AES ENCRYPTION OF IMAGE ON FPGA

Sreyas Janamanchi (IMT2022554)

Pradyumna G (IMT2022555)

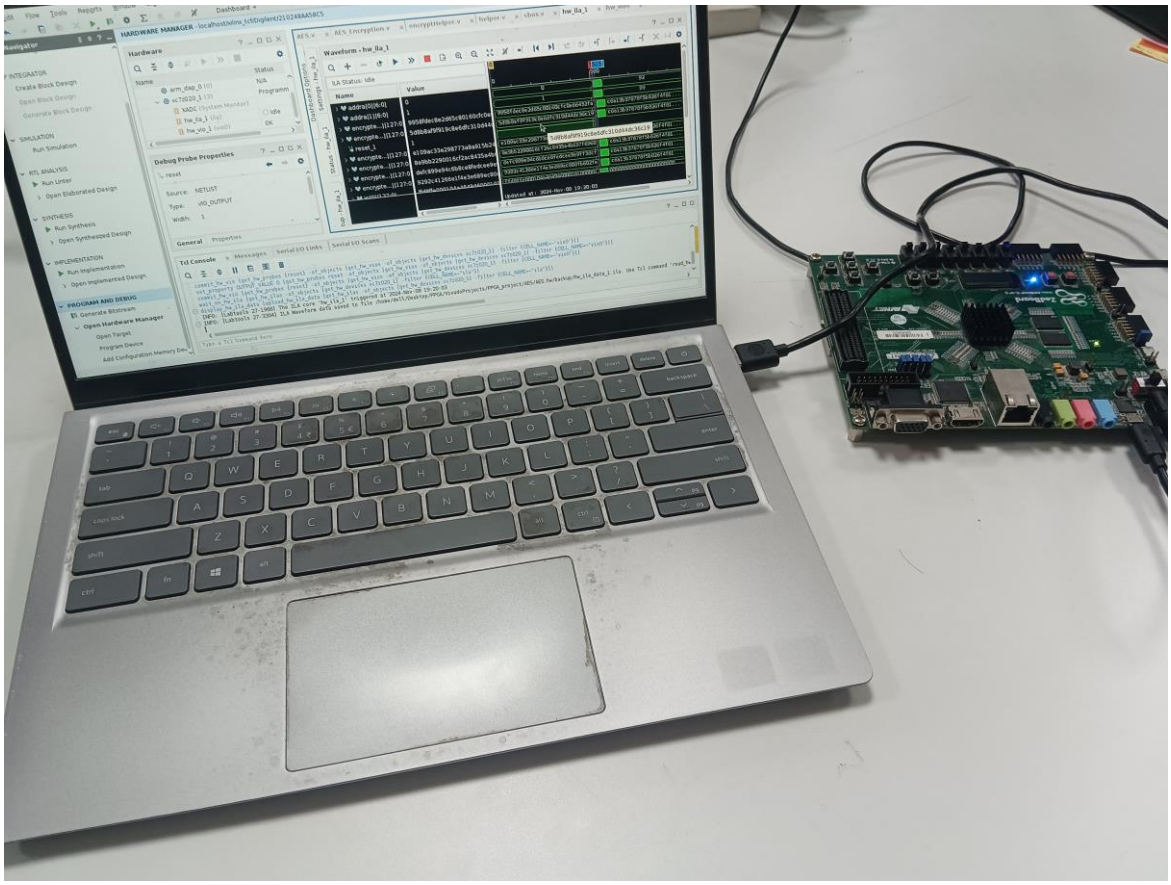
Chinmay Krishna R (IMT2022561)

BACKGROUND

The Advanced Encryption Standard (AES) is a trusted encryption algorithm widely used for securing sensitive data, offering key lengths of 128, 192, or 256 bits to prevent unauthorized access. In this project, we implement AES-128 encryption, a method extensively used to protect internet communications and sensitive files.

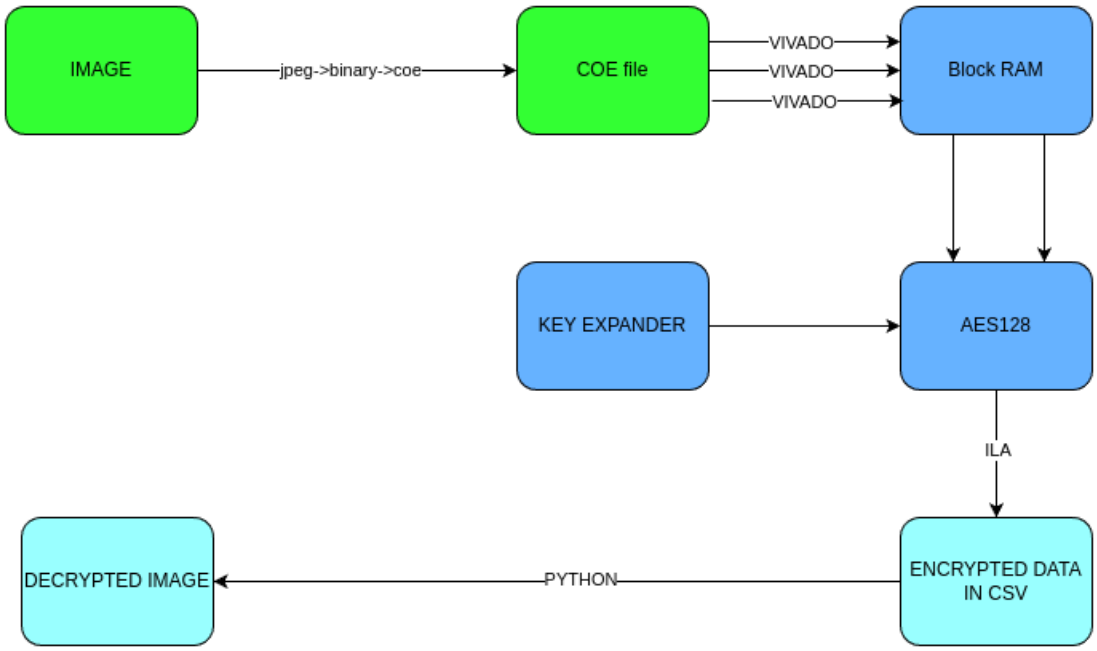
Implementing AES encryption on an FPGA enables secure, real-time processing of sensitive visual data, leveraging parallel processing for low latency and efficient resource utilization. This approach is ideal for time-sensitive applications such as surveillance and medical imaging, ensuring robust data security and adaptability in embedded systems.

METHODOLOGY



SETUP

In this project we perform acceleration of AES-128 encryption on an image using FPGA.
Here's the link to our code: https://github.com/Sreyas-J/ImageAES_FPGA
Here's the demo link: Demo.mp4
Reference: https://github.com/michaelehab/AES-Verilog, this gave us inspiration for a basic non-accelerated AES-128 encryption. The code implements composite arithmetic algorithm for optimization.



- The process begins with converting an image into a .coe file format. The .coe file is a series of 128-bit plaintexts.
- A round-robin algorithm was implemented in python to split the image data into three .coe files. Round-robin was chosen to make the project easily extendable to real-time image transfer.
- Each .coe file is used to create a block RAM (BRAM).
- We have instantiated six AES-128 modules. Each BRAM then provides input to two AES-128 modules using another round-robin to distribute data.
- The data is exported from ILA to a csv file.
- The csv is processed through Python code to reassemble the data in the correct order, accounting for a twelve-clock cycle delay. Multiple adjustments were required to synchronize the round-robin approach with the clock delay for accurate data retrieval.

To accelerate AES-128 encryption, the following optimization techniques were implemented: -

- **Parallel Computation**: Multiple AES encryption instances were instantiated, enabling parallel processing and significantly increasing throughput.

- **Single Key Expansion**: Key expansion is executed only once in the main module, and the expanded key is shared across all instances, eliminating redundant calculations.
- **Optimized MixColumn Operation**: Composite arithmetic using Galois fields was applied to optimize the MixColumn step, a core operation that is repeatedly performed during encryption.
- **Increased Clock Frequency**: The clock frequency was maximized to further boost the output rate, enhancing the overall speed of encryption.

REPORTS

BASIC IMPLEMENTATION

Clock frequency: 30.303MHz

Latency= 12 clock cycles

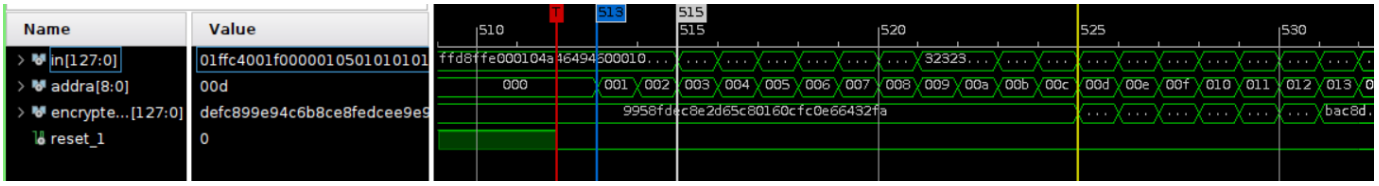
Latency: 396ns

Throughput= Clock frequency

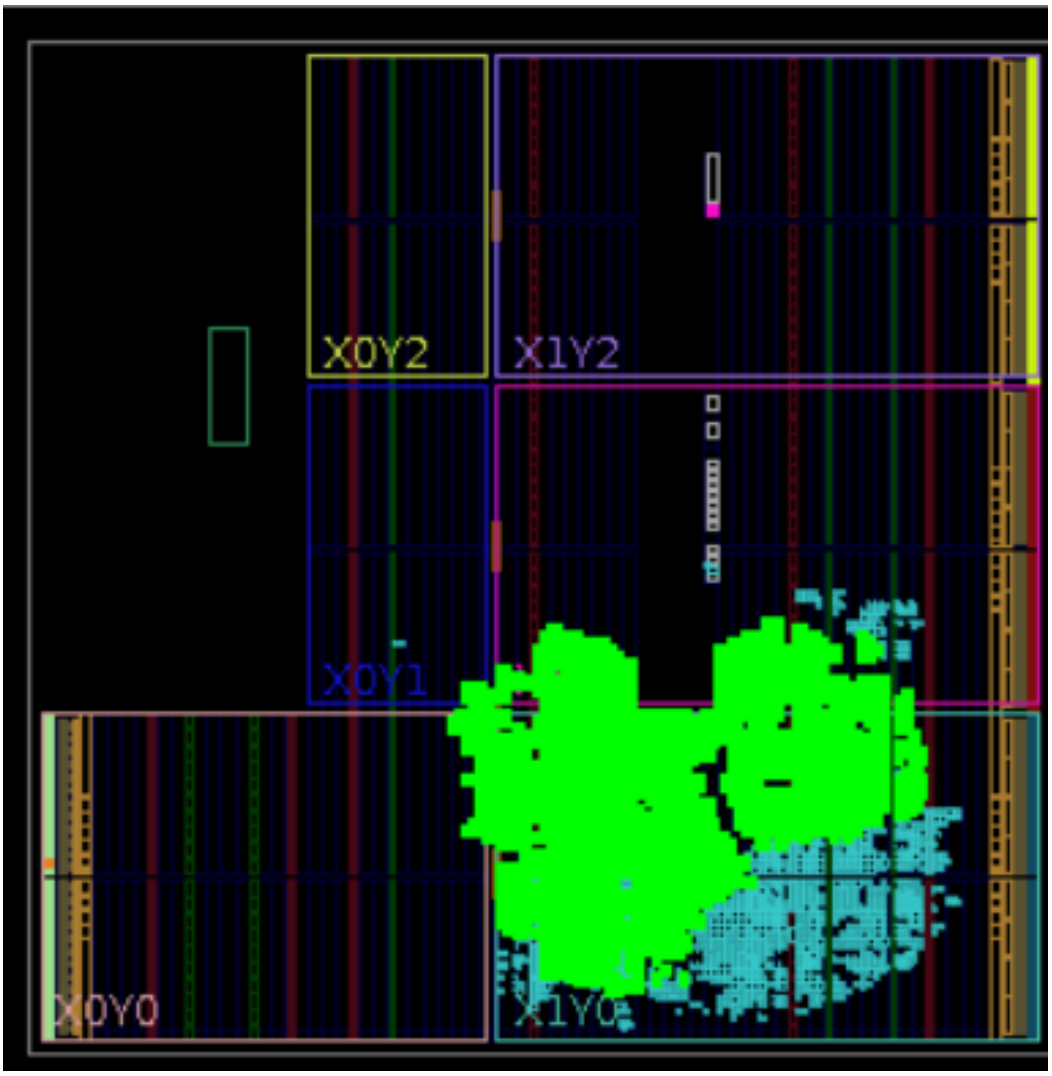
Throughput: $30.303*10^6$ ops/sec

Time taken for encrypting a 128px*128px jpeg image: 9504ns

Name	Waveform	Period (ns)	Frequency (MHz)
dbg_hub/inst/BSCANID.u_xsdbm_id/SWITCH_N	{0.000 16.500}	33.000	30.303



WE CAN SEE HERE THAT IT HAS A 12-CLOCK CYCLE DELAY FROM BRAM ADDRESS UPDATE TO ENCRYPTED VALUE OUTPUT (2 CLOCK CYCLES TO READ THE VALUE AND 10 CLOCK CYCLES FOR AES ENCRYPTION)



IMPLEMENTED LAYOUT

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	Bonded IOB (200)	BUFGCTRL (32)	BSCANE2 (4)
N AES	8232	4509	2590	1280	2515	7875	357	9.5	1	2	1
a (AES_Encrypt)	6460	1280	2560	1280	1718	6460	0	0	0	0	0
BRAM (blk_mem_gen_0)	0	0	0	0	0	0	0	2	0	0	0
dbg_hub (dbg_hub)	472	753	0	0	242	448	24	0	0	1	1
ila0 (ila_0)	1190	2236	30	0	606	857	333	7.5	0	0	0
vio0 (vio_0)	101	231	0	0	68	101	0	0	0	0	0

RESOURCE UTILISATION

ACCELERATED IMPLEMENTATION

Clock frequency: 119.403MHz

Worst negative slack: 0.795ns

Latency= 12 clock cycles

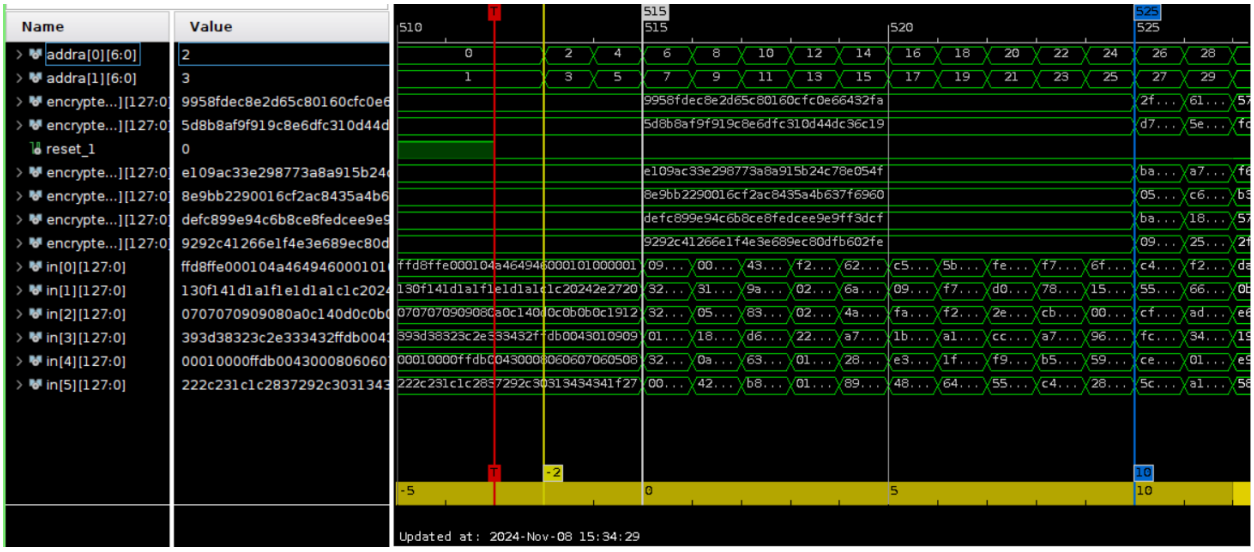
Latency: 100.5ns

Throughput= Clock_frequency*6

Throughput: $716.418*10^6$ ops/sec

Time taken for encrypting a 128px*128px jpeg image: 485.75ns

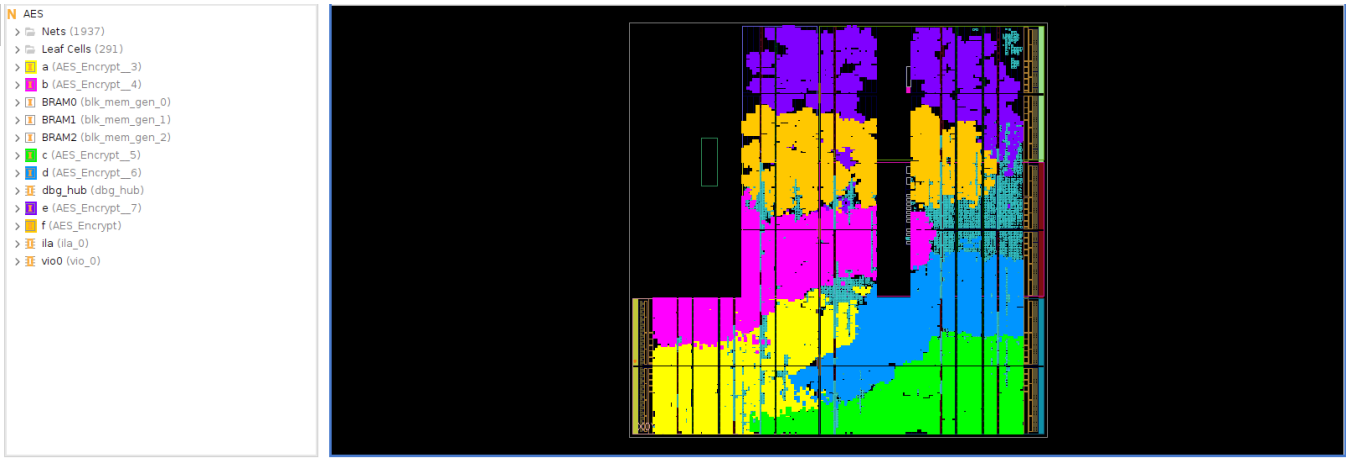
Name	Waveform	Period (ns)	Frequency (MHz)
clk	{0.000 4.188}	8.375	119.403



WE CAN SEE HERE THAT IT HAS A 12-CLOCK CYCLE DELAY FROM BRAM ADDRESS UPDATE TO ENCRYPTED VALUE OUTPUT (2 CLOCK CYCLES TO READ THE VALUE AND 10 CLOCK CYCLES FOR AES ENCRYPTION)

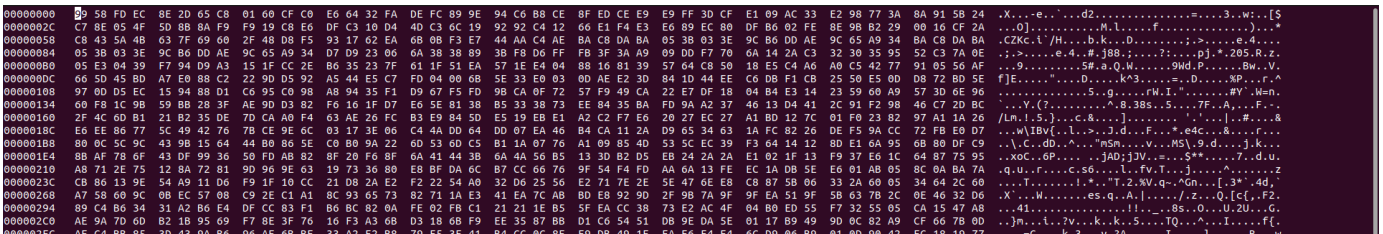
Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	Block RAM Tile (140)	Bonded IOB (200)	BUFGCTRL (32)	BSCANE2 (4)
N AES	45052	17742	15604	7762	12041	43131	1921	63.5	1	2	1
a (AES_Encrypt__3)	6641	1280	2560	1280	1727	6641	0	0	0	0	0
b (AES_Encrypt__4)	6647	1280	2560	1280	1728	6647	0	0	0	0	0
BRAM0 (blk_mem_gen_0)	0	0	0	0	0	0	0	4	0	0	0
BRAM1 (blk_mem_gen_1)	0	0	0	0	0	0	0	4	0	0	0
BRAM2 (blk_mem_gen_2)	0	0	0	0	0	0	0	4	0	0	0
c (AES_Encrypt__5)	6643	1280	2560	1280	1733	6643	0	0	0	0	0
d (AES_Encrypt__6)	6641	1280	2560	1280	1733	6641	0	0	0	0	0
dbg_hub (dbg_hub)	474	753	0	0	225	450	24	0	0	1	1
e (AES_Encrypt__7)	6640	1280	2560	1280	1819	6640	0	0	0	0	0
f (AES_Encrypt)	6643	1280	2560	1280	1765	6643	0	0	0	0	0
ila (ila_0)	4482	9063	244	82	2837	2585	1897	51.5	0	0	0
vio0 (vio_0)	101	231	0	0	70	101	0	0	0	0	0

RESOURCE UTILISATION



IMPLEMENTED LAYOUT

RESULTS



ENCRYPTED IMAGE AS A BINARY FILE (VIEWED USING HEXEDIT)

DECRYPTED IMAGE AS A BINARY FILE (VIEWED USING HEXEDIT)

INPUT IMAGE

DECRYPTED IMAGE

FUTURE ASPECTS

- Further acceleration can be achieved through pipelining. The encryption pipeline consists of nine instances, each handling rounds 2 through 10. When a plaintext is being processed in a specific round, the remaining instances are free to begin encrypting additional plaintexts, maximizing throughput and efficiency.
- The image can be dynamically transmitted from a microcontroller or processing system (PS) via a communication protocol such as UART or AXI. The encrypted image is then sent back to the microcontroller/PS, enabling automation and easing implementation for embedded systems.
- Further security can be improved by accelerating authentication algorithms like RSA and AES-CMAC on FPGA to verify data authenticity and integrity, in addition to confidentiality provided by AES encryption.