Laboratory Exam 2024

Objective: Use the capabilities of Pynq-Z2's FPGA to accelerate an example application

Steps:

1) Test the Jupyter notebook of your example application.

For this step you have to:

a) Run the notebook on a PC, check the output and the processing time.

b) Notice that the notebook is divided into two code cells. The first cell needs acceleration and is the code that will be implemented into the FPGA. This cell is composed of a compute function, which is used in the second cell, and some auxiliary data or functions that are only used internally by the compute function. The second code cell is used just for visualization and can be run on the Pynq-Z2's CPU.

c) Copy the notebook into the Pynq-Z2 board. Run the notebook and notice the processing time.

2) Build an IP core for your application.

For this step you have to:

a) Translate the source python code into C++ code (Any automatic tool like ChatGPT can be used)

b) Write a simple C++ testbench, to execute the compute function

c) In Vivado HLS, set up the IP core project and use the testbench to verify that the C++ code has the same behavior as the Python source code

d) Select the AXI interfaces the C++ main function will need in order to communicate with the Zynq SoC (for example, if the IP core reads the incoming data or writes the output data in a step by step fashion, use an AXI Stream interface, if the IP core needs to read or write just one value use an AXI lite interface).

e) Add the required pragmas to the compute C++ function to use those interfaces.

f) Add `#pragma HLS INTERFACE s_axilite port = return` into the compute function, and return a value of 0 when the process was successful

g) Modify the testbench as needed and check that the behavior of the code is still correct.

h) Synthesize and export the IP core

i) From the synthesis reports, notice:

I) Max frequency of the IP core

II) Latency of the IP core

III) FPGA's resource utilization of the IP core

j) In order to improve the performance of the IP core

I) Optimize the C++ code to better suit the FPGA capabilities.

II) Add HLS pragmas into the C++ code to specify how the High Level Synthesis tool should build the hardware to achieve better performance.

3) Using Vivado, build the bitstream of the acceleration system.

For this step you have to:

a) Add a block design

b) Add into the block design the Zynq SoC

c) Modify the Zynq SoC as needed (Add a HP slave interface, modify the PL max frequency to be lower than the max frequency of the IP core

d) Add into the block design the constructed IP core

e) Add any other required IP core (for example DMAs)

f) Connect the different IP cores

g) Build the bitstream

4) Write a Jupyter Notebook to run the acceleration system on the Pynq-Z2 board

For this step you have to:

a) Use the pynq library to load the bitstream into an overlay

b) Check the ip_dict of the overlay to verify that your IP core is present

c) If you used the AXI lite interface for any input/output parameter of the compute function

I) Use the ip_dict to get the address offset of the parameter

II) Write the required values into the register

d) Start your IP core by writing 0x01 into the control register (address 0x0)

e) If you used the AXI stream interface for any input/output parameter of the compute function

I) Use the ip_dict to get the name of the IP core for the corresponding DMA

II) Use pynq.allocate to create the required input/output buffers

III) Use the function transfer() to start the input/output DMA transfer

IV) Use wait() to give time to the IP core to finish processing the data

f) Show the IP core results, using a code similar to the second block of the original jupyter notebook

g) Notice the processing time

5) Write a report of your work.

For this step you have to:

a) Write an introduction explaining your example application, what it is, which are the main computational steps or the original Python code. (at least 200 words)

b) Explain the main steps you took to implement the IP core. Explain the structure of the translated C++ code, the testbench used to verify that the code was correct, the selection of the AXI interfaces, the synthesis results of the IP core (max freq, latency, hardware resources). Explain any steps you took to optimize the code for the FPGA. (no source code, at least 400 words)

c) Explain the main steps you took to synthesize the bitstream. Explain the IP cores you used, and the interconnection between them (at least 200 words)

d) Explain the main steps you took to write the Jupyter notebook to run the application (no source code, at least 400 words).

e) Write a conclusion section where you analyze the execution time results you obtained, compare them with the original jupyter notebook running on PC and on the Pynq-Z2, mention which steps could be taken in the future to improve the performance of the system (at least 200 words)

Each group has to present:

1) The source code and the testbench of the IP core
2) The bitstream (both the .bit and .hwh files)
3) The Jupyter notebook to run the system
4) The report

The evaluation score will be given following this criterion:

The testbench in step **2) c)** can correctly mimic the results of the original jupyter notebook
4 points

The testbench in step **2) g)** can correctly mimic the results of the original jupyter notebook
4 points

Any step **2) j)** was taken that improves the the max frequency or the latency from step **2) i)**
4 points

The Jupyter notebook of step **4) f)** can correctly mimic the results the original Jupyter notebook
4 points

The report complies with the requirements 5**) a, b, c, d, e**
4 points

The exam is considered approved if the group reaches a score of at least 10 points