T4 LAB - 3/12/2020

3.1 Bus Operation

```
# -------------------------------------------------------------------------- #
#
# Quartus Prime
# Version 16.1.2 Build 203 01/18/2017 SJ Standard Edition
# Date created = 14:04:54  December 03, 2020
#
# -------------------------------------------------------------------------- #
#
# Notes:
#
# 1) The default values for assignments are stored in the file:
#       playbus0_assignment_defaults.qdf
#    If this file doesn't exist, see file:
#       assignment_defaults.qdf
#
# 2) Altera recommends that you do not modify this file. This
#    file is updated automatically by the Quartus Prime software
#    and any changes you make may be lost or overwritten.
#
# -------------------------------------------------------------------------- #


set_global_assignment -name FAMILY "Cyclone V"
set_global_assignment -name DEVICE 5CSEMA5F31C6
set_global_assignment -name TOP_LEVEL_ENTITY playbus0
set_global_assignment -name ORIGINAL_QUARTUS_VERSION 16.1.2
set_global_assignment -name PROJECT_CREATION_TIME_DATE "14:04:54  DECEMBER 03, 2020"
set_global_assignment -name LAST_QUARTUS_VERSION "16.1.2 Standard Edition"
set_global_assignment -name SYSTEMVERILOG_FILE playbus0.sv
set_global_assignment -name SYSTEMVERILOG_FILE components.sv
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
set_global_assignment -name MIN_CORE_JUNCTION_TEMP 0
set_global_assignment -name MAX_CORE_JUNCTION_TEMP 85
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 256


set_location_assignment PIN_AE26 -to disp0[0]
set_location_assignment PIN_AE27 -to disp0[1]
set_location_assignment PIN_AE28 -to disp0[2]
set_location_assignment PIN_AG27 -to disp0[3]
set_location_assignment PIN_AF28 -to disp0[4]
set_location_assignment PIN_AG28 -to disp0[5]
set_location_assignment PIN_AH28 -to disp0[6]
set_location_assignment PIN_AJ29 -to disp1[0]
set_location_assignment PIN_AH29 -to disp1[1]
set_location_assignment PIN_AH30 -to disp1[2]
set_location_assignment PIN_AG30 -to disp1[3]
set_location_assignment PIN_AF29 -to disp1[4]
set_location_assignment PIN_AF30 -to disp1[5]
set_location_assignment PIN_AD27 -to disp1[6]

set_location_assignment PIN_AB12 -to sw0[0]
set_location_assignment PIN_AC12 -to sw0[1]
set_location_assignment PIN_AF9 -to sw0[2]
set_location_assignment PIN_AF10 -to sw0[3]
set_location_assignment PIN_AD11 -to RAMO
set_location_assignment PIN_AD12 -to ROMO
set_location_assignment PIN_AE11 -to SWBEN
set_location_assignment PIN_AC9 -to RAMW
set_location_assignment PIN_AD10 -to LEDLTCH
set_location_assignment PIN_Y16 -to n_clk
set_instance_assignment -name PARTITION_HIERARCHY root_partition -to | -section_id Top
set_global_assignment -name PARTITION_NETLIST_TYPE SOURCE -section_id Top
set_global_assignment -name PARTITION_FITTER_PRESERVATION_LEVEL PLACEMENT_AND_ROUTING -section_id Top
set_global_assignment -name PARTITION_COLOR 16764057 -section_id Top
```

Diagram of Code with pin configuration added on

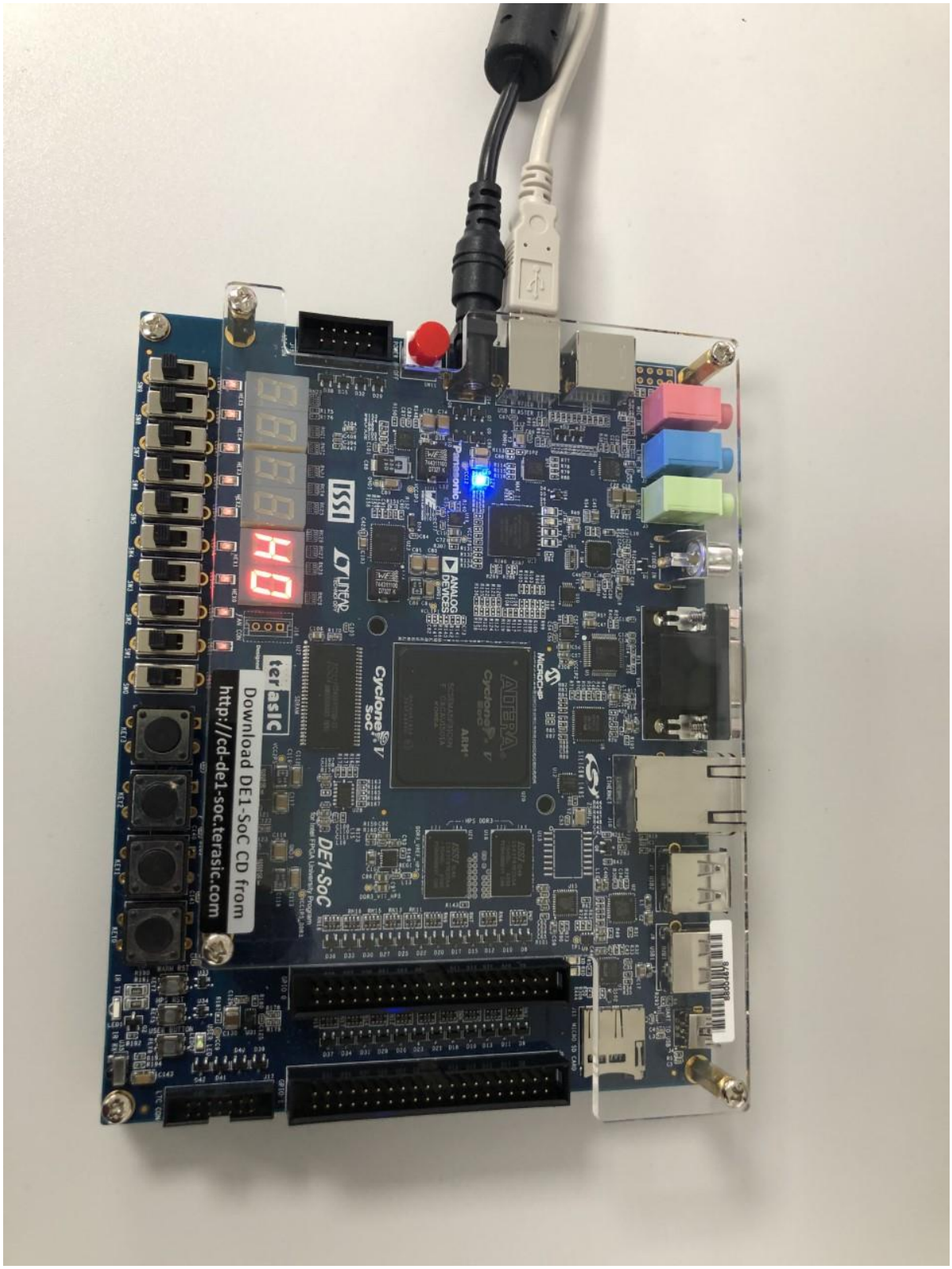


Diagram of FPGA with ‘HO’ displayed

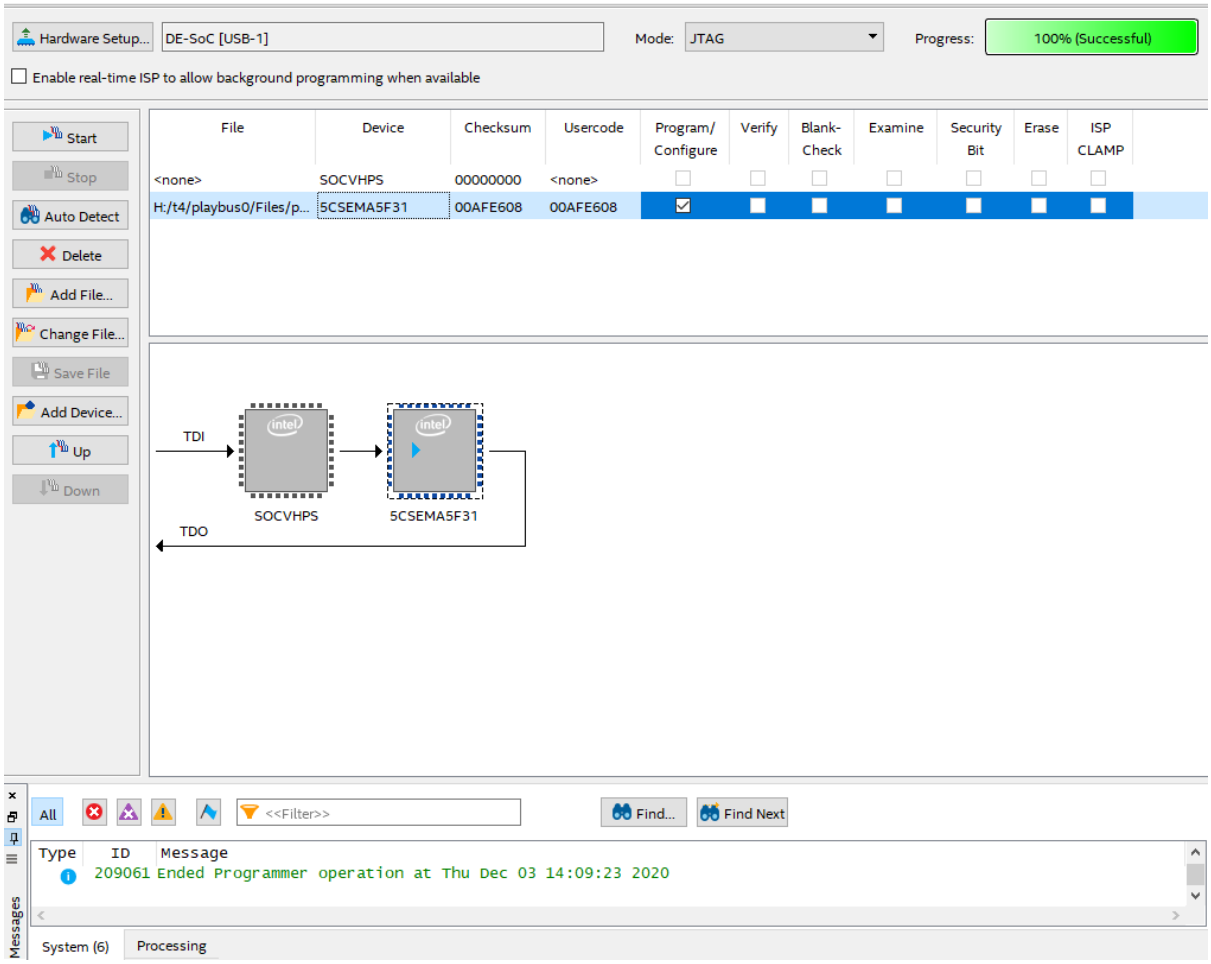
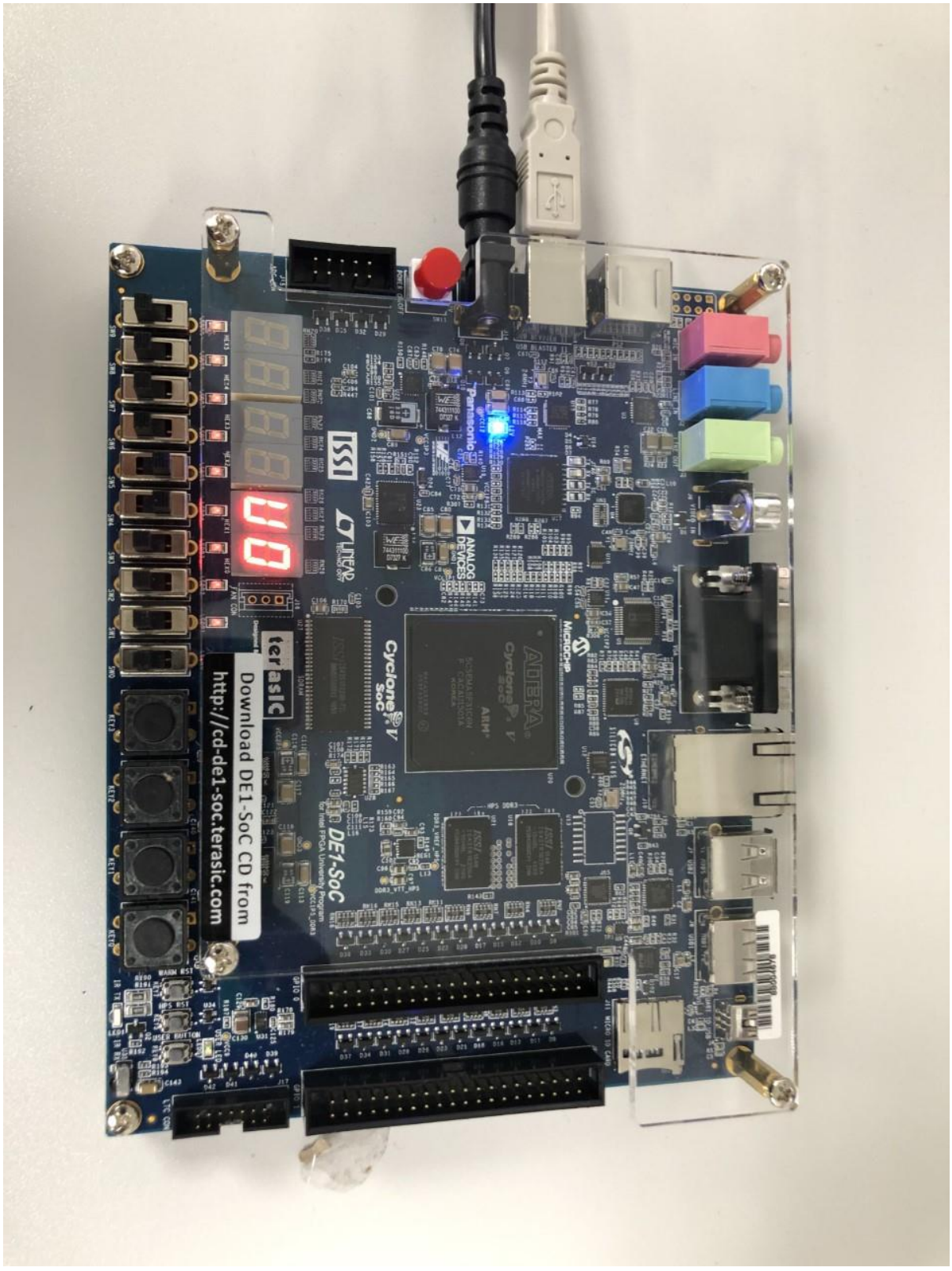


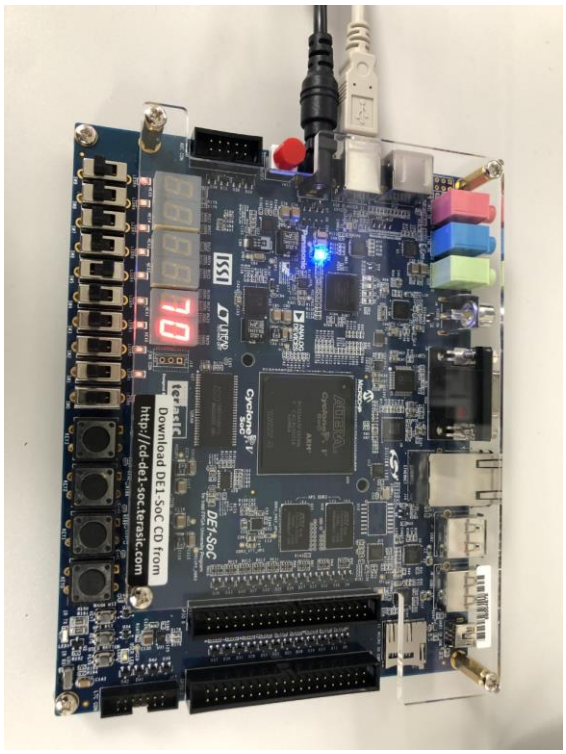
Diagram of Quartus Programmer with the code written to FPGA



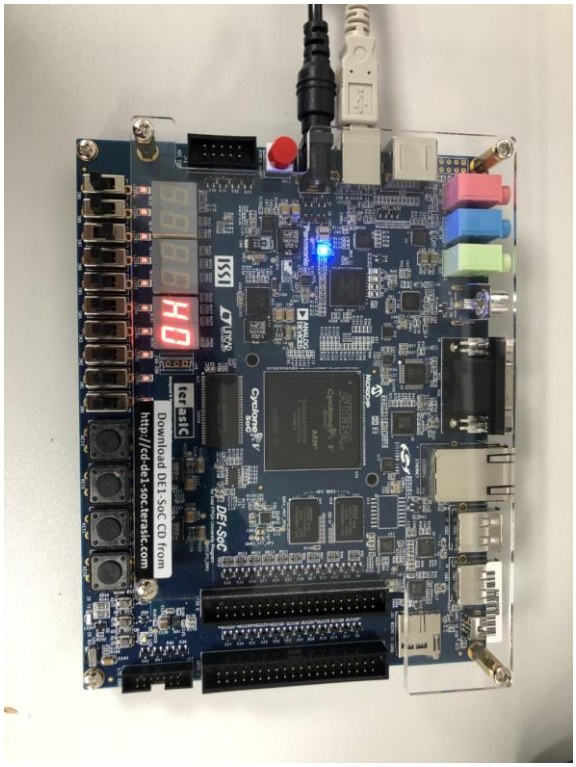
All the operations listed in TABLE 3 work correctly. I verified this by using the switches on the board SW0 – SW9 to copy data and read data.

Operation

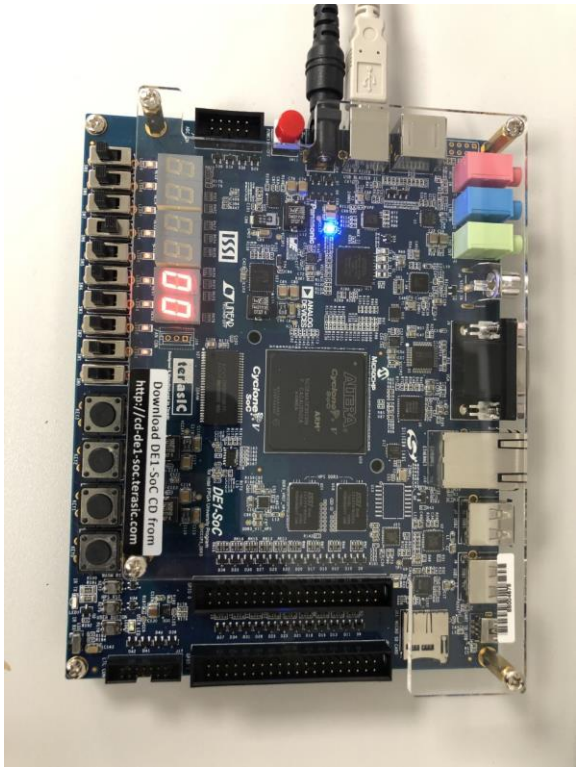
Read data from EPROM at address 5 onto bus - By flipping ROMO on the FPGA as shown below



Read data from RAM at address 5 onto bus - By flipping RAMW on the FPGA as shown below



Read data from Switches onto bus - By flipping SWBEN on the FPGA as shown below



Copy data from Switches into RAM at address 5 – By flipping on SWBEN and then RAMW. Then using RAMO to view the data

Copy data from EPROM into RAM at address 5 - By flipping on ROMO and then RAMW. Then using RAMO to view the data

Copy data from Switches into LEDs - By flipping on SWBEN and then LEDLTCH.

Copy data from EPROM at address into LEDs - By flipping on ROMO and then LEDLTCH.

Copy data from RAM at address into LEDs - By flipping on RAMW and then LEDLTCH.

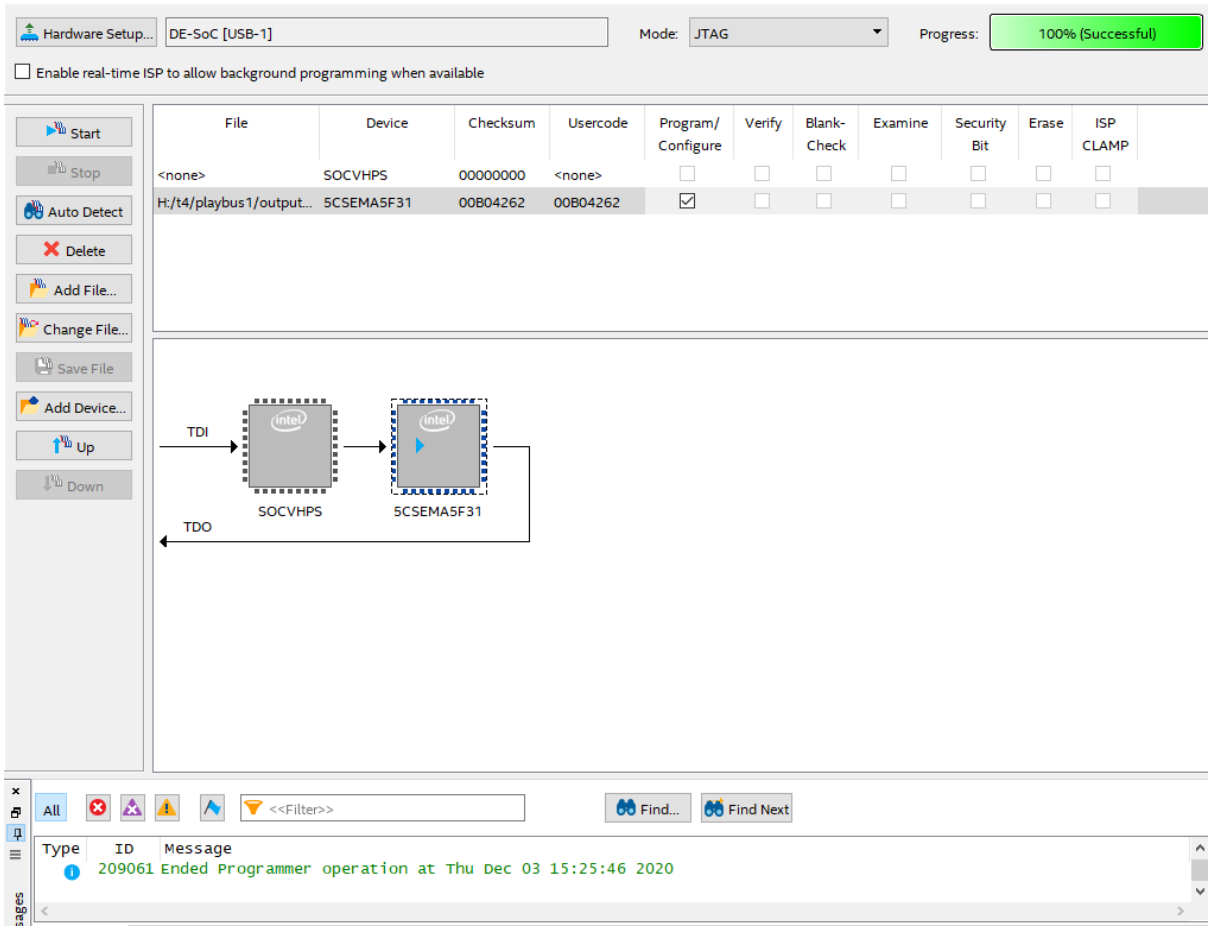
Q: How does the FPGA system determine when the data bus is in the ‘Z’ state?

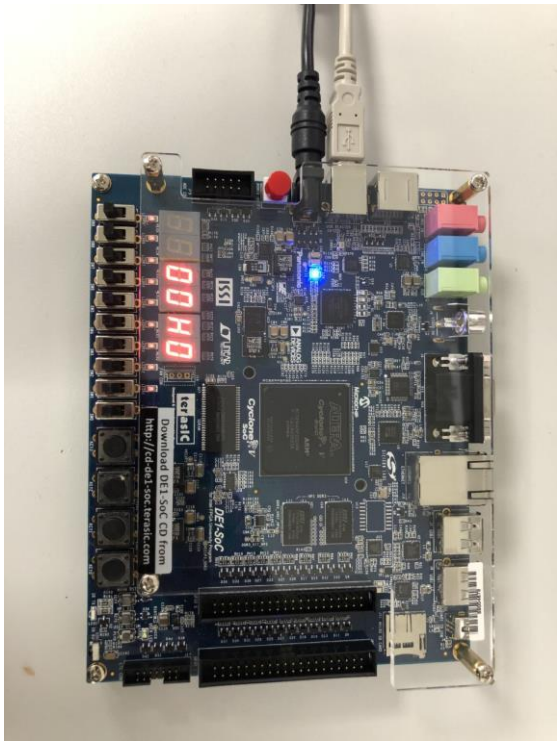
The Z state is represented by a ‘H’ on the FPGA.

Q: SystemVerilog and ModelSim can represent ‘Z’ and ‘X’ states, but are these states “real”? In other words, is it possible to design logic that would detect ‘Z’ or ‘X’ states and show them on a 7-segment display?

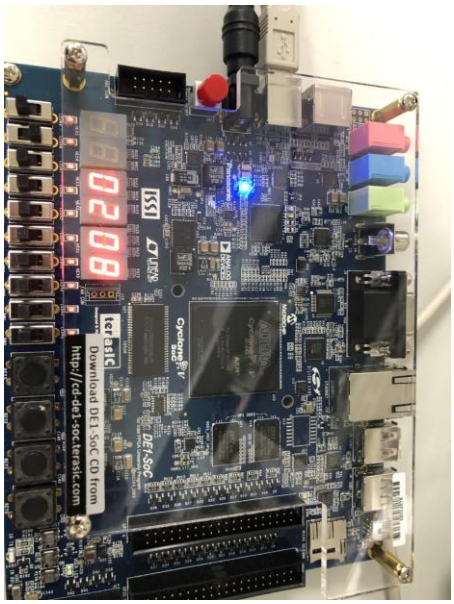
The states are not real and merely a simulation of the states represented on the 7 segment display. A ‘H’ state would not be possible as it involves taking the FPGA to either low nor high state. This is not possible as resistors will draw current.

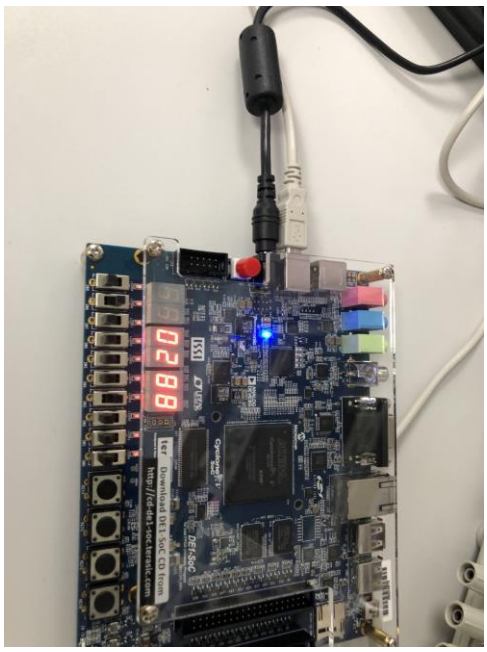
3.2 Controller Operation



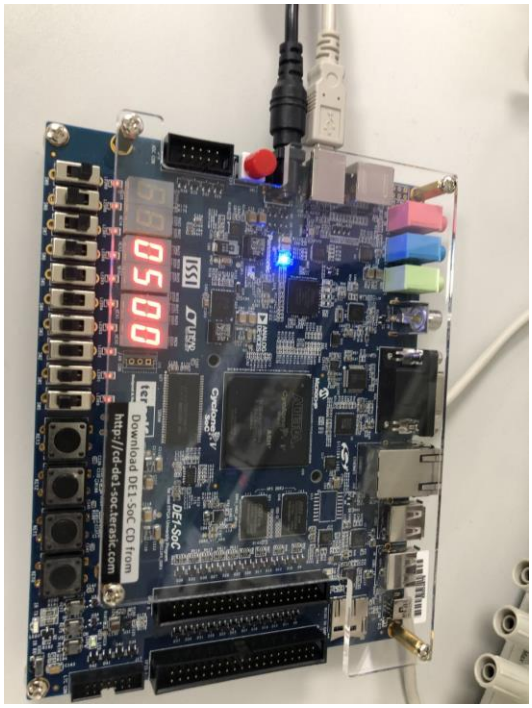


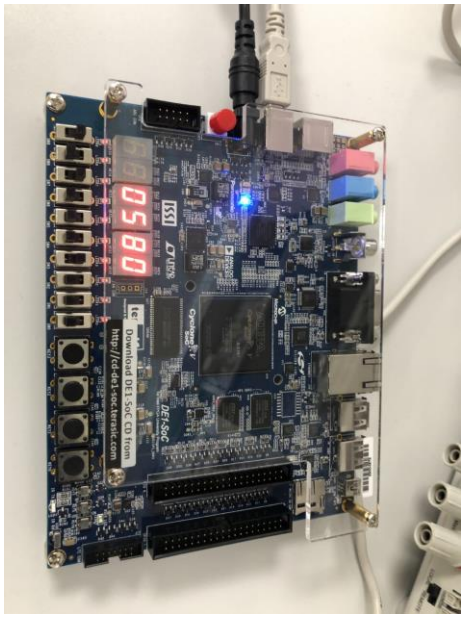
Function 2: Read data from Switches onto bus - Flip the switches in the switches section of the board





Function 5: Before –





After implemented –

Q: Are the signals that you observe consistent with the waveforms from your simulation? If not, why not?

Yes they are.

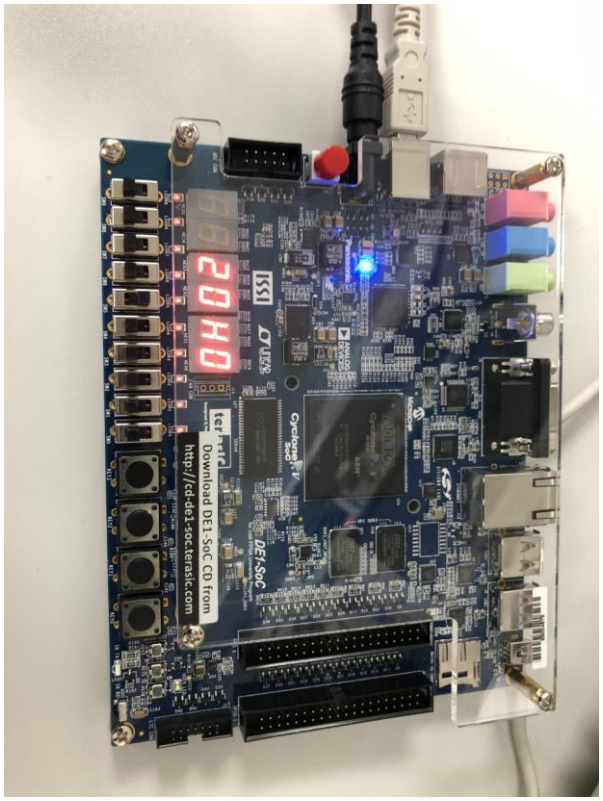
3.3

Q: Why is it a good idea to simulate code before downloading it onto the FPGA? Hint: How easy is it to observe what is happening inside your design on the FPGA compared with in the simulator

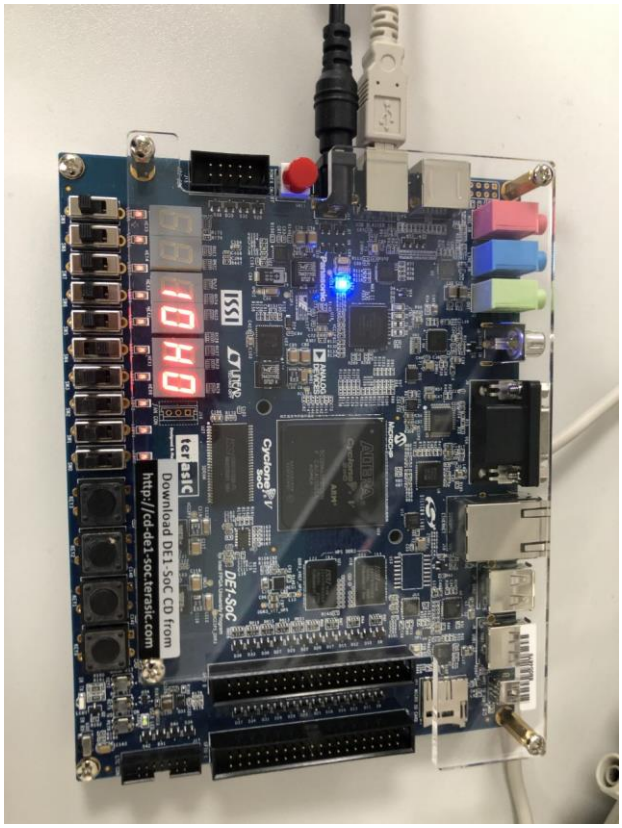
The simulation enables me to see the waveforms and actual working of the functions which will be harder to see on the FPGA

Operation

Read data from EPROM at address onto bus

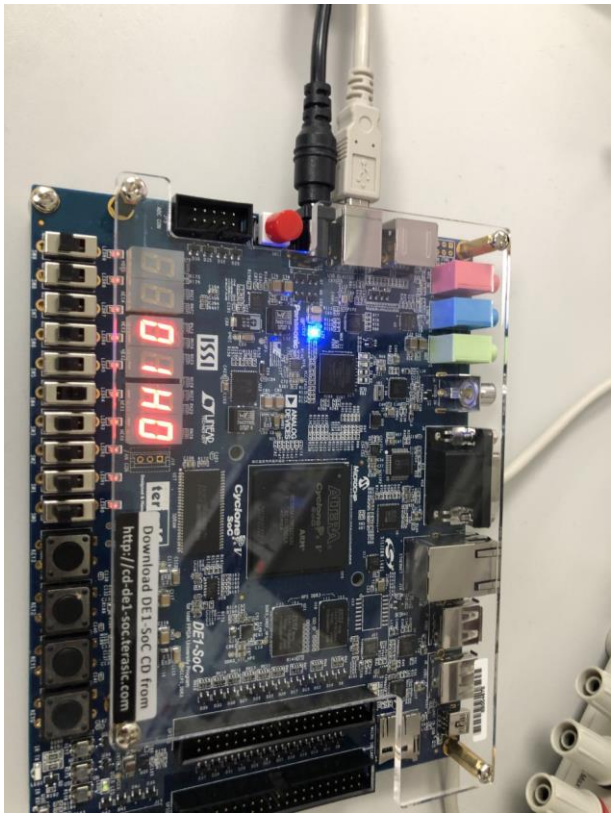


Read data from RAM at address onto bus

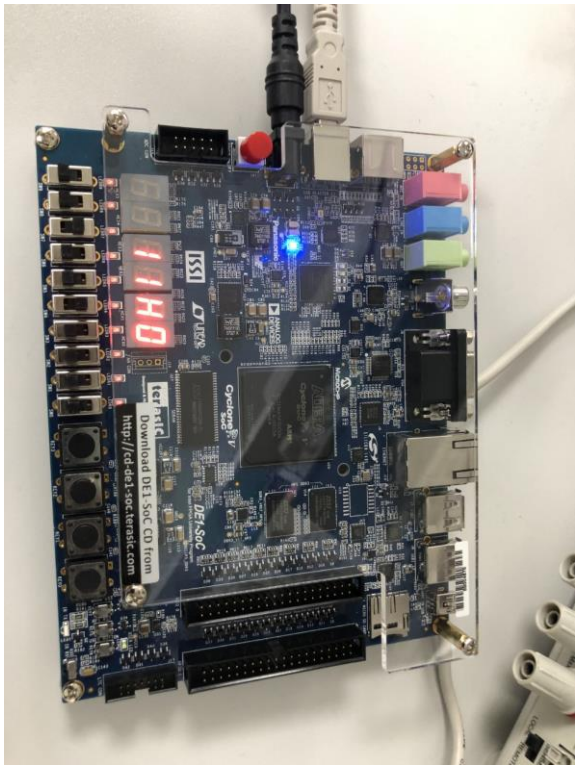


Copy data from Switches into RAM at address

Switches before-



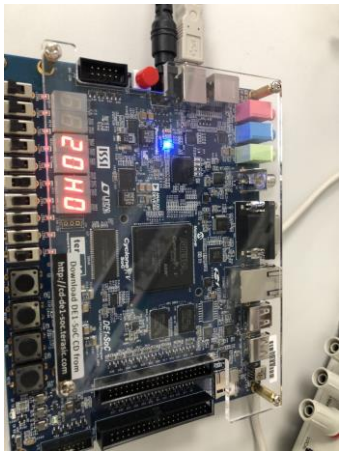
Switches after-



Copy data from EPROM into RAM at address

- RAM before

- Data at EPROM



Copy data from EPROM at address into LEDs

Copy data from RAM at address into LEDs

Q: For each of the functions, is the observed behaviour consistent with the simulations? If not, why not?

Yes

Q: For these new functions, can you read from and write to different addresses?

Yes, as shown above