Les activités Calcul et Données de l'institut

Les activités au sein des collaborations scientifiques

- dans toutes les expériences il y a besoin de sélectionner, enregistrer, référencer, transférer, stocker, reconstruire, réduire, analyser, publier les données
  - ➜ activités autour des infrastructures (stockage, calcul, bases de données, services) pour les développements propres aux besoins de l'expérience
  - ➜ activités autour des logiciels : système de déclenchement, reconstruction, analyse, visualisation
  - ➜ utilisation de techniques et technologies avancées : apprentissage automatique, HPC, HTC haut débit, CPU, GPU, FPGA…

➜ projets scientifiques

Les activités transverses aux expériences

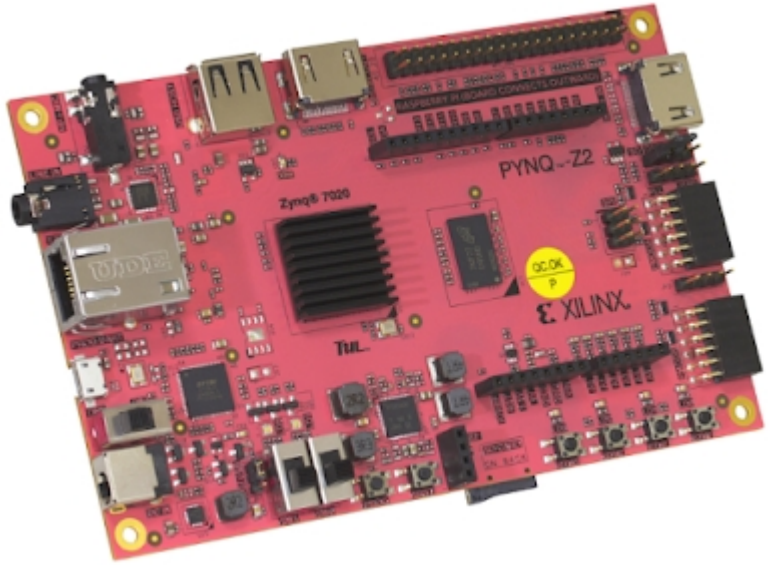
- les infrastructures : CC-IN2P3 et plateformes calcul et données
- les supports au CC et dans les labos
- les projets liés aux infrastructures
- les logiciels communs
- la R&D et la recherche
  - nouvelles techniques et technologies
  - la construction de l'EOSC

➜ projets calcul et données

FPGA pour le calcul : Xilinx PYNQ-Z2

02.04.2024

https://indico.in2p3.fr/event/32783/



Références :

[1] http://www.pynq.io/board.html

[2] https://www.tul.com.tw/images/PYNQ-Z2_PA_v2_pp_20201209_STD.pdf

[3] http://www.aiotlab.org/teaching/fpga/PYNQ%20Introduction.pdf

[4] https://www.youtube.com/watch?v=RiFbRf6gaK4

et

https://www.mouser.fr/new/dfrobot/dfrobot-pynqz2-dev-board/#Video-4

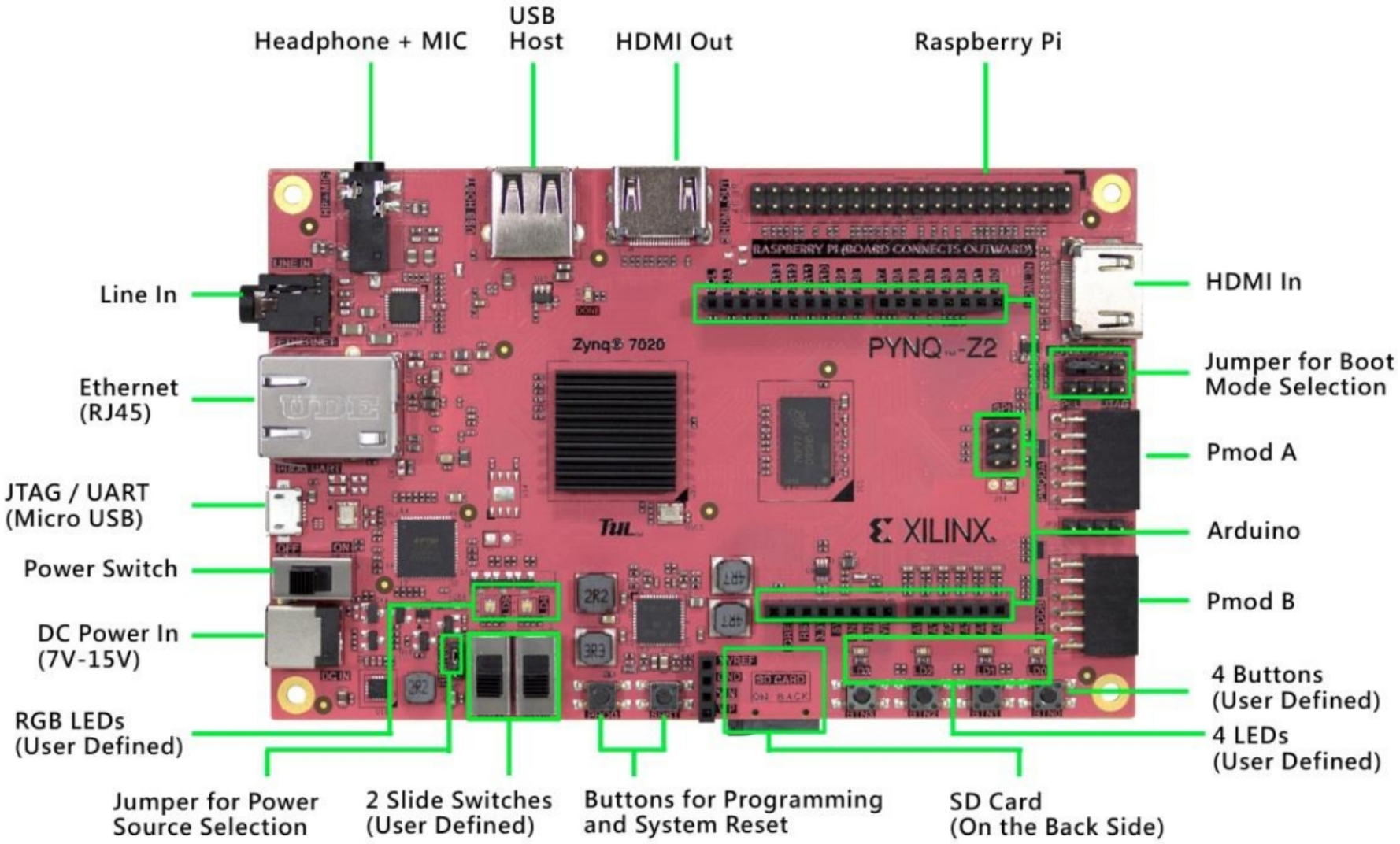
[5] https://www.mouser.fr/ProductDetail/DFRobot/DFR0600?qs=17u8i%2FzlE8%252BCAK0pRwKt0g%3D%3D

133.68 EUR (mars 2022) ⇒ 223.67 EUR (mars 2024)

[6] https://github.com/Xilinx/PYNQ

[7] https://github.com/Xilinx/PYNQ_Workshop?tab=readme-ov-file

PYNQ-Z2 SoC (System on Chip = ARM A9 + FPGA)



ZYNQ XC7Z020-1CLG400C

- 650MHz dual-core Cortex-A9 processor
- DDR3 memory controller with 8 DMA channels and 4 High Performance AXI3 Slave ports
- High-bandwidth peripheral controllers: 1G Ethernet, USB 2.0, SDIO
- Low-bandwidth peripheral controller: SPI, UART, CAN, I2C
- Programmable from JTAG, Quad-SPI flash, and microSD card
- Programmable logic equivalent to Artix-7 FPGA
  - 13,300 logic slices, each with four 6-input LUTs and 8 flip-flops
  - 630 KB of fast block RAM
  - 4 clock management tiles, each with a phase-locked loop (PLL) and mixed-mode clock manager (MMCM)
  - 220 DSP slices
  - On-chip analog-to-digital converter (XADC)

Memory

- 512MB DDR3 with 16-bit bus @ 1050Mbps
- 16MB Quad-SPI Flash with factory programmed 48-bit globally unique EUI-48/64™ compatible identifier
- microSD slot

Power

- Powered from USB or 7V-15V external power source

USB and Ethernet

- Gigabit Ethernet PHY
- Micro USB-JTAG Programming circuitry
- Micro USB-UART bridge
- USB 2.0 OTG PHY (supports host only)

Audio and Video

- HDMI sink port (input)
- HDMI source port (output)
- I2S interface with 24bit DAC with 3.5mm TRRS jack
- Line-in with 3.5mm jack

Switches, Push-buttons and LEDs

- 4 push-buttons
- 2 slide switches
- 4 LEDs
- 2 RGB LEDs

Expansion Connectors

- Two standard Pmod ports
  - 16 Total FPGA I/O (8 shared pins with Raspberry Pi connector)
- Arduino Shield connector
  - 24 Total FPGA I/O
  - 6 Single-ended 0-3.3V Analog inputs to XADC
- Raspberry Pi connector
  - 28 Total FPGA I/O (8 shared pins with Pmod A port)

Carte microSD + http://www.pynq.io/board.html pour image Linux

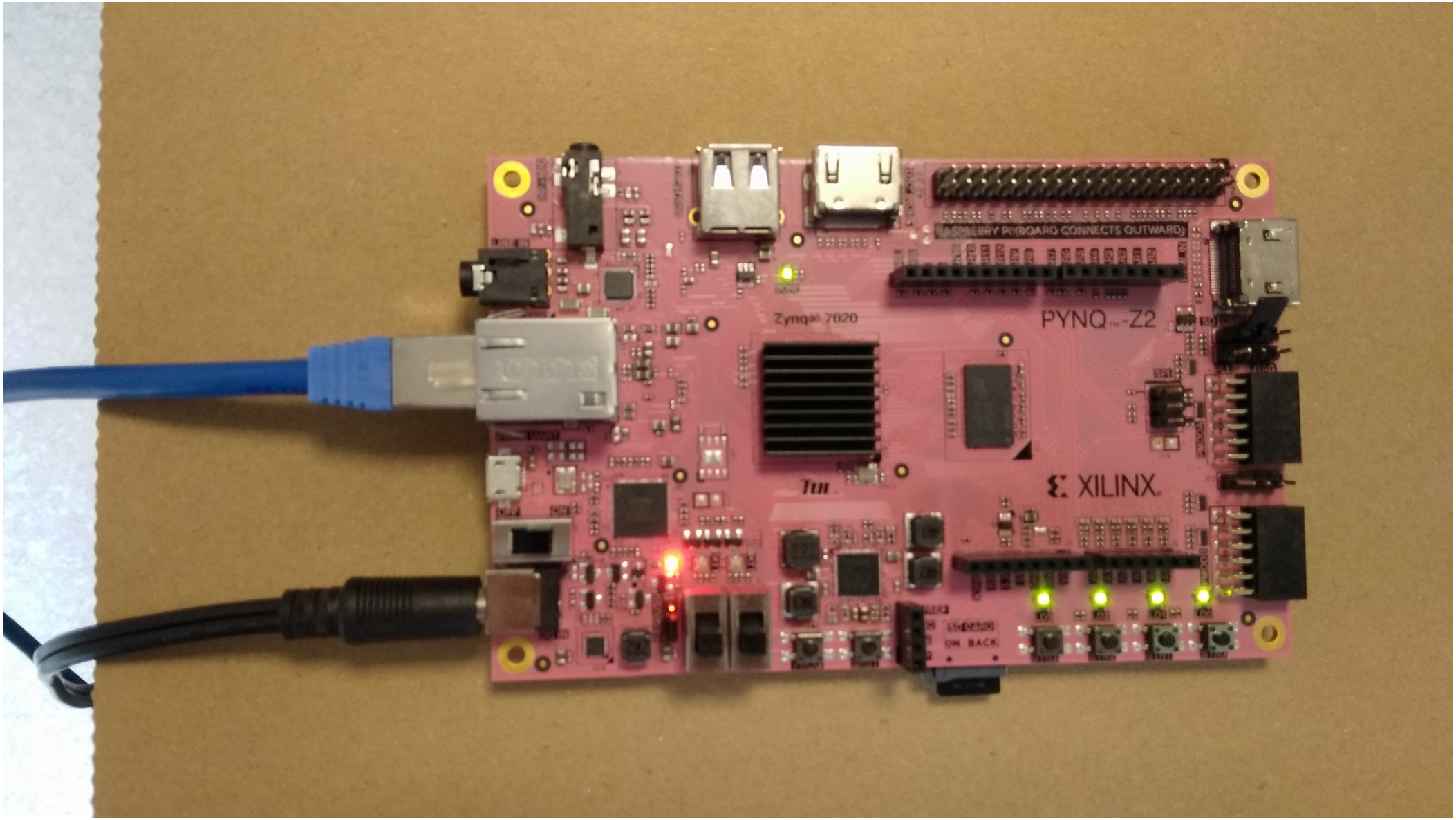
Downloadable PYNQ images

If you have a Zynq board, you need a PYNQ SD card image to get started. You can download a pre-compiled PYNQ image from the table below. If an image is not available for your board, you can build your own SD card image (see details below).

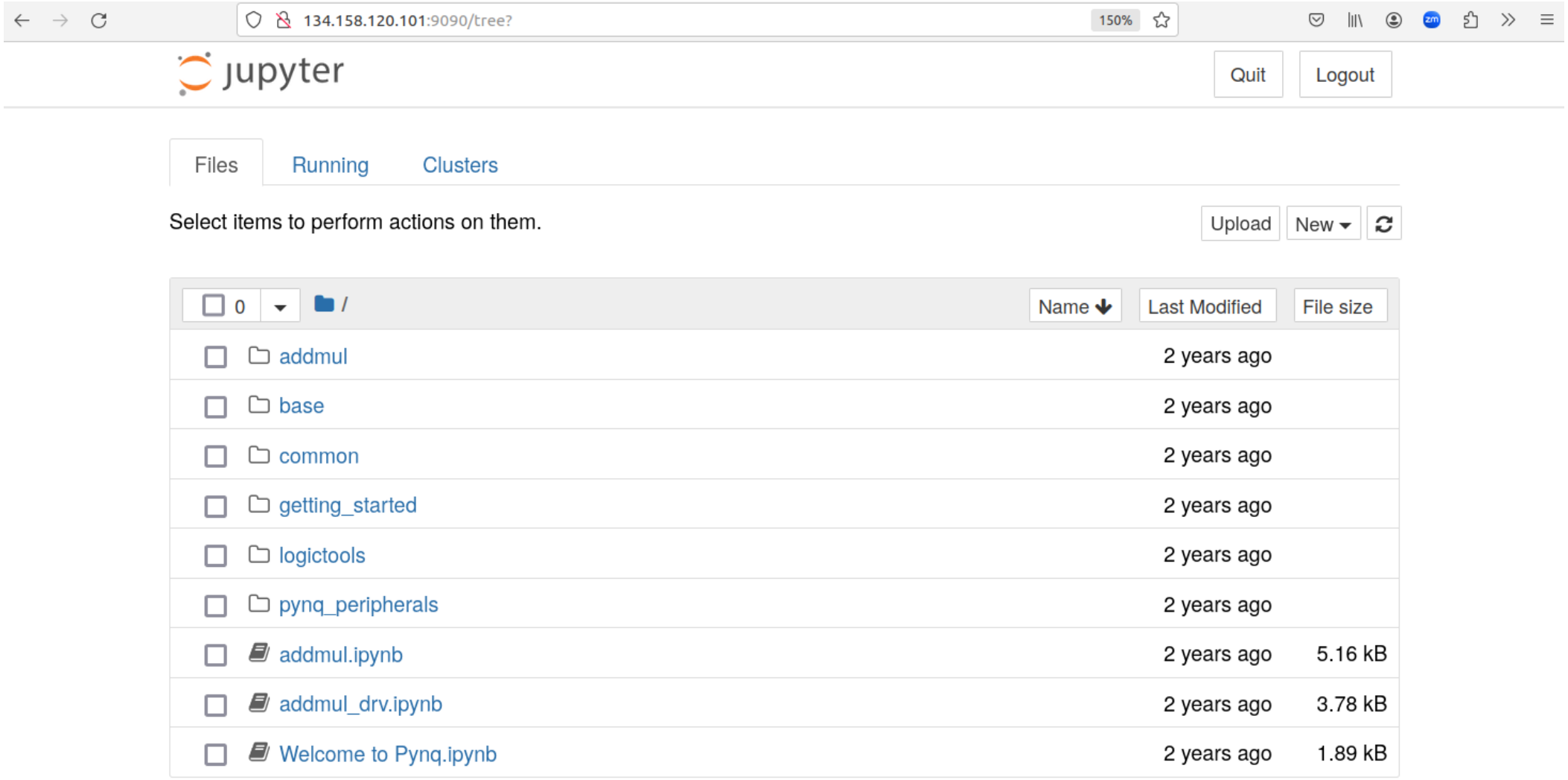
Board	SD card image	Previous versions	Documentation	Board webpage
PYNQ-Z2	v3.0.1	v2.7 v2.6	PYNQ setup guide	TUL Pynq-Z2
PYNQ-Z1	v3.0.1	v2.7 v2.6	PYNQ setup guide	Digilent Pynq-Z1
PYNQ-ZU	v3.0.1	v2.7 v2.6	GitHub project page	TUL PYNQ-ZU
Kria KV260*	Ubuntu 22.04		Kria PYNQ setup	Xilinx Kria KV260
Kria KR260*	Ubuntu 22.04		Kria PYNQ setup	Xilinx Kria KR260
ZCU104	v3.0.1	v2.7 v2.6	PYNQ setup guide	Xilinx ZCU104
RFSoC 2x2	v3.0.1	v2.7 v2.6	RFSoC-PYNQ	XUP RFSoC 2x2
RFSoC 4x2	v3.0.1	v2.7	RFSoC-PYNQ	XUP RFSoC 4x2
ZCU111	v3.0.1	v2.7 v2.6	RFSoC-PYNQ	Xilinx ZCU111
ZCU208	v3.0.1		RFSoC-PYNQ	Xilinx ZCU208
Ultra96V2	v3.0.1	v2.7 v2.6	Avnet PYNQ webpage	Avnet Ultra96V2
Ultra96 (legacy)	v3.0.1	v2.7 v2.6	See Ultra96V2	See Ultra96V2
ZUBoard 1CG	v3.0.1		GitHub project page	Avnet ZUBoard 1CG
TySOM-3-ZU7EV	v3.0.1	v2.7	GitHub project page	Aldec TySOM-3-ZU7EV
TySOM-3A-ZU19EG	v3.0.1	v2.7	GitHub project page	Aldec TySOM-3A-ZU19EG

*For the Kria KV260 and KR260, follow the links above for guide for getting started with the Ubuntu image, and then follow the Kria PYNQ setup instructions to install PYNQ.

Bureau 6121a



Connexion avec un navigateur via le IP de la carte (serveur Jupyter)



(accès possible aussi en tant que dispositif partagé vie Samba)

Ouvrir un terminal (en mode root)

OS = Ubuntu-like (20.04)

```
root@pynq:/home/xilinx/jupyter_notebooks# cat /etc/os-release
NAME="PynqLinux"
VERSION="2.7 (Austin)"
ID=pynqlinux
ID_LIKE=ubuntu
PRETTY_NAME="PynqLinux, based on Ubuntu 20.04"
VERSION_ID="2.7"
HOME_URL="https://www.pynq.io/"
SUPPORT_URL="https://discuss.pynq.io/"
BUG_REPORT_URL="https://www.pynq.io"
PRIVACY_POLICY_URL="https://www.pynq.io"
VERSION_CODENAME=Austin
UBUNTU_CODENAME=focal
root@pynq:/home/xilinx/jupyter_notebooks#
```

jupyter

Logout

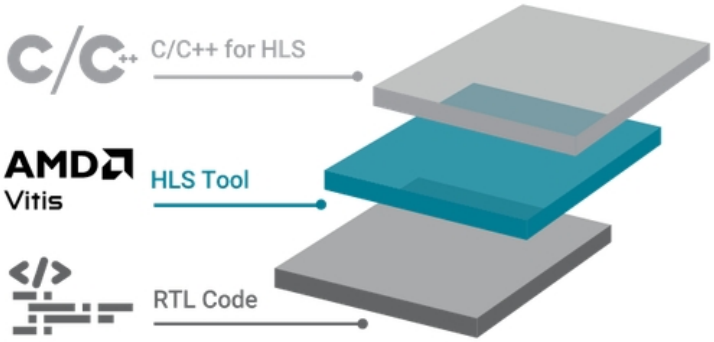
```
root@pynq:/# lscpu
Architecture:          armv7l
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    2
Socket(s):             1
Vendor ID:             ARM
Model:                 0
Model name:            Cortex-A9
Stepping:              r3p0
BogoMIPS:              650.00
Flags:                 half thumb fastmult vfp edsp neon vfpv3 tls vfpd32
root@pynq:/#
```

Synthèse de haut niveau (HLS, High Level Synthesis) avec Vitis HLS

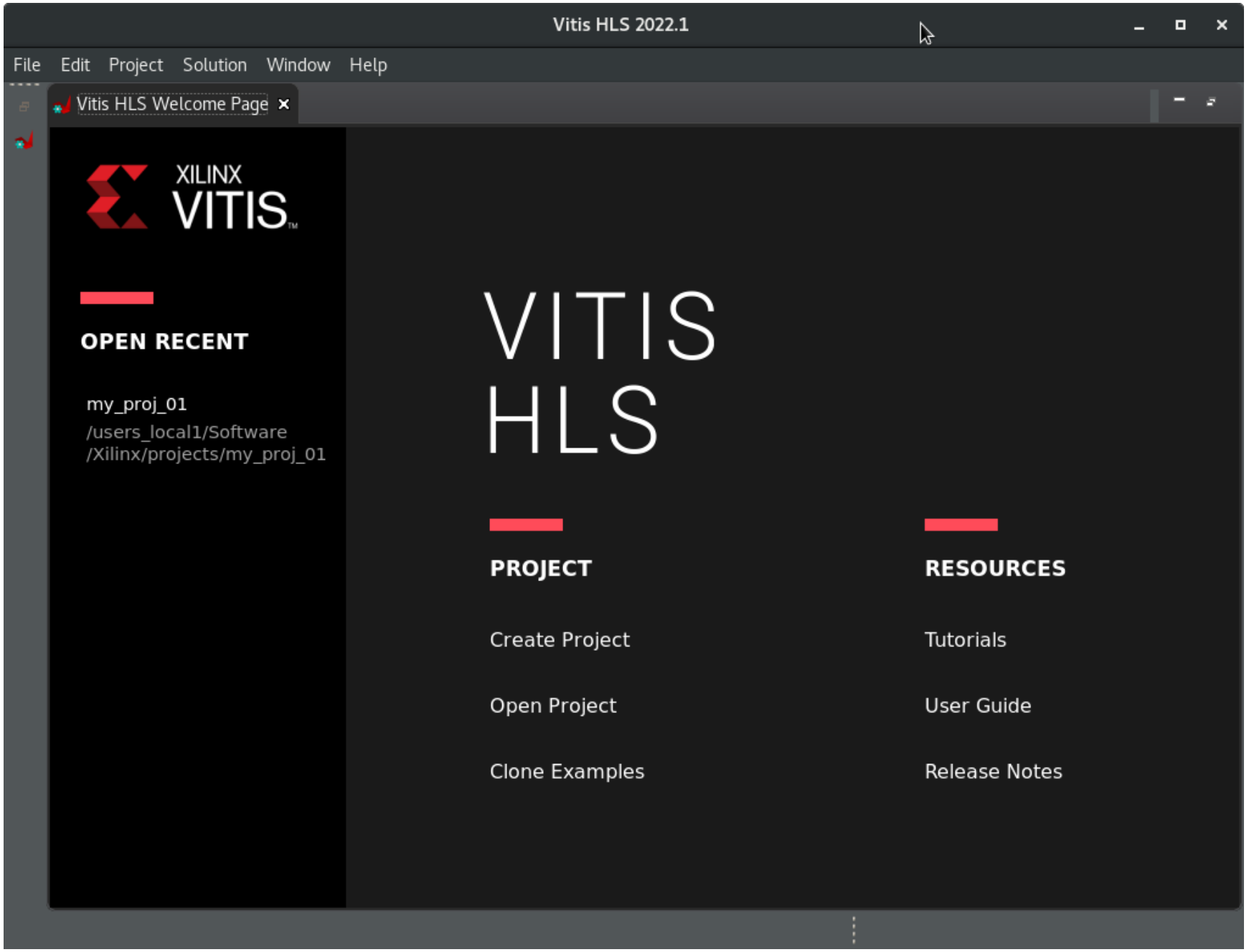
https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html

The Vitis™ HLS tool allows users to easily create complex FPGA algorithms by synthesizing a C/C++ function into RTL. The Vitis HLS tool is tightly integrated with both the Vivado™ Design Suite for synthesis and place & route and the Vitis™ unified software platform for heterogenous system designs and applications.

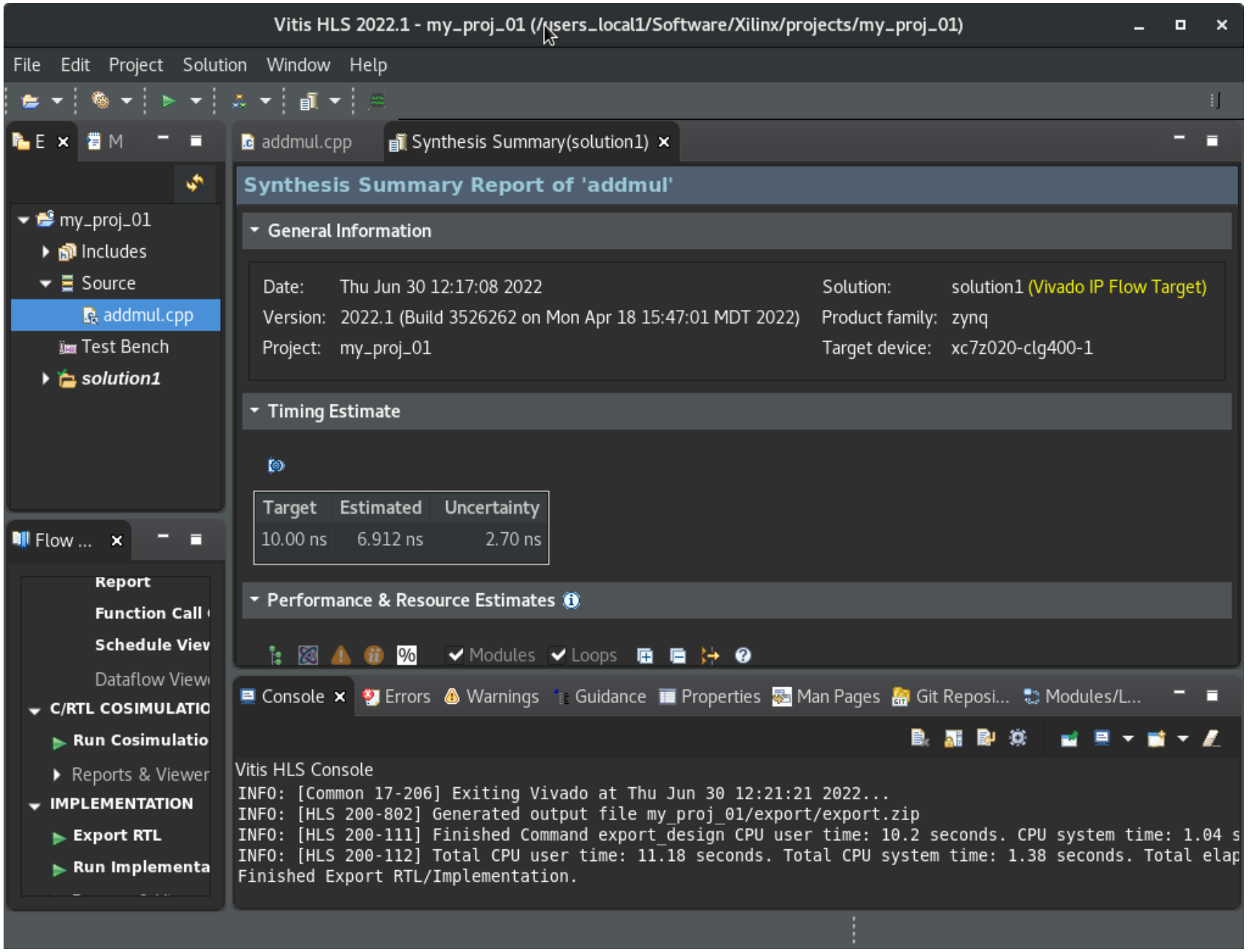
- Using the Vitis HLS flow, users can apply directives to the C code to create the RTL specific to a desired implementation.
- Multiple design architectures can be created from the C source code and a path for high-quality, correct-by-construction RTL is enabled.
- C simulation can be used to validate the design and allows faster iterations than a traditional RTL-based simulation.
- The Vitis HLS tool features a rich set of analysis and debugging tools that facilitate design optimization.



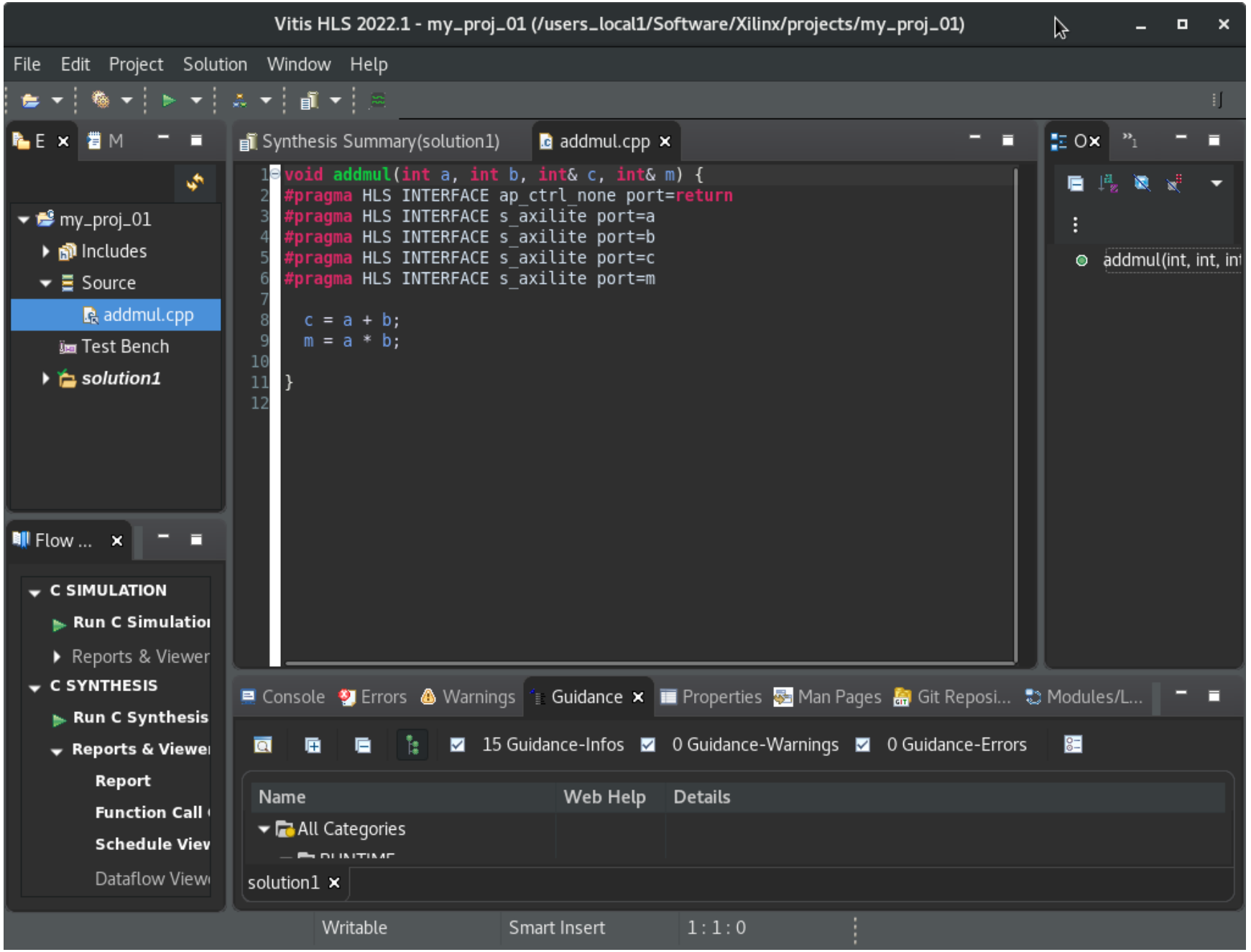
Vitis HLS 2022.1 sur clralicepc08



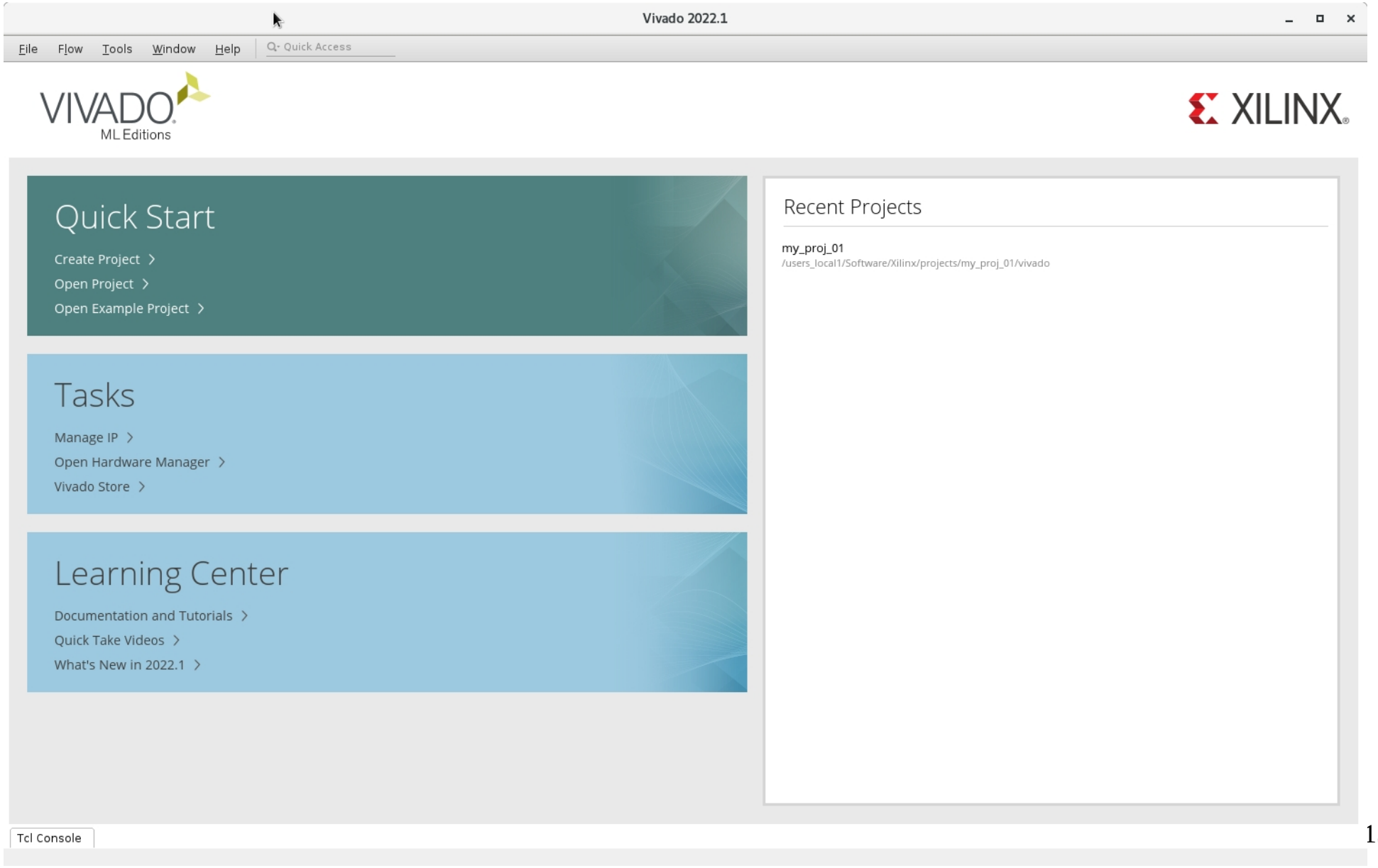
Un projet de multiplication de deux nombres



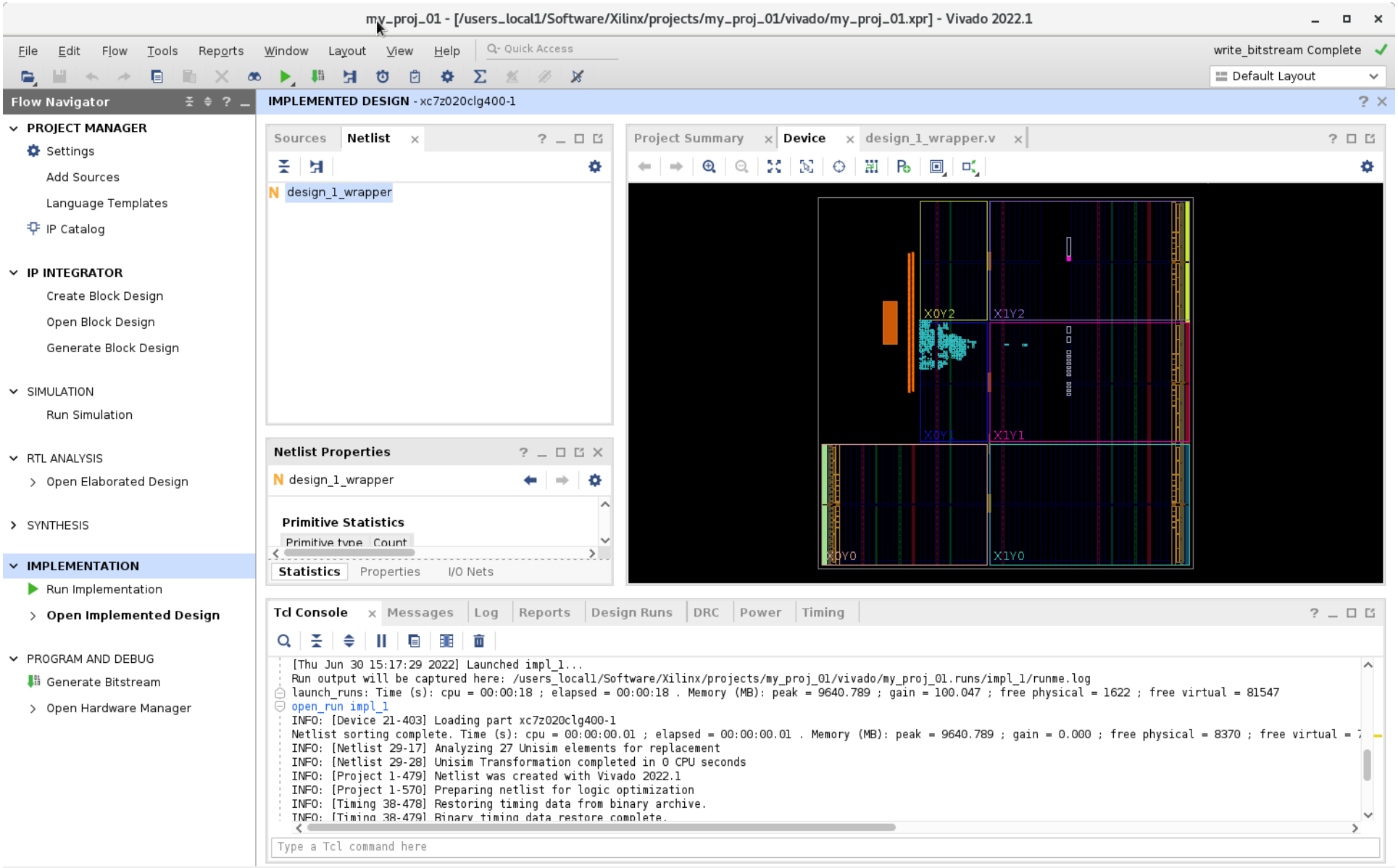
Le fichier source C++



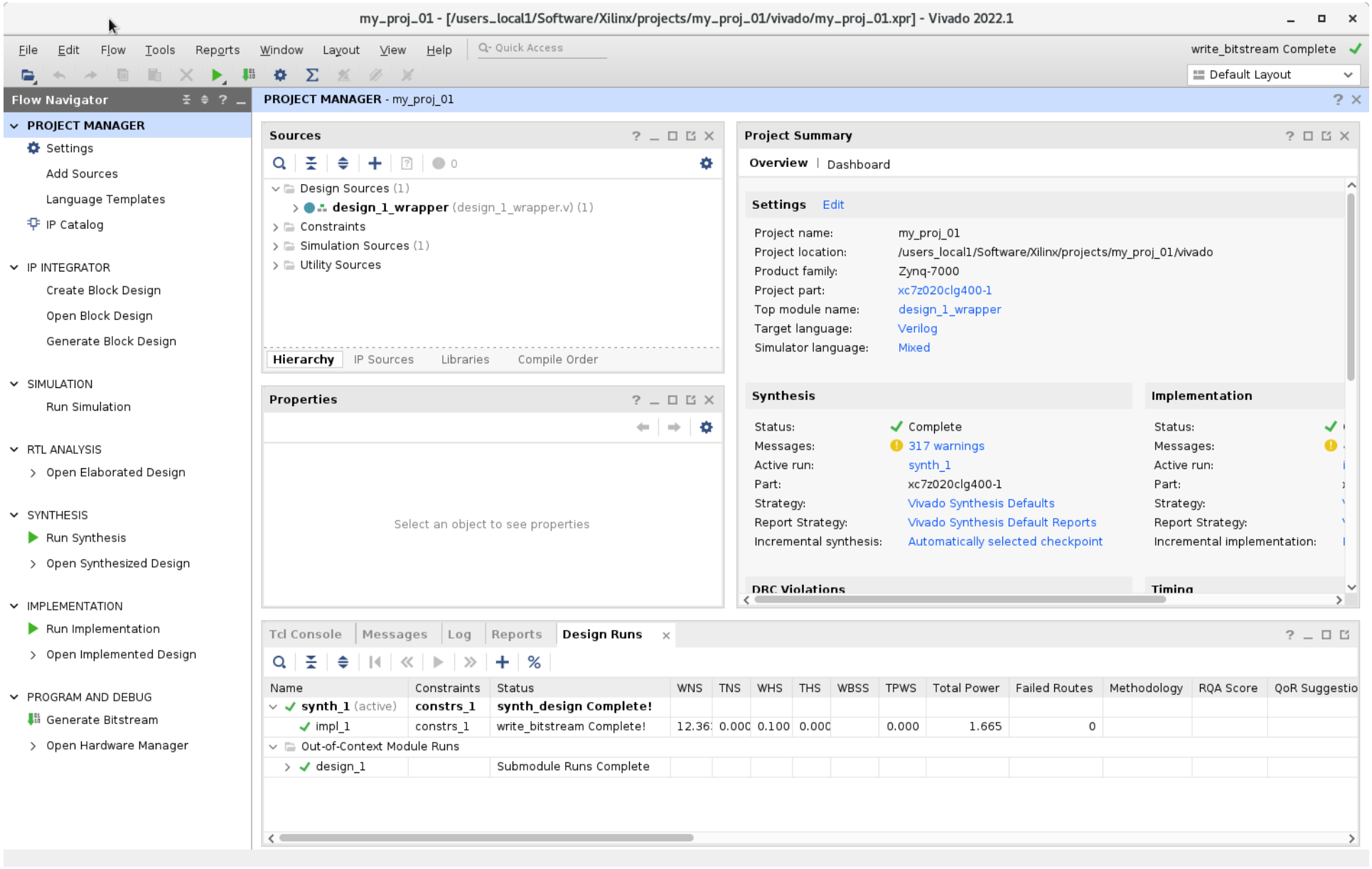
Après Vitis HLS ⇒ Vivado 2022.1 on clralicepc08



Le projet crée avec Vitis HLS

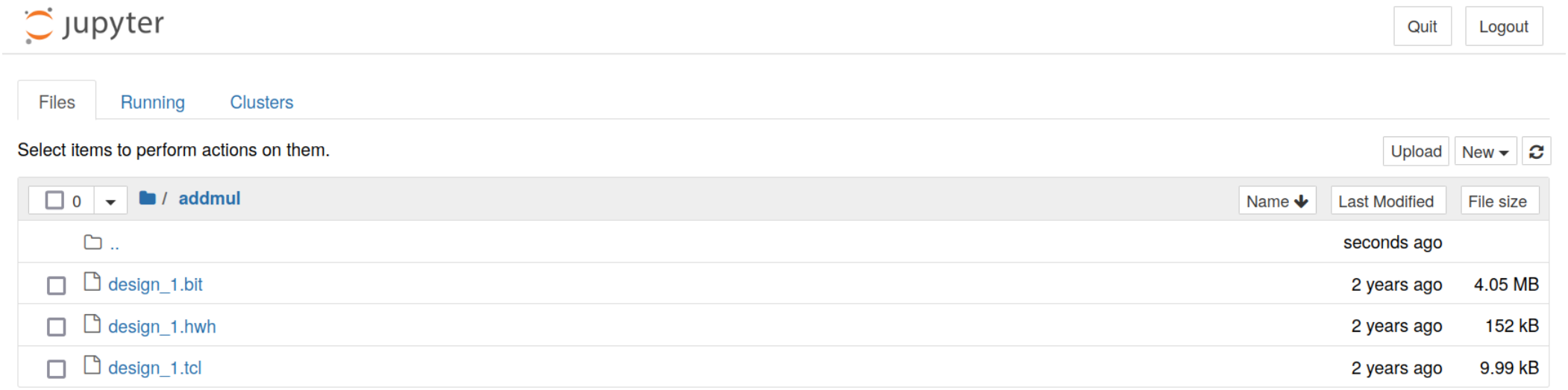


La génération du bitstream four le FPGA



Copier les fichiers résultant de la synthèse Vivado sur la carte PYNQ-Z2

`design_1.bit, design_1.hwh, design_1.tcl` (via le "Upload" du serveur Jupyter)



Addition et multiplication : écriture / lecture des registres

In [1]:
```
from pynq import Overlay
```

In [2]:
```
overlay = Overlay('/home/xilinx/jupyter_notebooks/addmul/design_1.bit')
```

In [3]:
```
overlay?
```

In [4]:
```
# see from previous output the name of the IP block "addmul_0"
add_ip = overlay.addmul_0
```

In [5]:
```
add_ip.register_map
```

Out[5]:
```
RegisterMap {
  a = Register(a=write-only),
  b = Register(b=write-only),
  c = Register(c=0),
  c_ctrl = Register(c_ap_vld=1, RESERVED=0),
  m = Register(m=0),
  m_ctrl = Register(m_ap_vld=1, RESERVED=0)
}
```

In [6]:
```
add_ip?
```

In [7]:
```
add_ip.register_map.a = 3
add_ip.register_map.b = 4
add_ip.register_map.c
```

Out[7]:
```
Register(c=7)
```

In [8]:
```
# alternatively, see the addresses definitions in xaddmul_hw.h
add_ip.write(0x10, 4)
add_ip.write(0x18, 5)
add_ip.read(0x20)
```

Out[8]:
```
9
```

In [9]:
```
add_ip.read(0x30)
```

Out[9]:
```
20
```

In [ ]:
```

```

Addition et multiplication : avec création d'un pilote (driver)

Creating a driver

In [1]:
```
from pynq import DefaultIP

class AddDriver(DefaultIP):
    def __init__(self, description):
        super().__init__(description=description)

    bindto = ['xilinx.com:hls:addmul:1.0']

    def add(self, a, b):
        self.write(0x10, a)
        self.write(0x18, b)
        return self.read(0x20)

    def mul(self, a, b):
        self.write(0x10, a)
        self.write(0x18, b)
        return self.read(0x30)
```

In [2]:
```
from pynq import Overlay
overlay = Overlay('/home/xilinx/jupyter_notebooks/addmul/design_1.bit')
```

In [3]:
```
overlay?
```

In [4]:
```
overlay.addmul_0.add(15,20)
```

Out[4]: 35

In [5]:
```
overlay.addmul_0.mul(15,20)
```

Out[5]: 300

In [ ]:

Overlay = un autre nom pour la bibliothèque hardware

La synthèse Vitis HLS :

`xaddmul_hw.h` ⇒

(addition et multiplication sur 32 bits)

`addmul.cpp`

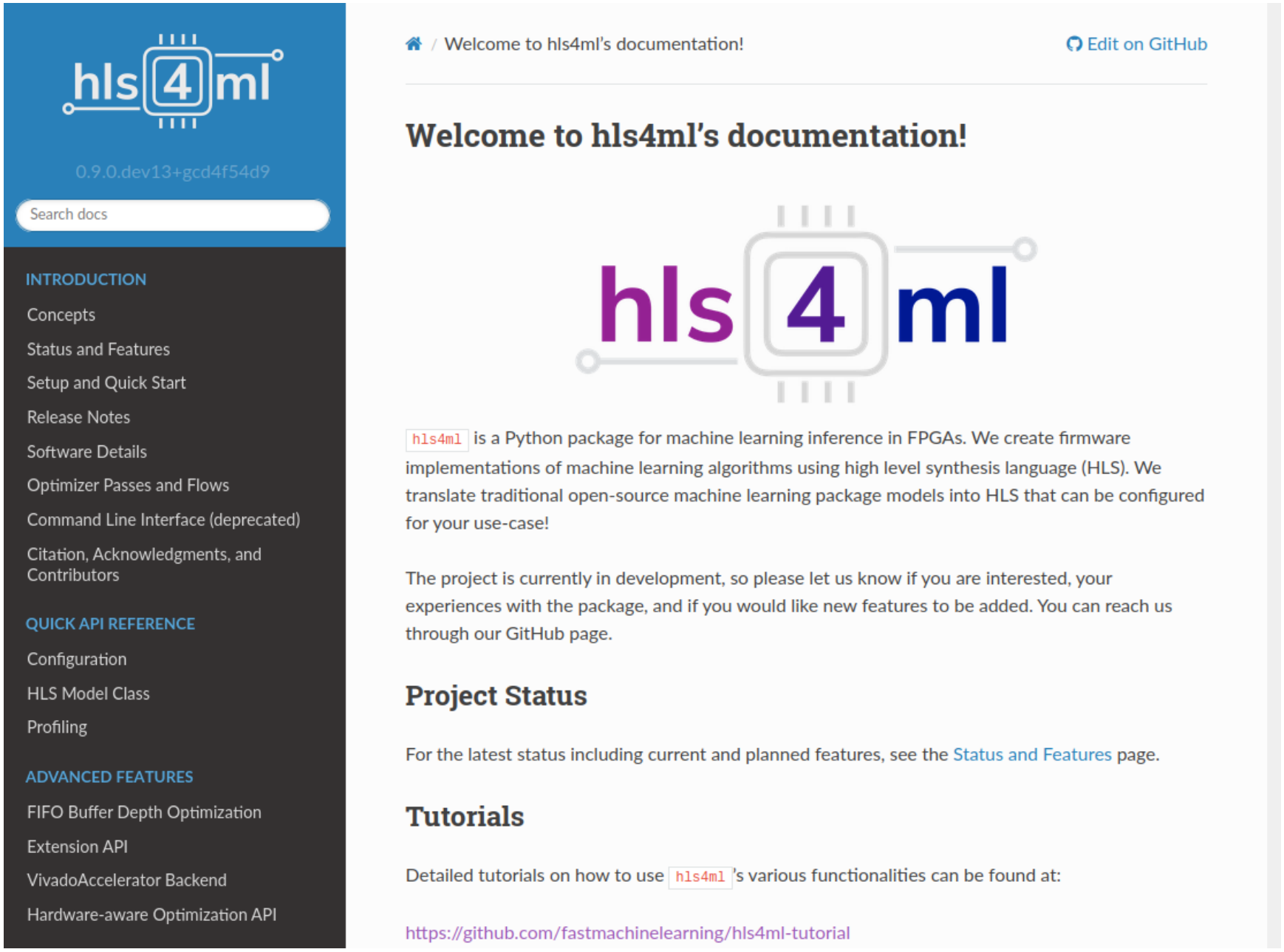
⇒

```
void addmul(
int a,
int b,
int& c,
int& m){ ...
```

```
// ==========================================================
// Vitis HLS - High-Level Synthesis from C, C++ and OpenCL v2022.1 (64-bit)
// Tool Version Limit: 2022.04
// Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
// ==========================================================
// control
// 0x00 : reserved
// 0x04 : reserved
// 0x08 : reserved
// 0x0c : reserved
// 0x10 : Data signal of a
//        bit 31~0 - a[31:0] (Read/Write)
// 0x14 : reserved
// 0x18 : Data signal of b
//        bit 31~0 - b[31:0] (Read/Write)
// 0x1c : reserved
// 0x20 : Data signal of c
//        bit 31~0 - c[31:0] (Read)
// 0x24 : Control signal of c
//        bit 0  - c_ap_vld (Read/COR)
//        others - reserved
// 0x30 : Data signal of m
//        bit 31~0 - m[31:0] (Read)
// 0x34 : Control signal of m
//        bit 0  - m_ap_vld (Read/COR)
//        others - reserved
// (SC = Self Clear, COR = Clear on Read, TOW = Toggle on Write, COH = Clear on Handshake)

#define XADDMUL_CONTROL_ADDR_A_DATA 0x10
#define XADDMUL_CONTROL_BITS_A_DATA 32
#define XADDMUL_CONTROL_ADDR_B_DATA 0x18
#define XADDMUL_CONTROL_BITS_B_DATA 32
#define XADDMUL_CONTROL_ADDR_C_DATA 0x20
#define XADDMUL_CONTROL_BITS_C_DATA 32
#define XADDMUL_CONTROL_ADDR_C_CTRL 0x24
#define XADDMUL_CONTROL_ADDR_M_DATA 0x30
#define XADDMUL_CONTROL_BITS_M_DATA 32
#define XADDMUL_CONTROL_ADDR_M_CTRL 0x34
```

1) High Level Synthesis for Machine Learning, https://fastmachinelearning.org/hls4ml/



Fast inference of deep neural networks in FPGAs for particle physics
Duarte et al. (2018), https://arxiv.org/pdf/1804.06913.pdf

Publications HLS4ML

Fast inference of deep neural networks in FPGAs for particle physics

Duarte et al. (FNAL, MIT, CERN, ...)

https://arxiv.org/abs/1804.06913.pdf

Fast convolutional neural networks on FPGAs with HLS4ML

Aarrestad et al.

https://arxiv.org/abs/2101.05108.pdf

Real-time semantic segmentation on FPGAs for autonomous vehicles with HLS4ML

Ghielmetti et al.

https://arxiv.org/abs/2205.07690.pdf

Compressing deep neural networks on FPGAs to binary and ternary precision with HLS4ML

Di Guglielmo et al.

https://arxiv.org/abs/2003.06308.pdf

https://indico.cern.ch/event/1389765/

Detector Seminar

Fast Machine Learning at the Edge for HEP Experiments

by Sioni Summers (CERN)

Friday 8 Mar 2024, 11:00 → 12:15 Europe/Zurich

40/S2-D01 - Salle Dirac (CERN)

Description Facing increasing data challenges at current and future HEP experiments, there is a growing demand for fast and efficient Machine Learning (ML) techniques deployed close to the detector. In this seminar, we explore the application of Fast ML in accelerating and optimizing ML algorithms for real-time analysis, triggering, and data reduction, focussing on two projects in use at CERN experiments: hls4ml and conifer. These tools offer a novel approach transforming ML models into efficient hardware descriptions suitable for implementation on field-programmable gate arrays (FPGAs) and other hardware accelerators. We present the state of the art techniques for compressing ML models for fast and lightweight inference. We highlight case studies and experimental results that have leveraged these tools and techniques to improve performance such as particle identification, event classification, and anomaly detection. Finally, we present key future development directions.

Coffee will be served at 10:30.

Recording | sps_ds_8-3-24.pdf | Video preview

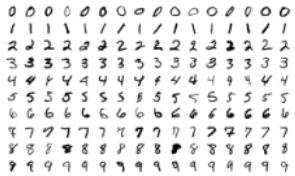
Organised by Michael Campbell

Webcast There is a live webcast for this event — Watch

Contact ep-seminars.colloquia@cern.ch

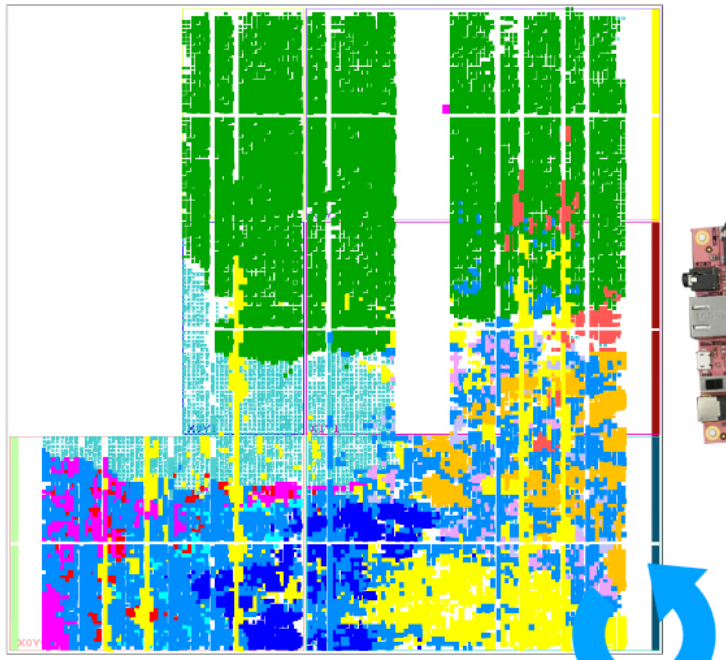
hls4ml - Dataflow Architecture

- Dataflow architecture: each layer is an independent compute unit
  - With tunable parallelism and quantization
- Fully on-chip: NN must fit within available FPGA resources (pynq-z2 floorplan shown)
  - Example: small CNN trained on MNIST

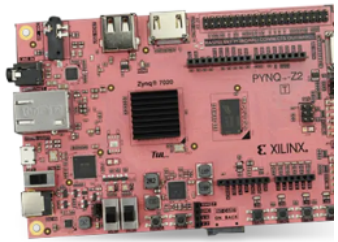


Conv2D
ReLU
MaxPool2D
Conv2D
ReLU
MaxPool2D
Flatten
Dense
Softmax

Prediction



FIFOs

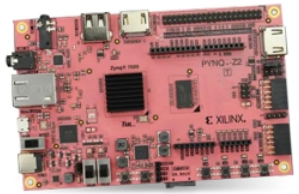


https://indico.cern.ch/event/1389765/

TinyML - MLPerf Tiny ™

- MLCommons group organise benchmarks of Machine Learning (MLPerf) - now also for low power devices (MLPerf Tiny)
- 4 benchmark datasets, open/closed division allowing/disallowing model retraining
- **hls4ml** in open category (for Quantisation Aware Training) achieves competitive performance
- In collaboration with AMD / Xilinx Research Labs developers of FINN project

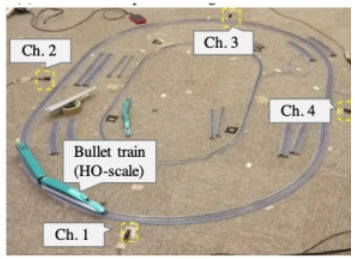
Benchmark		CIFAR-10			ToyADMOS		
Team	Device	Accuracy	Latency (ms)	Energy (uJ)	AUC	Latency (ms)	Energy (uJ)
hls4ml	Pynq-z2[1]	83.5%	7.64	12266	0.83	0.019	30.1
GreenWaves	GAP9 EVK[2]	85%	0.62	40.4	0.85	0.18	7.3
STMicro	Nucleo-L4R5ZI[3]	85%	54.3	8707	0.85	1.82	266.5
OctoML	Nucleo-L4R5ZI[3]	85%	389.2	21342	0.85	11.7	633.7

1 

2 

3 

CIFAR-10

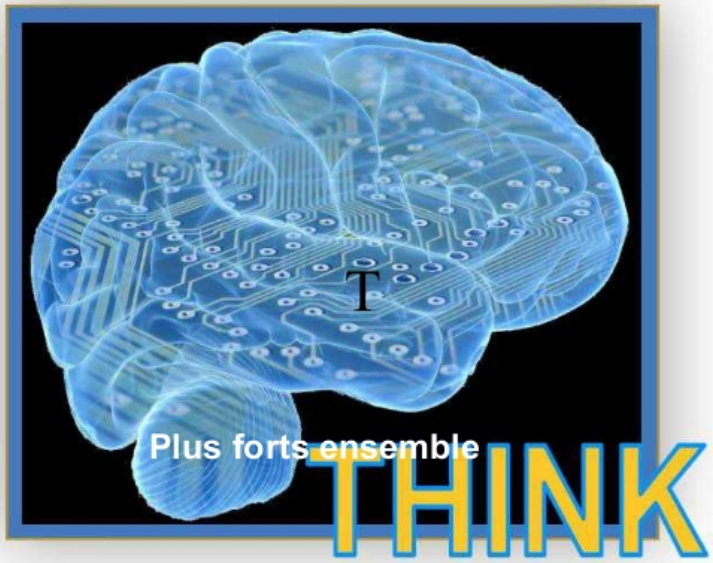


ToyADMOS

https://indico.cern.ch/event/1389765/

Le projet THINK

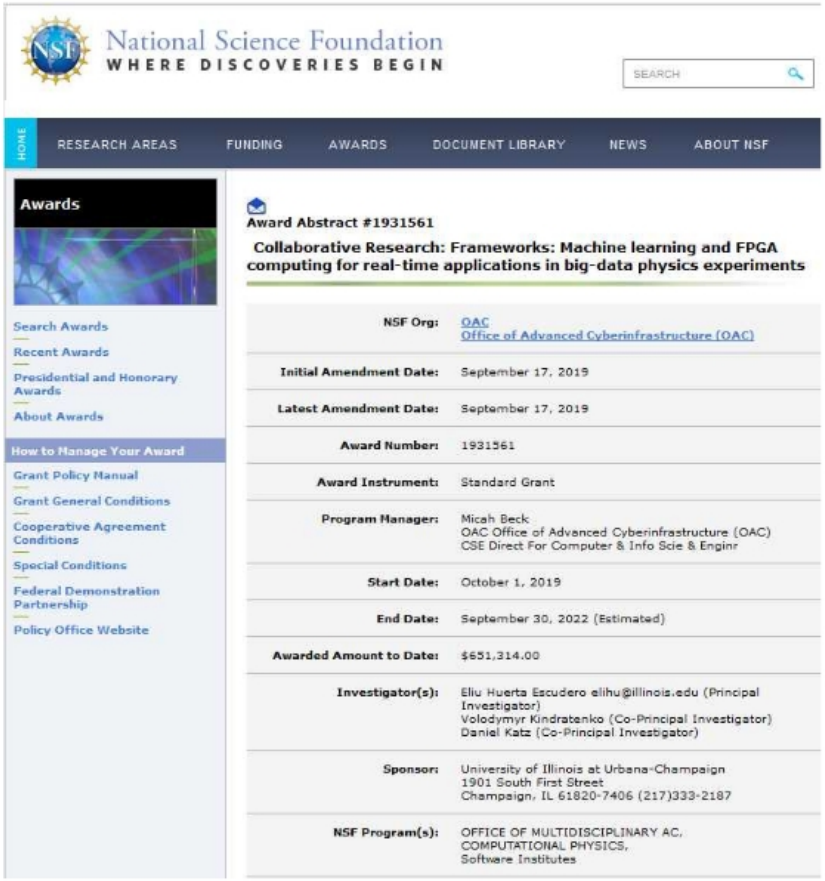
Testing Hardware Instantiations of Neural Kernels



IN2P3
Institut national de physique nucléaire et de physique des particules

J.-P. Cachemiche (CPPM)
V. Gligorov, O. Ledortz (LPNHE)
D. Etasse, J. Hommet (LPC Caen)
F. Bellachia, S. Lafrasse (LAPP
R. Bouet, F. Druillole, A. Rebii (CENBG)
F. Magniette, E. Sauvan (LLR)
G. Aad, Y. Boursier, T. Calvet, E. Fortin, E. Monnier (CPPM)
J. Frontera (CEA IRFU)

Projets similaires et premiers travaux



National Science Foundation
WHERE DISCOVERIES BEGIN

SEARCH

HOME | RESEARCH AREAS | FUNDING | AWARDS | DOCUMENT LIBRARY | NEWS | ABOUT NSF

Awards

Search Awards
Recent Awards
Presidential and Honorary Awards
About Awards

How to Manage Your Award
Grant Policy Manual
Grant General Conditions
Cooperative Agreement Conditions
Special Conditions
Federal Demonstration Partnership
Policy Office Website

Award Abstract #1931561

Collaborative Research: Frameworks: Machine learning and FPGA computing for real-time applications in big-data physics experiments

NSF Org:	OAC Office of Advanced Cyberinfrastructure (OAC)
Initial Amendment Date:	September 17, 2019
Latest Amendment Date:	September 17, 2019
Award Number:	1931561
Award Instrument:	Standard Grant
Program Manager:	Micah Beck OAC Office of Advanced Cyberinfrastructure (OAC) CSE Direct For Computer & Info Scie & Enginr
Start Date:	October 1, 2019
End Date:	September 30, 2022 (Estimated)
Awarded Amount to Date:	$651,314.00
Investigator(s):	Eliu Huerta Escudero elihu@illinois.edu (Principal Investigator) Volodymyr Kindratenko (Co-Principal Investigator) Daniel Katz (Co-Principal Investigator)
Sponsor:	University of Illinois at Urbana-Champaign 1901 South First Street Champaign, IL 61820-7406 (217)333-2187
NSF Program(s):	OFFICE OF MULTIDISCIPLINARY AC, COMPUTATIONAL PHYSICS, Software Institutes

Fast inference of deep neural networks in FPGAs for particle physics

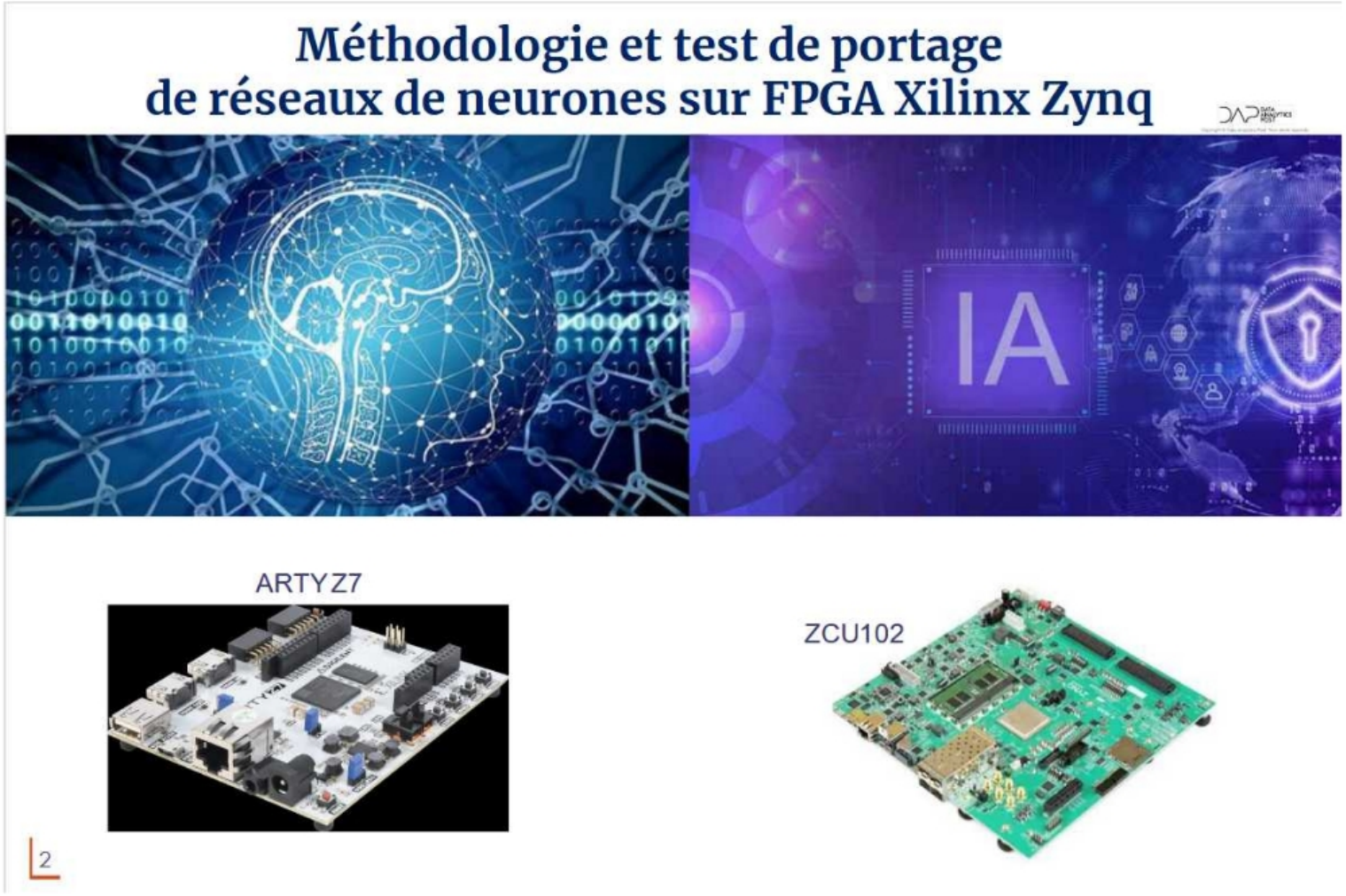
Javier Duarte, Song Han, Philip Harris, Sergo Jindariani, Edward Kreinar, Benjamin Kreis, Jennifer Ngadiuba, Maurizio Pierini, Ryan Rivera, Nhan Tran, Zhenbin Wu

(Submitted on 16 Apr 2018 (v1), last revised 28 Jun 2018 (this version, v3))

Recent results at the Large Hadron Collider (LHC) have pointed to enhanced physics capabilities through the improvement of the real-time event processing techniques. Machine learning methods are ubiquitous and have proven to be very powerful in LHC physics, and particle physics as a whole. However, exploration of the use of such techniques in low-latency, low-power FPGA hardware has only just begun. FPGA-based trigger and data acquisition (DAQ) systems have extremely low, sub-microsecond latency requirements that are unique to particle physics. We present a case study for neural network inference in FPGAs focusing on a classifier for jet substructure which would enable, among many other physics scenarios, searches for new dark sector particles and novel measurements of the Higgs boson. While we focus on a specific example, the lessons are far-reaching. We develop a package based on High-Level Synthesis (HLS) called hls4ml to build machine learning models in FPGAs. The use of HLS increases accessibility across a broad user community and allows for a drastic decrease in firmware development time. We map out FPGA resource usage and latency versus neural network hyperparameters to identify the problems in particle physics that would benefit from performing neural network inference with FPGAs. For our example jet substructure model, we fit well within the available resources of modern FPGAs with a latency on the scale of 100 ns.

Methodologie

Voir poster de Frédéric Duillole



3)

AT « Du capteur au cloud »

Emmanuel Bergeret, Samuel Manen,
Laurent Royer, David Sarramia

PAnAR 2021 - 25/11/2021

Organisation de l'AT

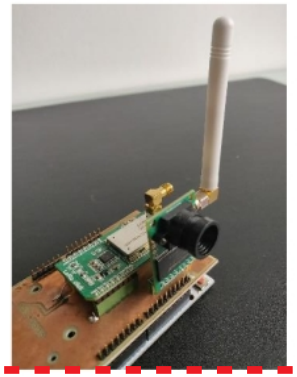
Indiquer ici l'organisation de l'AT au sein des collaborations éventuelles et en interne au laboratoire

Action thématique de l'AT	Cadre collaboratif	Rôle du LPC	Partenaires
IA intégré	Projet Région Soutien IN2P3	Conception d'asic spécifique	IP, CEA
Nœuds communicants innovants	I-SITE CAP 2025 (axes I et transverse)	Caractérisation fabrication de nœuds à fonctionnement énergétique très réduit	Geolab,YESITIS
Terra Forma	PIA	Design et conception capteurs+ cloud : synchronisation des réseaux	42 partenaires !!

Développements techniques

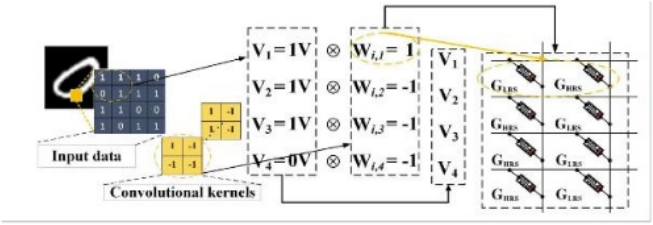
Indiquer ici les développements techniques principaux prévus dans les années à venir

Miniaturisation nœuds solo 1 semestre 2022

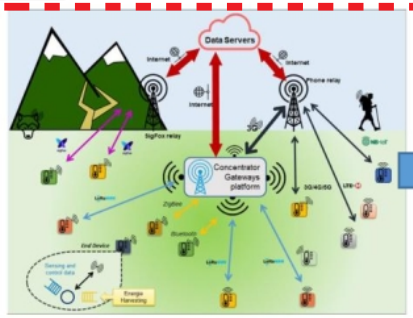


Passage prototype au nœud instrumental 1er semestre 2022

Circuits analogiques pour calcul CNN



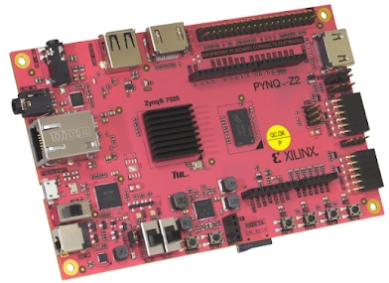
Fin 2022 /1er semestre 2023



Terraforma
Kick-off janvier 2022

Synchronisation multi-réseaux
Développement nœuds multiplateforme

Accélérateur	Labo	Charactéristiques	Software
DE1-SoC, Cyclone V SE ~400 € FPGA: 5CSEMA5F31(C6N)	LPC	ARM Cortex-A9 85k LE	Altera SDK (OpenCL) 
Cyclone V GT ~ 1100 € FPGA: 5CGTFD9E5F35(C7N)	LPC (clralicepc08)	228,000 LUT 458,000 FF 1,229 DRAM 300 DSP 301,000 LE	Intel SDK (OpenCL) 
MUSTANG-F100-A10 ~ 1650 € FPGA: Arria 10 1150 GX	LPC (clralicepc11)	1,150,000 LE	Intel SDK (OpenCL) Intel OpenVINO (ML) 

Accélérateur	Labo	Charactéristiques	Software
Xilinx Alveo U280 ~ 6800 € FPGA: XCU280	LLR (ACP)	1,203,870 LUT 2,472,213 FF 1,816 BRAM 9,020 DSP	Vitis AI 
Xilinx PYNQ-Z2 ~ 140 € ZYNQ-7000 SoC FPGA: XC7Z020-1CLG400C	LPC	ARM Cortex-A9 79,800 LUT 106,400 FF	Vitis HLS 

Annexes

Annexe A

■ *Fast Machine Learning at the Edge for HEP Experiments*, https://indico.cern.ch/event/1389765/

CIFAR-10 dataset, https://www.cs.toronto.edu/~kriz/cifar.html

ToyADMOS dataset, https://arxiv.org/abs/1908.03299

GAP9_EVK processor, https://greenwaves-technologies.com/gap9_processor/

Nucleo L4R5ZI board (STM32 microcontroller), https://www.st.com/en/evaluation-tools/nucleo-l4r5zi.html

Annexe B

Un problème de calcul : la multiplication des matrices

$$\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mm} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mm} \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mm} \end{bmatrix}$$

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj} \quad , \quad i = 1, \ldots, m \quad , \quad j = 1, \ldots, m$$

La complexité de calcul $= m^{\omega + o(1)} \quad , \quad 2 \leq \omega \leq 3 \quad , \quad o(1) \equiv (\forall \epsilon > 0)$

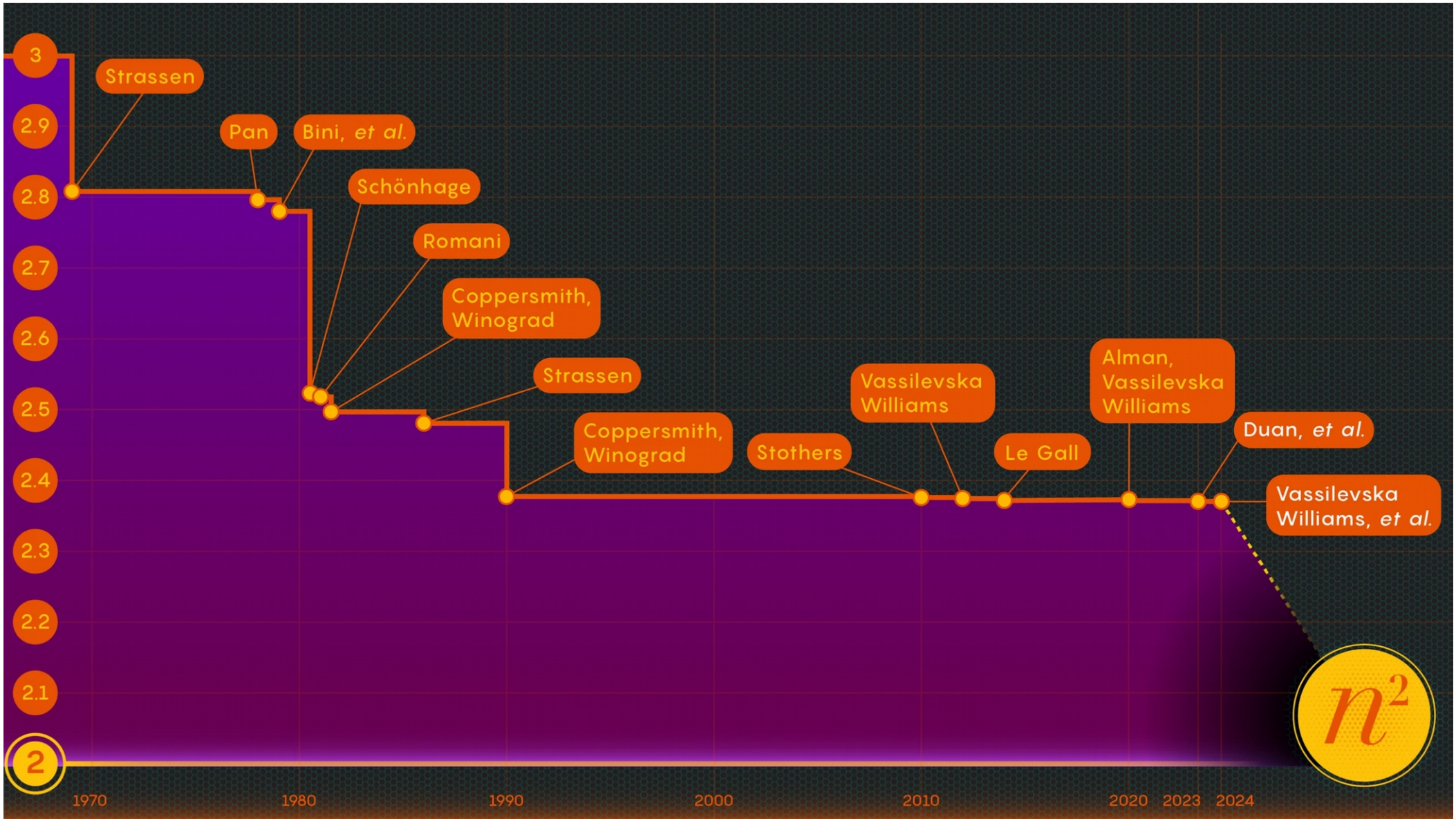
Table 3: Upper bounds of ω obtained by analyzing the m-th tensor power of the CW tensor.

m	Bounds from Our Methods	Bounds from Prior Works	References
1	N/A	2.387190	[CW90]
2	2.374631	2.375477	[CW90]
4	2.371919	2.372927	[Sto10, Wil12, LG14]
8	2.371866	2.372865	[Wil12, LG14]
16	N/A	2.372864	[LG14]
32	N/A	2.372860	[AW21b]

Faster Matrix Multiplication via Asymmetric Hashing

Ran Duan, Hongxun Wu, Renfei Zhou, https://arxiv.org/abs/2210.10173

Évolution de ω dans $m^{\omega+o(1)}$ de 1969 à aujourd'hui



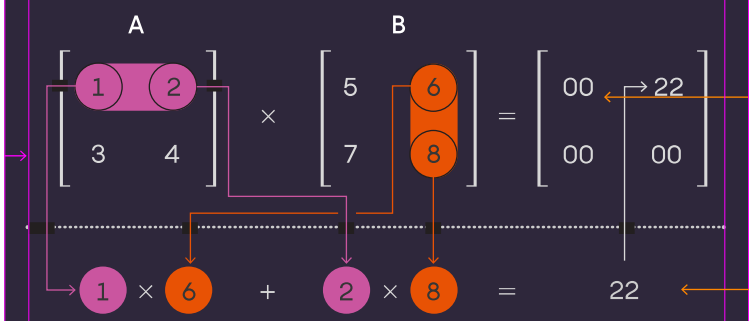
Merrill Sherman/Quanta Magazine https://www.quantamagazine.org/new-breakthrough-brings-matrix-multiplication-closer-to-ideal-20240307/

$8 = 2^3$ multiplications

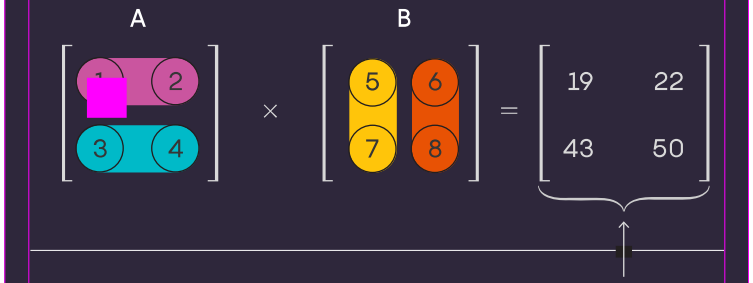
The Standard Algorithm

Combine the elements in a row of one 2-by-2 matrix and a column of another to generate one element in the product matrix.

CALCULATING ONE MATRIX ELEMENT



CALCULATING THE ENTIRE MATRIX



$1 \times 6 + 2 \times 8 = 22$

$1 \times 5 + 2 \times 7 = 19$

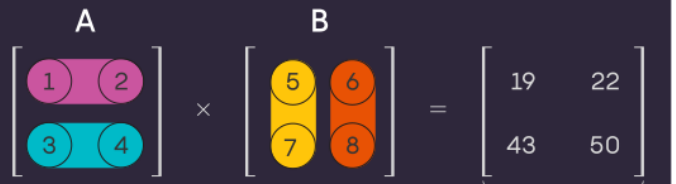
$3 \times 5 + 4 \times 7 = 43$

$3 \times 6 + 4 \times 8 = 50$

Eight multiplications

Strassen's Algorithm

This procedure for multiplying 2-by-2 matrices is less straightforward, but saves one multiplication step.



$(1 + 4) \times (5 + 8) = 65$

$(3 + 4) \times 5 = 35$

$(6 - 8) \times 1 = -2$

$(-5 + 7) \times 4 = 8$

$(1 + 2) \times 8 = 24$

$(-1 + 3) \times (5 + 6) = 22$

$(2 - 4) \times (7 + 8) = -30$

Seven multiplications

$65 + 8 - 24 + -30 = 19$

$-2 + 24 = 22$

$35 + 8 = 43$

$65 - 35 + -2 + 22 = 50$

$7 = 2^{2.807}$ multiplications