Lab on HW-SW digital systems codesign
SEL B.Sc. Curriculum (Spring Semester)
Embedded Systems Laboratory
EPFL-STI-IEL

Exercises Session 7
Out: 2025-04-03

Cache memory and memory hierarchies

In this session, we will conduct a small experiment to try to guess the size and associativity of the different data caches present in the ARM cores of the ZYNQ 7020 FPGA.

1 Experiment to determine the size and associativity of the caches

To find out the size of the caches and their associativity, we propose to use a program that adds all the elements of multiple vectors, traversing all the vectors in parallel. The experiment will have two configurable parameters: the length of the vectors, and the number of vectors. Since the vectors are allocated using a much larger size than the capacity of any cache in the system, and using powers of two, we know that they will alias on the same cache lines. These two parameters will allow us to observe the behavior of the cache while changing the number of simultaneous ways (sets) that the application would require to be able to access consecutive positions in the vectors from the cache. We will also be able to observe the effect of the working set size.

We can also repeat the execution of the application multiple times to compare the time required for the first execution, when all the vectors need to be read from main memory, with the time required in consecutive executions when, according to the working set size, the vectors may already be in the cache.

Use Table 1 as a reference to organize the obtained results.

▶ **Question:** What is the size of the data L1 cache? (The data L1 is individual for each core.)

▶ **Question:** What is the size of the L2 cache? (The L2 is shared between the cores, and holds both data and instructions.)

▶ **Question:** What is the associativity of the data L1 cache?

▶ **Question:** What is the associativity of the L2 cache?

The source code of the application can be downloaded from Moodle.

2 Additional experiment

As an additional experiment, use the provided code for the multiplication of two matrices. In this case, try to imagine how the rows and columns are accessed to estimate the size of the working set and to analyze its impact on performance.

Table 1: Example.

# of vectors	Size (elements)	First round (ns)	Next rounds (ns)	Working set size (B)	Time reduction warm cache	Time increase w.r.t. previous number of vectors)
1	512					
1	1024					
1	2048					
1	4096					
...	...					
1	524288					
1	1048576					
2	512					
2	1024					
2	2048					
2	4096					
...	...					
2	524288					
2	1048576					
...	...					

Table 2: Example.

Matrix size (dimension)	Elements per matrix	Total elements	Matrix size (B)	Working set (B)	Time (ns)	Time increase
8	64	192	256			
16	256					
24	576					
32	1024					
...	...					
512						
1024						

The provided code includes also a matrix multiplication implementation that first transposes the second matrix.

► **Question:** In which cases transposing first a matrix can help with performance?)

► **Question:** Why does performance increase in some cases, even if the application does extra work?)

In this experiment, a table similar to Table 2 may be useful to organize the results.