Getting started with Vitis & Vivado

06/27/2025

Setting Up

Tools

Download:

- Vitis
- Vivado (I used the 2025.1 version for both)

Physical:

- PYNQ Z2 board
- USB cables to connect to your computer

Set up your PYNQ board

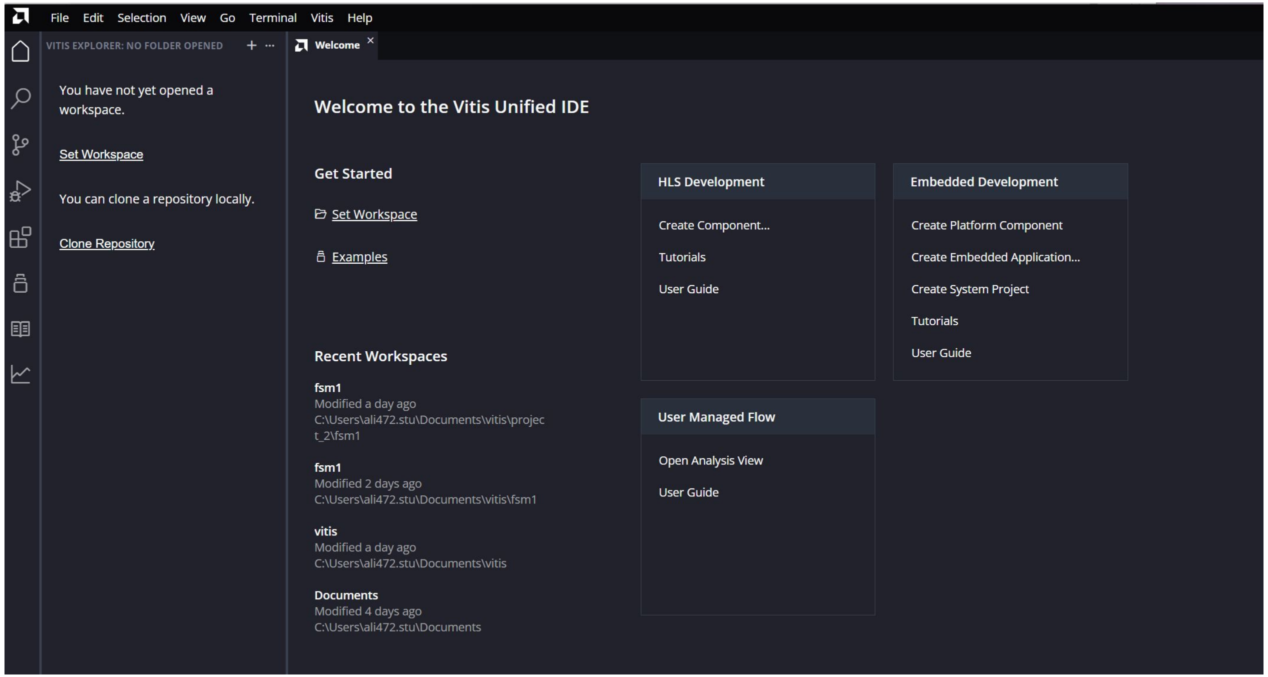
- Boot the PYNQ board with the SD card image
- Physically connect the board to your computer with a USB-C cord or other
- Power the board on (little black switch located near the edge of the board)
- Enable the board to access Wifi:
  - Desktop settings -> network and internet -> advanced network settings -> Ethernet 3 -> properties -> enter IP address (192.168.2.99, or google “PYNQ jupyter IP address”)
  - Only need to do this once
- Open a browser to the board’s IP
  - In the search bar, type “192.168.2.99” -> this should automatically open JupyterLab
  - If prompted for a username and/or a password, type ‘xilinx’ (all lowercase)

- Now you can use PYNQ’s Python APIs to interact with the board!

Vitis + Vivado Workflow

Step 1: Create new Vitis component

Includes specifying the part and the directory containing header files and the top level function



You have not yet opened a workspace.

Set Workspace

You can clone a repository locally.

Clone Repository

Welcome to the Vitis Unified IDE

Get Started

Set Workspace

Examples

Recent Workspaces

fsm1
Modified a day ago
C:\Users\ali472.stu\Documents\vitis\project_2\fsm1

fsm1
Modified 2 days ago
C:\Users\ali472.stu\Documents\vitis\fsm1

vitis
Modified a day ago
C:\Users\ali472.stu\Documents\vitis

Documents
Modified 4 days ago
C:\Users\ali472.stu\Documents

HLS Development

Create Component...
Tutorials
User Guide

Embedded Development

Create Platform Component
Create Embedded Application...
Create System Project
Tutorials
User Guide

User Managed Flow

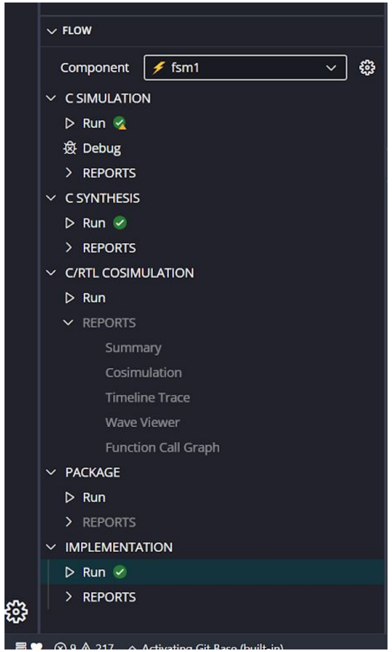
Open Analysis View
User Guide

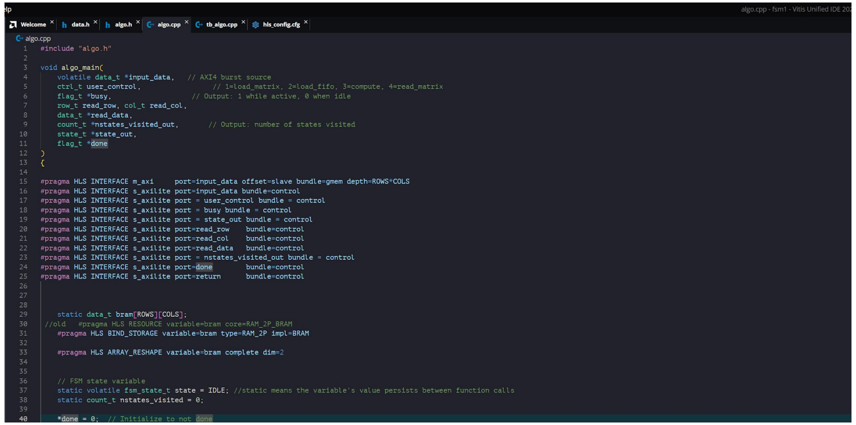
Step 2: Edit, simulate, synthesize, and implement C files in Vitis

(1)

(2)

(3)



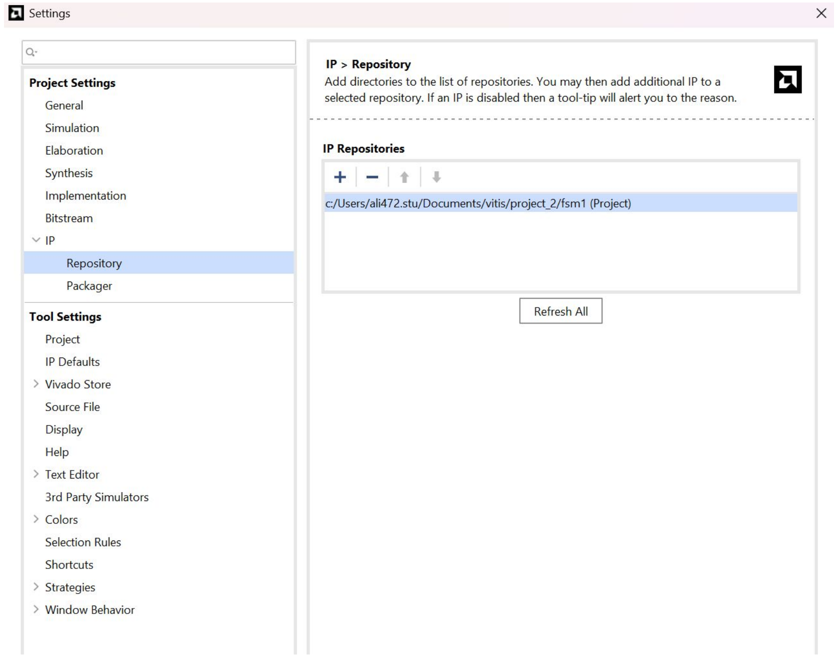


```cpp
#include "algo.h"

void algo_main(
    volatile data_t *input_data,   // AXI4 burst source
    ctrl_t user_control,                 // 1=load_matrix, 2=load_fifo, 3=compute, 4=read_matrix
    flag_t *busy,                    // Output: 1 while active, 0 when idle
    row_t read_row, col_t read_col,
    data_t *read_data,
    count_t *nstates_visited_out,       // Output: number of states visited
    state_t *state_out,
    flag_t *done
)
{

#pragma HLS INTERFACE m_axi     port=input_data offset=slave bundle=gmem depth=ROWS*COLS
#pragma HLS INTERFACE s_axilite port=input_data bundle=control
#pragma HLS INTERFACE s_axilite port = user_control bundle = control
#pragma HLS INTERFACE s_axilite port = busy bundle = control
#pragma HLS INTERFACE s_axilite port = state_out bundle = control
#pragma HLS INTERFACE s_axilite port=read_row    bundle=control
#pragma HLS INTERFACE s_axilite port=read_col    bundle=control
#pragma HLS INTERFACE s_axilite port=read_data   bundle=control
#pragma HLS INTERFACE s_axilite port = nstates_visited_out bundle = control
#pragma HLS INTERFACE s_axilite port=done        bundle=control
#pragma HLS INTERFACE s_axilite port=return      bundle=control



    static data_t bram[ROWS][COLS];
 //old   #pragma HLS RESOURCE variable=bram core=RAM_2P_BRAM
    #pragma HLS BIND_STORAGE variable=bram type=RAM_2P impl=BRAM

    #pragma HLS ARRAY_RESHAPE variable=bram complete dim=2


    // FSM state variable
    static volatile fsm_state_t state = IDLE; //static means the variable's value persists between function calls
    static count_t nstates_visited = 0;

    *done = 0;  // Initialize to not done
```

Example C code snippet

Step 3: Create new project in Vivado

Set repository to the upper-level folder containing the top level function and header files (from Project Manager -> settings)



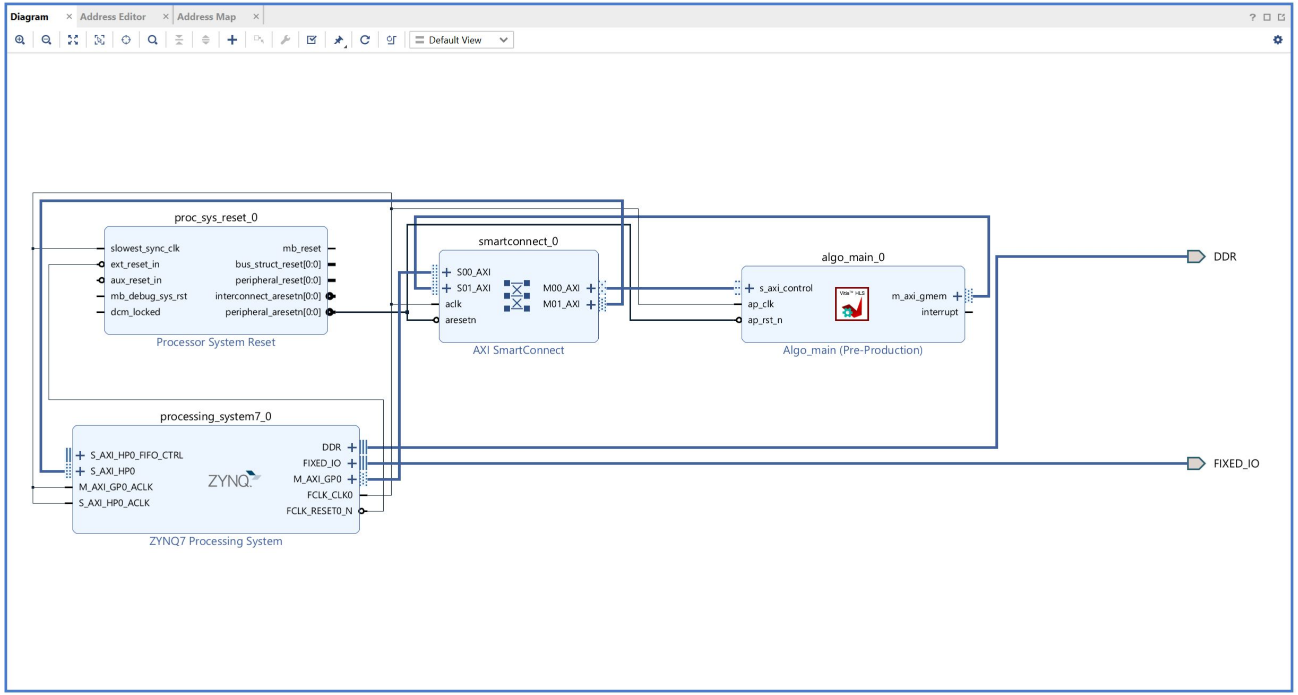
IP > Repository
Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason.

IP Repositories

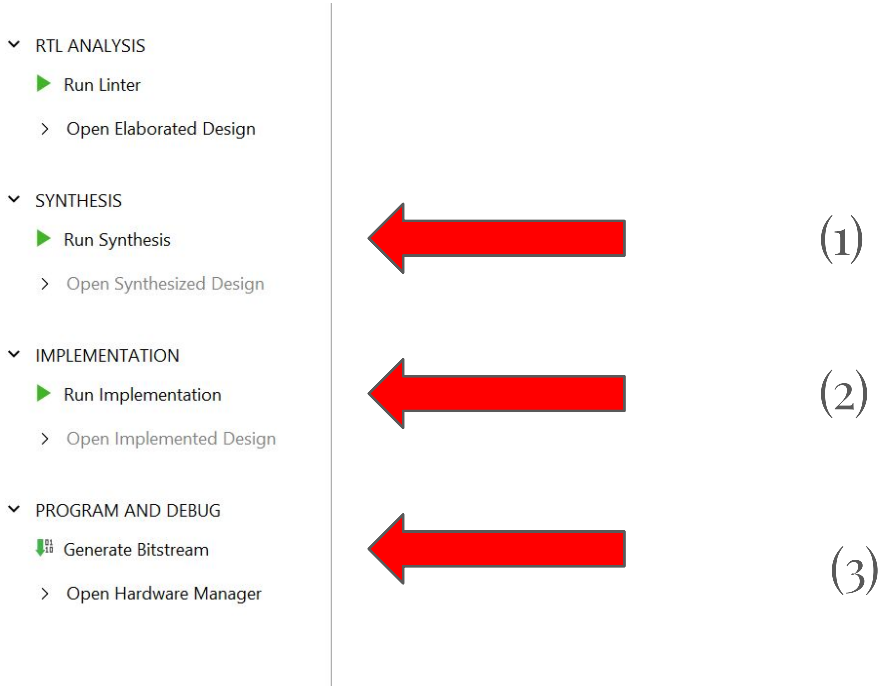
c:/Users/ali472.stu/Documents/vitis/project_2/fsm1 (Project)

Step 4: Create Block Diagram

Can accept automatic connections, but may need to create new connections and add ports (eg. for m_axi_gmem on the algo_main block)



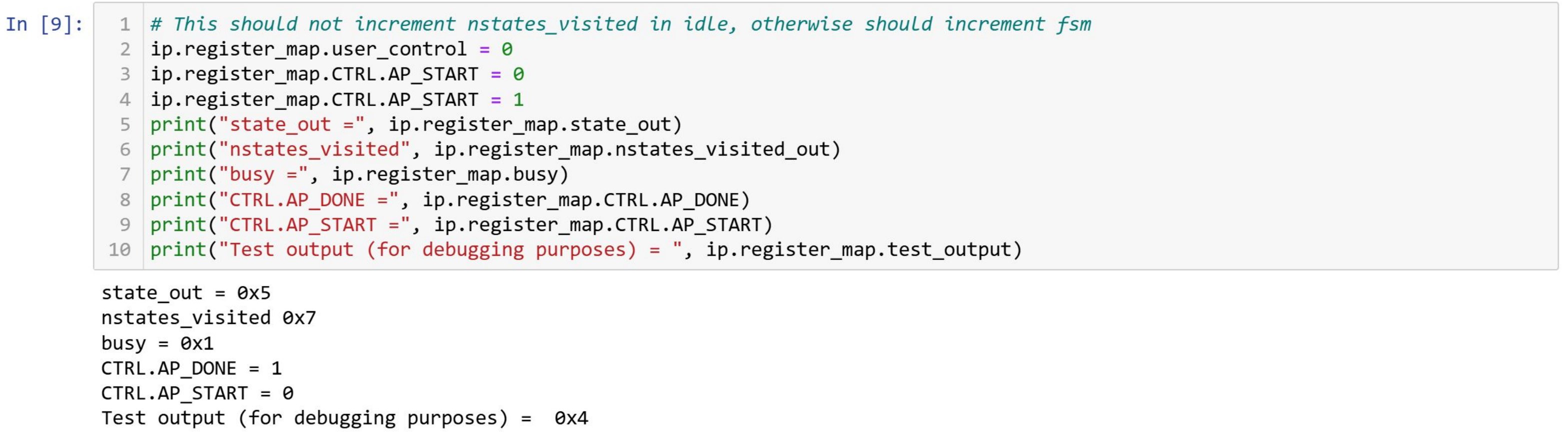
Step 5: Run synthesis, implementation, and generate bitstream



Found under the left-hand flow navigation bar

Step 6: Test hardware implementation with JupyterLab

- Log in to the correct IP address
- Upload .bit and .hwh files from Vivado to the same folder as the python script, ensuring that both files have the same name
- Run desired test cases and see outputs
- Note: repeat the workflow every time a change is made to the top function in Vitis



In [9]:

```python
# This should not increment nstates_visited in idle, otherwise should increment fsm
ip.register_map.user_control = 0
ip.register_map.CTRL.AP_START = 0
ip.register_map.CTRL.AP_START = 1
print("state_out =", ip.register_map.state_out)
print("nstates_visited", ip.register_map.nstates_visited_out)
print("busy =", ip.register_map.busy)
print("CTRL.AP_DONE =", ip.register_map.CTRL.AP_DONE)
print("CTRL.AP_START =", ip.register_map.CTRL.AP_START)
print("Test output (for debugging purposes) = ", ip.register_map.test_output)
```

```
state_out = 0x5
nstates_visited 0x7
busy = 0x1
CTRL.AP_DONE = 1
CTRL.AP_START = 0
Test output (for debugging purposes) =  0x4
```

Debugging tips and solutions to common problems

Problems I ran into:

C level (Vitis)

- Unexpected outputs from Jupyter test cases
  - Debug using the test bench (first level of debugging: assess the C code)
  - Co-sim requires specification of fifo depth (under m_axi pragma)

Vivado level

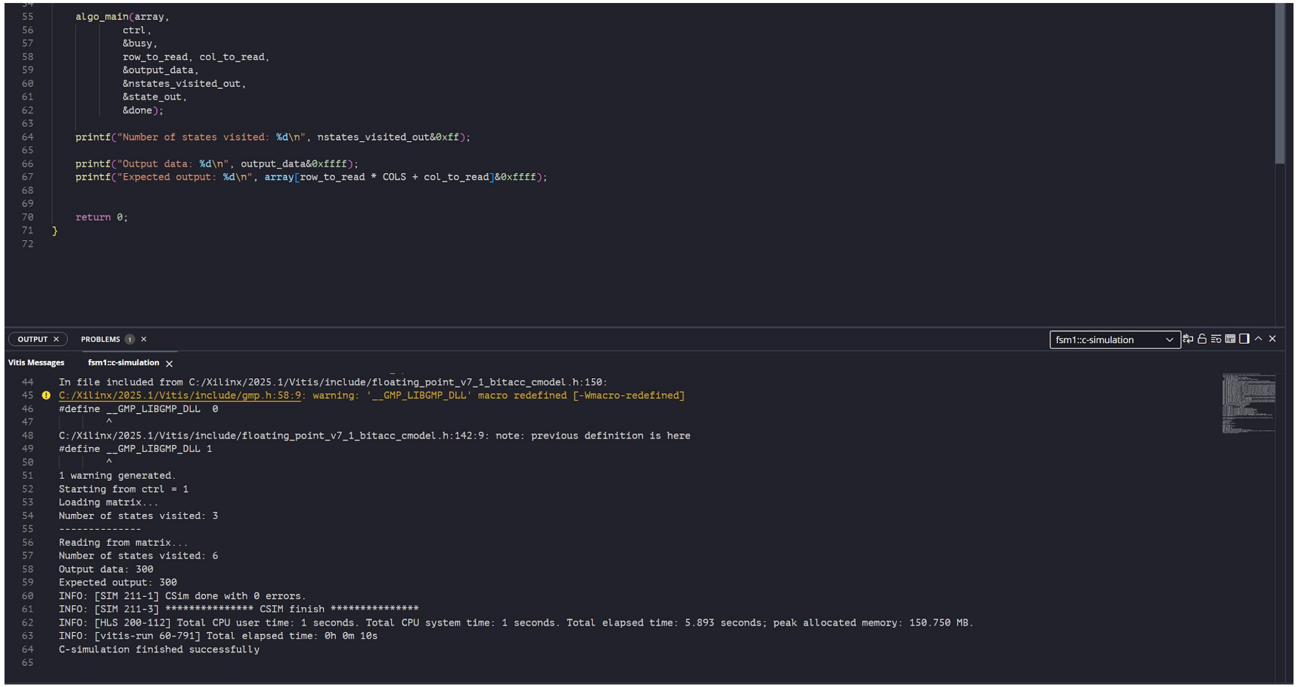
- FSM getting stuck in “load_matrix” state
  - Block diagram not properly connected
- FSM loading unexpected data
  - Ensure addresses have been properly assigned (in the block diagram); right click anywhere in the address editor and unassign and re-assign all

Jupyter level

- Overlay() referencing old files or outputting “resource busy”
  - Reboot the board or run PL.reset()

Debugging process

1) Assess correctness of C code with Vitis test bench (run C simulation)



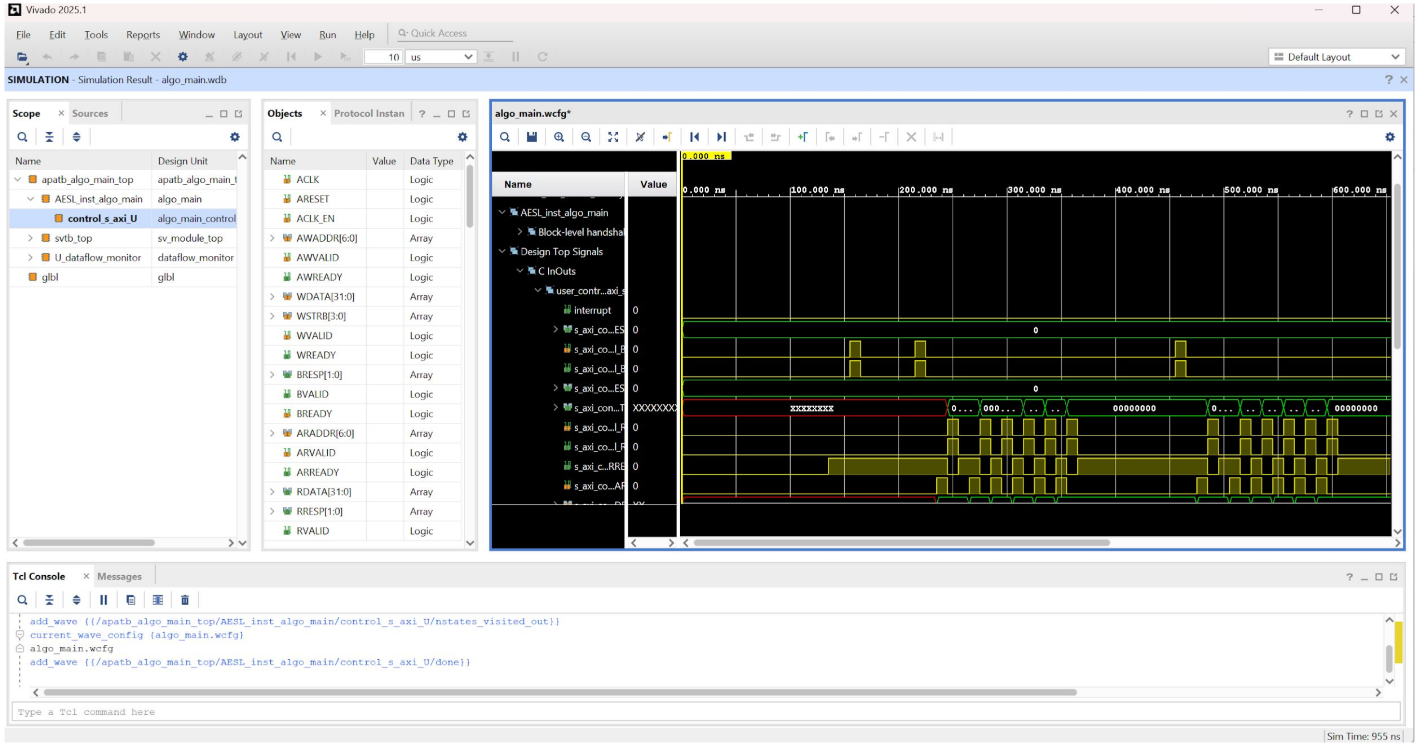
```c
    algo_main(array,
            ctrl,
            &busy,
            row_to_read, col_to_read,
            &output_data,
            &nstates_visited_out,
            &state_out,
            &done);

    printf("Number of states visited: %d\n", nstates_visited_out&0xff);

    printf("Output data: %d\n", output_data&0xffff);
    printf("Expected output: %d\n", array[row_to_read * COLS + col_to_read]&0xffff);


    return 0;
}
```

OUTPUT PROBLEMS 1 — fsm1::c-simulation

Vitis Messages — fsm1::c-simulation

```
In file included from C:/Xilinx/2025.1/Vitis/include/floating_point_v7_1_bitacc_cmodel.h:150:
C:/Xilinx/2025.1/Vitis/include/gmp.h:58:9: warning: '__GMP_LIBGMP_DLL' macro redefined [-Wmacro-redefined]
#define __GMP_LIBGMP_DLL  0
        ^
C:/Xilinx/2025.1/Vitis/include/floating_point_v7_1_bitacc_cmodel.h:142:9: note: previous definition is here
#define __GMP_LIBGMP_DLL 1
        ^
1 warning generated.
Starting from ctrl = 1
Loading matrix...
Number of states visited: 3
-------------
Reading from matrix...
Number of states visited: 6
Output data: 300
Expected output: 300
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ************** CSIM finish **************
INFO: [HLS 200-112] Total CPU user time: 1 seconds. Total CPU system time: 1 seconds. Total elapsed time: 5.893 seconds; peak allocated memory: 150.750 MB.
INFO: [vitis-run 60-791] Total elapsed time: 0h 0m 10s
C-simulation finished successfully
```

2) Assess predicted signal behaviour with the Vitis Co-simulation



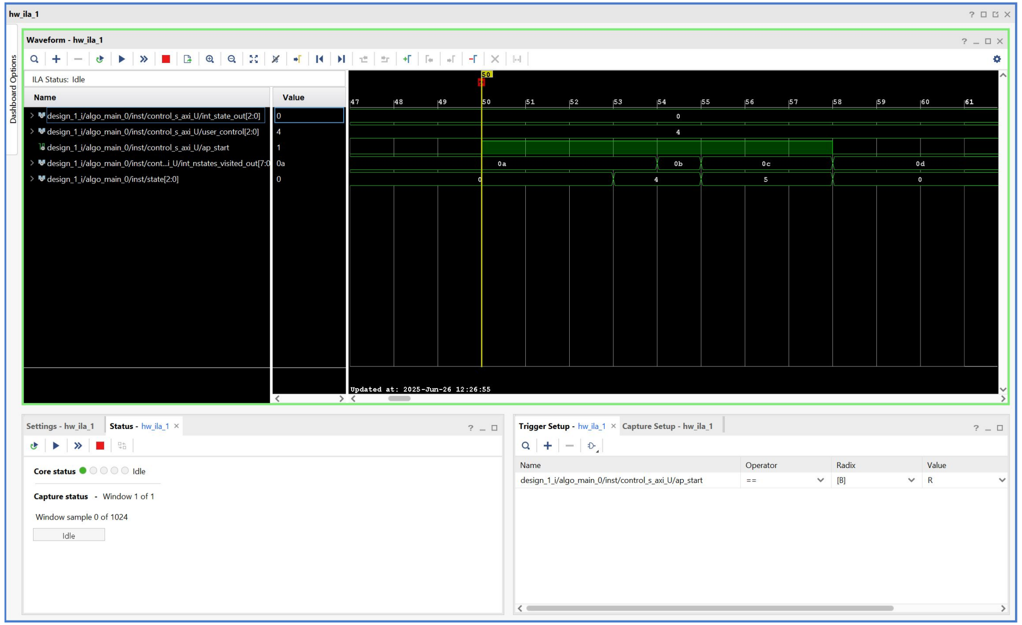
Name	Design Unit
apatb_algo_main_top	apatb_algo_main_t
AESL_inst_algo_main	algo_main
control_s_axi_U	algo_main_control
svtb_top	sv_module_top
U_dataflow_monitor	dataflow_monitor
glbl	glbl

Name	Value	Data Type
ACLK		Logic
ARESET		Logic
ACLK_EN		Logic
AWADDR[6:0]		Array
AWVALID		Logic
AWREADY		Logic
WDATA[31:0]		Array
WSTRB[3:0]		Array
WVALID		Logic
WREADY		Logic
BRESP[1:0]		Array
BVALID		Logic
BREADY		Logic
ARADDR[6:0]		Array
ARVALID		Logic
ARREADY		Logic
RDATA[31:0]		Array
RRESP[1:0]		Array
RVALID		Logic

```
add_wave {{/apatb_algo_main_top/AESL_inst_algo_main/control_s_axi_U/nstates_visited_out}}
current_wave_config {algo_main.wcfg}
algo_main.wcfg
add_wave {{/apatb_algo_main_top/AESL_inst_algo_main/control_s_axi_U/done}}
```

Type a Tcl command here

3) Assess actual signal output with Vivado ILA debug option



ILA Status: Idle

Name	Value
design_1_i/algo_main_0/inst/control_s_axi_U/int_state_out[2:0]	0
design_1_i/algo_main_0/inst/control_s_axi_U/user_control[2:0]	4
design_1_i/algo_main_0/inst/control_s_axi_U/ap_start	1
design_1_i/algo_main_0/inst/cont...i_U/int_nstates_visited_out[7:0	0a
design_1_i/algo_main_0/inst/state[2:0]	0

```
Updated at: 2025-Jun-26 12:26:55
```

Core status ● Idle

Capture status - Window 1 of 1

Window sample 0 of 1024

Idle

Name	Operator	Radix	Value
design_1_i/algo_main_0/inst/control_s_axi_U/ap_start	==	[B]	R

- Select nets to debug under Synthesis -> set up debug
  - When prompted, select Capture Control and Advanced Trigger
- Generate bitstream (will automatically ask to run implementation)
- Open Hardware Manager. If this is the first time configuring debug ILA cores for the project, select Open Target; else click Program Device
- Set up trigger to capture based on the value of a certain variable (if desired)
- Re-upload the .bit file to Jupyter, run the Overlay() code, enable the trigger in Vivado, and run the test case

Resources used:

Official PYNQ Z2 board set-up tutorial:

https://pynq.readthedocs.io/en/v2.3/getting_started/pynq_z2_setup.html

Official Vivado tutorials for setting up debugging using ILA cores:

Running the Set Up Debug Wizard • Vivado Design Suite Tutorial: Programming and Debugging (UG936) • Reader • AMD Technical Information Portal

Using the Vivado Integrated Logic Analyzer • Vivado Design Suite Tutorial: Programming and Debugging (UG936) • Reader • AMD Technical Information Portal