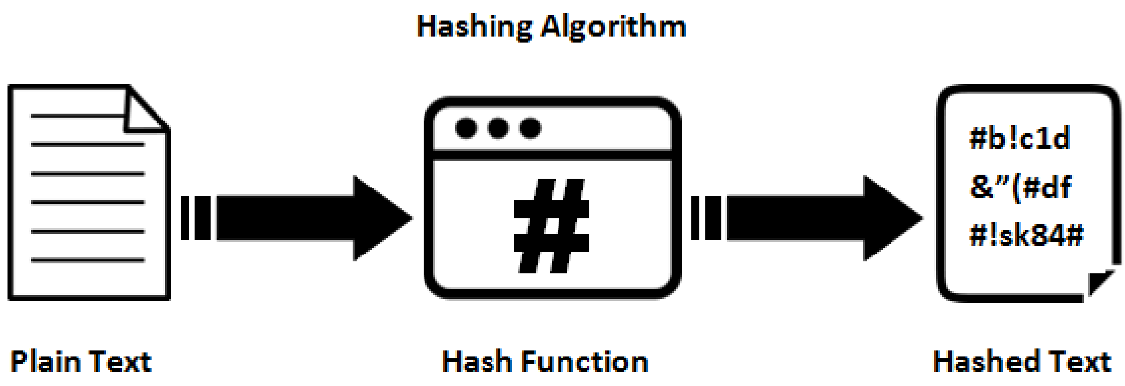
SHA-256 Implementation in Verilog

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that converts an input (a message of any length) into a fixed-size 256-bit hash value. The Verilog implementation of SHA-256 involves a series of logical and arithmetic operations performed on data blocks. Here's a brief overview of the key components and their functions, which you can use for your documentation.

What is a Cryptographic Hash Function?

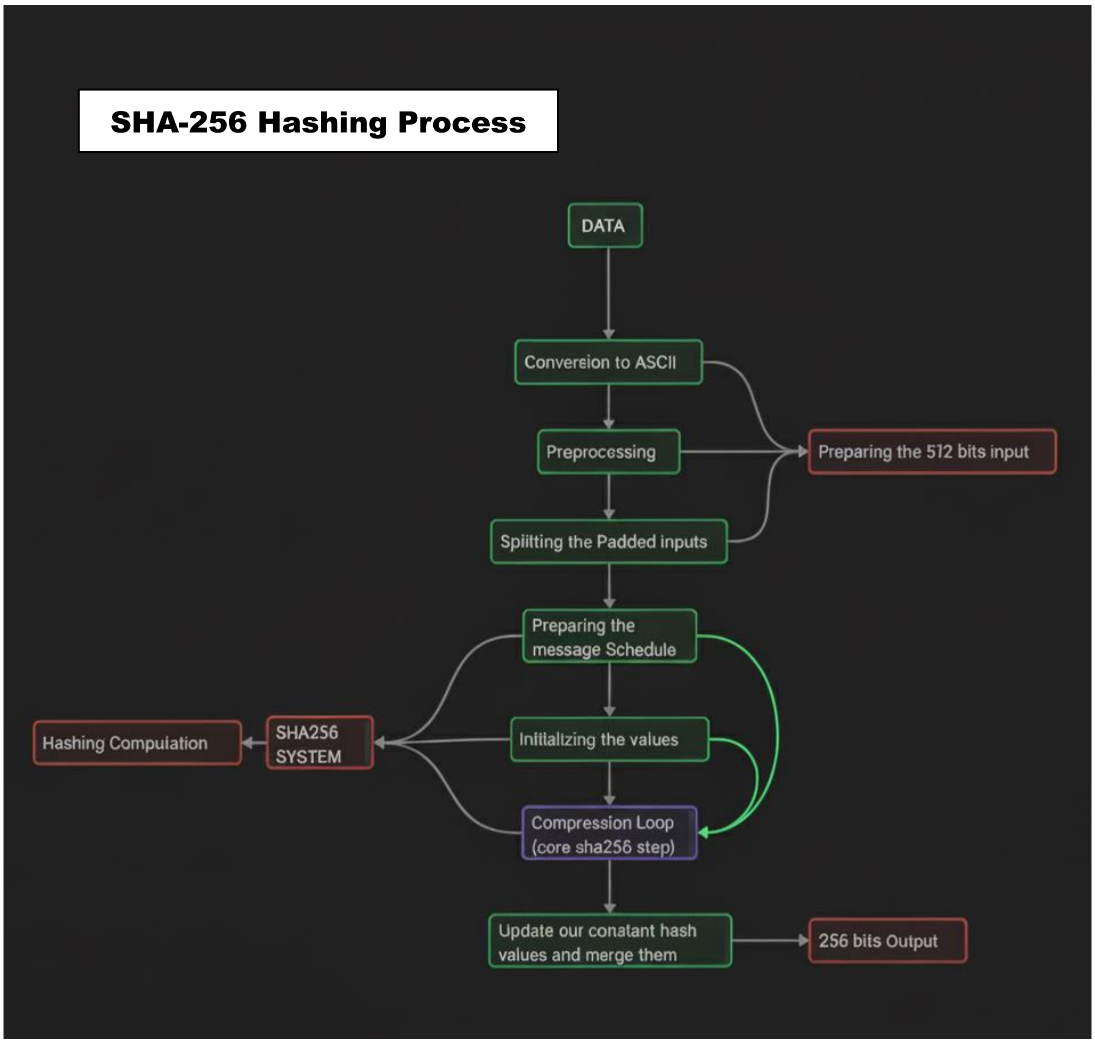
A cryptographic hash function is a one-way function. It's easy to compute the hash for any given input, but it's computationally infeasible to reverse the process—that is, to find the input from its hash. This property makes it useful for digital signatures, password storage, and data integrity verification. SHA-256 produces a unique 256-bit (32-byte) hash.



Simple Example :

Imagine you want to hash the word "hi".

1. The word "hi" is converted to ASCII.
2. It's then padded to become a 512-bit block.
3. The core hashing computation performs 64 rounds of operations on this block, using the initial hash values.
4. The final result of these operations is combined to give the 256-bit hash.


SHA-256 Hashing Process
DATA
Conversion to ASCII
Preprocessing
Preparing the 512 bits input
Splitting the Padded inputs
Preparing the message Schedule
Hashing Compulation
SHA256 SYSTEM
Initializing the values
Compression Loop (core sha256 step)
Update our constant hash values and merge them
256 bits Output

Conversion to ASCII

This initial step takes the input message, which is a string of characters, and converts it into a sequence of binary values. Each character is replaced by its corresponding **ASCII (American Standard Code for Information Interchange)** code. This creates a stream of numbers that the computer can use for calculations.

Preprocessing

This block pads the message so its length is a multiple of 512 bits. Padding involves adding a single '1' bit, followed by a series of '0' bits, and finally a 64-bit number representing the original message's length. This step is crucial to ensure the data fits the algorithm's fixed-size processing blocks.

Splitting the Padded Inputs

The padded message is divided into a series of **512-bit blocks**. If the original message was long, this step creates multiple blocks that will be processed one after another. This allows the algorithm to handle messages of any length efficiently.

Preparing the Message Schedule

Each 512-bit block is expanded into a **message schedule** of 64 words (each 32 bits long). The first 16 words are taken directly from the 512-bit block, and the remaining 48 words are generated using a specific formula. This schedule provides all the data needed for the core hashing rounds.

Initializing the Values

Before the main computation begins, a set of **eight initial hash values** (named H0 through H7) are loaded. These are specific 32-bit constants defined in the SHA-256 standard. They act as the starting point for the hashing process.

Compression Loop (Core SHA-256 Step)

This is the heart of the algorithm, where the hashing really happens. For each 512-bit block, this loop runs 64 times. In each round, it uses the current hash values, a word from the message schedule, and a round constant to perform a series of logical and arithmetic operations.

Update our constant hash values and merge them

After the compression loop finishes for a given 512-bit block, the initial hash values are updated with the results. This updated value becomes the starting hash for the next 512-bit block. After processing all blocks, these final updated hash values are concatenated to form the **final 256-bit hash output**.

Here are the essential formulas for a SHA-256 implementation.

Basic Logical Functions :

These are the three main functions used in each round of the algorithm.

1. Ch (Choice) function: Used for selecting bits from y or z based on the value of x.

   $$Ch(x,y,z)=(x\wedge y)\oplus(\neg x\wedge z)$$

2. Maj (Majority) function: Outputs the bit that is most common among x, y, and z.

   $$Maj(x,y,z)=(x\wedge y)\oplus(x\wedge z)\oplus(y\wedge z)$$

3. **Sigma Functions**: These are used in the compression function and involve right rotations and shifts. There are two types:
   - Σ0 for the compression function:

     $$\Sigma0(x)=ROTR2(x)\oplus ROTR13(x)\oplus ROTR22(x)$$

   - Σ1 for the compression function:

     $$\Sigma1(x)=ROTR6(x)\oplus ROTR11(x)\oplus ROTR25(x)$$

Message Schedule Formulas :

The 512-bit message block is expanded into 64 words (Wt) using these formulas.

1. For the first 16 words (t=0 to 15), they are taken directly from the message block:

   $$Wt=Mt$$

2. For the remaining 48 words (t=16 to 63), they are generated using the following formula:

   $$Wt=\sigma1(Wt-2)+Wt-7+\sigma0(Wt-15)+Wt-16$$

   - σ0 for the message schedule:

     $$\sigma0(x)=ROTR7(x)\oplus ROTR18(x)\oplus SHR3(x)$$

   - σ1 for the message schedule:

     $$\sigma1(x)=ROTR17(x)\oplus ROTR19(x)\oplus SHR10(x)$$

Note: All additions in these formulas are performed modulo 232.

Compression Function Round Formula :

The core of the algorithm is the 64-round compression loop. The main variable updates in each round are calculated using two temporary variables, T1 and T2.

- T1 Calculation:

  $$T1=H+\Sigma 1(E)+Ch(E,F,G)+Kt+Wt$$

  - H,E,F,G are 32-bit words from the current hash values.
  - Kt is the round constant for round t.
  - Wt is the word from the message schedule for round t.

- T2 Calculation:

  $$T2=\Sigma 0(A)+Maj(A,B,C)$$

  - A,B,C are 32-bit words from the current hash values.

These temporary variables are then used to update the hash values for the next round. The final 256-bit hash is the concatenation of the final eight hash variables.