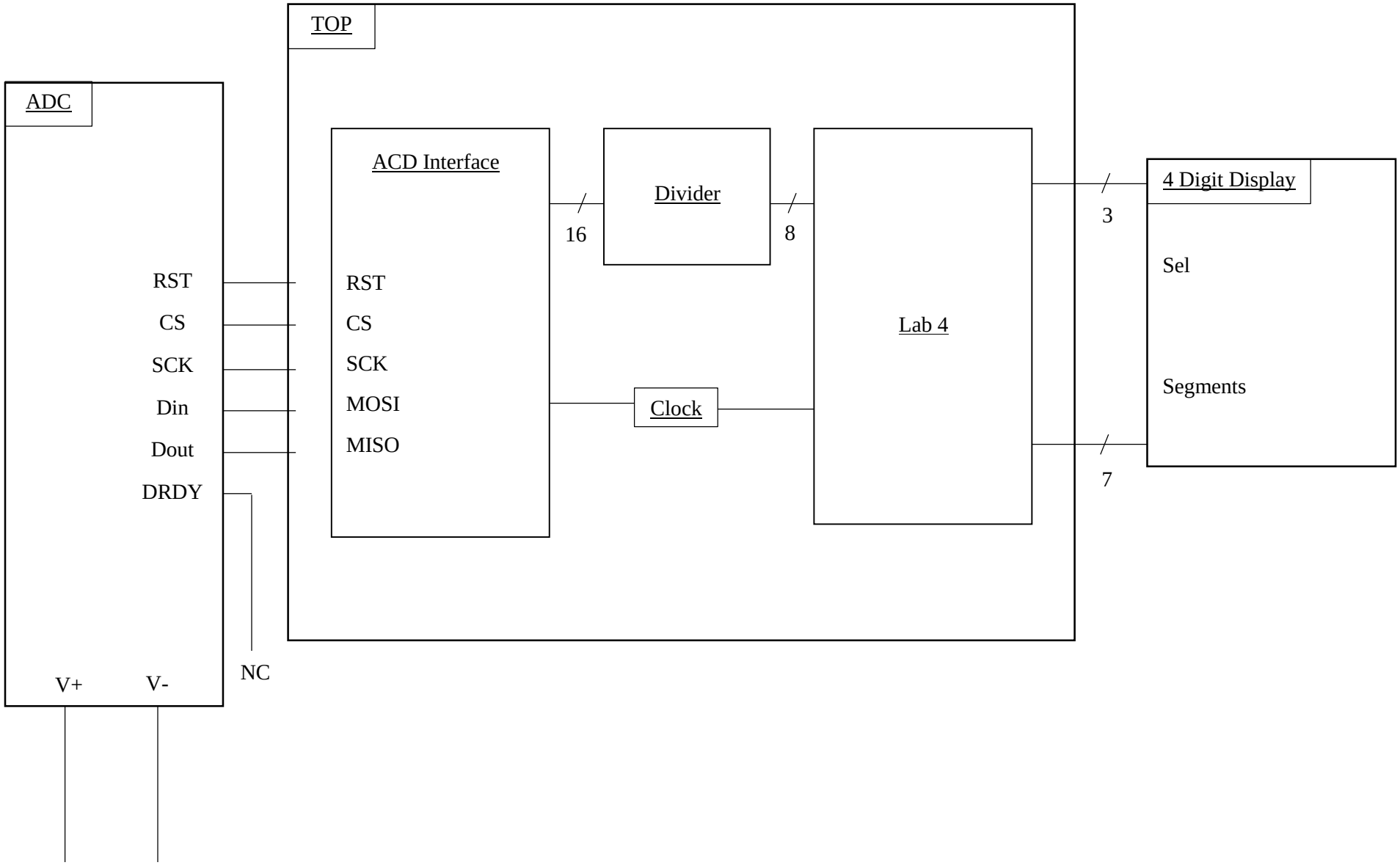
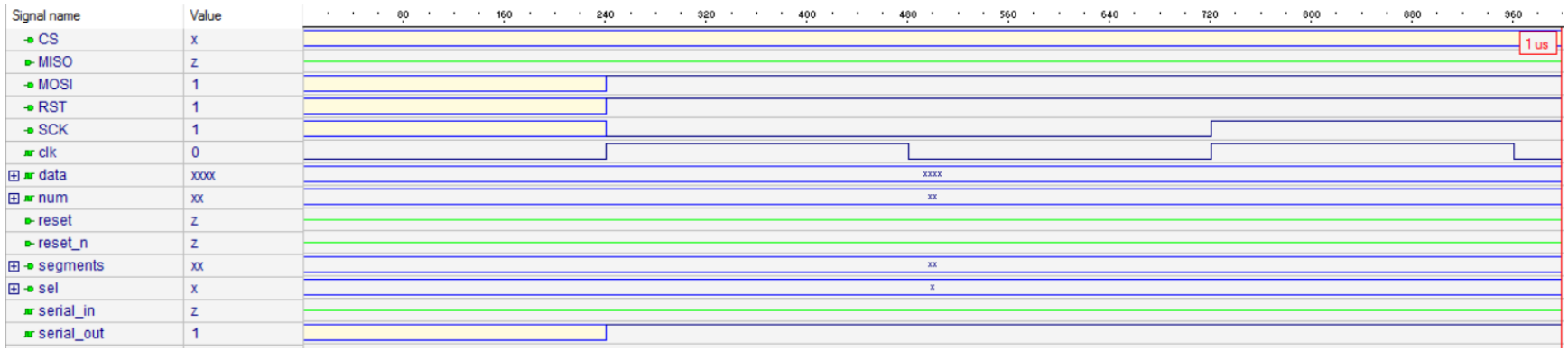
Rhea Mae Edwards




Signal name
Value
80
160
240
320
400
480
560
640
720
800
880
960
1 us
CS
x
MISO
z
MOSI
1
RST
1
SCK
1
clk
0
data
xxxx
num
xx
reset
z
reset_n
z
segments
xx
sel
x
serial_in
z
serial_out
1

```
  1  module AD7705(input reset, //Active High Reset
197  module divider(
198      input [15:0] ADC_data,
199      output reg [7:0] num
200      );
201
202      always @ (*)
203          begin
204              num = ((ADC_data)/2^16)*(333/10);
205          end
206
207  endmodule
208
209  module digit_parser(
226  module output_select(
247  module state_machine( //example of a Moore type state machine
305  module clock_counter(
333  module seven_segment(
354  module fpga(
355      input clk, //this was added
356      input [7:0] number,
357      input reset_n,
358      output [2:0] select,
359      output [6:0] segments );
360
361      wire clock;
362      //wire clk;
363      wire [3:0] ones;
364      wire [3:0] tens;
365      wire [3:0] hunds;
366      wire [3:0] thous;
367      wire [3:0] data;
368
```

```
    digit_parser dp(
        .number(number),
        .ones(ones),
        .tens(tens),
        .hunds(hunds),
        .thous(thous));

    output_select os(
        .select(select),
        .ones(ones),
        .tens(tens),
        .hunds(hunds),
        .thous(thous),
        .data(data));

    //This module is instantiated from another file, 'State_Machine.v'
    //It contains a Moore state machine that will take a clock and reset, and output LED combinations
    state_machine sm(
        .clk_i(clock),
        .reset_n(reset_n),
        .select(select));

    //This module is instantiated from another file, 'Clock_Counter.v'
    //It will take an input clock, slow it down based on parameters set inside of the module, and output the new clock. Reset functionality is also built-in
    clock_counter cc(
        .clk_i(clk),
        .reset_n(reset_n),
        .clk_o(clock));

    seven_segment ss(
        .data(data),
        .segments(segments));

    OSCH #("2.08") osc_int (                    //"2.03" specifies the operating frequency, 2.03 MHz. Other clock frequencies can be found in the MachX02's documentation
        .STDBY(1'b0),                           //Specifies active state
        .OSC(clk),                              //Outputs clock signal to 'clk' net
        .SEDSTDBY());

endmodule
```

```
410  module top(
411      input reset,
412      input reset_n,
413      input MISO, // Master Input Slave Output, connect to Dout on ADC module.  Set Pullmode to none
414      output RST, //ADC reset, connect to RST on ADC module
415      output CS, //Channel Select, connect to the CS pin on ADC module
416      output SCK, //Serial Clock, connect to SCK pin on ADC module
417      output MOSI, //Master Output Slave Input, connect to Din on ADC module.
418      output [2:0] sel,
419      output [6:0] segments
420      );
421
422      wire serial_out;
423      wire serial_in;
424      wire [15:0] data;
425      wire clk;
426      wire [7:0] num;
427
428      assign serial_in = MISO;
429      assign MOSI = serial_out;
430
431      AD7705 ADC(
432          .reset(reset),
433          .MISO(serial_in),
434          .MOSI(serial_out),
435          .SCK(SCK),
436          .CS(CS),
437          .RST(RST),
438          .ADC_data(data),
439          .clk(clk)
440          );
441
442      divider div (
443          .ADC_data(data),
444          .num(num)
445          );
446
447      fpga fpga (
448          .clk(clk),
449          .number(num),
450          .reset_n(reset_n),
451          .select(sel),
452          .segments(segments)
453          );
454
455  endmodule
```