Participants:

Jacob Munkholm Hansen, 20140479

Jens Jakob Mikkelsen, 201506215

Yang ‘Asian spy’ Hanshou, 201902791

**Embedded Real Time Systems – Assignment 2**

For the exercises, the following design will be used:

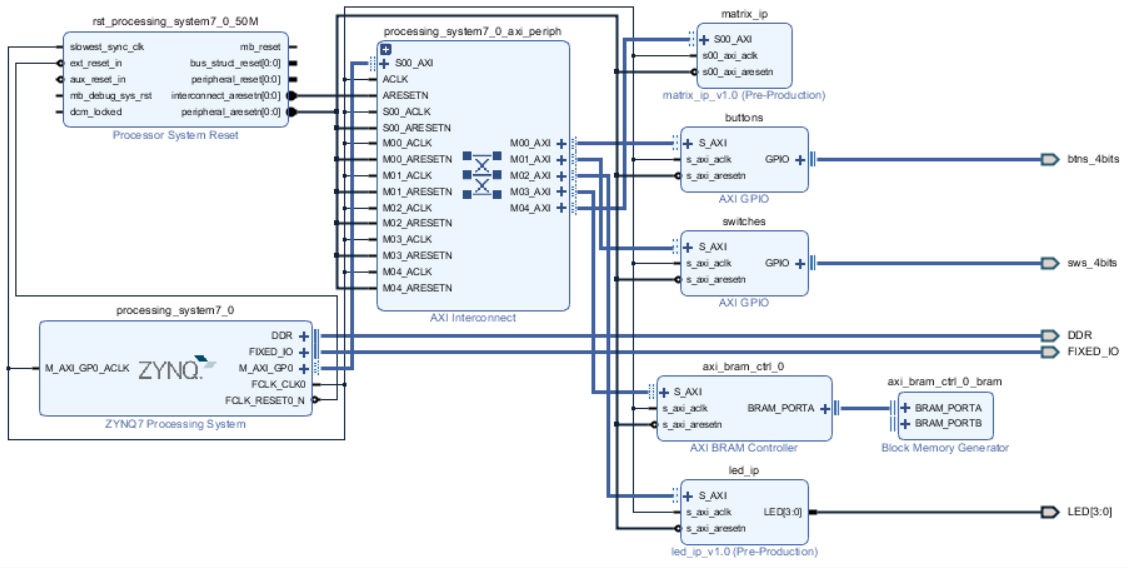


Figure 1 - System building blocks

The buttons and switches will be used in exercise 3 and 4, and the matrix\_ip will be used in exercise 5. The BRAM and led\_ip persists in the design created in the pre-exercises.

3) Create a Console Application

The main-program looks like this:

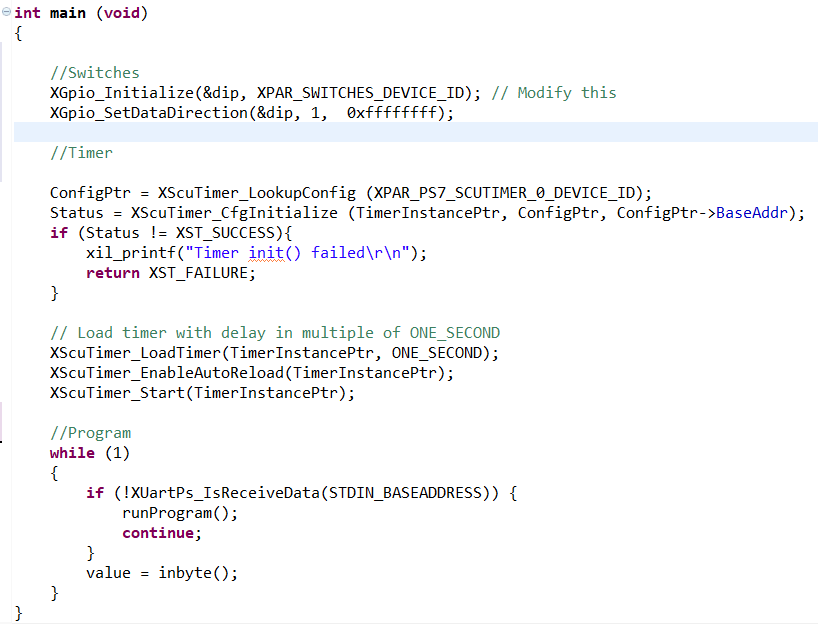


Figure - main program

The main function can be split into two parts; initialization of hardware components (Led’s and timer) and an always-running loop. Every iteration of the loop checks whether data has been received from the uart. If a new value is received, the ‘value’ will be set to it, allowing for the user to switch between the different parts of the program. This functionality is created in the runProgram() function:

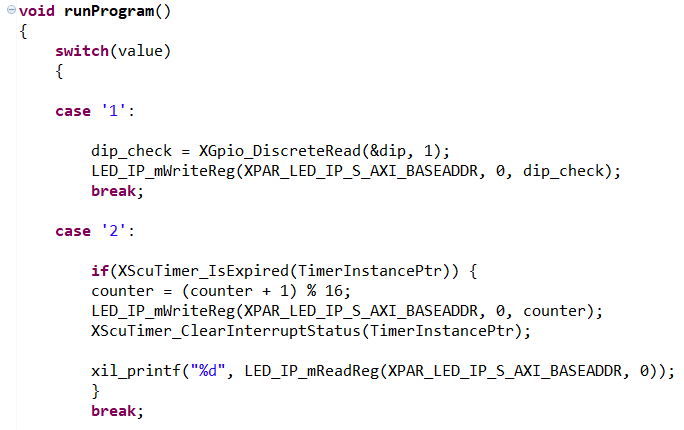


Figure 3 - runProgram() function

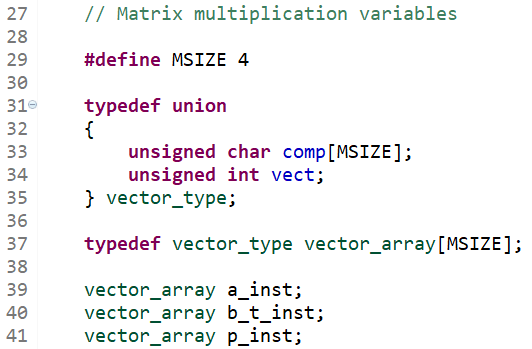
So every time the user writes from the UART, the program will check whether the value corresponds to the case values, and switch between them.

If the value is 1, the switches will be read, and the LED’s will be written to based on what switches are turned on/off.

If the value is 2, every time the timer expires the counter will increase up to 15 and then reset. The LED registers will then be written to, based on the value of the counter. As seen in figure 1, the timer is initialized with a value of one second (325000000 cycles), so it counts every second.

# 4) Create a Matrix Multiplication

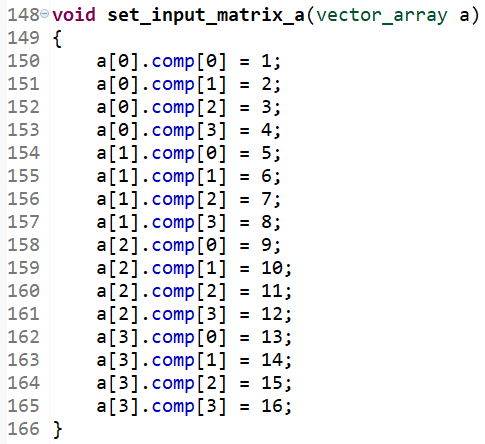
The matrix variables are defined and declared as seen below. The row/column size is restricted to be 4, and the matrices a\_inst, b\_t\_inst, and p\_inst are thus defined two-dimentional arrays of size 4x4:



Figur 3 - Matrix multiplication variables

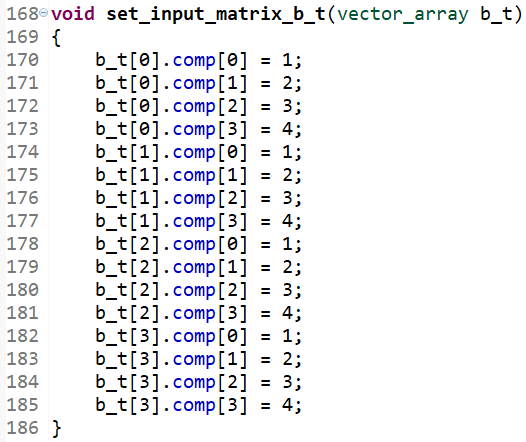
“set\_input\_matrices(…)” has been split into two separate functions “set\_input\_matrix\_a(…)” and “set\_input\_matrix\_b\_t(…)”.

Matrix a is set so that it reflects the matrix in the guidelines:



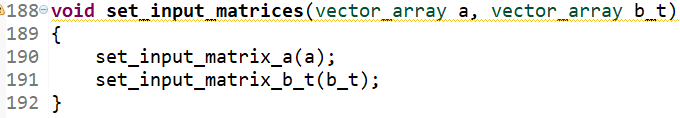
Figur 4 - set\_input\_matrix\_a(...) function

The same is done for matrix b\_t. Notice, however, that since b\_t is b transposed, the rows from the matrix in the guidelines corresponds to the columns of b\_t, so that all values of comp[0] are set to 1, comp[1] to 2, etc.:



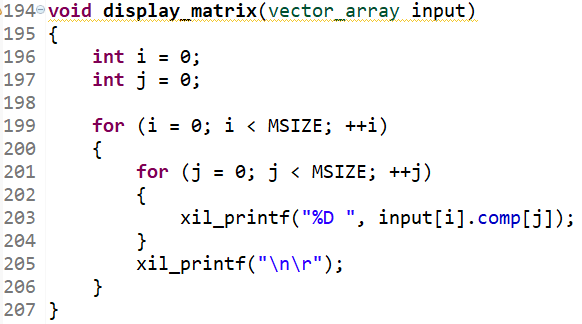
Figur 5 - set\_input\_matrix\_b\_t(...) function

The two functions are then called inside “set\_input\_matrices(…)”:



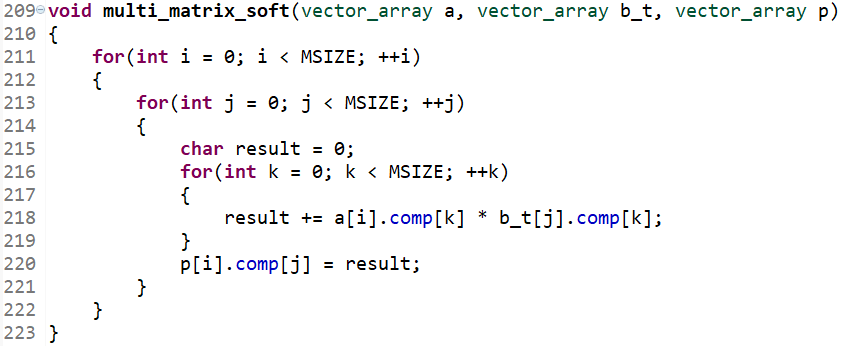
Figur 6 - set\_input\_matrices(...) function

To display a matrix, the function “display\_matrix(…)” has been implemented to loop through each row “i” and each row “j” and printing the value of each row-column element to the UART by calling “xil\_printf(…)”:



Figur 7 - display\_matrix(...) function

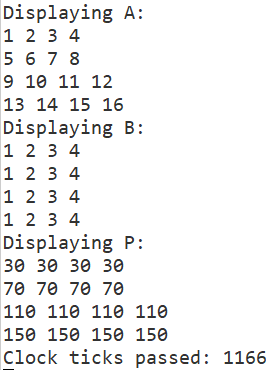
“multi\_matrix\_soft(…)” is implemented with two nested for-loops. “i” is the row number of matrices a and p; “j” is the row number of matrix b\_t and the column number of matrix p; “k” is the column number of matrices a and b\_t. The elements of matrix p are calculated by accumulating the sums of the row elements of matrices a and b\_t. This is because b\_t is matrix b transposed, otherwise it would be the column elements of matrix b that would be incorporated as a factor.



Figur 8 - multi\_matrix\_soft(...) function

The result of setting and displaying the matrices a and b as well as calculating the matrix b can be seen below. The clock ticks of calculating p are likewise displayed.

Please note that “B” below should be “B transposed” or something equivalent.

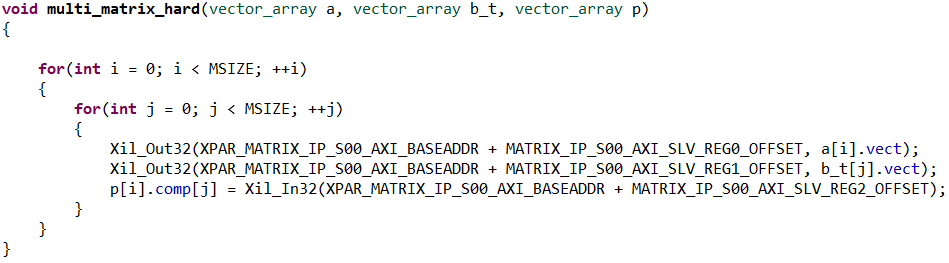


Figur 9 - UART output

# 5) Create a Hardware IP Acceleration

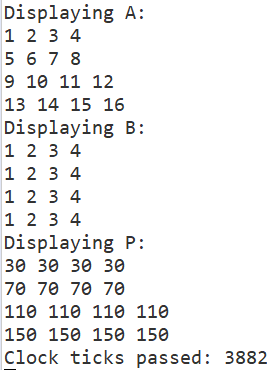
As seen in figure 1, the matrix\_ip is added to the system, allowing the use of the different IP core. As mentioned in the assignment paper, the PL is clocking at 100 MHz, while the PS runs at 650 MHz, so it is expected to be slower.

The code implementing the multiplication looks like this:



Whereas before each element in the matrix was read and used, now the new IP core allows a simpler implementation that just picks out each column for each row (first two lines in inner loop) and stores the result in the matrix p. The same functions as before is used to showing it.

The following is the result in the terminal:



So the same result as before, except now it is 3882/1166 = ~ 3.3 times slower.

# 7) HLS exercise with SystemC