



Rapport d'activité

Novembre 2002 – Août 2004

**Braise TRINCAZ
IUT GEII**

Administration de système
Mise en place d'un protocole de communication par interface VME
Elaboration d'un fréquencemètre



Responsable d'apprentissage à l'IUT : Monsieur Joël SENPAU ROCA
Responsable d'apprentissage au CEA : Monsieur Denis CALVET

REMERCIEMENTS

Je tiens tout d'abord à remercier Monsieur Joseph Lefoll pour avoir accepté d'être mon tuteur au CEA et pour l'accueil qu'il m'a réservé.

Je remercie également Monsieur Denis Calvet, mon chef de projet, pour m'avoir encadré dans un travail totalement nouveau pour moi.

Mes remerciements vont aussi à mes collègues du CEA qui m'ont aidé lorsque je rencontrais des difficultés d'ordre technique, et particulièrement à Madame Zoulikha Georgette qui m'a beaucoup soutenu lors de ces deux années passées à Saclay.

Dans le milieu universitaire, je remercie Monsieur Joël Senpau Roca, qui m'a accueilli à l'IUT de Cachan, ainsi que toute l'équipe pédagogique de la formation par apprentissage de cet IUT.

Résumé

Le stage de deux années, présenté ici, a été effectué au Commissariat à l'Energie Atomique (CEA) de Saclay dans le cadre de la formation par apprentissage de l'IUT GEII de Cachan. Les travaux ont été réalisés au sein du Département des Sciences de la Matière, plus précisément au Département d'Astrophysique, de Physique des Particules, de Physique Nucléaire et de l'Instrumentation Associée, le DAPNIA.

Les premiers mois ont porté sur un projet à l'échelle européenne, le Datagrid, permettant le stockage et l'utilisation de calcul au moyen de plusieurs ordinateurs reliés entre eux. Il s'agissait de faire de l'administration de système

Ensuite, le travail s'est centré sur un projet à l'échelle internationale, concernant l'amélioration d'un détecteur de physique des particules (appelé D0) utilisant les collisions d'un gigantesque accélérateur (le Tevatron), situé au Fermilab dans la banlieue de Chicago. Dans un premier temps, il fallait mettre en place un protocole de communication par interface VME en langage C, puis concevoir un fréquencemètre en VHDL.

Abstract

It was in CEA of Saclay that two years in apprenticeship were passed, more precisely in the Department of Astrophysics, Particles physics, Nuclear physics and the Associated Instrumentation, the DAPNIA.

During the first months, the work was related to an European project called Datagrid, including the storage and the power of many computers connected in a network, It was necessary to protect the system. Then During one year and half the work rose on an international project: a detector (called D0) in an accelerator (Tevatron), located in Fermilab, in the suburb of Chicago. The work consisted to program a communication VME in language C, then to realize a frequency meter in VHDL.

Table des matières

I.	INTRODUCTION.....	6
II.	L'ENTREPRISE ET SON PROJET	7
A.	LE CEA.....	7
1.	<i>Historique.....</i>	7
2.	<i>Les différents secteurs d'activités du CEA.....</i>	7
a)	Des activités très diversifiées	7
b)	Le DAPNIA.....	9
B.	LA PHYSIQUE DES PARTICULES ET L'EXPÉRIENCE D0	9
1.	<i>Physique des particules.....</i>	11
2.	<i>Le collisionneur Tevatron et l'expérience D0.....</i>	11
a)	Le Tevatron	11
b)	Le détecteur D0	12
III.	LA GRILLE DE CALCUL, UN OUTIL POUR LES PHYSICIENS	14
A.	INTRODUCTION.....	14
1.	<i>Le DataGrid</i>	14
2.	<i>Les acteurs du projet.....</i>	14
3.	<i>Les utilisations du DataGrid.....</i>	15
B.	ADMINISTRATION DE SYSTÈME.....	16
IV.	TRAVAIL SUR L'EXPÉRIENCE D0	17
A.	MISE EN PLACE D'UN PROTOCOLE DE COMMUNICATION PAR INTERFACE VME	17
1.	<i>Programmation du fichier principal Mpp.....</i>	18
2.	<i>Réalisation de bibliothèques sous Lynx OS</i>	19
B.	ELABORATION D'UN FRÉQUENCEMÈTRE.....	21
1.	<i>Contexte.....</i>	21
2.	<i>Description du projet</i>	21
3.	<i>Cahier des charges.....</i>	22
4.	<i>Travail demandé.....</i>	22
5.	<i>Outils disponibles.....</i>	23
6.	<i>Etude du fréquencemètre.....</i>	23
7.	<i>Intégration du fréquencemètre dans le testeur de liens à haut débit</i>	25
V.	BILAN DE MA FORMATION EN APPRENTISSAGE.....	28
VI.	ANNEXES.....	29
A.	ANNEXE 1 : TRIPWIRE	29
B.	ANNEXE 2 : PROGRAMME DE COMMANDE DU BUS VME	34
C.	ANNEXE 3 : PROGRAMME MPP TOURNANT SOUS DOS AVEC L'INTERFACE PCI/VME ..	41
D.	ANNEXE 4 : MAKEFILE SERVANT À LA CRÉATION DE BIBLIOTHÈQUES SOUS LYNX OS	43
E.	ANNEXE 5 : CONCEPTION DU FRÉQUENCEMÈTRE EN VHDL.....	48
F.	ANNEXE 6 : BANC DE TEST DU FRÉQUENCEMÈTRE.....	50
G.	ANNEXE 7 : CONCEPTION DU SIGNAL MODULE EN FRÉQUENCE.....	53

I. Introduction

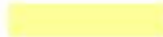
Durant mes deux années de formation en alternance à l'IUT de Génie Electrique et Informatique Industrielle (GEII) de Cachan, j'ai travaillé au Commissariat à l'Energie Atomique (le CEA) situé à Saclay dans le Département d'Astrophysique, de Physique des Particules, de Physique Nucléaire et de l'Instrumentation Associée, le DAPNIA.

Mon travail a été constitué de trois projets différents, tous reliés à la physique des hautes énergies (ou physique des particules).


Tout d'abord, durant quatre mois mon travail a porté sur le Datagrid, un projet à l'échelle européenne, permettant le stockage et l'utilisation de calcul au moyen de plusieurs ordinateurs reliés entre eux.

J'ai ensuite travaillé durant un an et demi sur l'amélioration d'un détecteur (appelé D0) de physique des particules utilisant les collisions d'un gigantesque accélérateur (le Tevatron), situé au Fermilab dans la banlieue de Chicago.

2002	2003	2004
	Janvier	Janvier
	Février	Février
	Mars	Mars
	Avril	Avril
	Mai	Mai
	Juin	Juin
	Juillet	Juillet
	Août	Août
	Septembre	
	Octobre	
Novembre	Novembre	
Décembre	Décembre	

 Datagrid : administration de système

 D0 : mise en place d'un protocole de communication par interface VME

 D0 : élaboration d'un fréquencemètre

II. L'entreprise et son projet

A. Le CEA

1. Historique

Le 18 octobre 1945, le Général de Gaulle crée le Commissariat à l'Energie Atomique (le CEA). Dès les premiers jours, Frédéric Joliot-Curie, Haut-Commissaire du CEA, et Raoul Dautry, Administrateur Général, décident la création d'un grand centre de recherche nucléaire situé à Saclay.

Son cadre de vie, entre la Bièvre et l'Yvette, ainsi que la proximité des universités, des écoles parisiennes et du CNRS semblent particulièrement adapté. En trois ans, toutes les disciplines se développent : physique et chimie nucléaire, biologie, médecine, technologie. Pendant vingt ans, le CEA de Saclay domine largement la recherche nucléaire en France.



CEA Saclay

Au fil du temps, le CEA s'engage plus intensément dans des coopérations nationales et internationales, par exemple dans le domaine de la physique.

Le CEA est reconnu comme un expert dans ses domaines de compétences, mais aussi comme un moteur de l'innovation et de la diffusion technologique grâce à son implication dans le tissu industriel et économique et grâce à de nombreux partenariats.

2. Les différents secteurs d'activités du CEA

a) Des activités très diversifiées

Les missions du CEA sont nombreuses et adaptées aux problématiques actuelles :

- Assurer une énergie plus compétitive et plus respectueuse de l'environnement.
Dans le domaine de l'électronucléaire, le CEA contribue à l'amélioration des performances du parc actuel de réacteurs et participe aux développements futurs.



Le CEA conduit ainsi des actions de recherche en partenariat avec les industriels pour répondre aux objectifs stratégiques d'allongement de la durée de vie des installations et d'amélioration des performances du combustible, incluant le retraitement/recyclage des combustibles usés afin d'en valoriser le contenu énergétique

Dans le domaine des économies d'énergie et de l'amélioration du rendement des systèmes de production d'énergie, le CEA met sa longue expérience de la modélisation numérique, des systèmes programmés intelligents, et de la microélectronique appliquée aux équipements thermiques à haute efficacité, au service des nouvelles technologies du froid et à la réduction des pollutions dans les systèmes énergétiques.

La maîtrise de la fusion thermonucléaire contrôlée pourrait permettre, dans l'avenir, de disposer d'une source quasi infinie d'énergie. Le CEA est un acteur clé des recherches dans ce domaine, notamment par son implication dans les études du projet international ITER (International Thermonuclear Experimental Reactor), étape essentielle vers la conception d'un réacteur de fusion électrogène.

- Contribuer à la défense

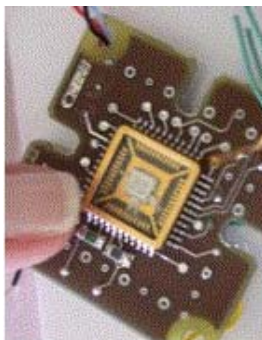


Dans le cadre de la politique de dissuasion nucléaire française, le CEA est chargé de la conception, de la fabrication, du maintien en condition opérationnelle et du démantèlement des têtes nucléaires. Par ailleurs, il conçoit et entretient les réacteurs nucléaires assurant la propulsion des bâtiments de la Marine nationale.

Dans ce cadre, le CEA s'équipe de supercalculateurs, outils indispensables à la simulation numérique et dispose aujourd'hui de la machine la plus puissante d'Europe.

- Mettre la recherche technologique au service de l'industrie

- Technologies de l'information



Les technologies de l'information font l'objet, au CEA, d'un important programme de recherche et développement, centré principalement sur les micro et nanotechnologies, l'instrumentation des systèmes complexes, et les technologies logicielles.

En micro et nanotechnologies, le CEA poursuit des travaux sur la technologie Silicium sur Isolant (SOI), l'évolution des circuits intégrés, ainsi que sur de nouveaux dispositifs innovants, comme les transistors à effet de champ.

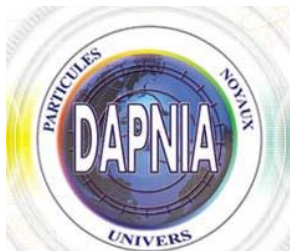
En instrumentation et technologies logicielles, l'objectif est de constituer de nouveaux systèmes complexes interconnectés conciliant à la fois robustesse,

fiabilité, confidentialité et convivialité. Le CEA travaille notamment le contrôle non destructif, la robotique interactive et les interfaces homme-machine.

- Technologies pour la santé
- Elargir les connaissances scientifiques
Indispensable à l'avancée des connaissances scientifiques, la recherche fondamentale joue un rôle central au sein du CEA. Outre les études amont dans les domaines de l'énergie, de la défense et de l'industrie, il poursuit différents programmes en sciences de la matière, sciences du vivant et sciences de l'environnement.

Mon travail au CEA a été réalisé au sein du département des sciences de la matière dans lequel les recherches portent sur la structure de la matière, à petite et à grande échelle. Elles s'articulent autour de la physique des particules, la physique nucléaire, les nano sciences et l'astrophysique. Le CEA collabore également à plusieurs programmes d'observation de l'Agence Spatiale Européenne (ESA). J'ai plus précisément travaillé au DAPNIA (Département d'Astrophysique, de Physique des Particules, de Physique Nucléaire et de l'Instrumentation Associée).

b) Le DAPNIA



Les physiciens du DAPNIA s'efforcent de répondre à plusieurs interrogations fondamentales : de quoi est fait le monde qui nous entoure, comment a évolué notre univers depuis le Big-Bang... Pour répondre à ces questions, des expériences sont construites, nécessitant des collaborations entre techniciens, ingénieurs et physiciens.

Le dapnia regroupe sept services (3 services de physique et 5 services d'électroniques) :

- Le SAp (Service d'Astrophysique) étudie les corps célestes sur une large gamme de longueurs d'onde, avec un fort accent porté sur les hautes énergies (rayons X et gamma) et l'infrarouge.
- Le SPhN (Service de Physique Nucléaire) est un service de recherche fondamentale en physique nucléaire expérimentale.
- Le SPP (Service de Physique des Particules) étudie les constituants ultimes de la matière et les interactions qui les régissent
- Le SDA (Service de Déclassement des accélérateurs) est chargé des opérations de démantèlement, d'assainissement et de déclassement des installations de l'Accélérateur Linéaire de Saclay.
- Le SACM (Service des Accélérateurs de Cryogénie et de Magnétisme) développe et met en œuvre de nouveaux concepts en matière d'accélérateurs de particules.
- Le SIS (Service d'Ingénierie des Systèmes) assure la conception des appareillages, systèmes ou équipements relevant des programmes et des projets du DAPNIA.
- Le SEDI (Service d'Electronique des Détecteurs et d'Informatique) effectue des recherches et développe de nouveaux concepts dans les domaines de la détection des particules ou du rayonnement, de l'électronique d'acquisition, ainsi que des outils logiciels associés. Il assure la conception, le suivi de la réalisation et la mise en œuvre d'ensembles de détection dans le cadre des programmes du département. Il conçoit et développe les circuits électroniques et les

logiciels destinés à acquérir et à traiter en « temps réel » les données issues de ces mêmes ensembles de détection. Il conçoit et développe les outils logiciels spécifiques destinés à l'analyse des données issues des expériences de physique. Il conçoit, développe, met en œuvre et maintient les moyens de caractérisation et d'essais des chaînes de détection. Il assure en outre le développement et la maintenance de l'environnement informatique nécessaire aux activités du département (serveurs, réseaux, bureautique, intranet ...), ainsi que l'approvisionnement et la gestion des composants électroniques utilisés sur les projets et expériences.

C'est au sein de ce service que j'ai effectué mon apprentissage.

B. La physique des particules et l'expérience D0

1. Physique des particules

Cette discipline s'attache à décrire la structure de la matière. Une théorie appelée le modèle standard a été développée et actuellement, elle n'a pas été mise en défaut par les expériences chargées de la tester et de la valider (ou de l'invalider). Les grandes idées de cette théorie sont les suivantes :

- Les grains de matière sont appelés quarks et leptons. Il existe 6 quarks et 6 leptons ainsi que 6 anti-quarks et 6 anti-leptons. Les deux quarks les plus légers composent les protons et les neutrons qui se trouvent dans les noyaux des atomes et l'électron qui se trouve aussi dans les atomes est un lepton léger.
- Ces particules interagissent à travers 4 forces fondamentales : l'électromagnétisme, la force forte, la force faible et la gravitation. Pour chacune de ces forces, il existe des particules messagères. Lorsque des grains de matière interagissent ils échangent ces particules messagères.

Les particules les plus massives n'existent pas sur terre à l'état naturel car elles se désintègrent très rapidement en particules plus légères. Pour les étudier, les physiciens doivent les produire. Pour ce faire, on utilise le fait que la masse n'est qu'une forme d'énergie. C'est la signification de la fameuse égalité d'Albert Einstein $E = mc^2$, où E est l'énergie, m la masse et c la vitesse de la lumière.

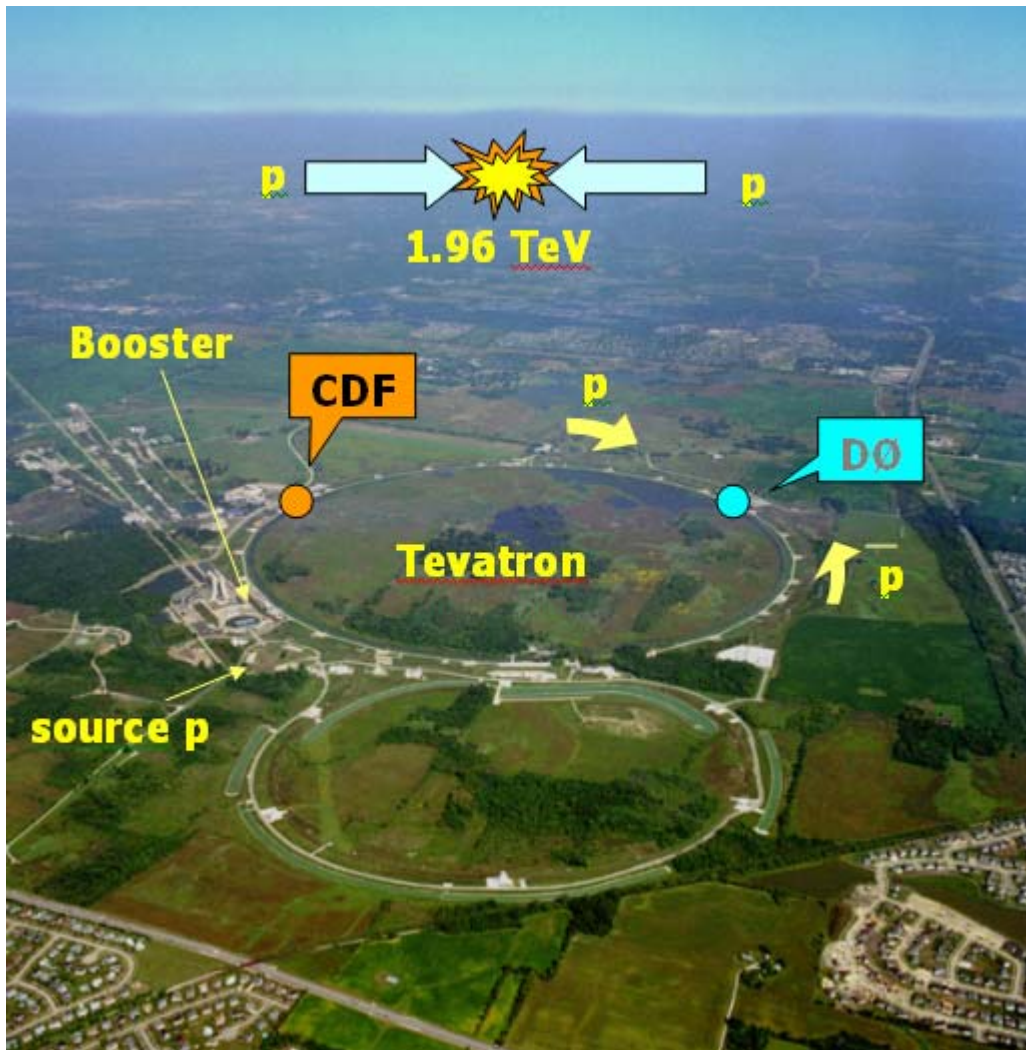
Pour produire ces particules plus massives, il faut accélérer des particules de faible masse (comme des protons ou des électrons) afin de leur donner une vitesse importante donc de l'énergie cinétique et de provoquer des collisions. Durant ces collisions, cette énergie cinétique est convertie en énergie de masse pour former des particules massives. Celles-ci sont ensuite étudiées grâce à des détecteurs situés au point de collision. Ces détecteurs enregistrent les produits des désintégrations des particules massives créées lors des collisions.

Le Tevatron est un accélérateur dans lequel ce sont des protons et des antiprotons qui entrent en collision. Au cours de mon stage, j'ai travaillé sur le détecteur D0 qui utilise les collisions du Tevatron.

2. Le collisionneur Tevatron et l'expérience D0

a) Le Tevatron

Le Fermi National Accelerator Laboratory est un laboratoire de physique des hautes énergies situé à 70km à l'ouest de Chicago. Il abrite l'accélérateur de particules le plus puissant du monde à l'heure actuelle, le Tevatron, qui a été mis en service en 1985 afin de découvrir le quark le plus massif (appelé le quark top). Deux détecteurs, nommés CDF et D0 sont situés sur le Tevatron. Après une première phase de prise de données de 1990 à 1996 qui a conduit à la découverte du quark top, l'accélérateur a été amélioré pour obtenir une énergie de collision plus importante et un nombre de collision par seconde plus élevé.



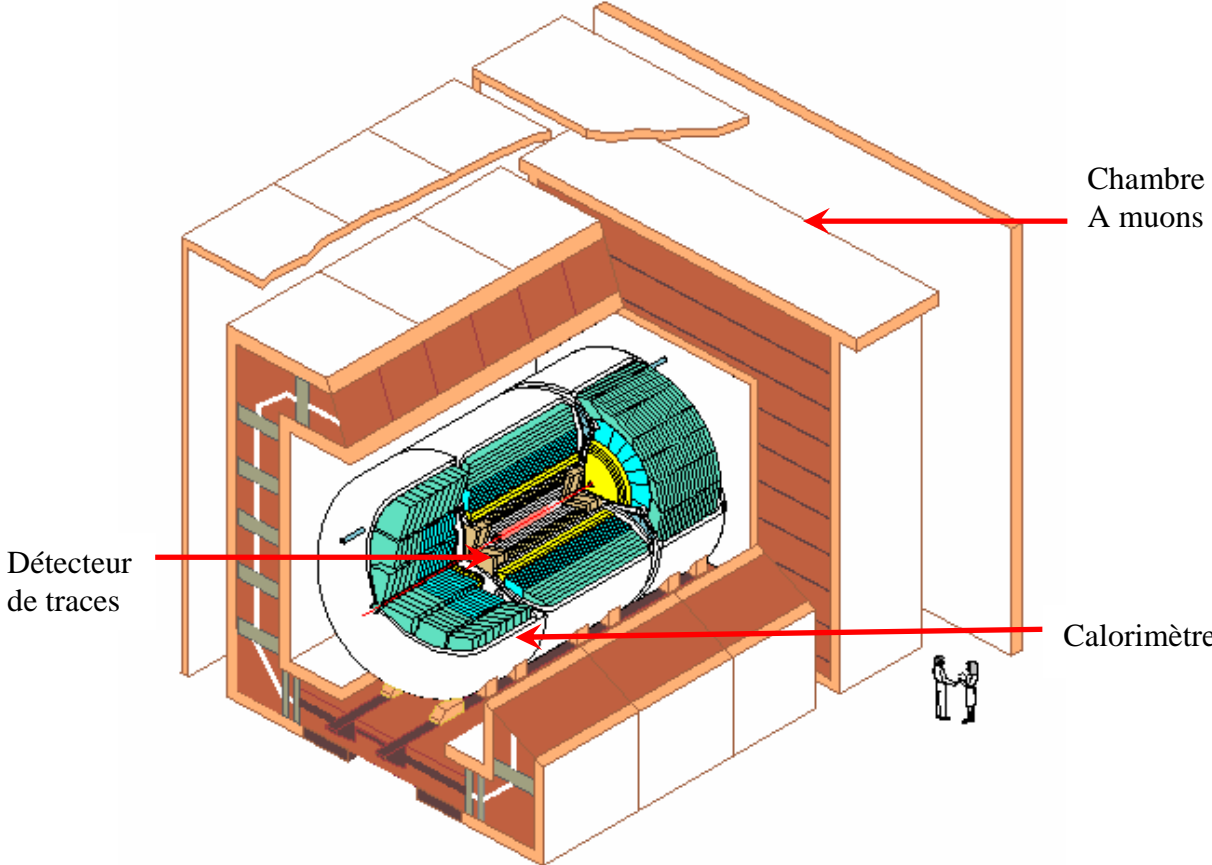
b) Le détecteur D0



Le détecteur D0 est composé de trois sous-systèmes principaux. Autour du point de collision se trouvent successivement un détecteur de traces, un calorimètre à uranium et argon liquide et un spectromètre à muons.

Comme cela a été dit, le Tevatron a été amélioré afin d'augmenter le nombre de collision par seconde et une autre amélioration dans ce sens est encore prévue.

L'électronique d'acquisition du détecteur doit pouvoir faire face à l'afflux de données inhérent à ces améliorations. J'ai travaillé dans ce domaine sur l'électronique du système de déclenchement du calorimètre. Le système de déclenchement du détecteur sert à déterminer si les données doivent être enregistrées en fonction de leur intérêt physique. Par exemple, si l'énergie déposée dans le calorimètre est importante, on peut supposer que des particules massives ont été produites lors d'une collision et que celle-ci présente un intérêt d'étude.



Le detecteur D0

III. La Grille de Calcul, un outil pour les physiciens

A. Introduction



Pour faire face à la demande de ressources informatiques nécessitées par les expériences de physique des particules qui doivent à la fois stocker et analyser leurs données, le DAPNIA est engagé dans un projet appelé le DataGrid ou grille de calcul. C'est sur ce projet que j'ai travaillé au cours des quatre premiers mois de mon apprentissage au CEA.

Ce nouveau type de calcul informatique a émergé au cours de ces dernières années. Cette technologie se base sur le calcul distribué : le principe est de répartir un ensemble de tâches sur plusieurs machines géographiquement distantes reliées par un réseau, comme par exemple internet. Des projets de calcul à échelle continentale ont ainsi pu voir le jour grâce à cette technologie, notamment le projet European DataGrid (EDG)

1. Le DataGrid

Le calcul distribué, ou Grid Computing, permet de combiner la puissance de calcul et l'espace de stockage de machines de la grille, et permet de ce fait de fournir les ressources nécessaires et suffisantes au traitement de grandes quantités de données.

Une grille de calcul est un type de système parallèle distribué qui permet le partage, la sélection et l'attribution de ressources distribuées à travers de multiples domaines administratifs, suivant leur disponibilité, leur possibilité, leur coût et les besoins de l'utilisateur.



2. Les acteurs du projet

Le principal instigateur de ce projet est le CERN (Laboratoire Européen pour la Physique des Particules). Cinq autres principaux partenaires se sont associés à ce dernier :

CNRS (France) Centre National de Recherche Scientifique

ESRIN (Italie) Centre de l'Agence Spatiale Européenne

INFN (Italie) Institut National de Physique Nucléaire

NIKHEF (Hollande) Institut National de Physique Nucléaire et des Hautes Energies

PPARC (Angleterre) Conseil de Recherche en Physique des Particules et Astronomie
Une quinzaine d'autres institutions se sont jointes au projet, dont le CEA.

3. Les utilisations du DataGrid

Dans le cadre du projet European DataGrid, trois secteurs scientifiques profitent de cette technologie qui s'avère être un outil majeur dans la recherche scientifique.

La physique des hautes énergies : Le CERN est actuellement en train de développer le LHC (Large Hadron Collider), le plus grand accélérateur de particules jamais conçu. Le LHC fournira une multitude de données suivant un débit très élevé. DataGrid offre une solution pour le stockage et le traitement des données résultantes. C'est dans ce cadre que j'ai travaillé sur ce projet.

Biologie et applications médicales : Le stockage et l'exploitation de données issues du décodage de génomes ne cessent de monopoliser chaque jour un peu plus les ressources informatiques des laboratoires. Les images médicales transitent chaque jour entre les différentes institutions (hôpitaux, centres de radiologie...), il y a une augmentation des besoins pour accéder aux données et les traiter. L'application de DataGrid à la biologie fournira la plat-forme idéale pour des algorithmes et facilitera ainsi l'échange de données médicales.

L'observation terrestre : Les missions de l'Agence Spatiale Européenne nécessitent la récupération, du ciel vers la Terre, de centaines de giga-octets d'images par jour. Des infrastructures terrestres ont été mises en place pour contenir la masse de données produites par les instruments des satellites. L'analyse des données relatives à l'ozone atmosphérique constitue l'une des applications de DataGrid.

B. Administration de système

A mon arrivée au DAPNIA, j'ai commencé par travailler sur le DataGrid.

La plate-forme que j'ai administrée était constituée de 7 ordinateurs sous Linux : 2 PC maîtres dénommés Gate01 et Gate02 ainsi que 5 PC esclaves nommés de node01 à node05. Les 7 machines étaient sur un réseau indépendant de celui du CEA, installé au sous-sol. L'administration de la plate-forme se faisait donc à distance grâce à l'utilitaire non graphique Putty.

Dans un premier temps, afin de me familiariser avec la programmation, j'ai dû réaliser un petit utilitaire en Perl (langage de programmation) permettant de compter les mots et les lignes dans un fichier texte. Ce n'était qu'un petit exercice afin d'apprendre à programmer, puisqu'une fonction existe déjà sous Linux : WC (Word Count). Ce fut l'unique programme que j'ai conçu en Perl, car ensuite j'ai travaillé sur la sécurisation de la plate-forme Datagrid.

En effet, le fait de répartir sur différentes machines toutes les données acquises crée une faille dans la sécurité des données : il faut que toutes les machines composant la grille puissent garantir l'intégrité des informations qu'elles stockent et se protéger des intrusions.

Ainsi j'ai eu à faire l'installation et la configuration de Tripwire, un firewall (pare-feu), outil permettant de contrôler les entrées / sorties des différentes machines, mais aussi de diffuser la liste des modifications apportées dans le mois. Pour la prise en main de cet outil, j'ai fait l'installation sur Gate01. Pour la configuration du pare-feu, j'ai codé les règles de gestion principalement la liste des fichiers ne devant pas être modifiés.

Une fois l'installation terminée sur ce PC, pour simplifier les tâches d'administration du pare-feu j'ai planifié sous l'horloge de Linux (le Cron) un script en shell permettant d'envoyer par mail le rapport de Tripwire chaque semaine à mon responsable ainsi qu'à moi. Pour les autres machines de la plate-forme (Gate02 et les 5 nodes), j'ai écrit un script en shell automatisant l'installation de Tripwire.

Chaque semaine, mon responsable et moi recevions par mail sous notre compte Windows, sous forme de fichier texte, les sept rapports des machines indiquant les modifications, le niveau de sécurité et d'autres informations concernant le microprocesseur (se reporter à l'annexe 1). Je n'avais pas à analyser le contenu de ces rapports mais je devais m'assurer qu'ils étaient bien générés et expédiés.

Afin de compléter la sécurisation de la plate-forme, je devais régulièrement exécuter sur le kernel (noyau de Linux) un logiciel écrit par mon responsable (nommé Filtlog) permettant de détecter toutes les tentatives d'intrusions du réseau et les rapporter dans un fichier texte. Je devais ensuite transmettre ce fichier à mon responsable pour analyse.

Datagrid a été une expérience enrichissante, puisque cela m'a permis de me familiariser avec le monde Linux.

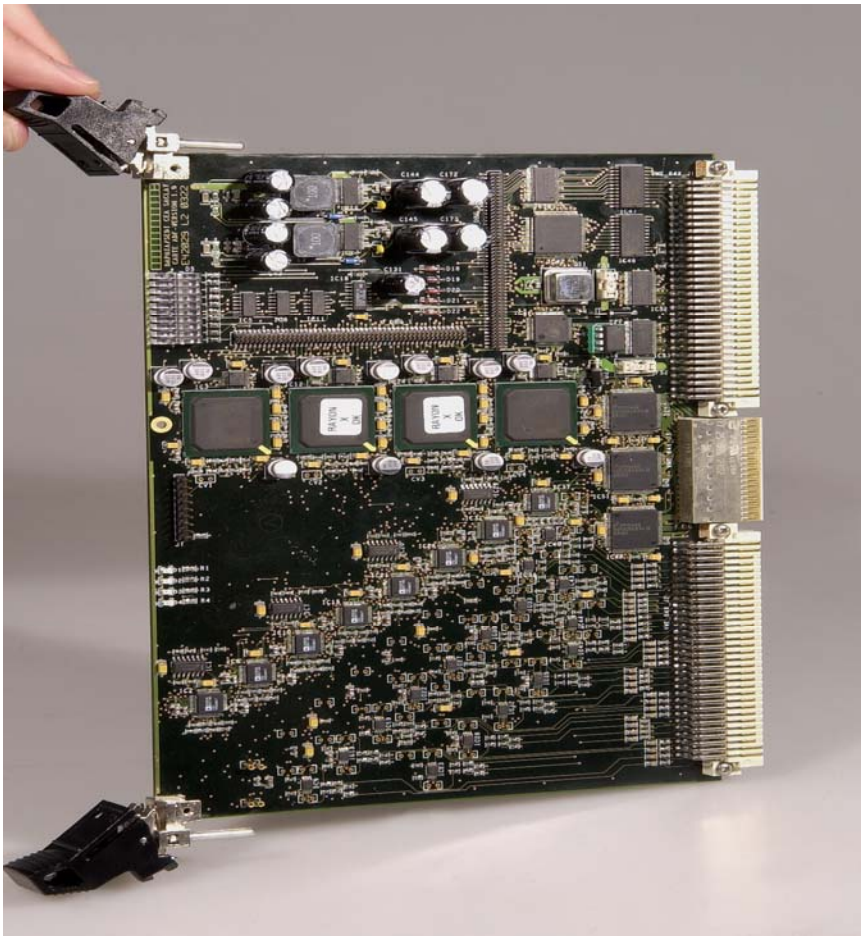
IV. Travail sur l'expérience D0

A. Mise en place d'un protocole de communication par interface VME

L'un des rôles du DAPNIA est donc de réaliser un système de déclenchement pour l'acquisition des données du calorimètre du détecteur D0 en vue d'une amélioration du Tevatron. C'est dans ce cadre de travail que Denis Calvet, mon responsable, a réalisé une carte électronique de conversion Analogique/Numérique contenant divers filtres : la carte ADF (Analog Digit Filter).

Cette carte sera chargée de convertir les signaux analogiques reçus du calorimètre en signaux numériques. Ces derniers seront ensuite envoyés à une carte « trigger » (ou déclenchement), carte permettant le tri et l'envoi d'informations concernant l'énergie déposée dans le calorimètre. La connexion entre la carte ADF et la carte « trigger » se fera numériquement, avec un débit de 2 Gbits/s.

Au final, 80 cartes ADF effectueront simultanément des conversions numériques des signaux provenant du calorimètre. Il y aura 4 châssis VME contenant 20 cartes chacun.

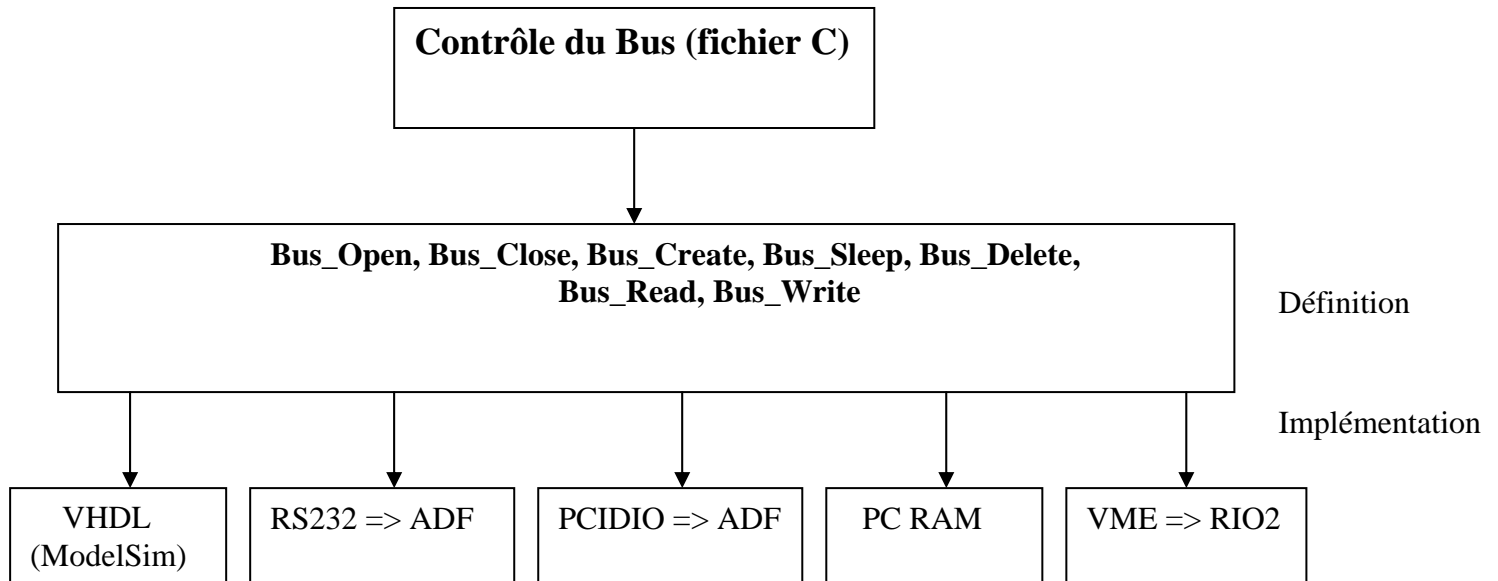


Carte électronique ADF

1. Programmation du fichier principal Mpp

Dans chaque châssis VME (Versa Module Eurocard), une carte PCI (pour PC interface) a été ajoutée afin de pouvoir contrôler d'un ordinateur les différentes actions sur le châssis VME.

Denis Calvet a réalisé différentes bibliothèques en langage C, permettant une communication avec le châssis VME, via différents ports (RS232 carte PCIDIO, PCI to VME)



Durant mon apprentissage, j'ai programmé un fichier principal (programme Main) c'est à dire un exécutable permettant différentes actions sur le châssis VME. Ce programme s'appelle Mpp : My peek poke (se référer à l'annexe 2) et sert à lire ou à écrire sur les adresses du châssis. 2 types de lecture et d'écriture sont disponibles,

ReadByte : permet de lire sur un octet : -r

ReadShort : permet de lire sur 2 octets : -R

WriteByte : permet de lire sur un octet : -w

WriteShort : permet de lire sur 2 octets : -W

La complexité de ce programme réside dans le fait que c'est la bibliothèque passée en paramètre qui détermine les différentes actions de lecture ou écriture sur le châssis, le programme Main devant être unique.

Le programme fonctionne sous DOS et utilise la syntaxe suivante (voir annexe 3) :

```
Mpp -v 10 -r 0x2b0000
```



Dans cet exemple, le programme va lire la donnée sur un octet à l'adresse 0x2b0000. Pour l'écriture, spécifiée par l'attribut -w au lieu de -r, il faut rajouter la donnée à écrire en paramètre de dernière position .

-v signifie verbose, suivi du niveau de détail de compte rendu d'exécution désiré. Plus la taille est grande, plus le programme affiche des informations.

2. Réalisation de bibliothèques sous Lynx OS

Le projet de protocole de communication étant bien avancé, les premières cartes ont été expédiées à Fermilab et la PCI du prototype n'a pas été restituée. Pour remplacer la carte PCI permettant la communication entre le PC et le châssis VME, nous avons utilisé une carte Rio2 immédiatement disponible, équipée d'un processeur tournant sous Lynx OS. Il fallait donc adapter les programmes du prototype qui avaient été écrits pour la carte PCI.



Carte Rio2

La deuxième partie de mon travail consistait à créer toutes les bibliothèques concernant la carte ADF sous Lynx OS à partir des fichiers C existants programmés par mon responsable, puis écrire la bibliothèque d'interfaçage permettant le contrôle du châssis par la carte Rio2 (BUS_VMERIO2).

Sous Windows, j'avais utilisé le logiciel Visual Studio pour écrire My Peek Poke. Ne disposant pas d'un tel outil convivial sous Lynx OS, j'ai dû utiliser l'utilitaire Makefile (similaire à celui disponible sous Linux), permettant de créer des fichiers de type bibliothèques ou bien des fichiers exécutables en passant par la commande make (voir annexe 4).

J'ai compilé et réalisé les différentes bibliothèques sous Lynx OS à partir des fichiers existants.

Malgré le fait que je disposais de multiples exemples de programmes en C, il m'était difficile d'utiliser ce langage au cours des premiers mois de mon apprentissage. Puis le fait d'avoir suivi les cours à l'IUT sur ce langage m'a aidé à mieux comprendre les programmes qu'il fallait modifier. Cependant cela n'a pas été suffisant pour que je puisse écrire moi-même le nouveau programme que je devais concevoir, BUS_VMERIO2.

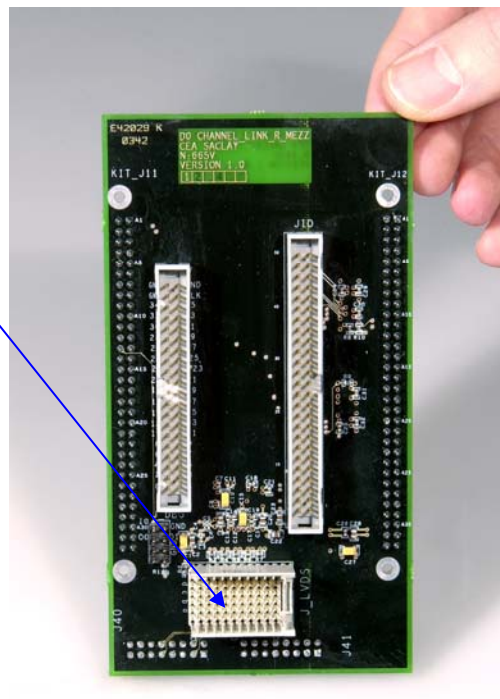
Finalement, après concertation avec mon responsable, j'ai continué à travailler pour le projet D0, mais en programmant en langage VHDL (Very High Speed Integrated Circuit Hardware Description Language), afin de concevoir un fréquencemètre.

B. Elaboration d'un fréquencemètre

1. Contexte

Comme cela a été expliqué, la connexion entre la carte ADF et la carte Trigger s'effectue à 2 Gbits/s. Une carte permettant de tester les liens à haut débit a été conçue et développée avant mon arrivée. Mon objectif a été d'ajouter à cette carte de test de lien à haut débit une fonction supplémentaire visant à calculer le débit sur le lien testé par la mesure de la fréquence de réception des paquets.

Récepteur
à 2 GBits/s



Carte de test de liens à haut débit

2. Description du projet

Afin de tester les sorties à haut débit sur des cartes électroniques en développement, un petit appareil de test a été conçu et réalisé. Ce testeur de liens à haut débit est constitué d'une carte mezzanine comportant un récepteur 2 Gbit/s connectée à un kit d'évaluation FPGA (Field Programmable Gate Arrays) Xilinx Virtex II du commerce (circuit logique programmable à un million de portes). Le contrôle de l'ensemble se fait au moyen d'un PC équipé d'une carte d'interface PCI / entrées sorties parallèles.

Ce testeur permet de compter le nombre de paquets reçus et d'enregistrer leur contenu (2048 mots sur 36 bits), de vérifier la parité pour chaque paquet et de compter les erreurs détectées.

Le testeur peut également générer localement une séquence pseudo aléatoire préétablie au niveau de l'émetteur. Cette séquence est envoyée sur la carte mezzanine et les paquets correspondants renvoyés par cette carte sont comparés à la séquence générée initialement par le testeur afin de détecter et compter les erreurs de transmission. Une fonctionnalité

manquante est la mesure de la fréquence de réception des paquets qui permet de déduire le débit effectif du lien testé. Le but du projet est de concevoir, développer et intégrer au testeur actuel un module de mesure de la fréquence des paquets reçus.

3. Cahier des charges

Le module à concevoir et à réaliser doit être décrit en langage VHDL. Il doit être synthétisable et, indépendant du composant cible. Il doit pouvoir être implanté dans un FPGA de type Xilinx Virtex II et devra utiliser le minimum de ressources.

Le débit nominal sur le type de liaison étudiée est 2.18016 Gbit/s soit 36 bits fois 60.56 Mhz après le circuit de désérialisation. Le signal d'horloge de 60.56 Mhz est reconstruit par le récepteur et est utilisé pour cadencer le bus de 36 bits reliant le récepteur série au FPGA du testeur. C'est la fréquence de ce signal d'horloge que l'on souhaite mesurer. La gamme de mesure du fréquencemètre doit être de ~30Mhz à ~65Mhz afin de pouvoir tester la liaison dans une large gamme.

Le module utilisera comme base de temps l'un des deux oscillateurs disponibles sur le kit d'évaluation (100Mhz ou 24Mhz).

La fréquence de réception est codée sur 16 bits et représente directement la grandeur d'intérêt sans conversion d'unité (60560 équivaut à 60560 kHz). Si la fréquence de réception dépasse la capacité du module, la valeur maximale pouvant être représentée doit être retournée.

La mesure de la fréquence sera rafraîchie à 1 Hz au minimum.

Le module devra placer ses mesures sur un bus de données 16 bits, mais les lectures seront faites par le logiciel du testeur en 2 transactions de 8 bits.

Le module comportera en plus des signaux d'interface nécessaires, un signal d'initialisation asynchrone (RESET). Tant que ce signal est actif, la fréquence retournée doit être nulle.

4. Travail demandé

- Concevoir l'architecture du module fréquencemètre.
- Coder en VHDL le module fréquencemètre suivant les spécifications énoncées.
- Coder en VHDL le banc de test de ce module fréquencemètre suivant les spécifications énoncées.
- Coder en VHDL le banc de test de ce module en simulant l'horloge de réception par une source modulée en fréquence.
- Valider le module fréquencemètre et son banc de test par simulation comportementale.
- Synthétiser et implémenter le module fréquencemètre dans un FPGA de type Xilinx VIRTEX II, et valider par simulation post-routage.
- Tester le composant sur le FPGA physique du kit d'évaluation Virtex II en utilisant un générateur de signaux carrés de fréquence variable et un analyseur logique.
- Adapter le plan d'adressage des registres du testeur de lien à haut débit pour y inclure le fréquencemètre.
- Intégrer le module développé dans le testeur de lien à haut débit actuel.
- Synthétiser et implanter le testeur de liens à haut débit ainsi modifié.
- Vérifier le fonctionnement de la fonction fréquencemètre du testeur complet au travers de son logiciel de contrôle.

5. Outils disponibles

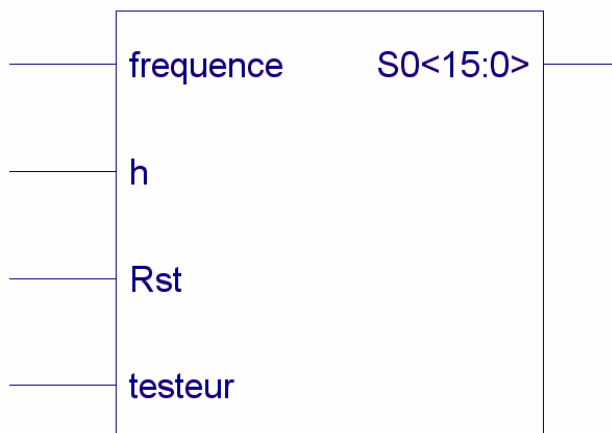
Je disposais d'un PC sous Windows, d'un kit d'évaluation Virtex II, d'une carte mezzanine récepteur 2 Gbits/s et d'un analyseur logique.

J'ai également utilisé un simulateur VHDL ModelSim, d'un outil de synthèse placement/routage Xilinx, du code source VHDL du testeur de lien à haut débit et du code source C du logiciel de contrôle ;

Lors de la conception du code VHDL utilisé par le fréquencemètre, je n'avais pas pris en compte la contrainte suivante du cahier des charges : « Le module devra placer ses mesures sur un bus de données 16 bits, mais les lectures seront faites par le logiciel du testeur en 2 transactions de 8 bits ». Cela signifie que le bus doit d'abord lire la fréquence sur les bits 0 à 7, puis de 8 à 15. J'ai donc dû légèrement modifier mon code et rajouter une entrée codée sur un bit, permettant de figer la valeur de la variable « fréquence » rendue en sortie du fréquencemètre tout au long des 2 phases d'acquisition. J'ai appelé cette entrée supplémentaire « testeur ».

6. Etude du fréquencemètre

Le fréquencemètre est donc composé de 4 entrées codées sur un bit chacune et d'une sortie codée sur 16 bits.



- **h** : signal d'horloge, j'ai choisi l'oscillateur local à 100 MHz (j'expliquerai après la raison)
- **Fréquence** : signal d'entrée à fréquence variable (c'est la fréquence de ce signal que l'on doit mesurer)
- **Rst** : signal permettant la mise à zéro de la sortie, ce signal est asynchrone avec l'horloge
- **testeur** : signal permettant de bloquer la sortie (afin de permettre la lecture de la fréquence par le bus)
- **S0** : signal de sortie codé sur 16 bits retournant la valeur de la fréquence (de 0 à 65534, si la fréquence est supérieure à 65534, le fréquencemètre retournera 65535)

La fréquence étant l'inverse de la période, elle représente le nombre d'évènements produits en 1 seconde. Comme la fréquence retournée par S0 est en kHz et non en Hz, il n'est pas nécessaire de mesurer sur 1 seconde, mais plutôt sur 1 ms.

L'oscillateur local est de 100 MHz (c'est à dire que 10 ns servent de base de temps pour 1 ms). Pour cela, il suffit de faire un compteur de 0 à 100000 afin d'avoir 1 ms.

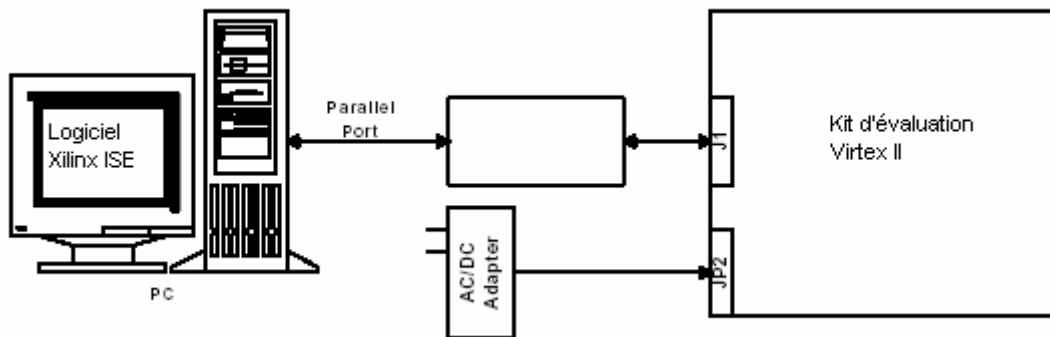
Dans le même temps pendant que le compteur s'incrémente jusqu'à 1 ms un autre compteur va s'incrémenter sur chaque front montant du signal fréquence. Dès que le premier compteur arrive à 1 ms, le second renvoie sa valeur à S0.

Une fois que le programme a été écrit et compilé, il faut le tester avec une simulation. Pour cela, il faut créer un banc de test VHDL, un fichier où l'on assigne les entrées sorties, les

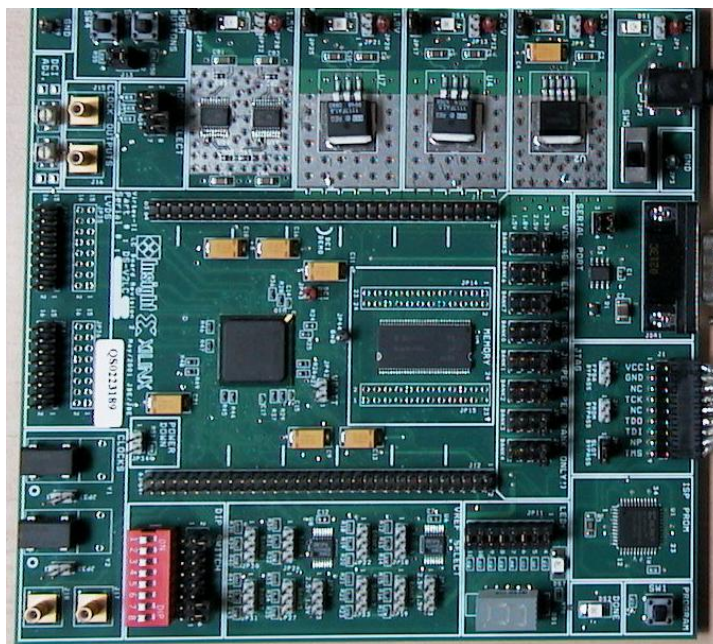
temps de simulation, et les différents évènements en entrée dans le temps. En plus de ce fichier, pour m'assurer que le fréquencemètre marche pour une large plage de fréquence, j'ai conçu un générateur en VHDL, dont la fréquence s'incrémente de 100 kHz toutes les 2 ms. Il part avec une fréquence de 1000 kHz (1 MHz) et finit avec une fréquence de 80000 kHz (80 MHz). L'inconvénient est que le temps de simulation est relativement long (1,4 secondes à simuler prend environ 10 minutes).

Une fois que le programme et les simulations ont été validés par ModelSim, il faut importer le programme principal sous le logiciel Xilinx ISE. Ce logiciel permet de synthétiser le fichier VHDL afin de voir la mémoire qu'il occupe, de simuler sous ModelSim le fichier qui sera implémenté dans le FPGA, en respectant les temps de réaction (lorsqu'un évènement se produit, la réponse n'est pas instantanée). De plus ce logiciel permet d'assigner les pins où se situent les entrées/sorties du FPGA.

A la fin seulement, on implémente le fichier final sur la carte d'évaluation Virtex II



Implémentation dans le FPGA



Kit d'évaluation Xilinx Virtex II

Une fois le fichier implémenté dans le FPGA, j'ai réalisé un diviseur de fréquence (divisant par 2^n) connecté à l'horloge de 100 MHz. Cela permet de vérifier que le fréquencemètre fonctionne pour différentes fréquences en utilisation réelle. Un véritable fréquencemètre est connecté devant le signal fréquence, afin de vérifier que la fréquence retournée en sortie soit juste. Pour mesurer les 16 bits de sortie, un analyseur logique relève les mesures et renvoie les données sur PC, afin que ces dernières puissent être traitées. Sur l'ordinateur, le logiciel utilisé s'appelle Agilent LogicWave ; il est fourni avec l'analyseur logique.



PC permettant le traitement des données reçues de l'analyseur logique.

Carte d'évaluation Virtex II, le fréquencemètre, et le diviseur de fréquence sont implémentés dans le FPGA.

Test physique du fréquencemètre

Analyseur logique, connecté au PC, relevant les 16 bits de la sortie du fréquencemètre sur la carte Virtex II.

Fréquencemètre, permettant de vérifier que la fréquence retournée soit juste.

Si le moindre problème a été détecté, mauvaise fréquence retournée, sortie ne réagissant pas aux commandes d'entrées, alors il faut repartir du début, et trouver l'erreur dans le code VHDL.

J'ai eu des problèmes au début avec le fréquencemètre, ce dernier retournait une valeur fausse lorsque la fréquence était inférieure à 10MHz. Finalement, après avoir apporté des modifications dans le code VHDL, le fréquencemètre marchait bien.

7. Intégration du fréquencemètre dans le testeur de liens à haut débit

Une fois le test du fréquencemètre validé, il faut l'intégrer dans le testeur à haut débit. Pour cela, il faut reprendre tout le projet sous ModelSim, c'est à dire tous les fichiers VHDL concernant le testeur (Chalink testeur), rajouter le fichier VHDL concernant le fréquencemètre, et créer le composant dans le fichier principal.

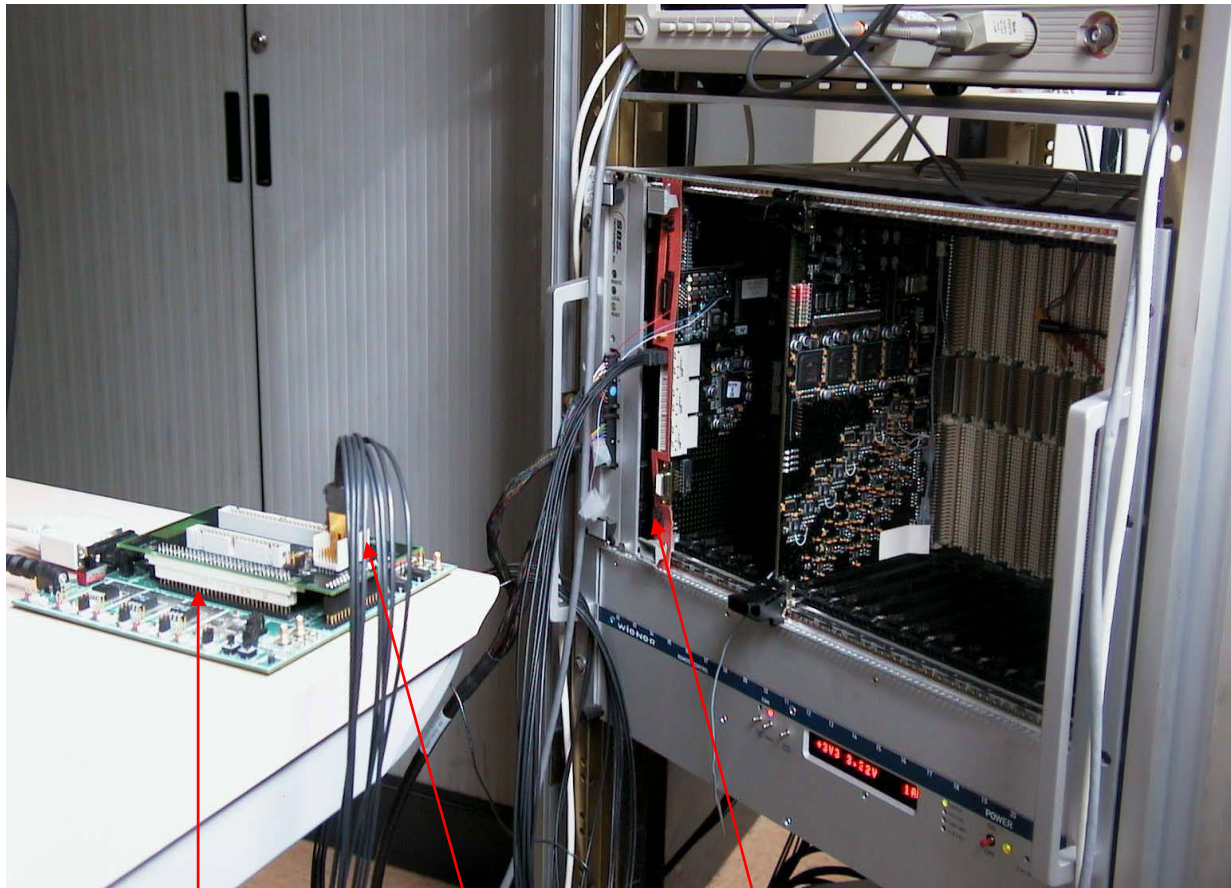
Name	Status	Type	Orde Δ	Modifi
utility_pkg.vhd	✓	VHDL	0	03/31.
cha_reg.vhd	✓	VHDL	1	08/02.
Sr_pl.vhd	✓	VHDL	2	06/20.
sr_pl_po.vhd	✓	VHDL	3	08/02.
trigger.vhd	✓	VHDL	4	08/03.
trigger_test.vhd	✓	VHDL	5	08/03.
lfsr.vhd	✓	VHDL	6	08/03.
lfsr_test.vhd	✓	VHDL	7	08/03.
cnt_sat.vhd	✓	VHDL	8	08/02.
gen_ctrl.vhd	✓	VHDL	9	08/03.
freq_meter.vhd	✓	VHDL	10	08/03.
ChalinkR.vhd	✓	VHDL	11	08/02.
ChalinkR_FPGA.vhd	✓	VHDL	12	04/29.
ChalinkR_FPGA_test.vhd	✓	VHDL	13	08/03.
constant_package.vhd	✓	VHDL	14	07/30.
shift_register.vhd	✓	VHDL	15	08/02.
Signal_Is.vhd	✓	VHDL	16	08/02.
Etude_chaine.vhd	✓	VHDL	17	08/02.
bloc_bus_control.vhd	✓	VHDL	18	08/02.
Serialisation.vhd	✓	VHDL	19	08/02.
Bloc_serial_deserial.vhd	✓	VHDL	20	08/02.
ChalinkR_RS232.vhd	✓	VHDL	21	08/02.
ChalinkR_RS232_FPGA.vhd	✓	VHDL	22	08/02.
acia.vhd	✓	VHDL	23	08/03.
acia_test.vhd	✓	VHDL	24	08/03.
ChalinkR_RS232_FPGA_test.vhd	✓	VHDL	25	07/28.

Fichier VHDL du fréquencemètre

Fichier principal à modifier afin d'y inclure le fréquencemètre

Logiciel ModelSim comprenant tous les fichiers VHDL du testeur de liens

La seule difficulté de l'intégration, concerne le blocage de la sortie avec l'entrée testeur lorsqu'on commence la lecture du bus.



Expérience finale

Carte de test connecté au kit Xilinx Virtex II (FPGA), relié au PC par liaison RS232

Connexion à 2 Gbits/s

Carte intégrée au châssis vme envoyant des données binaires aléatoires à la carte de test

Une fois le fréquencemètre implémenté, lorsque l'on tape « frequency » lors de l'utilisation de la carte mezzanine, la fréquence est retournée. Lors des tests, elle est d'environ 60.56 MHz, et comme la carte testeur reçoit des bits par paquet de 36, la connexion est de 36 fois 60.56 MHz, donc d'environ 2 Gbits/s ce qui prouve la réalité du lien haut débit.

V. Bilan de ma formation en apprentissage

Cette expérience de deux années de formation par la voie de l'apprentissage au CEA de Saclay a été pour moi extrêmement enrichissante. Tout d'abord, elle m'a permis de mieux percevoir le milieu de l'entreprise, tant du point de vue professionnel que du point de vue relationnel.

Professionnellement, j'ai appris à mettre en pratique ce que j'apprenais d'une manière théorique à l'IUT GEII de Cachan (comme par exemple la programmation en C ou en VHDL). Ceci m'a paru particulièrement important, car les études secondaires ne nous confrontent pratiquement pas au contexte de l'entreprise, mis à part une semaine de stage en classe de troisième.

Au début de mon stage au CEA de Saclay, avec le seul diplôme du baccalauréat (même si la série S et l'option techniques de l'ingénieur que j'avais choisie devaient m'y préparer), je me suis senti un peu dérouté par le travail qui m'était demandé, dont je percevais mal les objectifs et les implications. Il m'a fallu un certain temps pour mieux saisir ce que l'on attendait de moi, et je me suis alors senti plus à l'aise. En rédigeant ce bilan, je perçois combien cet apprentissage a représenté pour moi une chance de me former à un domaine qui m'était jusqu'alors totalement inconnu.

Le travail que j'ai effectué sur DataGrid puis sur l'expérience D0 m'a permis de m'impliquer dans un projet collectif.

Je n'ai pas eu l'opportunité de me rendre à Chicago pour voir le Tévatron, mais j'ai eu la chance de participer à une visite de 2 jours au CERN à Genève, organisée pour les apprentis du CEA (une dizaine). J'ai visité le chantier de construction du LHC (Large Hadron Collider). J'ai mieux perçu à cette occasion combien la physique des particules représentait un enjeu de recherche internationale intéressant. Je suis satisfait d'avoir pu collaborer, même d'une manière infime, à cette grande entreprise.

D'un point de vue relationnel, j'ai rencontré au CEA des professionnels passionnés par leur travail, qui ont su m'intéresser à la physique des particules. J'ai aussi pu constater combien le travail d'équipe sur un projet déterminé est stimulant, mais qu'il comporte aussi ses difficultés, comme par exemple le fait de retravailler sur des programmes que d'autres personnes ont déjà réalisés.

VI. Annexes

A. Annexe 1 : Rapport généré par Tripwire

Parsing policy file: /etc/tripwire/tw.pol

*** Processing Unix File System ***

Performing integrity check...

Wrote report file: /var/lib/tripwire/report/gate01.datagrid.cea.fr-20030727-042237.twr

Tripwire(R) 2.3.0 Integrity Check Report

Report generated by: root

Report created on: Sun Jul 27 04:22:37 2003

Database last updated on: Tue Apr 08 09:54:10 2003

=====
 =====
 Report Summary:
 =====
 =====

Host name: gate01.datagrid.cea.fr
 Host IP address: 192.54.208.1
 Host ID: None
 Policy file used: /etc/tripwire/tw.pol
 Configuration file used: /etc/tripwire/tw.cfg
 Database file used: /var/lib/tripwire/gate01.datagrid.cea.fr.twd
 Command line used: /usr/sbin/tripwire -m c

=====
 =====
 Rule Summary:
 =====
 =====

 Section: Unix File System

Rule Name	Severity Level	Added	Removed	Modified
Invariant Directories	66	0	0	0
Temporary directories	33	0	0	0
Tripwire Data Files	100	0	0	0
Critical devices	100	0	0	0
User binaries	66	0	0	0

Tripwire Binaries	100	0	0	0	
* Critical configuration files	100	0	0	2	
* Libraries	66	0	0	1	
Shell Binaries	100	0	0	0	
File System and Disk Administraton Programs	100	0	0	0	
Kernel Administration Programs	100	0	0	0	
Networking Programs	100	0	0	0	
System Administration Programs	100	0	0	0	
Hardware and Device Control Programs	100	0	0		
System Information Programs	100	0	0	0	
Application Information Programs	100	0	0		
Shell Releated Programs	100	0	0	0	
Critical Utility Sym-Links	100	0	0	0	
* Critical system boot files	100	0	0	1	
* System boot changes	100	0	1	58	
OS executables and libraries	100	0	0	0	
Security Control	100	0	0	0	
Login Scripts	100	0	0	0	
Operating System Utilities	100	0	0	0	
* Root config files	100	0	0	2	

Total objects scanned: 21274

Total violations found: 65

```
=====
=====
Object Summary:
=====
=====
```

```
-----
# Section: Unix File System
-----
```

```
-----
Rule Name: Libraries (/usr/lib)
Severity Level: 66
-----
```

Modified:

"/usr/lib/perl5/man/whatis"

```
-----
Rule Name: Critical configuration files (/var/lib/nfs/rmtab) Severity Level: 100
-----
```

Modified:

"/var/lib/nfs/rmtab"

Rule Name: System boot changes (/var/log)

Severity Level: 100

Modified:

"/var/log/boot.log"
"/var/log/boot.log.1"
"/var/log/boot.log.2"
"/var/log/boot.log.3"
"/var/log/boot.log.4"
"/var/log/cron"
"/var/log/cron.1"
"/var/log/cron.2"
"/var/log/cron.3"
"/var/log/cron.4"
"/var/log/httpd/access_log"
"/var/log/httpd/access_log.1"
"/var/log/httpd/access_log.2"
"/var/log/httpd/access_log.3"
"/var/log/httpd/access_log.4"
"/var/log/httpd/error_log"
"/var/log/httpd/error_log.1"
"/var/log/httpd/error_log.2"
"/var/log/httpd/error_log.3"
"/var/log/httpd/error_log.4"
"/var/log/maillog"
"/var/log/maillog.1"
"/var/log/maillog.2"
"/var/log/maillog.3"
"/var/log/maillog.4"
"/var/log/messages"
"/var/log/messages.1"
"/var/log/messages.2"
"/var/log/messages.3"
"/var/log/messages.4"
"/var/log/netconf.log"
"/var/log/netconf.log.1"
"/var/log/netconf.log.2"
"/var/log/netconf.log.3"
"/var/log/netconf.log.4"
"/var/log/secure"
"/var/log/secure.1"
"/var/log/secure.2"
"/var/log/secure.3"
"/var/log/secure.4"
"/var/log/spooler"
"/var/log/spooler.1"
"/var/log/spooler.2"
"/var/log/spooler.3"

"/var/log/spooler.4"
"/var/log/wtmp"
"/var/log/wtmp.1"
"/var/log/xferlog"
"/var/log/xferlog.1"
"/var/log/xferlog.2"
"/var/log/xferlog.3"
"/var/log/xferlog.4"

Rule Name: System boot changes (/var/lock/subsys)
Severity Level: 100

Modified:
"/var/lock/subsys/edg-crl-upgrade"
"/var/lock/subsys/edginfo-mds" "/var/lock/subsys/globus-gsi_wuftpd"
"/var/lock/subsys/globus-mds"
"/var/lock/subsys/sleppid"

Rule Name: System boot changes (/var/run)
Severity Level: 100

Removed:
"/var/run/globus-ftp.pids-all"

Rule Name: Critical configuration files (/etc/sysconfig) Severity Level: 100

Modified:
"/etc/sysconfig/hwconf"

Rule Name: Critical system boot files (/boot)
Severity Level: 100

Modified:
"/boot"

Rule Name: System boot changes (/dev/log)
Severity Level: 100

Modified:
"/dev/log"

Rule Name: Root config files (/root)
Severity Level: 100

Modified:
"/root"
"/root/.neditdb"

=====
=====
Error Report:
=====
=====

No Errors

*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc.
Tripwire is a registered trademark of Tripwire, Inc.
This software comes with ABSOLUTELY NO WARRANTY; for details use --version.
This is free software which may be redistributed or modified only under certain conditions;
see COPYING for details. All rights reserved. Integrity check complete.

B. Annexe 2 : Programme de commande du Bus VME

```

/*****
****

```

D0 Run

File: mpp.c

Description: Program used to control a VME INTERFACE

Author: B. Trincaz, btrincaz@dapnia.cea.fr

History:

Created June 2003

Note: Use the help with "mpp -h" if you want to use it correctly.

```

****/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "bus_if.h"
#include "getopt.h"

#define MAX_REASON_SIZE    2048
#define REPORT_MAX_SIZE    2048
#define MAX_INDENTATION    64
#define MAX_ADD_ARGUMENTS  128
#define DEFAULT_CMD_SERVER_PORT 4803

```

```

extern char *optarg;
extern int optint;
char *data;

```

```

char* what_is_data(unsigned int adr, unsigned short data)
{
    return (&"??");
}

```

```

char* what_is_address(unsigned int adr)
{

```

```

        return (&"??");
    }

int main( int argc, char **argv)

{
    short data1 = 12;
    short *data_short = &data1;
    unsigned char data3;
    unsigned short data4;
    int rep_max;
    char data2 = 'a';
    char *data_byte = &data2;
    char report[REPORT_MAX_SIZE];
    char lspace[MAX_INDENTATION];
    int choix = 0;
    int i;
    int opt;
    int argix = 0;
    int adr;
    int verbose = 0;

    file_string_socket trace;

    BusStruct *bus;
    BusArg bus_arg;
    BusArg_Clear(&bus_arg);
    bus = (BusStruct *) 0;
    rep_max = REPORT_MAX_SIZE;
    trace.p = stdout;
    trace.t = FILE_POINTER;

    sprintf(report, "");
    sprintf(lspace, "");

    bus_arg.bus_type = BUS_TYPE_PCIDIO96;
    bus_arg.DeviceNumber = 1;

    /******
    USE GETOPT TO ANALYSE ARGS
    *****/

    while (( opt = getopt(argc,argv,"v:R:W:w:r:h")) != -1)
    {

```

```

        switch (opt)
        {
            case 'r':
                argix+=2;
                if (argv[argix] == NULL)
                    { printf("Pas d'adresse consultez l'aide\n");
                    exit (0);
                    }
                else
                    {
                        if (sscanf(argv[argix], "0x%x", &adr) == 1)
                            printf("");
                        else
                            adr = atoi(argv[argix]);

                            choix = 1;
                    }
                break;

            case 'v':
                argix+=2;
                verbose = 1;
                if (argv[argix] == NULL)
                    {printf("Vous devez entrez un parametre derriere le
                    verbose\n");
                    exit (0);
                    }

                else
                    verbose = atoi(argv[argix]);

                break;

            case 'R':
                argix+=2;
                if (argv[argix] == NULL)
                    { printf("Pas d'adresse consultez l'aide\n");
                    exit (0);
                    }
                else
                    {
                        if (sscanf(argv[argix], "0x%x", &adr))
                            printf("");
                    }
        }

```

```
        else
            adr = atoi(argv[argix]);

            choix = 3;
        }
    break;

    case 'w':
argix+=2;
        if (argv[argix] == NULL)
            { printf("Pas d'adresse consultez l'aide\n");
              exit (0);
            }

            else
            {
                if (sscanf(argv[argix], "0x%x", &adr))
                    printf("");
            }
        else
            adr = atoi(argv[argix]);

            argix++;

            if (argv[argix] == NULL)
                {
                    printf("Il manque une adresse consultez l'aide\n");
                    exit (0);
                }

            else
                {
                    if (sscanf(argv[argix], "0x%x", &data3))
                        printf("");
                }

            else
                data3 = atoi(argv[argix]);

            choix = 2;
        }
    break;

    case 'W':
argix+=2;
        if (argv[argix] == NULL)
```

```

        {
            printf("Pas d'adresse consultez l'aide\n");
        exit (0);
        }

        else
        {
            if (sscanf(argv[argix], "0x%x", &adr))
                printf("");
        }
    else
        adr = atoi(argv[argix]);

    argix++;

    if (argv[argix] == NULL)
    {
        printf("Il manque une adresse consultez l'aide\n");
        exit (0);
    }

    else
    {
        if (sscanf(argv[argix], "0x%x", &data4))
            printf("");
    }

    else
        data4 = atoi(argv[argix]);

    choix = 4;
    }
    break;

    case 'h':
        printf("VME Interface \n\n-h : for help");
        printf("\n-v <niveau> pour le verbose\n-r <adresse> pour lire l'adresse sur un byte \n-
R <adresse> si l'on veut lire sur un short\n-w <adresse> <data> pour ecrire sur l'adresse un
byte\n-W <adresse> <data> pour ecrire sur l'adresse un short\n\nSi vous voulez faire une
action d'ecriture, vous devez placer le verbose\nau debut de l'instruction\n\n\n");
        exit (0);
        break;

    default:
        printf("use 'mpp -h'\n");
        exit (0);
        break;
    }

```

```
    }

    printf("\n");
    i = Bus_Create(&bus, &bus_arg, report, REPORT_MAX_SIZE, verbose, &trace,
lspace);
    if(i != 0)
    {
        printf("Erreur de creation du bus: \n\n%s\n", report);
        exit (0);
    }

    i = Bus_Open(bus, &bus_arg, report, REPORT_MAX_SIZE, verbose, &trace,
lspace);
    if(i != 0)
    {
        printf("Erreur de l'ouverture du bus\n\n%s\n", report);
        exit (1);
    }

    printf("\n");

switch (choix)
{

case 1 :
    i = Bus_ReadByte(bus, adr, data_byte, report, REPORT_MAX_SIZE, verbose-1,
&trace, lspace);
    break;

case 2 :
    i = Bus_WriteByte(bus, adr, data3, report, REPORT_MAX_SIZE, verbose-1, &trace,
lspace);
    break;

case 3 :
    i = Bus_ReadShort(bus, adr, data_short, report, REPORT_MAX_SIZE, verbose-1,
&trace, lspace);
    break;

case 4 :
    i = Bus_WriteShort(bus, adr, data4, report, REPORT_MAX_SIZE, verbose-1, &trace,
lspace);
    break;

}

if (i != 0)
```

```
    printf("Erreur de lecture ecriture du bus\n\n%s\n", report);

    i = Bus_Close(bus, report, REPORT_MAX_SIZE, verbose, &trace, lspace);
    if (i != 0)
        printf("Erreur de fermeture du bus: \n\n%s\n", report);

    printf("\n");

    i = Bus_Delete(&bus, report, REPORT_MAX_SIZE, verbose, &trace, lspace);
    if (i != 0)
        printf("Erreur de suppression du bus\n\n%s\n", report);

    printf("\n");
}
```


C. Annexe 3 : Programme Mpp tournant sous DOS avec l'interface PCI/VME

```
C:\Debug>mpp -v 10 -W 0x290000 0x127c
```

```
Bus_Create starting Bus type:SBS Bit3 PCI / VME Interface
Bus_Create completed.
Bus_Open starting
  bt_open done
  bt_clrerr done
  bt_set_info done
  bt_mmap done: region 0 mapped at local addr 0xd81000 remote_addr 0x0 size=8192K.
  bt_lock done
Bus_Open done.
```

```
Bus_WriteShort address 0x290000 data 0x127c starting
Bus_Read_Write starting
Bus_Read_Write completed err = 0.
Bus_WriteShort done: addr=0x290000 data=0x127c (?? ??)
Bus_Close starting
  bt_unlock done
  bt_unmmap region 0 done
  bt_chkerr done
  bt_close done
Bus_Close completed.
```

```
Bus_Delete starting
Bus_Close starting
  bt_close done
Bus_Close completed.
Bus_Delete completed.
```

```
C:\Debug>mpp -R 0x2b0000 -v 10
```

```
Bus_Create starting Bus type:SBS Bit3 PCI / VME Interface
Bus_Create completed.
Bus_Open starting
  bt_open done
  bt_clrerr done
  bt_set_info done
  bt_mmap done: region 0 mapped at local addr 0xd81000 remote_addr 0x0 size=8192K.
  bt_lock done
Bus_Open done.
```

```
Bus_ReadShort address 0x2b0000 starting
Bus_Read_Write starting
```

Bus_Read_Write completed err = 0.
Bus_ReadShort done: addr=0x2b0000 data=0x18 (?? ??)
Bus_Close starting
bt_unlock done
bt_unmmap region 0 done
bt_chkerr done
bt_close done
Bus_Close completed.

Bus_Delete starting
Bus_Close starting
bt_close done
Bus_Close completed.
Bus_Delete completed.

D. Annexe 4 : Makefile servant à la création de bibliothèques sous lynx OS

```

#-----
# Common variables
#-----
# Platform variables
VAR_OS = lynx
VAR_HOST = ces
VAR_PLATF = $(VAR_OS)/$(VAR_HOST)

ifdef LYNX24

VAR_VERS = 2.4
ADD_CFLAGS = -DOS_TIMER -D_POSIX_MESSADE_PASSING
ADD_LIB =

else #ifndef LYNX24

VAR_VERS = 2.5
ADD_CFLAGS = -fcommon -D_XOPEN_SOURCE_EXTENDED -
D_POSIX_MESSADE_PASSING
ADD_LIB = lynx

endif #ifdef LYNX24

# Change this directory: sources are under this
DIR_ROOT = /home/manip/mnt/traps/shared/btrincaz/c

PROJ = adf

DIR_PROJ = $(DIR_ROOT)/$(PROJ)
DIR_PROJ_OS = $(DIR_PROJ)/$(VAR_OS)
DIR_PROJ_OS_HOST = $(DIR_PROJ_OS)/$(VAR_HOST)

# Directory variables for object files
DIR_OBJ = /home/manip/mnt/traps/shared/btrincaz/compil
DIR_OBJ_PROJ = $(DIR_OBJ)/$(PROJ)
DIR_OBJ_PROJ_OS = $(DIR_OBJ_PROJ)/$(VAR_OS)
DIR_OBJ_PROJ_HOST = $(DIR_OBJ_PROJ_OS)/$(VAR_HOST)
DIR_OBJ_PROJ_HOST_VERS = $(DIR_OBJ_PROJ_HOST)/$(VAR_VERS)

DIR_BIN = /home/manip/mnt/traps/shared/btrincaz/bin
DIR_BIN_PROJ = $(DIR_BIN)/$(PROJ)
DIR_BIN_PROJ_OS = $(DIR_BIN_PROJ)/$(VAR_OS)
DIR_BIN_PROJ_HOST = $(DIR_BIN_PROJ_OS)/$(VAR_HOST)
DIR_BIN_PROJ_HOST_VERS = $(DIR_BIN_PROJ_HOST)/$(VAR_VERS)

```

```

DIR_OBJ_TARGET    = $(DIR_OBJ_PROJ_HOST)/$(VAR_VERS)

DIR_LIB_TARGET    = $(DIR_BIN_PROJ_HOST)/$(VAR_VERS)

DIR_BIN_TARGET    = $(DIR_BIN_PROJ_HOST)/$(VAR_VERS)

#-----
# Commands and options
#-----
CC                = gcc
DFLAGS            = -DLYNXOS -DPOSIX4_MEMLK
WFLAGS            = -pedantic -Wuninitialized -Wimplicit -Wreturn-type
CFLAGS            = -mthreads -O3 $(DFLAGS) $(WFLAGS) $(ADD_CFLAGS)
LD                = gcc

#-----
# Libraries
#-----
LIBS              = -ladf -llynx -lm -lutils -lmisc_util -lmerio2

LLIBS              = -L$(DIR_LIB_TARGET)

#-----
# Include files
#-----
INCLUDE = \
    -I$(DIR_PROJ) -I$(DIR_PROJ_OS) -I$(DIR_PROJ_OS_HOST)

#-----
# All
#-----
all : lib adf_board adf_crate adf_interpreter adf_system mcs

#-----
# Clean
#-----
clean_obj :
    rm $(DIR_OBJ_TARGET)/*.o

clean_lib :
    rm $(DIR_LIB_TARGET)/libadf.a

clean_bin :
    rm $(DIR_BIN_TARGET)/adf_board_test

#-----
# Utility Library
#-----
lib: $(DIR_LIB_TARGET)/libadf.a

$(DIR_LIB_TARGET)/libadf.a : \

```

```

$(DIR_OBJ_TARGET)/adf_board.o \
$(DIR_OBJ_TARGET)/adf_crate.o \
$(DIR_OBJ_TARGET)/adf_interpreter.o \
$(DIR_OBJ_TARGET)/adf_system.o \
$(DIR_OBJ_TARGET)/mcs.o \
    ar r $(DIR_LIB_TARGET)/libadf.a \
    $(DIR_OBJ_TARGET)/adf_board.o \
    $(DIR_OBJ_TARGET)/adf_crate.o \
    $(DIR_OBJ_TARGET)/adf_interpreter.o \
    $(DIR_OBJ_TARGET)/adf_system.o \
    $(DIR_OBJ_TARGET)/mcs.o \
    ranlib $(DIR_LIB_TARGET)/libadf.a

#-----
# Adf_board
#-----
$(DIR_OBJ_TARGET)/adf_board.o : \
    $(DIR_PROJ_OS)/adf_board.c \
    $(DIR_PROJ)/adf_board.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ_OS)/adf_board.c \
        -o $(DIR_OBJ_TARGET)/adf_board.o

#-----
# Adf_crate
#-----
$(DIR_OBJ_TARGET)/adf_crate.o : \
    $(DIR_PROJ_OS)/adf_crate.c \
    $(DIR_PROJ)/adf_crate.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ_OS)/timer.c \
        -o $(DIR_OBJ_TARGET)/timer.o

#-----
# Adf_interpreter
#-----
$(DIR_OBJ_TARGET)/adf_interpreter.o : \
    $(DIR_PROJ)/adf_interpreter.c \
    $(DIR_PROJ)/adf_interpreter.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ)/adf_interpreter.c \
        -o $(DIR_OBJ_TARGET)/adf_interpreter.o

#-----
# Adf_system
#-----
$(DIR_OBJ_TARGET)/adf_system.o : \
    $(DIR_PROJ)/adf_system.c \
    $(DIR_PROJ)/adf_system.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ)/adf_system.c \

```

```

-o $(DIR_OBJ_TARGET)/adf_system.o

#-----
# Mcs
#-----
$(DIR_OBJ_TARGET)/mcs.o : \
    $(DIR_PROJ)/mcs.c \
    $(DIR_PROJ)/mcs.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ)/mcs.c \
        -o $(DIR_OBJ_TARGET)/mcs.o

#-----
# Misc_util Library
#-----
libmisc: $(DIR_LIB_TARGET)/libmisc_util.a

$(DIR_LIB_TARGET)/libmisc_util.a : \
    $(DIR_OBJ_TARGET)/misc_util.o \
    ar r $(DIR_LIB_TARGET)/libmisc_util.a \
    $(DIR_OBJ_TARGET)/misc_util.o \
    ranlib $(DIR_LIB_TARGET)/libmisc_util.a

#-----
# Misc_util
#-----
$(DIR_OBJ_TARGET)/misc_util.o : \
    $(DIR_PROJ)/misc_util.c \
    $(DIR_PROJ)/misc_util.h
    $(CC) $(CFLAGS) $(INCLUDE) \
        -c $(DIR_PROJ)/misc_util.c \
        -o $(DIR_OBJ_TARGET)/misc_util.o

#-----
# Adf_board_test
#-----
adf_board_test : $(DIR_BIN_TARGET)/synchro_test

$(DIR_BIN_TARGET)/adf_board_test : \
    $(DIR_OBJ_TARGET)/adf_board_test.o \
    $(DIR_LIB_TARGET)/libutils.a
    $(LD) $(CFLAGS) \
        $(DIR_OBJ_TARGET)/adf_board_test.o \
        $(LLIBS) $(LIBS) -o $(DIR_BIN_TARGET)/adf_board_test

$(DIR_OBJ_TARGET)/adf_board_test.o : \
    $(DIR_PROJ)/adf_board_test.c \
    $(DIR_PROJ)/adf_interpreter.h \

```

```
$(DIR_PROJ)/os_al.h
$(CC) $(CFLAGS) $(INCLUDE) \
-c $(DIR_PROJ)/adf_board_test.c \
-o $(DIR_OBJ_TARGET)/adf_board_test.o
```

E. Annexe 5 : Conception du fréquencemètre en VHDL

```

-----
--
-- PROJECT: D0 Run Frequency meter
--
-- AUTHOR: B. Trincaz btrincaz@dapnia.cea.fr
--
-- DATE AND HISTORY:
-- July 2004
--
-----

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_arith.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library work;

-----

entity freq is
  port(
    RESET    : in std_logic;           -- RESET
    H        : in std_logic;           -- HORLOGE
    FREQUENCE : in std_logic;           -- SIGNAL DONT LA
    FREQUENCE VARIE
    S0       : out std_logic_vector( 15 downto 0 ); -- SIGNAL DE SORTIE CODE
    SUR 16 BITS 0 A 65535
    HOLD_B   : in std_logic;           -- SIGNAL PERMETTANT LA
    LECTURE SUR LE BUS
  );
end freq;

architecture metre of freq is

  signal cpt0 : std_logic_vector (16 downto 0); -- Compteur par rapport l'horloge (pour faire 1
  ms)
  signal c0    : std_logic_vector (15 downto 0); -- Compteur prenant en compte le nombre
  de pulsations
  signal raz   : std_logic;           -- Signal permettant de commuter entre l'horloge et la
  fréquence

```

```
begin
```

```
-----
--
-- Processus du compteur de frequence
--
```

```
-----
diviseur : process (FREQUENCE, RESET, raz)
```

```
begin
```

```
  if (RESET='1' or raz='1') then
```

```
    c0<= (others => '0');
```

```
  -- Activation du signal frequence sur front montant
```

```
  elsif FREQUENCE'event and FREQUENCE = '1' then
```

```
    if c0 /= "1111111111111111" then
```

```
      c0 <= c0 + 1 ;      -- On compte le nombre de pulsations sur 1 ms
```

```
    end if;
```

```
  end if;
```

```
end process diviseur;
```

```
-----
--
-- Processus de permettant de retourner les valeurs de la fréquence
--
```

```
-----
seq : process (H, RESET)
```

```
begin
```

```
  if RESET='1' then
```

```
    S0<= (others => '0');
```

```
    cpt0<=(others => '0'); -- Le reset doit être asynchrone par rapport à l'horloge
```

```
    raz<='0';
```

```
  elsif H'event and H='1' then
```

```
    if cpt0 = "00011000011010100000" then
```

```
      cpt0<=(others => '0');
```

```
      raz<='1';
```

```
      if HOLD_B = '1' then
```

```
        S0<= c0;
```

```
      end if;
```

```
    else
```

```
      cpt0<=cpt0+1; -- Activation du compteur pour avoir 1 ms
```

```
      raz<='0';
```

```
    end if;
```

```
  end if;
```

```
end process;
```

```
end metre;
```

F. Annexe 6 : Banc de test du fréquencemètre

```

-----
-
--
--                               TEST BENCH DU COMPTEUR
--
-----
-

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

-----
-----
entity tb_freq is
    constant FMINI : integer := 1000;
    constant FMAXI : integer :=70000;
    constant TINC  : time  := 2 ms;
    constant FINC  : integer := 100;
    constant SIMULATION_DURATION : time := TINC * (1 + (FMAXI - FMINI) /
FINC);
    constant H_REF_CLOCK_PERIOD : time := 5 ns;
end tb_freq;

-----
-----
architecture tb_metre of tb_freq is

-----
-----
-- Signaux intermédiaires du bench
-----
-----
    signal tb_h          : std_logic := '1';
    signal tb_Rst        : std_logic := '0';
    signal tb_frequence  : std_logic;
    signal tb_S0         : STD_LOGIC_VECTOR ( 15 downto 0 );
    signal tb_testeur    : std_logic := '1';
    signal fre           : integer;
    signal go            : BOOLEAN := TRUE;

-----
-----
-- Appel des composants
-----
-----

    component frequence
        generic (
            fmin : integer := 0;
            fmax : integer := 80000;
            T_incrementation : time := 2 ms;
            f_incrementation : integer := 100
        );
        port (
            RESET : in std_logic;

```

```

        CLK    : out std_logic;
        F_OUT  : out integer
    );
end component;

component freq
port(
    RESET      : in std_logic;           -- RESET
    H          : in std_logic;           -- HORLOGE
    FREQUENCE  : in std_logic;           -- SIGNAL
DONT LA FREQUENCE VARIE
    S0         : out std_logic_vector( 15 downto 0 ); -- SIGNAL DE
SORTIE CODE SUR 16 BITS 0 A 65535
    HOLD_B     : in std_logic            -- SIGNAL
PERMETTANT LA LECTURE SUR LE BUS
);
end component;

```

```

-----
-----
begin

```

```

    --
    -- A linear frequency modulated generator
    --
    j:frequence
    generic map (
        FMINI,
        FMAXI,
        TINC,
        FINC
    )
    port map(
        RESET => tb_Rst,
        CLK   => tb_frequence,
        F_OUT => fre
    );

    --
    -- A frequency meter
    --
    f:freq
    port map (
        RESET      => tb_Rst,
        H          => tb_h,
        FREQUENCE  => tb_frequence,
        S0         => tb_S0,
        HOLD_B     => tb_testeur
    );

    --
    -- Reset signals
    --
    tb_Rst <= '1' after 0 ns, '0' after 100 ns;

    --
    -- Reference clock
    --
    tb_h <= not tb_h after H_REF_CLOCK_PERIOD;

```

```
--  
-- Hold 1 measurement  
--  
tb_testeur <= '1' after 0 ns, '0' after ((TINC*10) + 500 us), '1'  
after ((TINC*12) + 500 us);  
  
--  
-- Detect end of simulation  
--  
go <= TRUE after 0 ns, FALSE after SIMULATION_DURATION;  
ASSERT go  
    REPORT "Simulation ended normally"  
    SEVERITY FAILURE;  
end tb_metre;
```

G. Annexe 7 : Conception du signal module en fréquence

-- Code VHDL permettant de produire une rampe de fréquence

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.math_real.all;
--use work.utility_pkg.all;

entity frequence is
  generic (
    fmin : integer := 0;
    fmax : integer := 80000;
    T_incrementation : time := 2 ms;
    f_incrementation : integer := 100
  );
  port (
    RESET : in std_logic;
    CLK    : out std_logic;
    F_OUT  : out integer
  );
end frequence;

architecture behavioral of frequence is

  signal fre      : integer := fmax;
  signal fre_real : integer := fmax;
  signal hor      : std_logic := '0';
  signal clk_i    : std_logic := '0';
  signal h_per    : time := (1000000 ns)/(2*fmax);

begin
  Clock: process(RESET, hor)
  begin
    if RESET = '1' then
      fre<=fmin;
    elsif hor'event and hor = '1' then
      if (fre<fmax) then
        fre<=fre + f_incrementation;
      end if;
    end if;
  end process;

  hor    <= not hor after T_incrementation/2;
  clk_i  <= not clk_i after h_per;
  h_per  <= (1000000 ns)/(2*fre);

  --
  -- Affect outputs
  --
  CLK    <= clk_i;
  F_OUT  <= 500000 ns / h_per;

end behavioral;

```