PCIe400 software development

J.-P. Cachemiche (CPPM)
Paul Bibron, Julien Langouët, Renaud Le Gac, Frédéric Réthoré

Sujets abordés

- Prérequis
- Modèles
- Pytest
- Outils communs

Prérequis

Bonne connaissance nécessaire de :

- Python3
- Pep8
  - PEP8 assure une homogénéisation de la façon de coder
  - renforce la lisibilité du code
- Pytest
  - Permet l'éxécution automatique des tests
  - Fonctions de haut niveau

Outil d'édition recommandé

- Pycharm
- Ou tout autre éditeur gérant le PEP8

Modèles

Approche objet

- Règle de base : 1 objet par composant
  - FPGA
  - PLL
  - Capteur de température
  - Mesure de tension
  - Mesure de courant
  - Transceiver optique
  - Etc …
- Eentuellement sous-objet s'il est complexe
  - Ex : FPGA
    - FPGA_Core
    - Sérialiseurs

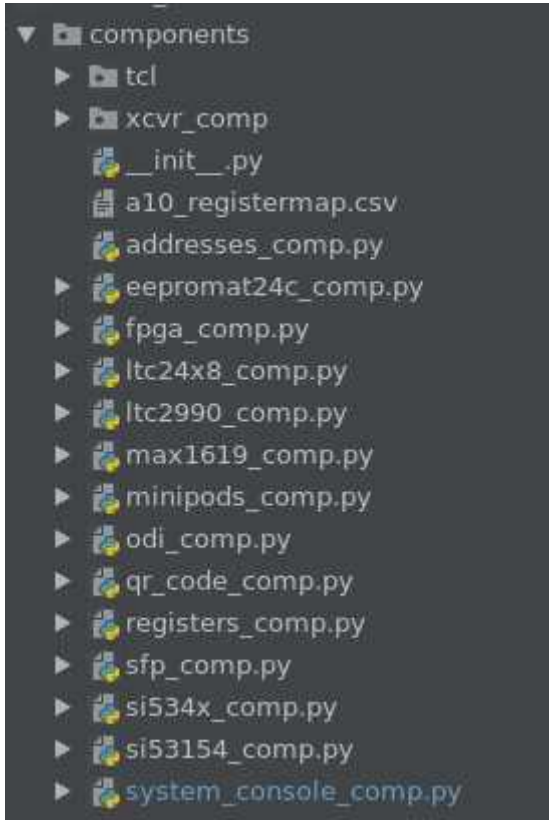
Méthodes

- Chaque composant incorpore toutes les méthodes qui permettent :
  - De l'initialiser (si nécessaire)
  - De lui envoyer des commandes
  - De lire des status

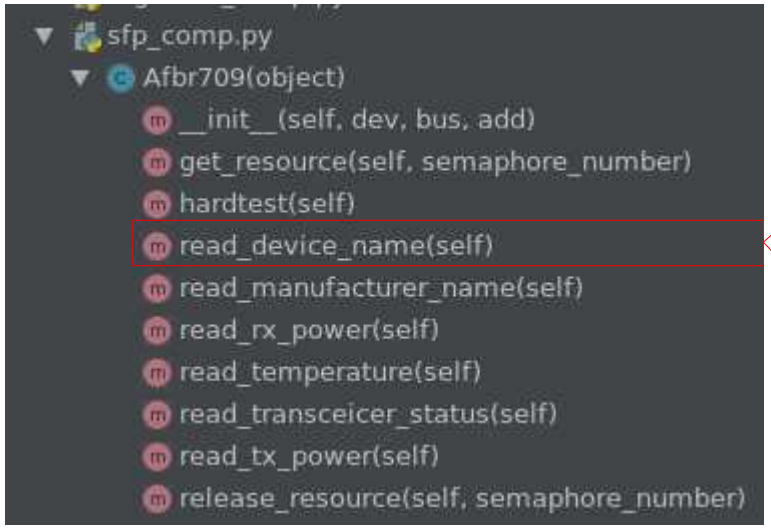
Tous les composants sont rangés dans un directory unique

Exemple

Liste des objets pour carte PCIe40



Tranceiver optique SFP+




```
def read_device_name(self):
    """

    Returns:
        str:
            device name

    """
    device_name = ''
    for reg in range(self.DEVICE_NAME_START, self.DEVICE_NAME_END + 1):
        self.get_resource(self.bus)
        dummy, value = i2c.read(self.dev, self.bus, self.add, reg)
        self.release_resource(self.bus)
        hexa = format(value, '#04x').split('x')[1]
        # device_name += binascii.unhexlify(hexa)
        device_name = hexa

    return device_name
```

PyTest

Principe

- La plupart des tests fonctionnels suivent le modèle Arrange-Act-Assert :
  - Organiser, ou mettre en place, les conditions du test
  - Agir en appelant une fonction ou une méthode
  - Affirmer qu'une condition finale est vraie

Avantages

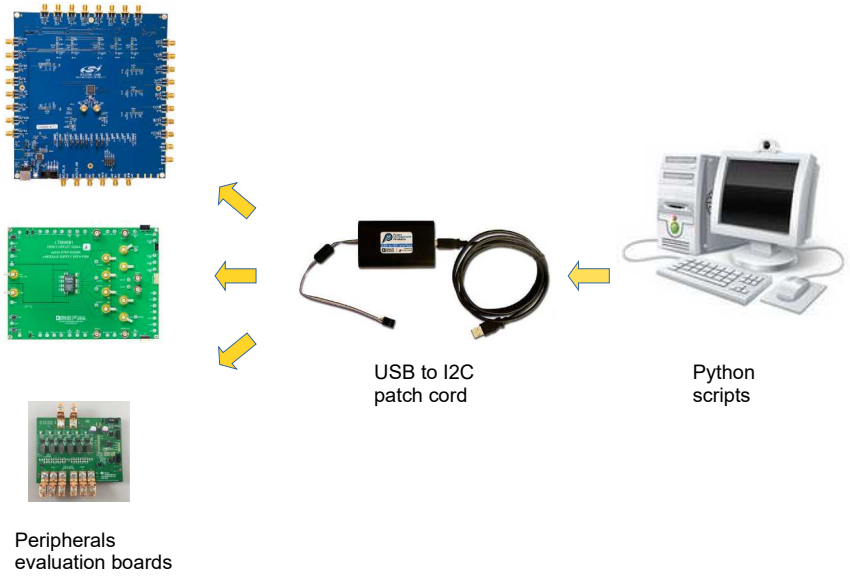
- Nombreuses fonctionalités de haut niveau venant sans développer de code
  - Tous les tests d'un même répertoire ou sous ce répertoire sont exécutés automatiquement
  - Filtrage par nom ou fraction du nom
    - Impose une discipline au niveau du nommage des tests
  - Filtrage inclusif ou exclusif.
  - Réexécution automatique des tests fails
  - Parallélisme
- Permet aussi bien de tester un objet individuel (composant) qu'un groupe d'objets interconnectés

Training

Organisation d'un training soit au CENBG, soit en Zoom

Prérequis :

- Montage d'une chaine de test pilotable à distance
  3 solutions :
  - Un setup très simple peut être mis en place avec une carte d'évaluation de périphérique
    - Necessité d'écrire les drivers de contrôle avec une interface USB/I2C
  - Utilisation d'une carte dévaluation Agilex et programmation d'un périphérique simple
    - Drivers déjà disponibles
  - Utilisation d'une carte PCIe40 et programmation d'un périphérique simple
    - Drivers déjà disponibles
  - Troisième solution certainement la plus simple si le training doit être mis en place dans un temps court



- Ouverture d'un serveur avec configuration appropriée vers l'extérieur

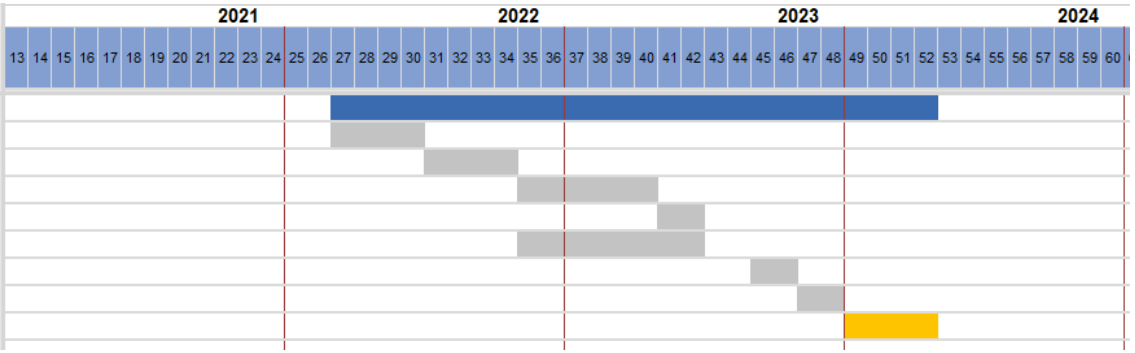
Outils communs

Vérifier que vos avez bien accès à :

- Box IN2P3 : https://box.in2p3.fr/index.php/apps/files/?dir=/PCIe400&fileid=18317950
- Mailing list : mailto:PCIE400-L@IN2P3.FR
- Git IN2P3 : https://gitlab.in2p3.fr
- Optionnellement Git CERN : https://gitlab.cern.ch

Rappel du planning

TASK DESCRIPTION	START	END	DURATION	RESOURCE
WP4: Software	1-Mar-22	30-Apr-24	26	
JTAG, PCIe drivers	1-Mar-22	30-Jun-22	4	CLab, CEF
DMA management	1-Jul-22	31-Oct-22	4	CLab, CEF
Python objects design for components	1-Nov-22	30-Apr-23	6	CENBG
Python object testing on evaluation cards	1-May-23	30-Jun-23	2	CENBG
Pytest layer design	1-Nov-22	30-Jun-23	8	CENBG
Pytest layer testing on real card	1-Sep-23	31-Oct-23	2	CENBG
DMA testing	1-Nov-23	31-Dec-23	2	CLab, CEF
Security margin	1-Jan-24	30-Apr-24	4	All



Conclusion

Un peu de préparation nécessaire avant le training

Quelques liens utiles :

- How to Write Beautiful Python Code With PEP 8
- Effective Python Testing With Pytest
- Pycharm