**Tasks**

1. Breakdown resources and tasks
   1. Download PMOD documentation from Digilent, schematic of PMOD, and SSD1331 OLED controller reference manual
   2. Determine I/O of PMOD
   3. Determine how to startup, turn off, and reset controller
   4. Determine different available commands
2. FPGA MOSI SPI interface
   1. Drive FPGA as master to control chip select, MOSI, and slave clock
   2. Should send 1 byte of data, or 2 hex values at a time
   3. Should drive D/C (data/command control) bit at start of every byte sent
3. MOSI Byte Buffer
   1. Take in X bytes to transmit concurrently, with individual D/C bits for each
4. Shape Wrapper
   1. Startup
   2. Draw a line
   3. Draw a rectangle
   4. Clear window
   5. Reset
5. Wrapper for SPI interface to change color of drawn shapes
6. Write text, update over certain amount of time
   1. Create table for ASCII values

**Nbit\_MOSI\_SPI**

Function

* Load 1 byte of data and serial transmit from MSB to LSB over o\_MOSI
* Drive o\_CS and o\_DC on neg edge so will be stable for slave to read

Parameter

* WIDTH: default 8, N bits to transmit in one set

Input

* I\_RST: asynchronous reset
* I\_SCK: clock for slave interface, has to transmit on it
* I\_DATA: byte to transmit to OLED interface slave (can either be command or data)
* I\_START: begin transmitting byte in i\_DATA
* I\_DC: D/C control to be updated on negedge at same time as data on MOSI line

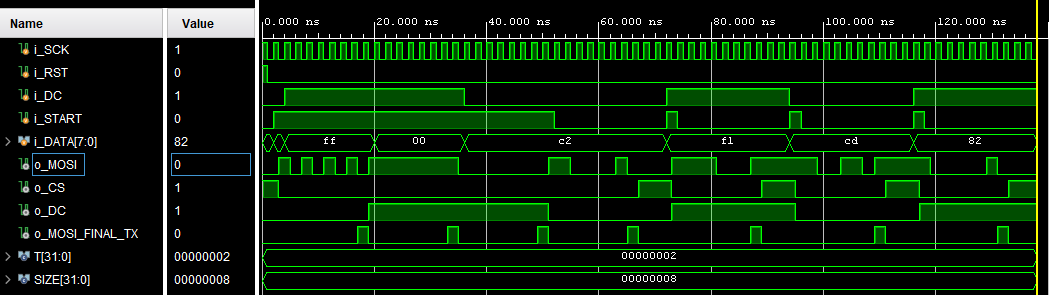
Output

* O\_MOSI: Output 1 bit at a time of i\_DATA, starting with MSB
* O\_CS: chip select, active low
* O\_DC: D/C to control data vs. command for the OLED controller
* O\_MOSI\_FINAL\_TX: asserted when last bit transmitted

States

* Idle: waiting for i\_START to be asserted once to transmit byte
* Transmit: transmit byte, stay in state if i\_START asserted by end, otherwise back to idle

**Nbit\_MOSI\_SPI\_tb**



**Nbyte\_MOSI\_SPI\_buffer**

Take in X inputs parallel, serially load them and drive start of master SPI MOSI

Paramters

* WIDTH: number of bits to transmit in one set
* N: max # of sets that can be parallel loaded

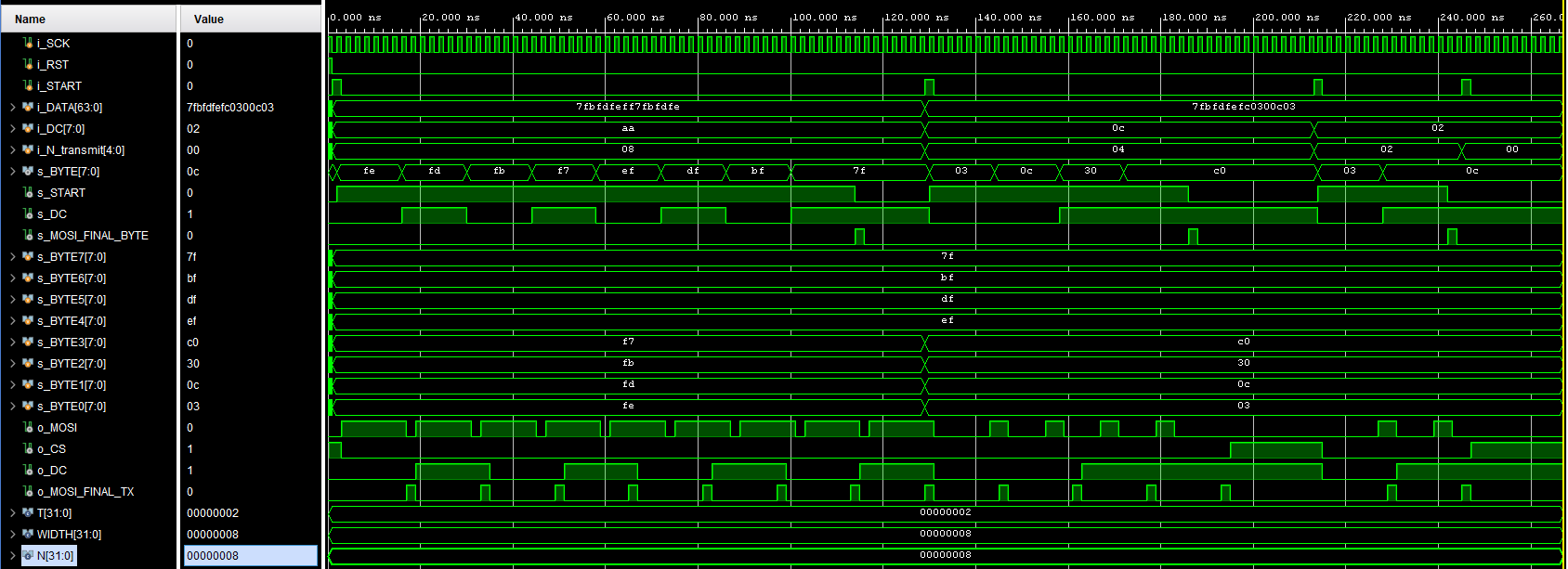
Inputs

* I\_SCK: SPI clock
* I\_RST: asynch reset
* [WIDTH\*N-1:0] i\_DATA: 2D bit array holding N sets to transmit over MOSI
* [N-1:0] i\_DC: D/C command for each byte
* I\_START: start from driver control, initiate parallel load of bytes
* [4:0] i\_N\_transmit: # of N bytes to transmit on load, will be counted

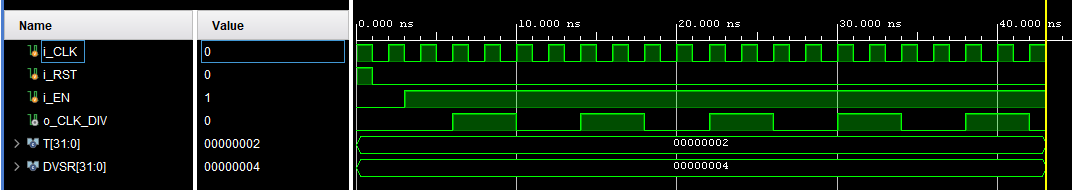
Outputs

* [WIDTH-1:0] o\_DATA: byte to feed to master SPI shift module
* O\_START: start bit to transmit another byte over MOSI, keep on to keep going
* O\_DC: D/C read from input DC for each respective byte
* O\_MOSI\_FINAL\_BYTE: final byte to transmit

**Nbyte\_MOSI\_SPI\_buffer\_tb**



**Clock Divider**



**OLED Interface wrapper**

Paramters

* WIDTH: number of bits to transmit in one set
* N: max # of sets that can be parallel loaded

Input

* I\_CLK: FPGA clock
* I\_RST: asynch reset
* [1:0] I\_MODE: display mode select, will determine state machine load
* I\_START: load bytes and transmit
* I\_ROW1: ROW1 byte data
* I\_ROW2: ROW2 byte data
* I\_ROW3: ROW3 byte data
* I\_COL1: COL1 byte data
* I\_COL2: COL2 byte data
* I\_COL3: COL3 byte data

Output

* O\_CS: Active low chip select
* O\_MOSI: Master out slave in serial bit
* O\_SCK: Slave clock
* O\_D/C: Data/Command control
* O\_RES: power reset
* O\_VCCEN: VCC Enable
* O\_PMODEN: Vdd Logic Voltage control

Uses the Nbit\_MOSI\_SPI to send serial bits to OLED display over MOSI. Also drives CS and D/C. This will include a clock divider to count/utilize the FPGA clock and drive O\_SCK for the byte buffer and slave OLED display.