Hardware Acceleration using FPGA

Franco Carbognani, Rhys Poulton – EGO

AHEAD2020 WP12 F2F meeting | 04 May 2022

Why FPGA?

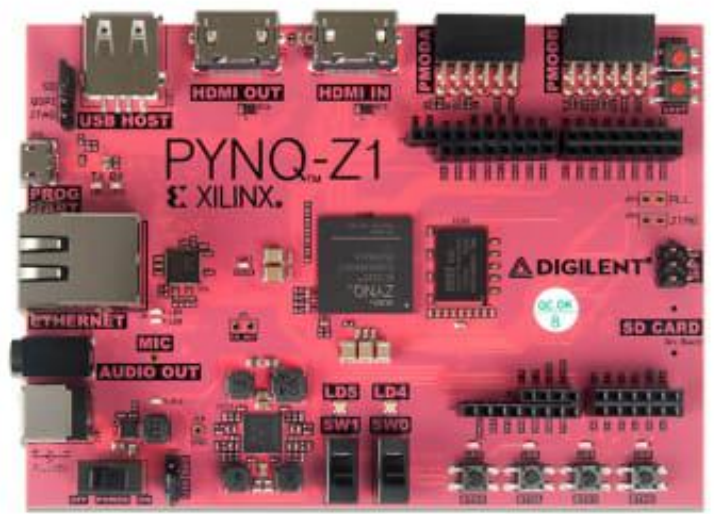
- Field Programmable Gate Array (FPGA)
  - An integrated circuit that can be programmed by the user.
  - Has a physical array of logic gates that can be customized to perform specific computing task
  - Highly optimized for a specific computing task

Why Python?

- More and more, the language of choice is Python:
  - It allows for compact and clear code that is relatively easy to learn for beginners.
  - It provides advanced programming concepts like exceptions, object-oriented programming, functional programming, threads, generators, etc.
  - With the Numpy and Scipy libraries, it has most of the complex math we need. These libraries use optimized Fortran and C-code under the hood, so they are fast (but we want them faster…..)
  - Python tremendously rich and diverse set of packages, libraries and tools (the so called “batteries included”) allows the development of applications at a speed unknown before.

FPGA + Python = PYNQ™

- PYNQ
  - PYNQ is an open-source project from Xilinx in python for interaction with the ZYNQ chip
- PYNQ-Z1 FPGA
  - Uses ZYNQ chip that contains Programmable logic to speed up signal processing
  - Jupyter Notebook based interactive computing environment
  - Can use hardware libraries and overlays to program the board in the jupyter environment



Why use FPGA for hardware acceleration?

- Part of the AHEAD2020 WP12.5 goal:
    - Improving and accelerating via hardware acceleration techniques (Pynq) the software applications used to characterize and reduce the noise

- Identified applications:
    - Use the PYNQ FPGA Board with a embedded Finite Impulse Response (FIR) filter to speed up whitening of the h(t) for Low Latency pipelines
    - Utilize the PYNQ FPGA Board to accelerate machine learning algorithms that have been implemented in Low Latency Pipelines

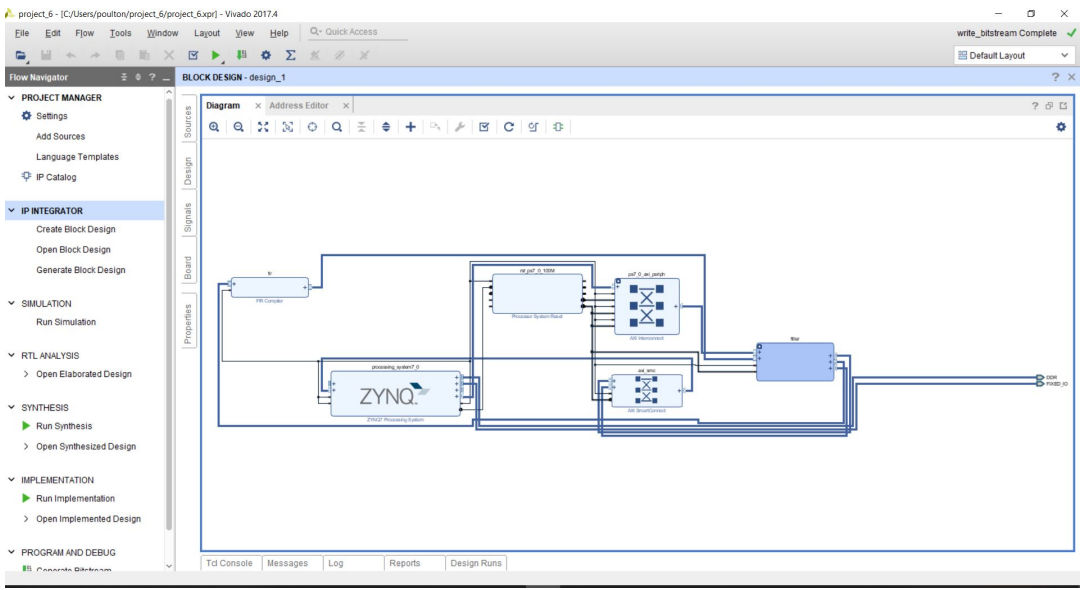
How to program the FPGA with a FIR filter?

- To embed a FIR filter in the FPGA an overlay has to be created which programs the board

- The overlay is created using the Vivado Design Suite
  - Uses Hardware Description Language to create designs

- The design of the FIR filter in the FPGA is shown right

- Once the overlay is created, it is then ready to be run in the Jupyter Notebook

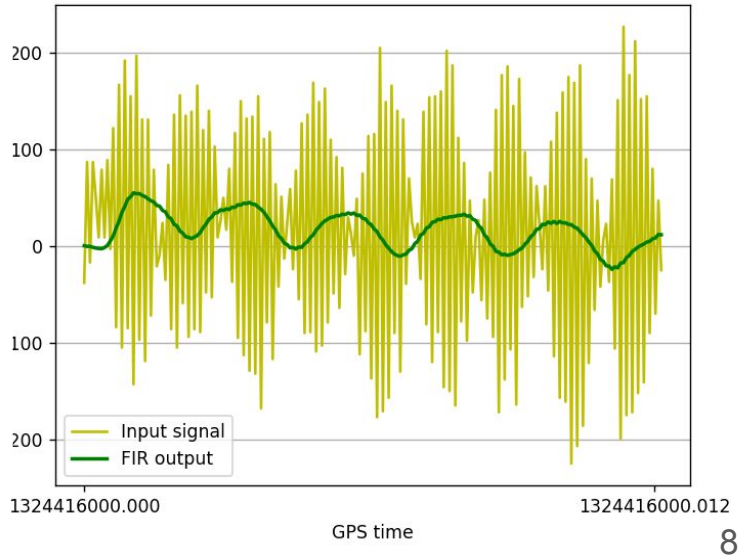


Running the FIR filter

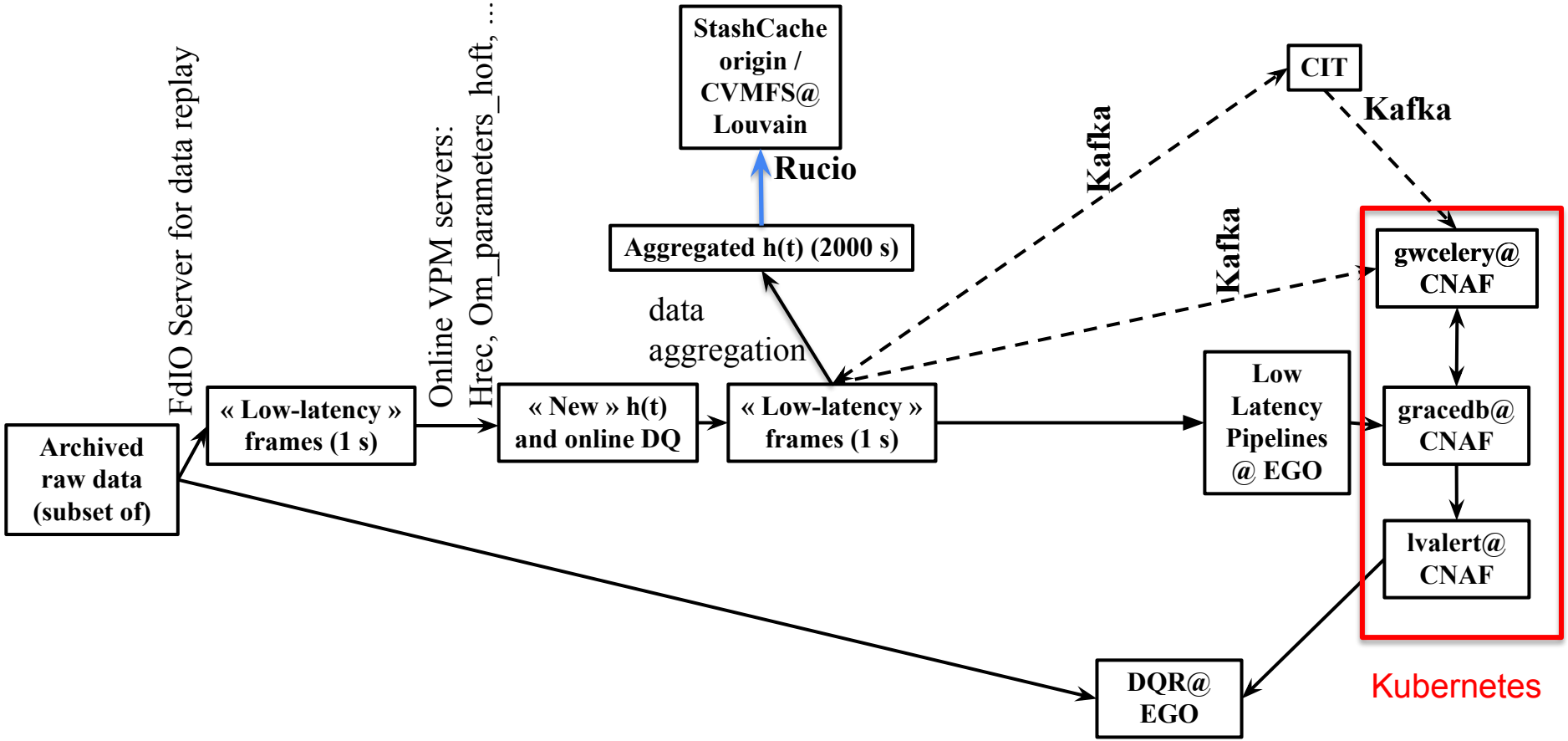
- The board has been fully embedded into the Virgo computing environment
- The overlay is loaded onto the FPGA by the jupyter notebook that is running on the board
- h(t) data is streamed to the board from the O3 Replay using Kafka
- Still need to calibrate the FIR coefficients to the h(t) signal
- Up to 100x faster than using scipy's whitening
- The FIR filter has been running for several weeks without problems so the board seems reliable

Acceleration of a FIR filter on the PYNQ-Z1

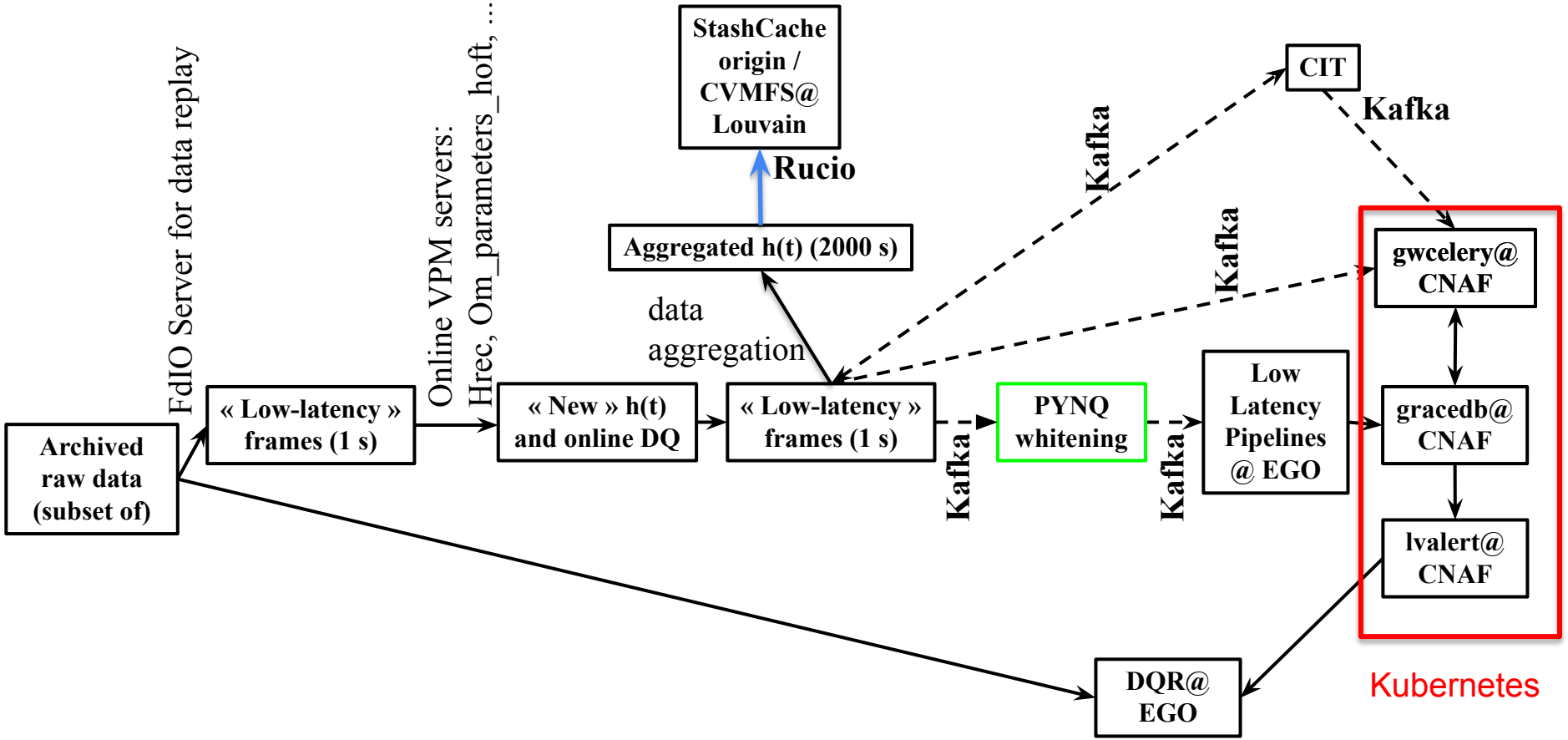
Accelerate a Python function on the Zynq-7000 using PYNQ. By testing a software FIR implementation, the SciPy function `lfilter`, and measuring it's performance. Then we load a custom overlay with a hardware FIR, and compare it's performance to the software implementation.



How PYNQ could fit into the low latency schema



How PYNQ could fit into the low latency schema



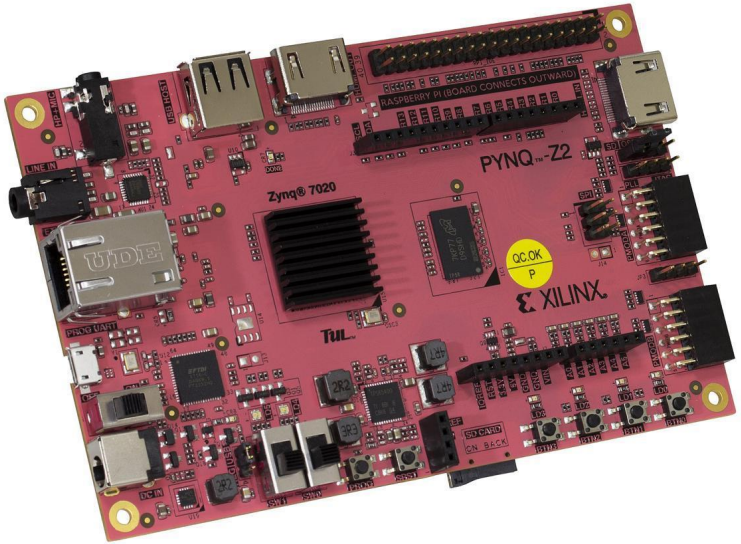
Next steps

- Need to calibrate the FIR filter coefficients to produce the desired whitening
- Have a low latency data analysis pipeline switch off its whitening and use pre-whitened data from the PYNQ board
- Targeting the coherent WaveBurst data analysis pipeline

Machine Learning using the PYNQ FPGA

- Next generation PYNQ-Z2 FPGA's
  - Procured two of those boards
  - Currently being setup.

- Target Machine Learning in Low Latency
  - GSTLAL early-warning pipeline
    - Detects BNS using the inspiral part of the waveform
  - WAVEFIER
    - Real time pipeline for the detection of transient signals and their classification
  - GW Celery
    - EM bright classification



Conclusions

- Currently have two use cases for PYNQ
  - Whitening for the data analysis pipelines: target cWB
  - Machine learning in either DA pipelines or GWcelery
- FPGA’s can provide a speedup of a factor of 100 times (respect to running on the PYNQ board itself, need to setup benchmarking respect to running on standard DA dedicated servers)
  - It can be implemented into the low latency schema to reduce the End-to-end latency