DIGITAL PULSE PROCESSING WITH OPEN FPGA ARCHITECTURE:

THEORY AND METHODS

A. Abba, D. Bianchi, F. Caponio, G. Cerretani, L. Colombini, A. Cusimano,, T. Masini, Y. Venturini

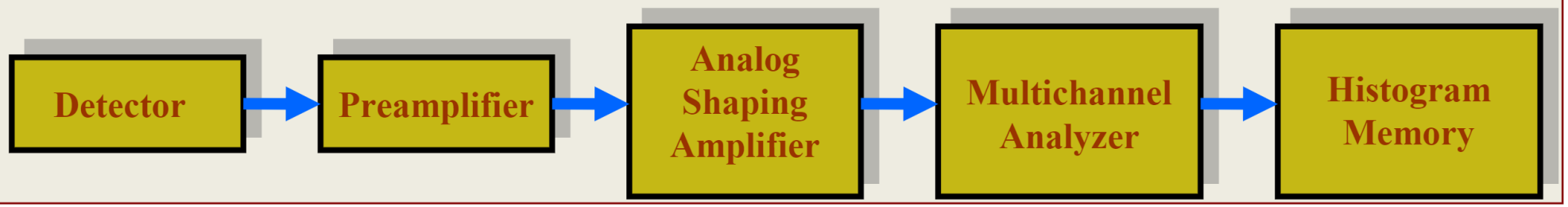
JOINT EUROPEAN MASTER IN NUCLEAR PHYSICS 2024

Seville, November 19-20th, 2024

DIGITAL PULSE PROCESSING BASICS

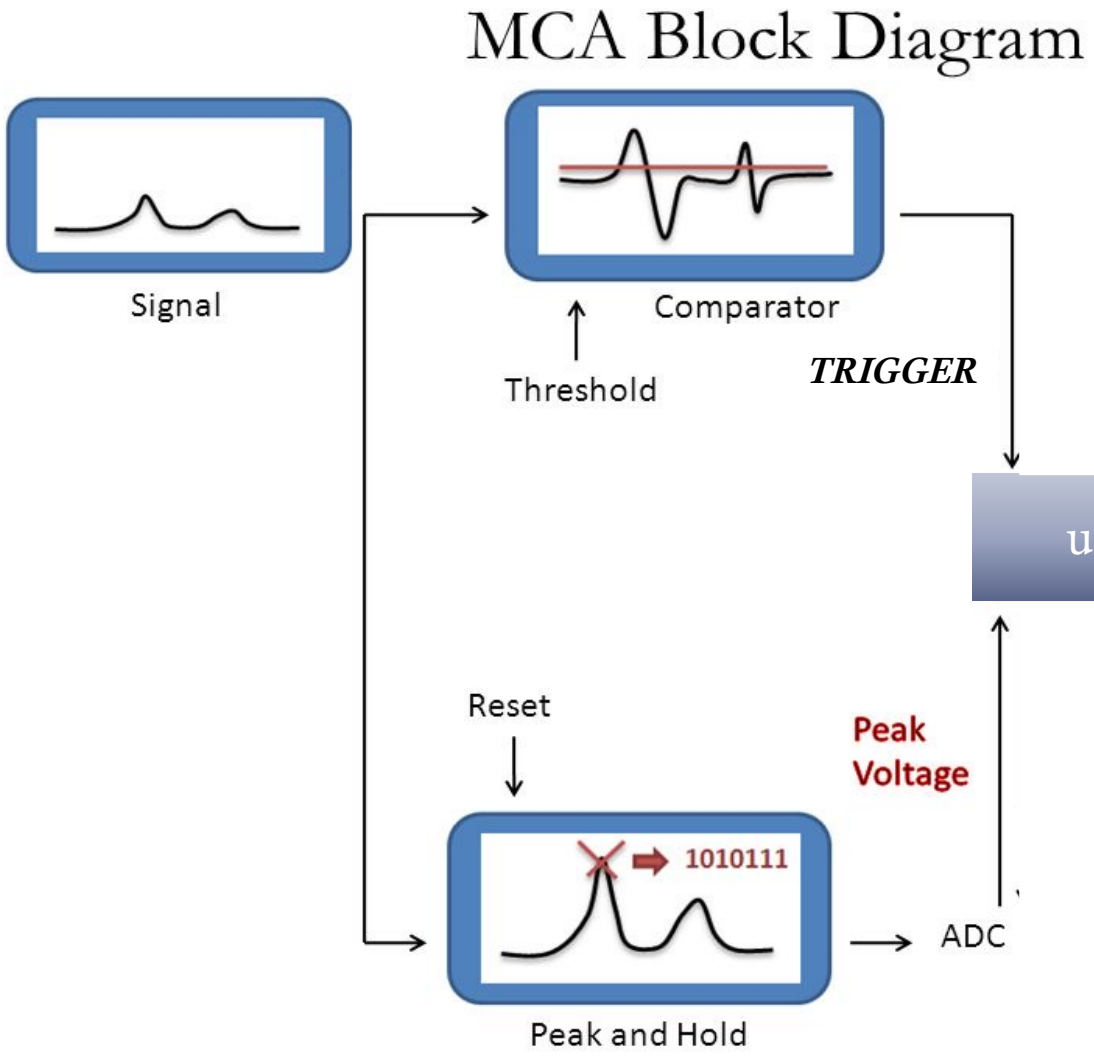
DIGITAL SIGNAL PROCESSING FOR NUCLEAR PHYSICS

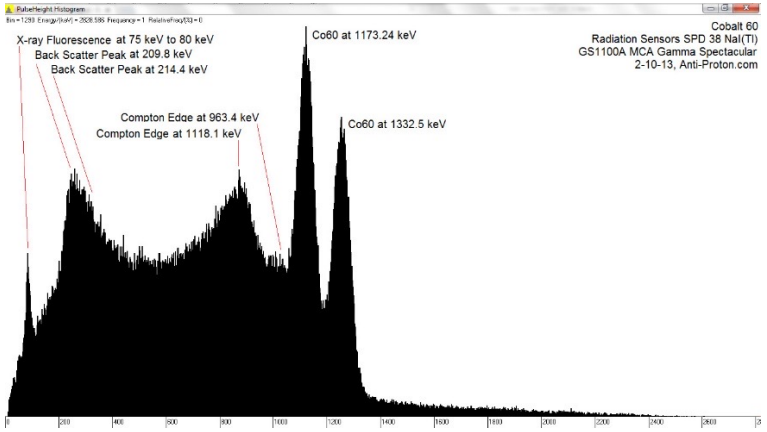
•Classical Spectrometer

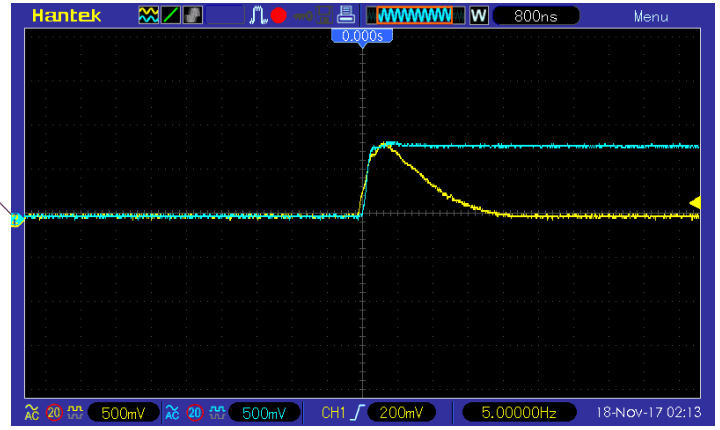


•Digital Spectrometer

ANALOG SPECTROMETERS

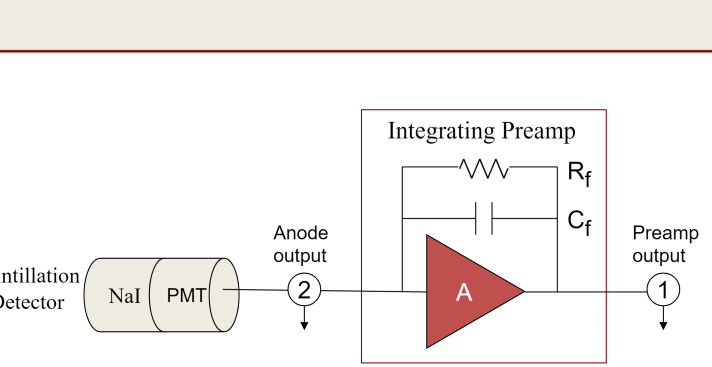


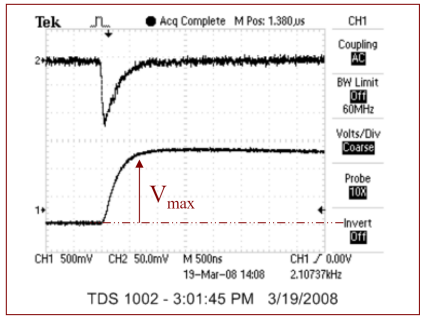




TRANSITION TO DIGITAL

•Digital Spectrometer



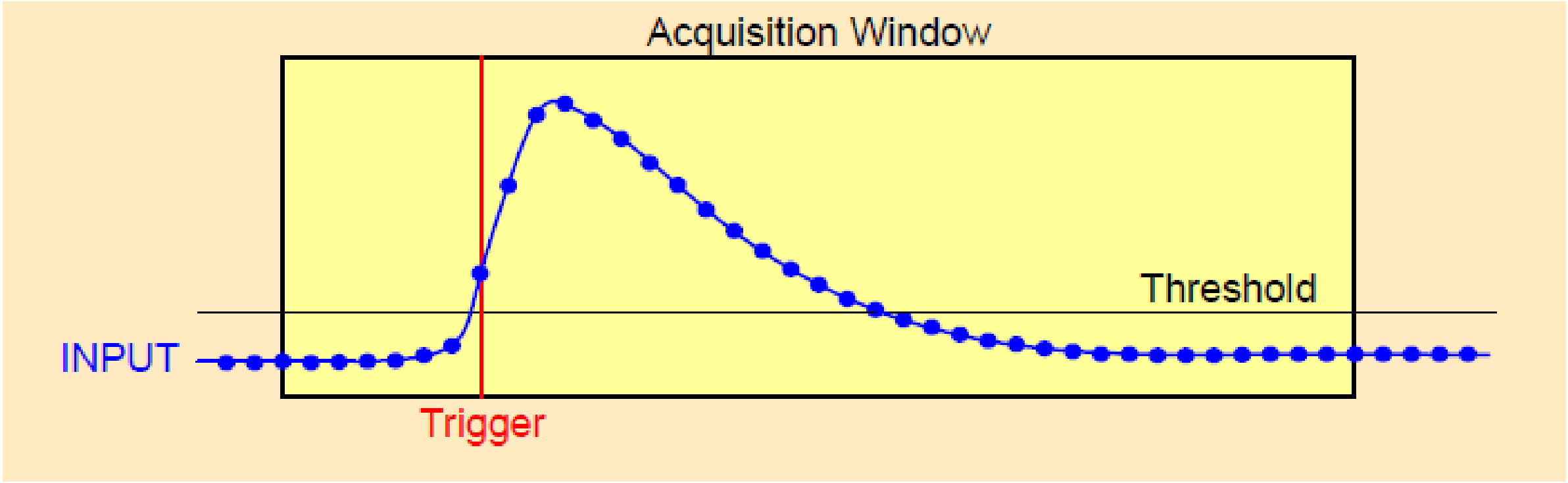


Detector → Preamplifier → High-Speed ADC → Digital Pulse Processing → Histogram Memory

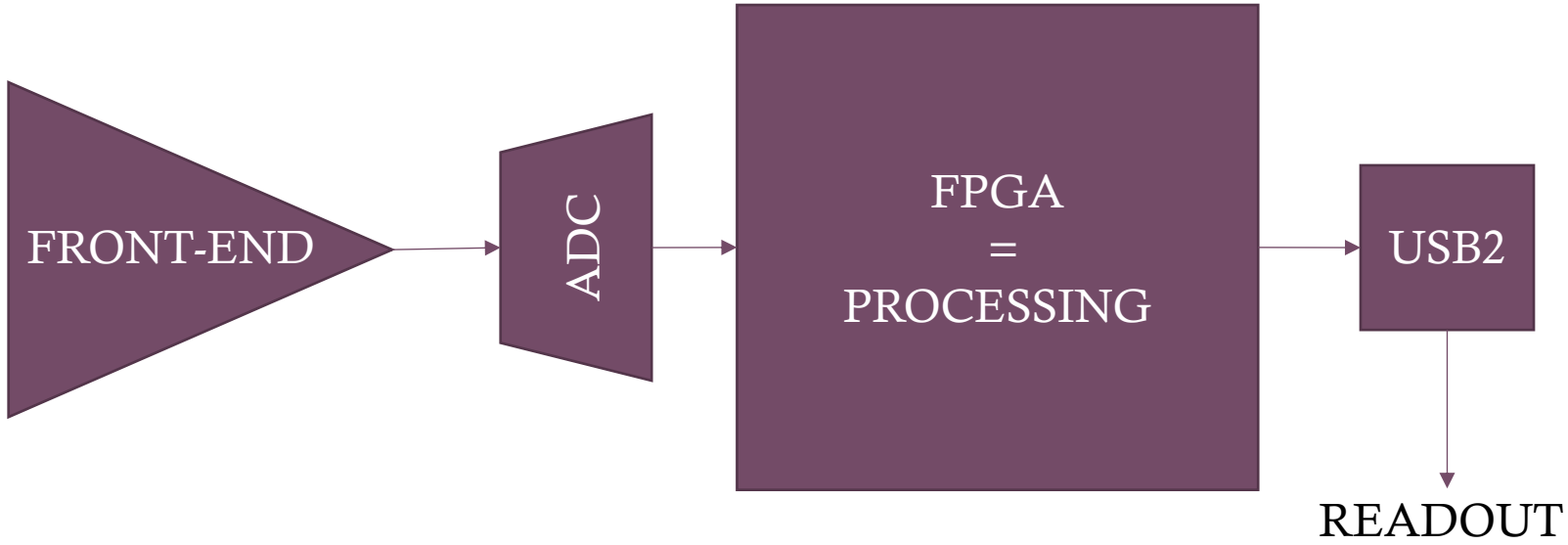
Performed in software/firmware

DIGITAL SIGNAL PROCESSING: THE ADC

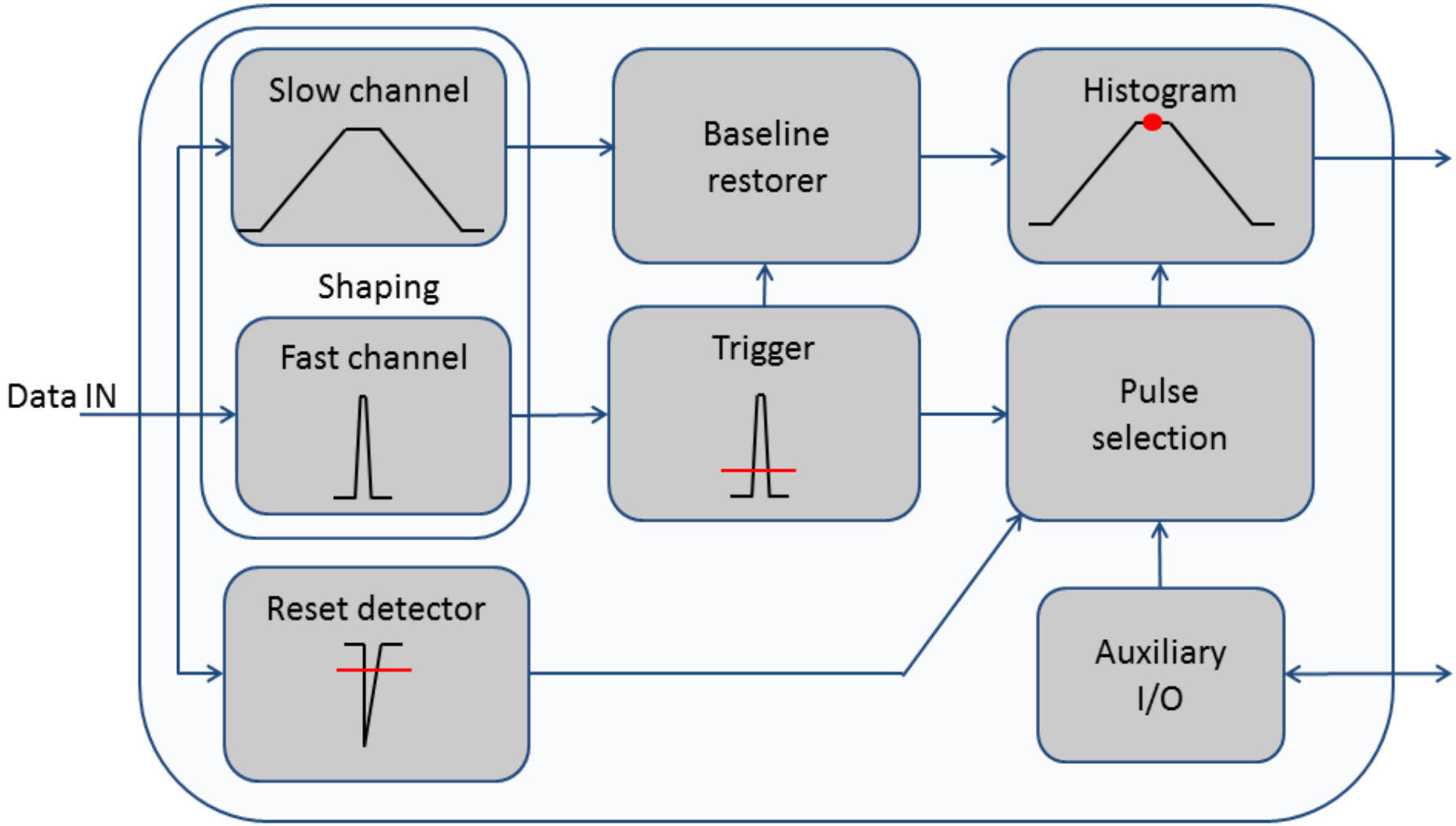
Digitize the analog signal



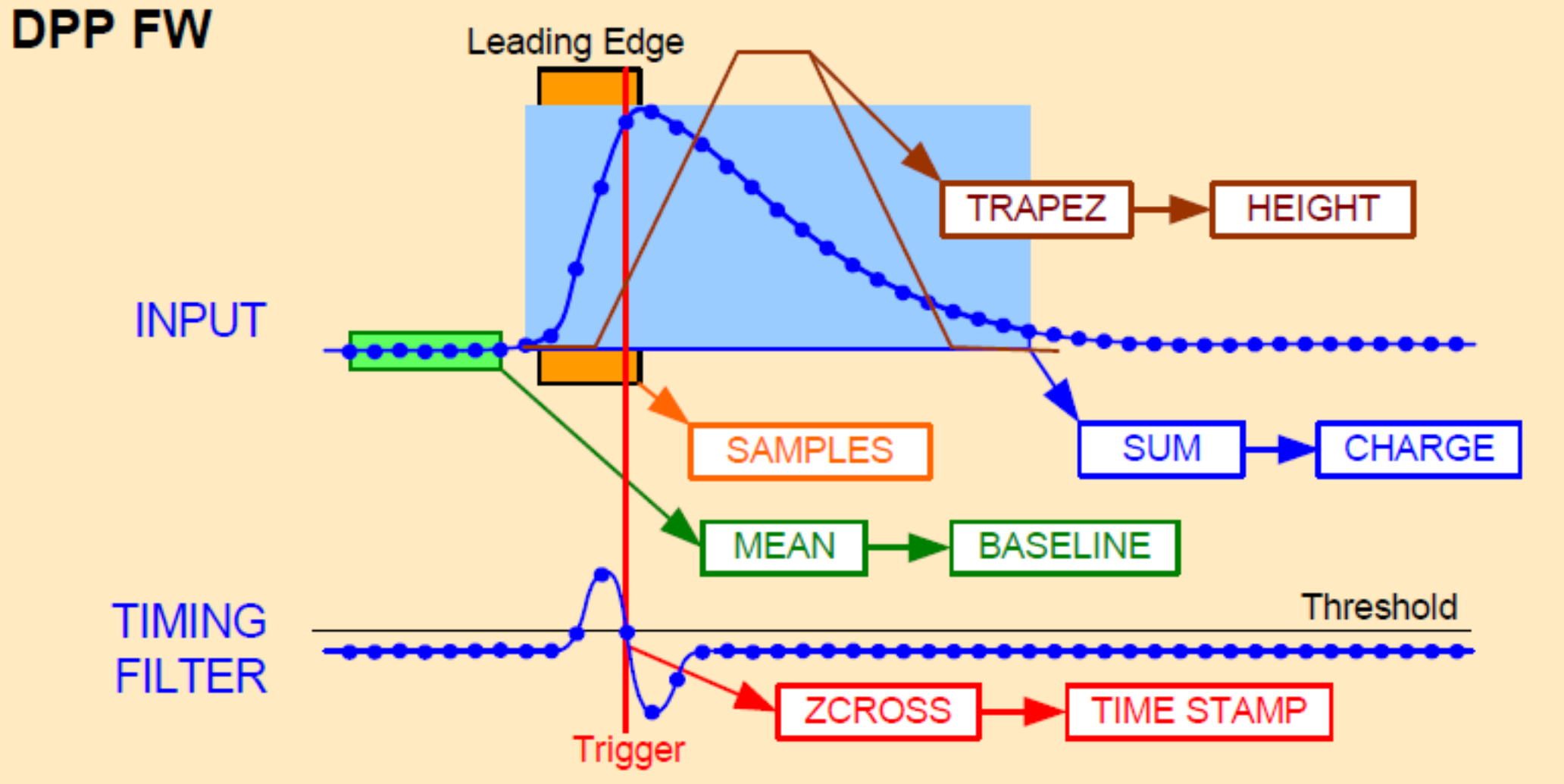
SIGNAL PROCESSING



BLOCK DIAGRAM FOR DIGITAL SPECTROMETERS



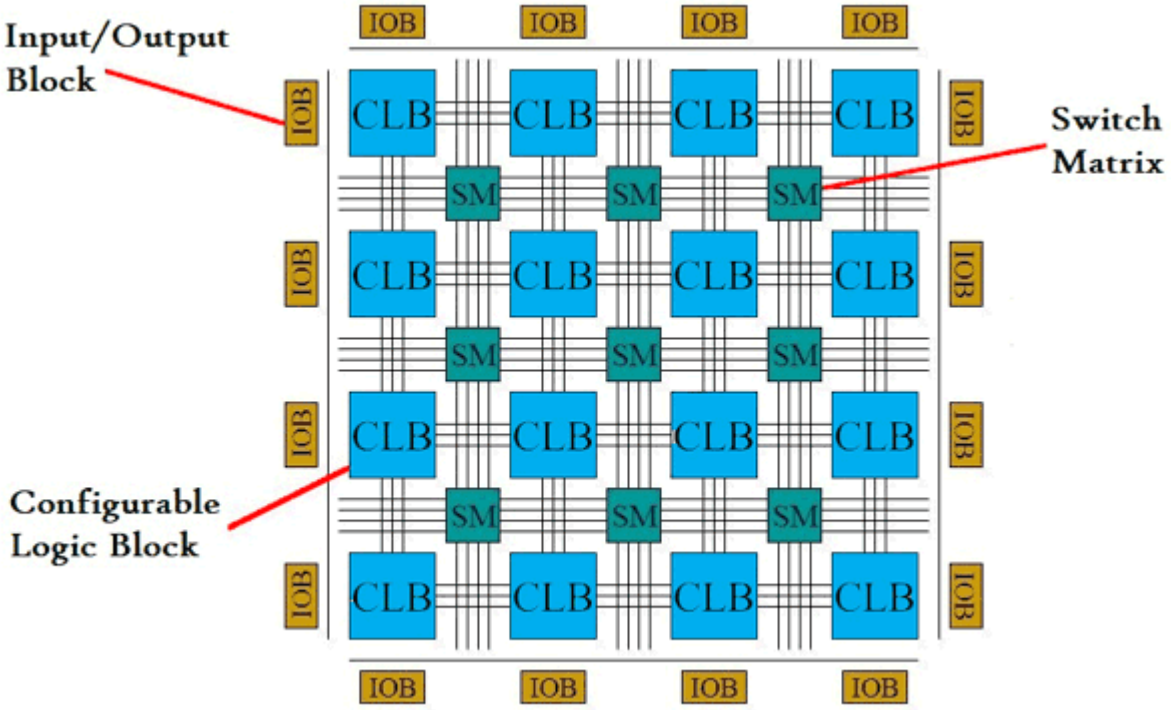
DIGITAL SIGNAL PROCESSING

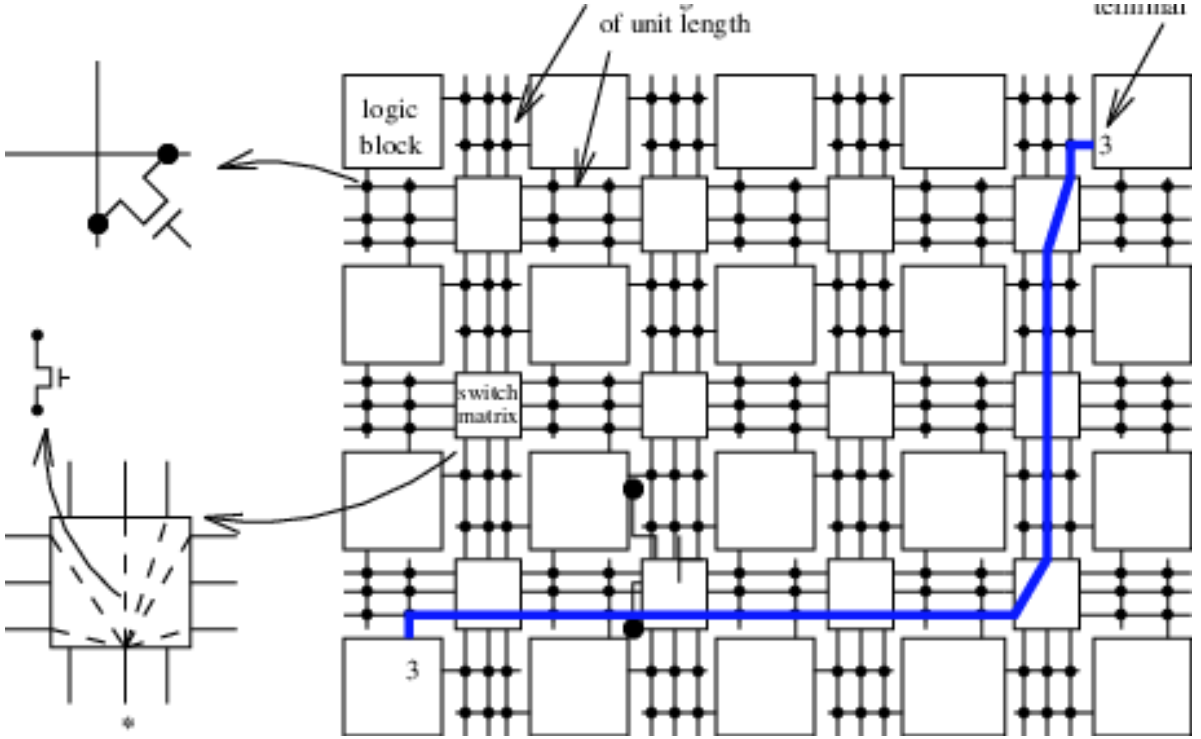


METHODS FOR FPGA PROGRAMMING

DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

Field Programmable Gate Array





DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

Programming FPGA

VHDL and Verilog

- The two leading hardware description languages are Verilog and VHDL.
- Verilog and VHDL are built on similar principles but have different syntax.
- VHDL is an acronym for the VHSIC Hardware Description Language.

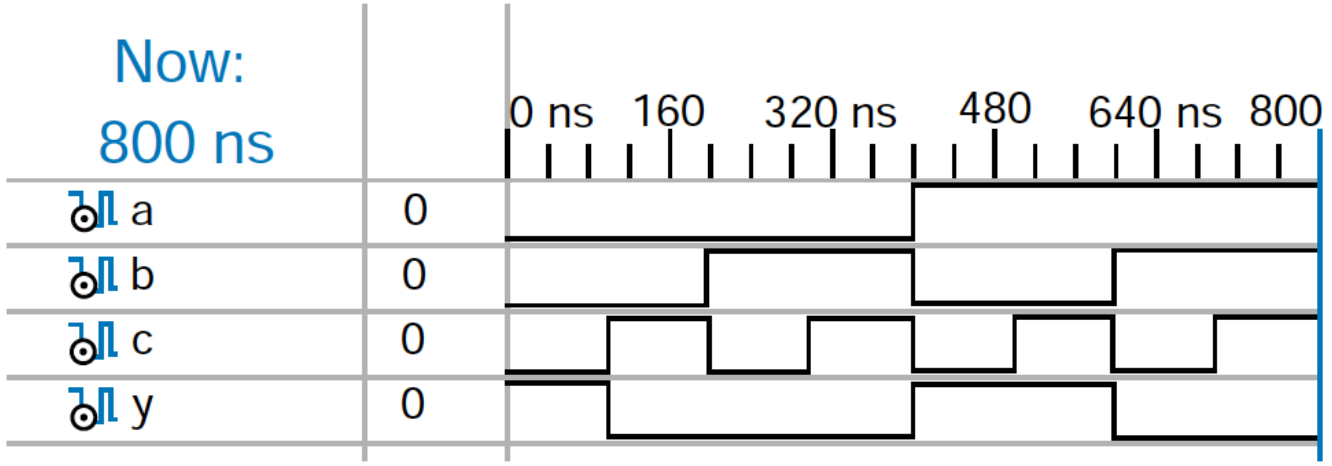
  VHSIC is in turn an acronym for the Very High Speed Integrated Circuits program of the US Department of Defense.
- They are not programming language. They describe the hardware structure to be implemented in the FPGA. They are a way to describe a circuit using a textual description

DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

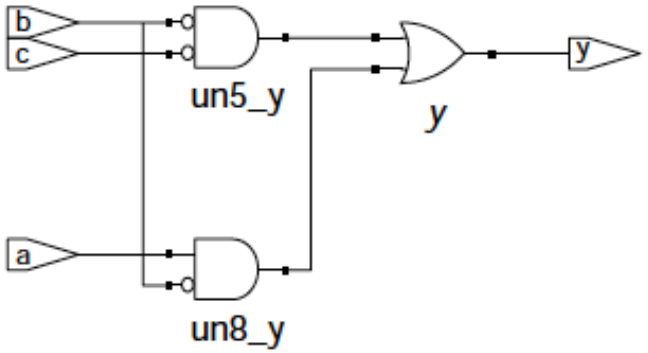
Programming FPGA

$$y = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + a\bar{b}c$$

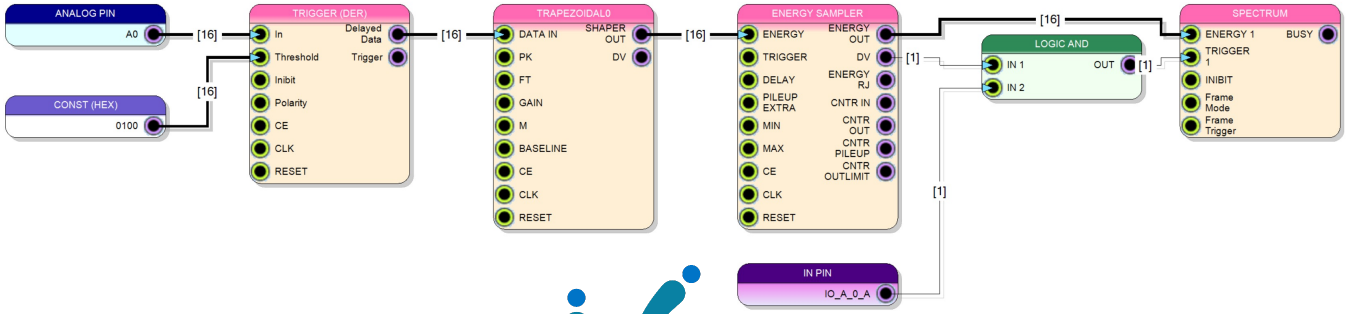
```
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity sillyfunction is
  port(a, b, c: in  STD_LOGIC;
       y:       out STD_LOGIC);
end;

architecture synth of sillyfunction is
begin
  y <= ((not a) and (not b) and (not c)) or
       (a and (not b) and (not c)) or
       (a and (not b) and c);
end;
```



Syntesis

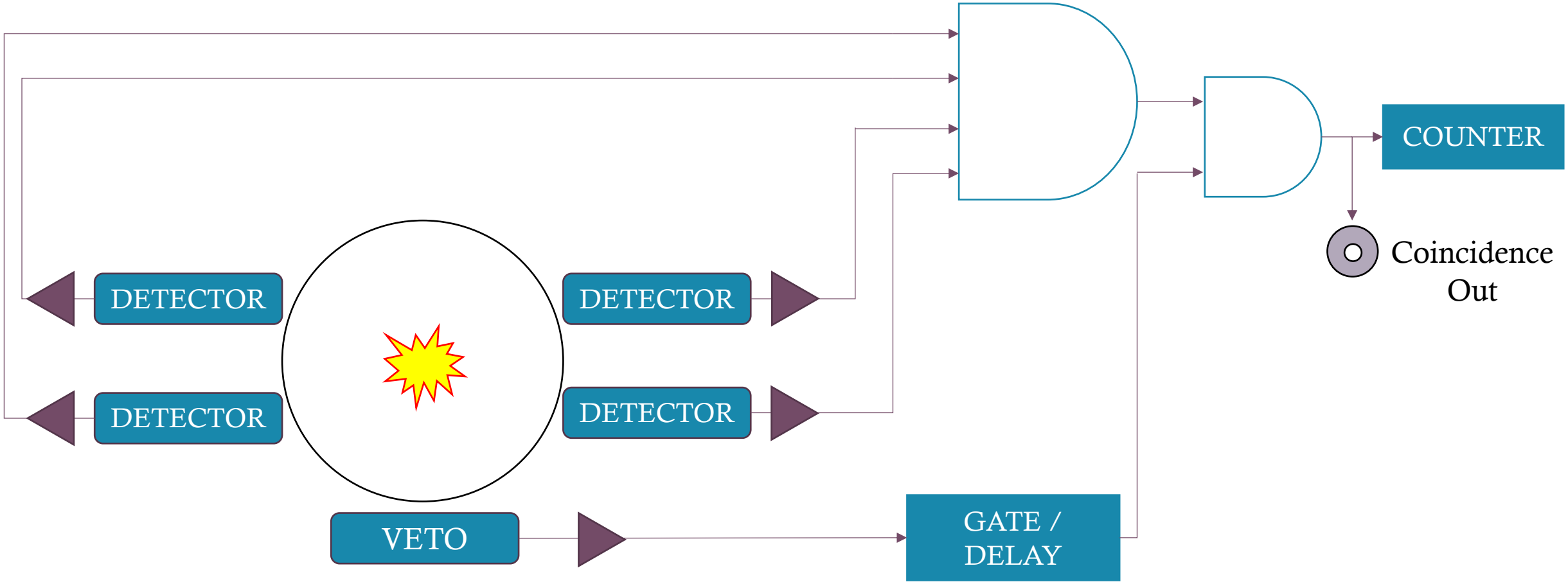


OPEN FPGA DIGITIZERS + SCI-COMPILER FIRMWARE GENERATOR

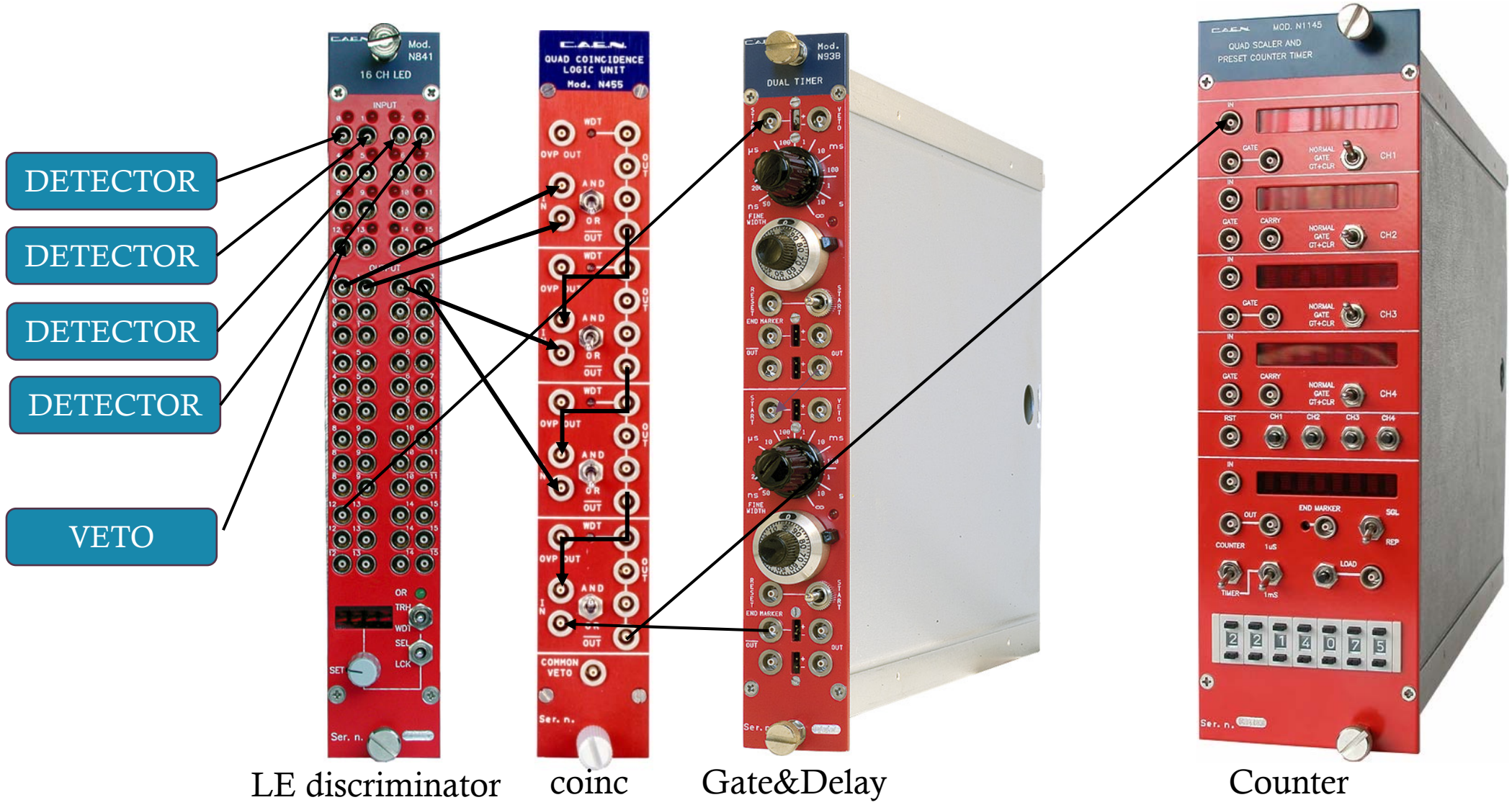


SCI-COMPILER

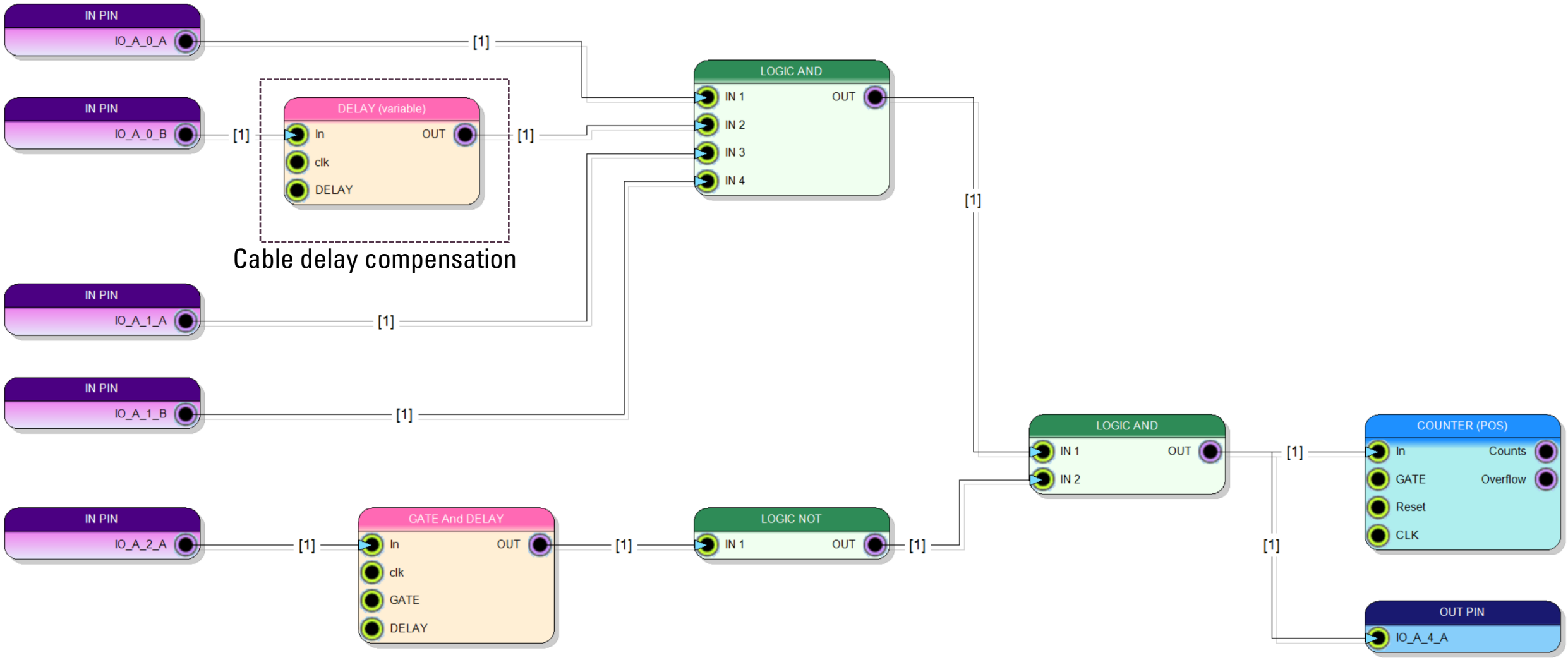
EXAMPLE: 4-COINCIDENCE EVENT COUNTER WITH VETO



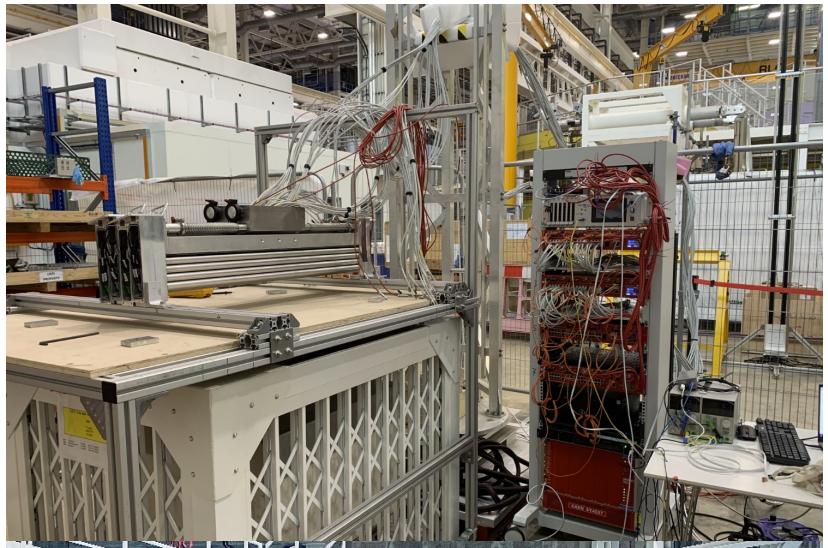
LET'S DO WITH NIM MODULES…

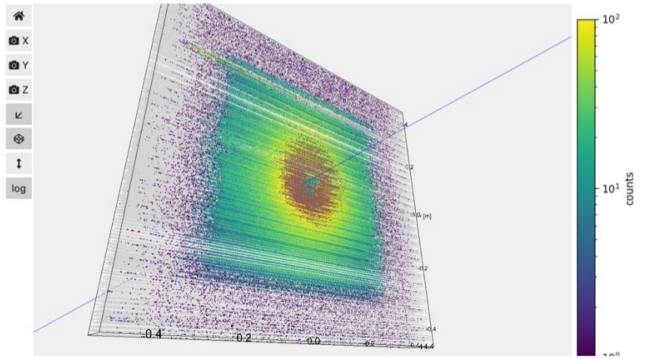


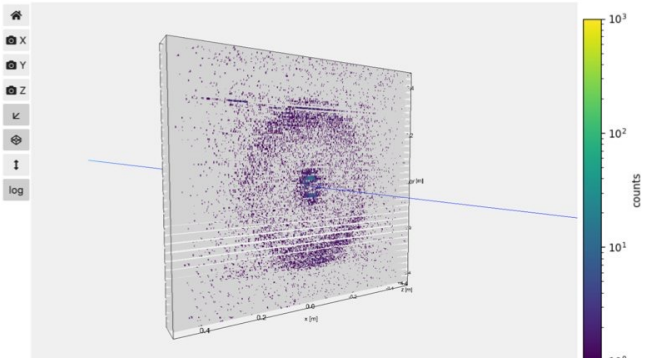
LET'S DO THE SAME IN SCI-COMPILER



ESS - LOKI SANS READOUT – 2300 CHANNELS



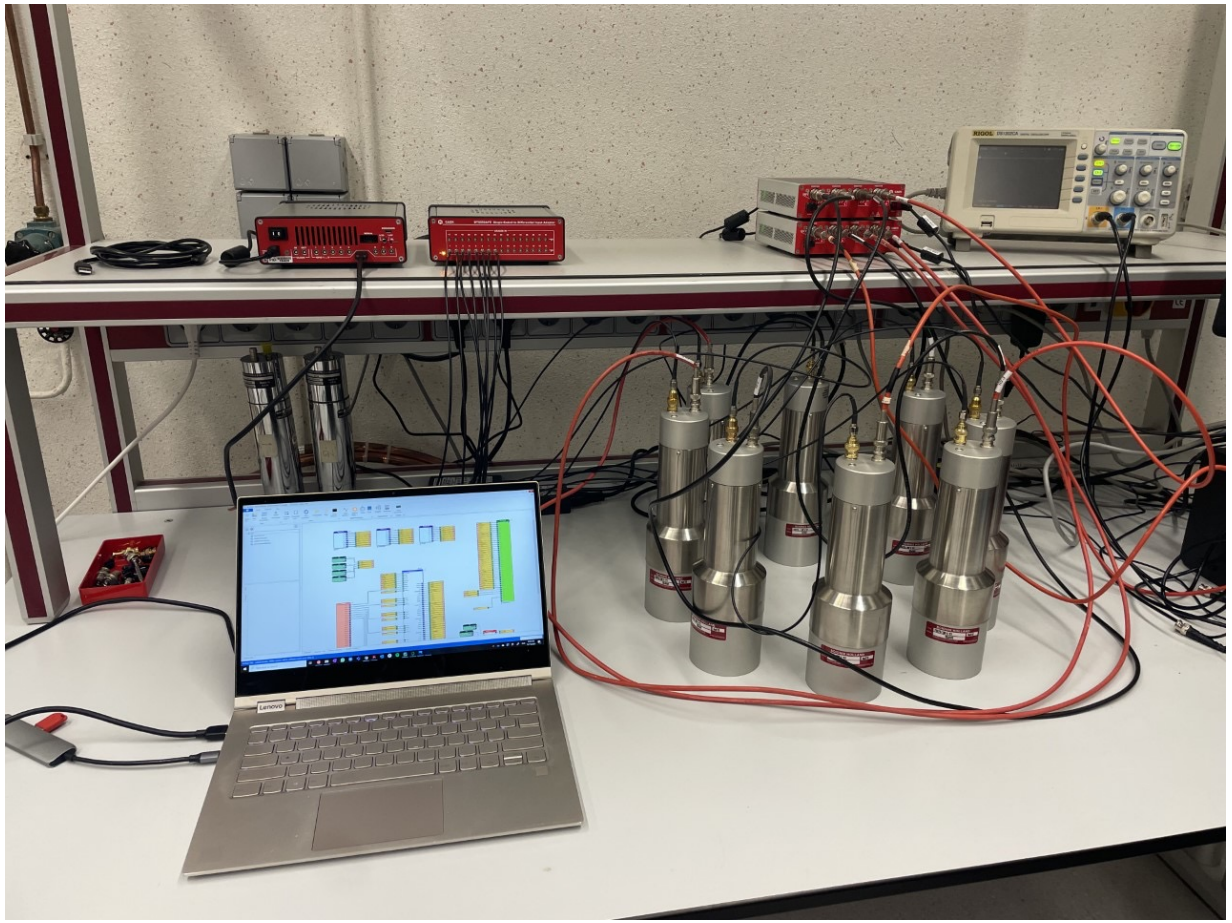




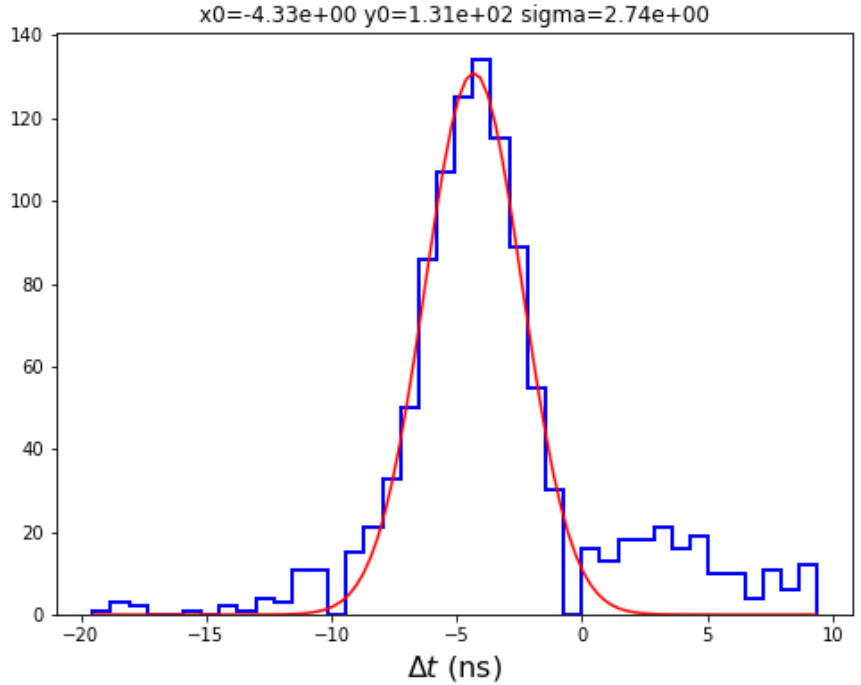
Real time processing from 2300 channels to readout straw tubes for LOKI experiment @ ESS.
ESS BIFROST, MIRACLES and VESPA are using R5560 and SciCompiler for readout

Davide Raspino - ISIS

SUB-NS TIME OF FLIGHT MEASURAMENT WITH DCFD



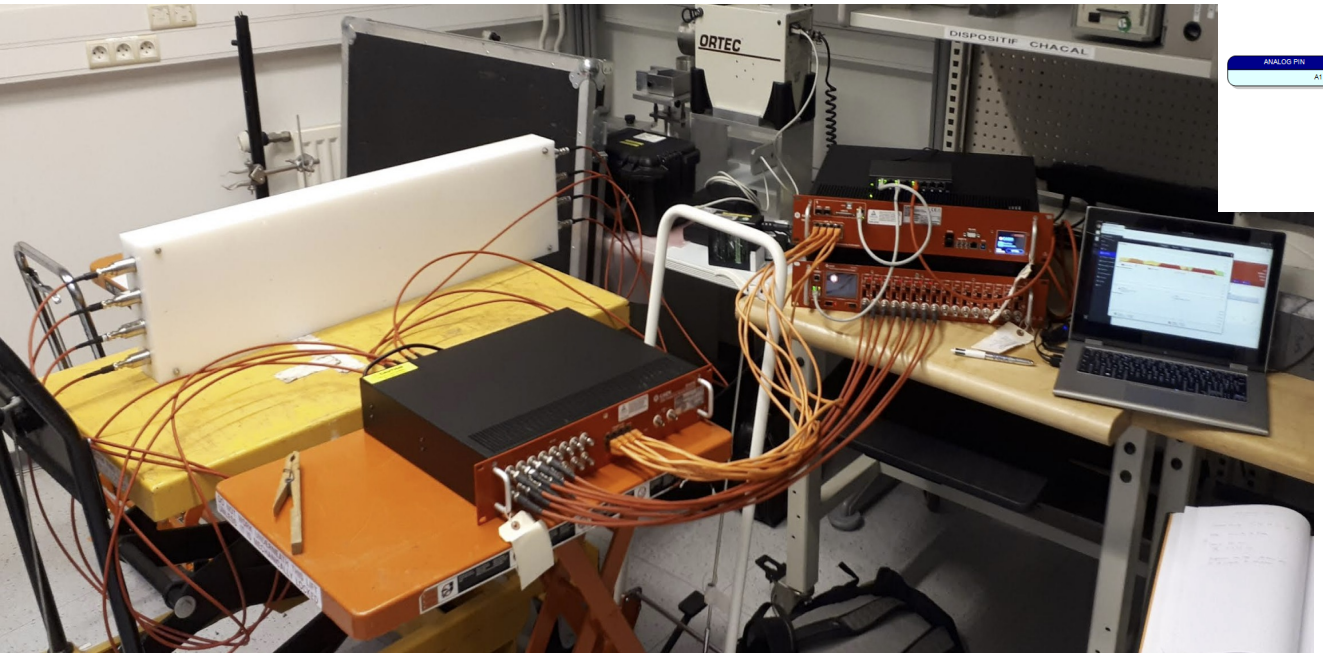
Sub-ns time of flight of correlated gamma measurement using DT5550, PMTs and a custom firmware developed using SciCompiler



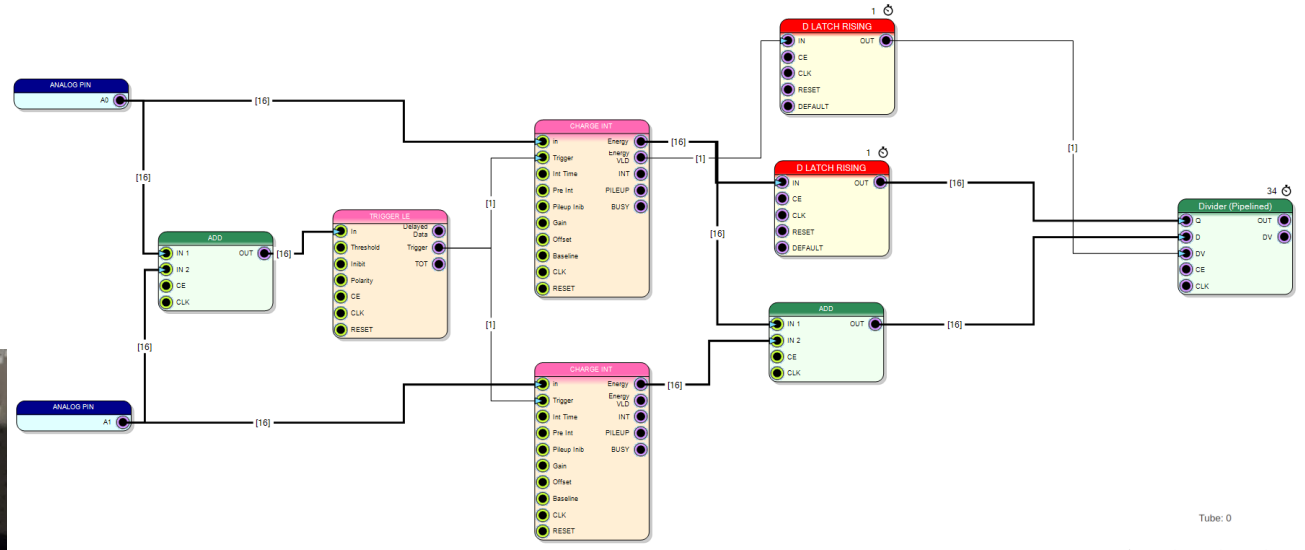
Auke Colijn - Nikhef

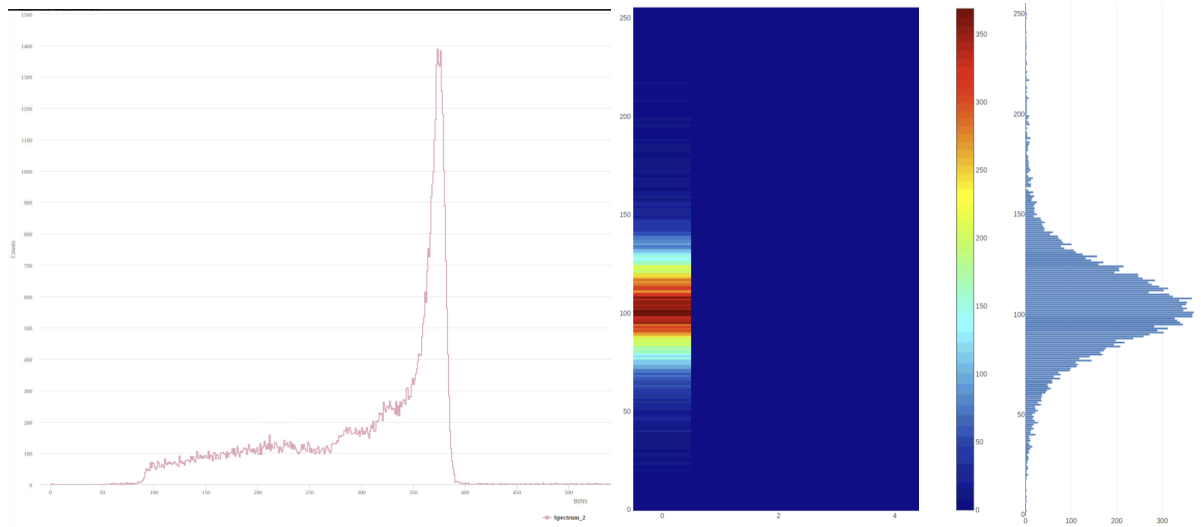
POSITION SENSE HE3 TUBE

Real time calculation of neutron interaction point using He3 tubes. 32 channels realtime center of mass and energy spectrum calculation.

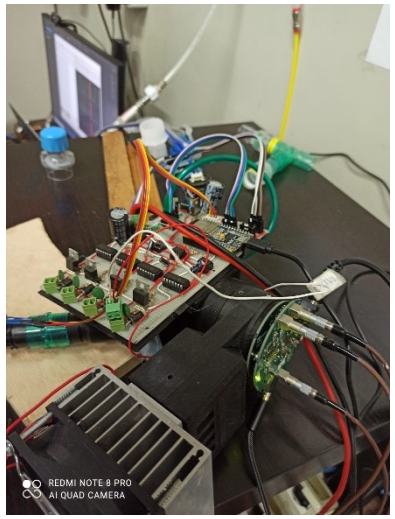


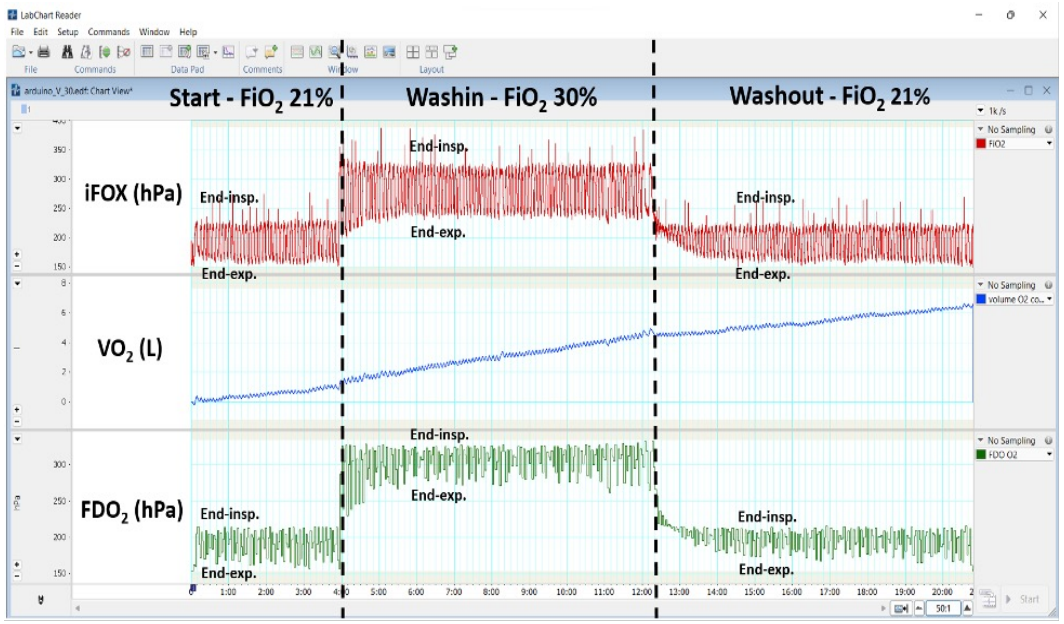
picture from IRSN

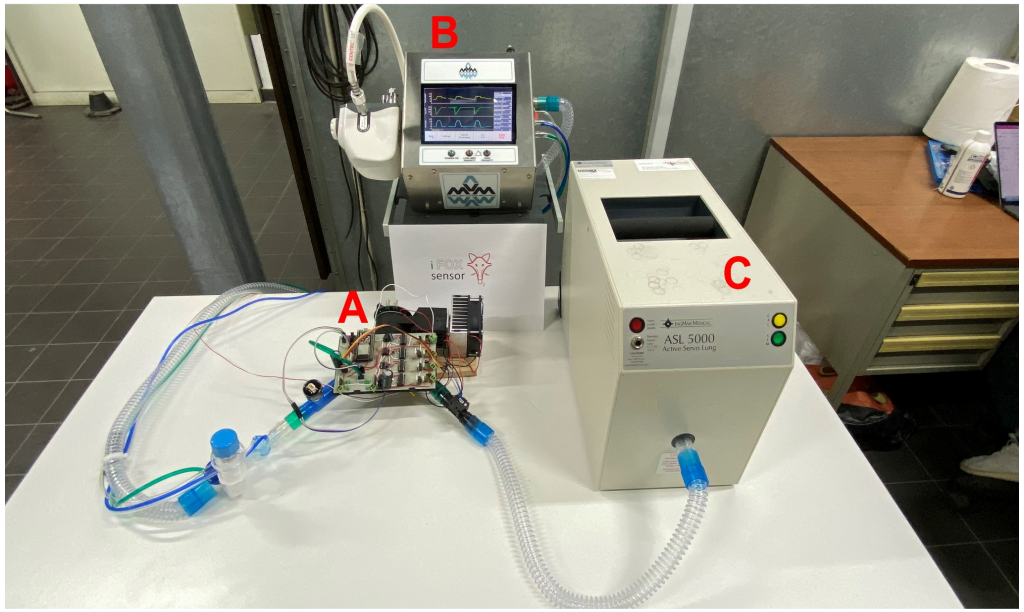


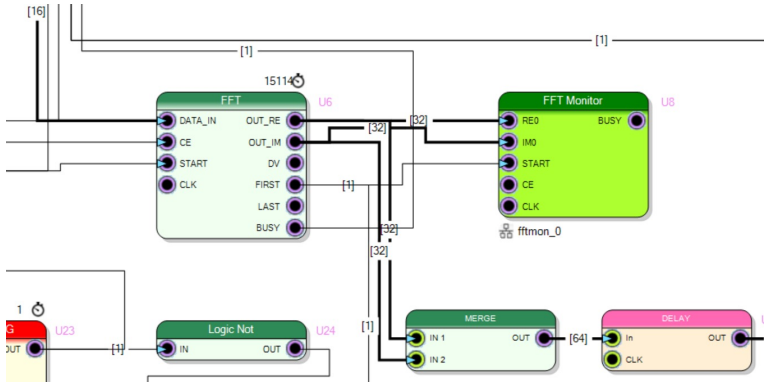


IFOX SENSOR – ITALIAN FAST OXYGEN SENSOR









Set up of the preclinical study by using the iFOX sensor (A), the Mechanical Ventilator Milano (MVM) (B) and the lung simulator ASL 5000 (C).

IFOX SENSOR

Marchio registrato numero 302022000044975

gabriele.croci@unimib.it or marco.tardocchi@istp.cnr.it

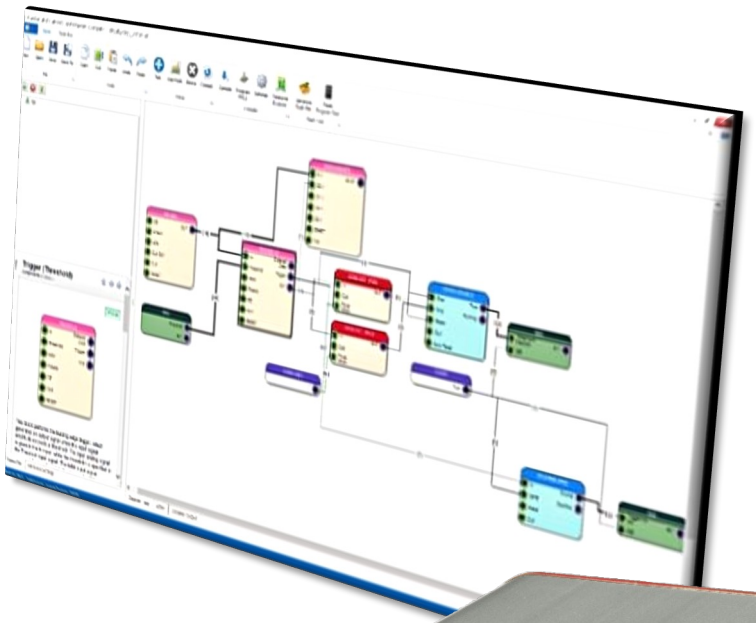
UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

ISTP Consiglio Nazionale delle Ricerche – ISTITUTO PER LA SCIENZA E TECNOLOGIA DEI PLASMI

HANDS-ON

EXPERIMENTAL SETUP

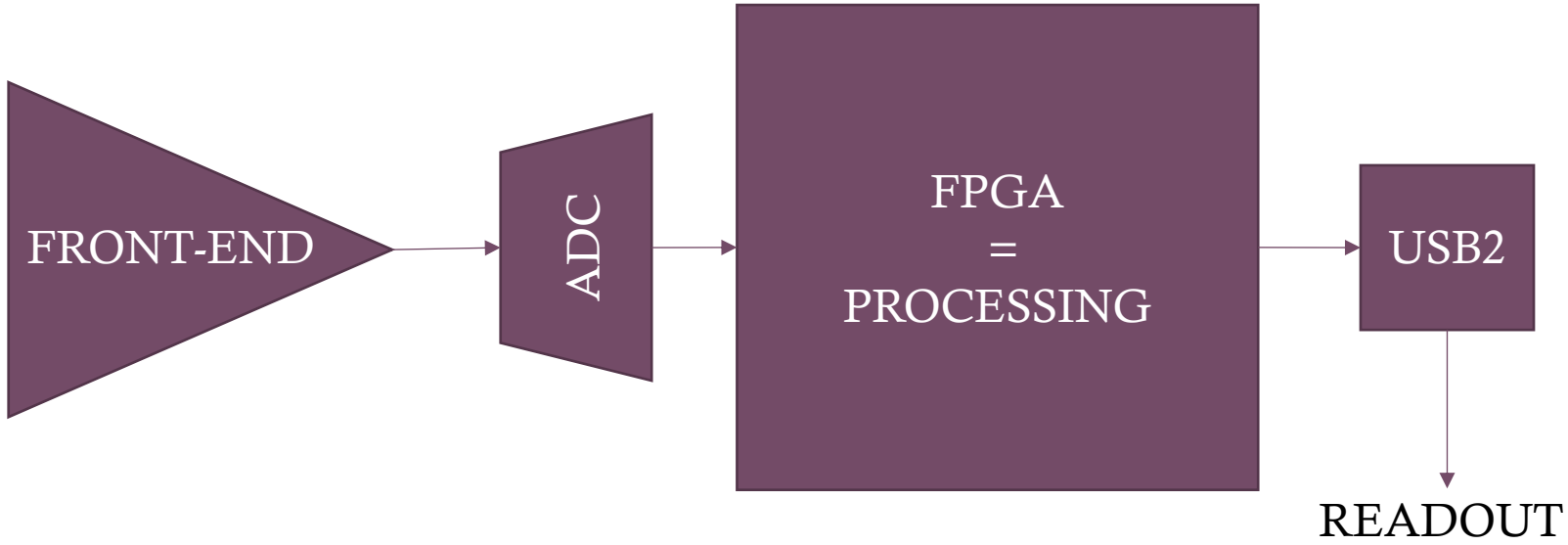
SP5650 – Open FPGA kit



DT1260
65 Ms/s, 12bit

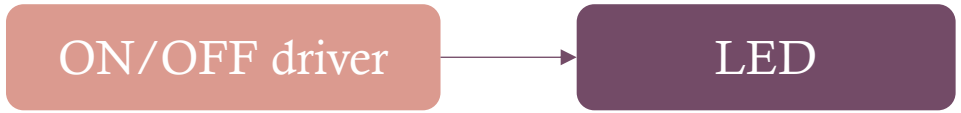
DT4800
Digital Detector Emulator

SIGNAL PROCESSING



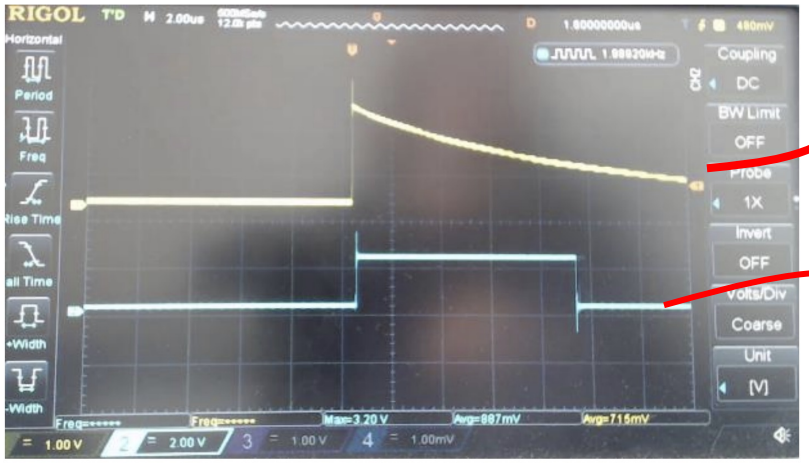
HANDS-ON EXERCISE N° 1

EX 1-BLINKING LED



HANDS-ON EXERCISE N° 2

EX 2-DISCRIMINATOR WITH PROGRAMMABLE THRESHOLD



Analog in

Threshold

DISCR

LED

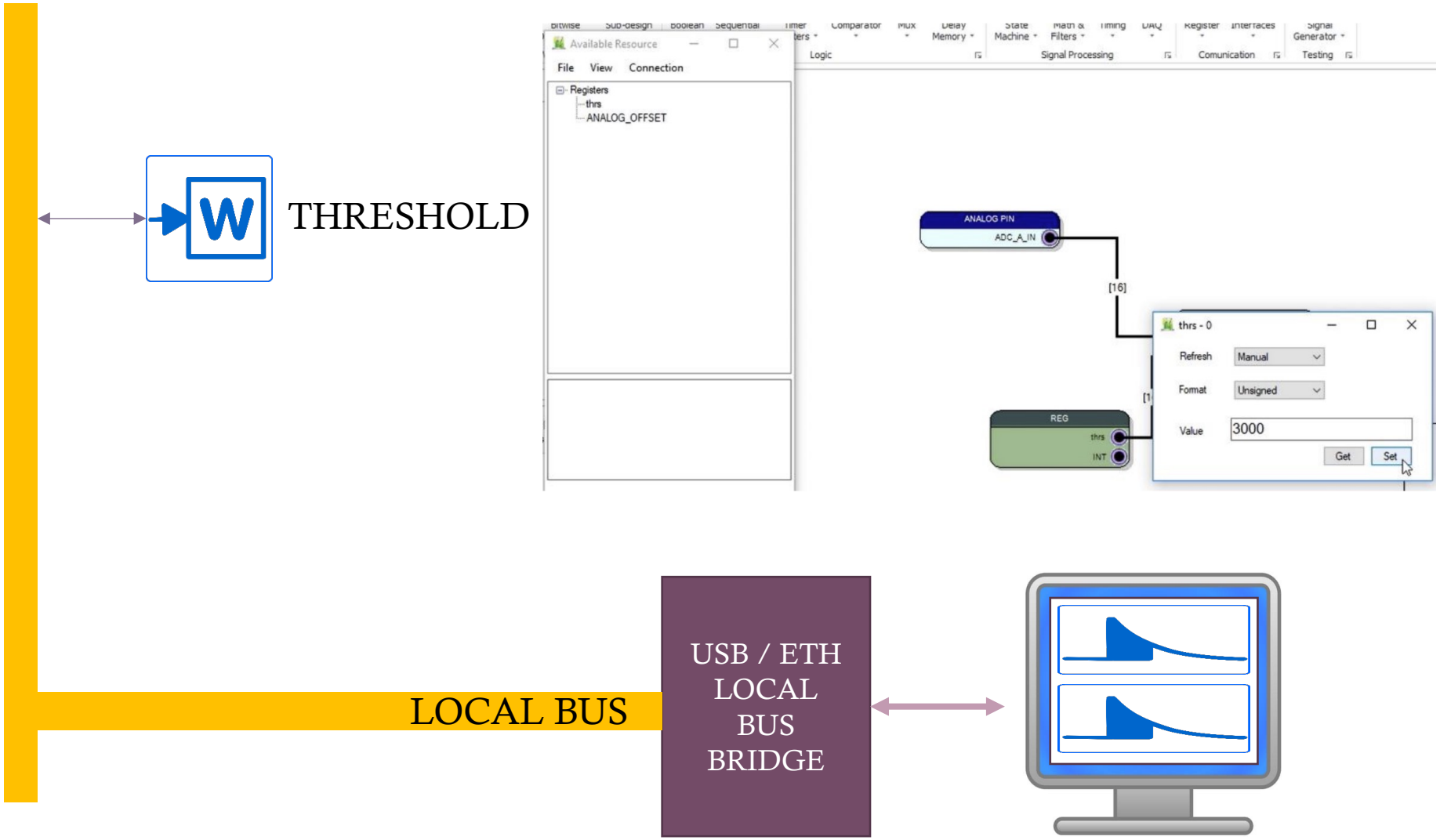
LEMO OUT

USB

Analog in signal

LED

EX 2-DISCRIMINATOR WITH PROGRAMMABLE THRESHOLD



HANDS-ON EXERCISE N° 3

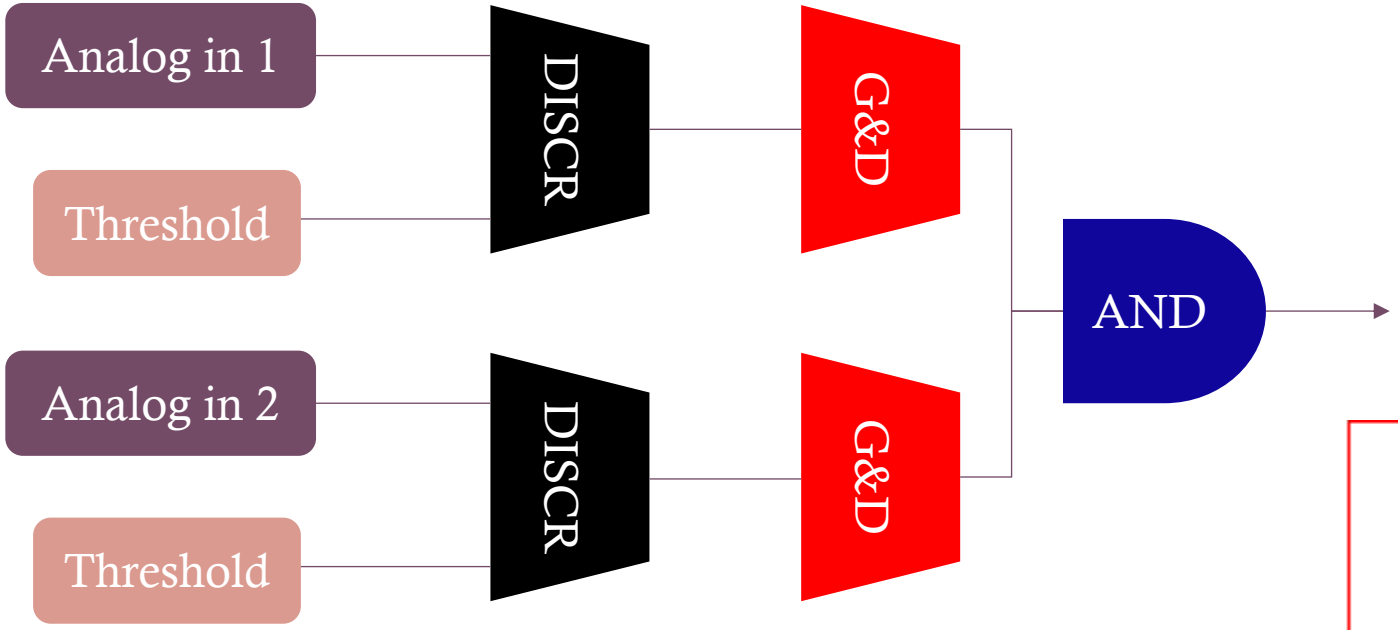
TWO-CH. COINCIDENCE **WITHIN A PROGRAMMABLE WINDOW**

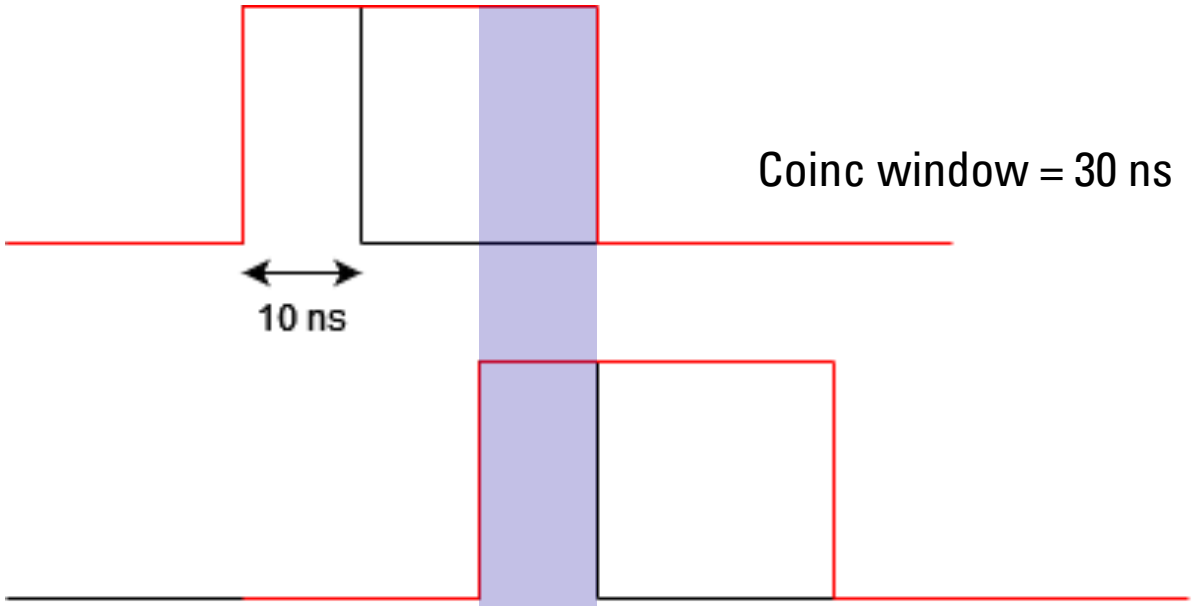
1. **Write down a conceptual schematic on paper**
2. **Translate the conceptual design in a Sci-Compiler block diagram**
3. Check results are as expected with external or internal tools

HINTS

- Two **analog** signals as input
- Coincidence means logic AND of two **digital** signals
- In physics, two signals are coincident when both falls within a certain window (even if they are not superimposed)
- You can delay/elongate signals to impose a coincidence window

STEP 1


Analog in 1
Threshold
DISCR
G&D
Analog in 2
Threshold
DISCR
G&D
AND


10 ns
Coinc window = 30 ns

STEP 2

HINTS

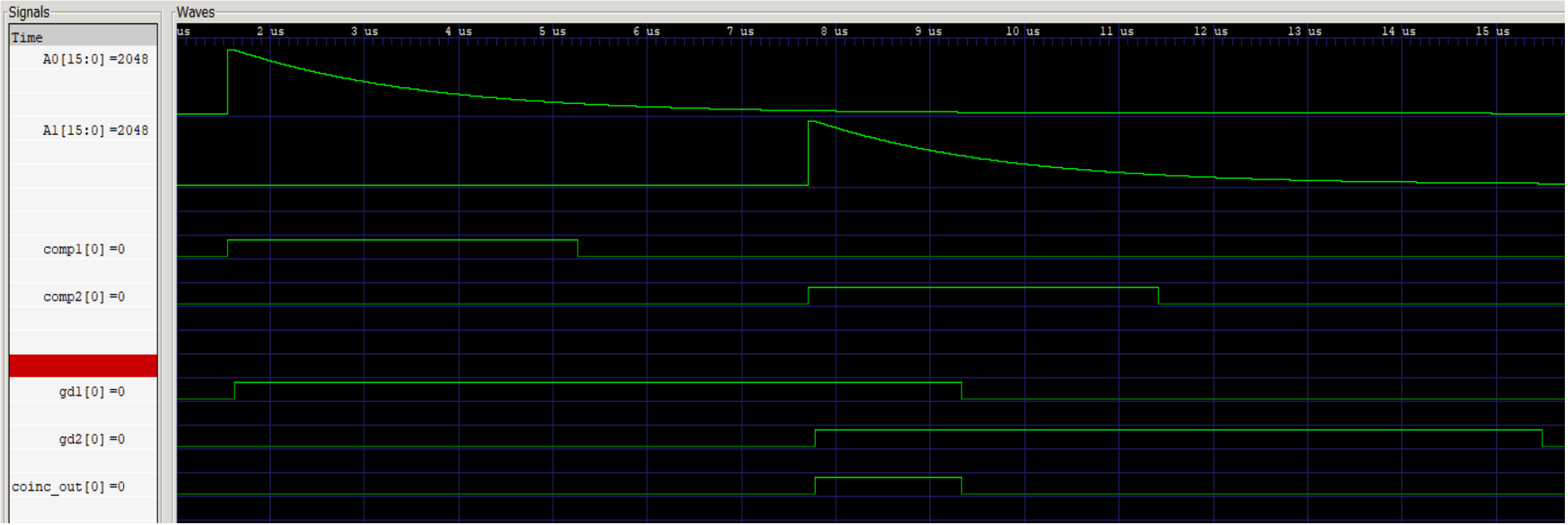
- Replicate the conceptual scheme
- Think about the previous example (90% of the work is done!)
- Remember to place tools for monitoring the results

PRELIMINARY STEP 3

SIMULATION

Signal delay = 500-100=6.4 us

w=500=8 us → coincidence window > signal delay → coinc out = 1



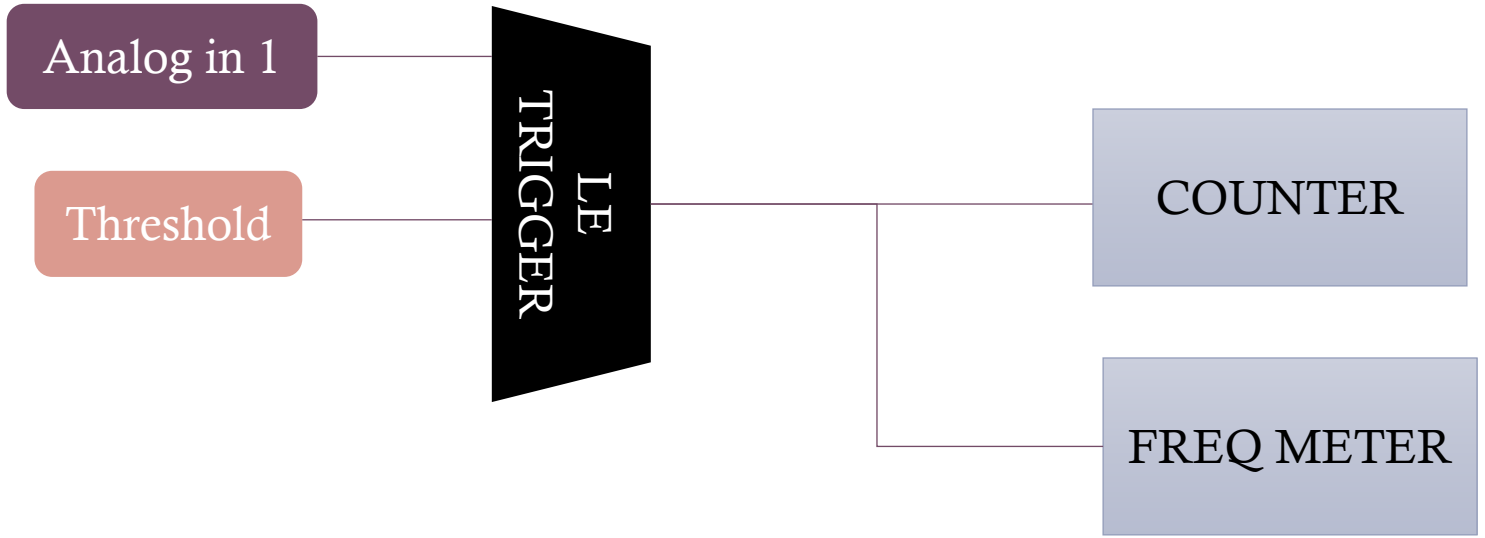
STEP 3

HINTS

- Set the right parameters to impose a coincidence window of your choice (for example w=500=8 us)
- Verify if two signals with deltaT<8 us generate a coincidence signal

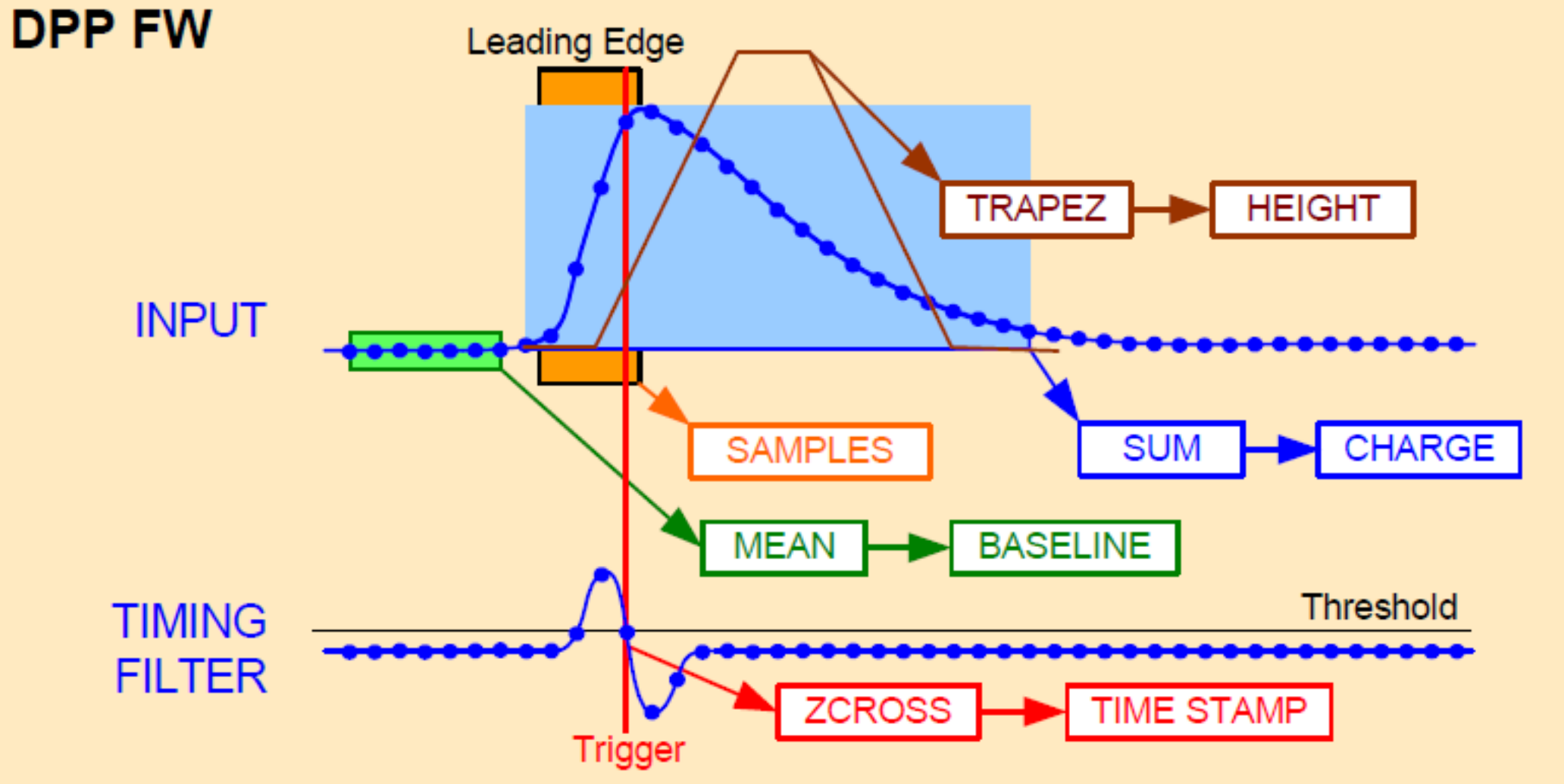
HANDS-ON EXERCISE N° 4

EVENT COUNTING

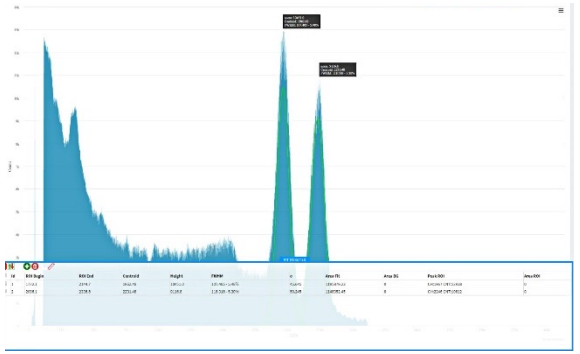


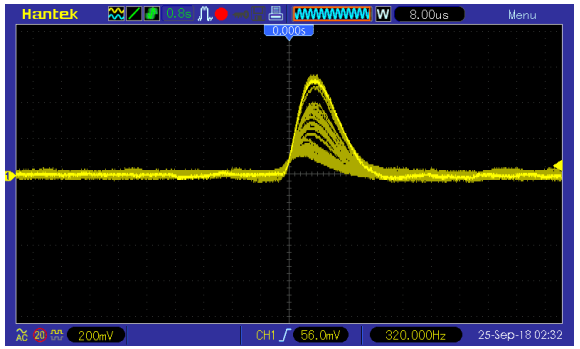
HANDS-ON EXERCISE N° 5

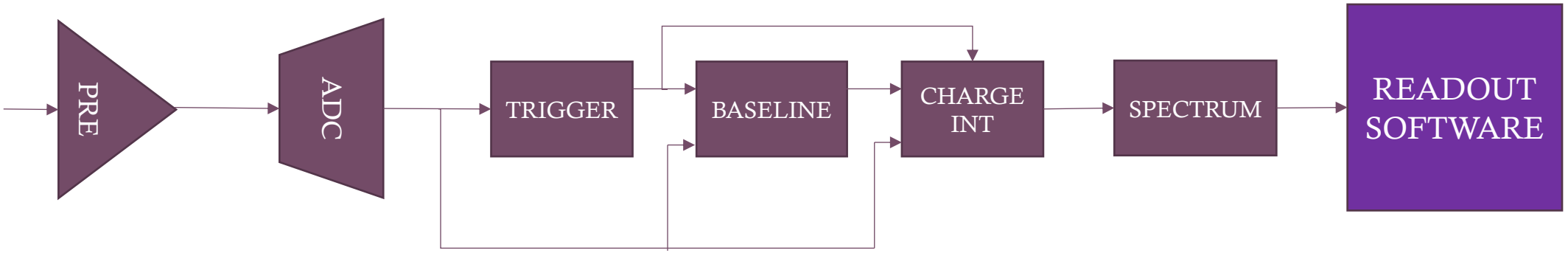
DIGITAL SIGNAL PROCESSING



CHARGE INTEGRATION







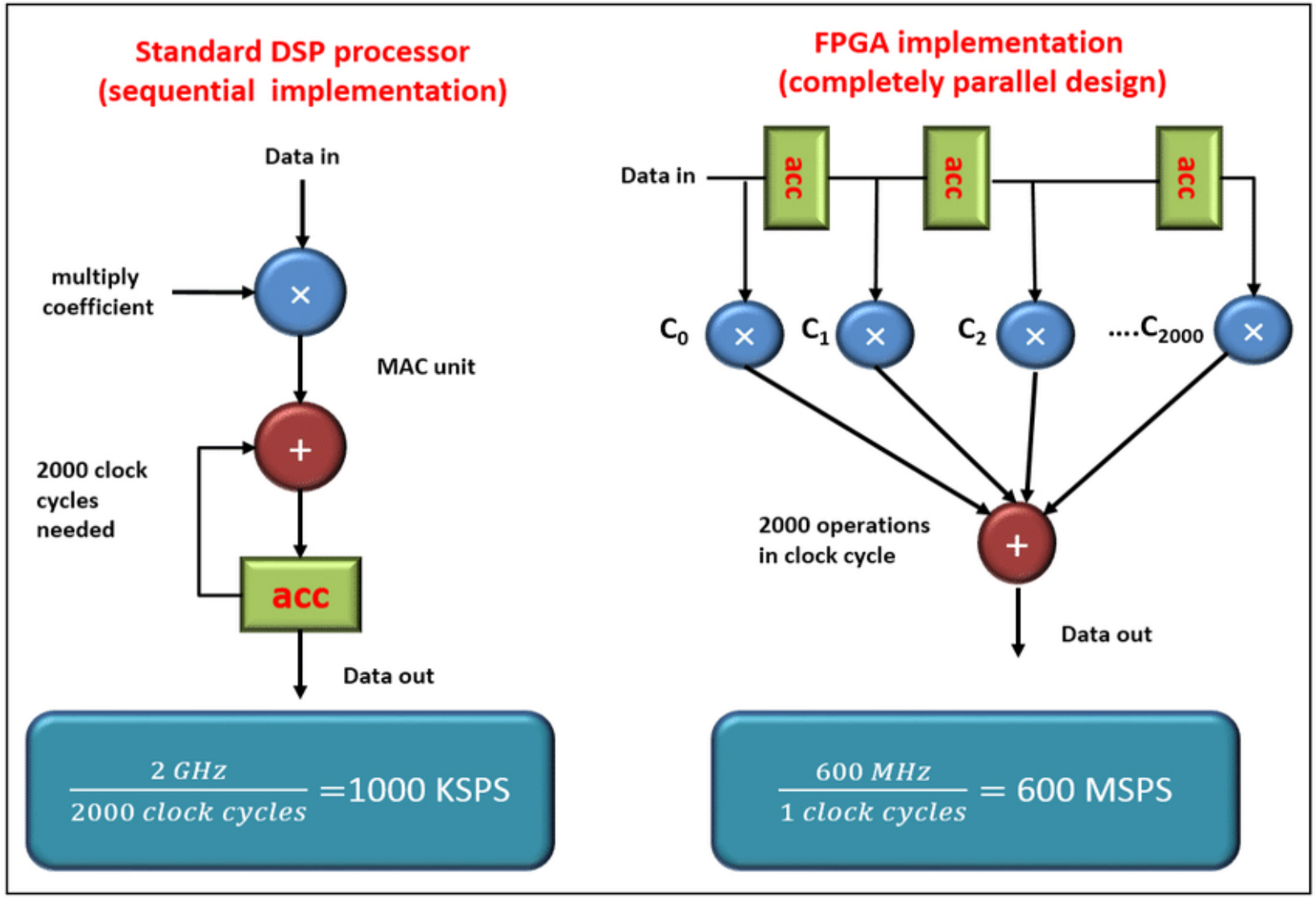
THANK YOU!

JOINT EUROPEAN MASTER IN NUCLEAR PHYSICS 2024

DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

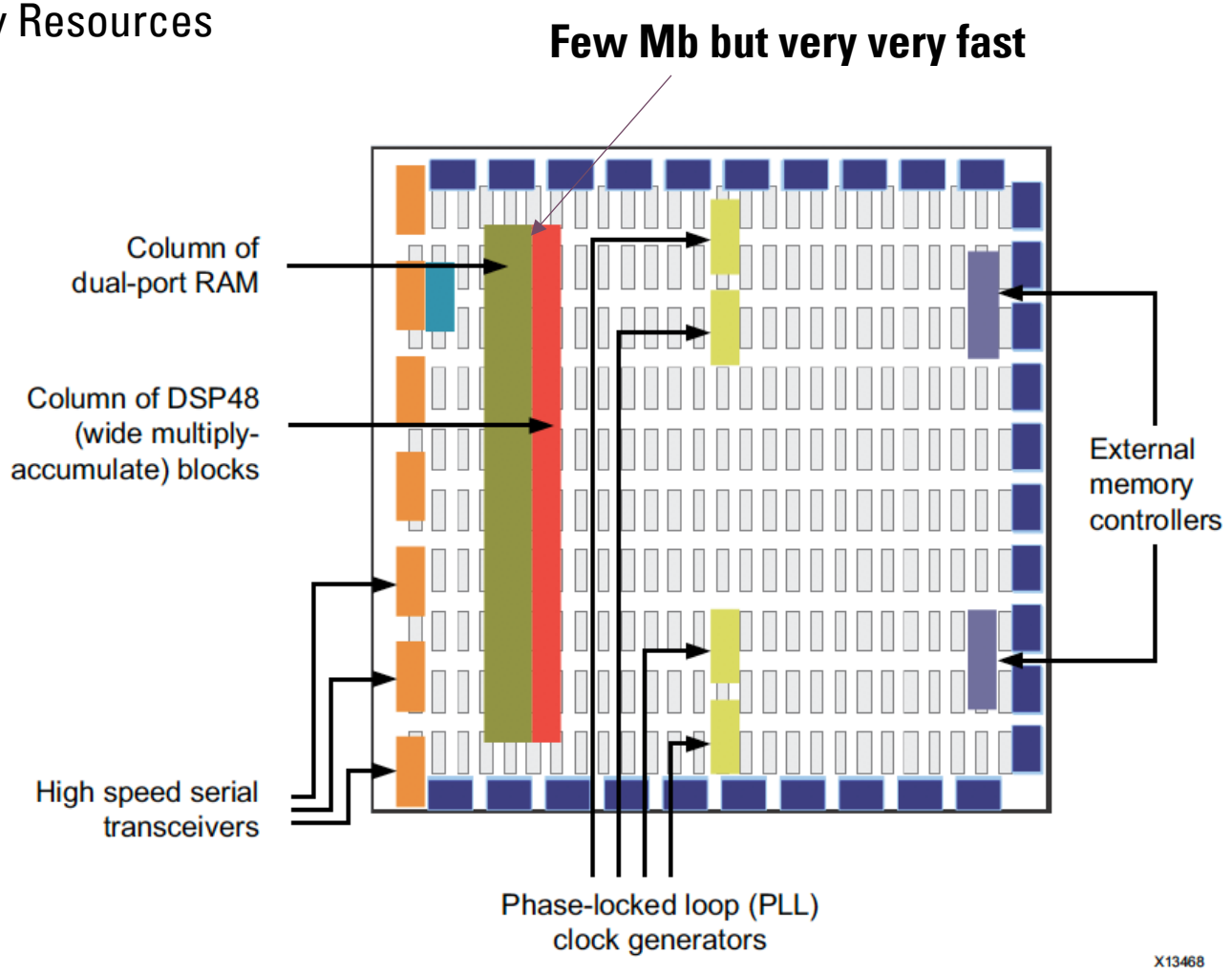
FPGA vs Microcontroller

Parallel vs Sequential



DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

Memory Resources



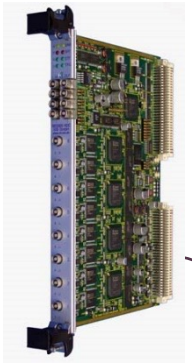
Large external DDR3/4 RAM

MODULAR ELECTRONICS READOUT SYSTEMS

- Analog and digital modules in NIM/VME
- Each module performs a specific operation
- Configuration = knobs and cables
- Poor flexibility
- Hard to debug and maintain

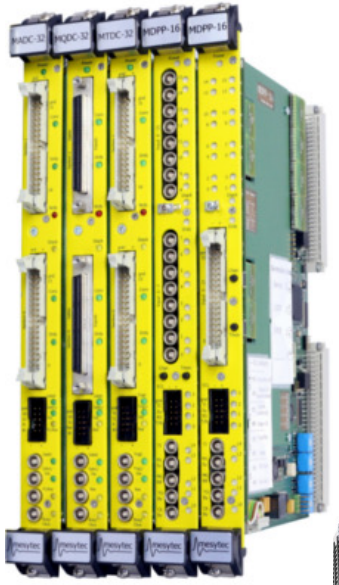
MODULAR ELECTRONICS READOUT SYSTEMS

SUB-NS
TIME MEASURAMENT



INTERCONNECTIONS

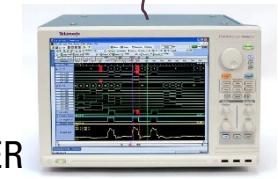
CHARGE INTEGRATION
PULSE HEIGHT ANALYSYS



GATE AND DELAY
SCALER

CUSTOM TRIGGER
AND COINCIDENCE LOGIC

WAVEFORM ACQUISITION

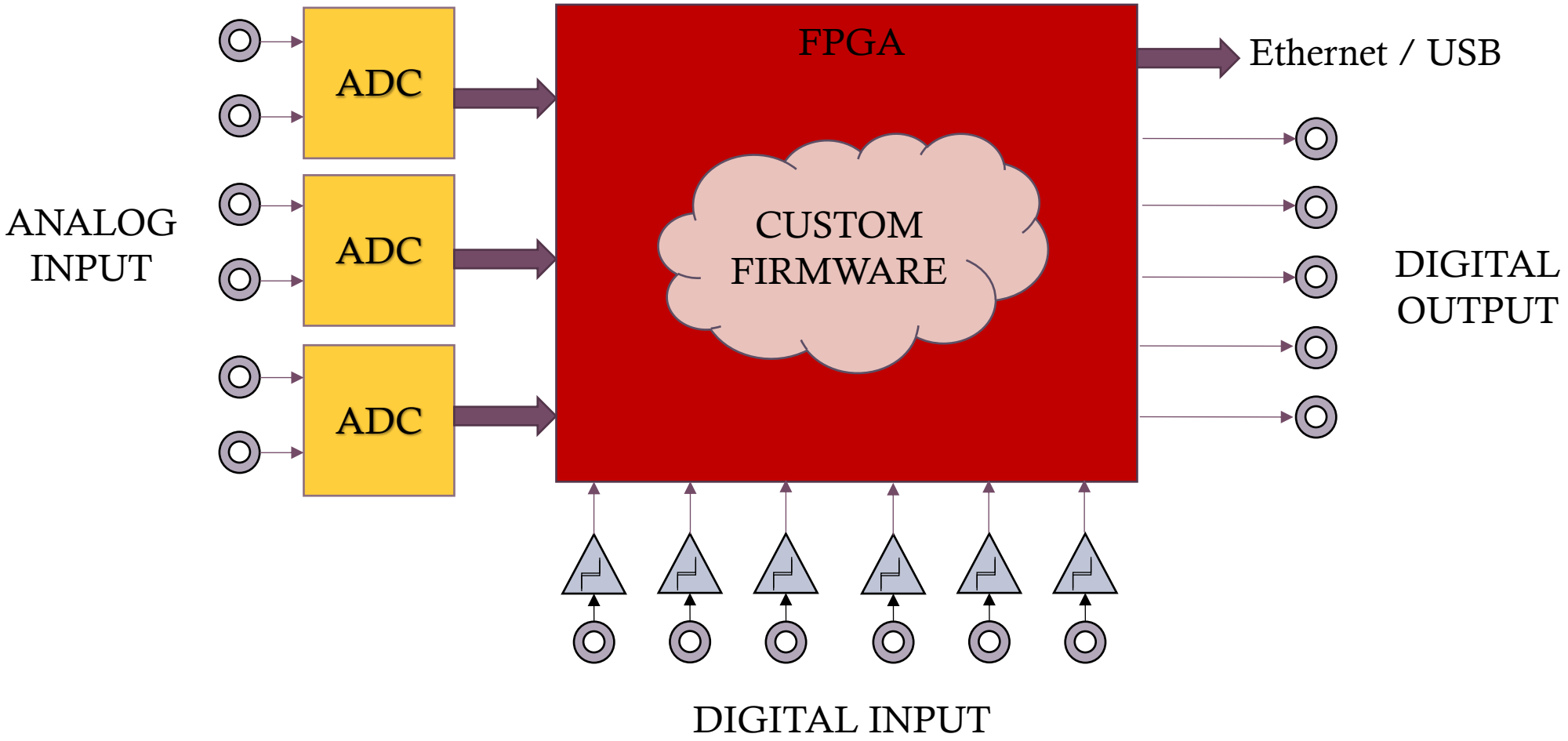


LOGIC ANALYZER

REMOTE CONTROL

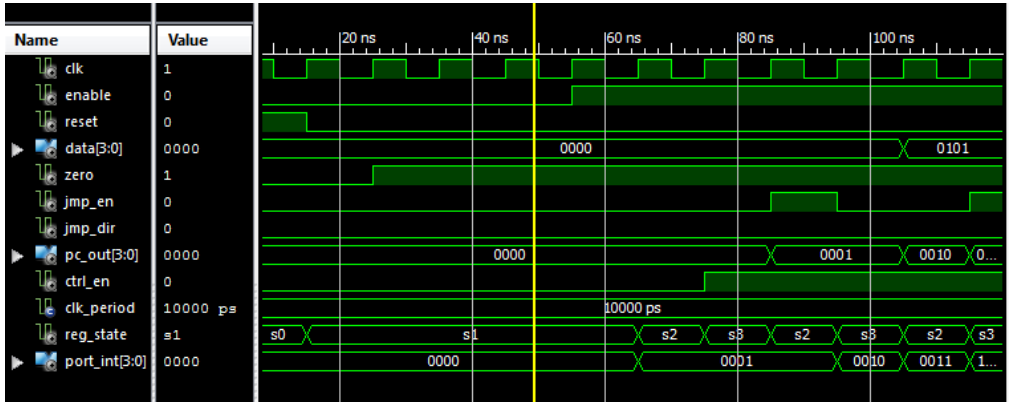
STANDARD NIM
LOGIC

REPLACE MODULAR ELECTRONICS WITH FPGA

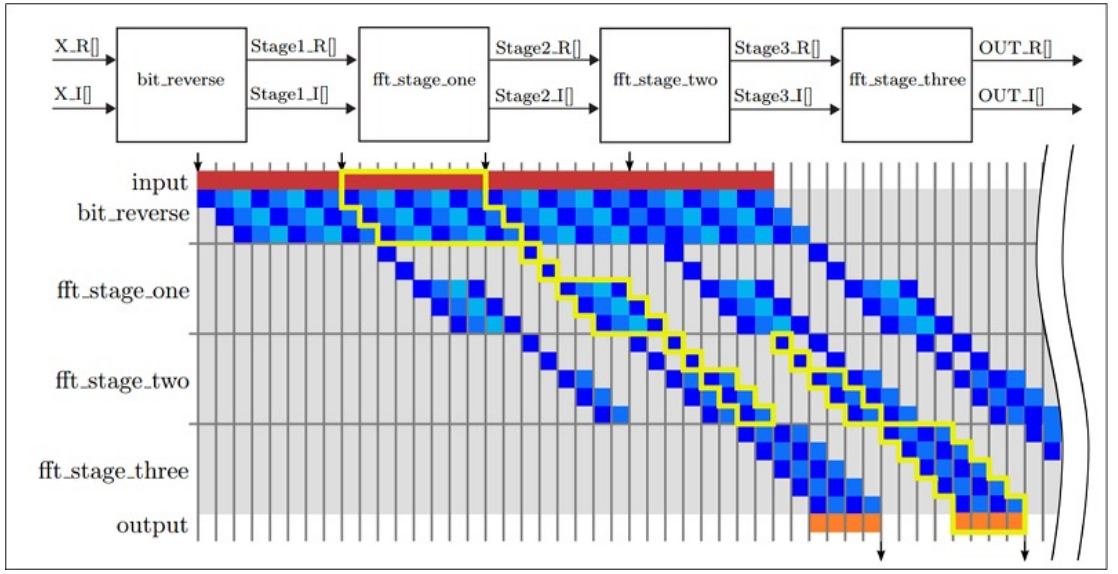


DEVELOPING IN VHDL/VERILOG IS HARD AND TIME CONSUMING

```
18  architecture Counter_Arch of AAC2M2P1 is begin
19
20      count_proc : process(CP,SR,PE,CEP,CET) begin
21
22          if (SR='0') then Q <= "0000";
23
24          elsif rising_edge(CP) then
25
26              if PE = '0' then Q <= P; --load
27
28              elsif CET = '1' and CEP = '1' then Q <= Q+1; -- count
29
30              end if;
31
32              if CET = '1' and Q = "1111" then TC <= '1'; -- Terminal count
33
34              else TC <= '0';
35
36
37              end if;
38
39          end if;
40
41      end process count_proc;
```

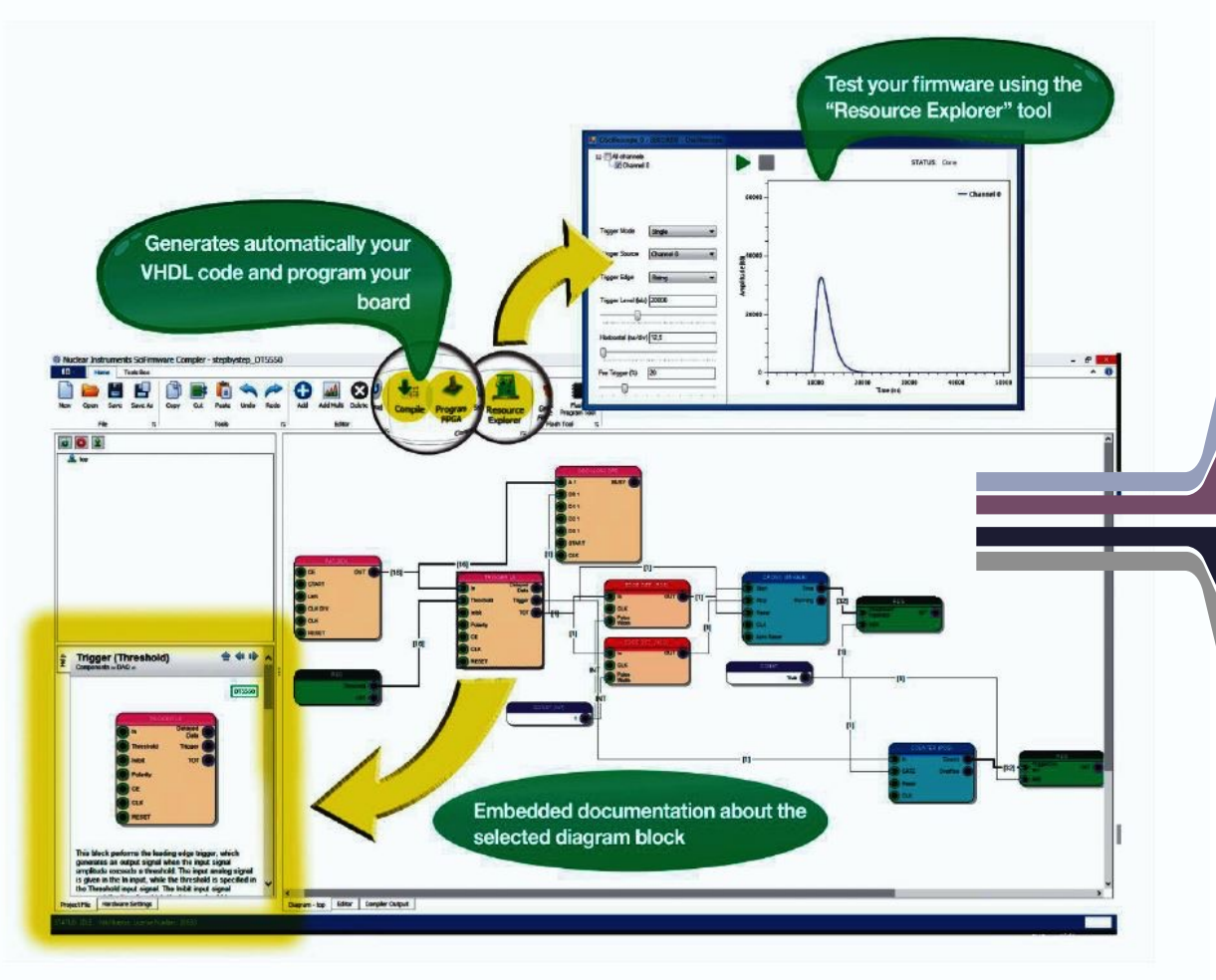


VHDL or Verilog programming skills



- ✗ VHDL/Verilog is difficult
- ✗ Deep knowledge of HW architecture
- ✗ Few instructions

WHAT IS SCI-COMPILER

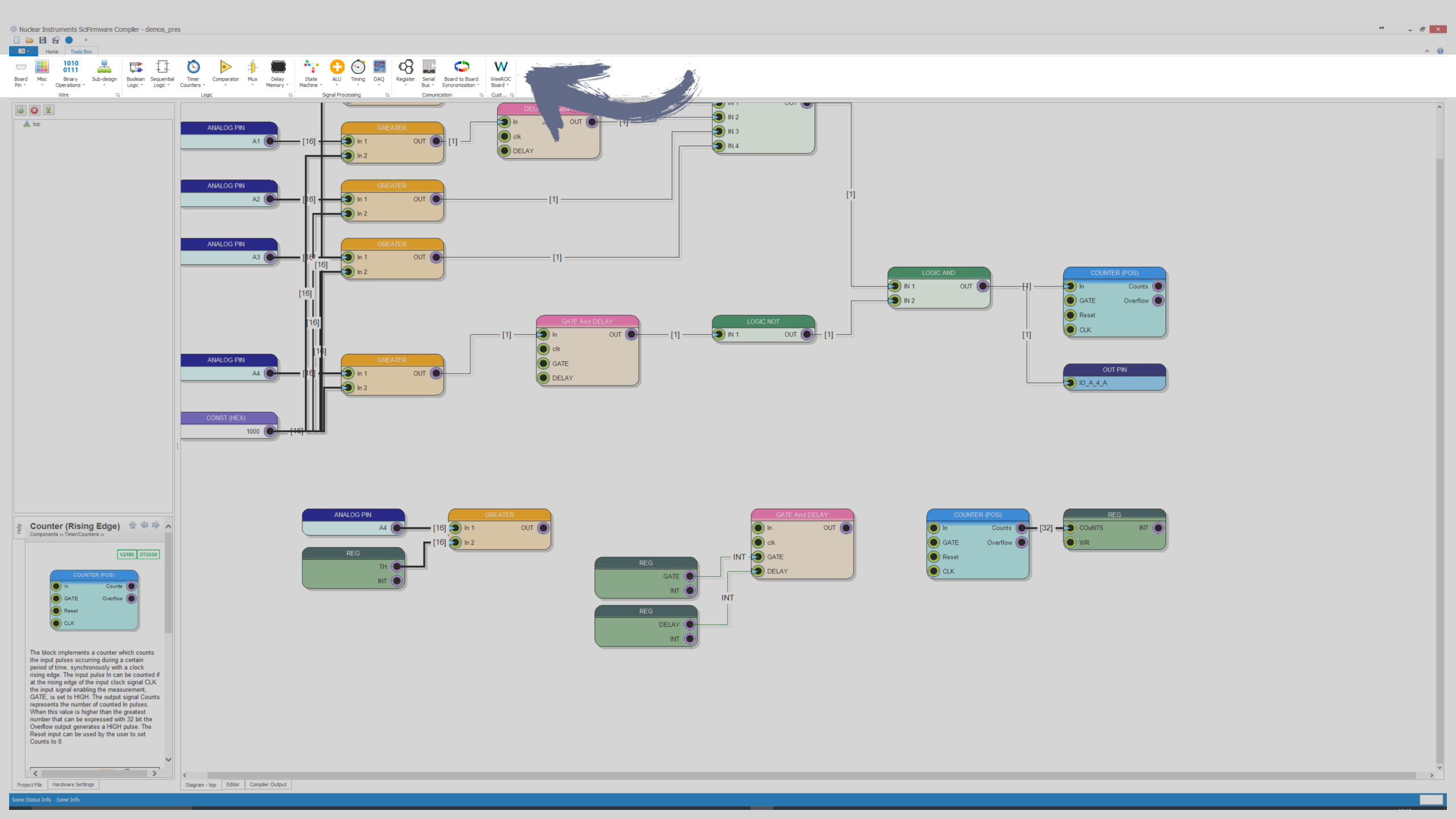


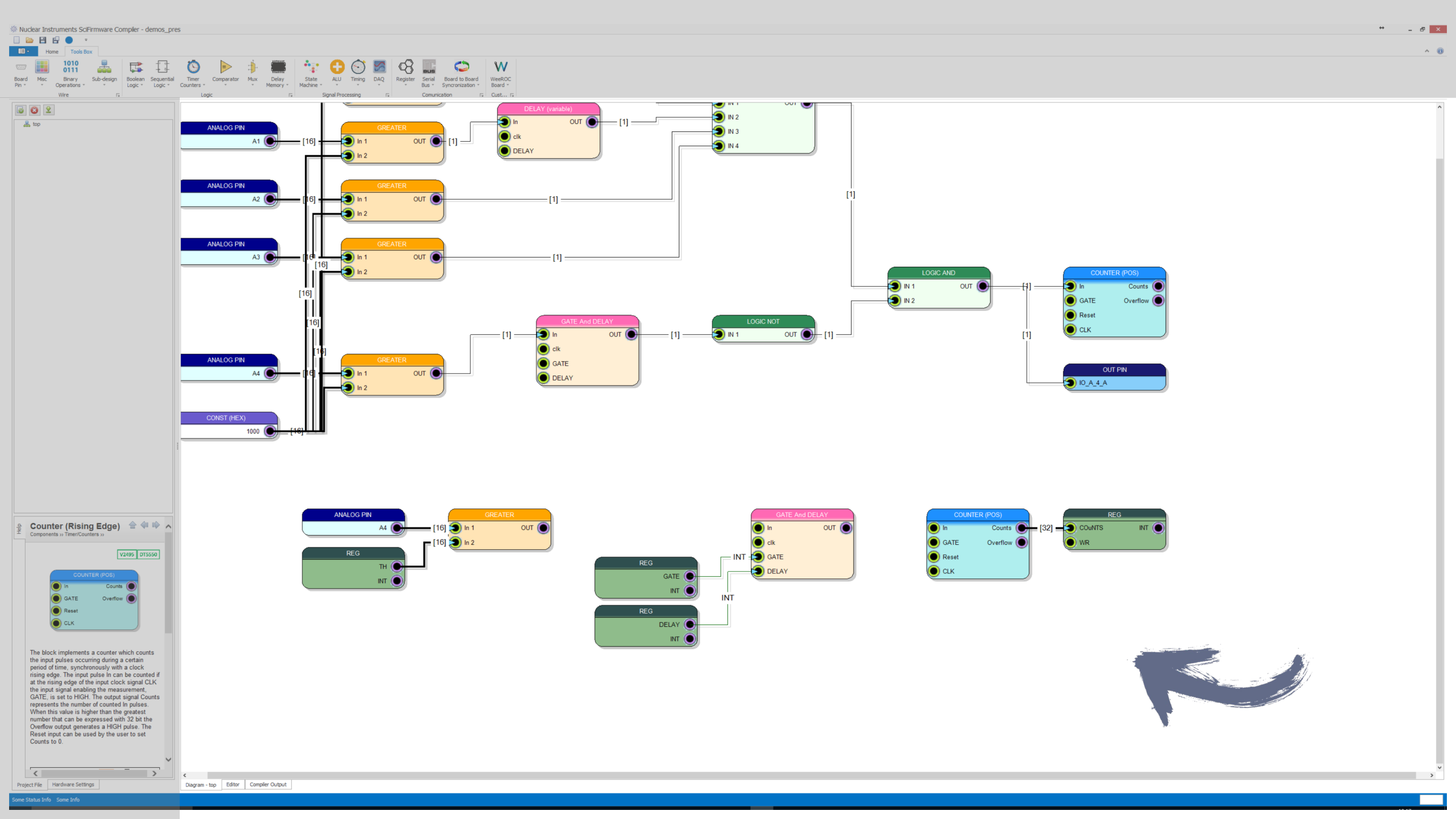
Block diagram — Define the functionalities of the firmware

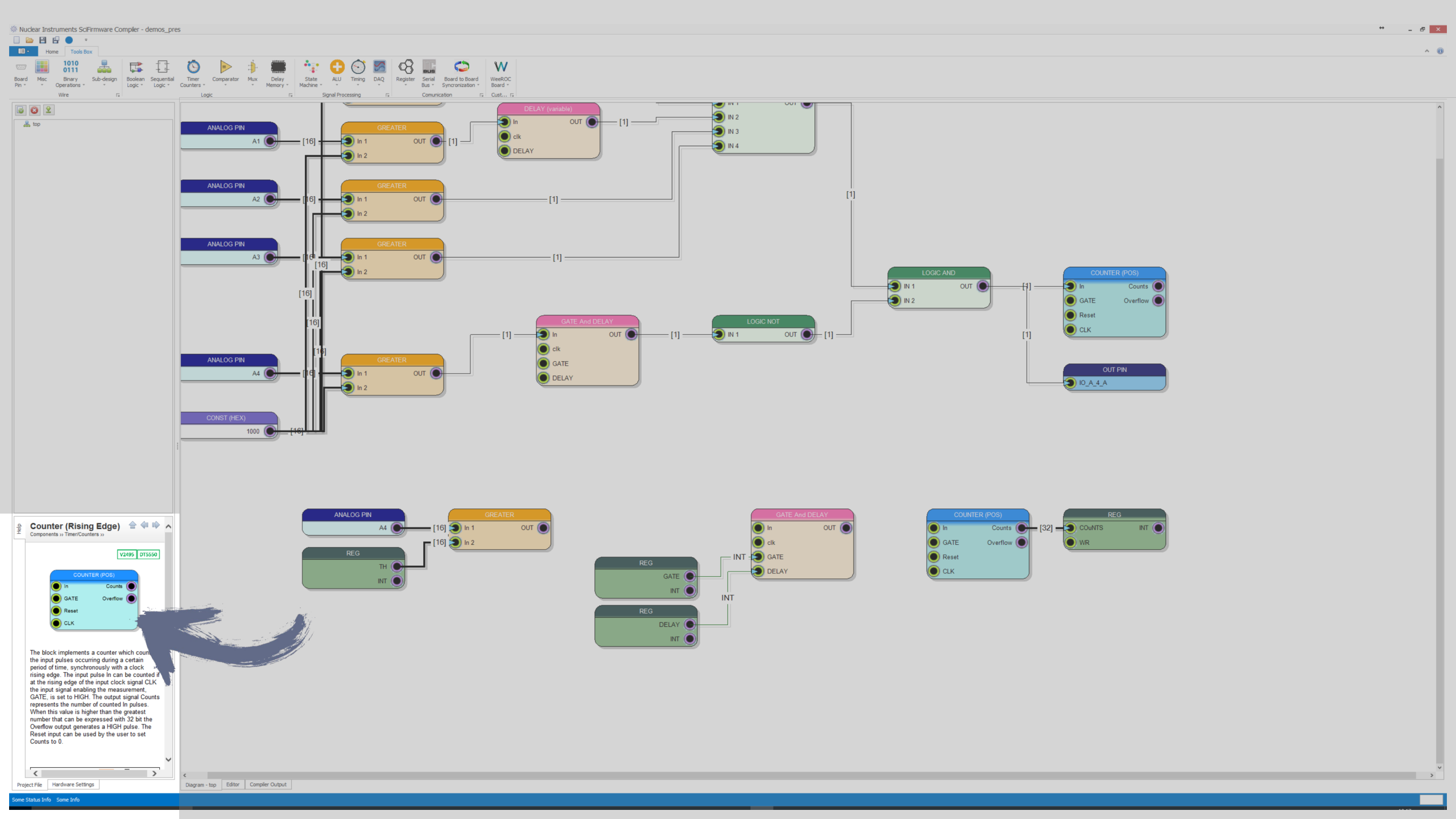
Function blocks — Build the diagram relying on a wide library of blocks for physics and nuclear engineering

Firmware — Automatic generation of VHDL and bitstream. No coding expertise required

Testing — Easy-to-use embedded tools for debugging


Nuclear Instruments SciFirmware Compiler - demos_pres
Home
Tools Box
Board Pin
Misc
Binary Operations
Sub-design
Boolean Logic
Sequential Logic
Timer Counters
Comparator
Mux
Delay Memory
State Machine
ALU
Timing
DAQ
Register
Serial Bus
Board to Board Syncronization
WeeROC Board
Wire
Logic
Signal Processing
Comunication
top
ANALOG PIN
A1
A2
A3
A4
GREATER
In 1
In 2
OUT
CONST (HEX)
1000
GATE And DELAY
In
clk
GATE
DELAY
LOGIC AND
IN 1
IN 2
LOGIC NOT
COUNTER (POS)
Counts
Overflow
Reset
CLK
OUT PIN
IO_A_4_A
REG
TH
INT
COuNTS
WR
Counter (Rising Edge)
Components ›› Timer/Counters ››
V2495
DT5550
The block implements a counter which counts the input pulses occurring during a certain period of time, synchronously with a clock rising edge. The input pulse In can be counted if at the rising edge of the input clock signal CLK the input signal enabling the measurement, GATE, is set to HIGH. The output signal Counts represents the number of counted In pulses. When this value is higher than the greatest number that can be expressed with 32 bit the Overflow output generates a HIGH pulse. The Reset input can be used by the user to set Counts to 0.
Project File
Hardware Settings
Diagram - top
Editor
Compiler Output
Some Status Info
Some Info

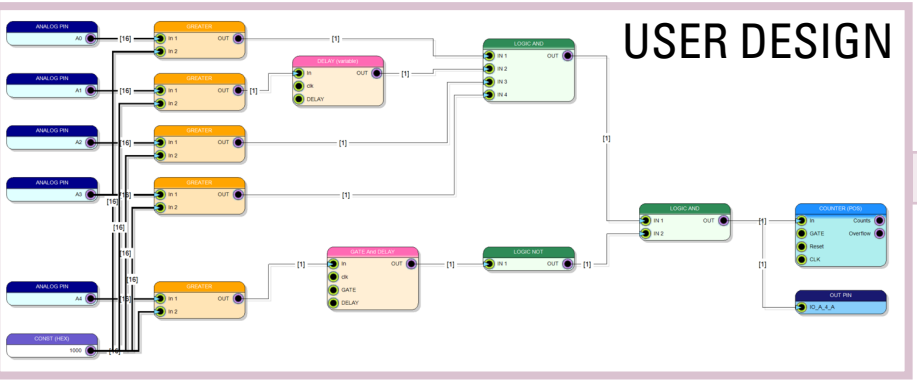

Nuclear Instruments SciFirmware Compiler - demos_pres
Home
Tools Box
Board Pin
Misc
Binary Operations
Sub-design
Wire
Boolean Logic
Sequential Logic
Timer Counters
Comparator
Mux
Delay Memory
Logic
State Machine
ALU
Timing
DAQ
Signal Processing
Register
Serial Bus
Board to Board Syncronization
Comunication
WeeROC Board
top
ANALOG PIN
A1
A2
A3
A4
GREATER
In 1
In 2
OUT
DELAY (variable)
In
clk
DELAY
IN 2
IN 3
IN 4
CONST (HEX)
1000
GATE And DELAY
GATE
LOGIC NOT
IN 1
LOGIC AND
COUNTER (POS)
Counts
Overflow
Reset
CLK
OUT PIN
IO_A_4_A
[16]
[1]
[32]
REG
TH
INT
COuNTS
WR
Counter (Rising Edge)
Components ›› Timer/Counters ››
V2495
DT5550
The block implements a counter which counts the input pulses occurring during a certain period of time, synchronously with a clock rising edge. The input pulse In can be counted if at the rising edge of the input clock signal CLK the input signal enabling the measurement, GATE, is set to HIGH. The output signal Counts represents the number of counted In pulses. When this value is higher than the greatest number that can be expressed with 32 bit the Overflow output generates a HIGH pulse. The Reset input can be used by the user to set Counts to 0.
Help
Project File
Hardware Settings
Diagram - top
Editor
Compiler Output
Some Status Info
Some Info


Nuclear Instruments SciFirmware Compiler - demos_pres
Home
Tools Box
Board Pin
Misc
Binary Operations
Sub-design
Wire
Boolean Logic
Sequential Logic
Timer Counters
Comparator
Mux
Delay Memory
Logic
State Machine
ALU
Timing
DAQ
Signal Processing
Register
Serial Bus
Board to Board Syncronization
Comunication
WeeROC Board
top
ANALOG PIN
A1
A2
A3
A4
GREATER
In 1
In 2
OUT
DELAY (variable)
In
clk
DELAY
IN 2
IN 3
IN 4
CONST (HEX)
1000
GATE And DELAY
GATE
LOGIC NOT
IN 1
LOGIC AND
COUNTER (POS)
Counts
Overflow
Reset
CLK
OUT PIN
IO_A_4_A
[16]
[1]
[32]
REG
TH
INT
COuNTS
WR
Counter (Rising Edge)
Components ›› Timer/Counters ››
V2495
DT5550
The block implements a counter which coun
the input pulses occurring during a certain
period of time, synchronously with a clock
rising edge. The input pulse In can be counted i
at the rising edge of the input clock signal CLK
the input signal enabling the measurement,
GATE, is set to HIGH. The output signal Counts
represents the number of counted In pulses.
When this value is higher than the greatest
number that can be expressed with 32 bit the
Overflow output generates a HIGH pulse. The
Reset input can be used by the user to set
Counts to 0.
Help
Project File
Hardware Settings
Diagram - top
Editor
Compiler Output
Some Status Info
Some Info

SCI-COMPILER: IP INTEGRATOR SOFTWARE

IP – Virtual Instruments

USER DESIGN

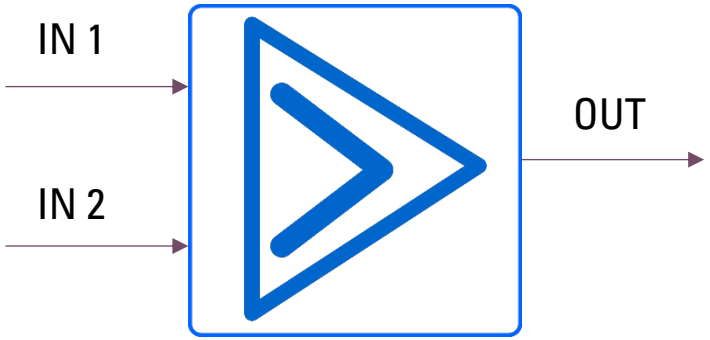


BUILDER ENGINE

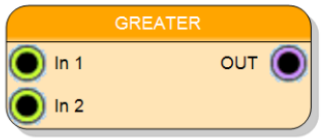
```
architecture Counter_Arch of AAC2M2P1 is begin

    count_proc : process(CP,SR,PE,CEP,CET) begin

        if (SR='0') then Q <= "0000";

        elsif rising_edge(CP) then

            if PE = '0' then Q <= P; --Load

            elsif CET = '1' and CEP = '1' then Q <= Q+1; -- count

            end if;

            if CET = '1' and Q = "1111" then TC <= '1'; -- Terminal count

            else TC <= '0';

            end if;

        end if;

    end process count_proc;
```

WHAT IS AN IP

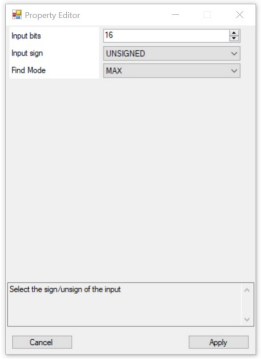
SYMBOL



COMPONENT



PROPERTY WINDOW

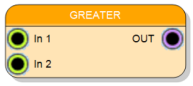


GUIDE

Greater

Components ›› Comparator ››

V2495 DT5550



The block implements a digital comparator that determines if the first input is greater than the second input. The comparison between the two inputs, which are two vectors of the same size representing numbers expressed in binary notation, is done bit by bit starting from the MSB and proceeding towards the LSB. When an inequality between two bits is found, if the considered bit of the first input vector is 1 and the correspondent bit of the second input vector is 0 it means that the first number is greater than the second. In this case the output of the block is a 1, otherwise it is a 0, meaning that the first input number is smaller than the second input number.

In 1	VECT	Size: 1...32

First Input signal

In 2	VECT	Size: 1...32

Second Input signal

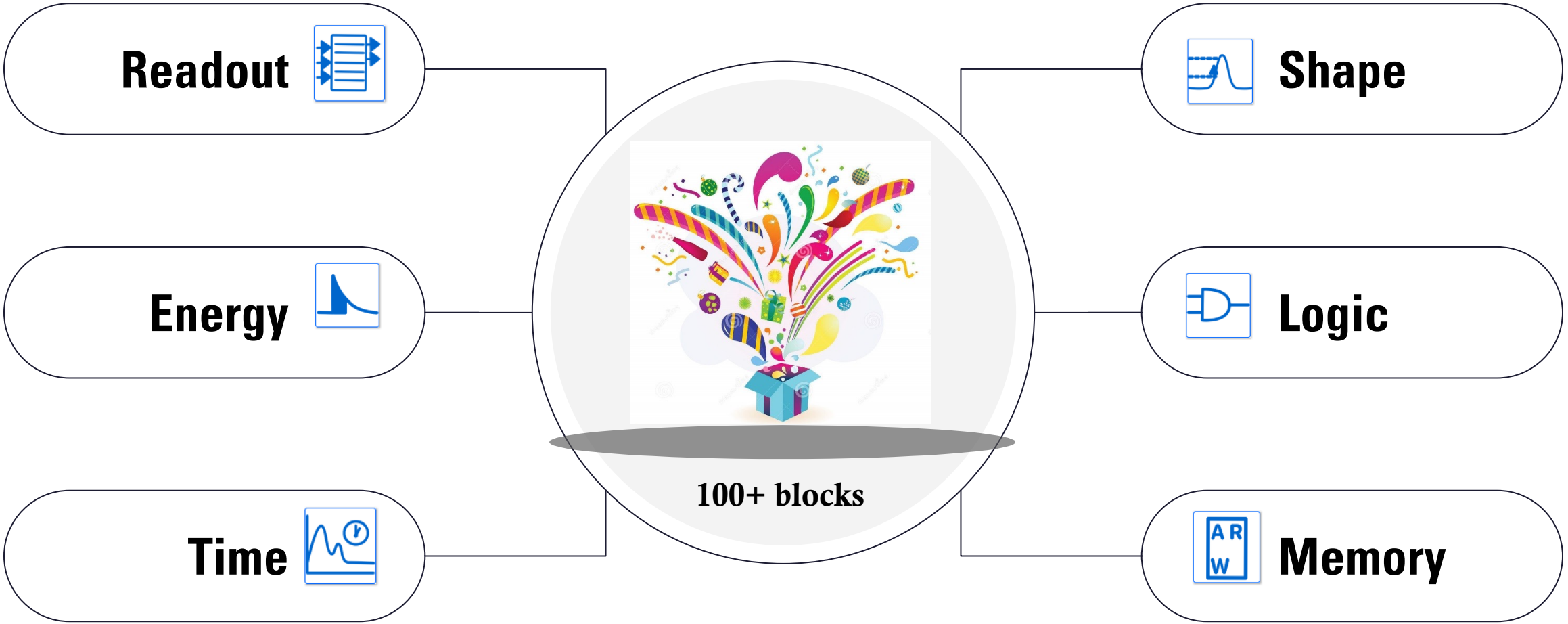
OUT	VECT	Size: 1

Output signal consisting of a 1 if the first input is greater than the second, in a 0 otherwise

IMPLEMENTATION

```
entity comparator is
    Generic (IN_SIZE : integer := 16;
             OPERATION : STRING := "equal";
             IN_SIGN : STRING := "signed";
             REGISTER_OUT : STRING := "true"
             );
    Port ( in1 : in STD_LOGIC_VECTOR (IN_SIZE-1 downto 0);
             in2 : in STD_LOGIC_VECTOR (IN_SIZE-1 downto 0);
             clk : in STD_LOGIC;
             comp_out : out STD_LOGIC);
end comparator;

architecture Behavioral of comparator is
signal i_out : std_logic := '0';
begin

IF_equal:
   if OPERATION = "equal" generate
      begin
         i_out <= '1' when in1=in2 else '0';
   end generate;


IF_notequal:
   if OPERATION = "not_equal" generate
      begin
         i_out <= '0' when in1=in2 else '1';
   end generate;
```

IP LIBRARY



COMPARISON WITH VHDL

Block Diagram

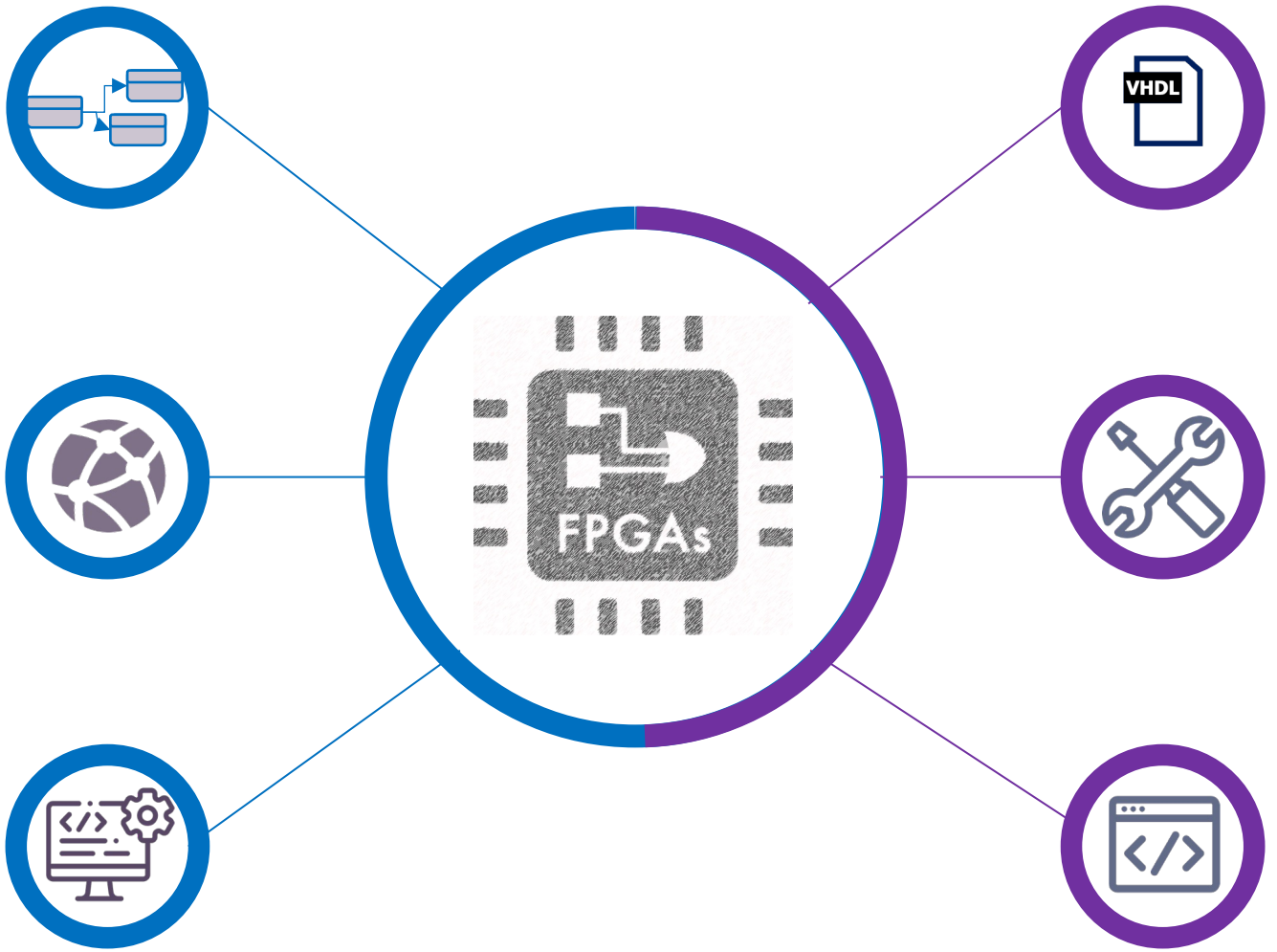
Focus on functional specifications only

All-in-one

Simulate, compile and test the firmware in the same environment

Software

High-level C++/Python SDK compatible with custom firmware



Hard-coding

Needs deep knowledge of target hardware

External tools

Different tools for simulation, compilation and testing

Testbench

HDL to C++ mapping

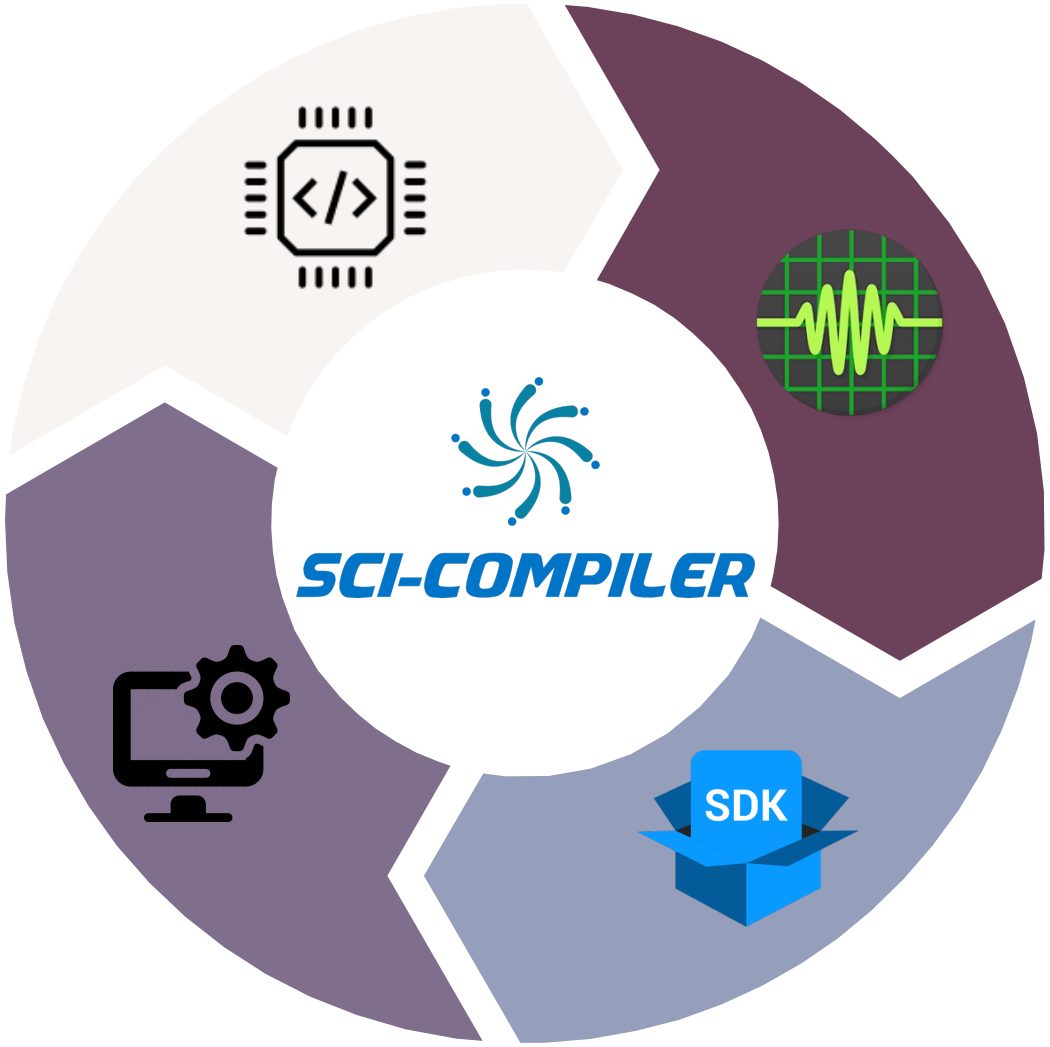
MAIN FEATURES

Compilation

Start from a block diagram to generate a custom firmware

Simulation

Embedded simulation tool based on GTKWave



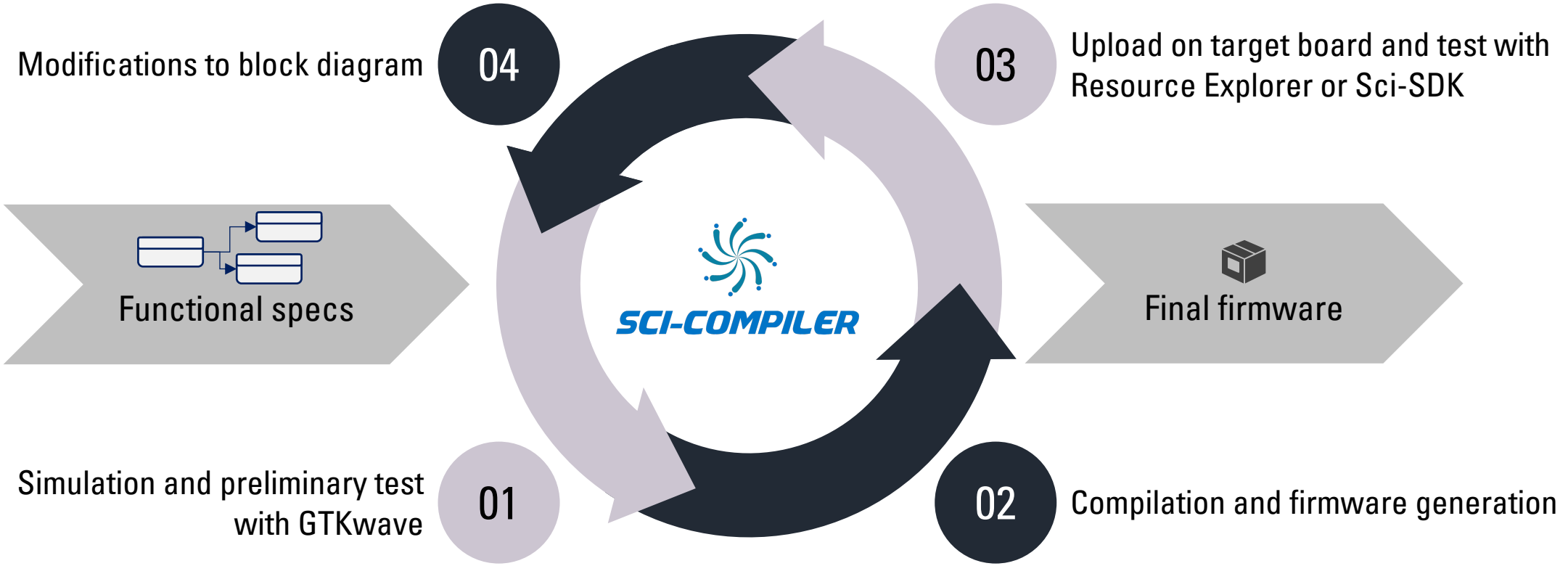
Resource Explorer

Embedded software for initial debug and testing

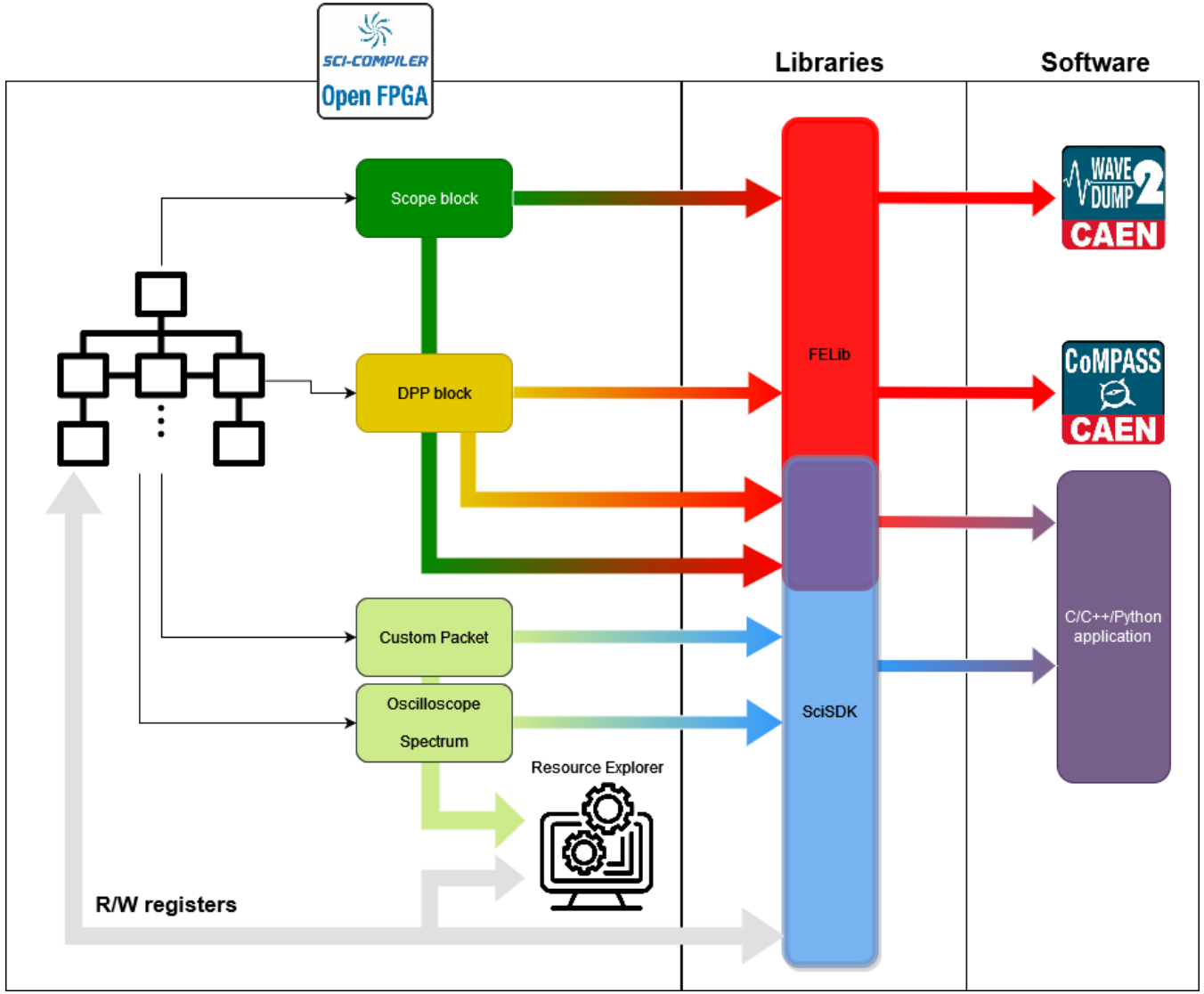
Sci-SDK

Adaptive SDK for any custom firmware and target hardware

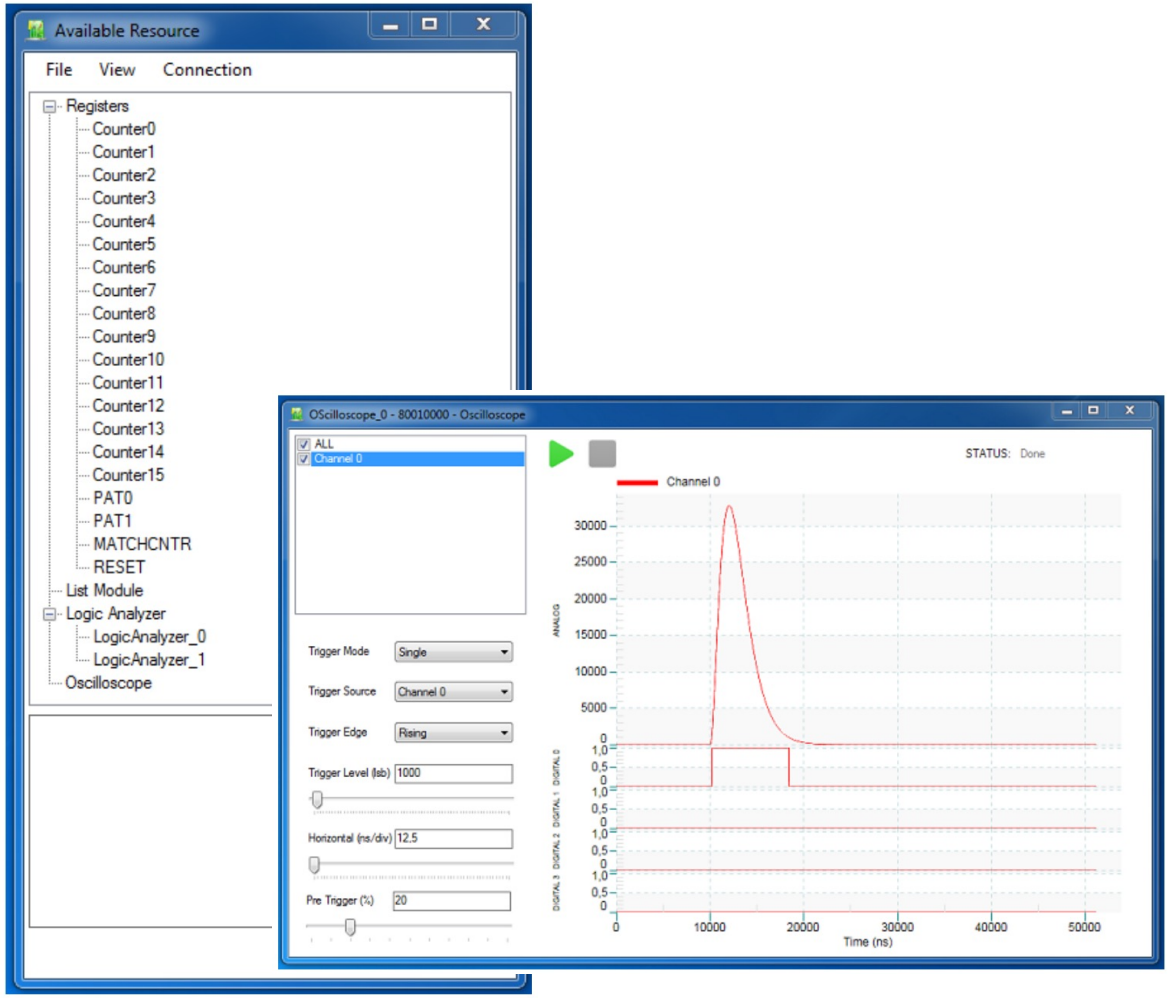
DESIGN FLOW


Modifications to block diagram
04
03
Upload on target board and test with Resource Explorer or Sci-SDK
Functional specs
SCI-COMPILER
Final firmware
Simulation and preliminary test with GTKwave
01
02
Compilation and firmware generation

DATA READOUT



DATA READOUT – RESOURCE EXPLORER AND SCISDK

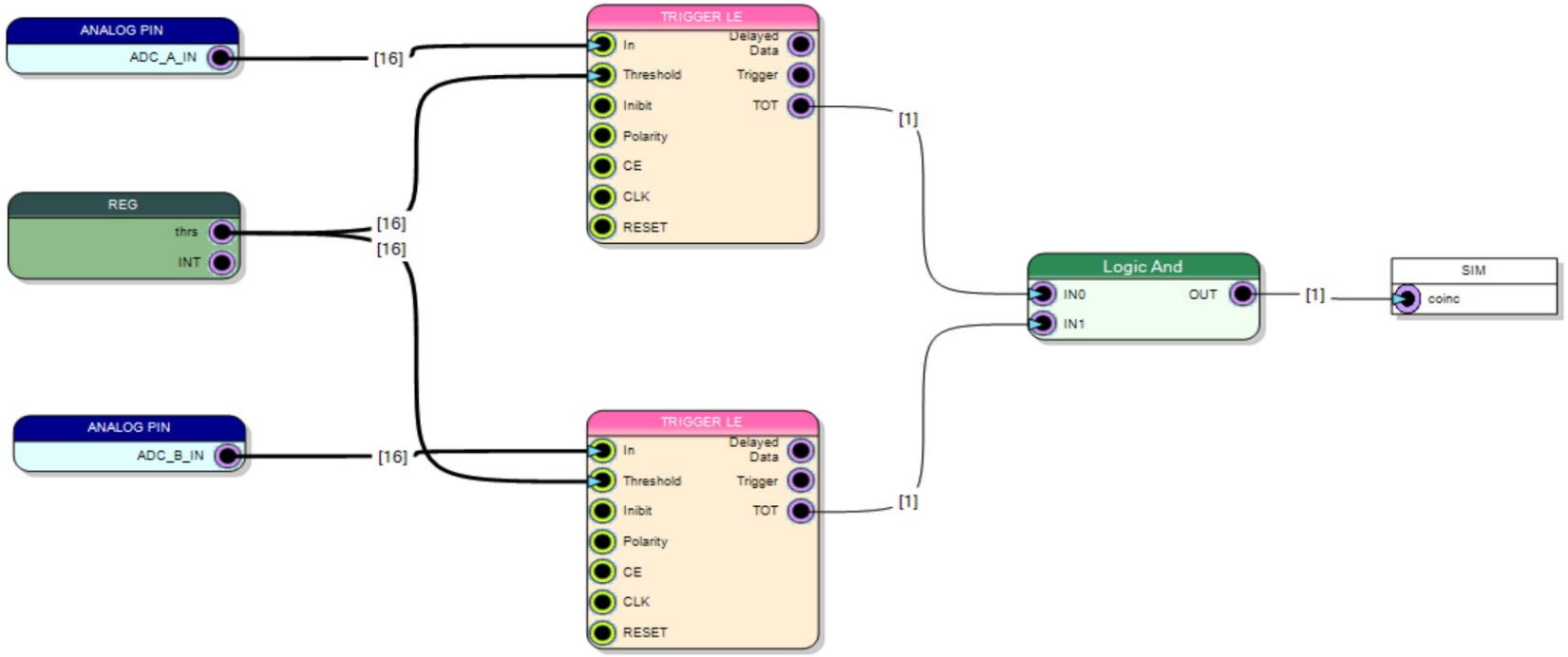



```c
int main()
{
    SCISDK_OSCILLOSCOPE *osc_data;

    sdk.AddNewDevice("usb:0006", "dt1260", "SCIDKTester.json", "board0");
    sdk.StrobeRegister("Registers/res", "pos");

    sdk.SetParameter("Oscilloscope_0/decimator", 10);
    sdk.SetParameter("Oscilloscope_0/trigger", "ext");
    sdk.SetParameter("Oscilloscope_0/acq_mode", "blocking");
    sdk.SetParameter("Oscilloscope_0/data_processing", "decode");

    sdk.ExecuteCommand("Oscilloscope_0", "start");

    sdk.AllocateBuffer("Oscilloscope_0", "decoded_buffer", (void **) &osc_data);

    for (int i = 0; i < 10; i++) {
        sdk.ReadData("Oscilloscope_0", osc_data);
        dump_to_file(osc_data);
    }

    return 0;
```

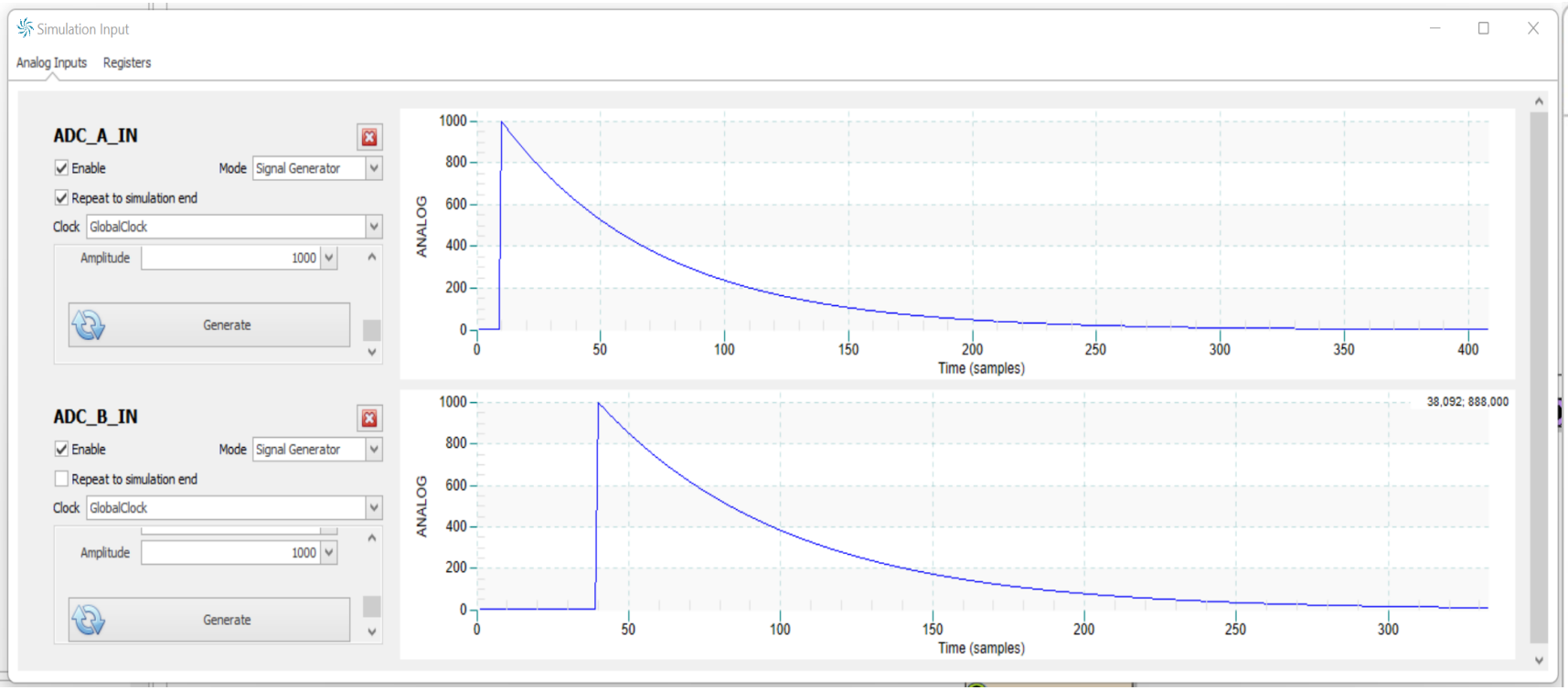
python™ VB.NET

INTEGRATED SIMULATION

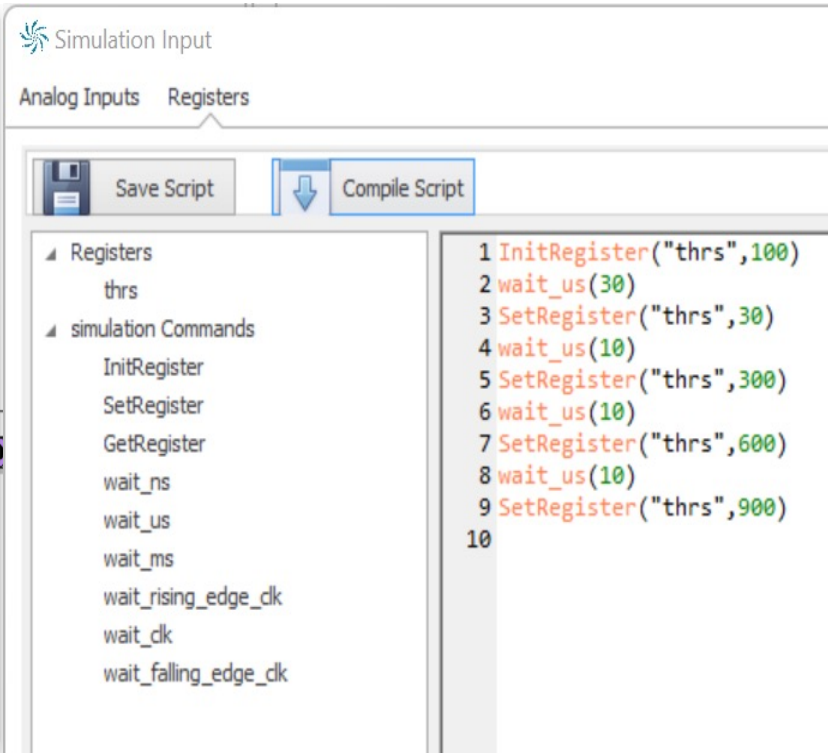
- Save your **time** → Compile can require hours, simulation few seconds
- Simpler debug → You can **inspect any net and signal**
- Better test **coverage** → You can insert critical probes and input stimulus to check how firmware perform



INTEGRATED SIMULATION – STIMULUS AND REGISTERS

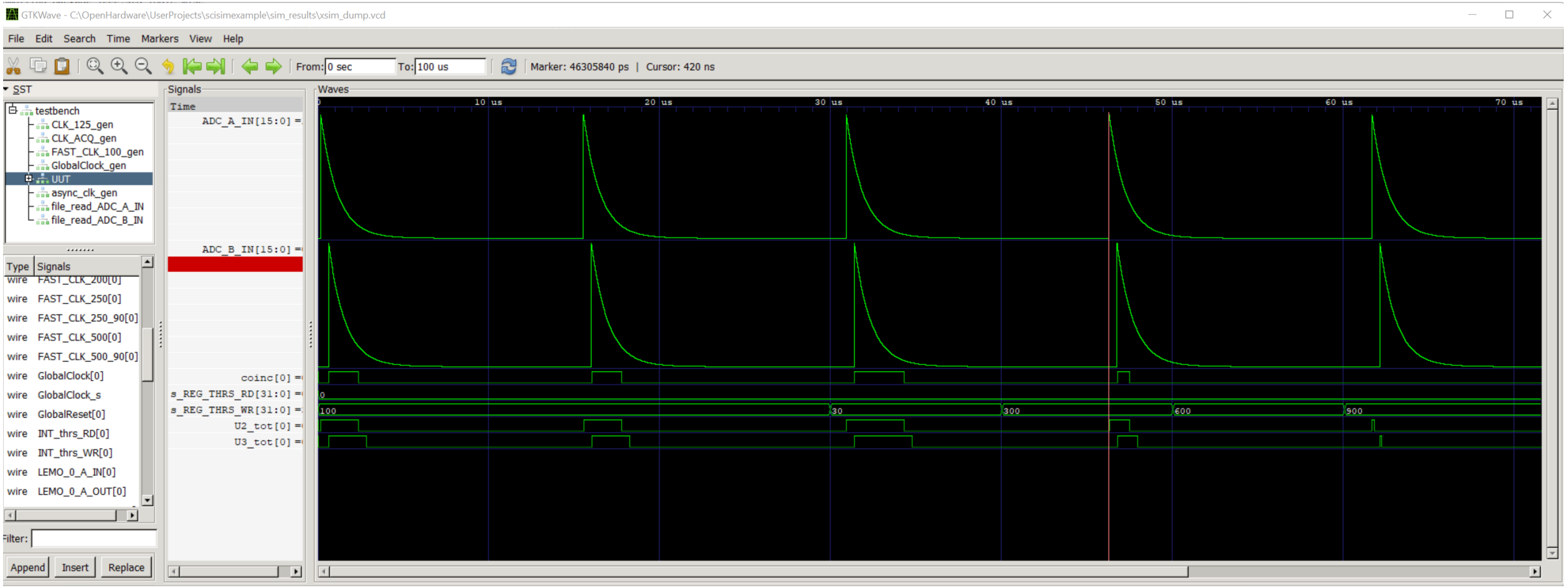


Generate analog signal to replace ADC data stream

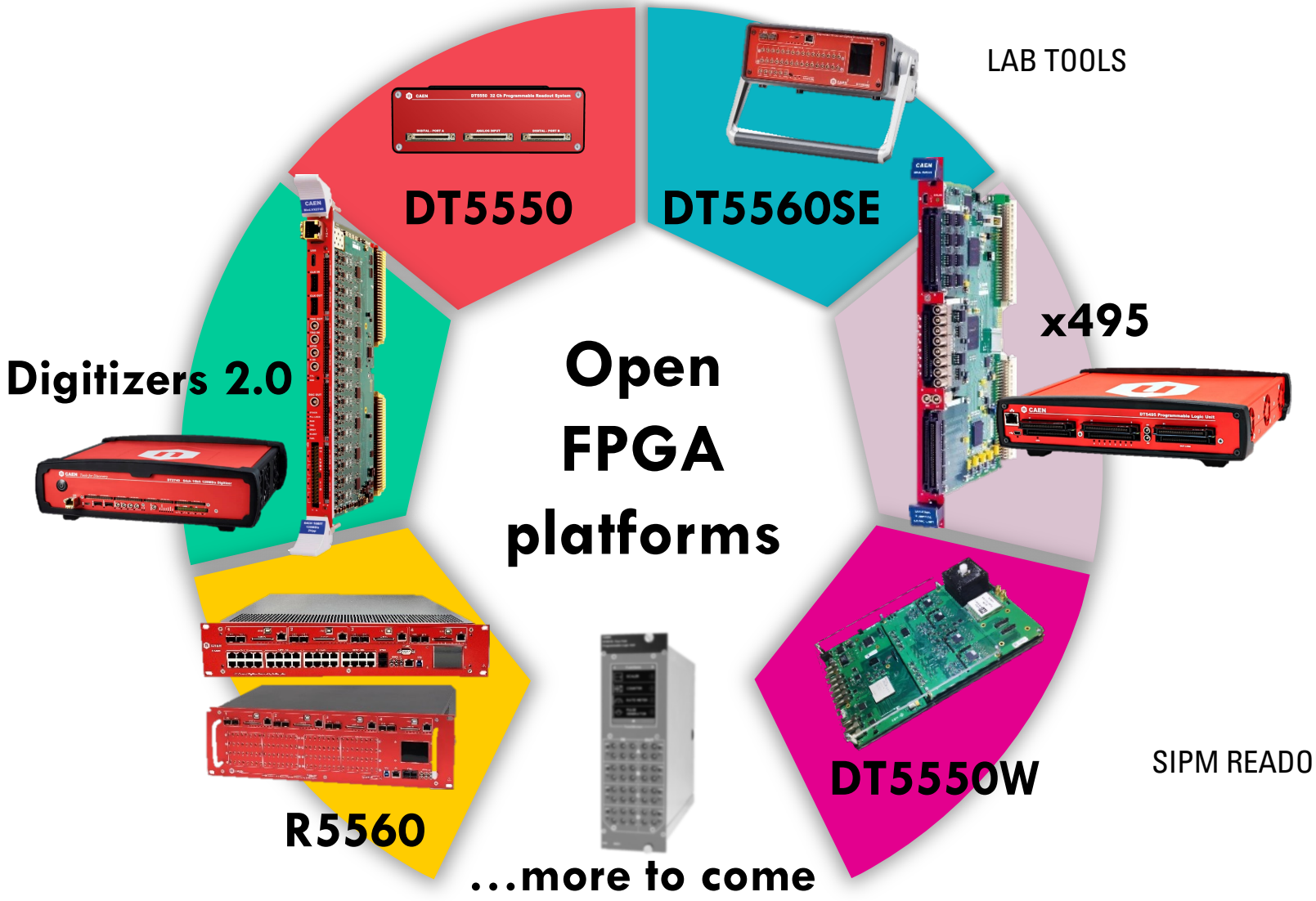


Write script to simulate Register RW

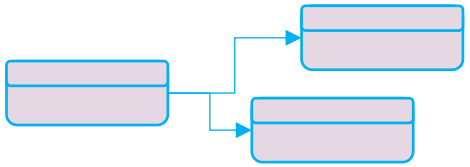
INTEGRATED SIMULATION - GTKWAVE



COMPATIBLE HARDWARE



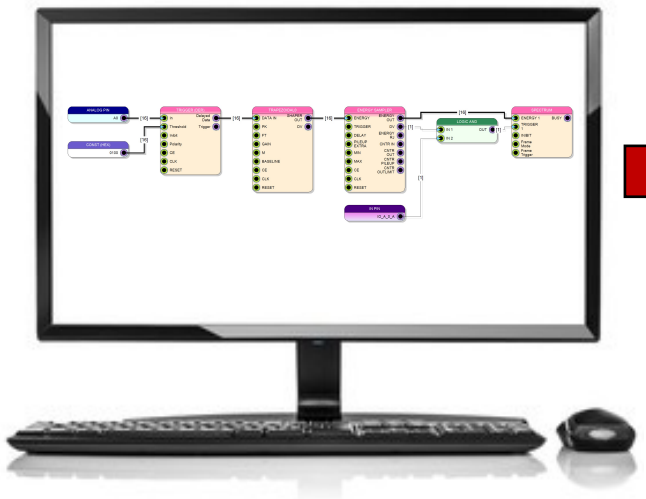
REMOTE SERVICE BASED ON MYCAEN CLOUD

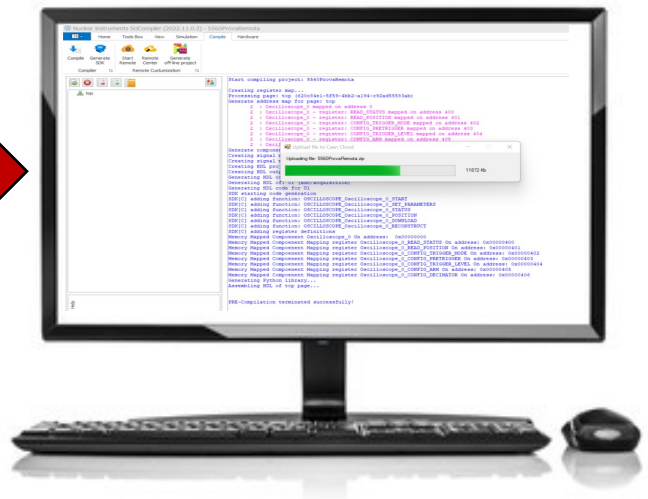


Design entry with SciCompiler

Upload project on the cloud

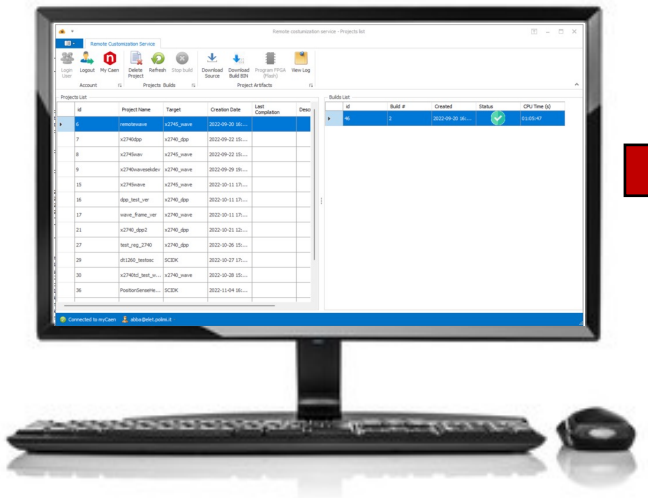
Remote Compile on MyCaen





REMOTE SERVICE BASED ON MYCAEN CLOUD

Download the firmware compiled



Install the firmware on the device

HANDS-ON

INTERNATIONAL SUMMER SCHOOL 2024

A new five-day international summer school, featuring an advanced program of courses, will be offered at the CAEN S.p.A Headquarters from July 22nd to 26th, 2024.

Students from different Universities and the presence of the Erasmus Mundus Master students are the ingredients of this exiting event!

Beginning with the fundamentals of physics, the courses will cover a range of applications of radiation detectors and nuclear measurements across diverse fields.

Emphasis will be placed on hands-on laboratory sessions utilizing cutting-edge nuclear instrumentation equipment.

SLIDE TITLE

Add text here

Add text here

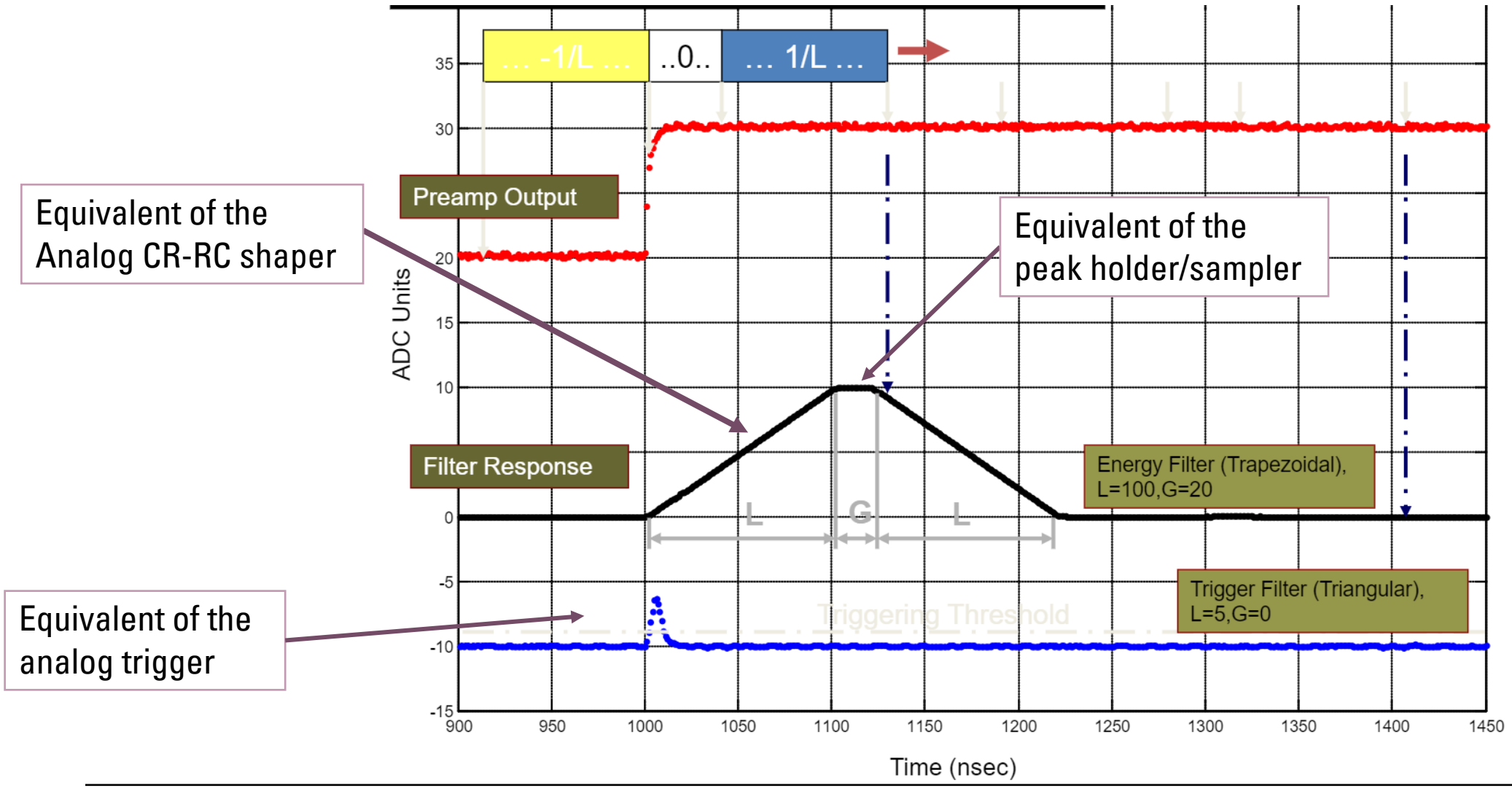
Add text here

Add text here

SLIDE TITLE

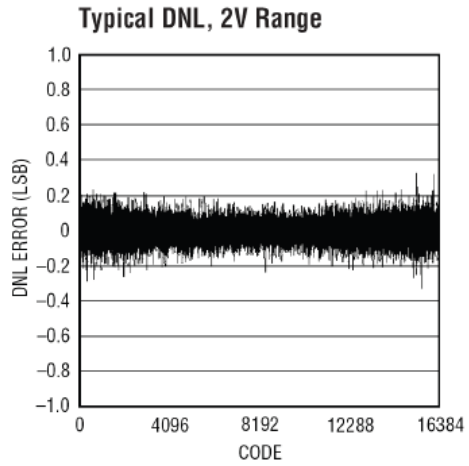
- Add text here
- Add text here
- Add text here
- Add text here

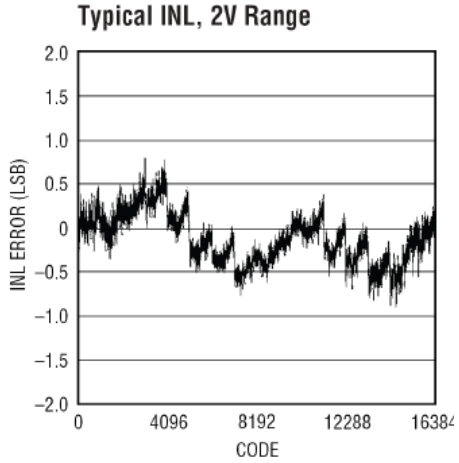
DIGITAL SIGNAL PROCESSING: DIGITAL SHAPER

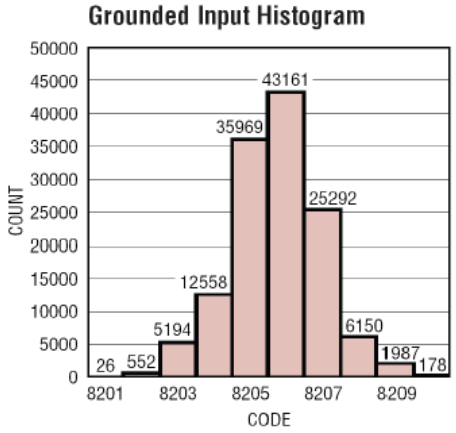


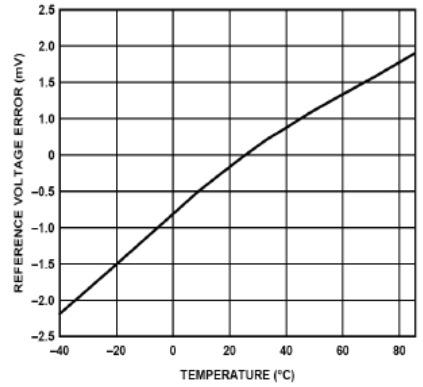
DIGITALIZATION OF THE SIGNAL

ADC Data Sheets





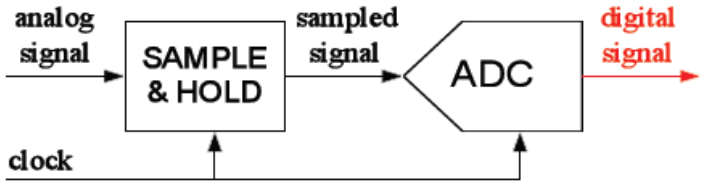


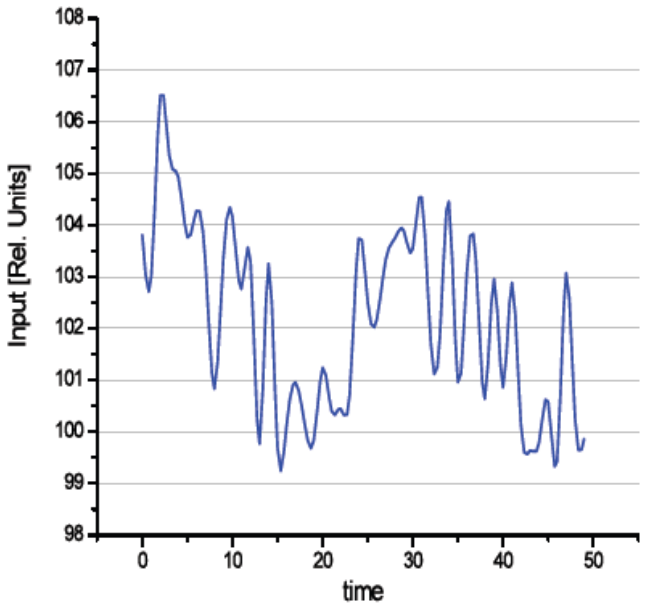


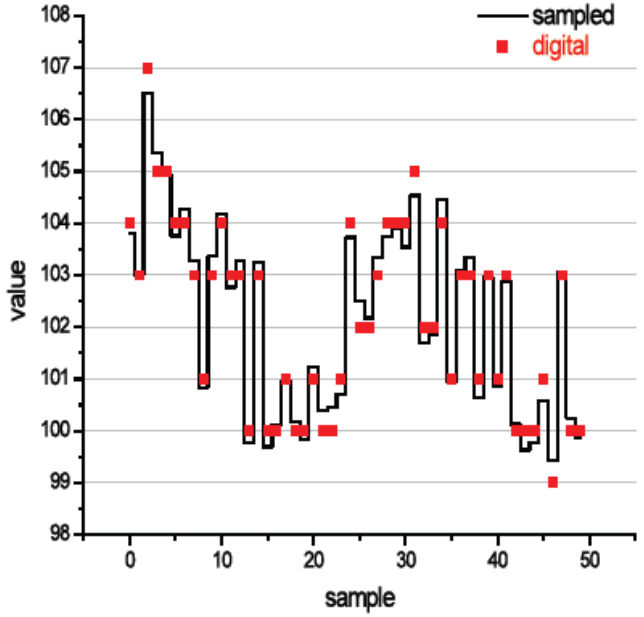
Linear Technology

DIGITALIZATION OF THE SIGNAL

Sampling ADC

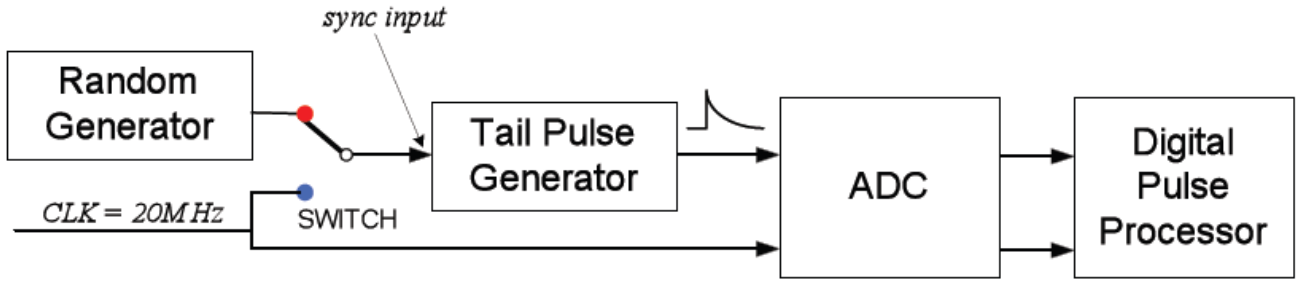




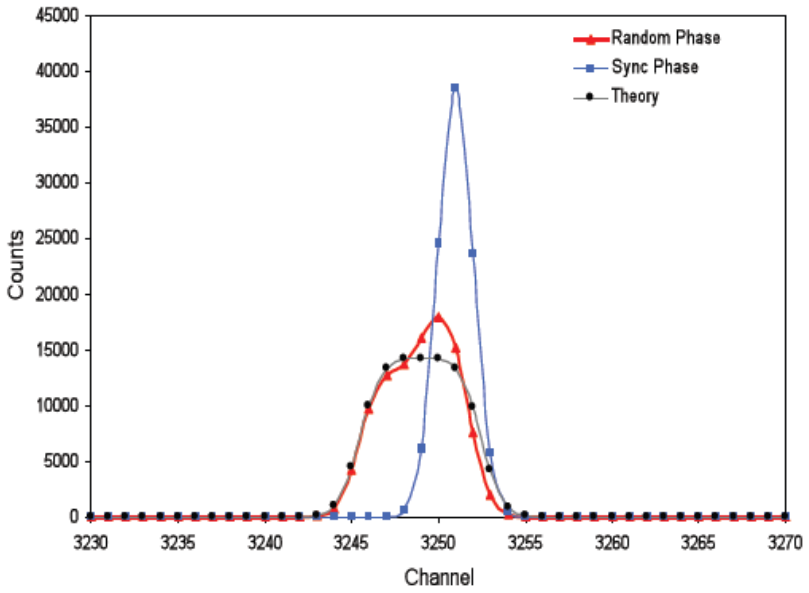


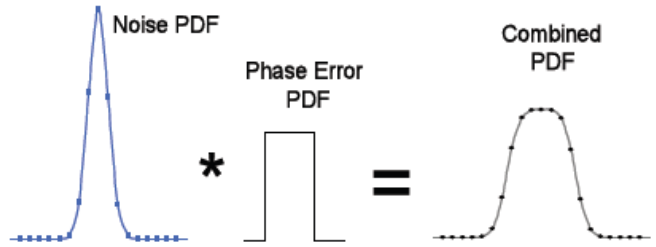
DIGITALIZATION OF THE SIGNAL

Phase Error Example – 20MHz Clock



$M_1 = 128,\ M2 = 0.5$



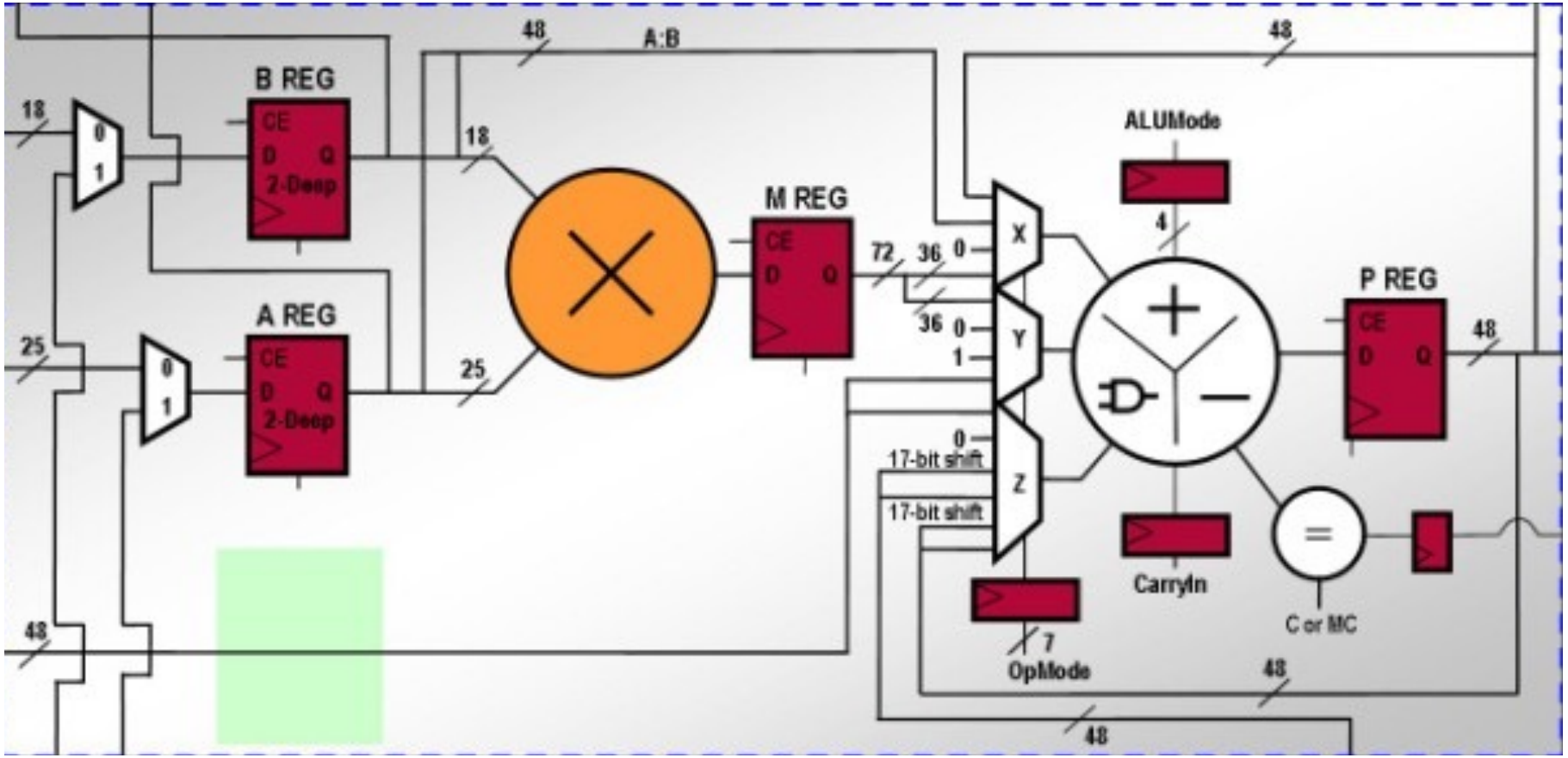


Probability Density Functions (PDF) Convolution

Phase Error PDF is approximately rectangular with about 7 channels width.

DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

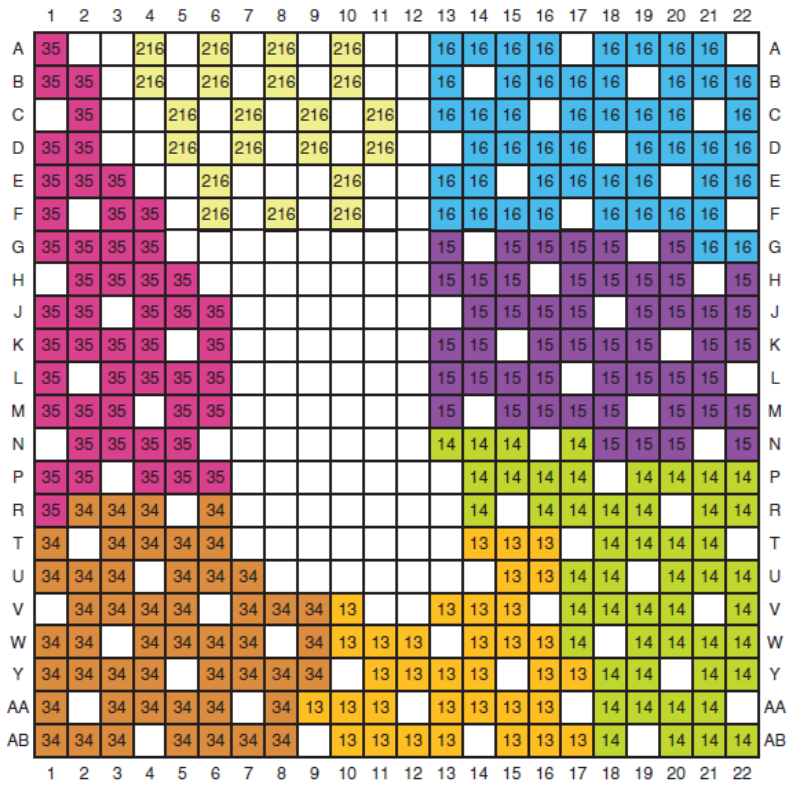
DSP / Embedded multiplier

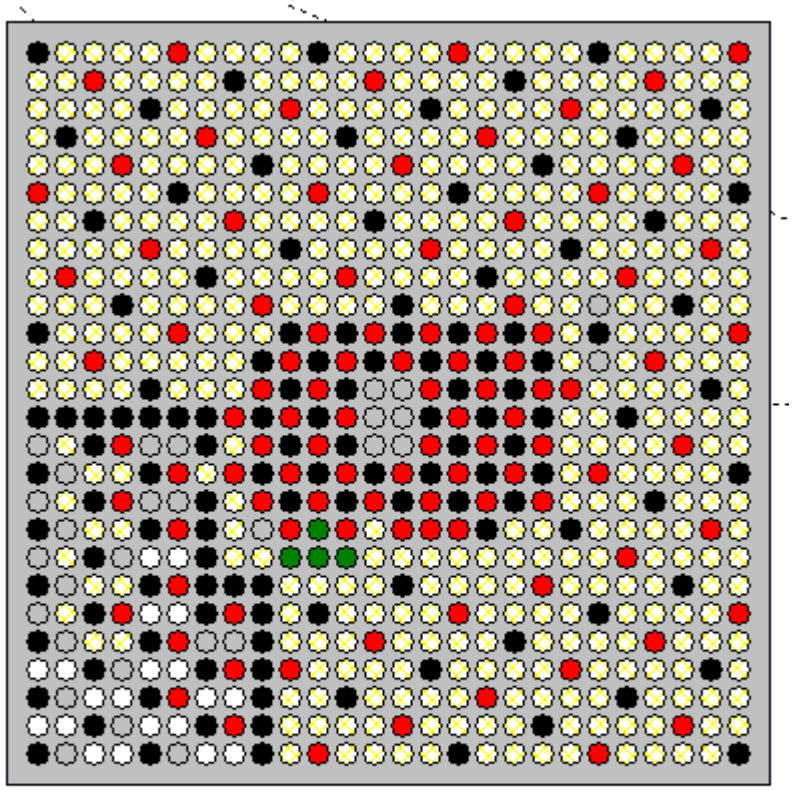


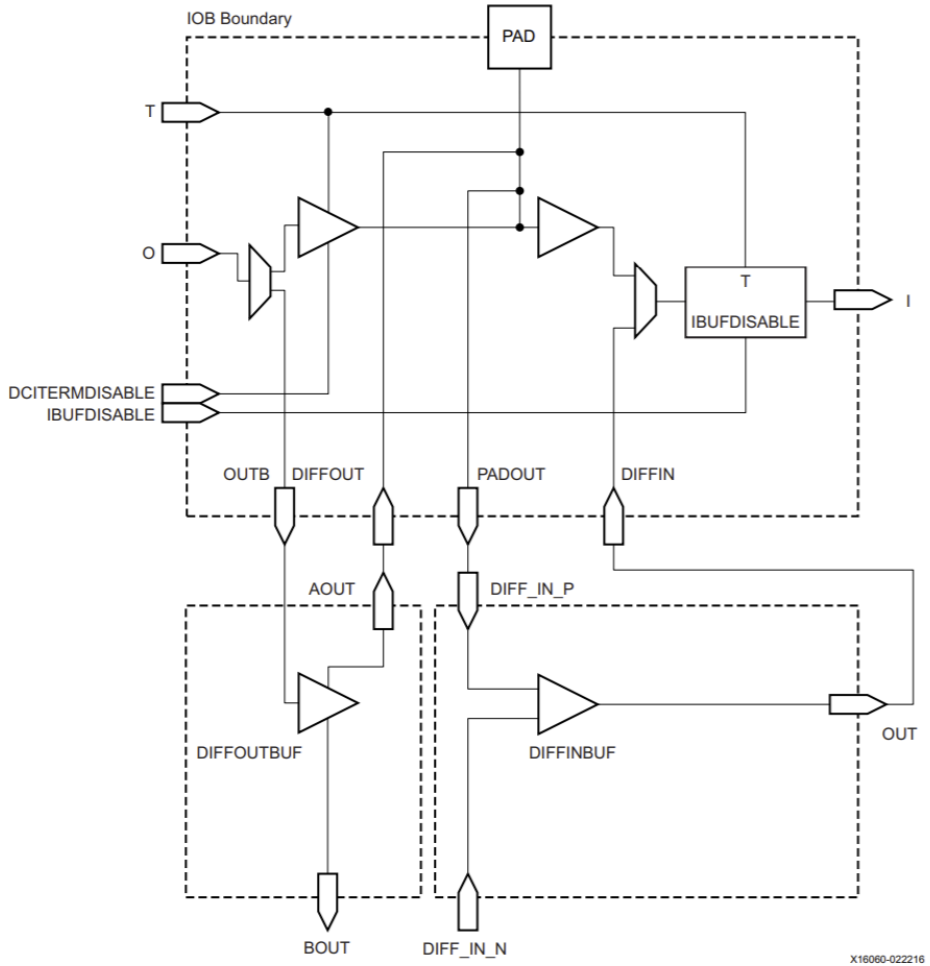
DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

FPGA configurable PIN

Ultrascale + I/O







DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

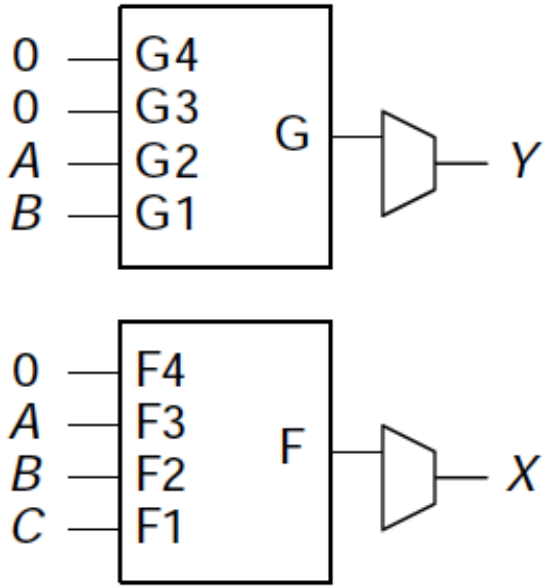
LUT based logic implementation

$$X = \overline{A}\,\overline{B}C + AB\overline{C} \text{ and } Y = A\overline{B}$$

	(A)	(B)	(C)	(X)
F4	F3	F2	F1	F
X	0	0	0	0
X	0	0	1	1
X	0	1	0	0
X	0	1	1	0
X	1	0	0	0
X	1	0	1	0
X	1	1	0	1
X	1	1	1	0

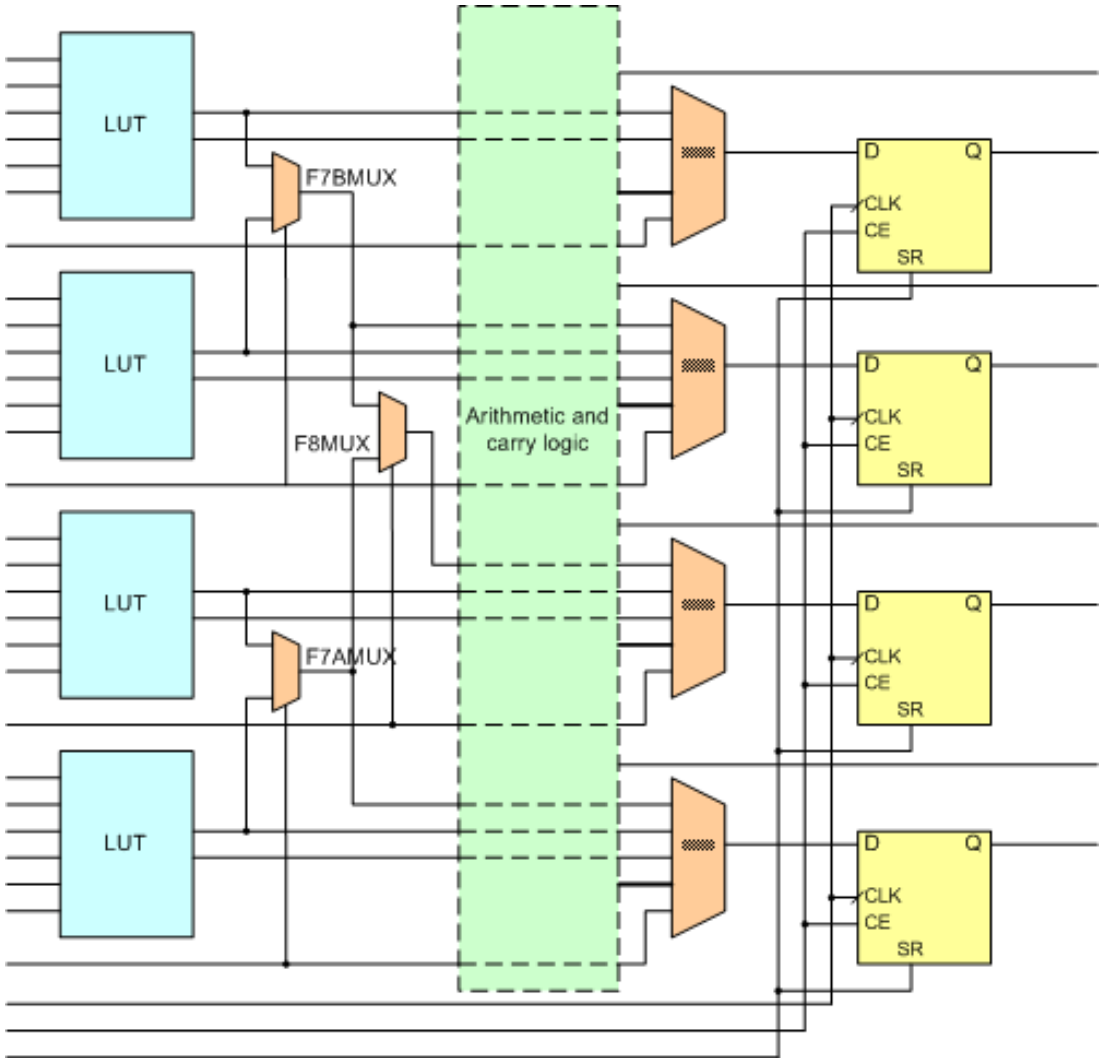
		(A)	(B)	(Y)
G4	G3	G2	G1	G
X	X	0	0	0
X	X	0	1	0
X	X	1	0	1
X	X	1	1	0

Xilinx Spartan CLB



DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

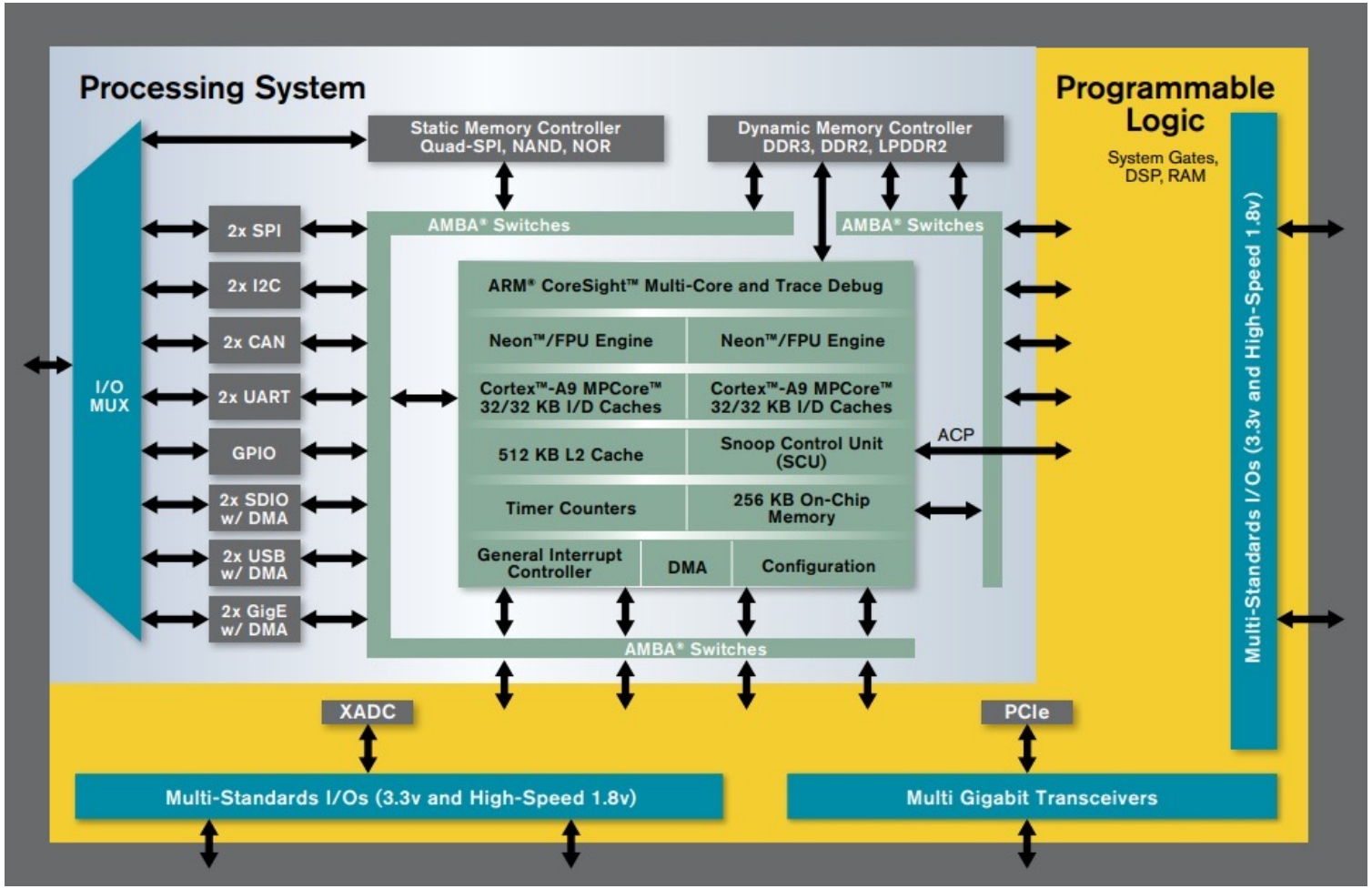
LUT structure



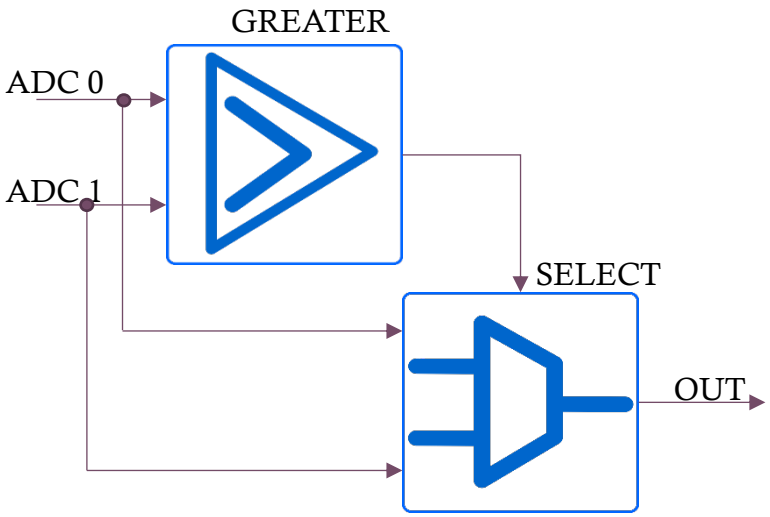
Xilinx Virtex-5 slice

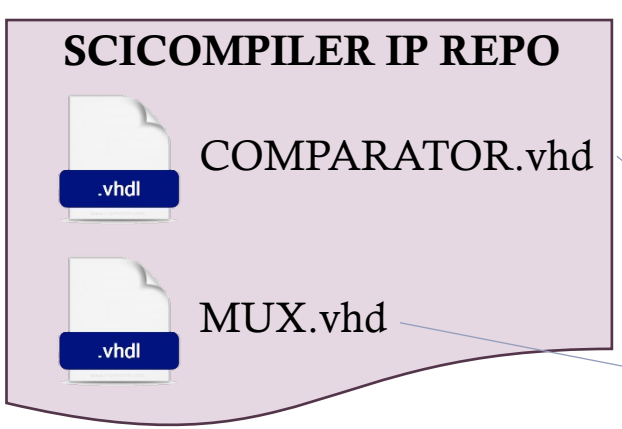
DIGITAL COMPONENTS FOR SIGNAL PROCESSING: FPGA

System on chip (SoC): Zynq



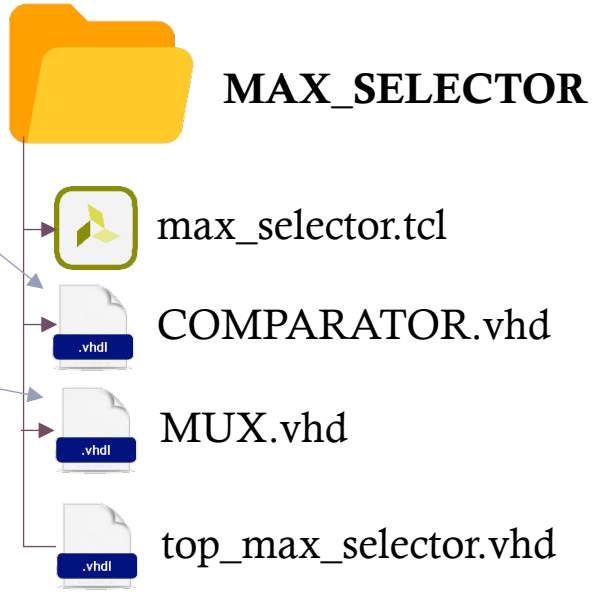
WHAT SCI-COMPILER DOES INSIDE



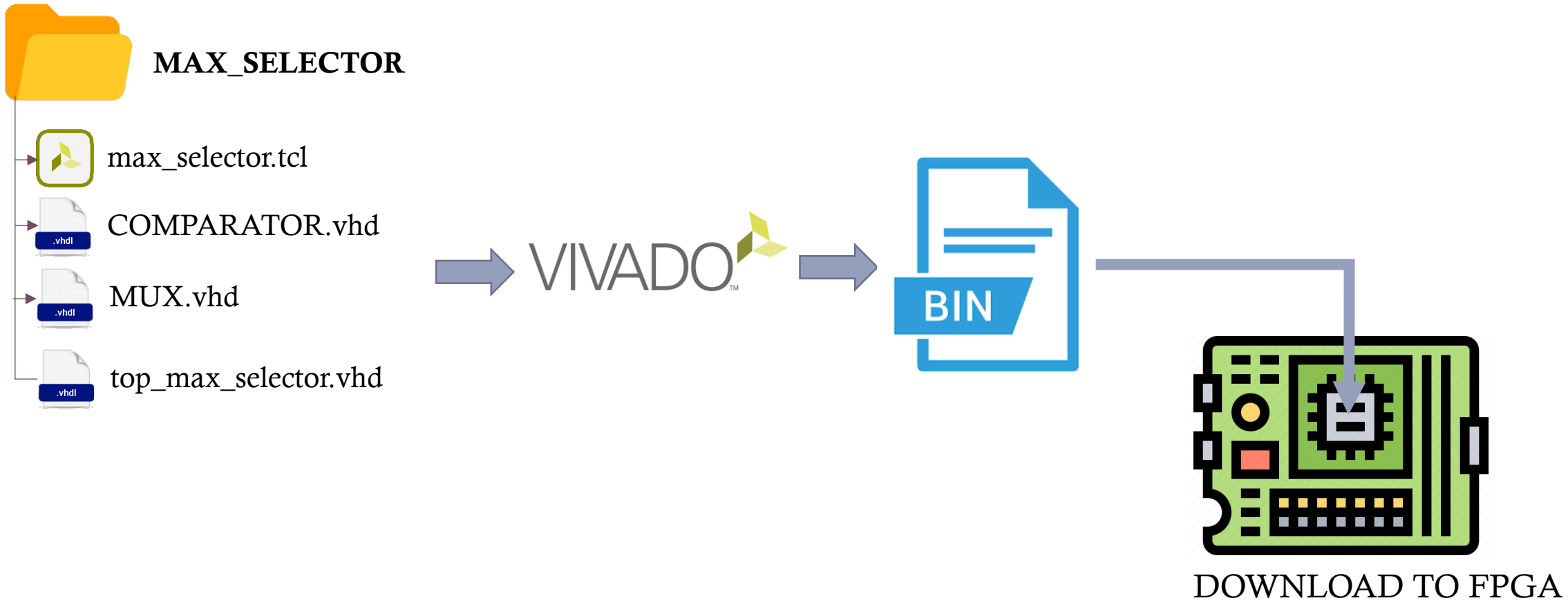


SCICOMPILER BUILDER

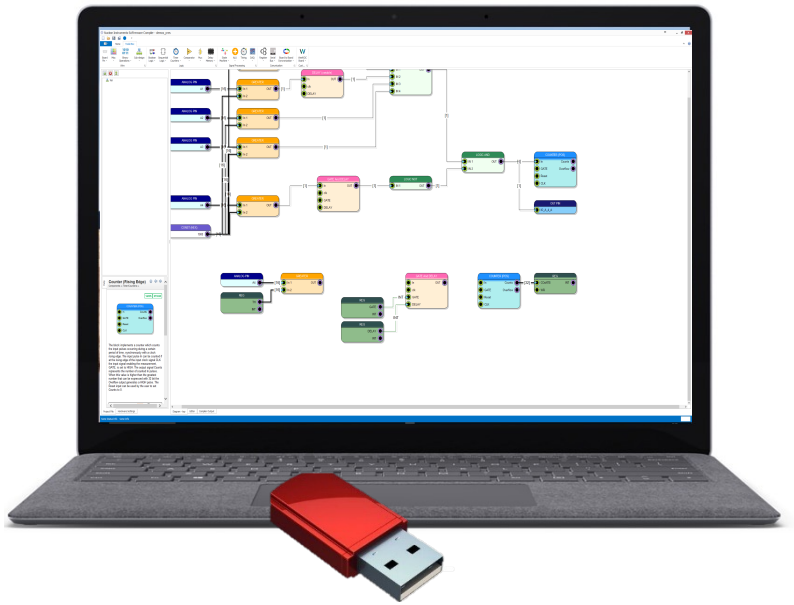
```
17  begin
18
19      U1: comparator
20      port map(
21          IN1 => ADC_0,
22          IN2 => ADC_1,
23          CLK => CLK,
24          COMP_OUT => U1_OUT
25      );
26
27      U2: MUX
28      port map(
29          IN1 => ADC_0,
30          IN2 => ADC_1,
31          SEL => U1_OUT,
32          OUT0 => U2_OUT
33      );
```



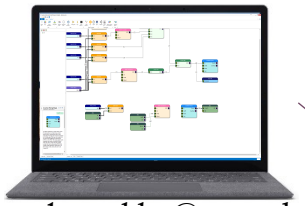
WHAT SCI-COMPILER DOES INSIDE



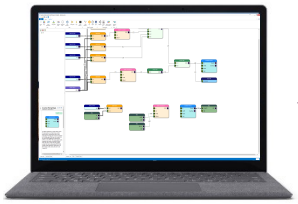
License Model



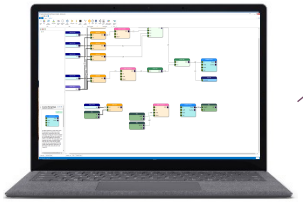
IN DEVELOPME NT



andrea.abba@cern.ch



joseph.criss@cern.ch



jasmin.white@cern.ch

CAEN License Server

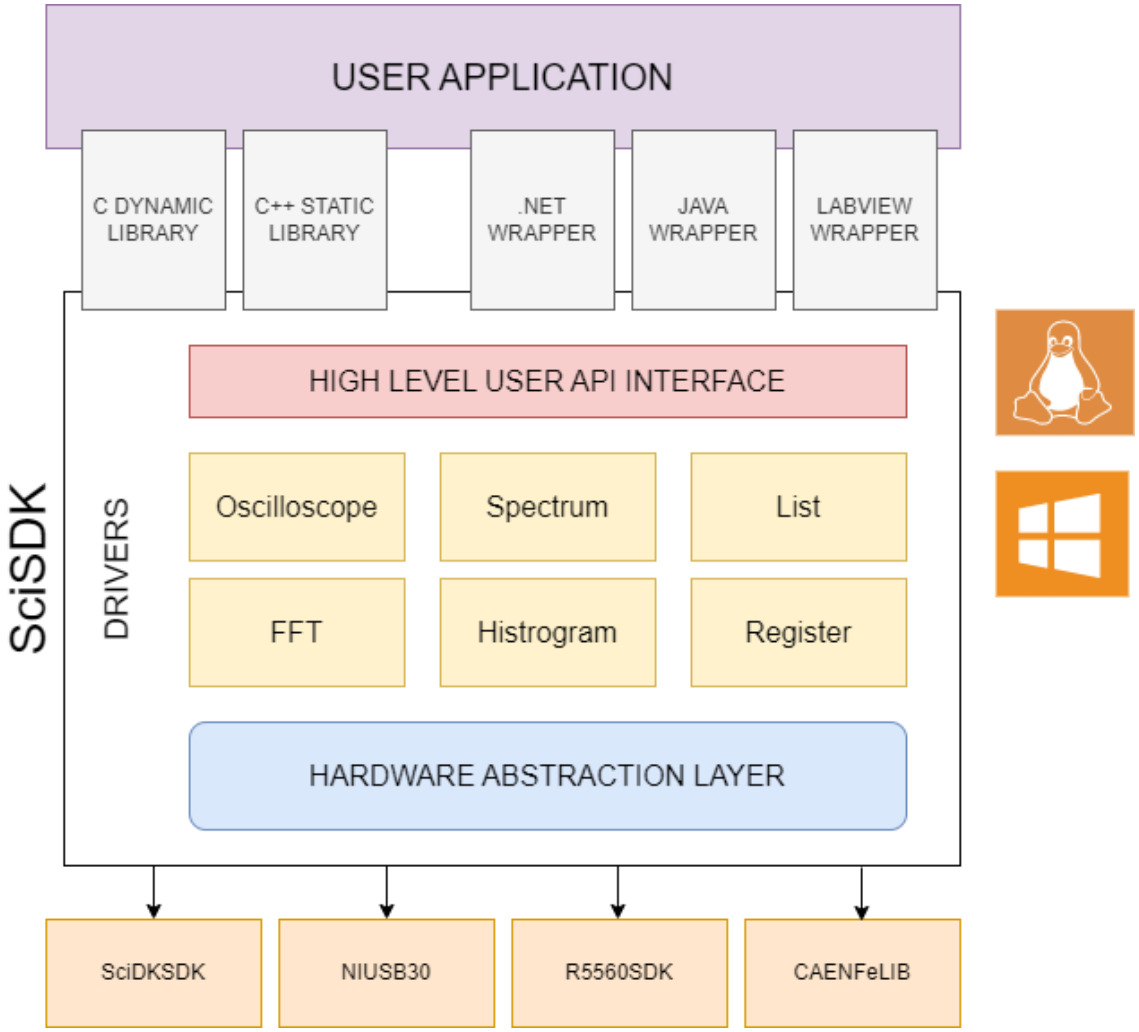
Stand Alone License:

- Works offline
- Perpetual software license
- Update and support for 12 months included, renewable
- Dongle key included: one dongle = one seat

Enterprise/Site License:

- Pool of license for the organization (minimum 10 licenses)
- Perpetual software license
- Update and support for 12 months included, renewable
- Online checkout of the license (admin web interface):
  - Single License can be static assigned to one user
  - Dynamic assignment of the license (FCFS)

SCISDK


USER APPLICATION
C DYNAMIC LIBRARY
C++ STATIC LIBRARY
.NET WRAPPER
JAVA WRAPPER
LABVIEW WRAPPER
SciSDK
HIGH LEVEL USER API INTERFACE
DRIVERS
Oscilloscope
Spectrum
List
FFT
Histrogram
Register
HARDWARE ABSTRACTION LAYER
SciDKSDK
NIUSB30
R5560SDK
CAENFeLIB

SCISDK

SciSDK_DLL.dll
|---> SCIDK_Lib.dll (optional, library required for DT1260/SCIDK)
|---> CAEN_FELib.dll (optional, library required for V/DT274X/FELib)
|---> CAEN_Dig2Lib.dl (optional, library required for V/DT274X/FELib)
|---> R5560_SDKLib.dll (optional, library required for R5560/R5560SE/DT5560)
|---> NiUSB30.dll (optional, library required for DT5550/DT5550W)
|---> libzmq-v140-mt-4_3_4.dll (optional, service library required for R5560/R5560SE/DT5560)
|---> libsodium.dll (optional, service library required for R5560/R5560SE/DT5560)

https://github.com/NuclearInstruments/SCISDK

SCISDK IN PYTHON

pip install scisdk

```python
from scisdk.scisdk import SciSDK

# initialize scisdk library
sdk = SciSDK()

#DT1260
res = sdk.AddNewDevice("usb:10500","dt1260", "./library/RegisterFile.json","board0")

if res != 0:
    print ("Script exit due to connetion error")
    exit()
```

SCISDK IN C

Install SciSDK from executable or compile linux version with automake

```
#include <SciSDK_DLL.h>
#include <NIErrorCode.h>

void* _sdk = SCISDK_InitLib();
char* res;
int ret = 0;

ret = SCISDK_AddNewDevice("usb:10500", "dt1260", "ListDT1260.json", "board0", _sdk);
if (ret == NI_OK) {
    cout << "Connected to the device." << endl;
}
else {
    SCISDK_s_error(ret, &res, _sdk);
    cout << res << endl;
    exit(-1);
}
```

SCISDK IN PYTHON: SET/GET REGISTERS

```
regA=100
regB=151
regC = 0
sdk.SetRegister("board0:/Registers/A", regA)
sdk.SetRegister("board0:/Registers/B", regB)
err, regC = sdk.GetRegister("board0:/Registers/C")
print("Register C = (A+B) value is ", regC)
```

SCISDK IN PYTHON: READ OSCILLOSCOPE

```python
# set oscilloscope parameters on the board
res = sdk.SetParameterString("board0:/MMCComponents/Oscilloscope_0.data_processing","decode")
res = sdk.SetParameterInteger("board0:/MMCComponents/Oscilloscope_0.trigger_level",1500)
res = sdk.SetParameterString("board0:/MMCComponents/Oscilloscope_0.trigger_mode","self")
res = sdk.SetParameterInteger("board0:/MMCComponents/Oscilloscope_0.trigger_channel", 0)
res = sdk.SetParameterInteger("board0:/MMCComponents/Oscilloscope_0.pretrigger", 150)
res = sdk.SetParameterInteger("board0:/MMCComponents/Oscilloscope_0.decimator", 0)
res = sdk.SetParameterString("board0:/MMCComponents/Oscilloscope_0.acq_mode", "blocking")

# allocate buffer (sciSDK detects automatically buffer type)
res, buf = sdk.AllocateBuffer("board0:/MMCComponents/Oscilloscope_0")
if res == 0:
    res = sdk.ExecuteCommand("board0:/MMCComponents/Oscilloscope_0.reset_read_valid_flag", "")
    # read data
    res, buf = sdk.ReadData("board0:/MMCComponents/Oscilloscope_0", buf)
```

SCISDK IN PYTHON: READ OSCILLOSCOPE

```
if res == 0:
        # store analog data inside a file
        analog_str = ""
        for i in range(0,buf.info.samples_analog):
            analog_str = analog_str + str(buf.analog[i]) + "\n"

        # store digital data inside a file
        digital_str = ""
        # read digital data from channel 1
        for i in range(0,buf.info.samples_digital):
            digital_str = digital_str + str(buf.digital[i]) + "\n«

        digital_str = ""
        # read digital data from channel 2
        for i in range(buf.info.samples_digital,buf.info.samples_digital*2):
            digital_str = digital_str + str(buf.digital[i]) + "\n"
```

SCISDK IN PYTHON: FREE MEMORY AND CLOSE CONNECTION

```
res = sdk.FreeBuffer("board0:/MMCComponents/Oscilloscope_0", buf)
sdk.DetachDevice("board0")
res = sdk.FreeLib()
```

SCISDK IN PYTHON: PLOT SPECTRUM

```python
# set board parameters
sdk.SetParameterString("board0:/MMCComponents/Spectrum_0.rebin", "0")
sdk.SetParameterString("board0:/MMCComponents/Spectrum_0.limitmode", "freerun")
sdk.SetParameterString("board0:/MMCComponents/Spectrum_0.limit", "100")

# execute command reset
sdk.ExecuteCommand("board0:/MMCComponents/Spectrum_0.reset", "")

# execute command start
sdk.ExecuteCommand("board0:/MMCComponents/Spectrum_0.start", "")


# allocate buffer
res, buf = sdk.AllocateBuffer("board0:/MMCComponents/Spectrum_0")
```

SCISDK IN PYTHON: PLOT SPECTRUM

```
def updateGraph(i, buffer): # function that provides to plot new data on graph
    res, buffer = sdk.ReadData("board0:/MMCComponents/Spectrum_0", buffer)# read data from
board
    if res == 0:
        xar = []
        yar = []
        for index in range(buffer.info.valid_bins):
            xar.append(index)
            yar.append(buffer.data[index])
        ax1.clear()
        ax1.plot(xar,yar)

# update graph every 50ms
ani = animation.FuncAnimation(fig, updateGraph, fargs=[buf,],interval=100)
# updateGraph(None, buf, decimator)
plt.show()
```

SCISDK IN PYTHON: READ DATA FROM LIST

```
res = sdk.SetParameterString("board0:/MMCComponents/List_0.thread", "false")
res = sdk.SetParameterInteger("board0:/MMCComponents/List_0.timeout", 500)
res = sdk.SetParameterString("board0:/MMCComponents/List_0.acq_mode", "blocking")

# allocate buffer raw, size 1024
res, buf = sdk.AllocateBuffer("board0:/MMCComponents/List_0", 1024)

res = sdk.ExecuteCommand("board0:/MMCComponents/List_0.stop", "")

res = sdk.ExecuteCommand("board0:/MMCComponents/List_0.start", "")

while True:
    res, buf = sdk.ReadData("board0:/MMCComponents/List_0", buf)
    if res == 0:
        for i in range(0, int(buf.info.valid_samples/8)):
            print(unpack('<hhL', buf.data[i*8:(i+1)*8]))
```

SCISDK IN PYTHON: READ DATA FROM CUSTOM PACKET

```
res = sdk.SetParameterString("board0:/MMCComponents/CP_0.thread", "false")
res = sdk.SetParameterInteger("board0:/MMCComponents/CP_0.timeout", 500)
res = sdk.SetParameterString("board0:/MMCComponents/CP_0.acq_mode", "non-blocking")
res = sdk.SetParameterString("board0:/MMCComponents/CP_0.check_align_word", "true")
res = sdk.SetParameterString("board0:/MMCComponents/CP_0.data_processing", "decode")

# allocate buffer raw, size 1024
res, buf = sdk.AllocateBuffer("board0:/MMCComponents/CP_0", 100)

res = sdk.ExecuteCommand("board0:/MMCComponents/CP_0.start", "")
```

SCISDK IN PYTHON: READ DATA FROM CUSTOM PACKET

```
while True:
    res, buf = sdk.ReadData("board0:/MMCComponents/CP_0", buf)
    if res == 0:
        for i in range(0, int(buf.info.valid_data)):
            for n in range(0, buf.data[i].n):
                print(buf.data[i].row[n])
            print("header:      ", hex(buf.data[i].row[0]))
            print("timestamp:   ",buf.data[i].row[1] + (buf.data[i].row[2] << 32) )
            print("counter 0:   ",buf.data[i].row[3])
            print("counter 1:   ",buf.data[i].row[4])
            print("counter 2:   ",buf.data[i].row[5])
            print("counter 3:   ",buf.data[i].row[6])
            print("-----------------")
```

TIME FOR A COFFE!

INTERNATIONAL SUMMER SCHOOL 2024

DIGITAL SIGNAL PROCESSING: THE ADC

Digitize the analog signal

ANALOG DEVICES

16-Bit, 80/100 MSPS ADC

AD9446

FEATURES

- 100 MSPS guaranteed sampling rate (AD9446-100)
- 83.6 dBFS SNR with 30 MHz input (3.8 V p-p input, 80 MSPS)
- 82.6 dBFS SNR with 30 MHz input (3.2 V p-p input, 80 MSPS)
- 89 dBc SFDR with 30 MHz input (3.2 V p-p input, 80 MSPS)
- 95 dBFS 2-tone SFDR with 9.8 MHz and 10.8 MHz (100 MSPS)
- 60 fsec rms jitter
- Excellent linearity
  - DNL = ±0.4 LSB typical
  - INL = ±3.0 LSB typical
- 2.0 V p-p to 4.0 V p-p differential full-scale input
- Buffered analog inputs
- LVDS outputs (ANSI-644 compatible) or CMOS outputs
- Data format select (offset binary or twos complement)
- Output clock available
- 3.3 V and 5 V supply operation

FUNCTIONAL BLOCK DIAGRAM

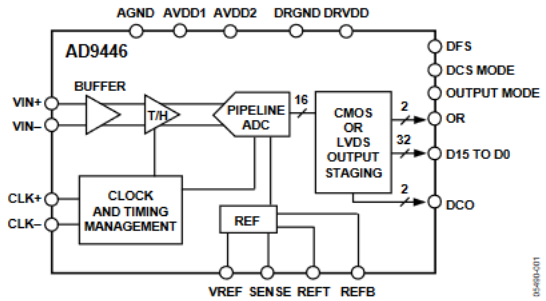


Figure 1.

TEXAS INSTRUMENTS

ADS5400-SP

SLAS669E – SEPTEMBER 2010 – REVISED MAY 2020

12-Bit, 1-GSPS Analog-to-Digital Converter

1 Features

- 1-GSPS sample rate
- 12-Bit resolution
- 2.1 GHz input bandwidth
- SFDR = 65 dBc at 1.2 GHz
- SNR = 57 dBFS at 1.2 GHZ
- 7 Clock cycle latency
- Interleave friendly: internal adjustments for gain, phase and offset
- 1.5 - 2 V_{PP} Differential input voltage, programmable
- LVDS-compatible outputs, 1 or 2 bus options
- Total power dissipation: 2.2 W
- On-chip analog buffer
- 100-pin ceramic nonconductive tie-bar package
- Military temperature range (–55°C to 125°C T_{case})

3 Description

The ADS5400 is a 12-bit, 1-GSPS analog-to-digital converter (ADC) that operates from both a 5-V supply and 3.3-V supply, while providing LVDS-compatible digital outputs. The analog input buffer isolates the internal switching of the track and hold from disturbing the signal source. The simple 3-stage pipeline provides extremely low latency for time critical applications. Designed for the conversion of signals up to 2 GHz of input frequency at 1 GSPS, the ADS5400 has outstanding low noise performance and spurious-free dynamic range over a large input frequency range.

The ADS5400 is available in a 100-Pin Ceramic Nonconductive Tie-Bar Package. The combination of the ceramic package and moderate power consumption of the ADS5400 allows for operation without an external heatsink. The ADS5400 is built on Texas Instrument's complementary bipolar process (BiCom3) and is specified over the full military temperature range (–55°C to 125°C T_{case}).

DIGITAL SIGNAL PROCESSING: THE ADC

Resolution

Simplistic assumption:

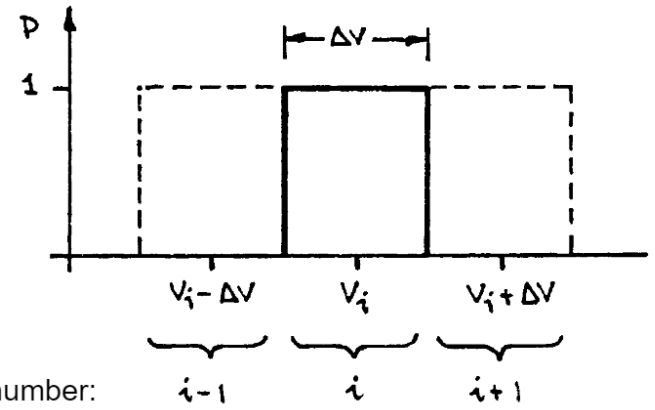
Resolution is defined by the number of output bits, e.g.

13 bits $\rightarrow$ $\frac{\Delta V}{V} = \frac{1}{8192} = 1.2 \cdot 10^{-4}$

True Measure: Channel Profile

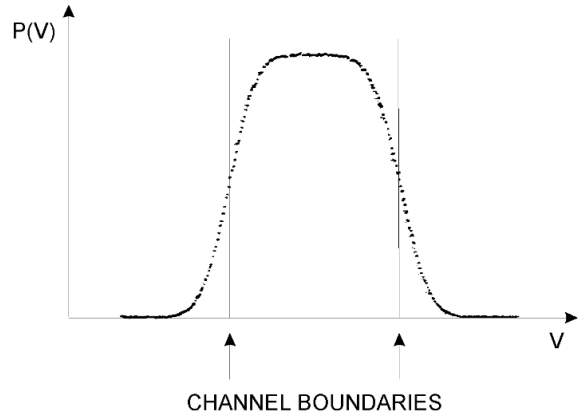
Plot probability vs. pulse amplitude that a pulse height corresponding to a specific output bin is actually converted to that address.

Ideal ADC:

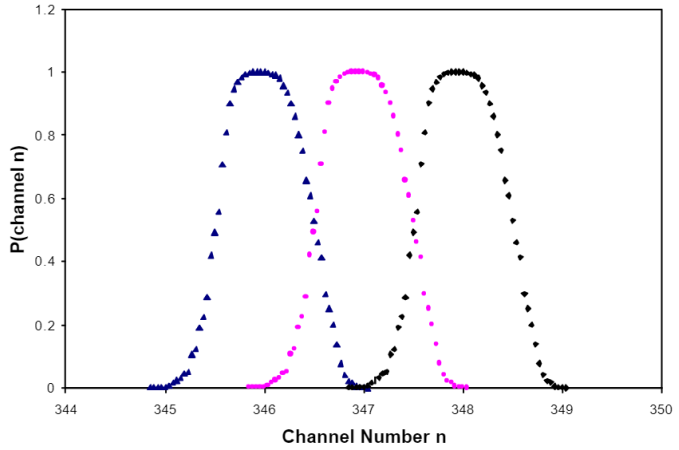


output bin number:

Measured channel profile (13 bit ADC)

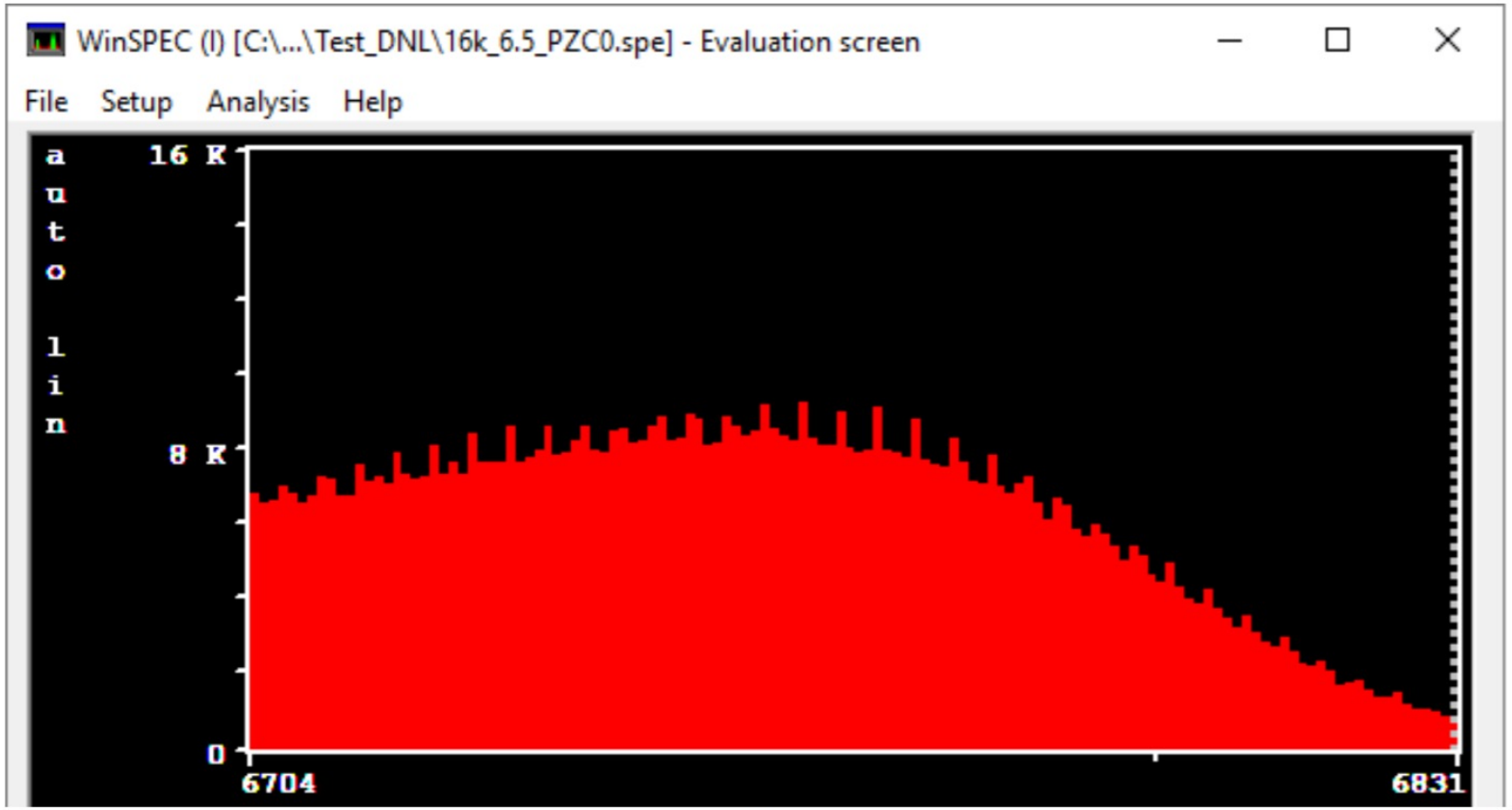


The profiles of adjacent channels overlap



TRANSITION TO DIGITAL: THE ADC

Non-Linearity effect on spectrum

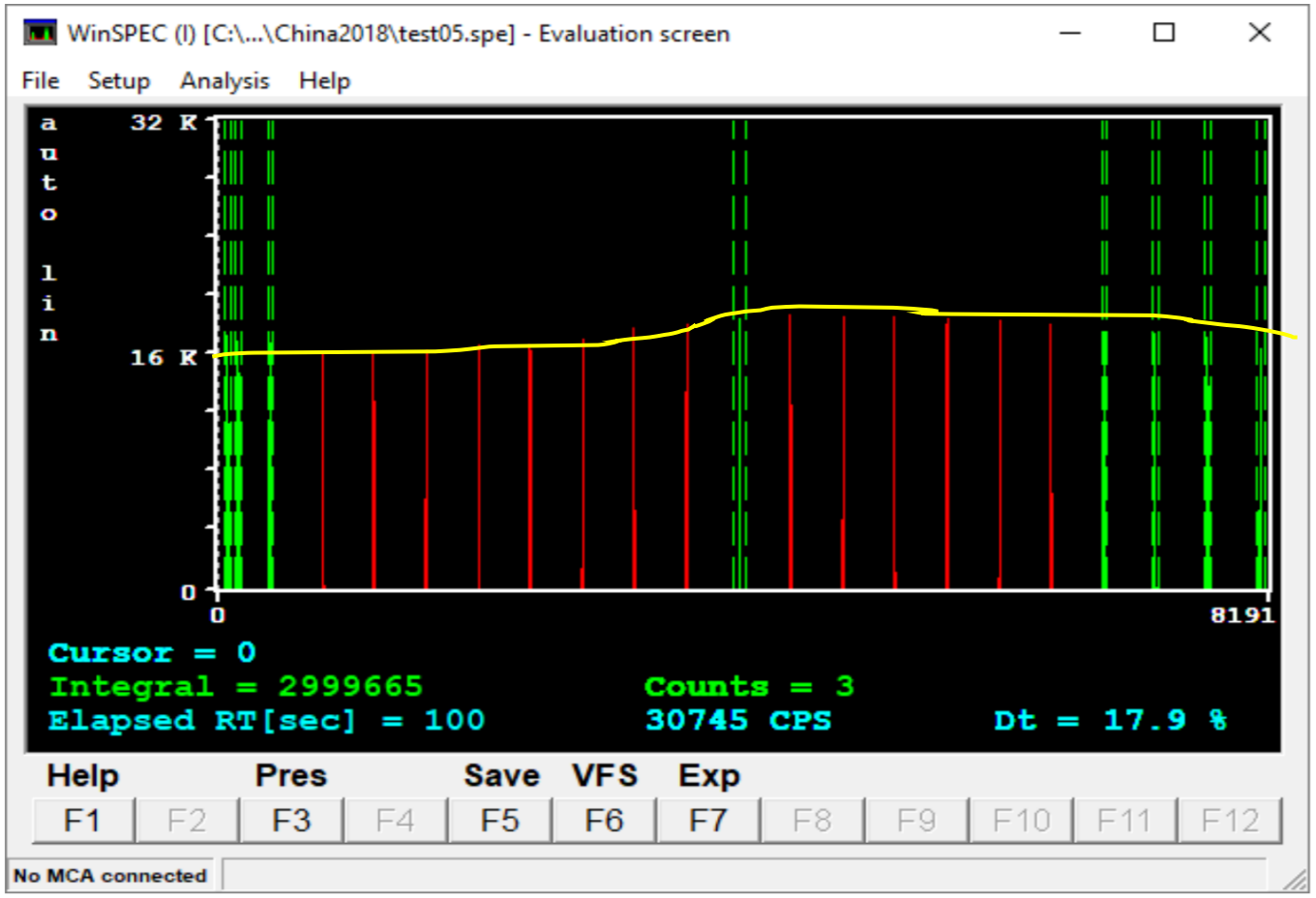


Different widths --> different probability --> peaks

Example for a spectrum measured with bad differential nonlinearity, up to 15% here

TRANSITION TO DIGITAL: THE ADC

Non-Linearity effect on spectrum

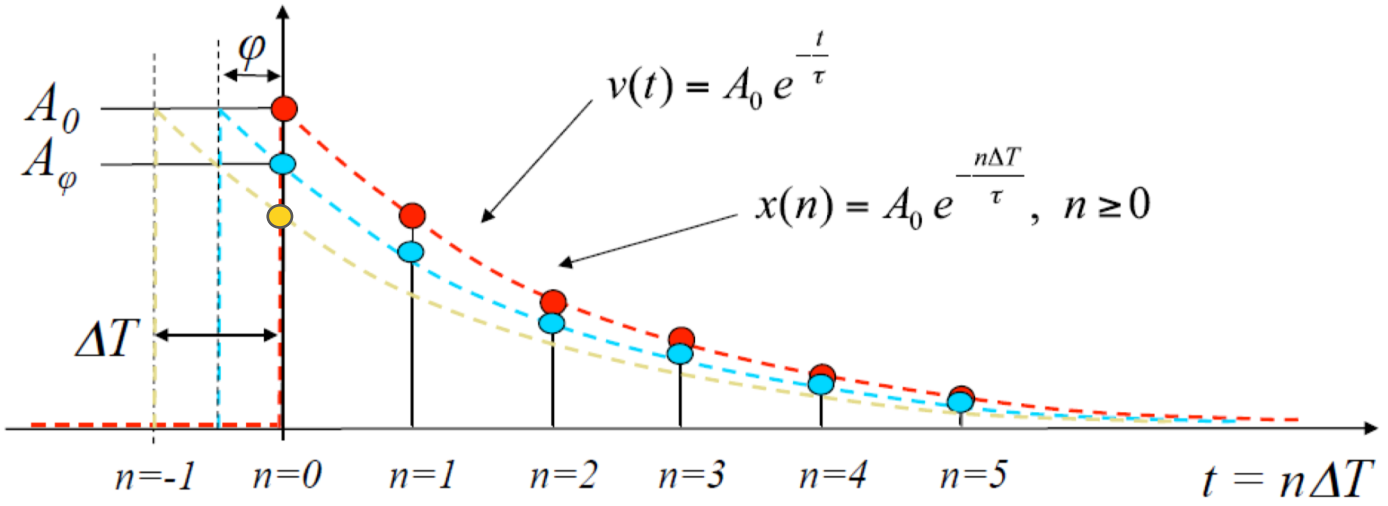


Deviation of the peak at different energies due to INL

“Low-frequency” modulation

Sampling Clock Phase Error

sampling is not synchronized with detector pulse arrival time!



$$A_\varphi(\varphi) = A_0 e^{-\frac{\varphi}{\tau}} \qquad 0 < \varphi \leq \Delta T \qquad e(\varphi) = A_0 - A_\varphi(\varphi) = A_0\left(1 - e^{-\frac{\varphi}{\tau}}\right)$$

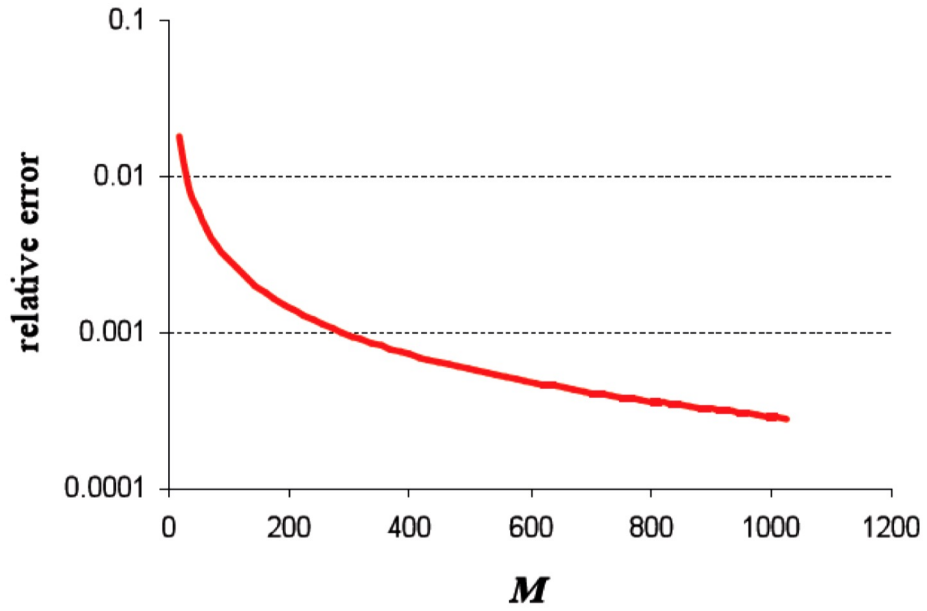
$$\text{if } \varphi << \tau \;\rightarrow\; 1 - e^{-\frac{\varphi}{\tau}} \approx \frac{\varphi}{\tau}$$

$$\boxed{e(\varphi) \approx A_0 \frac{\varphi}{\tau}}$$

Relative Phase Error – Exponential Pulse

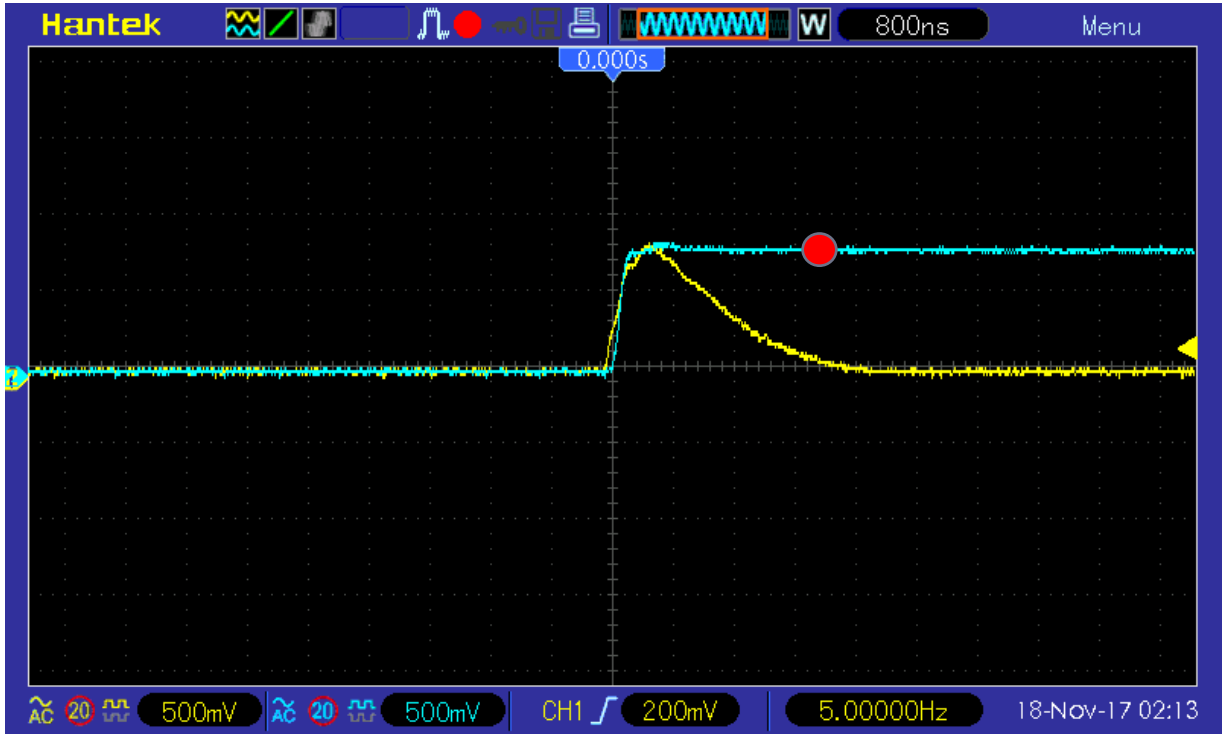
Phase error depends on the exponential signal amplitude!

relative error $\boxed{\varepsilon_\varphi \approx 0.29 \frac{1}{M}}$ $\tau = M \cdot \Delta T$



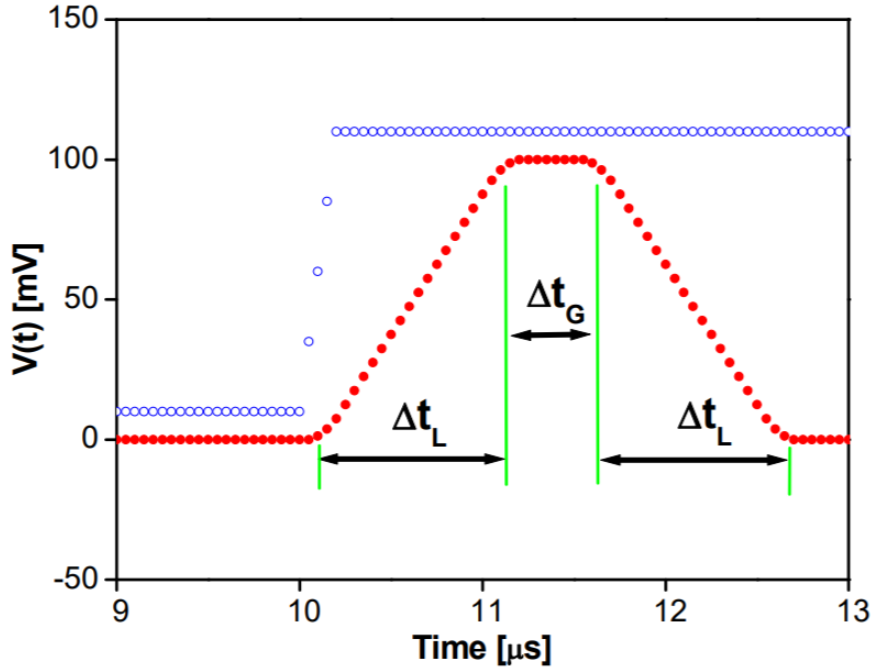
TRANSITION TO DIGITAL: REDUCE ADC NON IDEALITIES IMPACT

Classic MCA



ADC uses a single sample point

Digital MCA (PHA)



ADC uses hundreds of point

DIGITAL SIGNAL PROCESSING

1. **Design**: standard and tested components (ADC+FPGA), analog circuits has many components (tolerances), ageing of the analog components

2. **Resolution**: ADC resolution is increased by the averaging of multiple points

3. **Differential Non-Linearity**: almost neglectable because the signal is composed by several point sampled at random time moment. The instruments measure the area and not a single point.

4. **Conversion Time**: ADC samples continuously. Digital filter can be implemented to operate in real-time

5. **Stability**: digital circuit does not change operation parameters with ageing.