NTHU- EE525500 Design of Chip Security- Spring 2024

Lab02 FSM

Objective

Learn to use FSM to implement a circuit of **SHA2-256** and **hardware padding**

Prior Knowledge

1. Syntax of combinational and sequential circuits in Verilog.
2. Secure Hash Algorithm Standard (SHA) (FIPS 180-4).

Submission Content

1. Design a module:
    a. Calculate the correct outputs and comply with the IO protocol
    b. Pass the lint check (No warnings and errors)
    c. Could be synthesized (No warnings and errors)
    d. NOT contain latches or combination loop after synthesis
    e. It is necessary to use FSM in the circuit
2. A Report:
    a. Explain the design architecture and performance(clock_cycle/512-bit)
    b. Explain the testbench (how to verify the design module)
    c. Coverage report (screen capture of code coverage and FSM transition)
    d. If necessary, use Wavedrom to draw the waveform

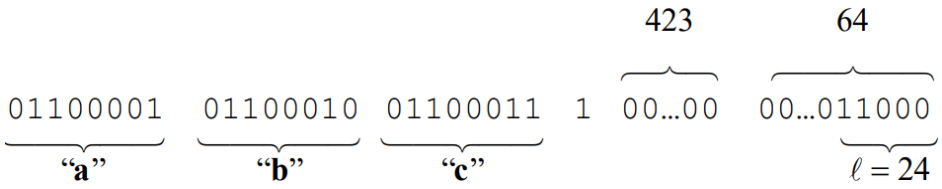
Module Specification

You need to implement the complete **SHA2_256 algorithm** and **hardware padding**

SHA-256 may be used to hash a message, M, having a length of ℓ bits, where where $0 \leq \ell < 2^{64}$. The algorithm uses (1) a message schedule of sixty-four 32-bit words, (2) eight working variables of 32-bit each, and (3) a hash value of eight 32-bit words. The result of SHA-256 is a 256-bit message digest.

The words of the message schedule are labeled W_0, W_1, …, W_{63}, The eight working variables are labeled a, b, c, d, e, f, g, and h. The words of the hash value are labeled H_0, H_1, …, H_7, which will hold the initial hash value, H^0, replaced by each successive imtermediate hash value H^i, and ending with the final hash value H^N.

Suppose that the length of the message, M, is ℓ bits. Append the bit "1" to the end of the message, followed by k zero bits, where is the smallest, non-negative solution to the equation $\ell + 1 + k = 448 \bmod 512$



For more detailed algorithm, please refer to *https://csrc.nist.gov/files/pubs/fips/180-2/final/docs/fips180-2.pdf*

A SHA256 module specification

Module name: sha2

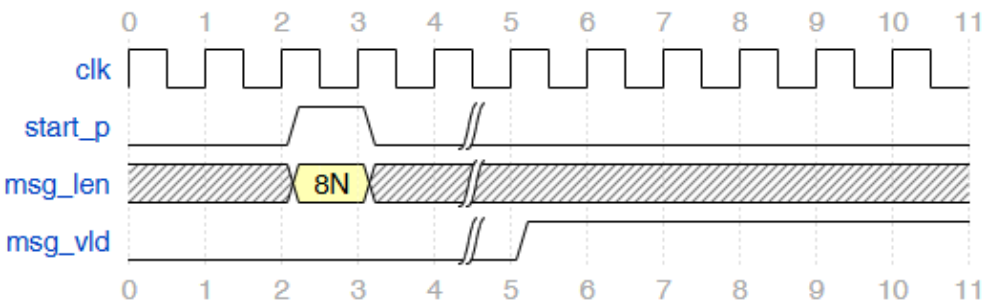
IO signals:

input	start_p	
input [63:0]	msg_len	//Unit: bit
input	msg_vld	
input [31:0]	msg_dat	//Total size is N * 8 bits
input [3:0]	msg_be	//Byte enable
input	msg_lst	//The last word of messages
output	msg_rdy	
output	dgst_done	
output [255:0]	dgst	
input	clk	//Periodic clock, 50MHz
input	rst_n	//Asynchronous reset, active low

The IO protocol of message-side is shown below:

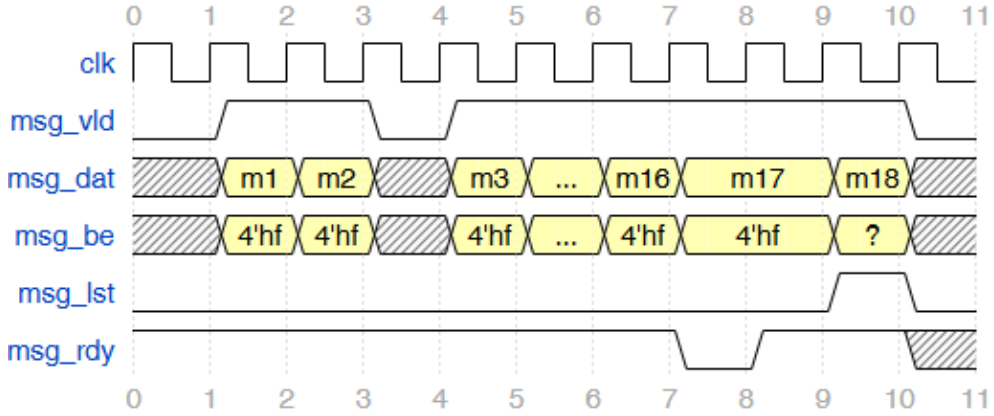
1. "rst_n" will be active at the beginning of the test
2. "start_p" rises one cycle time at the beginning of a message, Each rise indicates that a hash digest is to be generated.
3. "msg_len" indicates the total length of the message. It must be 8 * N bits (could be 0). "msg_len" is valid when "start_p" rises.

[Fig.1] Message Interface (1/2)



4. “msg_vld” may rise after “start_p,” indicating that “msg_dat” is valid. No matter what “msg_rdy” is, “msg_vld” is allowed not to be continuous
5. If the total length of message is 0, “msg_vld” would not rise
6. “msg_be” indicates which bytes of “msg_dat” are valid, usually is 4’hf, except for the last word of message may be 4’hf, 4’he, 4’hc or 4’h8 (for example, “msg_dat” is 32’h5a5a5a5a, 32’h5a5a5aXX, 32’h5a5aXXXX or 32’h5aXXXXXXXX)
7. If “msg_rdy” is low, “msg_dat” will remain unchanged until “msg_rdy” rises again. Pull up “msg_rdy” should not wait for “msg_vld” is high.

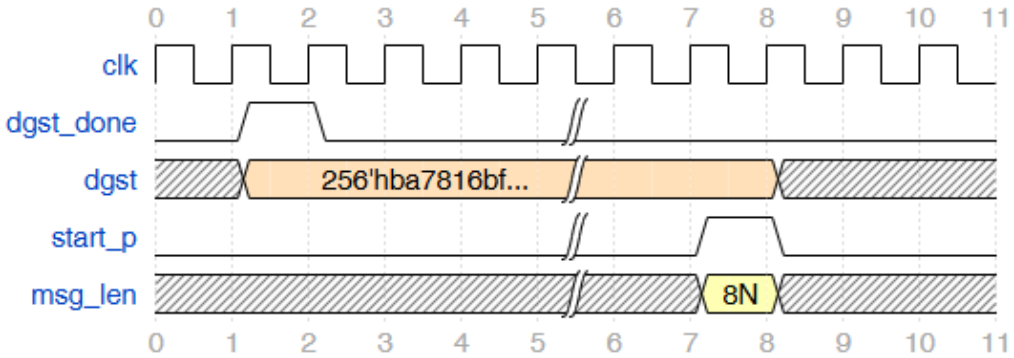
[Fig.2] Message Interface (2/2)



The IO protocol of digest-side is shown below:

8. After digest is claculated, “dgst_done” should be raised for one cycle
9. The dgst should remain unchanged until “start_p” rises again. Next “start_p” will not appear before the hash digest is generated.

[Fig.3] Digest Interface



Reference Environment

- lab02.zip
  It contains a folder, sha2/, which is the development environment for the module.
- If necessary, you can add your own submodules and test files into the environment. You can update ./sim/file.v or ./rtl/dut_include.v
- Please DO NOT:
  - Delete any files that were originally in the lab02.zip
  - Change the file hierarchy (do not change the location of the files)
  - Increase or decrease the module's IO or change its width

Submit

- lab02_xxxxxxx.zip (xxxxxxx is your student ID)
  Please run ./09_clean before you deliver. Use the following command to package the lab02 folder.
  zip -r lab02_xxxxxxx.zip lab02
- lab02_xxxxxxx.pdf
  Please write your report and submit it in pdf format

Credit

- Design contains FSM + Functional pass + Lint pass + Synthesis Pass (No latches or combinational loops) (70%)
- Design area, small area is preferred (20%)
- Report (10%)