ECE4530 Fall 2017
Homework 8: Bandwidth optimization for on-chip FPGA memory

Assignment posted on 28 November
Solutions due on 5 December 5:00PM

Homework 8 is an extension of the design discussed in Lecture 22, namely the integration of DMA controller to optimize the access latency to on-chip FPGA memory.

You will have to start from the example discussed in Lecture 22. This code is included in the repository for Homework 8. If you did not attend Lecture 22 (Tuesday 28 November), go the lecture notes on Canvas and study the material.

In the example, a DMA controller is used to scan out the 128 KByte of on-chip FPGA memory, and copy it into a 'mysum' coprocessor that accumulates all the bytes. To compare the speedup, the computation is also done by reading the 128 Kbyte memory content into the ARM and summing up the bytes in software.

Your task for Homework 8 is to investigate the effect of the data port width of the on-chip memory. You will increase the data port width from one byte to 4 bytes (32 bit), and adjust the software to benefit from this increased data port width. Finally, you will make a performance comparison similar to that of Lecture 22, in order to quantify the effect of increasing the data port width.

You can use example code and concepts developed in this homework as part of your codesign challenge, if you find that they are helpful.

How to start

1. When you accept the homework, you will get a repository identical to that of Lecture 22.
2. Make sure you understand that example. Re-read the notes of Lecture 22 if needed.
3. To increase the on-chip memory 1 data port width to 32 bit, and make sure that the software can make use of it, you will need to make modifications to the design. You can do so directly in the repository created for the Homework. The following items, as listed in the next few points, will have to be changed.
4. The on-chip memory 1 data port width has to be widened from one byte to 4 bytes (32 bit). Do this by double-clicking the on-chip memory components in QSYS and adjusting the port width parameter.
5. Enable word-width (32-bit wide) transfers in the DMA controller settings of the hardware in QSYS.

6. The C software `sumdemo` has to be adjusted so that DMA transfers from mem1 into the on-chip coprocessor mysum are word-wide instead of byte-wide.

None of these changes are complex, but you have to do them carefully in order to avoid coding errors. Once you widened the data access, rerun the software driver `sumdemo`. Add a file in the root directory of the repository called `output.txt` with the output of the program as captured from screen.

The expected output, if you changed the memory width correctly, is that the cycle count for the DMA-based memory-sum operation is around 530K clock cycles, as opposed to 2.1M clock cycles for the byte-based example discussed in class. In other words, the 4-times wider memory increases the speed of the operation in hardware four times. The speed of the software sum on the ARM remains unchanged, because we're not changing the byte-by-byte sum in software.

What to turn in

Push all the changes you made in the repository back into Github. Add the new file `output.txt`. Complete the assignment before the deadline, 5 December 2017 5PM. Good luck.