How Lab 9 works?

Main.c

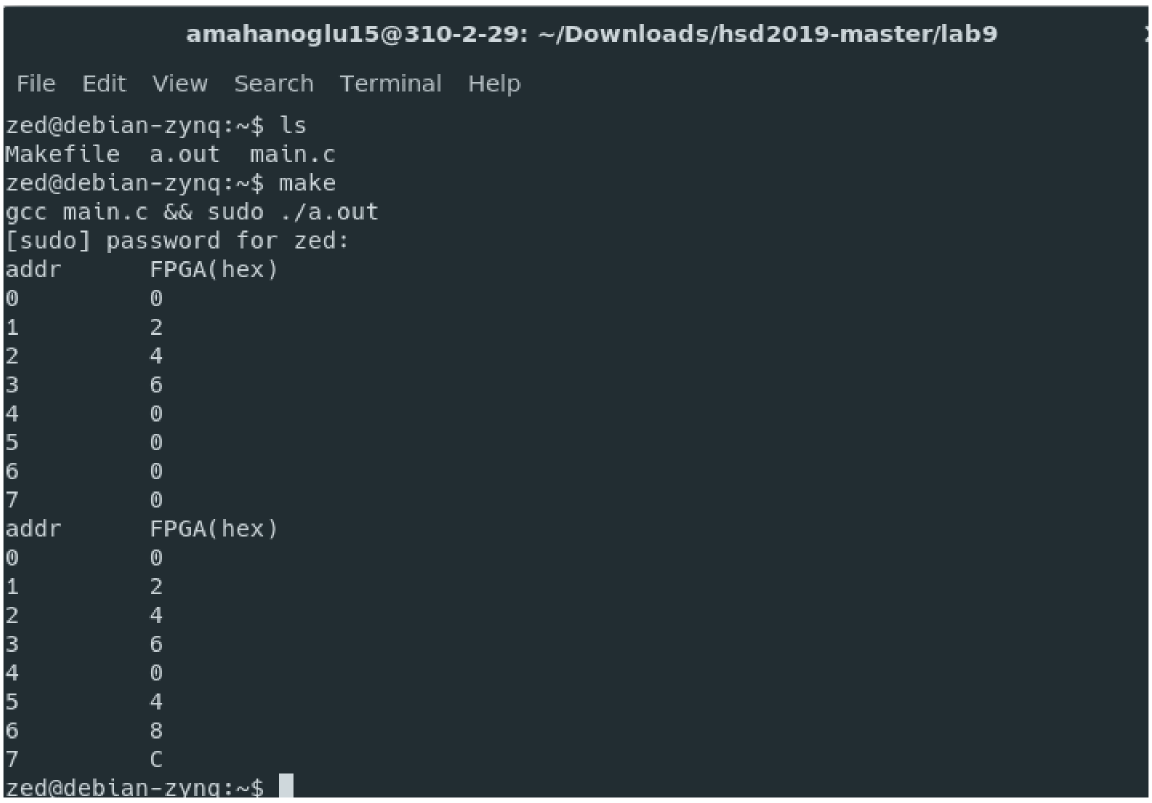
In the main.c code variable foo is used in order to access the contents of the file which is found at /dev/mem.

The key point in the main.c is the mapping of physical adress to a virtual adress which are called *fpga_bram and *fpga_ip. Thanks to the mmap function we can access (read and write) the contents of whatever file is found in /dev/mem.

I came to conclusion that lines between 17-25 are just written for testing purposes. It initializes a value to the adresses starting from `0x40000000`.

Lines 28 and 29 are interesting. Line 28 sets the value which is found in fpga_ip to `0x5555`, and in line 29 the processor waits until the value in fpga_ip is not `0x5555` anymore. (This was the first time I saw a while loop with only a conditional statement). After the value in fpga_ip is changed (due to the output of myip) the contents of fpga_bram is printed out.

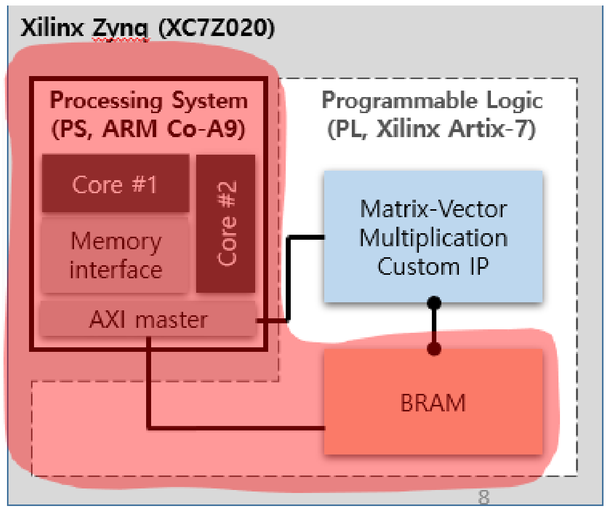
It must be noticed that the value found at adress fpga_ip (which is `0x43C00000` physically) is kind of a signal variable which lets main.c code to understand that the process in other layer (myip) is complete. In the result it can be seen that the contents of fpga_bram is also changed from 0,2,4,6,0,0,0,0 to 0,2,4,6,0,4,8,C


amahanoglu15@310-2-29: ~/Downloads/hsd2019-master/lab9
File Edit View Search Terminal Help

```
zed@debian-zynq:~$ ls
Makefile  a.out  main.c
zed@debian-zynq:~$ make
gcc main.c && sudo ./a.out
[sudo] password for zed:
addr      FPGA(hex)
0         0
1         2
2         4
3         6
4         0
5         0
6         0
7         0
addr      FPGA(hex)
0         0
1         2
2         4
3         6
4         0
5         4
6         8
7         C
zed@debian-zynq:~$
```

myip_v1_0_S00_AXI

In the myip code the value addressed by fpga_ip corresponds to slv_reg0 which is one of the slave registers initialized in myip. First time this register is changed is the line 232-239 where the for loop writes 8 bits each time from S_AXI_WDATA, corresponding to total of 32 bits, knowing the for loop iterates for 4 times. The content of the other registers are also changed in same manner.


Xilinx Zynq (XC7Z020)
Processing System (PS, ARM Co-A9)
Core #1
Core #2
Memory interface
AXI master
Programmable Logic (PL, Xilinx Artix-7)
Matrix-Vector Multiplication Custom IP
BRAM
8

In a general sense myip code generates the block “Processing System” including AXI bus, and “BRAM” block as it is seen in the figure above. A finite state machine of 4 states is implemented between the lines 415-436 where it controls the behaviour of BRAM. Instead of the block “matrix-vector multiplication custom IP, there is a line of code which equates data to be written to 1 bit shifted version of data read in the registers:

```
    assign BRAM WRDATA = BRAM RDDATA << 1;
```

What FSM does is that it first reads the values in the registers – which are 32 bit – and then shifts each 8 bits of data found in slv_reg which are 0, 2, 4, 6 which makes 0, 4, 8, C which is the result we observe.

Questions

(Where it is specified that the output of myip will be written in /dev/mem ?)

(I just noticed that this file has no extension. Most of the “open()” statements I saw, were used with .txt. However, it seems to me that the file located in /dev/mem has no extension?)