sha256

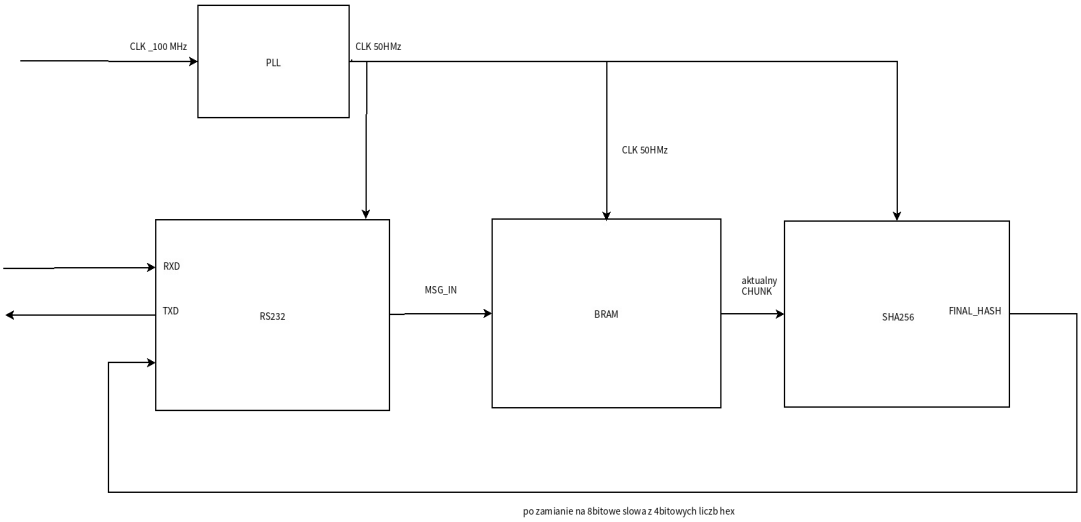
Project done for Warsaw University of Technology' s RIM course.

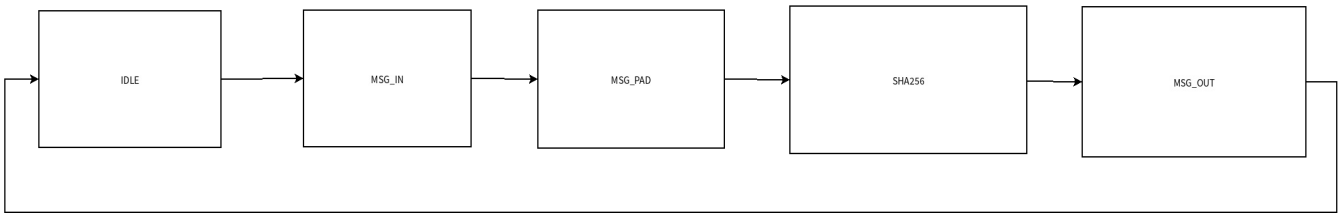
Written in VHDL.

SHA256 definition taken from FIPS 180-2 Secure Hash Standard.

Contains:

- Design Entities
  - uart module for loading messages and writing out the calculated hash
    - 9600 BaudRate
    - HalfDuplex
    - 1 stop bit
    - no parity bit
    - no flow control
    - no echo functionality
  - RAM to store the soon to be hashed msg
    - msg lenght is limited to 64 chunks
  - PLL block
    - Altera
      - inclk : 10MHz
      - outclk : 50MHz (required frequency for the UART)
    - Xilinx
      - inclk : 100Mhz
      - outclk : 50MHz (required frequency for the UART)
  - sha256 algorithm module
    - works on 512b chunks and reads the corresponding 32b msg words from the RAM
- TestBenches (reference hash values were taken from Sha256-Online site)
  - that tests uart functionality
  - for the sha256 algorithm module to test chunk_n>=1 messages
    - used assert statements to verify hash-calculated vs hash-ref
  - system tb that tests the integration of the componenets
    - has no direct comparison of the result vs reference
    - can be used to verify if whole pipeline works





Data Flow:

1. Data comes in through the UART module in form of 8bit ASCII characters, MSG ends whenever UART receives EOT(0x04) character
2. When the input msg is complete FSM inside the main module will start preprocesing the message by adding PADDING + placing the total lenght in the last CHUNK
   - This is already almost entirely taken care of when the msg is being written into memory since every 14th & 15th word is avoided thus what needs to be done in

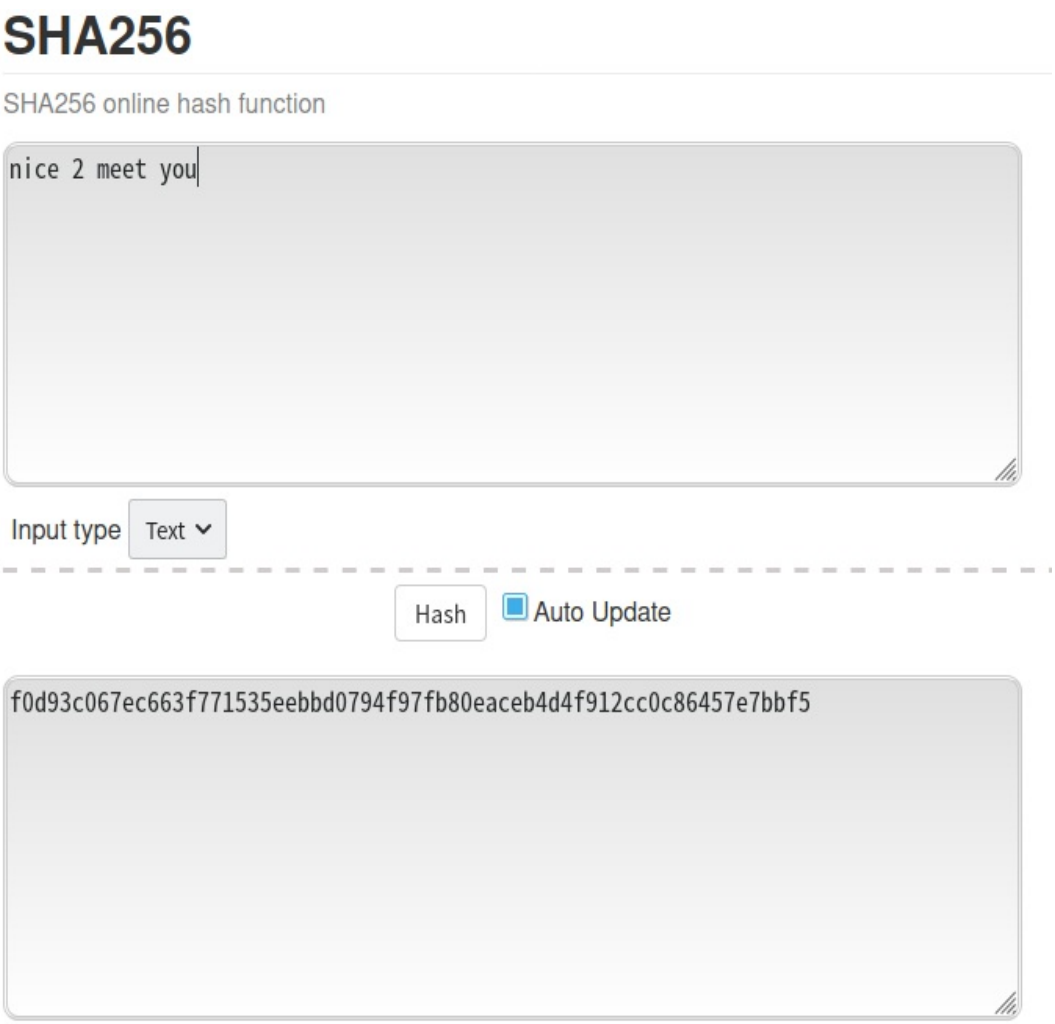
this step is clearing those addresses, appending x80 or updating the value with msgLenght

3. After the previous step the MSG is finally ready to be go into sha256 algorithm module where it worked on chunk by chunk
    - Algorithm was taken straight from FIPS 180-2 Secure Hash Standard
4. When every chunk has been processed and the final_hash is ready it' s state is stored in the register and It is will be transmitted to the USER through the UART module.
5. After every character in the 256 hash has been sent the module is ready for the next word.

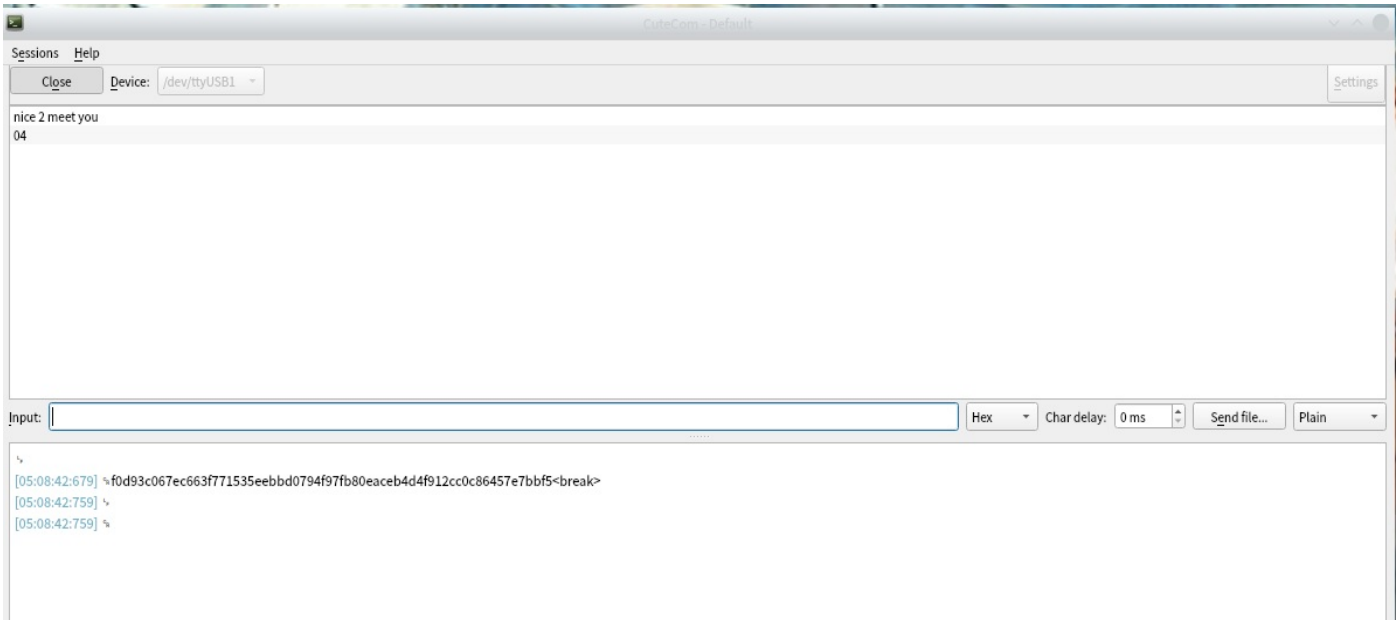
This flow doesn' t allow for pipelining but it was not it' s intention since the connection through UART is mainly used for Final Field Testing.

HW Testing (using Linux)

- Reference



- Result (connection to Board done using ***cutecom***)



1. `nice 2 meet you` sent without any END character (such as CR LF etc)
2. `04` sent as HEX to indicate message end

Project Summaries

Tried running on 2 FPGAs: - MAX10 (design too big) `Originally the project was meant to land on the MAX10 FPGA MAXimator Board from KAMAMI but it does not fit. Thus the project was regenerated and moved to a Xilinx Artix-7 100T Arty7 digilent board.` - Artix7 xa7a100tcsg324-2l on the Arty A7 Digilent Board

Resource Utilization

- **POST_SYNTH** Altera MAX10 on the maximator board using (Quartus Prime 15.1)

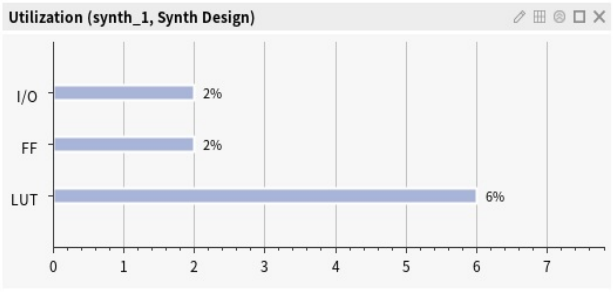
Resource	Utilization
Total logic elements	9315
Total combinational functions	7152
Dedicated logic registers	3079
Total registers	3079
Total pins	6
Total virtual pins	0
Total memory bits (M9K blocks)	32768(4)
Embedded Multiplier 9-bit elements	0
Total PLLs	1
UFM blocks	0
ADC blocks	0

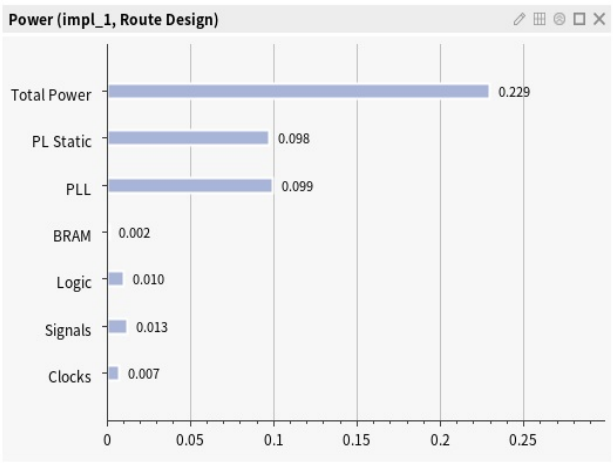
- **POST-IMPLEMENTATION** Xilinx Artix7 on the Arty A7 board (using Vivado 2020.2)

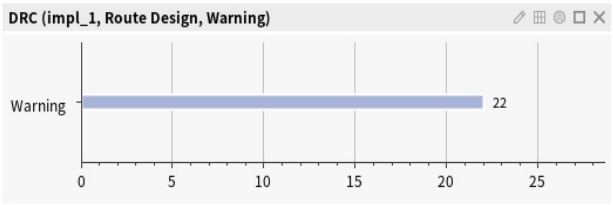
Resource	Utilization	Available	Utilization %
LUT	4041	63400	6.37
FF	3127	126800	2.47
BRAM	1	135	0.74
IO	6	210	2.86
BUFG	2	32	6.25
PLL	1	6	16.67

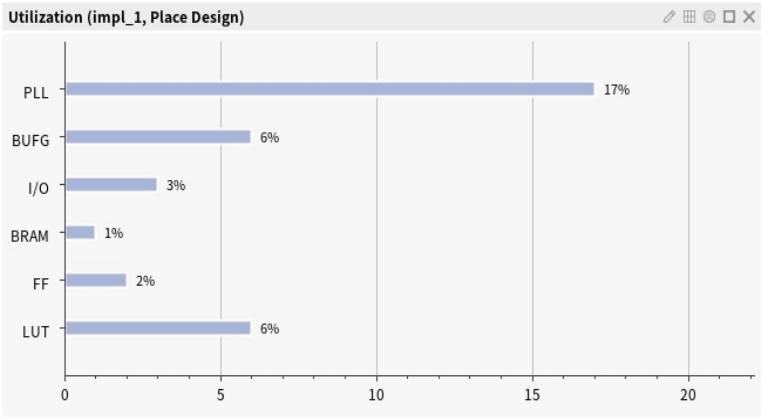
- Timing & Power

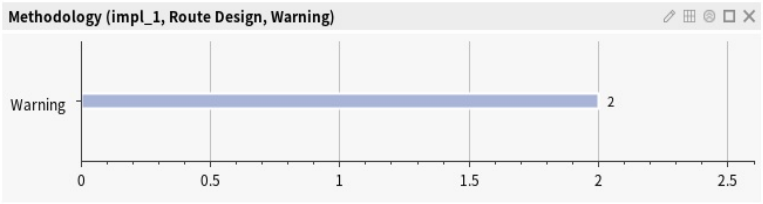
WNS	TNS	WHS	THS	TPWS	Total Power[W]
-0.045	-0.052	0.083	0.000	0.000	0.229











Helpful Links:

- https://github.com/skordal/sha256
- https://github.com/dsaves/SHA-256 (which helped greatly by providing full intermediate hash table for a specific test msg)