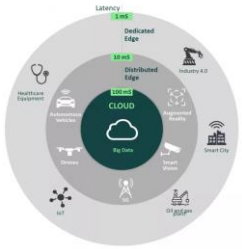
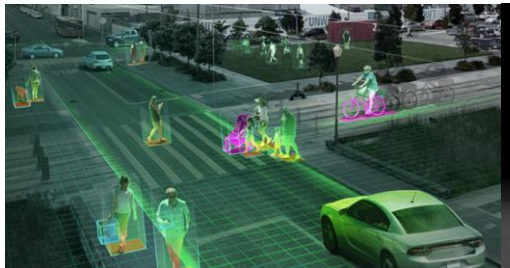
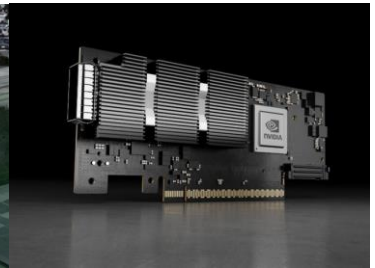
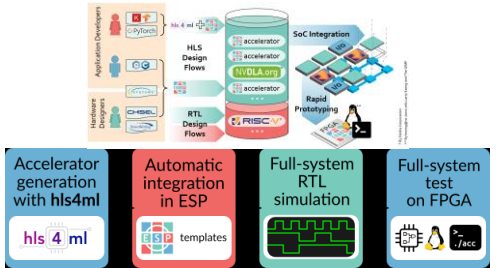
L’intelligence Artificielle dans les systèmes embarqués
– Principe & Méthodes –
– Applications avec FPGA Xilinx –

- ➔Principes et Objectifs
- ➔Les Réseaux de Neurones
- ➔Méthodologies
- ➔Modélisation Système pour l’apprentissage par renforcement









Définition

« *L'intelligence artificielle (IA) est un processus d'imitation de l'intelligence humaine qui repose sur la création et l'application d'algorithmes exécutés dans un environnement informatique dynamique.*

Son but est de permettre à des ordinateurs de penser et d'agir comme des êtres humains. »

Pour y parvenir, trois composants sont nécessaires :

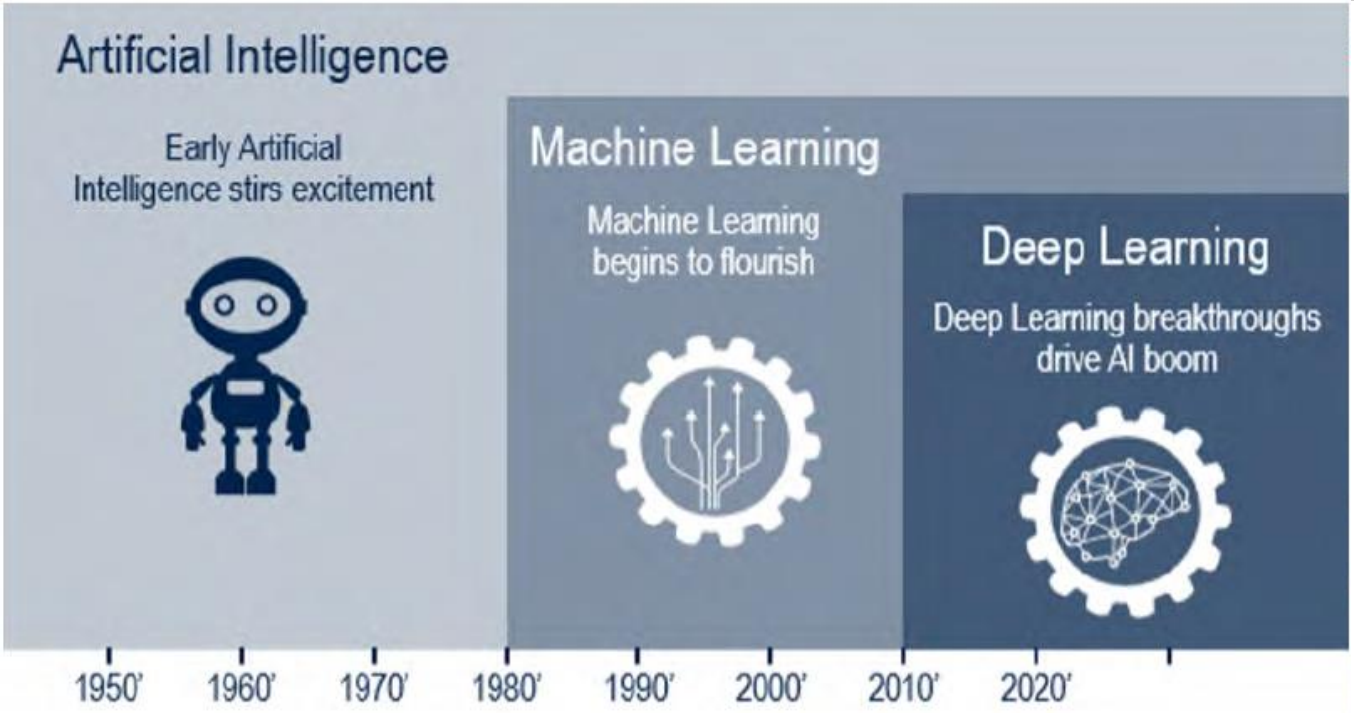
- ***Des systèmes informatiques***
- ***Des données avec des systèmes de gestion***
- ***Des algorithmes d'IA avancés (code)***

Pour se rapprocher le plus possible du comportement humain, l'intelligence artificielle a besoin d'une quantité de données et d'une capacité de traitement élevées. »

Système embarqué

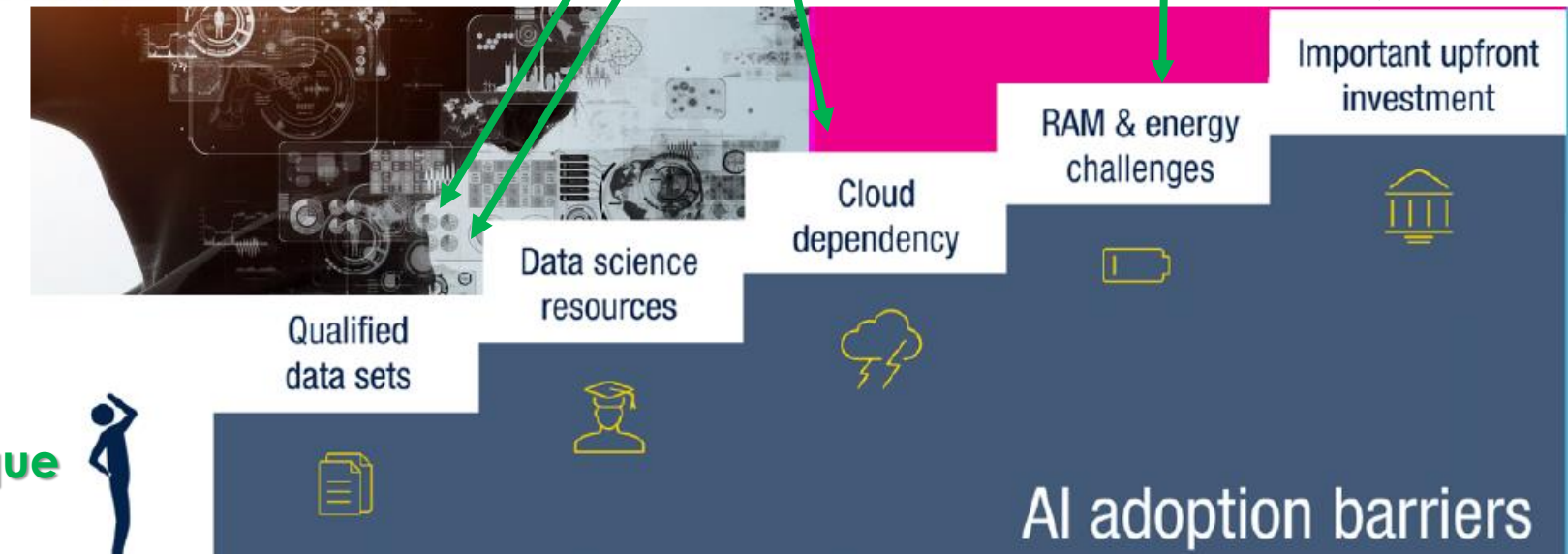
(https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/)

Historique



➔ **Evolution et progression de l'IA**

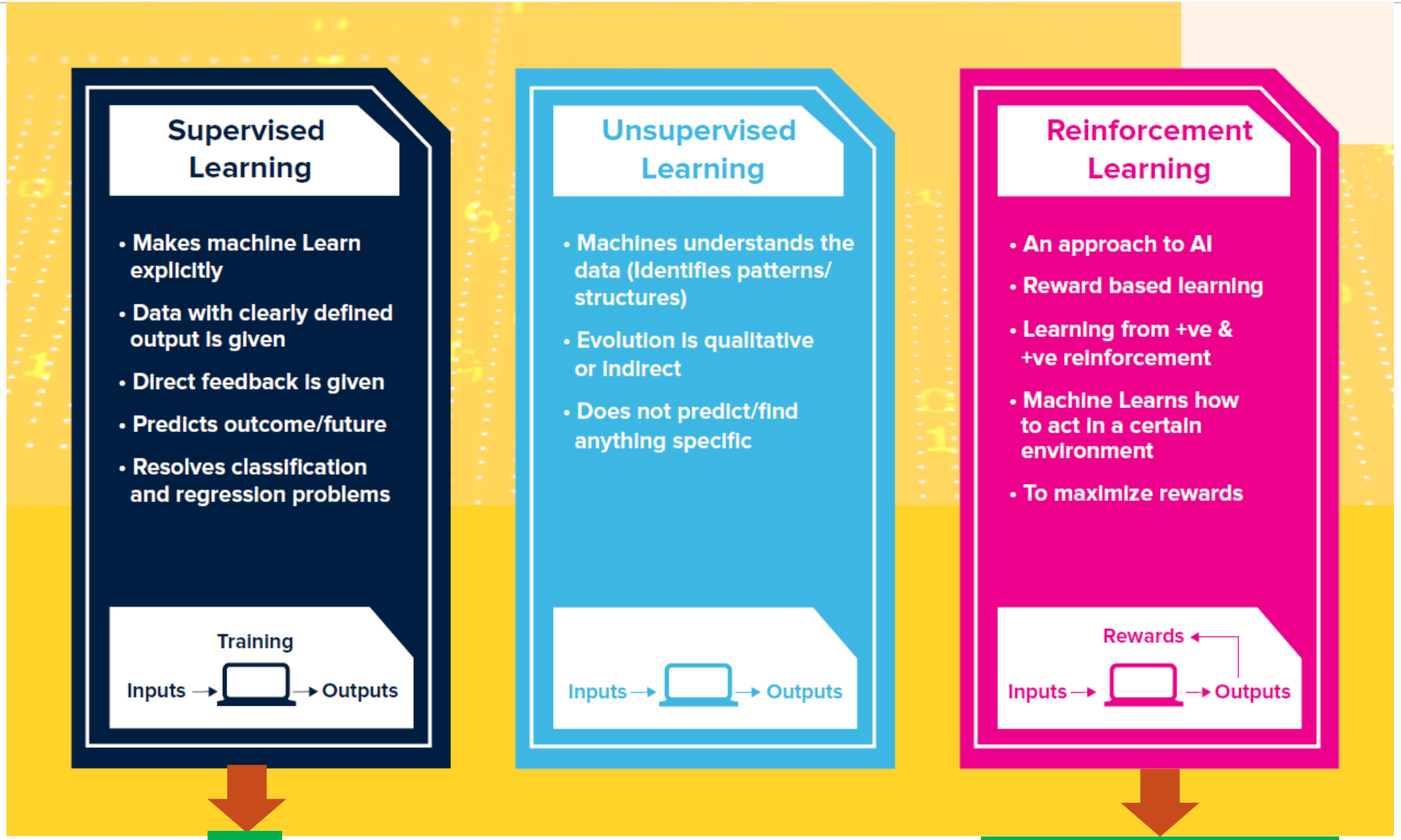
THINK:
Des données sélectionnées
Une formation
De la puissance de calcul
Sobriété (matériels)
Investissement de l'IN2P3



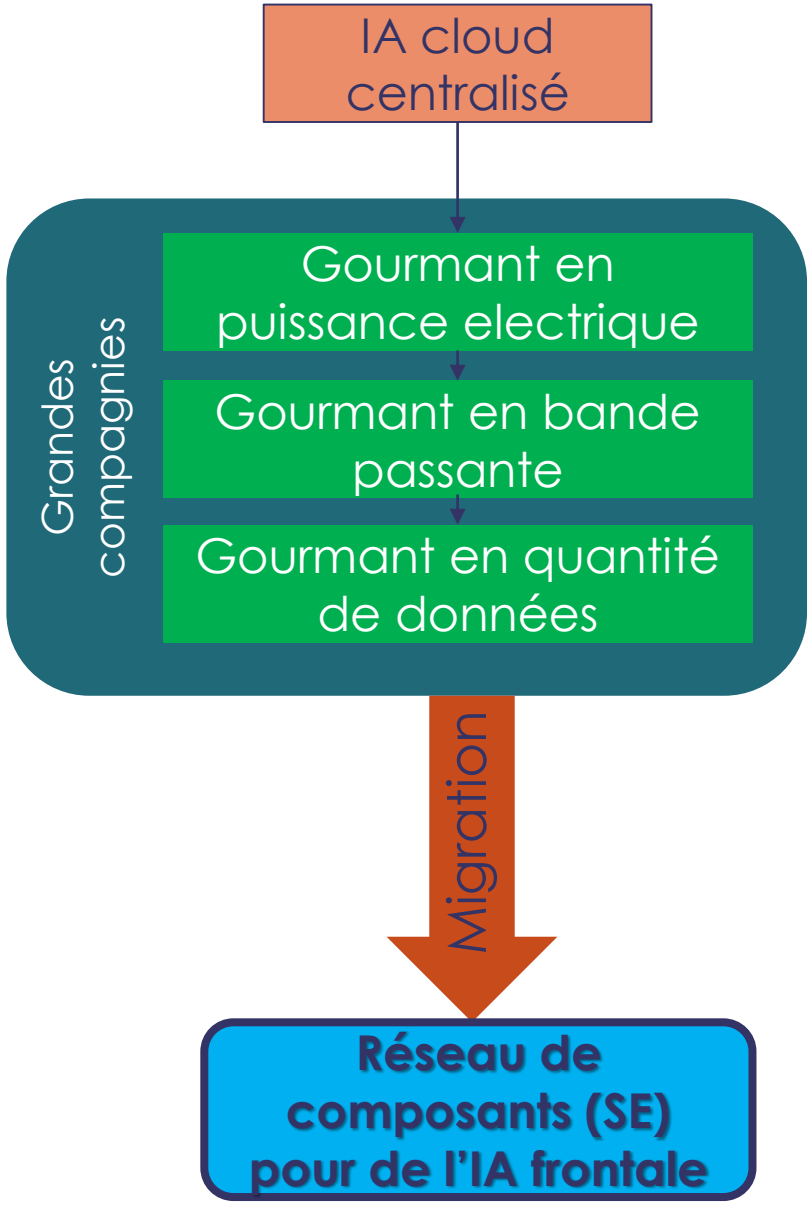
➔ **Les barrières pour adopter l'IA dans les produits pour les sociétés.**
-investissement en RH, matériels, données
-Grandes compagnies

➔ **ML dans SE : solution économique**

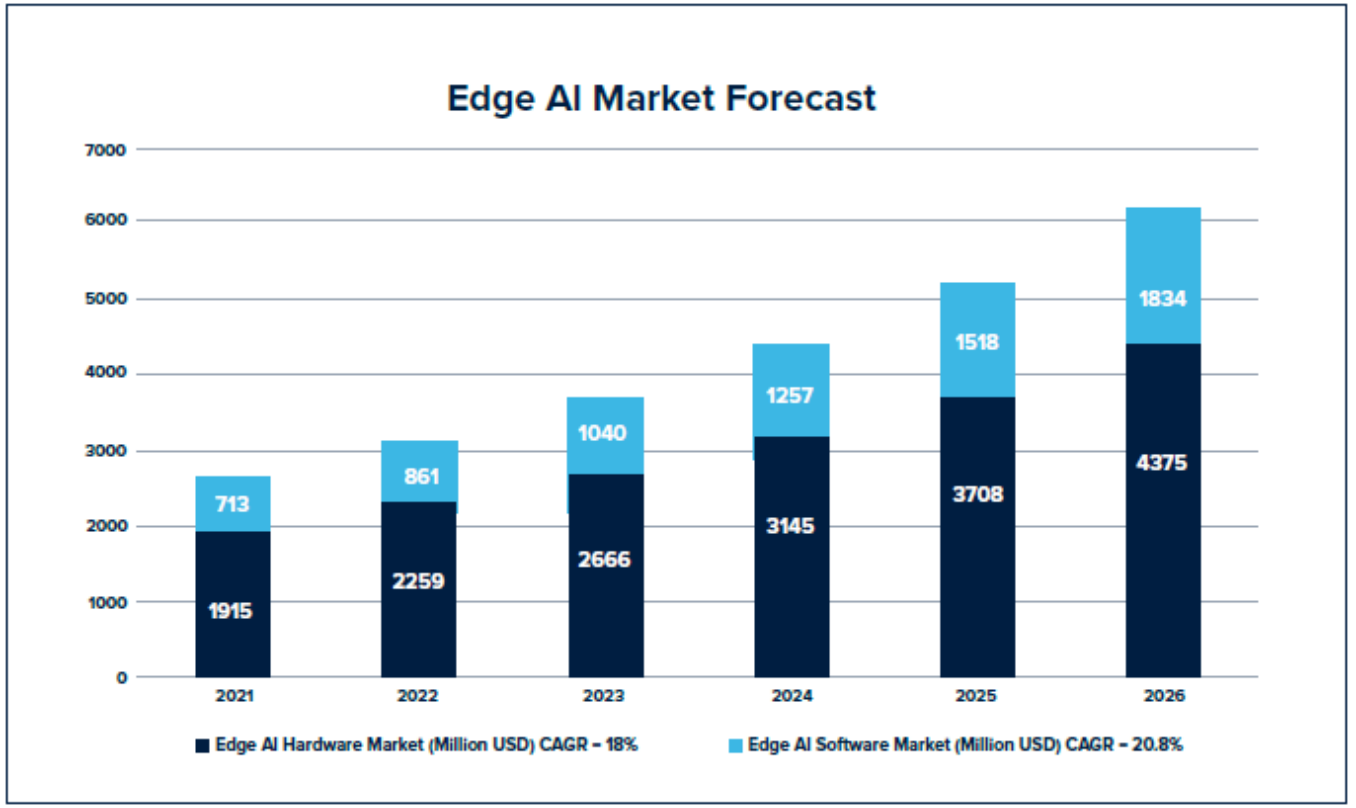
Les trois catégories d'apprentissage profond



Machine apprenante locale ➔ Responsive AI on the Edge



STMicroelectronics 2022



Les techniques de l'IA

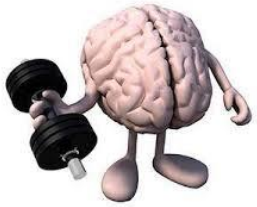
- Traitement des images et des vidéos (computer Vision)
  - Identification d'objets
  - Capture d'image
  - Etc…
- Traitement du langage naturel
  - Classification de texte
  - Lecture automatique
  - Reconnaissance de mot
  - Traduction
- Structuration des données
  - Classification des informations
  - Proposition (recommandation)
  - Maintenance
  - Détection de fraude

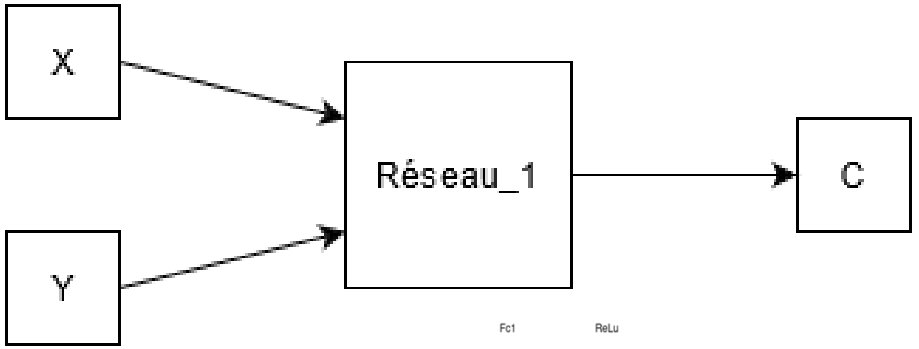
Les techniques de l'IA

- Séries temporelles
  - Prévision des données dans une serie temporelle de données (météo)
  - Classification des données en temps réel
- Données Audio
  - Reconnaissance de la parole
  - Reconnaissance des personnes suivant leur conversation
- Generateur d'information en apprentissage profond
  - Génération d'images
  - Génération d'information (texte, nombre …)
- Apprentissage par renforcement
  - Robotique
  - Simulation suivant meilleur choix
- Traitement des données avec graphes (chaînes)
  - Classification de nœuds
  - Arbre de decision
  - Algorithme genetique

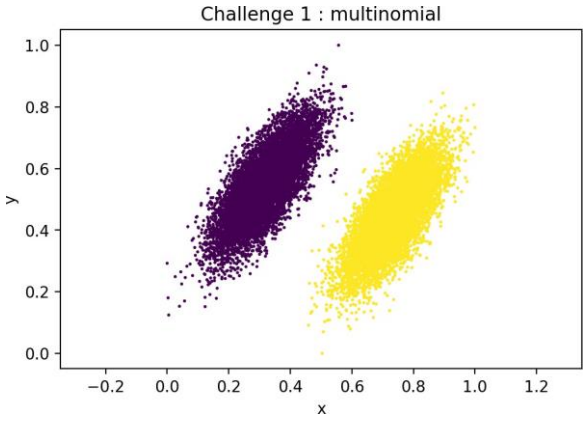
Imaginer comment se servir de ces techniques dans vos futurs domaines

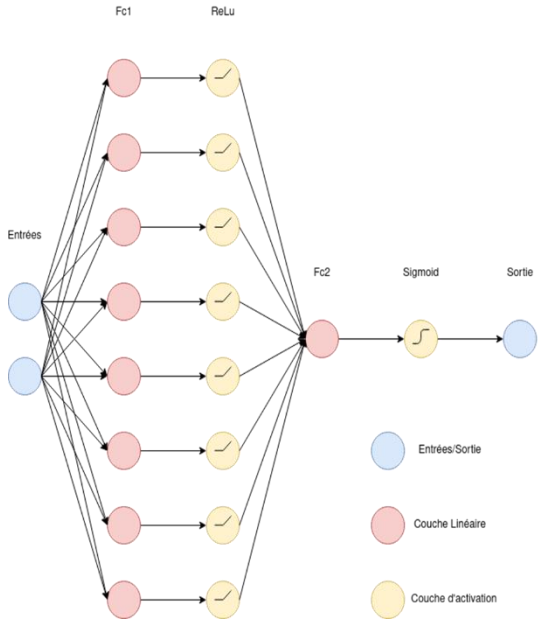
Un exemple de calcul: réseau de neurone peu profond

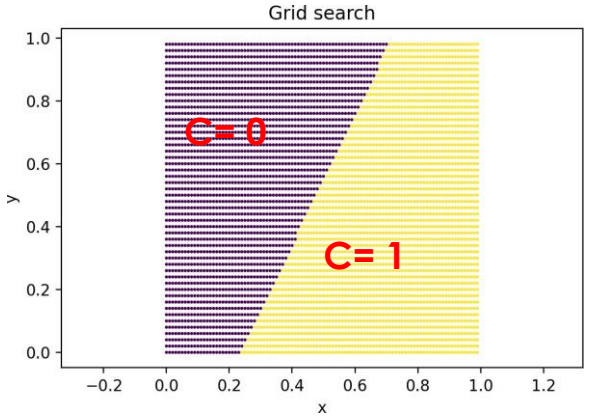




Principe: on fournit en entrée des coordonnées (x,y). Le réseau nous classifie les coordonnées selon la région d'appartenance. Ici, c'est une simple régression logistique avec une séparation de deux région linéaire.





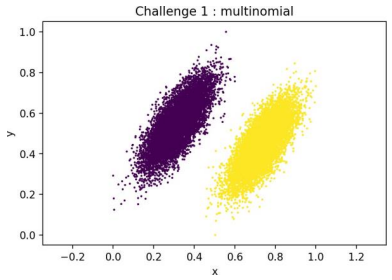


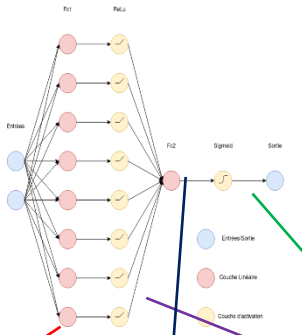
Il faut calculer

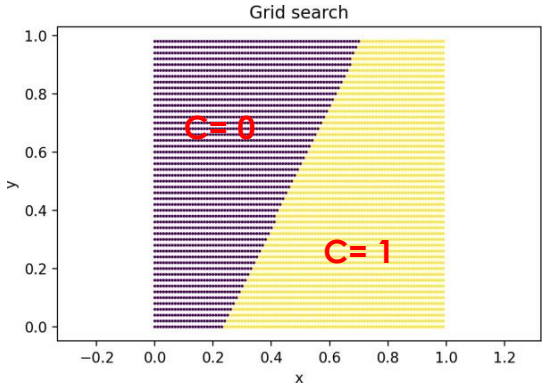
$$W = \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{pmatrix} \text{ et } B = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

$$s = \sigma(\mathbf{W}x + \mathbf{B})$$

Un exemple de calcul: réseau de neurone peu profond







X=0,2
Y=0,7

$$\begin{pmatrix} 1.7856 & -1.2345 \\ 2.4834 & -0.5797 \\ -0.3130 & 0.2658 \\ 0.3110 & -0.4116 \\ -0.4458 & 0.8010 \\ 3.4282 & -1.7166 \\ -0.6808 & 0.1631 \\ -0.5633 & 0.5694 \end{pmatrix} \text{X} \begin{pmatrix} 0.2 \\ 0.7 \end{pmatrix} + \begin{pmatrix} 0.0795 \\ -0.5202 \\ 0.4716 \\ -0.4349 \\ 1.3222 \\ -0.2305 \\ -0.3073 \\ 0.9498 \end{pmatrix} = \begin{pmatrix} -0.4275 \\ -0.4294 \\ 0.5951 \\ -0.6608 \\ 1.7938 \\ -0.7465 \\ -0.3293 \\ 1.2358 \end{pmatrix}$$

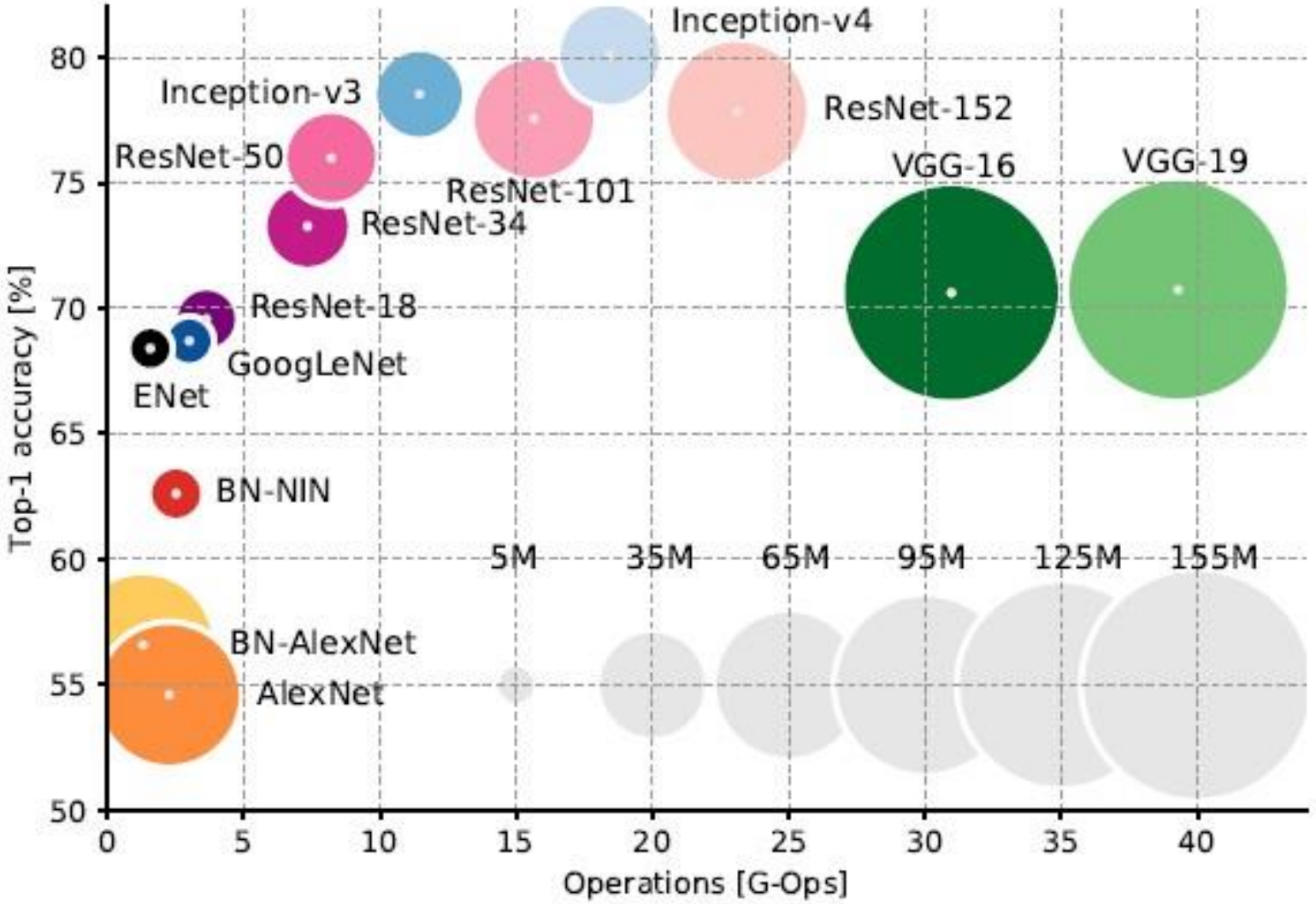
$$\begin{pmatrix} -0.4275 \\ -0.4294 \\ 0.5951 \\ -0.6608 \\ 1.7938 \\ -0.7465 \\ -0.3293 \\ 1.2358 \end{pmatrix} \xrightarrow{ReLu} \begin{pmatrix} 0 \\ 0 \\ 0.5951 \\ 0 \\ 1.7938 \\ 0 \\ 0 \\ 1.2358 \end{pmatrix}$$

$$(2.0441\ 2.5193 - 0.6255\ 0.0535\ - 1.5080\ 3.7406\ 0.1052\ - 1.2330)\ \text{X} \begin{pmatrix} 0 \\ 0 \\ 0.5951 \\ 0 \\ 1.7938 \\ 0 \\ 0 \\ 1.2358 \end{pmatrix} + (-1.4529) = (-6.0539)$$

$$(-6.0539) \xrightarrow{sigmoïde} (0.0023) \sim 0$$

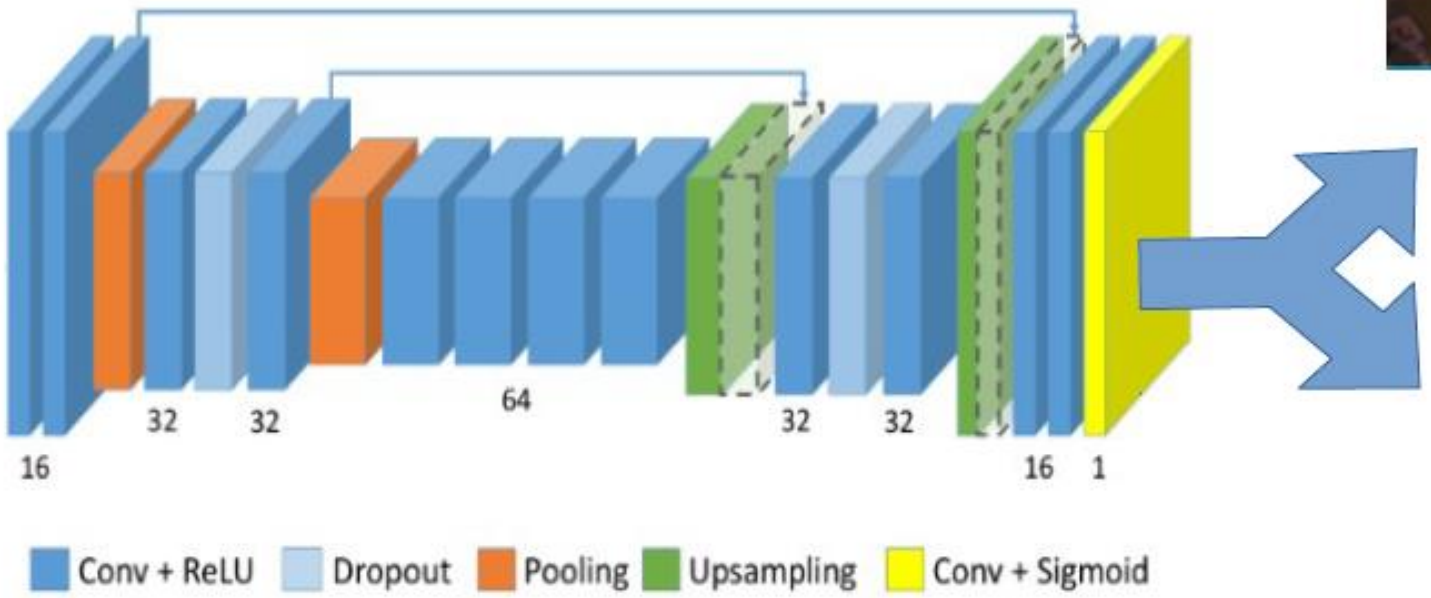
X=0,2
Y=0,7 Appartient à la zone **C= 0**

Les autres réseaux : Performances



Notre objectif

Implémenter les différents types de RN (DNN, CNN, RNN) dans des SE



L'approche embarquée : Une question d'optimisation

Entrées

RNN : modèle avec un énorme nombre de paramètres

Aspects matériels:

Créer un modèle synthétisable
Gestion des ressources

Objectifs

Transférer l'implementation sur ordinateur du réseau au niveau matériel

Optimiser le reseau pour trouver un compromis entre la performance (precision) et ressources matériels

Méthodes

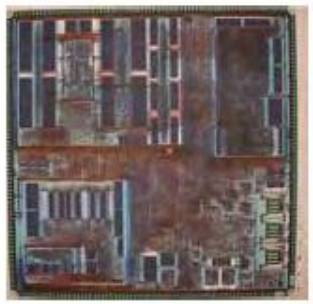
Elagage (Pruning):
Reduire la taille du réseau entrainé

Trouver un réseau optimisé en taille en même temps que l'entrainement

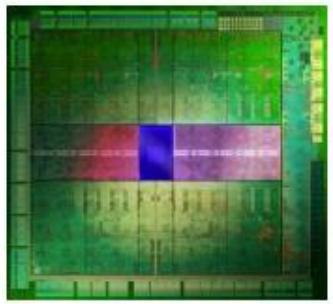
Votre travail : Votre mission

Système hétérogène à construire

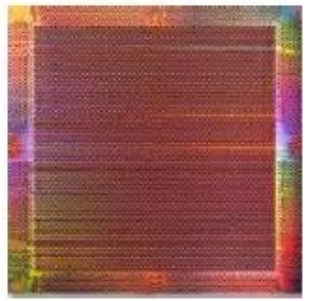
CPUs



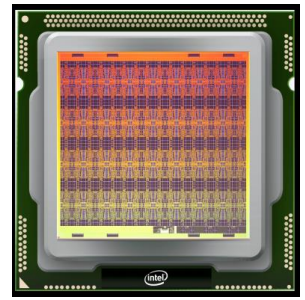
DSPs



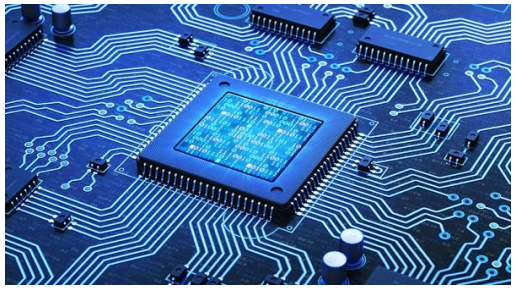
GPUs



FPGAs



NMC



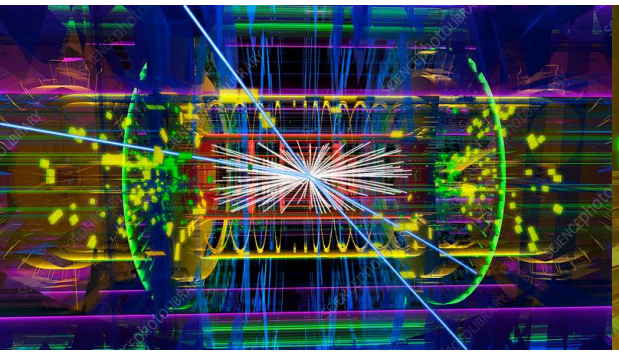
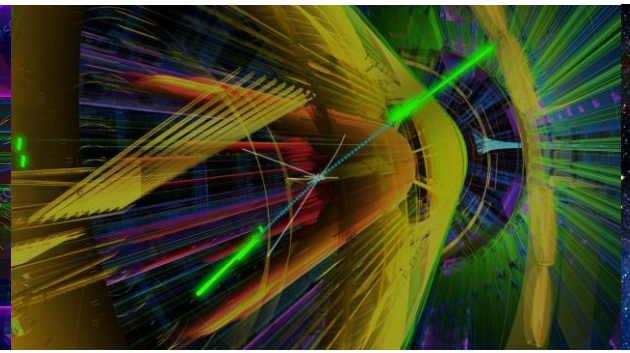
ASIC

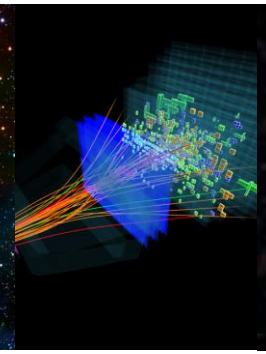
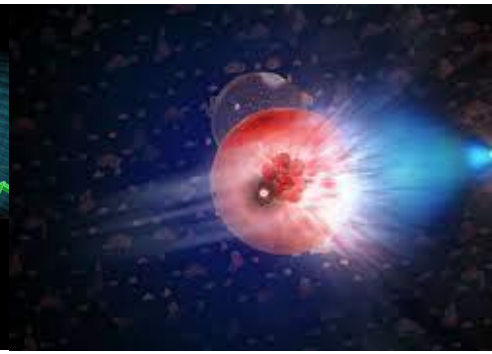
Les FPGA (SOC) allient un certains nombre d'avantages car ils rassemblent déjà un ensemble de fonctions dédiées:

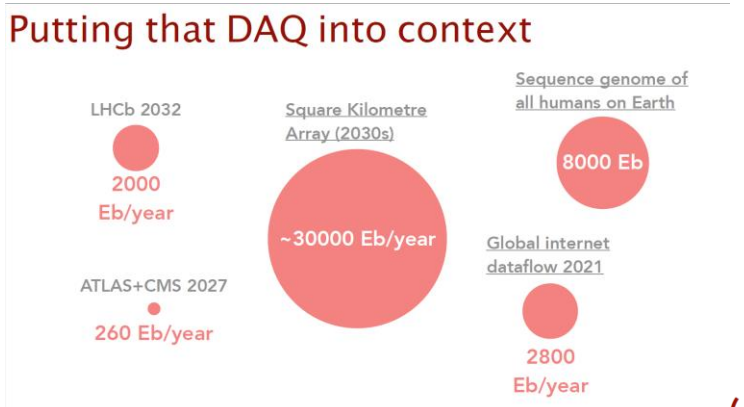
- CPU (ARM)
- DSP
- GPU
- IA engine
- DPU

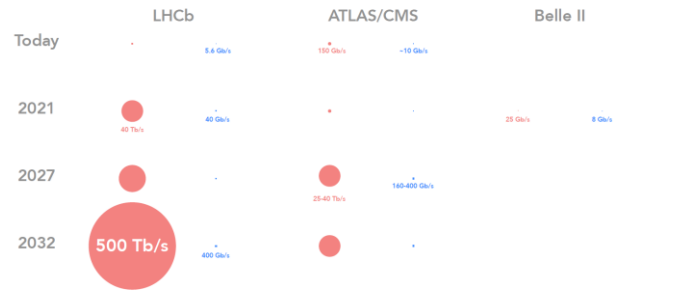
Les applications pilotant l'IA embarqué

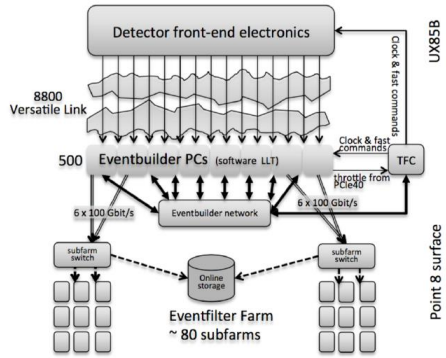
L'IA embarquée et la recherche fondamentale (en physique)

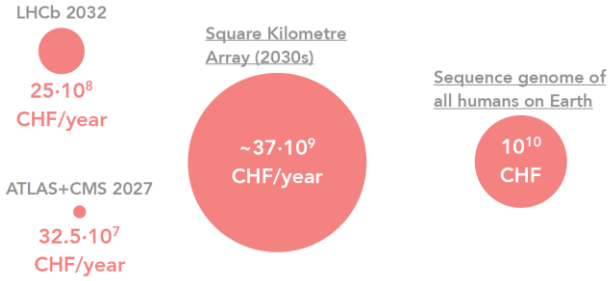











Haven't even discussed the cabling/networking: game over

Enjeux : Reduction des données en temps réel pour éviter le stockage sur disque (très couteux) :

➔ **Embarqué les algorithmes dans les nœuds de décisions**

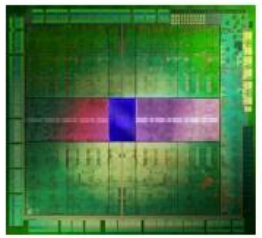
➔ **Optimiser les traitements (classifications….DL)**

➔ **Utiliser un mixte GPU, MPPA, FPGA**

Technologies Examples

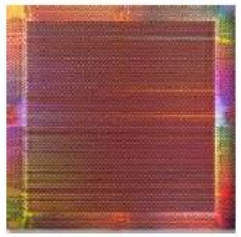
CPUs

MPPA



GPUs

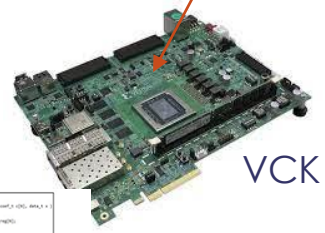
nVidia



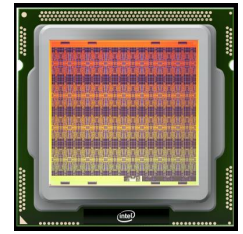
FPGAs

DE10-Agilex

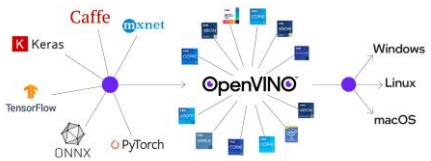
ZCU102,
ZCU104

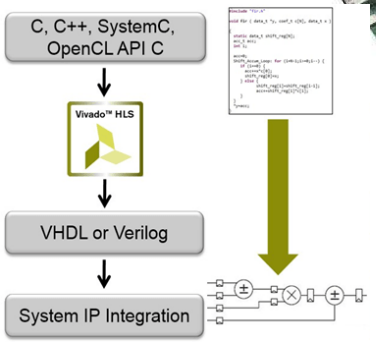


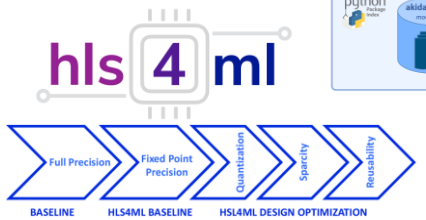
VCK190

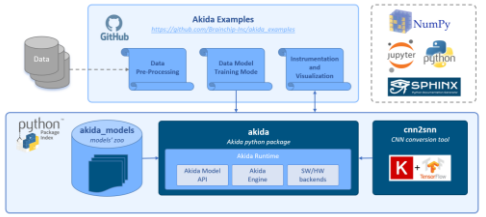


NMC



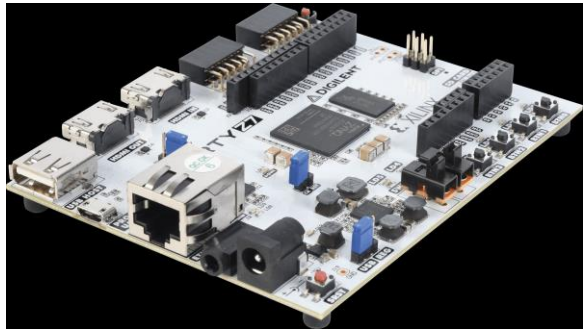






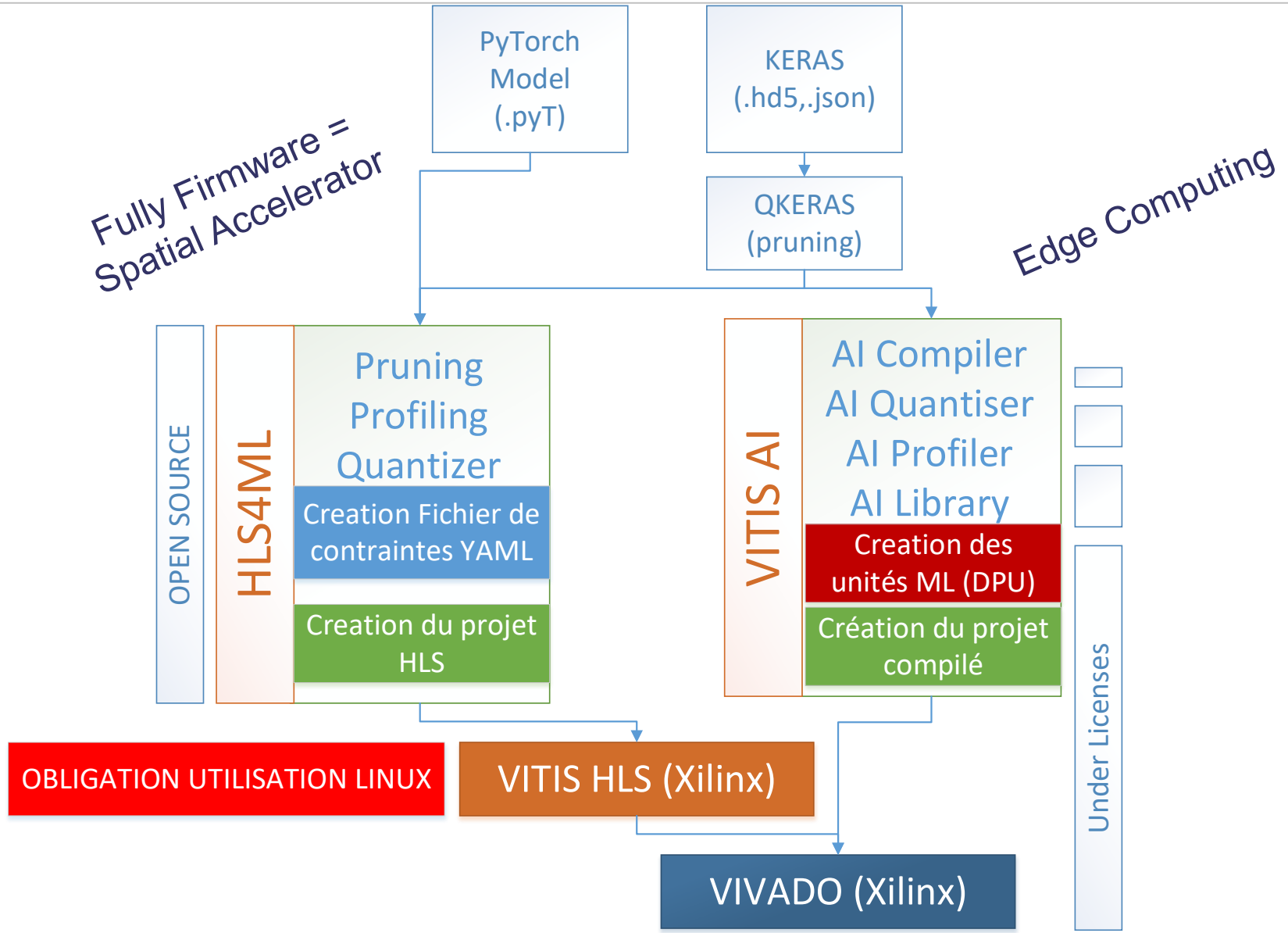
Méthodologie et test de portage de réseaux de neurones sur FPGA Xilinx Zynq avec DPU

ARTY Z7



ZCU102

2 façon d'intégrer l'IA en SE



INTEGRATION CONTINUE

PROCESSUS D'INTEGRATION D'UN RESEAU DE NEURONE PROFOND DANS UN FPGA

Mesures

Simulation

Bibliothèques des données d'apprentissage labellisées

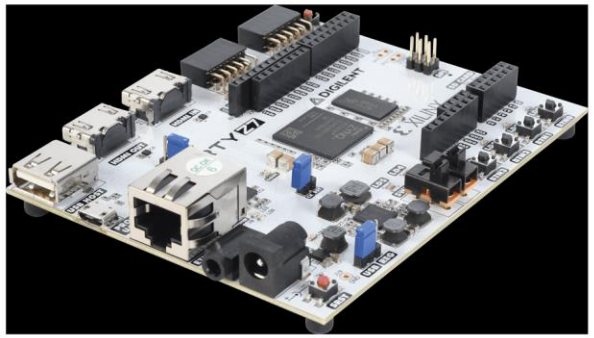
Analyse statistique des données (phase préparatoire)

Création du modèle :
- RNP
- RNC
- RNR …

Apprentissage: PyTorch Tensorflow-Keras

Poids

Modèle



Conversion des coefficients à virgules flottantes vers des nombres à virgules fixes

Redimensionnement des couches du réseau (prunning)

Reduction de l'empreinte mémoire et optimisation des calculs (Reuse factor)

Synthése et implementation du modèle dans un FPGA:
- « edge computing »
- Composants logiques (IP)

Validation avec jeux de données de test du composant

Validation sur données réelles du système (détecteur+ electronique)

hls4ml pipeline

Supported DL frameworks:

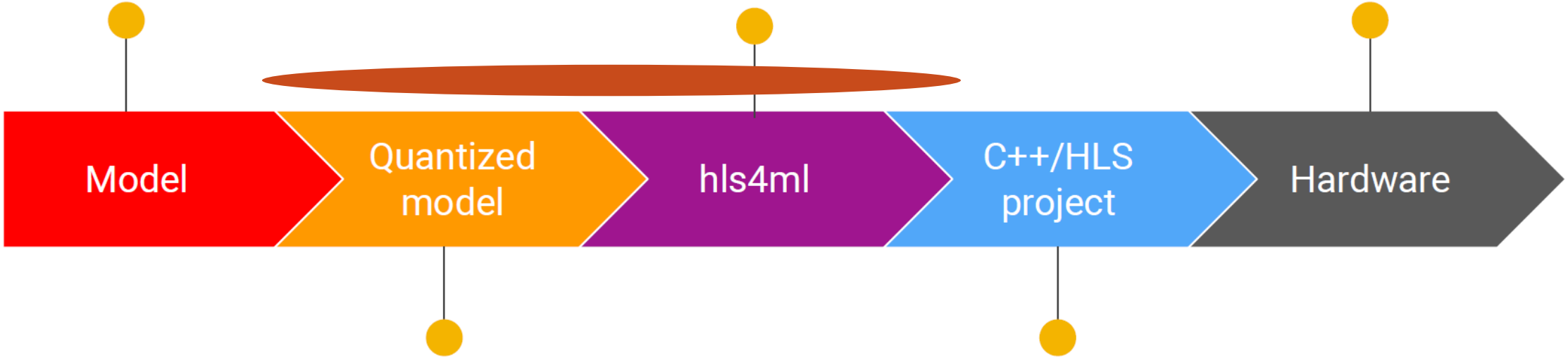
Keras
PyTorch
ONNX

Model conversion, optimization, profiling & tuning

hls4ml

Xilinx FPGAs, Intel/Altera FPGAs, Intel x86 CPUs

XILINX
intel

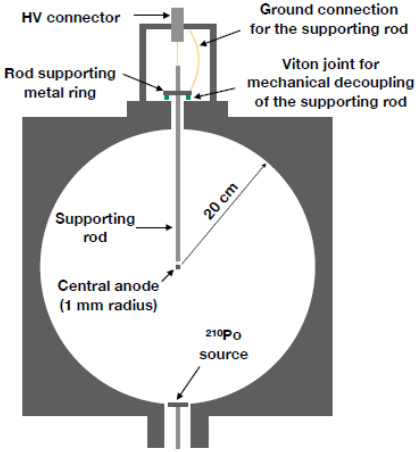


Quantization and pruning techniques:

- QKeras + AutoQ (Keras)
- Brevitas (PyTorch)

VIVADO
Mentor A Siemens Business
oneAPI
Quartus Prime

OWEN DNN



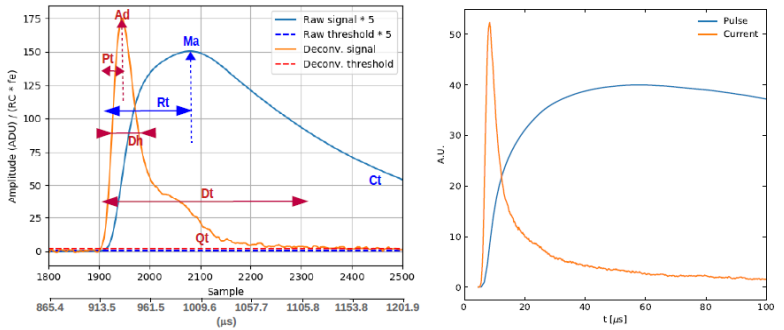
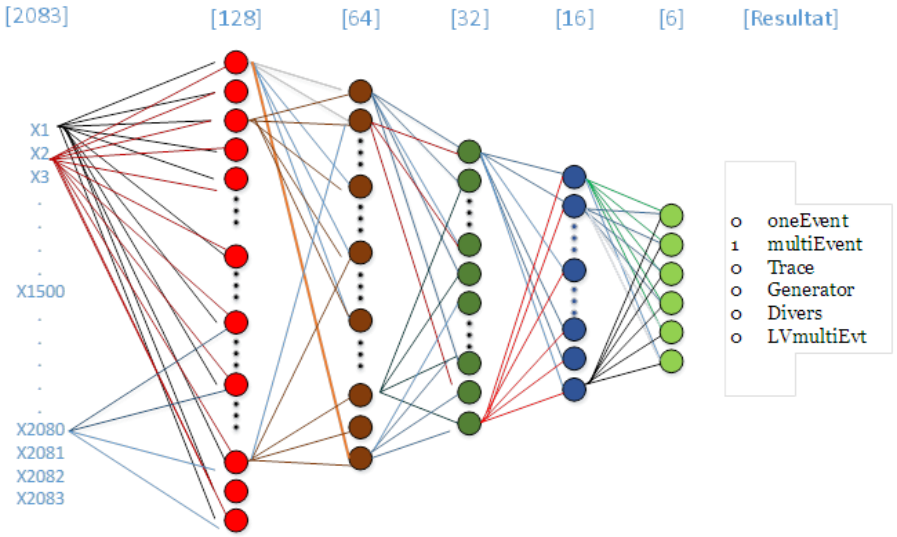
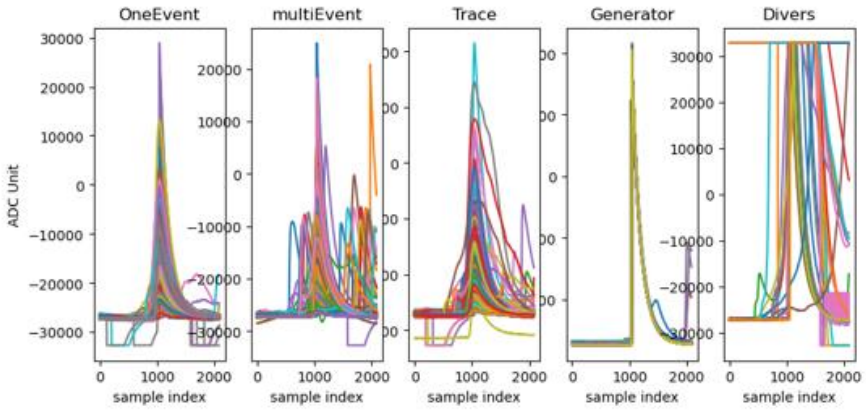
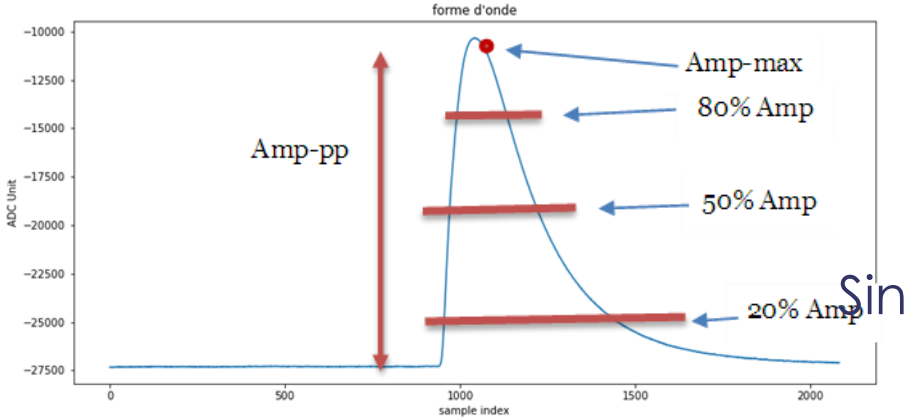


Figure 9. *(a) Real pulse with definition of observables used for the raw signal and for the deconvoluted one. (b) Example of simulated pulse, showing both the current and voltage signals.*

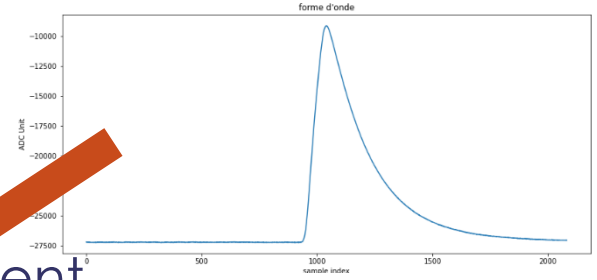




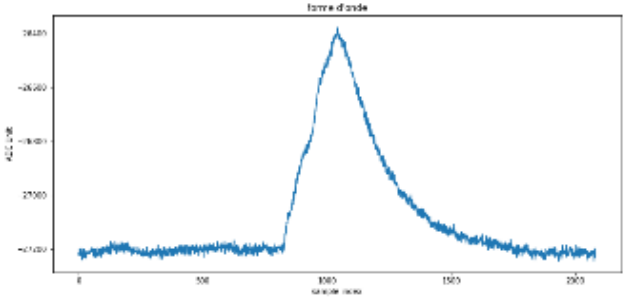
Learning phase with 21k labelled data



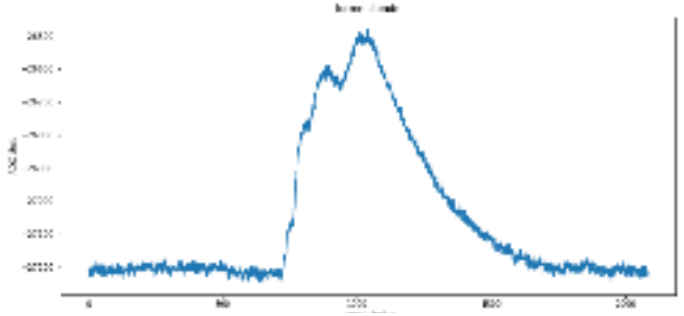
Single event

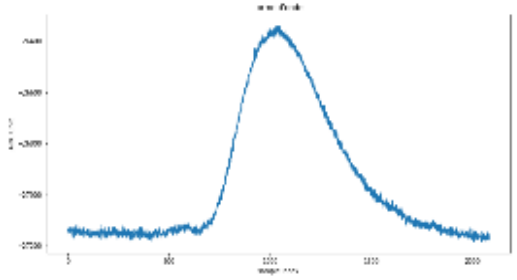


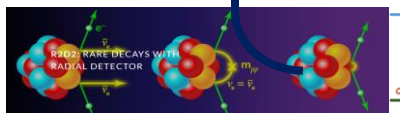
Trace



Multi particles in one event



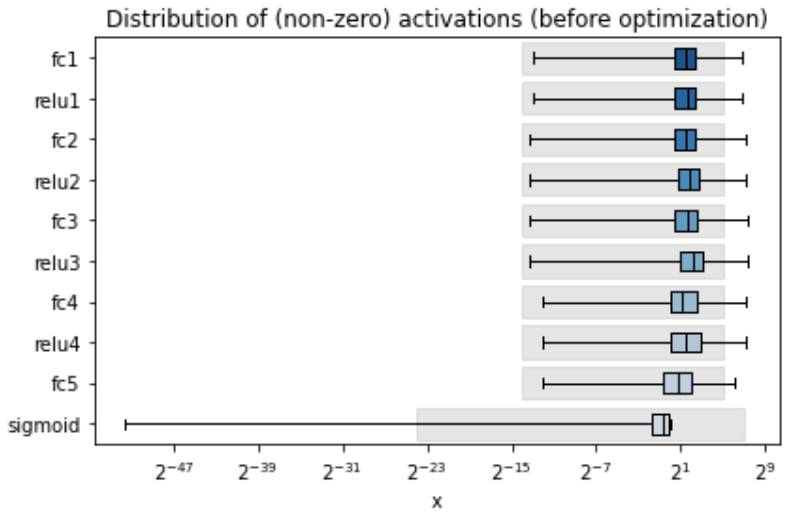


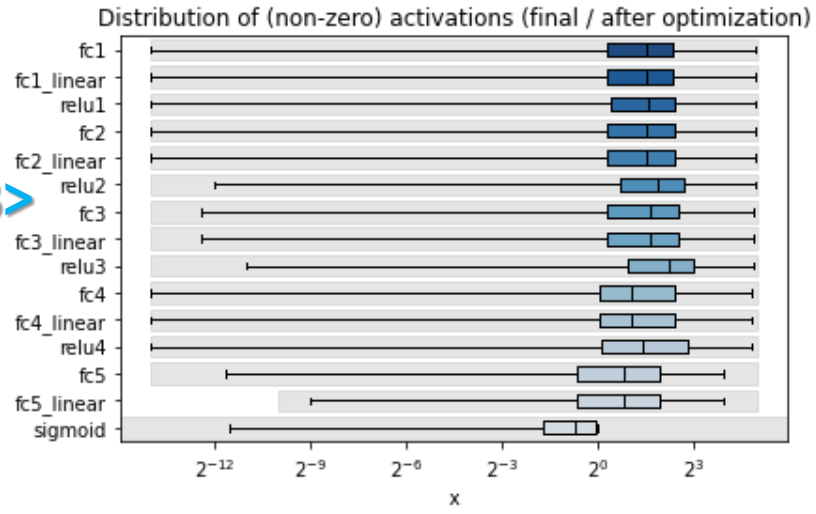


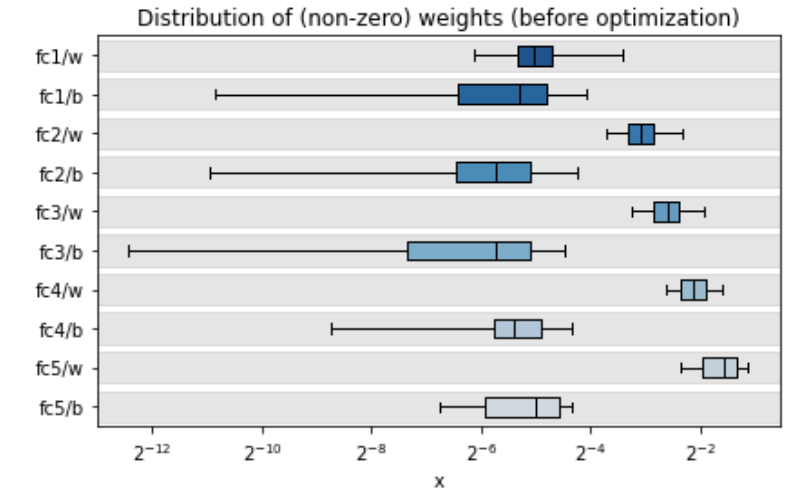
Profiling

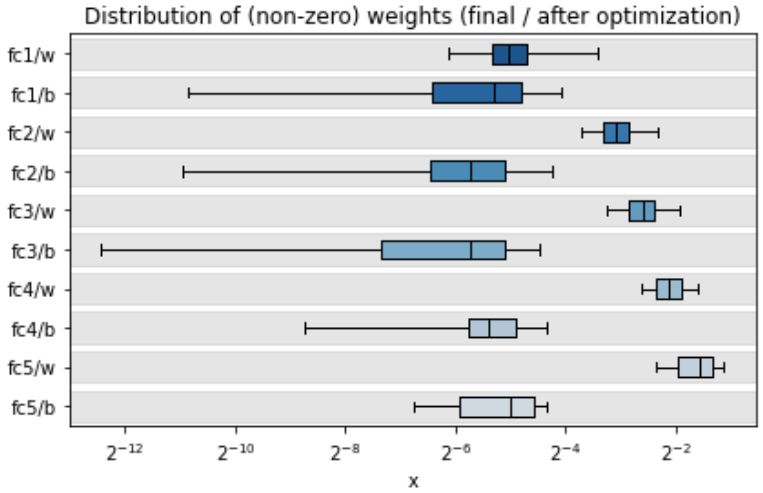
Floating point coding → Fixed point coding

HLS4ML ➔ Profiling and analyzing performances and coding









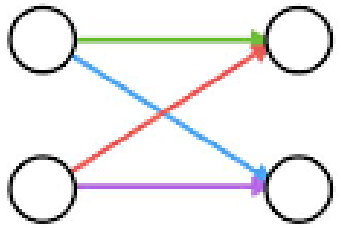
Les paramètres importants pour HLS4ML

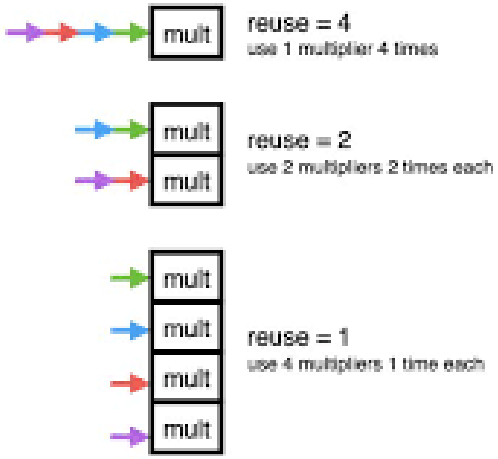
La **précision** de la quantification (Vigule flottante → Virgule fixe)

Exemple : ap_fixed<16,6>

15	0.75
001111	1100000000

Reuse Factor → Nombre de fois que l'IP boucle sur la ressource de calcul du FPGA





Strategy :

→ Latence : privilégie la vitesse d'écution

→ Minimise les ressources au détriment de la latence

HLS4ML Config

- **'model'** :

```
config['Model']['Precision'] = 'ap_fixed<32,10>'
config['Model']['ReuseFactor'] = 100
config['Model']['Strategy'] = 'latency' ]
```

MODEL

- **'type'** :

```
config['LayerType']['Dense']['Precision']['weight'] = 'ap_fixed<22,2>'
   config['LayerType']['Dense']['Precision']['bias']   = 'ap_fixed<22,2>'
   config['LayerType']['Dense']['Precision']['result'] = 'ap_fixed<32,10>'
   config['LayerType']['Dense']['ReuseFactor'] = 100
   config['LayerType']['Activation']['ReuseFactor'] = 100
   config['LayerType']['Activation']['Precision'] = 'ap_fixed<32,10>']
```

LAYER

- **'name'** :

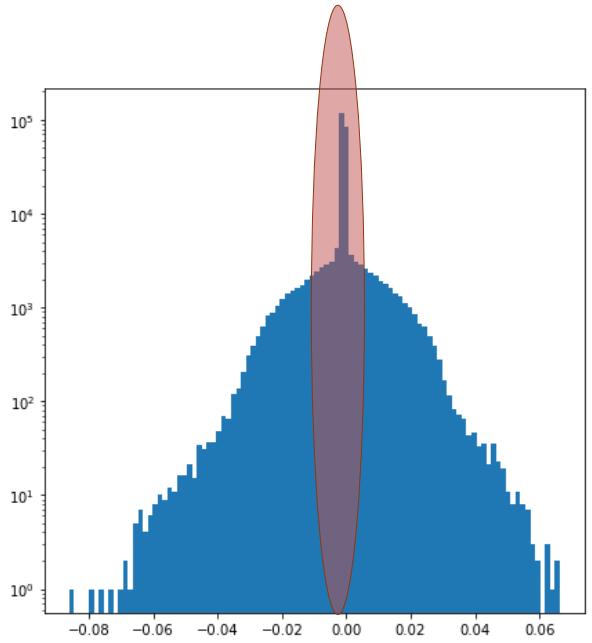
```
#config['LayerName']['input_10']['Precision']['result']   = 'ap_fixed<32,8>'
# config['LayerName']['fc1']['Precision']['weight']   = 'ap_fixed<32,8>'
# config['LayerName']['fc2']['Precision']['weight']   = 'ap_fixed<32,8>'
# config['LayerName']['fc3']['Precision']['weight']   = 'ap_fixed<32,8>'
# config['LayerName']['fc4']['Precision']['weight']   = 'ap_fixed<32,8>'
# config['LayerName']['fc5']['Precision']['weight']   = 'ap_fixed<32,8>'

# config['LayerName']['fc1']['Precision']['bias']    = 'ap_fixed<32,8>'
# config['LayerName']['fc2']['Precision']['bias']    = 'ap_fixed<32,8>'
# config['LayerName']['fc3']['Precision']['bias']    = 'ap_fixed<332,8>'
# config['LayerName']['fc4']['Precision']['bias']    = 'ap_fixed<32,8>'
# config['LayerName']['fc5']['Precision']['bias']    = 'ap_fixed<32,8>'
...
# config['LayerName']['sigmoid']['Precision']       = 'ap_fixed<32,8>'
# config['LayerName']['sigmoid']['table_t']        = 'ap_fixed<32,8>'
```

DETAIL

Pruning

```
pruning_params = {
     "pruning_schedule" :
tfmopt.sparsity.keras.PolynomialDecay(initial_sparsity=0.5,
final_sparsity=0.80,
begin_step=0,
end_step=20000)
```

```
model_for_pruning = prune.prune_low_magnitude(keras_model,
**pruning_params)
```



One layer with pruning action

With pruning

```
+ Latency:
    * Summary:
    +---------+---------+----------+----------+------+------+----------+
    |  Latency (cycles) |   Latency (absolute)  |   Interval  | Pipeline |
    |   min   |   max   |    min   |    max   | min  | max  |   Type   |
    +---------+---------+----------+----------+------+------+----------+
    |     4169|     4170| 20.845 us | 20.850 us |  4170|  4171| dataflow |
    +---------+---------+----------+----------+------+------+----------+
    ================================================================
    == Utilization Estimates
    ================================================================
    * Summary:
    +-----------------+---------+-------+--------+--------+-----+
    |       Name      | BRAM_18K| DSP48E|   FF   |   LUT  | URAM|
    +-----------------+---------+-------+--------+--------+-----+
    |DSP              |        -|      -|       -|       -|    -|
    |Expression       |        -|      -|       0|      32|    -|
    |FIFO             |        0|      -|    3660|   23424|    -|
    |Instance         |      340|    133|  204880|  141798|    -|
    |Memory           |        -|      -|       -|       -|    -|
    |Multiplexer      |        -|      -|       -|      36|    -|
    |Register         |        -|      -|       6|       -|    -|
    +-----------------+---------+-------+--------+--------+-----+
    |Total            |      340|    133|  208546|  165290|    0|
    +-----------------+---------+-------+--------+--------+-----+
    |Available        |     1824|   2520|  548160|  274080|    0|
    +-----------------+---------+-------+--------+--------+-----+
    |Utilization (%)  |       18|      5|      38|      60|    0|
    +-----------------+---------+-------+--------+--------+-----+
```

No pruning

```
    +-----------------+---------+-------+--------+--------+-----+
    |Total            |     1998|   6004|  217917|  245213|    0|
    +-----------------+---------+-------+--------+--------+-----+
    |Available        |     1824|   2520|  548160|  274080|    0|
    +-----------------+---------+-------+--------+--------+-----+
    |Utilization (%)  |      109|    238|      39|      89|    0|
    +-----------------+---------+-------+--------+--------+-----+
```

HLS4ML Results with OWEN NN

Input:
Matrix : 2083x32bit

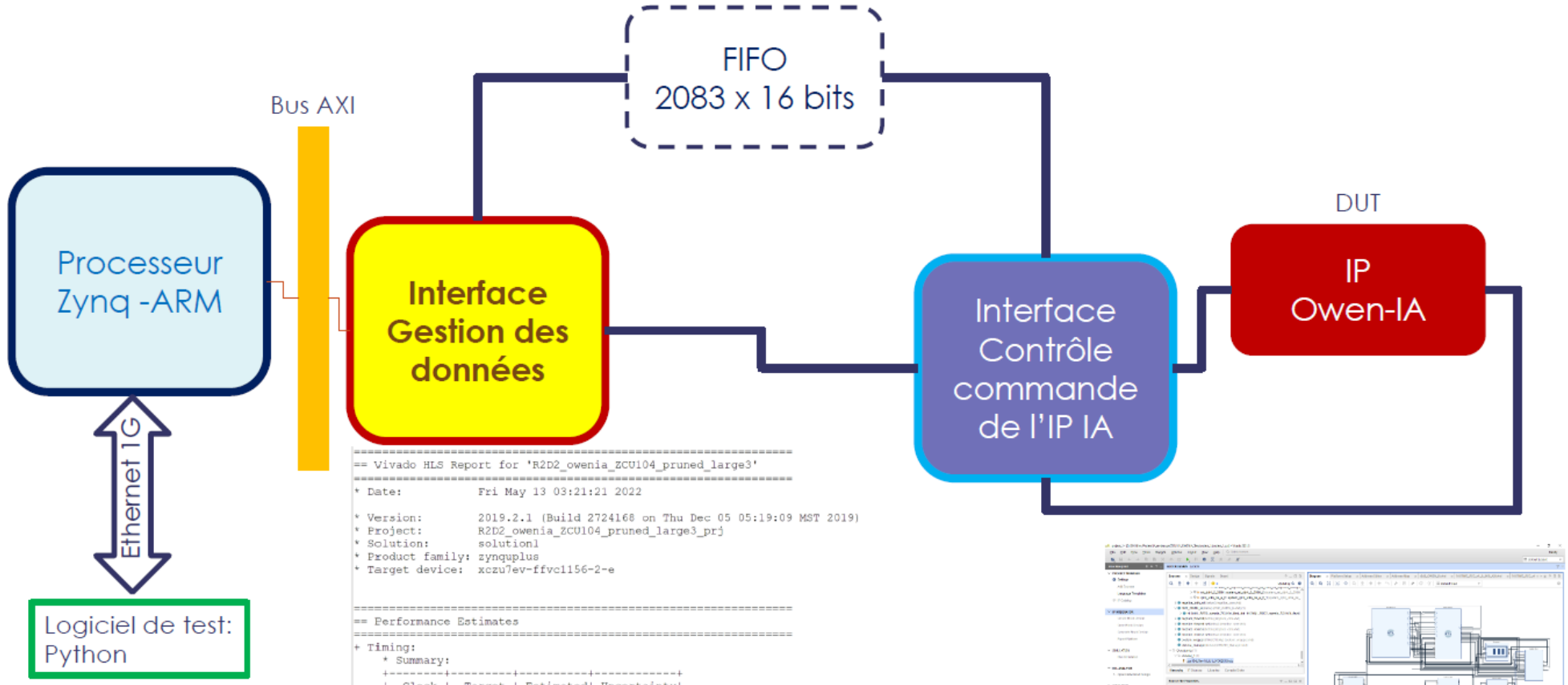
HLS input conversion

2083 Input of 32 bits (difficulties to manage)

Flatten: std_logic_vector(2083x32 downto 0)
→ vhdl to write

Development board	PC on Windows	ARTY Z7	ZCU102 (default setting)	ZCU102 (default setting)	ZCU102 (pruning)	ZCU104 (pruning)	ZCU104 (QKeras)	ZCU104 (QKeras)	ZCU102 (Qkeras)
Number of bits (Nbits, Integer size)	(32,32)	(16,6)	(16,6)	(32,8)	(20,6)	(20,6)	(16,6)	(20,6)	(16,3)
Number of DSP	20 Core	509/220	140	140	133	275	5852	5852	6004
Number of slice	--	170000	156263	156263	165290	163678	307125	250761	245213
Result speed	NA	NA	10,43µs	10,43µs	20,85µs	10,43µs	10,43µs	10,43µs	10,43µs
Score (good answer)	79,7%	45%	65%	76.3%	79,6%	74,2%	72-79%	72-79%	72-79%
Used ressources (DSP/LUT)	--	231%/310%	5%/57%	5%/57%	5%/60%	15%/75%	338%/104%	338%/108%	238%/89%

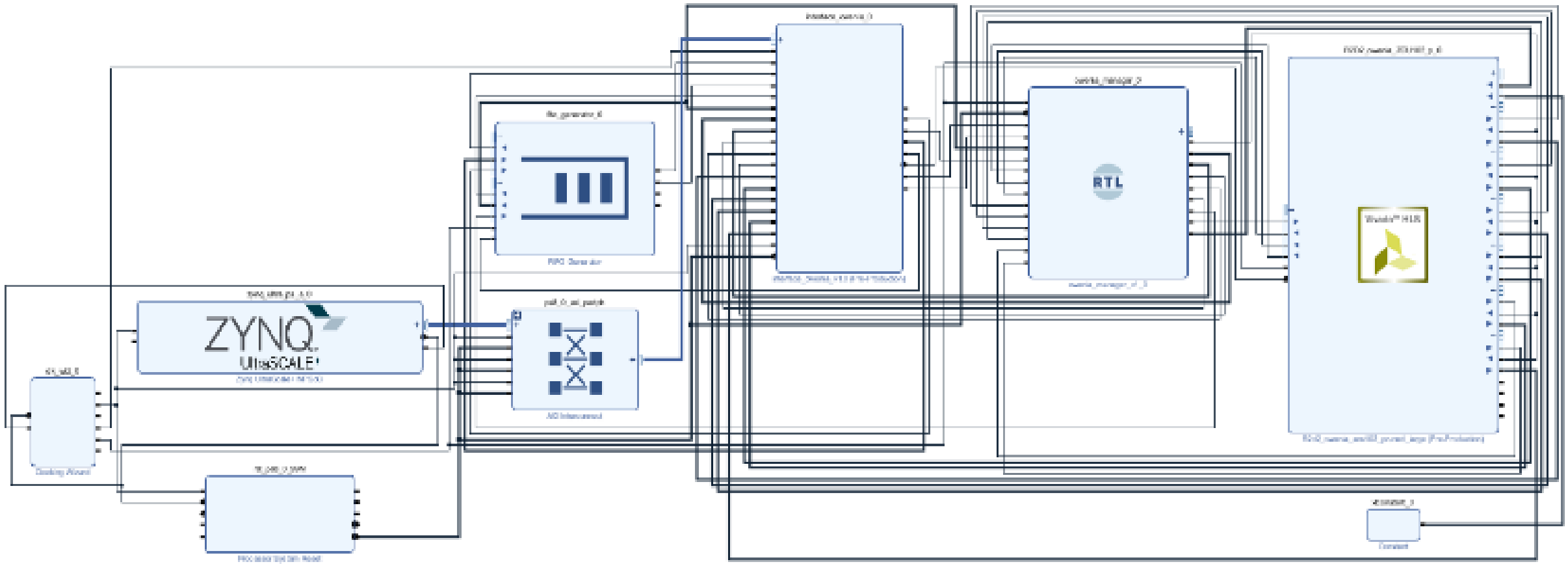
Owen Test in real condition



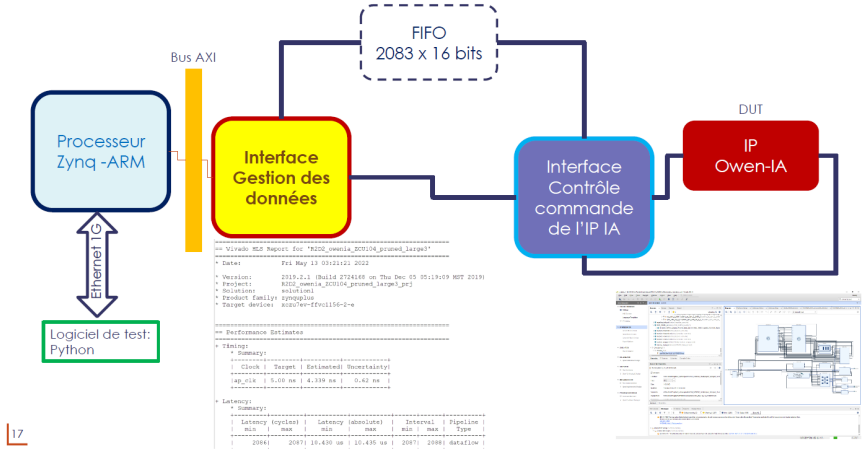

```
================================================================
== Vivado HLS Report for 'R2D2_owenia_ZCU104_pruned_large3'
================================================================
* Date:           Fri May 13 03:21:21 2022

* Version:        2019.2.1 (Build 2724168 on Thu Dec 05 05:19:09 MST 2019)
* Project:        R2D2_owenia_ZCU104_pruned_large3_prj
* Solution:       solution1
* Product family: zynquplus
* Target device:  xczu7ev-ffvc1156-2-e


================================================================
== Performance Estimates
================================================================
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 5.00 ns | 4.339 ns |   0.62 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+-----------+-----------+------+------+----------+
    | Latency (cycles)  |   Latency (absolute)  |   Interval  | Pipeline |
    |   min   |   max   |    min    |    max    |  min |  max |   Type   |
    +---------+---------+-----------+-----------+------+------+----------+
    |     2086|     2087| 10.430 us | 10.435 us |  2087|  2088| dataflow |
```

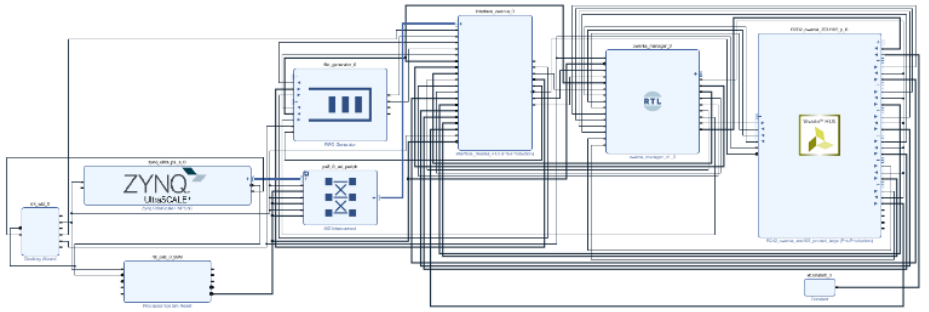

17

Owen Test in real condition



Owen Test in real condition





Categories	% Bonnes reponses	% Mauvaises réponses
oneEvent	52.4	47.6
MultiEvent	61.3	38.7
Trace	43.4	56.6
Generateur	94.0	6.0
Divers	98.3	1.7
Signaux_faibles	99.4	0.6

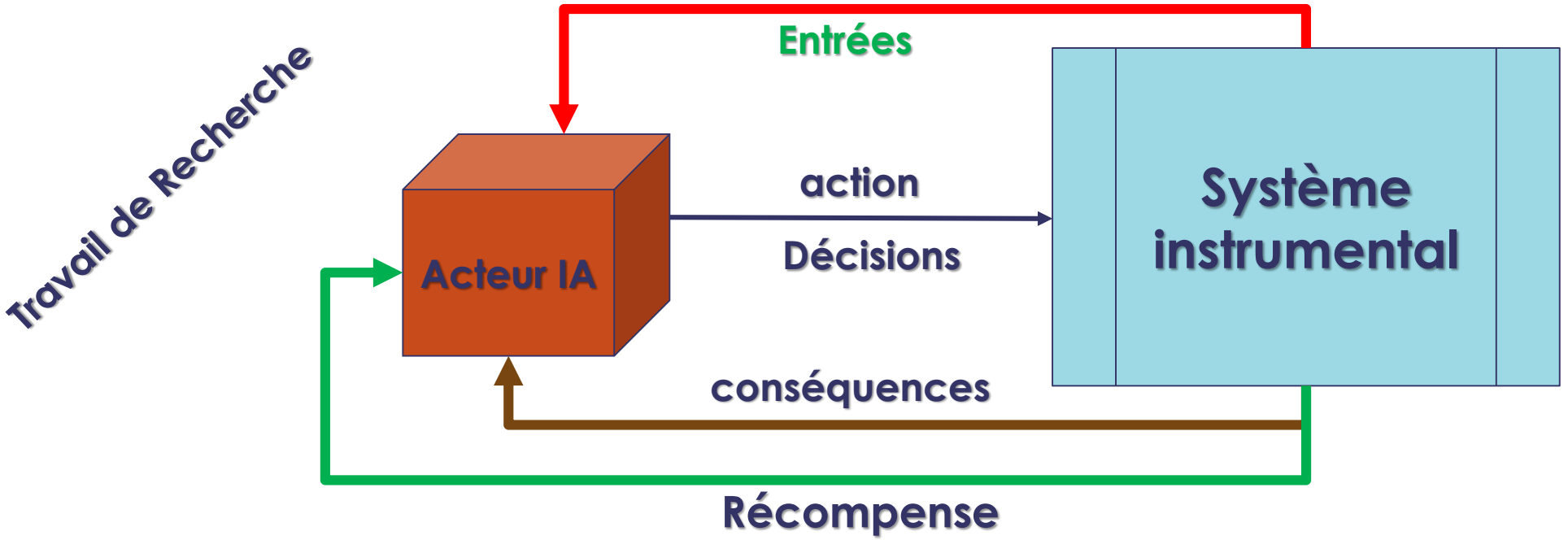
Problem of precision in coding number <20,3>
Lake of labelled input data during the learning phase

Apprentissage par renforcement

Apprentissage automatique → Problématique → **Génération de données labellisés**

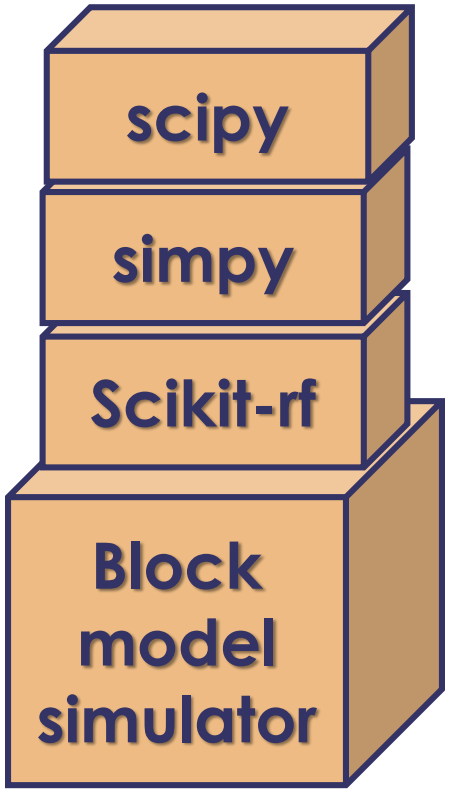
LHC-HL : Données du LHC
Autres projets : Prototype détecteur

IDEE : Créer un modèle d'ingénierie système dans un environnement de simulation

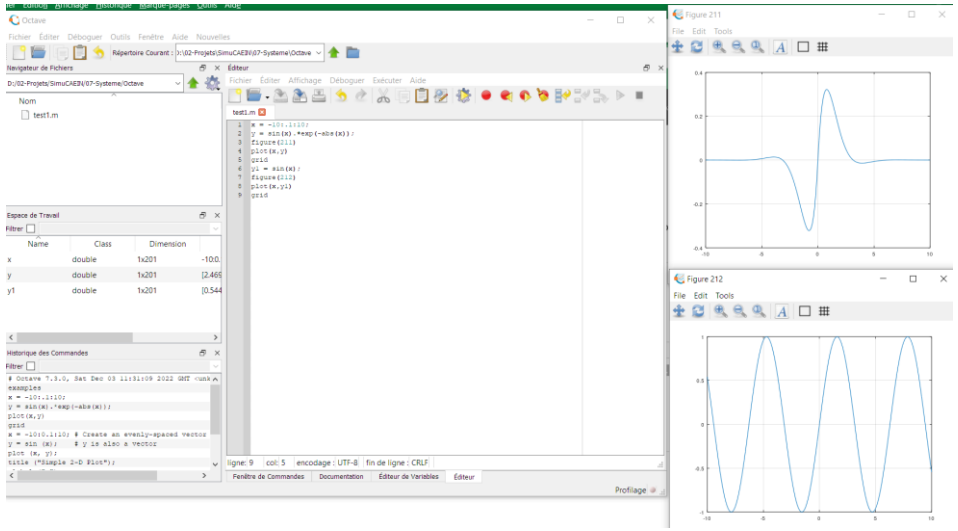


Les outils open sources matures d'ingeniering models

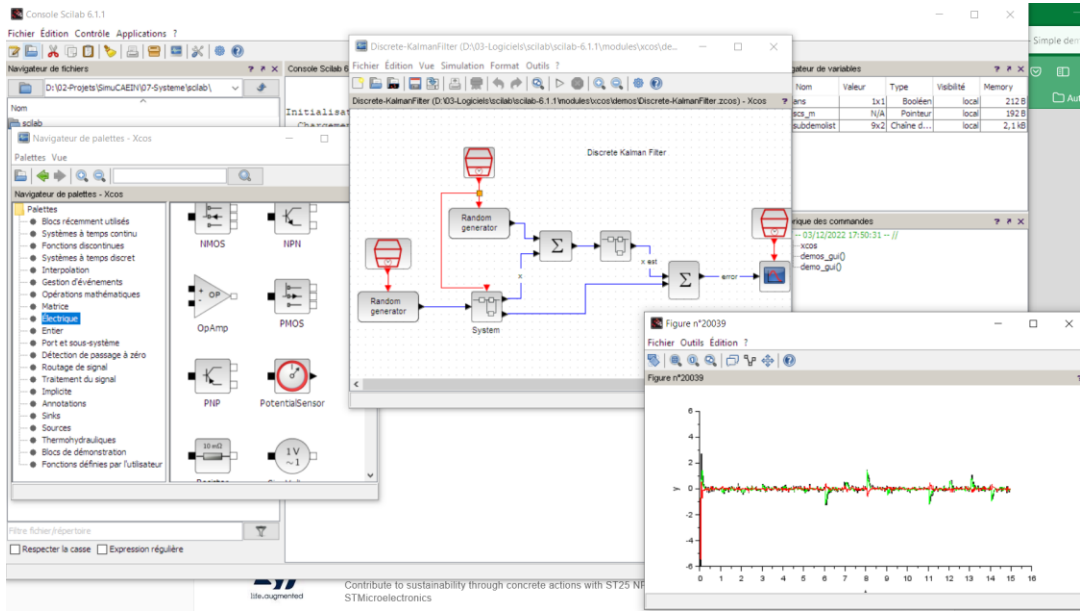
Projet Python



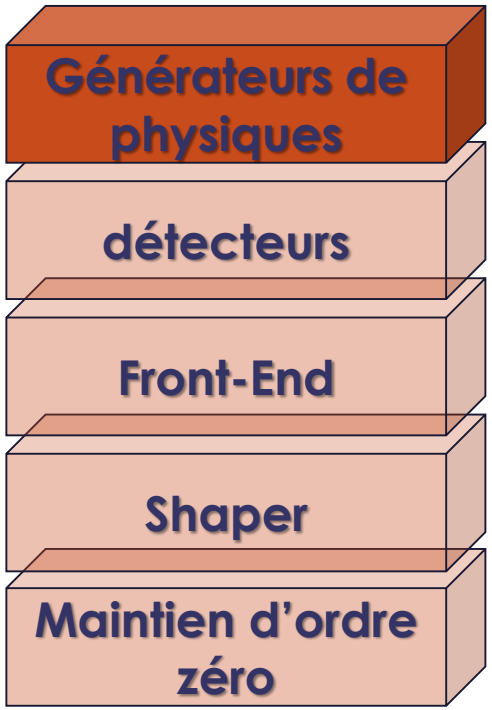
Octave

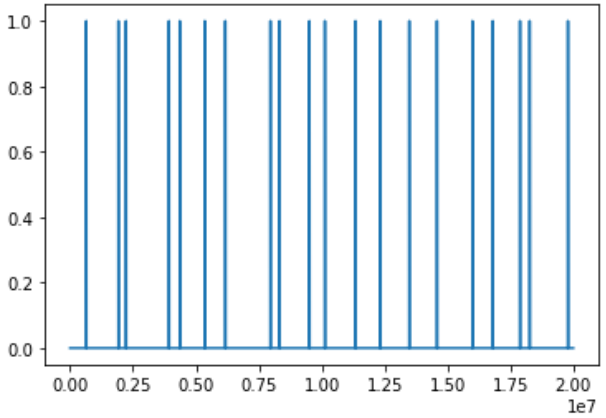


scilab

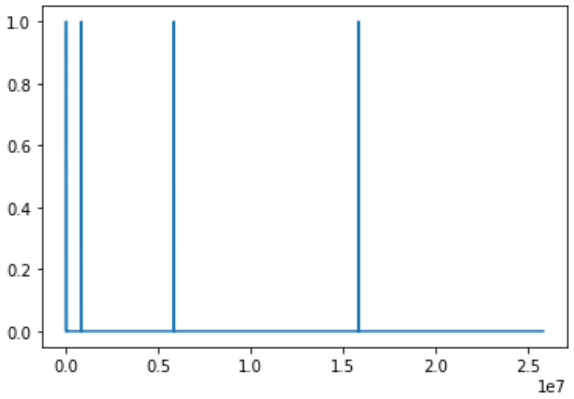


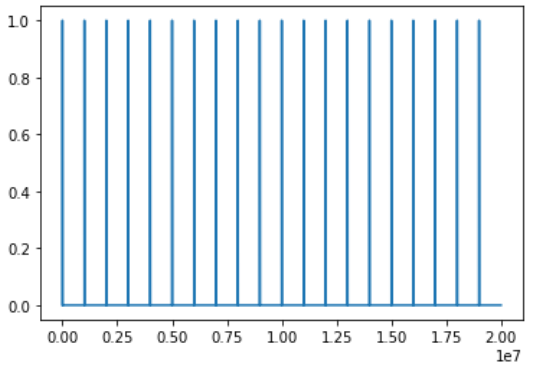
SimCAEIN : Modelisation chaine electronique (python)



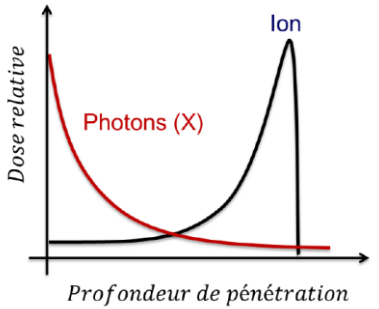


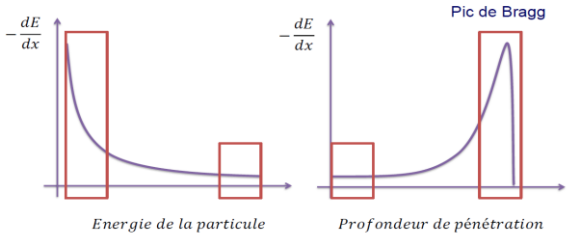

```
##########################
# SIGNAL GENERATOR
##########################

##########################
#  IMPULSION TRAIN       #
##########################
simu_time = 1e-3
wide_pulse = 1e-7
resolution_pulse = 40000
nbre_pulse = 20
dense_pulse = 20
percent_occupency = (nbre_pulse * wide_pulse) / simu_time
nbre_pas_simu_t = int(resolution_pulse / percent_occupency)
print('nbre_pas_simu_t:', nbre_pas_simu_t)
nbre_pas_simu = nbre_pas_simu_t
# nbre_pas_simu = 20000000
period_pulse = int(nbre_pas_simu / nbre_pulse)
imps = periodicPulse(nbre_pulse, period_pulse)
# imps = randomPulse(nbre_pulse, period_pulse)
# imps = poissonPulse(period_pulse, nbre_pas_simu, dense_pulse)
plt.plot(imps)
plt.show()
print(len(imps))
imps = imps[:len(imps)-1]
print(len(imps))

##########################
# SIMULATION
##########################
tsimu = np.linspace(0, simu_time, nbre_pas_simu, endpoint=False)
train = np.empty(nbre_pas_simu)
tdelay, yimpulse = calculDelay(tsimu, imps)
Nd = 0
nd_tab = []
for dly in yimpulse:
    Nd = randomNd(100)
    # Nd = Nd + 1
    sipm_prop.setPropSimu('NbreCelluleDeclenchee', Nd)
    sipm_model.setProperties(sipm_prop)
    t1, y1, y11 = oneSignalGenerator(sipm_prop, sipm_model,
                                     wide_pulse, resolution_pulse,
                                     display=False)
    length_pulse = len(y11)
    train[dly:dly + length_pulse] = [
        a + b for (a, b) in zip(train[dly:dly + length_pulse], y11)]
    nd_tab.append(Nd)
plt.plot(tsimu, train, 'k', c='g', linewidth=1.5,
         label='N=1000, Nf = '+str(nd_tab))
plt.title("train d'impulsion periodique d'impulion Sipm")
plt.legend(loc='best', shadow=True, framealpha=1)
plt.ylabel('Icharge (A)')
plt.xlabel('t(s)')
# plt.xlim(7.5e-5, 0.85e-4)
# plt.ylim(-1e-10, 100)
plt.show()
```

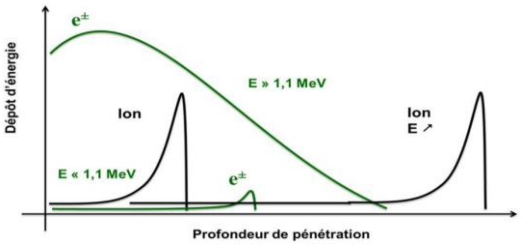




Signaux de Dirac avec une distribution:
-Normal
-Poisson
-Periodique







SimCAEIN : Modelisation chaine electronique (python)

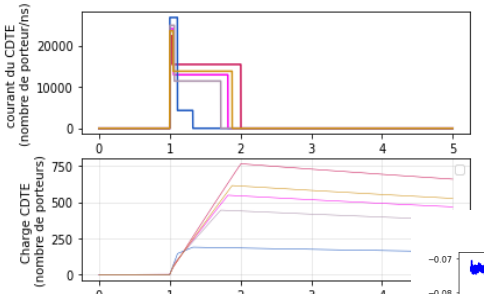
- Générateurs de physiques
- détecteurs
- Front-End
- Shaper
- Maintien d'ordre zéro

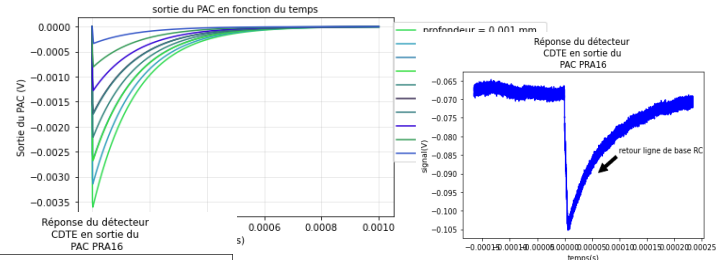
CdTe

$$t_e = \frac{L - x_o}{\mu_n . E_{bias}} \, , t_h = \frac{x_o}{\mu_h . E_{bias}}$$

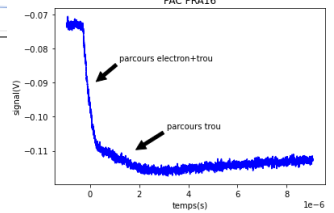
$$Q_e = \frac{n_o . q . \left\| \overrightarrow{E_{bias}} \right\|}{L} . [\mu_n \tau_n^c (1 - \exp\left(\frac{-(L - x_o)}{\mu_n \tau_n^c E_{bias}}\right)]$$

$$Q_h = \frac{n_o . q . \left\| \overrightarrow{E_{bias}} \right\|}{L} . [\mu_p \tau_p^c \left(1 - \exp\left(\frac{-x_o}{\mu_p \tau_p^c E_{bias}}\right)\right)]$$

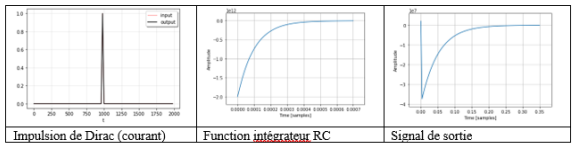


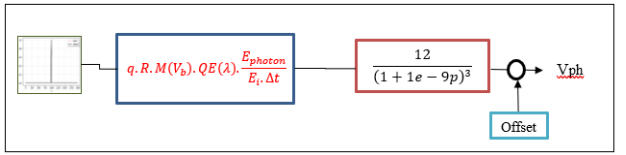


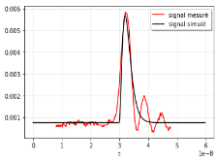
$$Q(t) = \frac{E_p . q}{E_i . 2.10^{-3}} . [8{,}47 . (1 - e^{\frac{-(2.10^{-3} - x_0)}{8{,}47}}) + 1{,}63.10^{-3} . (1 - e^{\frac{-x_0}{1{,}63.10^{-3}}})]$$

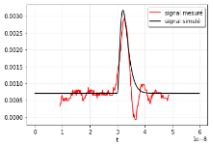


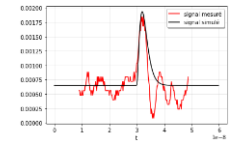
APD

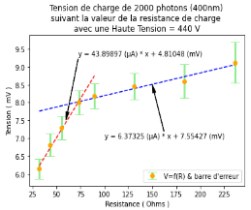


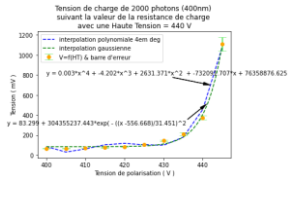




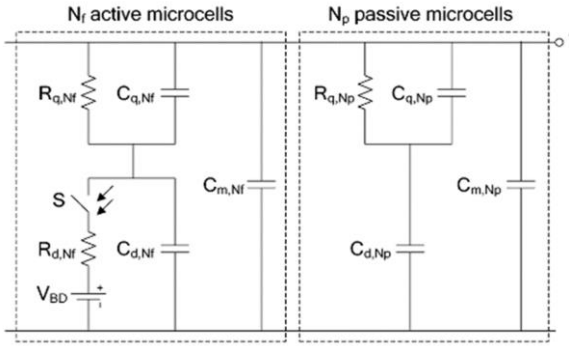


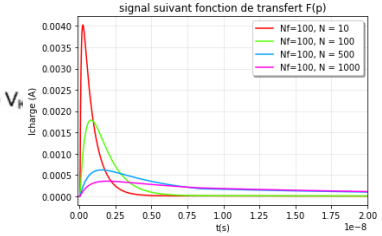


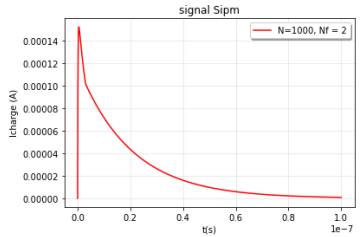


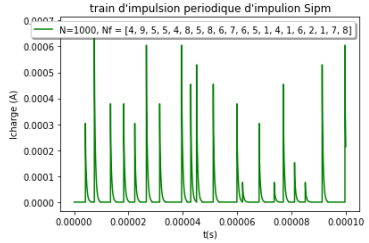


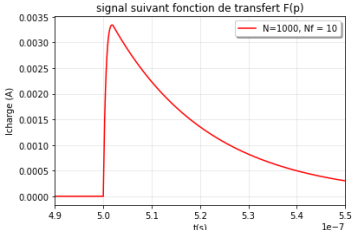
SIPM

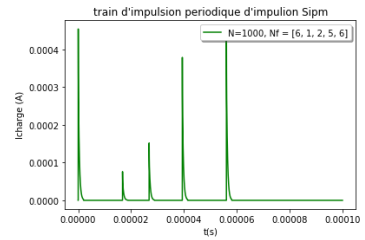


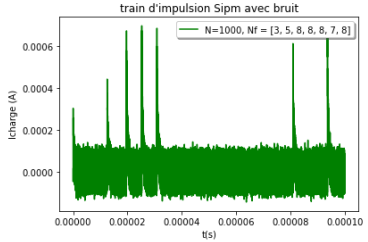












SimCAEIN : Modelisation chaine electronique (python)

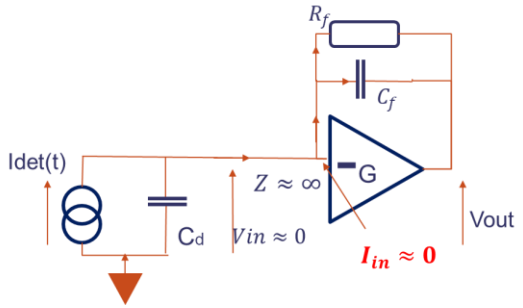
Générateurs de physiques

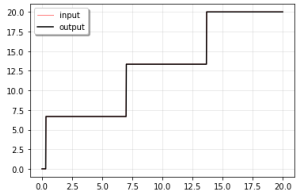
détecteurs

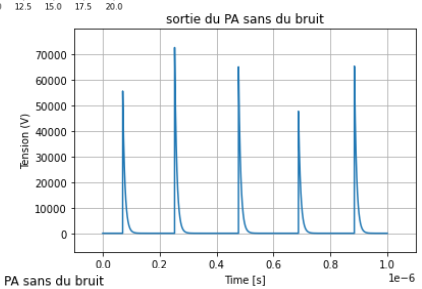
Front-End

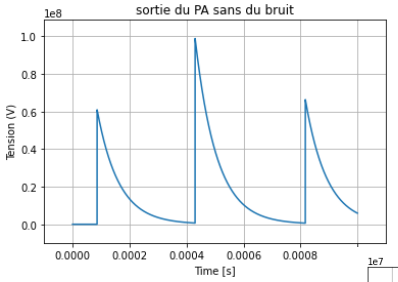
Shaper

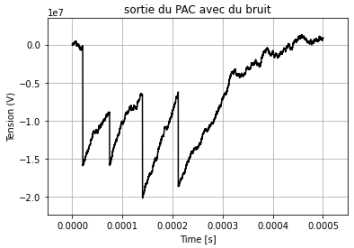
Maintien d'ordre zéro



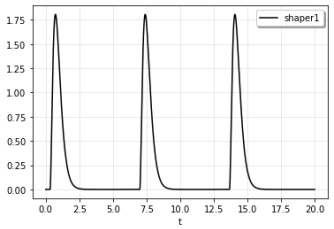


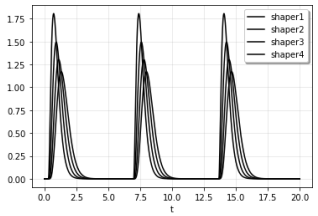


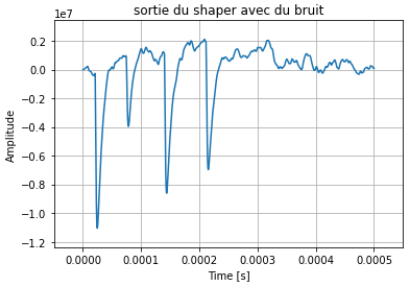




Filtrage Shaper





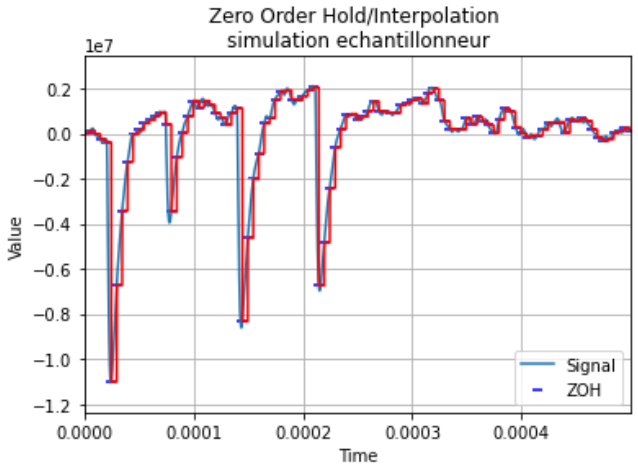


Maintien d'ordre zero

Codage

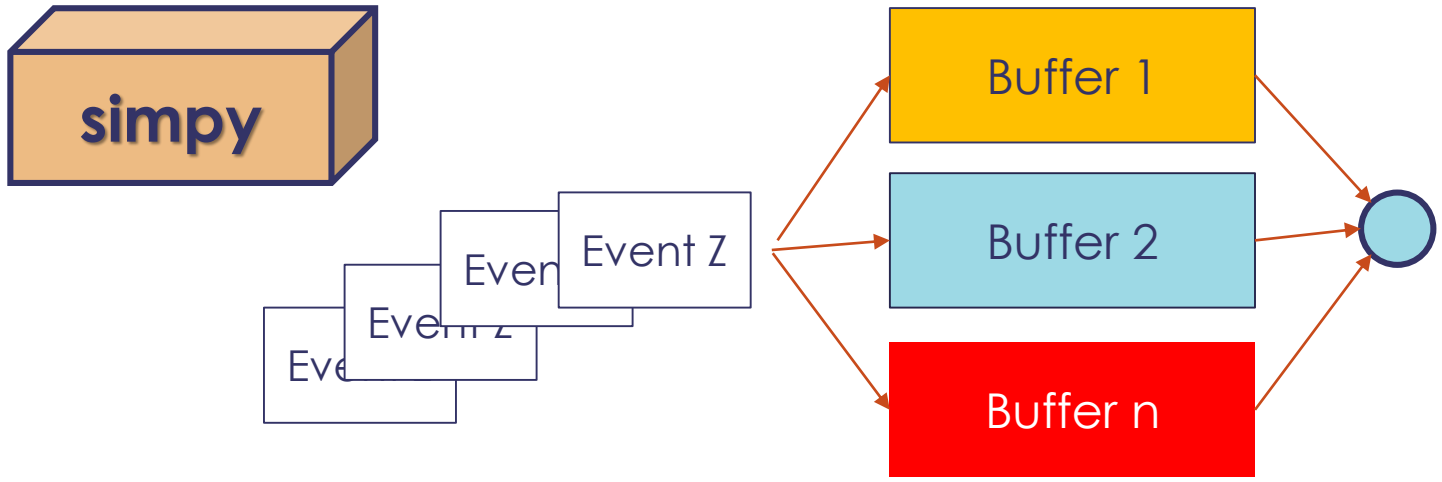
ADC

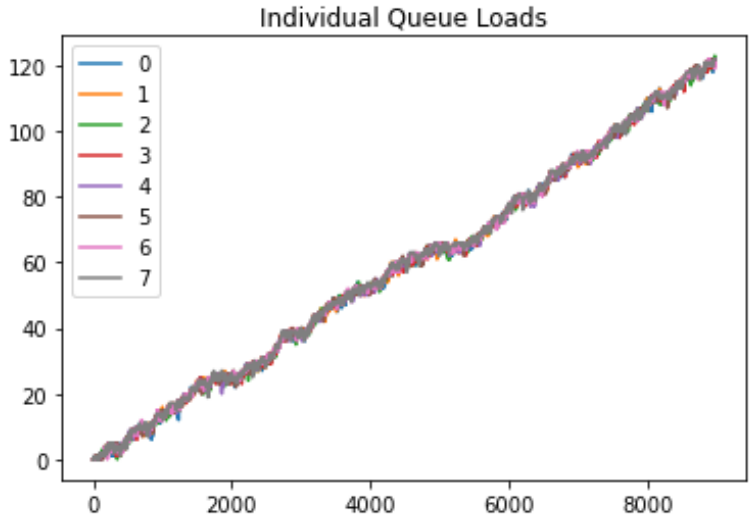
Fres ech = 2MHz

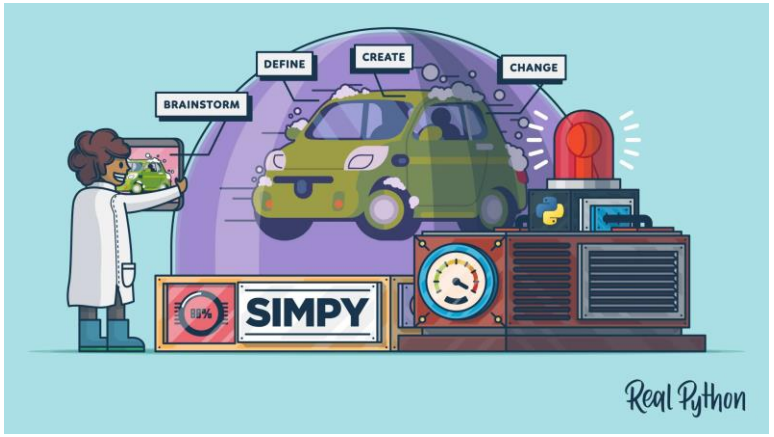


Fres ech = 200kHz

SimCAEIN : Modelisation chaine electronique (python)

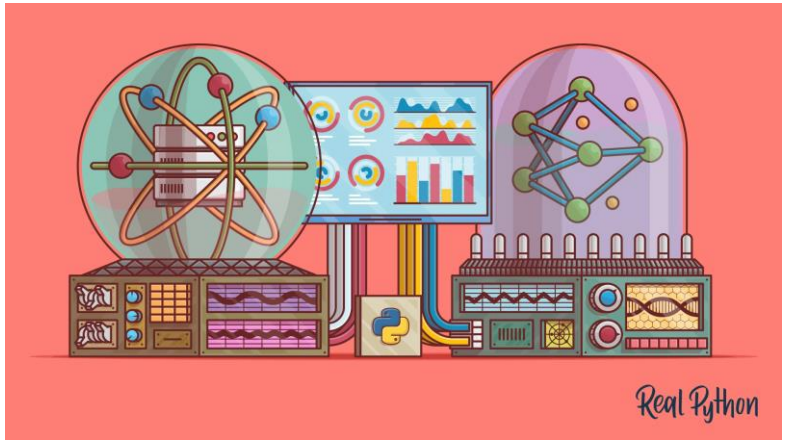






Modélisation et simulation de systèmes connectés

A tester sur une étude de temps mort



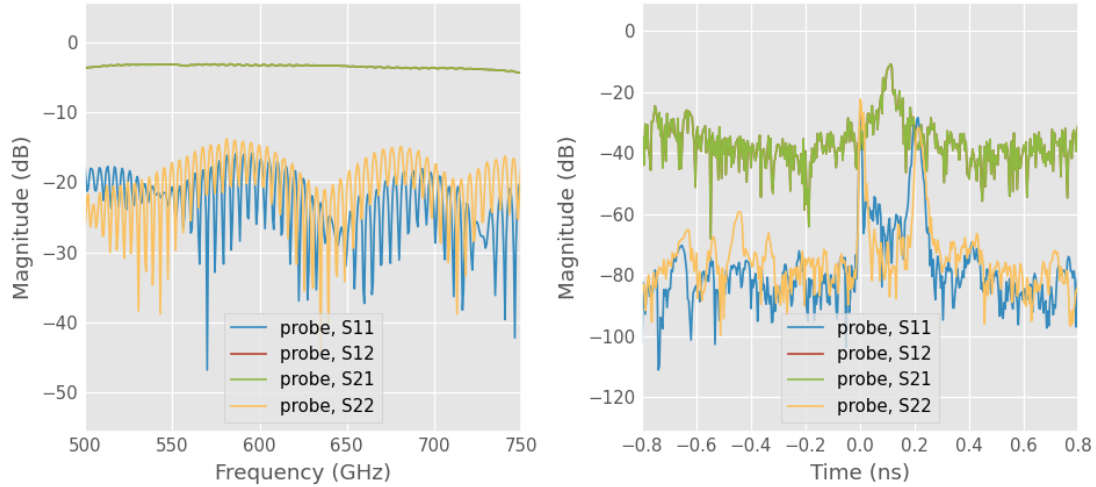
SimCAEIN : Modelisation chaine electronique (python)

Scikit-rf

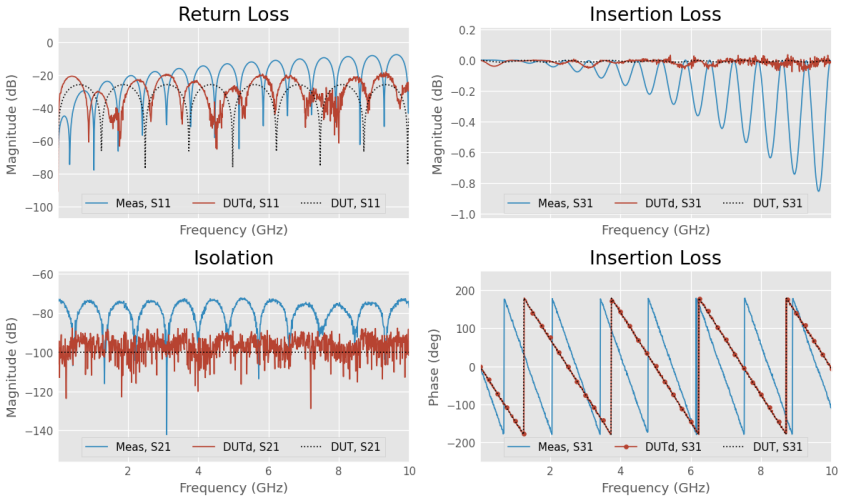
scikit-rf

Object Oriented RF Engineering

```python
from pylab import *
import skrf as rf
import matplotlib.pyplot as plt
rf.stylely()

# load data for the waveguide to CPW probe
probe = rf.Network('probe.s2p')

# we will focus on s11
s11 = probe.s11

#  time-gate the first largest reflection
s11_gated = s11.time_gate(center=0, span=.2, t_unit='ns')
s11_gated.name = 'gated probe'
```




```python
import matplotlib.pyplot as plt
import numpy as np
import skrf as rf
from skrf.media import CPW

rf.stylely()

# base parameters
freq = rf.Frequency(1e-3, 10, 1001, 'ghz')
cpw = CPW(freq, w=0.6e-3, s=0.25e-3, ep_r=10.6)

"""
          l1
    0----+-------2
         |
         = c1
         |
        GND

          l1
    1----+-------3
         |
         = c1
         |
        GND
"""

l1 = cpw.line(20, 'mm', z0=50, embed=True)
c1 = cpw.shunt_capacitor(C=0.15e-12, z0=50)
li = rf.connect(c1, 1, l1, 0)
li = rf.concat_ports([l1, l1], port_order='second')
Fix = li
Fix.name = 'Fix'
Fix.nudge(1e-4)
Left = Fix
# flip fixture for right side
Right = Fix.flipped()

"""
       l2
   0-------2
       l2
   1-------3
"""
l2 = cpw.line(50, 'mm', z0=50, embed=True)
DUT = rf.concat_ports([l2, l2], port_order='second')
DUT.name = 'DUT'
DUT.nudge(1e-5)

"""
          Left        Meas         Right
           l1          l2           l1
    0----+-------2 0-------2 0-------+----2
         |                             |
         = c1                          = c1
         |                             |
        GND                           GND
```



Conclusion

Fully Firmware = spatial accelerator

- Méthodologie sur FPGA OK
- Développer directement en VHDL pour optimiser les reseaux (Resource/latence)
- Utiliser l'apprentissage par renforcement (Engineering model)

Edge Computing

- Matériels OK (CPU, GPU, FPGA (SoC)
- Compiler son propre modèle de réseaux – à tester –
- Utiliser les nvx circuits VERSAL

THINK project (2024-2027) ➔ Welcome

- Formation
- Code framework en VHDL
- Applications au LHC
- Application aux Evenements Rares
- Apprentissage par renforcement

Rejoindre le réseau DAQ:
-Matériel
-Firmware
-Logiciels DAQ