KARNATAKA STATE COUNCIL FOR SCIENCE AND TECHNOLOGY

46S_BE_4171
IMPLEMENTATION OF AES 256 ENCRYPTION USING FPGA

COLLEGE: Rajarajeswari College of Engineering
DEPARTMENT: Electronics and Communication

NAME OF TEAM MEMBERS

Name: TRIKALESHWAR S
Email id: trikaleshwarshankar@gmail.com
Mobile No.: 9148760407

Name: SWETHA S
Email id: shwethasrinivas999@gmail.com
Mobile No: 8431695711

Name: TEJAS I TELKAR
Email id: tejastelkar2001@gmail.com
Mobile No.: 9901331770

Name: THAKSHTIH B K
Email id: thakshithblk4871@gmail.com
Mobile No.: 9844731920

NAME OF THE GUIDE

Name: Dr. Vijaya S M
Email id: vijaya@rrce.org
Contact No.: 8867590051

KEYWORDS

Advanced Encryption Standard, FPGA, Area optimisation, Pipelined architecture, 256-Bit key, Verilog, cyclic memory, multiplexing, Galois field multiplication.

INTRODUCTION

The Advanced Encryption Standard (AES) algorithm is one of the block cipher encryption algorithms that was published by National Institute of Standards and Technology (NIST) in 2000. This algorithm was developed by two professional cryptographers Joan Daemen and Vincent Rijmen. It represents a fundamental building block of many network security protocols to ensure data confidentiality in various applications ranging from data servers to low-power embedded systems. It finds applications in Mobile Phones, Smart Cards, Magnetism Cards, Intel Core Processors Family, AutomatedTeller Machines (ATM), WWW servers, SSD Devises, Ipsec and SSL Protocols, various other transmission protocols standardized by IEEE, IEEE 802.11i WPA2 standard Wi-Fi networks for secure encryption and digital video systems, etc., ensuring safety, security and reliability of data transmission. Implementation of AES algorithm can be done either in software or in hardware. But most of the practical real time applications prefer only the hardware implementation, since it is very fast, safe and highly reliable for high-speed processing as compared to software implementation.

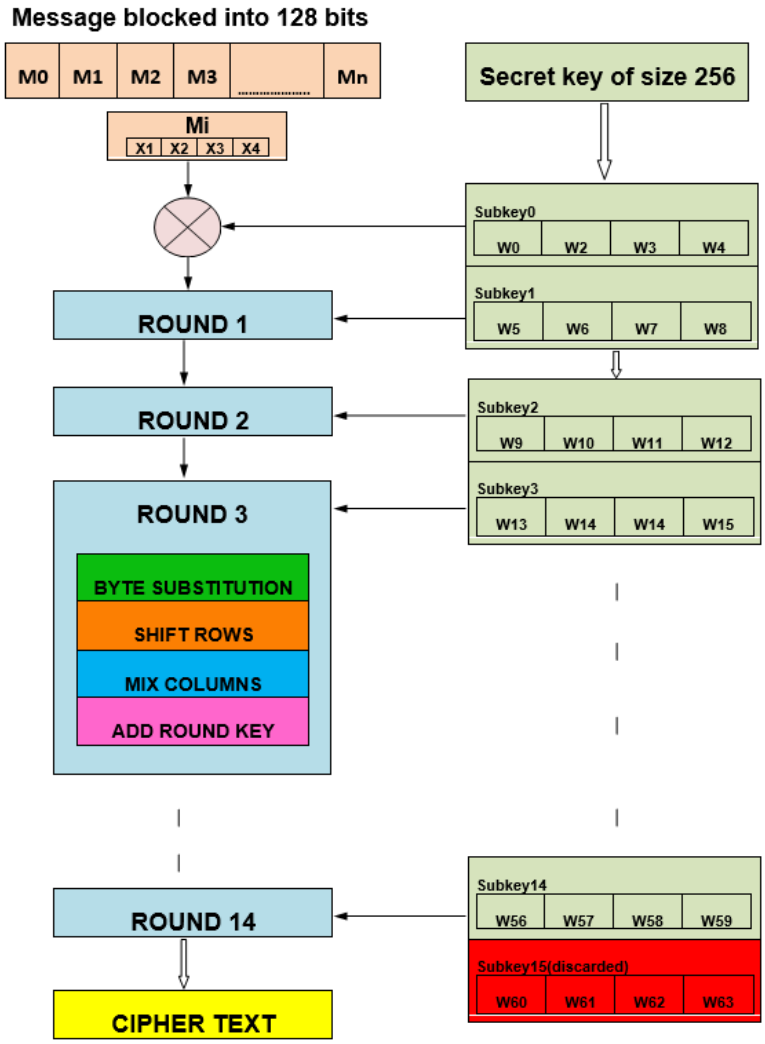
Advanced Encryption Standard (AES) algorithm is one on the most common and widely used symmetric block cipher algorithm in the world wide. This algorithm has an own particular structure to encrypt and decrypt sensitive data and is applied in hardware and software all over the world. It is extremely difficult for hackers to get the real data when encrypted by AES algorithm. Till date is not any evidence to crake this algorithm. AES has the ability to deal with three different key sizes such as AES 128, 192 and 256 bit and each of these ciphers has 128-bit block size. In this project we design hardware architecture and implement the Advanced Encryption Standard (AES) algorithm based on the Field Programmable Gate Array (FPGA) using High Level Language (HLL). This design is focused on the optimal usage of available resources; thus, it minimizes the hardware resource utilization and area parameters at best of our knowledge.

The previous work on this topic includes using of different architecture which yields better performances in terms of energy consumption or timing which relates to higher throughput or the cost of overall production of the design.

OBJECTIVE

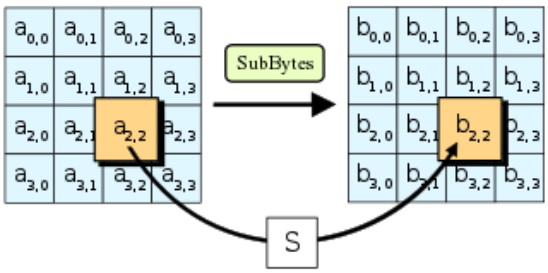
The AES-256 project has it's main objective to reduce the area consumed after the design synthesis, where the base paper referred to be around 64188 nm^2. So as a effort to reduce the same we have come up with a different approach to reduce the area but without considering other parameter such as the timing (performance), the power consumption, the throughput or even the accuracy but only the area parameter. We believe that when the major problem that is area is solved; then the tradeoff can be optimized for power and performance. All this is achieved by having a unique approach to the architecture used which eliminates the generation of hardware multiple times which does performs the same function over and over. The result of the same can be seen while implementing on the FPGA, there are flags which are designed to check whether the designed architecture encrypts the given test data successfully or not. The approach of saving is by using fewer always blocks and using case condition with defaults which reduces the generation of unwanted D-Flip Flops. By following few other techniques it's possible to reduce the area in the design code and checking the same during synthesis.

METHODOLOGY



1) Byte Substitution

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box.

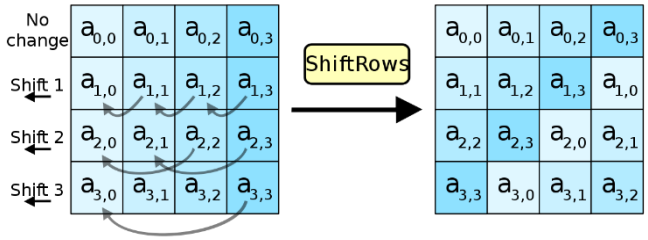


2)Shift Rows

This step is just as it sounds. Each row is shifted a particular number of times.

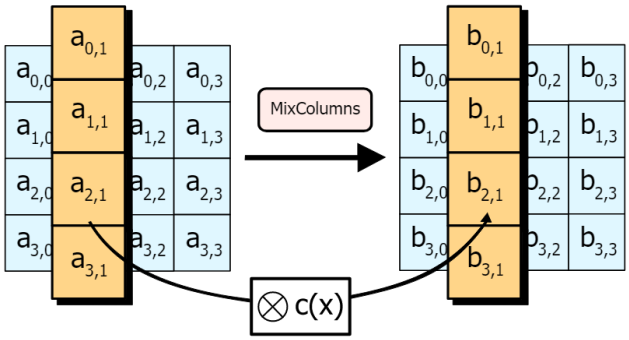
- The first row is not shifted
- The second row is shifted once to the left.

- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.



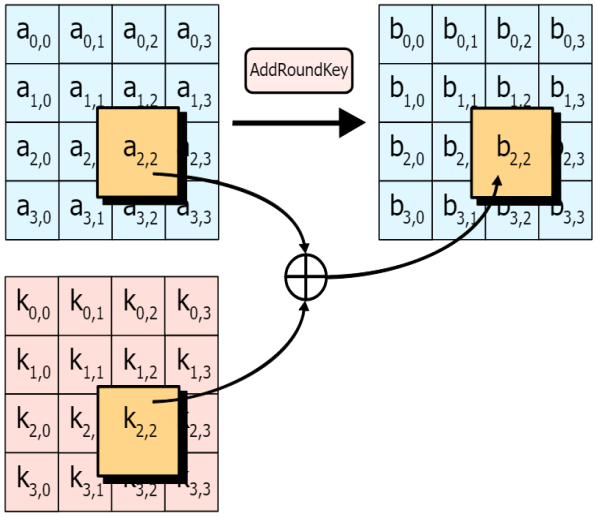
3) Mix Column

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

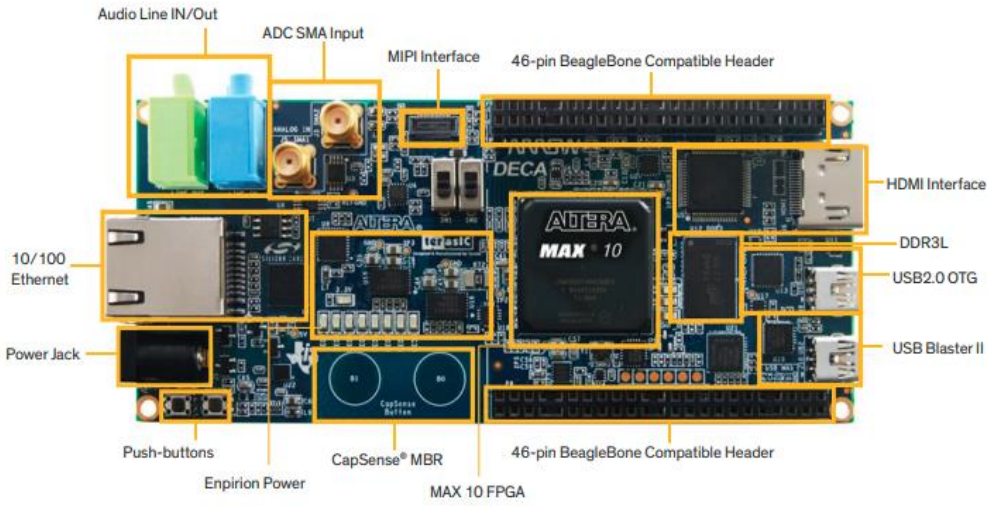


4)Add Round Key

Now the resultant output of the previous stage is XOR-ed with the corresponding round key.



HARDWARE COMPONENTS



Altera Deca Max

- 512MB RAM (16-bit data bus)
- Micro SD card socket
- 50K programmable logic elements
- 1,638 Kbits embedded memory
- 5,888 Kbits user flash memory
- Onboard USB-Blaster II (mini USB type B connector)
- 2 push-buttons
- 2 slide switches
- 8 blue user LEDs
- 5V DC input
- Note Pad ++

SOFTWARE

- Note Pad ++

  Code editor

- EDA playground, Icarus

  Open-source online tool for simulation of Verilog code.

- Quartus prime,

  Interfacing of Altera max 10.

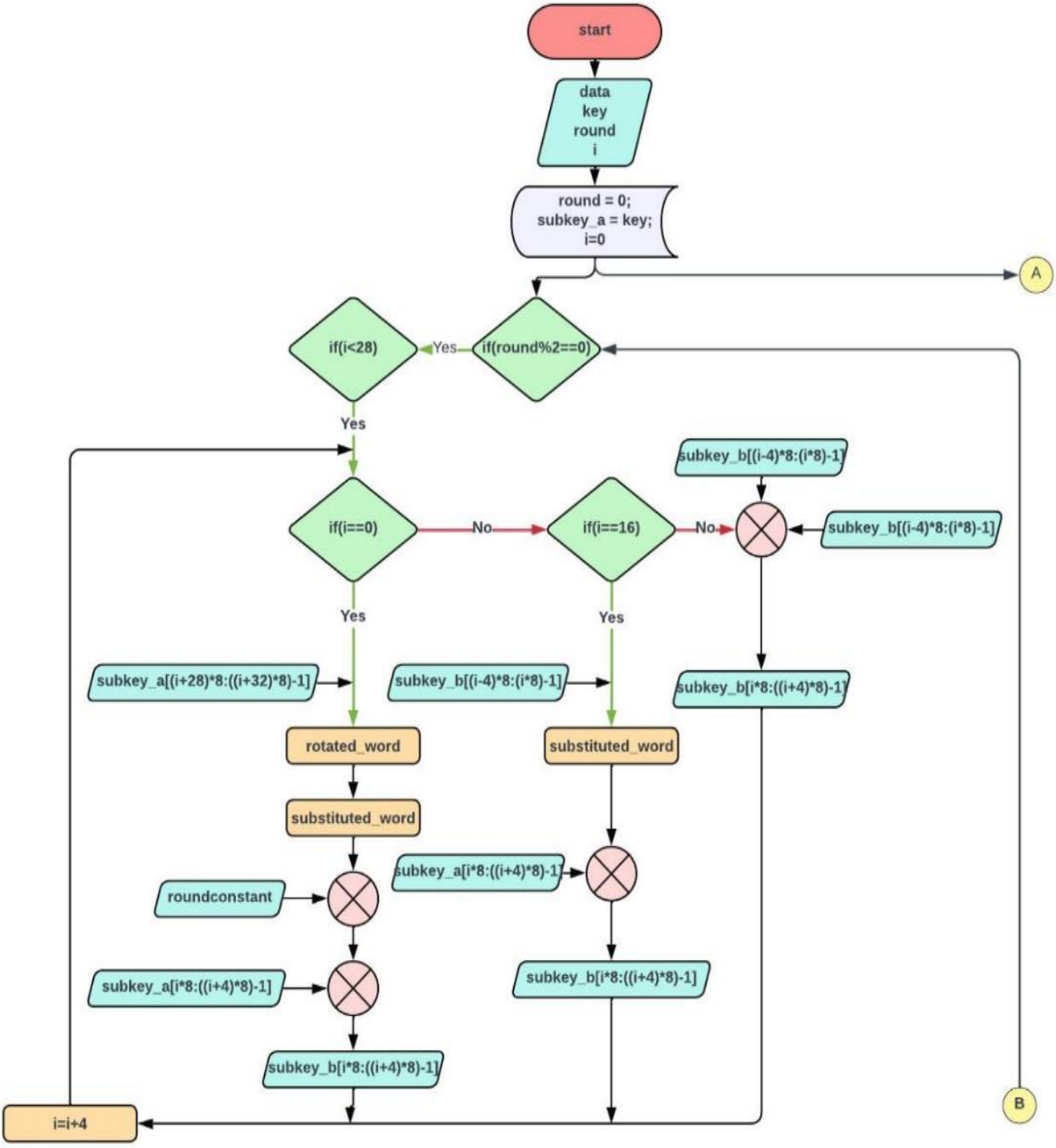
- Xilinx ISE

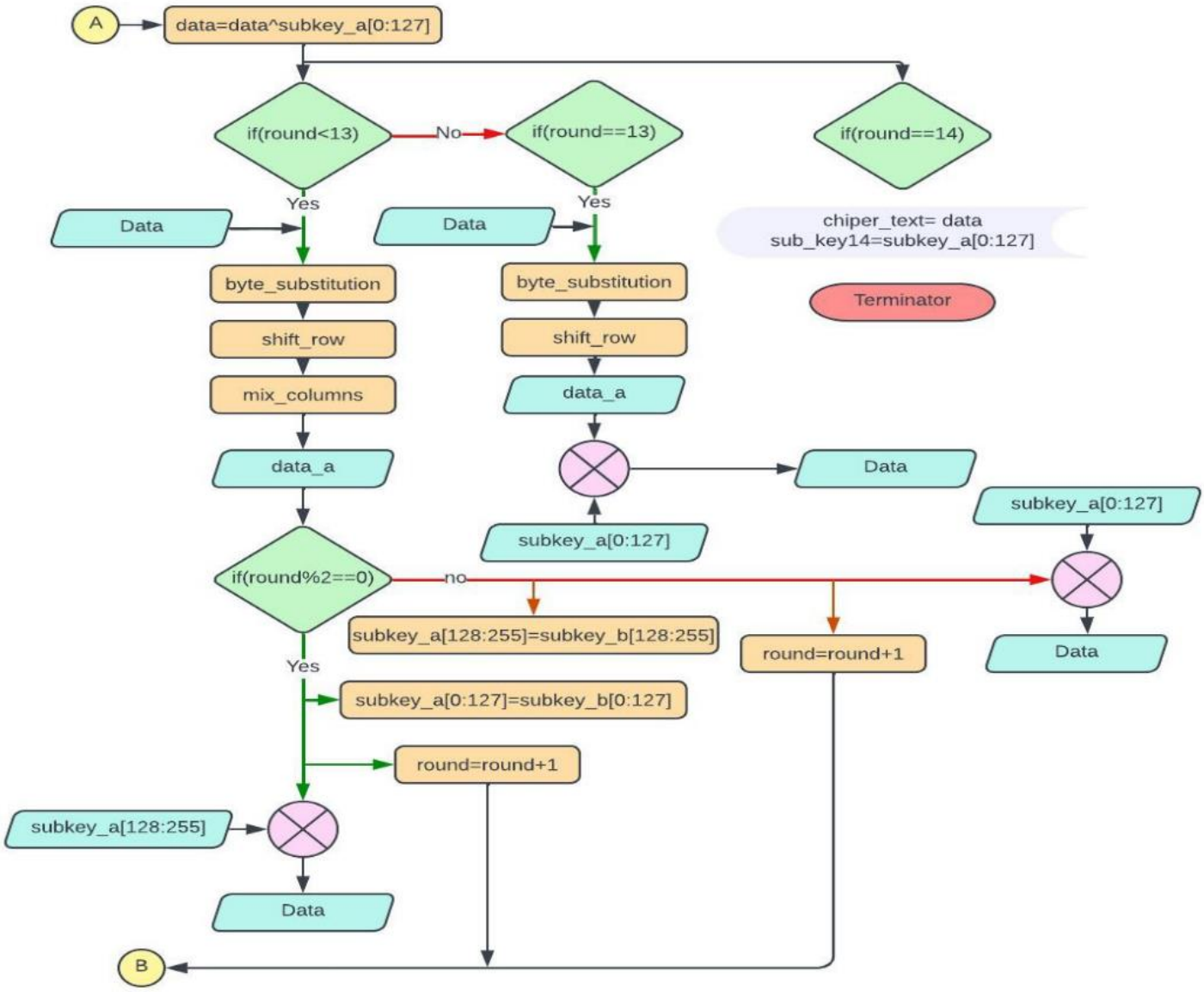
  Compilation and testing of codes.

- Cadence

  Licensed resource for simulation and reports.

ARCHITECTURE





Our architecture involves 3 techniques that's concerns to optimize area. These techniques are pipelining multiplexing and cyclic memory. Our inputs are clock reset 128 bit of data 256 bits of key, and we have set of registers such as subkey_a, subkey_b each of size 256 bits, data_a of 128 bits, roundconstant of size 32 bits. And integer such as round and i. Initially we move the key to register subkey_a and store value zero in round i and roundconstant as reset condition. We predefine the function such substituted word, rotated word ,byte_sustitution, shift row, and mix column.

In key expansion block we make give round and i in our sensitivity list as everytime there is any change in the senstitivity list it will carry out the function in that block.

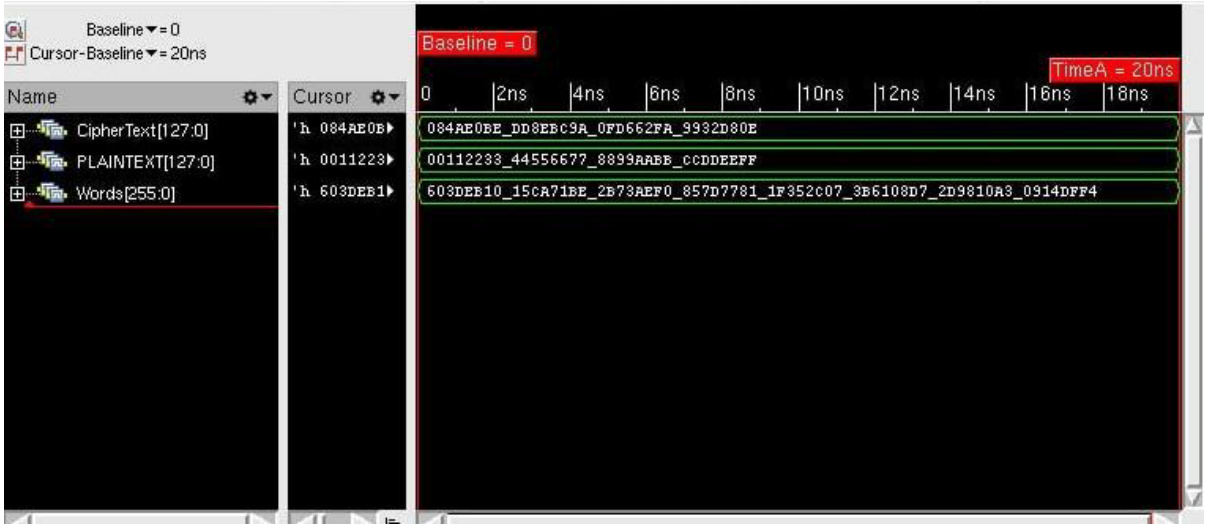
Roundconstant are defined on bases of the current round. if round is zero or two or four or six or eight or ten or twelve the process of driving subkey_b from subkey_a takes place. We of integer I to specify the address location of each word in subkey_a to map with subkey_b after undergoing operation defined in function such as substituted word, rotate word, adding round constant of that particular round, depending upon the location of each word. here we use multiplexing technique to make subkey_a to subkey b. We are reducing hardware.For example, substituted_word hardware which was required 14 time in now reduce to 2 and rotated word hardware which was required 8 time is reduced to 1'

In simple word subkey_a contains 2 sets subkeys and we deriver the next to subkeys we specified in AES alogithum. The integer I with represent the word in each. Integer I guide from which word of which subkey the next subkey must be derived with the help of multiplexer.

In the main block during round equal to zero we xor 1st 128 bits of subkey_a and Data and store it in data_a. This data_a undergoes the operation specified in function byte substitution, shift row, mix column then xor with the suitable subkey till the round is less than 13, and stored back in Data then the content of subkey_b is passed to subkey_a to maintain the further key expansions carry out along the side with cyclic memory. After completion of all this above process round is incremented by 1. When round is equal to 13 all the operation specified above is carried out expect the mix column. When round is again incremented to 14 the Data is transferred to output chipper text and 1st half of subkey_a is transferred to output sub_key14.

RESULT

DATA	00112233445566778899AABBCCDDEEFF
SECRET KEY	603DEB1015CA71BE2B73AEF0857D7781 1F352C073B6108D72D9810A30914DFF4
CIPHER DATA	146F2A291CB4798909A77836A60E3BC0 092A4AF9BA6704D751A38FE1B60F30DA



The above image shows that the result has been a success since all the 5 flags which were used to check with the test data given was found true. This signifies that the algorithm has worked in the right way as planned in the architecture. For additional verification the encrypted data was decrypted on an online platform which yielded the data that was given as the input. This way the re-verification of the algorithm was performed.

CONCLUSION

- AES 256 is considered one of the most secure encryption techniques currently available due to its key size and multiple rounds of encryption.
- Throughout this project, we focused on reducing the area parameter while implementing AES 256 on an FPGA.
- We were able to achieve this by developing our own architecture that streamlined the design and optimized it and the estimated area consumed by our project as per the results from the synthesis of Cadence EDA it was projected as **53802** with this, we can benchmark with the paper we have referred and the later has obtained **64188** which is a great reduction in the overall area used up in the design.
- However, the architecture is well designed to address the problem statement but still this design needs a lot of analysis on the timing constraints and the test for any possible glitches.
- There are still more possibilities to explore in area optimization or the other parameters such as performance and power which will be a trade-off in the area or any one of the other parameters if altered.
- Overall, this project has demonstrated the importance of encryption techniques such as AES 256 in ensuring data security and privacy in today's interconnected world.

INNOVATION

There are ongoing research and developments in the field of cryptographic algorithms and techniques that can enhance the overall security and performance of AES-256. Some potential areas of innovation and improvement in AES-256 can include:

1. **Side-Channel Attack Resistance**: Side-channel attacks exploit information leaked during the execution of an algorithm, such as power consumption or electromagnetic radiation. Innovations can focus on developing countermeasures to mitigate side-channel attacks and enhance the resistance of AES-256 against such attacks.
2. **Hardware Acceleration**: Innovations can be made in hardware implementations of AES-256, such as optimizing the design for high-speed encryption/decryption or developing specialized hardware architectures that provide better performance and efficiency.
3. **Quantum Resistance**: With the advent of quantum computers, there is a growing need for cryptographic algorithms that are resistant to quantum attacks. Innovations can involve developing post-quantum variants of AES-256 or combining it with other quantum-resistant algorithms to ensure long-term security.
4. **Authenticated Encryption**: AES-256 can be combined with authenticated encryption modes, such as AES-GCM (Galois/Counter Mode), to provide both confidentiality and integrity of the encrypted data. Innovations can focus on improving the efficiency and security of AES-256 in authenticated encryption scenarios.
5. **Secure Key Management**: Innovations can be made in key management schemes, key generation techniques, and key distribution protocols to enhance the overall security of AES-256. This includes secure key storage, key rotation, and proper key usage policies.

FUTURE SCOPE

1. **Post-Quantum Cryptography**: With the rise of quantum computing, there is a need for cryptographic algorithms that are resistant to quantum attacks. Future developments may focus on exploring post-quantum variants of AES-256 or completely new encryption algorithms that provide security against quantum adversaries.
2. **Efficiency and Performance Optimization**: Innovations can be made to improve the efficiency and performance of AES-256 implementations. This includes optimizing hardware designs, exploring new software algorithms, and leveraging parallel processing techniques to enhance the speed of encryption and decryption operations.
3. **Hardware Security**: As hardware-based attacks and vulnerabilities continue to be a concern, future research can focus on developing hardware security mechanisms to protect AES-256 implementations from side-channel attacks, fault injections, and tampering.
4. **Secure Implementation Guidelines**: Future work can involve establishing comprehensive guidelines and best practices for securely implementing AES-256 in various systems and platforms. This includes recommendations for key management, secure coding practices, and secure integration of AES-256 into larger cryptographic systems.
5. **Integration with Emerging Technologies**: AES-256 can be integrated with emerging technologies such as blockchain, Internet of Things (IoT), and cloud computing. Future research may explore how AES-256 can be effectively utilized and optimized in these contexts to ensure secure data storage, communication, and processing.
6. **Standardization and Certifications**: Ongoing efforts can focus on the standardization and certification of AES-256 implementations, ensuring interoperability, compatibility, and compliance with industry and government security standards.