

SYSTEM VERILOG:

Implementation of bitcoin blockchain through SHA256 hash algorithm

PART 1: SHA256

SHA256 explained:

SHA256 is a cryptographic secure hash algorithm used for bitcoin. This algorithm takes an input message, and outputs a unique 256 bit hash. Salient properties of sha256 include that the output is preimage resistant, second preimage resistant and is also collision resistant. The same input produces the same output, with a one to one mapping.

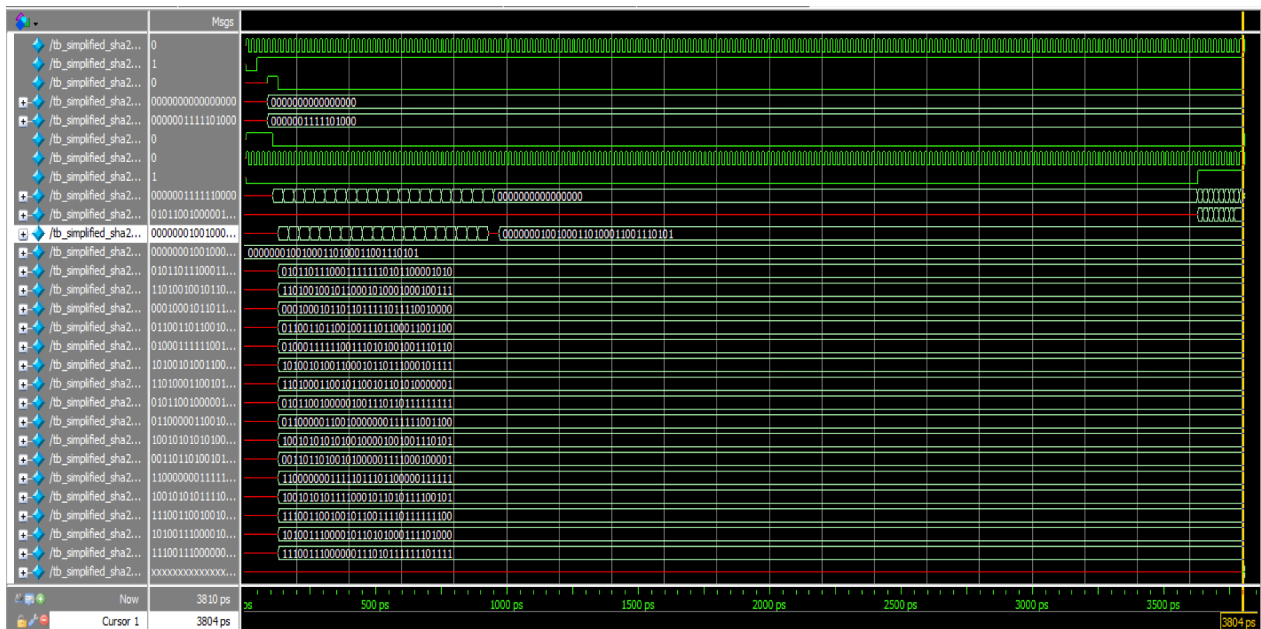
SHA256 implementation:

My implementation uses sequential logic through only one always_ff block and FSM logic. The FSM states are as follows: IDLE, READ_BUFFER, READ_MSG, BLOCK, COMPUTE, WRITE_BUFFER, WRITE_BUFFER2, WRITE. I initialized all the relevant variables and standard hashes in the IDLE state, and then moved on to READ the message, with the help of a buffer state. The message is then appropriately padded to the correct specification. We are left with a message consisting of $n \times 512$ bits. Each 512 bit block is then accessed iteratively in the

BLOCK state, and we fill in our $w[n]$ array with the weights accordingly. Each block is then sent to the COMPUTE state to process. Each block undergoes 64 rounds of processing. I implemented a **parallel** approach in the Compute state. The $w[n]$ array is then shifted to the left by one bit from $w[0]$ to $w[14]$. Only $w[15]$ is computed using a word expansion helper function. Assignments of new hash once the 64 iterations of each block are complete. Once we are through all the blocks, our hashes are updated to the new values. Then our $8 * 32$ bit hashes are written in memory with the help of WRITE_BUFFERS. We now have a secure and unique 256 bit hash for our input.

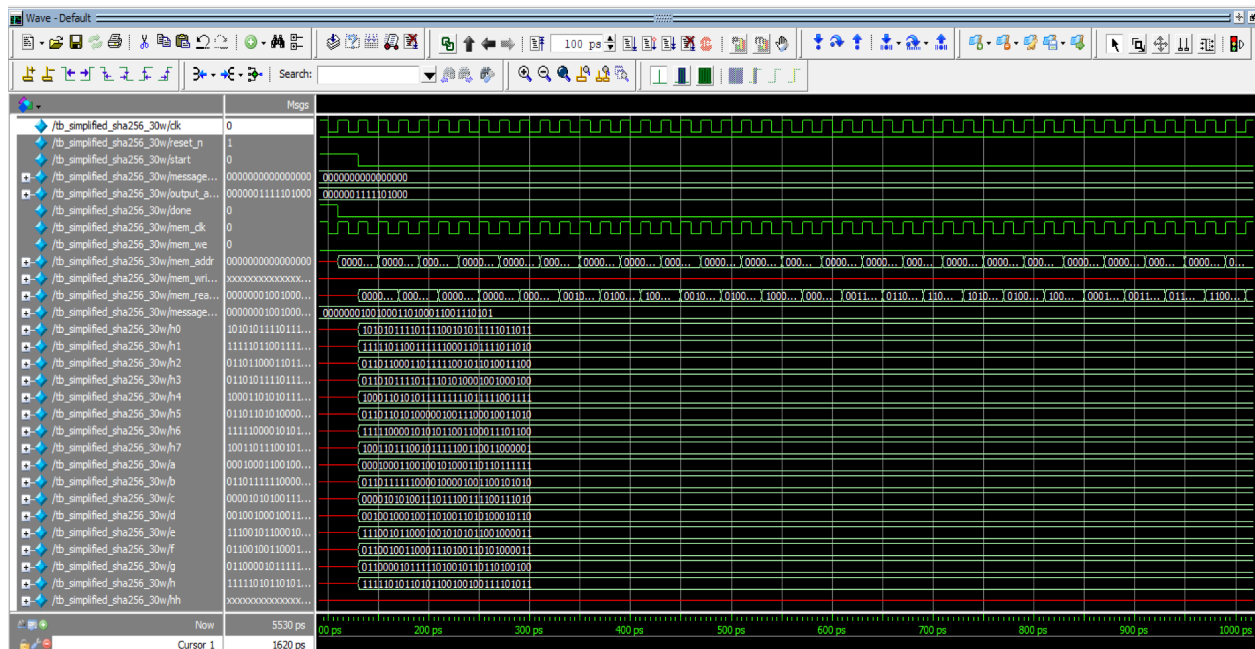
WAVEFORMS and TRANSCRIPTS:

For 20 words:



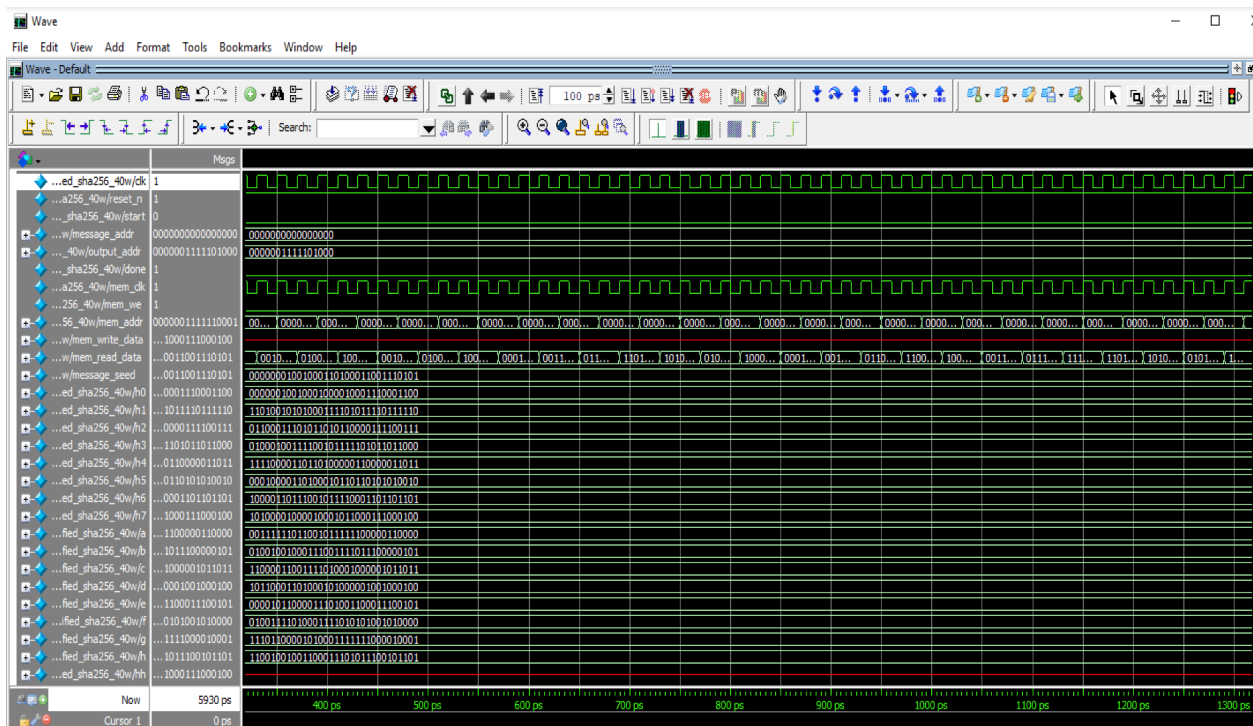
```
# -----
# COMPARE HASH RESULTS:
# -----
# Correct H[0] = 5b8feb0a Your H[0] = 5b8feb0a
# Correct H[1] = d258a227 Your H[1] = d258a227
# Correct H[2] = 116df790 Your H[2] = 116df790
# Correct H[3] = 66c9d8cc Your H[3] = 66c9d8cc
# Correct H[4] = 47e75276 Your H[4] = 47e75276
# Correct H[5] = a5316e2f Your H[5] = a5316e2f
# Correct H[6] = d1965a81 Your H[6] = d1965a81
# Correct H[7] = 5904edff Your H[7] = 5904edff
# *****
#
# CONGRATULATIONS! All your hash results are correct!
#
# Total number of cycles:          188
#
# *****
```

For 30 words:



```
# -----  
# COMPARE HASH RESULTS:  
# -----  
# Correct H[0] = abde57db Your H[0] = abde57db  
# Correct H[1] = fb3f1bda Your H[1] = fb3f1bda  
# Correct H[2] = 6c6f969c Your H[2] = 6c6f969c  
# Correct H[3] = 6bdea244 Your H[3] = 6bdea244  
# Correct H[4] = 8d5ff7cf Your H[4] = 8d5ff7cf  
# Correct H[5] = 6d41389a Your H[5] = 6d41389a  
# Correct H[6] = f85598ec Your H[6] = f85598ec  
# Correct H[7] = 9b97cccl Your H[7] = 9b97cccl  
# *****  
#  
# CONGRATULATIONS! All your hash results are correct!  
#  
# Total number of cycles:          274  
#  
#  
# *****
```

For 40 words:



```

# -----
# COMPARE HASH RESULTS:
# -----
# Correct H[0] = 0244238c Your H[0] = 0244238c
# Correct H[1] = d2a3d7be Your H[1] = d2a3d7be
# Correct H[2] = 63ad61e7 Your H[2] = 63ad61e7
# Correct H[3] = 44f2fad8 Your H[3] = 44f2fad8
# Correct H[4] = f0da0c1b Your H[4] = f0da0c1b
# Correct H[5] = 10dl6d52 Your H[5] = 10dl6d52
# Correct H[6] = 86e5e36d Your H[6] = 86e5e36d
# Correct H[7] = a108b1c4 Your H[7] = a108b1c4
# *****
#
# CONGRATULATIONS! All your hash results are correct!
#
# Total number of cycles:          294
#
# *****

```

RESOURCE UTILIZATION:


REGISTERS / ALUT'S:

	Resource	Usage
1	▼ Estimated ALUTs Used	1707
1	-- Combinational ALUTs	1707
2	-- Memory ALUTs	0
3	-- LUT_REGS	0
2	Dedicated logic registers	2136
3		

FLOW SUMMARY:

Fitter Status	Successful - Thu Dec 08 12:46:45 2022
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	simplified_sha256
Top-level Entity Name	simplified_sha256
Family	Arria II GX
Device	EP2AGX45DF29I5
Timing Models	Final
Logic utilization	8 %
Total registers	2136
Total pins	118 / 404 (29 %)
Total virtual pins	0
Total block memory bits	0 / 2,939,904 (0 %)
DSP block 18-bit elements	0 / 232 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)

FMAX:

Slow 900mV 100C Model Fmax Summary				
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	155.04 MHz	155.04 MHz	clk	

PART 2: BITCOIN HASH

BITCOIN HASH explained:

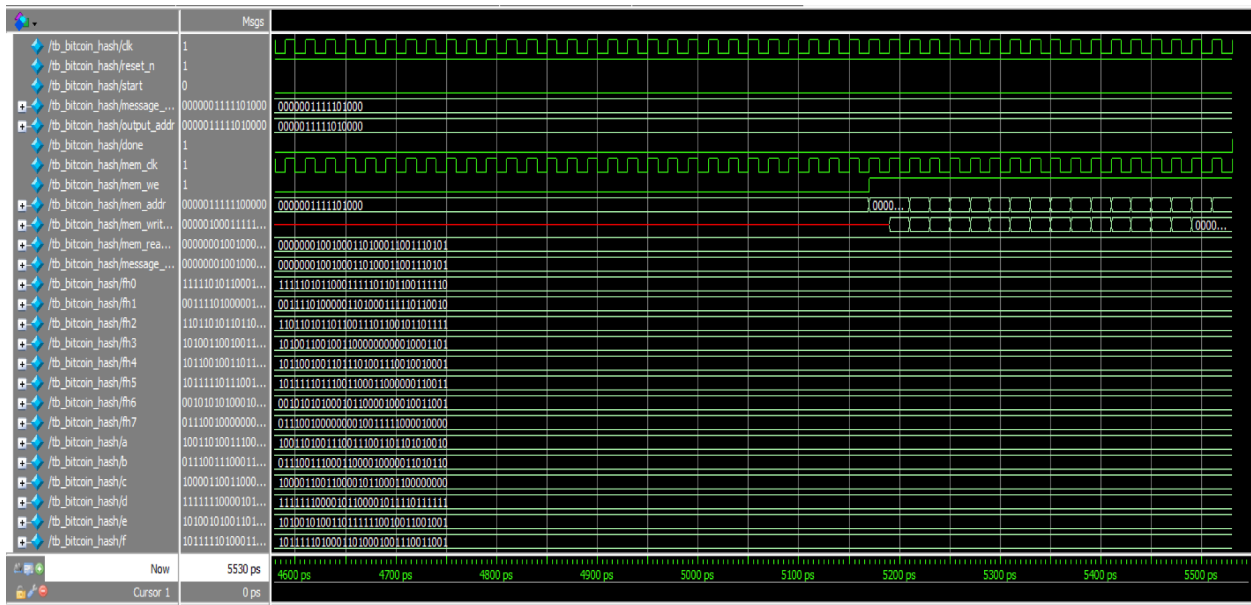
Bitcoin hashing utilizes the one-to-one and deterministic properties of the sha256 algorithm to connect blocks in a block chain. This Bitcoin hash algorithm connects one block of information to the next through the hashes produced from the block information. That is, the hash value of the previous block is used to calculate the unique hash of the present block and so on. Since the hashes are preimage and collision resistant, the chains that are produced are unique and secure.

BITCOIN HASH Implementation:

My bitcoin hash uses sequential logic through only one always_ff block and FSM logic. The FSM states are as follows: IDLE, READ_BUFFER1, READ_MSG, BLOCK, COMPUTE, WRITE_BUFFER, WRITE_BUFFER2, WRITE. For parallelism, I initialized 17 instances of simplified_sha256.v: 1 for phase 1 and the rest for phase 2 and 3. I initialized all the relevant variables in the IDLE state, and then moved on to READ the message, with the help of a READ_BUFFER. We pad the message appropriately upto 32 bits. For PHASE_1, we prepare our weights array w0[] consists of the 16 word input as they are. We then allow Inst0 of sha256 to calculate the hashes h0[8] for the w0 input. Once done, in preparation for phase2 We prepare 16 new weight arrays w0[16] to w15[16] and pad them appropriately including nonces 0 to 15. We then allow the 15 instances of sha256 to calculate the new hashes h1[8] to h16[8] using the hash h0 from phase 1. Once done calculating, we move on to PHASE_3. We prepare our weight arrays w0 to w15 with the new hashes from phase 2 and pad the remaining. In phase 3, we allow the sha256 instances to once again calculate the new hashes using the previous hashes. Once done, we have our final output in the form of h1[0] to h16[0]. We proceed to WRITE them into memory using 2 WRITE_BUFFERS.

WAVEFORMS and TRANSCRIPTS:

Waveform:




TRANSCRIPT:


```

# -----
# COMPARE HASH RESULTS:
# -----
# Correct H0[ 0] = a0211662 Your H0[ 0] = a0211662
# Correct H0[ 1] = bfbb6ccd Your H0[ 1] = bfbb6ccd
# Correct H0[ 2] = da017047 Your H0[ 2] = da017047
# Correct H0[ 3] = 1c34e2aa Your H0[ 3] = 1c34e2aa
# Correct H0[ 4] = 58993aea Your H0[ 4] = 58993aea
# Correct H0[ 5] = b41b7a67 Your H0[ 5] = b41b7a67
# Correct H0[ 6] = 04cf2ceb Your H0[ 6] = 04cf2ceb
# Correct H0[ 7] = 85ab3945 Your H0[ 7] = 85ab3945
# Correct H0[ 8] = f4539616 Your H0[ 8] = f4539616
# Correct H0[ 9] = 0e4614d7 Your H0[ 9] = 0e4614d7
# Correct H0[10] = 6bec8208 Your H0[10] = 6bec8208
# Correct H0[11] = ce75ecf2 Your H0[11] = ce75ecf2
# Correct H0[12] = 672cb1a0 Your H0[12] = 672cb1a0
# Correct H0[13] = 4d48232a Your H0[13] = 4d48232a
# Correct H0[14] = cfe99db3 Your H0[14] = cfe99db3
# Correct H0[15] = 047d81b9 Your H0[15] = 047d81b9
# *****
#
# CONGRATULATIONS! All your hash results are correct!
#
# Total number of cycles:          274
#
#

```

RESOURCE UTILIZATION:

Flow Summary	
 <<Filter>>	
Flow Status	Successful - Thu Dec 08 13:49:22 2022
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	bitcoin_hash
Top-level Entity Name	bitcoin_hash
Family	Arria II GX
Device	EP2AGX45DF29I5
Timing Models	Final
Logic utilization	94 %
Total registers	27473
Total pins	118 / 404 (29 %)
Total virtual pins	0
Total block memory bits	0 / 2,939,904 (0 %)
DSP block 18-bit elements	0 / 232 (0 %)
Total GXB Receiver Channel PCS	0 / 8 (0 %)
Total GXB Receiver Channel PMA	0 / 8 (0 %)
Total GXB Transmitter Channel PCS	0 / 8 (0 %)
Total GXB Transmitter Channel PMA	0 / 8 (0 %)

Slow 900mV 100C Model Fmax Summary

 <<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	138.05 MHz	138.05 MHz	clk	

	Resource	Usage
1	▼ Estimated ALUTs Used	17184
1	-- Combinational ALUTs	17184
2	-- Memory ALUTs	0
3	-- LUT_REGS	0
2	Dedicated logic registers	27473

SUMMARY:

bitcoin_hash.sv (MIN DELAY DESIGN)						
#ALUTs	#Registers	Area	Fmax (MHz)	#Cycles	Delay (microsec)	Area*Delay (millisec*area)
17184	27473	44657	138.05	274	1.985	88599.488