**IMPLEMENTATION OF MD-5 ALGORITHM USING VERILOG**

**I. ABSTRACT:**

This report consists of the implementation of the hardware-based Message Digest Algorithm, (MD-5), which is designed using the hardware description language, Verilog. MD-5 is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message.

**II. HASH FUNCTIONS:**

A Hash Function is any function that can be used to map data of arbitrary size to fixed-size. It is a sort of operation that takes an input and produces a fixed-size string which is called the hash value. The input string can be of any length depending on the algorithm used. The produced output is a condensed representation of the input message and is called as a message digest. A very small change in the input results with a completely different hash value. In the MD-5 algorithm, the hash is of 128-bits (16 bytes).

**III. THE MD-5 ALGORITHM:**

**1. Appending Padding bits:**

Padding means adding extra bits to the original message. In MD5 padding is done such that the total bits are 64 less, being a multiple of 512 bits length. In padding bits, the only first bit is 1, and the rest of the bits are 0.

Length (original message + padding bits) = 512 \* i – 64 where i = 1,2,3 . . .

**2.** **Appending Length bits:**

After padding, 64 bits are inserted at the end, which is used to record the original input length. The resulting message has a length multiple of 512 bits.

**3. Initializing MD buffers:**

A four-word buffer (A, B, C, D) is used to compute the values for the message digest. Here A, B, C, D are 32- bit registers with the following values:

A='h67452301 B='hefcdab89 C='h98badcfe D='h10325476

**4. Processing each block:**

* Each 512-bit block gets broken down further into 16 sub-blocks of 32 bits each. There are four rounds of operations, with each round utilizing all the sub-blocks, the buffers, and a constant array value.
* The constant array is denoted as t [1] -> t [64]. Each of the sub-blocks are denoted as M [0] -> M [15].The order of processing the instructions is as follows:

B, C, and D are passed onto a non-linear process. Then the result is added with the value present at A. Then, the constant value for that iteration is added. There is a circular shift applied to the string. As a final step, it adds the value of B to the string and is stored in buffer A. The steps are run for every buffer and every sub-block. After all rounds are performed, the buffer A, B, C, D contains the MD5 output.

* The operations that are performed are:

1. Add modulo 2^ (32)

2. D[i] – 32-bit message.

3. B[i] – 32-bit constant.

4. <<<n – Left shift by n bits.

* **Auxiliary Functions**: Auxiliary functions take three inputs (32-bits word) and give an output of 32-bit word. These functions apply logical AND, OR and XOR to the inputs. The non-linear process above is different for each round of the sub-block.

Round 1: (b AND c) OR ((NOT b) AND (d))

Round 2: (b AND d) OR (c AND (NOT d))

Round 3: b XOR c XOR d

Round 4: c XOR (b OR (NOT d))

**IV. IMPLEMENTATION OF MD-5 ALGORITHM USING VERILOG:**

We have used the Xilinx Vivado tool to implement the design of the algorithm in Verilog. MD5 consists of several rounds of message block processing and state updates. Each round involves a series of logical and arithmetic operations and the current state of the algorithm. The state consists of four 32-bit registers: A, B, C, and D, which are initialized with predefined constants.

The message block is divided into 16, 32-bit words, which are used as input to the round operations. The implementation of MD5 in Verilog typically involves defining the state registers and the round operations as combinational or sequential logic.

This Verilog module takes an 8-bit message input and a clock signal as inputs, and produces a 128-bit hash output. It uses the functions defined in the MD5 specification to compute the round operations, and updates the state registers on each clock cycle using a clocked process. The output hash is generated by concatenating the state registers.

The message padding module would take in the input message and add padding bits to it so that its length is a multiple of 512 bits.

The message schedule module would take in the padded message and generate a 64-entry message schedule array based on it.

The round function module would take in a 512-bit block of the message schedule array and perform a series of logical and arithmetic operations on it to produce a 128-bit hash value.

Once these modules have been designed, they can be connected to form the overall MD5 circuit. This circuit would take in the input message and output the corresponding hash value.

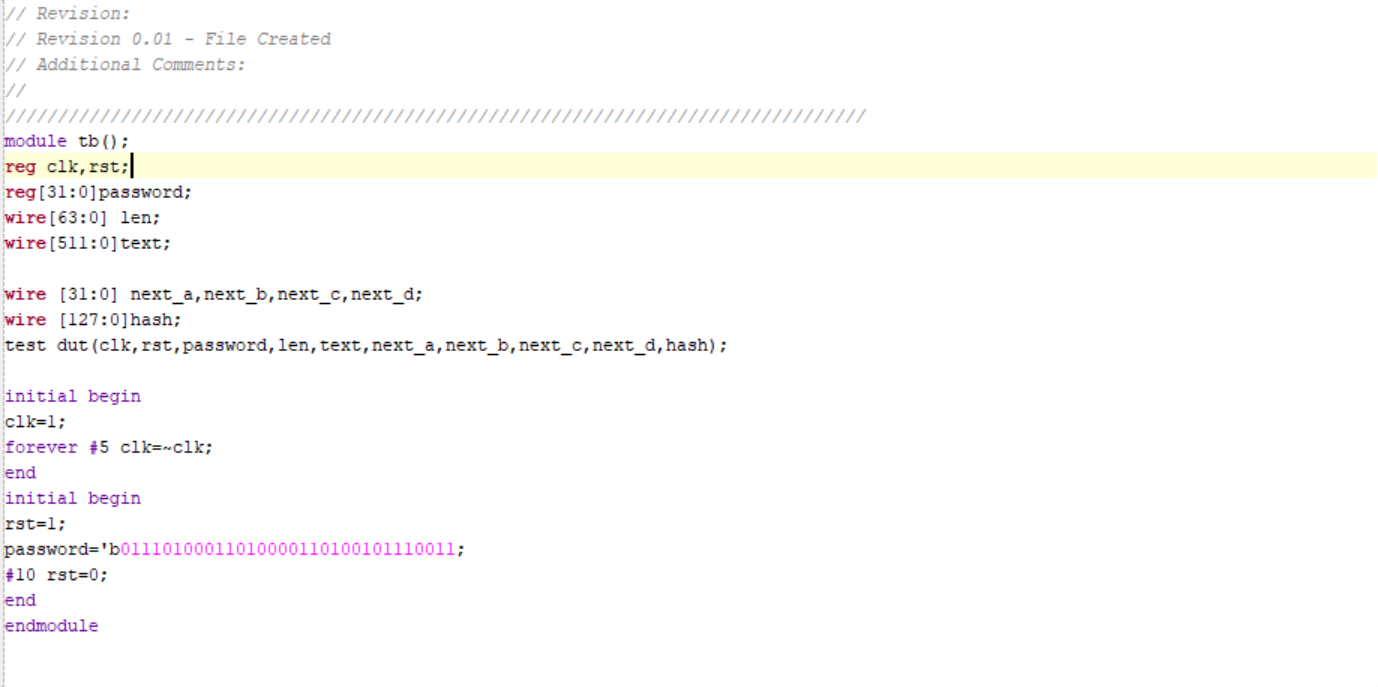
In Verilog, the design of these module involves defining input and output ports, specifying internal registers and wires, and implementing the necessary logic and arithmetic operations.

Top of Form

Bottom of Form

**V. TESTBENCH:**

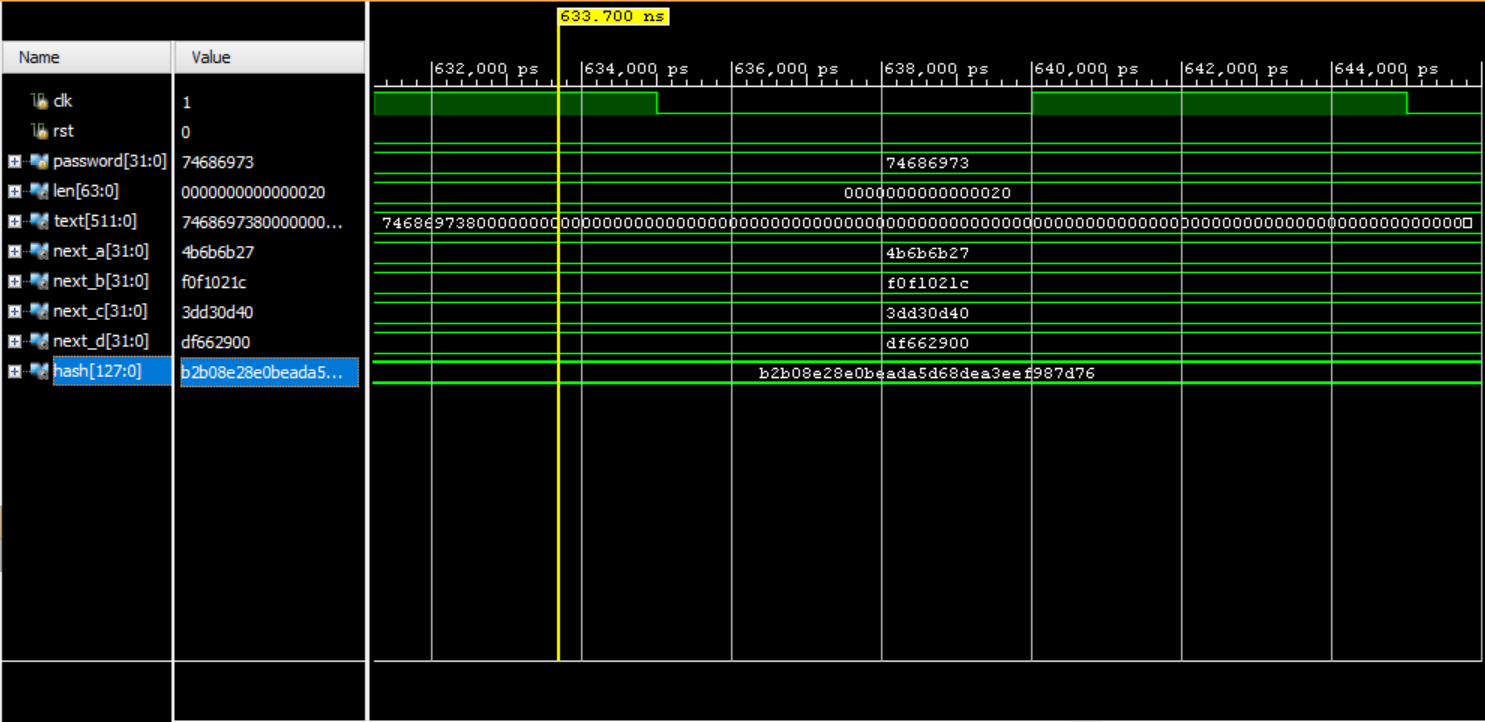
A testbench for an MD5 implementation in Verilog involves generating input messages of various lengths and verifying that the corresponding output hash values are correct. The test bench that we have implemented is as follows:



Here the variable ‘password’ is the input message, ‘rst’ is the reset input and ‘clk’ is the clock used in the circuit. The variable ‘hash’ gives the final output of the MD-5 algorithm. The ‘hash’ is of 128 bits or 16 bytes in length.

**VI. SIMULATION RESULTS:**

The output hash corresponding to the input is as obtained as below. The variable hash contains the MD5 digest of the input message, password.



**VII. APPLICATIONS:** One of the most important usages of MD5 is one-way password encryption. While storing passwords it is important that the password string itself is not stored inside the database. Instead MD5 or any other hashing function is used to generate a hash value of the original password input. When the password is entered first, the user password will be stored as the hash value inside the database. The next time the user uses the password; it is again hashed and then compared with the stored hash value. If both the hash values match then user gets access.

**VIII. CONCLUSION:**  MD5 is an efficient one-way hash function that generates a hash value of the input. This algorithm provides means to secure any messages and an effective way to check the integrity of any data between serves. Though there have been collisions found for the MD-5 algorithm, it is one of the widely used algorithms to secure passwords.