AES-128 Specification

Introduction

The `aes_128` module implements the AES (Advanced Encryption Standard) algorithm with a 128-bit key size. AES is a symmetric key encryption standard widely used across the globe for securing data. This module adheres to the AES-128 specification, performing encryption through a series of transformations on the input data (`state`) using a given key. The design is structured to process data in a pipelined manner, ensuring efficient encryption operations.

Architecture

The architecture of the `aes_128` module is composed of several sub-modules that implement the key expansion and the encryption rounds of the AES algorithm. The top-level module, `aes_128`, orchestrates the encryption process by managing the flow of data through the following components:

- **Key Expansion**: Implemented by the `expand_key_128` module, which generates round keys from the initial key.
- **Rounds**: Consists of nine `one_round` modules and a `final_round` module. Each `one_round` module performs a single AES round transformation, while the `final_round` module completes the encryption with a slightly different transformation.

The design operates synchronously with a single clock domain, ensuring that all operations are aligned with the clock signal.

Interface

aes_128 Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
state	128	In	Input data block to be encrypted
key	128	In	Encryption key
out	128	Out	Encrypted output data block

expand_key_128 Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
in	128	In	Input key for expansion
rcon	8	In	Round constant for key expansion
out_1	128	Out	Expanded key output (registered)

Signal	Width	In/Out	Description
out_2	128	Out	Expanded key output

one_round Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
state_in	128	In	Input state for the round transformation
key	128	In	Round key for the transformation
state_out	128	Out	Output state after the round transformation

final_round Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
state_in	128	In	Input state for the final round transformation
key_in	128	In	Final round key
state_out	128	Out	Output state after the final round transformation

table_lookup Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
state	32	In	Input state word for table lookup
p0	32	Out	Output word after table lookup transformation
p1	32	Out	Output word after table lookup transformation

Signal	Width	In/Out	Description
p2	32	Out	Output word after table lookup transformation
p3	32	Out	Output word after table lookup transformation

S4 Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
in	32	In	Input word for substitution
out	32	Out	Output word after substitution

T Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
in	8	In	Input byte for transformation
out	32	Out	Output word after transformation

S Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
in	8	In	Input byte for S-box substitution
out	8	Out	Output byte after S-box substitution

xS Module

Signal	Width	In/Out	Description
clk	1	In	Clock signal for synchronization
in	8	In	Input byte for xS transformation
out	8	Out	Output byte after xS transformation

Timing

The AES-128 module operates synchronously with the clock signal. Each round transformation, including the final round, is completed in one clock cycle. The key expansion process also aligns with the clock, ensuring that expanded keys are available for each round as needed. The overall latency from input to output is determined by the number of rounds, with each round taking one clock cycle.

Usage

To use the `aes_128` module, follow these steps:

1. Provide the 128-bit input data (`state`) and the 128-bit encryption key (`key`) to the module inputs.
2. Ensure the clock signal (`clk`) is active and stable.
3. After 10 clock cycles, the encrypted output (`out`) will be available at the module's output port.
4. The module does not require any reset signal, as it operates continuously with the provided clock.

This module is designed for integration into larger systems requiring AES encryption, providing a straightforward interface for secure data processing.

Functional Description (Generated by funcgen)

Verilog Design Modules Description

Module: aes_128

File: aes_128.v

Purpose

The `aes_128` module implements the AES-128 encryption algorithm, which is a symmetric key encryption standard. This module orchestrates the key expansion, encryption rounds, and final output generation using 10 rounds specific to AES-128.

Ports

- **clk**: (input, 1-bit) - Clock signal for synchronous design.
- **state**: (input, 127:0-bit) - Initial plaintext input state to be encrypted.
- **key**: (input, 127:0-bit) - The cipher key used for encryption.
- **out**: (output, 127:0-bit) - The encrypted output after AES processing.

Internal Signals

- **s0, k0**: (reg, 127:0-bit) - Registers to hold the initial state and key post XOR operation.
- **s1 to s9, k1 to k9, k0b to k9b**: (wire, 127:0-bit) - Intermediary state and key signals used across rounds in AES encryption workflow.

Functionality

- **Sequential Logic**:

  - On each clock edge, initialize the state (`s0`) with an XOR operation between `state` and `key`. This also initializes `k0` with `key` only.
- **Combinational Logic**:
  - The module continuously propagates keys and states through `expand_key_128` and `one_round` modules.
- **Instantiations**:
  - `expand_key_128` (Instances: a1 to a10): These instances expand the AES key for subsequent rounds using key scheduling, incrementally generating each round key.
  - `one_round` (Instances: r1 to r9): Each `one_round` instance performs one of the first 9 AES rounds, transforming the state using the corresponding round key.
  - `final_round` (Instance: rf): Handles the last round of AES encryption, which differs slightly from the previous 9 rounds as it omits the MixColumns step.

Module: expand_key_128

File: aes_128.v

Purpose

Generates the subsequent round keys required for AES encryption from an initial key input.

Ports

- **clk**: (input, 1-bit) - Clock signal for synchronous behavior.
- **in**: (input, 127:0-bit) - Input key for the round when executed.
- **rcon**: (input, 7:0-bit) - Round constant used in key expansion.
- **out_1**: (output, 127:0-bit) - Expanded key output for immediate usage in rounds.
- **out_2**: (output, 127:0-bit) - Additional key output, same as `out_1`, demonstrating architecture redundancy or pipelining.

Internal Signals

- **k0 to k3, v0 to v3, k0a to k3a, k0b to k3b**: Various intermediary variables for key generation stages, leveraging bit shifts and S-box transformations.

Functionality

- **Sequential Logic**:
  - On clock's positive edge, capture intermediate values for key expansion facilitated by S-box transformations.
- **Combinational Logic**:

  - Initial key values (`k0` etc.) are transformed through bitwise and S-box operations to result in expanded keys.

Instantiations

- **S4**: Used to apply an S-box to transform parts of the key being expanded.

Module: one_round

File: round.v

Purpose

Implements a single round of the AES encryption excluding the last round.

Ports

- **clk**: (input, 1-bit) - Clock signal for timing coordination.
- **state_in**: (input, 127:0-bit) - State of AES data before this round.
- **key**: (input, 127:0-bit) - Subkey for the current round.
- **state_out**: (output, 127:0-bit) - Transformed state after this round.

Internal Signals

- **s0 to s3, z0 to z3**: (wire, 31:0-bit) - Intermediate representations of state transformation.
- **p00, p01, ... , p33**: Intermediate table lookup results to apply transformations per AES specifics.

Functionality

- **Sequential Logic**:
  - The result of state transformation (`z0` to `z3`) is latched every clock cycle to `state_out`.
- **Combinational Logic**:
  - State is transformed using extensive XOR operations and lookup tables provided by `table_lookup`.

Instantiations

- **table_lookup**: Used four times to transform the sub-state words into outputs for further processing.

Module: final_round

File: round.v

Purpose

Executes the last round of AES encryption, which does not include the MixColumns stage.

Ports

- **clk**: (input, 1-bit) - Clock signal.
- **state_in**: (input, 127:0-bit) - Input state for the final round.
- **key_in**: (input, 127:0-bit) - Key used for the final round of encryption.
- **state_out**: (output, 127:0-bit) - Final encrypted output post-AES operation.

Internal Signals

- Uses the same nomenclature for signal and transformation as in `one_round`, implying a similar role but for the ultimate round.

Functionality

- **Sequential Logic**:
  - Final transformation results are registered in `state_out`.
- **Instantiations**
- **S4**: Handles the final transformation of four words of the AES state input through S-box substitution.

Module: table_lookup

File: table.v

Purpose

Transforms 32-bit state words into substituted values based on AES S-box concepts.

Ports

- **clk**: (input, 1-bit) - Synchronization signal.
- **state**: (input, 31:0-bit) - Input word from state.
- **p0 to p3**: (output, 31:0-bit) - Sub-outputs for the transformed word after applying S box and digging deeper with further manipulations.

Functionality

- **Combinational Logic**:
  - Decouples state into bytes and applies transformations using S-box-based `T` transformation for advanced operations.

Instantiations

- **T**: Used to provide specific functions and permutations for bytes as per AES needs.

Inter-Module Connections

The design is structured hierarchically, starting from the `aes_128`, which serves as the top-level module. `aes_128` coordinates among key expansion (`expand_key_128`) and iterative encryption stages (`one_round`, `final_round`).

1. **aes_128**:
   - **Key Expansion**: Uses `expand_key_128` instances to prepare keys for all rounds.
   - **Encryption Rounds**: Utilizes `one_round` for the first nine processing stages involving state and key transformations.
   - **Final Round**: Executes through a `final_round` instance, differing from regular rounds by omitting mix columns.
2. **Subcomponents**:
   - **one_round & final_round**: These modules further deploy `table_lookup` and `S4` modules to apply AES-specific transformations, including the byte substitution and mixing operations via S-boxes.

Data flows through initialized states and keys, transformed in rounds, eventually culminating in the final AES output cipher.

This layered structure not only adheres to the AES standard but also optimizes the hardware implementation through efficient modularity, ensuring ease in debugging and enhancement.