. 4. It is quite similar to initialization phase described above. The different elements are: no more T1 transform (and its result no longer directed to XOR), no more D feedback from the final XOR and T6 transform inserted to the output. T6 involves 2 XOR operations applied to 4-bit vector D:

W'[0] = D[1] XOR D[0]

W'[1] = D[2] XOR D[3]

W' is the final result, thus the throughput of stream encryption is 2 bits per clock cycle.

III. Architecture

The device has 131 inputs and 64 outputs. Input vectors are the 64-bit input data and 64-bit encryption key. The other inputs are global clock, reset and a strobe signal forcing start of encryption process. The output is 64-bit vector with encrypted data. The design is full-synchronous with latch registers controlling the proper data flow and storing the appropriate values for the next iteration. Each iteration from algorithm description is performed in a single clock cycle. The encryption was partitioned to 3 modules – block cipher, stream cipher and control.

The architecture of block cipher is inspired by algorithm schematic shown in Fig. 2. The core of data flow are eight 8-bit registers. The s-box look-up table was implemented as a huge combinatorial coder based on multiplexing. Specific element is the key extension mechanism. It is sequential logic providing 6 iterations. It is enabled at the beginning of encryption process and then kept running for a few clock cycles. Then the result – extension key, is latched in a series of registers and the machine is turned off by disabling all the internal latch registers.

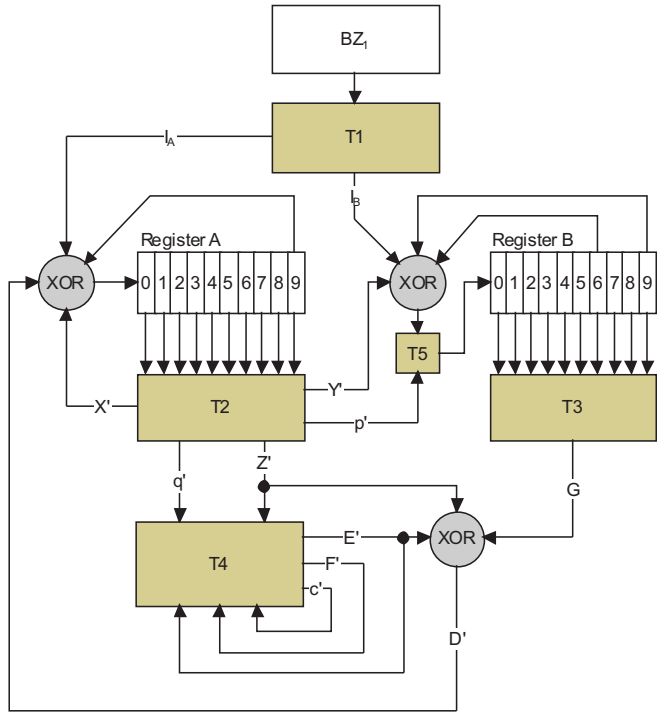

Fig. 3. Schematic of DVB-CSA stream encryption; initialization phase.

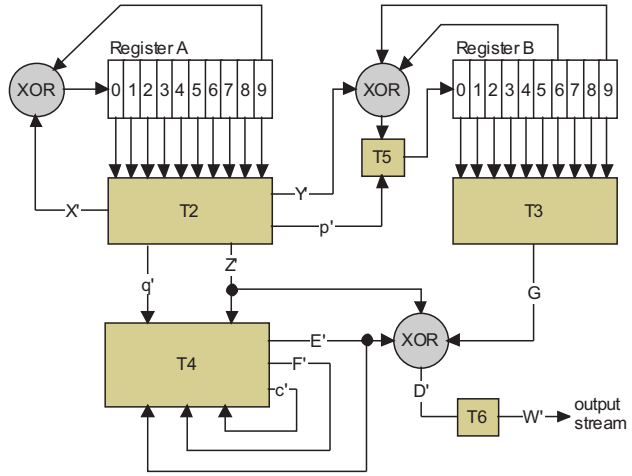

Fig. 4. Schematic of DVB-CSA stream encryption; generation phase.

Architecture of stream encryption module is shown in Fig. 5 and Fig. 6. Two schematics show the logic around registers A and B, appropriate for initialization and generation phases respectively. In initialization phase some extra logic is used, to perform T1 transform (Fig. 3). Other differences are: T6 block bypassed and D signal routed to the XOR gate at register A. The same circuitry is used for both phases, with appropriate switching, multiplexing, bypassing. Phase switching process does not harm A and B registers – their states are preserved, in accordance with the algorithm.

Block cipher requires 56 cycles to process 64 bit data. Stream cipher delivers 2 bits per clock cycle, which makes 32 cycles for 64 bits. These blocks may operate concurrently in pipeline providing a throughput of 1 data block (64-bit)

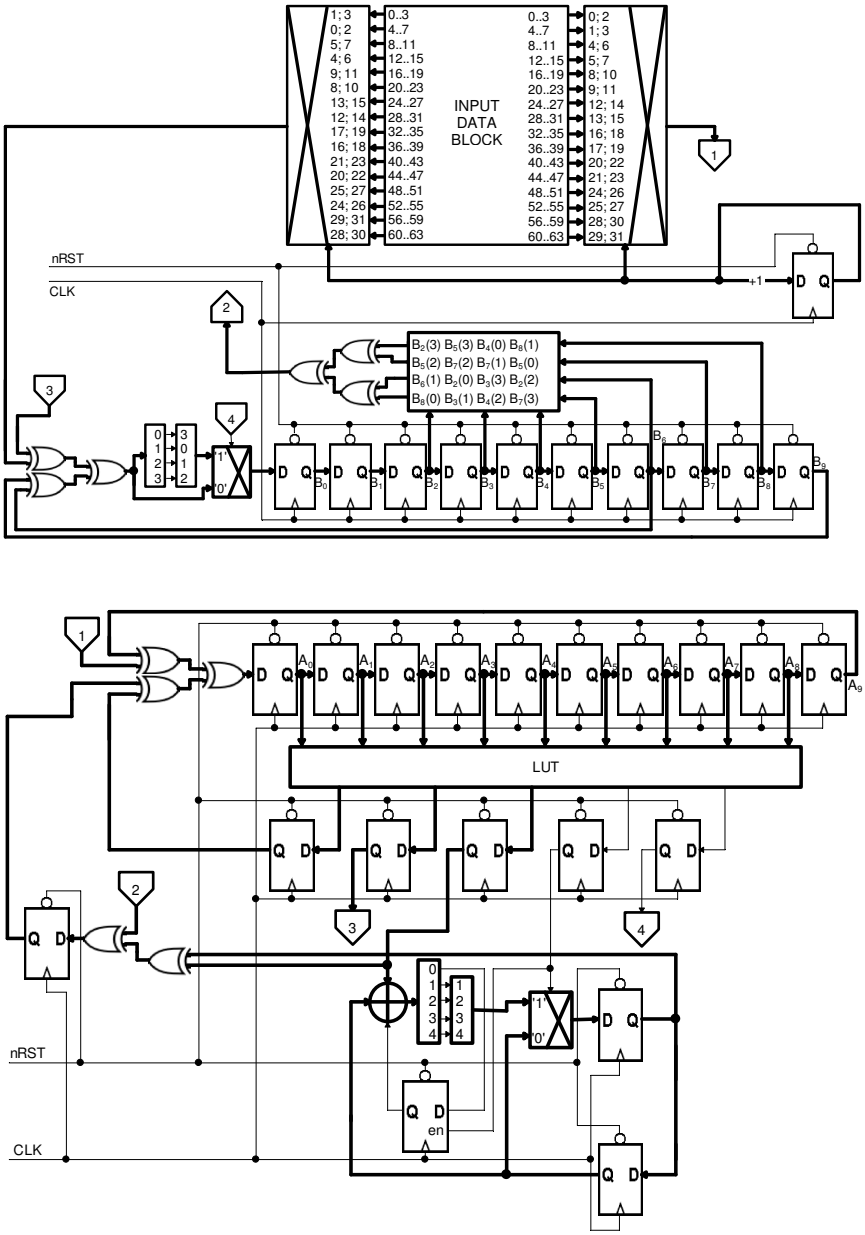

Fig. 5. DVB-CSA hardware; stream cipher configured for initialization phase.

per 58 clock cycles (block cipher slice plus 2 cycles for registers reload). A higher performance could be achieved by implementation of 2 block cipher units working in parallel and this way reducing the slice to 32-34 cycles.

IV. Implementation

Presented architecture was implemented in hardware description language – VHDL. The code was processed with Xilinx ISE tools and synthesized for Spartan 3 family FPGA. Synthesis results are summarized in Table I. The amount of FPGA resources allocated for the design – 1738 registers and 1645 LUTs mimicking combinational logic let roughly estimate the overall complexity of proposed architecture to 100k gates.

The design was simulated in Xilinx ISE environment. Verification plan covered the typical input/output patterns known from specifications. At the beginning the results were not matching. It was revealed that the reason was a little mistake in a document with algorithm description used for the design. The problem was solved by combination of experimental

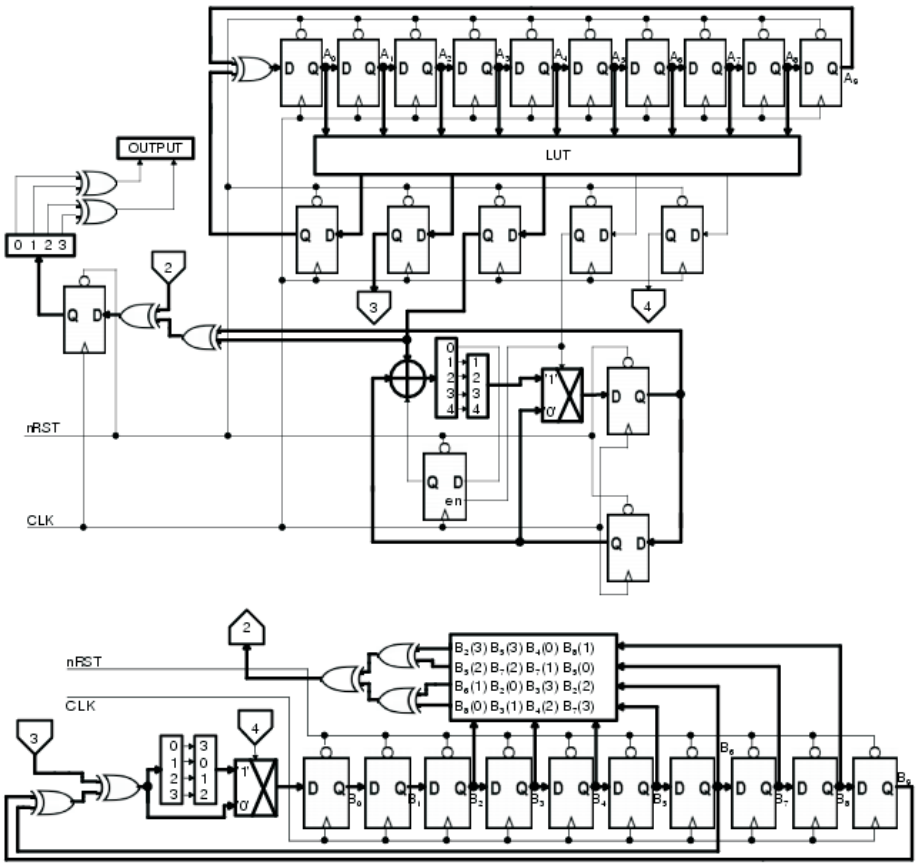

Fig. 6. DVB-CSA hardware; stream cipher configured for generation phase.

TABLE I
DVB-CSA ENCRYPTER SYNTHESIS RESULTS (XILINX SPARTAN 3)

Maximum clock frequency	97 MHz
Minimum input arrival time before clock	10.5 ns
Maximum output required time after clock	4.0 ns
Number of Slices	1644
Number of IOs	194
Number of Slice Registers	1738
Number of Slice LUTs	1645

research involving changes of the architecture and reverse engineering of C code available. Eventually the proper construction was found and successfully verified. Shall be stated however that the rough verification plan applied does not give absolute proof of compliance with DVB-CSA mechanisms used for the real broadcast.

V. STATIC TIMING ANALYSIS

Timing analysis results listed in Table I show that maximum clock frequency is quite high – very close to 100 MHz. The minimum input arrival time before clock, slightly above 10 ns, is not great result however. It exceeds the minimum clock period estimation given above. Thus for the proper operation, the clock frequency must be decreased e.g twice. Alternative solution is insertion of additional registers to the critical inputs. Detailed analysis of timing report revealed that the critical path of combinatorial logic leads from H0 register in Fig. 2 to H5 register. It contains XORing with the key, T trasnform (s-box), U transform (permutation) and eventually XOR with H6 register. Simultaneously the critical path from input to the register is pretty same – it leads from the key input to H5 register. Thus the bottleneck for both clock frequency and input arrival time before clock edge is located in the same part of design. In these circumstances insertion of a register somewhere between T and U blocks shall improve two timing parameteres. Such insertion however would destroy the natural synchronicity of encryption algorithm, based on shift register H, and would require significant redesign of both hardware and the algorithm itself. Shall be taken into account that ciphers in general are somewhat delicate matter and any change in data flow may impact the resistivity to attacks.

Maximum required time after clock (4 ns) is more reasonable. It may be reduced again by insertion of registers to the signals coming out from T6 transform logic. Concerning considerations given above, for decreased clock frequency no changes are required whilst for the redesign approach the output logic needs appropriate redesign too.

VI. Conclusions

Brief analysis of DVB-CSA algorithm led to conclusion that it is more complex than popular AES solutions like Rijndael [4]. Consequently hardware implementation is more complicated too [5]. This carefulness in protection seems questionable, taking into account that confidentiality of e.g. TV broadcast of a sport event is absolutely incomparable with internet banking money transfer. On the other hand however, prospective attacks on DVB transmission are performed in much more convenient conditions, with relatively easy and legal access to both encrypted and decrypted data of unlimited size. And the loses for the industry may be much higher than those associated with bank operation. This may explain the higher level of algorithm complication and consequently justify the need for bigger digital circuitry.

Specialized architecture for DVB-CSA encryption algorithm was proposed. Transfer from mathematical description to RTL level architecture was difficult because of algorithm complexity and simultaneously quite easy because of sequential nature of its mechanisms. Several intermediate variables appearing in DVB-CSA are updated every iteration, sometimes referring to their own previous values, which fits perfectly to a concept of register in digital electronics. There are also several operations which may be performed in parallel which fits to natural concurrence of hardware. Eventually the cipher makes an impression of being designed for hardware implementation rather than software approach. Consequently, the VHDL code appears to be more clear than C program used for reference.

Timing analysis performed in the context of FPGA technology revealed that straightforward transfer of algorithm to the dedicated hardware leads to serious timing problems. Stack of operations implemented in a long chain of combinatorial logic limits clock frequency and induces requirements for external devices which are hard to meet. More efficient approach would involve a little redesign of genuine DVB-CSA approach, perhaps followed by revision of existing knowledge about its resistivity to attacks. This issue may be very interesting for further research. For the solution presented in the paper, correct operation of encypter may require clock frequency reduced to e.g. 50 MHz.

Successful verification and synthesis proved that design is functionally correct and may be run on real hardware. Complexity estimated to 100k gates and clock frequency close to 100 MHz, gained for the first intuitive trial, show that DVB-CSA encryption is not a challenge for contemporary FPGA technology. The expected pipeline throughput is around 100 Mbps. Authors have shown that it may be physically implemented together with appropriate communication mechanisms and go for competition with existing software solutions.

Acknowledgments

Authors would like to thank Mr. Mateusz Golicz from Sileman Ltd., Ruda Ślaska, Poland, for inspiration and help with presented experimental design.

References

[1] "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems," EN 300 468 V1.3.1, 1998.

[2] S. Bewick, "Descrambling dvb data according to etsi common scrambling specification," UK Patent Application GB2 322 994A / GB2 322 995A, 1998.

[3] "ETSI Technical Report 289: Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems," European Telecommunications Standards Institute, 1996.

[4] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," in *Proc. 1st AES Conference*, Ventura CA, USA, 1998.

[5] P. M. Szecówka and D. Tekla, "Hardware implementation of rijndael encryption algorithm," in *Proc. 11th International Conference Mixed Design of Integrated Circuits and Systems, (MIXDES)*, Szczecin, Poland, 2004, pp. 561–564.

[6] R. P. Weinmann and K. Wirt, "Analysis of the dvb common scrambling algorithm," in *Proc. Conference on Communications and Multimedia Security, CMS*, Kluwer Academic Publishers, 2004.

[7] W. Li, "Security analysis of dvb common scrambling algorithm," in *Proc. The First International Symposium on Data, Privacy, and E-Commerce (ISDPE)*, 2007, pp. 271–273.

[8] L. Simpson, M. Henricksen, and W. S. Yap, "Improved cryptanalysis of the common scrambling algorithm stream cipher," in *Proc. 14th Australasian Conference on Information Security and Privacy*, Brisbane, Australia, Springer-Verlag, 2009.