

Développements autour de l'IPBUS

O. Bourrion

CNRS-IN2P3-LPSC Grenoble

1er juin 2016

- 1 Introduction
- 2 Firmware
 - Partie UDP
 - Partie IPbus
 - Limitations
 - Utilisation de l'IPBus
- 3 Interface logicielle
- 4 Utilisation au LPSC
- 5 Documents utiles

1 Introduction

2 Firmware

- Partie UDP
- Partie IPbus

- Limitations
- Utilisation de l'IPBus

3 Interface logicielle

4 Utilisation au LPSC

5 Documents utiles

Qu'est ce que l'IPBUS ?

Présentation rapide \Rightarrow développement à partager !
Documentation et codes source disponibles.

- C'est une connectivité Ethernet pour le hardware.
- Le protocole est spécifié par/pour le CERN (ou Bristol) :
 - Ensemble firmware (VHDL) et logiciel (C++)
 - Coté FPGA, passerelle Ethernet \Leftrightarrow bus A32/D32
 - Pour maximiser le débit, un paquet contient une ou plusieurs transactions
 - UDP avec couche de sécurisation
 - C'est le client (PC) qui fait les requêtes
 - Numérotation des paquets et contrôle de continuité
 - Possibilité de redemander un paquet perdu
 - Multiples paquets en transit
- Caractéristiques d'origine :
 - IPV4
 - Fonctionnalité ICMP (ping)
 - ARP (Address Resolution Protocol)
 - RARP (protocole pour l'affectation automatique d'IP par un serveur)
 - Indifférent à l'Endianess

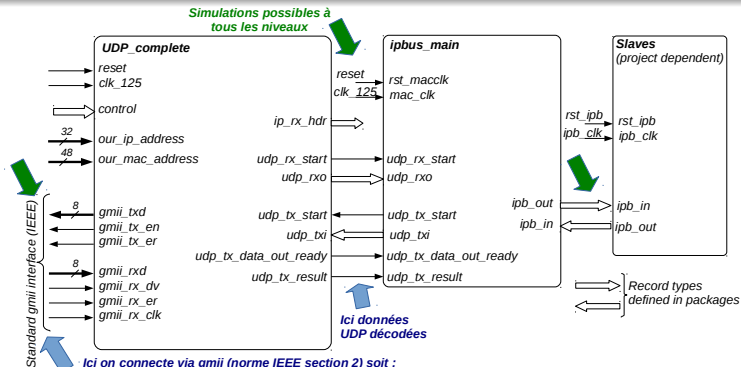
Motivations pour un IPBUS version LPSC

- ❶ Impossibilité de séparer la partie UDP de la partie IPBus
- ❷ Simulation au niveau Ethernet pas possible coté firmware
- ❸ Logiciel lourd à utiliser pour des applications de taille réduite
- ❹ Utilisation de FIFO spécifiques à Xilinx, ou d'IP payantes (MAC)

- ❶ Simulation au niveau Ethernet possible
 - ❷ Ajout de brique client DHCP et NTP.
 - ❸ Utilisation de RAM inferring pour les FIFO et description d'une MAC
- ⇒ Portable sur une autre plateforme (ALTERA), sous réserve d'instancier le lien série adéquat en 1000BASE-SX
- ❹ Ecriture d'un driver logiciel avec Qt (multiplateforme)

- ⇒ Compatibilité complète entre soft et firmware LPSC↔CERN
- ⇒ Gain sur les ressources utilisées, version LPSC ~30% plus compacte
- ⇒ Sur spartan 6 avec NTP et DHCP : ~6000FF, ~6400LUT, 35BRAM

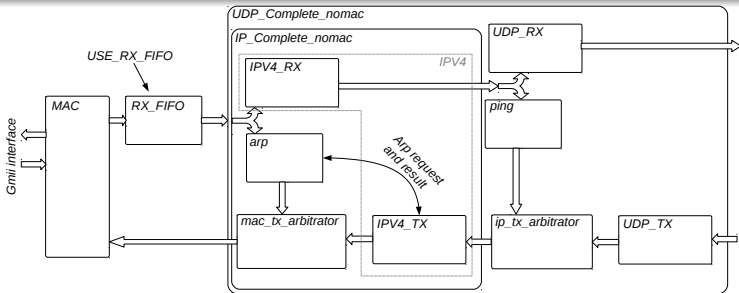
Vue générale



- Ceci est la configuration de base (sans dhcp_client, ni ntp_client)
- Génération d'horloge/reset non représentée.
- Les blocs UDP_complete et ipbus_main s'utilisent comme des IP.
- Le bloc slaves contient tous les éléments esclaves du projet. Il est différent pour chaque projet.

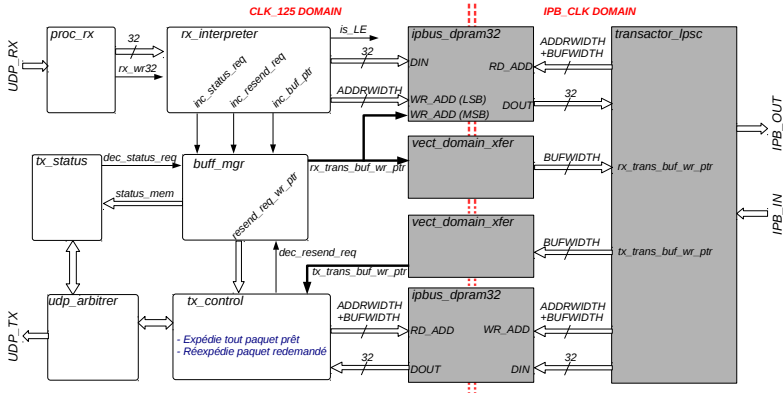
- 1 Introduction
- 2 Firmware
 - Partie UDP
 - Partie IPbus
- 3 Limitations
- 4 Utilisation de l'IPBus
- 5 Interface logicielle
- 6 Utilisation au LPSC
- 7 Documents utiles

Schéma bloc de UDP_complete



- MAC : Gère l'accès au média et génère/vérifie les checksum
- RX_FIFO est facultative (occupe 1 BRAM), voir paramétrage
- ARP :
 - Le bloc arp répond aux requêtes si l'IP du bloc est demandée
 - Lorsqu'il répond, il stocke l'adresse MAC du demandeur dans sa table
 - Si IPv4_TX veut envoyer un paquet → demande d'adresse MAC
 - Si IP connue, réponse immédiate, sinon requête ARP sur le réseau
- Arbitrators : gèrent les accès aux TX, priorité au flux de donnée
- Ping : fonctionne jusqu'à la taille maxi Ethernet (1492 octets)

Schéma bloc de la partie IPBUS



Paramètres *generic* :

- **BUFWIDTH** : nombre de buffer pour le streaming (2^{BUFWIDTH})
- **ADDRWIDTH** : taille maxi d'un buffer en word 32 ($2^{\text{ADDRWIDTH}}$), une taille de 512 words permet un MTU de 2040 octet
- **IPBUSPORT** : port UDP utilisé pour la transaction IPbus

MAC

- ne gère pas les collisions (nécessaire en half-duplex) ni les pauses.

Calculs de checksum (Ethernet)

Les blocs ci-dessous ne génèrent/calculent pas de checksum pour ne pas ajouter de latence. Ce n'est pas imposé par la norme. La MAC assure déjà une bonne émission/réception.

- IPV4_RX
- UDP_RX
- UDP_TX (champ forcé à zéro → OK en IPV4)

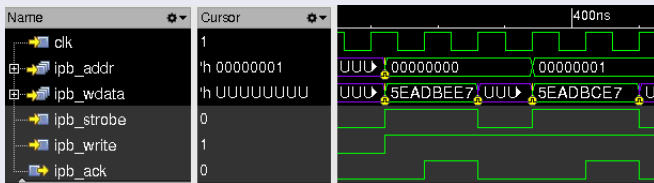
IPBus

- Ne gère qu'un seul client à la fois (Si il y a besoin de sécurisation)

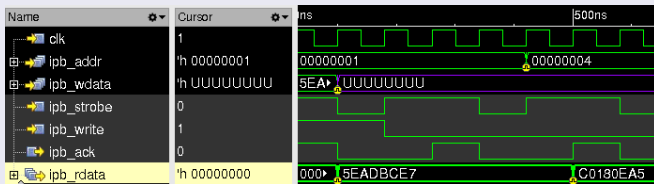
- Protocole adresse/donnée 32 bit (lecture/écriture)
- Signal de *strobe* pour valider la transaction
- Signal *ack* pour finir le cycle → possibilité de ralentir les accès (jusqu'à 255 clk)
- possibilité pour l'esclave de signaler une erreur via *err*
- Durée minimum d'un cycle IPBus : 2 ipb_clk
- Pour une même transaction, pas de temps mort pour changer d'adresse
- Au changement de transaction, 1 cycle perdu
- Au changement de paquet, plusieurs cycles perdus

Protocole de l'IPBus (coté slaves) II

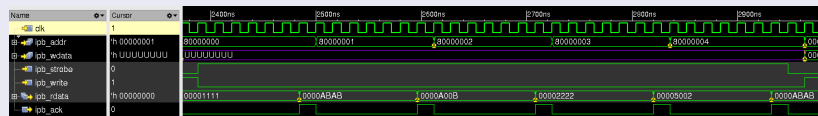
Ecriture simple



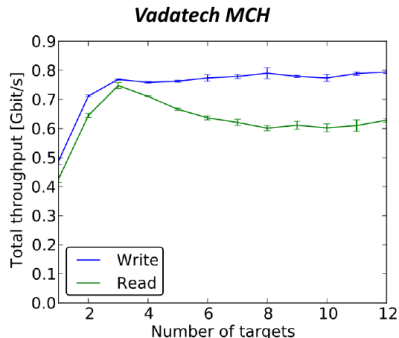
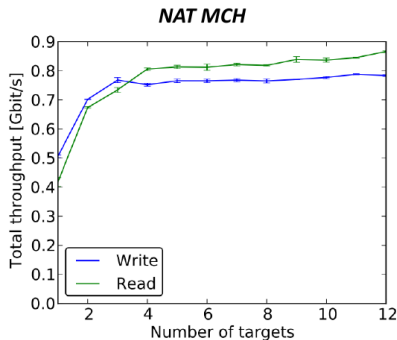
Lecture simple



Lecture multiple, ralentie par ack

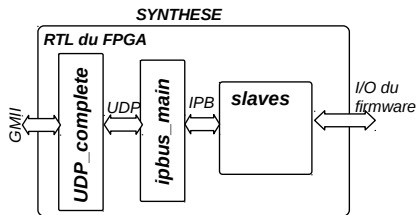
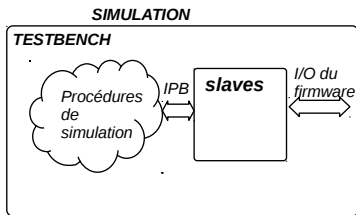


Mesures effectuées* avec firmware et logiciel CERN (600 MB transférés).



* Graphes et conditions extraits de <https://indico.cern.ch/event/299180/contributions/1659676/>, Tom Williams, TWEPP-14. Plus de données disponibles.

- Créer un composant contenant tous les esclaves et le décodeur d'adresse
 - Le composant *top* utilisant les esclaves ne doit ajouter que la couche UDP et l'interface Ethernet
 - Pour accélérer temps de simulation, simuler uniquement au niveau slave en utilisant les procédures du package de simulation.
- ⇒ C'est la possibilité de simuler sans la couche Ethernet.



- 1 Introduction
- 2 Firmware
 - Partie UDP
 - Partie IPbus
- 3 **Interface logicielle**
 - Limitations
 - Utilisation de l'IPBus
- 4 Utilisation au LPSC
- 5 Documents utiles

En plus du setup CERN, il existe une bibliothèque logicielle Qt (LPSC)
Peut être compilée dans le projet ou liée si vous créez un *.so ou un *.dll

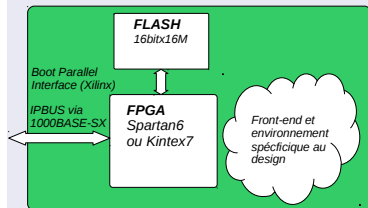
Méthodes publiques de la classe TIPbusClient.h

```
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();
```

- Une méthode de connexion à la cible
- 6 méthodes pour créer des transactions
- Une méthode pour forcer l'envoi d'un paquet IPBus même si il n'est pas rempli de transactions.

- 1 Introduction
- 2 Firmware
 - Partie UDP
 - Partie IPbus
- 3 Limitations
- 3 Utilisation de l'IPBus
- 3 Interface logicielle
- 4 **Utilisation au LPSC**
- 5 Documents utiles

Plateforme de base



⇒ Flash pour firmware et numéro MAC

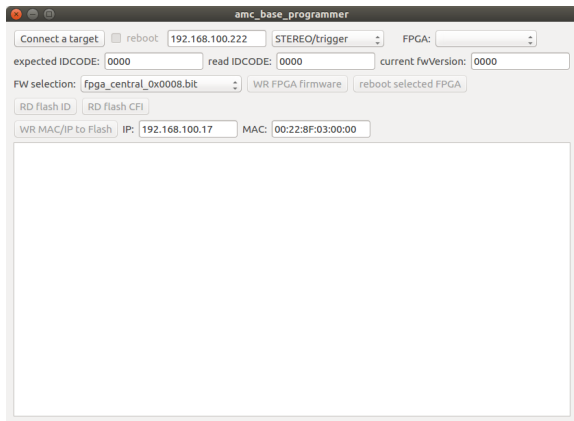
⇒ Un slave pour contrôler la flash

⇒ Un logiciel de programmation maison

Projets :

- NIKA, radio-astronomie millimétrique :
 - 3 châssis MTCA
 - 20 cartes de readout, 3 cartes de timing.
- STEREO, expérience neutrino :
 - 1 châssis MTCA
 - 10 cartes front-end, 1 cartes de LED, 1 carte DAQ
 - Voir exposé suivant de C. Vescovi

En plus des logiciels DAQ spécifiques aux projets, un logiciel commun pour la programmation in-situ (firmware et/ou IP et/ou MAC).



- Le logiciel de programmation se déploie avec l'application
- Identification des firmware par numéro (!effacement accidentels)

- 1 Introduction
- 2 Firmware
 - Partie UDP
 - Partie IPbus
- 3 Limitations
- 4 Utilisation de l'IPBus
- 5 Interface logicielle
- 6 Utilisation au LPSC
- 7 Documents utiles

- Wiki IPBus <https://svnweb.cern.ch/trac/cactus/wiki>
- Protocole IPBus :
<https://svnweb.cern.ch/trac/cactus/browser/trunk/doc>
- LogiCORE IP Tri-Mode Ethernet MAC user guide (ug777)
- Opencore de la MAC :
http://opencores.org/project,ethernet_tri_mode
- IEEE standard for Ethernet section one (802.3-2012) : Spécification de la couche MAC
- IEEE standard for Ethernet section two (802.3-2012) : Spécification de l'interface gmii
- Opencore pour la couche UDP :
http://opencores.org/project,udp_ip_stack
- Dépôt SVN <https://lpsc-secure.in2p3.fr/svn/akido/Ethernet/>
(login : guest, passwd : guest)
- Obtention d'une plage d'adresse MAC par le CNRS
<http://www.mrct.cnrs.fr/spip.php?article21>