Embedded Linux in SoC-FPGA

Fabian Andres Castaño Usuga (PhD in Electronics Engineering and Computer Science)

Email: fabian.castano@udea.edu.co

Universidad de Antioquia (DUNE collaboration member since June 2022)

SoC-FPGA Architecture Overview

Processor System (PS):

- ARM cores, GPU, FPU, I/O

Programmable Logic (PL):

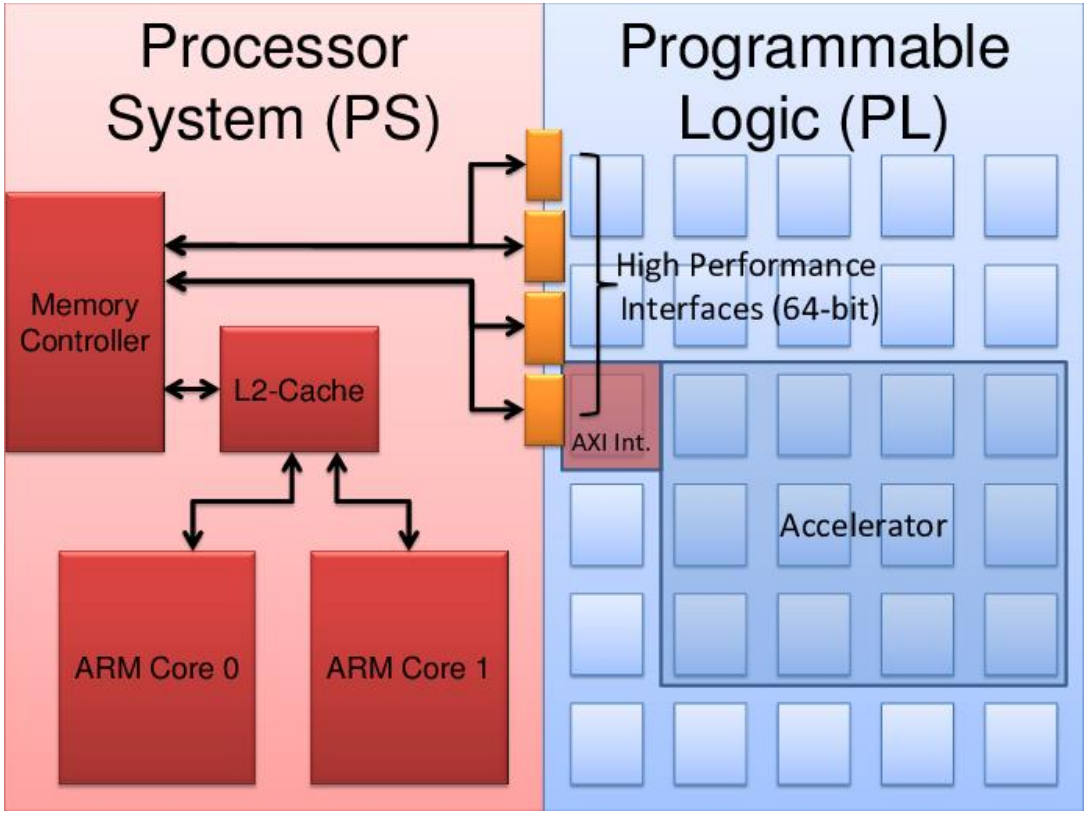
- reconfigurable hardware, AXI interface

AXI bus:

- connects PS and PL for data/control transfer

Software layer:

- manage resources in PL side from PS side



Development Levels in SoC-FPGA

- **Baremetal**: direct hardware access, full determinism

•No OS, runs directly on hardware
•Lowest latency and full control
•Ideal for ADC control or signal generation
Pros: deterministic, small footprint
Cons: no multitasking, hard to scale

- **FreeRTOS**: lightweight multitasking, real-time control

•Real-Time Operating System
•Multitasking with deterministic scheduling
•Best for concurrent control or sensors
Pros: predictable, modular
Cons: limited ecosystem

- **Embedded Linux**: full OS, drivers, Python integration

•Full OS with networking and multitasking
•Run Python, C++, shell scripts
•Access hardware via /dev/uio, /dev/mem, or drivers
Pros: flexible, feature-rich
Cons: higher latency, non-deterministic

How would you design this?

- Implementation of 50 MSps Oscilloscope with filters selection

Filters' selection	Data manager	High Speed ADC communication

How would you design this?

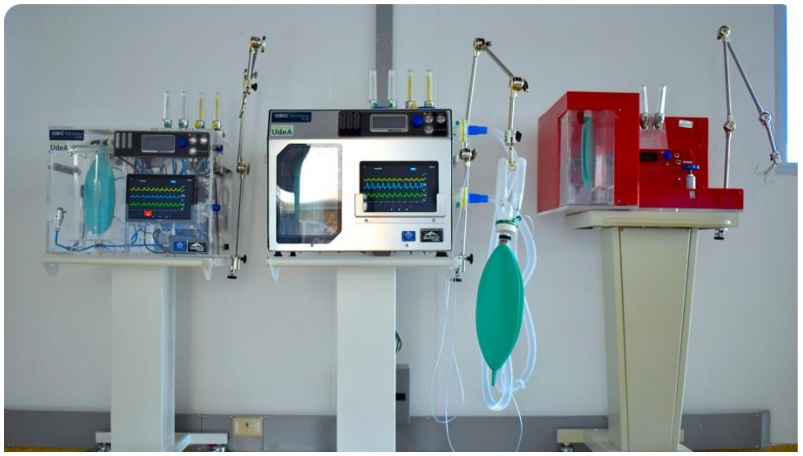
- Control and configuration of high performance multi-stack DAQ

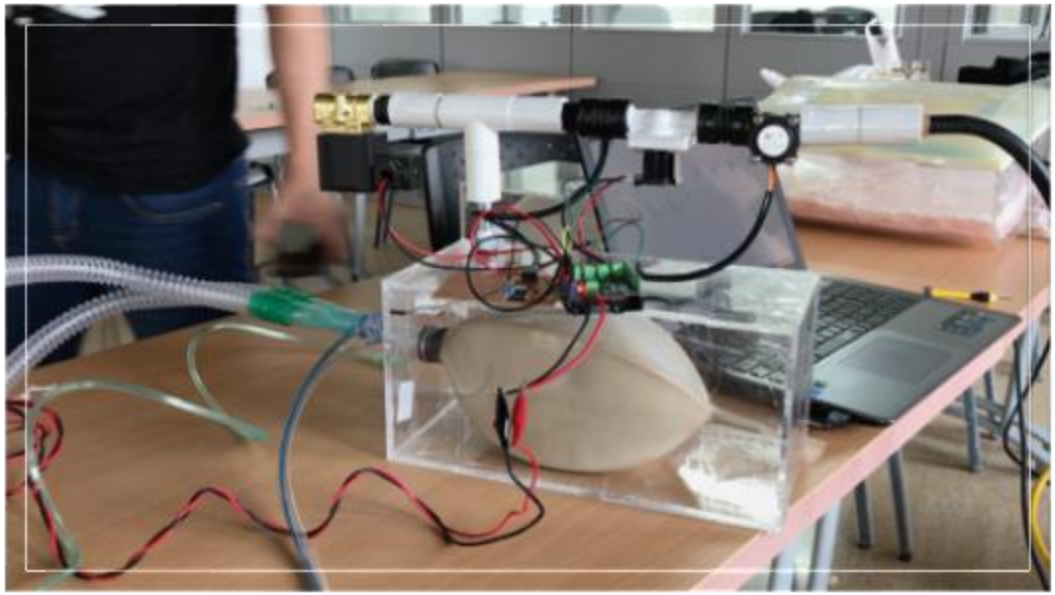
Slow control of each distributed system	Data processing and transport	Signal acquisition in each detector

How do I introduce myself to Embedded Linux?

InnspiraMED (Innovation and Inspiration from Medellín)

- COVID 19 Pandemic brings a lot of challenges
  - Public health
  - Social and Economics
  - **Technology**





GIBIC

InnspiraMED

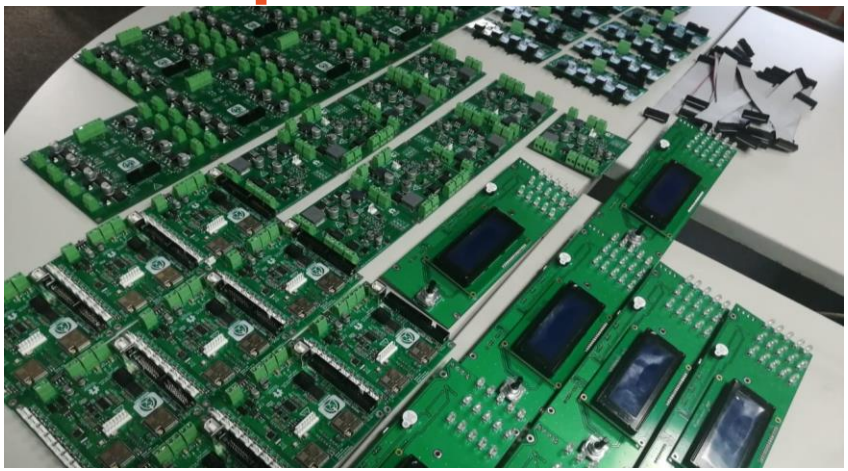
ICTP The Abdus Salam International Centre for Theoretical Physics

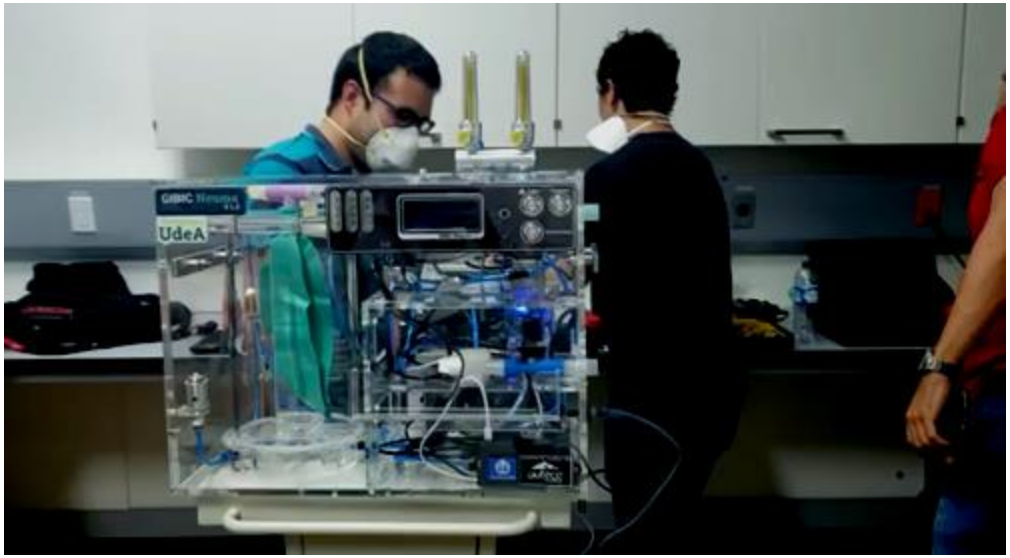
UNIVERSIDAD DE ANTIOQUIA
Facultad de Ciencias Exactas y Naturales

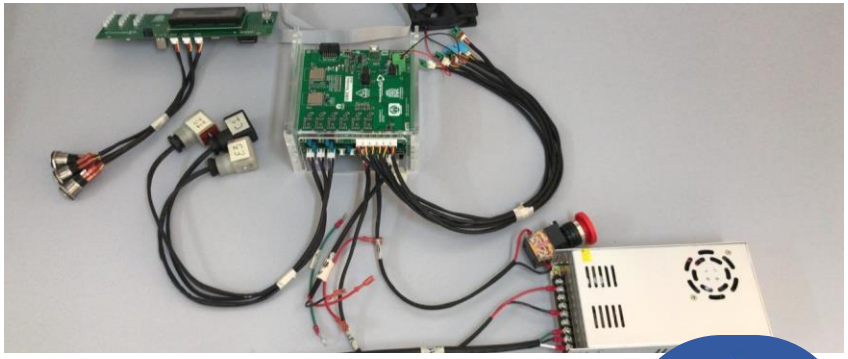
DUNE

InnspiraMED (Innovation and Inspiration from Medellín)

- How to build an architecture for a critical equiment and in a short period of time?







GIBIC

InnspiraMED

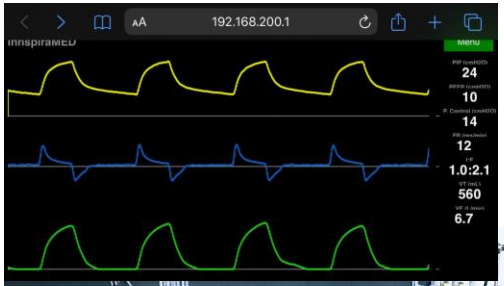
ICTP The Abdus Salam International Centre for Theoretical Physics

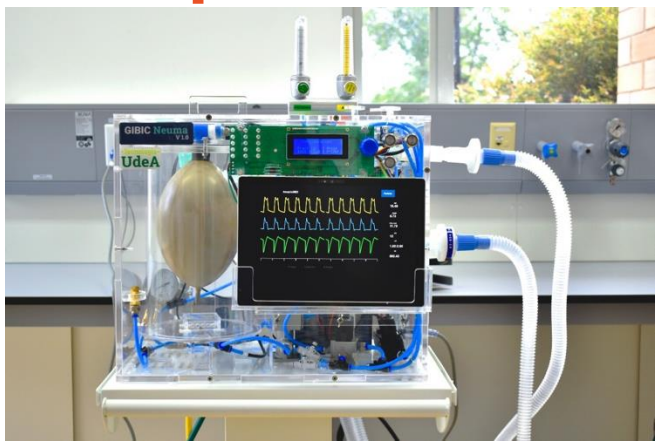
UNIVERSIDAD DE ANTIOQUIA
Facultad de Ciencias Exactas y Naturales

DUNE

InnspiraMED (Innovation and Inspiration from Medellín)

- How ensure the health of physicians?





GIBIC

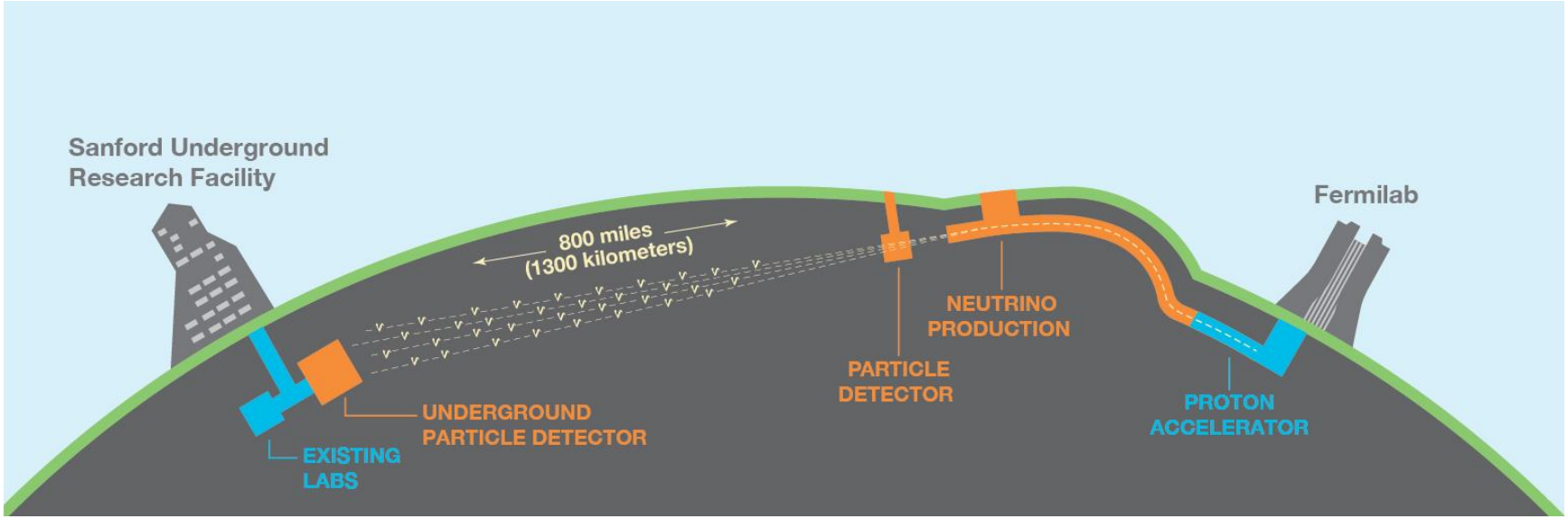
InnspiraMED

The Abdus Salam International Centre for Theoretical Physics

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ciencias Exactas y Naturales

DUNE

Deep Underground Neutrino Experiment

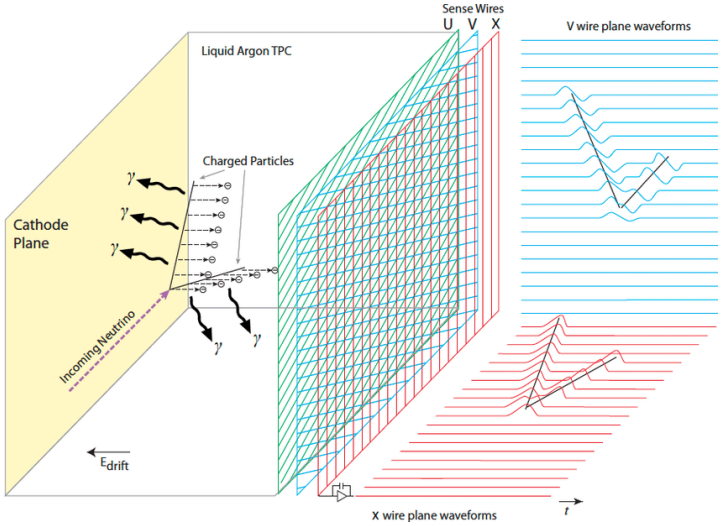


- Investigate Neutrino oscillation to prove charge parity violation (CP)
- Determine the order of the mass of the neutrinos
- Study supernovae, the formation of neutron stars and black holes
- **Run up at summer 2028**

The Abdus Salam International Centre for Theoretical Physics

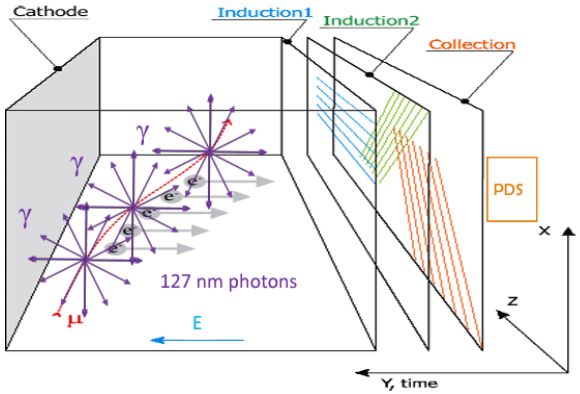
UNIVERSIDAD DE ANTIOQUIA
Facultad de Ciencias Exactas y Naturales

Interaction chain (photons and charge)



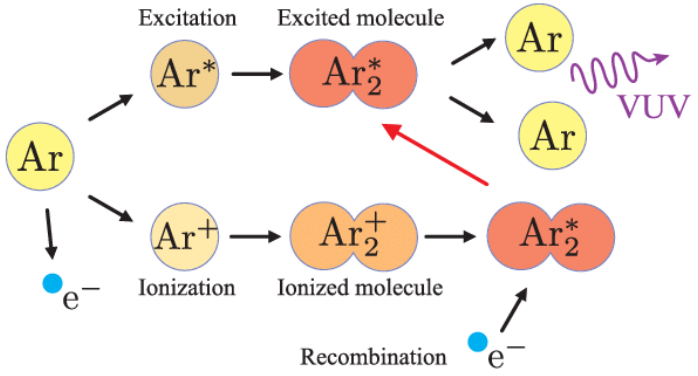
LArTPC Interaction

Cross-Section = $1 \times 10^{-45}\ m^2$



P(x) = 2E-14

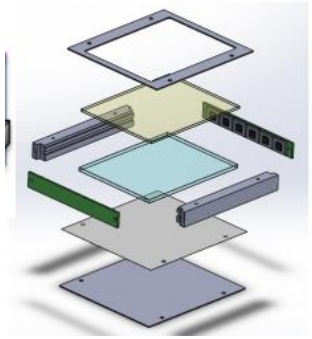
1 in 5 Billion of neutrinos

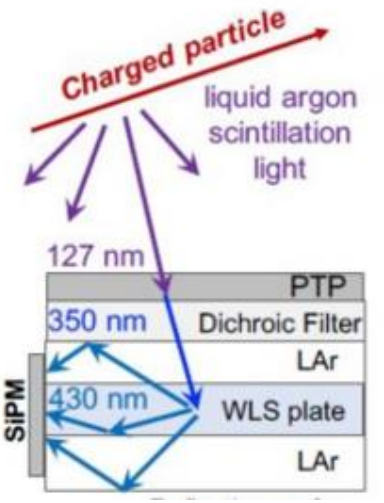


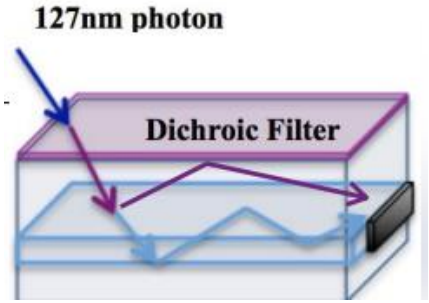
https://www.youtube.com/watch?v=R5G1_hW0ZUA

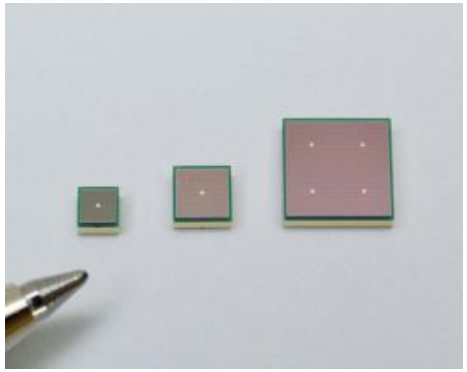
Photons' trap and detector

X-Arapuca

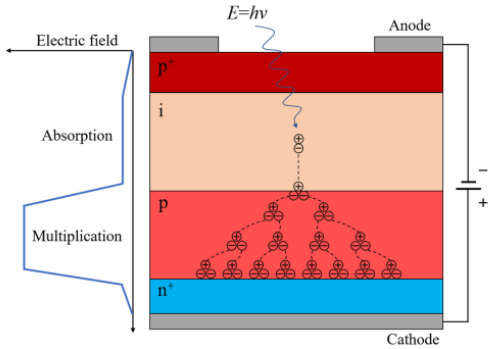




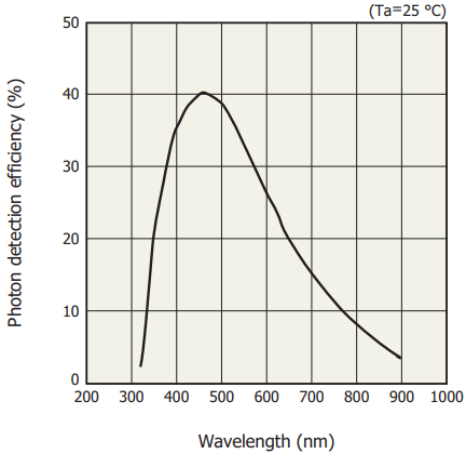




MPPC (Multi-Pixel Photon Counter)

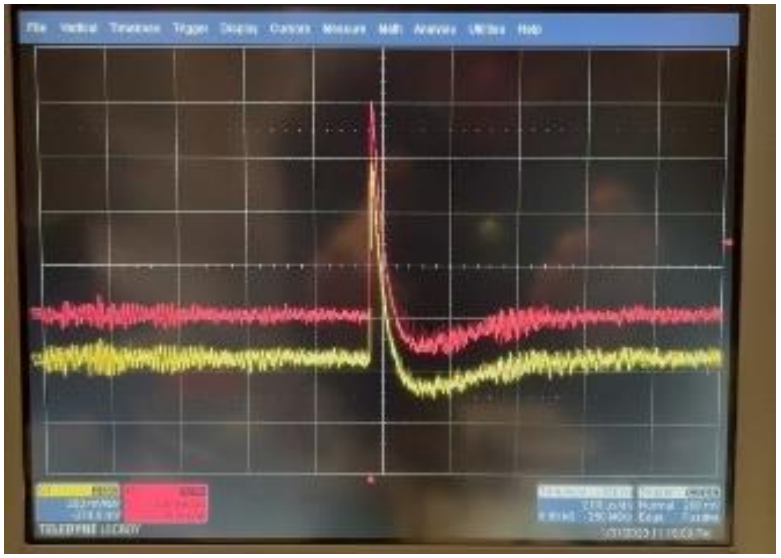


SiPM Hamamatsu S13360-6050VE

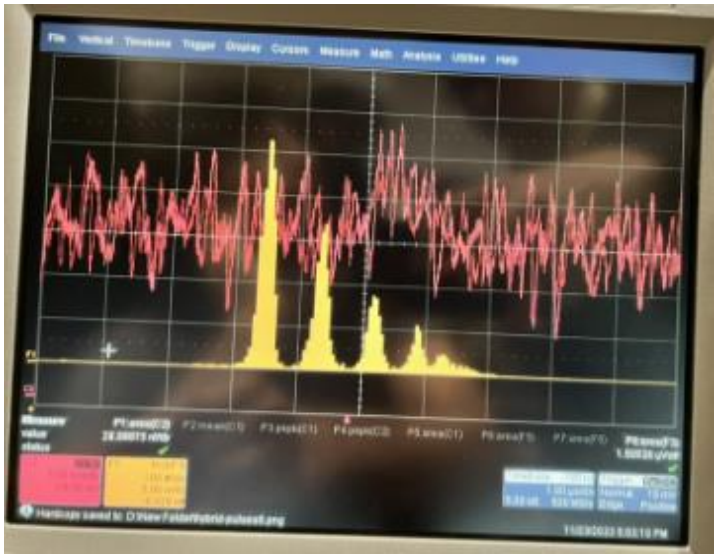


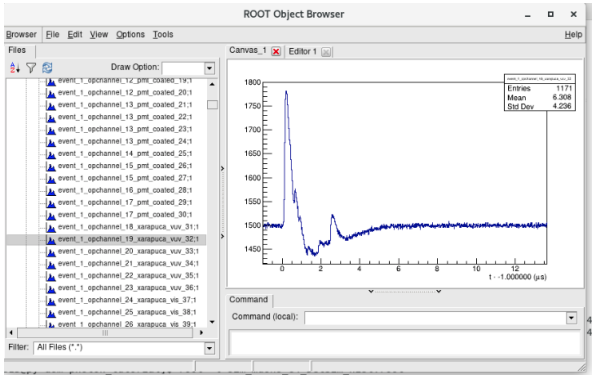
Photon Detection System (PDS)

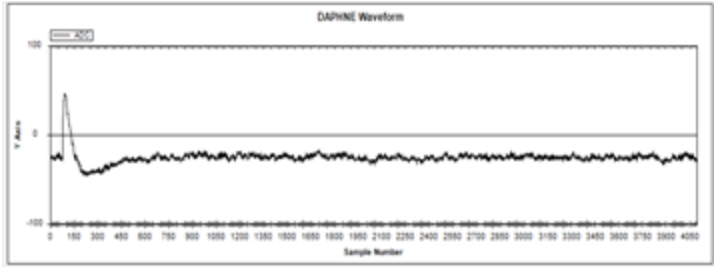
Analog signal form SiPM



Single Photon Detection

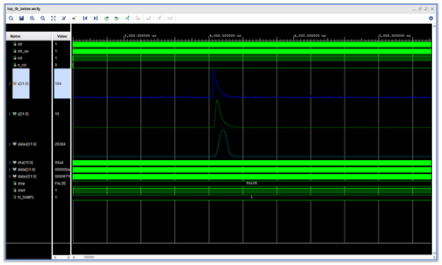






Event duration = 4 us

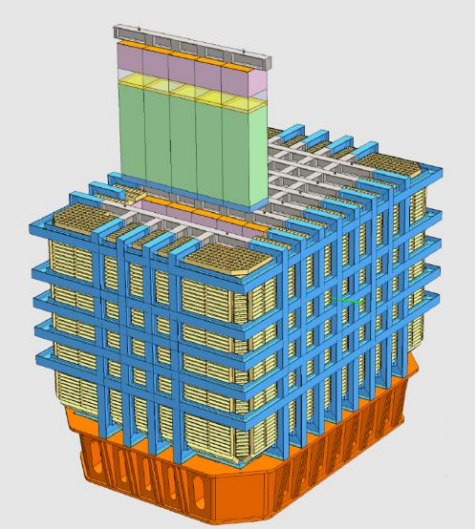
Discrimination by height of signal (**Energy of discharge**)

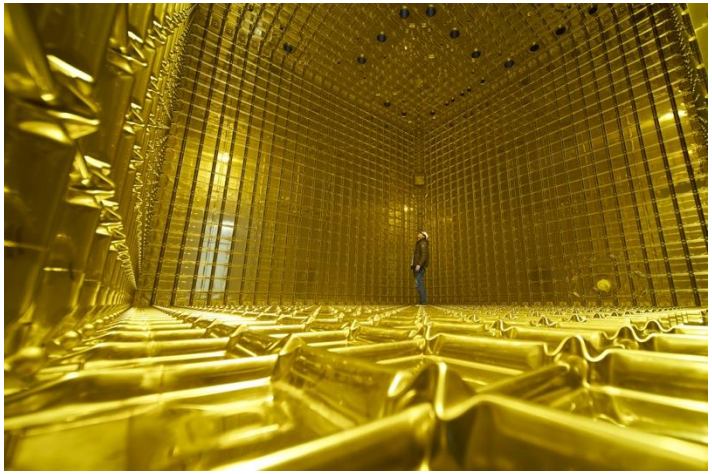


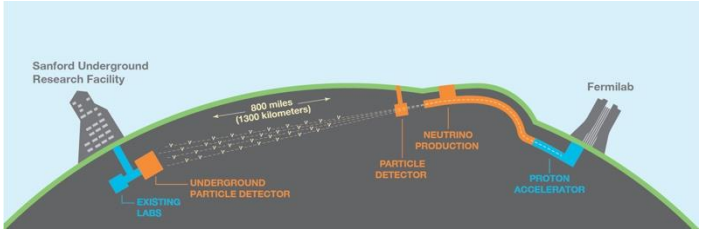
DUNE Detectors (PDS)

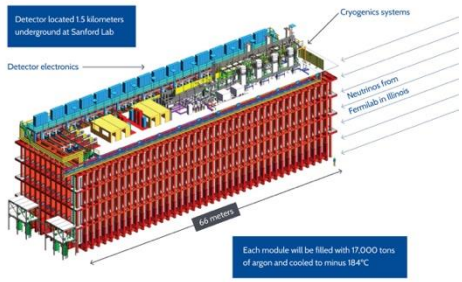
- Near and Far Detector

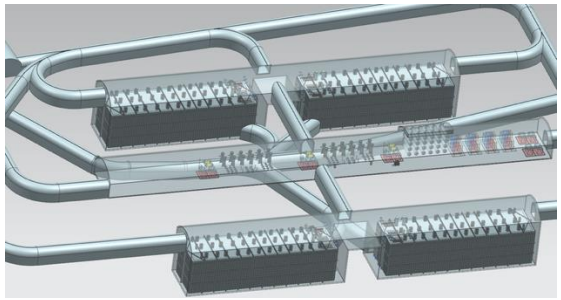
PDS
- 360.000 SiPM
- 6000 Xarapuca
- 150 DAPHNE
- 4 TB/s





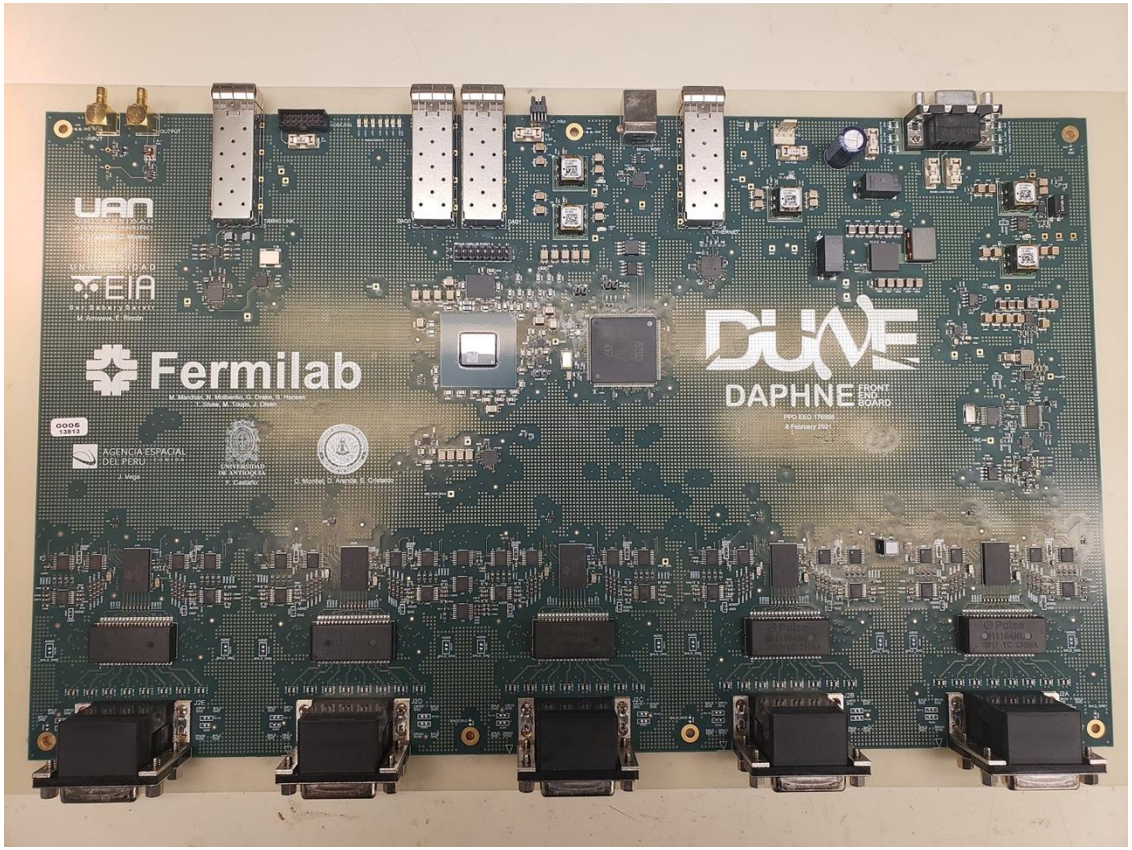




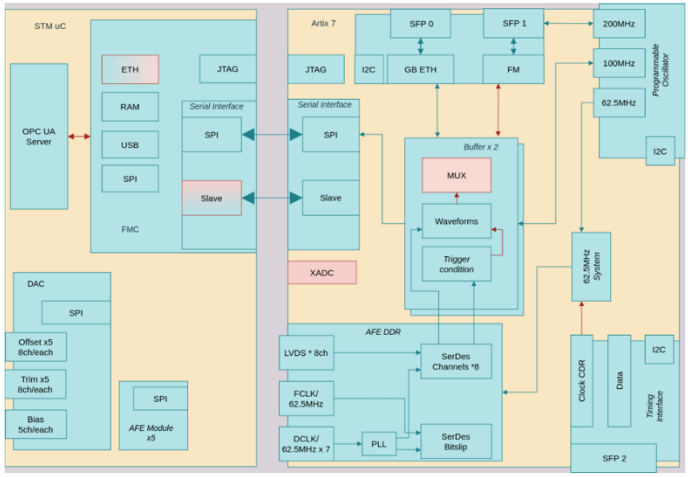


DAPHNE V1 and V2A

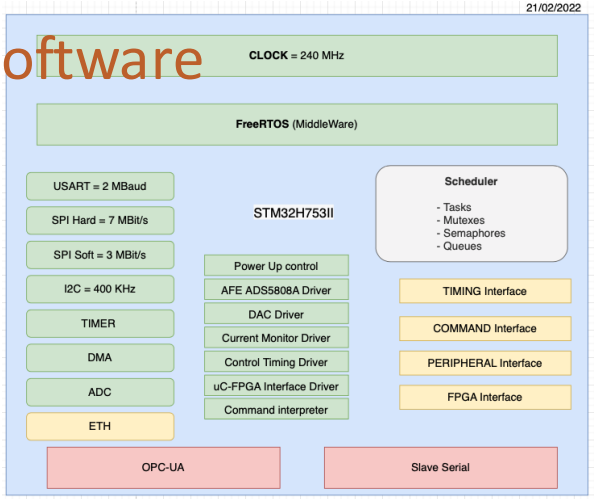
Hardware



Firmware

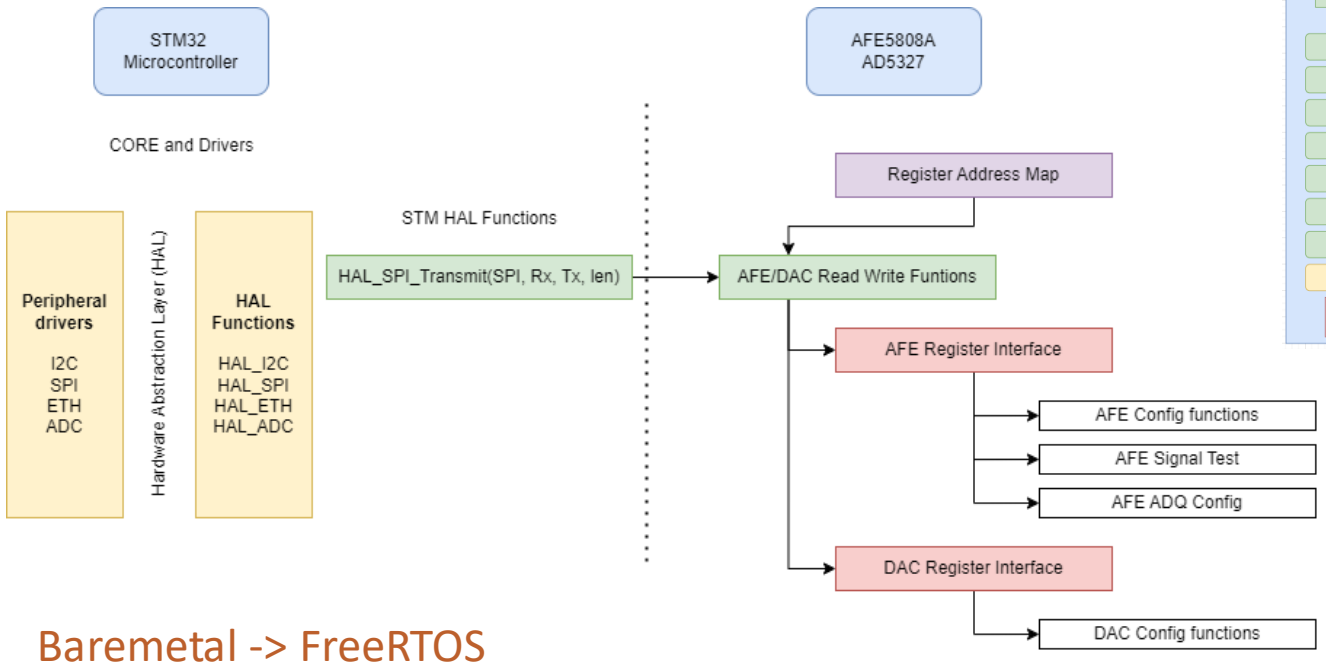


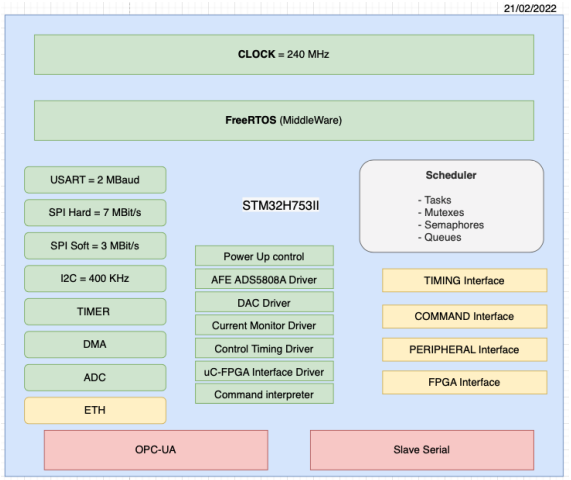
Software



Distributed design

- DAPHNE v1-v2a Slow Control Firmware architecture





Baremetal -> FreeRTOS

DAPHNE V3

- Kria KR260

Feature	Specification
Application Processor	Quad-core Arm Cortex-A53 @ 1.5 GHz
Real-Time Processor	Dual-core Cortex-R5F @ 600 MHz
Programmable Logic	256 K Logic Cells, 1 248 DSPs
Memory	4 GB DDR4 + 26.6 Mb On-Chip SRAM
Connectivity	PCIe Gen2 ×4, USB 3.0, GigE, DisplayPort
Power	7.5 W typical, passive cooling

XILINX VITIS

VIVADO

PetaLinux

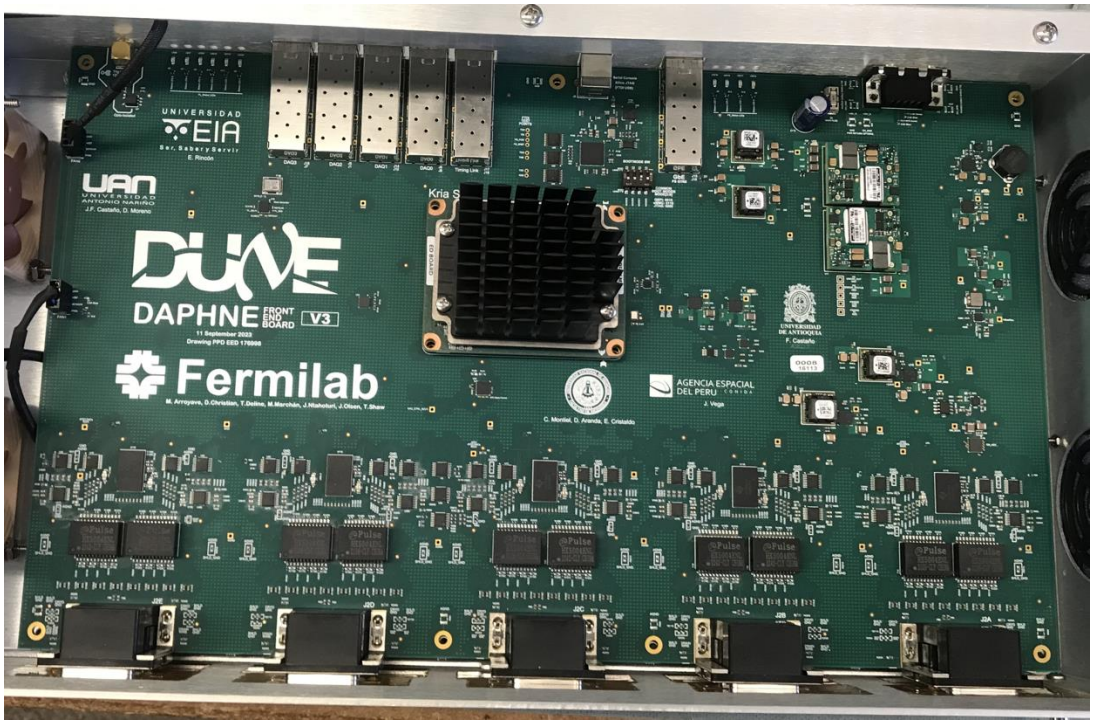
GitHub

DAPHNE V3

- Kria KR260 – DAPHNE V3

  Design of new architecture for Data Acquisition System

  - Acquisition of development boards for the collaboration



XILINX VITIS™

VIVADO™

https://www.hackster.io/fabioc9675/empowering-dune-setting-up-vivado-vitis-and-petalinux-b879ee

PetaLinux

GitHub

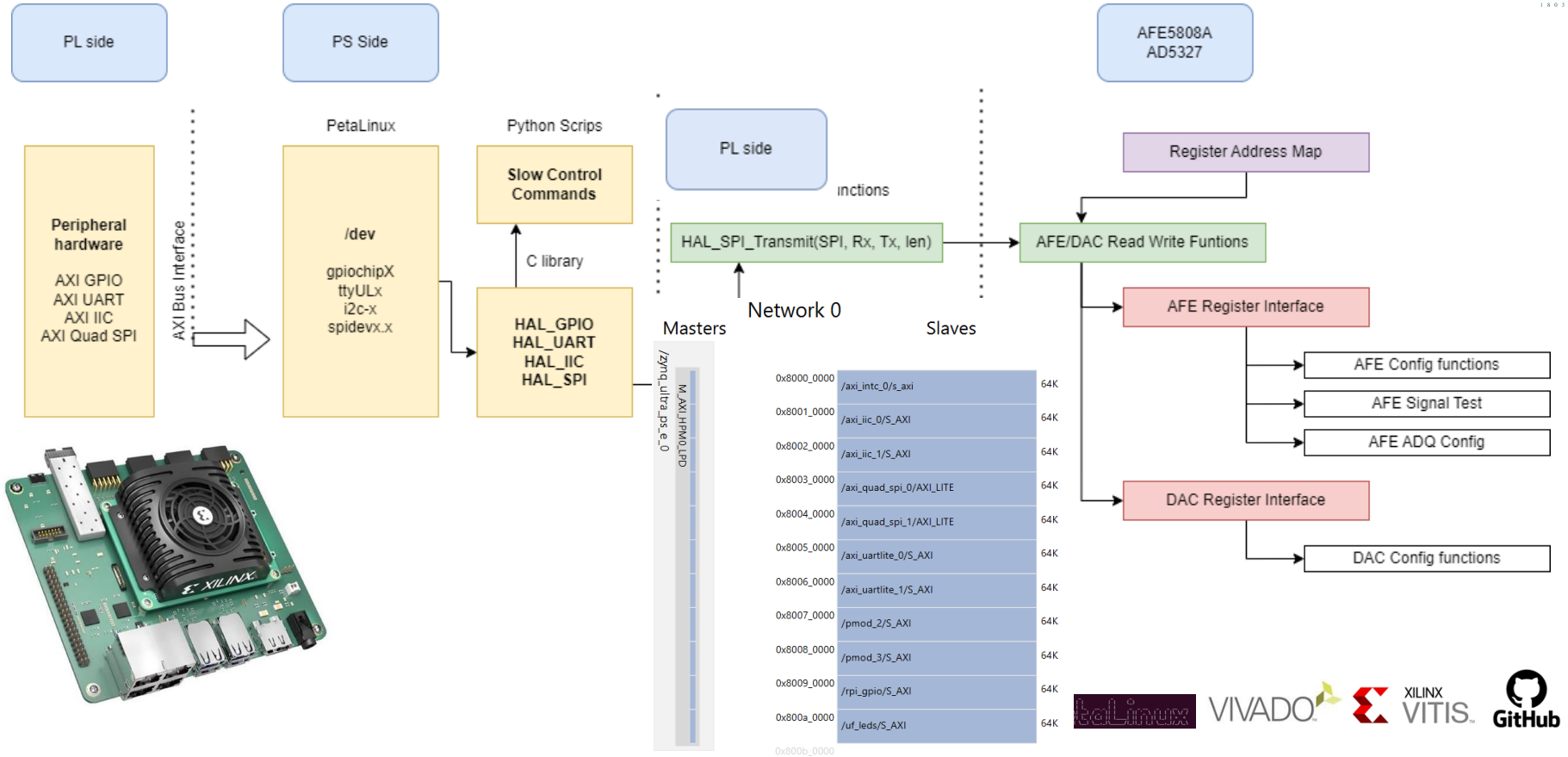
ICTP The Abdus Salam International Centre for Theoretical Physics

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ciencias Exactas y Naturales

DUNE

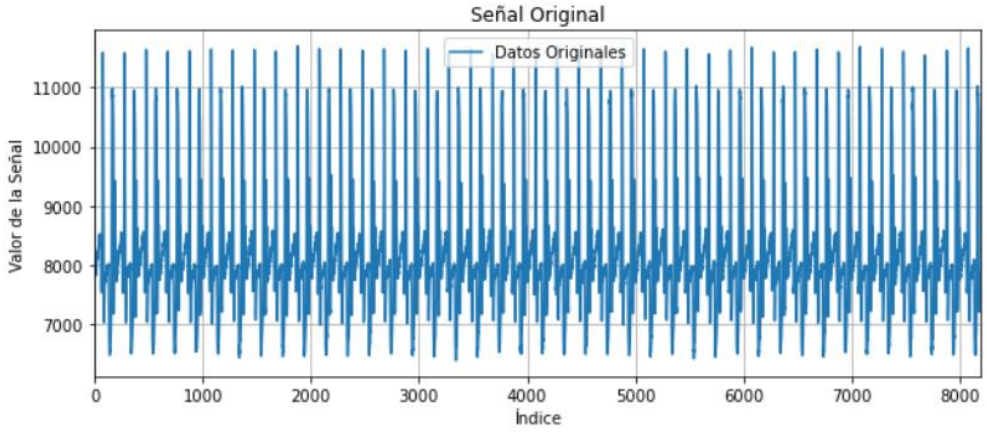
Co-Processor design

- DAPHNE v3 Slow Control Firmware architecture



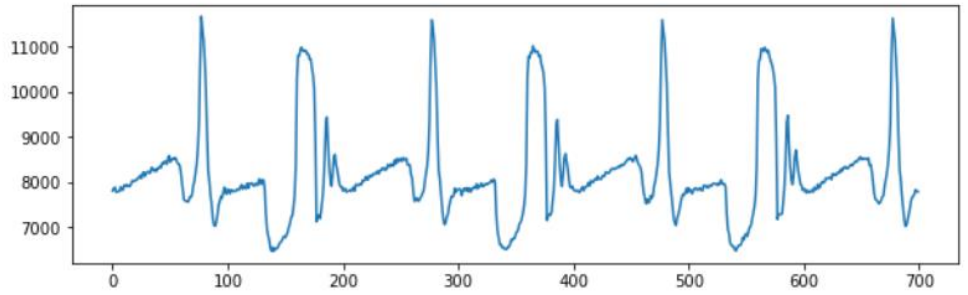
Alignment of AFE data out

- Data acquisition
  - Spy buffer alternative
- FiFo size
  - 8192 of 32 bits
  - 40 MHz
- Direct Memory Access
  - AXI Stream DMA




```
In [65]: plt.figure(figsize=(10, 3))
         plt.plot(data[6000:6700])

Out[65]: [<matplotlib.lines.Line2D at 0xffff78849970>]
```



https://github.com/fabioc9675/DUNE_Daphne_v3_AFE/blob/devFabianAFEHOG_ipcore/Petalinux/file_transfer/AFE_Test.ipynb

Demonstration

- Repository:

https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop

Embedded Linux in SoC-FPGA

- Tutorials
  - Platform config

    https://www.hackster.io/fabioc9675/empowering-dune-setting-up-vivado-vitis-and-petalinux-b879ee
  - Project initialization

    https://www.hackster.io/fabioc9675/empowering-dune-a-guide-to-starting-with-kria-kr260-vivado-fe39c5
  - PetaLinux image compile

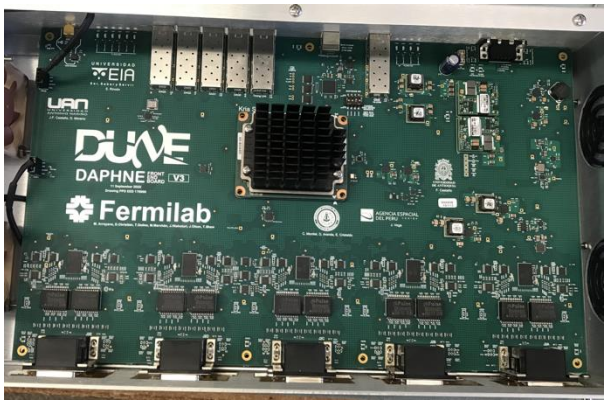
    https://www.hackster.io/fabioc9675/empowering-dune-creating-petalinux-2022-2-os-image-for-kria-2fbca1
  - Hardware support

    https://www.hackster.io/fabioc9675/empowering-dune-support-for-hard-peripherals-rpi-pmod-kria-505ded
  - AXI UARTLite

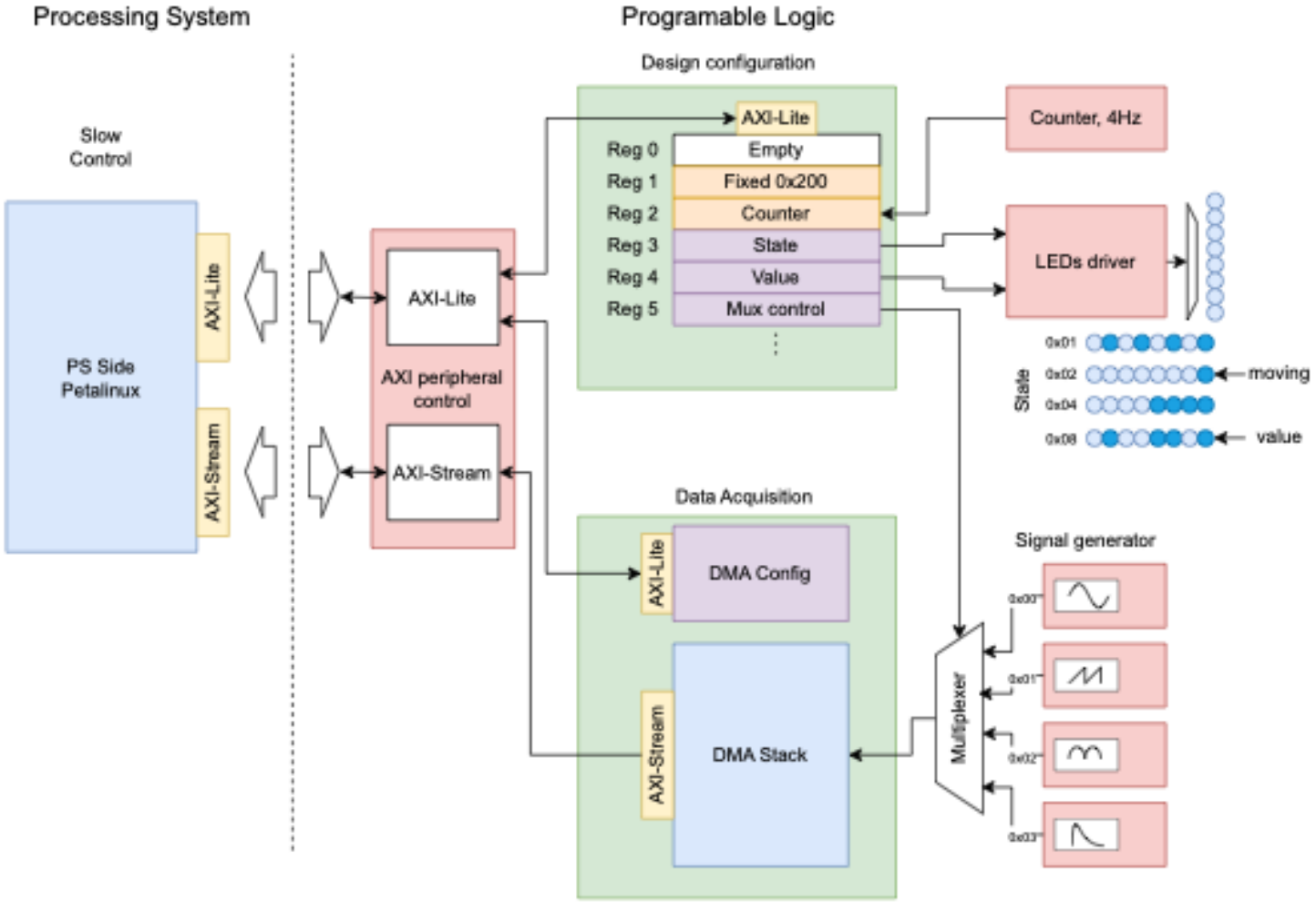
    https://www.hackster.io/fabioc9675/empowering-dune-axi-uartlite-using-petalinux-2022-2-in-kria-7e11cf
  - AXI IIC

    https://www.hackster.io/fabioc9675/empowering-dune-axi-iic-using-petalinux-2022-2-in-kria-a71f25
  - AXI Quad SPI

    https://www.hackster.io/fabioc9675/empowering-dune-axi-quad-spi-using-petalinux-2022-2-in-kria-ad6d72



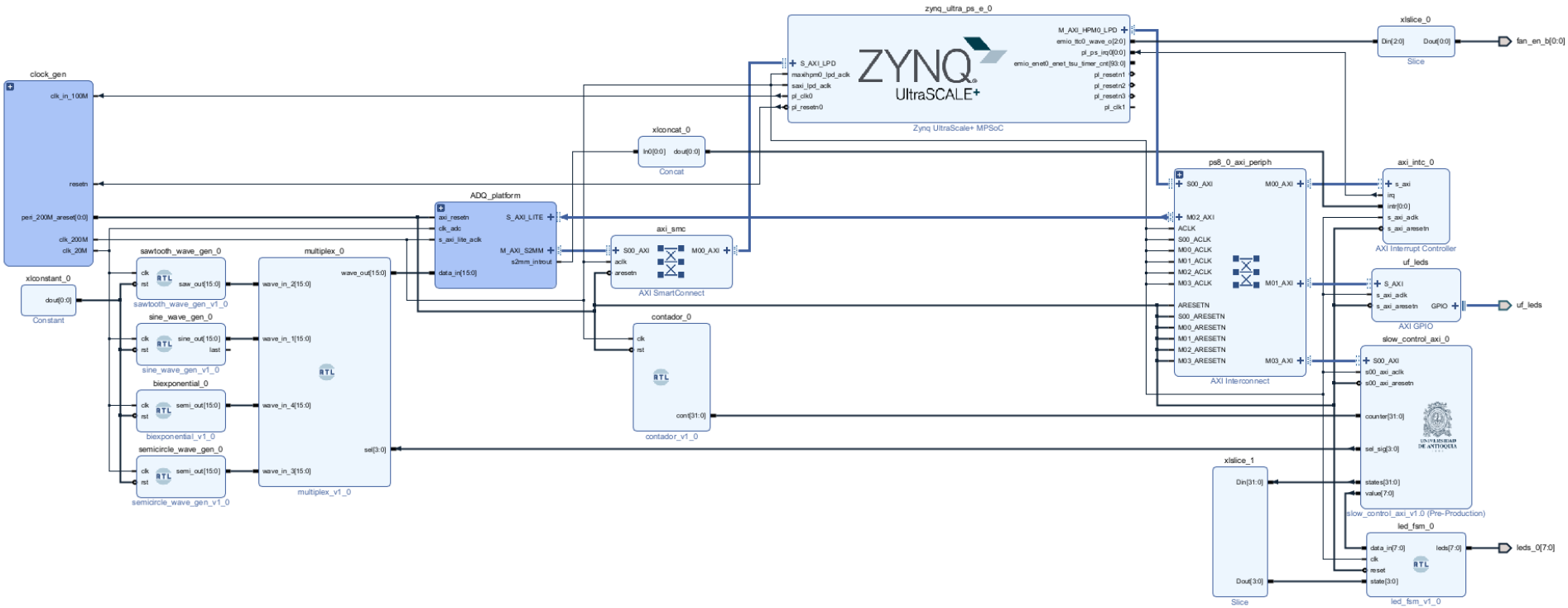
Application Example, basic slow control



Block Design, Vivado

- Application Example (Simple Slow Control Application)

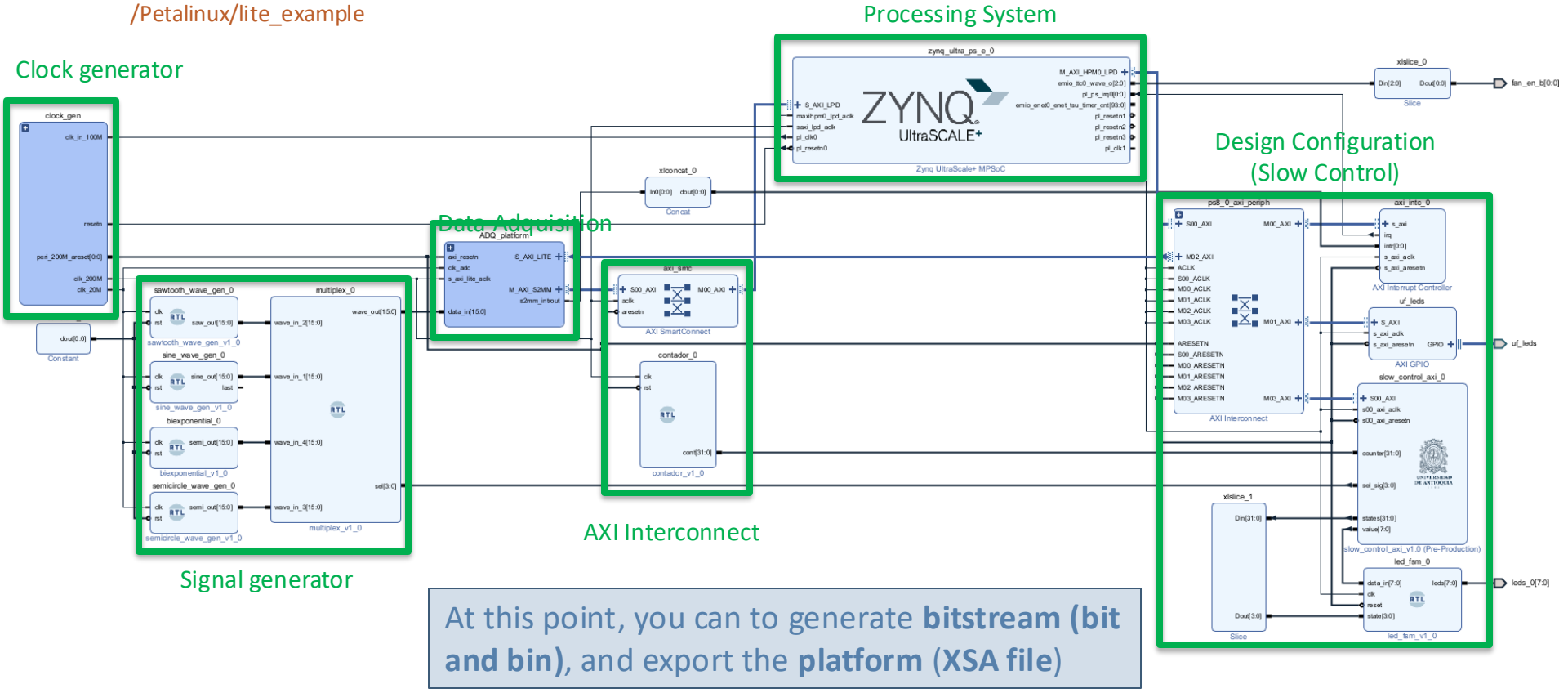
  https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/tree/main/Petalinux/lite_example



Block Design, Vivado

- Application Example (Simple Slow Control Application)

  https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/tree/main/Petalinux/lite_example



At this point, you can to generate **bitstream (bit and bin)**, and export the **platform** (**XSA file**)

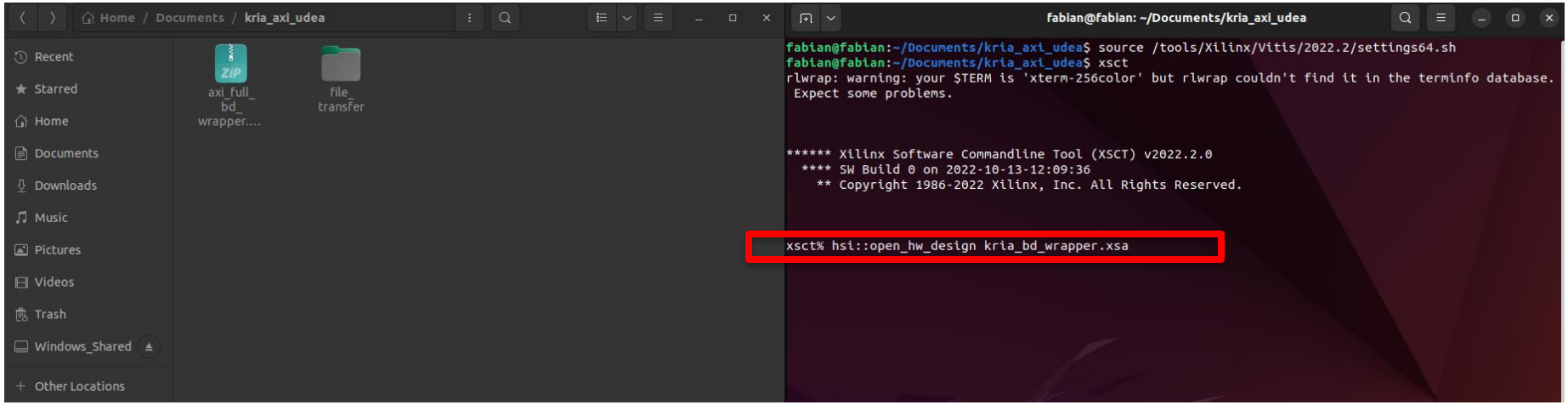
Generate Device Tree Overlay

- This step needs to be done on **Linux OS**, due to the **petalinux** is just officially supported this OS

```
source /tools/Xilinx/Vitis/2022.1/settings64.sh
xsct

xsct% hsi::open_hw_design kria_base.xsa
```

https://www.hackster.io/fabioc9675/empowering-dune-support-for-hard-peripherals-rpi-pmod-kria-505ded

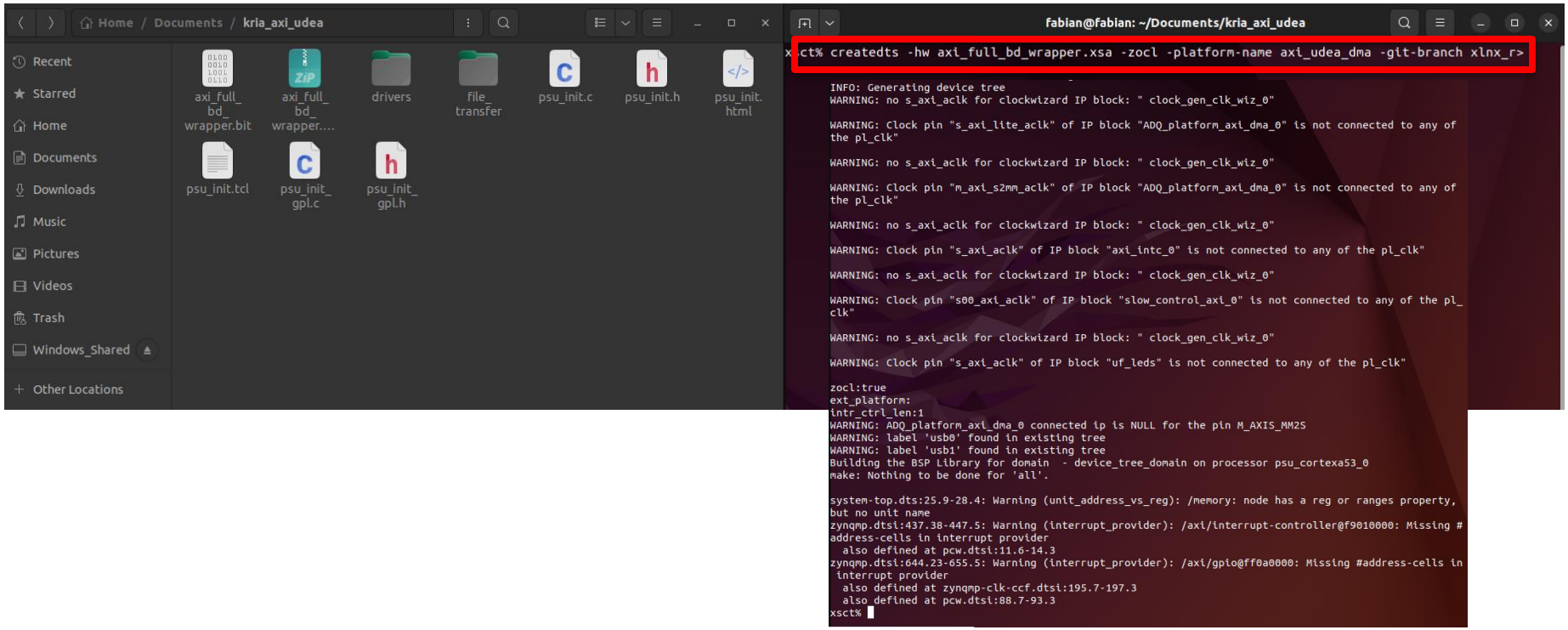


Generate Device Tree Overlay

- This step needs to be done on **Linux OS**, due to the **petalinux** is just officially supported this OS

https://www.hackster.io/fabioc9675/empowering-dune-support-for-hard-peripherals-rpi-pmod-kria-505ded

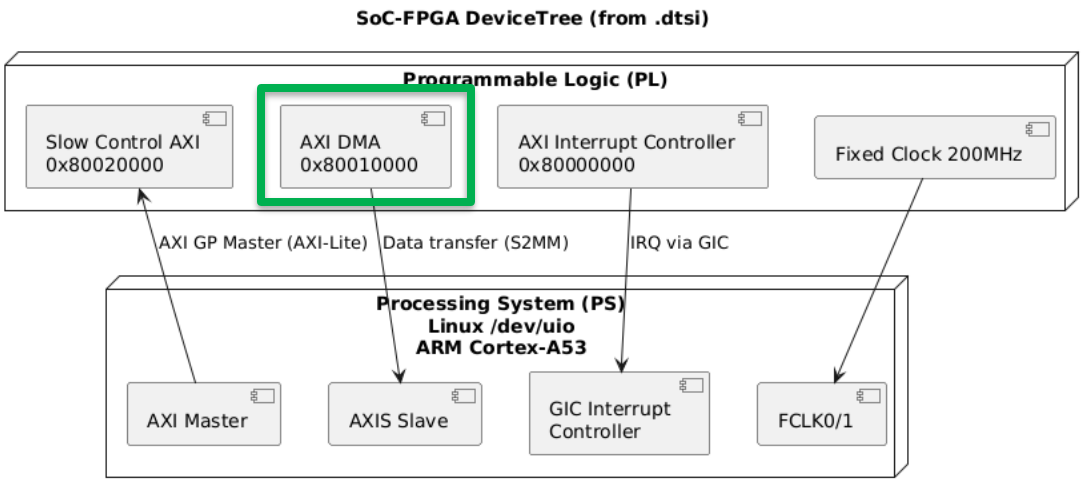
```
xsct% createdts -hw kria_base.xsa -zocl -platform-name kria_kr260 -git-branch xlnx_rel_v2022.2 -overlay -compile -out ./dtg_kr260_v0
```



Generate Device Tree Overlay

- This step needs to be done on **Linux OS**, due to the **petalinux** is just officially supported this OS

https://www.hackster.io/fabioc9675/empowering-dune-support-for-hard-peripherals-rpi-pmod-kria-505ded

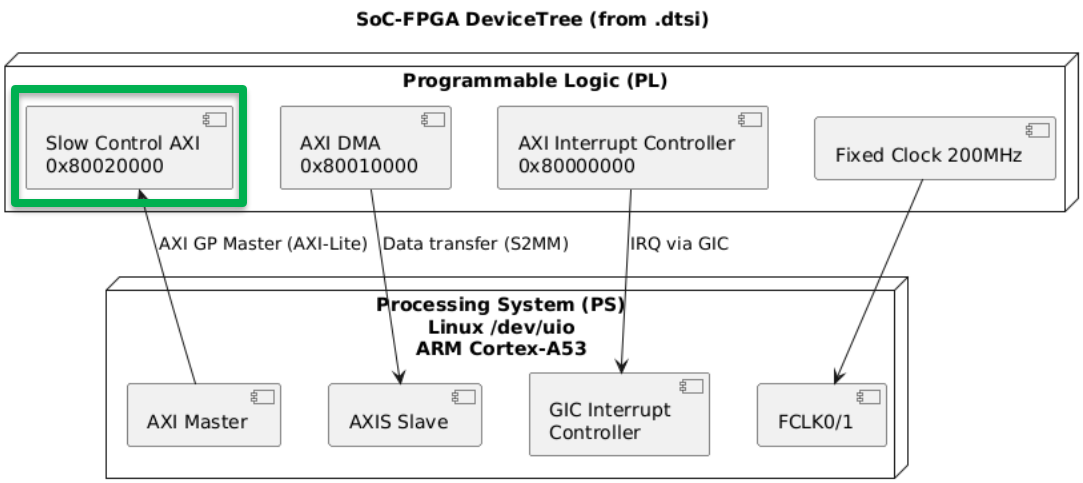



```
fragment@2 {
 target = <&amba>;
 overlay2: __overlay__ {
  #address-cells = <2>;
  #size-cells = <2>;
  ADQ_platform_axi_dma_0: dma@80010000 {
   #dma-cells = <1>;
   clock-names = "s_axi_lite_aclk",
          "m_axi_s2mm_aclk";
   clocks = <&misc_clk_0>, <&misc_clk_0>;
   compatible = "xlnx,axi-dma-7.1",
          "xlnx,axi-dma-1.00.a";
   interrupt-names = "s2mm_introut";
   interrupt-parent = <&axi_intc_0>;
   interrupts = <0 2>;
   reg = <0x0 0x80010000 0x0 0x10000>;
   xlnx,addrwidth = <0x20>;
   xlnx,sg-length-width = <0x10>;
   dma-channel@80010030 {
    compatible = "xlnx,axi-dma-s2mm-channel";
    dma-channels = <0x1>;
    interrupts = <0 2>;
    xlnx,datawidth = <0x20>;
    xlnx,device-id = <0x0>;
   };
  };
  misc_clk_0: misc_clk_0 {...};
  slow_control_axi_0: slow_control_axi@80020000
  {...};
  uf_leds: gpio@800a0000 {...};
 };
};
```

Generate Device Tree Overlay

- This step needs to be done on **Linux OS**, due to the **petalinux** is just officially supported this OS

https://www.hackster.io/fabioc9675/empowering-dune-support-for-hard-peripherals-rpi-pmod-kria-505ded




```
fragment@2 {
 target = <&amba>;
 overlay2: __overlay__ {
  #address-cells = <2>;
  #size-cells = <2>;
  ADQ_platform_axi_dma_0: dma@80010000 {...};
  misc_clk_0: misc_clk_0 {...};
  low_control_axi_0: slow_control_axi@80020000
  {
   /* This is a place holder node for a
      custom IP, user may need to update
      the entries */
   clock-names = "s00_axi_aclk";
   clocks = <&misc_clk_0>;
   compatible = "xlnx,slow-control-axi-1.0";
   reg = <0x0 0x80020000 0x0 0x10000>;
   xlnx,s00-axi-addr-width = <0x6>;
   xlnx,s00-axi-data-width = <0x20>;
  };
  uf_leds: gpio@800a0000 {...};
 };
};
```

At this point, you can to compile the **.dtsi** and generate the **.dtbo**, also upload the **.dtbo**, **.bit.bin** and **.json** files to the **petalinux** system

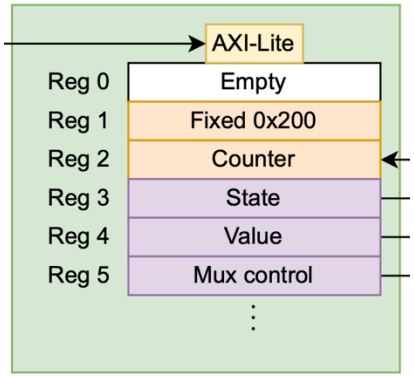
```
sudo xmutil loadapp <custom overlay>
```

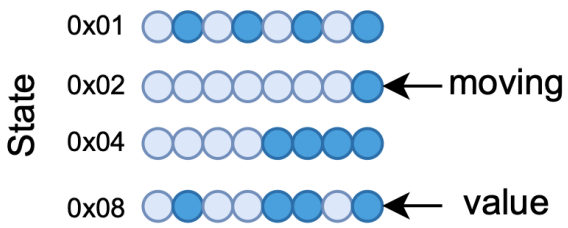
Demo

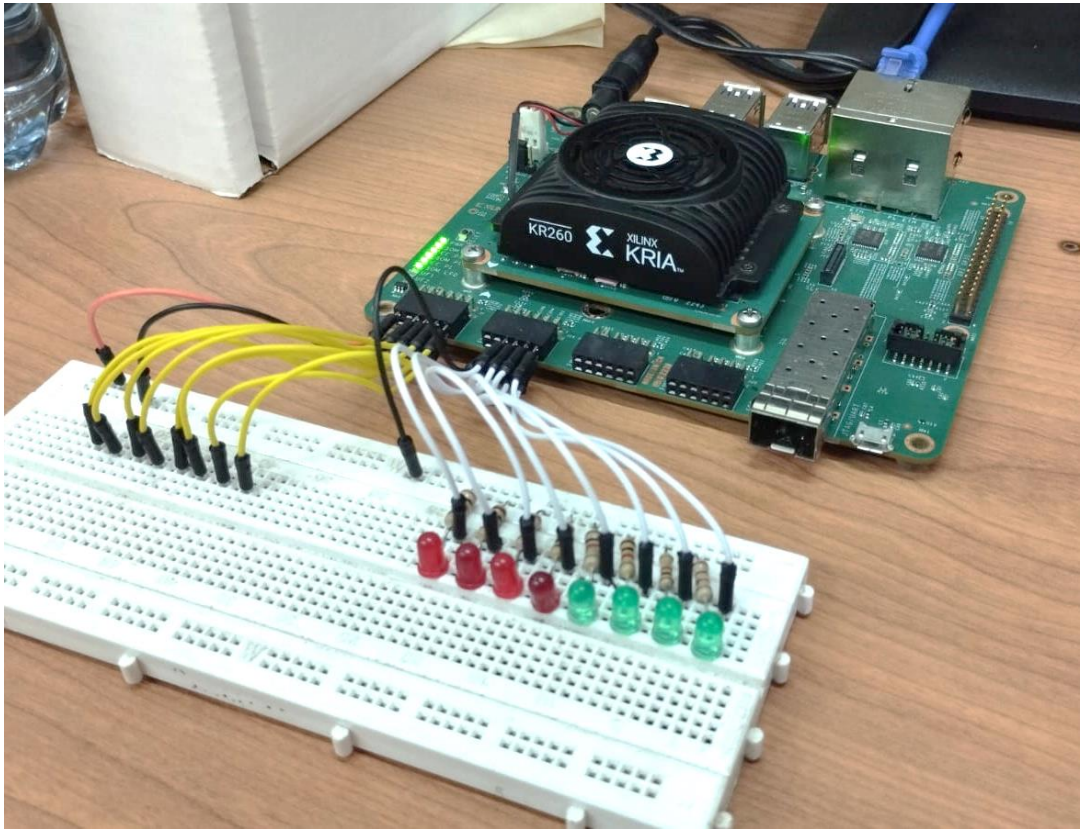
Interaction with hardware from BASH

```
sudo devmem 0x80020004

0x200
```

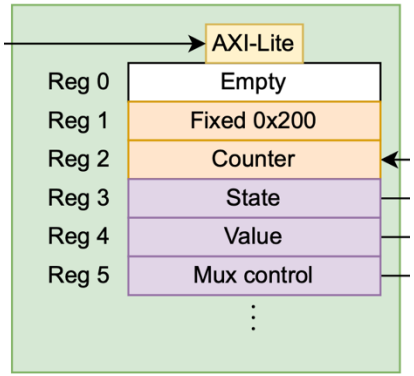


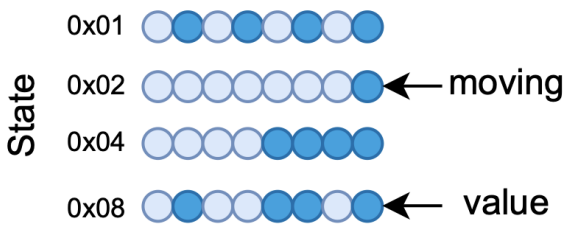


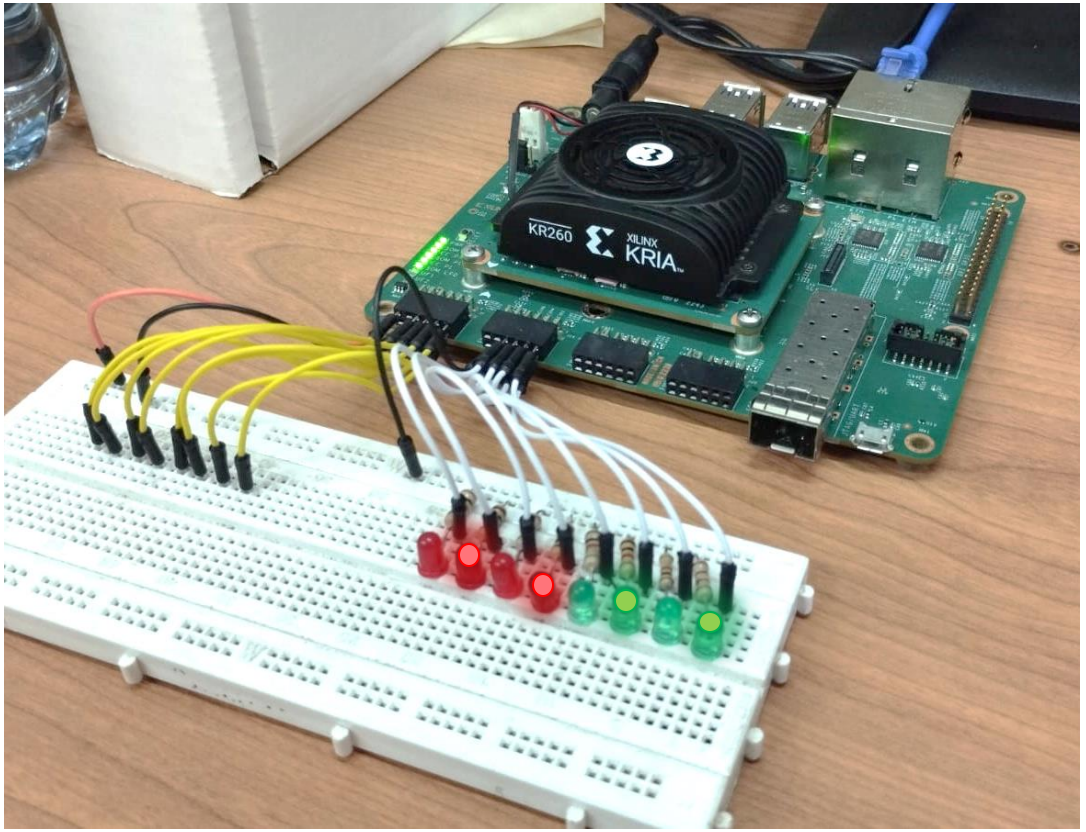


Interaction with hardware from BASH

```
sudo devmem 0x8002000C 32 0x01
```

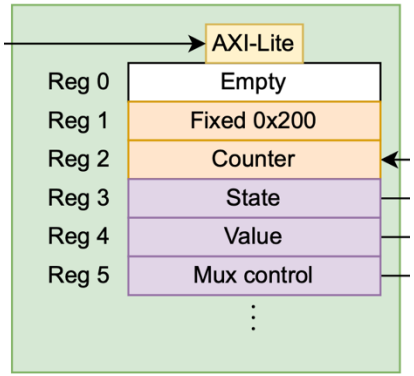


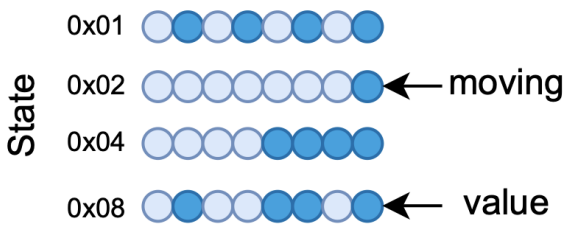


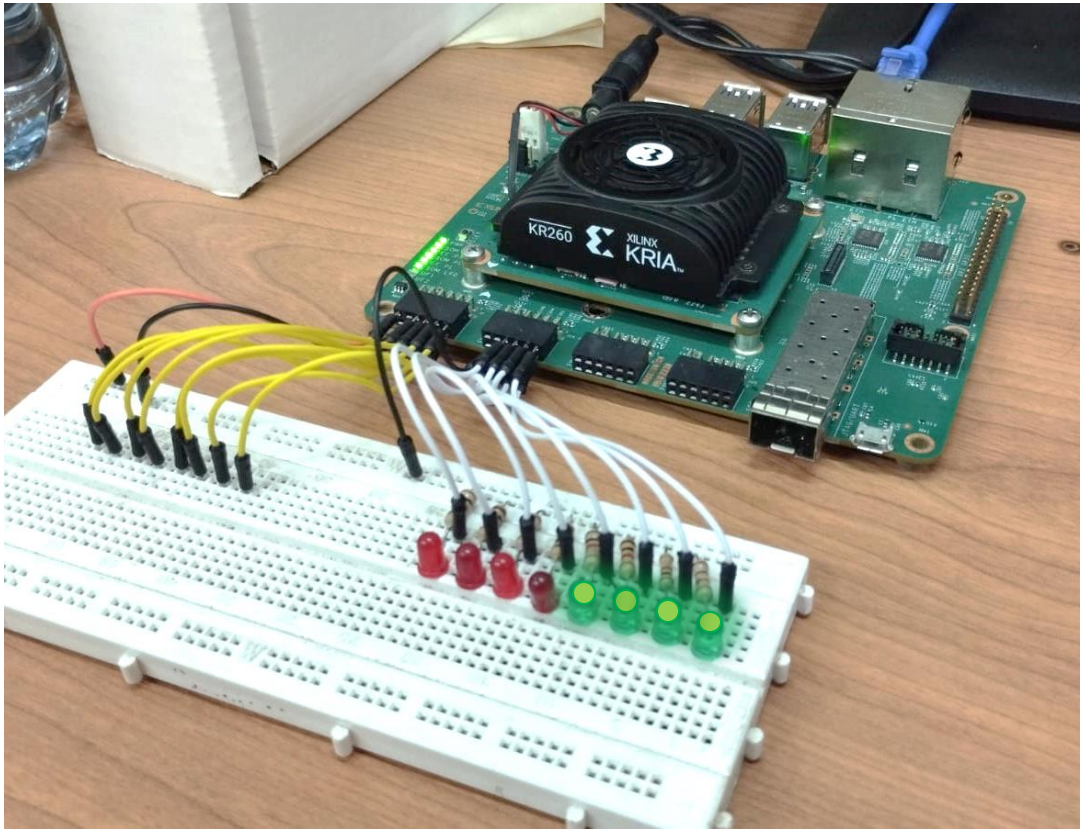


Interaction with hardware from BASH

```
sudo devmem 0x8002000C 32 0x04
```



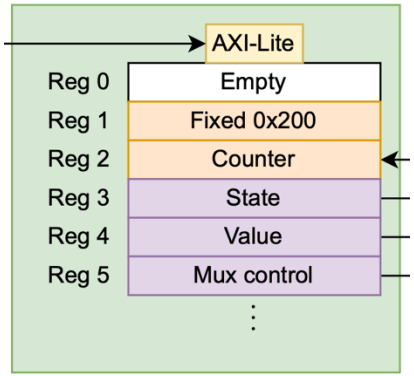


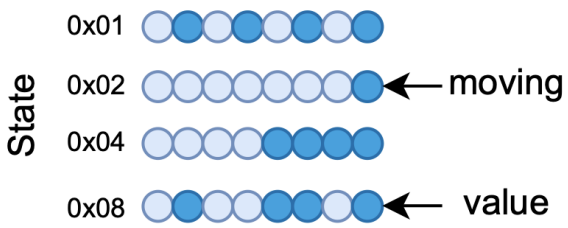


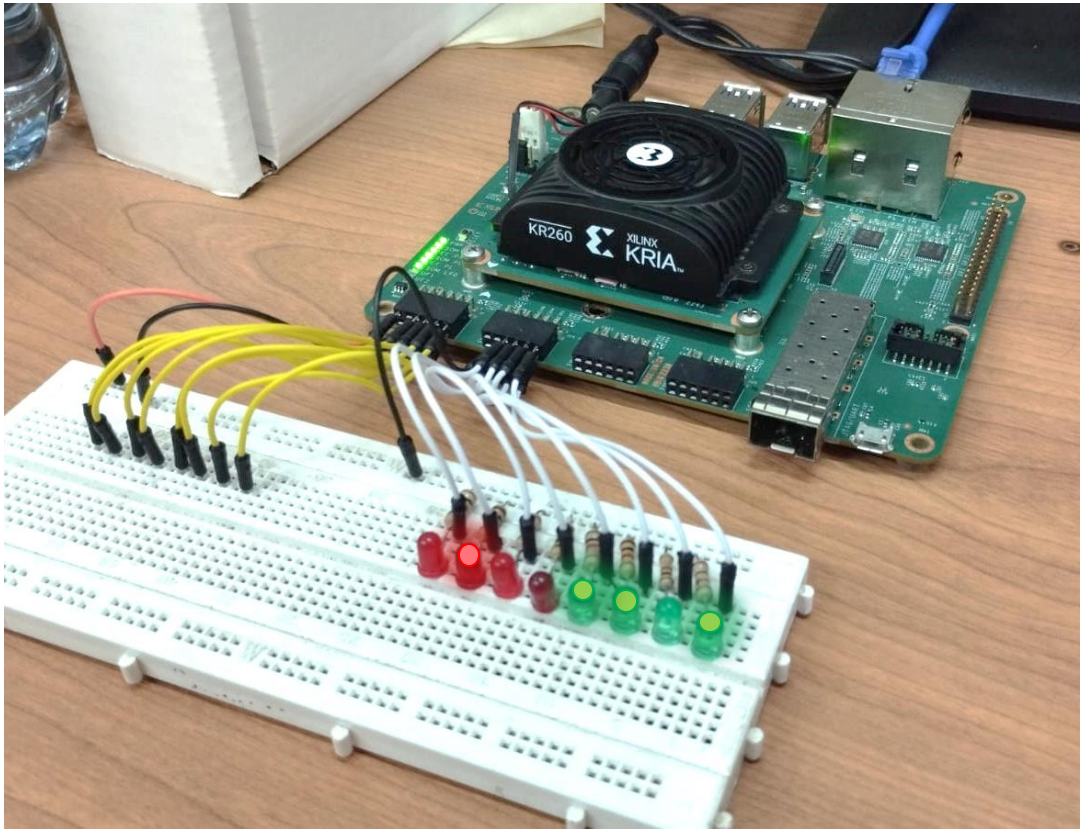
Interaction with hardware from BASH

```
sudo devmem 0x8002000C 32 0x08

sudo devmem 0x80020010 32 0x4D
```







C driver (.so) and python interaction

- It is possible to write a C driver to communicate with the device

```
uint32_t axi_read_reg(uintptr_t base_addr, uint32_t offset) {
  int mem_fd;
  void *mapped_base;
  uint32_t value = 0xFFFFFFFFU;

  long page_size = sysconf(_SC_PAGESIZE);
  if (page_size <= 0) {
    perror("sysconf(_SC_PAGESIZE)");
    return value;
  }

  mem_fd = open("/dev/mem", O_RDONLY | O_SYNC);
  if (mem_fd < 0) {
    perror("open(/dev/mem)");
    return value;
  }

  uintptr_t page_addr = (base_addr + offset) & ~(page_size - 1);
  uintptr_t page_offset = (base_addr + offset) - page_addr;

  mapped_base = mmap(NULL, page_size, PROT_READ, MAP_SHARED, mem_fd, page_addr);
  if (mapped_base == MAP_FAILED) {
    perror("mmap");
    close(mem_fd);
    return value;
  }

  volatile uint32_t* reg_ptr = (volatile uint32_t*)((uint8_t*)mapped_base + page_offset);
  value = *reg_ptr;

  munmap(mapped_base, page_size);
  close(mem_fd);

  return value;
```

https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/tree/main/Petalinux/lite_example

https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/blob/main/Petalinux/CodeTest/dma_ctypes.ipynb

Open the device

Map the memory address

Read the memory space and close

C driver (.so) and python interaction

- It is possible to write a C driver to communicate with the device

```
void axi_write_reg(uintptr_t base_addr, uint32_t offset, uint32_t value) {
  int mem_fd;
  void *mapped_base;

  long page_size = sysconf(_SC_PAGESIZE);
  if (page_size <= 0) {
   perror("sysconf(_SC_PAGESIZE)");
   return;
  }

  mem_fd = open("/dev/mem", O_RDWR | O_SYNC);
  if (mem_fd < 0) {
   perror("open(/dev/mem)");
   return;
  }

  uintptr_t page_addr = (base_addr + offset) & ~(page_size - 1);
  uintptr_t page_offset = (base_addr + offset) - page_addr;

  mapped_base = mmap(NULL, page_size, PROT_READ | PROT_WRITE, MAP_SHARED, mem_fd, page_addr);
  if (mapped_base == MAP_FAILED) {
   perror("mmap");
   close(mem_fd);
   return;
  }

  volatile uint32_t* reg_ptr = (volatile uint32_t*)((uint8_t*)mapped_base + page_offset);
  *reg_ptr = value;
  __sync_synchronize();

  munmap(mapped_base, page_size);
  close(mem_fd);
```

https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/tree/main/Petalinux/lite_example

https://github.com/fabioc9675/ICTP_Embedded_Linux_workshop/blob/main/Petalinux/CodeTest/dma_ctypes.ipynb

Open the device

Map the memory address

Write the memory space

Demo

Thank you!

Support slides

How would you design this?

- How would you define the architecture of the following systems?
  - Implementation of 100 MSps Oscilloscope with filters selection
  - Control of robotic arm system with PID
  - Control and configuration of high performance multi-stack DAQ

How would you design this?

- Control of robotic arm system with PID

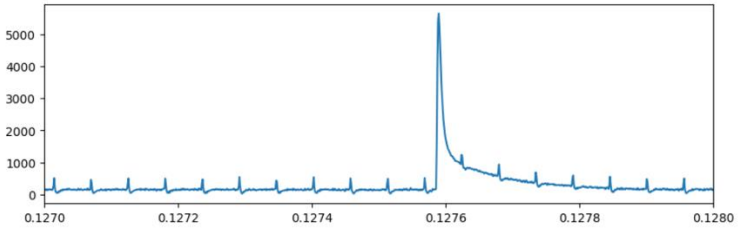
PID control calculation	Motor control signals	Motor position sensor

Simulation model building

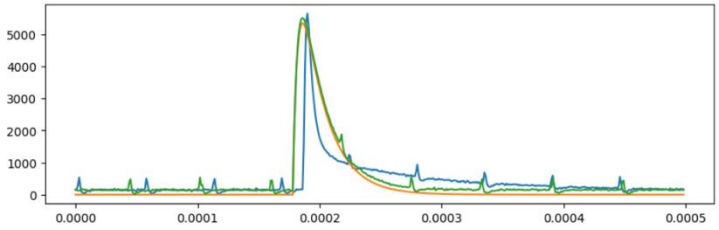
- Biexponential model (Radiation events' model)

$$y(t) = A(e^{(t-t_0)/\tau_D} - e^{(t-t_0)/\tau_R})$$

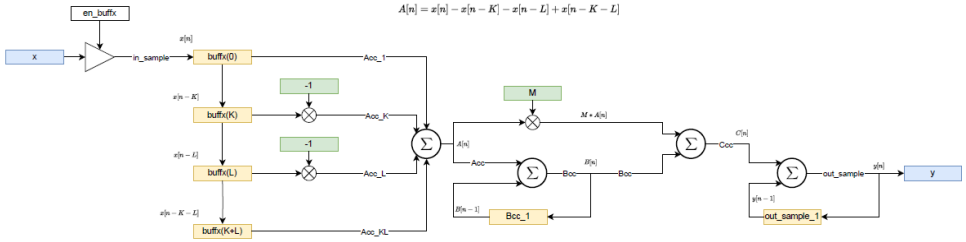
- Photon Event sampled (SiPM)



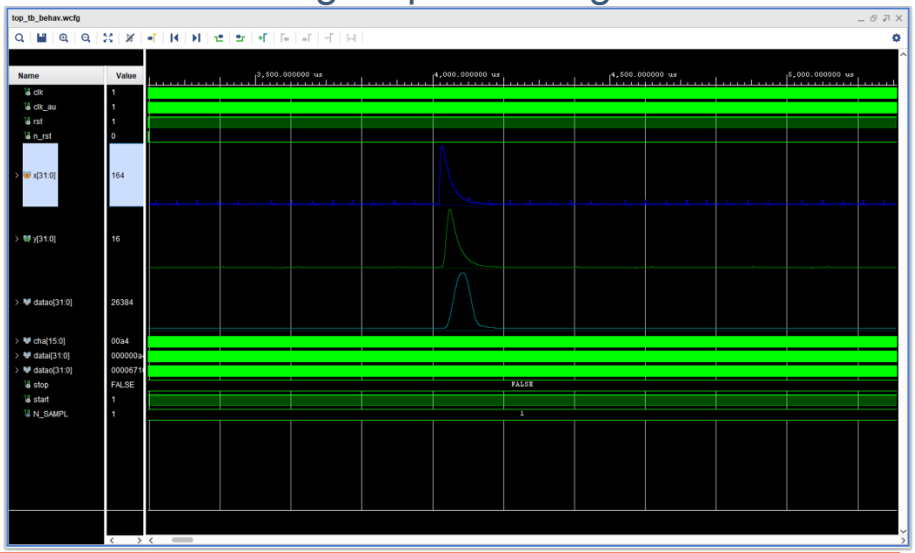
- Baseline noise extraction and event model fitting



- Trapezoidal Shaper Filter and DPP implementation

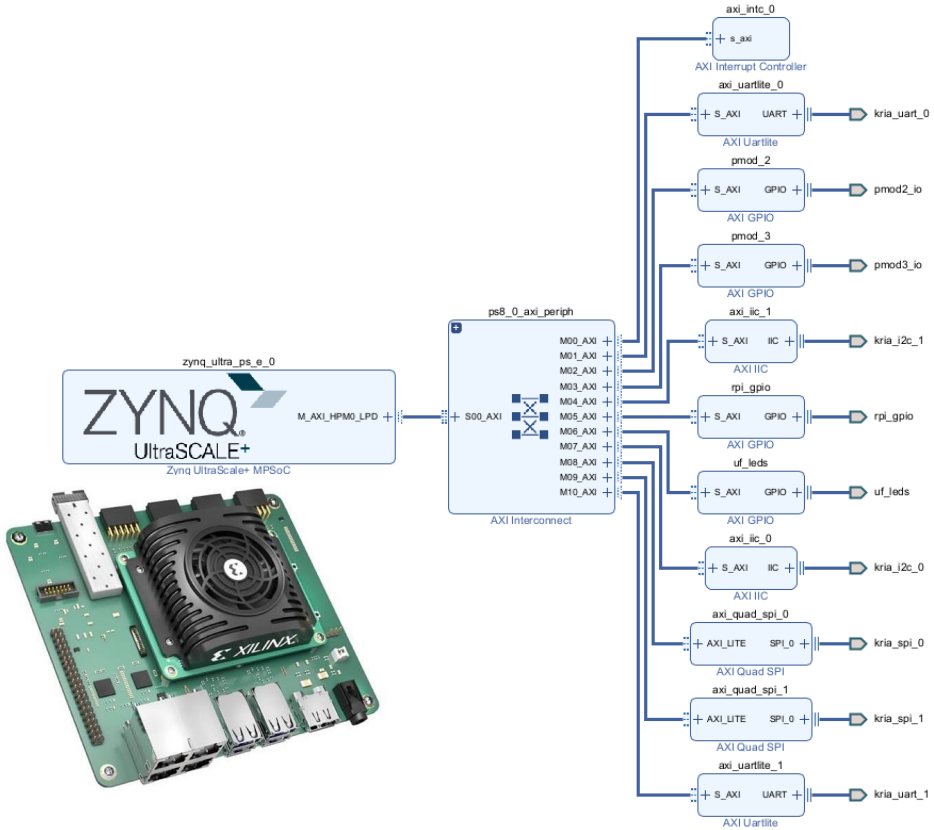


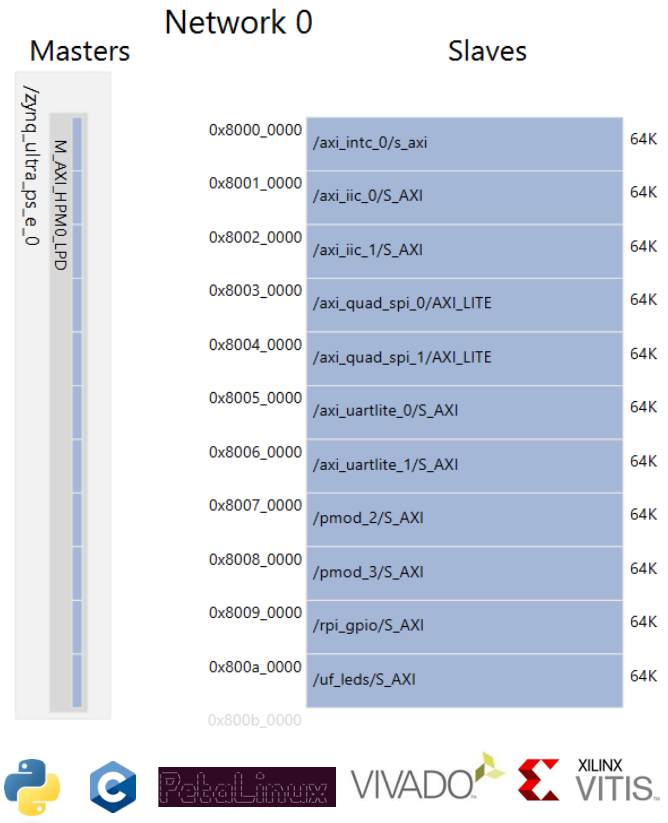
- Event signal processing



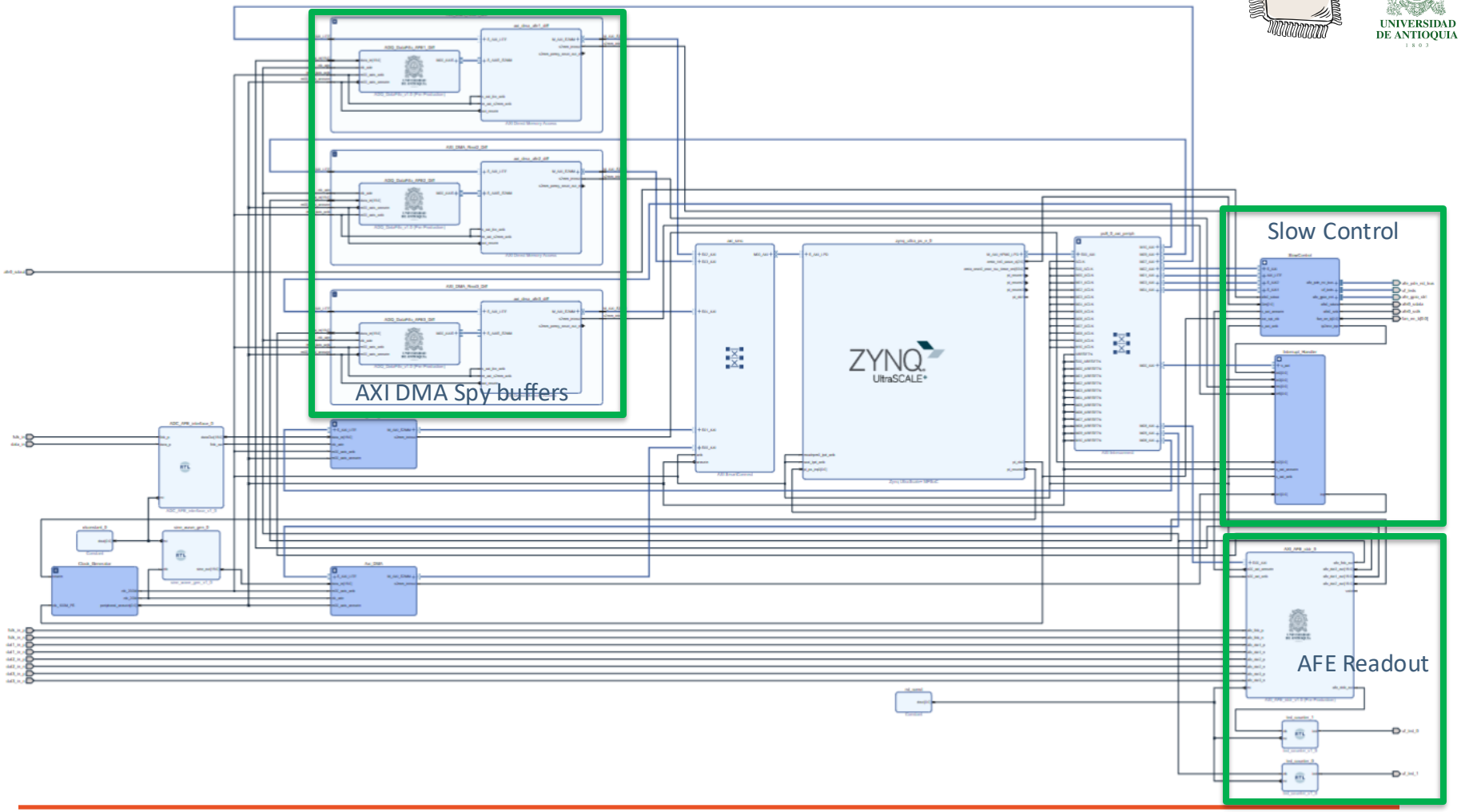
Co-Processor design

- Vivado Block Design



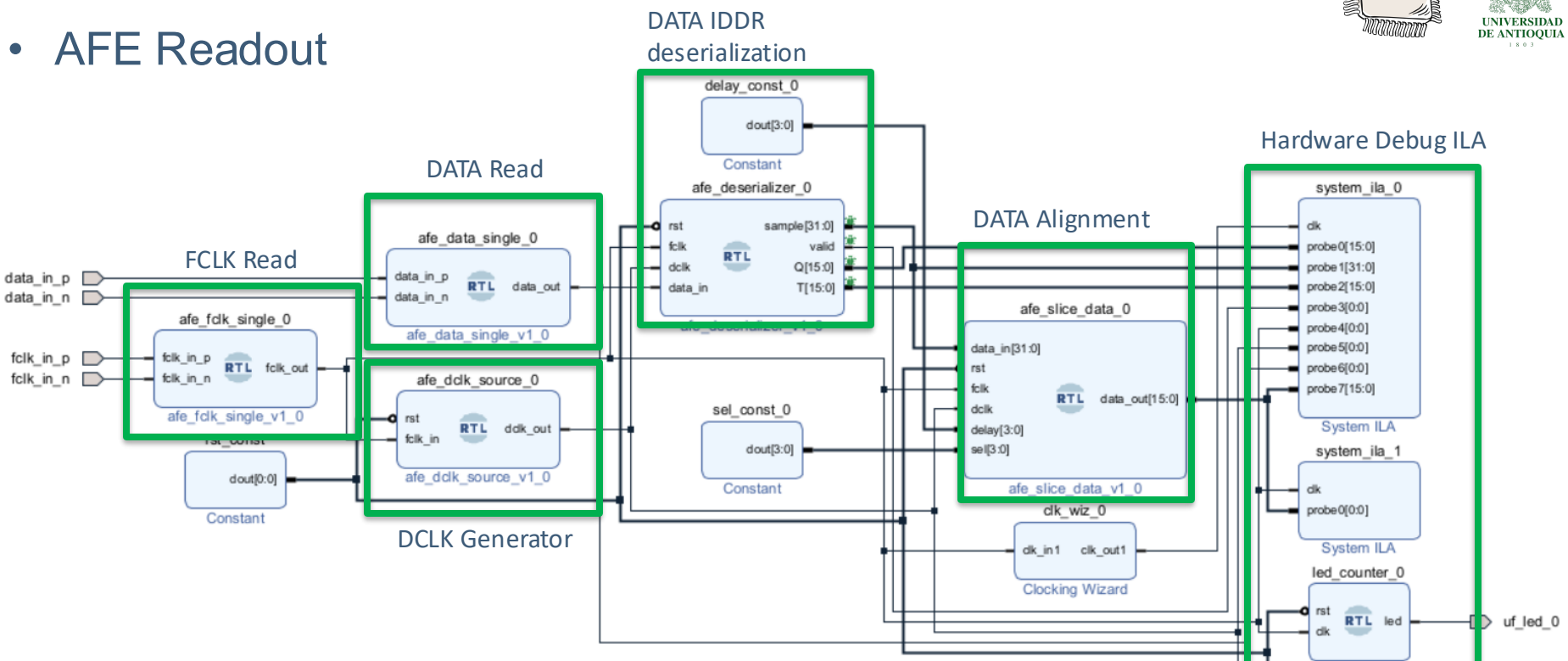


Hardware Block Design



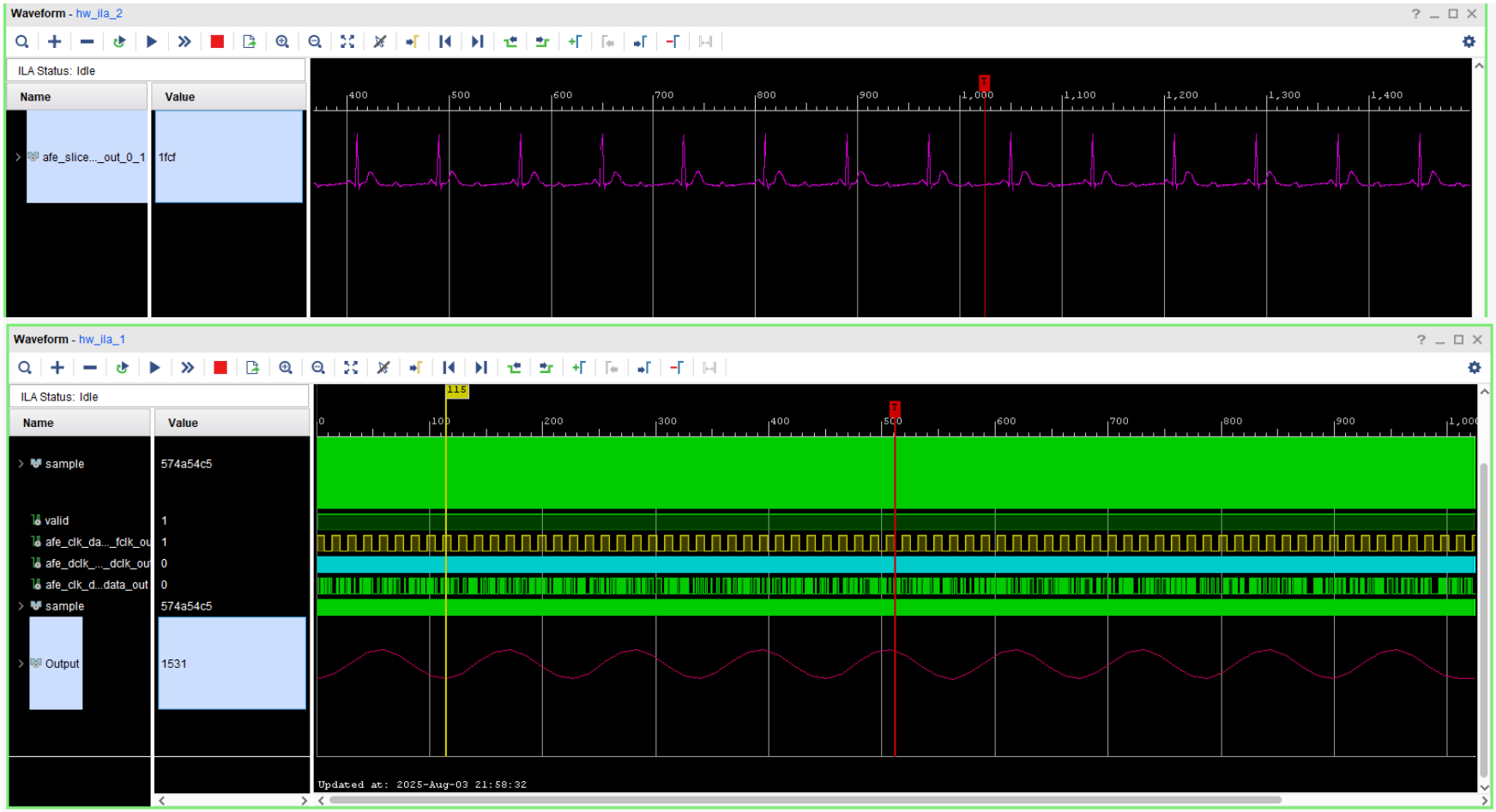
AFE Readout IPCore Custom

- AFE Readout



Name	CLB LUTs (117120)	CLB Registers (234240)	CARRY8 (14640)	F7 Muxes (58560)	F8 Muxes (29280)	CLB (14640)	LUT as Logic (117120)	LUT as Memory (57600)	Block RAM Tile (144)	Bonded IOB (189)	HPIOB_M (58)	HPIOB_S (58)	HDIOB_M (35)	HDIOB_S (35)	HPIOBDIF FINBUF (82)	BITSLICE_RX_TX (1248)	GLOBAL CLOCK BUFFERs (352)	MMCM (4)	PS8 (1)
AXI_AFE_iddr_0 (kria	137	241	0	0	0	34	137	0	0	0	0	0	0	0	4	3	3	1	0
Axi_DMA (Axi_DMA_in	857	1259	20	0	0	209	780	77	10.5	0	0	0	0	0	0	0	0	0	0
AXI_DMA_Real (AXI_[	853	1259	20	0	0	215	776	77	10.5	0	0	0	0	0	0	0	0	0	0
AXI_DMA_Real1_Diff	859	1259	20	0	0	209	782	77	10.5	0	0	0	0	0	0	0	0	0	0

DEBUG with ILA

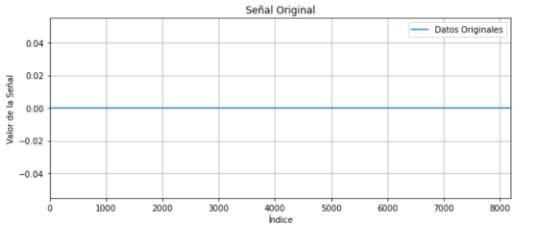


Alignment of AFE data out

- Alignment sequence test

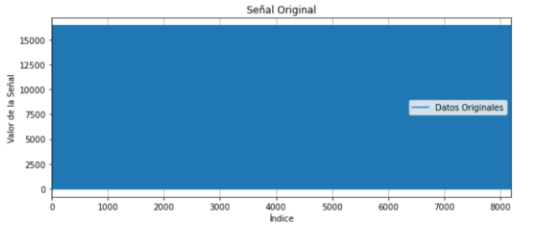
In [35]:
```
# prueba de todos ceros
my_library.HAL_AFE_All0sTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de ceros")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de ceros
['0000000000000', '0000000000000', '0000000000000', '0000000000000', '0000000000000',
-----------------------------------------
```



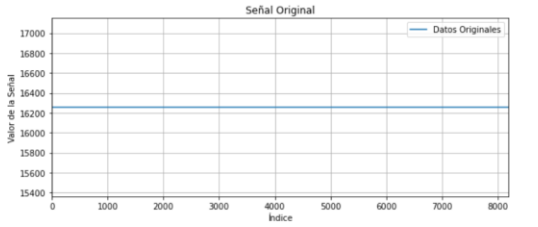
In [40]:
```
# prueba de toggle
my_library.HAL_AFE_ToggleTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de toggle")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de toggle
['1111111111111', '0000000000000', '1111111111111', '0000000000000', '1111111111111',
-----------------------------------------
```



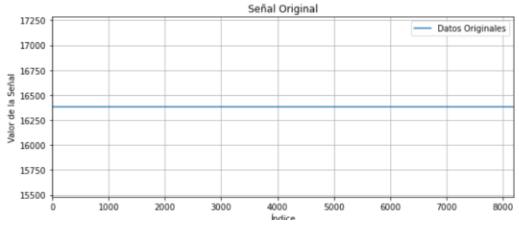
In [45]:
```
# prueba de Sync
my_library.HAL_AFE_SyncTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de sync")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de sync
['1111111000000', '1111111000000', '1111111000000', '1111111000000', '1111111000000',
-----------------------------------------
```



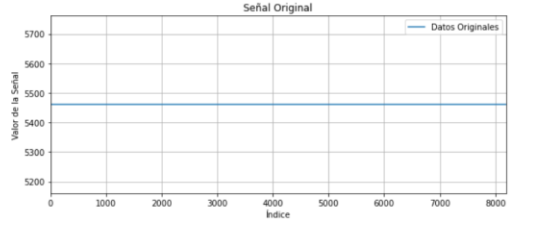
In [37]:
```
# prueba de todos unos
my_library.HAL_AFE_All1sTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de unos")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de unos
['1111111111111', '1111111111111', '1111111111111', '1111111111111', '1111111111111',
-----------------------------------------
```



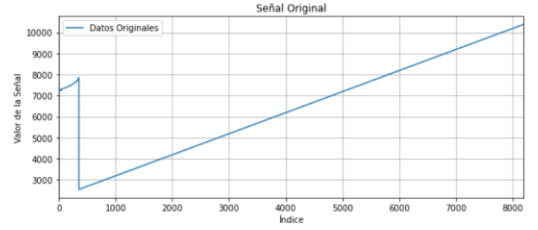
In [42]:
```
# prueba de deskew
my_library.HAL_AFE_SkewTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de deskew")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de deskew
['0101010101010', '0101010101010', '0101010101010', '0101010101010', '0101010101010',
-----------------------------------------
```



In [49]:
```
# prueba de ramp
my_library.HAL_AFE_RampTest(my_library.hafe0)
my_library.HAL_AFEWriteRegister(hafe0, 4, ctypes.byref((ctypes.c_uint16)(0x18)))
print("Prueba de ramp")
prueba_data()
print("-------------------------------------------------")
```

```
Prueba de ramp
['0110000010110', '0110000010110', '0110000010111', '0110000110000', '0110000110001',
-----------------------------------------
```



https://github.com/fabioc9675/DUNE_Daphne_v3_AFE/blob/devFabianAFEHOG_ipcore/Petalinux/file_transfer/AFE_Test.ipynb

C driver (.so) and python interaction

- It is possible to write a C driver to communicate with the device

In [14]:
```
import ctypes

# Carga la librería compilada
axi = ctypes.CDLL("./libaxi_mmap.so")

# Define tipos de argumentos y retorno
axi.axi_read_reg.argtypes = [ctypes.c_ulong, ctypes.c_uint]
axi.axi_read_reg.restype = ctypes.c_uint

axi.axi_write_reg.argtypes = [ctypes.c_ulong, ctypes.c_uint, ctypes.c_uint]
axi.axi_write_reg.restype = None

# Dirección base del bloque AXI
BASE_ADDR = 0x80020000
```

In [15]:
```
# Leer un registro
value = axi.axi_read_reg(BASE_ADDR, 0x04)
print(f"Valor leído: 0x{value:08X}")
```

```
Valor leído: 0x00000200
```

In [16]:
```
# Leer un registro
value = axi.axi_read_reg(BASE_ADDR, 0x08)
print(f"Valor leído: 0x{value:08X}")
```

```
Valor leído: 0x00003CD4
```

In [18]:
```
axi.axi_write_reg(BASE_ADDR, 0x0C, 1)

value = axi.axi_read_reg(BASE_ADDR, 0x0C)
print(f"Registro escrito con 0x{value:08X}")
```

```
Registro escrito con 0x00000001
```

In [19]:
```
axi.axi_write_reg(BASE_ADDR, 0x0C, 2)

value = axi.axi_read_reg(BASE_ADDR, 0x0C)
print(f"Registro escrito con 0x{value:08X}")
```

```
Registro escrito con 0x00000002
```

In [20]:
```
axi.axi_write_reg(BASE_ADDR, 0x0C, 4)

value = axi.axi_read_reg(BASE_ADDR, 0x0C)
print(f"Registro escrito con 0x{value:08X}")
```

```
Registro escrito con 0x00000004
```

In [21]:
```
axi.axi_write_reg(BASE_ADDR, 0x0C, 8)

value = axi.axi_read_reg(BASE_ADDR, 0x0C)
print(f"Registro escrito con 0x{value:08X}")

axi.axi_write_reg(BASE_ADDR, 0x10, 0b11001010)

value = axi.axi_read_reg(BASE_ADDR, 0x10)
print(f"Registro escrito con 0x{value:08b}")
```

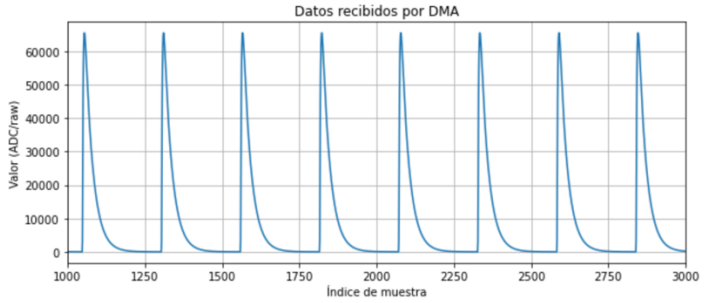
File display

```
Registro escrito con 0x00000008
Registro escrito con 0x11001010
```

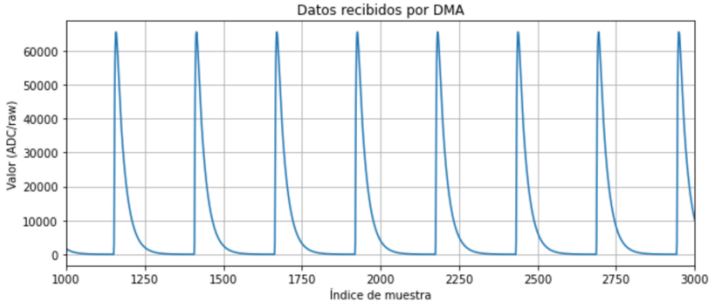
Reading a DMA example

https://github.com/fabioc9675/ICTP_DMA_Yocto_Prj/tree/ICTP_Presentation/Petalinux/CodeTest

```
Opening handle via /dev/mem (C lib)...
Initial STATUS (from C lib) = 0x00005011
Starting DMA (C lib)...
DMA finished OK (C lib)
Tiempo total DMA (Python mmap): 6.915 ms
Tiempo total DMA (Python mmap): 0.118 ms
First 16 words: [    0     0     0     0 10437 10036  9650  9279  8922  8579  8249  7932
  7627  7333  7051  6780]
```

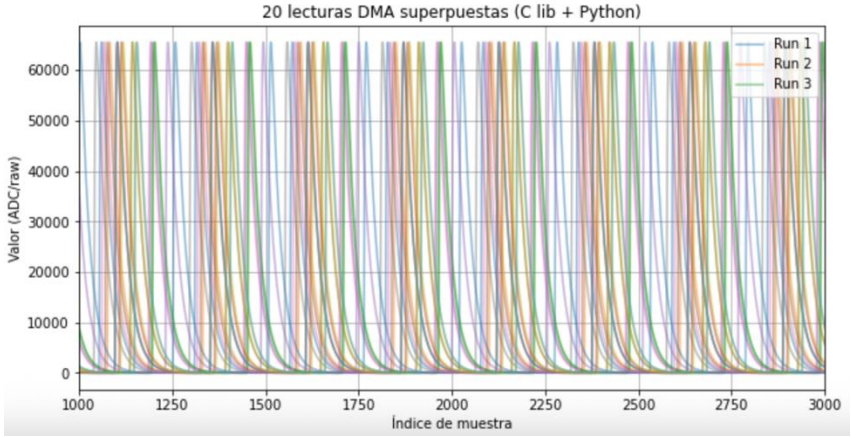



```
Initial STATUS (from C lib) = 0x00005011
Starting DMA (C lib)...
DMA finished OK (C lib)
Tiempo total DMA (Python mmap): 4.320 ms
Tiempo total DMA (Python mmap): 0.123 ms
```



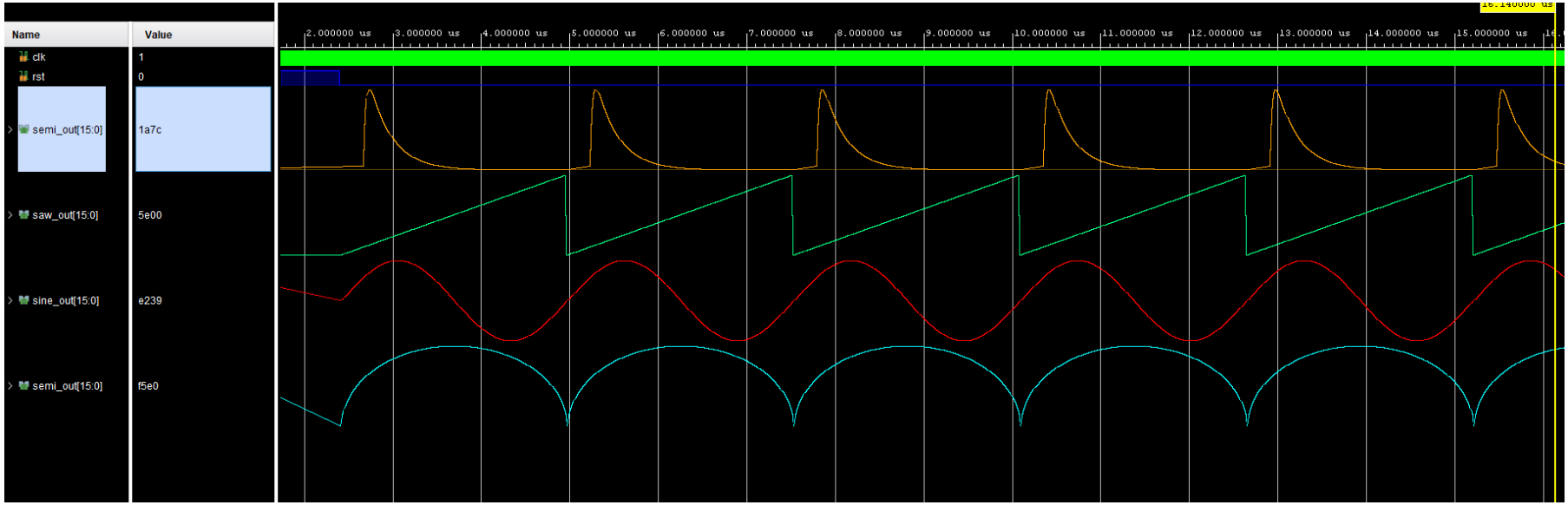

```
Run 06: DMA = 0.522 ms, Read = 0.037 ms
Run 07: DMA = 0.532 ms, Read = 0.037 ms
Run 08: DMA = 0.614 ms, Read = 0.041 ms
Run 09: DMA = 0.542 ms, Read = 0.039 ms
Run 10: DMA = 0.518 ms, Read = 0.038 ms
Run 11: DMA = 0.523 ms, Read = 0.037 ms
Run 12: DMA = 0.516 ms, Read = 0.037 ms
Run 13: DMA = 0.532 ms, Read = 0.038 ms
Run 14: DMA = 0.545 ms, Read = 0.041 ms
Run 15: DMA = 0.553 ms, Read = 0.044 ms
Run 16: DMA = 0.520 ms, Read = 0.037 ms
Run 17: DMA = 0.517 ms, Read = 0.037 ms
Run 18: DMA = 0.539 ms, Read = 0.043 ms
Run 19: DMA = 0.521 ms, Read = 0.040 ms
Run 20: DMA = 0.518 ms, Read = 0.037 ms

=== Benchmark summary (20 runs, C lib) ===
DMA avg:   0.569 ms (± 0.247)
Read avg:  0.043 ms (± 0.011)
```

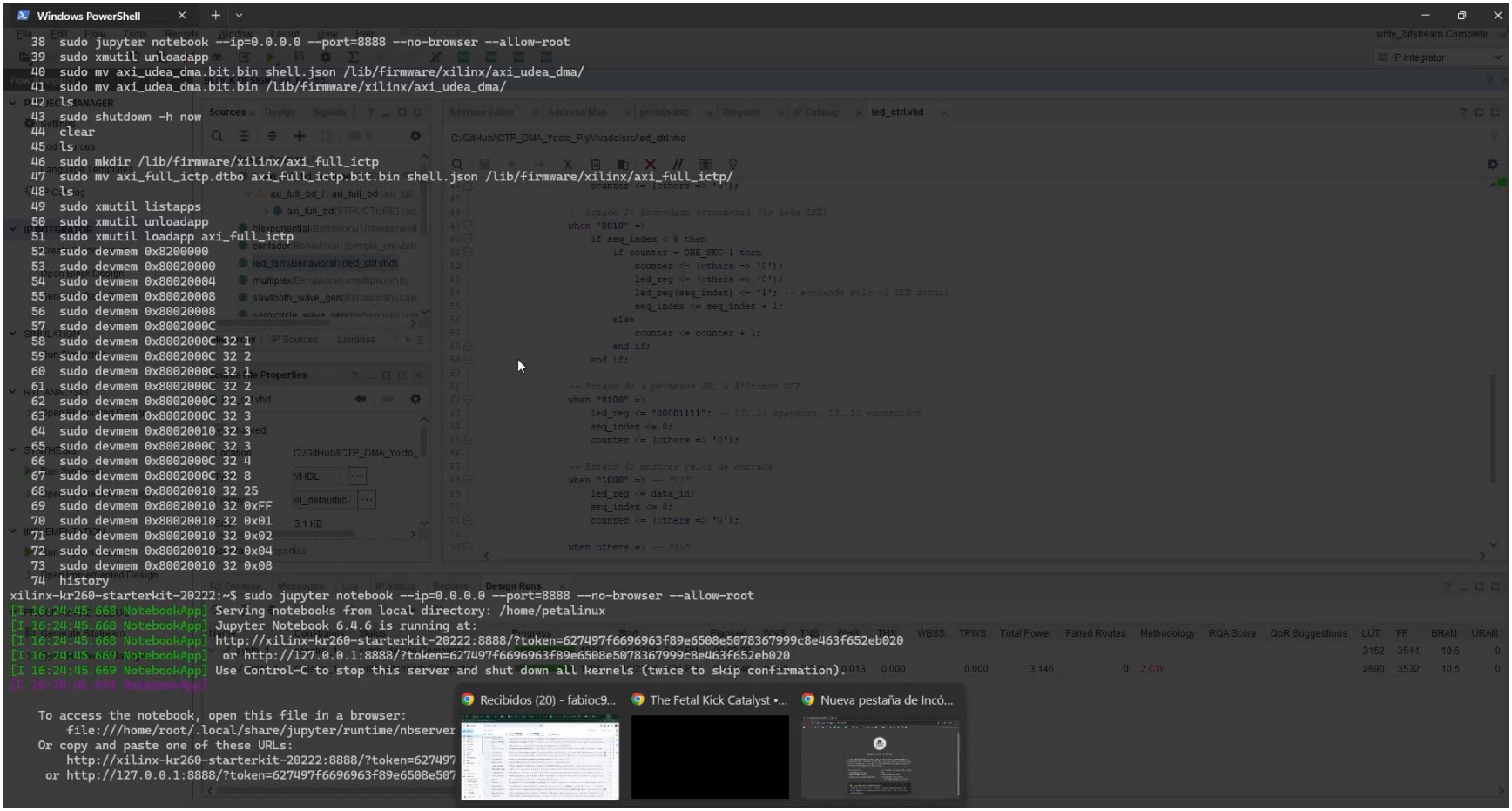


Reading a DMA example

https://github.com/fabioc9675/ICTP_DMA_Yocto_Prj/tree/ICTP_Presentation/Petalinux/CodeTest



Linux controller



Linux controller