UCD School of Electrical and Electronic Engineering

EEEN40280 Digital & Embedded System

System on Chip Design Report

Name: Jingyi Hu **Student Number: 17200558**

Working with: Anton Shmatov, James Carron

I certify that ALL of the following are true:

1. I have read the *UCD Plagiarism Policy* and the *College of Engineering and Architecture Plagiarism Protocol*. (These documents are available on Blackboard, under Assessment.)
2. I understand fully the definition of plagiarism and the consequences of plagiarism as discussed in the documents above.
3. I recognise that any work that has been plagiarised (in whole or in part) may be subject to penalties, as outlined in the documents above.
4. I have not previously submitted this work, or any version of it, for assessment in any other module in this University, or any other institution.
5. I have not plagiarised any part of this report. The work described was done by the team, but this report is all my own original work, except where otherwise acknowledged in the report.

Signed: Jingyi Hu **Date:** April 2019

Introduction

In this assignment, we divided our task into – software design, hardware SPI interface, 8-digit display interface and VGA signal generator design. All the hardware blocks are connected to the AHB-Lite bus. As a result, we built our embedded system on a FPGA (N0.62 board) where an accelerometer (ADXL362) can be used to measure acceleration in x-axis, y-axis, and Z-axis. These data will be demonstrated as an icon on a VGA display. Besides, we also used the 8-digit 7-segment display to display the data measured by an integrated temperature sensor of ADXL362.

System Functionality

We designed a system that could display the temperature on two 4-digit 7-segment display. At the same time, the data of each x,y,z axis would be demonstrated as an icon on the VGA display. The horizontal movement of the icon depends on the value out from the x-axis registers of the accelerometer (board tilts along the x-axis), while the vertical movement of the icon is related to the value stored in the y-axis registers ((board tilts along the y-axis). The size of the icon is used to represent the vertical movement of the board which moves up to create a bigger icon, and moves down to create a smaller icon.

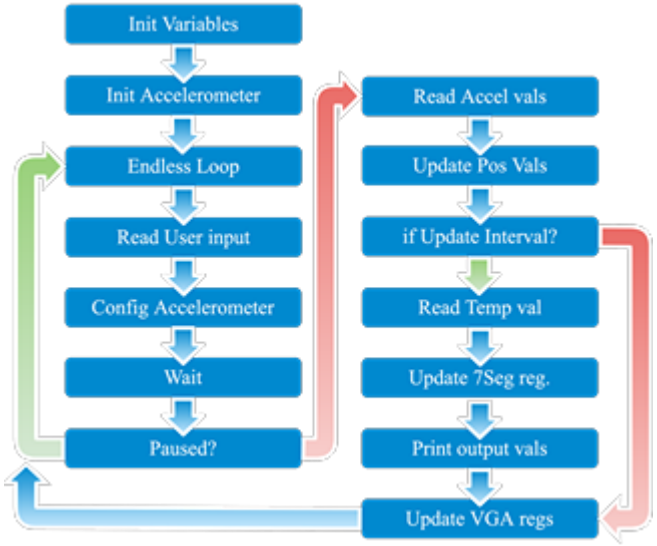
Moreover, the 16 switches were also used for multiple options. From the left to the right, the first 9 switches can be used to change the RGB of the background colour on the VGA display; the next 3 for the RGB of the cube; and the sensitivity of measurement (±2g, ±4g, ±8g,) for next 3. Lastly, the rightmost switch is for pausing.

For the hardware part, we identified the main blocks as ‘vga_sync’, ‘vga_pixels’, ‘vga_pixpos’, ‘spi’ and ‘LEDSEG’. Only the display interface block got a subblock ‘hex2seg’.

For the software part, the ‘main.c’ controls the hardware to get the value out from the registers within the accelerometer. And process the data in order to display it on the either LED display or VGA display.

Software Part

The flow chart of the software program shows below:



Created by:
James Carron

After getting the initialisations of the accelerometer and the variable, an endless loop starts working. First, update the values from the switches and write the sensitivity information into the accelerometer. Next, keep sending the instrument command and the address where it holds the data that we want to get the data from the accelerometer.

For the VGA display, once the software gets the data, an IIR filter will be applied to the data. Next, update the positions of the cube, cast them to integer discrete pixel values and write to VGA register.

For the LED display, the data out from the temperature register will be converted into a hexadecimal value and this value will be pushed to the 7-segment display.

Hardware Block

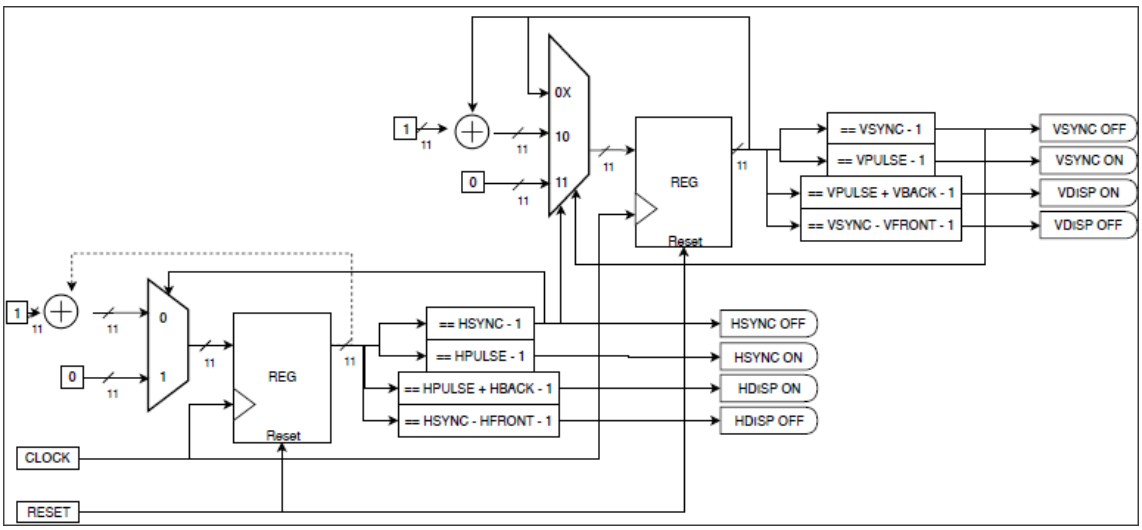
AHB-Lite VGA Display Block (The graphs for VGA are create by: Anton Shmatov)

Once the software got the coordinate position signals, it controls registers within the AHB-Lite bus to send HWDATA, whose 11 LSB bits is either x, y or z-axis data and 12 LSB is either background or cube colour, to relative signals in VGA blocks.

Block ‘vga_snyc’ creates two standard sync signals, horizontal and vertical and the pixel control signals to make sure the refresh time in horizontal and vertical place based on the pixel clock which is 108 MHz.

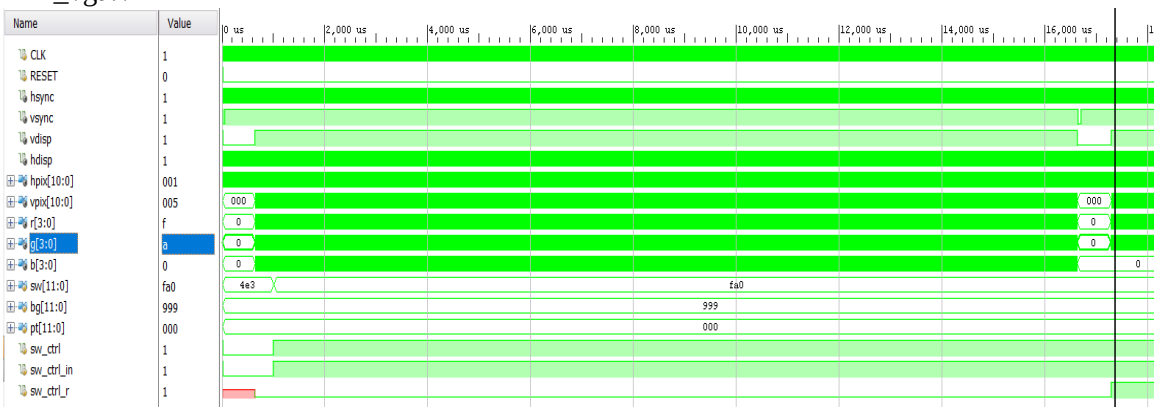
The 12 bits value of the colour will be processed in ‘vga_pixels’ to specify the vgaRed signal, vgaGreen signal and vgaBlue signal.

RTL Diagram of VGA Display

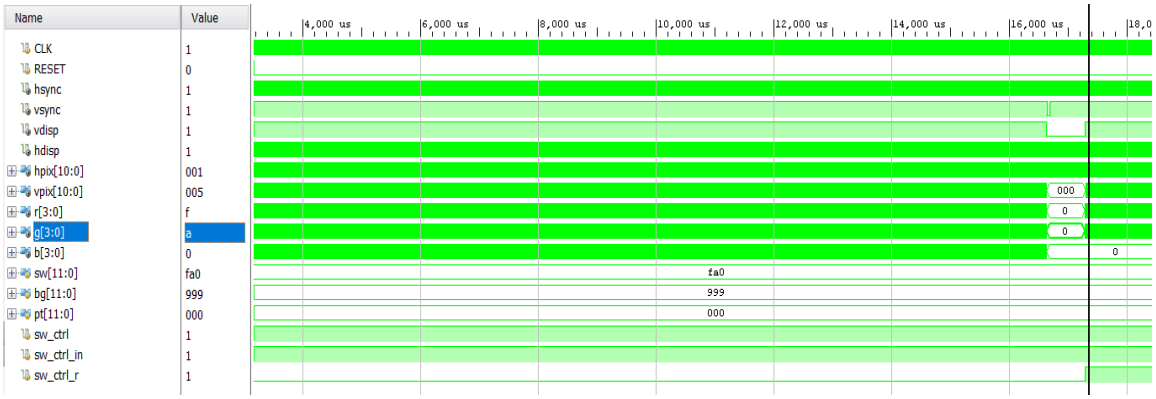


VGA Display Verification

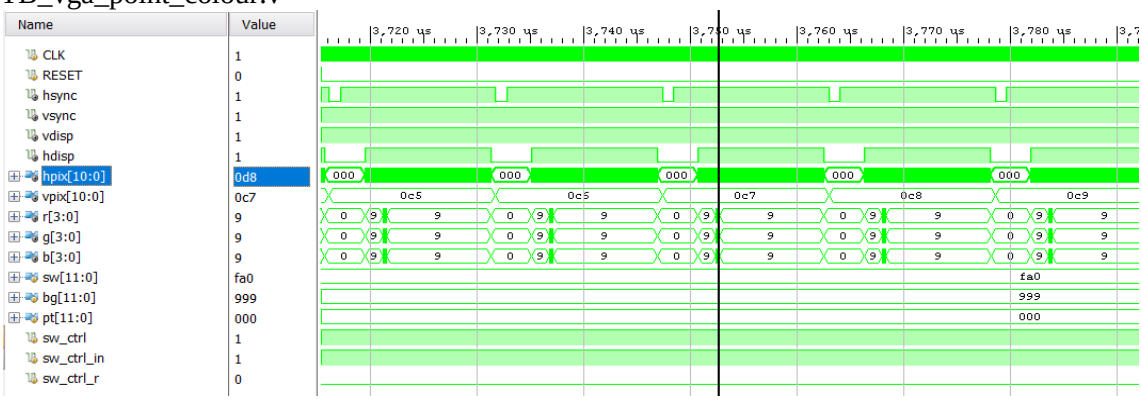
TB_vga.v



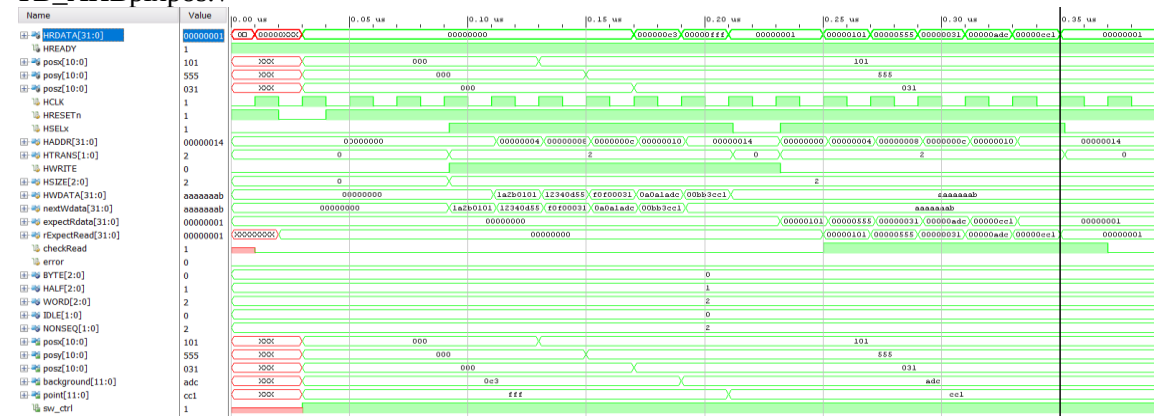
TB_swctrl.v



TB_vga_point_colour.v



TB_AHBpixpos.v



AHB-Lite SPI Interface Block (The graphs for SPI are create by: Jingyi Hu)

The accelerometer communicates via 4-wire SPI and operates as a slave, therefore, we need to design a SPI master interface which operates in SPI mode 0 with CPOL=0 and CPHA = 0.

The SPI interface is an input- output block, designed to be a slave on AHB-Lite bus. The block provides 3 1-bit output ports and 1 1-bit input port to the accelorameter. This hardware is described in Verilog file AHBSpi.v and a testbench is provided as TB_AHBspi.v.

The SPI Interface logic module is composed of 11 registers and combination logics. All of the registers operate on the bus clock with synchronous active-low reset (the RTL diagram of SPI interface is shown in next section):

- Registers to hold signals from address phase of each bus transaction:
  - *a) W_control Register:* The write enable signal is generated during the address phase and held in flip-flop for use during the data phase.
  - *b) R_control Register:* A read transaction is identified during the address phase and held for use in the data phase.
  - *c) Addr Register:* This register holds bits 3:2 of HADDR from the address phase of each bus transaction for use the data phase.
- Registers for output ports:
  - *d) Write Data Register:* A register holds the least significant 8 bits of HWDATA from the data phase of each bus transaction if the value of relevant address signal is 0 and the write enable signal is 1, and held in the register for use in SPI master- slave communication.
  - *e) Slave Selection Register :* This register holds bit 0 of HWDATA from the data phase of each bus truncation if rHADDR and rWrite are 1s. And the output chip_select is connected to the output port CS of SPI interface for SPI slave selection.
  - *f) SCLK generation Register:* This register used to generate a new clock for SPI master-slave communication because the SPI slave Accelerometer required that the SPI clock speeds are 1 MHz to 8MHz. And a new clock will be generated at each SPI master-slave transaction. The SCLK clock we choose is 3.125 MHz, therefore, each time a rising edge is detected 8 times, the SCLK_i will invert when the next rising edge arrives.
  - *g) 7-bit counter Register:* The counter will start counting from 0 to 127. After reset, the output of the counter will be assigned 7-bit 0s where start_count is 0 as well. Once start_count goes high, the counter will start up-counting to 127 if start_count is 0 or the output of counter is not all 0s.
  - *h) CountIni Register:* This register is used to generate the signal start_count. After reset, clock_count is 7-bit 0s and start_count is 0 as well. When each SPI master-slave transaction occurs, a new SCLK is needed as mentioned above. Only the write enable signal rWrite will be set to high and rHADDR signal is 0, the start_count will be assigned to 1for enable the 7-bit counter. Besides that, the start_count signal will be set back to 0 when the next rising edge of HCLK arrives once the counter starts counting.
  - *i) Flag Register:* This register is used to indicate a complete round of successful data transaction. After reset, the output spi_ready will goes to high for tell that the SPI master and slave are ready for data transaction. Once there is data between SPI master and its slave, this signal spi_ready will be set to 0 as

indication that our SPI master and slave are currently busy and any new write data from the bus will be kept wait. Only the spi_ready goes to high again, a new data exchange can be performed.

j) MOSI Register: This register allows data from master to slave at correct timing. nMOSI will be assigned with new 8 bits data every time, new least significant 8-bit HWDATA from bus need to be transmitted to the accelerometer when there is no data transaction between SPI master and slave (start_count = 1,clock_count =7'b0); or nMOSI will be assigned the old 8-bit HWDATA but with 1 bit left shift at falling edge of SCLK since out SPI interface is required to operate in mode 0; otherwise, nMOSI will hold its current value.

k) MISO Register: A register used to receive the data sent back from the accelerometer at the rising edge of SCLK. The output nreadData only holds the latest 8 bits data.

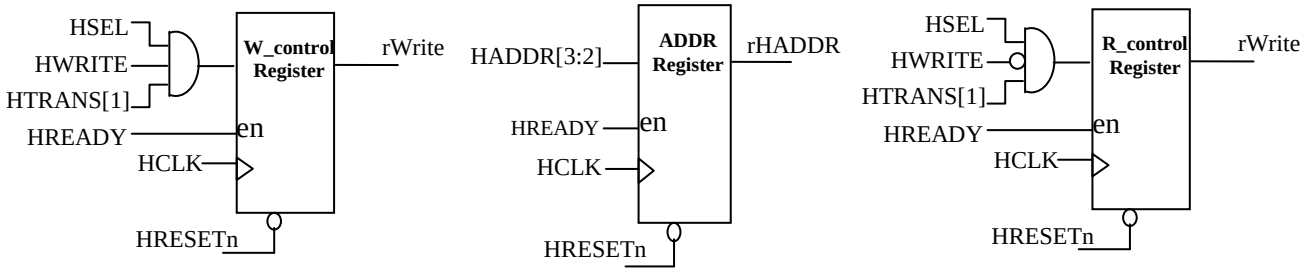
- Read transfer from SPI master to AHB-Lite bus:

l) Read Multiplexer: This multiplexer selects the required 32-bit value for read transfers, based on the held address bits rHADDR.

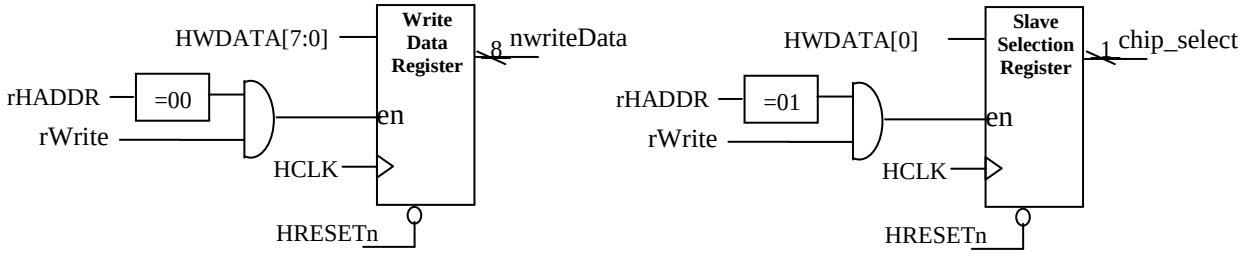
HREADYOUT is always 1 as there are no wait states.

Register	Relativ Address	Access
Write Data Register	0x0	R/W
Slave Selection Register	0x4	R/W
MISO Register	0x8	R
Flag Register	0xC	R

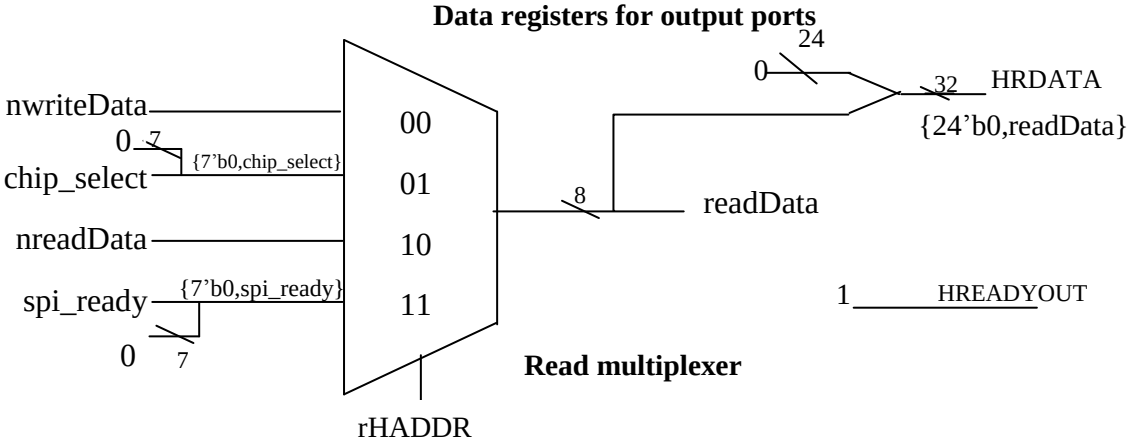
RTL Diagram of SPI interface



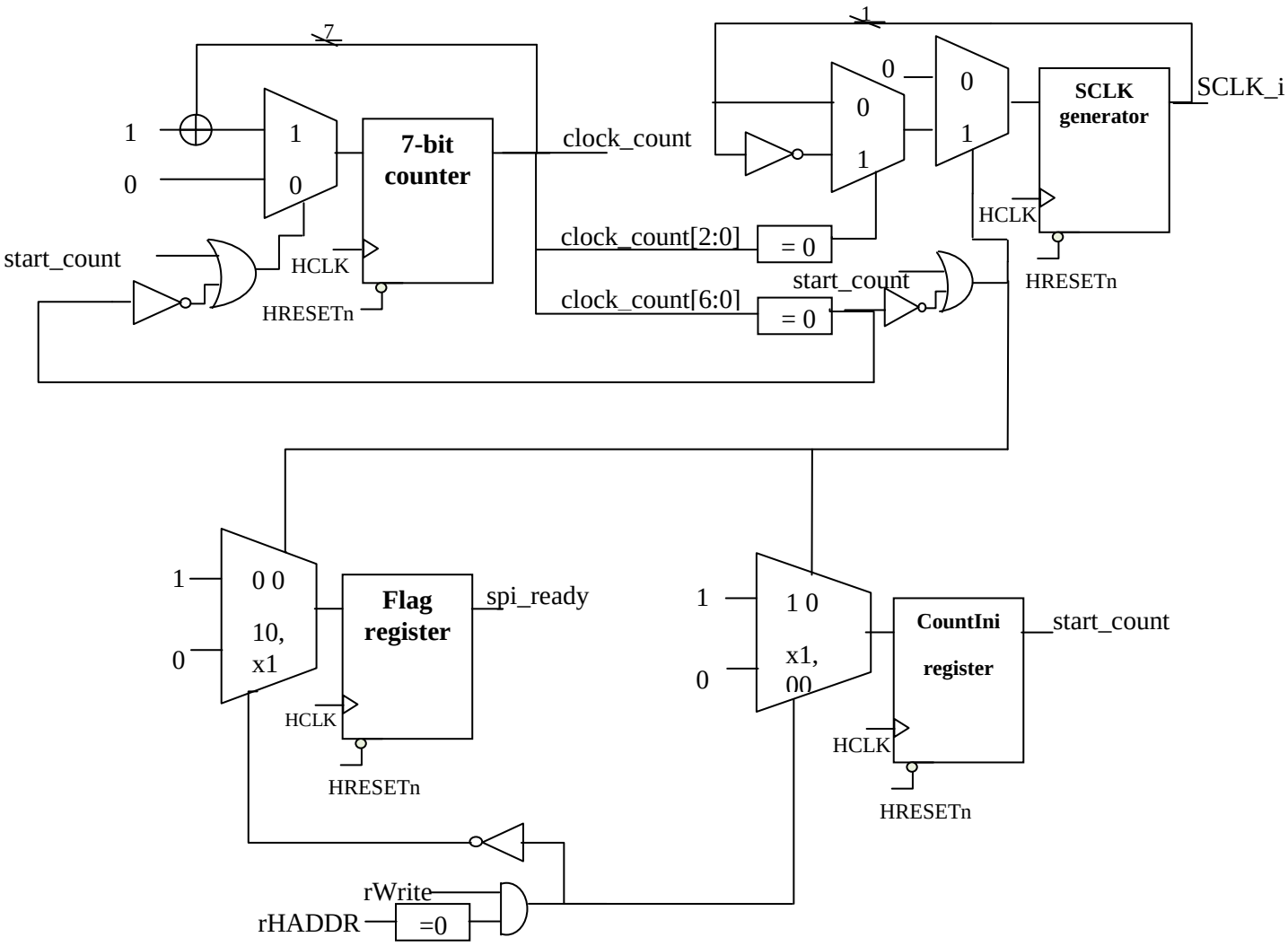
Registers to hold signals from address phase



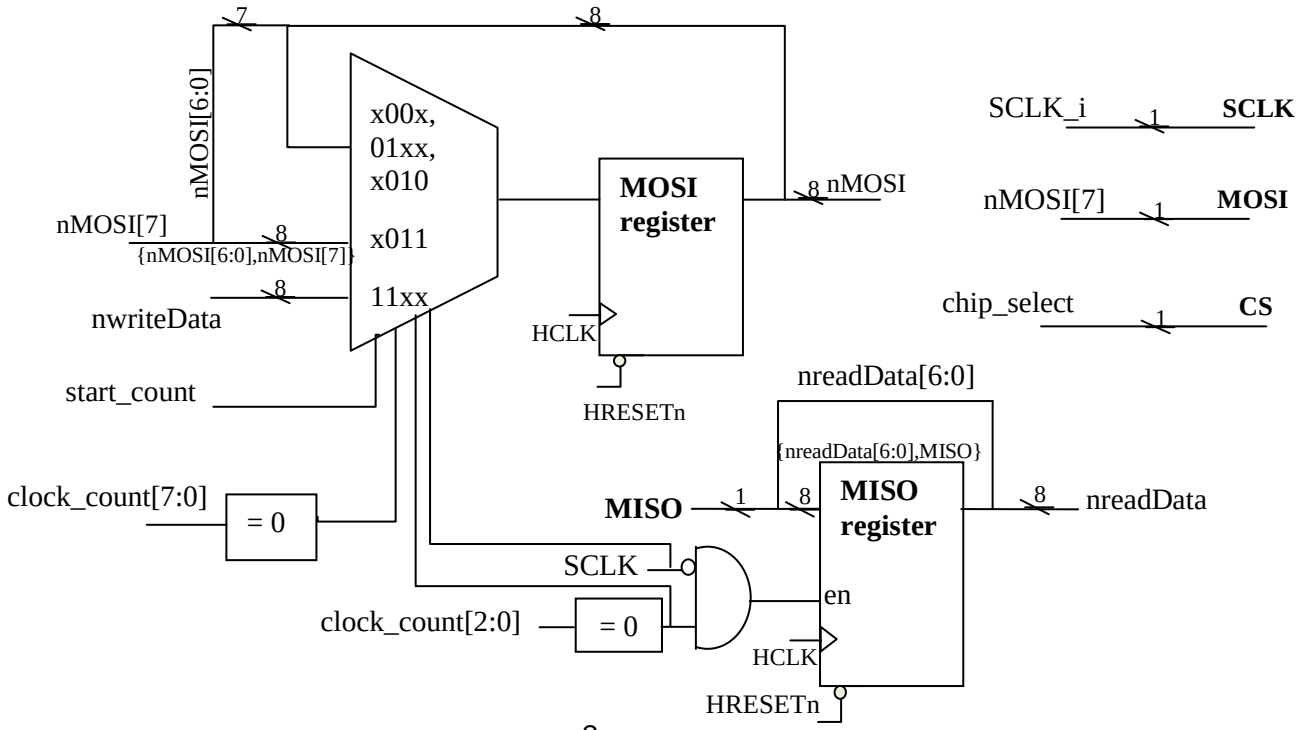
Data registers for output ports



Read multiplexer



SPI master-slave communication control part



SPI master-slave data transceive part

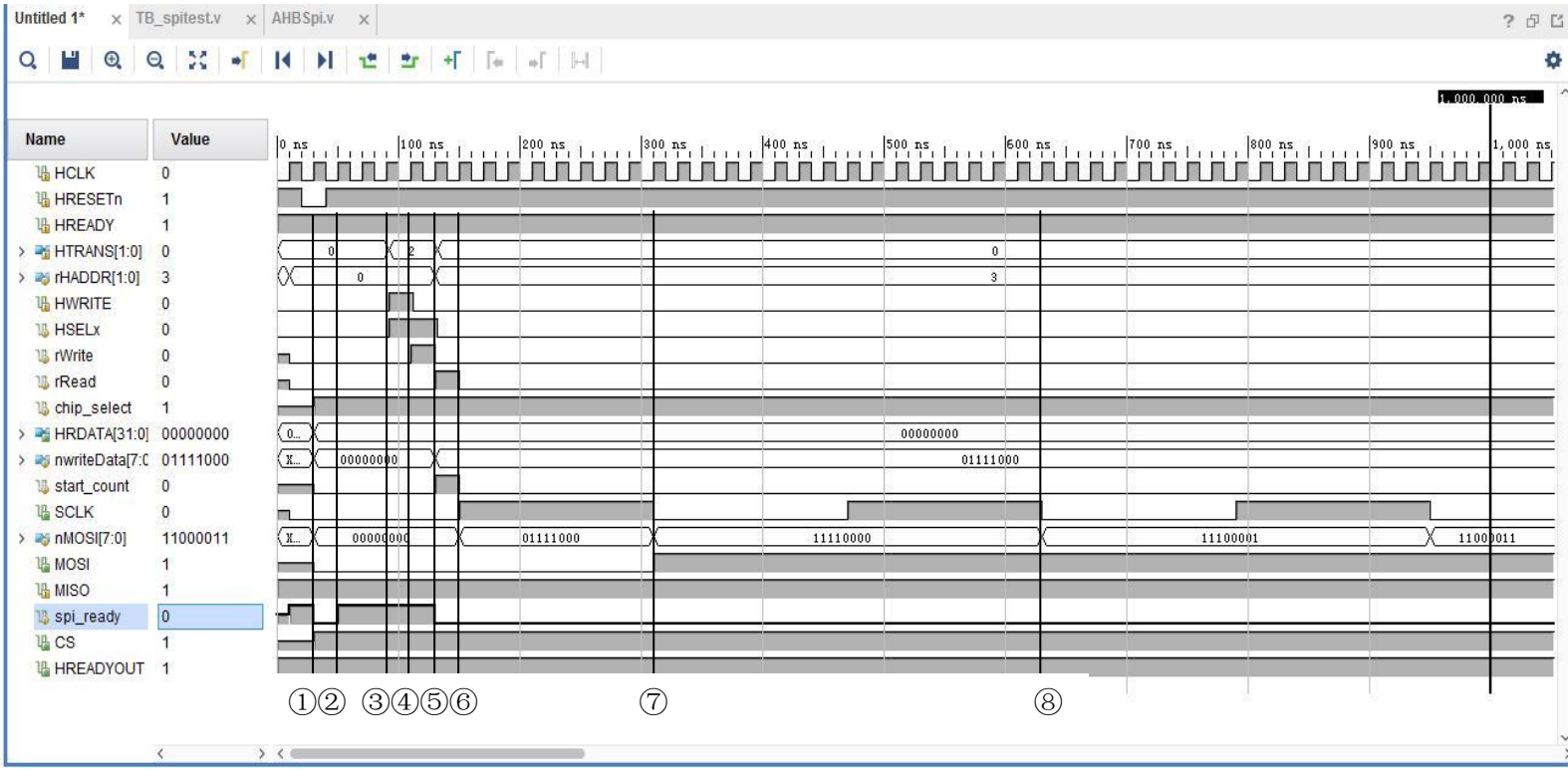
SPI Interface Verification

Start with your verification plan, followed by the testbench you created to implement that plan.

Include some details of testing on the hardware – this will probably be a subset of your verification plan.

In order to ensure that the SPI interface's behaviour is correct, the following situations have been verified:

1) From new write data 32'h 12345678, the only value sent to accelerometer should be 78;
2) Check whether the read multiplexer is working correctly use AHBread() function with data = 32'h00000000 and relative address = 0xC;
3) Assign HWDATA a new data 32'h12340555 before SPI transaction finished;
4) Assign HWDATA a new data 32'h10011001 after SPI transaction finished;
5) Set CS signal to 0

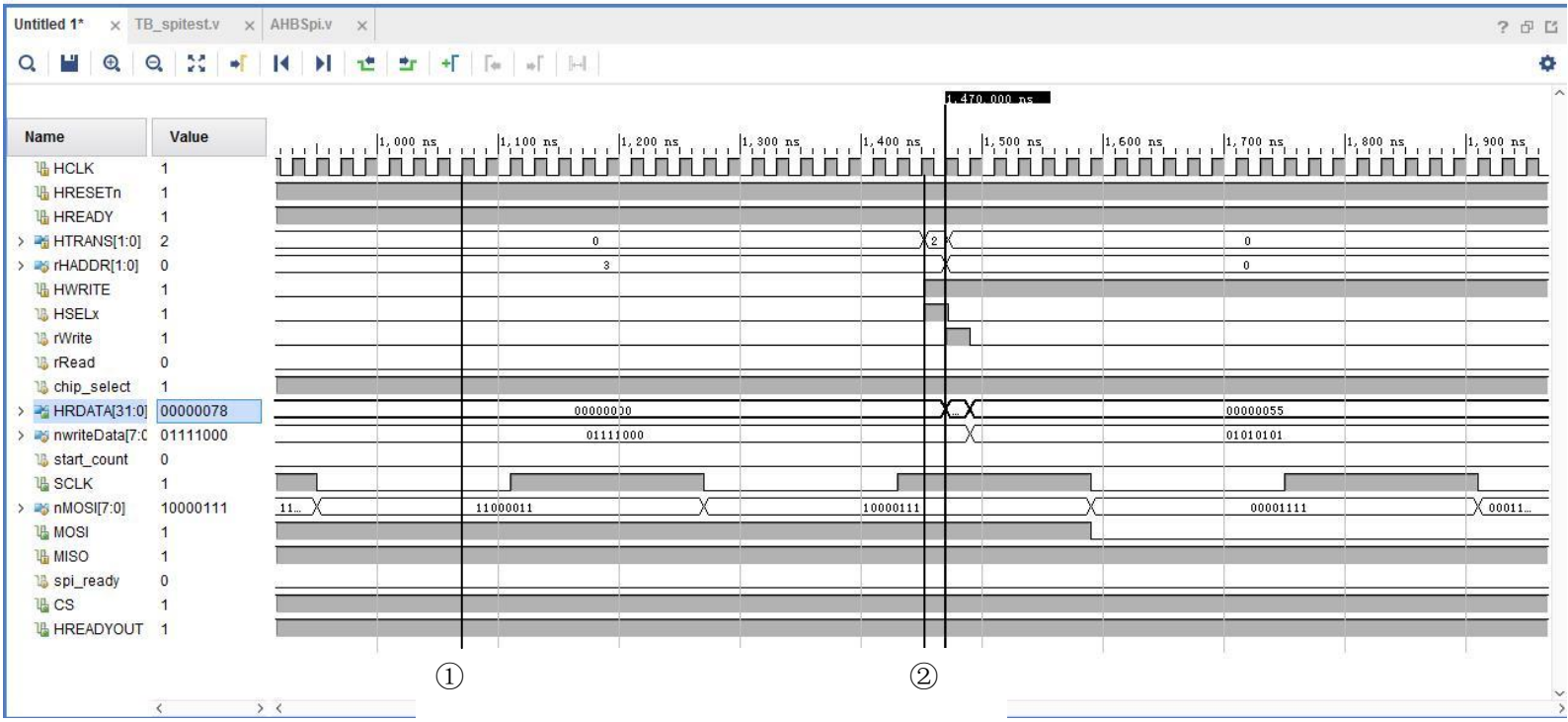


The timing diagrams are shown below:

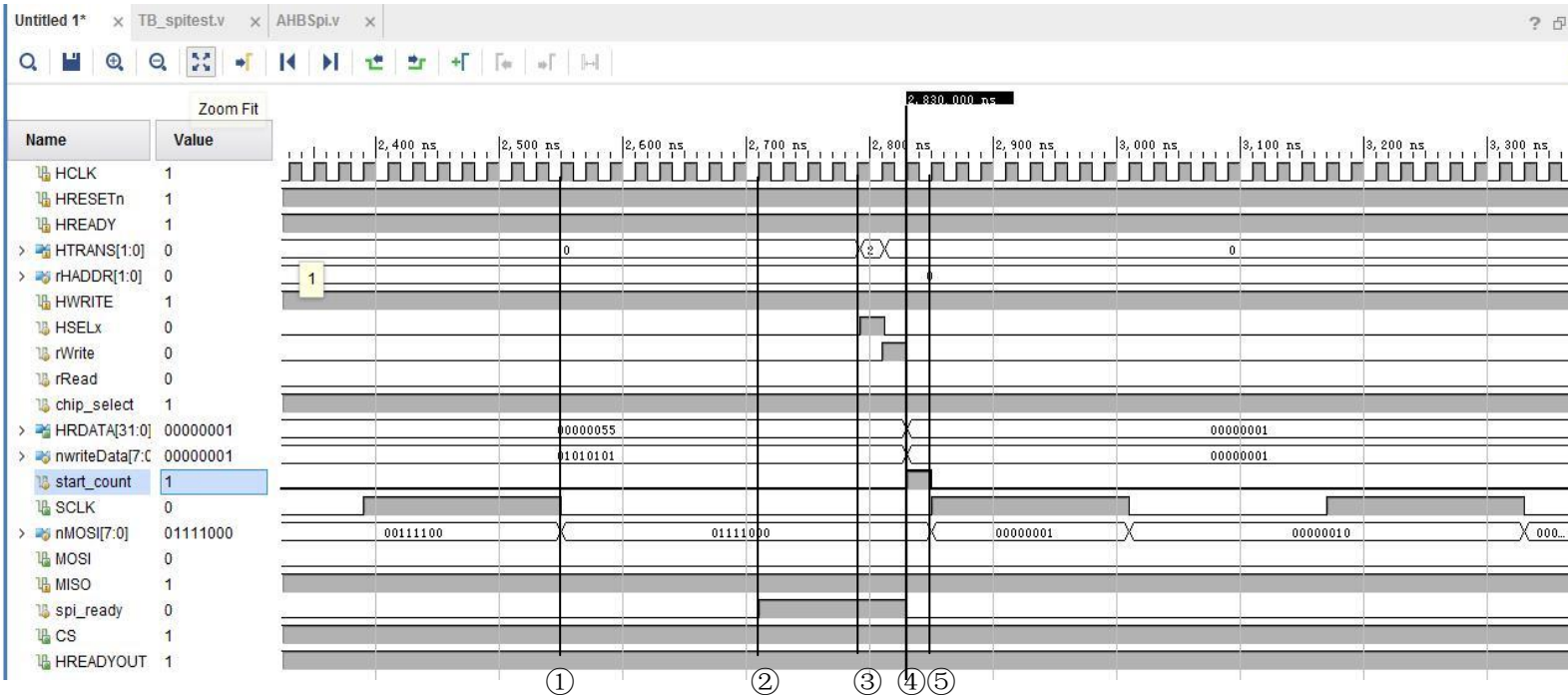
- ① After active-low reset, the 3 outputs from the registers which hold signals from address phase (rWrite, rHADDR, rRead) changes to 0 at the following positive edge of bus clock. Same for clock_count, SCLK, nMOSI, nreadData, start_count, spi_ready which are all 0s, except chip_select which is 1 because CS is active low.
- ② At the next rising edge of HCLK, HRESETn goes to high. SPI is not busy so that spi_ready goes to high for indication that SPI master is available for new data transaction.
- ③ ④ According to our testbench, we started our first AHBwrite at 90 ns. Firstly, HSELx goes to high for selecting SPI interface as the current working slave on AHB-Lite bus. Besides, for write transfer, HTRANS[1] needs to be 1 as well as

HWRITE (③). They will generate a write enable signal rWrite and will change from 1 to 0 at the next following rising edge of HCLK (shown as ④)

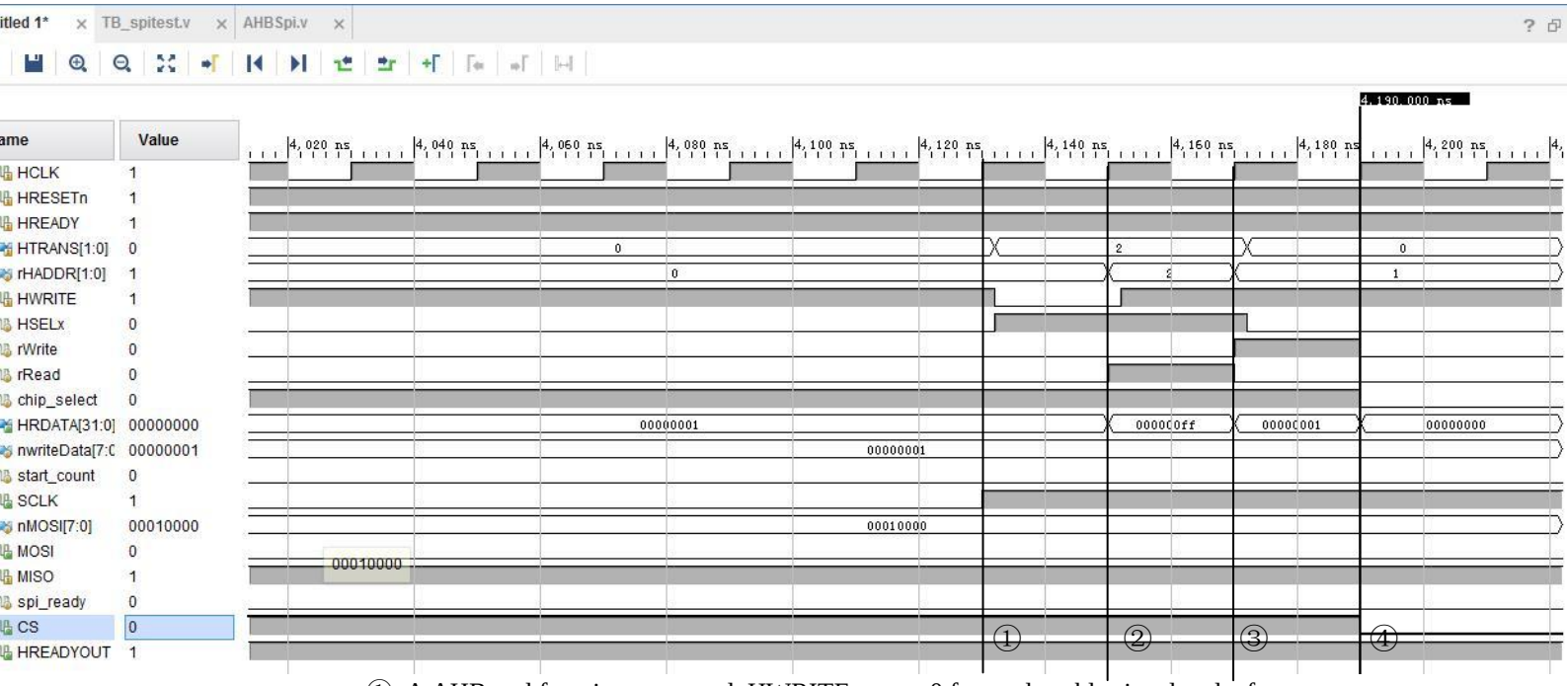
- ⑤ start_count goes to high at the next rising edge of HCLK after rWrite goes to high to enable the counter to start counting. At the same time, nwriteData will hold the relevant data from the data phase of bus. Therefore, spi_ready goes to low to indicate that SPI interface currently is busy.
- ⑥ Counter starts counting after start_count goes to high and counter is idle. SPI clock also starts to be generated for SPI master-slave data transaction. Meanwhile, nMOSI holds the data and transmit them to accelerometer at the precise time. The output port of SPI interface MOSI holds the first bit needed to be transmitting at the next falling edge of SCLK because SPI works in mode 0.
- ⑦ After 8 rising edges of HCLK have been detected, SCLK inverts when the first bit has been transmitted to the accelerometer and MOSI was assigned the value of the next pending transmission bit by round left shifting of nMOSI.
- ⑧ Same as Step 7



- ① Check the address 0x8 when nreadData will be assigned to HRDATA. rHADDR[1:0]=HADDR[3:2] = 3.
- ② Before the current SPI master-slave transaction is finished, we sent another group of data. As spi_ready is low, thus, new nwrite data will not be assigned to nMOSI.



- ① The transmission of the last one bit will be completed at the falling edge of SCLK clock.
- ② Counter finished counting, spi_ready goes to high to indicate that SPI interface is ready for next data trend.
- ③ SPI interface has been selected as the slave by bus. And the next rising edge od the bus clock, rWrite goes to high.
- ④ When start_count changes to 1, the counter starts counting at the following rising edge of bus clock in order to generate a new SCLK and nMOSI would be assigned with the new nwritedata.
- ⑤ The new write data is ready for bit transaction at the following falling edge of SCLK.



- ① A AHBread function presented, HWRITE sets to 0 for read enable signal and a few nanoseconds delay can be observed.
- ② rRead goes to high, rHADDR=2 allows data goes through the read multiplexer
- ③④ Check CS signal works properly. rHADDR is 1, last bit of HWDATA is 1 and CS actives low, thus, CS goes to low for selecting the SPI slave.

AHB-Lite 7-Segment Display Block (The graphs for LEDs are create by: Jingyi Hu)

8 digits 7-segment display are used to display the temperature of accelerometer. In order to illuminate a segment, both the anode and the cathode are driven low when active. In order for each of the 4 digits to be displayed on the board continuously, there is a need for the 8 digits to be entered every 1 to 16ms meaning that the 8-digit need to be refreshed at a frequency of about 1KHz to 60Hz.

The display interface is designed to be a slave on AHB-Lite bus. The block provides two 8-bit output ports. This hardware is described in Verilog file AHBLed2.v and a testbench is provided as TB_ledtest.v.

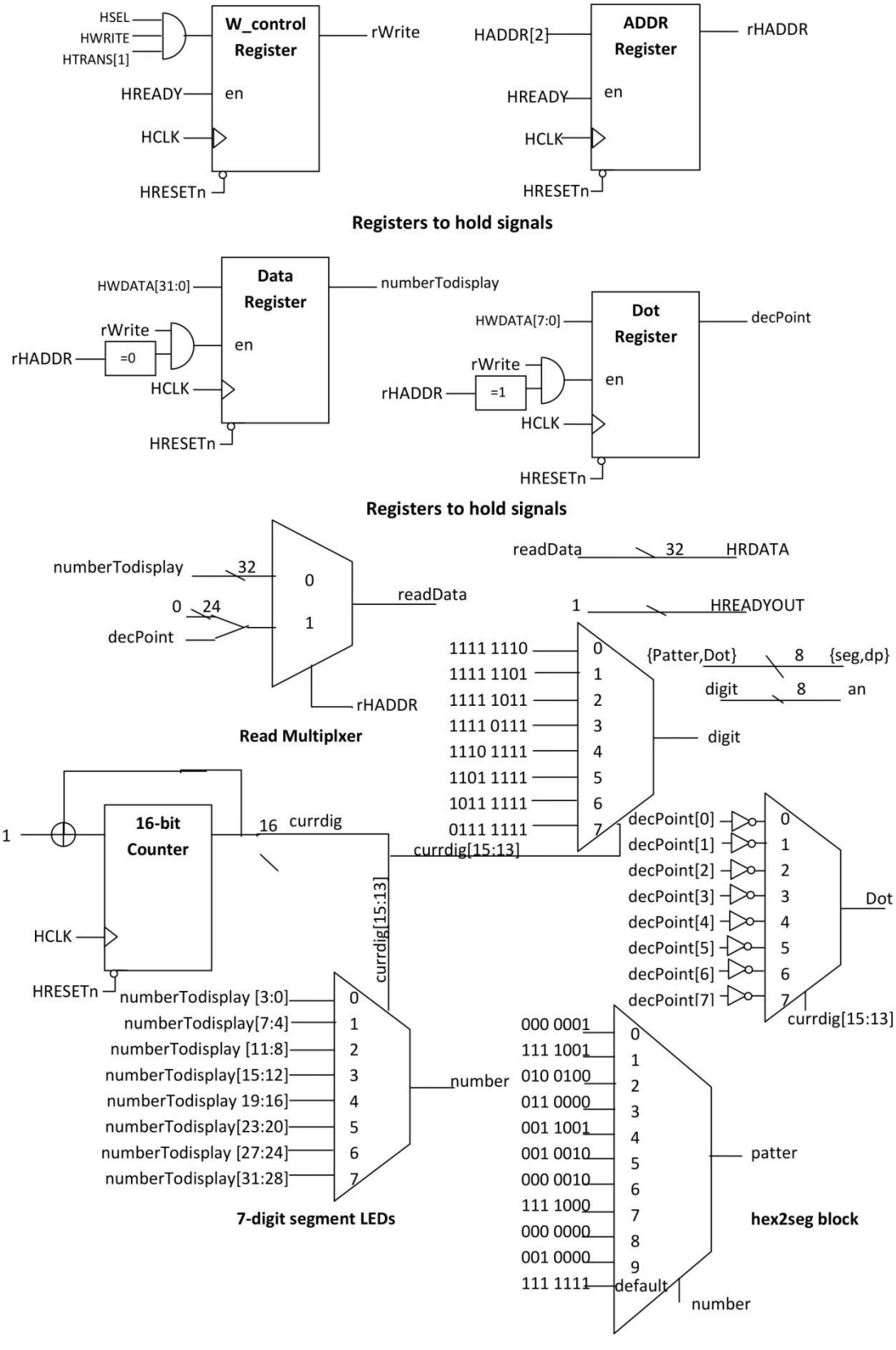
The display interface logic module is composed of 5 registers and combination logics. All of the registers operate on the bus clock with synchronous active-low reset (the RTL diagram of display interface is shown in next section):

- Registers to hold signals from address phase of each bus transaction:
  a) *W_control Register:* The write enable signal is generated during the address phase and held in flip-flop for use during the data phase.
  b) *Addr Register:* This register holds bits 3:2 of HADDR from the address phase of each bus transaction for use the data phase.
- Registers for output ports:
  c) *Data Register:* A register holds all 32 bits of HWDATA from the data phase of each bus transaction if the value of relevant address signal is 0 and the write enable signal is 1, and held in register. Each 4 bits, from the least significant to most significant , will send to the subblock *hex2seg* for corresponding segments information.
  d) Dot Register : This register holds bits 7 to 0 of HWDATA from the data phase of each bus truncation if rHADDR and rWrite are 1s. And connect to a multiplexer for select its corresponding bit according to time.
  e) *16-bit Counter:* This register used to slow down the bus clock down to around *763*Hz, therefore, refresh period is 1.3ms to meet the requirement. For each value of the 3 bits off currdig we change the value of digit to cycle through the digits on the display. At the same time, change the value of number to a different part of numberTodisplay value because of now displaying a different digit.
  Assign dot which is the hexadecimal point to one bit of the inverse of point.
- Read transfer from SPI master to AHB-Lite bus:
  f) Read Multiplexer: This multiplexer selects the required 32-bit value for read transfers, based on the held address bits rHADDR.

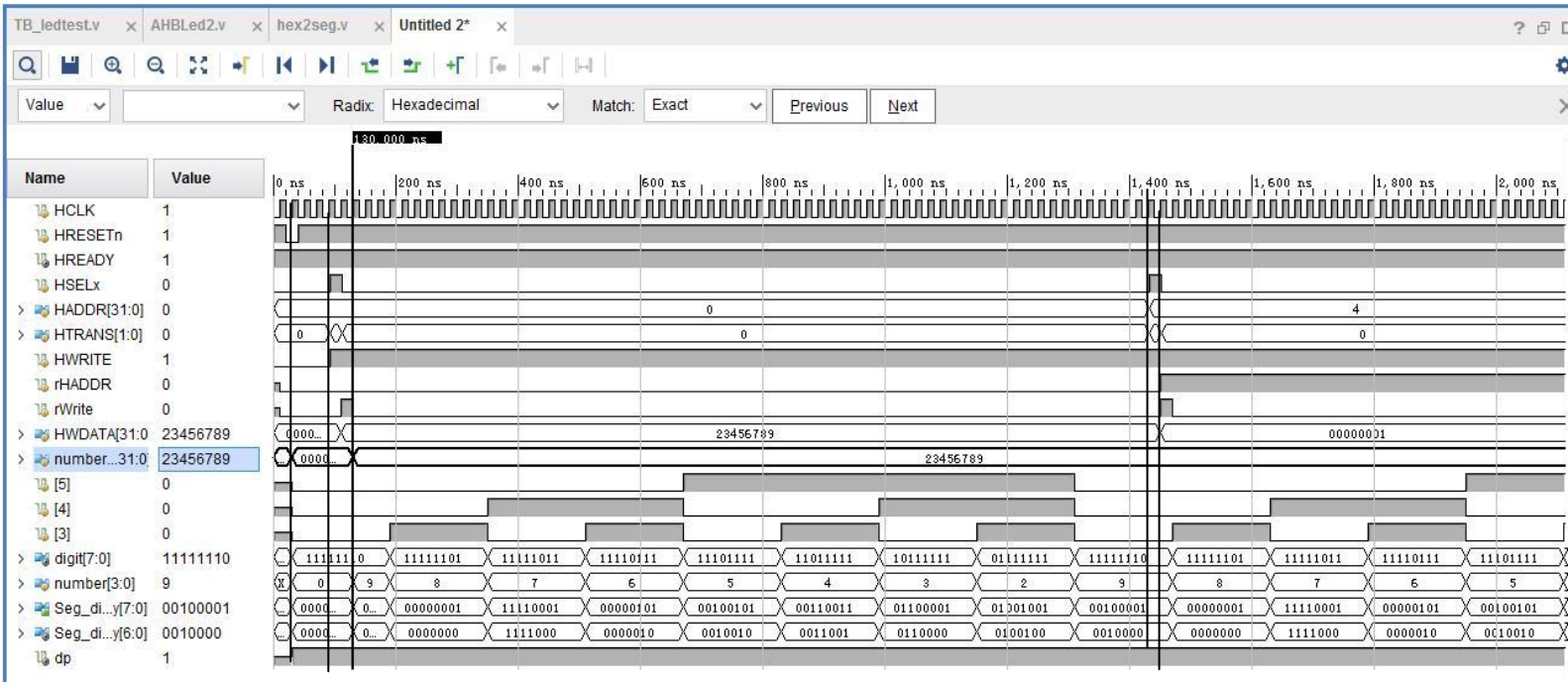
HREADYOUT is always 1 as there are no wait states.

Register	Relativ Address	Access
Data Register	0x0	R/W
Dot Register	0x4	R/W

RTL Diagram of Display Interface

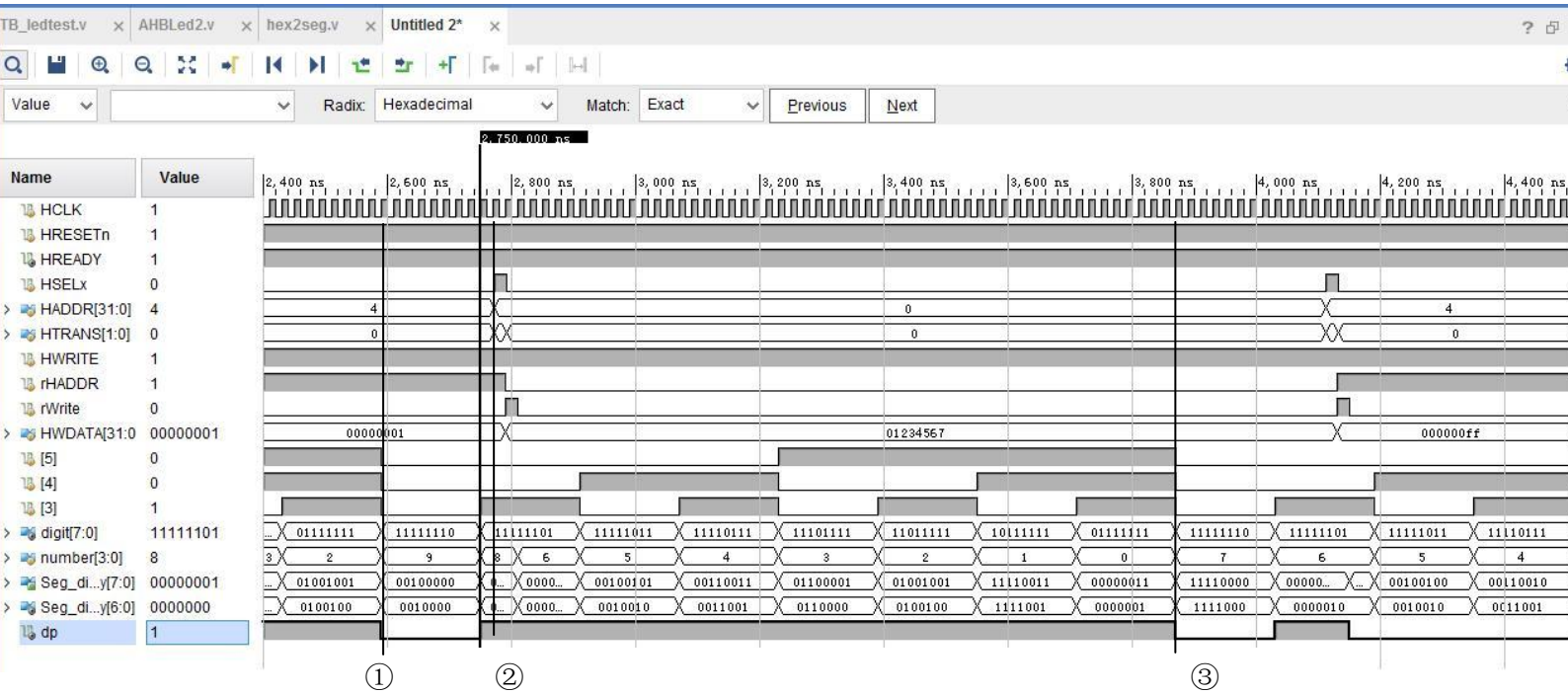


Display Interface Verification (A 6-bit counter used in testing for short graph)

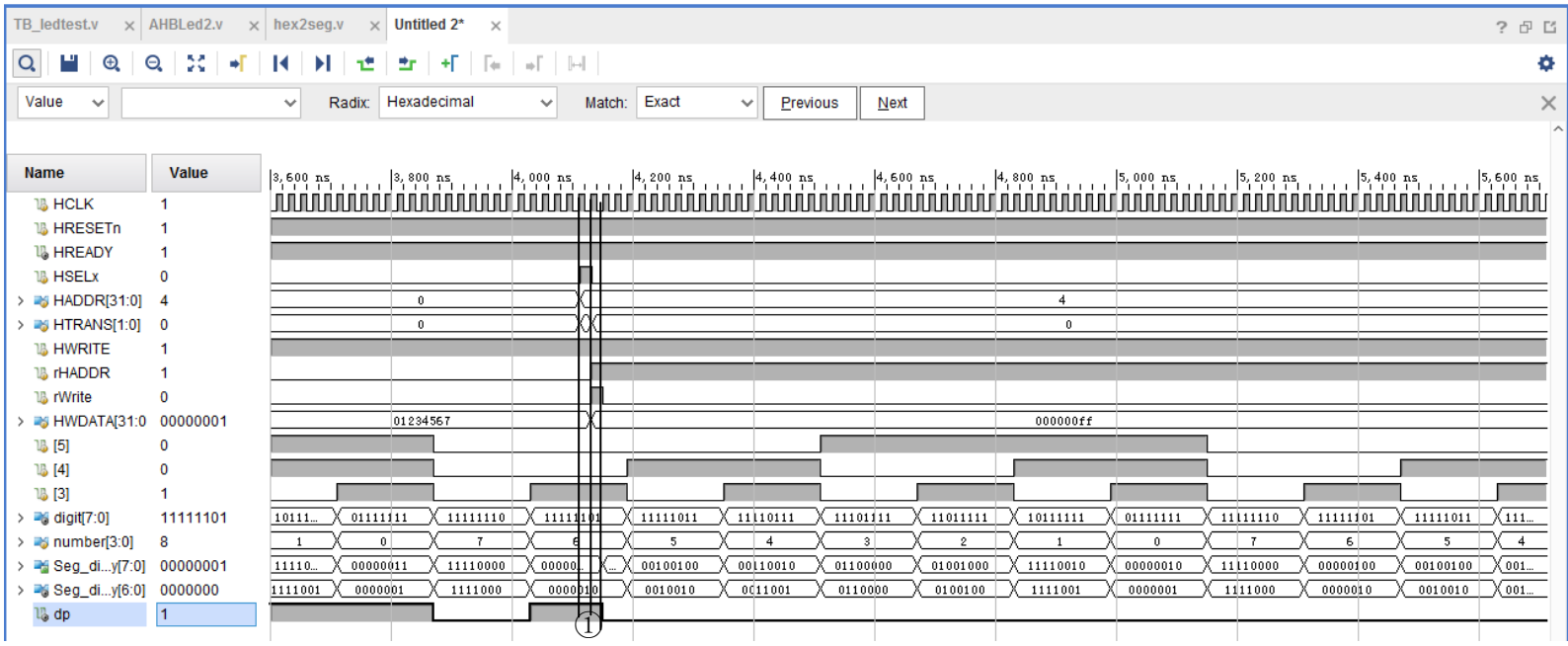


① ②③ ④

- ① After reset, all signals out from the registers are set to 0s. Dot is invert of signal decPoint. The counter starts counting at the 1st positive edge when reset is finished.
- ② AHB-Lite bus selects the display interface as the slave (HSELx =1) for write transfer (HWRITE =1). rWrite changed to 1 at the following positive edge of the bus clock.
- ③ numberTodisplay was assigned the 32 bits value of HWDATA 1 clock after the rWrite went to high. Because the relative bits of counter is 0, so the rightmost digit of the display value demonstrates on the LED display. Counter counts up, once the bits of counting value we care about changed, the correspond digit will display the relevant value.
- ④ The rightmost led displays have been refreshed. HADDR = 4 and rWrite=1 for write the decPoint signal which changes at the next following rising edge. However, as it can be seen in the testing, decPoint[0] changed right after its corresponding led digit refreshed. Thus, dp did not change until the next time of refresh if decPoint stays same on that time.

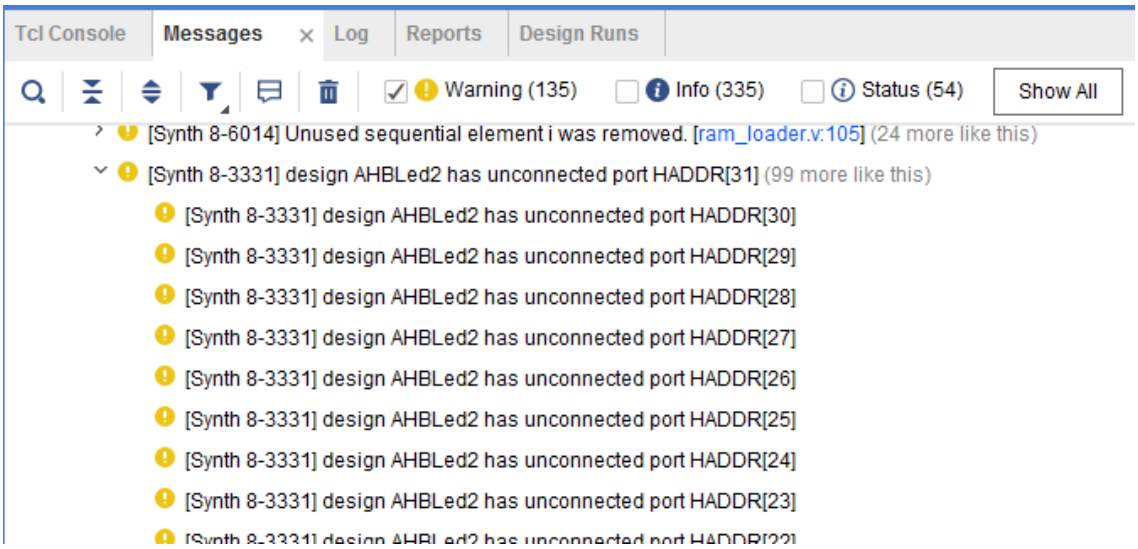


- ① When another refresh period comes, dp goes to low for turn on the rightmost decimal point.
- ② Another displaying values has been assigned. The rightmost value 7 has missed its digit period, but the reset value did not. Therefore, they changed first and the value 7 would display on the LED display when its next digit period arrives.
- ③ 7 should be shown on the LED display now, with the decimal point as well.

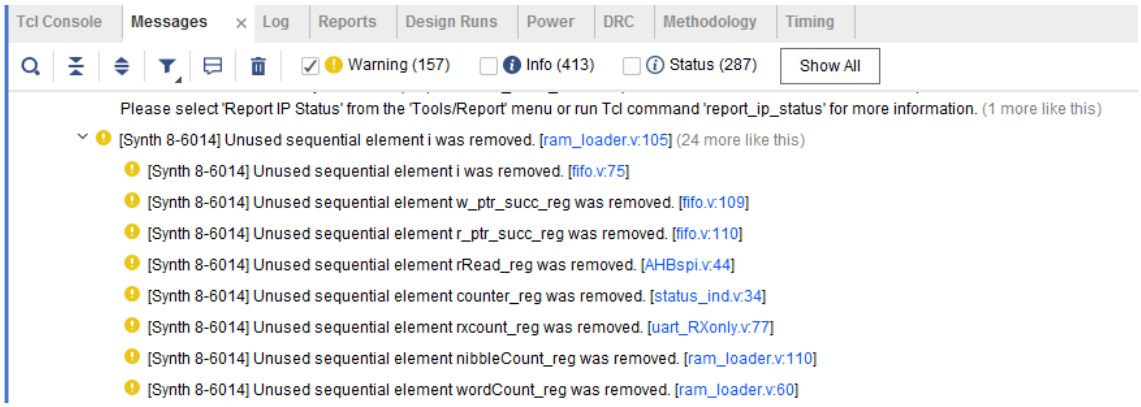


- ① Turn on all decimal point. Signal dp is 0 all the time due to decimal point active

Synthesis



These warnings are not the problem because these design blocks will not use these bits of each port.



There warnings here might have been caused by the different optimizing ways between simulator and synthesizer.

Resources:

AHBliteTop.v	AHBbram.v	AHBgpio.v	AHBMUX.v	CORTEXM0DS.v
AHBDCD.v	AHBprop.v	AHBuart2.v	clock_gen.v	Cortexm0ds_logic.v
AHB.Led2.v	ram_loader.v	reset_gen.v	status_ind.v	uart_RXoly.v
AHBSpi.v	vga_sync.v	Vga_pixels.v	uart_RXoly.v	ROMcode4.coe
hex2seg.v	fifo.v	AHBpixpos.v		Nexys4_Master.xdc

Testbench:

TB_ledtest.v	TB_spitest.v	TB_AHBgpio.v	TB_AHBuart2.v	TB_toplevel.v

Software:

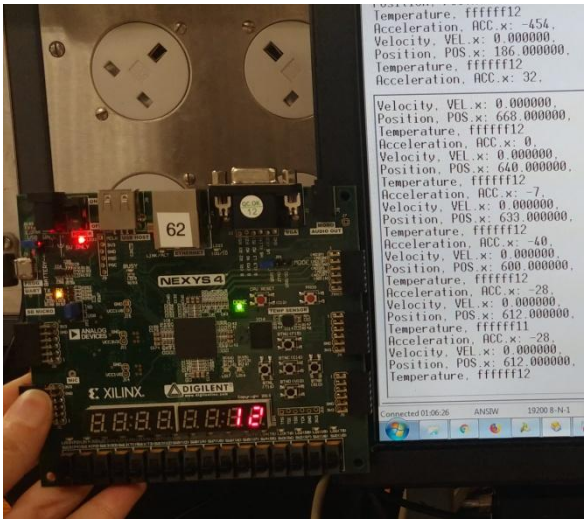
DES_MO_Soc.h	main.c

Implementation

Design Timing Summary

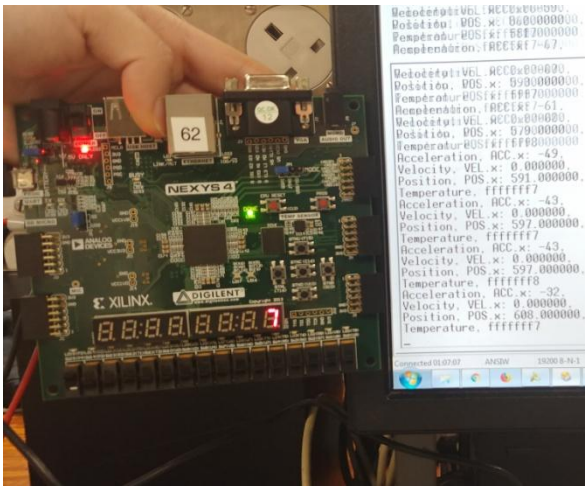
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.390 ns	Worst Hold Slack (WHS):	0.032 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4762	Total Number of Endpoints:	4762	Total Number of Endpoints:	1720

All user specified timing constraints are met.



Comparing the result shown on the LED diplay with the value showed in hyperterm. The value larger than 9 will make the LED digits blank.

The temperature value displayed on the LED is 12 and the value on the hyperterm windows is 12 as well



Placed my cold finger on the accelerometer.

The temperature value changed to 7 as shown on the hyperterm windows. The temperature value 7 was also displayed on the LED display.