# Assignment 4 - FPGA Implementation

Submission – R. Mewan Chandira

Functionality of the mvm_uart_system RTL and the python script

Mvm_uart_system

This system is a wrapper for serial communication modules and computational modules. For serial communication two modules uart_rx and uart_tx are designed where uart_rx convert input serial data to parallel data while uart_tx convert output parallel data to serial data for sending. The main computation block is a wrapper named axis_matvec_mul with two modules named matvec_mul and skid_buffer with a shifter. skid_buffer handles axi stream signals and matvec_mul handles the matrix multiplication, the intended operation of the design.

Python script

Since the communication is serial between computer and Zybo, the following code initatie the serial connection.

```
#serial.Serial(NAME_OF_UART_PORT, BAUD_RATE, READ_TIME_OUT)
ser = serial.Serial('COM3', 115200, timeout=0.050)
```

The script generate two matrices (k,x) with random integers of range ±128 such that they can undergo vector multiplication. Each element in these arrays are written in 1 byte. The result of the vector multiplication is stored in another array (y_exp) with 4 bytes elements.

The arrays k and x are then prepared to sent through a serial connection by flattening and concatenated to a single 1 dimensional array.

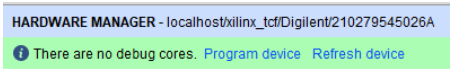
```
kx = np.concatenate([x, k.flatten()])
kx_bytes = kx.tobytes()
```

Next the script monitors the receiving connection and extract the resultant matrix as an array (y). Then the script compares the received array and the computed result and display an error message if any mismatches present.

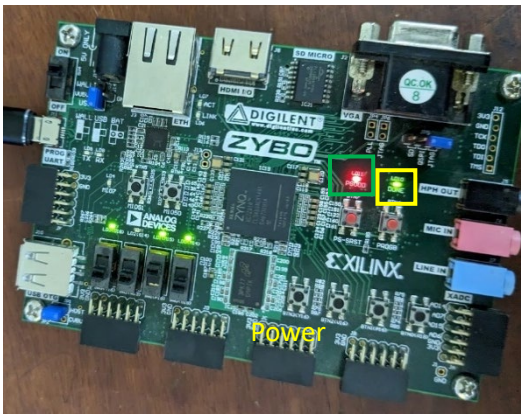
```
assert (y == y_exp).all(), f"Output doesn't match: y:{y} != y_exp:{y_exp}"
```

Process of programming the FPGA (using Vivado)

1. In the creation of a new RTL project after adding the required source files select the FPGA board for implementation (Zybo in the session conducted) as the default part and create the project.
2. Add the constraints file to map the properties of the board according to the RTL design. Set the top module appropriately from the sources tab, in case of the design containing several source files. Next from generate bit stream from Program and Debug section.
3. Upon successful completion, a separate tab named Open Hardware manager will appear. Now connect the FPGA board to the computer. A red colour bulb will indicate the power on state of the board. Once the device is recognized the following message will appear.



4. Click on the program device button to program the device. Once it is successfully completed a green colour bulb will light up as follows.




## Utilization reports of synthesis and implementation

### Resources

Synthesis

| Name | Slice LUTs (17600) | Slice Registers (35200) | Bonded IOB (100) | BUFGCTRL (32) |
|---|---|---|---|---|
| fpga_module | 5399 | 3373 | 4 | 1 |
| mvm_uart_syste | 5399 | 3373 | 0 | 0 |

Implementation

| Name | Slice LUTs (17600) | Slice Registers (35200) | Slice (4400) | LUT as Logic (17600) | Bonded IOB (100) | BUFGCTRL (32) |
|---|---|---|---|---|---|---|
| fpga_module | 5394 | 3373 | 1691 | 5394 | 4 | 1 |
| mvm_uart_sys | 5394 | 3373 | 1691 | 5394 | 0 | 0 |

### Timing

Synthesis

| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 2.279 ns | Worst Hold Slack (WHS): | 0.139 ns | Worst Pulse Width Slack (WPWS): | 3.500 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: | 6737 | Total Number of Endpoints: | 6737 | Total Number of Endpoints: | 3374 |

**All user specified timing constraints are met.**

Implementation

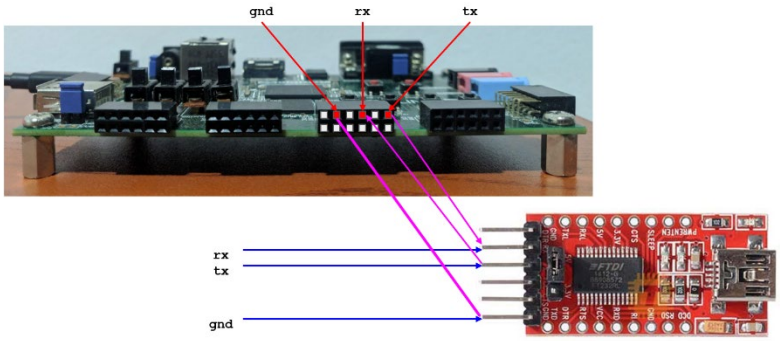
| Setup | | Hold | | Pulse Width | |
|---|---|---|---|---|---|
| Worst Negative Slack (WNS): | 0.247 ns | Worst Hold Slack (WHS): | 0.036 ns | Worst Pulse Width Slack (WPWS): | 3.500 ns |
| Total Negative Slack (TNS): | 0.000 ns | Total Hold Slack (THS): | 0.000 ns | Total Pulse Width Negative Slack (TPWS): | 0.000 ns |
| Number of Failing Endpoints: | 0 | Number of Failing Endpoints: | 0 | Number of Failing Endpoints: | 0 |
| Total Number of Endpoints: | 6737 | Total Number of Endpoints: | 6737 | Total Number of Endpoints: | 3374 |

**All user specified timing constraints are met.**

Matrix vector multiplication

Video of performing a matrix multiplication could not be captured due to several issues in troubleshooting. The following is the complete process in performing the matrix multiplication.

USB-tty is used to communicate between FPGA and the computer using serial communication. First connect USB-tty to the FPGA board using JC Pmod Ports as shown below.




Now connect the USB-tty to computer and get the USB serial port of connection by device manager. Rename the python script with the port identification as follows.

```
#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} **********************
```

Now program the FPGA and reset the circuit with sw[0] on the board. Connect the USB-tty and run the python script. Upon successful completion the results will be as follows.

```
(base) C:\Users\mcrms>python mvm_uart_system.py
****************** TEST 1 ************************



 k=array([[-123,   90, -109,  -88,   80,  117,   -8,   -3],
       [  -2,  -84,   16,   68,   20,  -68,  114,  -31],
       [  89,  105,  -35,   48,  -15, -114,   55,  -25],
       [ -23,  123,  109,  -95,  -58,  -25,  -53,   36],
       [  42,  -88, -127,   63,  121,   27,  100,  -68],
       [ -15,  -47,   60,   79,   96, -121,  121,   88],
       [  -8,   33,  -87,  -34,  -59,  -90,  116,  -38],
       [ 109,   52,   55,   43, -117,  -16,   80, -119]], dtype=int8)

x=array([ -37,   53,  -38,  -96, -123,  -82, -110,  100], dtype=int8)

kx=array([ -37,   53,  -38,  -96, -123,  -82, -110,  100, -123,   90, -109,
       -88,   80,  117,   -8,   -3,   -2,  -84,   16,   68,   20,  -68,
       114,  -31,   89,  105,  -35,   48,  -15, -114,   55,  -25,  -23,
       123,  109,  -95,  -58,  -25,  -53,   36,   42,  -88, -127,   63,
       121,   27,  100,  -68,  -15,  -47,   60,   79,   96, -121,  121,
        88,   -8,   33,  -87,  -34,  -59,  -90,  116,  -38,  109,   52,
        55,   43, -117,  -16,   80, -119], dtype=int8)

Sent: kx_bytes= b'\xdb5\xda\xa8\x85\xae\x92d\x85Z\x93\xa8Pu\xf8\xfd\xfe\xac\x10D\x14\xbcr\xe1Yi\xdd0\xf1\x8e7\xe7\xe9{m\xa1\xc6\xe7\xcb$*\xa8\x81?y\x1bd\xbc\xf1\xd1<O`\x87yX\xf8!\xa9\xde\xc5\xa6t\xdam47+\x8b\xf0P\x89'


Received:

y=array([], dtype=int32)

 y_exp=array([  3057, -24038,   1637,  30962, -42337, -18196,   6692, -12492])
```

```
****************** TEST 8 ************************



 k=array([[ 114,   60,  -80,   85,  -10,   85,   -3,   -2],
       [ -44, -109,   53,  -64,  122,   13,   23,  -94],
       [ -31,   28,   49,   -2, -126,   -8,  -68,  -28],
       [ -18,  115,  -33,   33,  -94, -104,  -65,  114],
       [ -83,   82,   77,  -52,  -48,  -19,  -28,  112],
       [ -43,   48,  -91,  -11,  -90,   94,  -34,  -93],
       [  40,  -66, -128,   86,   86,  122,   74,   26],
       [  97,    7,   -1,  -34,  -46,  -13,  120,  -71]], dtype=int8)

x=array([ -41, -115,  -63,   85,   38, -116,   29,  -75], dtype=int8)

kx=array([ -41, -115,  -63,   85,   38, -116,   29,  -75,  114,   60,  -80,
        85,  -10,   85,   -3,   -2,  -44, -109,   53,  -64,  122,   13,
        23,  -94,  -31,   28,   49,   -2, -126,   -8,  -68,  -28,  -18,
       115,  -33,   33,  -94, -104,  -65,  114,  -83,   82,   77,  -52,
       -48,  -19,  -28,  112,  -43,   48,  -91,  -11,  -90,   94,  -34,
       -93,   40,  -66, -128,   86,   86,  122,   74,   26,   97,    7,
        -1,  -34,  -46,  -13,  120,  -71], dtype=int8)

Sent: kx_bytes= b'\xd7\x8d\xc1U&\x8c\x1d\xb5r<\xb0U\xf6U\xfd\xfe\xd4\x935\xc0z\r\x17\xa2\xe1\x1c1\xfe\x82\xf8\xbc\xe4\xees\xdf!\xa2\x98\xbfr\xadRM\xcc\xd0\xed\xe4p\xd50\xa5\xf5\xa6^\xde\xa3(\xbe\x80VVzJ\x1aa\x07\xff\xde\xd2\xf3x\xb9'


Received:

y=array([ -9486,  16405,  -8938,  -9546, -24130,  -7294,  10636,    956])

 y_exp=array([ -9486,  16405,  -8938,  -9546, -24130,  -7294,  10636,    956])
```

```
****************** TEST 12 ************************



 k=array([[-110,  -12,   79,  -66,  -75, -120,   87,   59],
       [-103,  -24,  -88,  -77,   22,  -71,  -82,  105],
       [  41,  -15, -114,  -53,   11, -110,    2,  -30],
       [  54,  -55,   17,   83,  -31,  124,   69, -126],
       [ -37,  118,   68,   31,   30,   55,  -48, -121],
       [ 124,   75, -112,   53,  -68,   57,   16,   99],
       [ -75,  -95, -117, -108,    4,  112,  120,    1],
       [ -67,   40,  122,  -30,  -25,  -32,  -53,  111]], dtype=int8)

x=array([  31,  108,  -10,   53, -113,  -72,  -66, -100], dtype=int8)

kx=array([  31,  108,  -10,   53, -113,  -72,  -66, -100, -110,  -12,   79,
       -66,  -75, -120,   87,   59, -103,  -24,  -88,  -77,   22,  -71,
       -82,  105,   41,  -15, -114,  -53,   11, -110,    2,  -30,   54,
       -55,   17,   83,  -31,  124,   69, -126,  -37,  118,   68,   31,
        30,   55,  -48, -121,  124,   75, -112,   53,  -68,   57,   16,
        99,  -75,  -95, -117, -108,    4,  112,  120,    1,  -67,   40,
       122,  -30,  -25,  -32,  -53,  111], dtype=int8)

Sent: kx_bytes= b'\x1fl\xf65\x8f\xb8\xbe\x9c\x92\xf4O\xbe\xb5\x88W;\x99\xe8\xa8\xb3\x16\xb9\xaei)\xf1\x8e\xcb\x0b\x92\x02\xe26\xc9\x11S\xe1|E\x82\xdbvD\x1f\x1e7\xd0\x87|K\x905\xbc9\x10c\xb5\xa1\x8b\x94\x04px\x01\xbd(z\xe2\xe7\xe0\xcbo'


Received:

y=array([ -3521, -11448,   7527,   2584,  20478,   8497, -33675,  -3040])

 y_exp=array([ -3521, -11448,   7527,   2584,  20478,   8497, -33675,  -3040])
```

```
****************** TEST 20 ************************



 k=array([[  90, -127,  112,   87,  -10,  -42,   29,  -19],
       [ -57,  122,  -25,   32,  -51,   96,  -85,  112],
       [  76,   37,   33, -116,  -85, -119,   99,  -74],
       [-110,  -65, -124,  -66,  125,  -37, -112,   19],
       [ 122, -104,   88,   59,  -47,  104,   17,  -48],
       [ 102,  117,   93,  117,  106,   93,   69,  -94],
       [  41,  -15,   97,   36,  -66, -116,   82,  -94],
       [ -65, -114,   99,    9,  -23,   57,   61,  -22]], dtype=int8)

x=array([ 18,  37,  99,  38,  75, -18,   0, -13], dtype=int8)

kx=array([  18,   37,   99,   38,   75,  -18,    0,  -13,   90, -127,  112,
        87,  -10,  -42,   29,  -19,  -57,  122,  -25,   32,  -51,   96,
       -85,  112,   76,   37,   33, -116,  -85, -119,   99,  -74, -110,
       -65, -124,  -66,  125,  -37, -112,   19,  122, -104,   88,   59,
       -47,  104,   17,  -48,  102,  117,   93,  117,  106,   93,   69,
       -94,   41,  -15,   97,   36,  -66, -116,   82,  -94,  -65, -114,
        99,    9,  -23,   57,   61,  -22], dtype=int8)

Sent: kx_bytes= b'\x12%c&K\xee\x00\xf3Z\x81pW\xf6\xd6\x1d\xed\xc7z\xe7 \xcd`\xabpL%!\x8c\xab\x89c\xb6\x92\xbf\x84U\xbe}\xdb\x90\x13z\x98X;\xd1h\x11\xd8fu]uj]E\xa2)\xf1a$\xbe\x8cR\xa2\xbf\x8ec\t\xe99=\xea'


Received:

y=array([11568, -4780, -1675, -9375,  4425, 27316,  9514,  2290])

 y_exp=array([11568, -4780, -1675, -9375,  4425, 27316,  9514,  2290])



(base) C:\Users\mcrms>
```