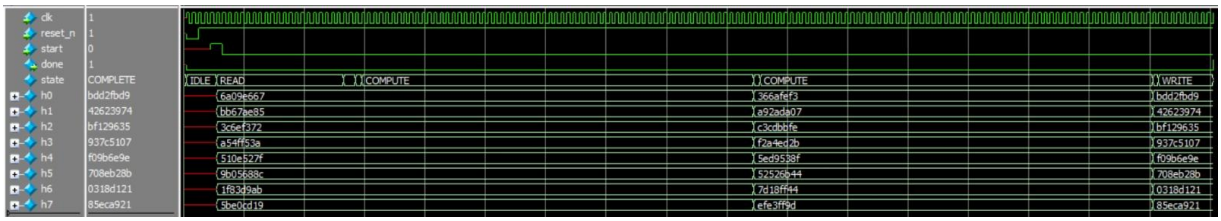
SUMEDH RAMAPRASAD- A59009811
SHREYAS BORSE- A59009564
PRAHARSHA MAHURKAR- A59009712

PART-1 SHA-256

WAVEFORM



The initial hash values are updated and the new hash values from the SHA-256 module can be seen in the simulation window. At the end of the simulation, done signal is high (after the WRITE state).

Total Cycles required and Comparison of Hash Values

```
# ---------------------
# 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:         168
#
#
# ***************************
#
# ** Note: $stop    : D:/DOWNLOADS/WI 2022/ECE 111/PROJECT/simplified_sha256/tb_simplified_sha256.sv(263)
#    Time: 3410 ps  Iteration: 2  Instance: /tb_simplified_sha256
# Break in Module tb_simplified_sha256 at D:/DOWNLOADS/WI 2022/ECE 111/PROJECT/simplified_sha256/tb_simplified_sha256.sv line 263
```

Maximum Frequency of operation.

```
+-----------------------------------------------------+
; Slow 900mV 100C Model Fmax Summary                  ;
+------------+-----------------+------------+------+
; Fmax       ; Restricted Fmax ; Clock Name ; Note ;
+------------+-----------------+------------+------+
; 158.35 MHz ; 158.35 MHz      ; clk        ;      ;
+------------+-----------------+------------+------+
This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods.  FMAX is only computed for paths
where the source and destination registers or ports are driven by the same clock.  Paths of different clocks, including generated clocks,
are ignored.  For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with
FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and
other slack reports for sign-off analysis.
```

Area utilized

```
+---------------------------------------------------------------------------------------+
; Analysis & Synthesis Summary                                                          ;
+-----------------------------------+---------------------------------------------------+
; Analysis & Synthesis Status       ; Successful - Thu Mar 10 19:11:52 2022             ;
; Quartus Prime Version             ; 20.1.0 Build 711 06/05/2020 SJ Lite Edition       ;
; Revision Name                     ; simplified_sha256                                 ;
; Top-level Entity Name             ; simplified_sha256                                 ;
; Family                            ; Arria II GX                                       ;
; Logic utilization                 ; N/A                                               ;
;     Combinational ALUTs           ; 2,422                                             ;
;     Memory ALUTs                  ; 0                                                 ;
;     Dedicated logic registers     ; 2,128                                             ;
; Total registers                   ; 2128                                              ;
; Total pins                        ; 118                                               ;
; Total virtual pins                ; 0                                                 ;
; Total block memory bits           ; 0                                                 ;
; DSP block 18-bit elements         ; 0                                                 ;
; Total GXB Receiver Channel PCS    ; 0                                                 ;
; Total GXB Receiver Channel PMA    ; 0                                                 ;
; Total GXB Transmitter Channel PCS ; 0                                                 ;
; Total GXB Transmitter Channel PMA ; 0                                                 ;
; Total PLLs                        ; 0                                                 ;
; Total DLLs                        ; 0                                                 ;
+-----------------------------------+---------------------------------------------------+
```

SUMEDH RAMAPRASAD- A59009811
SHREYAS BORSE- A59009564
PRAHARSHA MAHURKAR- A59009712

Part- 1 of the project is generating new hash value using K constant and original 32bit word in the messages. We have generalized the number of blocks calculation for different length of messages, although given inputs has fixed word lengths of 640. Every input message is multiple of 512 bits and hence they are divided into blocks depending on padding.

Summary:

Firstly, the RTL includes four functions; one is `determine_num_blocks, sha256_op, wt and rightrotate`
The *determine_num_blocks* calculate the number of Mn. The rightrotate function is used to calculate S0 and S1 in the sha256_op and wt, which gets the required Hash value. The `sha256_result` signal holds the final value and are assigned combinationally.

Now the simplified_SHA RTL is modelled as 6 state FSM design i.e.

a. IDLE → When start =1, all initial hash values [H0…H7] and [a…h] are equal. current address value is set to message address (read location). Offset increments each cycle util all 20 message words are read (write enable is set to 0). Now, FSM will move to READ state. (Note: at the end done is high if the state is IDLE).

b. READ → In Read state, message words are fetched form the memory until all words are read. Offset value is incremented when a new word is read from the memory. FSM moves to READ_DELAY state when all message words have been read from the memory.

c. READ_DELAY → In the READ_DELAY state, 21ST word in the message is assigned 1 (at MSB location) as a partition in actual message and 0s are padded. message length is also padded (31st and 32nd words) and the state is changed to BLOCK state. Now, we have both M_0 and M_1 ready for SHA256 calculation.

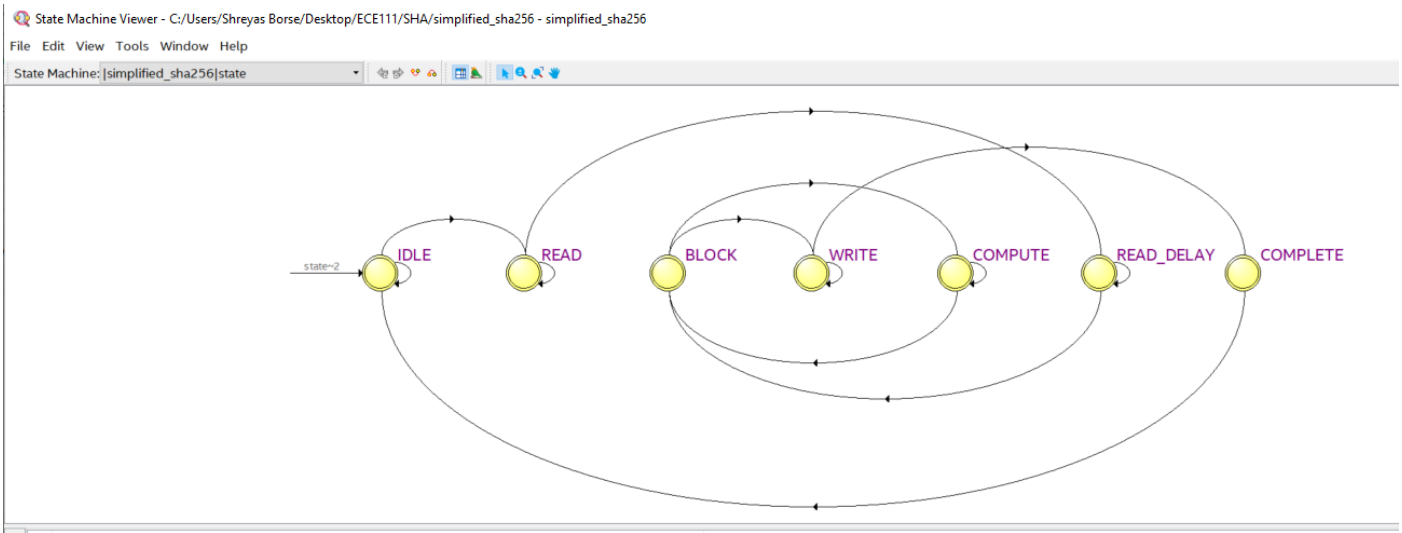
d. BLOCK→ We have optimized the number of **w[m]** used to 16 registers. Each 512-bit message block is loaded into **w[m]**. FSM state now changes to COMPUTE as we have the data to be hashed ready. FSM reaches BLOCK state after 64 iterations in COMPUTE state are completed. We check if all the 512-bit message blocks have been processed or any remain. If all blocks are processed, then the FSM moved to WRITE state. Otherwise, **w[m]** registers are loaded with the next 512-bit message block; FSM moves again to COMPUTE state.

e. COMPUTE → In COMPUTE state, **sha256_op** function is run for 64 iterations. The **w[15]** block is loaded with the result from the **wt** function. Data in each **w[m]** register is left shifted to **w[m-1]** location. This allows us to use fewer registers to save area. Tstep variable tracks the number of iterations. After 64 iterations are complete: new message digest is calculated by adding the initial hash values [**h0-h7**] used in current iteration, and the **a-h** values calculated after 64 iterations of **sha256_op** function. FSM now moves back to BLOCK state.

f. WRITE→ Here the sha256 result calculated after processing all 512-bit message blocks is loaded back into the memory. Offset variable is incremented after each memory write. Once offset reaches value of 8 (i.e., when all computed hash values are loaded into memory), FSM moves back to IDLE state. The module is now ready to process the next message block from memory.

SUMEDH RAMAPRASAD- A59009811
SHREYAS BORSE- A59009564
PRAHARSHA MAHURKAR- A59009712

FSM TRANSITION DIAGRAM FROM QUARTUS.

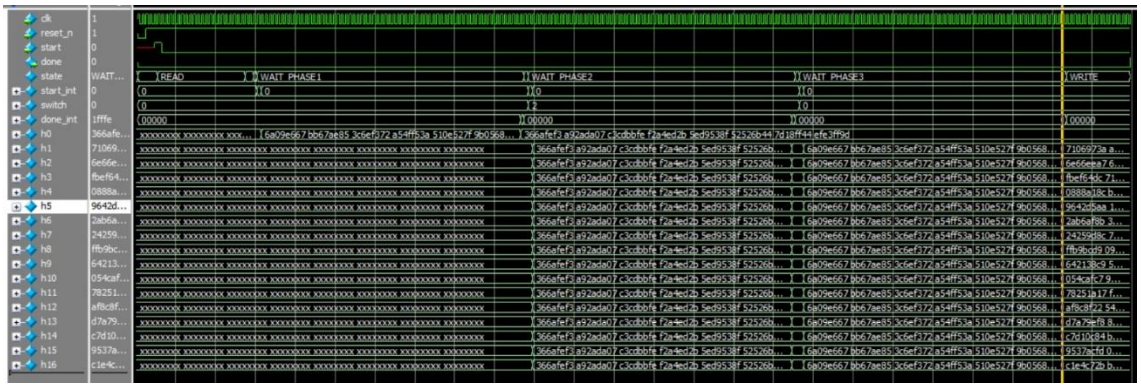


	Source State	Destination State
1	BLOCK	COMPUTE
2	BLOCK	WRITE
3	COMPLETE	IDLE
4	COMPUTE	COMPUTE
5	COMPUTE	BLOCK
6	IDLE	IDLE
7	IDLE	READ
8	READ	READ
9	READ	READ_DELAY
10	READ_DELAY	READ_DELAY
11	READ_DELAY	BLOCK
12	WRITE	WRITE
13	WRITE	COMPLETE

SUMEDH RAMAPRASAD- A59009811
SHREYAS BORSE- A59009564
PRAHARSHA MAHURKAR- A59009712

PART - 2 BITCOIN HASH

Waveform:



The hexadecimal output values of Bitcoin module and the correct hash values are written into the memory when the done signal is 1 at end of the Write state.

Total Cycles required and Comparison of Hash Values

```
# ---------------------
# 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:         247
#
#
# ****************************
#
# ** Note: $stop    : D:/DOWNLOADS/WI 2022/ECE 111/PROJECT/Group_10_submission_partial/bitcoin_hash/RTL/tb_bitcoin_hash.sv(334)
#    Time: 4990 ps  Iteration: 2  Instance: /tb_bitcoin_hash
# Break in Module tb_bitcoin_hash at D:/DOWNLOADS/WI 2022/ECE 111/PROJECT/Group_10_submission_partial/bitcoin_hash/RTL/tb_bitcoin_hash.sv line 334
# End time: 19:09:33 on Mar 10,2022, Elapsed time: 0:00:27
# Errors: 0, Warnings: 5
```

Maximum Frequency of operation.

```
+-----------------------------------------------------+
; Slow 900mV 100C Model Fmax Summary                  ;
+------------+-----------------+------------+--------+
; Fmax       ; Restricted Fmax ; Clock Name ; Note ;
+------------+-----------------+------------+--------+
; 141.74 MHz ; 141.74 MHz      ; clk        ;      ;
+------------+-----------------+------------+--------+
This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods.  FMAX is only computed for paths where the
source and destination registers or ports are driven by the same clock.  Paths of different clocks, including generated clocks, are ignored.  For
paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in
terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for sign-off analysis.
```

Area utilized

```
+---------------------------------------------------------------------------------+
; Analysis & Synthesis Summary                                                    ;
+-----------------------------------+---------------------------------------------+
; Analysis & Synthesis Status       ; Successful - Thu Mar 10 19:21:44 2022       ;
; Quartus Prime Version             ; 20.1.0 Build 711 06/05/2020 SJ Lite Edition ;
; Revision Name                     ; bitcoin_hash                                ;
; Top-level Entity Name             ; bitcoin_hash                                ;
; Family                            ; Arria II GX                                 ;
; Logic utilization                 ; N/A                                         ;
;     Combinational ALUTs           ; 13,785                                      ;
;     Memory ALUTs                  ; 0                                           ;
;     Dedicated logic registers     ; 23,036                                      ;
; Total registers                   ; 23036                                       ;
; Total pins                        ; 118                                         ;
; Total virtual pins                ; 0                                           ;
; Total block memory bits           ; 0                                           ;
; DSP block 18-bit elements         ; 0                                           ;
; Total GXB Receiver Channel PCS    ; 0                                           ;
; Total GXB Receiver Channel PMA    ; 0                                           ;
; Total GXB Transmitter Channel PCS ; 0                                           ;
; Total GXB Transmitter Channel PMA ; 0                                           ;
; Total PLLs                        ; 0                                           ;
; Total DLLs                        ; 0                                           ;
+-----------------------------------+---------------------------------------------+
```

Part-2 – The simplified_sha256 module designed in Part-1 is re-used after doing the following modifications:

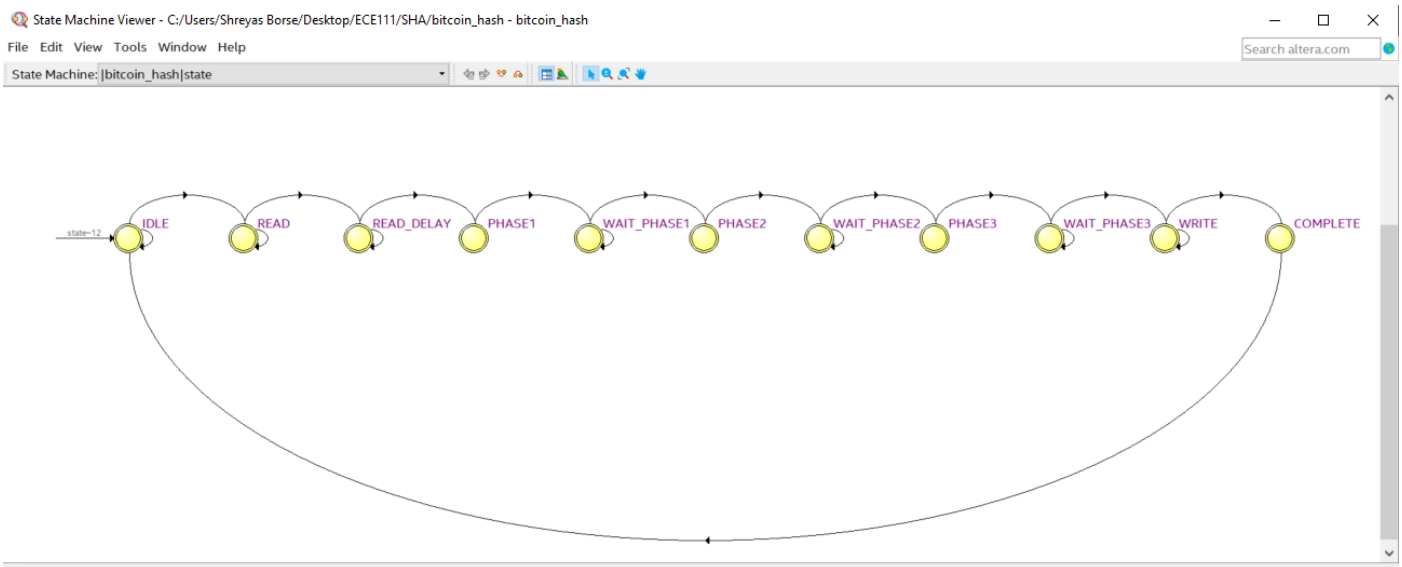
a. READ, READ_DELAY, WRITE states from FSM are removed
b. COMPLETE state has been added – FSM reaches this state after 64 iterations of **sha256_op**
c. New ports **in, switch** have been added to support the below feature:
   When **switch=0**, initial hash values loaded are same as that in Part-1
   When **switch=1**, hash values present on the **in** port are loaded into the design

Now the bitcoin_hash RTL is modelled as 10 state FSM design i.e.

a. IDLE → When start =1, current address value is set to message address (read location). Offset, Start_int, switch is set to 0 AND each cycle until all 20 message words are read (write enable is set to 0). Now, FSM will move to READ state. (Note: at the end done is high if the state is IDLE).
b. READ → In read state, the read location memory is written to message until offset 20 (), and offset is Incremented and state is transitioned to READ_DELAY.
c. READ_DELAY → In Read Delay state, we are loading from 21st to 32nd word of message similar to part 1. FSM moves to Phase 1 State.
d. PHASE_1 → Here the SHA_256 operation is done on 1ST 512 BITS of message block. Start signal is given to 1 SHA instance. FSM State moves to WAIT_PHASE1
e. WAIT_PHASE1 →FSM waits for SHA instance to completes its operations. Once its complete, **w0 -w15** are loaded with remaining message words, nonces, Padding and message size. FSM moves to PHASE_2.
f. PHASE_2 → Here 16 SHA instances are triggered parallelly, and Hash value generated from PHASE_1 is provided as seed for PHASE_2 COMPUTATION. FSM moves to WAIT_PHASE_2 state.
g. WAIT_PHASE_2 → FSM waits for 16 SHA instances to completes their operations, once its complete **w0 – w15** are loaded with the Hash Values generated from phase_2 computation. FSM moves to Phase_3 State.
h. PHASE_3 → Here 16 SHA instances triggered parallelly, and initial Hash values are same as Phase_1. FSM moves to WAIT_PHASE_3 state.
i. WAIT_PHASE_3 → FSM waits for 16 SHA instances to completes their operations, once its done FSM moves to WRITE state.
j. WRITE → The HASH values computed from Phase_3 are loaded to Memory. Offset variable tracks number of memory writes. Once done FSM moves to Complete State.
k. COMPLETE → done is set to High to Indicate completion of all Steps. FSM moves to IDLE state and is ready to process Next Message.

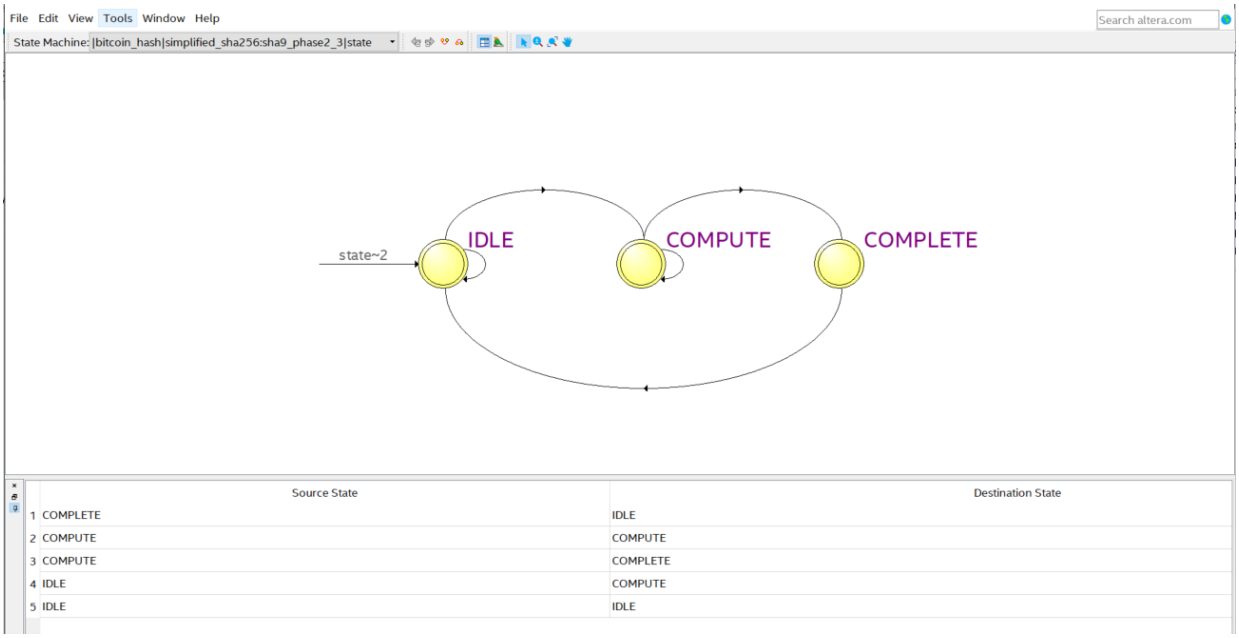
SUMEDH RAMAPRASAD- A59009811
SHREYAS BORSE- A59009564
PRAHARSHA MAHURKAR- A59009712

FSM TRANSITION DIAGRAM FROM QUARTUS.



	Source State	Destination State
1	COMPLETE	IDLE
2	IDLE	IDLE
3	IDLE	READ
4	PHASE1	WAIT_PHASE1
5	PHASE2	WAIT_PHASE2
6	PHASE3	WAIT_PHASE3
7	READ	READ_DELAY
8	READ	READ
9	READ_DELAY	READ_DELAY
10	READ_DELAY	PHASE1
11	WAIT_PHASE1	WAIT_PHASE1
12	WAIT_PHASE1	PHASE2
13	WAIT_PHASE2	WAIT_PHASE2
14	WAIT_PHASE2	PHASE3
15	WAIT_PHASE3	WAIT_PHASE3
16	WAIT_PHASE3	WRITE
17	WRITE	WRITE
18	WRITE	COMPLETE

State Table
Transitions / Encoding



	Source State	Destination State
1	COMPLETE	IDLE
2	COMPUTE	COMPUTE
3	COMPUTE	COMPLETE
4	IDLE	COMPUTE
5	IDLE	IDLE