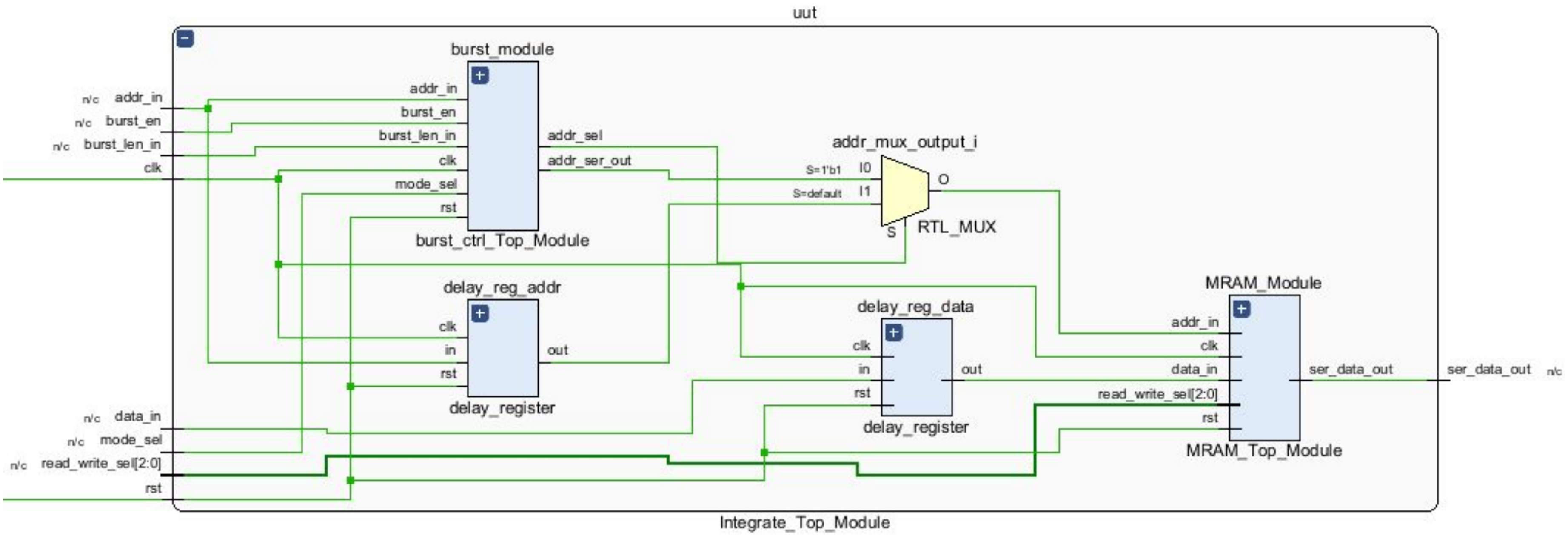
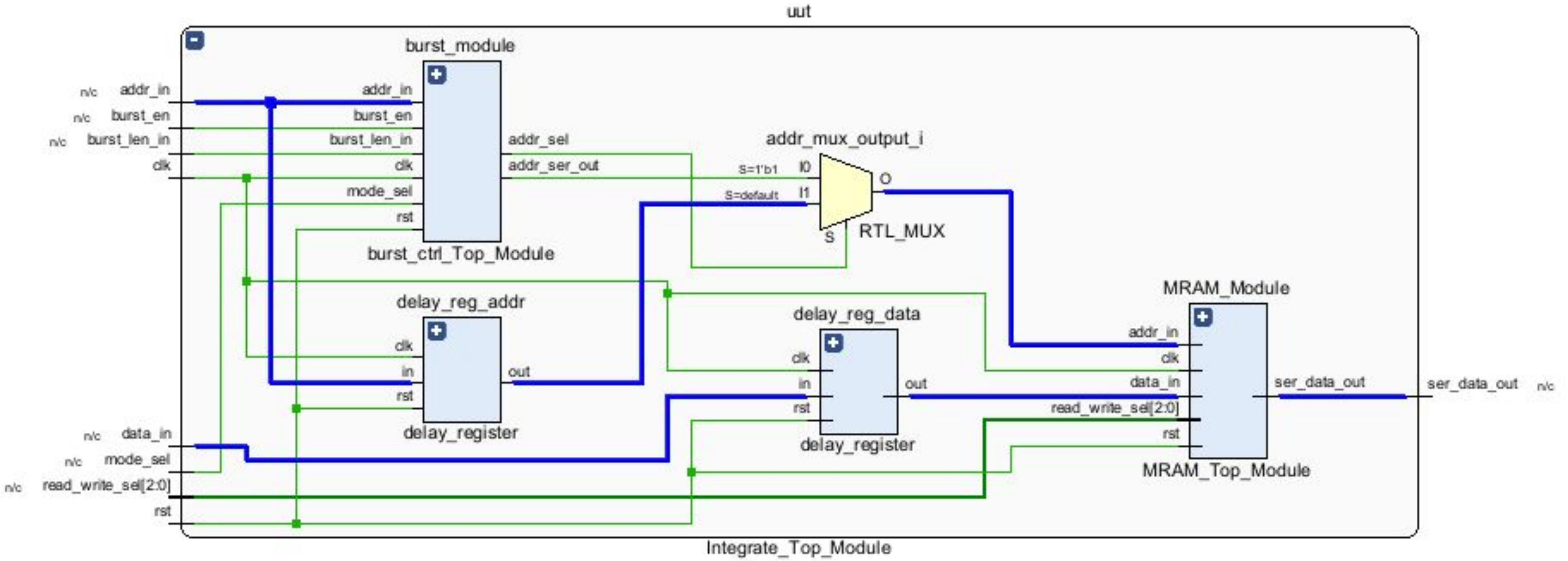
Timing analysis

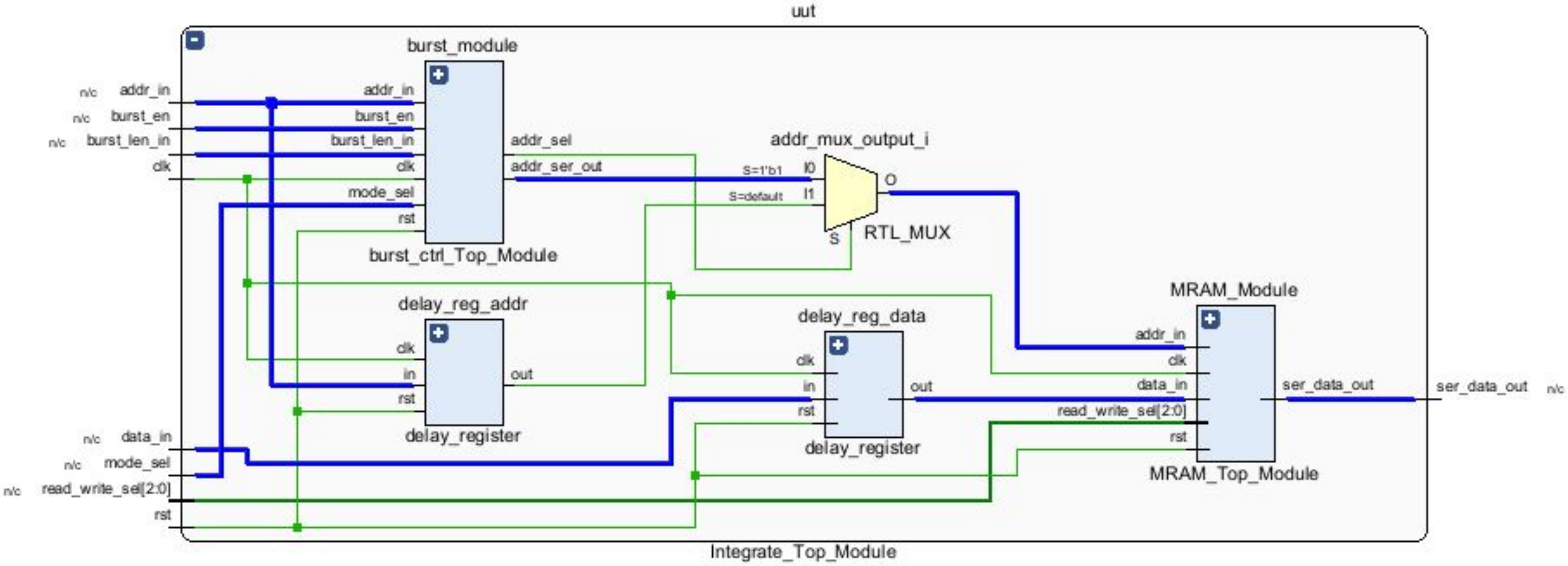
Integrated Overview of Controller



During Single Transfer / First loop of Burst Transfer



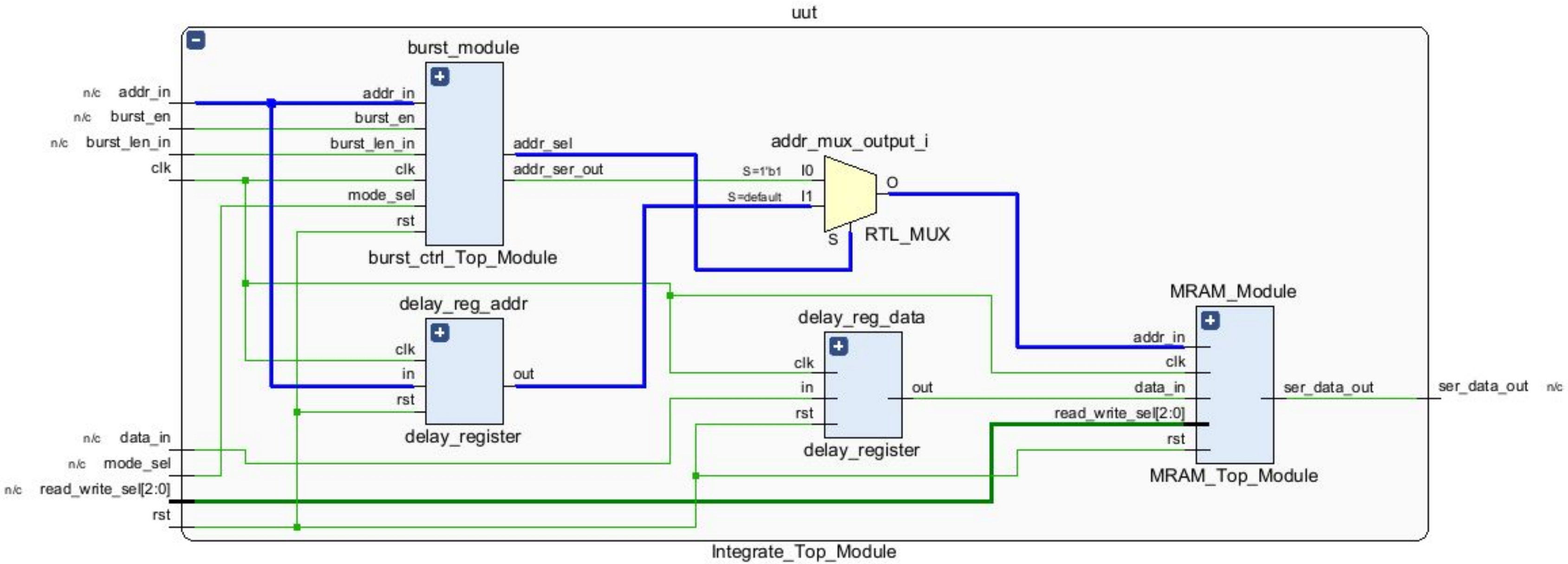
During Burst Transfer



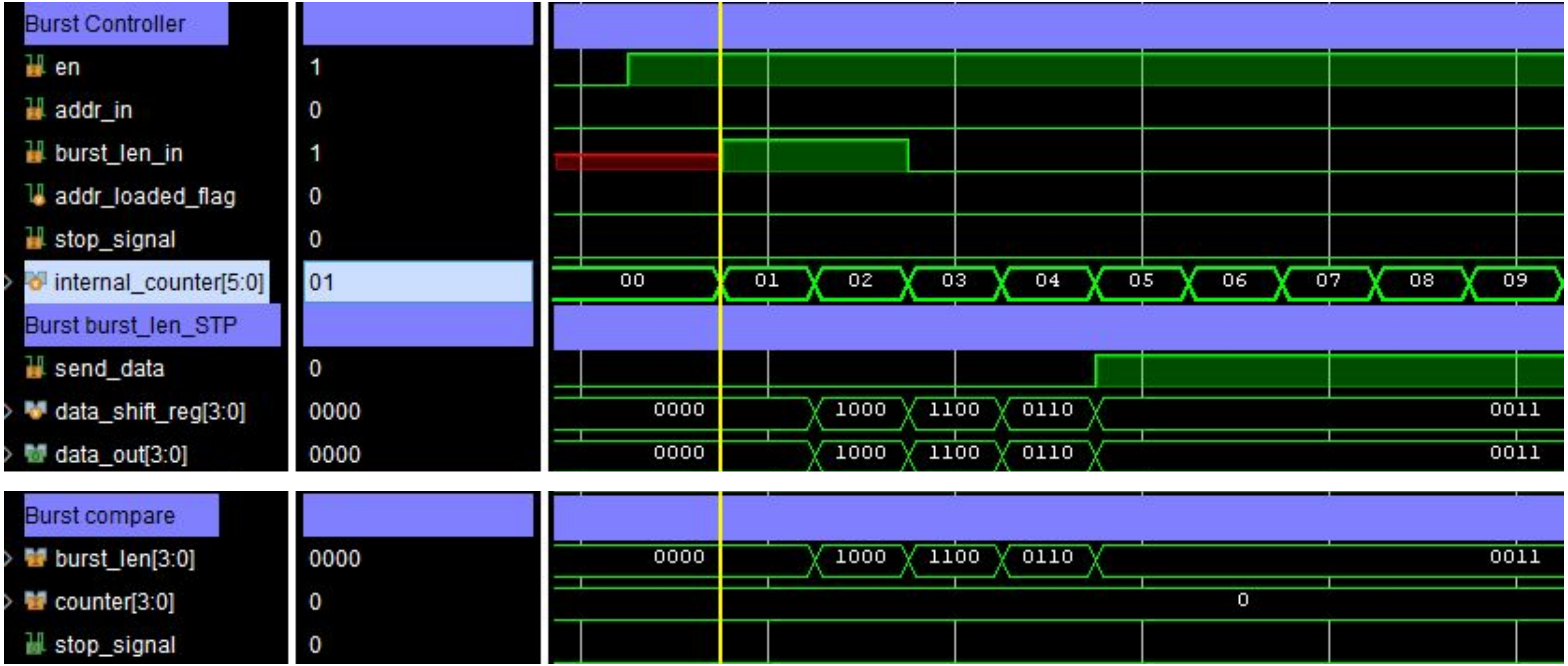
Burst Module

If Single Transfer is selected, don’t do anything.

Done through setting the mux to 0.



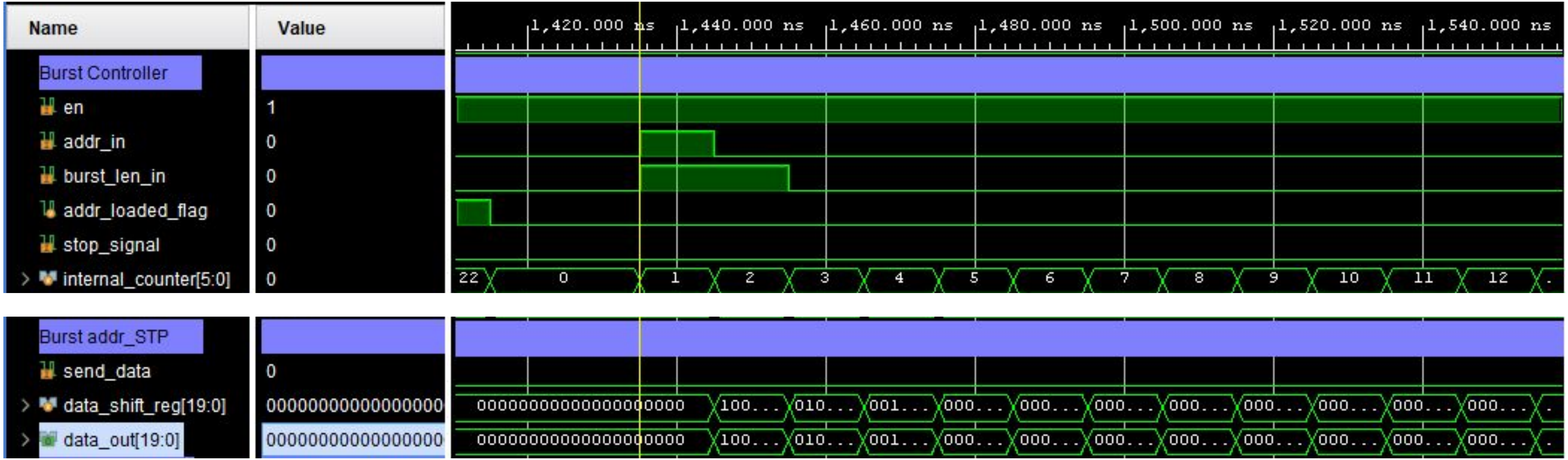
Burst Module - Burst len



^burst len gets fed in during counter = 1 and gets shifted into the STP module at counter = 2

At counter=5, burst_len has been fully shifted in.

Burst Module - Initial addr

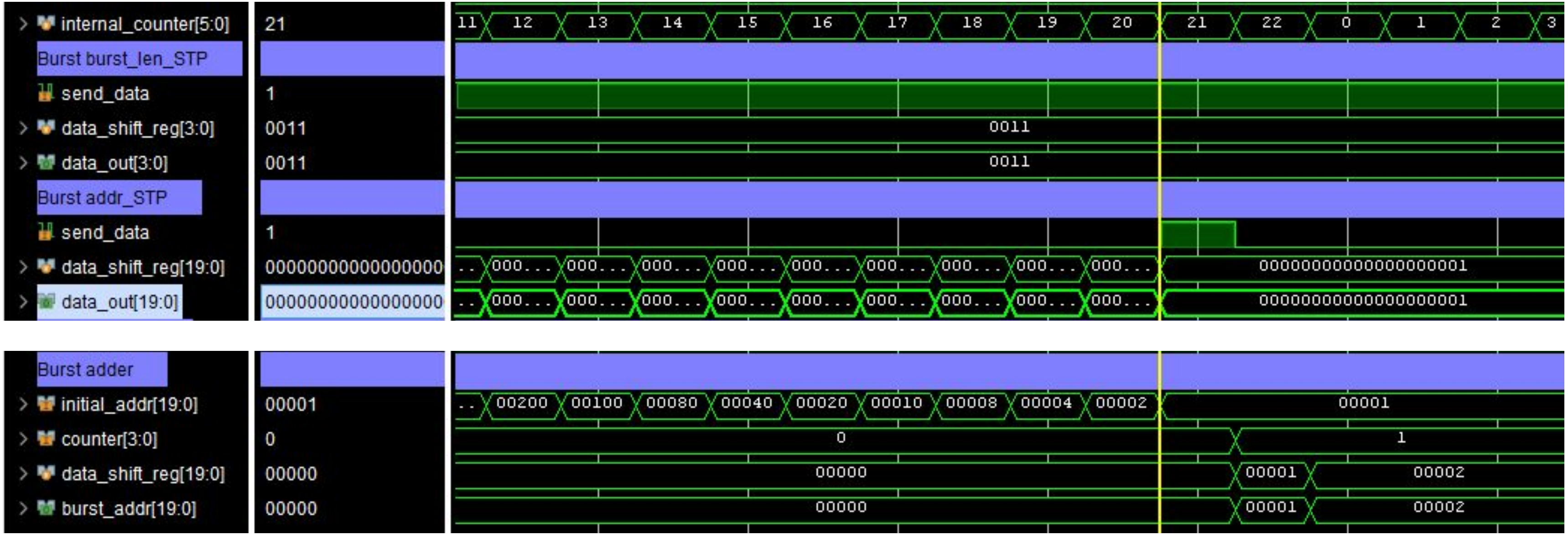


Counter 0: Enable gets pulled high,

Counter 1: Addr gets fed in,

Counter 2: STP module starts to move this data into internal shift reg,

Burst Module - Initial addr



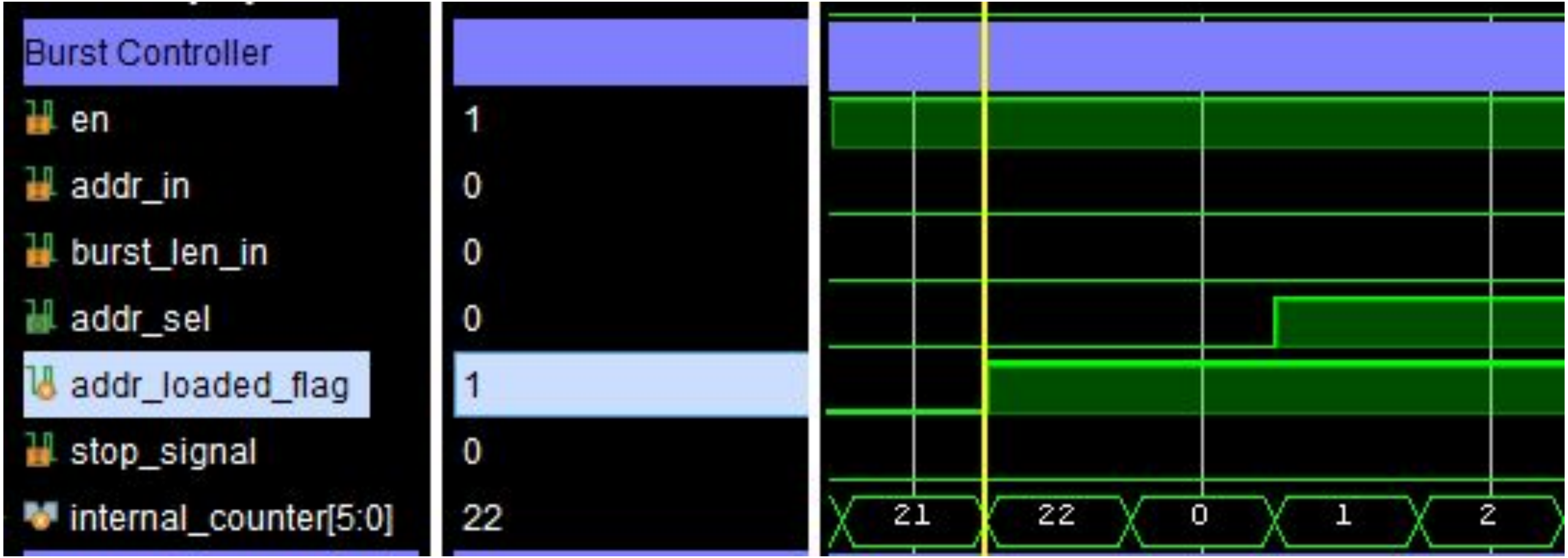
```
internal_counter[5:0]   21
Burst burst_len_STP
send_data               1
data_shift_reg[3:0]     0011
data_out[3:0]           0011
Burst addr_STP
send_data               1
data_shift_reg[19:0]    00000000000000000
data_out[19:0]          00000000000000000

Burst adder
initial_addr[19:0]      00001
counter[3:0]            0
data_shift_reg[19:0]    00000
burst_addr[19:0]        00000
```

Counter 21: STP module has fully read in the addr and sends it to adder module

Burst Module - Initial addr

After initial addr is loaded, set the loaded flag and addr_sel(mux control)

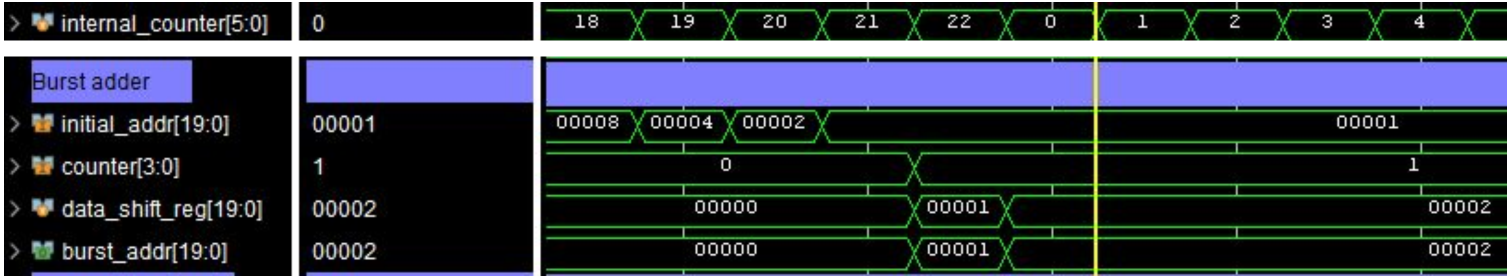
Addr read for the STP/PTS module starts when counter = 2



```
Burst Controller
en                      1
addr_in                 0
burst_len_in            0
addr_sel                0
addr_loaded_flag        1
stop_signal             0
internal_counter[5:0]   22
```

Burst Module - Adder

Takes initial addr and counter module’s value and adds them tgt

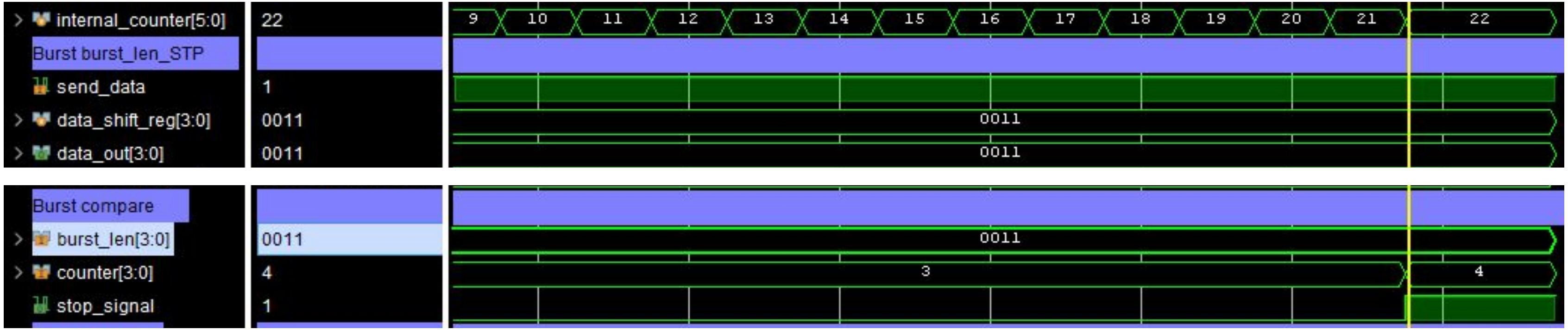


Counter 22: Initial addr and counter are both available in the adder module

Counter 0 of next loop: Adds initial address and counter value together

Burst Module - compare module

Cycle 22: When compare module detects that the counter value exceeds the burst len, assert stop signal to the burst control to stop sending address serially



internal_counter[5:0] 22

Burst burst_len_STP

send_data 1

data_shift_reg[3:0] 0011

data_out[3:0] 0011

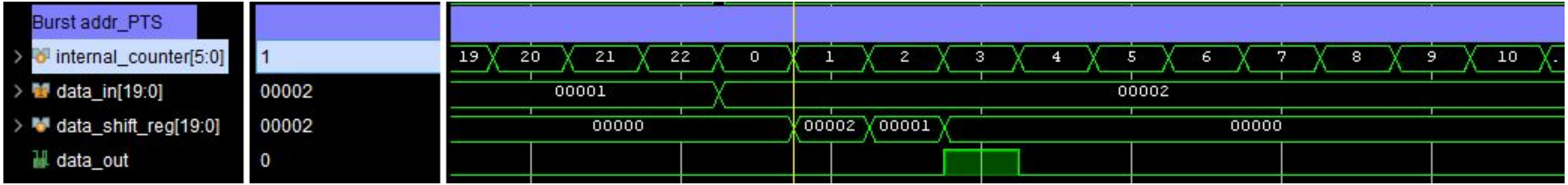
Burst compare

burst_len[3:0] 0011

counter[3:0] 4

stop_signal 1

Burst Module - address PTS module

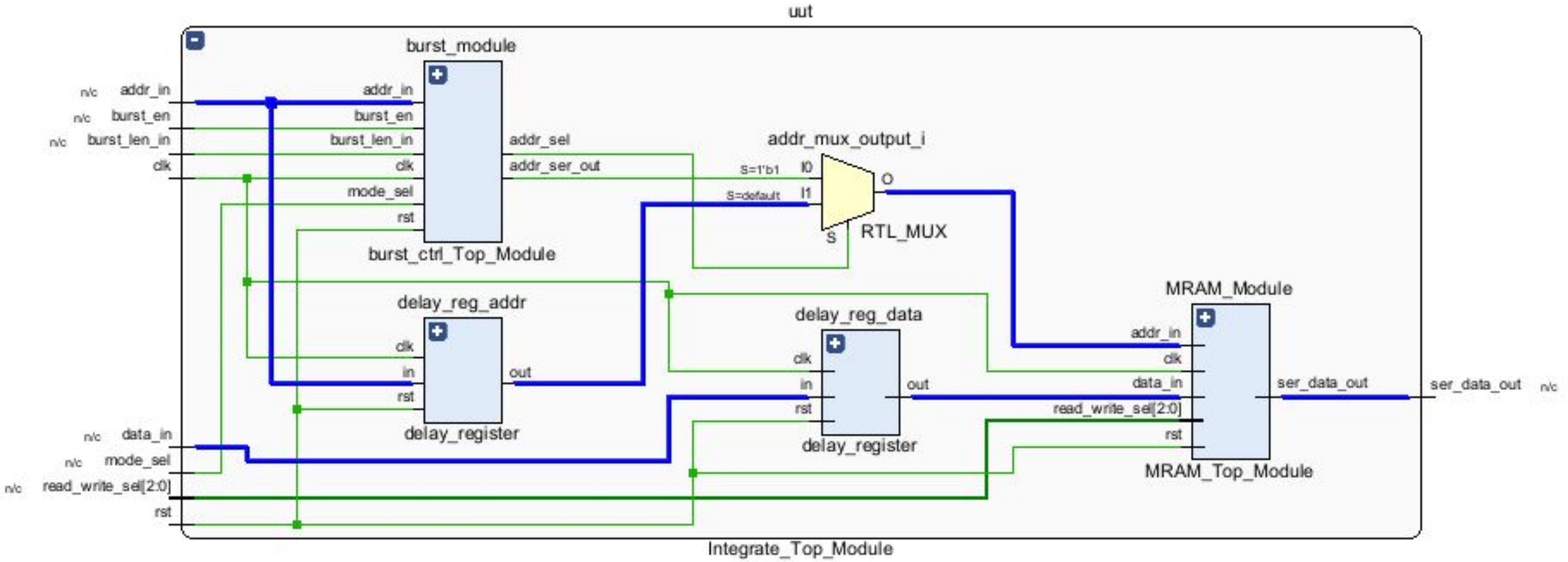


Counter 1: Next address to be accessed is loaded into the internal shift register

Counter 2: Starts to output the address serially. Address to be output here is 2 (from LSB to MSB)

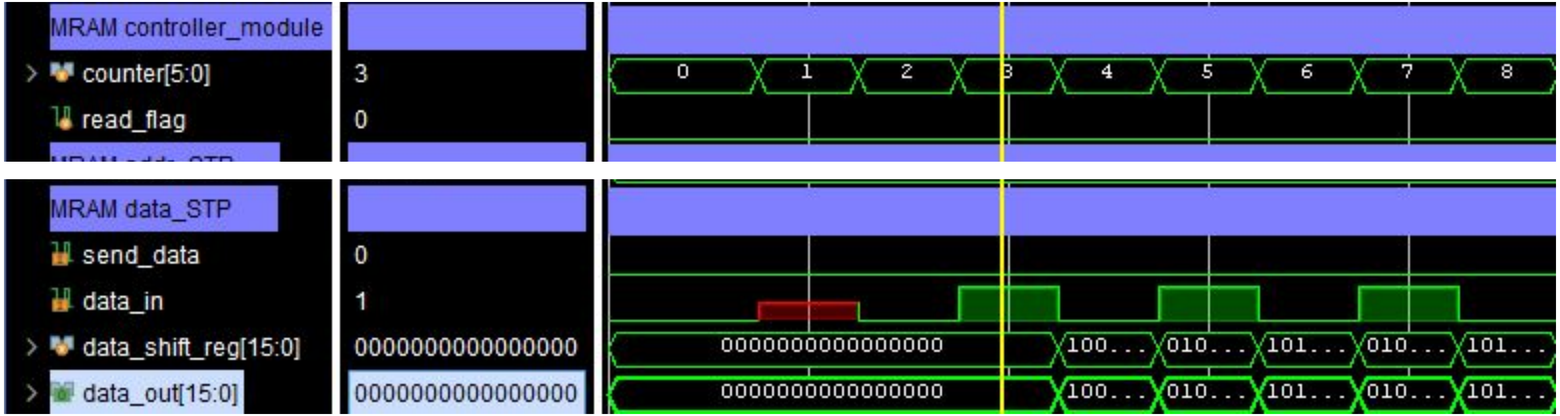
- Counter 2: output bit 0
- Counter 3: output bit 1
- Counter 4 onwards: output bit 0
- Overall: 20’b0000_0000_0000_0000_0010

MRAM STP/PTS module -



MRAM STP/PTS module - data/addr STP module (write op)

At counter = 2, data gets read and takes 1 cycle to be shifted into the internal shift register



MRAM controller_module

counter[5:0] 3

read_flag 0

MRAM data_STP

send_data 0

data_in 1

data_shift_reg[15:0] 0000000000000000

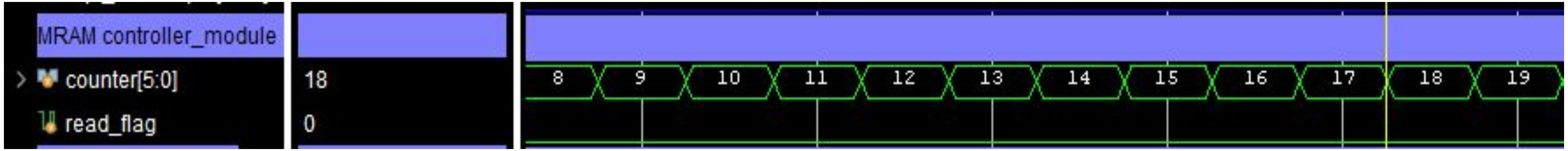
data_out[15:0] 0000000000000000

MRAM STP/PTS module - data/addr STP module (write op)

At counter = 18, data is fully in the internal shift register.

At counter = 22, addr is fully in the internal shift register.

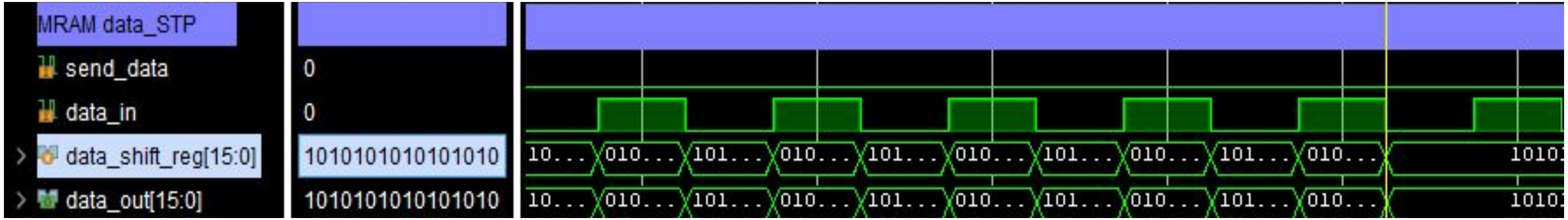
Can be sent to the MRAM alongside the control signals



MRAM controller_module

counter[5:0] 18

read_flag 0



MRAM data_STP

send_data 0

data_in 0

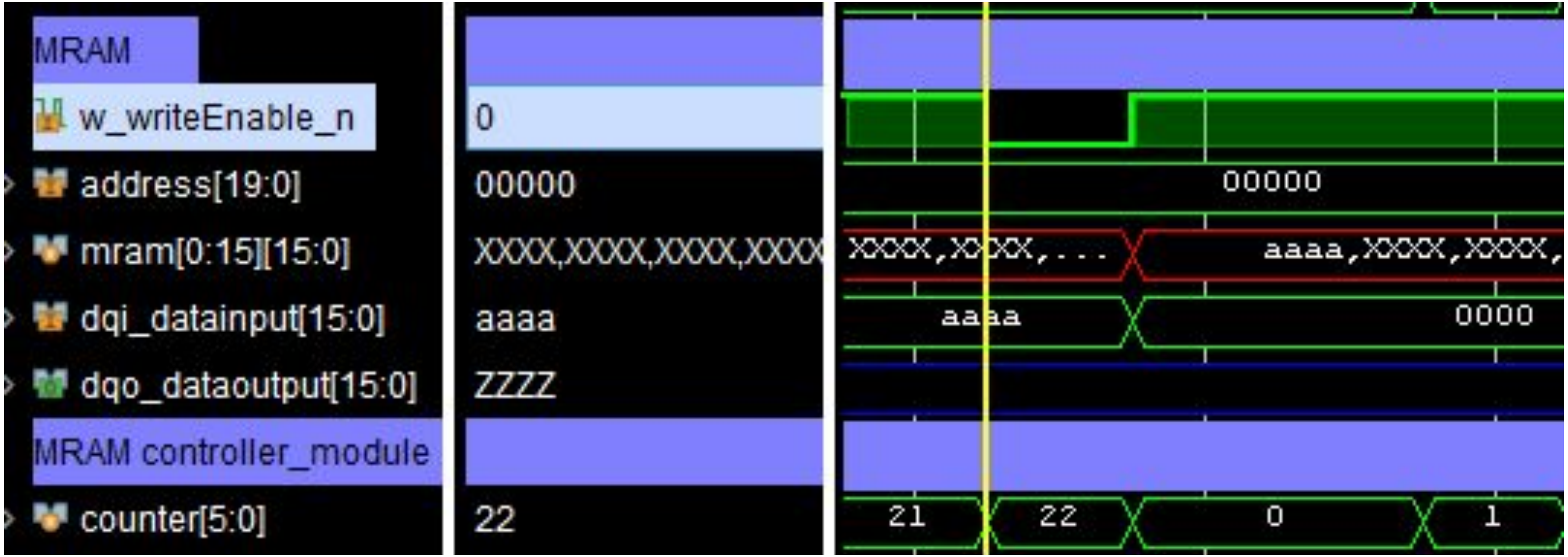
data_shift_reg[15:0] 1010101010101010

data_out[15:0] 1010101010101010

MRAM STP/PTS module - addr STP module (write op)

At counter = 22, signals are asserted to write to the MRAM

At counter = 0 of the next cycle, it is in the MRAM

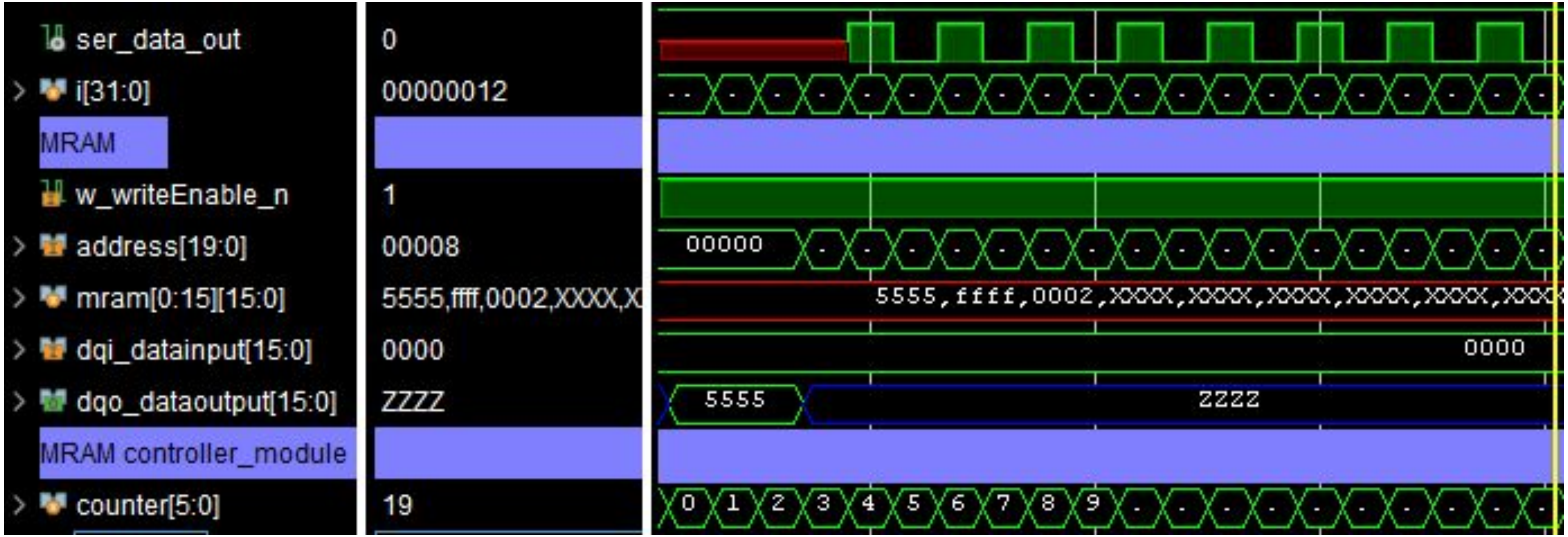


MRAM	
w_writeEnable_n	0
address[19:0]	00000
mram[0:15][15:0]	XXXX,XXXX,XXXX,XXXX
dqi_datainput[15:0]	aaaa
dqo_dataoutput[15:0]	ZZZZ
MRAM controller_module	
counter[5:0]	22

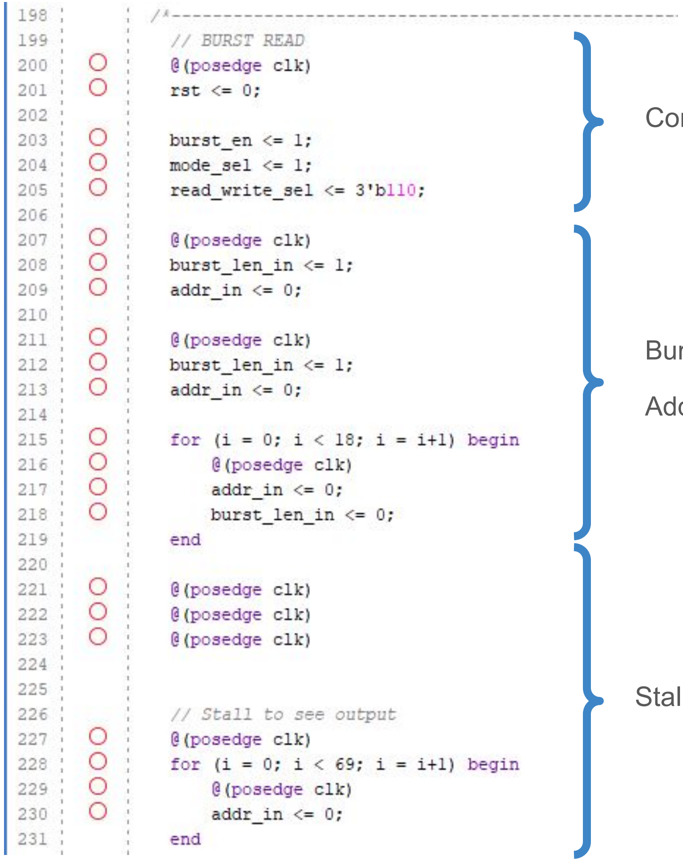
MRAM STP/PTS module - Read operation

Data is read out in the next loop after address is provided

Addr = 0; data in address 0 is 5555h, 0101_0101_0101_0101b. Data is output from LSB to MSB



Testbench - burst read example



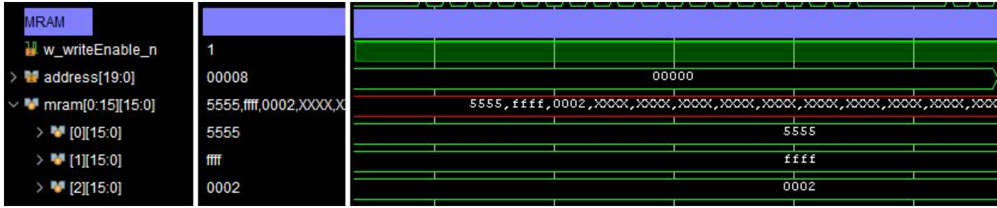
```verilog
/*--------------------------------------------------------
    // BURST READ
    @(posedge clk)
    rst <= 0;

    burst_en <= 1;
    mode_sel <= 1;
    read_write_sel <= 3'b110;

    @(posedge clk)
    burst_len_in <= 1;
    addr_in <= 0;

    @(posedge clk)
    burst_len_in <= 1;
    addr_in <= 0;

    for (i = 0; i < 18; i = i+1) begin
        @(posedge clk)
        addr_in <= 0;
        burst_len_in <= 0;
    end

    @(posedge clk)
    @(posedge clk)
    @(posedge clk)


    // Stall to see output
    @(posedge clk)
    for (i = 0; i < 69; i = i+1) begin
        @(posedge clk)
        addr_in <= 0;
    end
```

Control signals

Burst len = 3

Addr = 0

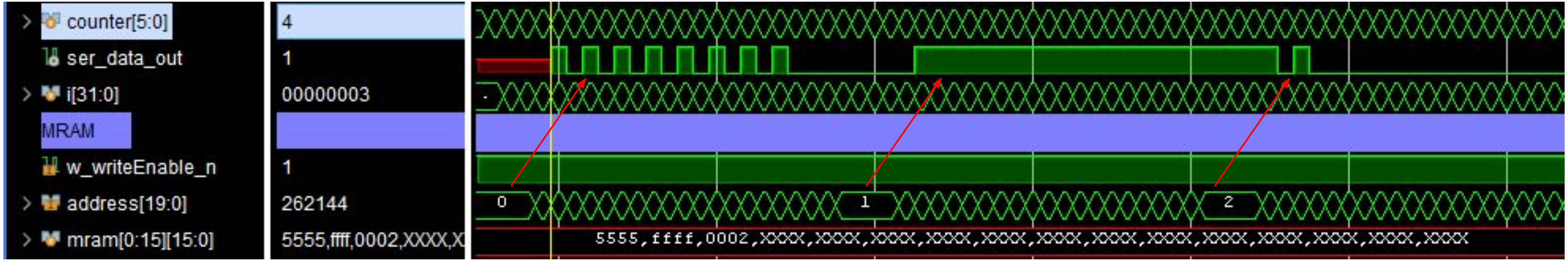
Stall to see output



```
MRAM
w_writeEnable_n      1
address[19:0]        00008
mram[0:15][15:0]     5555,ffff,0002,XXXX,X
  [0][15:0]          5555
  [1][15:0]          ffff
  [2][15:0]          0002
```

00000
5555,ffff,0002,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXX
5555
ffff
0002

Testbench - burst read example



counter[5:0]	4
ser_data_out	1
i[31:0]	00000003
MRAM	
w_writeEnable_n	1
address[19:0]	262144
mram[0:15][15:0]	5555,ffff,0002,XXXX,X

-
0
1
2
5555,ffff,0002,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX

Integrated module summary

1 loop consists of 23 cycles (0-22)

Writing:

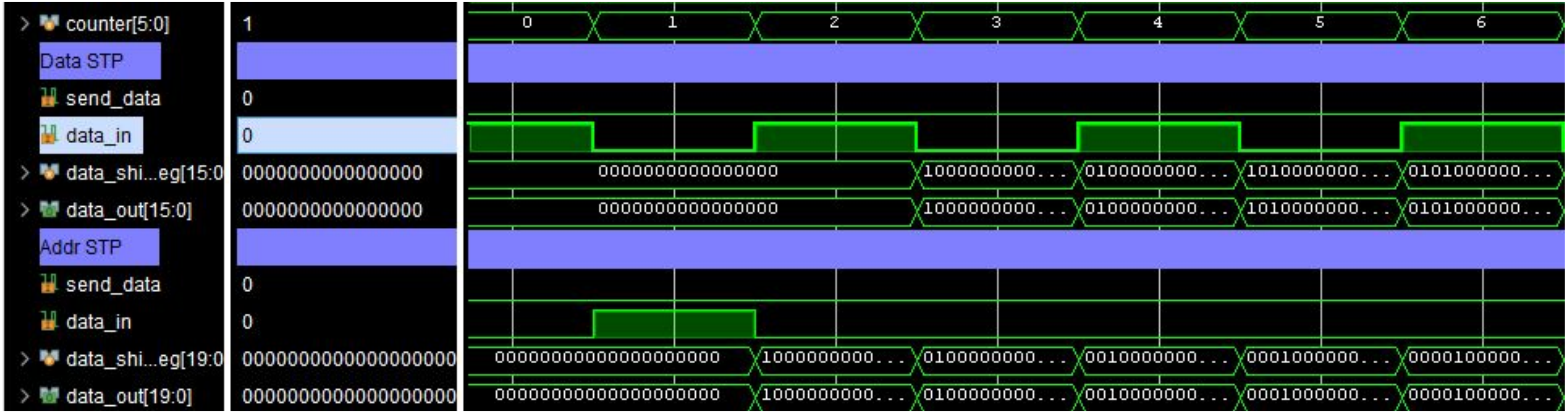
- Takes 23 cycles

Reading:

- Takes 23 cycles to provide address
- Data is output on the next loop, cycle 4 to 19

MRAM STP/PTS module standalone

Data is being shifted into the STP module's shift register during counter 1 instead of counter 2 as opposed to the integrated module

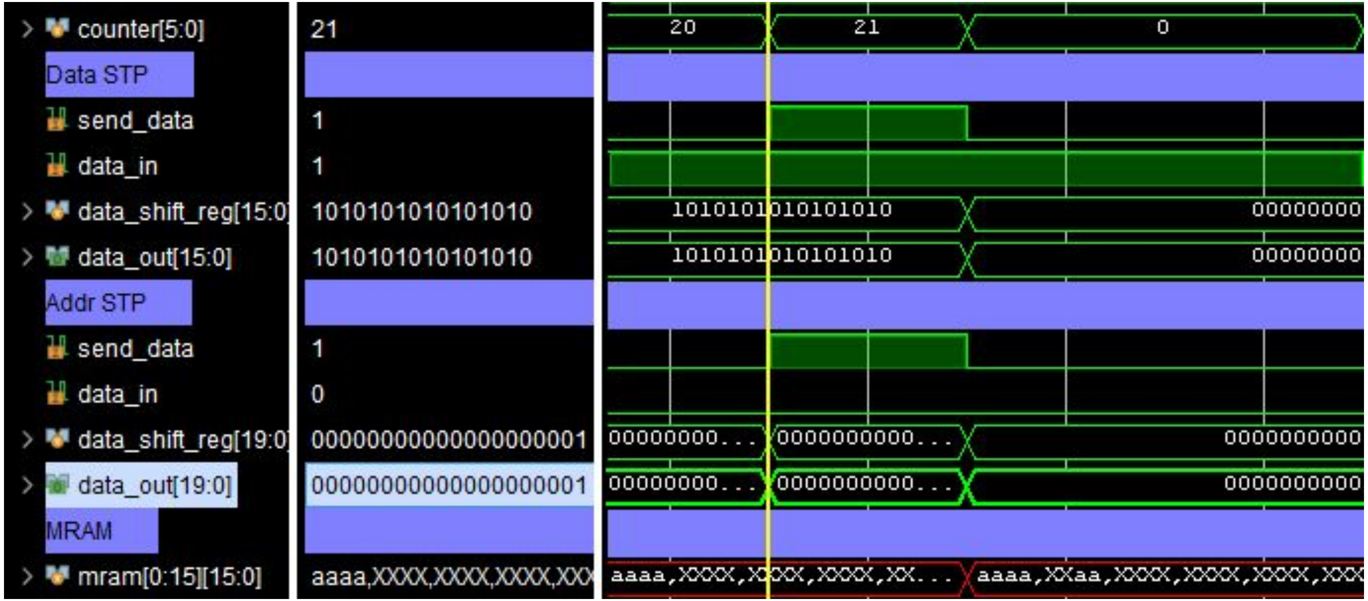


^addr and data is being shifted into the STP modules from counter=1 onwards

MRAM STP/PTS module standalone

MRAM is being written to on cycle 0 of the next loop

Therefore, writing takes 22 cycles (1 faster than the integrated module



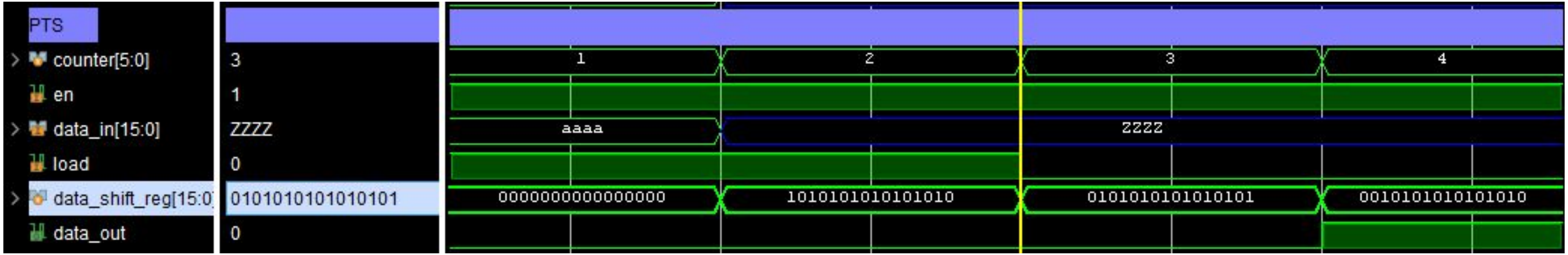
counter[5:0]	21
Data STP	
send_data	1
data_in	1
data_shift_reg[15:0]	1010101010101010
data_out[15:0]	1010101010101010
Addr STP	
send_data	1
data_in	0
data_shift_reg[19:0]	00000000000000000001
data_out[19:0]	00000000000000000001
MRAM	
mram[0:15][15:0]	aaaa,XXXX,XXXX,XXXX,XXX

^addr is fully inside the shift register at counter = 21.

MRAM STP/PTS module standalone

Reading of data:

- Address needs to be provided first. Therefore, initial 21 cycles needed
- In the next loop, data gets serialy output from counter 3 onwards



PTS

counter[5:0] 3

en 1

data_in[15:0] ZZZZ

load 0

data_shift_reg[15:0] 0101010101010101

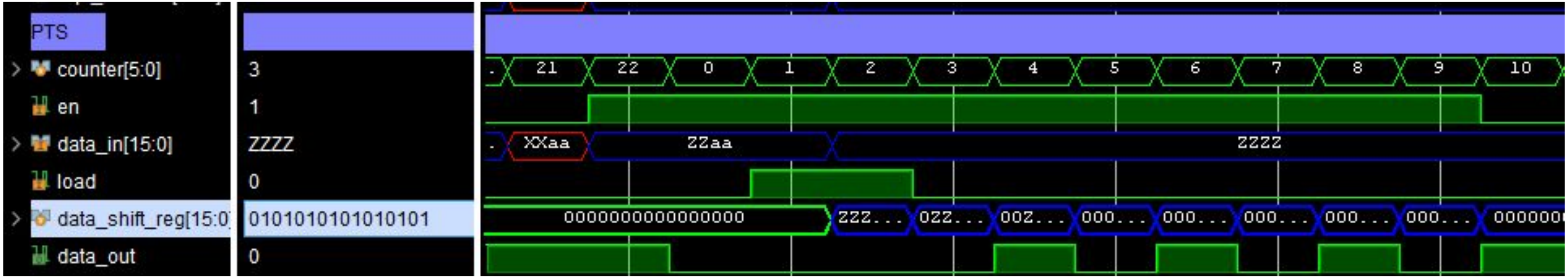
data_out 0

^Data gets loaded in at counter = 2, shifted out at counter = 3

MRAM STP/PTS module standalone

Reading of data:

- Subsequent reads in succession will case the loop to take 23 cycles due to shifting of data


PTS
counter[5:0] 3
en 1
data_in[15:0] ZZZZ
load 0
data_shift_reg[15:0] 0101010101010101
data_out 0
21 22 0 1 2 3 4 5 6 7 8 9 10
XXaa ZZaa ZZZZ
0000000000000000 ZZZ... 0ZZ... 00Z... 000... 000... 000... 000... 000... 0000000

^half word read, lower byte

MRAM STP/PTS module standalone Summary

Writing:

- 22 cycles

Reading:

- Initial 22 cycles for writing address
- Subsequent 23 cycle per loop (due to shifting of data into registers I believe)
- Data is output on the next loop, cycle 3 to 18

Questions n stuff

- Can I clarify, how does my memory controller fit into the PULP microprocessor? Does it act something like a shared memory between the PULP and accelerator that Yan Rui(the other FYP student) is developing?
- How can I request for the Zboard FPGA? Do I have to email Jeremiah for it?
- I only managed to finish this integration as there were quite a bit of bugs and synchronization issues between the 2 modules. I will continue to work on the other design I have in mind over the next few weeks.