University of Pennsylvania
Department of Electrical and System Engineering
System-on-a-Chip Architecture

ESE532, Spring 2017 HW5: Memory and I/O Sunday, February 12

Due: Friday, February 17, 5:00PM

In this assignment, we will investigate how we can change the memory organization of the encoder to improve its performance. We will build upon the encoder that we mapped onto MicroBlazes in homework 4.

The emphasis of this homework is on the revision of your memory organization. If you need too much time for performing the microbenchmarks, reach out to the TA. In case that does not help, make reasonable assumptions on the outcomes of the benchmarks and use those for the second part of the homework. Because we expect that implementing a new memory organization will be too demanding for a one-week assignment following the overlong homework 4, we do not ask you to implement it.

Revised Hardware Platform

We provide a revised version of the hardware platform of homework 4 at the the course website. The main differences with respect to the homework 4 platform are that the MicroBlaze processors have:

- larger local data memories. We increased the local data memories from 8 KB to 32 KB.
- larger stream buffers and victim buffers. Note that we do not expect you to know what these are. All you need to know is that they may improve the performance.
- longer instruction cache lines
- write-back cache policy in data cache. The previous platform used a write-through policy.
- no floating point unit. You probably won't need floating point operations anyway.
- no divider. You probably won't need integer division either.

On the flip side, there are only 8 processors in this platform.

Video

Because the 1920 × 1080-pixel frames result in excessive runtimes, you can assume that we will encode a stream with 480 × 270 pixels per frame. This should improve caching oportunities.

Software

The software for this assignment can be downloaded from here. The archive contains several projects that can be loaded into SDSoC. An example of how to use DMA is in the `hw5_arm_test` project. The `hw5_mb0_test` project contains a function, `read_counter` to read a clock counter on a MicroBlaze. It should return the number of cycles at the MicroBlaze frequency. Please check the output with a stopwatch to ensure that it works on your system.

Local Memory

Each MicroBlaze has local memory. This memory is not accessible by the ARM or other MicroBlazes. The memory is located at address 0x00000000. This memory is not cached, but it is located on the PL close to the processor, so the latency is comparable to a cache. If you use JTAG mode for configuration, you can also assign variables directly to local memory by adding the `section` attribute, and setting it to the section name of the local memory. You can find the section names under `MEMORY` at the top of the `lscript.ld` file. This method will probably not work when the FPGA is configured from the SD-card.

Debugging

In the previous assignment, we needed the SD-card because we needed full control of the boot order of the processors. In the meantime, we have figured out how you could use JTAG instead. Follow the following procedure to run code on a MicroBlaze:

1. Make sure that the jumpers on your board are set to JTAG mode.
2. Make sure that the board is connected using 2 USB cables and powered on.
3. Open the *Debug Configurations* dialog.
4. Select the *hw5_arm_test Debug* configuration.
5. Start the debug session by pressing the *Debug* button.
6. Run the application by pressing *Resume*.
7. Now, the MicroBlaze should be configured on the FPGA. To run an application on the MicroBlaze, open the *Debug Configurations* dialog again.
8. Select the *hw5_mb0_test Debug* configuration.
9. Start the debug session by pressing the *Debug* button.
10. If SDSoC asks whether it should terminate the previous session, confirm it.

Once you have performed this procedure, you can restart the MicroBlaze application any time by relaunching just the *hw5_mb0_test Debug* configuration. Only if you turn your board off, you will have to run the ARM application again. Note that running the `init_board` function is all you have to run to get the MicroBlaze started.

Homework Submission

Please be concise in your answers.

1. **Learning and navigating the tools.**

   Document all the problems you encountered in working the assignment and how you solved them. For each, include:

   (a) Step (if applicable)

   (b) OS you were running

   (c) Error message or unexpected behavior you encounted

   (d) How the problem was resolved

   (e) How you found the solution (e.g., how did you experiment or reason through an answer, which document had the answer, where on the web did you find an answer, which student or TA was able to point out the answer)

2. **Teamwork**

   (a) Document the work sessions with your partner (where, when, duration).

   (b) How did you collaborate on the work? (pair programming is acceptable, explain)

   (c) What background basics, tricks, or techniques did you learn from your partner?

   (d) Provide highlights for cases where working together allowed you to understand deeper, combine insights, avoid pitfalls, and/or build confidence in your directions and solutions.

3. **Microbenchmarks** We will investigate the performance of certain memory transfers that could be of interest in your encoder implementation. You don't have to implement these in the encoder. You can also create an image with only the processor that you need. Make sure caching is enabled on the DRAM, and disabled on the OCM. Use optimization level `-O2`. Verify whether data was transmitted correctly before reporting your results.[1]

   (a) Measure the delay and compute the bandwidth for a number of data transfers. Put the numerical results in a table such as the one below the questions, and add the relevant code in an appendix of your report. A table and code are sufficient. We do not accept screenshots of code.

      i. Report the time needed to copy 10 MB of data from the DRAM to the OCM using the ARM.

      ii. Report the time needed to transfer 10 MB of data frames from the DRAM to the OCM via DMA configured by the ARM.

      iii. What is the slowdown or speedup that you obtained using DMA?

[1] Perhaps use a summation or signature on the data.

iv. Report the time needed to copy 1 MB of data from the DRAM to the local memory of one of the MicroBlaze processors. Make sure you invalidate the data cache before you read to ensure that you are not reading from cache.

v. Report the time needed to copy 1 MB of data from the OCM to the local memory of one of the MicroBlaze processors.

vi. What is the slowdown or speedup that you obtained for transferring data from the OCM vs data from the DRAM?

vii. Report the time needed to read 64K 16-byte chunks from random locations in the DRAM by the MicroBlaze. Make sure your read operations are not optimized away by computing a signature or summation and printing the result for example. For the "random" locations, have the i'th read be to `(base+i*37*16)%size`, where `base` is an address such that `base` through `base+size-1` are in specified memory. Use a `base` of `0x18000000` and a `size` of 64M for the DRAM.

viii. Report the time needed to read 64K 16-byte chunks from random locations in the OCM by the MicroBlaze. `size` is 256K for the OCM.

ix. Report the time needed to read 64K 16-byte chunks from random locations in local memory by the MicroBlaze. `size` is 32K for the local memory.

(b) Give two possible advantages of using DMA.

Transfer	Duration	Bandwidth (B/s)
DRAM → OCM copy		
DRAM → OCM DMA		
DRAM → OCM speedup		
DRAM → local memory		
OCM → local memory		
OCM → local speedup		
Random read DRAM		
Random read OCM		
Random read local memory		

4. **Revised Memory Organization** Based on the results of the previous question, describe the encoder memory organization that you expect will have the best performance in detail. As in homework 4, we expect you to decompose the `motion_estimation` function. The remainder of the encoder will be run on the ARM core. Assume that frame i is dependent on the motion vectors of frame $i-1$. Keep in mind that certain memories (and caches) have lower latencies than others, so it is beneficial to keep data that is accessed more frequently close to the consumer.

(a) Describe your strategy for decomposing the parallelism among multiple microblaze cores in the `motion_estimation` function.

(b) Give a memory layout that shows where in each memory you store the different data and code used by the encoder and the communication buffers. Indicate the starting and ending address and size of each area or memory.

(c) Describe which data is transferred between the processors. Illustrate the data movement in a block diagram with the relevant memories and processors.

(d) Describe using which method(s) the data is transfered. Motivate your transfer methods.

(e) If you divide the data into smaller chunks, describe how you do this. Illustrate the decomposition using a diagram. Indicate the data dimensions, and how you traverse the data (e.g. using arrows). Motivate your choices.

(f) Estimate the performance you believe your design can achieve when using the 8 microblaze cores, assuming data movement (including all the microblaze reads) throughput, latency, and dependencies determine performance.