Ethernet 1GbEth sur FPGA, présentation de l'IPBUS

O. Bourrion

CNRS-IN2P3-LPSC Grenoble

26 sept 2019

Plan

Plan

Qu'est ce que l'IPBUS ?

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

- C'est une connectivité Ethernet pour le hardware.
- Le protocole a été spécifié par/pour le CERN (ou Bristol) :
  - Ensemble firmware (VHDL) et logiciel (C++)
  - Coté FPGA, passerelle Ethernet ⇔ 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

1. Impossibilité de séparer la partie UDP de la partie IPBus
2. Simulation au niveau Ethernet pas possible coté firmware
3. Logiciel lourd à utiliser pour des applications de taille réduite
4. Utilisation de FIFO spécifiques à Xilinx, ou d'IP payantes (MAC)

1. Simulation au niveau Ethernet possible
2. Ajout de briques client DHCP et NTP.
3. Utilisation de RAM inferring pour les FIFO et description d'une MAC

$\Rightarrow$ Portable sur une autre plateforme (ALTERA), sous réserve d'instancier le lien série adéquat en 1000BASE-SX

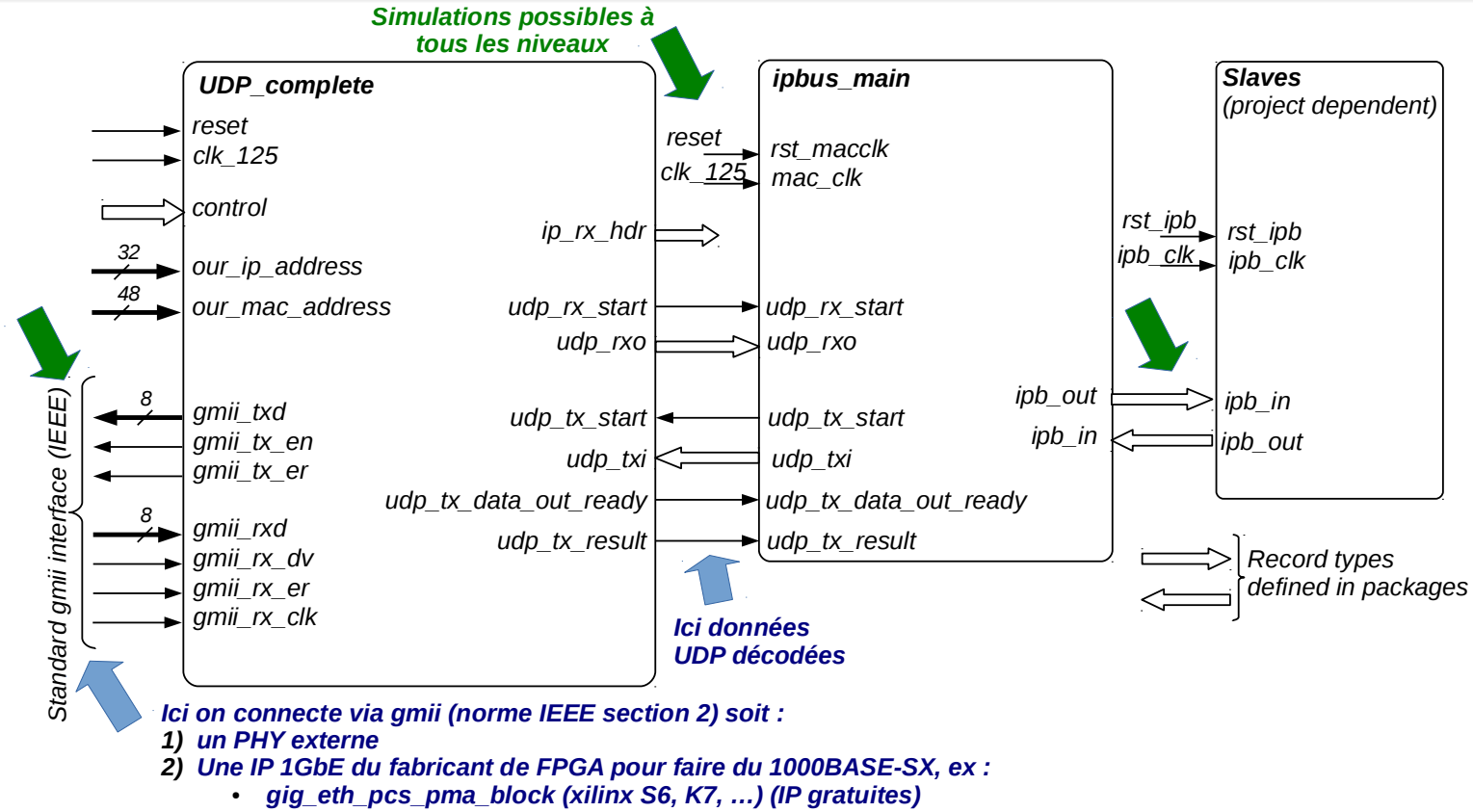
4. Ecriture d'un driver logiciel avec Qt (multiplatforme)

$\Rightarrow$ Compatibilité complète entre soft et firmware LPSC$\leftrightarrow$CERN

$\Rightarrow$ Gain sur les ressources utilisées, version LPSC $\sim$30% plus compacte

$\Rightarrow$ Sur spartan 6 avec NTP et DHCP : $\sim$6000FF, $\sim$6400LUT, 35BRAM

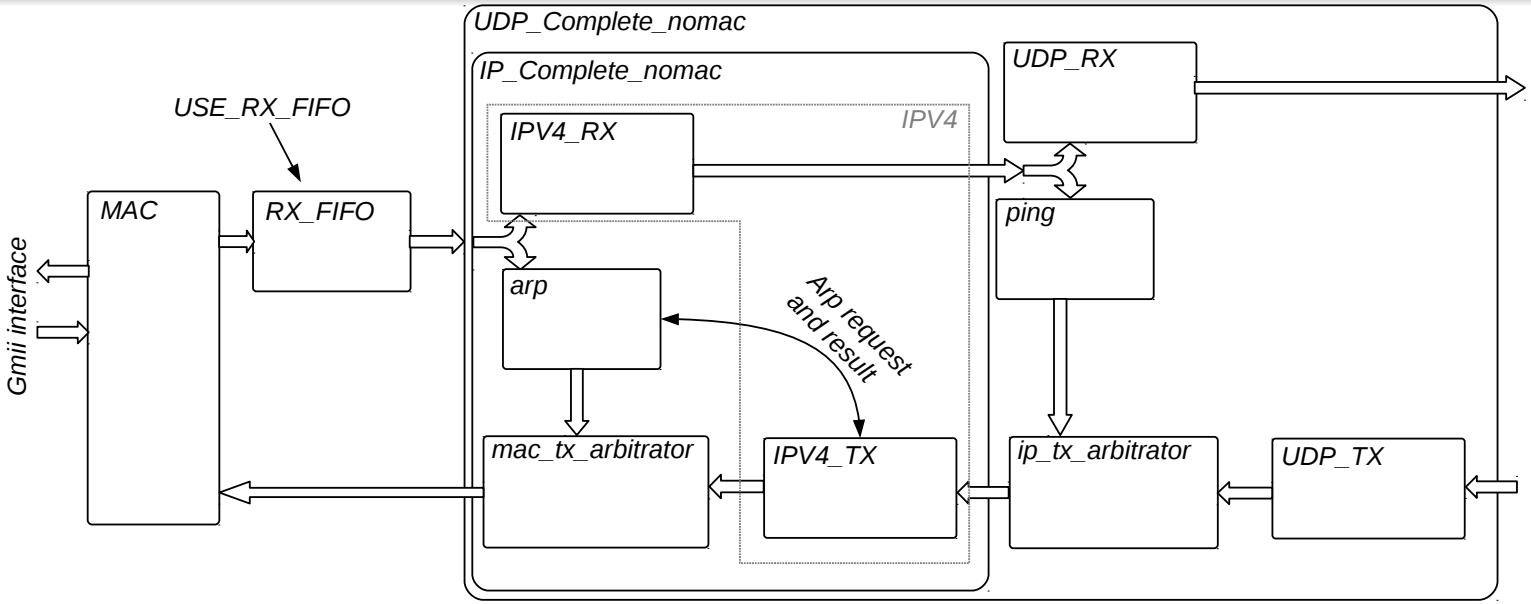
Vue générale



- 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.

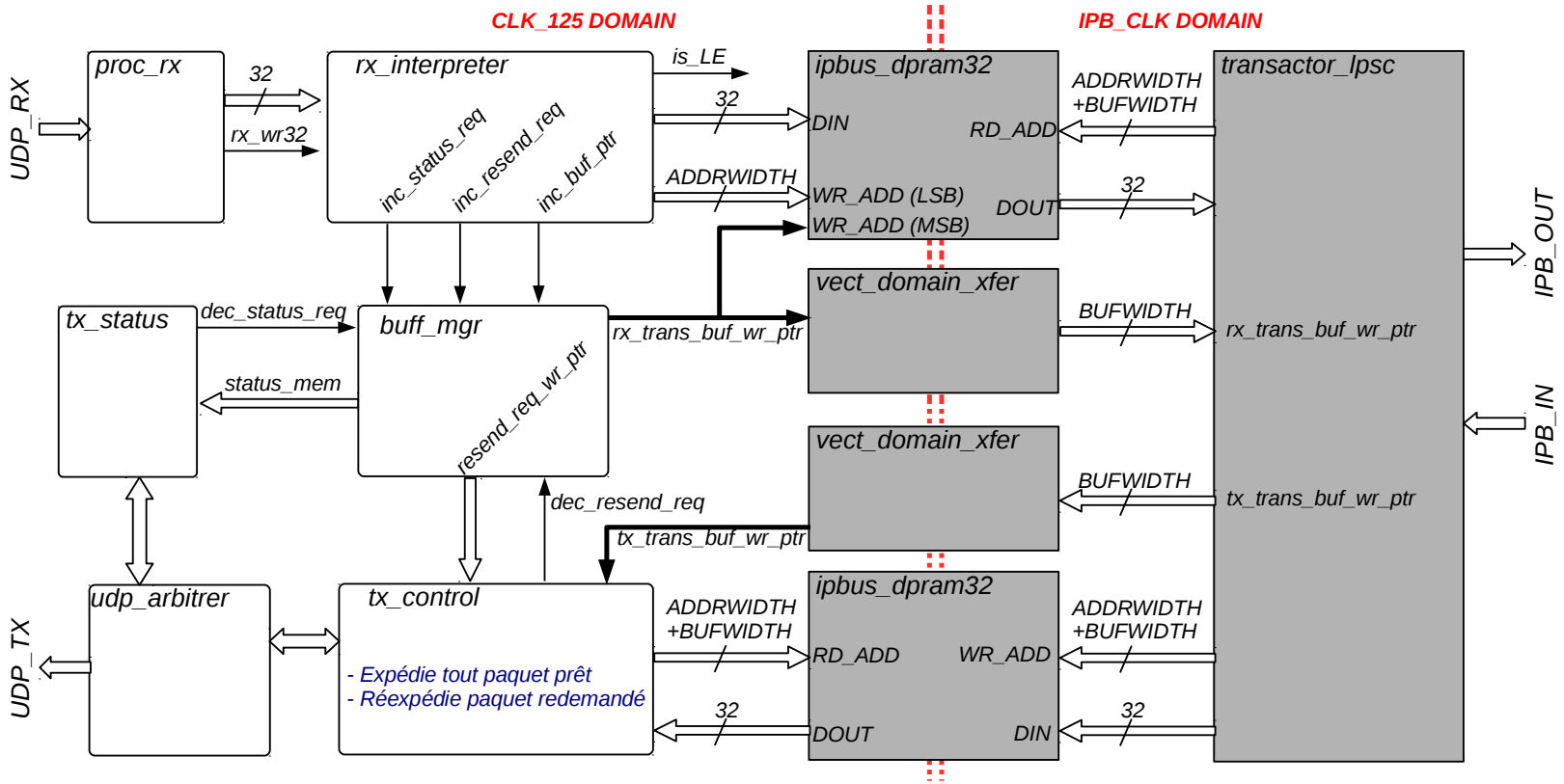
Plan

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

Limitations

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 $\rightarrow$ OK en IPV4)

IPBus

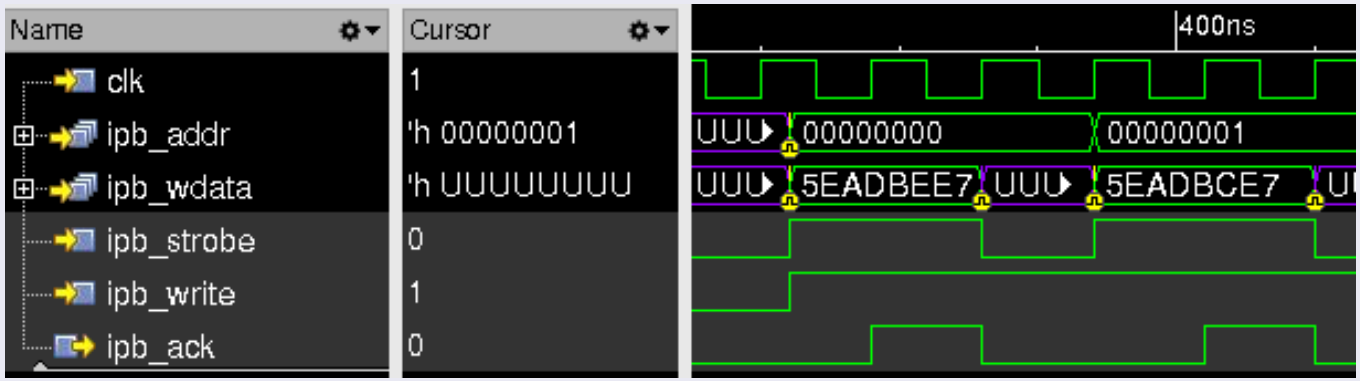
- Ne gère qu'un seul client à la fois (Si il y a besoin de sécurisation)

Protocole de l'IPBus (coté slaves) I

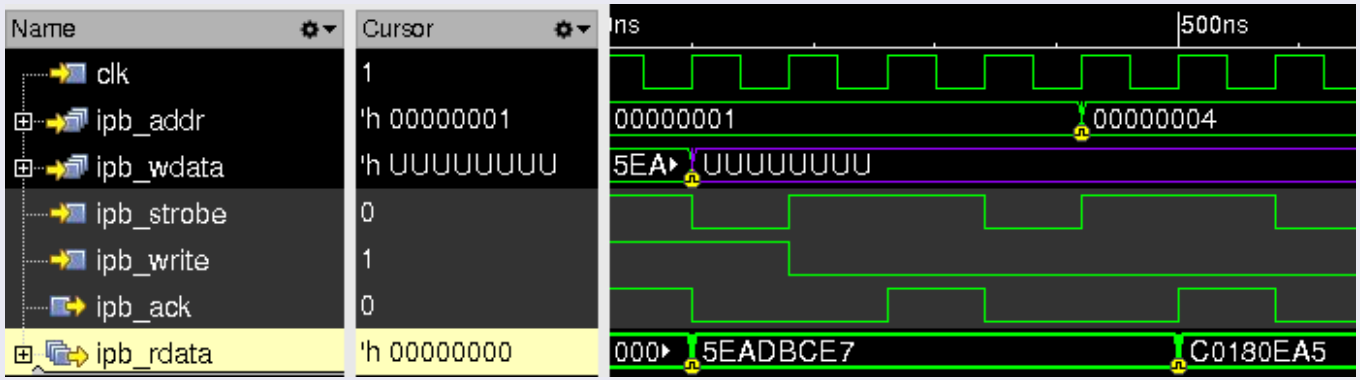
- Protocole adresse/donnée 32 bit (lecture/écriture)
- Signal de *strobe* pour valider la transaction
- Signal *ack* pour finir le cycle $\rightarrow$ possibilité de ralentir les accès (jusqu'à 255 clk)
- possibilité pour l'esclave de signal une erreur via *err*
- Durée minimum d'un cycle IPBus : 2 ipb_clk
- 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

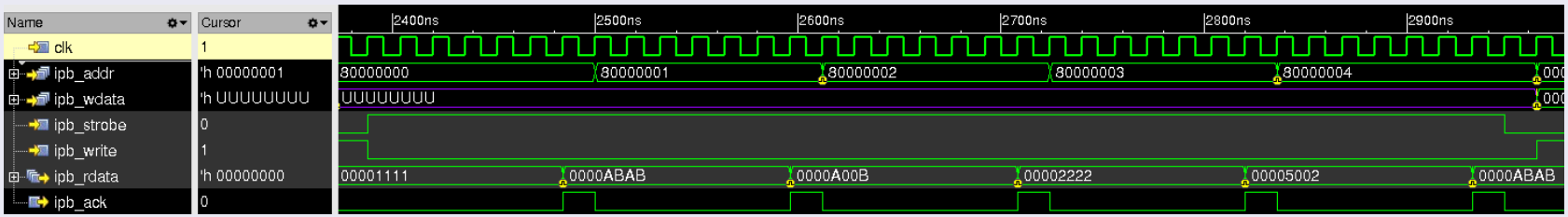


Lecture simple



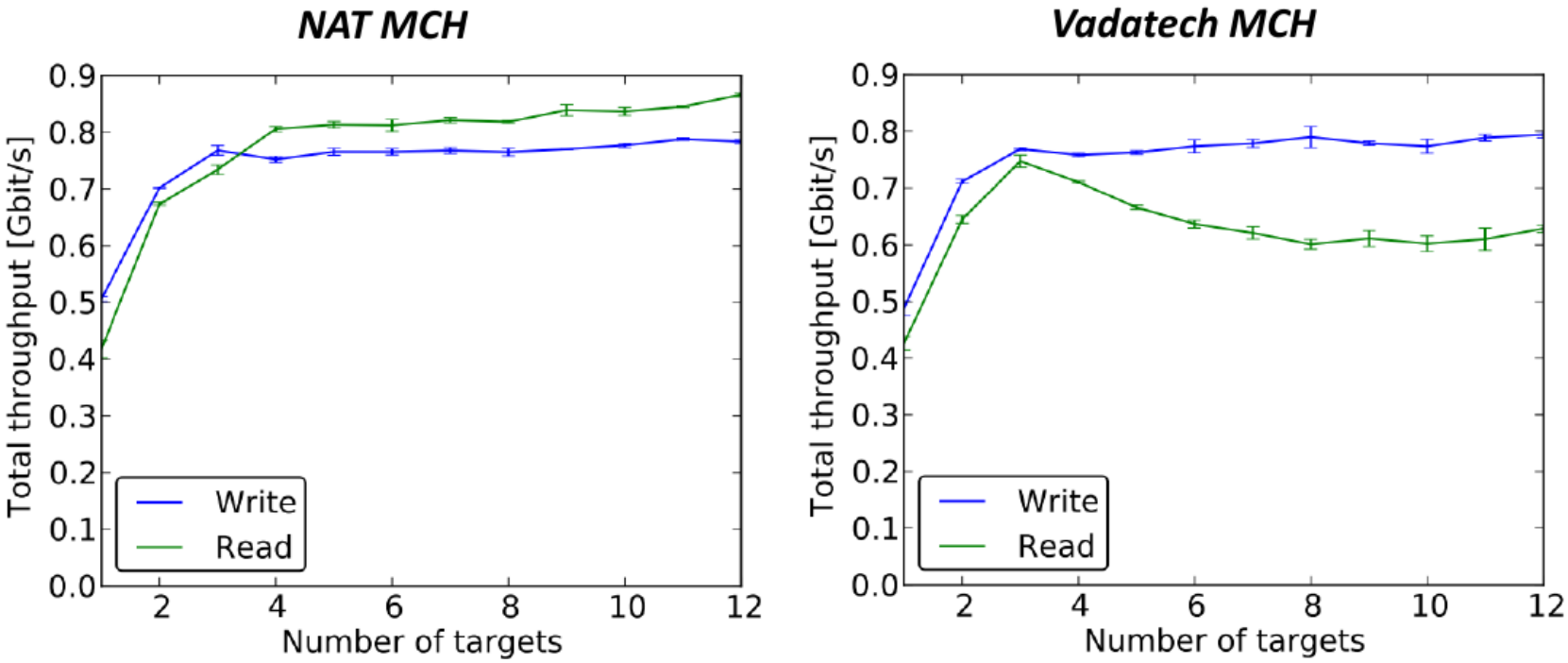
Protocole de l'IPBus (coté slaves) III

Lecture multiple, ralentie par ack



Performances

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

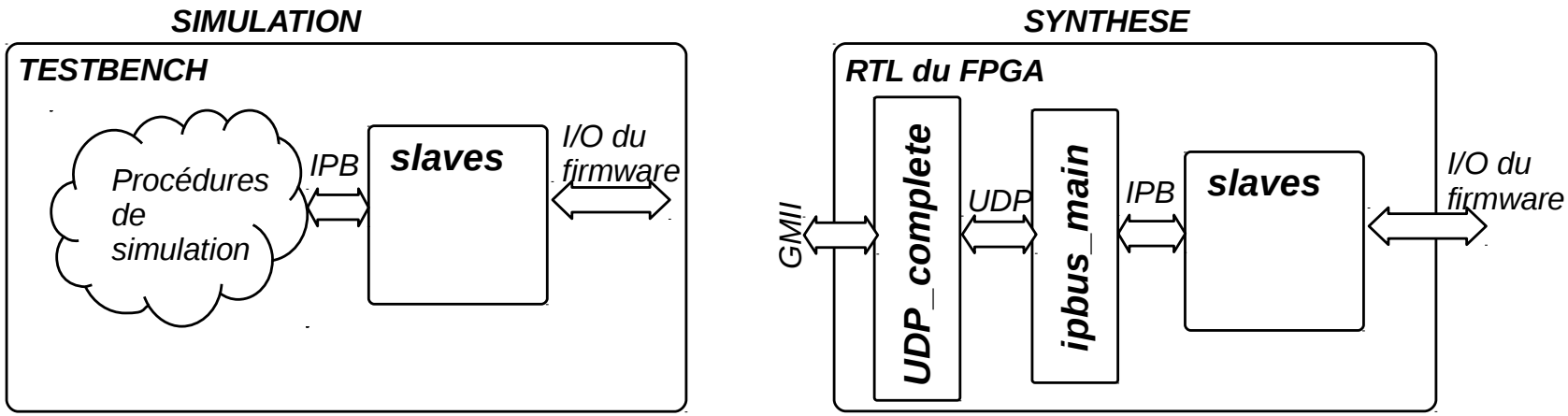


* Graphes et conditions extraits de https://indico.cern.ch/event/299180/contributions/1659676/, Tom Williams, TWEPP-14. Plus de données disponibles.

Mise en œuvre

- 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 (et la couche DHCP si souhaitée)
- Pour accélérer temps de simulation, simuler uniquement au niveau slave en utilisant les procédures du package de simulation.

$\Rightarrow$ C'est la possibilité de simuler sans la couche Ethernet.



Plan

Interface logicielle

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

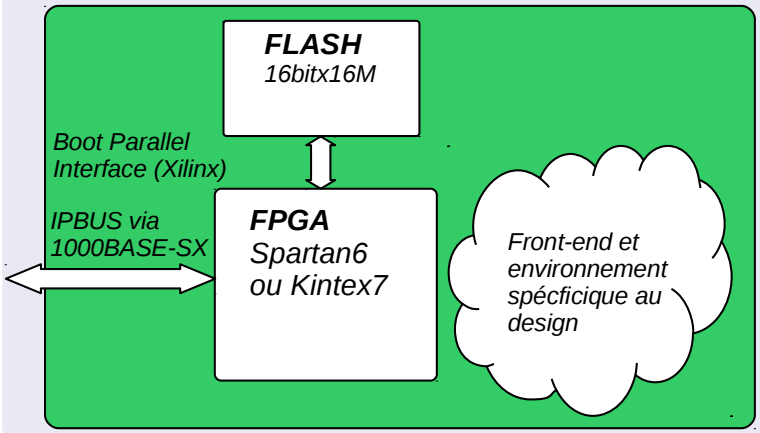
```
void connectClient(QString addStr, int port);
void read_std(int add, quint32 *buff,int size);
void read_ni(int add, quint32 *buff,int size);
void rmw_bit(int add, quint32 *buff,quint32 ANDterm,quint32 ORterm);
void rmw_sum(int add, quint32 *buff,quint32 A);
void write_std(int add, quint32 *buff,int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

Plan

Utilisation au LPSC

Plateforme de base



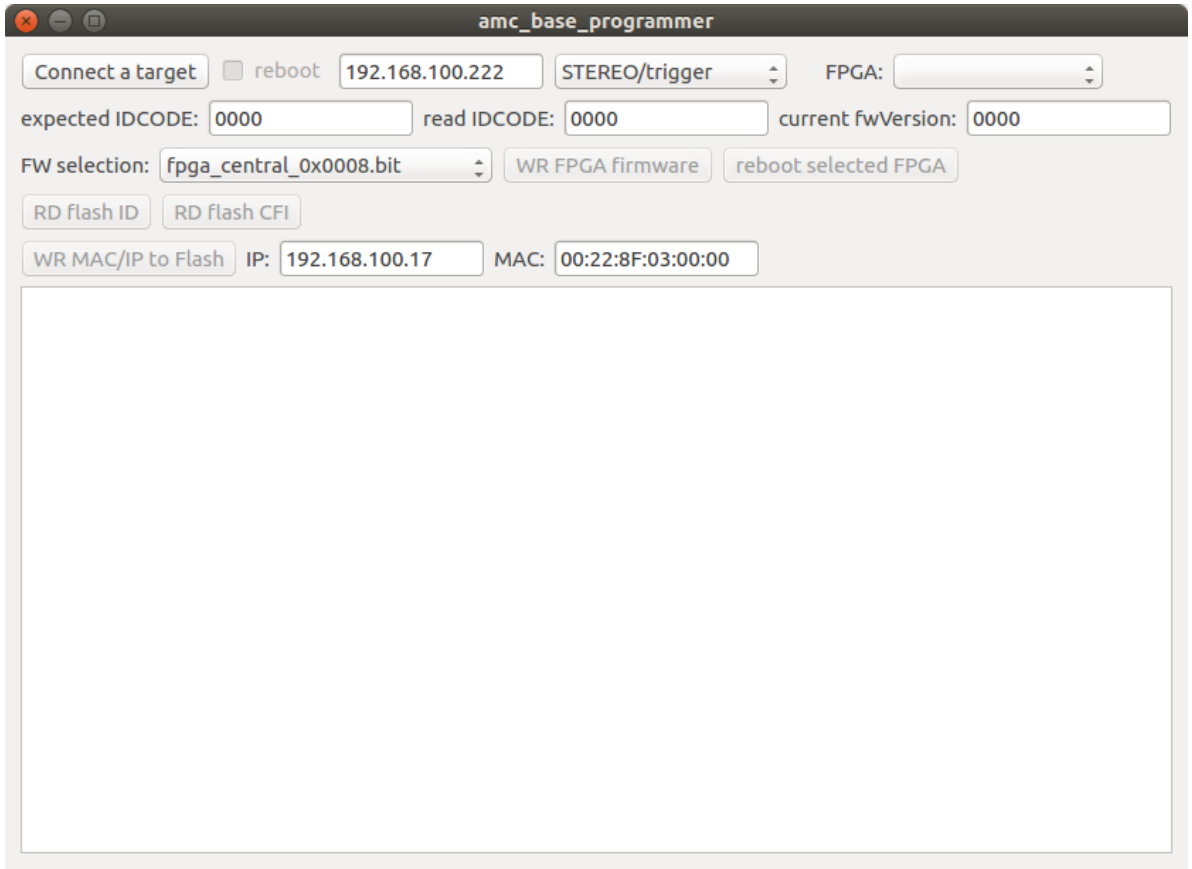
⇒Flash pour firmware et numéro MAC
⇒Un slave pour contrôler la flash
⇒Un logiciel de programmation maison

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

Logiciel de programmation in-situ

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)

Plan

Documents utiles

- Wiki IPBus http://ipbus.web.cern.ch/ipbus/
- 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
- Dépot firmware https://gitlab.in2p3.fr/AKIDO/ipbus-fw (accès à demander)
- Dépot software https://gitlab.in2p3.fr/AKIDO/ipbus-sw (accès à demander)
- Obtention d'une plage d'adresse MAC par le CNRS http://www.mrct.cnrs.fr/spip.php?article21