MATTHEW ALEGRADO
ANDREI SECOR
TRIPTI CHANDA

Project Proposal

Develop a Bitcoin hashing RTL model using SHA256 hashing functionality.

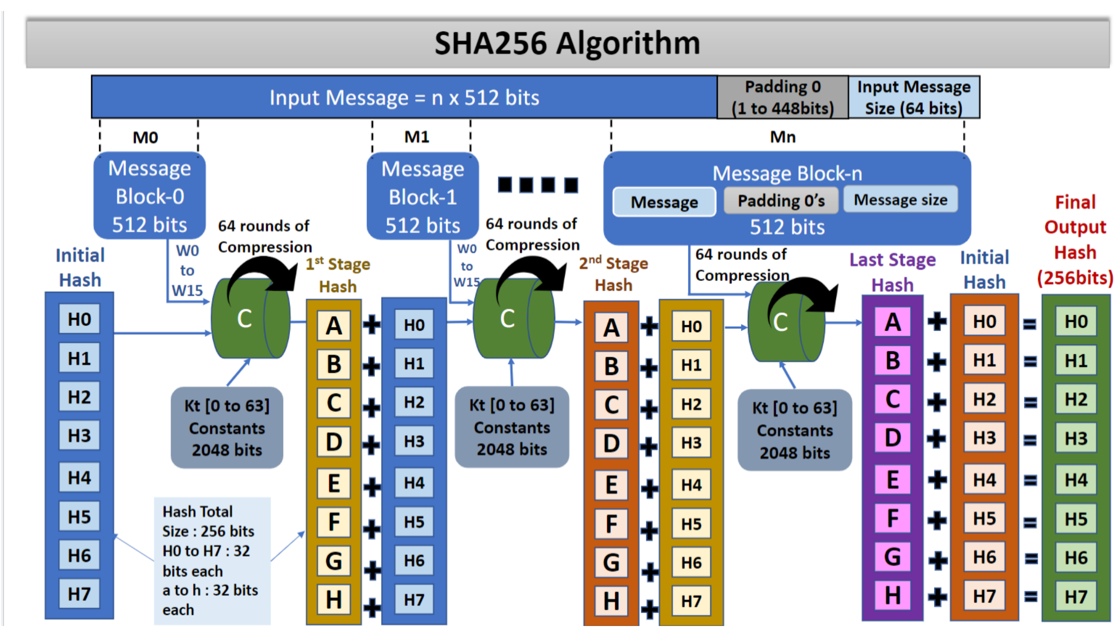
What is a SHA256 hashing function?

SHA-256, a cryptographic hash function algorithm, is specifically designed to transform data of any size into a fixed-size output. The primary objective of this function is to efficiently map data of varying lengths to a standardized "hash" that is incredibly challenging to reverse-engineer, making it suitable for a wide range of applications. In order to meet these requirements, hash functions must adhere to key properties, including the inability to reverse the hashing process and the rarity of finding two distinct inputs that yield the same hash.

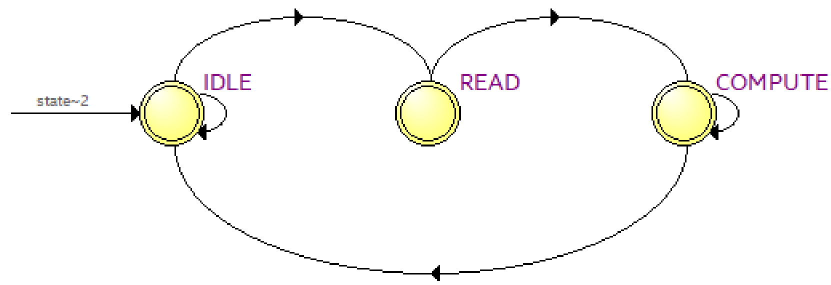
SHA-256 belongs to a family of standardized hash functions developed to fulfill these objectives. This particular hash function assumes that the input messages are no greater than 264 bits in length. The algorithm processes the data sequentially in blocks of 512 bits, ultimately producing a hash value with a fixed length of 256 bits.

The key properties of a cryptographic hash function are:

- Compression: A fixed size of the output regardless of the input data size.
- Avalanche Effect: A small change in the input will result in a drastic change in the output data.
- Determinism: The hash function will always produce the same output for the same input.
- Quick computation: The hash function should be able to produce the hash value efficiently.
- Pre-image resistance: Given a hash value, determining the original input should be computationally infeasible.
- Collision resistance: Finding two different inputs that produce the same hash value would be highly improbable.



SHA256 algorithm implemented by us



State Machine of our implementation

Explanation of the SHA256 algorithm that we have implemented

The SHA-256 algorithm uses an FSM to keep track of the device's state. The machine stays in the idle state until prompted to start, then initializes the hash values h0-h7 to preset constants and moves to the READ state. In this state, memory is read to a message array on every clock cycle, with the memory address being incremented each time, until all 20 words have been fed into the array.

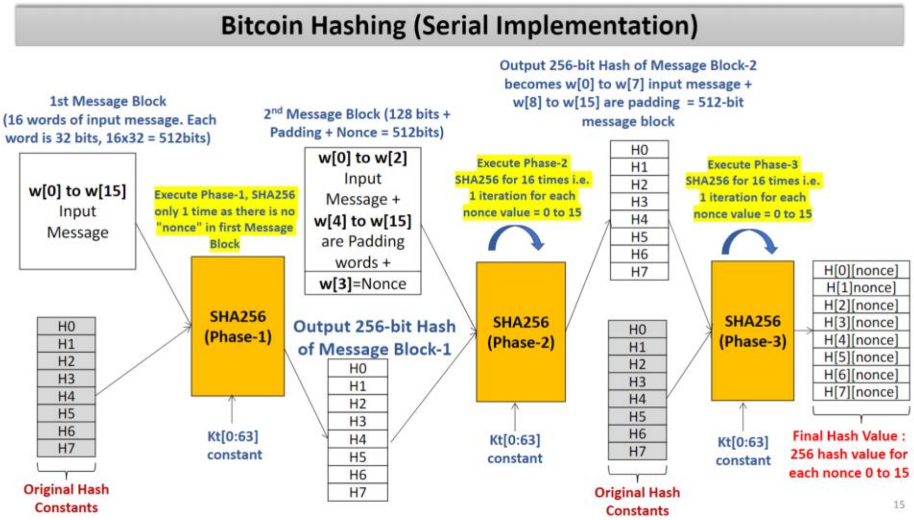
It then moves to the BLOCK state, which finds the next 512-bit block to process. If needed, it will add padding and the message length at the end in the format: 1, some number of 0s, and a 64-bit message length. Each block will then go through 64 rounds of the COMPUTE stage, where we use the sha256_op function to calculate the a-h variables, which we add to the h0-7 values after all 64 cycles. Once this has been performed on each block, the algorithm writes out the data stored in h0-h7 to memory one at a time in a similar repeating fashion to reading.

Additionally, we have implemented the following optimizations to our algorithm:

- Decreasing the number of multiplexers required to operate on the word array.
- Pre-computation of the repeated values that are used in critical paths.

How do we implement Bitcoin hashing using SHA256?

Bitcoin hashing uses the SHA-256 algorithm to keep track of and authenticate all Bitcoin transactions. When adding a new block to the existing blockchain (data of all the previous transactions), a signature is generated, which is the SHA-256 hash of the data from the previous block in the blockchain. Then the next block includes the new transactions along with the signature of the previous block. This keeps an accurate record of all previous transactions because of the cryptographic properties that the SHA-256 function has making it nearly impossible to change a block on the chain without changing the signature. Furthermore, blocks are only qualified to be added to the chain if they have 7 consecutive starting zeros in their signature. As such, part of the data, called the "nonce", is changed, and the result is hashed until the resulting hash satisfies this property; this is the best-known algorithm to find a nonce with a proper hash due to the cryptographic properties of the SHA-256 function. Once a nonce is found that generates a proper signature, the resulting block may be added to the Bitcoin blockchain.



Basic Bitcoin hashing implementation – Serial mode

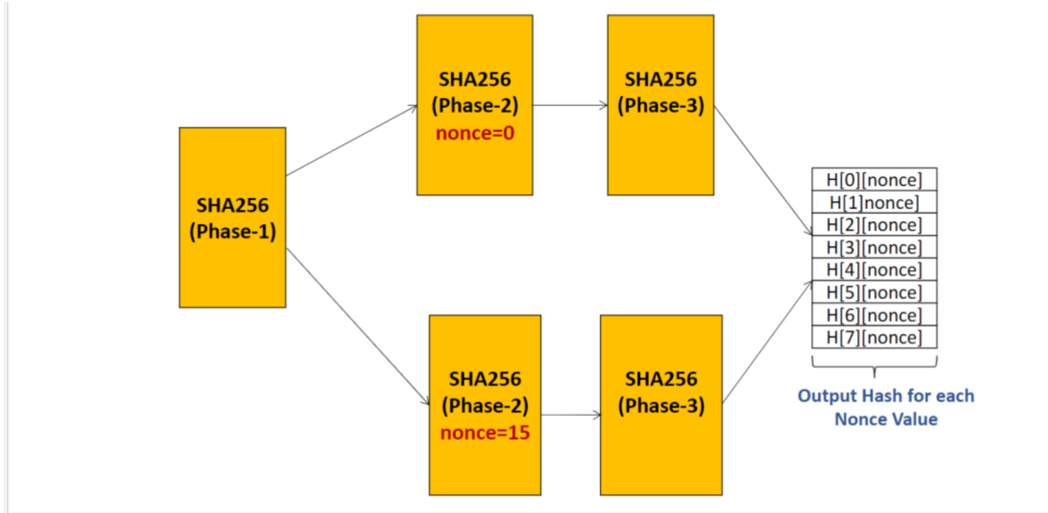
Explanation of the Bitcoin hashing algorithm that we have implemented

The bitcoin hashing algorithm uses a modified SHA-256 algorithm from part 1, which has been optimized for this application, and calculates the hash value of each message, with different associated nonces, in parallel. When prompted to start, the machine moves from the IDLE state to the INIT_READ state, which reads the message from memory serially. Once that finishes, the machine moves to PHASE_ONE_READ, which corresponds to the 1st block of the message being processed. The message block and the initial hash constants are loaded into the SHA-256 instance. The module then enters the PHASE_ONE_CALCULATE state, where the SHA-256 does the usual calculations and outputs the hash values for phase two.

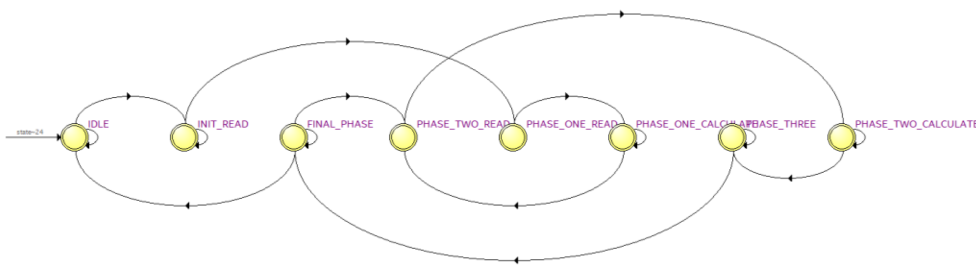
During PHASE_TWO_READ, the new hash values from phase 1 are loaded into all the SHA-256 instances, of which there is a number preset by a parameter. The variable phase_iter keeps track of how many times the algorithm has run phase 2, which depends on the number of instances of the SHA-256 module. During PHASE_TWO_CALCULATE, each instance's hash values are loaded with the same array input, except for the fourth entry, which is the nonce. Each instance uses a unique nonce that updates each phase cycle.

After these calculations, it moves onto PHASE_THREE, which loads in the original hash constants and uses the output of phase 2 as the message to hash (with some padding); specifically, the new input array has the 8 outputs of phase two at [0:7], [8] is 32'h80000000, [15] is 32'd256 (message size padding), and the rest are zeros. The output h0 for each SHA-256 instance is saved to the final_out_h0 array. After this, the machine moves to FINAL_PHASE.

In FINAL_PHASE, the h0 values are serially written to memory. If there are still nonces that need to be calculated, then the machine moves back to PHASE_TWO_READ. Otherwise, the algorithm is done and moves back to IDLE, marking its completion with the done signal set to high.



Parallel Implementation of the SHA256 Algorithm



State Machine of our implementation

Transcript of the SHA256 Implementation

```
VSIM 5> run -all
# --------
# MESSAGE:
# --------
# 01234567
# 02468ace
# 048d159c
# 091a2b38
# 12345670
# 2468ace0
# 48d159c0
# 91a2b380
# 23456701
# 468ace02
# 8d159c04
# 1a2b3809
# 34567012
# 68ace024
# d159c048
# a2b38091
# 45670123
# 8ace0246
# 159c048d
# 00000000
# ***************************
#
# ---------------------
# COMPARE HASH RESULTS:
# ---------------------
# Correct H[0] = bdd2fbd9 Your H[0] = bdd2fbd9
# Correct H[1] = 42623974 Your H[1] = 42623974
# Correct H[2] = bf129635 Your H[2] = bf129635
# Correct H[3] = 937c5107 Your H[3] = 937c5107
# Correct H[4] = f09b6e9e Your H[4] = f09b6e9e
# Correct H[5] = 708eb28b Your H[5] = 708eb28b
# Correct H[6] = 0318d121 Your H[6] = 0318d121
# Correct H[7] = 85eca921 Your H[7] = 85eca921
# ***************************
#
# CONGRATULATIONS! All your hash results are correct!
#
# Total number of cycles:         166
#
#
# ***************************
#
# ** Note: $stop    : C:/Users/andre/OneDrive/Documents/ECE 111/Final/Project_Files/simplified_sha256/tb_simplified_sha256.sv(262)
#    Time: 3370 ps  Iteration: 2  Instance: /tb_simplified_sha256
```

Waveform of the SHA256 Implementation

[Attached at the end in Appendix B]

Transcript of the Bitcoin Hashing Implementation

```
VSIM 6> run -all
# ---------------
# 19 WORD HEADER:
# ---------------
# 01234567
# 02468ace
# 048d159c
# 091a2b38
# 12345670
# 2468ace0
# 48d159c0
# 91a2b380
# 23456701
# 468ace02
# 8d159c04
# 1a2b3809
# 34567012
# 68ace024
# d159c048
# a2b38091
# 45670123
# 8ace0246
# 159c048d
# ***************************
#
# ---------------------
# COMPARE HASH RESULTS:
# ---------------------
# Correct H0[ 0] = 7106973a Your H0[ 0] = 7106973a
# Correct H0[ 1] = 6e66eea7 Your H0[ 1] = 6e66eea7
# Correct H0[ 2] = fbef64dc Your H0[ 2] = fbef64dc
# Correct H0[ 3] = 0888a18c Your H0[ 3] = 0888a18c
# Correct H0[ 4] = 9642d5aa Your H0[ 4] = 9642d5aa
# Correct H0[ 5] = 2ab6af8b Your H0[ 5] = 2ab6af8b
# Correct H0[ 6] = 24259d8c Your H0[ 6] = 24259d8c
# Correct H0[ 7] = ffb9bcd9 Your H0[ 7] = ffb9bcd9
# Correct H0[ 8] = 642138c9 Your H0[ 8] = 642138c9
# Correct H0[ 9] = 054cafc7 Your H0[ 9] = 054cafc7
# Correct H0[10] = 78251a17 Your H0[10] = 78251a17
# Correct H0[11] = af8c8f22 Your H0[11] = af8c8f22
# Correct H0[12] = d7a79ef8 Your H0[12] = d7a79ef8
# Correct H0[13] = c7d10c84 Your H0[13] = c7d10c84
# Correct H0[14] = 9537acfd Your H0[14] = 9537acfd
# Correct H0[15] = c1e4c72b Your H0[15] = c1e4c72b
# ***************************
#
# CONGRATULATIONS! All your hash results are correct!
#
# Total number of cycles:         249
#
#
# ***************************
#
# ** Note: $stop    : C:/Users/andre/OneDrive/Documents/ECE 111/Final/Project_Files/bitcoin_hash/tb_bitcoin_hash.sv(334)
#    Time: 5030 ps  Iteration: 2  Instance: /tb_bitcoin_hash
```

Waveform of the Bitcoin hashing Implementation

[Attached at the end in Appendix A]

For Bitcoin Hashing Algorithm Implementation

Resource Usage

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	13875
2		
3	Combinational ALUT usage for logic	14252
1	-- 7 input functions	0
2	-- 6 input functions	691
3	-- 5 input functions	2297
4	-- 4 input functions	120
5	-- <=3 input functions	11144
4		
5	Dedicated logic registers	23134
6		
7	I/O pins	118
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	23135
13	Total fan-out	153236
14	Average fan-out	4.07

Maximum Clock Speed

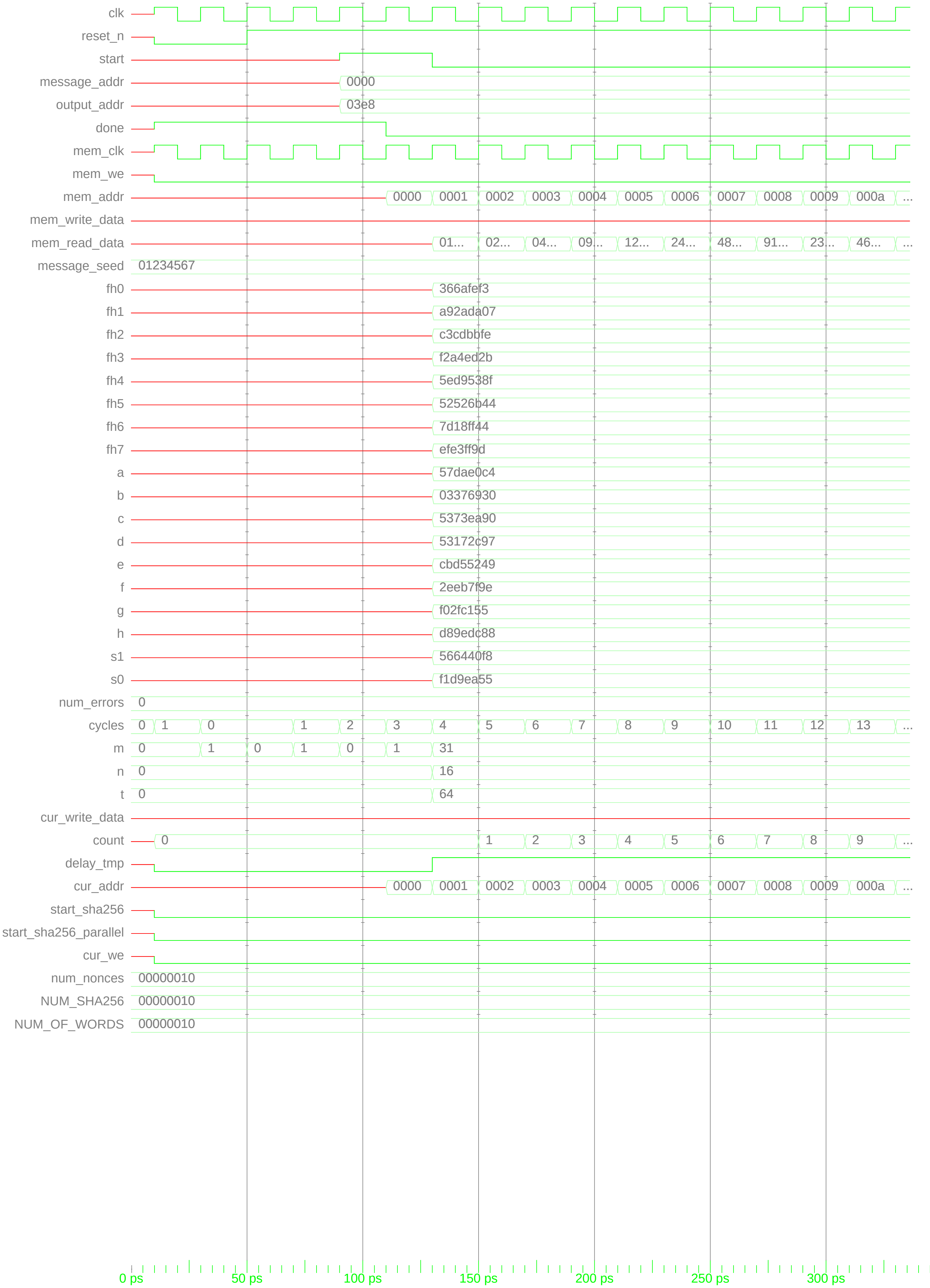
	Fmax	Restricted Fmax	Clock Name
1	167.08 MHz	167.08 MHz	clk

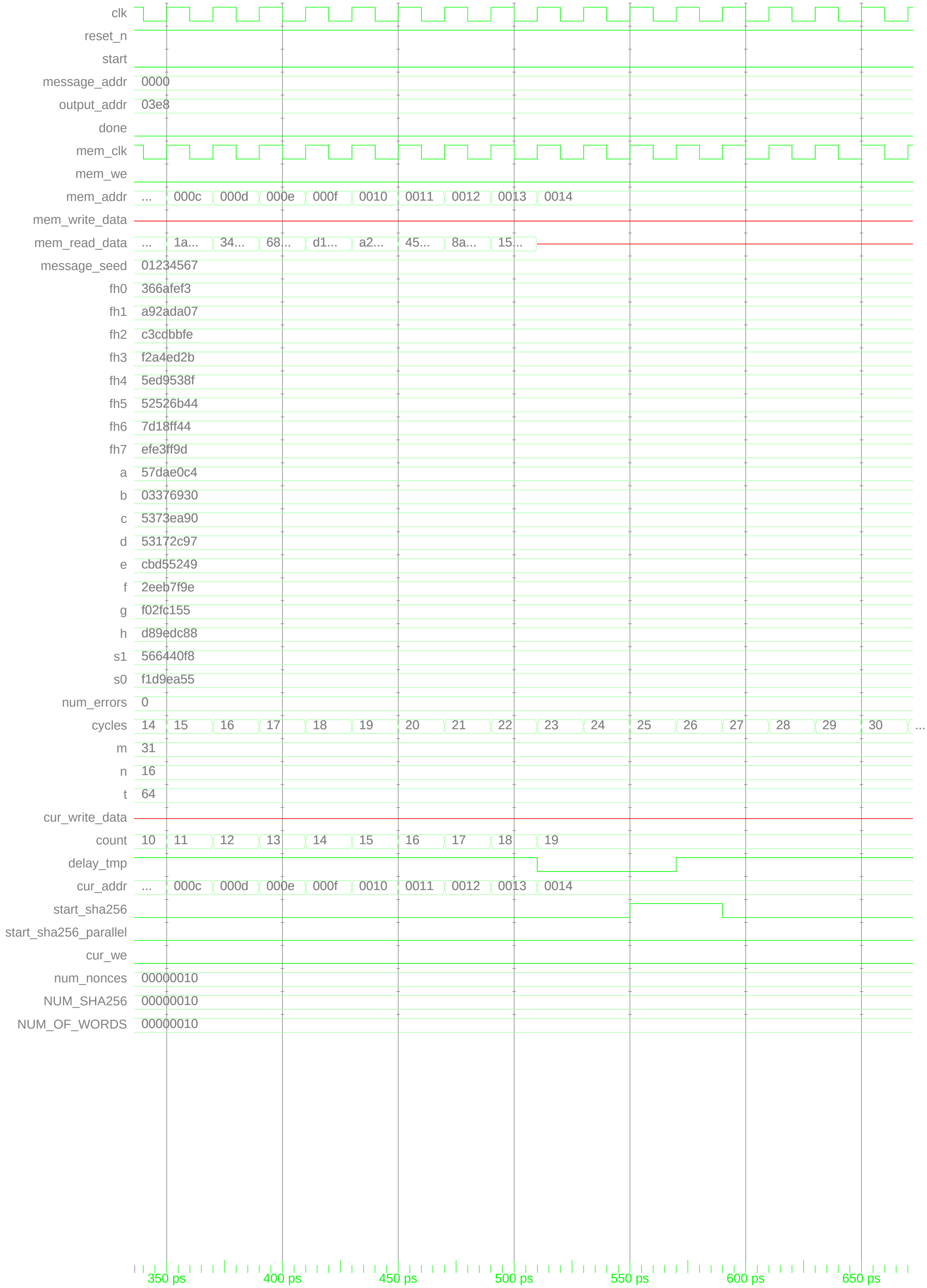
Fitter Report

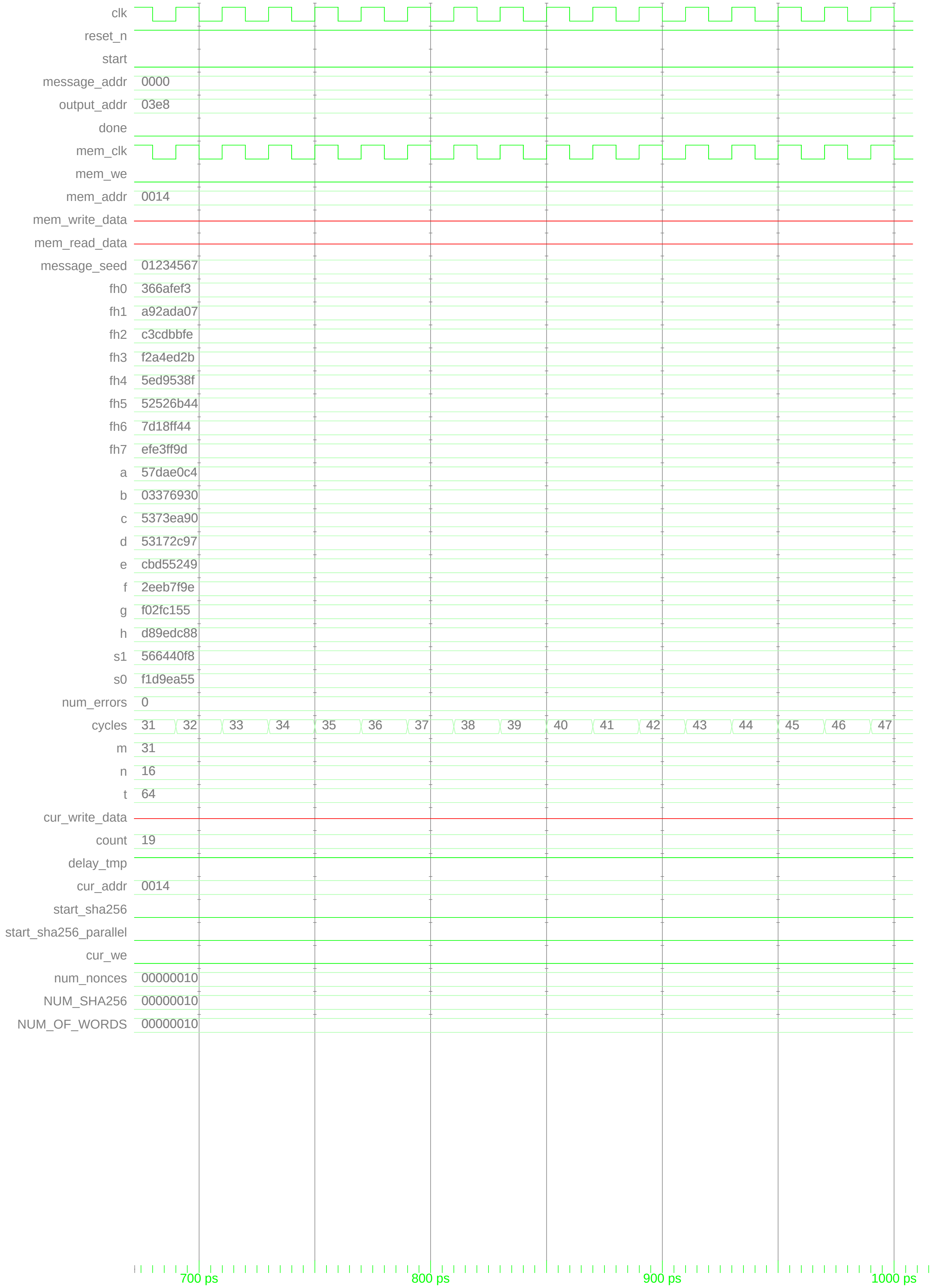
Fitter Status	Successful - Sat Jun 10 18:02:58 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	bitcoin_hash
Top-level Entity Name	bitcoin_hash
Family	Arria II GX
Device	EP2AGX45DF29I5
Timing Models	Final
Logic utilization	87 %
Total registers	19396
Total pins	118 / 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 %)
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 2 (0 %)

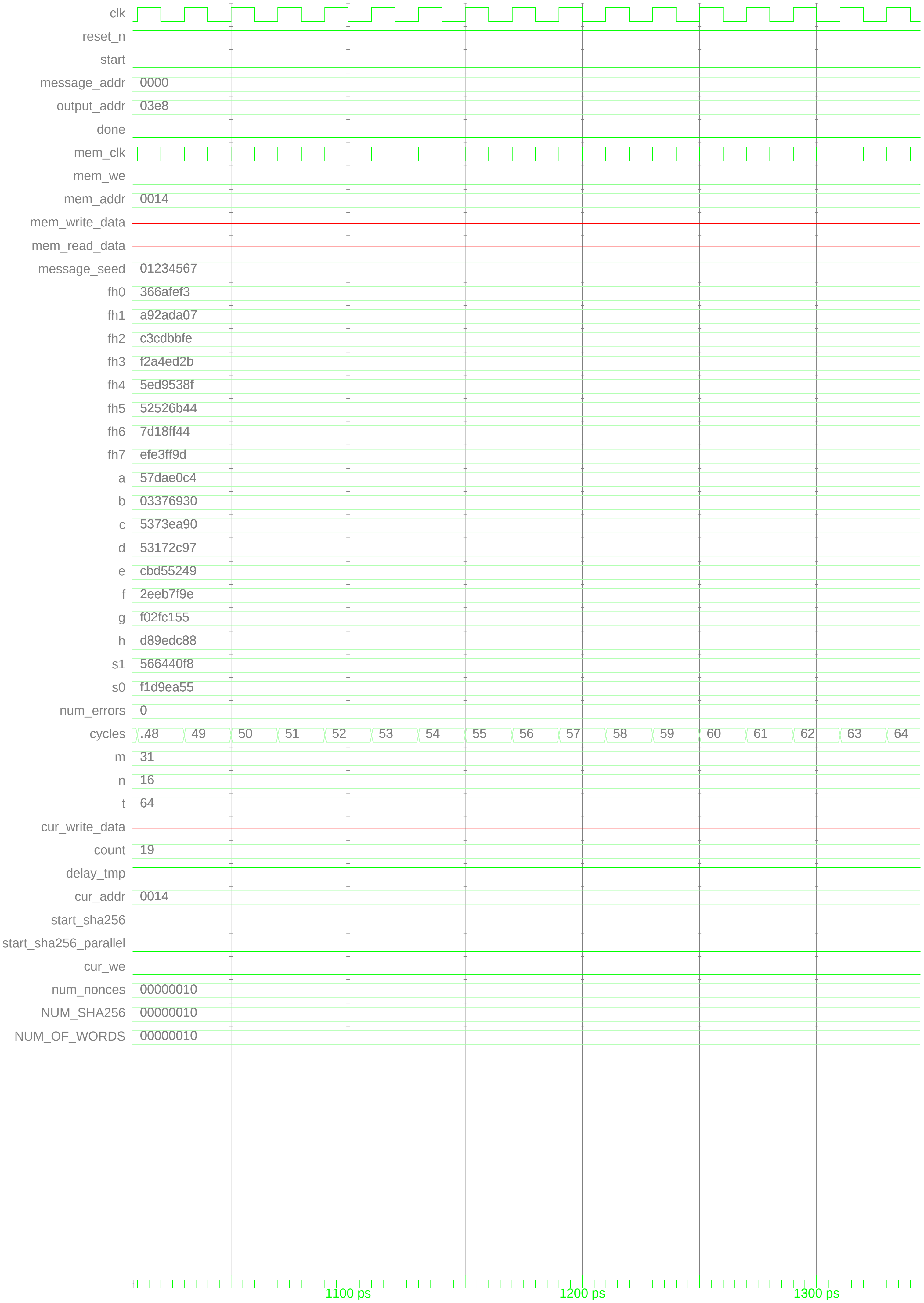
Appendix A

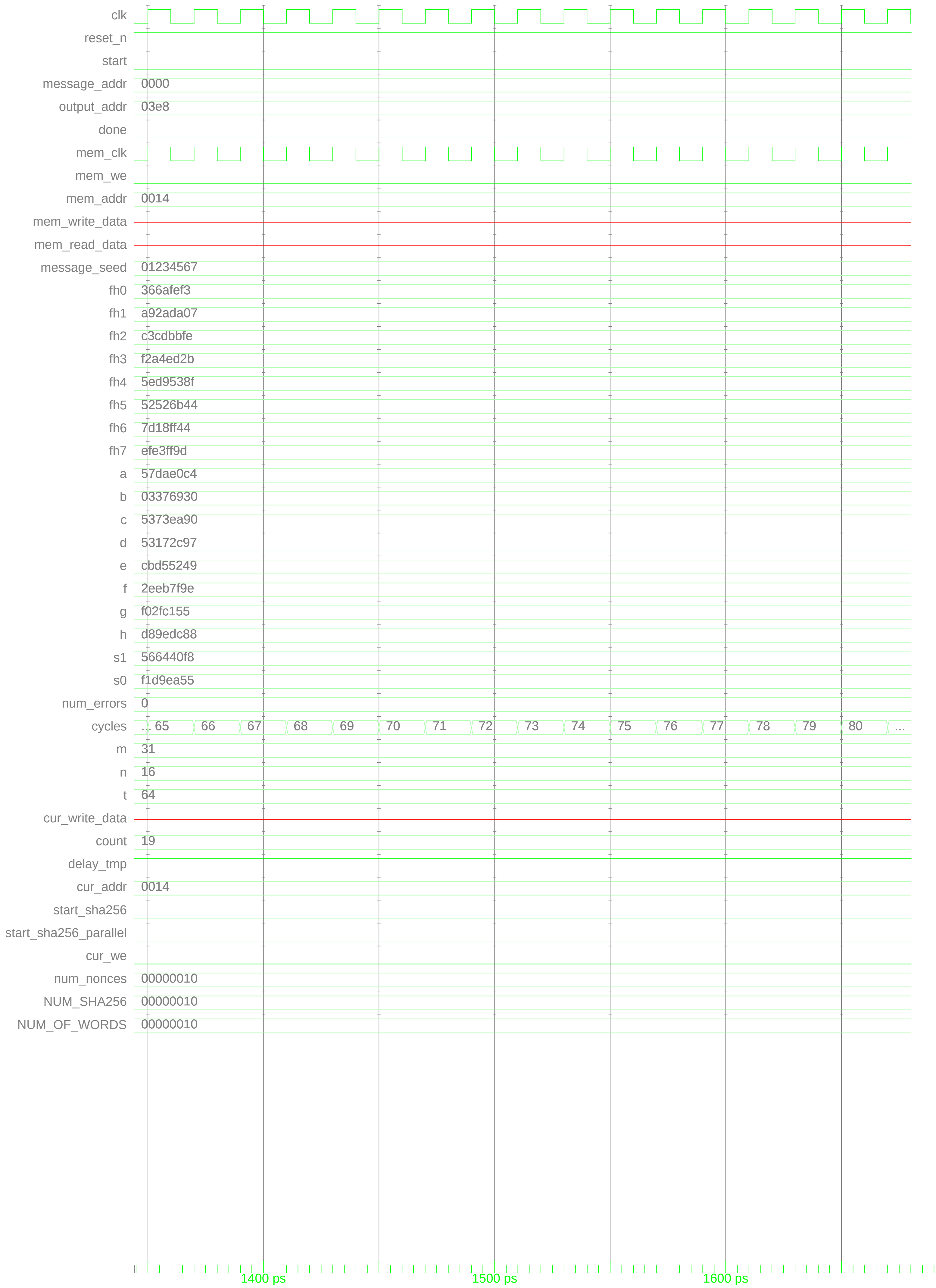
[Waveform for Bitcoin hashing Algorithm]


clk
reset_n
start
message_addr
0000
output_addr
03e8
done
mem_clk
mem_we
mem_addr
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000a
mem_write_data
mem_read_data
01...
02...
04...
09...
12...
24...
48...
91...
23...
46...
message_seed
01234567
fh0
366afef3
fh1
a92ada07
fh2
c3cdbbfe
fh3
f2a4ed2b
fh4
5ed9538f
fh5
52526b44
fh6
7d18ff44
fh7
efe3ff9d
a
57dae0c4
b
03376930
c
5373ea90
d
53172c97
e
cbd55249
f
2eeb7f9e
g
f02fc155
h
d89edc88
s1
566440f8
s0
f1d9ea55
num_errors
cycles
m
31
n
16
t
64
cur_write_data
count
delay_tmp
cur_addr
start_sha256
start_sha256_parallel
cur_we
num_nonces
00000010
NUM_SHA256
00000010
NUM_OF_WORDS
00000010
0 ps
50 ps
100 ps
150 ps
200 ps
250 ps
300 ps

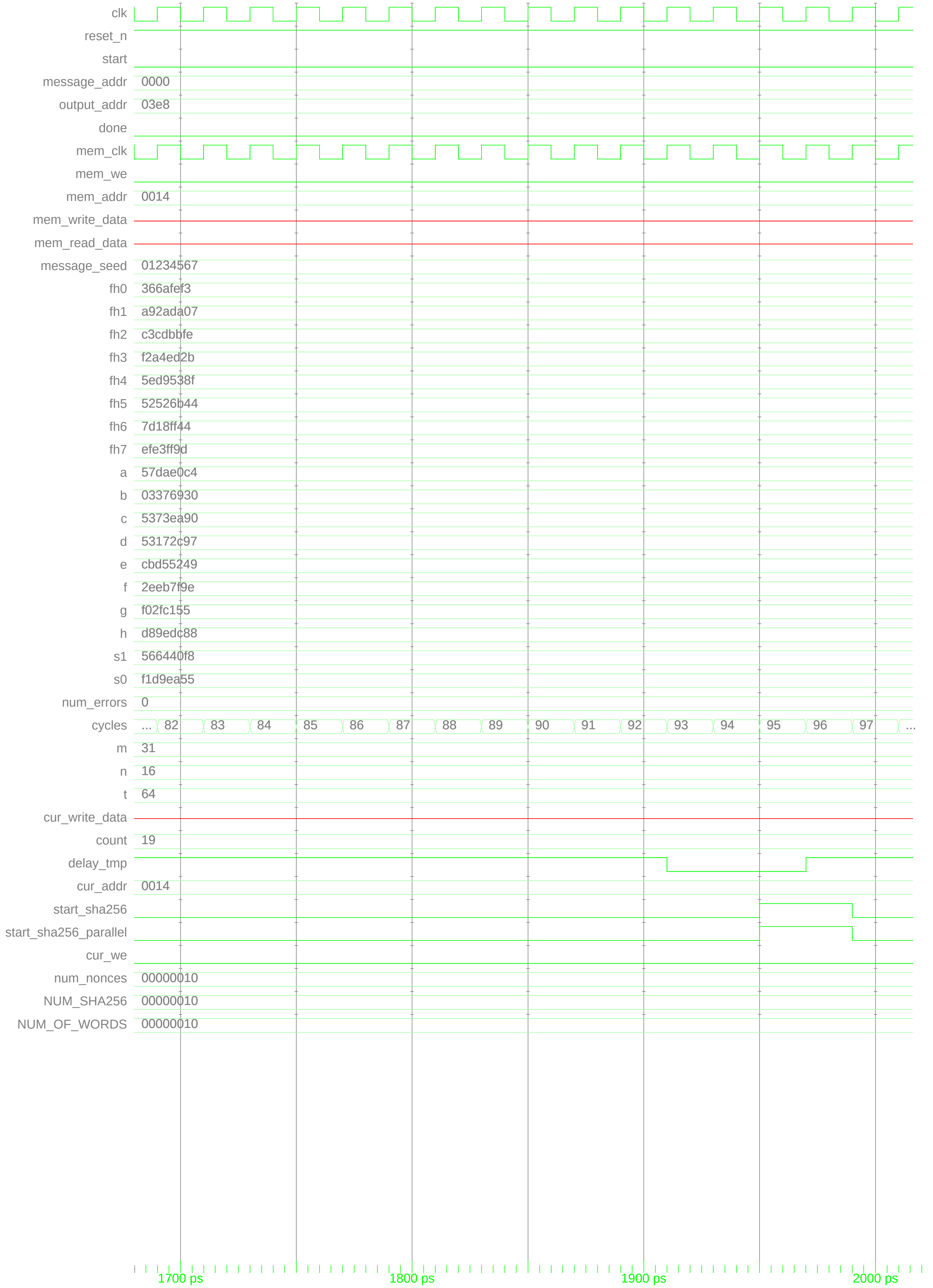

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr ... 000c 000d 000e 000f 0010 0011 0012 0013 0014
mem_write_data
mem_read_data ... 1a... 34... 68... d1... a2... 45... 8a... 15...
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 ...
m 31
n 16
t 64
cur_write_data
count 10 11 12 13 14 15 16 17 18 19
delay_tmp
cur_addr ... 000c 000d 000e 000f 0010 0011 0012 0013 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
350 ps
400 ps
450 ps
500 ps
550 ps
600 ps
650 ps

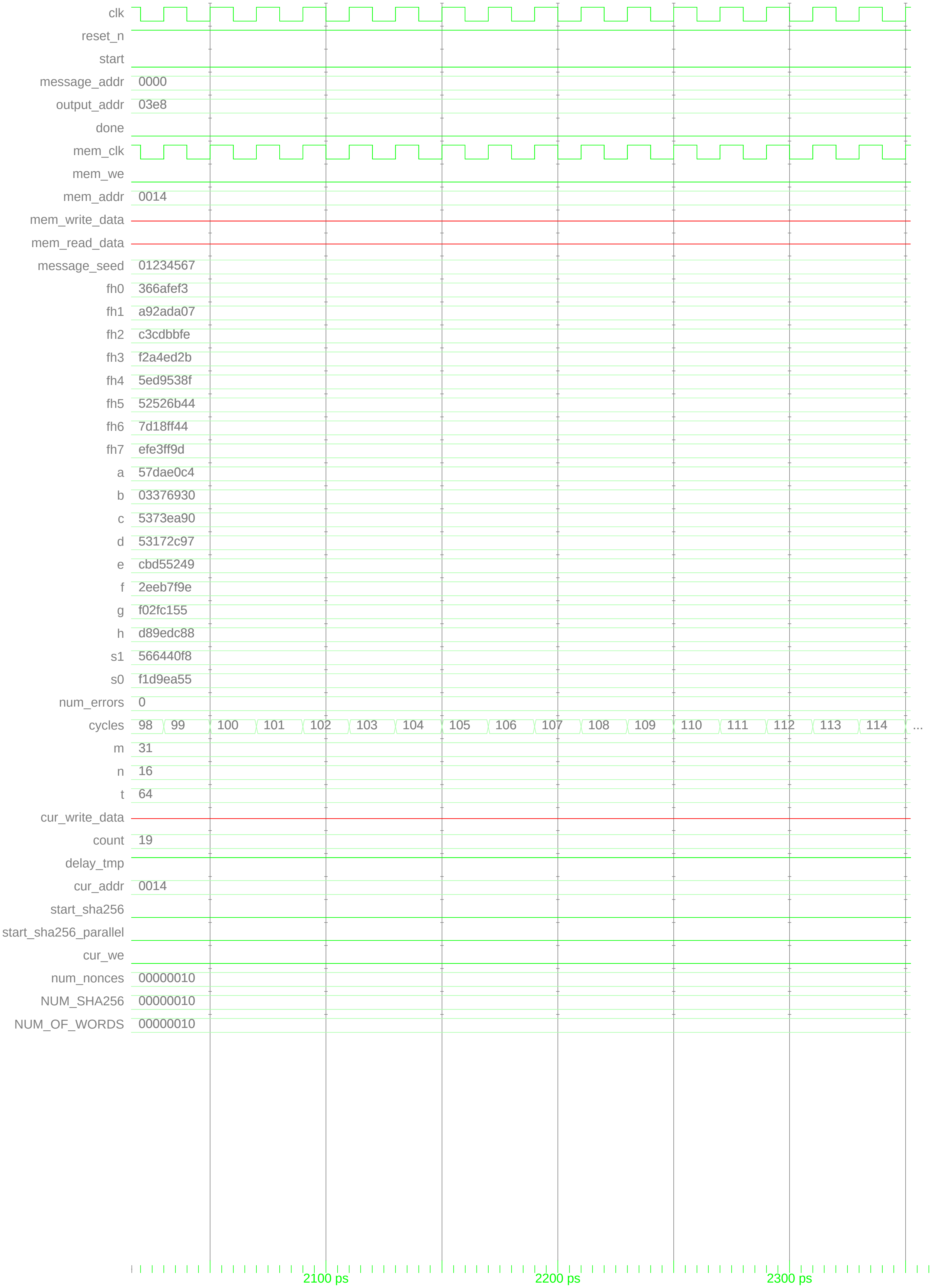

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
700 ps
800 ps
900 ps
1000 ps

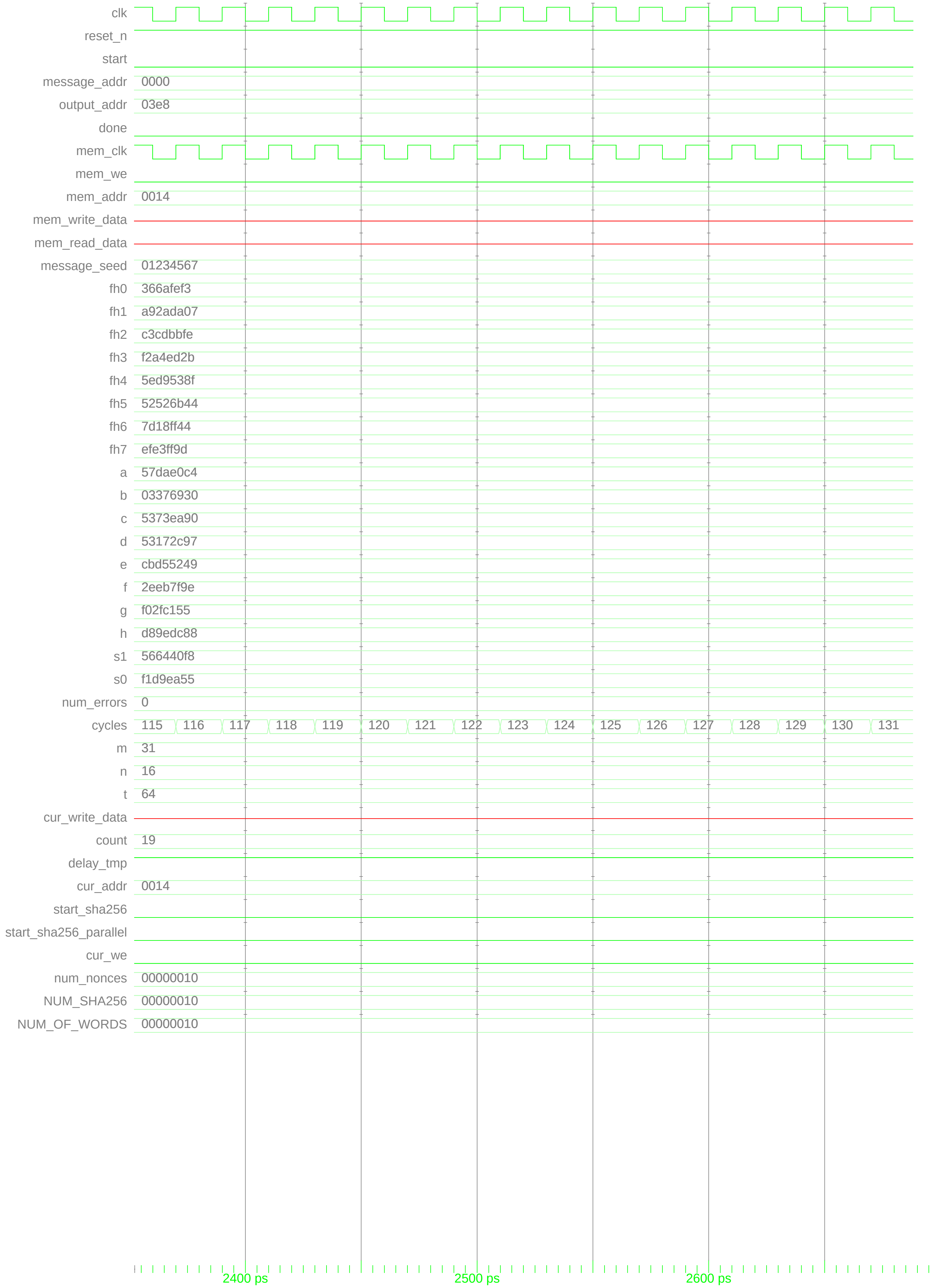

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles .48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
1100 ps
1200 ps
1300 ps

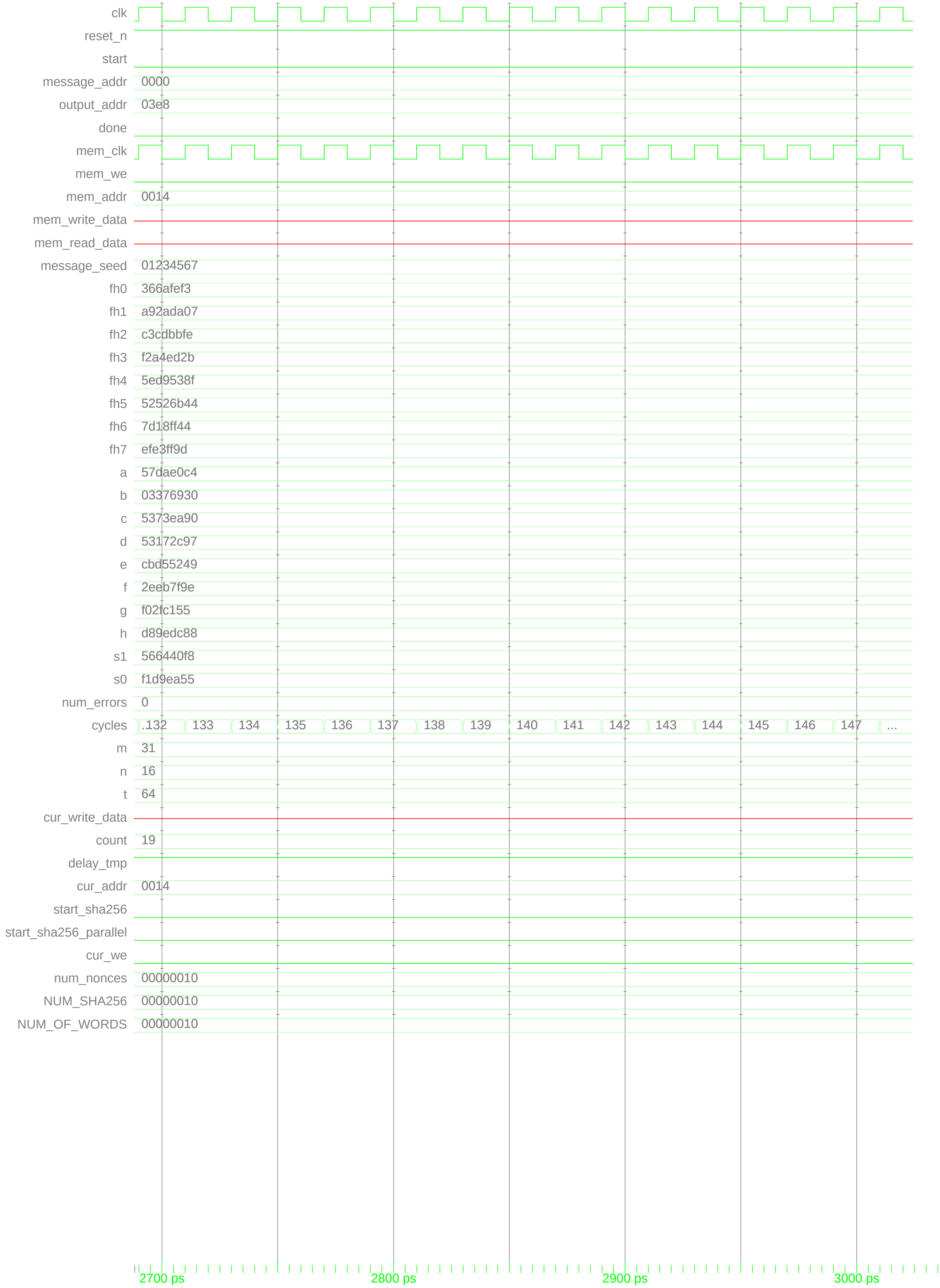


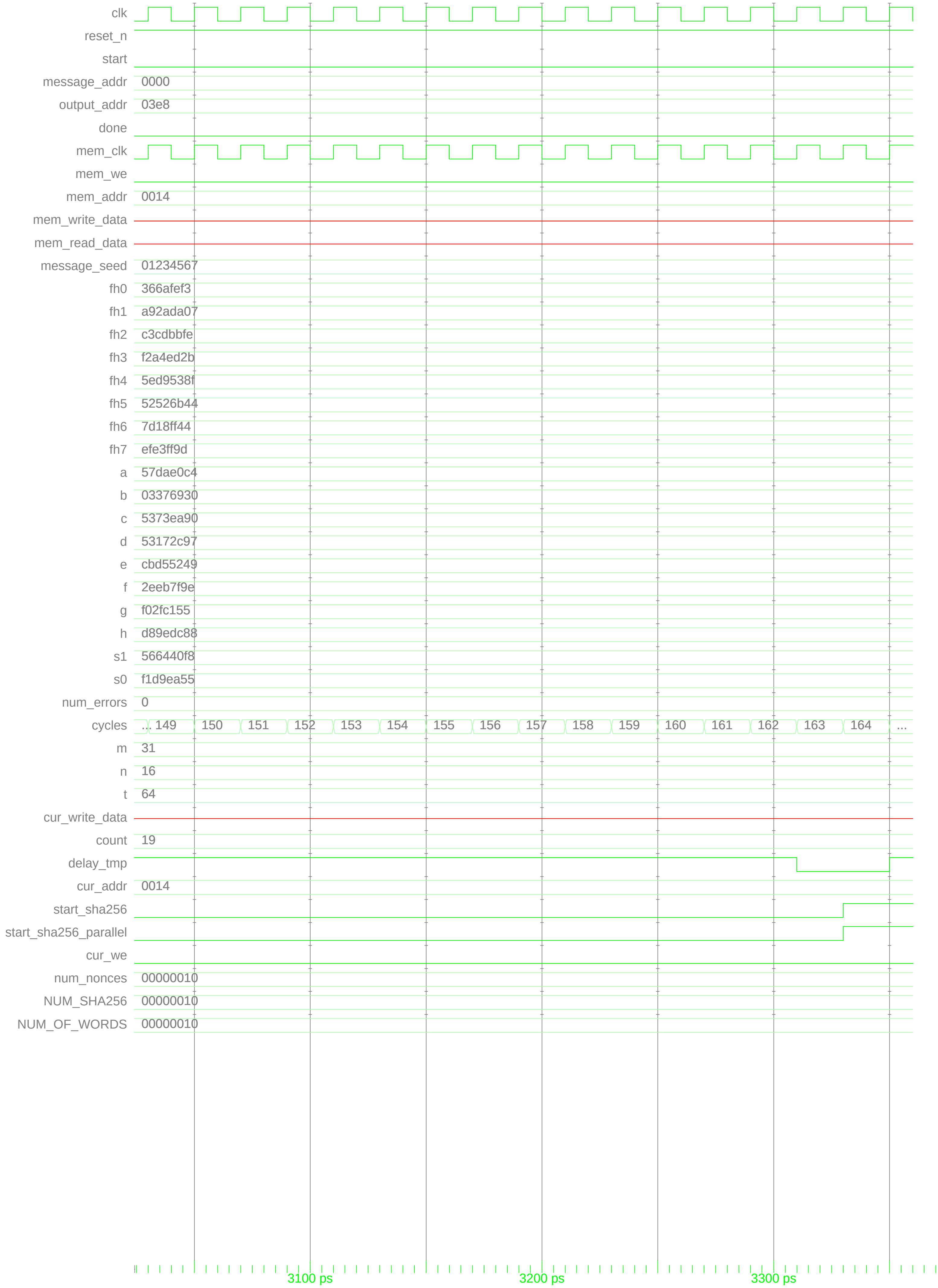
clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
1400 ps
1500 ps
1600 ps

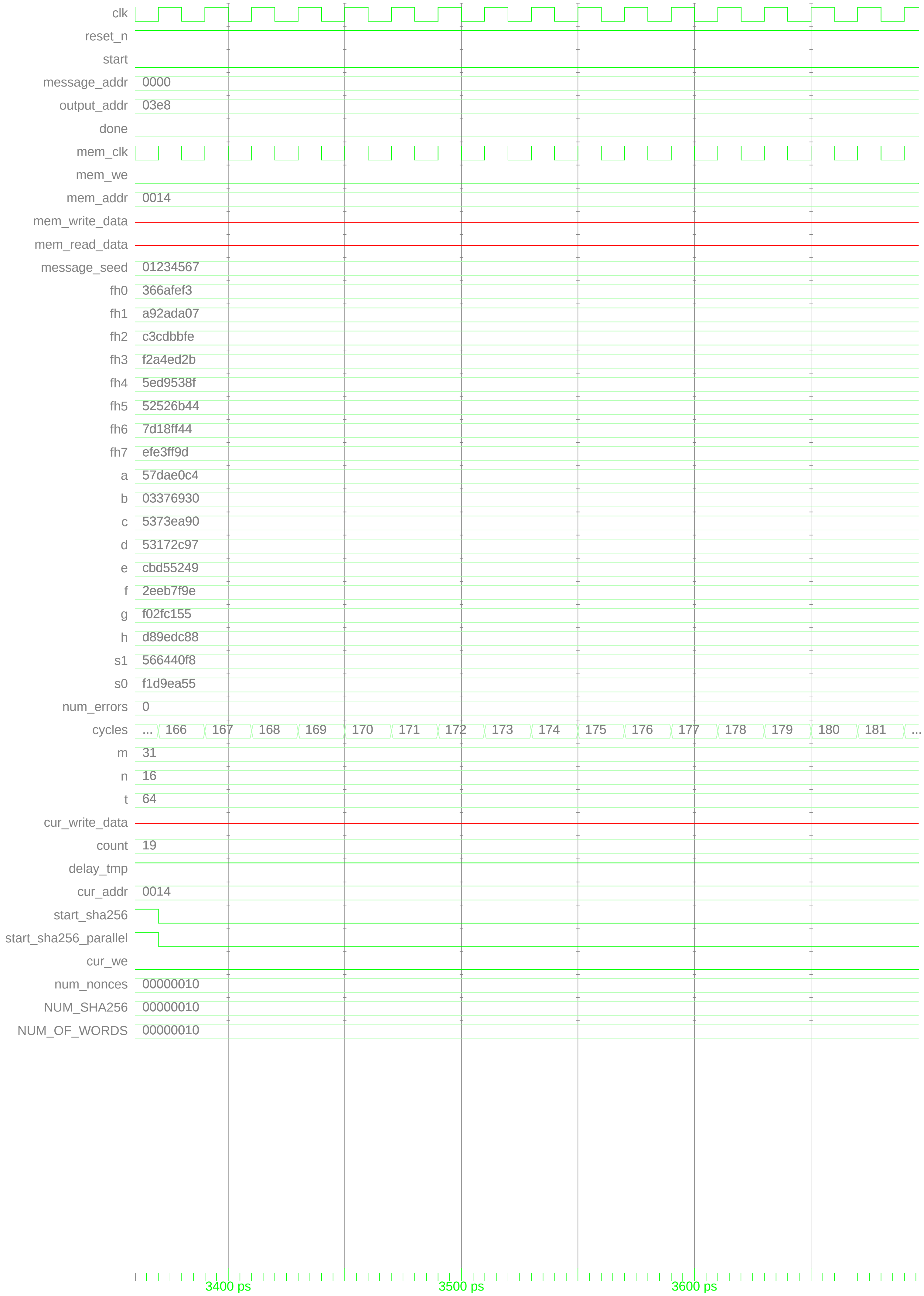

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
1700 ps
1800 ps
1900 ps
2000 ps

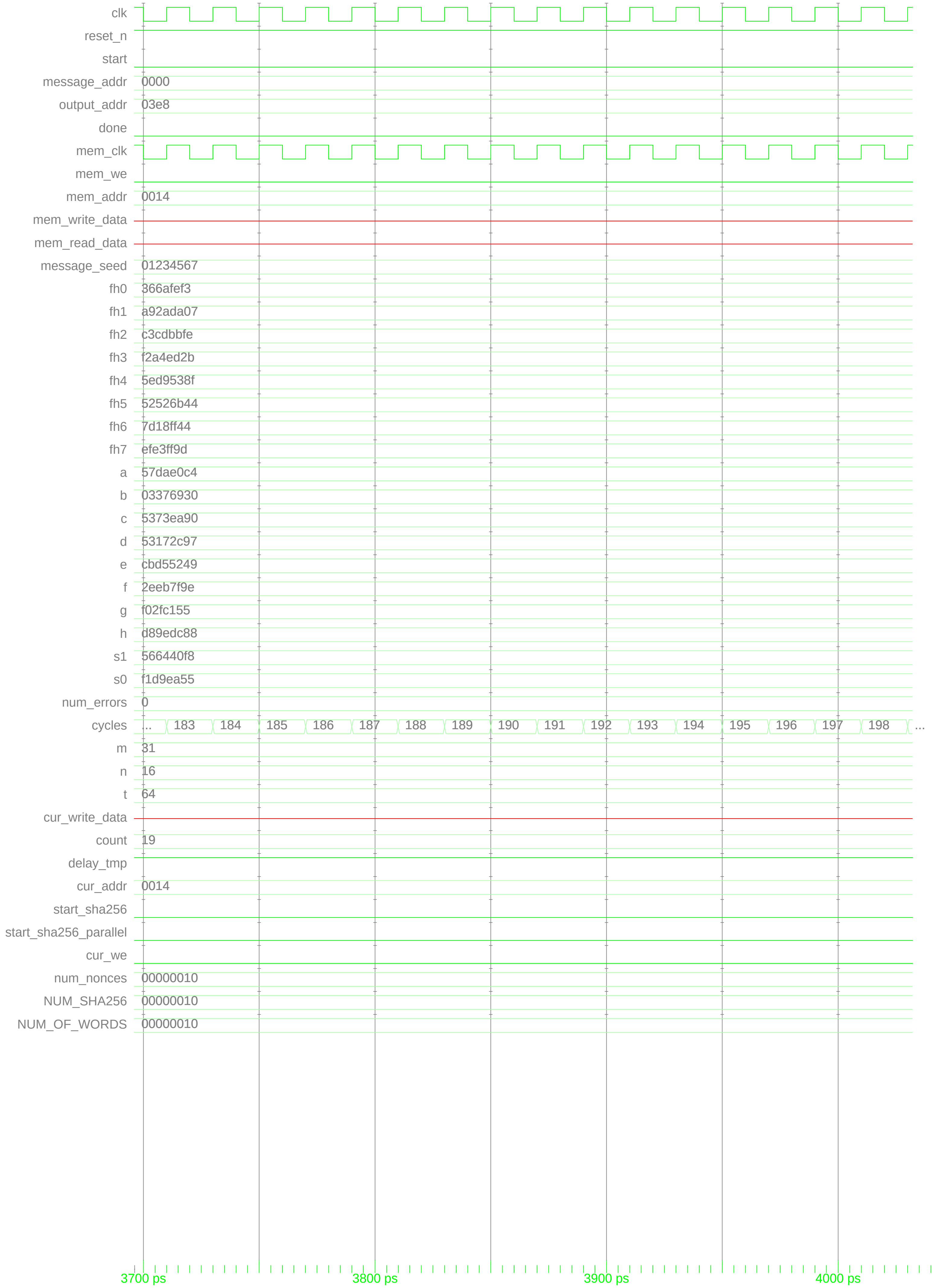

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
2100 ps
2200 ps
2300 ps

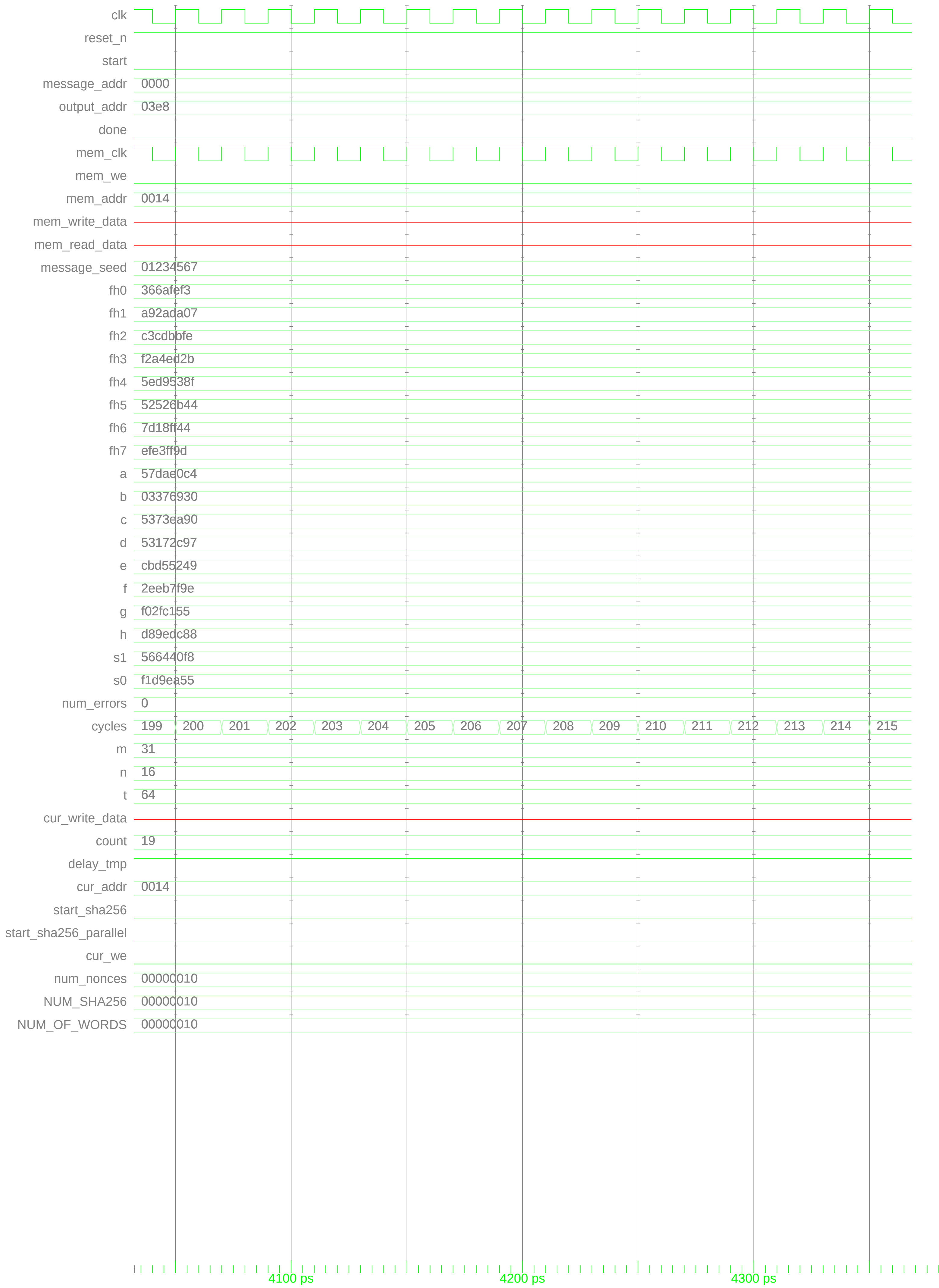

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
2400 ps
2500 ps
2600 ps

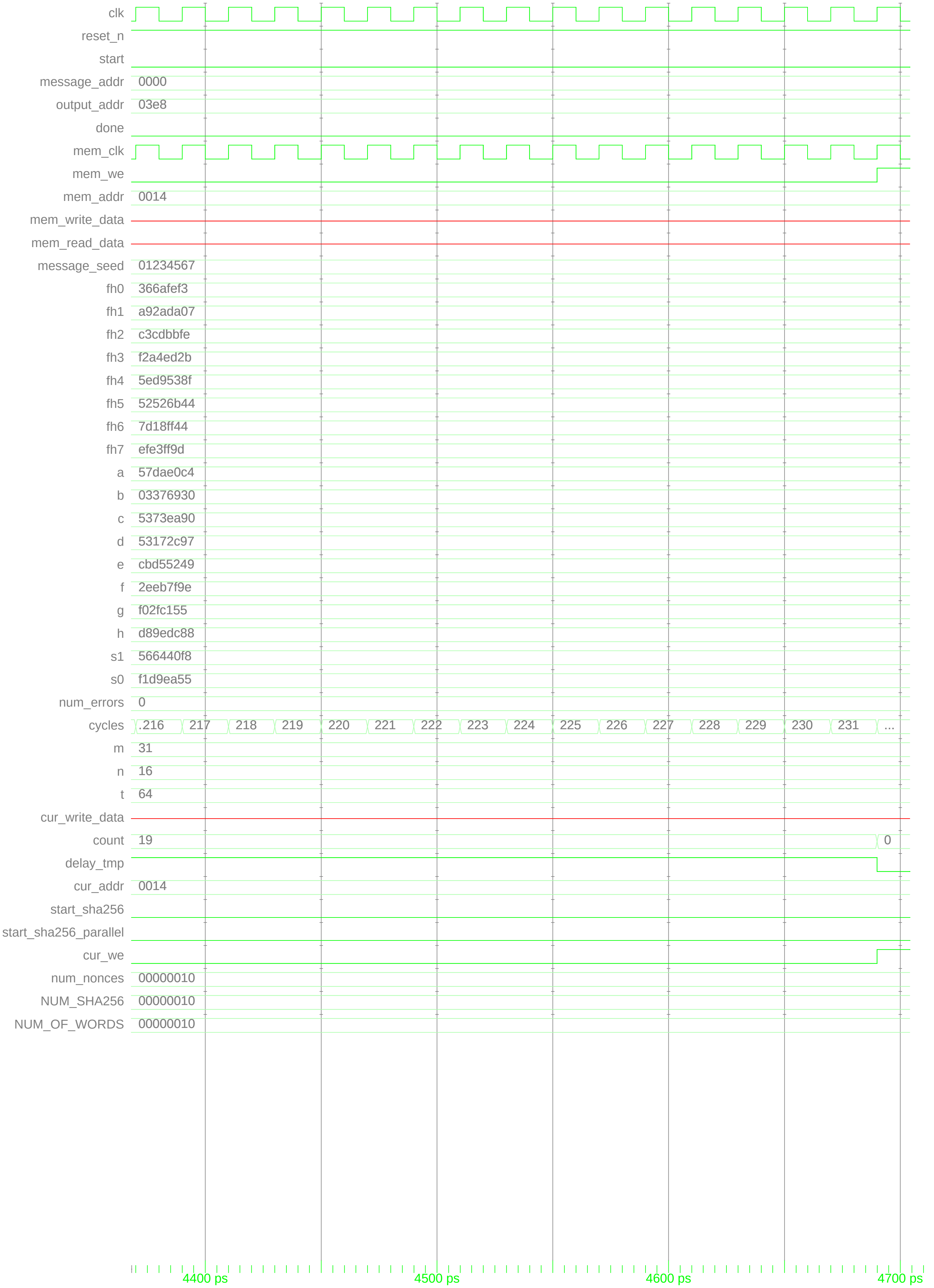

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles .132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
2700 ps
2800 ps
2900 ps
3000 ps

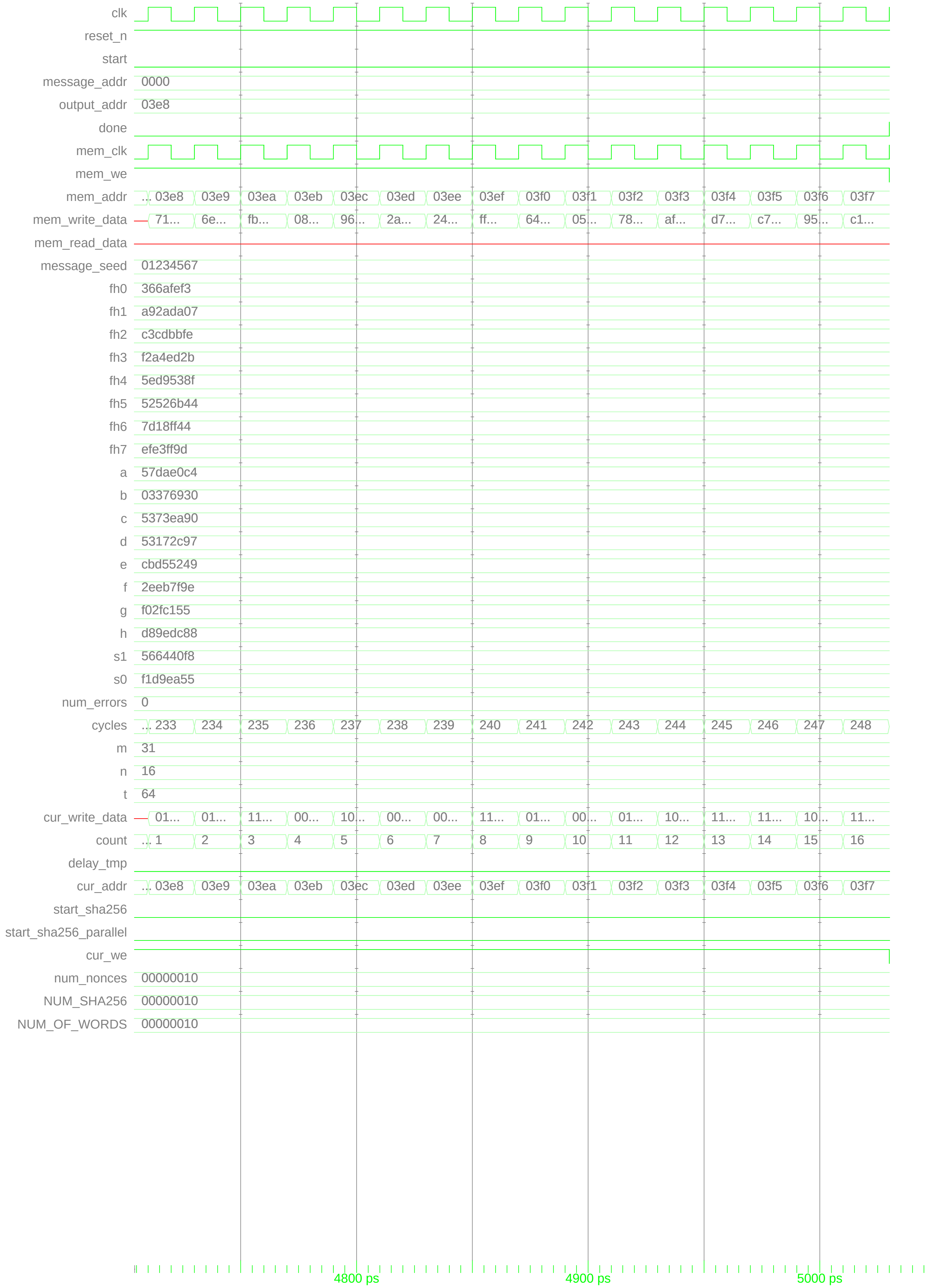

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
3100 ps
3200 ps
3300 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
3400 ps
3500 ps
3600 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 ...
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
3700 ps
3800 ps
3900 ps
4000 ps

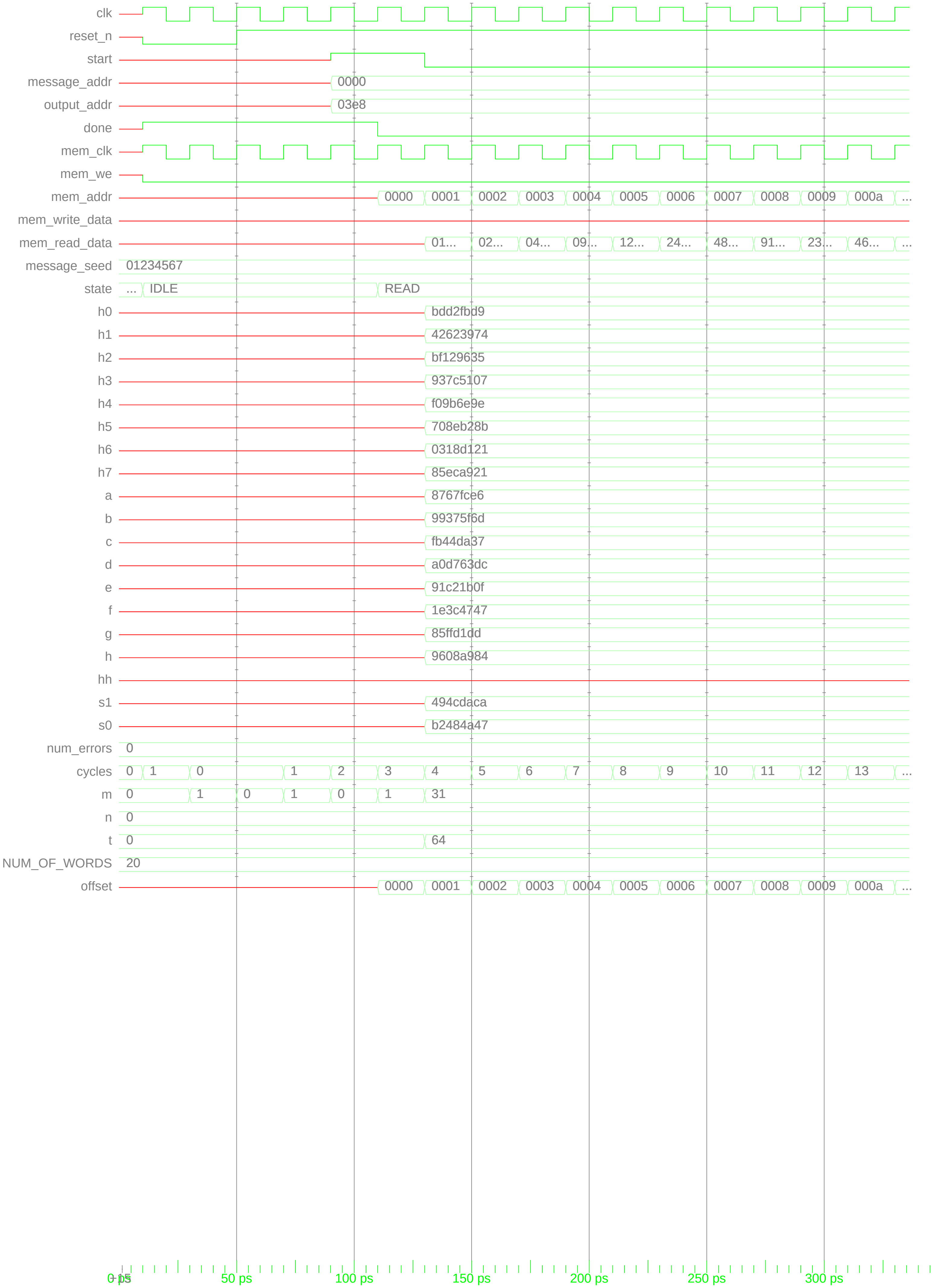

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
m 31
n 16
t 64
cur_write_data
count 19
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
4100 ps
4200 ps
4300 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0014
mem_write_data
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles .216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 ...
m 31
n 16
t 64
cur_write_data
count 19 0
delay_tmp
cur_addr 0014
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
4400 ps
4500 ps
4600 ps
4700 ps

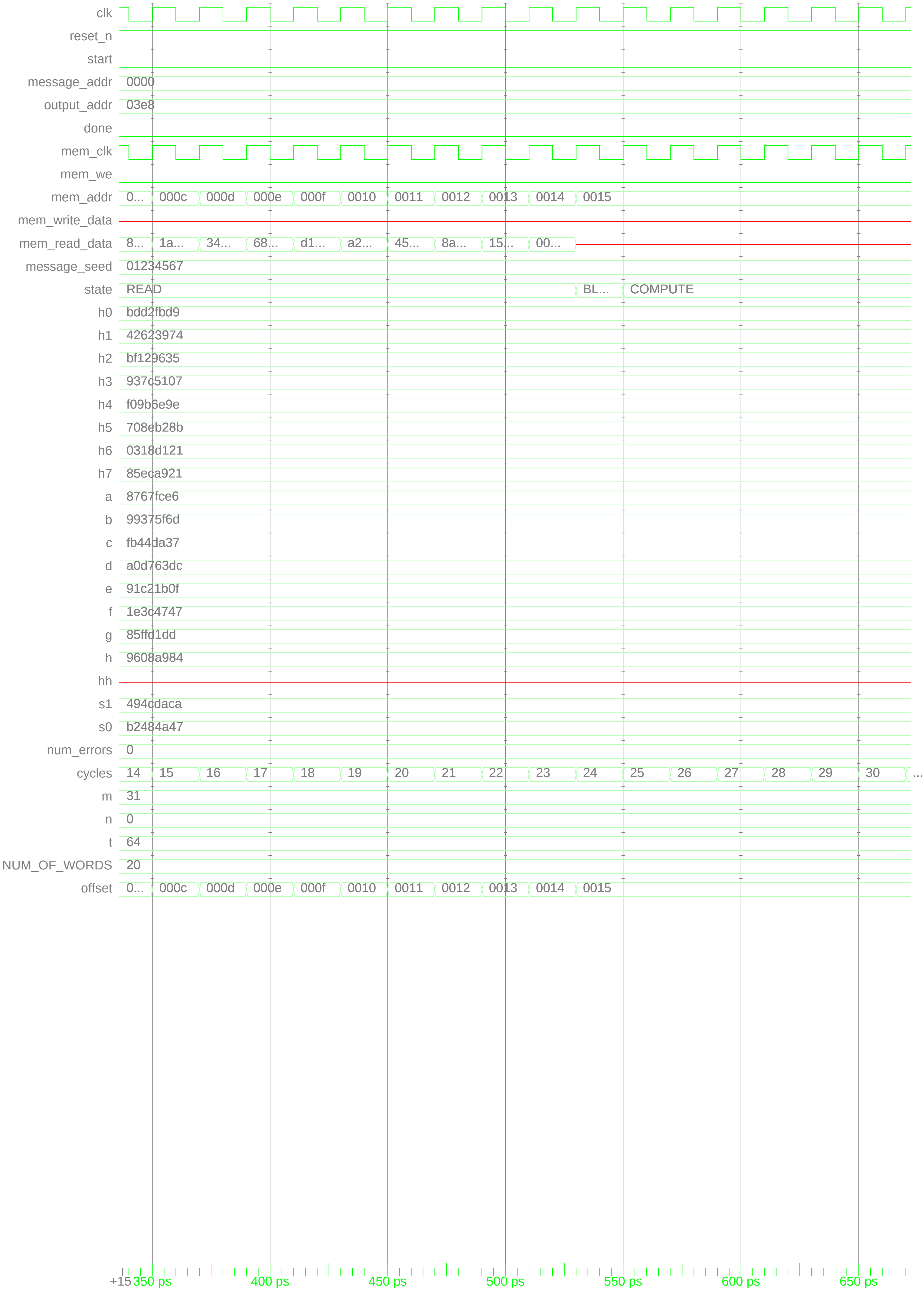

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr ... 03e8 03e9 03ea 03eb 03ec 03ed 03ee 03ef 03f0 03f1 03f2 03f3 03f4 03f5 03f6 03f7
mem_write_data 71... 6e... fb... 08... 96... 2a... 24... ff... 64... 05... 78... af... d7... c7... 95... c1...
mem_read_data
message_seed 01234567
fh0 366afef3
fh1 a92ada07
fh2 c3cdbbfe
fh3 f2a4ed2b
fh4 5ed9538f
fh5 52526b44
fh6 7d18ff44
fh7 efe3ff9d
a 57dae0c4
b 03376930
c 5373ea90
d 53172c97
e cbd55249
f 2eeb7f9e
g f02fc155
h d89edc88
s1 566440f8
s0 f1d9ea55
num_errors 0
cycles ... 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248
m 31
n 16
t 64
cur_write_data 01... 01... 11... 00... 10... 00... 00... 11... 01... 00... 01... 10... 11... 11... 10... 11...
count ... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
delay_tmp
cur_addr ... 03e8 03e9 03ea 03eb 03ec 03ed 03ee 03ef 03f0 03f1 03f2 03f3 03f4 03f5 03f6 03f7
start_sha256
start_sha256_parallel
cur_we
num_nonces 00000010
NUM_SHA256 00000010
NUM_OF_WORDS 00000010
4800 ps
4900 ps
5000 ps

Appendix B

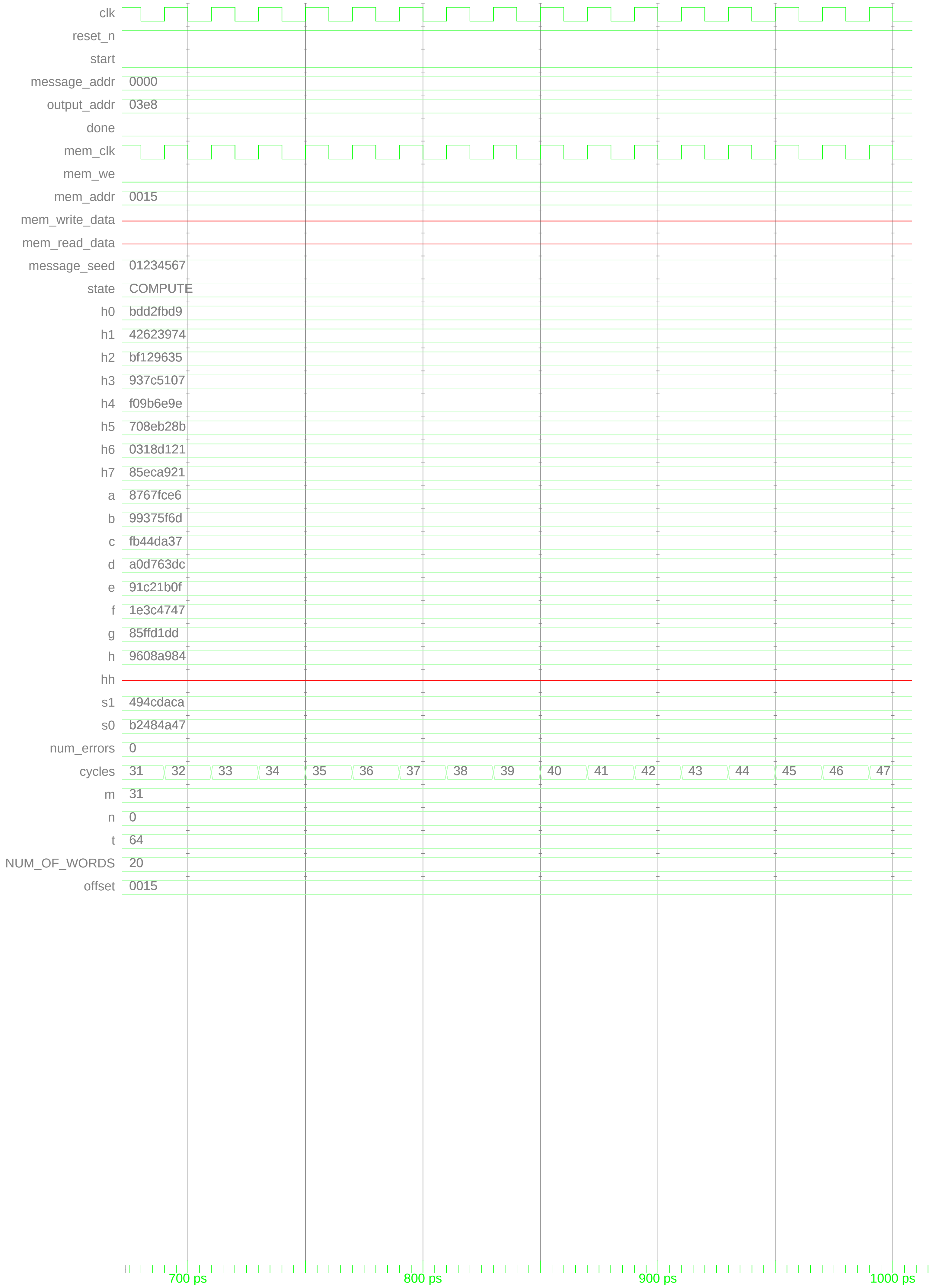
[Waveform for SHA256 Algorithm]

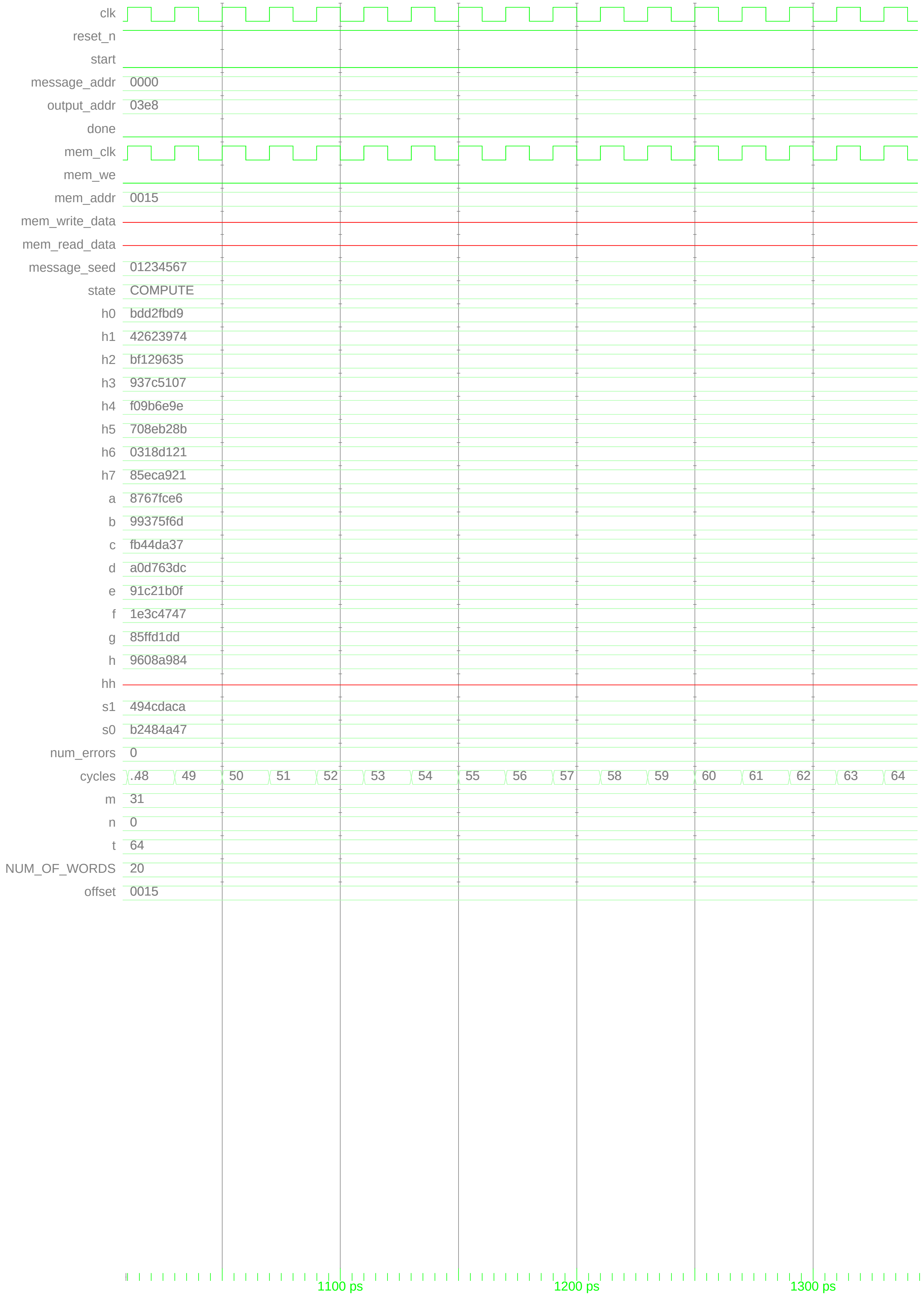


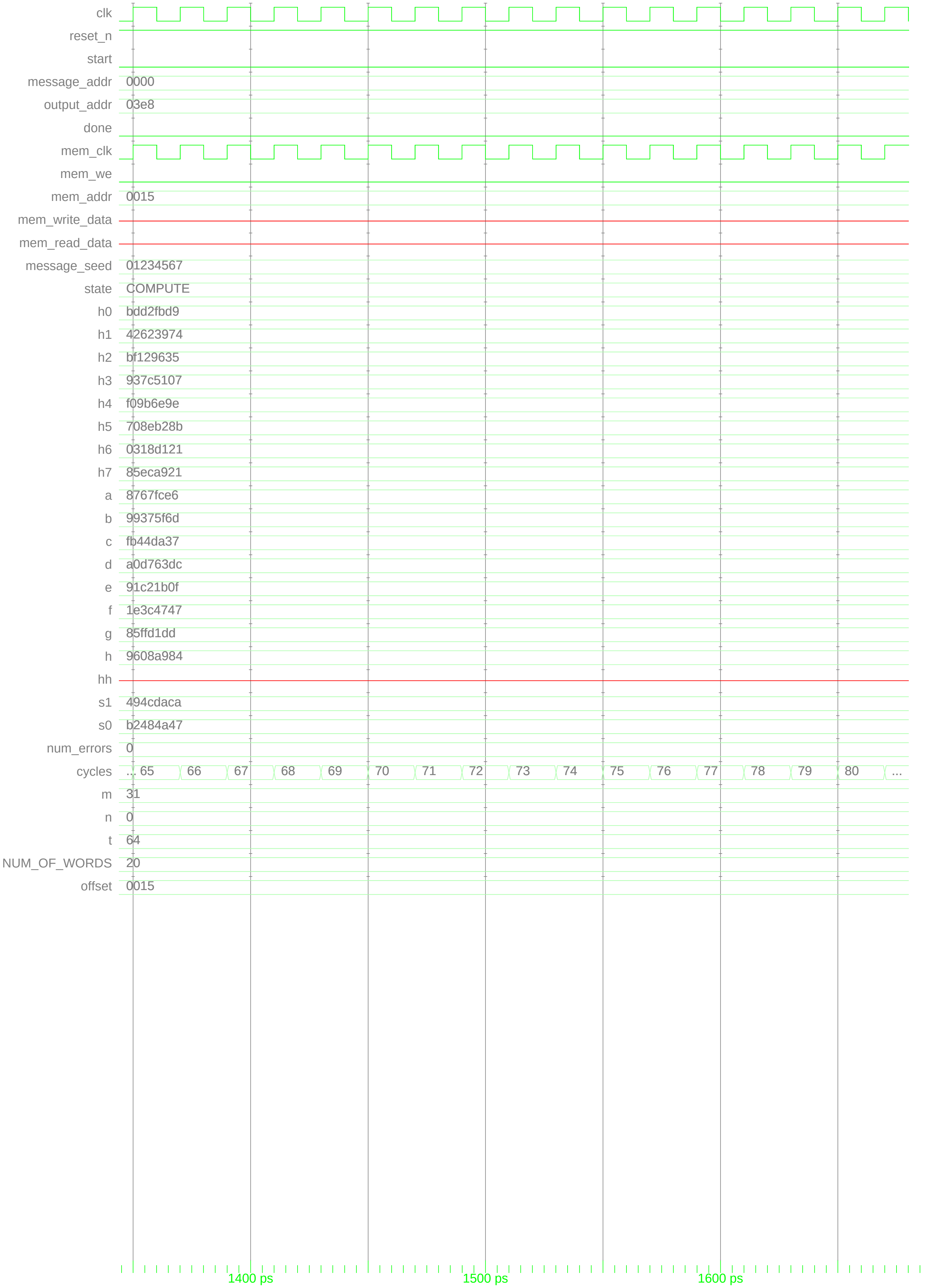
Entity:tb_simplified_sha256 Architecture: Date: Sat Jun 10 21:11:21 PDT 2023 Row: 1 Page: 1

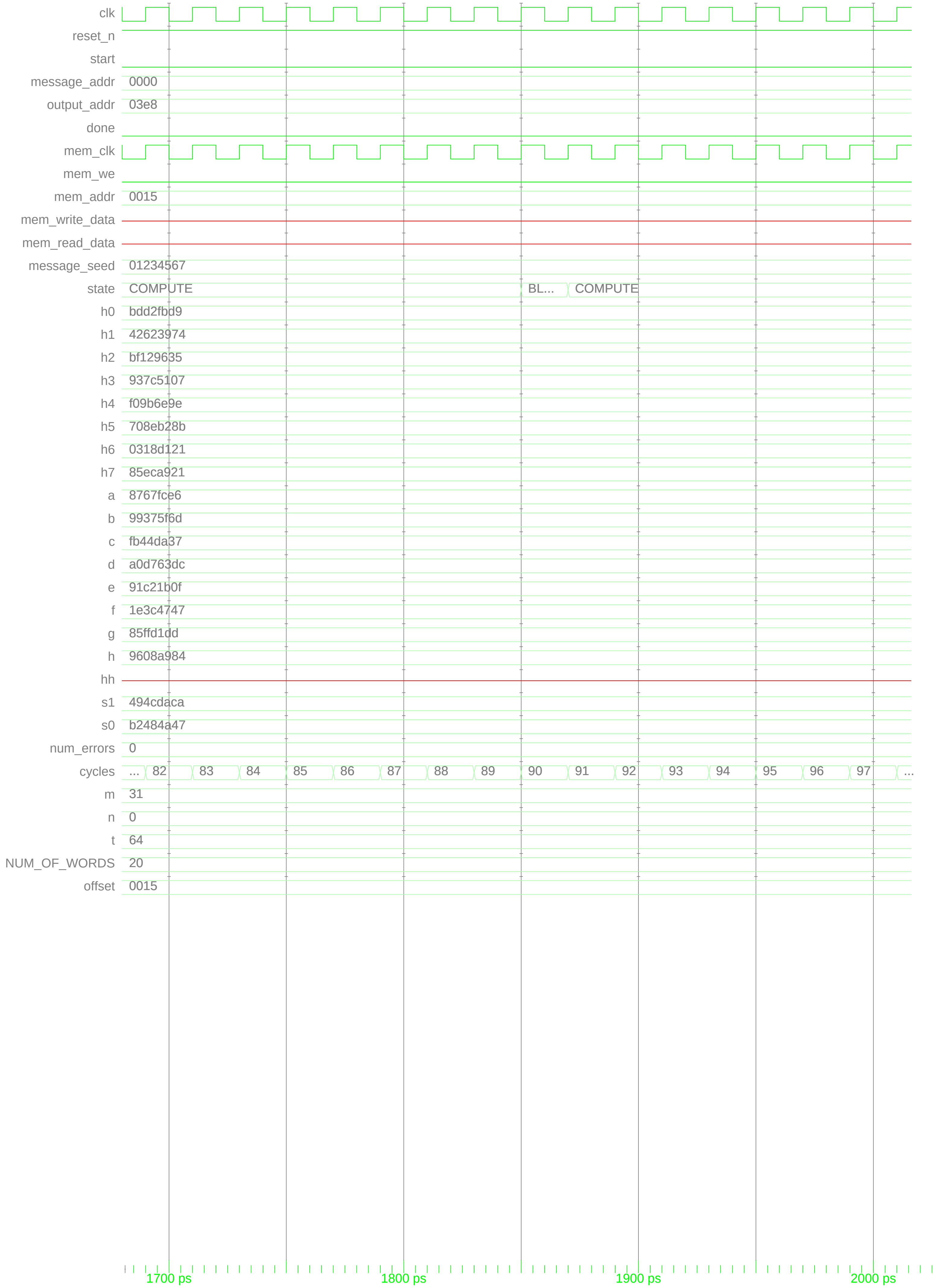


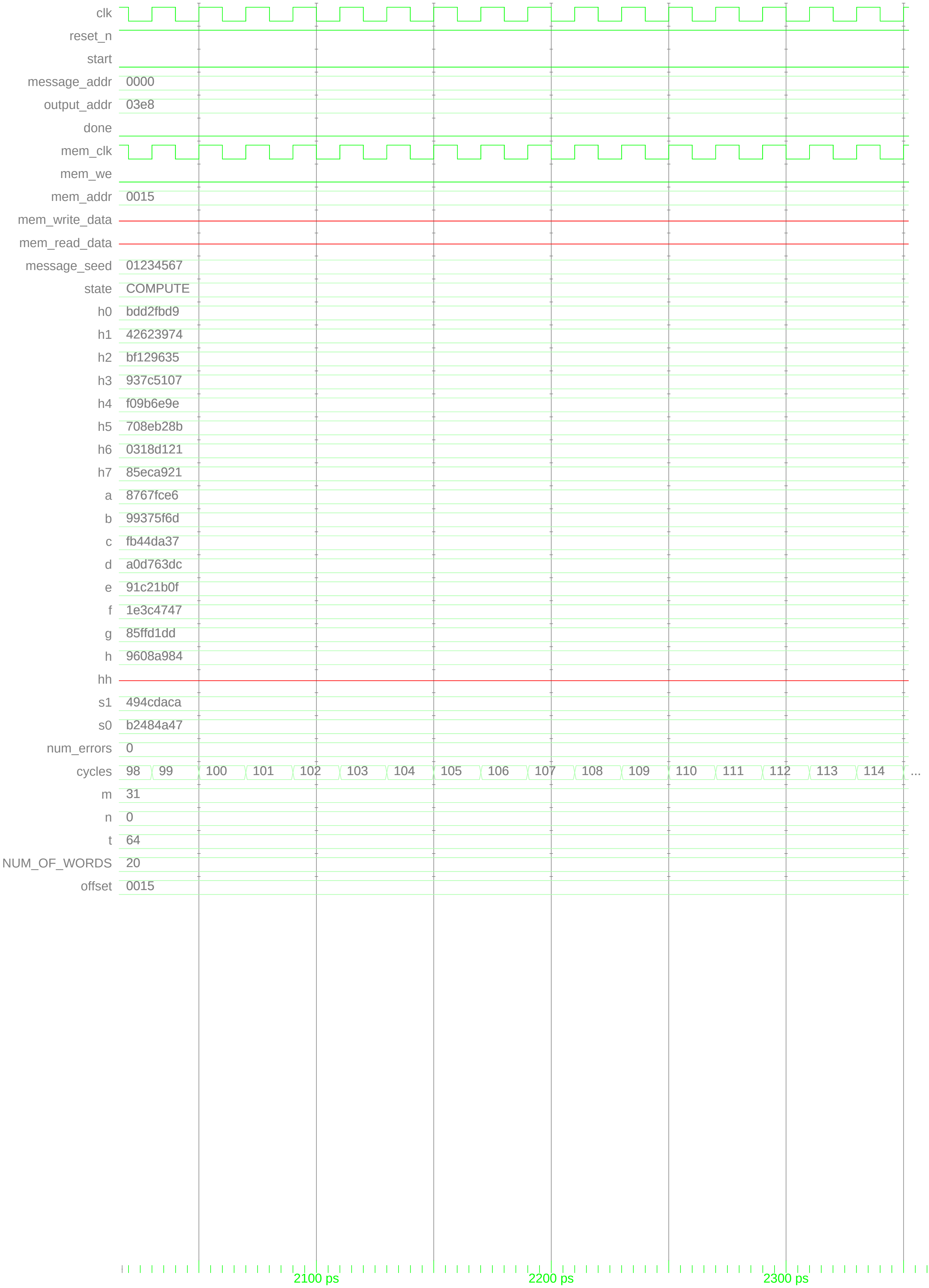
Entity:tb_simplified_sha256 Architecture: Date: Sat Jun 10 21:11:21 PDT 2023 Row: 2 Page: 2

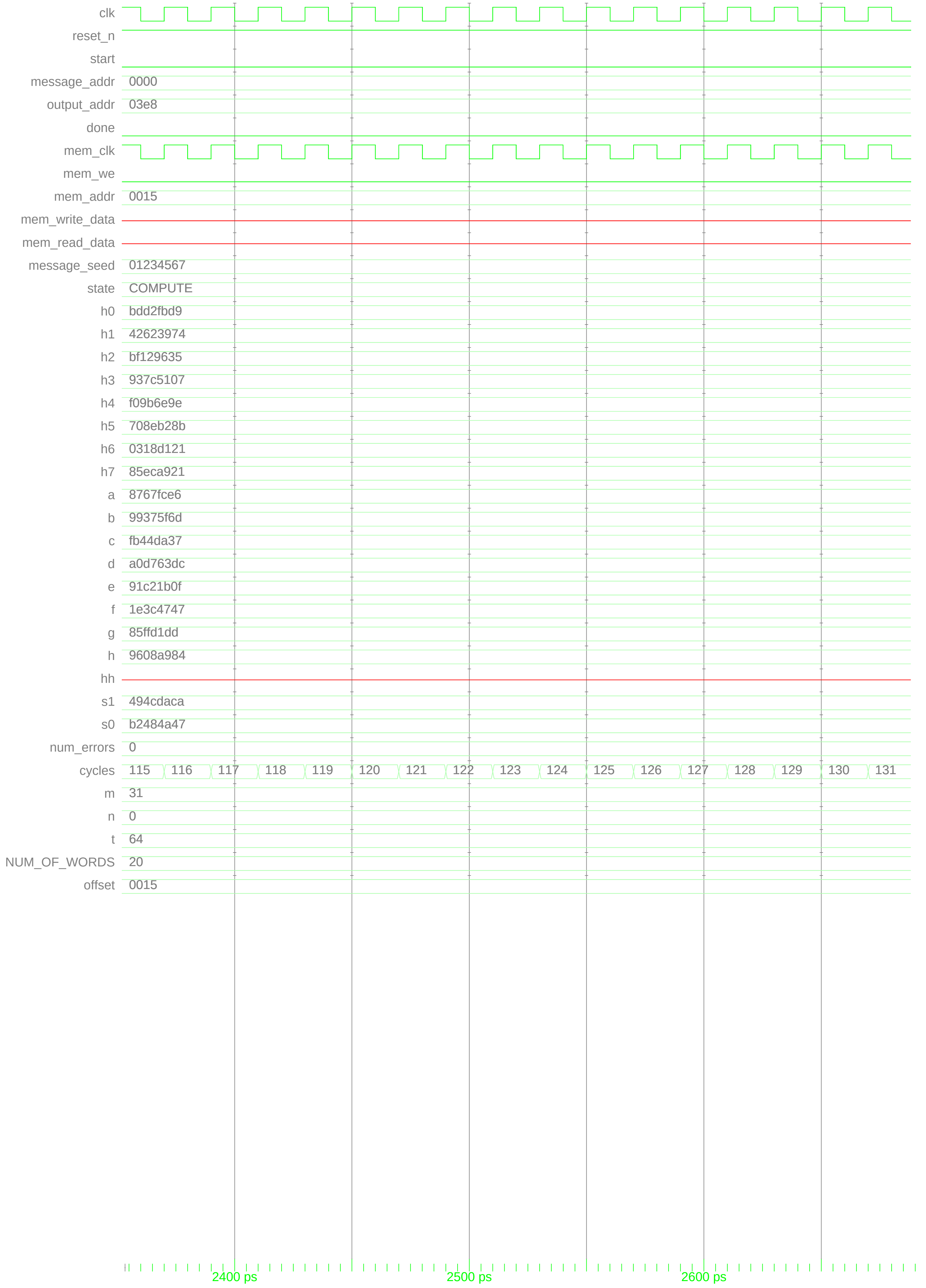

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
700 ps
800 ps
900 ps
1000 ps

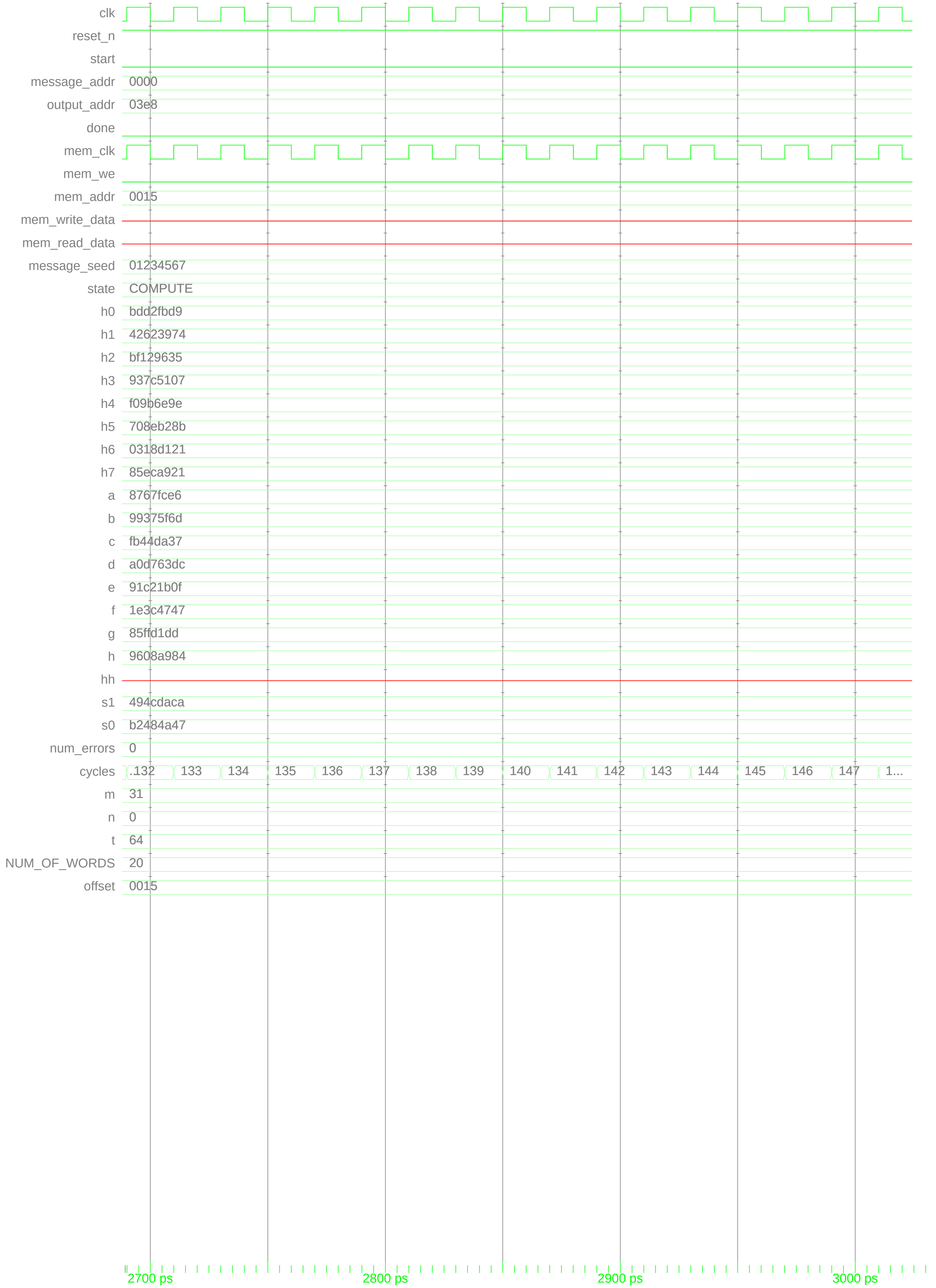

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles .48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
1100 ps
1200 ps
1300 ps

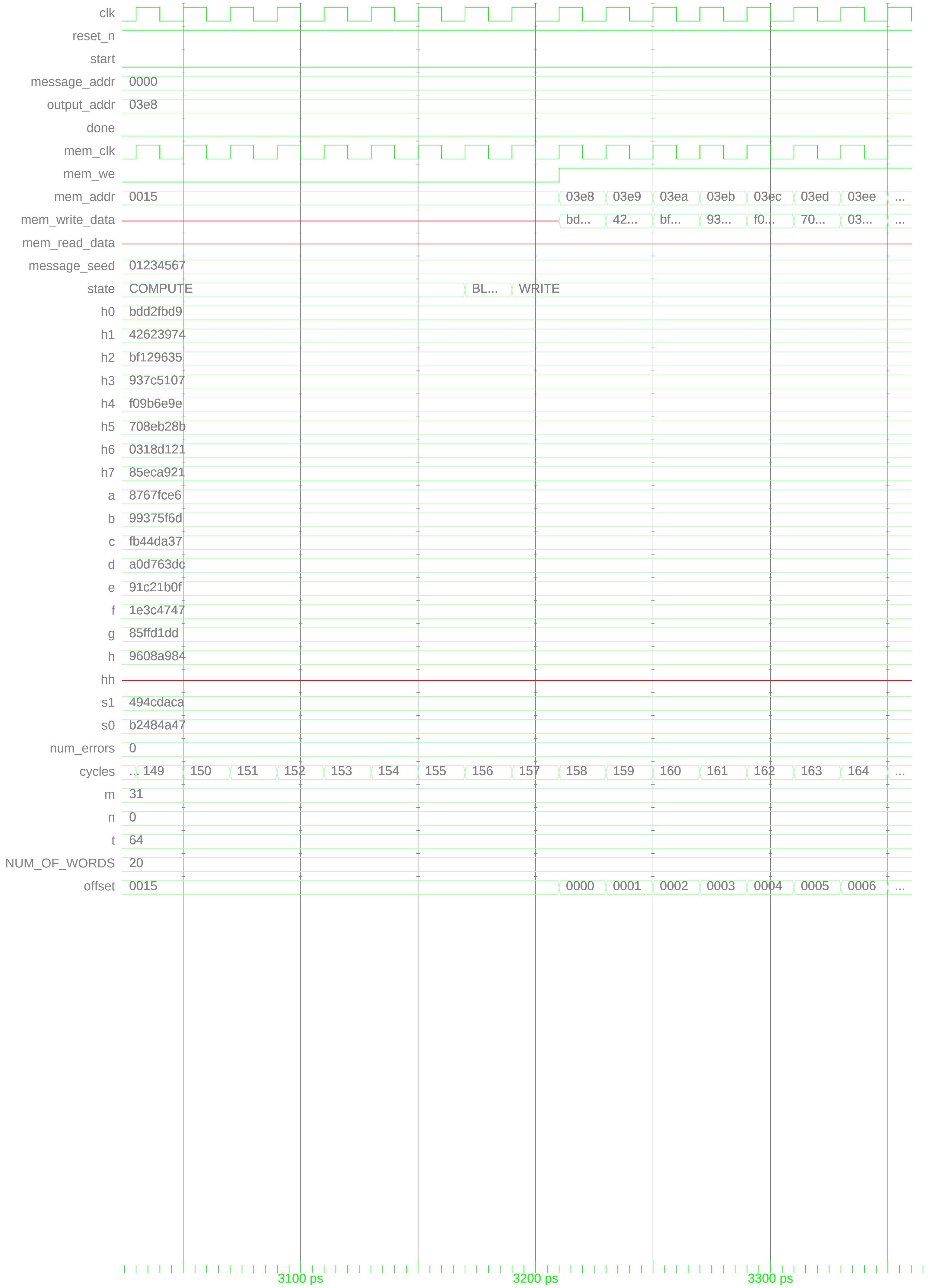

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles ... 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 ...
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
1400 ps
1500 ps
1600 ps

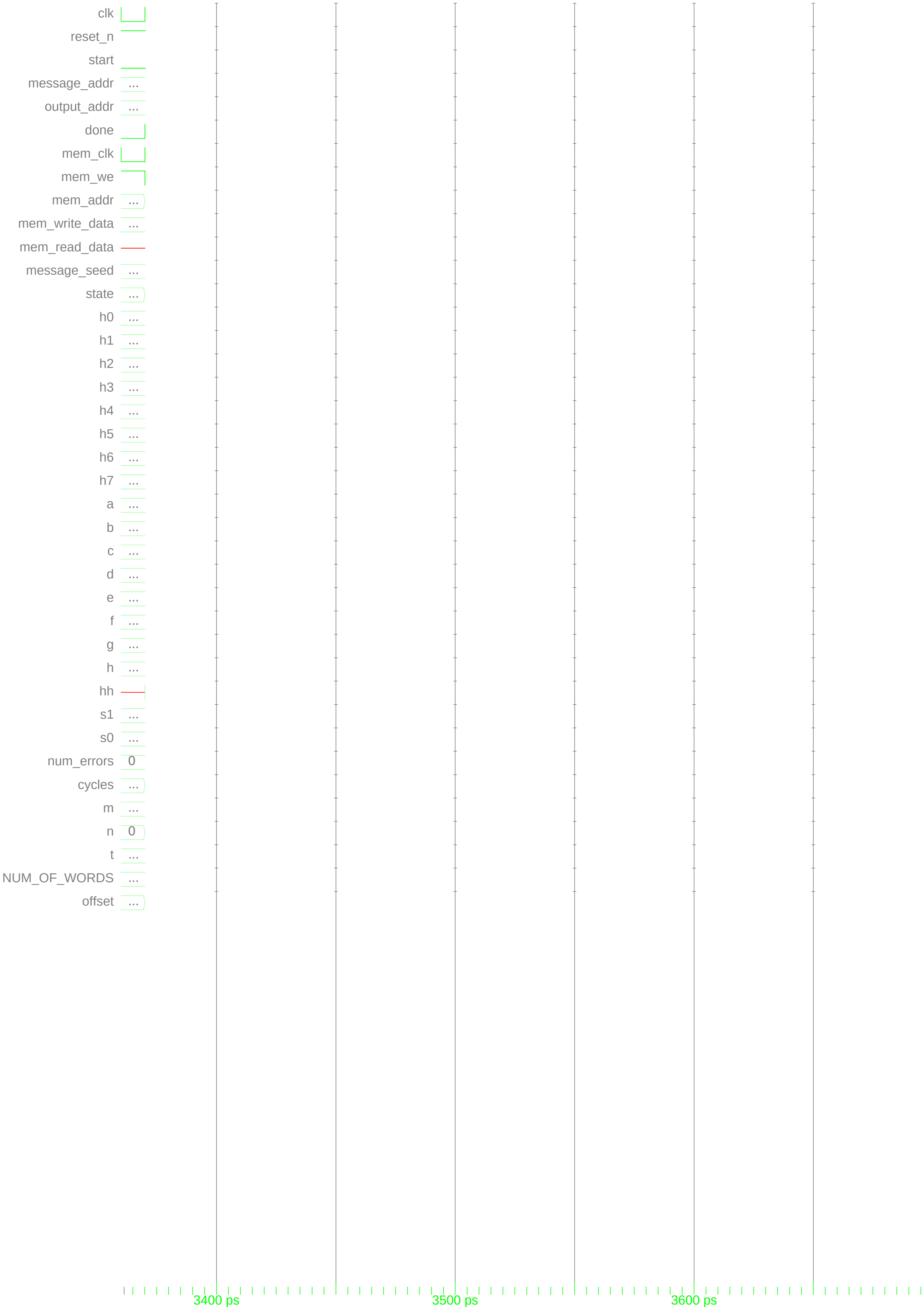

clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE BL... COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles ... 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 ...
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
1700 ps
1800 ps
1900 ps
2000 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 ...
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
2100 ps
2200 ps
2300 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
2400 ps
2500 ps
2600 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015
mem_write_data
mem_read_data
message_seed 01234567
state COMPUTE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles .132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 1...
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015
2700 ps
2800 ps
2900 ps
3000 ps


clk
reset_n
start
message_addr 0000
output_addr 03e8
done
mem_clk
mem_we
mem_addr 0015 03e8 03e9 03ea 03eb 03ec 03ed 03ee ...
mem_write_data bd... 42... bf... 93... f0... 70... 03... ...
mem_read_data
message_seed 01234567
state COMPUTE BL... WRITE
h0 bdd2fbd9
h1 42623974
h2 bf129635
h3 937c5107
h4 f09b6e9e
h5 708eb28b
h6 0318d121
h7 85eca921
a 8767fce6
b 99375f6d
c fb44da37
d a0d763dc
e 91c21b0f
f 1e3c4747
g 85ffd1dd
h 9608a984
hh
s1 494cdaca
s0 b2484a47
num_errors 0
cycles ... 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 ...
m 31
n 0
t 64
NUM_OF_WORDS 20
offset 0015 0000 0001 0002 0003 0004 0005 0006 ...
3100 ps
3200 ps
3300 ps


clk
reset_n
start
message_addr
output_addr
done
mem_clk
mem_we
mem_addr
mem_write_data
mem_read_data
message_seed
state
h0
h1
h2
h3
h4
h5
h6
h7
a
b
c
d
e
f
g
h
hh
s1
s0
num_errors
cycles
m
n
t
NUM_OF_WORDS
offset
0
0
3400 ps
3500 ps
3600 ps