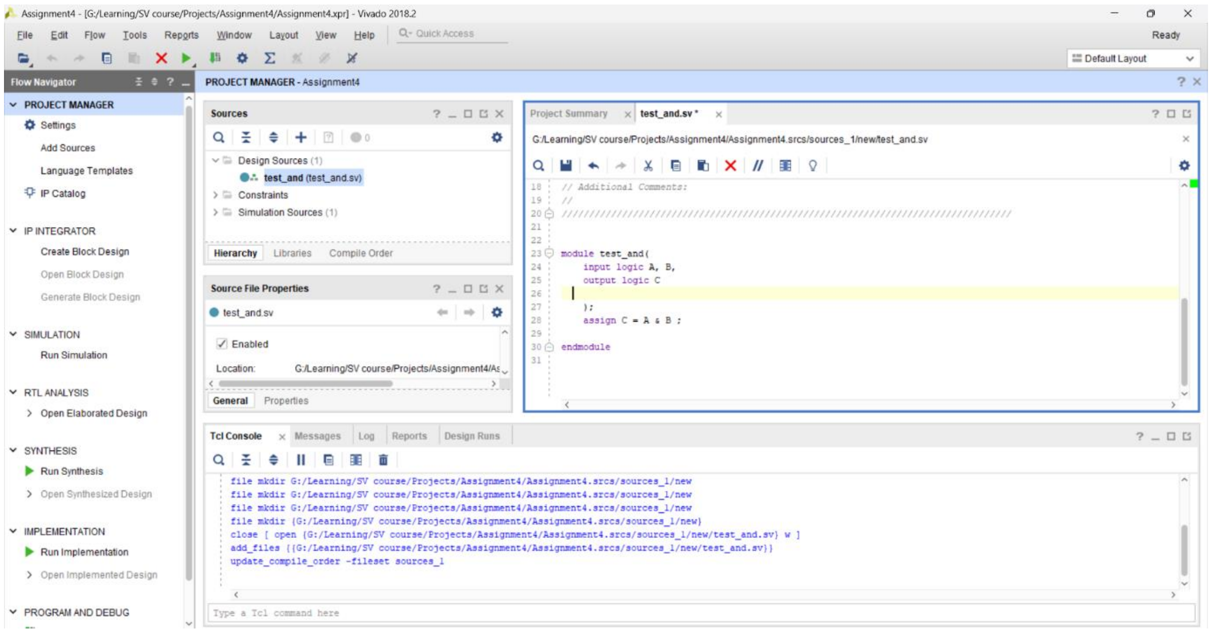
System Verilog for ASIC/FPGA Design & Simulation 2023 -In01
Assignment 4 – R.P.U.A. Pathirana
FPGA Implementation

In this lab session following digital designs were implemented on the Xilinx zybo FPGA board.

1. And Gate
2. 4-bit counter
3. UART Matrix Vector Multiplier

This report contains screenshots of the steps and results of the implementation.

1. AND Gate Implementation

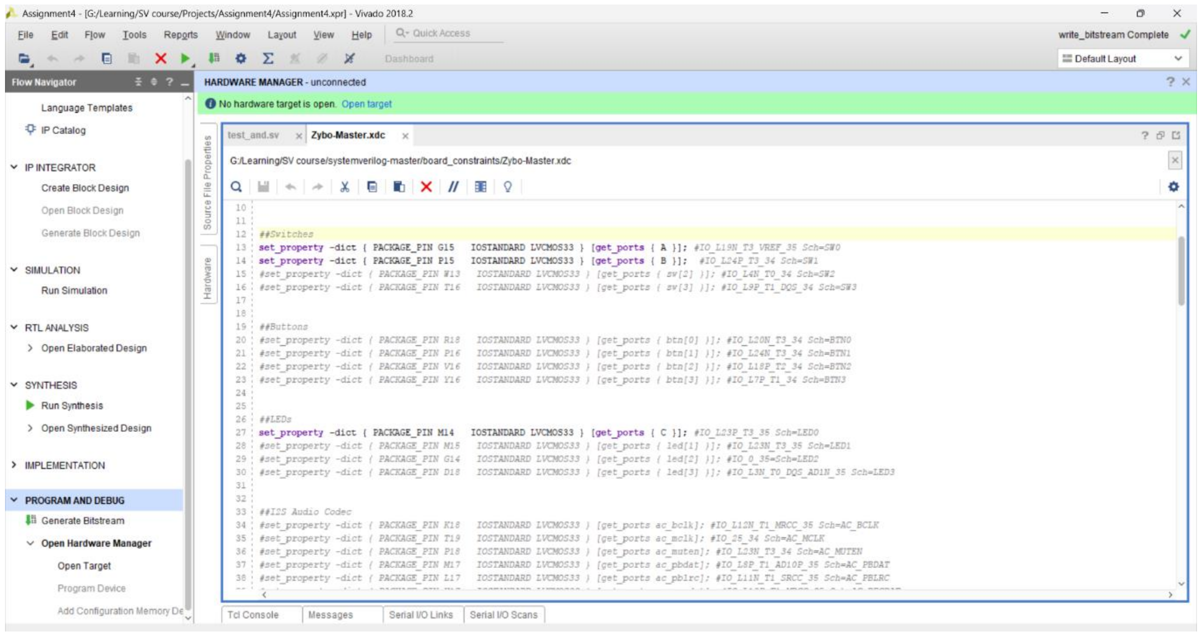


G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new/test_and.sv

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test_and(
    input logic A, B,
    output logic C

    );
    assign C = A & B ;

endmodule
```

```
file mkdir G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new
file mkdir G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new
file mkdir G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new
file mkdir {G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new}
close [ open {G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new/test_and.sv} w ]
add_files {{G:/Learning/SV course/Projects/Assignment4/Assignment4.srcs/sources_1/new/test_and.sv}}
update_compile_order -fileset sources_1
```

Modifying the contrarian file



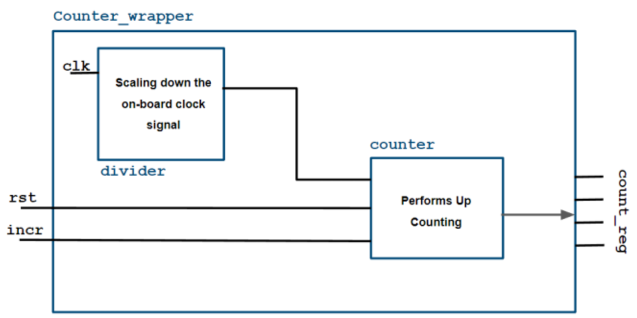
G:/Learning/SV course/systemverilog-master/board_constraints/Zybo-Master.xdc

```


##Switches
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { A }]; #IO_L19N_T3_VREF_35 Sch=SW0
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { B }];  #IO_L24P_T3_34 Sch=SW1
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=SW2
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3


##Buttons
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L20N_T3_34 Sch=BTN0
#set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L24N_T3_34 Sch=BTN1
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L18P_T2_34 Sch=BTN2
#set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L7P_T1_34 Sch=BTN3


##LEDs
set_property -dict { PACKAGE_PIN M14   IOSTANDARD LVCMOS33 } [get_ports { C }]; #IO_L23P_T3_35 Sch=LED0
#set_property -dict { PACKAGE_PIN M15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L23N_T3_35 Sch=LED1
#set_property -dict { PACKAGE_PIN G14   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_0_35=Sch=LED2
#set_property -dict { PACKAGE_PIN D18   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L3N_T0_DQS_AD1N_35 Sch=LED3


##I2S Audio Codec
#set_property -dict { PACKAGE_PIN K18   IOSTANDARD LVCMOS33 } [get_ports ac_bclk]; #IO_L12N_T1_MRCC_35 Sch=AC_BCLK
#set_property -dict { PACKAGE_PIN T19   IOSTANDARD LVCMOS33 } [get_ports ac_mclk]; #IO_25_34 Sch=AC_MCLK
#set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports ac_muten]; #IO_L23N_T3_34 Sch=AC_MUTEN
#set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports ac_pbdat]; #IO_L8P_T1_AD10P_35 Sch=AC_PBDAT
#set_property -dict { PACKAGE_PIN L17   IOSTANDARD LVCMOS33 } [get_ports ac_pblrc]; #IO_L11N_T1_SRCC_35 Sch=AC_PBLRC
```

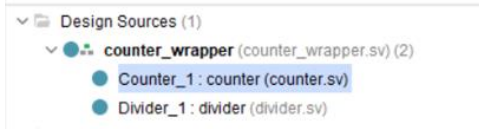
Then the bit-stream was generated. The board was connected to the PC and the code was uploaded.

Working Video

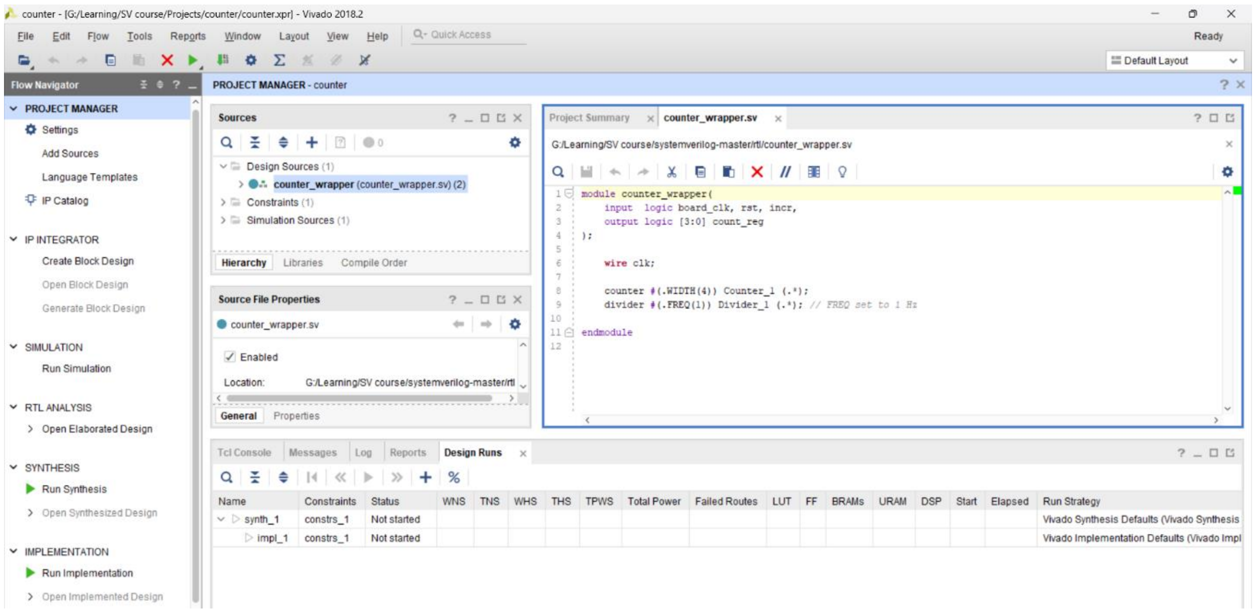
2. 4-bit Counter implementation



Since there is a couple of modules inside the counter wrapper, the counter wrapper was selected as the top module



- Design Sources (1)
  - counter_wrapper (counter_wrapper.sv) (2)
    - Counter_1 : counter (counter.sv)
    - Divider_1 : divider (divider.sv)



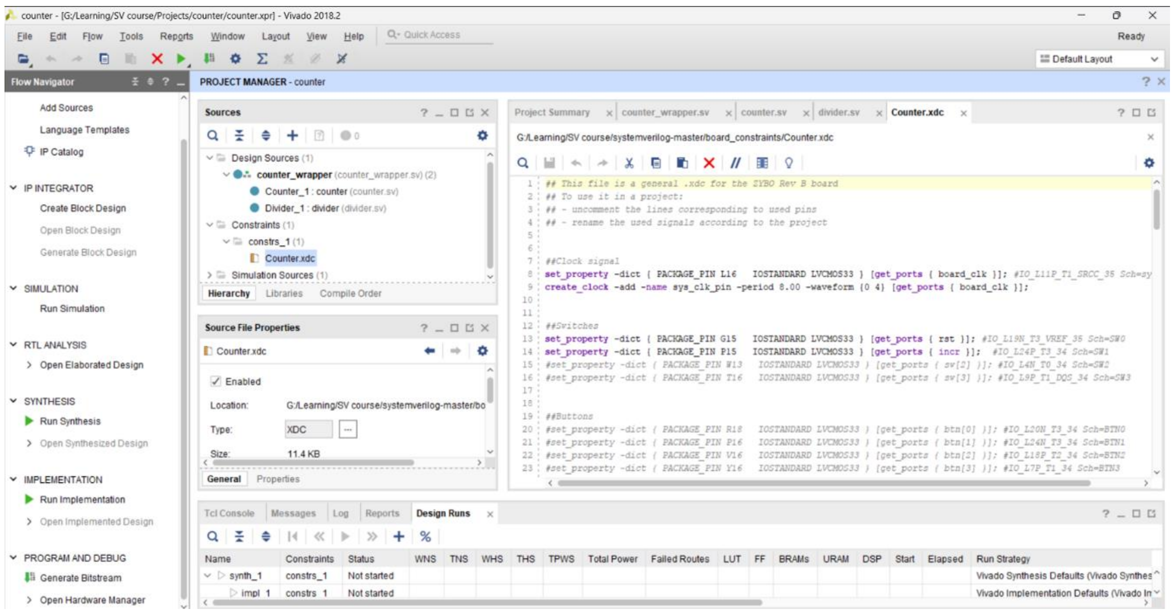
counter - [G:/Learning/SV course/Projects/counter/counter.xpr] - Vivado 2018.2

G:/Learning/SV course/systemverilog-master/rtl/counter_wrapper.sv

```
module counter_wrapper(
    input  logic board_clk, rst, incr,
    output logic [3:0] count_reg
);

    wire clk;

    counter #(.WIDTH(4)) Counter_1 (.*);
    divider #(.FREQ(1)) Divider_1 (.*); // FREQ set to 1 Hz

endmodule
```

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Impl

Modified the contrarian file



counter - [G:/Learning/SV course/Projects/counter/counter.xpr] - Vivado 2018.2

G:/Learning/SV course/systemverilog-master/board_constraints/Counter.xdc

```
## This file is a general .xdc for the ZYBO Rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project


##Clock signal
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { board_clk }]; #IO_L11P_T1_SRCC_35 Sch=sy
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { board_clk }];


##Switches
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { rst }]; #IO_L19N_T3_VREF_35 Sch=SW0
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { incr }];  #IO_L24P_T3_34 Sch=SW1
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=SW2
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3


##Buttons
#set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { btn[0] }]; #IO_L20N_T3_34 Sch=BTN0
#set_property -dict { PACKAGE_PIN P16   IOSTANDARD LVCMOS33 } [get_ports { btn[1] }]; #IO_L24N_T3_34 Sch=BTN1
#set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { btn[2] }]; #IO_L18P_T2_34 Sch=BTN2
#set_property -dict { PACKAGE_PIN Y16   IOSTANDARD LVCMOS33 } [get_ports { btn[3] }]; #IO_L7P_T1_34 Sch=BTN3
```

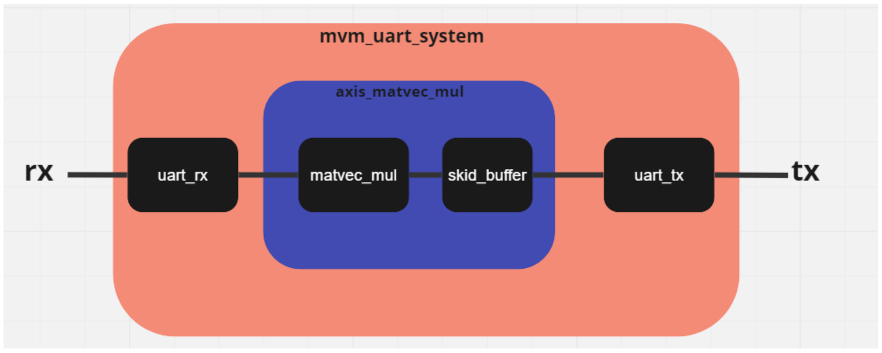
Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthes
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado In

Then the bit-stream was generated. The board was connected to the PC and the code was uploaded.

Working Video

3. UART Matrix Vector Multiplier Implementation

This was the main task of the practical session.



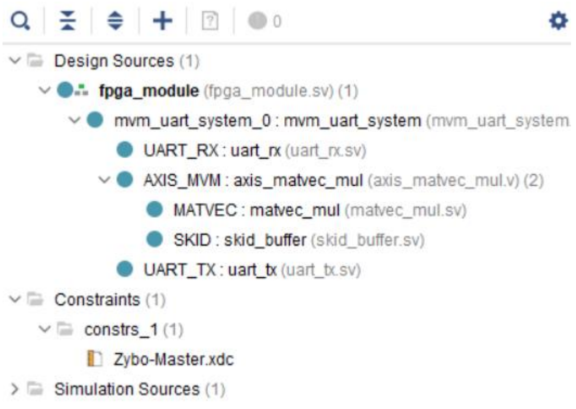
Mvm_uart_system module consists of several low-level modules as shown in the above figure. The functionalities of each low-level module can be explained as follows.

Uart_rx: converts the serial input to parallel data

Matvec_mul: performs the multiplication on input

Skid_buffer: handles the AXI stream signals

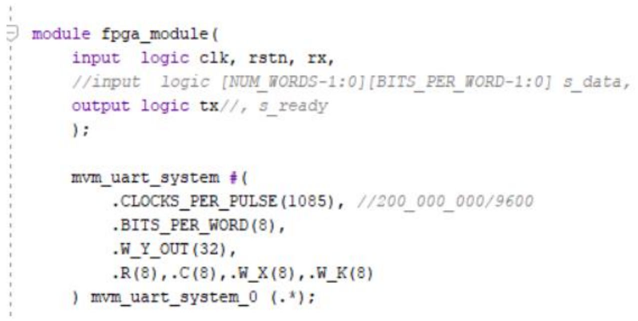
Uart_rx: translates the result to serial output



```
Design Sources (1)
  fpga_module (fpga_module.sv) (1)
    mvm_uart_system_0 : mvm_uart_system (mvm_uart_system
      UART_RX : uart_rx (uart_rx.sv)
      AXIS_MVM : axis_matvec_mul (axis_matvec_mul.v) (2)
        MATVEC : matvec_mul (matvec_mul.sv)
        SKID : skid_buffer (skid_buffer.sv)
      UART_TX : uart_tx (uart_tx.sv)
Constraints (1)
  constrs_1 (1)
    Zybo-Master.xdc
Simulation Sources (1)
```

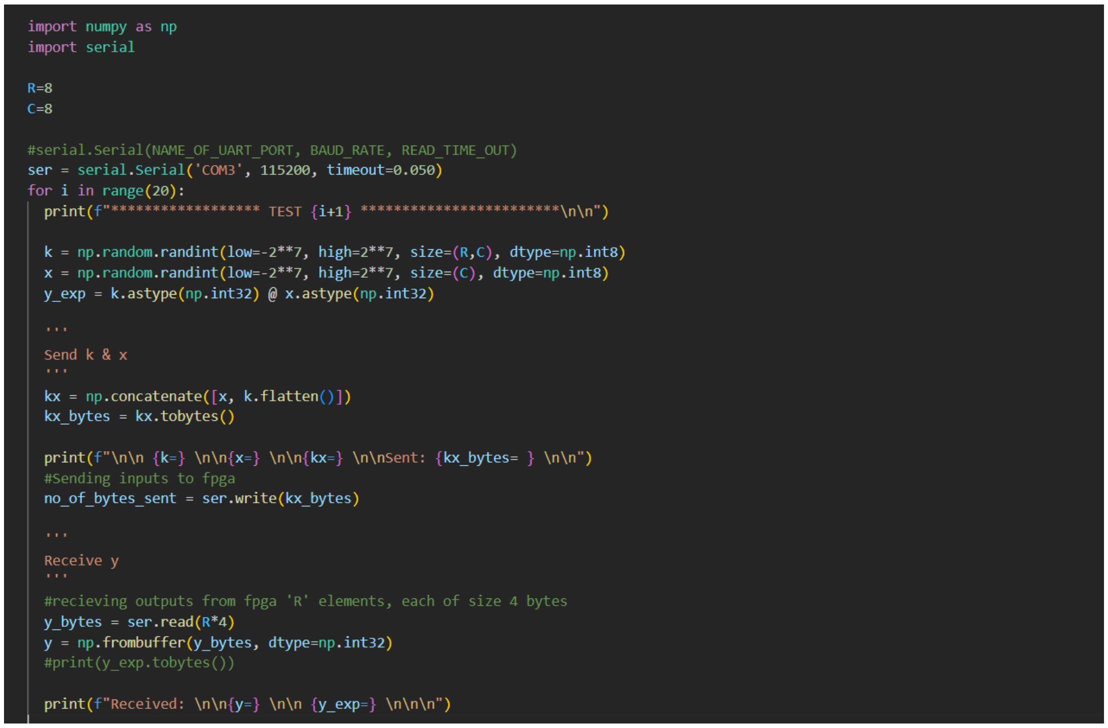
Parameters selected.

R = 8 , C =8 , W_X = 8 , W_K = 8



```
module fpga_module(
    input  logic clk, rstn, rx,
    //input  logic [NUM_WORDS-1:0][BITS_PER_WORD-1:0] s_data,
    output logic tx//, s_ready
    );

    mvm_uart_system #(
        .CLOCKS_PER_PULSE(1085), //200_000_000/9600
        .BITS_PER_WORD(8),
        .W_Y_OUT(32),
        .R(8),.C(8),.W_X(8),.W_K(8)
    ) mvm_uart_system_0 (.*);
```

Functionality of the python script



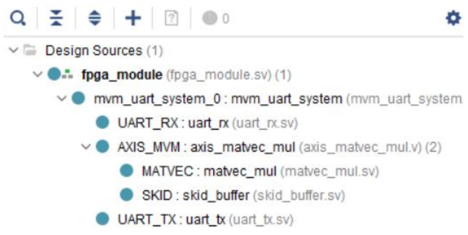
```python
import numpy as np
import serial

R=8
C=8

#serial.Serial(NAME_OF_UART_PORT, BAUD_RATE, READ_TIME_OUT)
ser = serial.Serial('COM3', 115200, timeout=0.050)
for i in range(20):
  print(f"***************** TEST {i+1} **********************\n\n")

  k = np.random.randint(low=-2**7, high=2**7, size=(R,C), dtype=np.int8)
  x = np.random.randint(low=-2**7, high=2**7, size=(C), dtype=np.int8)
  y_exp = k.astype(np.int32) @ x.astype(np.int32)

  '''
  Send k & x
  '''
  kx = np.concatenate([x, k.flatten()])
  kx_bytes = kx.tobytes()

  print(f"\n\n {k=} \n\n{x=} \n\n{kx=} \n\nSent: {kx_bytes= } \n\n")
  #Sending inputs to fpga
  no_of_bytes_sent = ser.write(kx_bytes)

  '''
  Receive y
  '''
  #recieving outputs from fpga 'R' elements, each of size 4 bytes
  y_bytes = ser.read(R*4)
  y = np.frombuffer(y_bytes, dtype=np.int32)
  #print(y_exp.tobytes())

  print(f"Received: \n\n{y=} \n\n {y_exp=} \n\n\n")
```

Python script was used to generate two matrices such as k and x and then send them to the FPGA board through a serial port. After connecting the FTDI converter to the PC, the name of the UART port was included in the script. The Python script also calculates the expected multiplied output and compares it with the received matrix. It runs 20 different test cases and gives the output.

Our output as a txt file

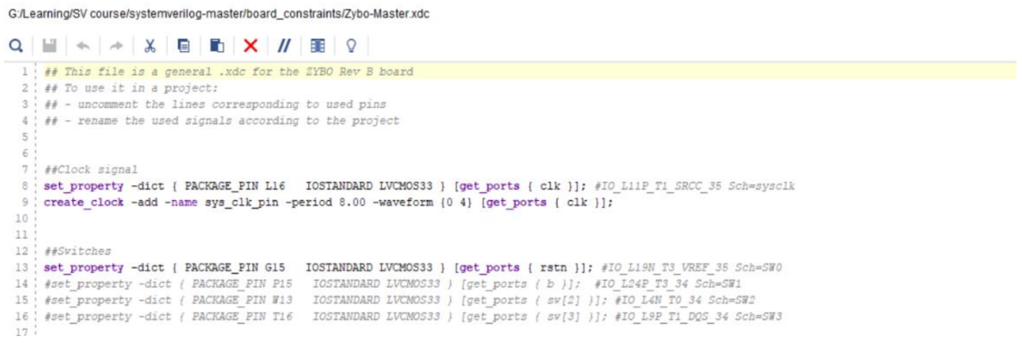
Process of programming the FPGA.

1. A new project was created and the zybo board was selected.
2. Added all the relevant source files and the FPGA module was selected as the top module.



```
Design Sources (1)
  fpga_module (fpga_module.sv) (1)
    mvm_uart_system_0 : mvm_uart_system (mvm_uart_system
      UART_RX : uart_rx (uart_rx.sv)
      AXIS_MVM : axis_matvec_mul (axis_matvec_mul.v) (2)
        MATVEC : matvec_mul (matvec_mul.sv)
        SKID : skid_buffer (skid_buffer.sv)
      UART_TX : uart_tx (uart_tx.sv)
```

3. The board constraint file was added and the modifications were done accordingly.



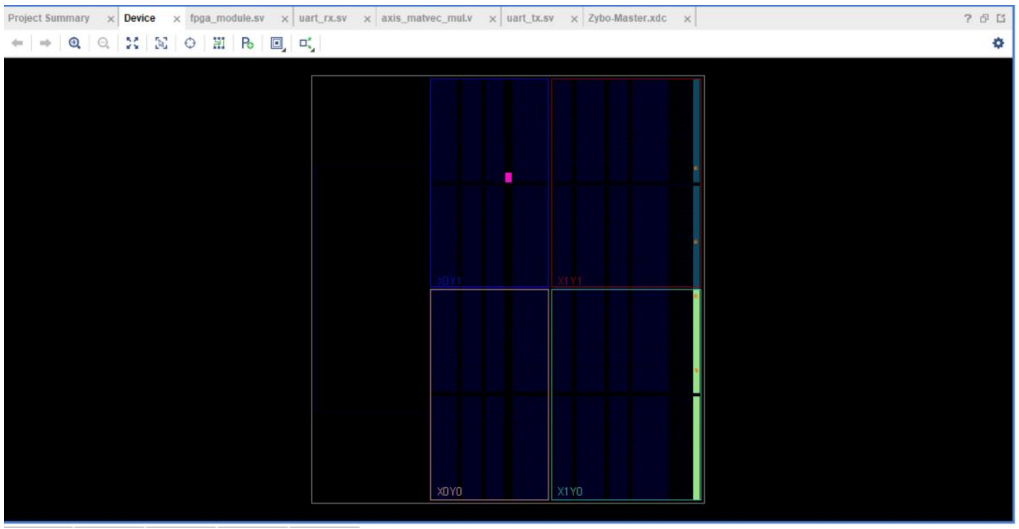
G:/Learning/SV course/systemverilog-master/board_constraints/Zybo-Master.xdc

```
## This file is a general .xdc for the ZYBO Rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project


##Clock signal
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L11P_T1_SRCC_35 Sch=sysclk
create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports { clk }];


##Switches
set_property -dict { PACKAGE_PIN G15   IOSTANDARD LVCMOS33 } [get_ports { rstn }]; #IO_L19N_T3_VREF_35 Sch=SW0
#set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { b }];  #IO_L24P_T3_34 Sch=SW1
#set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L4N_T0_34 Sch=SW2
#set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L9P_T1_DQS_34 Sch=SW3

```

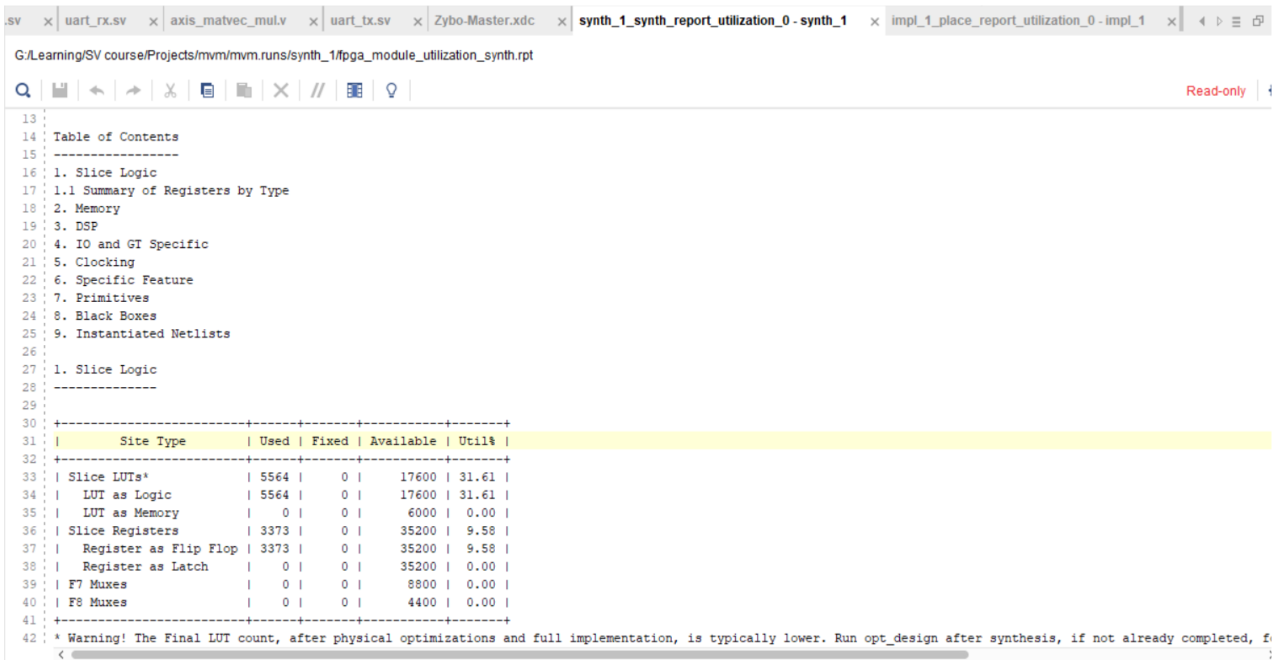
```
##Pmod Header JC
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { tx }]; #IO_L10P_T1_34 Sch=JC1_P
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports { jc_n[0] }]; #IO_L10N_T1_34 Sch=JC1_N
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { rx }]; #IO_L1P_T0_34 Sch=JC2_P
#set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { jc_n[1] }]; #IO_L1N_T0_34 Sch=JC2_N
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports { jc_p[2] }]; #IO_L8P_T1_34 Sch=JC3_P
#set_property -dict { PACKAGE_PIN Y14   IOSTANDARD LVCMOS33 } [get_ports { jc_n[2] }]; #IO_L8N_T1_34 Sch=JC3_N
#set_property -dict { PACKAGE_PIN T12   IOSTANDARD LVCMOS33 } [get_ports { jc_p[3] }]; #IO_L2P_T0_34 Sch=JC4_P
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { jc_n[3] }]; #IO_L2N_T0_34 Sch=JC4_N
```

4. Then the synthesis was run.

Synthesized design



Synthesis resource utilization report

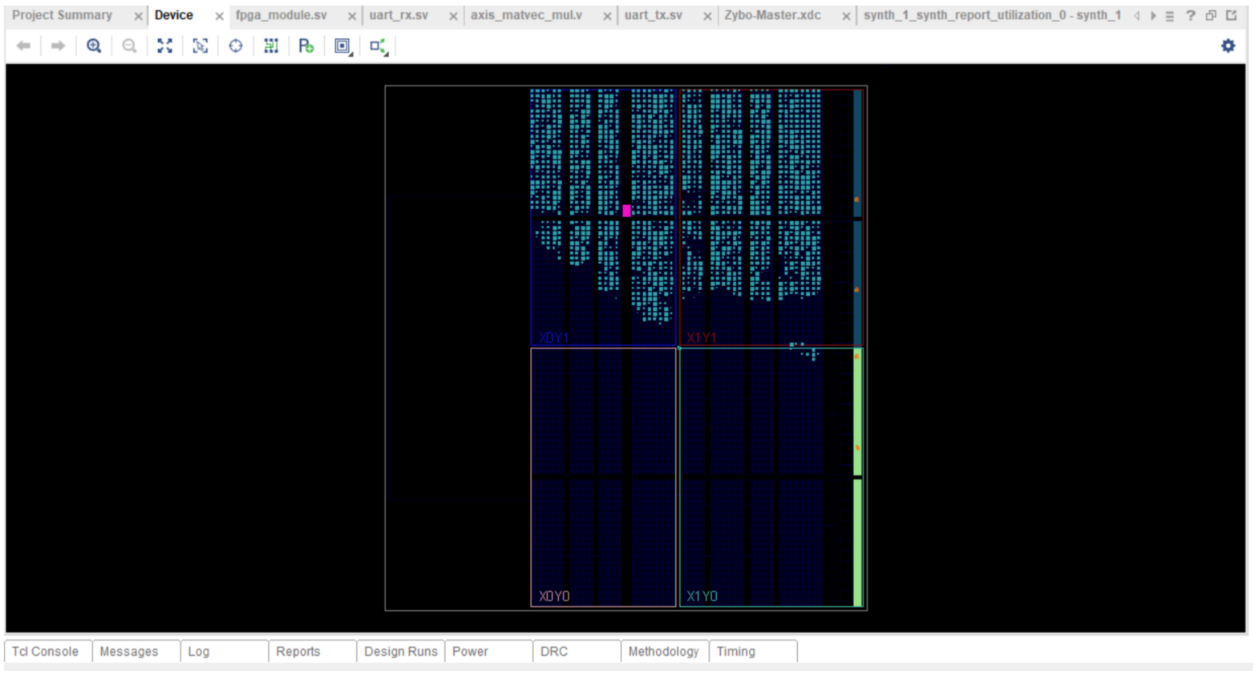

.sv
uart_rx.sv
axis_matvec_mul.v
uart_tx.sv
Zybo-Master.xdc
synth_1_synth_report_utilization_0 - synth_1
impl_1_place_report_utilization_0 - impl_1
Read-only

G:/Learning/SV course/Projects/mvm/mvm.runs/synth_1/fpga_module_utilization_synth.rpt

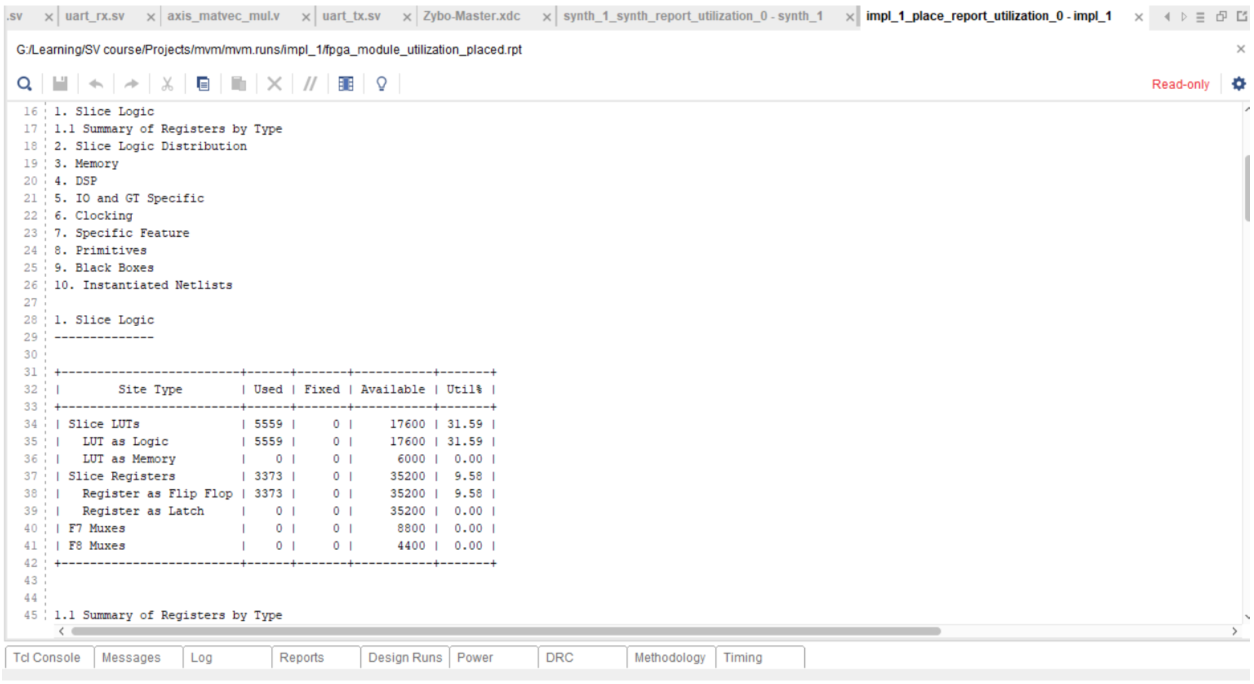
```

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             | 5564 |     0 |     17600 | 31.61 |
|   LUT as Logic          | 5564 |     0 |     17600 | 31.61 |
|   LUT as Memory         |    0 |     0 |      6000 |  0.00 |
| Slice Registers         | 3373 |     0 |     35200 |  9.58 |
|   Register as Flip Flop | 3373 |     0 |     35200 |  9.58 |
|   Register as Latch     |    0 |     0 |     35200 |  0.00 |
| F7 Muxes                |    0 |     0 |      8800 |  0.00 |
| F8 Muxes                |    0 |     0 |      4400 |  0.00 |
+-------------------------+------+-------+-----------+-------+
* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, f
```

5. Run implementation

Implemented design



Implementation resource utilization report

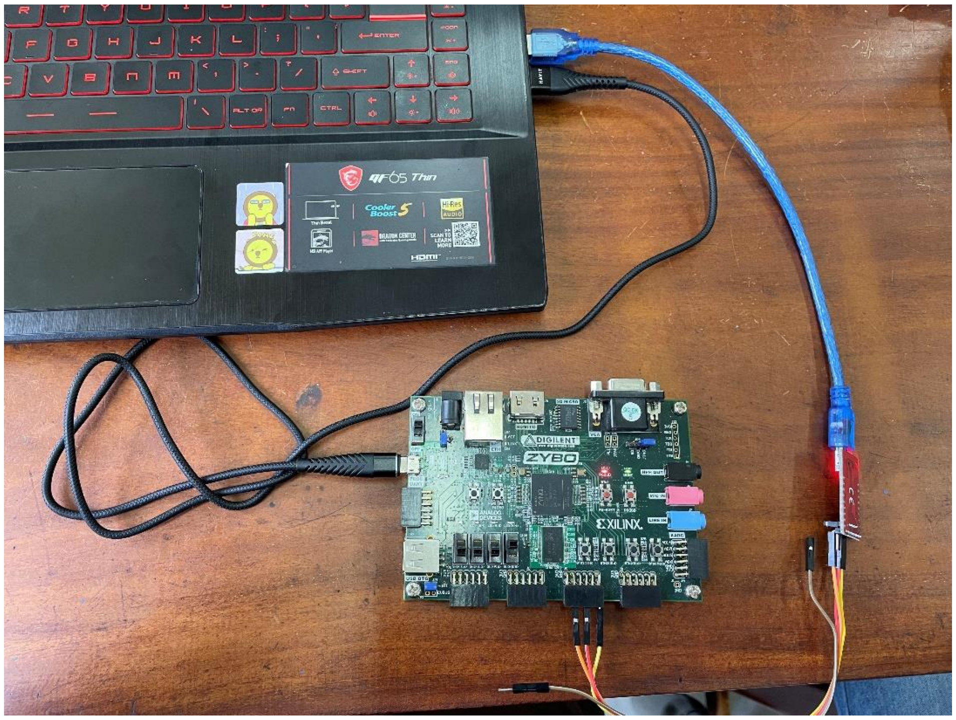


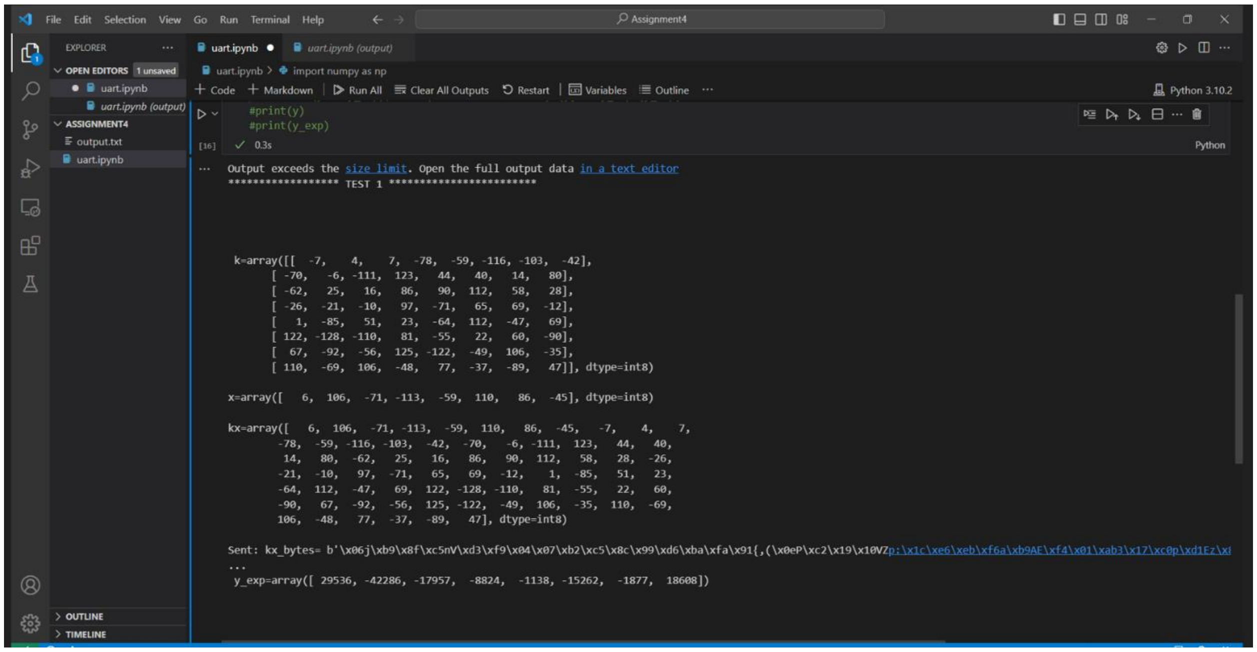
G:/Learning/SV course/Projects/mvm/mvm.runs/impl_1/fpga_module_utilization_placed.rpt

```
1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              | 5559 |     0 |     17600 | 31.59 |
|   LUT as Logic          | 5559 |     0 |     17600 | 31.59 |
|   LUT as Memory         |    0 |     0 |      6000 |  0.00 |
| Slice Registers         | 3373 |     0 |     35200 |  9.58 |
|   Register as Flip Flop | 3373 |     0 |     35200 |  9.58 |
|   Register as Latch     |    0 |     0 |     35200 |  0.00 |
| F7 Muxes                |    0 |     0 |      8800 |  0.00 |
| F8 Muxes                |    0 |     0 |      4400 |  0.00 |
+-------------------------+------+-------+-----------+-------+


1.1 Summary of Registers by Type
```

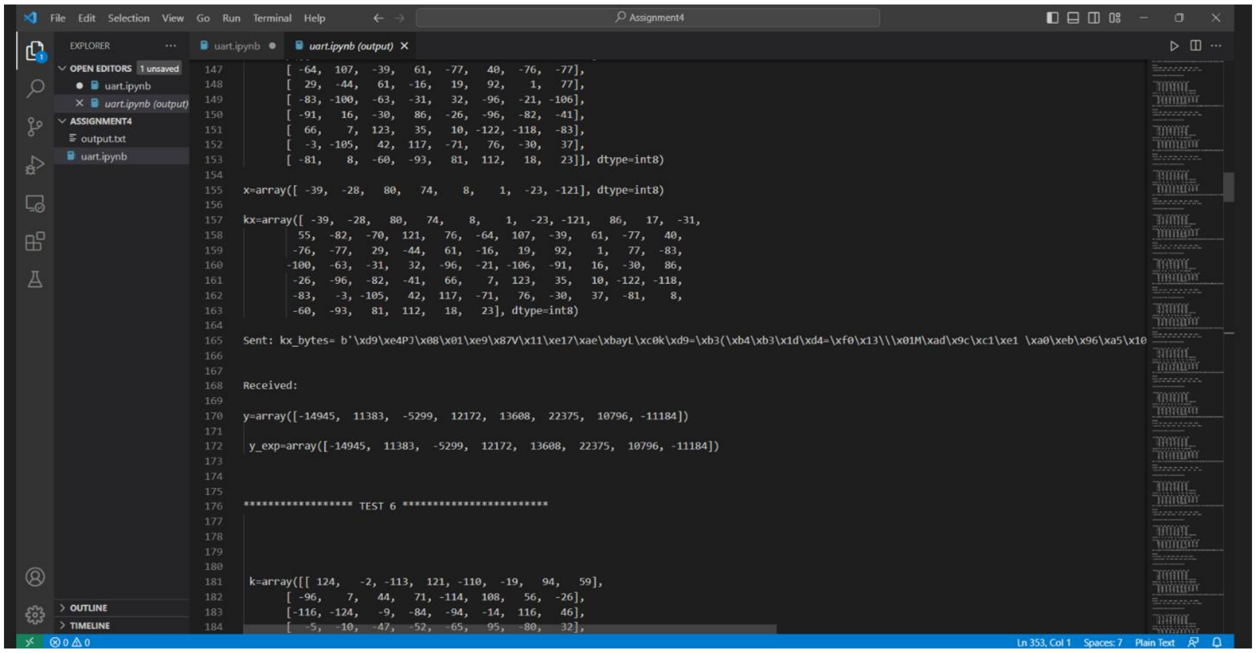
6. Bitstream generation
7. Connecting zybo board to a USB port using a FTDI converter.
8. Modifying the Python script
9. Turn on sw[0]
10. Power on zybo and send the cord to the FPGA using the hardware manager.
11. Run the python script.

Working the design




File Edit Selection View Go Run Terminal Help
Assignment4
EXPLORER
OPEN EDITORS 1 unsaved
uart.ipynb
uart.ipynb (output)
ASSIGNMENT4
output.txt
uart.ipynb
uart.ipynb > import numpy as np
+ Code + Markdown Run All Clear All Outputs Restart Variables Outline
Python 3.10.2
OUTLINE
TIMELINE

```
    #print(y)
    #print(y_exp)
[16]  ✓ 0.3s

Output exceeds the size limit. Open the full output data in a text editor
****************** TEST 1 ************************




 k=array([[  -7,    4,    7,  -78,  -59, -116, -103,  -42],
       [ -70,   -6, -111,  123,   44,   40,   14,   80],
       [ -62,   25,   16,   86,   90,  112,   58,   28],
       [ -26,  -21,  -10,   97,  -71,   65,   69,  -12],
       [   1,  -85,   51,   23,  -64,  112,  -47,   69],
       [ 122, -128, -110,   81,  -55,   22,   60,  -90],
       [  67,  -92,  -56,  125, -122,  -49,  106,  -35],
       [ 110,  -69,  106,  -48,   77,  -37,  -89,   47]], dtype=int8)

x=array([   6,  106,  -71, -113,  -59,  110,   86,  -45], dtype=int8)

kx=array([   6,  106,  -71, -113,  -59,  110,   86,  -45,   -7,    4,    7,
        -78,  -59, -116, -103,  -42,  -70,   -6, -111,  123,   44,   40,
         14,   80,  -62,   25,   16,   86,   90,  112,   58,   28,  -26,
        -21,  -10,   97,  -71,   65,   69,  -12,    1,  -85,   51,   23,
        -64,  112,  -47,   69,  122, -128, -110,   81,  -55,   22,   60,
        -90,   67,  -92,  -56,  125, -122,  -49,  106,  -35,  110,  -69,
        106,  -48,   77,  -37,  -89,   47], dtype=int8)

Sent: kx_bytes= b'\x06j\xb9\x8f\xc5nV\xd3\xf9\x04\x07\xb2\xc5\x8c\x99\xd6\xba\xfa\x91{,(\x0eP\xc2\x19\x10VZp:\x1c\xe6\xeb\xf6a\xb9AE\xf4\x01\xab3\x17\xc0p\xd1Ez\x
...
 y_exp=array([ 29536, -42286, -17957,  -8824,  -1138, -15262,  -1877,  18608])
```



```
      [ -64,  107,  -39,   61,  -77,   40,  -76,  -77],
      [  29,  -44,   61,  -16,   19,   92,    1,   77],
      [ -83, -100,  -63,  -31,   32,  -96,  -21, -106],
      [ -91,   16,  -30,   86,  -26,  -96,  -82,  -41],
      [  66,    7,  123,   35,   10, -122, -118,  -83],
      [  -3, -105,   42,  117,  -71,   76,  -30,   37],
      [ -81,    8,  -60,  -93,   81,  112,   18,   23]], dtype=int8)

x=array([ -39,  -28,   80,   74,    8,    1,  -23, -121], dtype=int8)

kx=array([ -39,  -28,   80,   74,    8,    1,  -23, -121,   86,   17,  -31,
         55,  -82,  -70,  121,   76,  -64,  107,  -39,   61,  -77,   40,
        -76,  -77,   29,  -44,   61,  -16,   19,   92,    1,   77,  -83,
       -100,  -63,  -31,   32,  -96,  -21, -106,  -91,   16,  -30,   86,
        -26,  -96,  -82,  -41,   66,    7,  123,   35,   10, -122, -118,
        -83,   -3, -105,   42,  117,  -71,   76,  -30,   37,  -81,    8,
        -60,  -93,   81,  112,   18,   23], dtype=int8)

Sent: kx_bytes= b'\xd9\xe4PJ\x08\x01\xe9\x87V\x11\xe17\xae\xbayL\xc0k\xd9=\xb3(\xb4\xb3\x1d\xd4=\xf0\x13\\\x01M\xad\x9c\xc1\xe1 \xa0\xeb\x96\xa5\x10


Received:

y=array([-14945,  11383,  -5299,  12172,  13608,  22375,  10796, -11184])

 y_exp=array([-14945,  11383,  -5299,  12172,  13608,  22375,  10796, -11184])



****************** TEST 6 ************************




 k=array([[ 124,   -2, -113,  121, -110,  -19,   94,   59],
       [ -96,    7,   44,   71, -114,  108,   56,  -26],
       [-116, -124,   -9,  -84,  -94,  -14,  116,   46],
       [  -5,  -10,  -47,  -52,  -65,   95,  -80,   32],
```

Full output as a txt file

Working Video (on other team member's PC)