User Guide for ATMEL Serial Firmware DataFlash® Verilog model

1. Introduction:

This documentation describes the verification environment and the features that offer to a test engineer.

2. Verilog Model for DataFlash:


3. Introduction of modules:

Device model:
AT26DFxxx model will receive opcode, data from the SI line. Some of the tasks in this model are created for performing internal operations like programming the memory, easing the memory, checking protection status of particular sector of memory, Protection and Unprotection of sectors. Few tasks are created to retrieve 8-bit data from memory and to send it through SO, sending protection status of sectors in the SO line.

Testbench:
This module is responsible for sending opcode, address and data. Based on which particular trigger input asserted, corresponding opcode, address and data transfer will start. For example, if tr_byt_prog is asserted, then the byte programming opcode will be transferred in serial_in line in serial manner. Then the address obtained through m_address is transmitted serially. data_num will intimate the number of data to be written into the memory. Based on the data_num value, data in w_data will be transmitted serially through serial_in. The testbench_AT26DFx.v module has HOLDB control. Hence this signal can be passed based on the HOLDB value.

Top:
topx_testbench.v module is responsible for selecting the device for which the model has to work, sending opcode, address and data in SI line and obtaining the read array data from SO line. It also passes all the parameter needed for the device model (AT26DFxxx.v) internally. It contains all the test cases to test the model for each device model (AT25DF041A, AT26DF081A, AT26DF161A, and AT26DF321) and also for all the possibilities of error conditions.
4. Signals and Events of AT26DFxxx.v module:

AT26DFxxx model will receive opcode, data from the SI line. Some of the tasks in this model are created for performing internal operations like programming the memory, easing the memory, checking protection status of particular sector of memory, Protection and Unprotection of sectors. Few tasks are created to retrieve 8-bit data from memory and to send it through SO, sending protection status of sectors in the SO line.

4.1 Port Signals:

Input ports for this model are: CSB, SCK, SI, WPB, and HOLDB. Output port for this model is SO.

CSB is Chip Select Signal. Generated from testbench and is a active low signal.
SCK is Serial Clock Signal. Generated from testbench.
SI is Serial Input. Generated from testbench and provides model with either opcode, address or data.
WPB is Write Protect signal. Generated from testbench and is a active low signal.
HOLDB is a Hold signal. Generated from testbench and is a active low signal. (This port will not be available for AT26DF321 model since this model doesn't support hold feature)
SO is a output port for the model. It can provide data, status register content, Manufacture ID and protection status of memory.

4.2 Top level signals:

memory is an array of 8 bit memory of selected device.
MEMSIZE specifies the maximum memory size for each device.
status_reg signal contains value of the 8-bit Status register register.
sector_reg specifies the sector protection status of the sectors for selected memory model.
deep_power_down signal intimates whether the device is in Deep Power Down Mode (Login 1) or in Stand-by Mode(Logic 0).
rd_dummy is used to differentiate between Read Array low frequency (Logic 0) and high frequency (Logic 1).
seq_byte_prog signal intimates start of Sequential Byte Programming.
current_address signal stores the 24 bit address.
data_in signal stores the 8 bit data received.
read_data retrieves 8-bit data from the memory in the specified address.
SO_on enable signal for SO
SO_reg signal holds the value for SO
SPRL signal holds Sector Protection Register Locked bit status of Status register
SPM signal holds the Sequential Programming Mode status of Status register.
EPE signal holds Erase/Program Error of Status register
WPP signal Write Protect Pin status Status register
SWP signal holds the Software Protection Status of Status register
WEL signal holds the Write Enable Latch of Status register
RDYnBSY signal holds the Ready/Busy status of Status register

4.3 List of events:

**EDPD**: Enter into Deep Power Down
**RDPD**: Resume from Deep Power Down
**RA**: Read Array (both low frequency and high frequency)
**BE4**: Block Erase of 4KB block
**BE32**: Block Erase of 32KB block
**BE64**: Block Erase of 64KB block
**CE**: Chip erase
**BP**: Byte Programming
**SBP**: Sequential Byte programming (This event will not be asserted for AT26DF321 model since this model doesn't support Sequential Byte programming feature)
**WE**: Write Enable
**WD**: Write Disable
**PS**: Protect Sector
**UPS**: Unprotect Sector
**RSPR**: Read Sector Protection Register
**RSR**: Read Status Register
**WSR**: Write Status Register
**MIR**: Manufacture Id Read

4.4 List of parameters:

**fRDLF**: SCK Frequency for read Array Low frequency in MHz
**fSCK**: Serial clock (SCK) Frequency in MHz
**tDIS**: Output Disable time in ns
**tV**: Output Valid time in ns
**tOH**: Output Hold time in ns
**tSCKH**: SCK High time in ns
**tSCKL**: SCK Low time in ns
**tHLQZ**: HOLD! Low to Output High-z time in ns
**tHHQX**: HOLD! High to Output Low-z time in ns
**tEDPD**: Chip Select high to Deep Power-down time in ns
**tRDPD**: Chip Select high to Stand-by Mode time in ns
**tWRSR**: Write Status Register Time in ns
**tSECP**: Sector Protect Time in ns
**tSECUP**: Sector Unprotect Time in ns
**tBP**: Byte Program Time in ns
**tPP**: Page Program Time in ns
**tBLKE4**: Block Erase Time 4-kB time in ns
**tBLKE32**: Block Erase Time 32-kB time in ns
tBLKE64: Block Erase Time 64-kB time in ns
tCHPE: Chip erase time in ns
DEVICE: Specifies the selected card model
PRELOAD: memory preloading condition
MEMORY_FILE: memory preload file

5 Tasks used in AT26DFxxx.v module:

Some of the tasks in AT26DFxxx model are created to receive opcode, data from the SI line. Some of the tasks are created for performing internal operations like programming the memory, easing the memory, checking protection status of particular sector of memory, Protection and Unprotection of sectors. Few tasks are created to retrieve 8-bit data from memory and to send it through SO, sending protection status of sectors in the SO line.

get_opcode: This task receives 8 bits of opcode.
get_data: This task receives 8 bits of data.
read_out_array: This task is to read 8-bit data from main Memory
protect_sector: This task is used for Sector protection
unprotect_sector: This task is used for Sector Unprotection
check_protection: This task is used to Check Sector Protection
send_protection_status: This task is used for Sending Sector Protection status
page_program: This task is used to receive data for byte / page programming
byte_program: This task is used for Byte program / Page Program / Sequential Byte programming for devices
erase_4kb: This task is used to Erase a 4kB block
erase_32kb: This task is used to Erase a 32kB block
erase_64kb: This task is used to Erase a 64kB block
erase_chip: This task is used for Chip Erase.

6. Status and Error printing of AT26DFxxx.v module:

Based on the opcode received, any of the following will appear on the screen as the model is being compiled:
 Opcode(ab) for Resume from Deep Power-down received
if the device is in Deep Power-down, and if any other opcode except “ab” appears, then the following print will appear:
 Opcode XX is not allowed: device in Deep Power-down
 Opcode(0b) for Read Array received
 Opcode(03) for Read Array (Low Freq) received
 Opcode(20) for 4 KB Block erase received
 Opcode(52) for 32 KB Block erase received
 Opcode(d8) for 64 KB Block erase received
 Opcode(c7) for Chip erase received
 Opcode(02) Byte Program received
Opcode(af) Sequential Byte Program received
if device selected is AT26DF321, then this printing will appear
Sequential Byte Program is not supported for device AT26DF321
Opcode(06) for Write Enable received
Opcode(04) for Write Disable received
Opcode(36) for Protect Sector received
Opcode(39) for Unprotect Sector received
Opcode(3c) for Read Sector Protection Register received
Opcode(05) for Status Register Read received
Opcode(01) for Write Status Register received
Opcode(9f) for Read Manufacturer and Device ID received
Opcode(b9) for Deep Power-down received
if none of the above opcodes received, then the following print will appear:
Unrecognized opcode XX

Inside Tasks:

In read_out_array task after receiving each byte of data in consecutive memory locations,
following printing will appear:
Data: XX read from memory location XXXXXX

When calling protect_sector, following printing will appear:
Entered Protect Sector task

When entering unprotect_sector, anyone of the following printing will appear:
Entered Unprotect sector for Device: AT25DF041A
Entered Unprotect sector for Device: AT26DF081A
Entered Unprotect sector for Device: AT26DF161A
Entered Unprotect sector for Device: AT26DF321
Unprotect Sector failed

if the check_protection process fails, following printing will appear:
Check Sector Protection failed

In page_program after each byte of data receiving, following printing will appear:
“One byte of data: XX received for Page Program”

In byte program after writing a byte of data in memory, following printing will appear:
“One Byte of data XX written in memory in location XXXXXX”

In erase_4kb, erase_32kb, erase_64kb and erase_chip following printing will appear respectively:
4kB Block with start address XX is going to be erased
32kB Block with start address XX is going to be erased
64kB Block with start address XX is going to be erased
Chip Erase is going to be started
Entering into Deep Power-down execution, following printing will appear:
If device is busy(RDYnBSY==1):
“Device is busy. Deep Power-down command cannot be issued”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX Mhz for SPI interface. Deep Power-down is not allowed.”
At the end of operation:
“Device enters into Deep Power-down mode. Send 'Resume from Deep Power-down' to resume”

Entering into Resume from Deep Power-down execution, following printing will appear:
If device is busy(RDYnBSY==1):
“Device is busy. Deep Power-down command cannot be issued”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX Mhz for SPI interface. Resume from Deep Power-down is not allowed.”
At the end of operation:
“Device Resumes from Deep Power-down mode”

Entering into Manufacturing ID Read execution, following printing will appear:
If device is busy(RDYnBSY==1):
“Device is busy. Manufacturing ID Read cannot be issued”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX Mhz for SPI interface. Manufacturing ID Read is not allowed”
At the end of operation:
“Manufacture ID and Device ID of Device sent”

Entering into Status Register Read execution, following printing will appear:
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX Mhz for SPI interface. Status Register Read is not allowed.”
“Status register content of Device transmitted”

Entering into Status Register Write execution, following printing will appear:
If device is busy(RDYnBSY==1):
“Device is busy. Write Status Register is not allowed”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Write Status Register is not allowed.”
Any of the following print will appear based on the write status register operation.
   “SPRL Software locked”
   “SPRL Hardware locked. Lock cannot be done”
   “SPRL Software locked. Lock cannot be done”
   “SPRL Hardware locked. Unlock cannot be done”
   “Sector Protection Register UnLocked”
“SPRL in unlocked state”
“SPRL locked. Global Protect/Unprotect cannot be done”
“Global Protection issued for Sector protection register”
“Global Protection issued for Sector protection register”
“Global Unprotect issued for Sector protection register”
“SPRL locked. Global Protect/Unprotect cannot be done”

If WEL bit is not set prior to write status register operation, following print will appear:
“WEL bit not set. Write Status Register is not allowed”
“Write Status Register operation completed”

**Entering into Write Enable execution, following printing will appear:**
If frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Write Enable is not allowed.”
“Write Enable Latch Set”

**Entering into Write Disable execution, following printing will appear:**
If frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Write Disable is not allowed.”
“Write Enable Latch Reset”

**Entering into Read Array execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Read Array is not allowed”
If frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Read Array is not allowed”

**Entering into Protect Sector execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Protect Sector is not allowed”
If frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Protect Sector is not allowed”
If WEL bit is set/SPRL is unlocked the received address is protected:
“Sector for Address XXXXXX is Protected”
If WEL bit is not set/SPRL is locked
“WEL bit not set or Sector Protection Register Locked. Protect Sector is not allowed”

**Entering into Unprotect Sector execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Un-Protect Sector is not allowed”
If frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Un-Protect Sector is not allowed”
if WEL bit is set/SPRL is unlocked the received address is Unprotected:
“Sector for Address XXXXXX is UnProtected”
If WEL bit is not set/SPRL is locked”
“WEL bit not set or Sector Protection Register Locked. Un-Protect Sector is not allowed”

**Entering into Read Sector Protection Register execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Read Sector Protection Register is not allowed”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Read Sector Protection
Register is not allowed”
Based on sector protection condition any one of the following print will appear:
“Sector for Address XXXXXX is Protected”
“Sector for Address XXXXXX is Unprotected”
“Read Sector Protection Register completed”

**Entering into Byte Program execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Byte Program is not allowed”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Byte
Program is not allowed”
if CSB deasserted in wrong point of time, following print will appear and operation will get
aborted:
“Chip Select deasserted in non-even point. Byte/Page Program is not allowed”
If WEL bit is not set:
“WEL bit not set. Byte Program is not allowed”
Based on sector protection condition any one of the following print will appear:
“Sector for Address XXXXXX is Protected. Byte Program cannot be performed”
“Sector for Address XXXXXXX is UnProtected, Byte Program can be performed”
“Byte write completed”

**Entering into Sequential Byte Program execution, following printing will appear:**
If device is busy(RDYnBSY==1):
“Device is busy. Sequential Byte Program is not allowed”
if frequency exceeds specified fSCK:
“WARNING: Frequency should be less than XX MHz for SPI interface. Sequential
Byte Program is not allowed”
if the sequential operation reaches end of memory:
“Sequential Byte Program for device reaches end of memory. No Wrapping
allowed. Sequential Byte Program is not allowed”
If WEL bit is not set:
“WEL bit not set. Sequential Byte Program is not allowed”
Based on sector protection condition any one of the following print will appear:
“Sector for Address XXXXXXX is Protected. Sequential Byte Program cannot be
performed”
“Sector for Address XXXXXX is Unprotected, Sequential Byte Program can be performed”
“Sequential Byte write completed”

**Entering into 4KB Block Erase execution, following printing will appear:**
If device is busy(RDYnBSY==1):
  “Device is busy. 4KB Block Erase is not allowed”
if frequency exceeds specified fSCK:
  “WARNING: Frequency should be less than XX MHz for SPI interface. 4KB Block Erase is not allowed”
If WEL bit is not set:
  “WEL bit not set. 4KB Block Erase is not allowed”
Based on sector protection condition any one of the following print will appear:
  “Sector for Address XXXXXX is Protected. 4KB Block Erase cannot be performed”
  “Sector for Address XXXXXX is Unprotected, 4KB Block Erase can be performed”
“4kB Block with start address XXXXXX erased”

**Entering into 32KB Block Erase execution, following printing will appear:**
If device is busy(RDYnBSY==1):
  “Device is busy. 32KB Block Erase is not allowed”
if frequency exceeds specified fSCK:
  “WARNING: Frequency should be less than XX MHz for SPI interface. 32KB Block Erase is not allowed”
If WEL bit is not set:
  “WEL bit not set. 32KB Block Erase is not allowed”
Based on sector protection condition any one of the following print will appear:
  “Sector for Address XXXXXX is Protected. 32KB Block Erase cannot be performed”
  “Sector for Address XXXXXX is Unprotected, 32KB Block Erase can be performed”
“32KB Block with start address XXXXXX erased”

**Entering into 64KB Block Erase execution, following printing will appear:**
If device is busy(RDYnBSY==1):
  “Device is busy. 64KB Block Erase is not allowed”
if frequency exceeds specified fSCK:
  WARNING: Frequency should be less than XX MHz for SPI interface. 64KB Block Erase is not allowed”
if WEL bit is not set:
  “WEL bit not set. 64KB Block Erase is not allowed”
Based on sector protection condition any one of the following print will appear:
  “Sector for Address XXXXXX is Protected. 64KB Block Erase cannot be performed”
  “Sector for Address XXXXXX is UnProtected, 64KB Block Erase can be
performed"
64kB Block with start address XXXXXX erased"

**Entering into Chip Erase execution, following printing will appear:**
If device is busy(RDYnBSY==1):
   “Device is busy. Chip Erase is not allowed”
if frequency exceeds specified fSCK:
   “WARNING: Frequency should be less than XX MHz for SPI interface. Chip Erase
   is not allowed”
if WEL bit is not set:
   “WEL bit not set. Chip Erase is not allowed”
Based on sector protection condition any one of the following print will appear:
   “All Sectors in the chip are Unprotected. Chip Erase can be performed”
   “Some or all Sectors in the chip is Protected. Chip Erase cannot be performed”
Chip Erase Completed”

**if the input clock frequency (fSCK) exceeds maximum limit, following printing will appear:**
   Frequency exceeds max limit (14285 ms). Time period detected is XX”

7. **Signals and Events of testbench_AT26DFx.v module:**

This module is responsible for sending opcode, address and data. Based on which
particular trigger input asserted, corresponding opcode, address and data transfer will start. For example, if tr_byt_prog is asserted, then the byte programming opcode will be transferred in
serial_in line in serial manner. Then the address obtained through m_address is transmitted serially. data_num will intimate the number of data to be written into the memory. Based on the
data_num value, data in w_data will be transmitted serially through serial_in. The
testbench_AT26DFx.v module has HOLDB control. Hence can be passed based on HOLDB
value.

7.1 **Port Signals:**

<table>
<thead>
<tr>
<th>Port</th>
<th>Signal name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>clk</td>
<td>Clock input</td>
</tr>
<tr>
<td>Input</td>
<td>HOLDB</td>
<td>Hold</td>
</tr>
<tr>
<td>Input</td>
<td>SO_data</td>
<td>SO from BFM</td>
</tr>
<tr>
<td>Input</td>
<td>tr_read_stat</td>
<td>trigger for read status reg</td>
</tr>
<tr>
<td>Input</td>
<td>tr_write_stat</td>
<td>trigger for write status reg</td>
</tr>
<tr>
<td>Input</td>
<td>tr_wr_en</td>
<td>trigger for Write Enable</td>
</tr>
<tr>
<td>Input</td>
<td>tr_wr_dis</td>
<td>trigger for Write Disable</td>
</tr>
<tr>
<td>Input</td>
<td>tr_man</td>
<td>trigger for read Manufacturer ID</td>
</tr>
<tr>
<td>Input</td>
<td>tr_pwr_dwn</td>
<td>trigger for Deep Power-Down</td>
</tr>
<tr>
<td>Input</td>
<td>tr_res_pwr_dwn</td>
<td>trigger for resume from Deep Power-Down</td>
</tr>
<tr>
<td>Input</td>
<td>tr_byt_prog</td>
<td>trigger for Byte programming</td>
</tr>
</tbody>
</table>
### 7.2 Top level signals:

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_status</td>
<td>opcode for read status reg</td>
</tr>
<tr>
<td>write_status</td>
<td>opcode for write status reg</td>
</tr>
<tr>
<td>write_enable</td>
<td>opcode for Write Enable</td>
</tr>
<tr>
<td>write_disable</td>
<td>opcode for Write Disable</td>
</tr>
<tr>
<td>manufacturer</td>
<td>opcode for Manufacturer ID</td>
</tr>
<tr>
<td>deep_power_down</td>
<td>opcode for Deep Power-Down</td>
</tr>
<tr>
<td>res_deep_power</td>
<td>opcode for resume from Deep Power-Down</td>
</tr>
<tr>
<td>byte_program</td>
<td>opcode for Byte programming</td>
</tr>
<tr>
<td>read_array</td>
<td>opcode for Read array</td>
</tr>
<tr>
<td>read_array_l</td>
<td>opcode for Read array in low frequency</td>
</tr>
<tr>
<td>seq_program</td>
<td>opcode for Sequential programming</td>
</tr>
<tr>
<td>protect</td>
<td>opcode for Protect sector</td>
</tr>
<tr>
<td>unprotect</td>
<td>opcode for Unprotect sector</td>
</tr>
<tr>
<td>read_protect</td>
<td>opcode for read protection register</td>
</tr>
<tr>
<td>erase_4</td>
<td>opcode for 4KB Block erase</td>
</tr>
<tr>
<td>erase_32</td>
<td>opcode for 32KB Block erase</td>
</tr>
<tr>
<td>erase_64</td>
<td>opcode for 64KB Block erase</td>
</tr>
<tr>
<td>chip_erase</td>
<td>opcode for Chip erase</td>
</tr>
<tr>
<td>x_val</td>
<td>don’t care values for Read array in low frequency</td>
</tr>
</tbody>
</table>

### 7.3 List of events:

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_rd_array</td>
<td>trigger for Read array</td>
</tr>
<tr>
<td>tr_rd_array_l</td>
<td>trigger for Read array in low frequency</td>
</tr>
<tr>
<td>tr_seq_byt</td>
<td>trigger for Sequential programming</td>
</tr>
<tr>
<td>tr_protect</td>
<td>trigger for Protect sector</td>
</tr>
<tr>
<td>tr_unprotect</td>
<td>trigger for Unprotect sector</td>
</tr>
<tr>
<td>tr_rd_protect</td>
<td>trigger for read protection register</td>
</tr>
<tr>
<td>tr_be4</td>
<td>trigger for 4KB Block erase</td>
</tr>
<tr>
<td>tr_be32</td>
<td>trigger for 32KB Block erase</td>
</tr>
<tr>
<td>tr_be64</td>
<td>trigger for 64KB Block erase</td>
</tr>
<tr>
<td>tr_ce</td>
<td>trigger for Chip erase</td>
</tr>
<tr>
<td>data_num</td>
<td>no. of data to be transmitted/read</td>
</tr>
<tr>
<td>no_addr</td>
<td>No address for consecutive sequential program</td>
</tr>
<tr>
<td>m_address</td>
<td>address for protect/unprotect/read arrays/program/erase</td>
</tr>
<tr>
<td>w_data</td>
<td>write data for Byte programming / Sequential programming</td>
</tr>
<tr>
<td>serial_in</td>
<td>SI for BFM</td>
</tr>
<tr>
<td>out_data</td>
<td>data from read array to top</td>
</tr>
</tbody>
</table>
### Event Purpose

<table>
<thead>
<tr>
<th>Event</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>read_stat</td>
<td>read status reg</td>
</tr>
<tr>
<td>wr_en</td>
<td>Write Enable</td>
</tr>
<tr>
<td>wr_dis</td>
<td>Write Disable</td>
</tr>
<tr>
<td>manufact</td>
<td>Manufacturer ID</td>
</tr>
<tr>
<td>pwr_dwn</td>
<td>Deep Power-Down</td>
</tr>
<tr>
<td>res_pwr_dwn</td>
<td>resume from Deep Power-Down</td>
</tr>
<tr>
<td>byt_prog</td>
<td>Byte programming</td>
</tr>
<tr>
<td>rd_array</td>
<td>Read array</td>
</tr>
<tr>
<td>rd_array_l</td>
<td>Read array for low frequency</td>
</tr>
<tr>
<td>seq_byte</td>
<td>Sequential programming</td>
</tr>
<tr>
<td>protect_sector</td>
<td>Protect sector</td>
</tr>
<tr>
<td>unprotect_sector</td>
<td>Unprotect sector</td>
</tr>
<tr>
<td>read_sector</td>
<td>read protection register</td>
</tr>
<tr>
<td>write_stat</td>
<td>write status reg</td>
</tr>
<tr>
<td>erase4</td>
<td>4KB Block erase</td>
</tr>
<tr>
<td>erase32</td>
<td>32KB Block erase</td>
</tr>
<tr>
<td>erase64</td>
<td>64KB Block erase</td>
</tr>
<tr>
<td>erase_chip</td>
<td>Chip erase</td>
</tr>
</tbody>
</table>

#### 7.4 List of parameters:

- **tDS**: Data in Setup time
- **tDH**: Data in Hold time

#### 8. Signals and Events of topx_testbench.v module:

topx_testbench.v module is responsible for selecting the device for which the model has to work, sending opcode, address and data in SI line and obtaining the read array data from SO line. It also passes all the parameter needed for the device model (AT26DFxxx.v) internally. It contains all the test cases to test the model for each device model (AT25DF041A, AT26DF081A, AT26DF161A, and AT26DF321) and also for all the possibilities of error conditions.

#### 8.1 Top level signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>trg_read_stat</td>
<td>trigger for read status reg</td>
</tr>
<tr>
<td>trg_write_stat</td>
<td>trigger for write status reg</td>
</tr>
<tr>
<td>trg_wr_en</td>
<td>trigger for Write Enable</td>
</tr>
<tr>
<td>trg_wr_dis</td>
<td>trigger for Write Disable</td>
</tr>
<tr>
<td>trg_man</td>
<td>trigger for read Manufacturer ID</td>
</tr>
</tbody>
</table>
### Signal | Purpose
--- | ---
trg_pwr_dwn | trigger for Deep Power-Down
trg_res_pwr_dwn | trigger for resume from Deep Power-Down
trg_byt_prog | trigger for Byte programming
trg_rd_array | trigger for Read array
trg_rd_array_l | trigger for Read array in low frequency
trg_seq_byt | trigger for Sequential programming
trg_protect | trigger for Protect sector
trg_unprotect | trigger for Unprotect sector
trg_rd_protect | trigger for read protection register
trg_be4 | trigger for 4KB Block erase
trg_be32 | trigger for 32KB Block erase
trg_be64 | trigger for 64KB Block erase
trg_ce | trigger for Chip erase
t_data_num | no. of data to be transmitted/read
t_no_addr | No address for consecutive sequential programming
t_address | address for protect/unprotect/read arrays/programming/erase
t_data | write data for Byte programming / Sequential programming
CSB_out | CSB signal for BFM
WPB_out | WPB signal for BFM
HOLDB_out | HOLDB signal for BFM
SO_in | SO from BFM
SI_out | SI for BFM
SCK_out | SCK for BFM

### 8.2 List of parameters:

### Parameter | Purpose
--- | ---
frDLF | SCK Frequency for read Array (Low freq - 03h opcode)
frSCK | Serial clock (SCK) Frequency in MHz
frSCKH | SCK High time
frSCKL | SCK Low time
frCSH | Chip Select high time
frCSLS | Chip Select Low Setup time
frCSLH | Chip Select Low hold time
frCSHS | Chip Select high Setup time
frCSHH | Chip Select high hold time
tDS | Data in Setup time
tDH | Data in Hold time
tHLS | HOLD! Low Setup Time
tHHS | HOLD! High Setup Time
tHLH | HOLD! Low Hold Time
tHHH | HOLD! High Hold Time
tWPS | Write Protect Setup Time (only when SPRL=1)
tWPH | Write Protect Hold Time (only when SPRL=1)
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>tWRSR</td>
<td>Write Status Register Time</td>
</tr>
<tr>
<td>tSECP</td>
<td>Sector Protect Time</td>
</tr>
<tr>
<td>tSECUP</td>
<td>Sector Unprotect Time</td>
</tr>
<tr>
<td>tEDPD</td>
<td>Chip Select high to Deep Power-down (3 us)</td>
</tr>
<tr>
<td>tRDPD</td>
<td>Chip Select high to Stand-by Mode</td>
</tr>
<tr>
<td>tPP</td>
<td>Page Program Time</td>
</tr>
<tr>
<td>tBP</td>
<td>Byte Program Time</td>
</tr>
<tr>
<td>tBLKE4</td>
<td>Block Erase Time 4-kB (0.350 sec)</td>
</tr>
<tr>
<td>tBLKE32</td>
<td>Block Erase Time 32-kB</td>
</tr>
<tr>
<td>tBLKE64</td>
<td>Block Erase Time 64-kB</td>
</tr>
<tr>
<td>tCHPE</td>
<td>Chip erase time</td>
</tr>
<tr>
<td>DEVICE</td>
<td>Specifies the selected card model</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>Memory preloading condition</td>
</tr>
<tr>
<td>MEMORY_FILE</td>
<td>Memory preload file</td>
</tr>
</tbody>
</table>

8.3 List of Switches:

<table>
<thead>
<tr>
<th>Switch</th>
<th>Switch status</th>
</tr>
</thead>
<tbody>
<tr>
<td>041</td>
<td>Selects device model for device AT25DF041A</td>
</tr>
<tr>
<td>081</td>
<td>Selects device model for device AT26DF081A</td>
</tr>
<tr>
<td>161</td>
<td>Selects device model for device AT26DF161A</td>
</tr>
<tr>
<td>321</td>
<td>Selects device model for device AT26DF321</td>
</tr>
<tr>
<td>MODE3</td>
<td>Selects SPI mode 3</td>
</tr>
<tr>
<td>LOAD</td>
<td>Selects Memory preloading with the content in memory.txt</td>
</tr>
</tbody>
</table>

Any one of the 4 switches (041 / 081 / 161 / 321) has to be selected for proper selection of device parameters in AT26DFxxx.v model.

For SPI mode of operation, default is SPI mode0. If switch MODE3 is selected, then SPI mode3 operation will be carried out.

LOAD switch can be used to preload the card memory with values specified in memory.txt file. If LOAD switch is not used, then memory is preloaded with erased state.

For selecting a particular ATMEL DataFlash model, only any one of 4 switches need to be selected. All other parameters like maximum frequency, timing for SI, SO, HOLDB, CSB, and WPB will internally passed to the corresponding modules.

9. Tasks used in topx_testbench.v module:

For deep power down, resume from deep power down, read manufacturer ID, chip erase, write enable and write disable only opcode need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after 2*8*tSCKH2 (+ *32*tSCKH for read
manufacturer ID alone). For deep power down, resume from deep power down and chip erase, additional delay of tEDPD, tRDPD, tCHPE is given respectively to complete the operation.

For **read array in low and high frequency**, opcode and address need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH + 2*24*tSCKH\) this delay is for read array in high frequency alone. Not for low frequency)+ the time to receive data in SO.

For **protect, unprotect and read sector protection**, opcode and address need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH + 2*24*tSCKH\) (the time to receive data in SO (for read sector protection)). Additional delay of tSECP (for protect sector) or tSECUP (for unprotect sector).

For **4KB erase, 32KB erase and 64KB erase**, opcode and address need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH + 2*24*tSCKH\). Additional delay of tBLKE4 (4KB erase) or tBLKE32 (for 32KB erase) or tBLKE64 (for 64KB erase) is issued to complete the operation.

For **Byte/Page programming**, opcode, address and data need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH + 2*24*tSCKH + 2*(t\_data\_num)*tSCKH\) (this delay is for sending data in SI). Additional delay of tPP is issued to complete the operation. t_data_num represents the number of data that is going to be transferred.

For **Sequential Byte programming**, for the first time, opcode, address and data need to be supplied. Hence particular trigger signal is asserted and deasserted after \(2*8*tSCKH + 2*24*tSCKH + 2*(t\_data\_num)*tSCKH\) (this delay is for sending data in SI). Additional delay of tBP is issued to complete the operation. For the second time onwards opcode and data need to be supplied. Hence particular trigger signal is asserted and deasserted after \(2*8*tSCKH + 2*8*tSCKH\) (this delay is for sending data in SI). Additional delay of tBP is issued to complete the operation.

For **write status register**, opcode and data need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH(\text{opcode}) + 2*8*tSCKH(\text{data})\). Additional delay of tWRSR is issued to complete the operation.

For **read status register**, only opcode need to be supplied. Hence particular trigger signal and CSB are asserted and deasserted after \(2*8*tSCKH(\text{opcode}) + 2*(\text{min} 1 \text{ and max as needed})*tSCKH\).

**10. Status and Error printing of topx_testbench.v module:**

For data validation any on of these two prints will appear.
If written data and read data are correct,

```
“Write Data: XX, Read Data: XXXXXX Valid Data received”
```
If written data and read data are incorrect,
“Write Data: XX, Read Data: XXXXXX Invalid Data received”

11. Instantiation of memory model and test cases in different EDA Tools:

The verilog memory model doesn't need any change for instantiating in both ModelSim and in VCS. Only the script need to be changed. All the verilog files (device model, testbench and top) should be kept in a folder and filelist.v file should contain path to access these three files. For example if all these 3 files kept in a folder, filelist.v can contain:

```
top26x_testbench.v
AT26DFxxx.v
testbench_AT26DFx.v
```

For memory preloading condition, memory.txt should be kept at the working directory, where compilation and simulation takes place.

11.1 For VCS all possible scripts are listed below:

For SPI MODE0 operation

```vcs -R -PP +define+041 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT25DF041A in SPI Mode0

```vcs -R -PP +define+081 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF081A in SPI Mode0

```vcs -R -PP +define+161 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF161A in SPI Mode0

```vcs -R -PP +define+321 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF321 in SPI Mode0

For SPI MODE3 operation

```vcs -R -PP +define+041+MODE3 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT25DF041A in SPI Mode3

```vcs -R -PP +define+081+MODE3 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF081A in SPI Mode3

```vcs -R -PP +define+161+MODE3 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF161A in SPI Mode3

```vcs -R -PP +define+321+MODE3 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF321 in SPI Mode3

For card memory preloading, few example is given below:

```vcs -R -PP +define+321+LOAD +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF321 in SPI Mode0 with memory preloaded with data in memory.txt.

```vcs -R -PP +define+081+MODE3 +vcs+lic+wait -f filelist.v -l at26x.log```
above used for compilation and simulation for device AT26DF321 in SPI Mode3 with memory preloaded with data in memory.txt.
above used for compilation and simulation for device AT26DF081A in SPI Mode3 with memory preloaded with data in memory.txt.

### 11.2 For ModelSim all possible scripts are listed below:

For the first time these two lines need to be executed for creating work library.

- **vlib** work
- **vmap** work

For SPI MODE0 operation

- **vlog** +define+041 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT25DF041A in SPI Mode0
- **vlog** +define+081 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF081A in SPI Mode0
- **vlog** +define+161 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF161A in SPI Mode0
- **vlog** +define+321 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF321 in SPI Mode0

For SPI MODE3 operation

- **vlog** +define+041 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT25DF041A in SPI Mode3
- **vlog** +define+081 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF081A in SPI Mode3
- **vlog** +define+161 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF161A in SPI Mode3
- **vlog** +define+321 -novopt -f fileslist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF321 in SPI Mode3

For card memory preloading, few example is given below:

- **vlog** +define+161+LOAD -novopt -f filelist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF321 in SPI Mode0 with memory preloaded with data in memory.txt.
- **vlog** +define+041+MODE3+LOAD -novopt -f filelist.v +acc -R -c -do "run -all" -l at26x.log - above used for compilation and simulation for device AT26DF081A in SPI Mode3 with memory preloaded with data in memory.txt.