Cartes pour le pilotage et l'acquisition d'ASIC / MAPS solutions « Custom or COTS (Commercial off-the-shelves) » ?

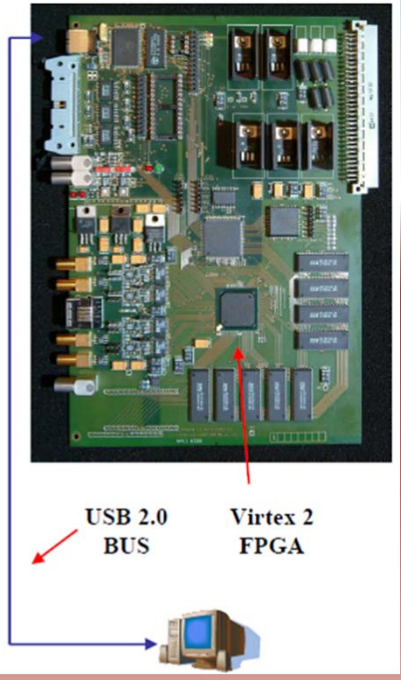
2006
Dernière carte DAQ USB développée dans notre groupe

Analog
- 1-4 ADC 12 bits – 40 Mhz
- 1 ADC 14 bits – 100 MHz
- 10^6 pixels **shared** 1-4 Inputs

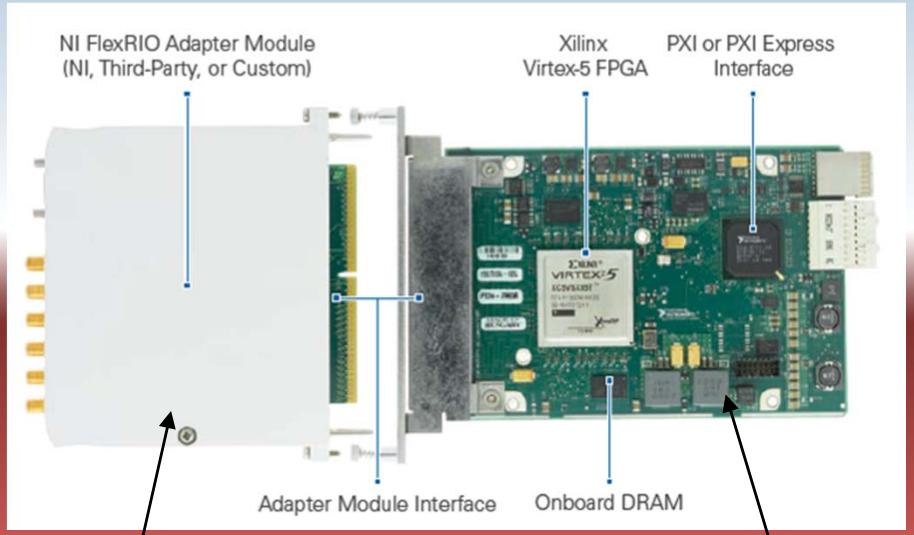
Digital
- **Pattern** generator **& Trigger**
- 16 **Digital inputs**

Daq
- Transfer **15 MB/ s USB 2.0**

Firmware
- CDS Calculation (Done)



Depuis 2008
Architecture Flex RIO de NI



Adapter Module I/O
Digital (up to 20 x 1 Gb/s)
Analog (up to 14 bits 250 MSPS)

Flex RIO board
66 digital I/O ⇔ Bus PXIe
FPGA - Custom FW

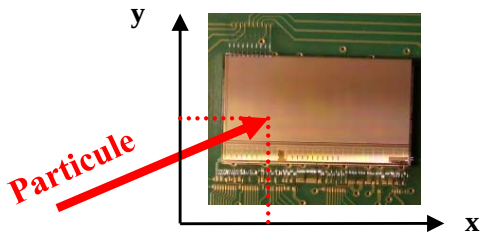
Custom HW / COTS : Question tranchée ou compromis permanent ?
Notre expérience dans les collaborations EUDET, AIDA, BEAST …

Plan

- Qui sommes nous : Groupe microélectronique IPHC ?
- Un Télescope de faisceau ... « Pour observer quoi ? »
- Le DAQ du Télescope de faisceau EUDET Custom / COTS
- FW / SW – LabVIEW FPGA / VHDL – LabVIEW / C
- Nos futurs besoins en DAQ : COTS ou Custom HW
- Conclusion

- **Objectifs et Cycle de conception**
  - **La conception d'ASIC et de capteurs à pixels (MAPS) pour les détecteurs des expériences de physique**
  - **La conception d'un ASIC / MAPS nécessite un cycle prototypage pour atteindre les performances requises**
    - **Conception Microélectronique**
    - **Fabrication du prototype (fonderie microélectronique AMS, TSMC, TowerJazz)**
    - **Test et Caractérisation de plusieurs exemplaires du prototype**
    - **Corrections et améliorations du design Microélectronique … convergence vers l'ASIC / le MAPS final**

R&D PICSEL
Capteur à pixels = MAPS (Monolithic Active Pixels Sensor



- **But = Détecter les coordonnées x, y de passage des particules**
- **Exemple Mimosa 26**
  - **Minimum Ionising MOS Active pixel sensor**
  - **Matrice 2 cm² - 660 000 pixels 18,4 µm x 18,4 µm**
  - **8680 frames / s**

CAO Microélectronique

R&D PICSEL ➔Physics with Integrated Cmos Sensors and ELectron machines

Cycle de conception et caractérisation d'un MAPS ➔ Conception & Caractérisation de ~ 3 MAPS / An

Thématique PICSEL
5 Chercheurs
Définition MAPS
Analyse « Beam tests »


J.BAUDOT

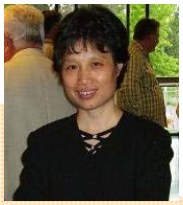

M.WINTER


I.RIPP-BAUDOT


A.PEREZ PEREZ


A.BESSON

R&D Microélectronique
10 Ingénieurs
Conception MAPS


C.HU


C.COLLEDANI


M.KACHEL


I.VALIN


F.MOREL


H.PHAM


A.DOROKHOV


G.DOZIERE


A.HIMMI


G.BERTOLONE

Test Microélectronique
5 Ingénieurs
Conception bancs de Test
Caractérisation des MAPS


G.CLAUS


M.SPECHT


K.JAASKELAINEN

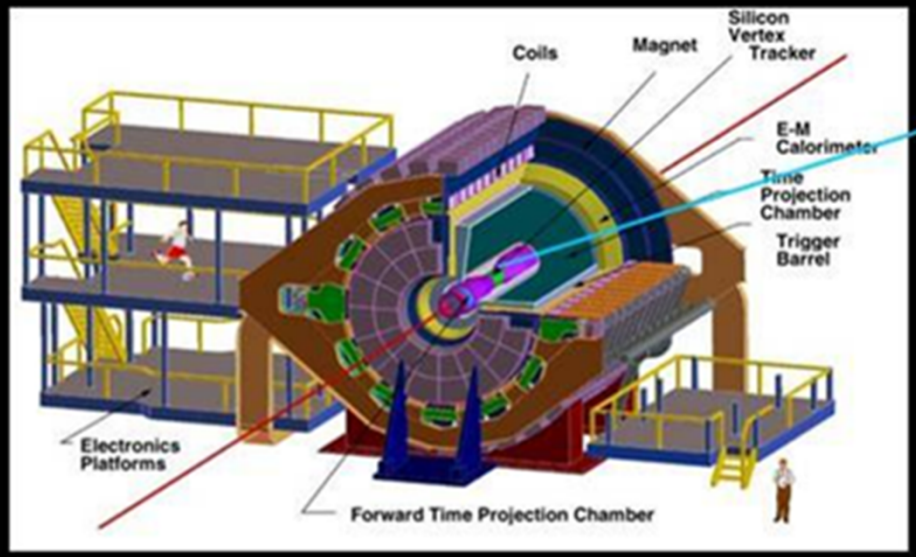

M.GOFFE

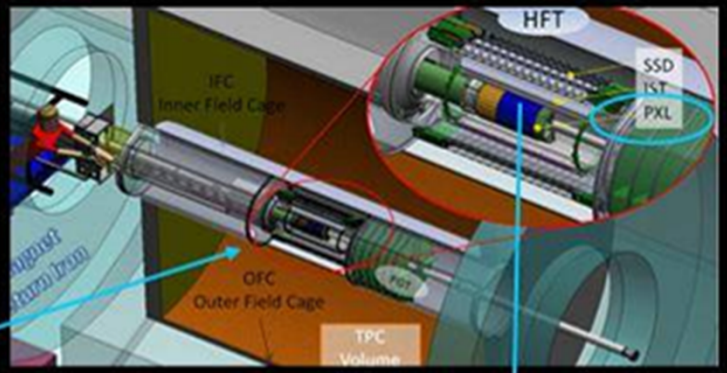

M.SZELEZNIAK

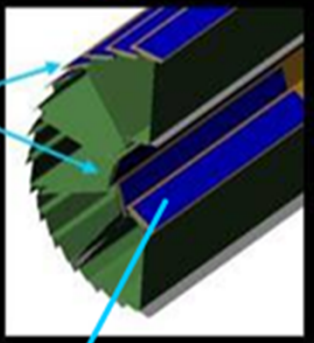
Résultats ➔ STAR Premier détecteur au monde équipé de MAPS

Upgrade Expérience STAR (Brookhaven National Laboratory) - USA **Opérationnel depuis 2013**

- Collisions de faisceaux d'Ions lourds (Pb)
  - → Coordonnées X,Y,Z des particules émises
  - → Capteur à Pixels 2D = Coordonnées X,Y
  - → Plusieurs couches de capteurs = Coordonnée Z
- Ajout de deux couches de capteurs à pixels → 2012 – 2013





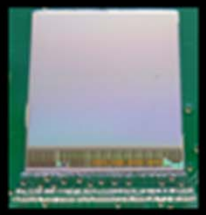


Deux couches de capteurs Mimosa 28

- Longueur 20 cm
- Rayon 2,5 cm & 8 cm
- surface 1600 cm² recouverte de 400 capteurs 2 cm x 2 cm
- Flux données max ~ 16 Go/s

Capteur Mimosa 28

- 2 cm x 2 cm ~ 1 milion pixels (1024 x 960)
- 5 000 images / seconde
- Sortie numériques 320 Mb/s

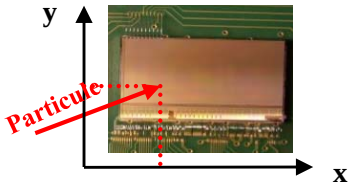


10 capteurs Mimosa 28 assemblés pour former une réglette

Télescope de faisceau ...

Fonction du MAPS

➢ Détecter les coordonnées x, y de passage des particules



Comment caractériser le MAPS ?

➢ Le placer dans un faisceau de particules

➢ Comparer les couples (x,y) qu'il génèrent à ceux de détecteurs de référence

➢ Ces détecteurs de référence forment le Télescope de faisceau

Deux générations de Télescopes

➢ Capteurs de référence = Détecteurs silicium à pistes

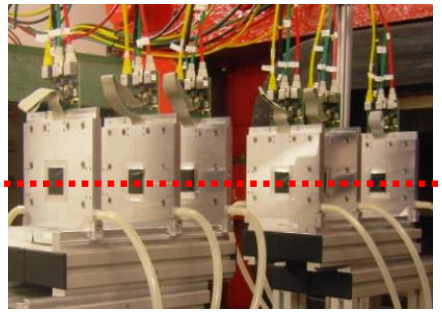
➢ Capteurs de référence = C'est aussi un MAPS ➔ Télescope EUDET

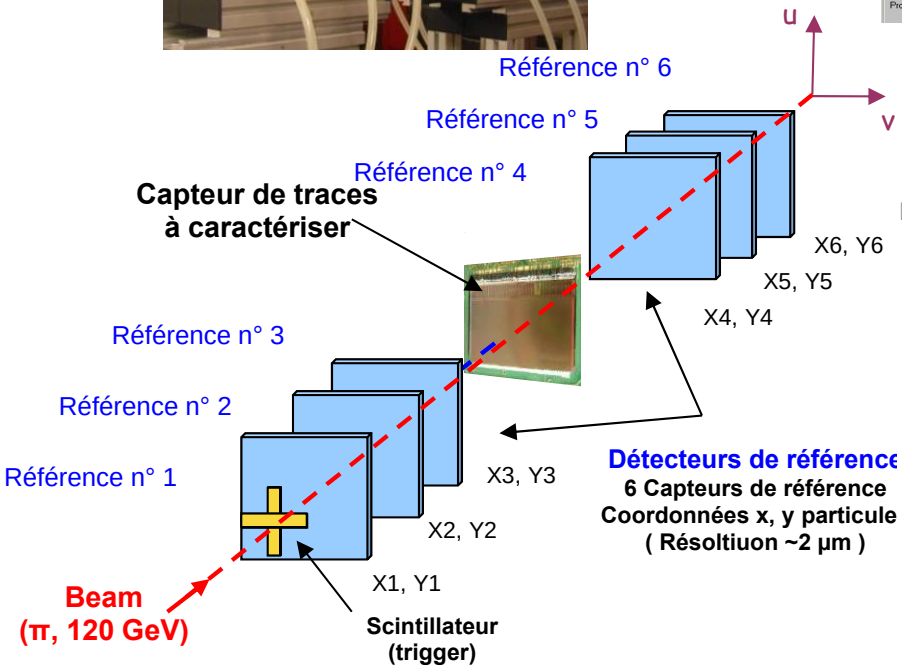
Télescope de faisceau du Projet Européen EUDET FP6 2006-2010

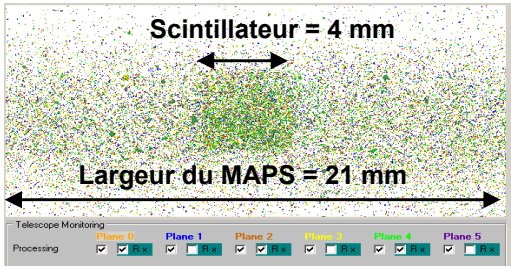
➢ IPHC ➔ Conception des capteurs de référence : Mimosa 26

➢ EUDET ➔ Construction d'un Télescope de faisceau hautes performances

Télescope de faisceau EUDET
6 Capteurs de éférence Mimosa 26

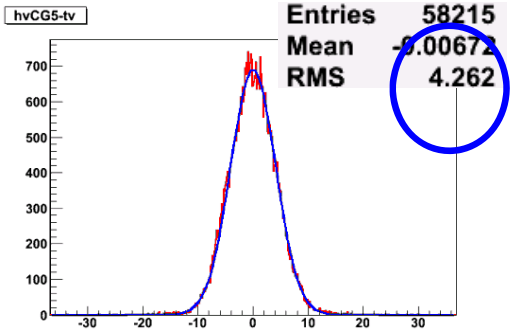






On-line monitoring

Off-line analysis



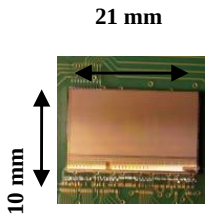
Résolution – Analyse off-line
Mimosa 26 : σx ~ σy ~ 4 – 4,5 µm

Mimosa 26 : Ce n'est plus « un proto de R&D »

➔ C'est le premier MAPS destiné à une expérience : Le Télescope EUDET

Le capteur Mi 26 (2009)

- 663 552 Pixels
- Surface ~ 2 cm²
- ~ 8680 images / seconde
- Suppression de zéro intégrée



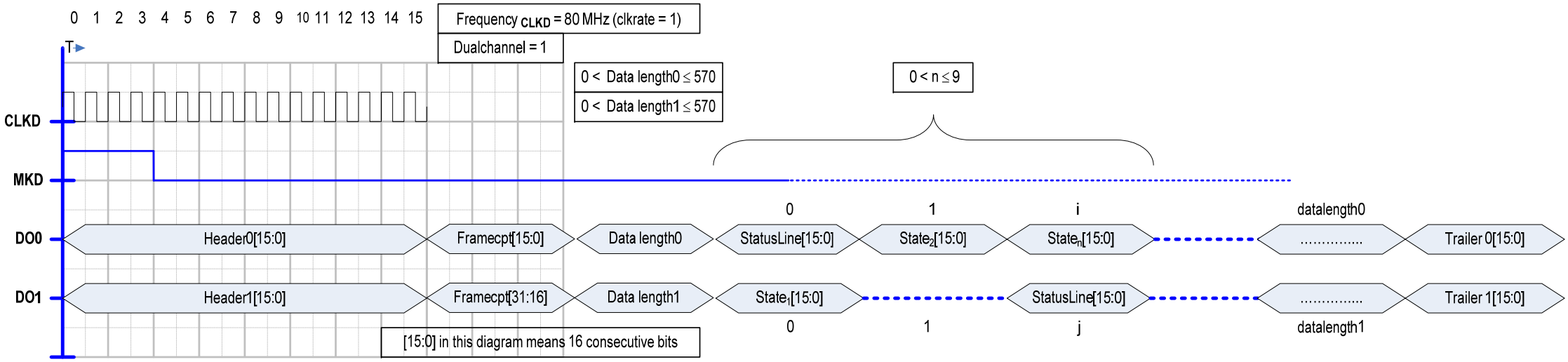
Mimosa 26

20 MB/s

Protocole série synchrone

- Lien série propriétaire 4 fils
  - Horloge 80 MHz (CLKD)
  - Signal de synchro (MKD)
  - Deux lignes de données (D00, D01)
- Cadence 80 MHz – 9216 bits / trame ➔ 20 MB/s

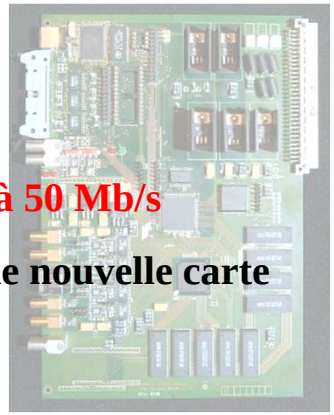
Protocole de lecture de Mimosa 26

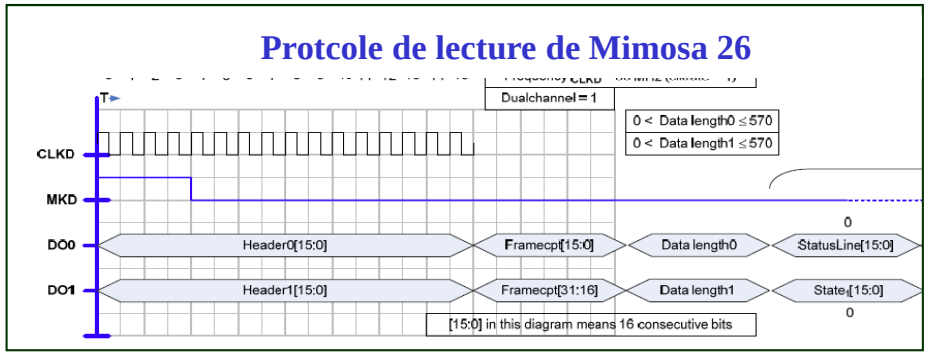


Les raisons de notre migration « DAQ Custom » vers COTS NI …

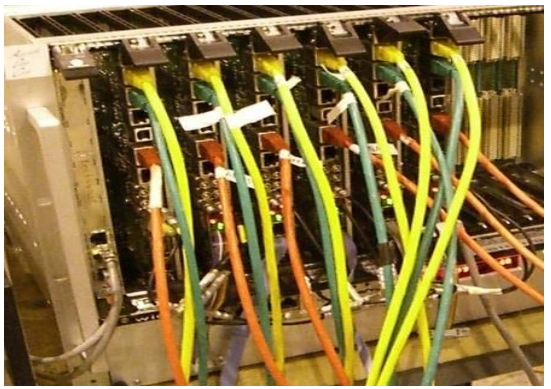
- Le capteur Mimosa 26
  - Deux sorties série 80 Mb/s
  - Carte DAQ USB « Custom » limitée à 50 Mb/s
  - Pas de ressources pour développer une nouvelle carte

Carte DAQ USB





- Une demande de la collaboration EUDET …
  - La collaboration développe un DAQ VME pour le Télescope
    - Carte EUDRB – INFN Ferrara



Télescope de faisceau EUDET et son DAQ VME

- Volonté de dupliquer le Télescope ➔ Dupliquer son DAQ
  - La collaboration nous demande de proposer une solution COTS NI
    - Raison : Pas de « manpower » pour produire les cartes VME

➔ Nous avons étudié et proposé une solution NI

Télescope EUDET : Migration DAQ par étapes ...

En deux étapes … composer avec les contraintes de la réalité …

2009 → Le coup de feu

- Il faut un équipement pour caractériser Mimosa 26
  - Au laboratoire et en faisceaux de test au CERN
  - Mais pas de contrainte de lecture continue du capteur
    - Le système peut avoir du temps mort !
- « Solution soft et souple » → Carte PXI 6562
  - Carte d'acquisition digitale 16 voies 200 MHz
  - Désérialisation par logiciel → 94 % de temps mort

Carte PXI 6562 dans son châssis

→ Pas besoin d'expert firmware ☺

2010 → La réflexion à long terme

- Il faut une solution DAQ pour les expériences
  - Le temps mort n'est plus permis
  - Il faut lire en continu un Télescope
    - EUDET 2010 → 6 Capteurs → 6 x 20 = 120 MB/s
  - Ce système doit être extensible
    - AIDA 20XX → 72 Capteurs → 72 x 20 = 1,4 GB/s
- DAQ PXIe → Carte Flex RIO 7962R
  - Codage de la désérialisation on board
  - Bande passante du bus PXIe (x 4 = 800 MB/s)

Carte Flex RIO PXIe 7962R

→ Nécessité d'un expert firmware …

Télescope EUDET : Les Atouts des cartes Flex RIO

NI FlexRIO FPGA Modules

NI PXI-795xR, NI PXIe-796xR *NEW!*

- NI FlexRIO FPGA modules
- PXI and PXI Express
- Xilinx Virtex-5 SXT and LX FPGAs
- Programmable with the LabVIEW FPGA Module
- Up to 512 MB onboard DRAM
- Peer-to-peer data streaming between PXI Express modules at more than 800 MB/s
- Up to 16 DMA channels
- 132 single-ended or 66 differential data lines to adapter module interface
- Up to 66 Gbits/s adapter module bandwidth
- Multi-adapter module synchronization for high-channel-count applications
- Adapter modules available from NI and third parties as well as through custom development with the NI FlexRIO Adapter Module Development Kit (MDK)

Operating Systems
- Windows 7/Vista/XP/2000
- LabVIEW Real-Time

Required Software
- LabVIEW
- LabVIEW FPGA Module

Recommended Software
- NI FlexRIO Adapter Module Development Kit

Driver Software
- NI-RIO
- NI FlexRIO adapter module support

Model	Bus/Form Factor	FPGA	FPGA Slices	FPGA DSP Slices	FPGA Memory (Block RAM)	Onboard Memory (DRAM)
NI PXIe-7965R	PXI Express	Virtex-5 SX95T	14,720	640	8,784 kbits	512 MB
NI PXIe-7962R	PXI Express	Virtex-5 SX50T	8,160	288	4,752 kbits	512 MB
NI PXIe-7961R	PXI Express	Virtex-5 SX50T	8,160	288	4,752 kbits	0 MB
NI PXI-7954R	PXI	Virtex-5 LX110	17,280	64	4,608 kbits	128 MB
NI PXI-7953R	PXI	Virtex-5 LX85	12,960	48	3,456 kbits	128 MB
NI PXI-7952R	PXI	Virtex-5 LX50	7,200	48	1,728 kbits	128 MB
NI PXI-7951R	PXI	Virtex-5 LX30	4,800	32	1,152 kbits	0 MB

Caractéristiques

- 66 I/O différentielles
- Module d'I/O = « Adaptor module »
  - LVDS inputs module → NI 6585
  - Développer son propre module
- Implantation de firmware utilisateur
  - Virtex 5
- PXIe bus x 4
  - ~ 200 MB/s / Lane
  - 4 lanes / board = ~ 800 MB/s ?
- Mémoire → 512 MB DRAM

Les atouts !

- Firmware développé par l'utilisateur
- Bande passante ~ 800 MB/s
  - Peer-to-peer data streaming
- Le module d'entrée
  - L'acheter chez NI
  - Développer le sien

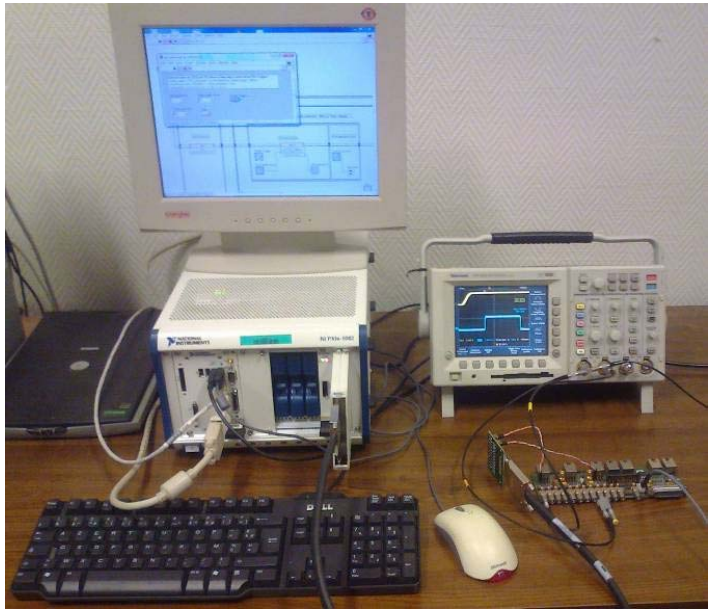
Télescope EUDET : Flex RIO et « Adapter » module

- Ressources sur la carte Flex RIO PXIe 7962R
  - FPGA Virtex 5
  - Deux blocs de DRAM bus 64 bits - 512 MB chacun
  - Bus PXIe vers le CPU → BW annoncée ~ 800 MB/s

Module NI 6585
32 I/O LVDS – 200 (300) MHz

Carte Flex RIO
PXIe 7962R

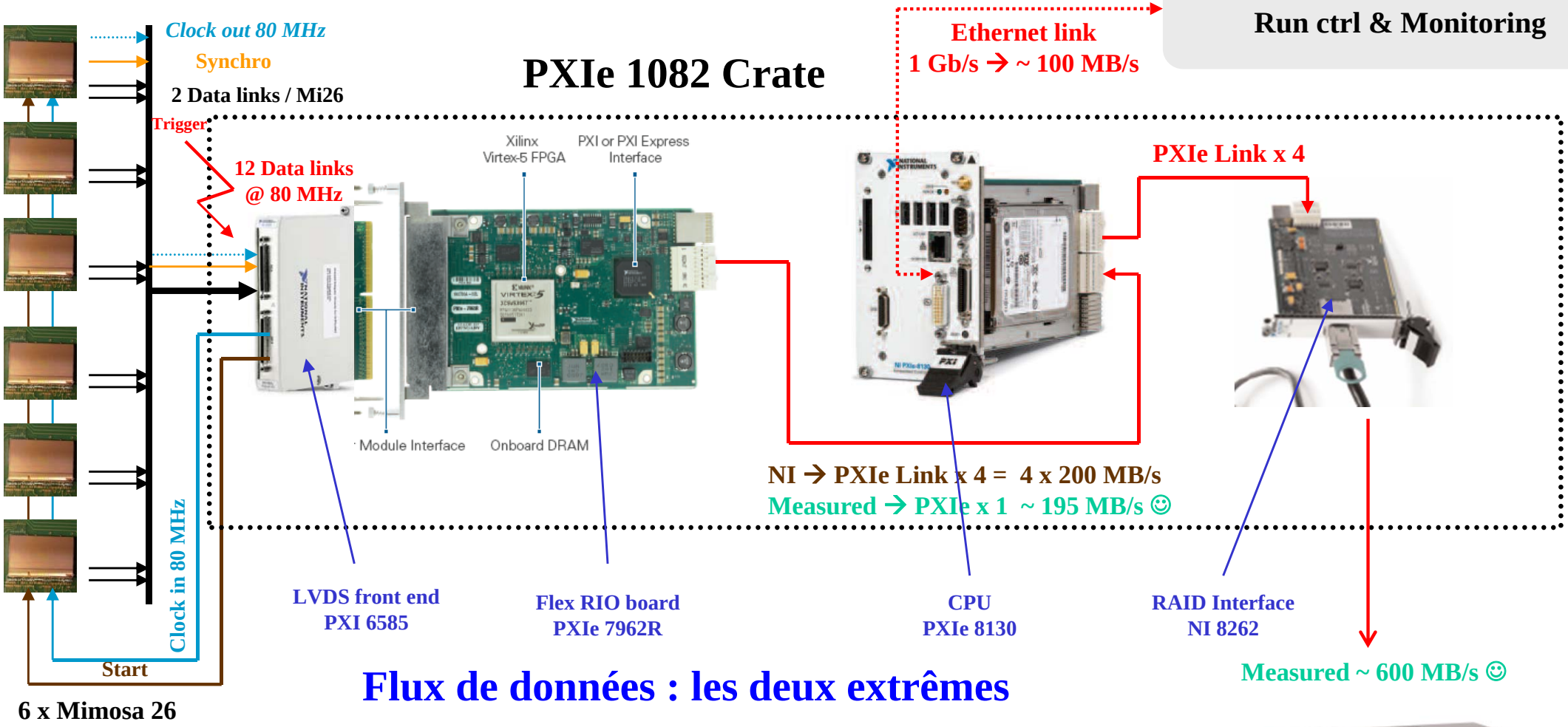
- Interface d'entrées / sorties via module LVDS NI 6585
  - Deux ports DDCA et DDCB – Sur chacun :
    - 1 x horloge en entrée → DDC Clock In
    - 1 x horloge en sortie → DDC Clock Out
    - 4 x lignes d'usage général (Synchro, trigger, etc …) → PFI 1,2,3,4
    - Un bus 16 bits → DDC 00..15
  - Bit rate maximal par ligne d'I/O
    - 200 Mbps en SDR = Single Data Rate (Horloge 200 MHz)
    - 300 Mbps en DDR = Double Data rate (Horloge 150 MHz)
      - Génération sur les deux fronts d'horloge



Setup utilisé pour évaluer la Flex RIO

Télescope EUDET : Architecture du DAQ

DAQ → 1 Flex RIO + 1 CPU + 1 RAID



Flux de données : les deux extrêmes

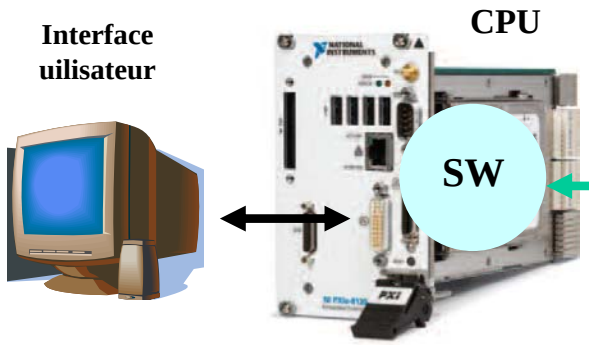
- "Faisceau faible intensité" → Flux données << 120 MB/s → Ethernet
- "Faisceau forte intensité" → Flux données ~ 120 MB/s → RAID

RAID Disk - 3 TB
HDD 8264

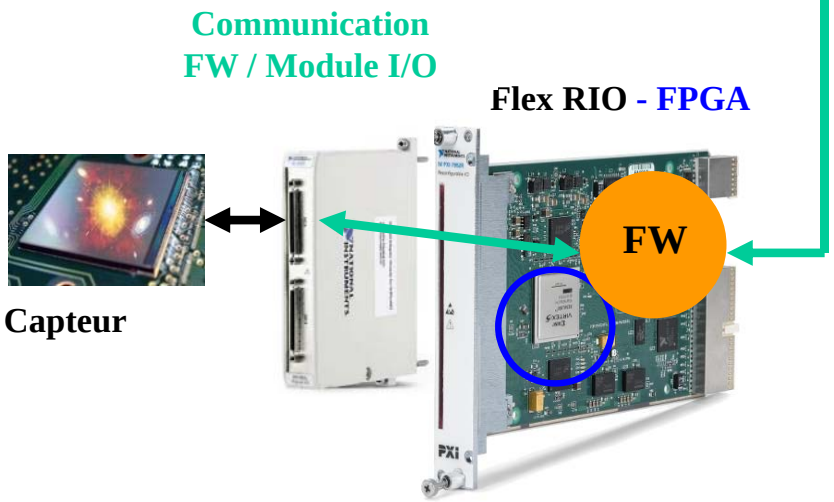
More about DAQ → http://www.eudet.org/e26/e28/Eudet-Memo-2010-25, 26, 27, 28

Télescope EUDET : Firmware ou Software

- **Software → SW**
  - Exécuté par le CPU
  - Facilité Développement → Pas besoin de connaissances HW
  - Points faibles
    - Réponse temps réel
    - Flux de données élevé
    - Traitement parallèles



- **Firmware → FW**
  - Exécuté par un FPGA
  - Difficulté Développement → Connaissances HW requises
  - Points forts
    - Réponse en temps réel
    - Flux de données élevés
    - Traitement parallèles



- **Intérêt majeur de l'architecture RIO (Flex RIO, Compact RIO, Single board RIO)**
  - Communication SW/FW & FW/Module I/O → « Déjà écrite & Opérationnelle ! »

Specifications …

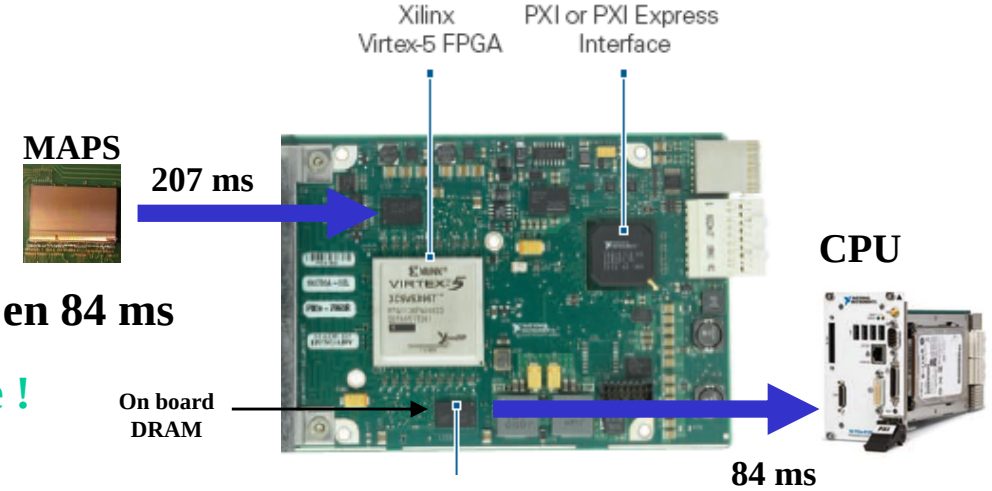
Specifications du DAQ pour le Telescope EUDET

- 0 % temps mort → Latence bus PXIe & SW ?
- Minimiser le temps de développement du FW
  - Car disponibilité limité d'un Ingénieur FW

Approche pragmatique …

Approche pragmatique → Etudier le HW Flex RIO avant de se lancer ...

- Latence → Bufferiser sur la carte
  - Flex RIO a deux blocs indépendant de DRAM 512 MB
  - Le MAPS rempli un block DRAM / CPU lit l'autre
  - Acquisition de 1800 trames 25 MB en 207 ms / Lecture en 84 ms
    - → 123 ms marge / 207 ms → 60 % temps CPU libre !



- Temps de développement du FW → Temps de développement du SW (C dans DLL)
  - Garder le FW aussi simple que possible : Deserialisation + Trigger → Tâches HW simples
  - Déplacer le traitement vers le SW : Frames sizing + Event building → Tâches SW +/- complexes

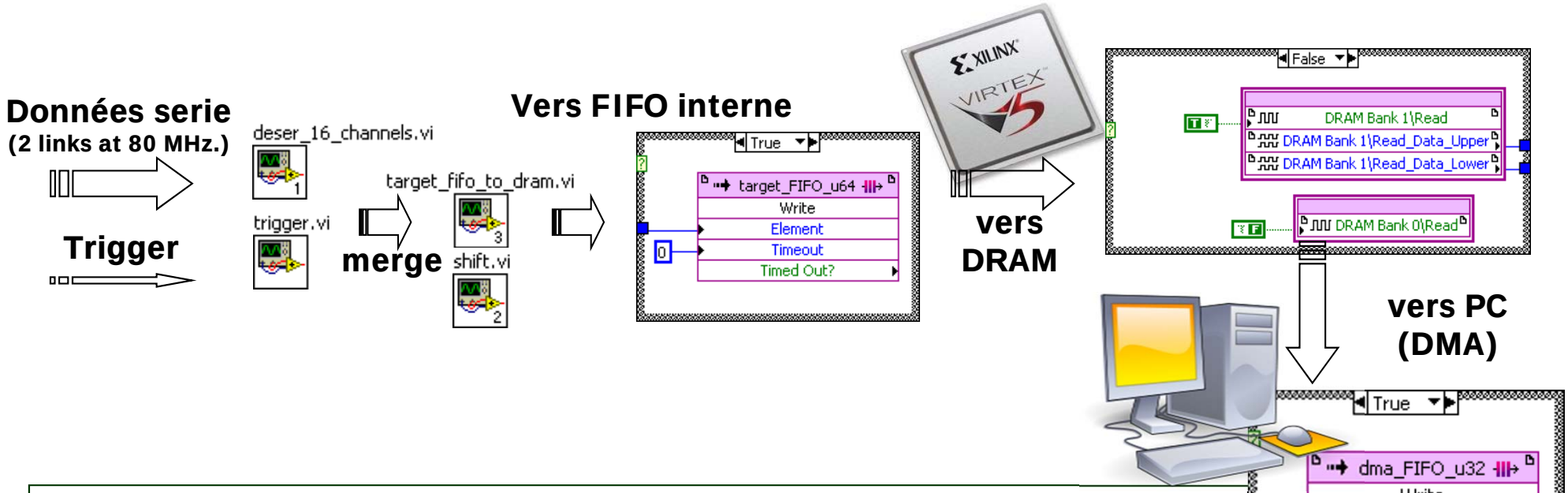
Télescope EUDET : FW DAQ vue générale

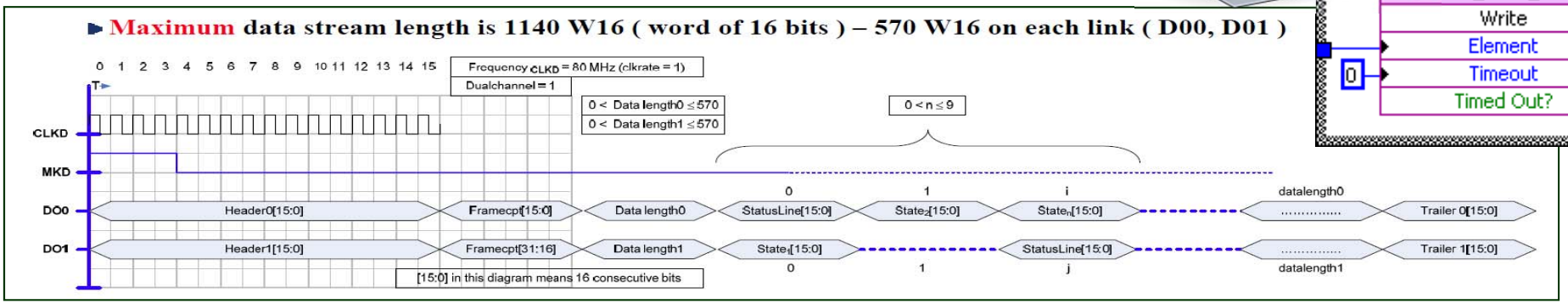
Firmware développé par l'utilisateur (Nous !)

- Désérialisation des données (6 Capteurs lus en //)
- Swap entre deux blocs de DRAM (Réduire le risque de temps mort SW)
  - Mimosa 26 remplit une RAM / Le bus PXIe via le Software vide l'autre
- Première version développée en LabView FPGA → Pour l'exercice de style ;-)

C. Santos (FW)
Actuellement APC

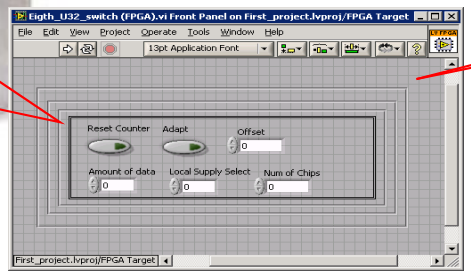
G. Claus (SW)



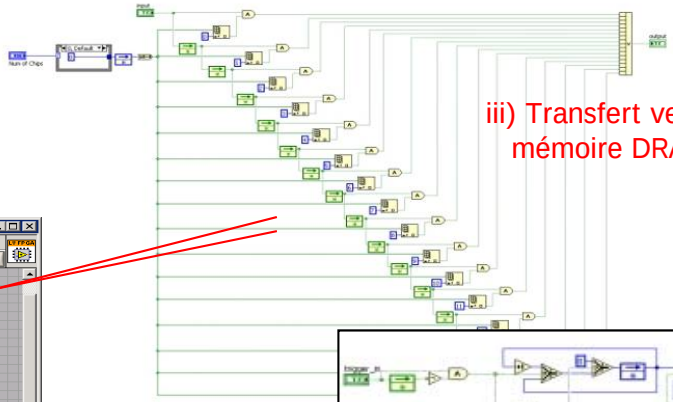


Télescope EUDET : FW DAQ vue détaillée …

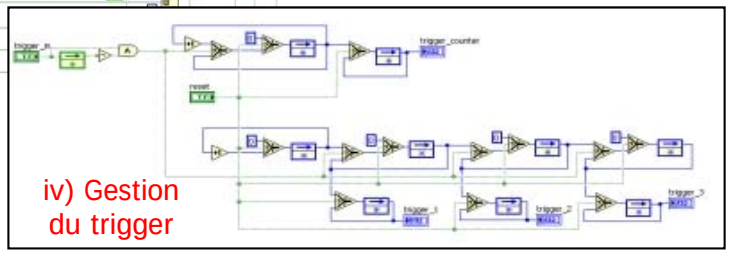
FPGA vi



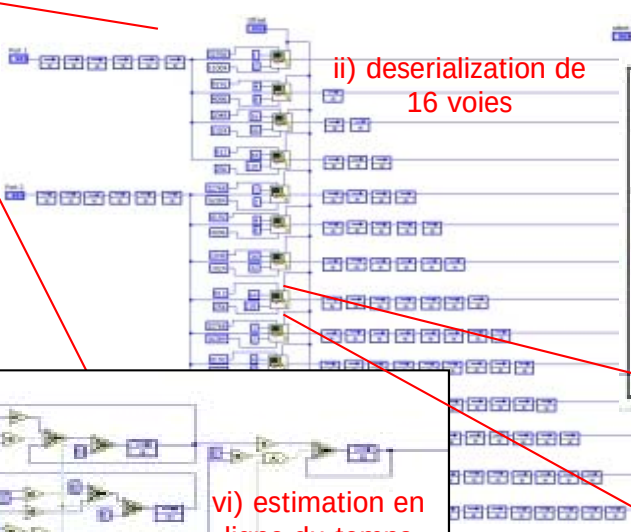
deser_16_channels.vi



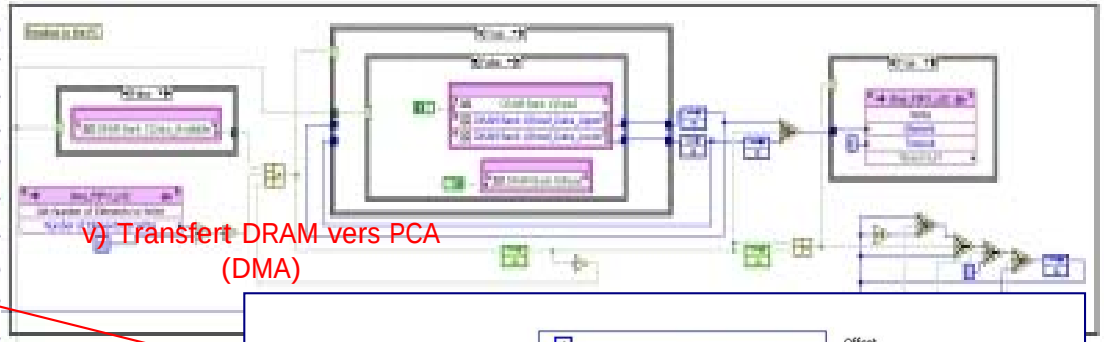
iii) Transfert vers la mémoire DRAM



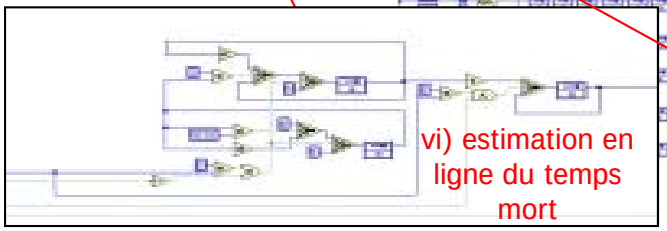
iv) Gestion du trigger



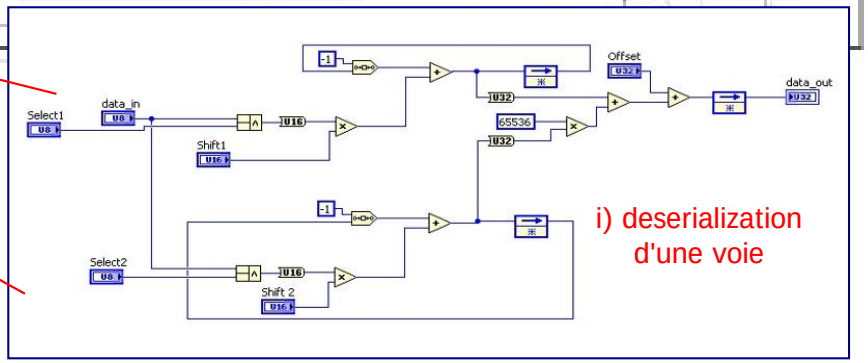
ii) deserialization de 16 voies



v) Transfert DRAM vers PCA (DMA)



vi) estimation en ligne du temps mort



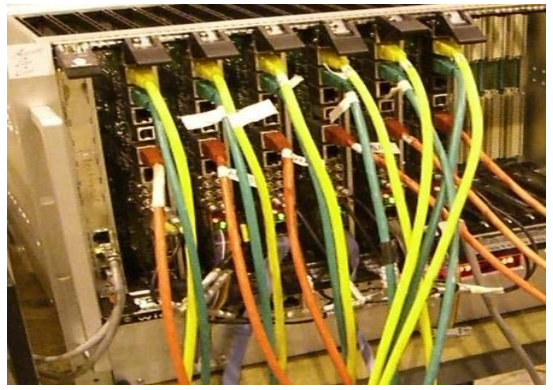
i) deserialization d'une voie

Télescope EUDET : DAQ Performances & Coût

- **Flex RIO répond aux besoins du projet avec de la marge de sécurité**
  - Pas de temps mort → 60 % « temps libre » – FPGA rempli à ~ 55 %
  - Upgrades du système → Upgrades SW (facile) / FW ne changera pas

- **Coût Flex RIO PXIe (NI) / Carte EUDRB VME (INFN Ferrara)**
  - Solution EUDRB → ~ 22 k€
    - 6 Carte EUDRB → 6 x EUDRB = 6 x 2 k€ = 12 k€
    - 1 Châssis VME ~ 5 k€
    - 1 Carte CPU ~ 5 k€
  - Solution Flex RIO → ~ 15 k€
    - 1 Carte Flex RIO + NI 6585 → 7 k€
    - 1 Châssis PXIe ~ 3 k€
    - 1 Carte CPU ~ 5 k€
  - Conclusion
    - Solution Flex RIO réduit le coût du HW de 30 % (7 k€)
    - Pas besoin de « Manpower » pour la production et le test des cartes

Carte EUDRB VME (INFN)
1 Carte / Mimosa 26 nécessaire



DAQ INFN Telescope EUDET
VME - 6 Cartes EUDRB

DAQ IPHC – NI Telescope EUDET
PXIe - 1 Carte Flex RIO + NI6585

Une carte Flex RIO peut acquérir jusqu'à 16 x Mimosa 26
→ 440 € / Mimosa 26

Télescope EUDET : DAQ Temps de développement & Transfert

- **Temps de développement du DAQ Flex RIO / EUDRB**
  - **Evaluation FlexRIO + Développement DAQ FW, SW ➔ ~ 9 mois ETP (Aucun dév HW)**
  - **Développement de la carte EUDRB (HW, FW, SW) ➔ ~ 24 mois ETP (Dont beaucoup de dév HW)**
    - **Première version : Carte mère VME + mezzanine analogique (Telescope Analogique)**
    - **Seconde version : Carte mère VME + mezzanine numérique (Telescope EUDET final)**
  - **Conclusion : Flex RIO 9 ETP / EUDRB 24 ETP**
    - **On économise le temps de développement du HW**
    - **Le développement du FW n'est pas forcément plus rapide avec Flex RIO et ce n'est ni une surprise, ni nécessaire**

- **Transfert à la collaboration EUDET**
  - **IPHC : Documentation + Organisation d'une semaine de formation ➔ ~ 1 mois ETP**
    - **FW LabVIEW FPGA fournit « as is » sans documentation ➔ Upgrade FW par l'IPHC si nécessaire**
    - **SW LabVIEW et code C, documenté de A à Z, formation ➔ Upgrade SW par la collaboration EUDET**
  - **EUDET (DESY – Hambourg) : Intégration & Déploiement des Télescopes**
    - **Intégration du DAQ Flex RIO dans le superviseur DAQ EUDET**
    - **Assemblage, validation et mise en service des Télescopes**

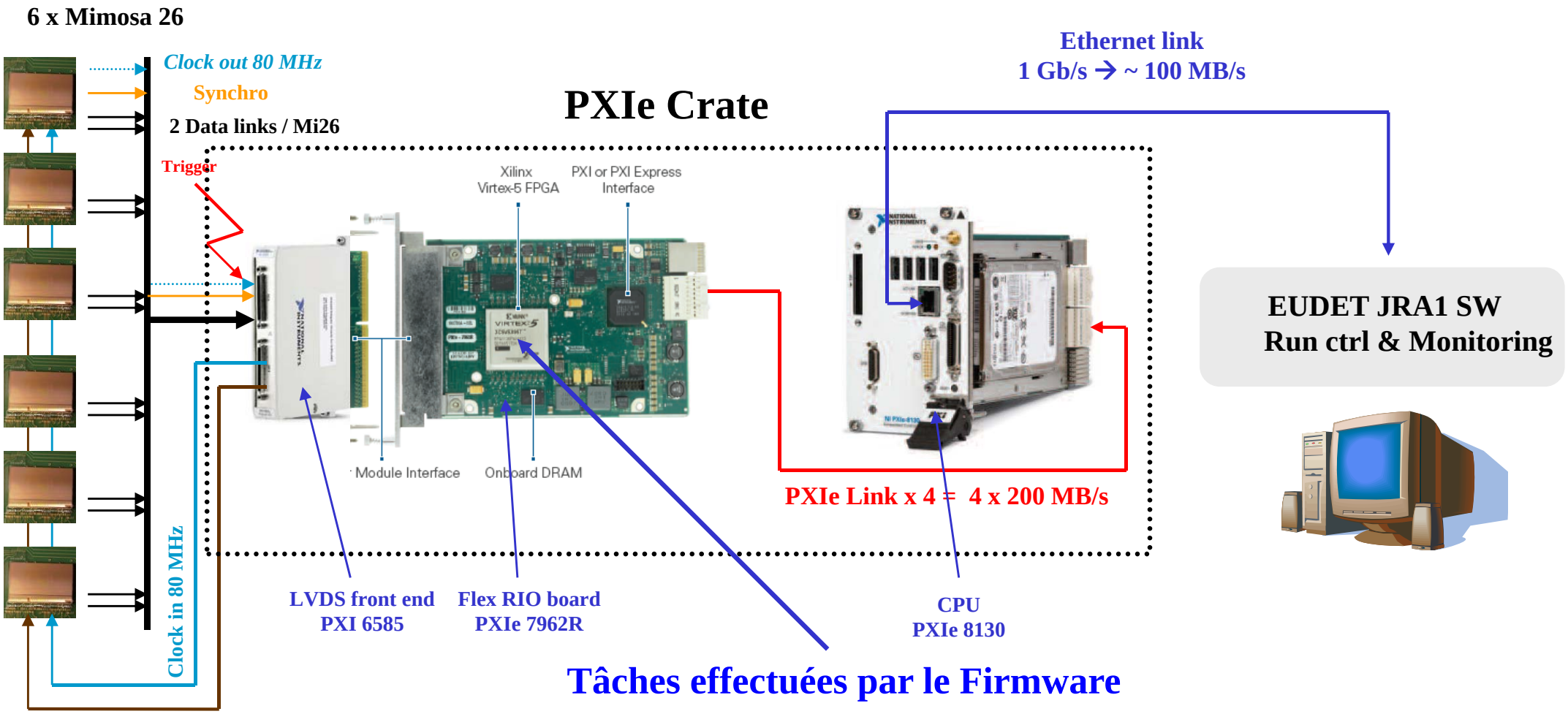
Télescope EUDET : Conclusion

- La solution Flex RIO a permis de ☺
  - D'atteindre les performances requises : Temps mort = 0 % - 60 % Temps CPU dispo
  - De développer un DAQ en 9 mois ETP (Ceci en // d'autres activités ...)
  - Réduire le coût du DAQ de 30 % (22 k€ à 15 k€)
  - S'affranchir du « Manpower » pour le suivi de production et le test de cartes « Custom »
  - De créer des copie du Télescope EUDET ... 6 Télescopes mis en service depuis 2011
    - DESY Hambourg 2 x, CERN Genève 2 x, Bonn, Ottawa

Télescope de faisceau EUDET
6 x Mimosa 26

- A l'IPHC déployer 10 DAQ clef en main pour les capteurs Mimosa 26 / Mimosa 28
  - Hambourg PLUME, Hambourg SALAT, CEA Saclay, IPNL Lyon, Univ Colombia, LBNL Brookhaven, Univ Aarhus, GSI Darmstadt 2 x, Strasbourg IMABIO

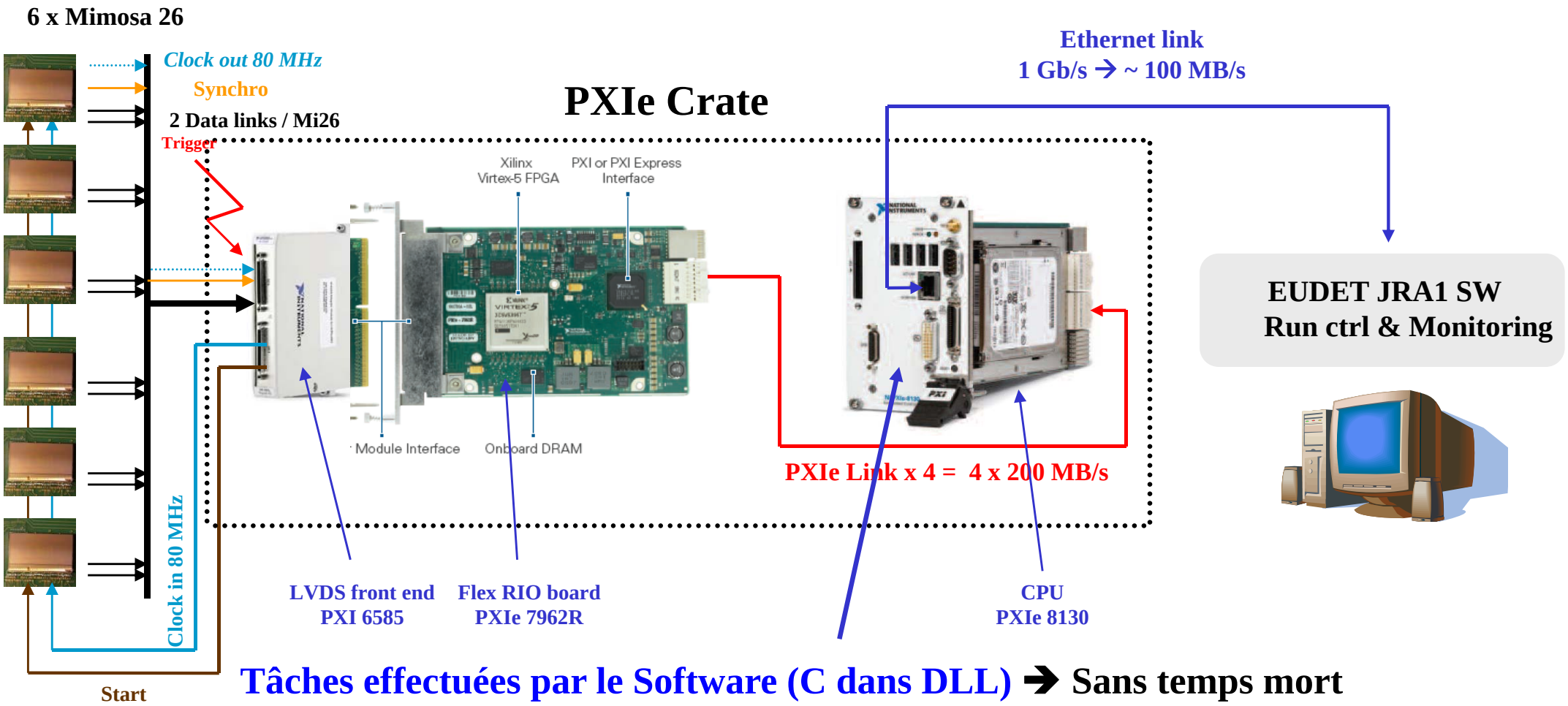
FW / SW : Tâches du FW DAQ EUDET



Tâches effectuées par le Firmware

- **Désérialise les données** (12 x W16 deserializer) → Indépendant du format header, trailer … → **Flux de W16**
- **Acquiert toutes les trames** → Ne se préoccupe pas du trigger (Ce sera le job du SW)
- **Acquiert toute la longueur des trames** → Ne tient pas compte du champ « Data length » (Assez de BP PXIe)
- **Stocke la position du trigger** (No line de Mi26) → Jusqu'à 288 triggers / trame
- **Stocke le compteur de triggers** fournit par la TLU (Trigger Logic Unit) → Jusqu'à 288 triggers / trame

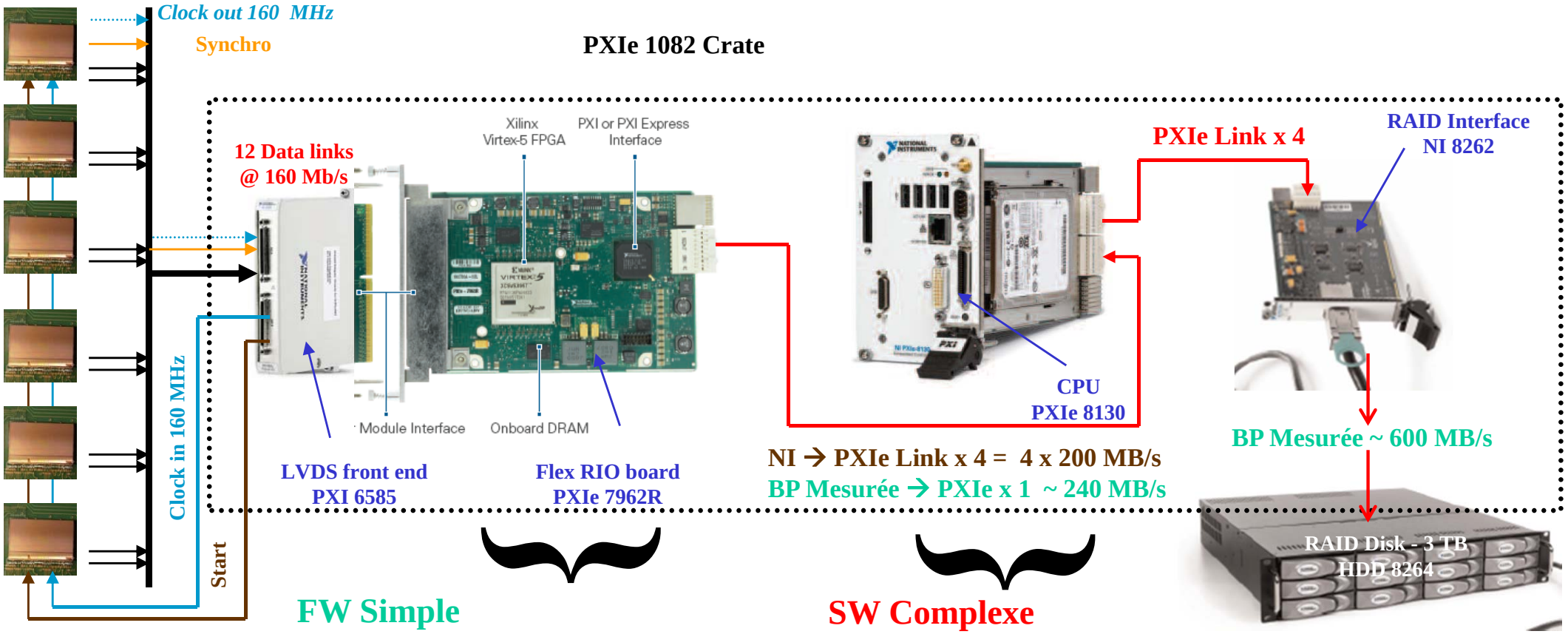
FW / SW : Tâche du SW DAQ EUDET



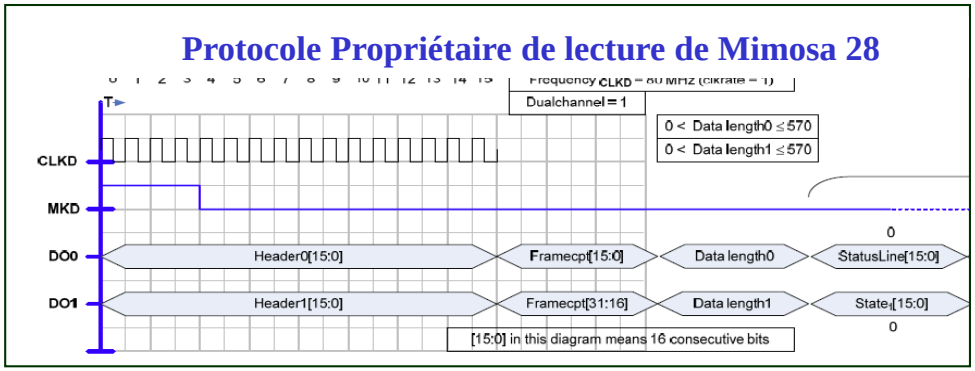
Tâches effectuées par le Software (C dans DLL) → Sans temps mort

- Extrait SEULEMENT les trames avec trigger du flux de données généré par la Flex RIO (8680 trames/s)
- Coupe les trames à la longueur utile → Définie par le champ « Data Length » de la trame de Mimosa 26
- Créé des enregistrements de taille variable → Sauvegarde sur disque / Emission sur Ethernet
- 2010 EUDET : 6 x Mi 26 = 1 Flex RIO + CPU PXIe 8130 → 120 MB/s traité par SW Temps mort 0 %, 60 % CPU libre
- 2016 BEAST : 24 x Mi26 = 2 Flex RIO + CPU PXIe 8133 → 480 MB/s traité par SW Temps mort 8 %, 70 % CPU libre

FW / SW : Le bon compromis ➔ Procéder par étapes



- Désérialise data & Stocke Triggers
- Détecte Triggers
- Extrait la partie utile des trames @ 8 680 trame/s



Procéder par étapes

1. **Valider** les traitement complexes ➔ **SW**
2. **Migrer** ces traitements ➔ **FW**

L'environnement LabView FPGA est incontournable
Mais il est possible d'intégrer du code VHDL dans le projet

- HDL Interface node
  - Limité à quelques dizaines de lignes
- User define CLIP
  - Pas de limitation en nombre de ligne
  - Possibilité de spécifier des contraintes de timing
- Socketed CLIP
  - Interface HDL vers périphériques externe
- IP Intégration node
  - Outil d'importation d'IP

K. Jaaskelainen

... Questions ?

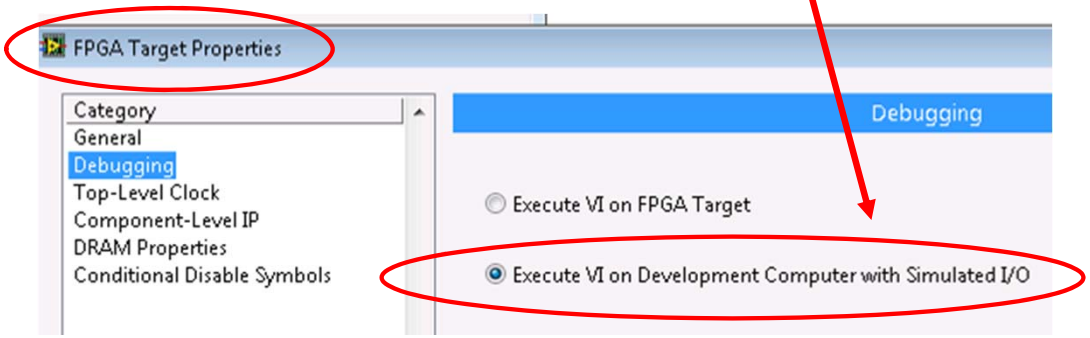
➔ Kimmo JAASKELAINEN

- On peut transposer certaines méthodes de travail de LabView à LabView FPGA
  - Couche supérieure et Interface vers matériel ➔ LabView soft ➔ LabView FPGA
  - Traitement complexes ou critiques en temps d'exécution ➔ C dans une DLL ➔ VHDL

FW / SW : FW ➔ Comment debugger ?

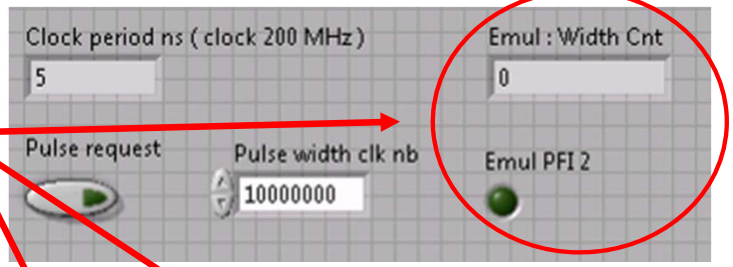
➔ Simulation fonctionnelle via Emulateur

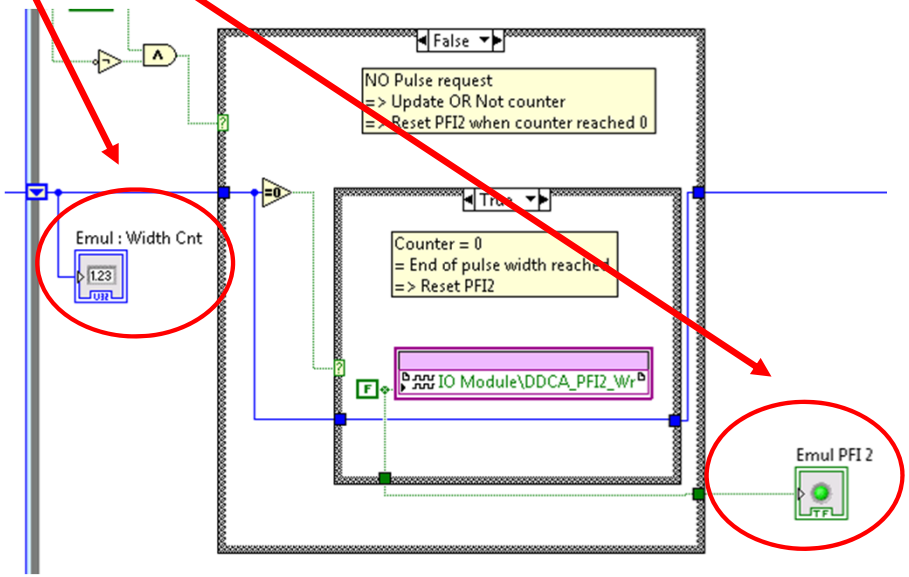
- Ajouter des Indicateurs de debug dans VI FPGA
- Sélectionner le mode « Execute VI on development computer »



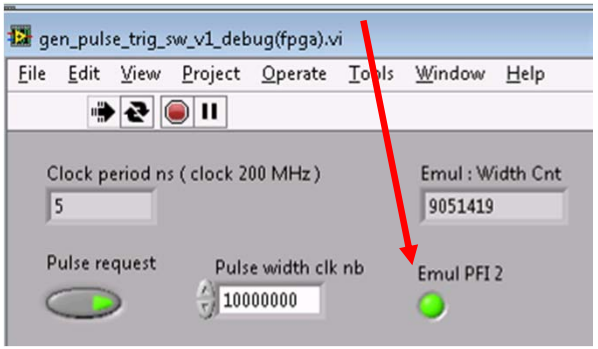
- Emuler le fonctionnement
  - Outils de debuggage LabVIEW classique
    - Points d'arrêts
    - Mode pas à pas

Générateur d'impulsions

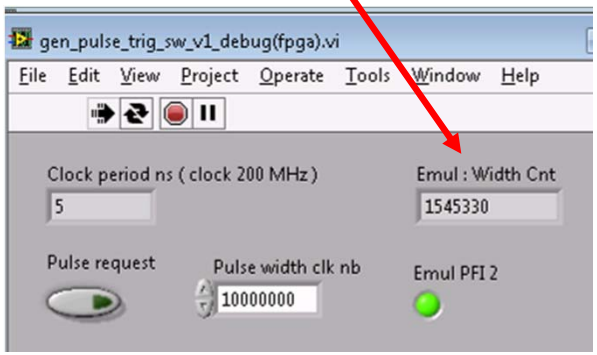




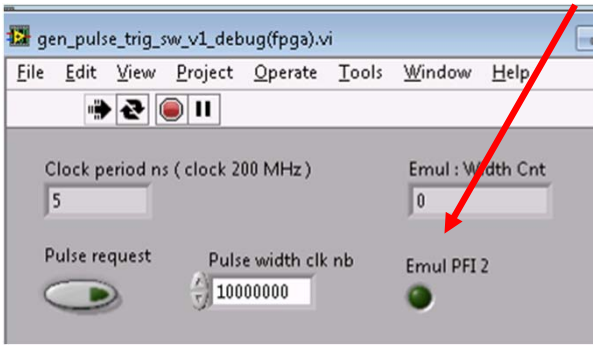
Emulation : Pulse request ➔ Set PFI



Emulation : Décomptage



Emulation : Fin d'impulsion ➔ Reset PFI

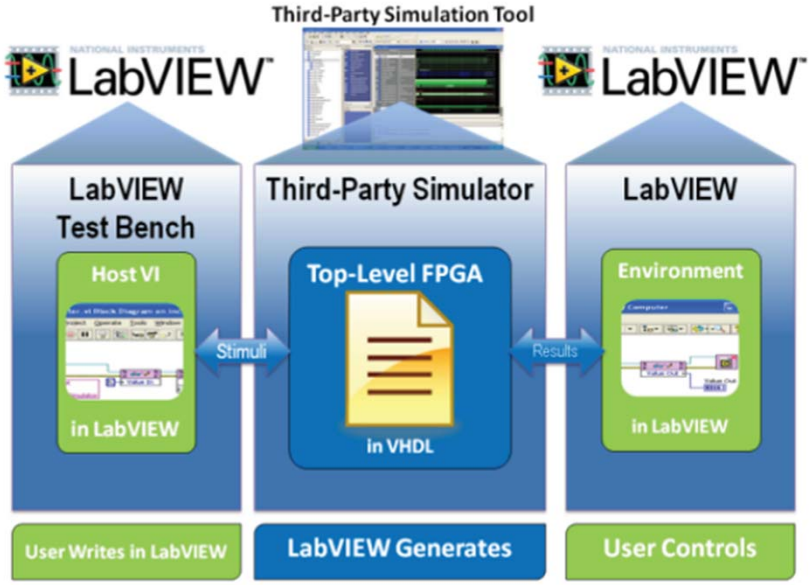


FW / SW : FW ➔ Comment simuler le code LV FPGA ?

Pourquoi simuler ➔ Gagner du temps (Compilation > 30 min …) – NI ➔ Simulation « Cycle accurate »

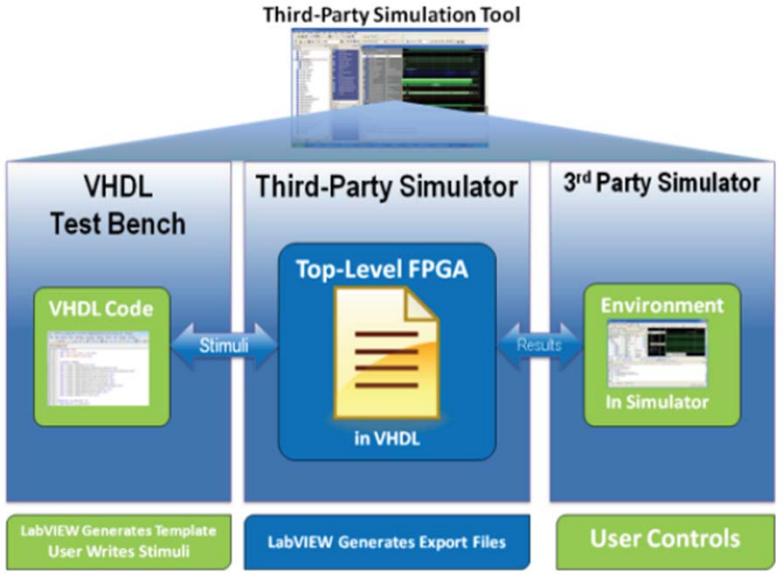
- Simulation « Cycle accurate » (WP 12917) ➔ http://www.ni.com/white-paper/12917/en
- Co-simulation avec mentor Graphics (WP 11574) ➔ http://www.ni.com/white-paper/11574/en
- Méthode d'exportation pour simulation avec Xilinx ISim (WP 12942) ➔ http://www.ni.com/white-paper/12942/en

Co-Simulation avec simulateur tiers



- Test vectors & Test results ➔ LabVIEW
- Simulation ➔ ModelSim & Questa (Mentor Graphics)
- Pas de connaissances VHDL requises

Export vers simulateur tiers



- Test vectors & Test results ➔ VHDL (ou autre HDL)
- Simulation ➔ Mentor Graphics + ISim (Xilinx)
- Maîtrise VHDL requise

Choix de développement du FW

- **Première démo du FW développée en 2009 avec LabVIEW FPGA 2009**

- **Premier FW déployé en 2010 LabVIEW FPGA 2009 + 8 x CLIP pour coder en VHDL**
  - **Désérialisation d'un Mimosa 26**
  - **Gestion du trigger**
  - **Gestion des domaines d'horloges multiples**
  - **etc … ➔ vers un code LabVIEW FPGA minimal**

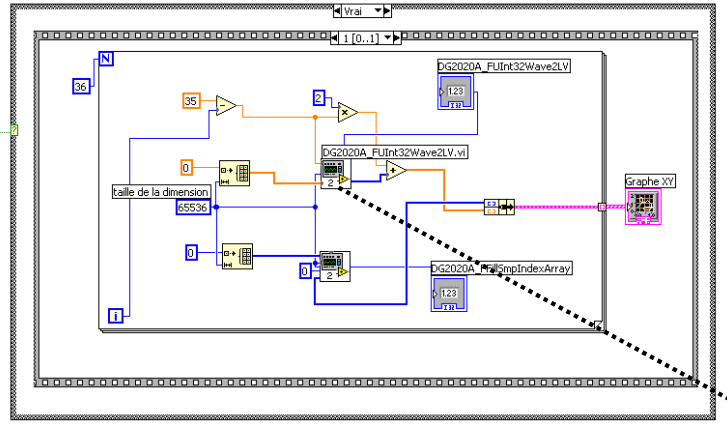
C. Santos
Actuellement APC

- **FW utilisé « as is » jusqu'en 2015**
  - **Seuls des upgrades SW ont été nécessaire – Comme prévu ☺**

- **Second FW développé en 2015 LabVIEW FPGA 2014 + CLIP pour coder en VHDL**
  - **Futur projets ➔ Upgrade FW probablement requis – Nouveau Ingénieur FW**
  - **Co-Simulation LabVIEW FPGA + Mentor Graphics Questa SIM**
  - **Upgrade des IPs XILINX obsolètes (COREGEN) ISE10.1 ➔ ISE 14.7**
  - **Emulation de MAPS et de la TLU dans l'environnement de simulation**

K. Jaaskelainen

Choix de développement du SW



LabView / C – JAVA (ou autre …) ?

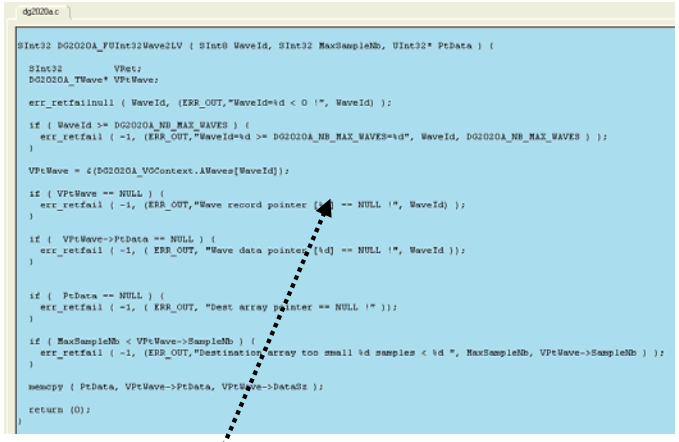
Graphique / Textuel ?

Un schéma / Un programme ?

Lenteur / Efficacité d'exécution ?

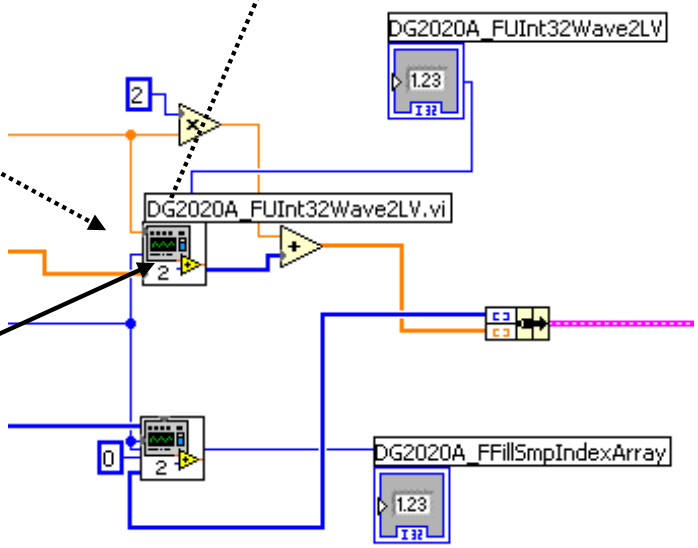
Efficacité / Lenteur de développement ?

Code « Jetable » / Réutilisable ?



Notre philosophie …

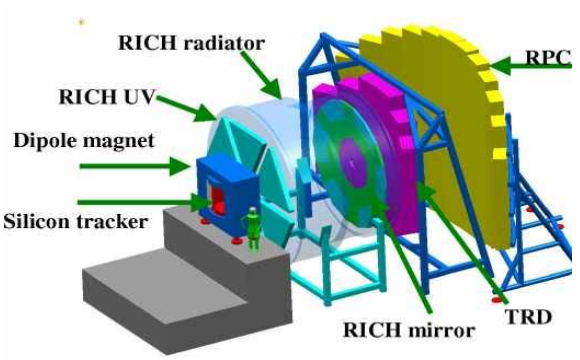
- Labview
  - Interface graphique (GUI)
  - Driver des cartes (Si fournis par constructeur → Pas pour les écrire !)
  - Traitements : FFT, etc … (Si ils existent → Pas pour les écrire !)
- Le langage C
  - Pourquoi le C plus qu'un autre ? … → Pour sa portabilité
  - Pour le code « complexe » → Traitement des données
    - Efficacité du code (temps d'exécution)
    - Lisibilité – Maintenance – Réutilisabilité du code
- Une solution : Encapsuler du C dans LabView
  - Appels de fonctions C via DLL dans LabView
  - LabWindows CVI



Code LabView
contenant des appels de fonctions C via DLL

Nos besoins futurs : COTS / « Custom HW »

- **Liens série rapides**
  - Bit rate >= 1 Gb/s + Horloge embarquée dans les donnéees ➔ Protocole 8B/10B
  - Exemple MAPS développé pour CBM (Darmstadt GE) ➔ 3 liens à ~ 1 Gb/s
- **Solutions NI**
  - Flex RIO + Adapter module ➔ Pas de solution
    - NI 6587 – 16 I/O – 1 Gb/s (500 MHz DDR) – Ne gère pas le 8B/10B
    - Coût Flex RIO ~ 6000 € + Adapter module ~ 2000 €
  - PXIe-6591R ➔ Coût 15 400 € (Mais DAQ manip 640 Mimosa 26 ➔ 24 €/ Mi 26)
    - 8 Voies - 500 Mbps – 12,5 Gbps (2 Connecteurs Mini – SAS HD)
  - PXIe-6592R ➔ Coût 11 300 € (Mais DAQ manip 256 Mimosa 26 ➔ 44 €/ Mi 26)
    - 4 Voies – 500 Mbps – 10 Gbps (4 Connecteurs SFP+)



CBM (Compressed Baryonic Matter) > 2018

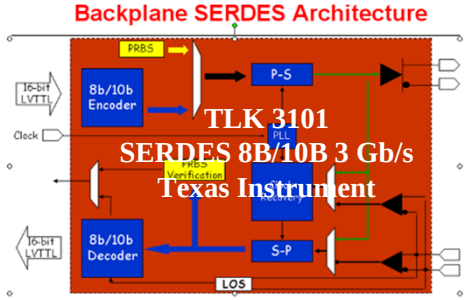
PXIe 6591 R

PXIe 6592 R

- **Solution COTS CERN / « Custom HW »**

Custom HW / COTS : Question tranchée ou compromis permanent ?
Notre expérience dans les collaborations EUDET, AIDA, BEAST ...

  - CERN – GBT (Giga Bit Tranceiver), GLIB (GBT Link Interface Board), CTA (CMS Tracker AMC) - µTCA
  - Carte « custom » TLK 3101 3 Gb/s pour Flex RIO + NI6585
    - Carte développée, testée en mode « loop back »
    - Reste à développer le FW côté Flex RIO



Conclusion – Custom HW / COTS NI Flex RIO

- **Les points forts**
  - **Le coût des COTS NI est similaire ou inférieur / « Custom HW » ➔ DAQ Télescope EUDET coût réduit de 30 %**
  - **La duplication du HW DAQ ne coute pas de RH (Pas de suivi de production, de test des cartes, de dépannage)**
  - **Maintenir des compétences en développement FW dans le groupe**
    - **Grâce au temps économisé sur le développement HW, le suivi de production, le test des cartes**
  - **La conception FW peut être effectuée en VHDL & LabVIEW FPGA ➔ Augmente le nb de développeurs potentiels**
  - **Le débit sur le bus PXIe (n x 250 MB/s) et la puissance CPU permettent de**
    - **Implémenter par SW des fonctions « en principe » réservées au FW**
    - **Réduire le temps de développement – Implémenter ensuite ces fonctions par FW pour gagner en performances**
- **Les limites**
  - **Point de vue HW**
    - **Pas de possibilité d'utiliser les MGT (Multi Gigabit Transceiver) sur la famille de cartes Flex RIO Virtex 5**
    - **Pas d'adapter module 8B/10 – Coût de la ligne de produits PXIe 6591 & 6192 trop élévé / Nos applications**
  - **Point de vue développement**
    - **Le temps de compilation (~ 30 minutes projet simple) ➔ Intérêt de la Co-simulation – Source VHDL cryptés**
    - **Windows / Linux**

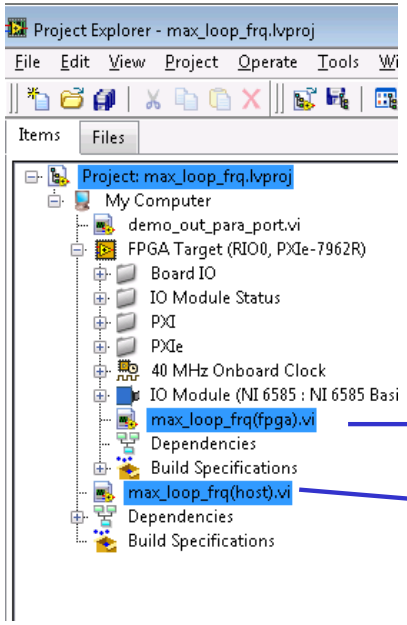
Backup slides

- Quelques exemples de code LabVIEW FPGA
  - Démo No 1 : La procédure de A à Z (P31)
  - Démo No 2 : Ecriture / Lecture d'un registre (P36)
  - Démo No 25 : Sérialiseur 16 bits 200 MHz codé en LabVIEW FPGA (P38)
  - Démo No 26 : Sérialiseur 16 bits 200 MHz codé en VHDL via user CLIP (P41)

- La R&D PICSEL (P48)
- Les étapes de la caractérisation d'un MAPS (P49)
- L'équipe Test (P50)
- Les bancs de caractérisation de MAPS basés sur des DAQ COTS (NI) (P51)
- Les détecteurs MAPS équipés de DAQ des COTS (NI Flex RIO) (P52)
- Les limites du compromis FW / SW (P53)
- Le DAQ du projet BEAST (P54-56)
- Acquisition de liens série > 1 Gbps – GBT / GLIB (P57-59)
- Une autre comparaison « Custom HW » / COTS - DAQ pour bancs de test labo (P60-P62)

Démo No 1 : Comment concevoir SW et FW ?

Explorateur de projet



Projet ➔ Contient Deux VIs

- SW ➔ max_loop_frq(host).vi (LabVIEW)
- FW ➔ max_loop_frq(fpga).vi (LabVIEW FPGA)
  - Code LabVIEW FPGA ➔ VHDL
  - Compilation = Outils Xilinx

Répertoire démo ➔ 1__max_loop_frq

Firmware LabVIEW FPGA

Face avant FW

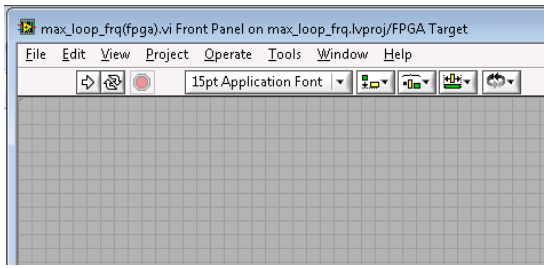
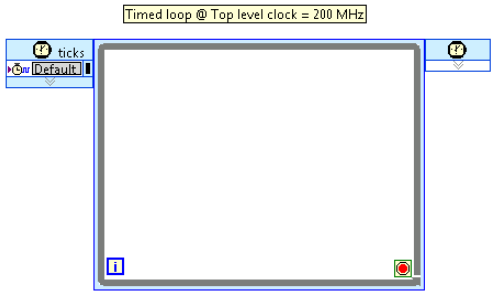


Diagramme FW



Software LabVIEW

Face avant SW

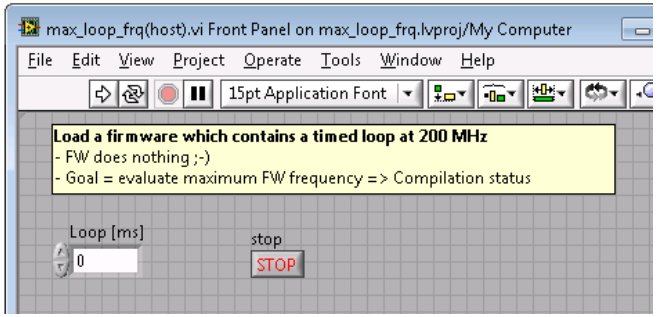
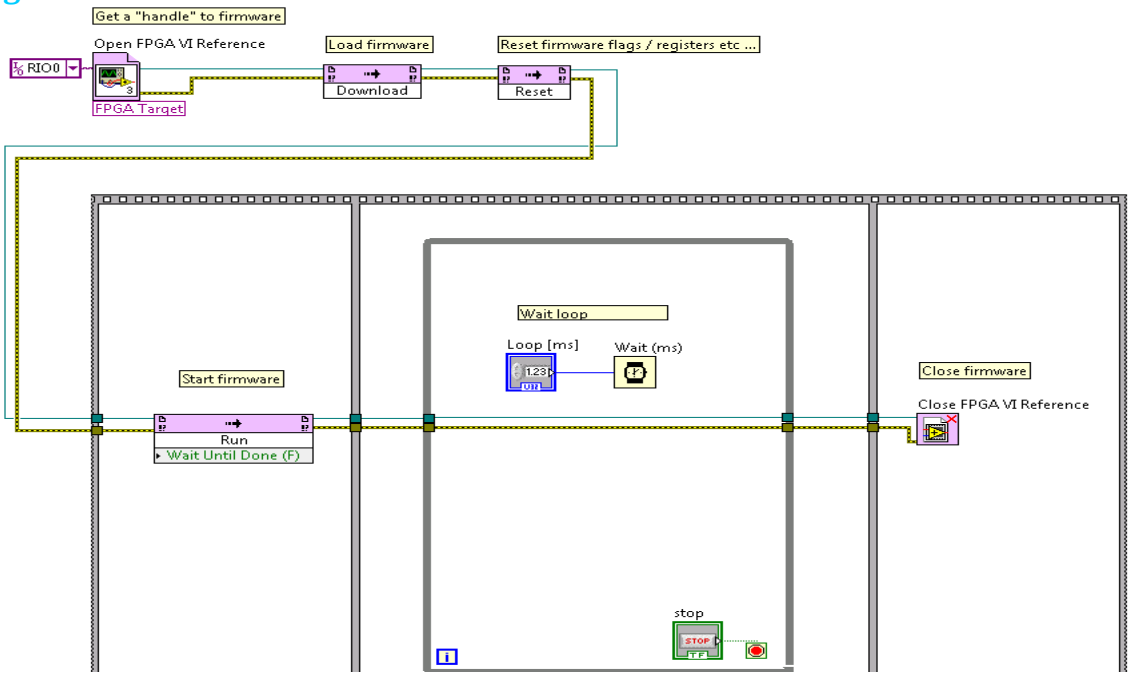


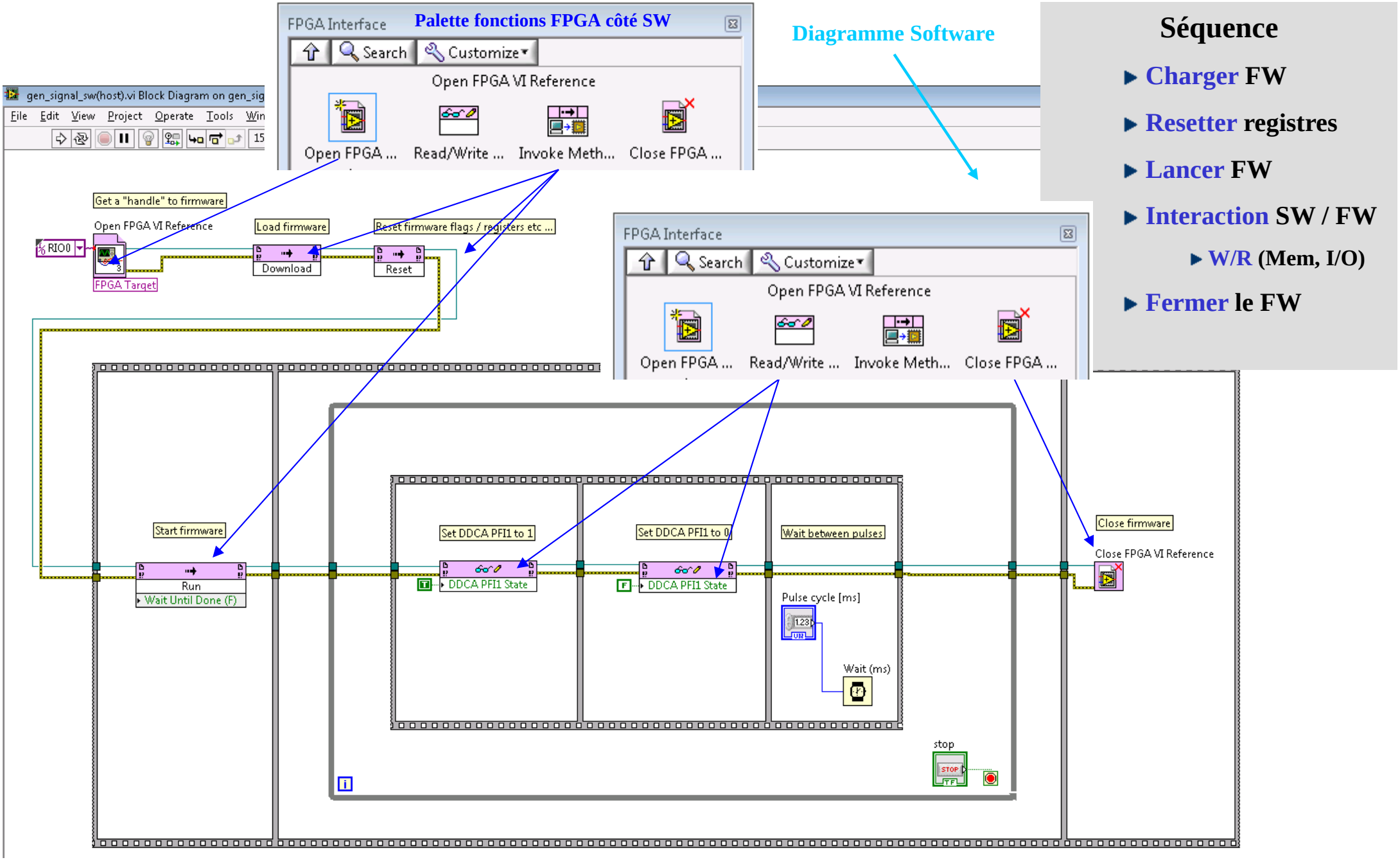
Diagramme SW



Mêmes outils de conception côté FW que SW

- Face avant ➔ Palette Control & Indicateurs
- Diagramme ➔ Palette de fonctions
  - Mais fonctionnalités limitées côté FW / SW

Démo No 1 : Comment le SW pilote le FW ?

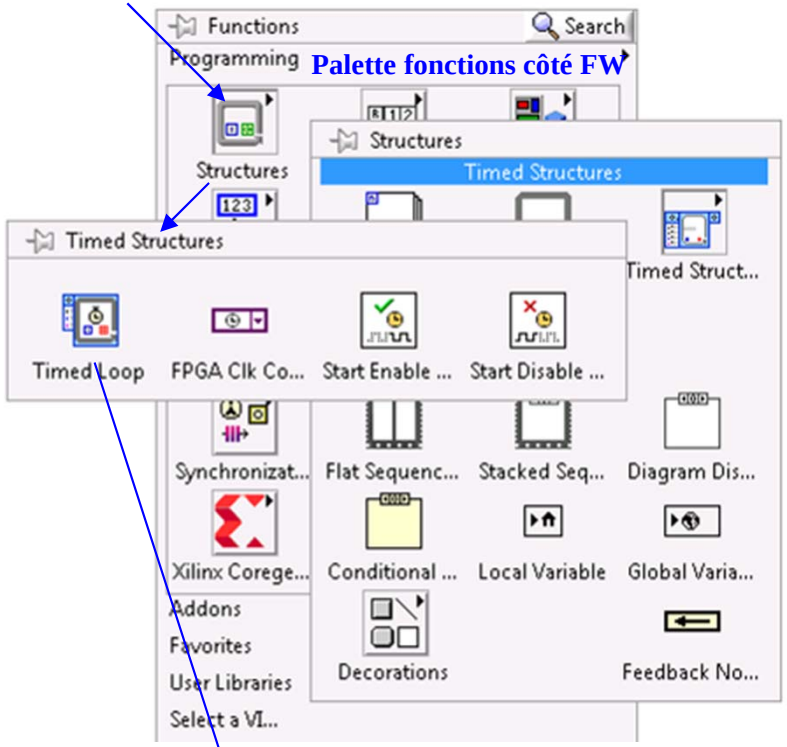


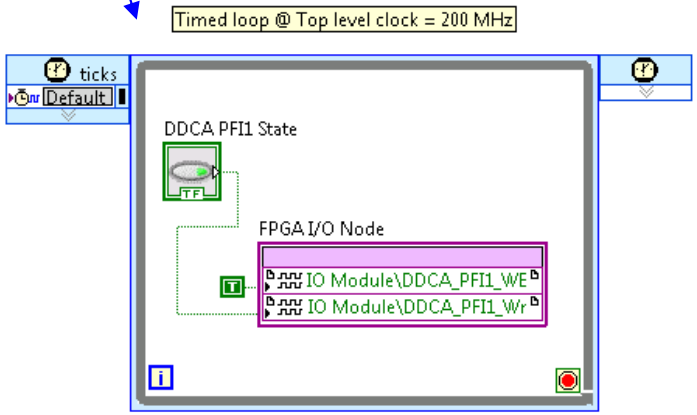
Séquence

- **Charger** FW
- **Resetter** registres
- **Lancer** FW
- **Interaction** SW / FW
  - **W/R** (Mem, I/O)
- **Fermer** le FW

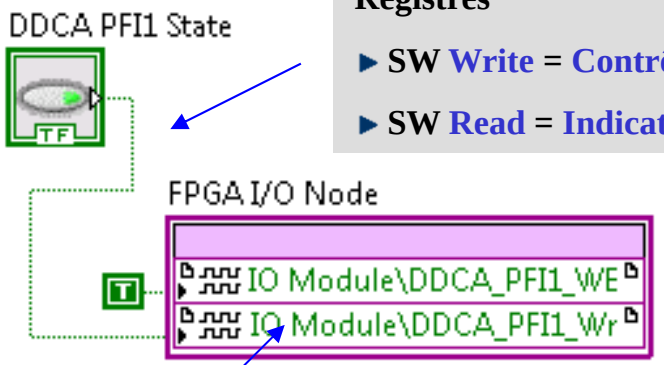
Démo No 1 : Comment écrire le code FW ?

Structures de contrôle d'exécution
« boucles, tests, etc » +/- comme en SW





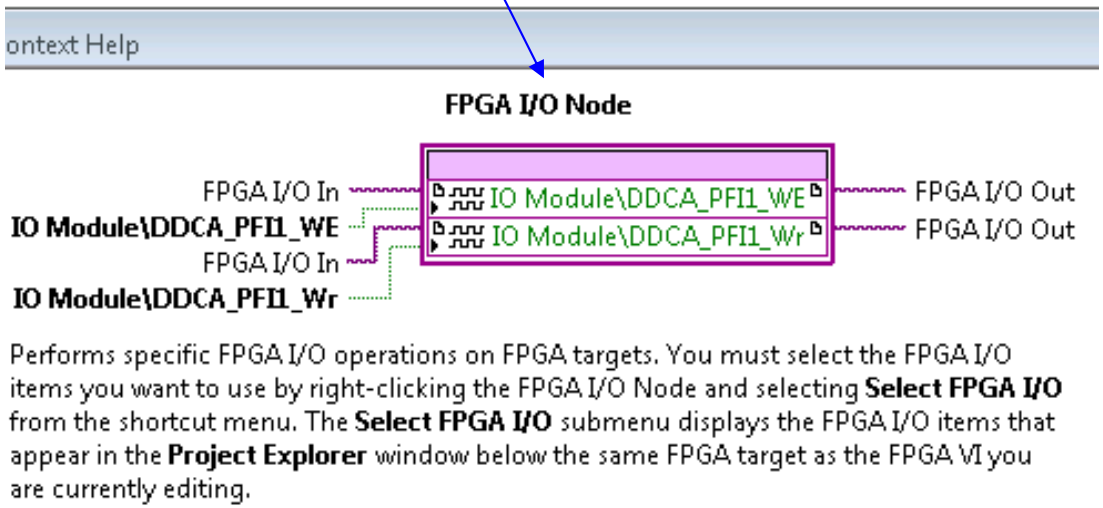
Des Entrées / Sorties vers le matériel
Registres, Port I/O, Mémoires, etc …



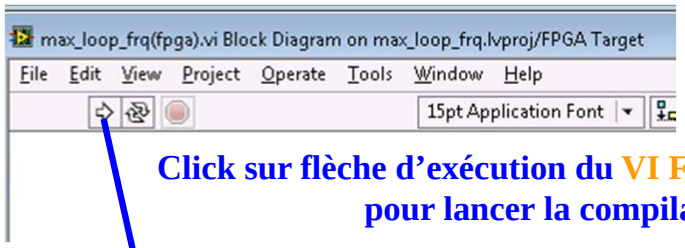
Registres
- SW **Write** = **Contrôles**
- SW **Read** = **Indicateurs**

Nœuds Entrée / Sortie
- **Fixent direction des I/O : In / Out**
- **Ecrire & Lire les I/O**

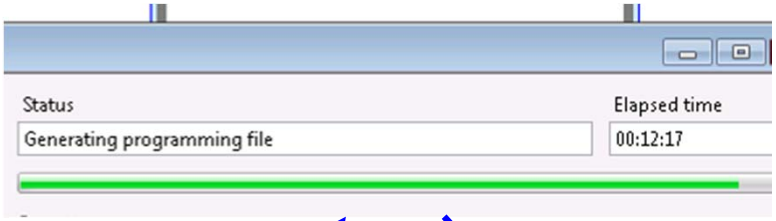
Autres composants I/O (Fonction du HW)
- **FPGA I/O Property node**
- **FPGA I/O Method node**



Démo No 1 : Compilation → Exécute les outils Xilinx pour vous ☺



Click sur flèche d'exécution du VI FW pour lancer la compilation



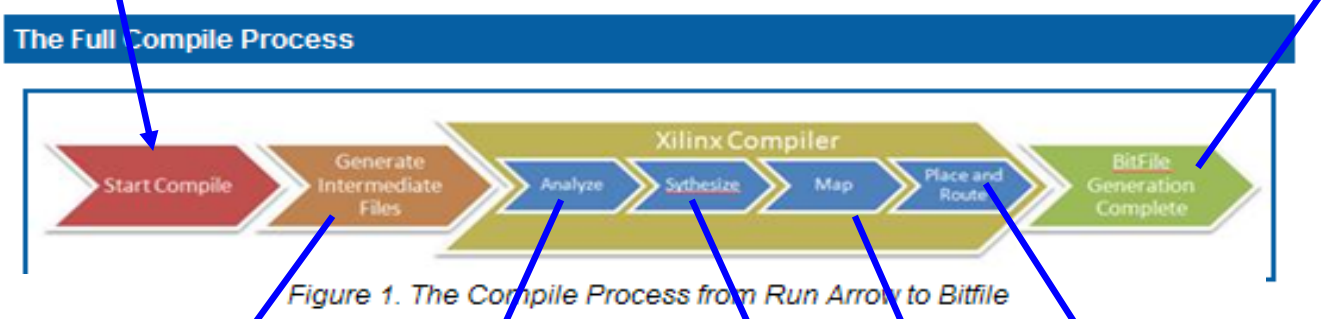
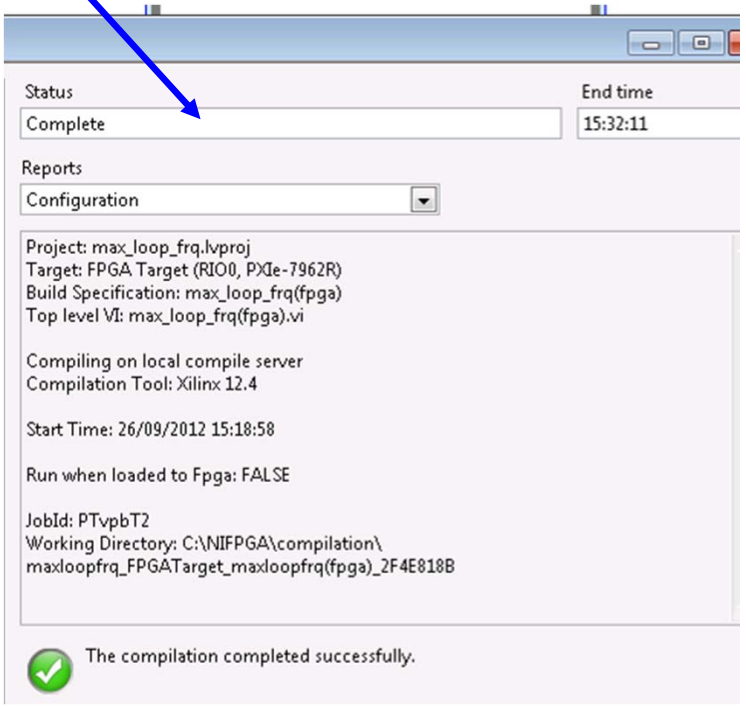
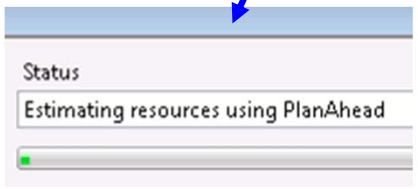
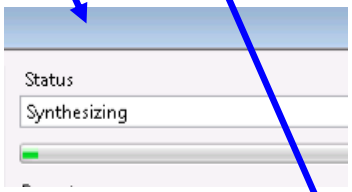
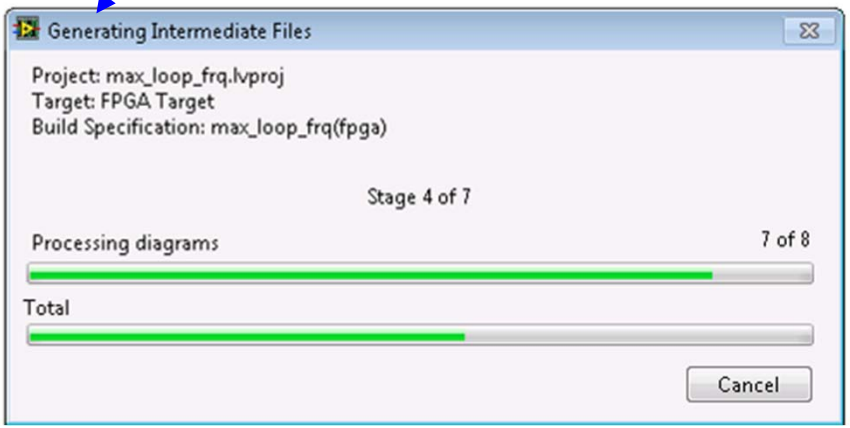


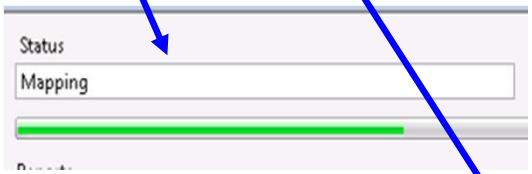
Figure 1. The Compile Process from Run Arrow to Bitfile

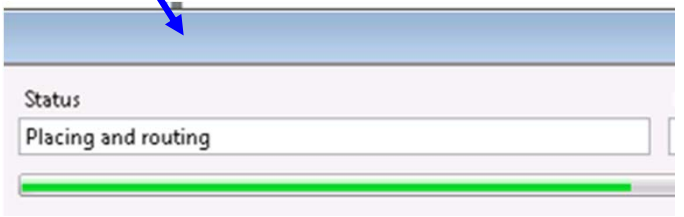






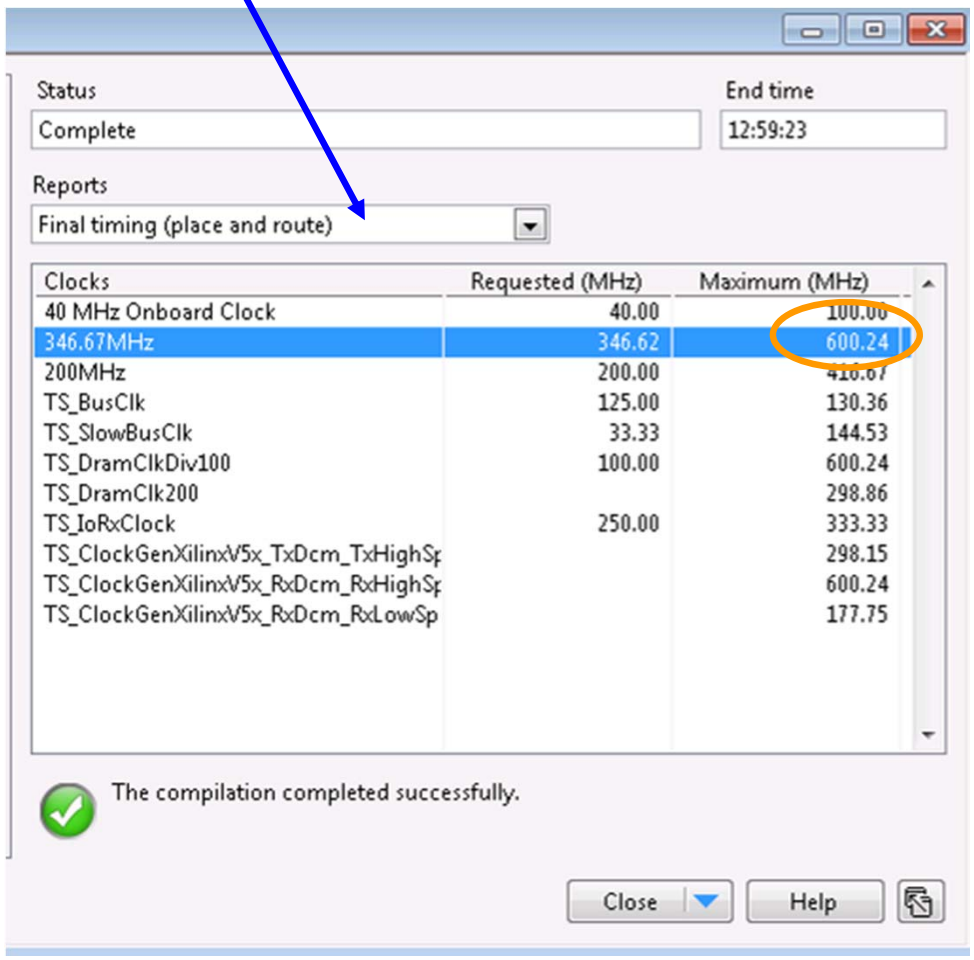






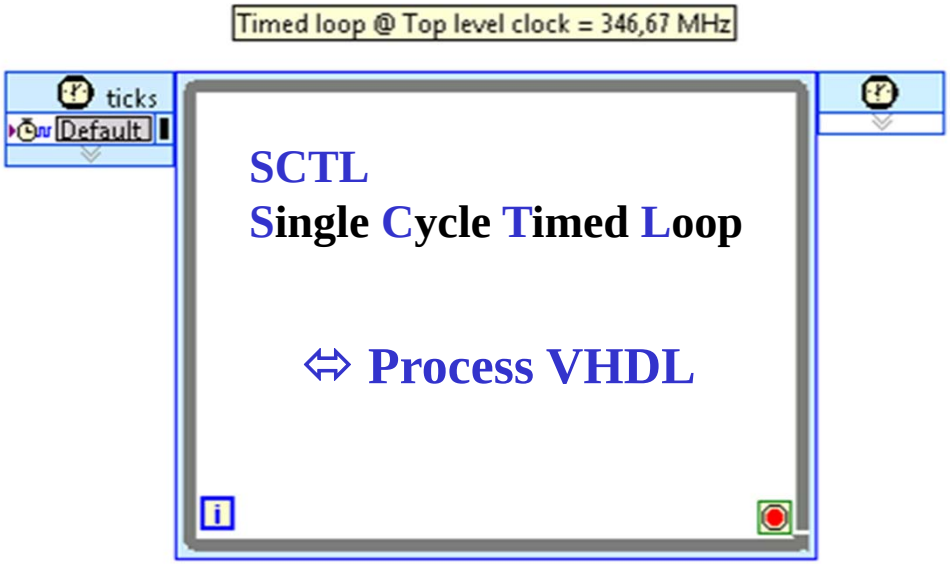
Démo No 1 : Fréquence maximale

Rapport de compilation



Fréquence maximale Timed Loop

- Time Loop vide → Difficile de faire mieux
  - 600 MHz Max (Sur Flex RIO PXIe 7962R)
- Limites HW FPGA
- Limites de l'implémentation du code

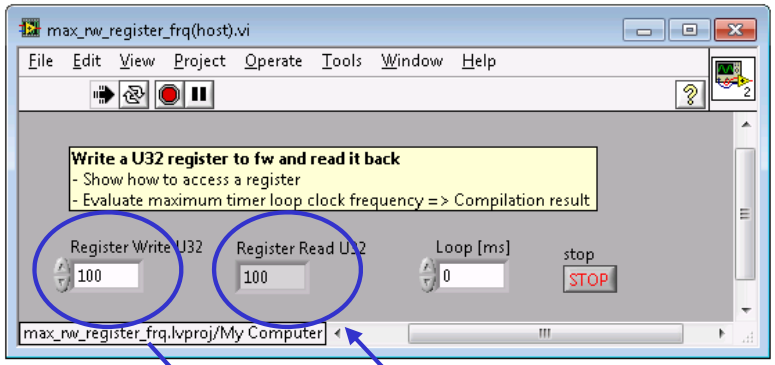


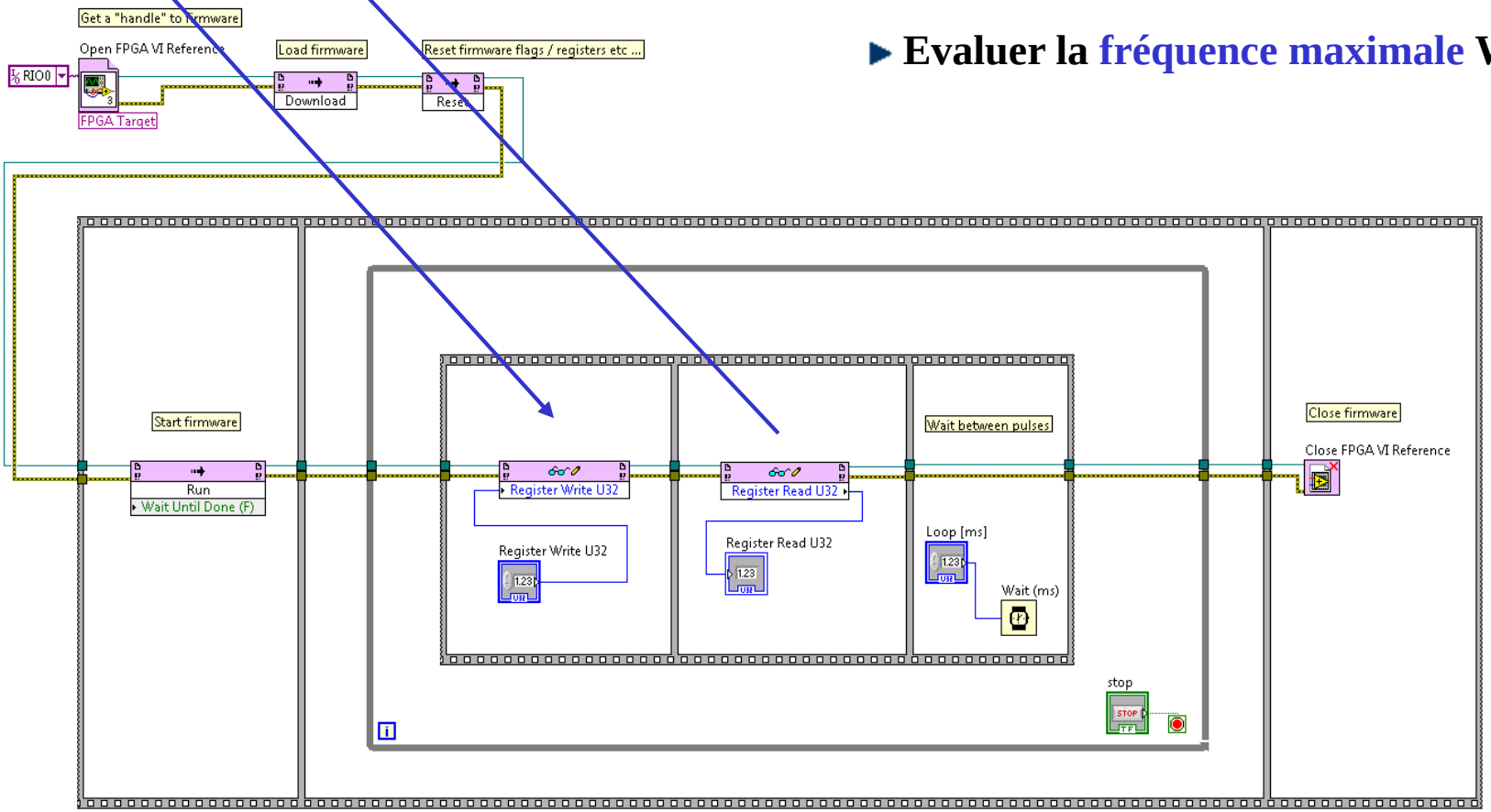
Temps de compilation

- Compter quelques dizaines de minutes
  - 14 minutes pour ce projet vide

Démo No 2 : Ecriture / Lecture d'un registre - Le SW

Répertoire démo → 2__max_rw_u32_register_fw_frq

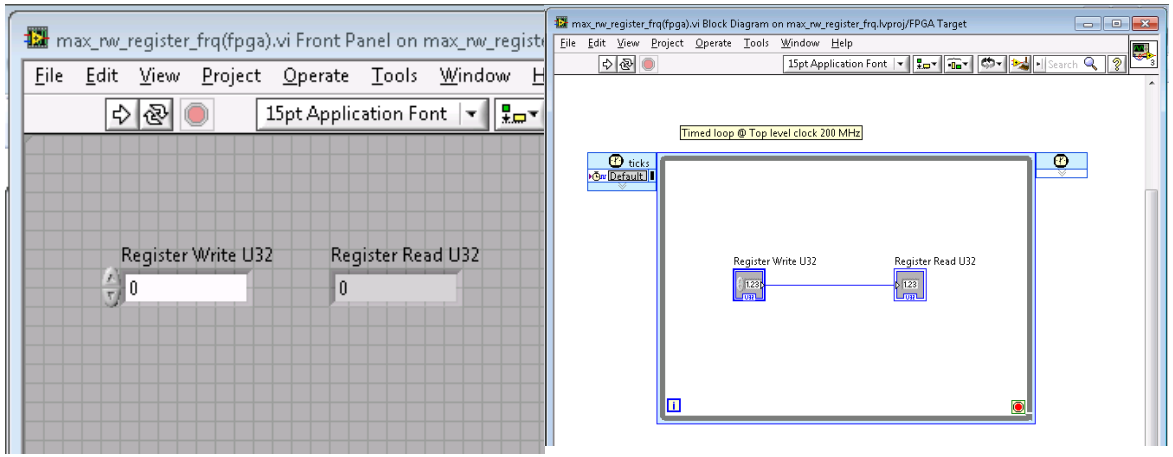


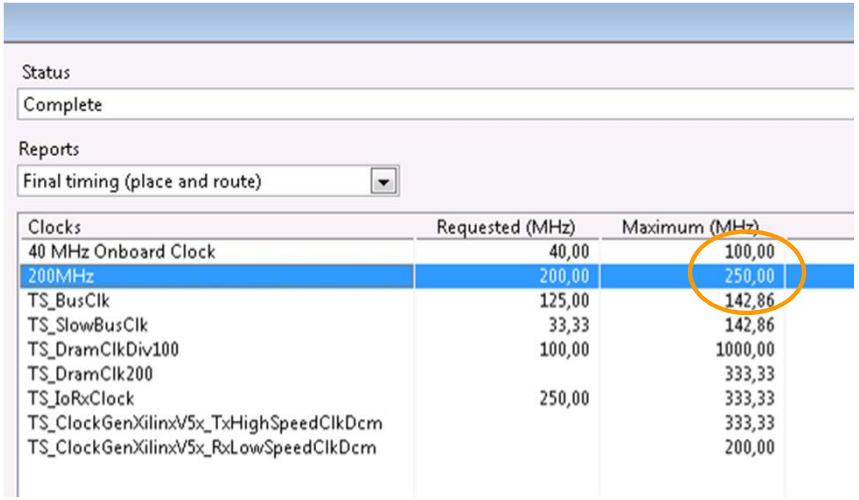


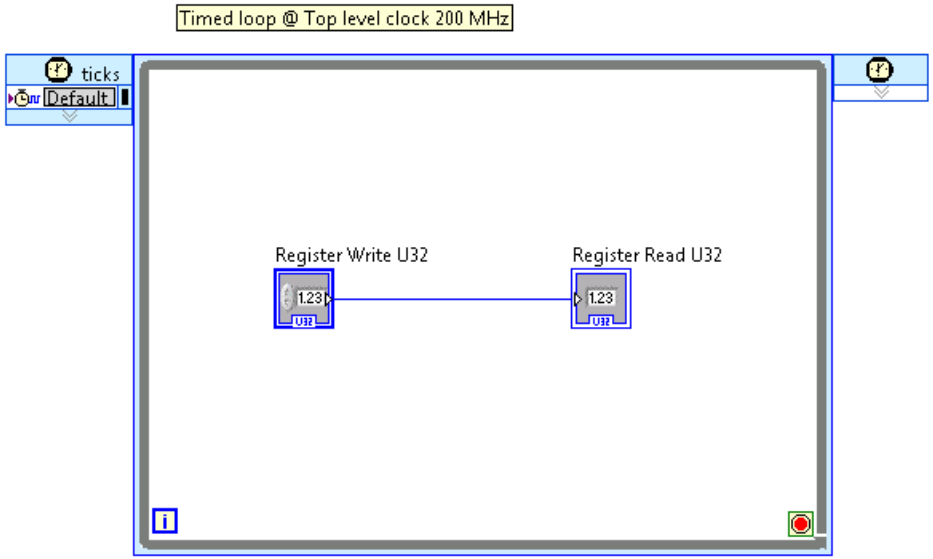
But ?

- SW écrit dans registre (32 bits) du FW
- SW relit contenu du registre
- SW sensé relire ce qu'il a écrit ;-)
- Evaluer la fréquence maximale W/R

Démo No 2 : Ecriture / Lecture d'un registre - Le FW







Fréquence maximale W/R ?

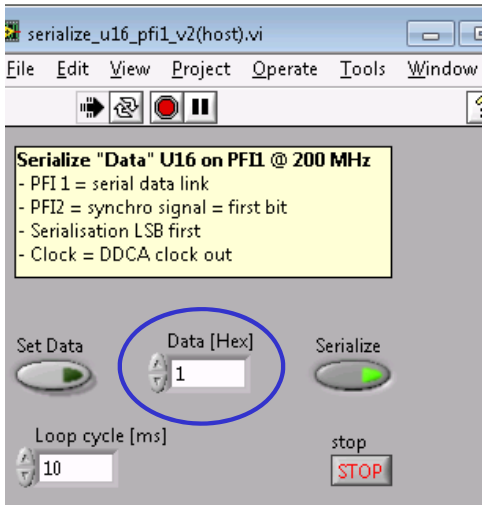
- Côté FW → Compilation → Max 250 MHz
- Mais aucune info sur F Max côté SW
  - Accès à 250 MHz ? … peu probable …
  - Pas de moyen de mesure … pour l'instant
    - Voir démo No 5, 6, 7

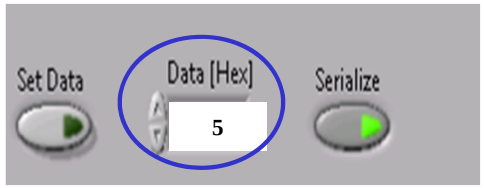
Démo No 25 : Sérialiser - Le SW

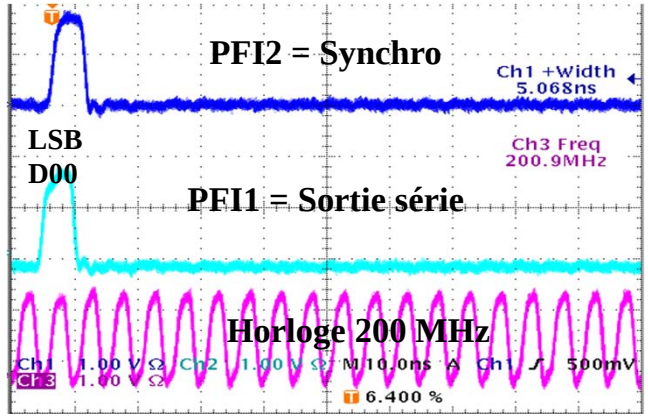
Répertoire démo → 25(42)__serialize_u16_pfi1_v2

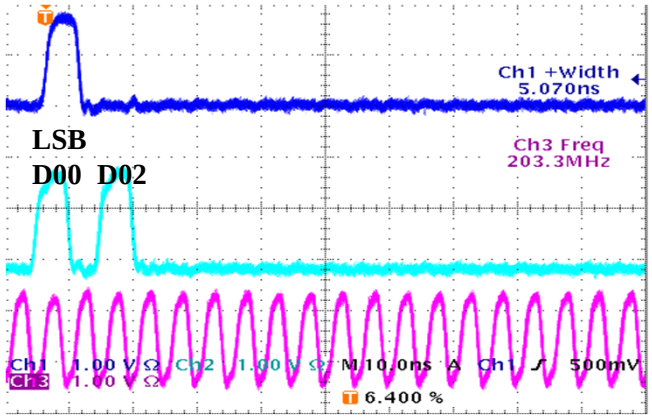
But ?

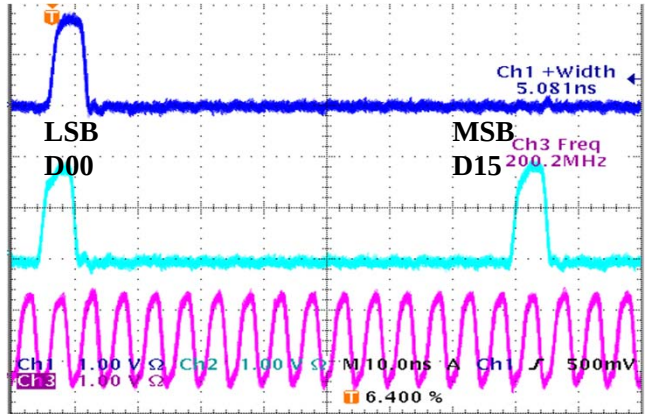
- Générer un lien série
  - Horloge 200 MHz
  - Synchro début mot = PFI2
  - Sortie série = PFI1
  - Sérialisation LSB first
  - Un seul mot configuré
    - Registe → Pas de DMA
- Trois exemples de données
  - $ 0001
  - $ 0005
  - $ 8001





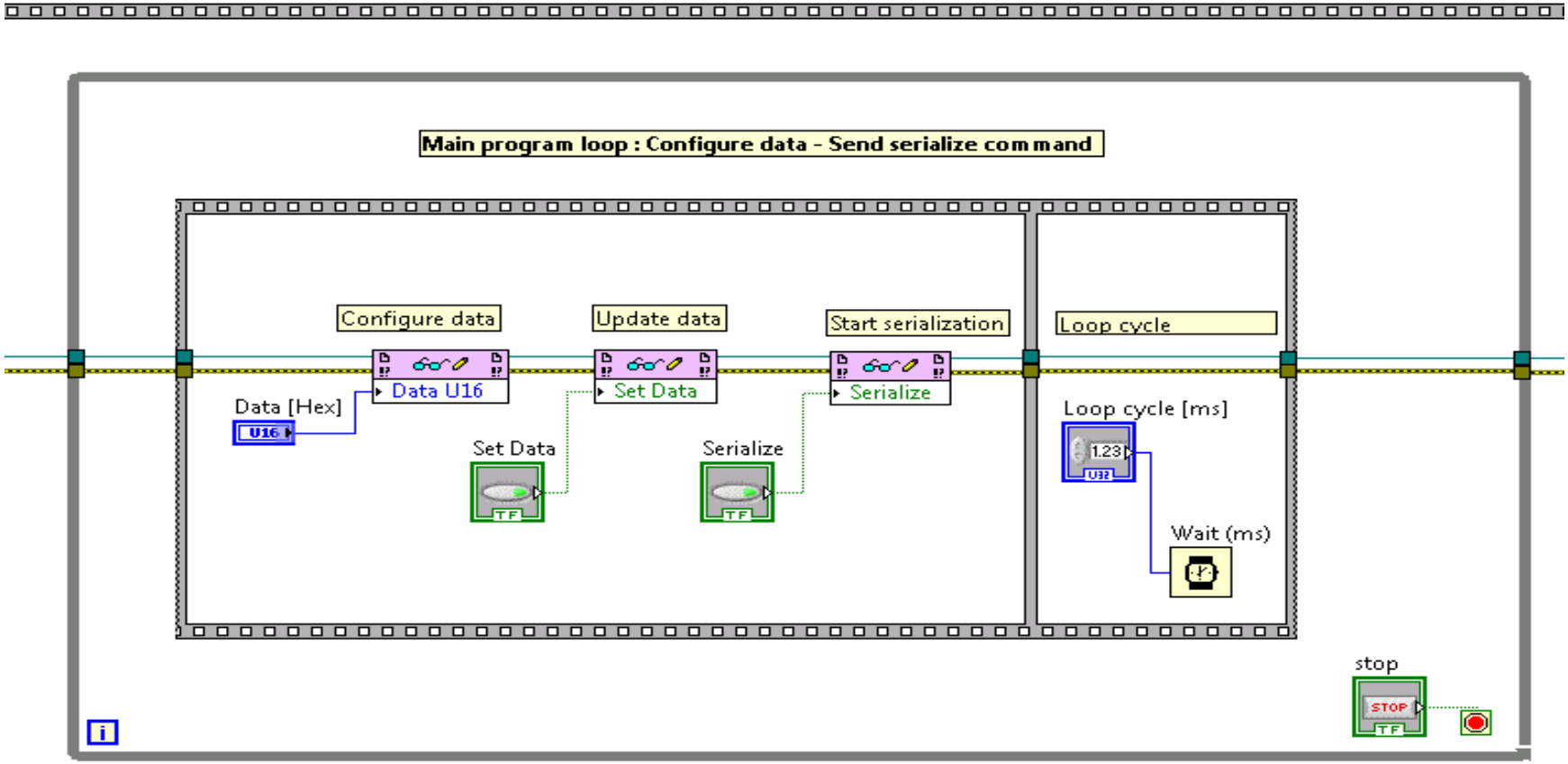




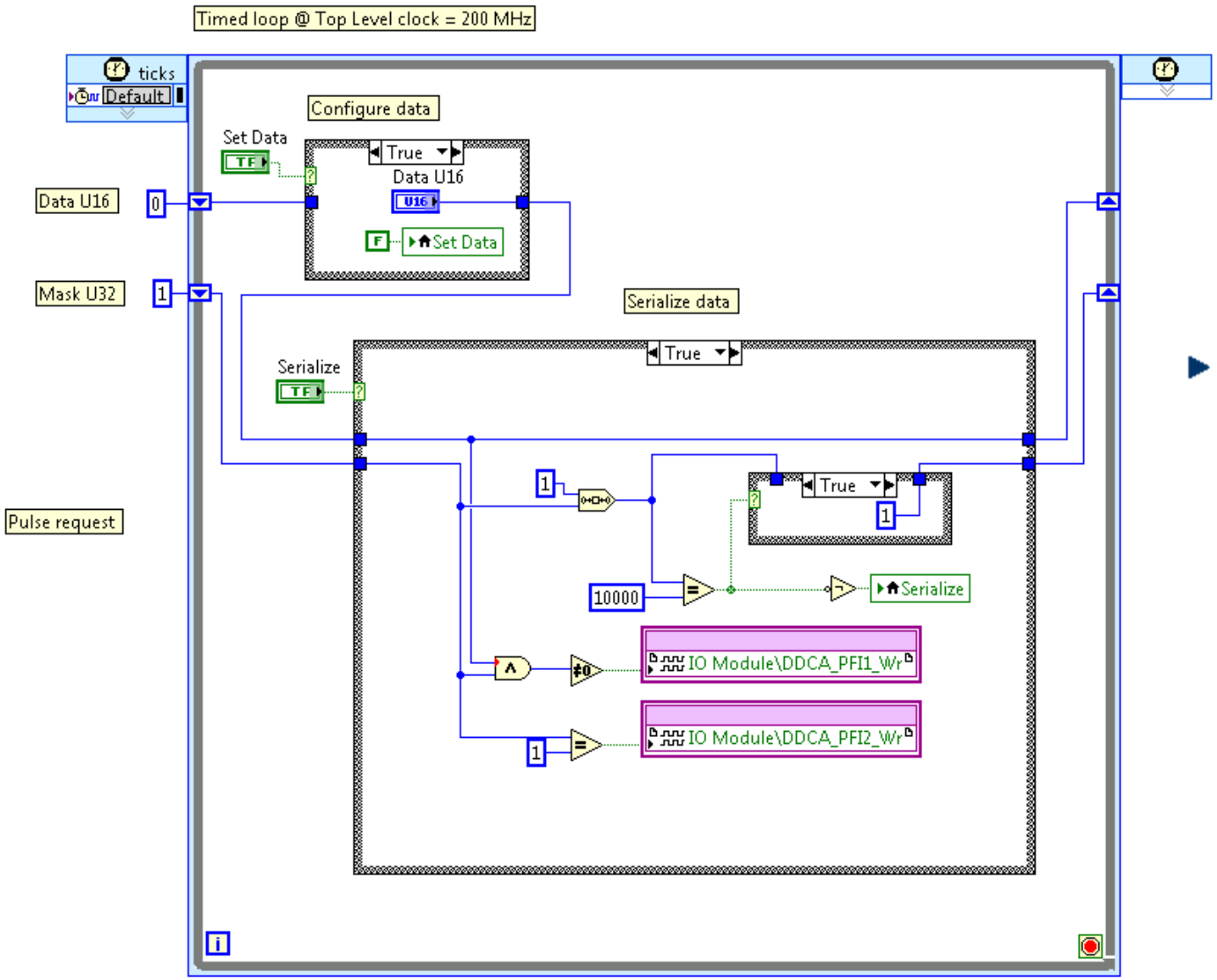


Rôle du SW ?

- Configure la donnée
- Donne l'ordre de sérialiser



Démo No 25 : Sérialiser - Le FW



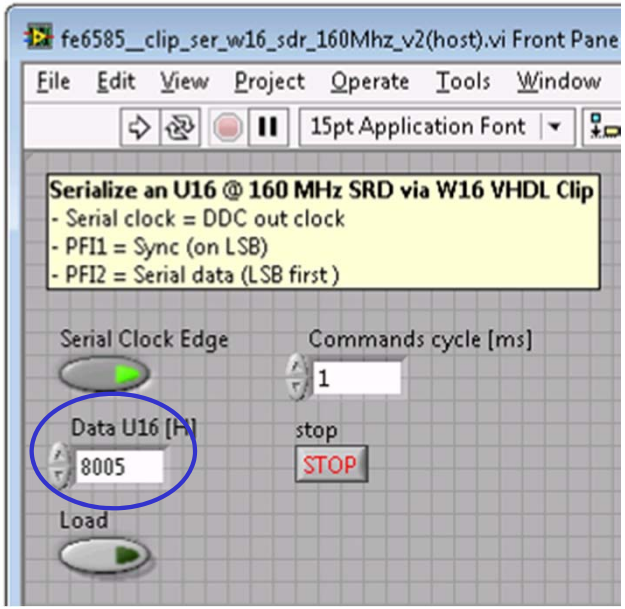
Comment ça marche ?

- **Une « Timed loop » @ 200 MHz**
  - **Charge la donnée à sérialiser**
  - **Génère un masque pour extraire les bits**
  - **Sort (Data AND Mask) sur PFI1**
  - **Set PFI2 lors de la sortie du premier bit**

Démo No 26 : Sérialiser (FW en VHDL) - Le SW

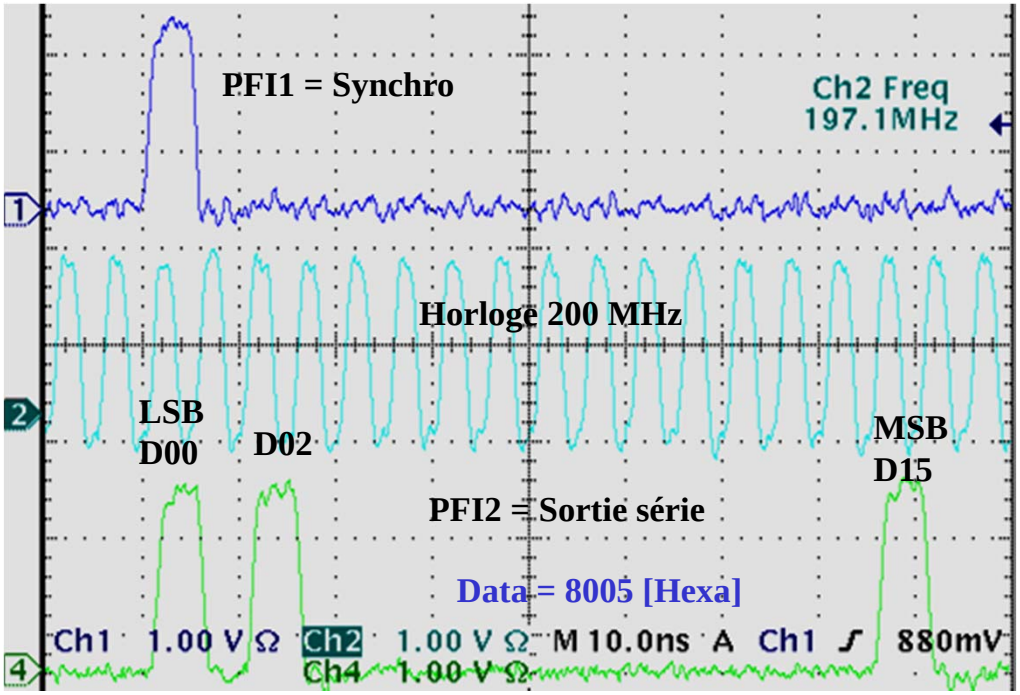
But ?

- Générer un lien série
  - Horloge 200 MHz
  - Synchro début mot = PFI1
  - Sortie série = PFI2
  - Sérialisation LSB first
  - Un seul mot configuré
    - Registe → Pas de DMA
- Un exemple de donnée sérialisée
  - $ 8005



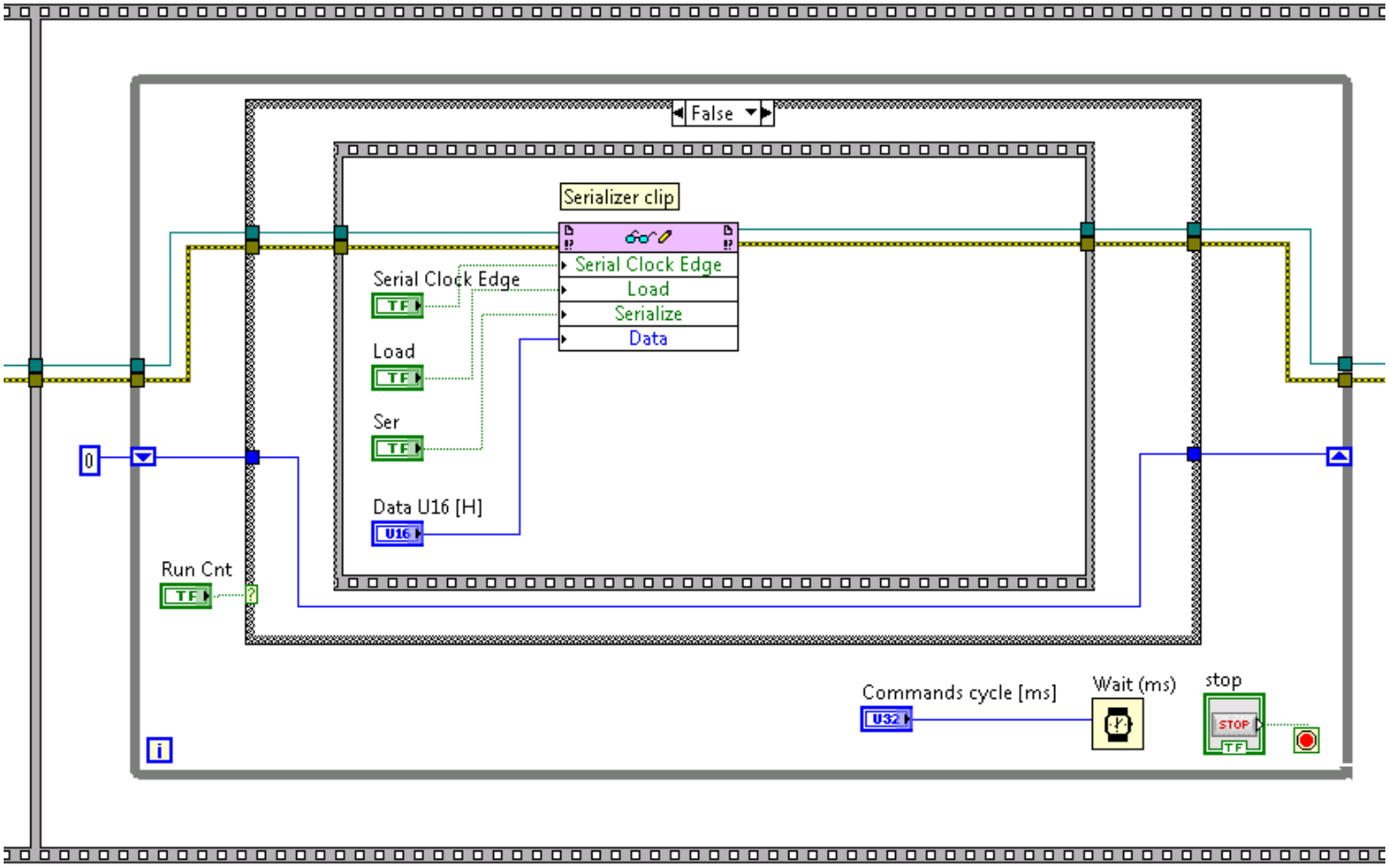
Différence / Demo No 25

- Le FW de désérialisation
  - Pas écrit en LabVIEW FPGA
  - Ecrit en VHDL via un User CLIP
- User CLIP ?
  - User Component Level IP



Rôle du SW ?

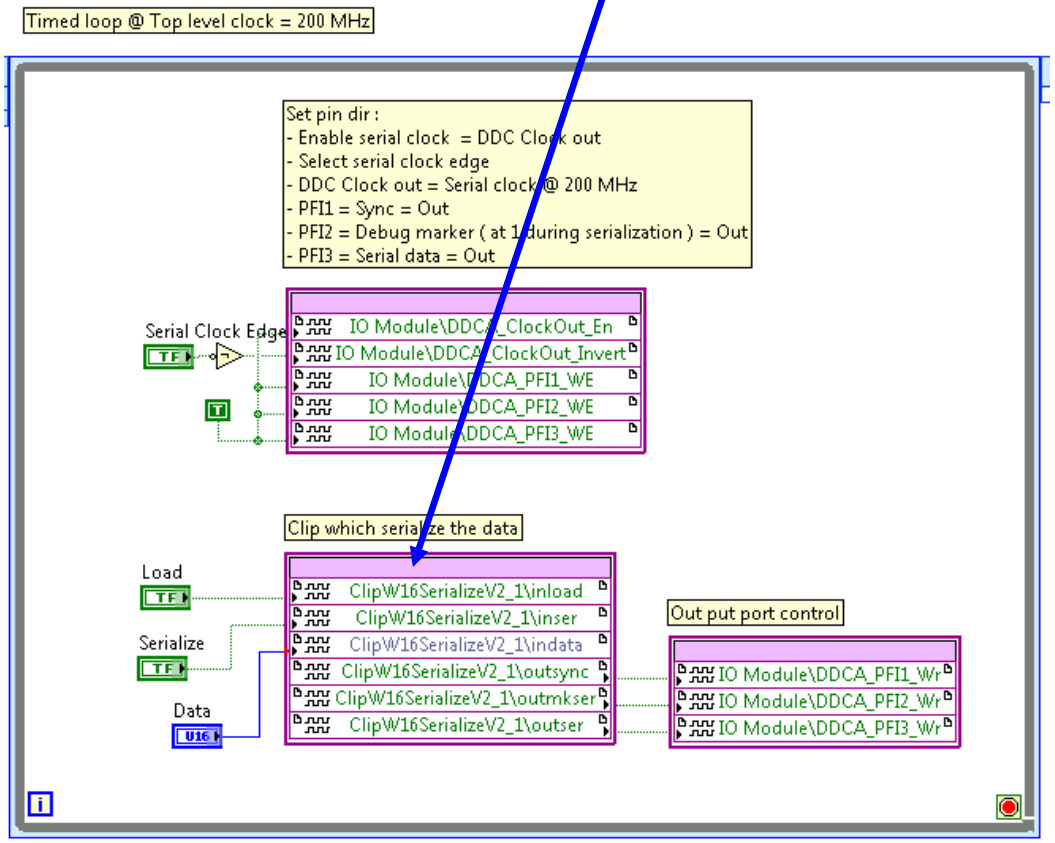
- Configure la donnée
- Donne l'ordre de sérialiser

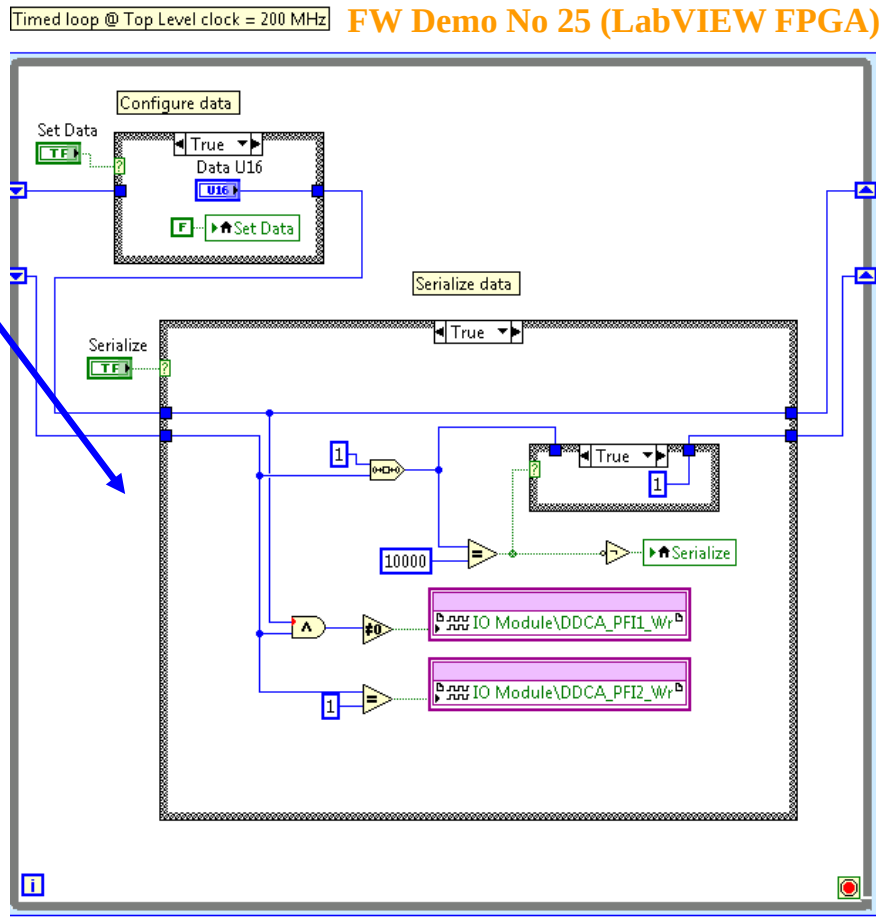


Démo No 26 : Sérialiser (FW en VHDL) - Le FW LabVIEW FPGA

Comment ça marche ?

- Un seul VI (CLIP) contient le code de désérialisation
  - Code écrit en VHDL et « encapsulé » via le CLIP
- A comparer au code de la démo No 25

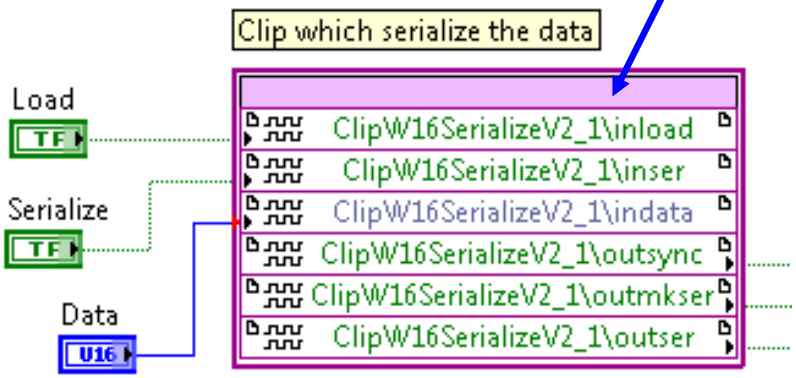




Démo No 26 : Sérialiser (FW en VHDL) - Le FW VHDL

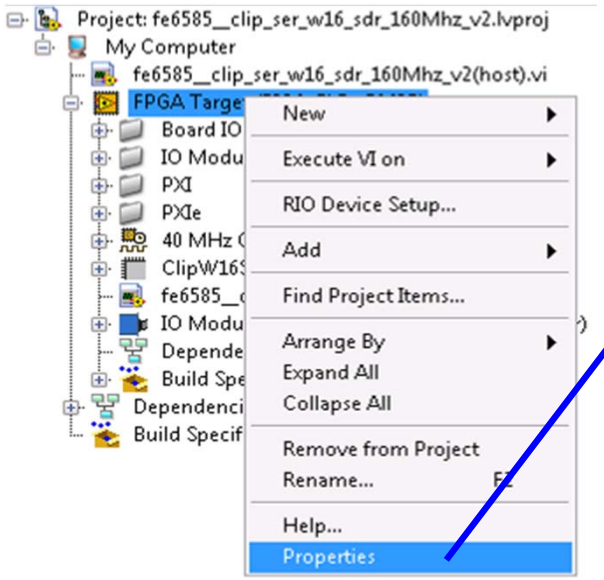
Comment procéder ?

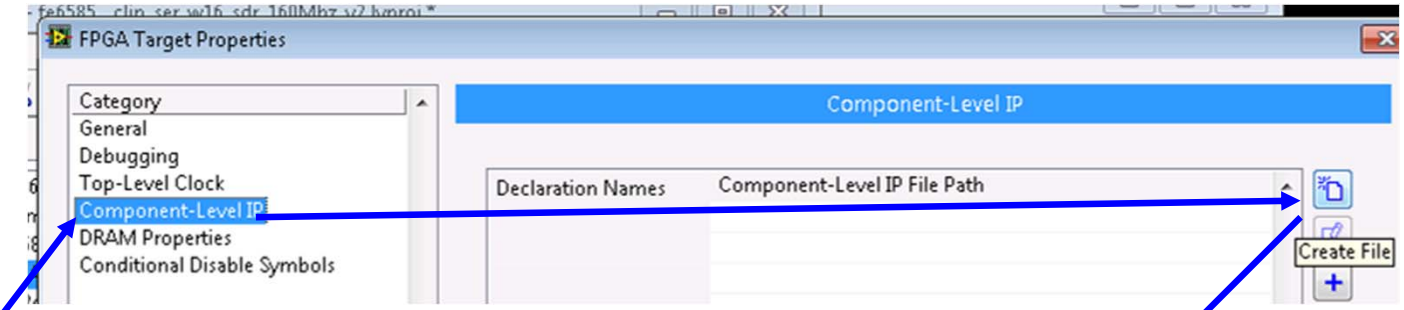
- Ecrire le code en VHDL
- Vérifier la syntaxe
  - Peut se faire sous LabVIEW FPGA
- L'intégrer dans le projet via un CLIP

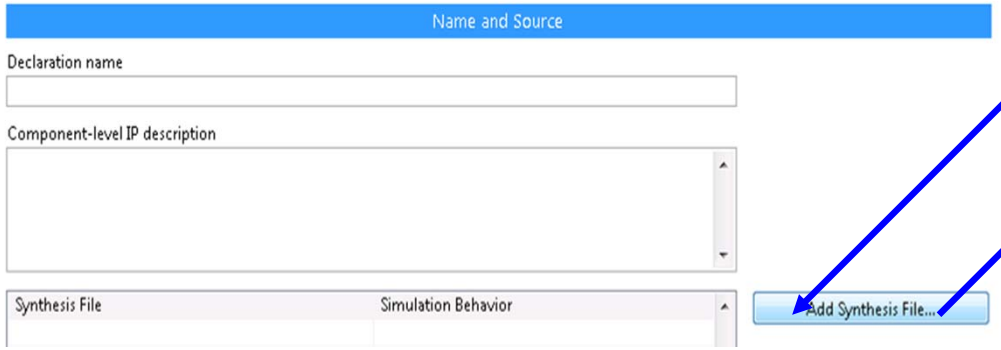


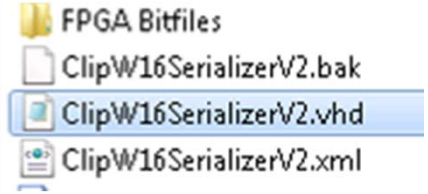

```
LIBRARY ieee;

  use ieee.std_logic_1164.all;
  use ieee.numeric_std.all;
  use IEEE.STD_LOGIC_ARITH.ALL;

  ENTITY EClipW16SerializerV2 is
     port (
            clk      : in std_logic;  -- Clock
            aReset   : in std_logic;  -- Reset
            InLoad   : in std_logic;  -- Command : Load "InData" in serializer register
            InSer    : in std_logic;  -- Command : Serialize data
            InData   : in std_logic_vector (15 downto 0); -- Data to serialize
            OutSync  : out std_logic; -- Sync signal generated on InData LSB
            OutMkSer : out std_logic; -- Debug gignal ar 1 during serialization of data
            OutSer   : out std_logic  -- Serial output
          );
  END EClipW16SerializerV2;


  ARCHITECTURE AClipW16SerializerV2 of EClipW16SerializerV2 is

   BEGIN

    PROCESS (aReset, InLoad, InSer, clk)

    VARIABLE VData   : std_logic_vector (15 downto 0); -- Used to shift source data
    VARIABLE VSync   : std_logic_vector (15 downto 0); -- Used to generate Sync pulse & to d
                                                       -- MSB is shifted : VSync(15) = 1 =>

    VARIABLE VRunSer : bit; -- Flag used to enable / disable serialization sequence

    BEGIN

      -- Reset

      IF (aReset = '1') THEN
        OutMkSer <= '0';
        VData    := "0000000000000000";
        VSync    := "0000000000000000";
        OutMkSer <= '0';
        OutSer   <= '0';
        VRunSer  := '0';

      -- Load

      ELSIF (InLoad = '1') THEN
        OutMkSer <= '0';
        OutSer   <= '0';
```

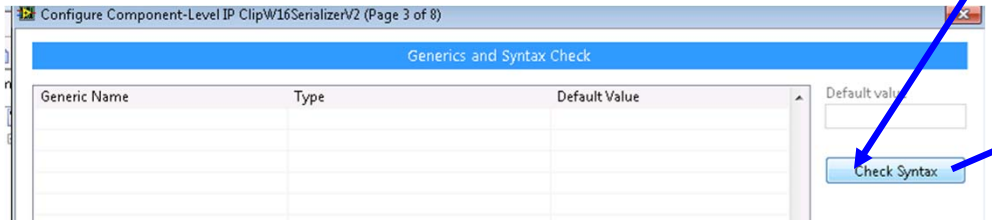
Note : Créer User CLIP → Intégrer le code VHDL

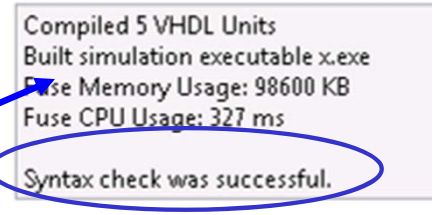


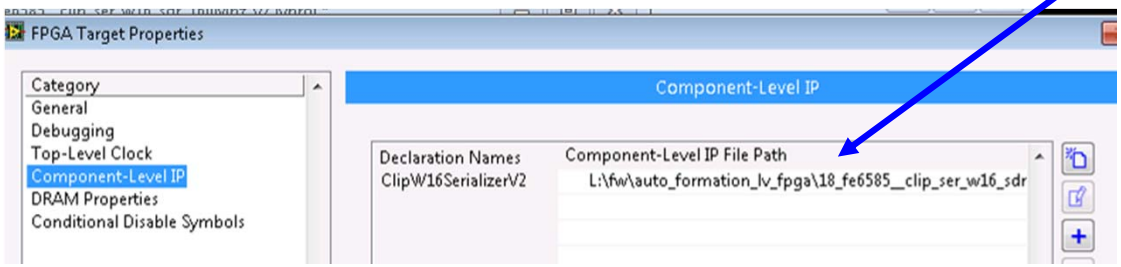








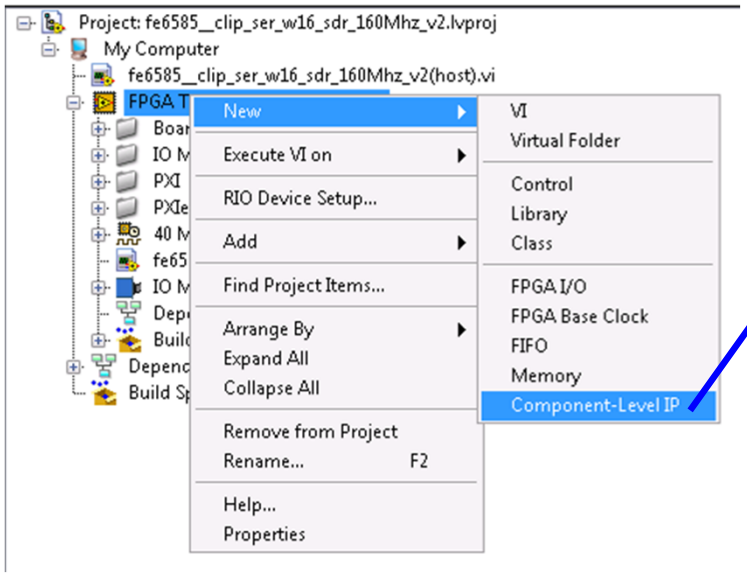




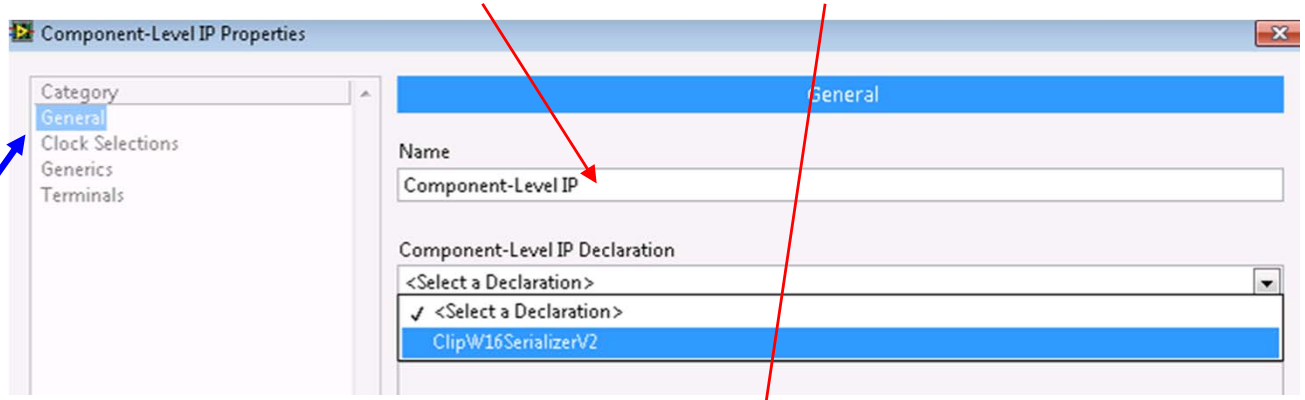
Procédure

- **FPGA Target**
  - **Properties**
    - **CLIP**
      - **Create file**
      - **Add Synthesis file**
      - **Check Syntax**
      - **Next, next …**
- **Code VHDL intégré**

Note : Créer User CLIP → Ajouter le CLIP au projet

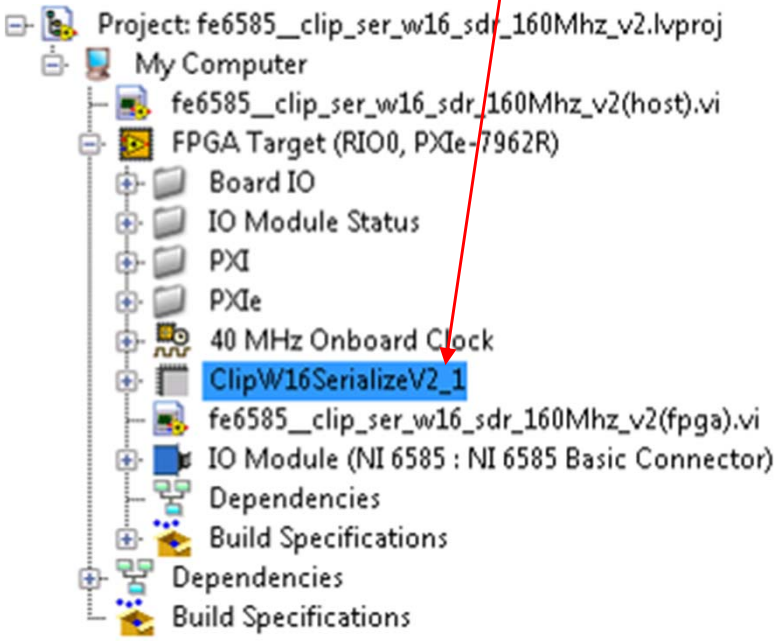


Donner un nom au CLIP → Ex : ClipW16SerializeV2_1



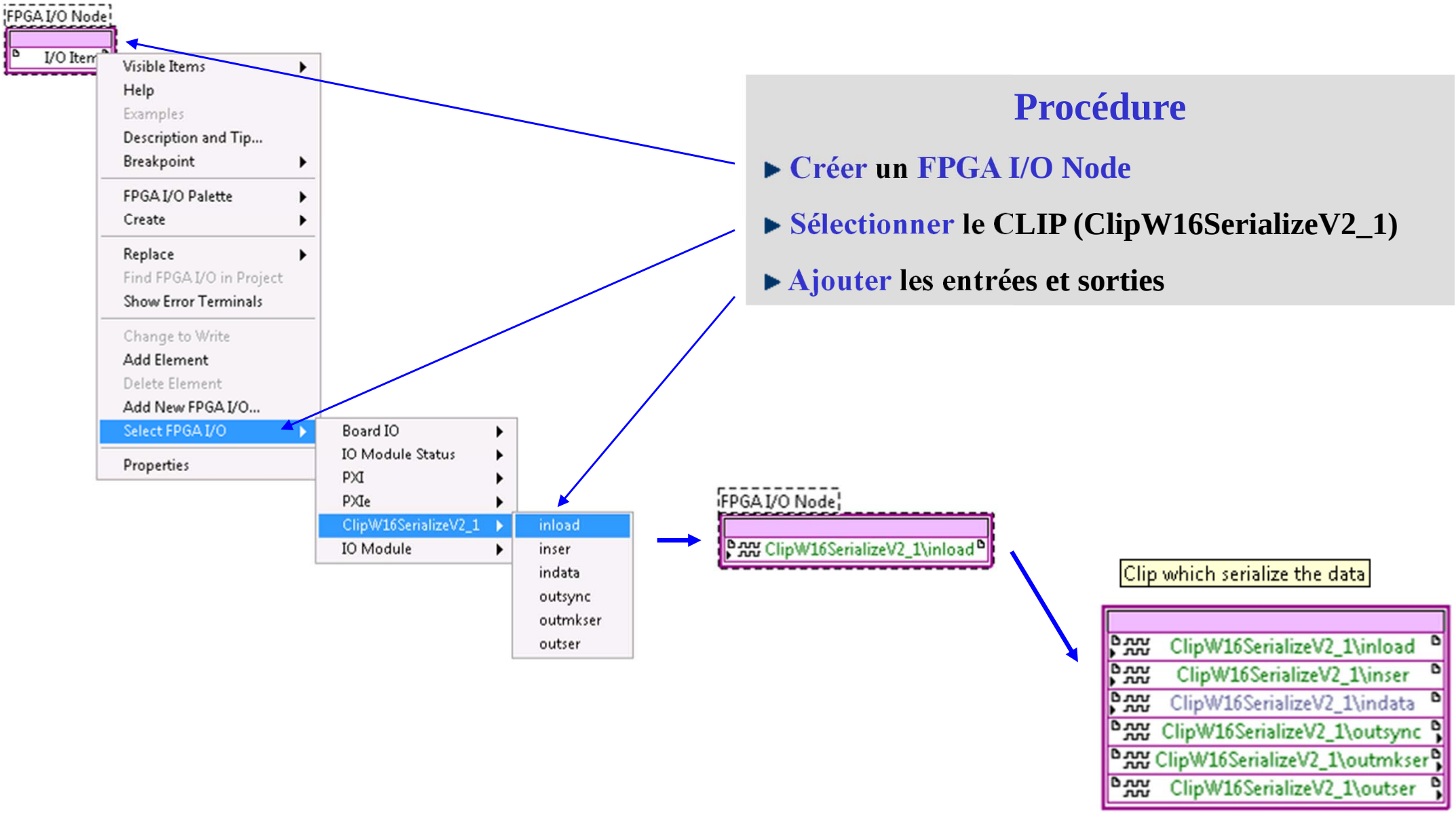
Procédure

- **FPGA Target**
  - **Properties CLIP**
    - **Create file**
    - **Add Synthesis file**
    - **Check Syntax**
    - **Next, next …**
- **CLIP créé**



CLIP ajouté au projet

Note : Créer User CLIP → Utiliser le CLIP dans le diagramme



Procédure

- **Créer** un **FPGA I/O Node**
- **Sélectionner** le **CLIP (ClipW16SerializeV2_1)**
- **Ajouter** les **entrées et sorties**

CLIP prêt à être utilisé

La R&D PICSEL

Physics with Integrated Cmos Sensors and ELectron machines

- IPHC (Strasbourg)
  - Département de Recherches Subatomiques
    - La R&D PICSEL (Marc WINTER)

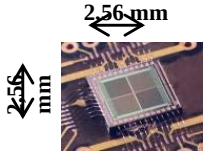
- R&D Capteurs monolithiques à pixels → MAPS
  - Donnent la position d'impact des particules (x, y)

- Plus de 30 capteurs conçus et validés depuis 1999
  - Simple matrice de pixels à lecture analogique
  - MAPS numériques à suppression de zéros intégrée

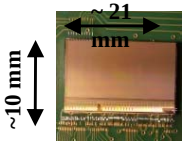
- L'organisation du groupe
  - Définition des projets → 5 Chercheurs
  - Conception Microélectronique → 10 Ingénieurs
  - Caractérisation des capteurs → 5 Ingénieurs
    - Conception des bancs de test & Caractérisation
  - Etudiants, Doctorants

Le capteur : C'est une matrice de pixels

- MAPS analogique → Lecture analogique série
- MAPS numérique → Discriminateur / colonne
- MAPS « intelligent » → Suppression de zéros intégrée



1999 - Mimosa 1
6,5 mm² - 600 images / s



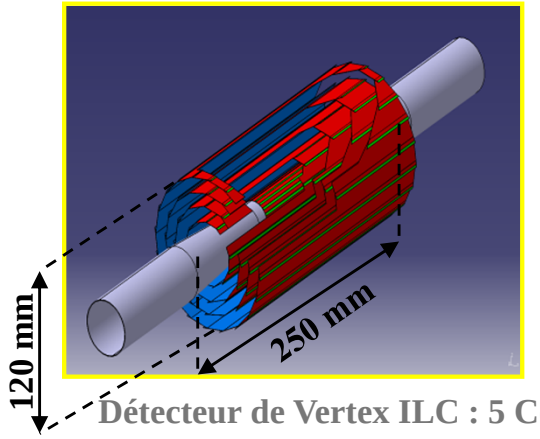
2008 - Mimosa 26
4 cm² - 10 000 images / s

En 10 ans de R & D …

- Surface x 100 : ~ 4 mm² → 4 cm²
- Temps lecture / 10 : ~ 1 ms → 0,1 ms
- Tolérance aux radiations x ~ 10
- Amincissement / 10 : ~ 500 µm → 50 µm

Les Applications des capteurs

- Futurs Détecteurs de vertex : STAR – ILC – CBM



Exemple : ILC

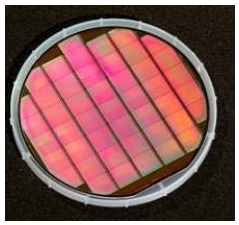
- ✓ ~ 300 10^6 pixels
- ✓ Pitch 20-10 µm
- ✓ Temps de lecture 25-200 µs
- ✓ 50 KRad/an
  10^{11} $n_{eq}/cm^2/an$

Détecteur de Vertex ILC : 5 Couches de MAPS ~ 300 10^6 pixels

Caractérisation d'un MAPS : Test fonctionnel - Calibration – Tests en faisceau

Test Fonctionnel

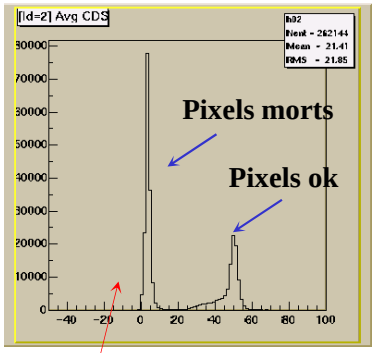
Rendement de production [%]



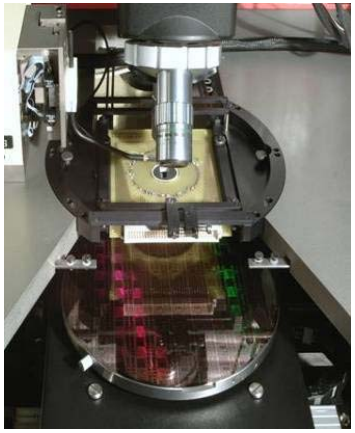
Galette 6 pouces
33 MAPS

↓

Test Sous Pointes

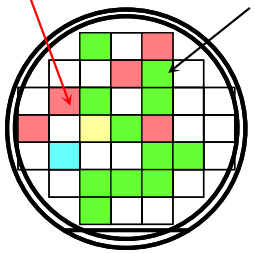


Affichage en ligne



Mauvais

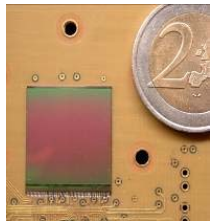
Bon



Carte Galette

Calibration - Photons X

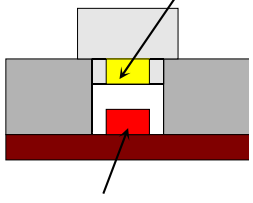
Gain [µV/e-] – Collection charge [%]

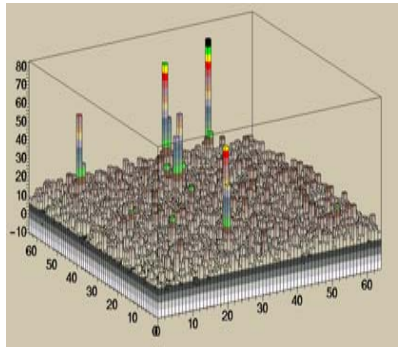


MAPS
&
Source

↓

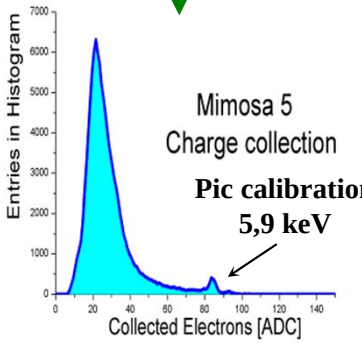
Fe55
Source





Affichage en ligne

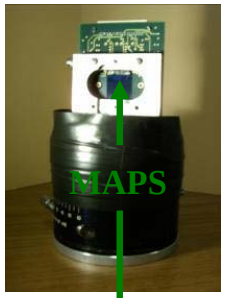
↓

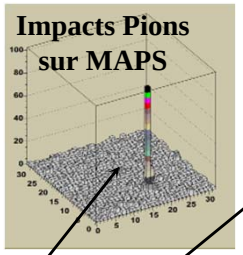


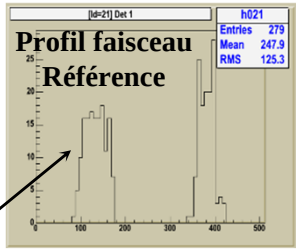
Analyse hors ligne

Tests en faisceau (CERN – DESY)

Résolution [µm] - Efficacité détection [%]

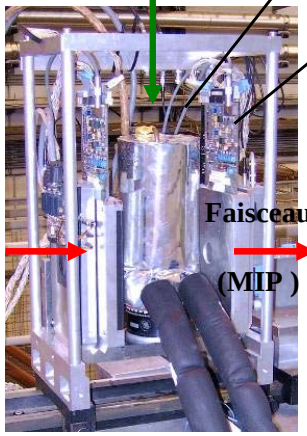


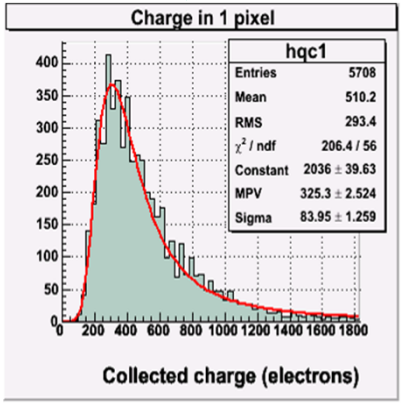




Affichage en ligne

↓





Télescope : Résolution ~ 1 um

Analyse hors ligne

Ces bancs de test nécessitent des systèmes d'acquisition de données

Mais seulement ~ 30 % des ressources humaines affectées au DAQ → 70 % « Manip »

L'Equipe Test

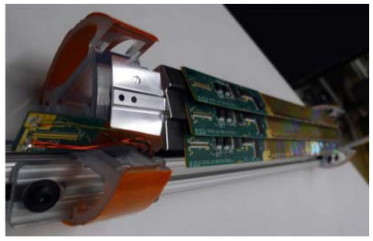
M. Specht
- SW slow ctrl, DAQ
- Intégration DAQ
- Validation bancs de test
- Schéma & PCB

Telescope EUDET - 2009

K. Jaaskelainen
- Dev FW VHDL - LV FPGA
- SW slow ctrl
- Caractérisation ASICs
- Schéma & PCB

DAQ Beam Test - 2009

G. Claus
- Coordination
- SW DAQ
- Validation DAQ
- Formation collaborateurs



1 secteur de STAR

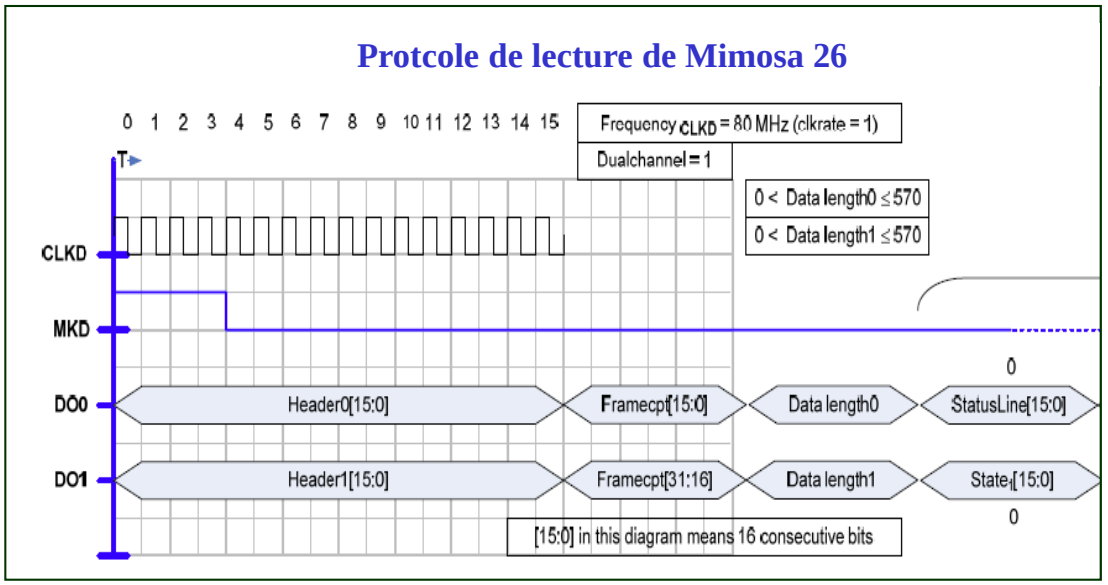
Insertion du détecteur STAR - 2013

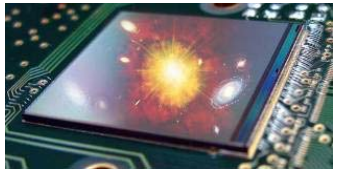
M. Goffe
- Caractérisation MAPS
- Formation collaborateurs
- Responsabilité beam-test
- SW d'analyse labo

M. Szelezniak
- Intégration des expériences (STAR, BEAST)
- Caractérisation
- Mécanique détecteurs & Refroidissement
- Schéma & PCB

MAPS Acquis par les cartes NI

- **Carte Flex RIO PXIe 7962R**
  - **Mimosa 26**
    - **Clock à 80 MHz + Mkd (Synchro)**
    - **2 Data links SRD à 80 Mb/s**
    - **Longueur d'une trame = 9 216 bits / 115,2 us**
  - **Mimosa 28**
    - **Clock à 160 MHz + Mkd (Synchro)**
    - **2 Data links SDR à 160 Mb/s**
    - **Longueur de trame = 29 696 bits / 185,6 us**
  - **Plusieurs capteurs ➔ Tous synchrones**
    - **Telescope EUDET = 6 x Mi26 = 12 liens LVDS @ 80 Mb/s**
    - **Telescope SALAT = 12 x Mi28 = 24 liens LVDS @ 160 Mb/s**
    - **Détecteur BEAST = 24 x Mi26 = 48 liens LVDS @ 80 Mb/s**





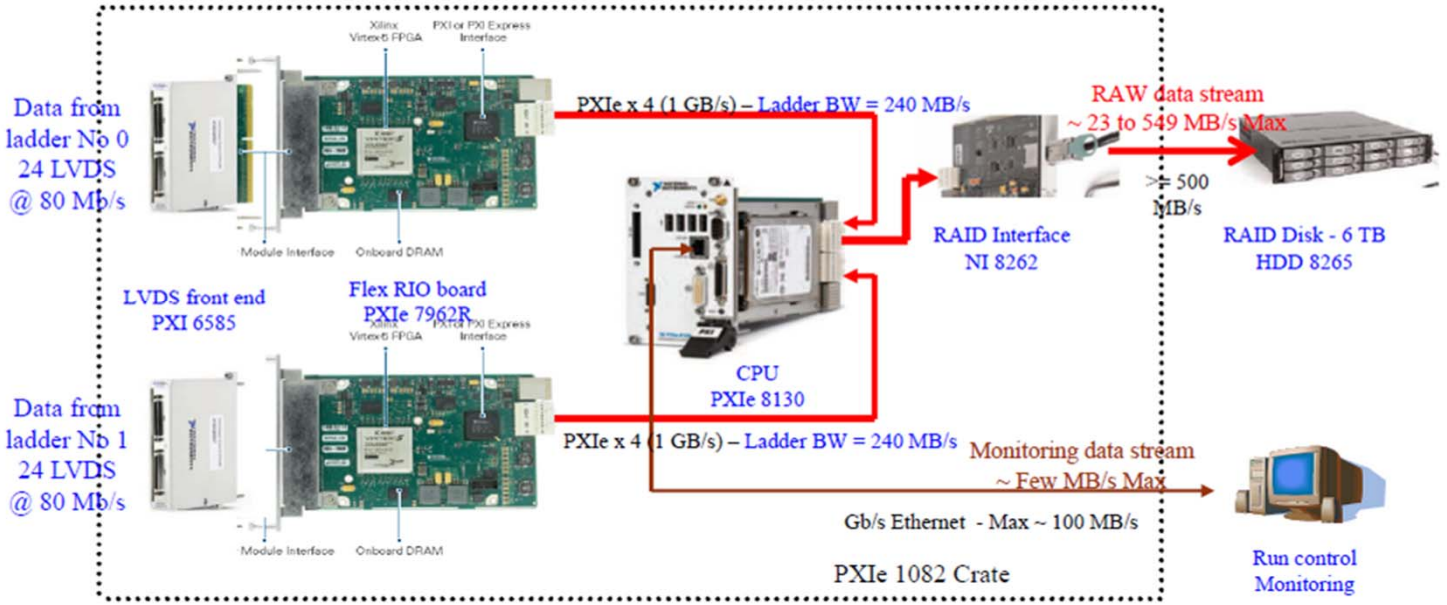
Le DAQ court … toujours derrière le MAPS …

- **Carte PXE NI6562**
  - **FSBB**
    - **Clock à 160 MHz + Mkd (Synchro)**
    - **1 Data link DDR à 320 Mb/s**
    - **Longueur d'une trame = 13 312 bits / 41,6 us**

- **Désérialization par SW – Bufferisation « on board » ➔ Avec un cycle CERN (8s/45s) seulement < 30 % temps mort**

DAQ basés sur Flex RIO : EUDET – AIDA - BEAST

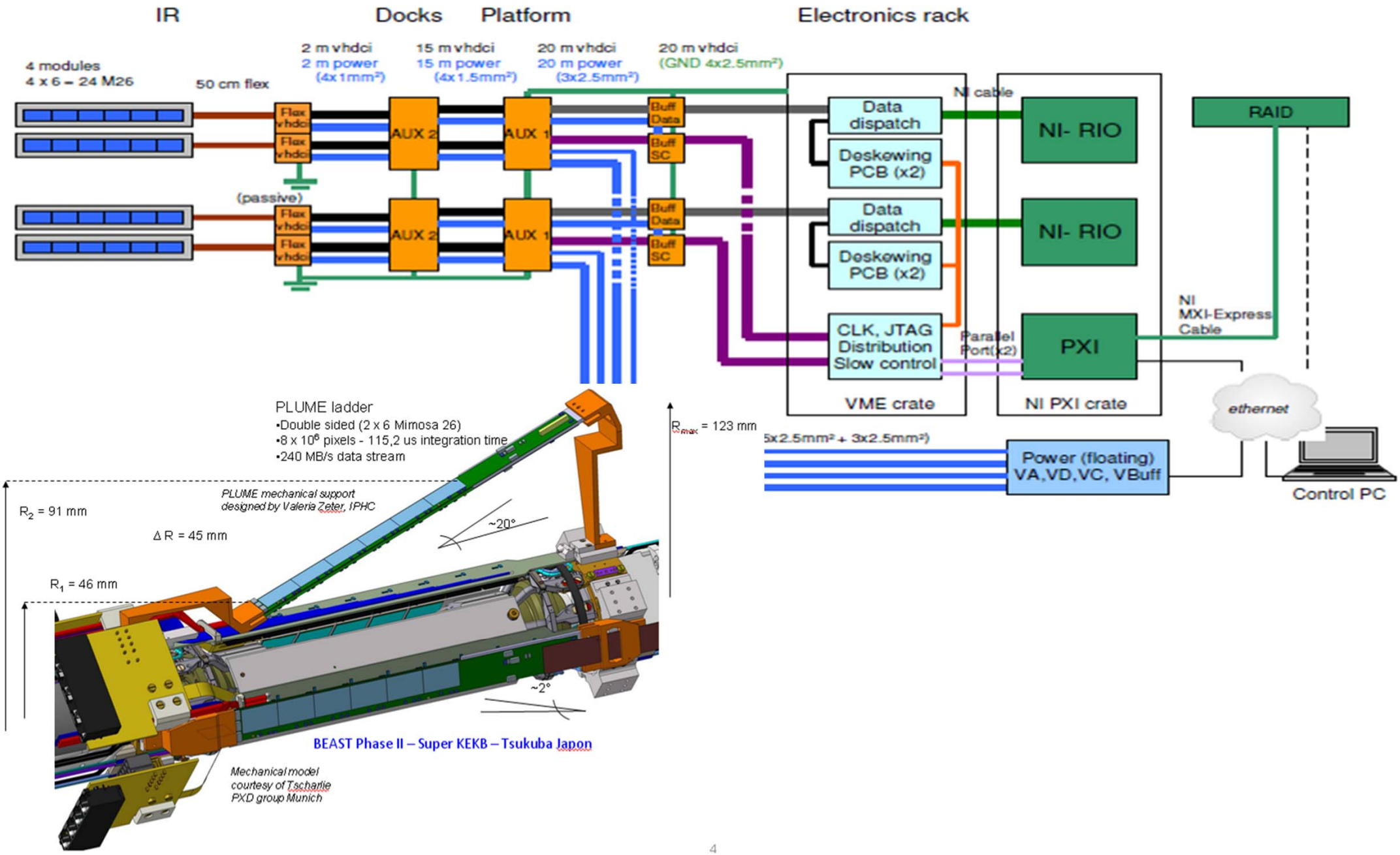
- **Démonstrateur DAQ Télescope EUDET adopté par la collaboration**
  - **DAQ opérationnel et sans temps mort – 6 x Mimosa 26 = 120 MB/s permanent vers le disque**
  - **Transfert technologique & Formation des collaborateurs (Décembre 2010)**
  - **8 Télescopes déployés par la collaboration EUDET**
  - **8 DAQ « Flex RIO » déployés par le groupe PICSEL pour ses capteurs**
  - **Coût total d'un système ~ 17 K€ DAQ + 7 K€ pour le RAID ➔ 24 K€**
    - **Châssis 3 K€, CPU 5 K€, FlexRIO 6 k€, FE 1 K€, Câbles 1 K€, Installation/Assurance 2 K€**
- **Upgrades effectuées**
  - **Telescope SALAT – 12 x Mimosa 28 = 480 MB/s ~ 50 % de temps mort**
  - **DAQ pour BEAST – 24 x Mimosa 26 = 480 MB/s ~ 8 % de temps mort ➔ Assemblé comme un lego en 1 semaine ☺ !**



Compromis FW / SW : Les limites

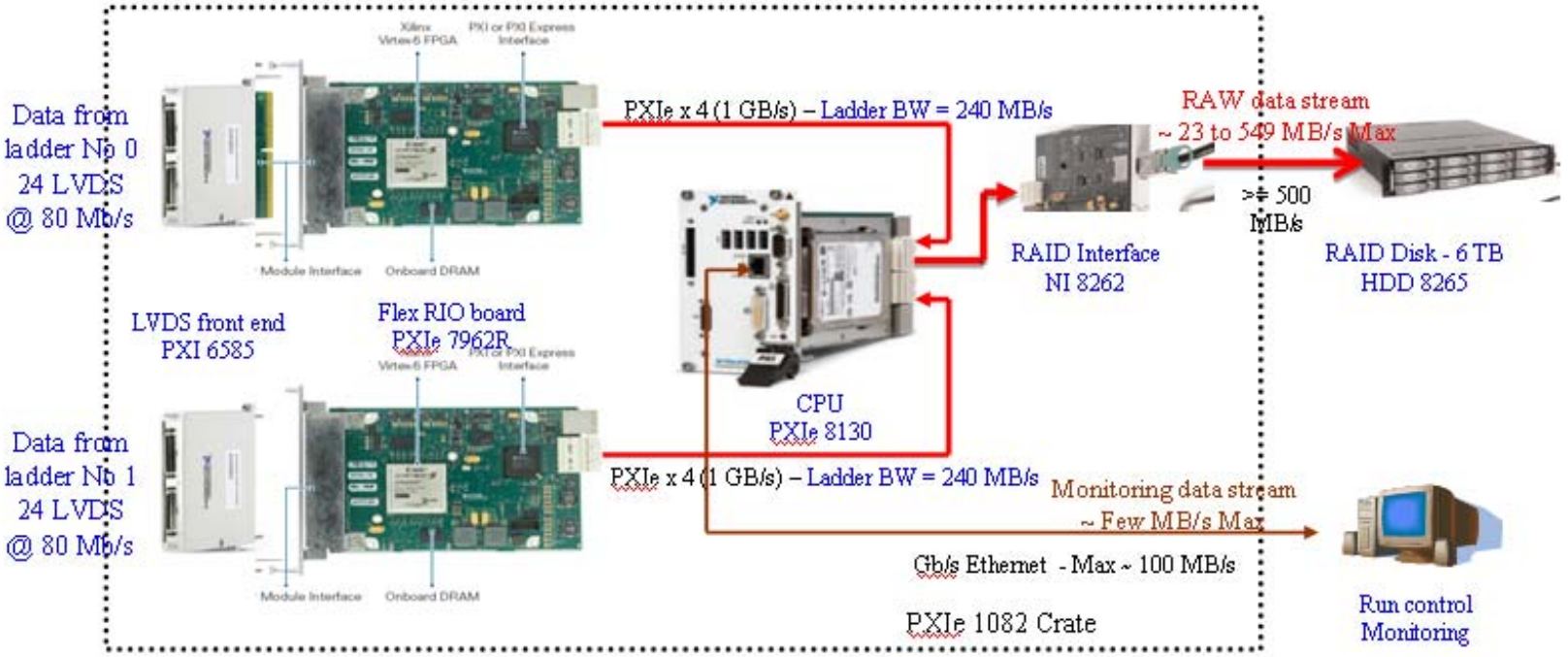
- Ce DAQ a néanmoins une limite ☹ ... qui n'en est pas vraiment une ☺
  - Pas liée au HW NI (Débit PXIe avec notre FW ~ 220 MB/s < Maximum ~ 800 MB/s)
  - Due aux décisions de conception (prises en toute connaissance de causes)
    - Minimiser le développement FW ➔ Reporter sur SW ➔ Basé sur puissance CPU + Débit bus PXIe
      - Flux données MAPS % Nb hits – Mais la taille variable de la trame est gérée par le SW, pas par le FW
    - Temps mort « plafonné » par débit PXIe donc >= 100 x (1 – Débit PXie / (Débit maximal MAPS * Nb MAPS))
    - Parfait pour les petits détecteurs
      - Télescope EUDET 6 x Mimosa 26 : Temps mort = 100 x (1 – 220 / (6 x 20)) = -1 ;-) ➔ 0 %
      - Télescope IPHC 6 x Mimosa 28 : Temps mort = 100 x (1 – 220 / (6 x 40)) = ~ 8,5 %
    - Convient moins aux grands détecteurs
      - Télescope AIDA SALAT 12 x Mimosa 28 : Dead time = 100 x (1 – 220 / (12 x 40)) = 54 %
  - Peut être résolu par un upgrade FW
    - Traitement actuellement réalisé par SW ➔ Intégré plus tard dans le FW
    - Développement par étapes : Fournir rapidement un outil aux chercheurs – L'améliorer ensuite

DAQ Projet BEAST1/3 - 2017



3 – DAQ : Hardware NI PXIe

- Based on the IPHC PXIe NI FlexRIO board (EUDET & AIDA Telescopes)
  - **Input :**
    - Beam cycle signal : Time stamping with 0,2 us resolution
    - 2 Ladders data stream = 2 x 8 .10^6 pixels @ 8680 fr/s = 24 x Mimosa 26 = 48 links @ 80 Mb/s = 480 MB/s
  - **Output**
    - Ladders RAW data stream @ 8680 frames/s → RAID system ~23 to 549 MB/s (see slides 11, 12)
    - Processed data Eg : Nb hits / Mi26 / Time unit → Ethernet Few kB/s (see slide 13)



11

G.Claus BEAST II MiniWorkshop, KEK 31 January 2016

3 – DAQ : Validation Test & Next steps

- Test of a DAQ prototype to check principle & performances ➔ Done
  - Worst case data stream (Mi26 saturated ⇔ DAQ input = 240 MB/s / Ladder)
  - Test done on 10^6 frames runs = 5760 beam cycles = 115 s (one test on 10^7 frames : same results)
    - Mi26 mode 0 : 92 hits / Mi26 / frame ➔ 0 % dead-time – 15 % CPU occupancy
    - Mi26 mode 1 : 188 hits / Mi26 / frame ➔ 4 to 7,5 % dead-time – 30 % CPU occupancy

2 x Mimosa 26

➔ One ladder emulated by one Mi26
➔ Ladder = Mi26 data links x 12

PXIe DAQ – 2 x FlexRIO boards

DAQ demo SW

- Next steps ➔ August
  - Final DAQ application
  - Interface / EPICS for Monitoring ➔ Next slides

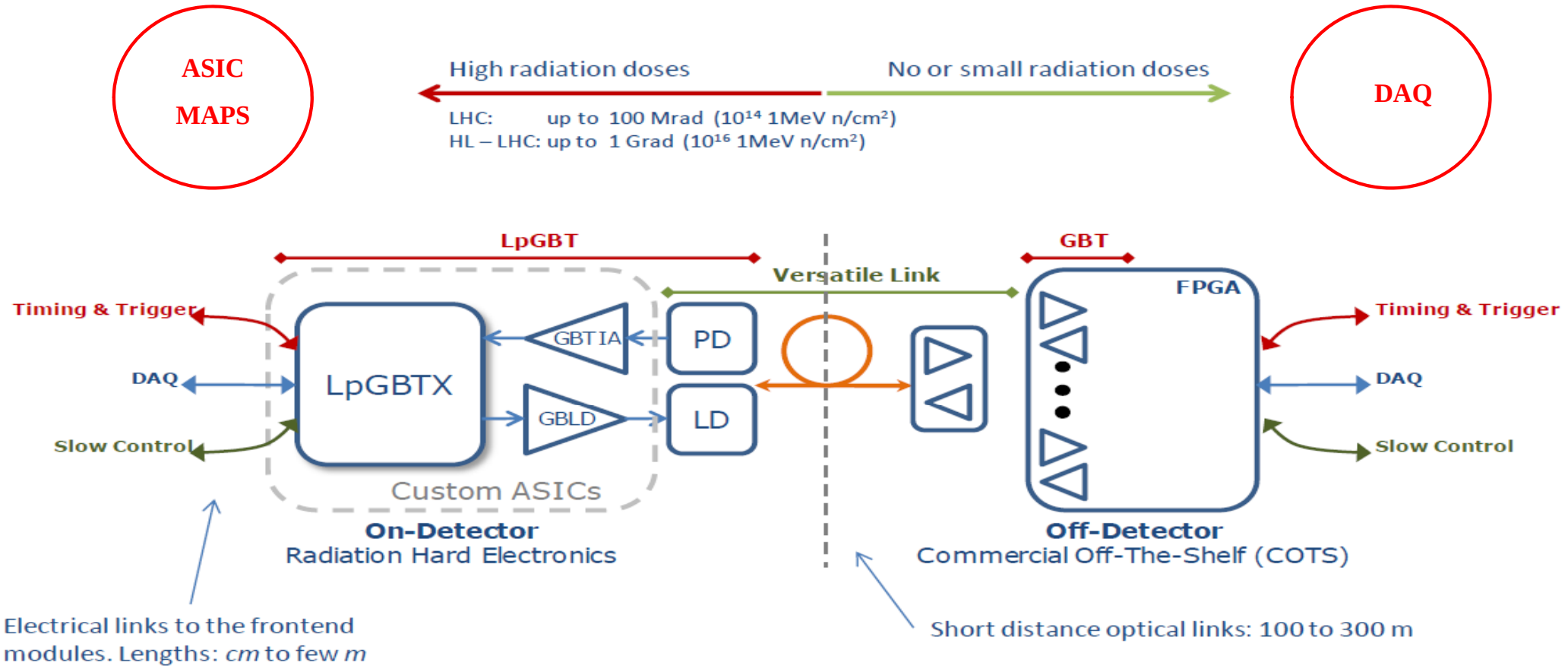
12

G.Claus BEAST II MiniWorkshop, KEK 31 January 2016

DAQ pour liens Gbps : GBT / GLIB 1/3

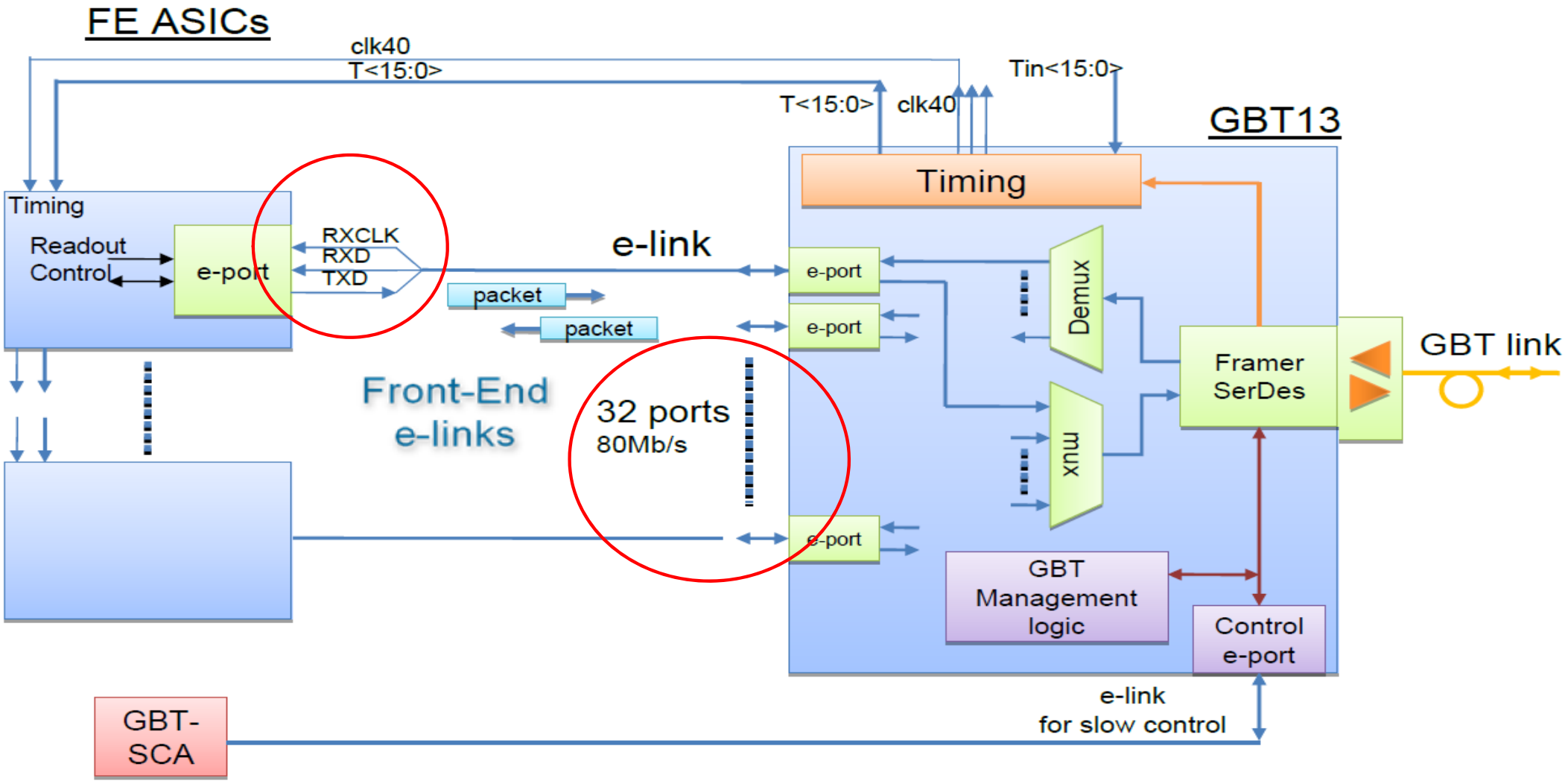
- **Nouveaux MAPS (CBM, …)**
  - **Liens 1 Gb/s série synchrone (Clk, Mkd, Data0)**
    - **HW (NI6585) et FW Flex RIO actuel limité à 320 Mb/s – Mais FE 1 Gb/s (NI6587) existe et dispo**
    - **Possible avec Flex RIO + FE dont nous disposons mais upgrade FW majeur ➔ RH pas dispo à ce jour**
  - **Liens >= 1 Gb/s 8B/10B (avec clock embarquée dans les données)**
    - **Pas de FE 8B/10B pour la Flex RIO ➔ Pas de réutilisation des Flex RIO déjà acquises**
    - **Il existe des Flex RIO spécifiques 8B/10B mais coût exorbitant (11 à 15 K € > Flex RIO + FE = 7-8 K€)**
      - **NI PXIe 6592R – 4 voies 0,5 à 10 GS/s ➔ 11 K€**
      - **NI PXIe 6591R – 8 voies 0,5 à 12,5 GS/s ➔ 15 K€**
    - **Pas de solution avec Flex RIO pour des liens 8B/10B**
- **Temps développement, Coût et Intégration**
  - **Si une solution +/- clef en main existe via GLIB ➔ Réduit les RH nécessaire**
  - **Coût des systèmes NI >> GLIB ?**

LpGBT & VL+ Link Architecture



Same "philosophy": radiation hard components in the detectors and COTS in the counting room

GBT13 Block Diagram



Solution « Custom HW » / COTS : Cas d'un DAQ labo 1/3

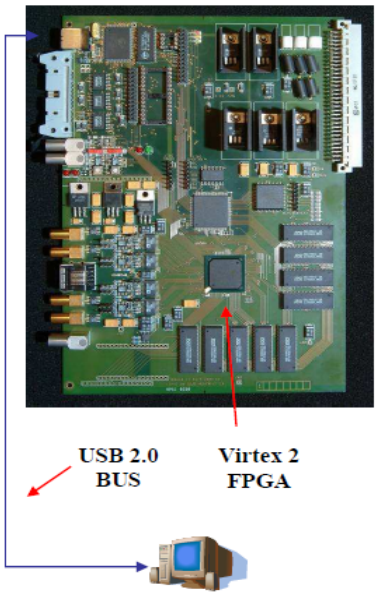
Carte USB
Entrées Analogiques / Numériques

Analog
- 1-4 ADC 12 bits – 40 Mhz
- 1 ADC 14 bits – 100 MHz
- 10^6 pixels **shared** 1-4 Inputs

Digital
- **Pattern** generator **& Trigger**
- 16 **Digital inputs**

Daq
- Transfer **15 MB/ s USB 2.0**

Firmware (To Be Done)
- CDS Calculation (Done)
- Pedestal substraction
- Data sparcification



- **Développée au laboratoire = « Custom board »**
- **Evolution cartes VME-LynxOS ➔ USB-Windows**
  - **Afin de réduire le coût des DAQ**
- **La carte existe – Mais ne convient pas !**
  - **Limite des entrées numériques**
    - **Mimosa 26 = 80 MHz > USB board = 50 MHz**

Développer une nouvelle carte ?
Leçons suite au développement de la carte USB ...

- **Développement trop long ... 2003 à 2006 = 4 ans !**
  - **HW : 1 an (CNAM) – FW : 3 Mois – SW : 1 Mois**
  - **Temps mort ~ 2 ans (Autres projets plus prioritaires)**
- **Production sous traitée ... Chronophage !**
  - **51 cartes produites en 3 lots – 2005, 2006, 2007**
  - **Interface / sous traitant + test des cartes**
    - **Organisation sous traitance ~ 1 mois**
    - **Suivi production + Test des cartes ~ 2 mois /lot**
    - **Total 1 Ingénieur x 7 mois**
- **Cycle de 4 ans + 7 mois de test des cartes ... incompatible avec l'objectif 1 upgrade majeur de DAQ / an !**
- **Le coût d'un DAQ complet ~ 3000 € / VME ~ 6000 €**
  - **PC Windows : 1000 € - Châssis VME : 2000 €**
  - **Carte 2000 € (PCB, composants 1500 €, montage 500 €)**

Solution « Custom HW » / COTS : Cas d'un DAQ labo 2/3

Carte NI6562
I/O Numériques – Bus PXIe

Ajouter une photo

De la carte

Carte PXI 6562 dans son châssis

- Carte National Instruments = « COTS »
- Format PXIe
  - 16 I/O @ 200 Mb/s SDR ou 8 I/O @ 400 Mb/s DDR
  - Données acquises en SRAM 2 à 256 Mb / I/O
  - Nombreuses fonctions de trigger
  - Lecture par bus PXI jusqu'à ~ 130 MB/s
- La carte convient pour ... ☺
  - Mi 26 (2008) : 2 Sorties 80 Mb/s SDR
  - Mi 28 (2011) : 2 Sorties 160 Mb/s SDR
  - FSBB (2014) : 2 Sorties 320 Mb/s DDR

NI 6562 est la solution ?
Bilan du DAQ basé sur la carte NI 6562

- Démonstration de la faisabilité ... en ½ journée !
  - Emulation Mimosa 26 (en fab) par géné de patterns
  - Test effectué avec NI : SW NI6562 mode oscilloscope
- Première démo DAQ ... en 1 semaine !
  - LabVIEW pour le pilotage de la carte
  - Désérialisation des données codée en C dans une DLL
- DAQ pour caractériser Mi 26 au labo ... en deux mois !
  - Nombreux modes de r/o à coder ...
  - Outils de monitoring en ligne des données
- Upgrade pour Télescope de 6 x Mi26 ... en deux mois !
  - Désérialisation par SW ➔ 94 % temps mort
  - Mais le temps mort n'était pas une contrainte forte
- Le coût d'un DAQ complet ~ 10 000 €
  - Châssis PXIe, CPU, Carte NI6562

Solution « Custom HW » / COTS : Cas d'un DAQ labo 3/3

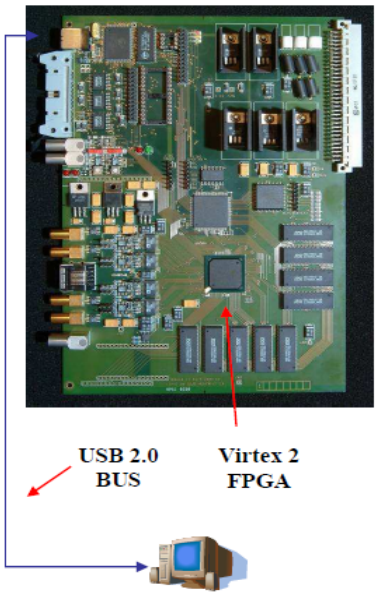
Carte USB = « Custom »
Entrées Analogiques / Numériques

Analog
- 1-4 ADC 12 bits – 40 Mhz
- 1 ADC 14 bits – 100 MHz
- 10^6 pixels **shared** 1-4 Inputs

Digital
- **Pattern** generator **& Trigger**
- 16 **Digital inputs**

Daq
- Transfer **15 MB/ s USB 2.0**

Firmware (To Be Done)
- CDS Calculation (Done)
- Pedestal substraction
- Data sparcification



- **Développement s'étale sur 4 ans ...**
- **Evolution limitée, carte dispo déjà dépassée ...**
- **Débit sur bus USB carte / PC faible ~ 15 MB/s**
- **Copie du DAQ pour collaborateurs chronophage**
  - **~ 2 Mois / an suivi de production des cartes**
- **Coût modéré ~ 3000 € DAQ complet**

Carte NI6562 = « COTS »
I/O Numériques – Bus PXIe

Carte PXI 6562 dans son châssis

- **Développement en 4 mois !**
- **Evolution ... plus que demandée Mi26, Mi28, FSBB**
- **Débit sur bus PXI carte / CPU ~ 80 MB/s**
- **Copie du DAQ pour collaborateurs aisée**
  - **Demande de devis à NI**
- **Coût significatif ~ 10 000 € DAQ complet**