IN2P3
Institut national de physique nucléaire et de physique des particules

LPSC
GRENOBLE | MODANE

Développements d'intérêt général au sein du réseau DAQ

Olivier Bourrion (LPSC)

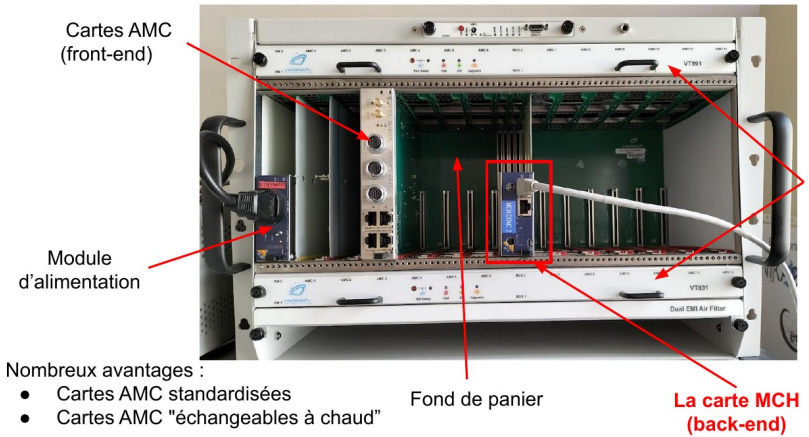
Sujets abordés

- Développements pour microTCA
- Ethernet 1G/10G sur FPGA
- Standard JESD204B pour DAC et ADC
- (Bientôt) *PCIe Over Cable*

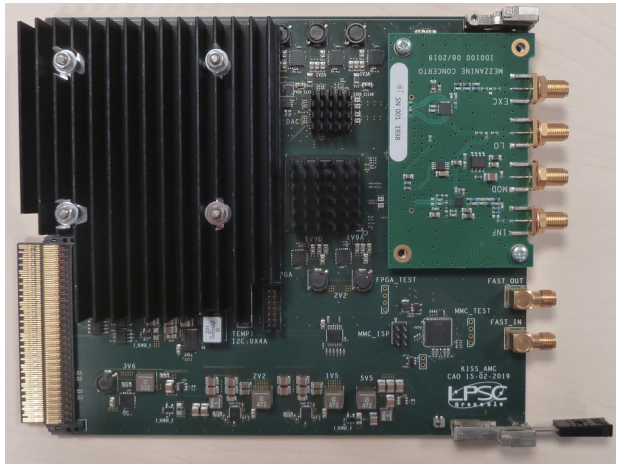
Rappels sur le microTCA

- Standard
  - électrique, mécanique, thermique et de **management => 1er point bloquant**
    - **hotswap**
    - **Capteurs**
    - **compatibilité**
  - Liens point à point en étoile sur le fond de panier
  - Maintenu par PICMG (PCI Industrial Computer Manufacturer Groups)
- Utilisé par (non exhaustif) :
  - CMS
  - DUNE
  - OXYGEN et IDROGEN
  - STEREO/Ricochet
  - Instruments KIDs : NIKA, CONCERTO
  - FASTER

Management du châssis



Mezzanine Carrier Hub
du fabricant NAT

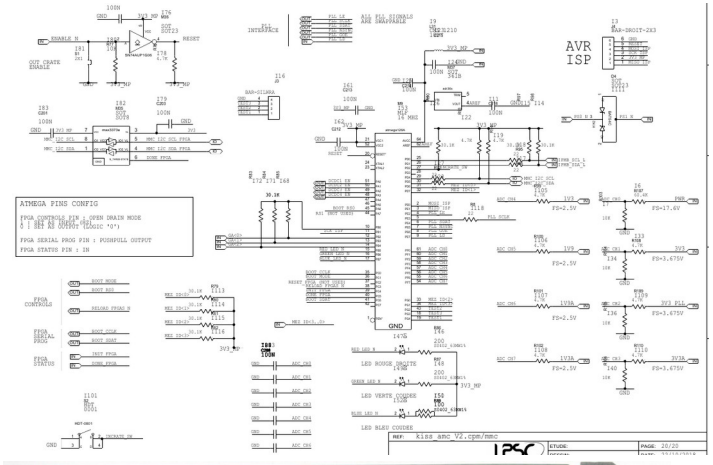


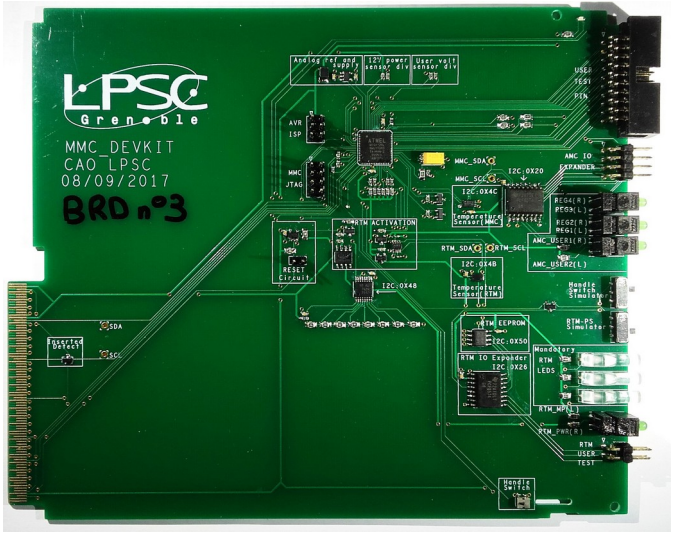
Advanced Mezzanine Card

- Chaque carte AMC doit déclarer ses capteurs et sa consommation et répondre aux requêtes de la carte contrôleuse (MCH)
  - => développement du module MMC (hardware + logiciel)
- La carte MCH comporte un switch Ethernet pour adresser tous les slots

Module MMC

- Construit autour d'un microcontrôleur 8 bits low cost (ATMEGA)
- Solution originale via CERN, mais empreinte mémoire nettement allégée et environnée d'outils et de modes opératoires
- Sous module git pour le firmware microcontrôleur
  - https://gitlab.in2p3.fr/AKIDO/MMC-base
  - Utilisé sur >8-10 cartes différentes
- Kit de développement low cost disponible pour la prise en main





Outils xTCA

- Outils pythons pour pilotage par IPMITOOLS
  - https://gitlab.in2p3.fr/DAQGEN/xTCA-tools
- Pilotage du châssis, des slots
  - Extinction, reboot, ventilateurs
- Lecture des capteurs
- Reprogrammation distante du MMC
- Reconfiguration de secours des FPGA
- ...

```
bourrion@lpsc2210x:~/soft/xTCA-tools/crate_mgr$ ./control.py 192.168.100.160 sdr 9
| AMC HOT_SWAP     | 0x01              | ok |
| 12V              | 12,49 Volts       | ok |
| 3.3V MP          | 3,34 Volts        | ok |
| 1.0V             | 1,01 Volts        | ok |
| 1.0V MGT         | 1,00 Volts        | ok |
| 1.2V MGT         | 1,20 Volts        | ok |
| 1.8V             | 1,81 Volts        | ok |
| 1.8V MGT         | 1,80 Volts        | ok |
| 3.3V             | 3,34 Volts        | ok |
| 5V               | 5,01 Volts        | ok |
| 24V              | 24,19 Volts       | ok |
| I core           | 0 Amps            | ok |
| I periph         | 0,21 Amps         | ok |
| I sensors        | 0 Amps            | ok |
| Power IN         | 2,42 Watts        | ok |
| Board Temp 1     | 25 degrees C      | ok |
| Board Temp 2     | 24 degrees C      | ok |
| FPGA Temp        | 35,12 degrees C   | ok |
| FPGA Status      | 0x00              | ok |
```

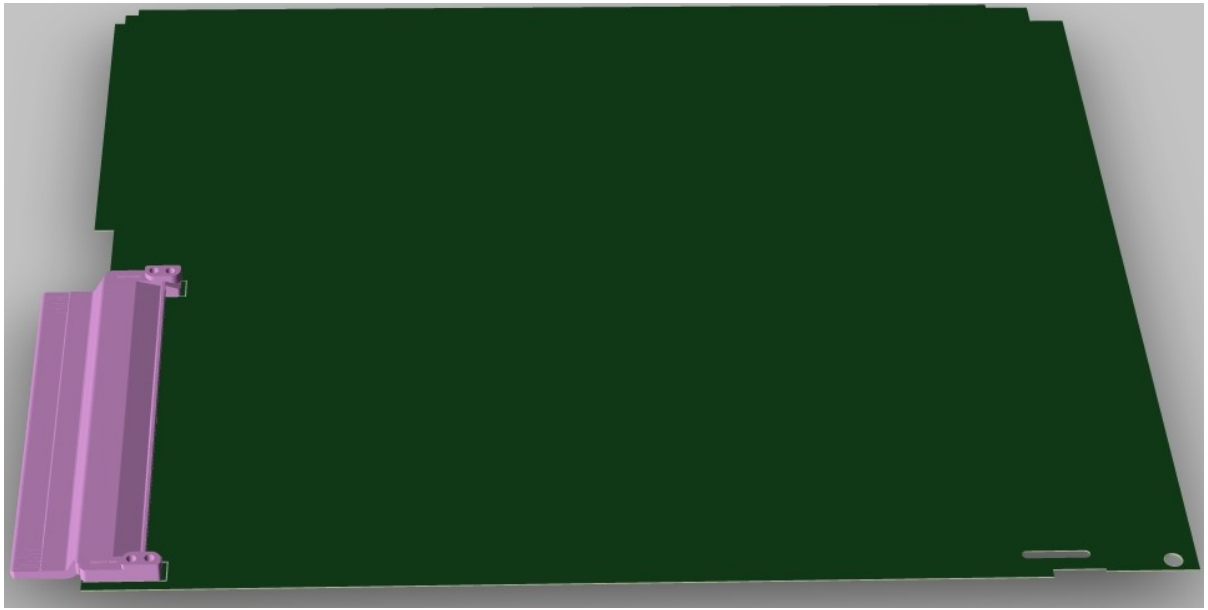
Etats des capteurs du slot 9

```
bourrion@lpsc2210x:~/soft/xTCA-tools/crate_mgr$ ./control.py 192.168.100.160 frulist
AMC          : Kiss_MCMPM_v2     ID=13   slot=9  IP=192.168.100.119      MAC=00:22:8f:03:00:47   MMC_fw=3.0
AMC          : Stereo_FE8_V2     ID=6    slot=2  IP=192.168.100.33       MAC=00:22:8f:03:00:0c   MMC_fw=2.1
POWER MODULE: NAT-PM-AC600       ID=50   site=0
FAN TRAY     : VT VT095          ID=41
MCH          : NAT-MCH-CM        ID=0
FAN TRAY     : VT VT095          ID=40
```

Liste du matériel dans le châssis

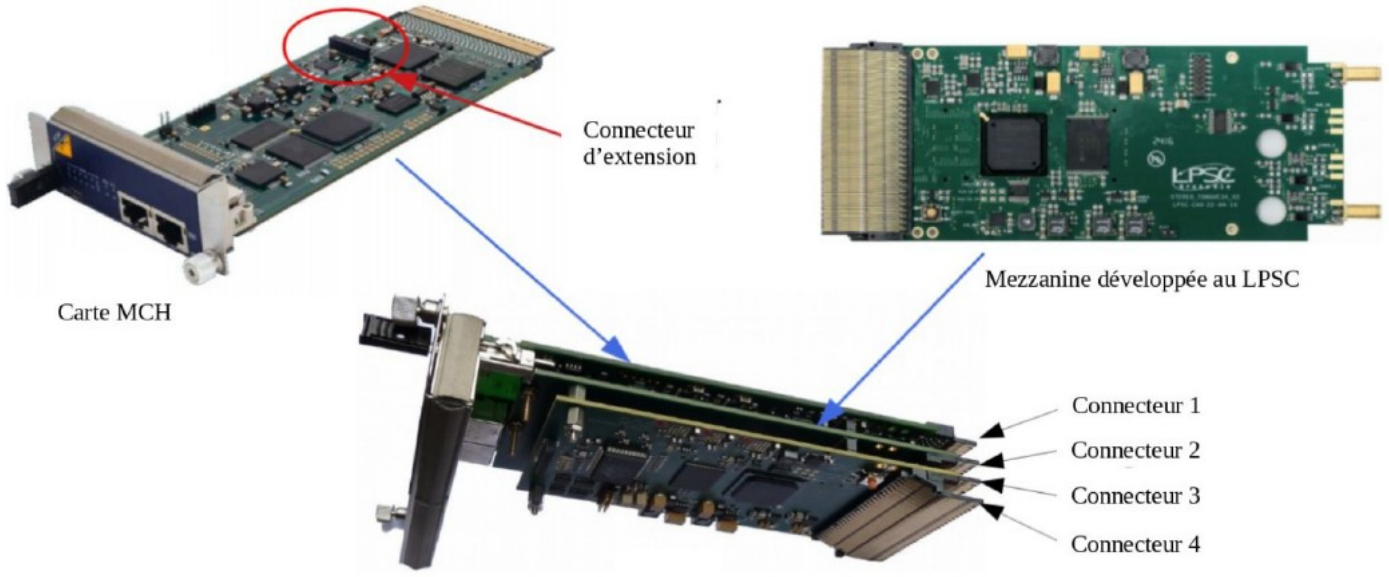
Mécanique/électronique

- Disponibles en bibliothèque partagée IN2P3 :
  - Contours de carte avec positionnement des éléments clés
  - Contours de carte avec connecteur PCB
  - Composants clés (connecteurs, MMC, ...)



Extension MCH

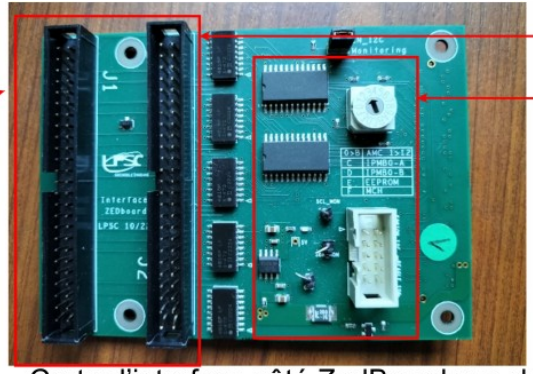
- Exploitée pour :
  - STEREO/RICOCHET (trigger et DAQ)
  - Instruments KIDs (NIKA, KISS, CONCERTO), distribution timing
- V3 bientôt disponible (Nouveau FPGA, interface white-rabbit, 3 ADC 10 bits 40Mech/s, 3 IO)



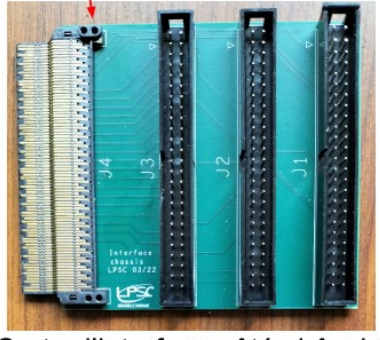
OpenMCH et R&T Timed (1/2)

- Besoins :
  - **EOL** de la NAT-MCH => solution propriétaire car une extension a été développée et est en exploitation
  - Distribution directe du temps par white-rabbit nécessite un switch SYNC-E
- Conception initiée via un apprentissage ingénieur (E. Conrat) d'une solution à base de SOC ZYNQ et avec un switch SYNC-E
  - Prototype de firmware+ gateware sur ZEDboard+ extension hardware pour piloter châssis (voir diapo suivante)
  - Carte openMCH en routage

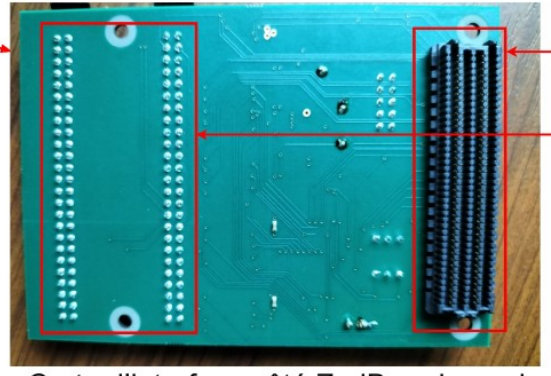
OpenMCH et R&T Timed (2/2)



Carte d'interface côté ZedBoard vue du dessus



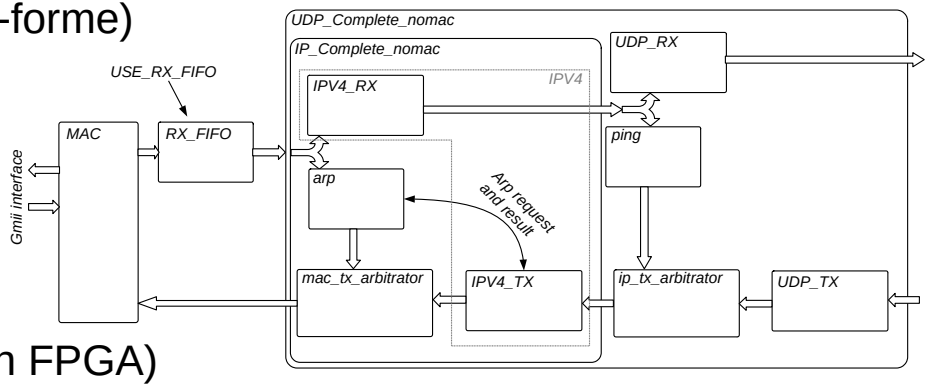
Carte d'interface côté châssis



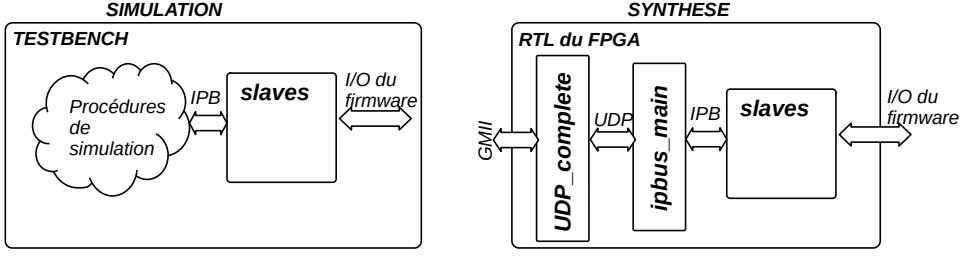
Carte d'interface côté ZedBoard vue du dessous

Ethernet 1G/10G sur FPGA

- Au départ une solution dénommée « ipbus » du CERN (firmware + logiciel), qui est un UDP sécurisé avec récupération de trame perdue
- Variante LPSC :
  - Réécriture du firmware pour le rendre aussi agnostique que possible de la cible et plus léger
  - Écriture d'outils logiciels légers sous Qt (multiplate-forme)



- Firmware :
  - Gère l'UDP, IPV4, le ping, l'ARP
  - Ressources de -30 % (pour le 1Gb)
  - Ajout DHCP+ hostname (au lieu de RARP)
  - Ajout d'un pilote de flash externe (reprogrammation FPGA)
  - Ajout d'outils de simulations à différents niveaux (procédure, package, faux serveur DHCP, …)
  - Dépôt : https://gitlab.in2p3.fr/AKIDO/ipbus-fw (accès à demander)



Outils logiciels IPBUS

- API permettant d'utiliser l'ipbus
- Outil de configuration de flash (en GUI ou script pour des dizaines de cibles)
- Script LUA pour diagnostique transactions ipbus via wireshark
- https://gitlab.in2p3.fr/AKIDO/ipbus-sw

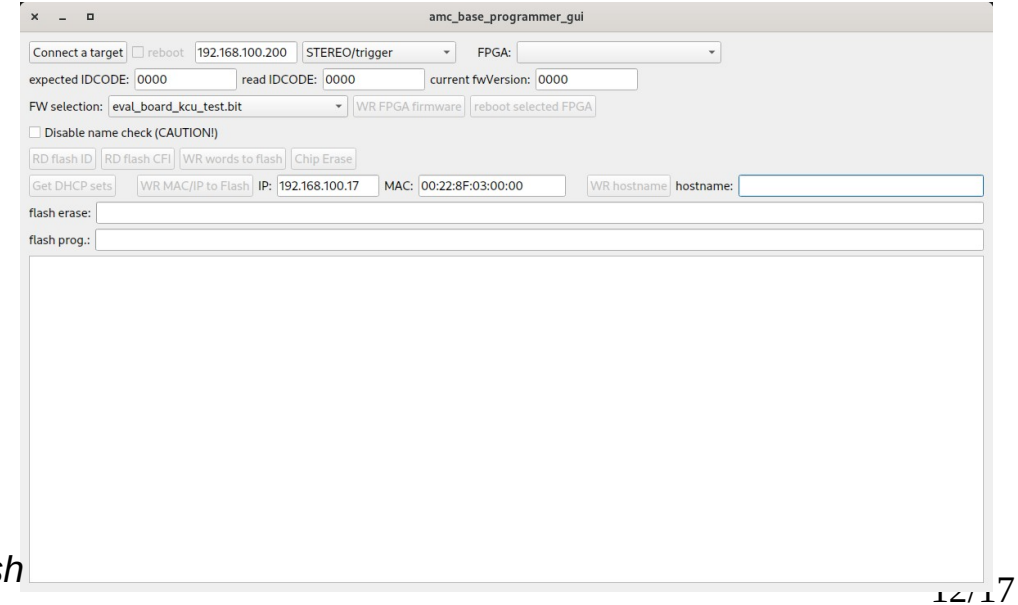
Méthodes publiques de la classe TIPbusClient.h

```
void connectClient(QString addStr, int port);
void read_std(int add, quint32 *buff,int size);
void read_ni(int add, quint32 *buff,int size);
void rmw_bit(int add, quint32 *buff,quint32 ANDterm,quint32 ORterm);
void rmw_sum(int add, quint32 *buff,quint32 A);
void write_std(int add, quint32 *buff,int size);
void write_ni(int add, quint32 *buff,int size);
void setTimeout(int val);
void dispatch();
```

```
#test comment, verification
192.168.100.33 STEREO/FE8_V1 CHECK
#192.168.100.19 NIKA/NIKEL_central_V1 CHECK
192.168.100.50 NIKA/NIKEL_AMC CHECK

#test comment, programmation
#192.168.100.1 STEREO/trigger PROG fpga_merger_proto8 ../fpga_merger_proto8_A014.bit
#192.168.100.33 STEREO/FE8_V1 PROG fpga_stereo_fe8_V1  ../fpga_stereo_fe8_V1_0xD01D.bit
#192.168.100.33 STEREO/FE8_V1 PROG fpga_stereo_fe8_V1  ../fpga_stereo_fe8_V1_0xD01D.bit
#192.168.100.50 NIKA/NIKEL_AMC PROG FPGA_SPLIT ../fpga_split_0x0012.bit
#192.168.100.50 NIKA/NIKEL_AMC PROG FPGA_PROC ../FPGA_proc_0xF20E.bit
```

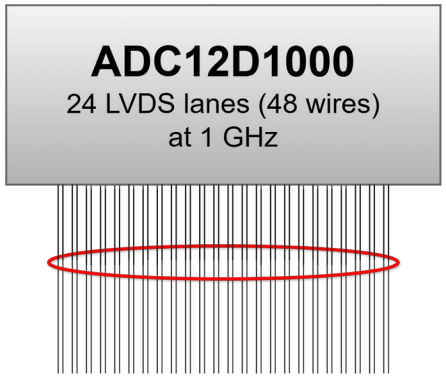
Script de configuration (mode console)

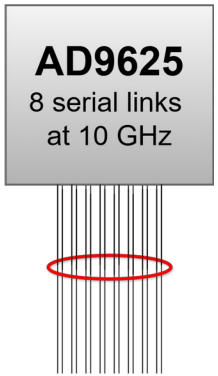


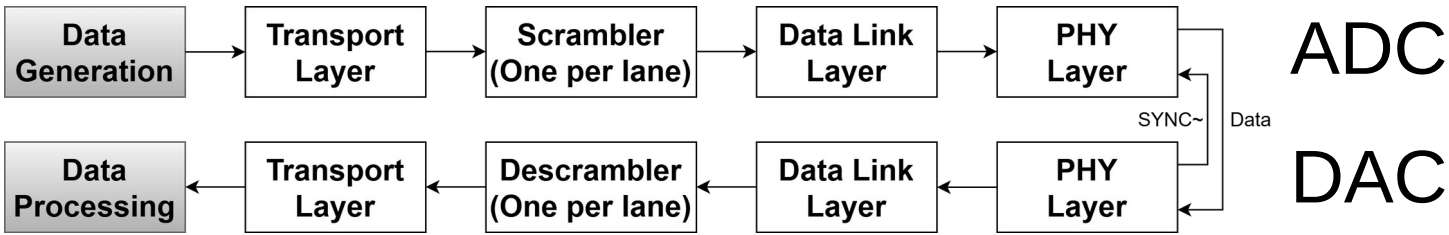
GUI de programmation des flash

JESD204B (1/3)

- Protocole de communication par lien série pour la communication avec les convertisseurs (ADC ou DAC)
  - Simplification routage et gain de taille de boîtier
  - Moins de problème de diaphonie et de consommation
- Contraintes :
  - Liens séries rapides (xx Gb/s)
  - Protocole à gérer, plusieurs couches
  - Possibilité de « scrambling » pour limiter CEM
- Solutions :
  - propriétaires (analog device et Texas Instrument), mais licences et compréhension au niveau transport layer requise tout de même
  - LPSC https://gitlab.in2p3.fr/AKIDO/JESD204B-fw

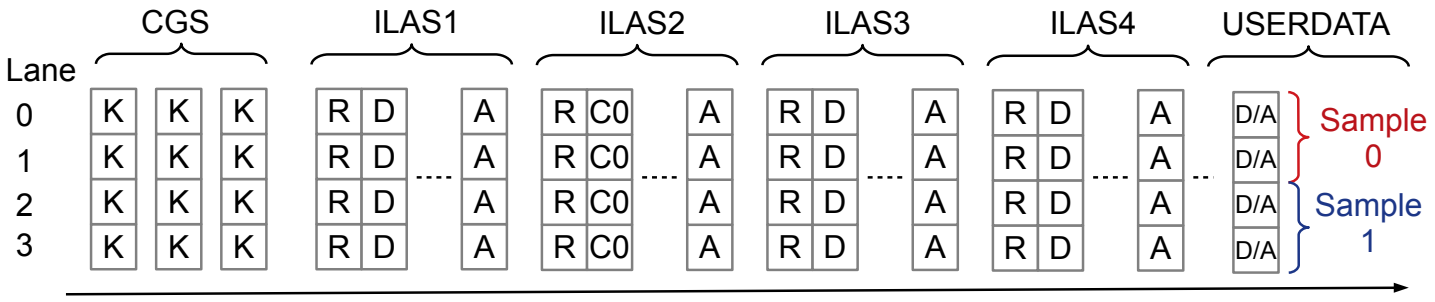






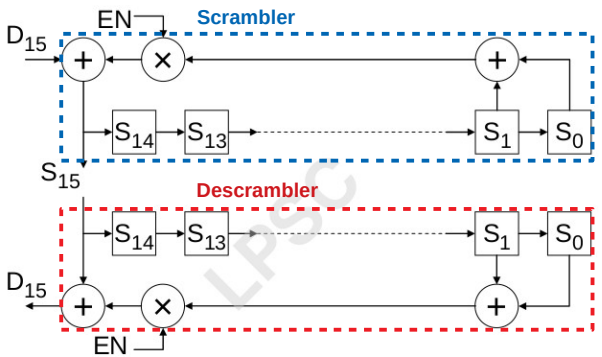
JESD204B (2/3)

- *CGS (Code Group Synchronization)* : alignement de chaque lien
- *ILAS (Initial Lane Alignement Sequence)* : synchronisation inter liens
- *USER DATA* : transmission des données utiles avec occasionnellement des remplacements de caractères (suivi communication)



- Scrambling

JEDEC Standard No. 204B – 5.2.4 – Figure 30



$D = D_0, D_1, D_2, \cdots = unscrambled\ data\ in \quad (MSB\ first)$

$S = S_0, S_1, S_2, \cdots = scrambler\ state = scrambled\ data\ out$

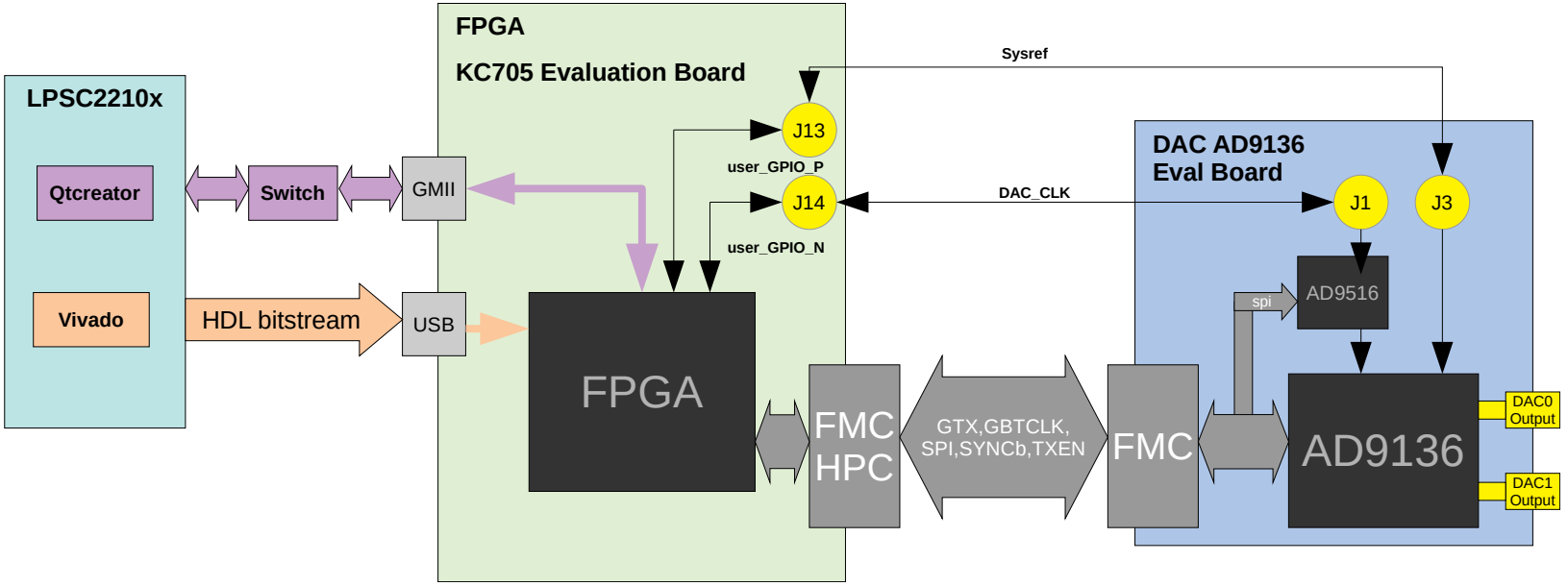
$serial\ update: S_{15} = D_{15} + S_1 + S_0\ ;\ D_{15} = S_{15} + S_1 + S_0$

$$parallel\ update: \begin{cases} S_{31} = D_{31} + S_{17} + S_{16}\ ;\ D_{31} = S_{31} + S_{17} + S_{16} & (LSB\text{-}16b) \\ S_{30} = D_{30} + S_{16} + S_{15}\ ;\ D_{30} = S_{30} + S_{16} + S_{15} & \\ \cdots & \\ S_{23} = D_{23} + S_9 + S_8\ ;\ D_{23} = S_{23} + S_9 + S_8 & (LSB\text{-}8b) \\ S_{22} = D_{22} + S_8 + S_7\ ;\ D_{22} = S_{22} + S_8 + S_7 & \\ \cdots & \\ S_{17} = D_{17} + S_3 + S_2\ ;\ D_{17} = S_{17} + S_3 + S_2 & \\ S_{16} = D_{16} + S_2 + S_1\ ;\ D_{16} = S_{16} + S_2 + S_1 & (MSB) \end{cases}$$

LPSC

JESD204B (3/3)

- Firmware https://gitlab.in2p3.fr/AKIDO/JESD204B-fw
- Applicatifs de test https://gitlab.in2p3.fr/AKIDO/JESD204B-sw
- Exemples fournis :
  - Kit eval ADCAD9625 (12 bits 2.6 Gech/s)
  - Kit eval DAC AD9136 (2x 16 bits 2 Gech/s)
  - Usage ADC AD9699 (14 bit 3 Gech/s)



PCIe Over Cable

- Objectifs :
  - Développer un écosystème **facile à réutiliser** (*ipbus like*) et **low cost** (PCIe gen3): firmware + software + outils
  - Permettre d'agréger plusieurs cartes via un switch PC vers fibres optiques (MPO en interface pour faire du breakout)
- Débuté en septembre 24 avec apprenti ingénieur (D. Huin)

Kit adnaco du commerce (en gen2)

Performance des liens PCI Express

Version	Année de lancement	Codage		Taux de transfert par ligne[a]	Bande passante				
					x1	x2	x4	x8	x16
1.0 / 1.1	2003	NRZ	8b/10b	2,5 GT/s	250 Mo/s	500 Mo/s	1 Go/s	2 Go/s	4 Go/s
2.0	2007			5 GT/s	500 Mo/s	1 Go/s	2 Go/s	4 Go/s	8 Go/s
3.0	2010			8 GT/s	985 Mo/s	1,969 Go/s	3,938 Go/s	7,877 Go/s	15,754 Go/s
4.0	2017		128b/130b	16 GT/s	1,969 Go/s	3,938 Go/s	7,877 Go/s	15,754 Go/s	31,508 Go/s
5.0	2019			32 GT/s	3,938 Go/s	7,877 Go/s	15,754 Go/s	31,508 Go/s	63,015 Go/s
6.0	2022	PAM4 FEC	FLIT 242 octets/256 octets	64 GT/s (32 GBd)	7,563 Go/s	15,125 Go/s	30,25 Go/s	60,5 Go/s	121 Go/s
7.0	2025 (prévu)			128 GT/s (64 GBd)	15,125 Go/s	30,25 Go/s	60,5 Go/s	121 Go/s	242 Go/s

Conclusion

- Beaucoup de choses peuvent (et sont partagées)
  - Merci de contribuer en retour (commentaires, découverte de bug, corrections, ...)
- Contributeurs/contacts :
  - MMC : D. Tourres
  - Extension MCH : O. Choulet/O. Bourrion
  - xTCA tools : D. Tourres/ O. Bourrion
  - OpenMCH : D. Tourres/ O. Bourrion
  - ipbus/Ethernet : O. Bourrion
  - JESD204B : J. Bounmy
  - POC : J. Bounmy/D. Tourres
  - Composants et contours de cartes : N. Ponchant