VOL. 10, NO. 16, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

RECONFIGURABLE RESOURCE SHARING VLSI ARCHITECTURE FOR RC5 ALGORITHM

M. Vanitha and S. Subha
School of Information and Technology Engineering, VIT University, Vellore, India
E-Mail: mvanitha@vit.ac.in

ABSTRACT

The RC5 Algorithm is a symmetric block based, cipher which has been chosen because of its features such as simplicity of operation, implementation and its parameterizable nature. This work tries to realize the RC5 cipher on an ASIC chip and on a FPGA. The design is optimized to improve latency, throughput, area and power constraints using techniques such as loop wrapping, pipelining, parallel processing and resource sharing. A hardware implementation of the cipher has the advantage of improved speed of operation compared to a software implementation and it also improves its security. The FPGA Implementation has been done on the DE1 board while reports were taken using Xilinx ISE. The design was made reconfigurable to accept two values of rounds and keys. The ASIC Implementation was done using a fixed choice of parameters. The results achieved for area, throughput and power for an ASIC Implementation is presented. The proposed solution could be used for security in a range of applications such as wireless sensor network nodes, network devices such as routers, servers and in mobile devices.

Keywords: reconfigurable, symmetric block, throughput, power constraints, loop wrapping, security.

1. INTRODUCTION

In the past two decade with the advent of internet and rapid improvement in wireless technologies, a wide range of wireless devices have come into existence. These include wireless sensors, smart cards, laptops, smart phones etc. The list is likely to grow a lot bigger as hardware becomes cheaper and more applications begin taking to the wireless medium for data transfer. This brings an issue of privacy, which was not so much of an issue in wired devices. This is because of the very nature of the wireless medium, where the user can tune in to a particular frequency and listen to all the data that is being transmitted (at that frequency) in the surroundings. Thus security is a must when the wireless medium is considered. Security in data communication is when only the intended recipient(s) is able to understand the message the sender is sending. This can be done using cryptographic algorithms.

The RC5 cipher [1, 6] was developed by Ron Rivest. The cipher is block-based and symmetric. The advantage of the RC5 cipher over other ciphers as the DES [2] is its simplicity of implementation and its flexibility due to its parameterizable nature. Also simplicity of operations is of paramount importance to improve operation speed. The RC5 Algorithm is simpler compared to the widely used present day cryptographic standards as the Advanced Encryption Standard(AES) while providing security levels which are safe. Hence it could be used as an alternative to these algorithms when the requirements on area, delay and power are very stringent. This is especially true in wireless sensor network nodes and other smart mobile devices. Further the reconfigurable nature of the algorithm makes it possible to specify the security level required for the particular application by specifying the key size and the number of rounds. Hence a common architecture can be used for all the wireless applications without compromising either security or speed, as the case may be.

2. PREVIOUS WORKS

There are various methods of implementations suggested in the literatures. An area optimized implementation by N. Sklavos [3] *et al*. uses an encryption-decryption core with resource sharing. But the circuit uses a separate block RAM which makes data tapping possible. Another area optimized architecture by J. Liang [4] *et al*. use loop unrolling to achieve greater throughput and a barrel shifter to reduce the area of the shifter. An FPGA Pipelining Approach by A. Ruhan Bevi [5] *et al*. further improves throughput to 6.9 Gbps while the previous architecture achieves around 530 Mbps. The FPGA implementation by Elkelaany [2] *et al.* uses a reconfigurable architecture to accept the key size, word size and the number of rounds as parameters from the user.

There cipher implementation followed for the project consists of the initialization and an encryption-decryption module (EDM). The EDM affects the throughput of the entire system because the initialization is done only once, whenever the key changes. So the initialization module is area optimized, compromising on the speed of the system. While the EDM is optimized to improve throughput and reduce latency, while making a compromise for the area. Throughput is improved by using a pipelining approach and the latency is reduced by using barrel shifter for shift operations

3. RC5 ALGORITHM

The algorithm uses series data dependent rotations heavily to randomize the data during encryption. The decryption stage performs the inverse of the

operations performed in the encryption stage to obtain the original data or plain text. Both the encryption and decryption stages use the expanded version of the key called as S array for their operations.

The flexibility of the algorithm is due to the fact that the word length (W), key size (b) and the number of rounds (R), are variable. Their values can be adjusted depending on the requirements. The word length specifies the number of bits in each word which the algorithm takes as input. Increasing the word length increases the throughput. But in software implementations, it is necessary to consider the register size of the CPU. Any length greater than the size of the CPU registers degrades performance. The key size is the length of the key (in bytes). Increasing the key size improves security by reducing vulnerability to rainbow or brute force attacks. The number of rounds specifies the number of iterations in the encryption and decryption procedures. Apart from randomizing the data even further, it increases the encryption and decryption times which is a trade-off for security, because it makes brute force attacks difficult or even not possible.

3.1 S-Box initialization

This stage takes the key as input which is used to initialize the S-Array. The S-Array is used in the encryption and decryption stages. The length of the array depends on the number of rounds chosen. The block diagram representation of the S-array Pre-initialization and Key word initialization as L-array is shown in Figures 1(a) and (b) respectively. The flow chart of the operation of S-array initialization is shown in Figure-2.

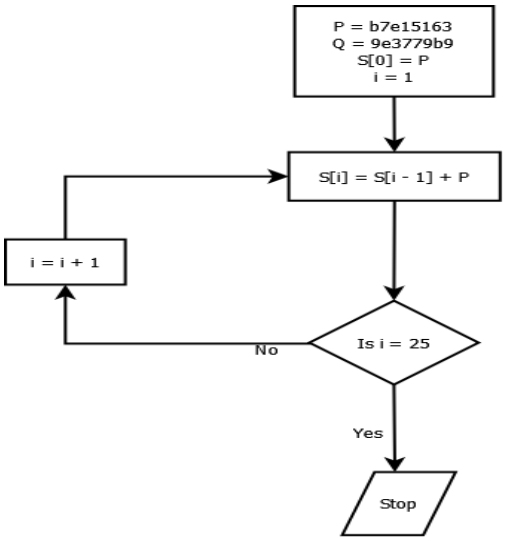

Figure-1(a). S-Array Pre-initialization.

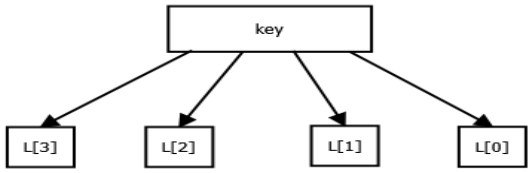

Figure-1(b). L-Array initialization.

3.2 Encryption

In the Encryption module, 2 words of data are taken and encrypted with the help of the S Array created in the previous stage. The block diagram representation for one stage of encryption is given in Figure-3. The major operation includes shifting, addition, complement and EXOR.

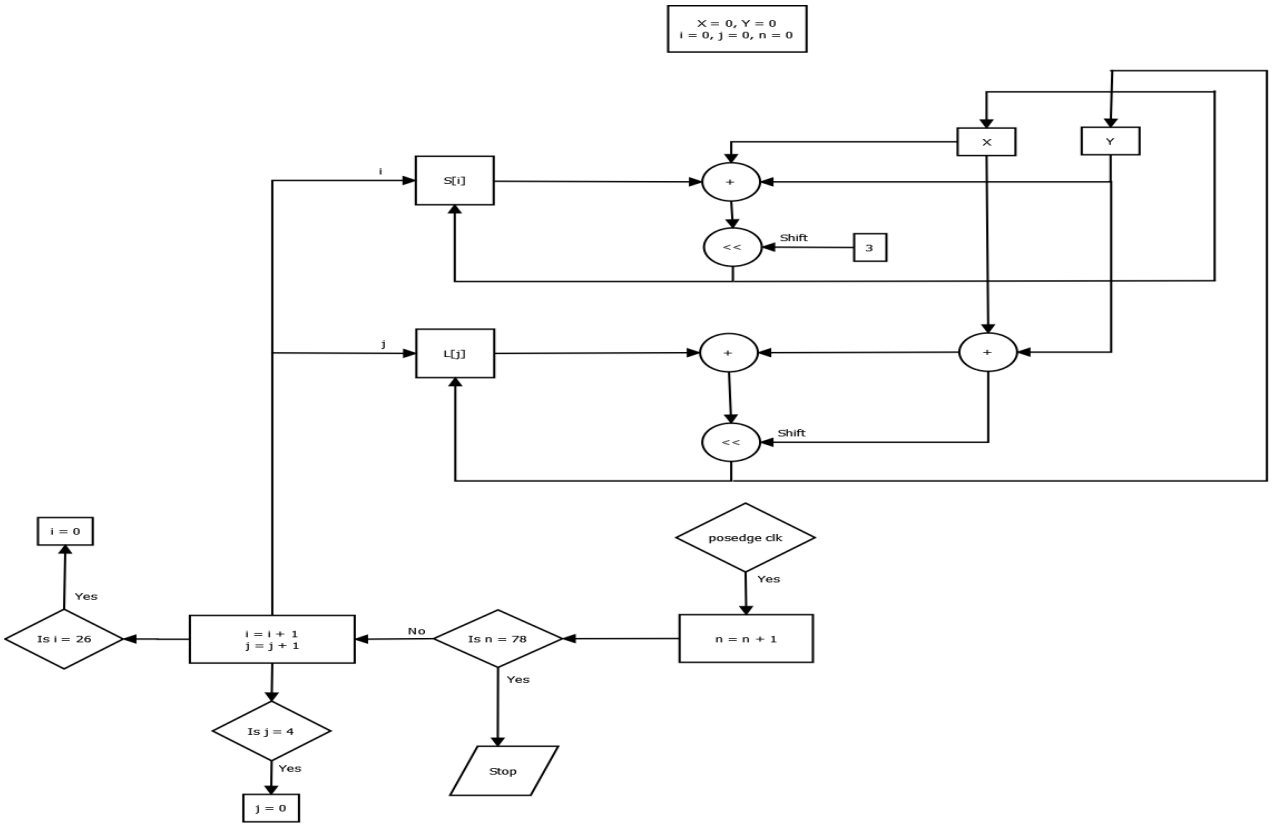

Figure-2. S-Array initialization.

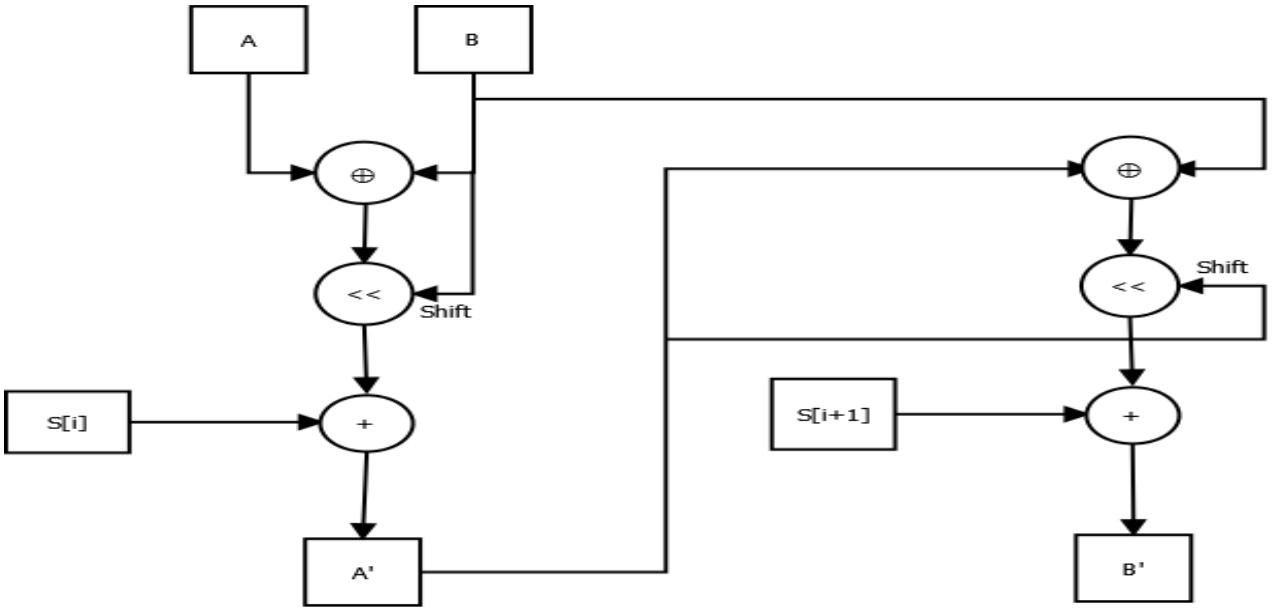

Figure-3. Encryption flowchart.

3.3 Decryption

Decryption module is the reverse of, the operations performed in the encryption stage, which is used to obtain the plaintext. The block diagram representation for one stage of decryption is shown in Figure-4.

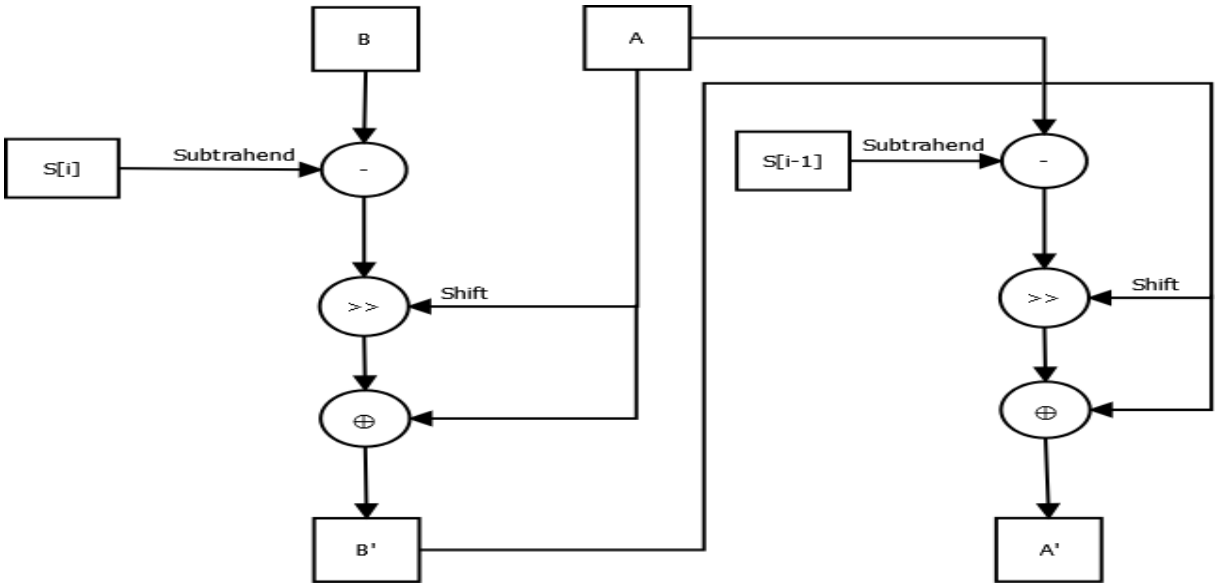

Figure-4. Decryption flowchart.

4. TECHNIQUES FOLLOWED

The VLSI architecture developed uses the following techniques to come up with efficient hardware architecture. The detailed discussions of the techniques are as follows.

4.1 Loop wrapping

If a sequential approach is taken, the critical path is the path between the key input and encrypted data at the output. This path hence decides the clock period. It is hence beneficial to divide the operations in the initialization phase over multiple clock cycles. Moreover the initialization module is used only once for each new key. The multiple loop present in the initialization phase iterates tens times. (78 for 12 rounds).

Without clock it is implemented as a set of identical modules in cascade. This 'FOR' loop can be wrapped into a single module with a clock and enable input. The enable input stays high for the required number of clock cycles after which it is disabled. This also reduces the area of the initialization module by a significant percentage. The wrapped and unwrapped architecture for multiple rounds is shown in Figures 5(a) and (b) respectively.

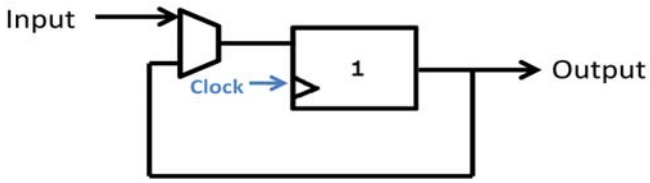

Figure-5(a). Wrapped Loop.

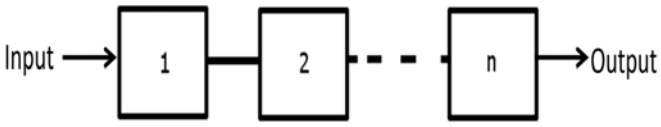

Figure-5(b). Unwrapped Loop.

4.2 Pipelining

The encryption and decryption portion in the core consists of many 'FOR' loops which are again synthesized as a series of identical modules connected in series. Whenever new data appears at the input each module does a small portion of the encryption operation and passes it to the next module. Each module remains idle after before and after performing its operations. This path is connecting

all the modules is also the critical path which decides the clock period and hence the throughput of the system. It is hence beneficial to follow a pipelining approach by providing a clock input to each of these modules and enabling them each time the output from the previous module is available. This increases throughput, so that output appears at the end of each clock cycle.

The drawback of pipelining is the increase in latency and a slight increase in area. The pipelined architecture is shown in Figure-6.

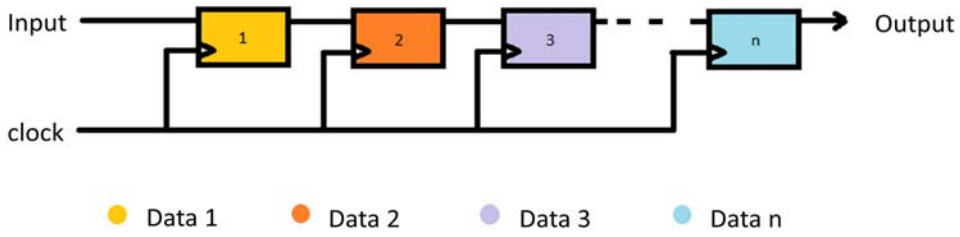

Figure-6. Pipelined architecture.

4.3 Resource sharing

A common resource sharing architecture is used for both encryption and decryption round. This reduces the need for additional registers by reusing the existing hardware. The resources are shared based on the need by appropriate timing network. There are also resources shared in the combinational logic itself, which prevents the need for additional identical circuits. A resource sharing architecture is shown in Figure-7.

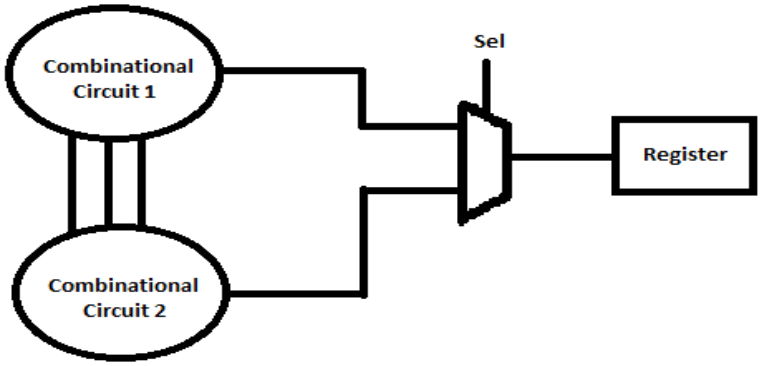

Figure-7. Resource sharing.

5. IMPLEMENTATION

The RC5 algorithm is implemented both in software and hardware. The functional verification of the encryption and decryption is done in software and hardware architecture for RC5 is implemented in both FPGA and ASIC platform.

5.1 Software

The RC5 Algorithm was first implemented using C language. The output obtained for particular values of input and parameters was functionally verified by performing the encryption and decryption. The simulated result shown in Figure-9 proves the correctness of the code modeling using C.

The same algorithm was again modeled using Verilog HDL for hardware development. The simulation was done using ModelSim. The output obtained after simulation is shown in Figure-10. The simulated results were compared with the C output to ensure correct implementation.

5.2 Hardware

For hardware implementation we plan to go for both FPGA and ASIC mode of implementation. The FPGA implementation results are compared with the existing and a dedicated ASIC chip is developed for RC5.

5.2.1 FPGA

The verilog code was synthesized and dumped onto the DE1 FPGA board using QUARTUS II Software. The key and data were coded into the Verilog [7, 8 and 10]. The output was displayed using the hexadecimal display present on the board.The design was made reconfigurable to accept two different values of round and

keys. The output obtained from the board was verified with the output of the C program programmed with same values of inputs and parameters to ensure correct functionality.

The Xilinx ISE was used to get the reports for comparison with existing [2] results in literature. The target board chosen was Virtex 6 board and FPGA device chosen was 6vlx75tff484-3.

5.2.2 ASIC

The verilog code was synthesized using 90nm technology from TSMC. Unlike the FPGA platform this architecture is not reconfigurable in the real sense. The verilog code was first compiled, elaborated and simulated using the NC Launch tool in Cadence to verify output and check for errors. Then the RC compiler was invoked to synthesize the technology mapped design of the circuit. The tool generates the netlist and constraint files used in the backend process. Also the Area, Timing and Power reports were taken after the synthesis process. The Encounter tool was used for backend process. After the various stages, the GDSII file was generated. The report generated was used to check for violations and comparison.

6. SIMULATION AND SYNTHESIZED RESULTS

The synthesized reports for both FPGA and ASIC platform [9] are compared with the existing and reports are tabulated. Table-1 shows the power consumption of RC5 architecture in ASIC with 90nm. The area and timing reports are shown in Tables 2 and 3. The resource comparison for FPGA synthesize is shown in Figure-4. The throughput comparison shows 62% throughput improvement and 56.7% area reduction compared with [2].The photocopy of the routed ASIC chip is shown in Figure-8 and the complete FPGA implementation setup in Figure-11.

Table-1. ASIC power report.

Instance	Cells	Leakage power (μW)	Dynamic power (μW)	Total power (μW)
rc5_v6	32735	3.25	16984.76	1698801

Table-2. ASIC area report.

Instance	Cells	Cell area
rc5_v6	32735	69525

Table-3. ASIC timing report.

Timing reports	
Clock Period	12ns
Timing Slack	1ps
Throughput	5.33 Gbps

Table-4. Results comparison for FPGA.

Architecture	Throughput (Mbps)	Slices used
Elkeelany *et al* [2]	300	3618
Proposed (without pipelining)	241	5343
Proposed (with pipelining)	488	5673

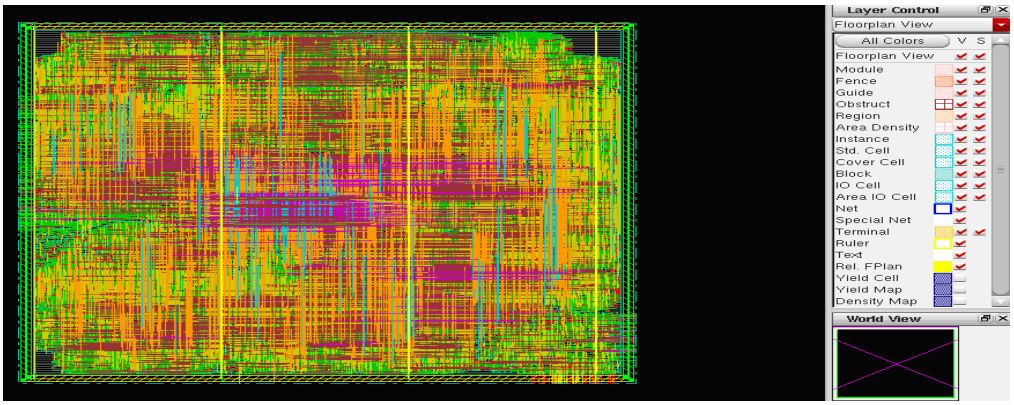

Figure-8. Routed layout.

```
Plaintext:      56788765 12344321
Ciphertext:     9630d50d 06983782
Key:            00000000000000000000000ab123cd456
Process returned 0 (0x0)   execution time : 5.053 s
Press any key to continue.
```

Figure-9. Output of RC5 C implementation.

```
Transcript
# Compile of test_v1.v was successful.
# Compile of rc5_v1.v was successful.
# 2 compiles, 0 failed with no errors.
VSIM 12> vsim work.test_v1
# vsim work.test_v1
# Loading work.test_v1
# Loading work.rc5_v1
VSIM 13> run
# Plaintext:      xxxxxxxxxxxxxxxx
# Ciphertext:     xxxxxxxxxxxxxxxx
# Key:            xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# Plaintext:      5678876512344321
# Ciphertext:     xxxxxxxxxxxxxxxx
# Key:            00000000000000000000ab123cd456
# Plaintext:      5678876512344321
# Ciphertext:     9630d50d06983782
# Key:            00000000000000000000ab123cd456
Transcript
```

Figure-10. Output of Verilog simulation using model SIM.

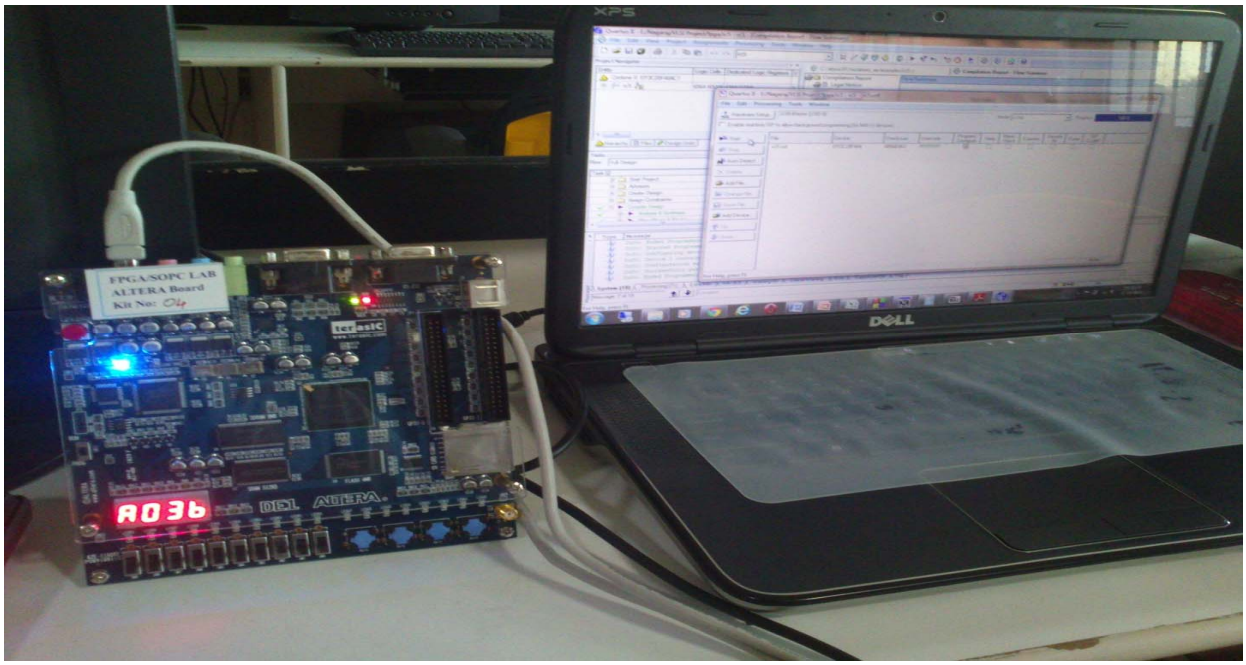

Figure-11. FPGA setup for RC5 DE1 output snapshot.

7. CONCLUSIONS

In this paper, we have done both FPGA and ASIC Implementations of the RC5 cipher. The FPGA Implementation has been made reconfigurable to demonstrate the flexibility inherent in the algorithm. The ASIC Implementation has been optimized using techniques such as loop wrapping, resource sharing, pipelining to improve parameters such as area, power and throughput. The FPGA implementation has been done on DE1 board and the Xilinx ISE tool has been used to generate various reports. The ASIC implementation has been done using Cadence Tools. The FPGA result has been compared with the implementation by Elkelaany [2]. The proposed architecture shows a throughput improvement of 62% with area reduction of 56%. The ASIC implementation can be used as a module in existing network devices where security is of importance. Further the parameters used could be changed to suit the application requirement of speed or security.

REFERENCES

[1] Ronald L. Rivest. 1995. The RC5 Encryption Algorithm. Proceedings of the 1994 Leuven Workshop on Fast Software Encryption (Springer 1995), pp. 86-96

[2] Omar Elkelaany, Adegoke Olabisi. 2008. Performance Comparisons, Design, and Implementation of RC5 Symmetric Encryption Core using Reconfigurable Hardware. Journal of Computers. 3(3): 48-55.

[3] N. Sklavos, C. Machas, O. Koufopavlou. 2003. Area Optimized Architecture and VLSI Implementation of RC5 Encryption Algorithm. In: Proc. of 10th IEEE International Conference on Electronics, Circuits and Systems (IEEE ICECS'03). 1: 172-175, United Arab Emirates.

[4] Jing Liang, Qin Wang, Yue Qi, Feng Yu. 2009. An Area Optimized Implementation of Cryptographic Algorithm RC5. Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference.

[5] A. Ruhan Bevi, S.S.V. Sheshu, S. Malarvizhi. 2012. FPGA Pipelined Architecture for RC5 Encryption. Digital Information and Communication Technology and its Applications (DICTAP), 2012 Second International Conference.

[6] Bruce Schneier. 1996. Applied Cryptography. Wiley.

[7] Samir Palnitkar. 1996. Verilog HDL - A Guide to Digital Design and Synthesis. Prentice Hall.

[8] Charles H Roth. 1995. Introduction to Logic Design. West Pub. Co.

[9] Michael D Ciletti. 1999. Modeling, Synthesis and Rapid Prototyping with the Verilog HDL. Prentice Hall PTR.

[10] M Morris Mano. 1979. Digital Logic and Computer Design. Prentice Hall.