ES3F1 Lab 1

Lab 1 is an introduction to Xilinx's Zynq SoC Architecture using the Digilent Zybo Z7-20 series FPGA boards. We'll walk through setting up the processing system in hardware, exporting this from Vivado to the Xilinx SDK environment and writing some software in C to run on one of the ARM cores.

Objectives

Successfully export a basic Processing System layer in hardware to the SDK in order to write a C program that computes the multiplication of two square matrices and returns the solution + the time taken for program execution.

Follow the set up example in the rtl instructions, moving onto the src instructions when you've successfully exported the hardware bitstream.

To get started with a project in the Xilinx SDK, you may use the template.c as a starting point.

1. Create an instance of a timer at the entry point of your program, using the "xtime_l.h" library. We'll use this later to time our multiplication function. The example below shows you how to set up an instance of the timer and how to use the `XTime_GetTime()` function.

```
#include "xtime_l.h"
#include "xil_printf.h"

void example(){

   XTime tVariable;

   XTime_GetTime(&tVariable);

   xil_printf("%d", tVariable);

}
```

- Hint: Think about how to measure a time interval. You will need to take a reading before and after the event, to find the difference.

2. Create a C function that takes a int pointer to a pointer and initialises each of the elements in the passed 2D array (**which represents a matrix**) with a random integer using the `rand()` function. Make sure to allocate memory for each 2D array (both rows and columns!). Don't forget to seed your `rand()` function with `srand(YOUR_SEED_VALUE)` to ensure that you don't get the same results each time the program runs.

- Hint: Ideally you want YOUR_SEED_VALUE to be a different number each time you start your application.

3. Format and print your random matrices to the serial console so you can check the input values against your output matrix later on.

- Hint: Use loop (or possibly multiple loops) along with `xil_printf("%d", tVariable);` to increment through the rows and columns of your matrix

3. Allow for a macro to set the size of the input 2D array (i.e. `#define SIZE`). We'll use this to see the effect on matrix size against the required clock cycles for computation.

4. Write a function that takes 3 array pointers and multiplies the first two together, storing the result in the 3rd.

5. Use the timer from step 1. to sample the time before and after the multiplication and calculate the time taken for the computation.

6. Format and print your output solution matrix to the serial console. Do this outside of your timer so as not to affect the timing operation.

7. Don't forget to deallocate memory and run `cleanup_platform()` at the end of your program to release any used instances from memory.

Note - You can print data to the serial terminal (within the Xilinx SDK) using the "xil_printf.h" library. This function however does not support floating point values, so you will need to use printf() from the <stdio.h> library with standard "%f" formatting, instead.

Extra

1. Modify your function to handle 2D arrays that contain floats, how does this affect the computation time?

2. Think about how we might do this multiplication on an FPGA? Could we offload the multiplication from the processing system to the FPGA to improve performance? What are the constraints with working with floating point numbers on an FPGA?