ADVANCE ENCRYPTION STANDARD

ABSTRACT:

This project details the hardware implementation of the Advanced Encryption Standard (AES) algorithm. We rigorously define the four core transformations: SubBytes (using GF(2^8) multiplicative inverse and affine transformation), ShiftRows, MixColumns (involving polynomial multiplication modulo (x^4+1), and AddRoundKey. The project further includes the Key Expansion process. The objective is to realize a 128-bit block size and 128-bit key length AES core, providing a foundational step towards a high-throughput, low-latency cryptographic hardware design.

Step-by-Step Implementation Outline

1. Core Arithmetic Module (The Foundation)

This module is the most complex, handling the finite field arithmetic crucial for SubBytes and MixColumns.

Module	Purpose	Key Operations
GF256_Math	Implements fundamental GF(2^8)math.	**Multiplication** (polynomial multiplication modulo (x^8+x^4+x^3+x+1).
SBox	Implements the SubBytes transformation.	1. Find the **Multiplicative Inverse** of an 8-bit input in GF(2^8). 2. Apply the **Affine Transformation** (matrix multiplication and XOR with a constant vector).

2. Round Transformation Modules

These modules perform the transformations on the 128-bit (4x4 byte) State Array.

Module	Transformation	Input/Output	Notes
ShiftRows	Cyclic shift of rows 1, 2, and 3.	128-bit state in/out.	A simple wiring/re-arrangement of bytes.
MixColumns	Matrix multiplication in GF(2^8) with the fixed matrix.	32-bit column in/out.	This will utilize the GF256_Math module repeatedly. For speed, consider parallelizing the four column operations.
AddRoundKey	State $\oplus$ Round Key.	128-bit state in/out, 128-bit Round Key in.	A simple 128-bit XOR operation.

3. Key Expansion Module

This is typically implemented as a combinatorial/sequential block to generate all 11 Round Keys (for AES-128).

Module	Purpose	Key Operations
KeyExpansion	Generates 11 Round Keys from the 128-bit Master Key.	**g-function** implementation (RotWord, SubWord, XOR with Rcon), followed by **XOR chain** to generate the rest of the key schedule words.

4. Top-Level Module (`AES_128`)

This module orchestrates the entire encryption process.

- **Initialization:** AddRoundKey (using the initial Master Key).
- **Rounds 1 to 9:** Loop through **SubBytes, ShiftRows, MixColumns, AddRoundKey**.
    - It will need a Round Counter and a register to hold the current State.
- **Final Round (Round 10): SubBytes, ShiftRows, AddRoundKey** (no MixColumns).
- **Data Flow:** Connect the State Register to the input of each transformation module and feed the output back into the State Register. Use the Key Expansion output as the input for the AddRoundKey module in each round.