High Throughput VLSI Architecture for RC5 Algorithm

M.Ramkumar raja [#1], Dr.K.Thanushkodi [*2], S.Arul Jothi [#3]

[#] Department of Electronics and Communication, CIET
AnnaUniversity
[1] ramkumarrajavlsi@gmail.com
[*] Director, Akshaya College of Engineering & Technology
AnnaUniversity
[2]thanush12@gmail.com
[#] Department of Electronics and Communication, SREC
AnnaUniversity
[3] arulbe2005@gmail.com

***Abstract*— In this project we realize the RC5 cipher on ASIC chip and on 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 RC5 Algorithm is symmetric block based cipher which has been chosen because of its features such as simplicity of operation and implementation and its parameterizable nature. 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. Since pipelining could not be implemented for the FPGA, the throughput was lower than that achievable. 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.**

I. 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.

Cryptographic algorithms can be divided on the basis of key usage as Symmetric and Asymmetric ciphers. In symmetric ciphers a key is used as a parameter to the encryption algorithm which takes the data and converts it into a random sequence of characters which have no relation(ideally) to the original data. This random sequence of characters is known as cipher text. This cipher text is sent to the receiver over the medium. The receiver then gives the same key as input to the decryption algorithm and converts the cipher text back to the plain text. If the key used for encryption is not the same as the key used for decryption, the cipher is asymmetric. Asymmetric ciphers are mainly used to exchange the keys for exchanging the symmetric keys which are used to establish a secure connection between devices. Asymmetric ciphers are not used extensively because they are inherently slower compared to symmetric ciphers. Ciphers can also be divided as stream based or block based ciphers based on the size difference between the cipher text and the corresponding plain text. In Block based ciphers the length of plain text is same as the cipher text but in stream based ciphers, the cipher text is usually longer than the plaintext. Though stream ciphers are less complex and easier to implement compared to Block based ciphers, they have security issues arising due to the pseudo random generators used in them.

The RC5 cipher[1] 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.

II. RC5 ALGORITHM

A. RC5 Algorithm

The algorithm uses a 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 infeasible.

1) S Array 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 flowchart and block diagram is given in Fig 1 and Fig 2.

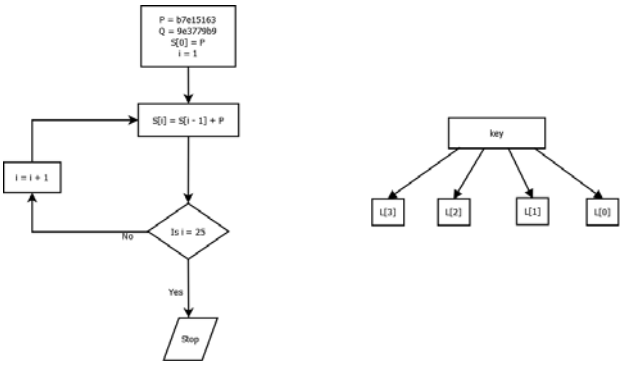

Fig. 1. S-Array and L-Array Pre-Initialization

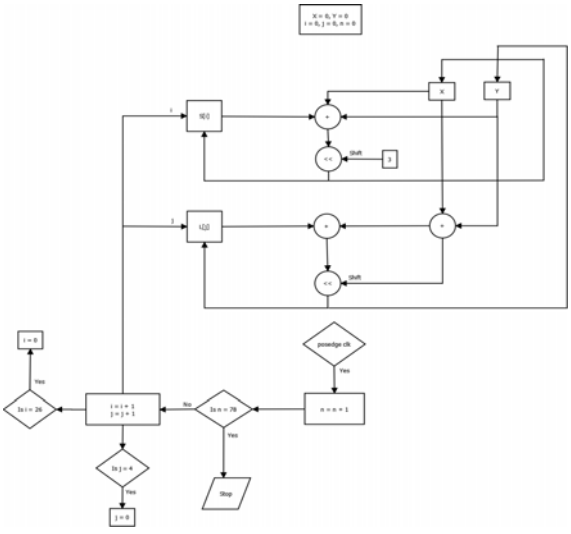

Fig. 2. S-Array Initialization

2) Encryption: In this stage, 2 words of data are taken and encrypted with the help of the S Array created. The pseudo code and block diagram for one stage of encryption is given in Fig 3.

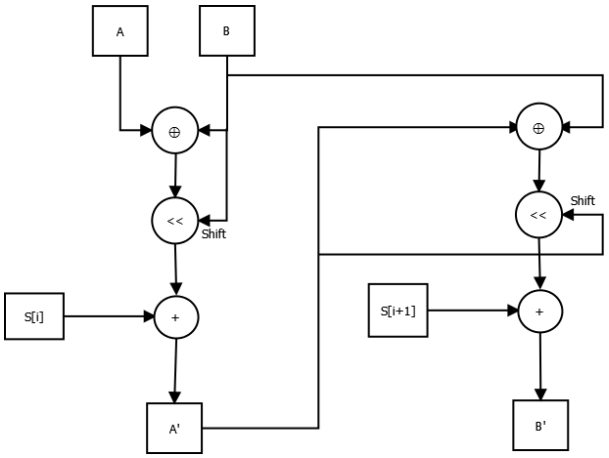

Fig. 3. Encryption Flowchart

3) *Decryption:* During this stage the reverse of the operations performed in the encryption stage is used to obtain the plaintext. The pseudo code and block diagram for one stage of decryption is given in Fig 4.

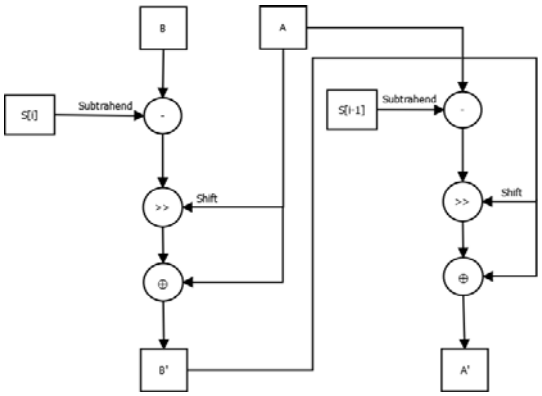

Fig. 4. Decryption Flowchart

B. ASIC

Application Specific Integrated Circuits are those which are customized for a specific purpose. These include simple ICs as logic gates and timers to complex systems as Microprocessors and Memories which contain many thousands of gates.

These circuits, like FPGAs are described using Hardware Description Languages.

These Circuits are divided, based on extent of customization as given below.

1. Full Custom ASIC

2. Semicustom ASIC

3. Programmable ASIC

In Full Custom ASICs, some or all logic cells are customized and all mask layers are customized. The Mask Layers Describe transistors and their interconnects.

In Semi Custom ASICs, all the logic cells are predefined and some or all of the mask layers are customized. These are again of two types: Standard cell Based and Gate Array based.

In Programmable ASICs, all the logic cells are predefined and none of the mask layers are customized. These include PLDs and FPGAs.

1) FPGA: A Field Programmable Gate Array is an integrated circuit designed to be configured by the customer after manufacturing. The FPGA configuration can be specified using Hardware Description Language. FPGAs usually have large resources of logic gates and block RAMs which makes it possible to implement complex digital computations.

An FPGA contains programmable logic components called logic blocks and a hierarchy of reconfigurable interconnects that allows different blocks to be connected to each other. The logic blocks can be configured to perform any combinational function. They also include a storage unit such as a flip-flop or entire blocks of memory. The figure shows the structure of a simple logic block.

The advantages of using FPGA include minimal prototyping cost and the ability to make the design reconfigurable[2].

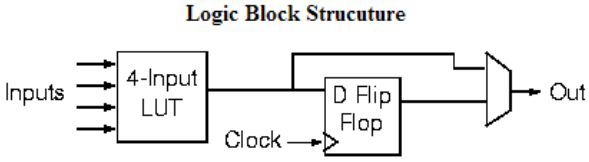

Image Credit: Vaughn Betz, University of Toronto
Fig. 5. Decryption Flowchart

Many such logic blocks are then connected via routing channels as shown in figure. The interconnects are enabled based on the logic to be realized.

FPGA chips are usually provided on development boards by manufacturers. These development boards contain many peripherals such as Memories(RAM, SDRAM, Flash), General Purpose Input Output Ports, Serial Ports(RS232, VGA), Clock Crystals, Switches(Push, DPDT), Displays(LED, LCD) etc.

These peripherals can be used in the design by mapping the I/O pins in the design with pins of these devices.

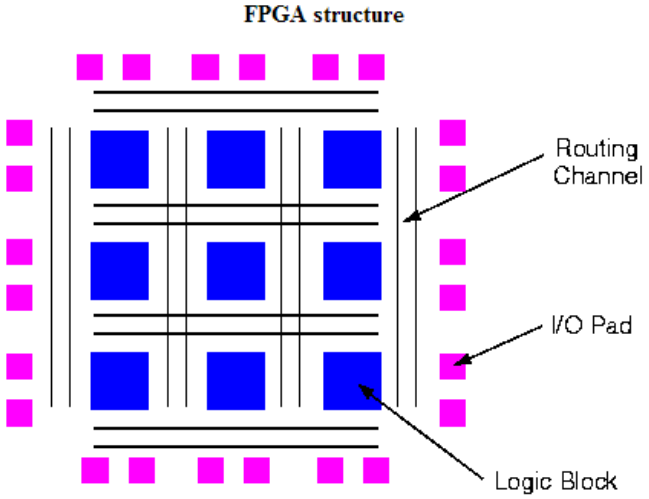

Image Credit: Vaughn Betz, University of Toronto
Fig. 6. FPGA Structure

2) Standard Cell Based ASIC: A Standard Cell is a group of transistor and interconnect structures that provide a particular logic function. These cells are usually provided by the manufacturers in the form of libraries. Theses libraries contain many of the commonly used logic functions in the form of standard cells which can be used directly in circuits. Each library corresponds to a particular semiconductor technology such as 45, 90 or 180nm. Further within each technology the libraries are again classified based on inherent optimizations used in the design of the cells as slow, fast and typical. Users can choose libraries based on their requirements of area, power and timing.

These circuits are designed using EDA tools where the HDL code is compiled synthesized into a circuit consisting of standard cells and their interconnections.

The steps involved in the conversion of verilog code to a standard cell design is explained below:

1. Compilation

In this stage the top module is compiled, elaborated and simulated to check for any errors and to ensure desired functionality. Another important function is that the compiler reports any unsynthesizable constructs present in the code.

2. RTL Synthesis

Initially the setup file was created where the top module, libraries, optimizations and names of the generated net and sdc files are specified.

It is also possible to add commands to report the power consumed, timing delay and area occupied by the circuit, to name a few.

Once this stage is completed, a technology mapped netlist and a constraints file is generated.

It is also generates reports for timing, area and power which have been tabulated in the next section.

3. Backend

This is the final stage in the process. It involves a series of steps.

a. System Partitioning
b. Floor Plan
c. Power Plan
d. Placement
e. Clock Tree Synthesis
f. Routing
g. DRC/LVS
h. Tape out

System Partitioning is the process of dividing a complex design into a set of smaller modules. After division modules can be placed either on the same chip, the same board or on different boards.If they are placed within the same chip then on connecting them, we have Intrachip delay. If they are placed on the same board, then we have Intraboard delay. If they are placed on different boards, then we have Interboard delay.Interboard delays tend to be much higher compare to the other two and hence must be avoided. While interchip delays are higher when compared to Intrachip delays.So as a general thumb rule, whenever there are lots of interconnections between modules we first prefer Intrachip connections before the others.This task is done by Partitioning Algorithms which optimize the partitioning with respect to constraints as performance, area, amount of interconnects etc.

Floor Plan

In this stage, the total geometry of the chip is decided. The I/O pads and pre-designed macro cells are placed. Enough space is left between the cells for routing optimizations and power connections.

Power Plan

In this stage, the source(VSS) and ground(VDD) rings are added to the design. Additionally power stripes can also be placed, if the design contains a large number of cells, so that all the cells can access the source and ground rings with minimum interconnect length.

Placement

This stage involves placing components on the chip.

Clock Tree Synthesis

Clock is the most important signal in the design of any digital system based on sequential logic. The following are the pre-requisites of the clock signal.The standard digital system expect the clock signal to be present everywhere within the system at the same time.The clock signal is to be connected to all flip flops. Hence it must have a high fan out.In this stage, the tool inserts multi level buffers to drive the load. It also balances the timing by ensuring equal wirelengths for all the connections to the clock.

Routing

Special Route: They are used to connect I/O power rings with the block power pins.

Nano Route: Performs signal routing ie. it connects different cells according to the netlist

III. ARCHITECTURE

There are various methods of implementations suggested in literature.

The 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.

A. Loop Wrapping

If a sequential approach is taken, the critical path is the path between the key input and output S Array. 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 for loop present in the initialization phase iterates tens of 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.

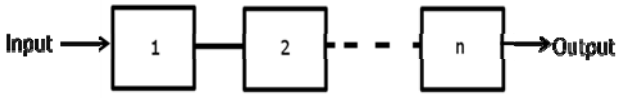

Fig. 7. Unwrapped Loop

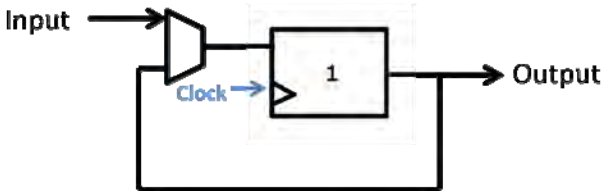

Fig. 8. Wrapped Loop

B. Pipelining

The encryption and decryption portions in the core consists of 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.

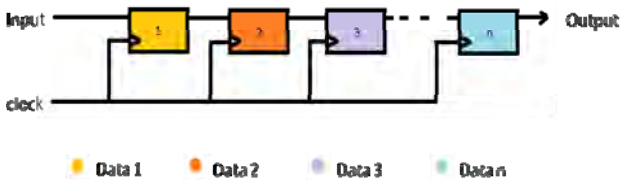

Fig. 9. Pipelining

C. Resource Sharing

A common architecture is used for both encryption and decryption round. This reduces the need for additional registers by reusing the existing hardware.

There are also resources shared in the combinational logic itself, which prevents the need for additional identical circuits.

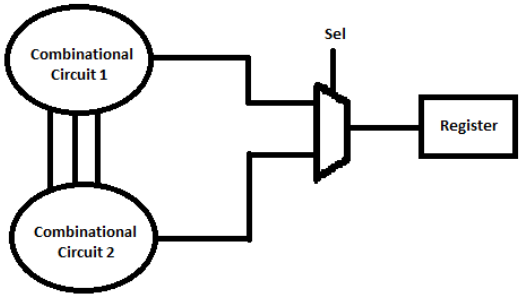

Fig. 10. Resource Sharing

IV. IMPLEMENTATION

A. Software

The RC5 Algorithm was first implemented using C language. The output obtained for particular values of input and parameters was verified with the output obtained in the original paper to ensure correctness of the code.

The Algorithm was again implemented using Verilog HDL. The output obtained during simulation was compared with the C output to ensure correct implementation.

The above step was repeated after every modification of the code and before taking final results.

B. Hardware

1) FPGA: The verilog code was synthesized and dumped onto the DE1 FPGA board using QUARTUS II Software. The key and data were hard coded into the verilog code. 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 results in literature. The target board chosen was Virtex 6 board and FPGA device chosen was 6vlx75tff484-3.

2) ASIC: The verilog code was fixed to a particular choice of parameters unlike the FPGA based reconfigurable implementation. The verilog code was first compiled, elaborated and simulated using the NC 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 post the synthesis. 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.

V. RESULTS

A. Simulation

The code for the algorithm was written using C language and Verilog HDL.

The outputs obtained for particular choices of parameters and input data were compared to ensure that the correctness of the algorithm's implementation.

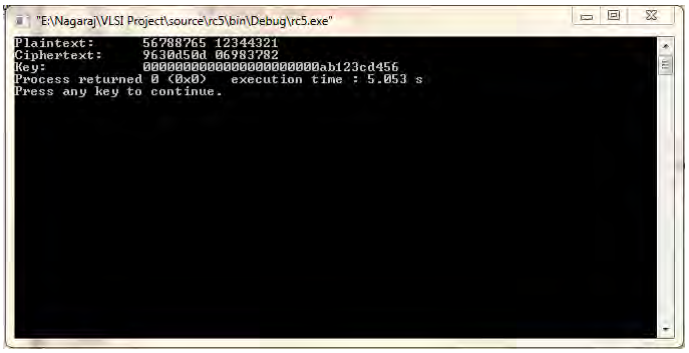

Fig. 11. Output of C implementation

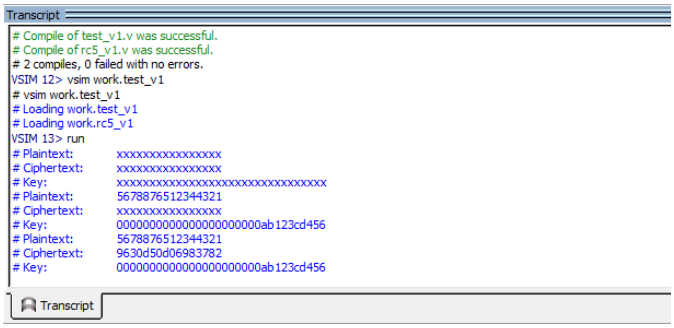

Fig. 12. Output of Verilog Simulation using ModelSIM

B. FPGA

1) DE1 Board

The following shows a snapshot of the output obtained from the DE1 board.

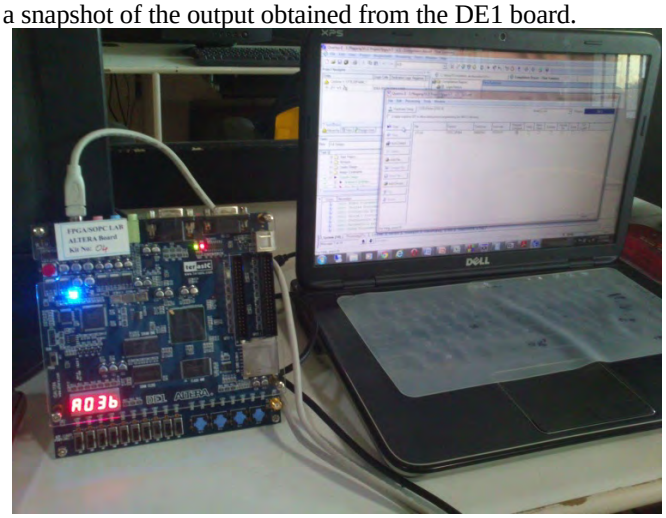

Fig. 13. DE1 Output Snapshot

2) Report for Virtex 6 Board using Xilinx ISE

TABLE I. VIRTEX6 REPORT

FPGA Board	Virtex 6
Clock Period	30.622ns
Selected Device	6vlx75tff484-3
Slice Logic: Number of Slice Registers Number used as Flip Flops Number used as Latches Number used as Latch-thrus Number used as AND/OR logics Number of Slice LUTs Number used as logic Number using O6 output only Number using O5 output only Number using O5 and O6 Number used as ROM Number used as Memory Number used exclusively as route-thrus Number with same-slice register load Number with same-slice carry load Number with other load	Utilization 1,665 out of 93,120(1%) 1,607 0 0 58 5,343 out of 46,560(11%) 5,336 out of 46,560(11%) 4,236 210 890 0 0 out of 16,720(0%) 7 0 7 0
Throughput	241.73 Mbps
On-Chip Power Summary: On-Chip Clocks Logic Signals IOs Leakage Total	 Power(mW) 7 5 6 0 1292 1312

C. ASIC

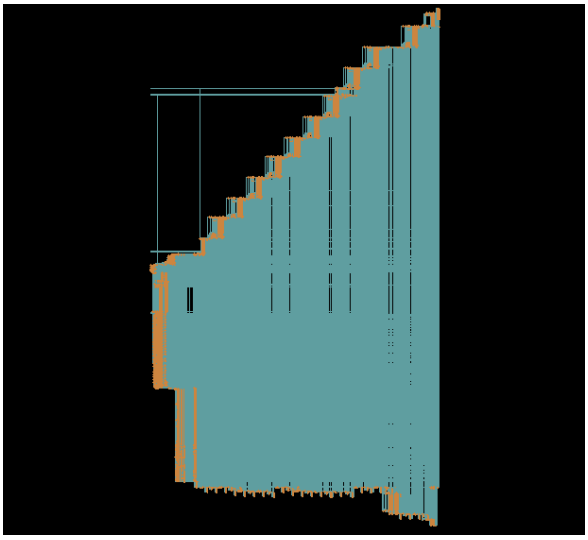

Fig. 14. Schematic

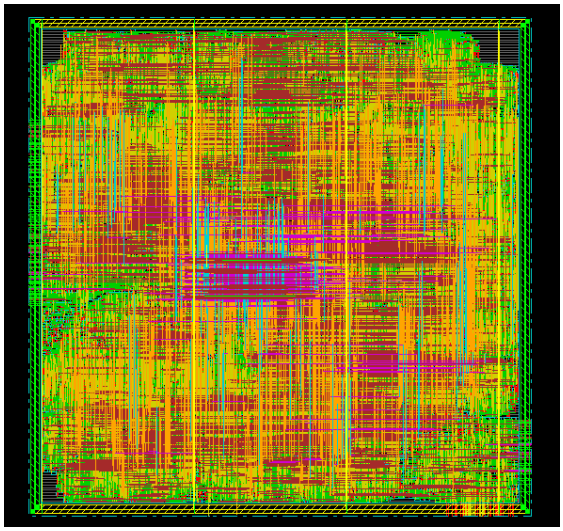

Fig. 15. Routed Layout

1) Reports

TABLE II. POWER REPORT

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
rc5_v6	32735	3250.042	16984760.553	16988010.595

TABLE III. AREA REPORT

Instance	Cells	Cell Area
rc5_v6	32735	69525

TIMING REPORT

Clock Period	12ns
Timing Slack	1ps
Throughput	5.33 Gbps

D. Results Comparison for FPGA

TABLE IV. RESULT COMPARISON

Architecture	Throughput(Mbps)	Slices Used
Elkeelany et al.	300	3618
Implemented(without pipelining)	241	5343
Implemented(with 2 stage pipelining)	560	5504

VI. CONCLUSION

In this project, we have done 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 results has been compared with the implementation by Elkelaany[2]. The throughput obtained in the current implementation was lower because of the absence of pipelining. 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, “The RC5 Encryption Algorithm”, Proceedings of the 1994 Leuven Workshop on Fast Software Encryption (Springer 1995), pages 86-96

[2] Omar Elkelaany, Adegoke Olabisi, “Performance Comparisons, Design, and Implementation of RC5 Symmetric Encryption Core using Reconfigurable Hardware”, Journal of Computers, vol. 3, no.3, pp. 48-55, Mar. 2008.

[3] N. Sklavos, C. Machas, O. Koufopavlou, "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), vol. 1, pp.172-175, United Arab Emirates, Dec. 2003.

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

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

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

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

[8] Charles H Roth, "Introduction to Logic Design", West Pub. Co., 1985.

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

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