The proposed SHA-256 algorithm implementation is divided into two major blocks. All the control signals are controlled from a separate block named as Control Unit. All the hardware components are implemented in data path block (i.e. counter, ROM, XOR and other operations).

1. Control Unit

In this block an FSM is proposed to control the data path different modes. A 2-bit state register is used for state transition. A four state FSM is proposed. In IDLE state all the active low resets are asserted into all Datapath components. In RESET state it waits for the start another control input signal from user to drive the state from RESET to BUSY. In BUSY state all the enable signals are raised high. In this state a feedback signal from the data path is used that goes high after computing all 64 rounds and drives the FSM state from BUSY to DONE. In DONE state valid signals is raised for only one clock cycle and FSM state register is again pushed back to IDLE state.

2. Data Path

All the hardware components are placed inside this block. All these are globally controlled by Control Unit. A feedback signal is also passed back from counter.

2.1.Counter

A 6-bit counter is used to provide the address to two major blocks that is message schedule block and round constants. Its start counting when Control unit asserts the control signal of counter enable high. It counts from 0 to 63 and asserts the flag signal high and changes the state of control unit to DONE.

2.2.Round Constant

In round constant block a simple rom is used that is holding all the values of round constant provided by NIST. The ROM is read based on the address provided by counter. This value is passed to round block.

2.3. Message Schedular

It provides all 64 message weights (i.e. 4 bytes) based on counter value to the round block. Here the combinational circuit is implemented in separate (*package*) file and later on used to find message weights.

2.4.Round Block

This is the most critical block of the whole algorithm all the processing of the round constant along with message word (i.e. 4 byte) are done with Initial value of hash. Later on, this Initial value is updated with the each coming round. All the combinational circuits are implemented in a separate (*pacakge*) file and these functions are used inside the round block. The message weight and round constant comes from message schedular and round constant block respectively.