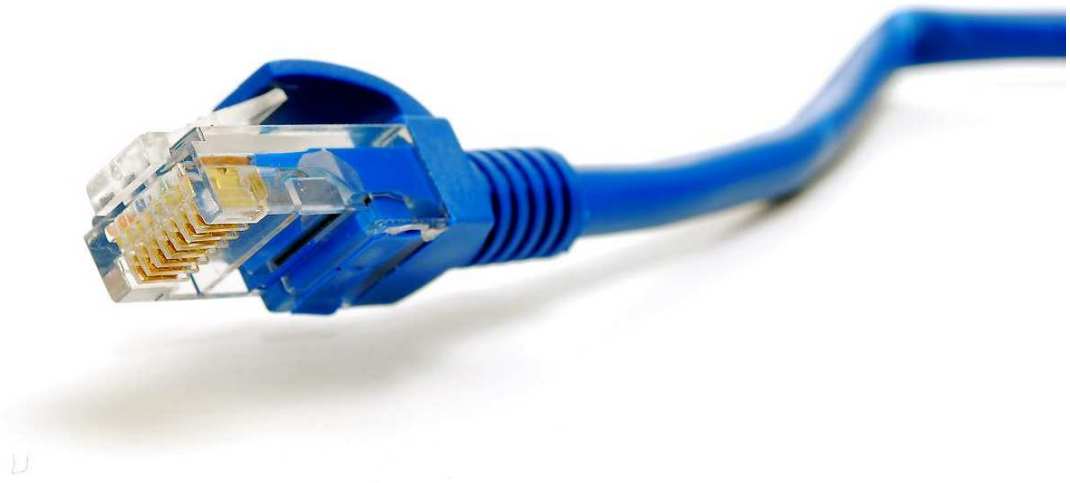
RAPPORT DE STAGE DE MASTER 1

FRANÇOIS BRENOT

CONCEPTION D'UNE LIAISON GIGAETHERNET



Année scolaire | 2010 - 2011

SOMMAIRE

REMERCIEMENTS

Tout d'abord je souhaite remercier Rémi CORNAT qui m'a permis d'effectuer mon stage au sein du laboratoire Leprince-Ringuet et qui, tout au long de mon projet, m'a soutenu. Je remercie également tout le corps professoral de l'Université Blaise Pascal de Clermont Ferrand en particulier François BERRY et Samuel MANEN pour m'avoir mis en contact avec ce laboratoire. Enfin je remercie mes collègues pour le soutien technique qu'ils m'ont apporté.

I. INTRODUCTION

Ce dossier est un compte rendu de projet réalisé dans le cadre d'un stage de Master 1 à l'université Blaise Pascal de Clermont-Ferrand. Ce dernier s'est déroulé au sein du LLR (Laboratoire Leprince Ringuet). Ce document présente ce stage en quatre axes. Tout d'abord, le cadre, le fonctionnement, les acteurs et les projets du LLR sont présentés. Dans une seconde partie, le projet est présenté de manière globale avec ses objectifs, puis de façon plus précise en développant l'objet du stage. Enfin, il s'achève sur l'état d'avancement du projet ainsi que son futur.

II.PRESENTATION DU LLR

1. Le Laboratoire Leprince-Ringuet (LLR)

Le Laboratoire Leprince-Ringuet (LLR) est une unité mixte de recherche (UMR7638) de l'Institut National de Physique Nucléaire et de Physique des Particules (IN2P3) du Centre National de la Recherche Scientifique (CNRS) et de l'Ecole Polytechnique. Il est implanté sur le site de l'Ecole Polytechnique à Palaiseau (91). Avant de détailler les missions du laboratoire Leprince Ringuet, il est important de voir les structures qui encadrent les travaux du LLR (c'est-à-dire le CNRS et l'IN2P3).

1.1 Le centre National de la Recherche Scientifique

Le Centre national de la recherche scientifique est un organisme publique de la recherche fondamentale (Établissement publique à caractère scientifique et technologique, placé sous la tutelle du Ministre chargé de la Recherche). Il produit du savoir et met ce savoir au service de la société. Avec plus de 34 000 personnes (dont 11 450 chercheurs et 14 180 ingénieurs, techniciens et administratifs), un budget primitif pour 2011 de 3,204 milliards d'Euros, une implantation sur l'ensemble du territoire national, le CNRS exerce son activité dans tous les champs de la connaissance, en s'appuyant sur 1200 unités de recherche et de service.

Ses activités

Le CNRS est présent dans toutes les disciplines majeures, regroupées au sein de huit départements scientifiques :

– Institut des sciences biologiques (INSB).

– Institut de chimie (INC).

– Institut écologie et environnement(INEE).

– Institut des sciences humaines et sociales (INSHS).

– Institut des sciences informatiques et de leurs interactions (INS2I).

– Institut des sciences de l'ingénierie et des systèmes(INSIS).

– Institut national des sciences mathématiques et leurs interactions (INSMI).

– Institut de physique (INP).

– Institut national de physique nucléaire et de physique de particules (IN2P3).

– Institut national des sciences de l'univers (INSU).

Le CNRS développe des collaborations entre spécialistes de différentes disciplines et les universités. Cela permet ainsi d'ouvrir de nouveaux champs d'investigations répondant aux besoins de la société. Des actions interdisciplinaires de recherche sont notamment menées dans les domaines suivants :

– Le Vivant et ses enjeux sociaux.

– Information, communication et connaissance.

– Environnement, énergie et développement durable.

– Nanosciences, nanotechnologies et matériaux.

– Astroparticules : des particules à l'Univers.

Dynamisme du CNRS

– 1 200 unités de recherche et de service dont 95 % en partenariat avec l'enseignement supérieur et les autres organismes de recherche en France ;

– 4382 brevets principaux en fin 2010, 495 nouveaux brevets déposés en 2010, 864 licences activées;

– 593 entreprises innovantes créées depuis 1999 ;

– 5000 chercheurs étrangers accueillis annuellement dans les laboratoires, 1714 chercheurs étrangers statutaires au CNRS, 85 accords de coopération avec 60 pays, 343 programmes internationaux de coopération scientifique, 114 laboratoires Européens et internationaux associés et 93 groupements de recherche Européens et internationaux, 22 unités mixtes internationales.

1.2 L'Institut National de Physique Nucléaire et de Physique des particules (IN2P3)

Créé en 1971, l'Institut National de Physique Nucléaire et de Physique des Particules (IN2P3) est un institut du CNRS dont la mission est de promouvoir et de fédérer les activités de recherche dans les domaines de la physique.

Ses missions :

Les expériences de recherche fondamentale effectuées au sein de l'Institut visent à identifier les constituants fondamentaux de la matière et à comprendre les interactions entre ces différents éléments. Ces expériences doivent permettre de déterminer l'origine des masses des particules et de comprendre le comportement de certains constituants élémentaires, allant des particules composites jusqu'aux noyaux. La mise en œuvre de grands programmes implique un partage des tâches et des échanges constants au sein de collaborations nationales et internationales. Ainsi la concertation et l'interaction sont deux aspects qui permettent de définir l'activité de recherche des laboratoires de l'IN 2P3.

Des liens privilégiés sont noués entre les chercheurs de l'IN2P3 et d'autres secteurs de la recherche tels que :

– L'astrophysique (aussi bien en nucléaire qu'en particulaire) et la cosmologie ;

– La chimie, dans les programmes de chimie nucléaire et de radiochimie

– La physique du solide et celle des matériaux ;

– La biologie par l'utilisation de l'imagerie médicale et des accélérateurs à des fins thérapeutiques.

Ses laboratoires :

Cette liste étant je vous invite à vous rendre sur le site de l'IN2P3 :

http://www.in2p3.fr/presentation/instances/instances_in2p3.htm

Ce stage est effectué au sein de l'un d'entre eux le laboratoire **Leprince-Ringuet** LLR.

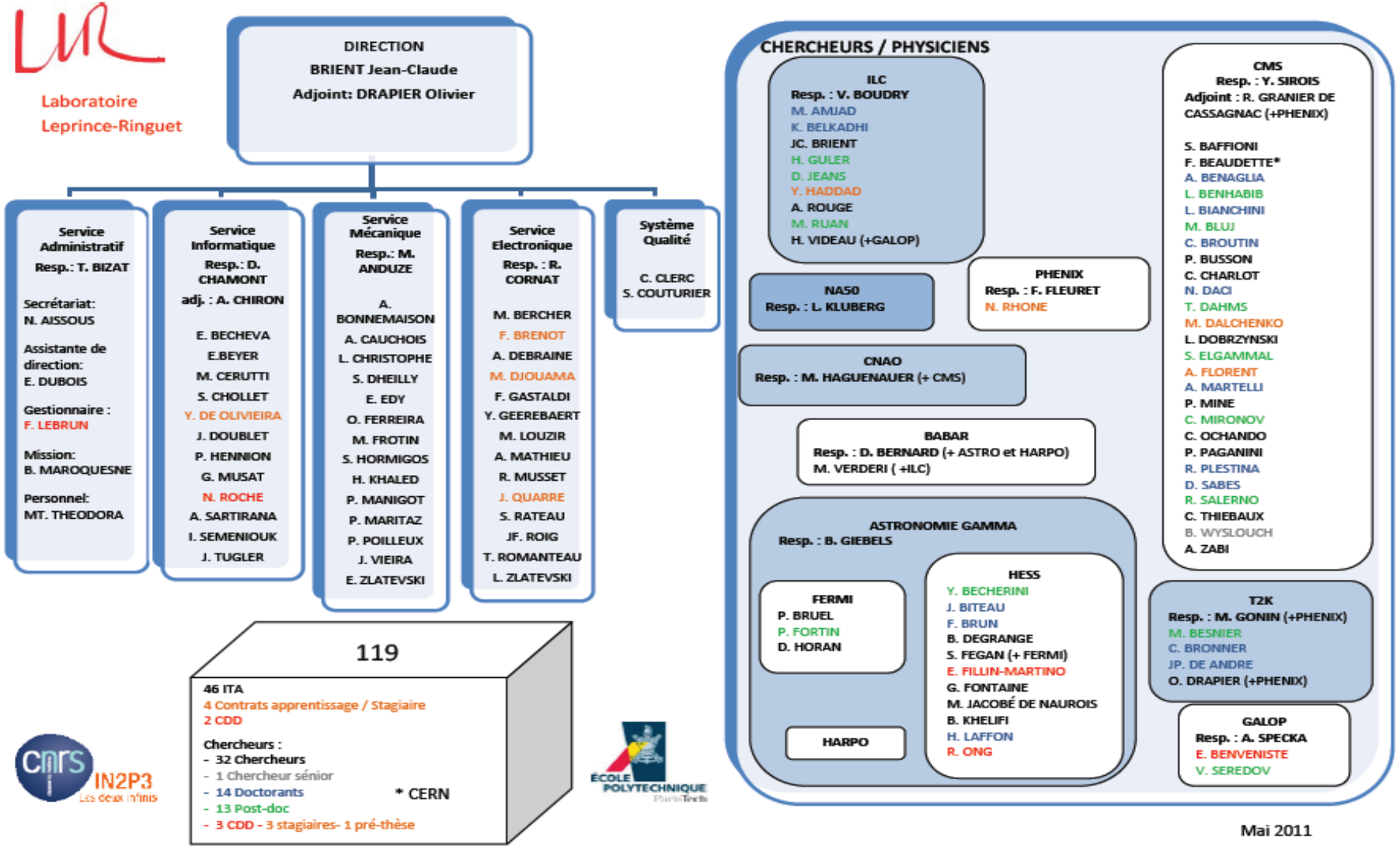


Figure 1 : Structure du laboratoire LLR

1.3 Présentation du Laboratoire Leprince-Ringuet LLR

Le laboratoire Louis Leprince Ringuet est situé dans l'enceinte de l'Ecole Polytechnique à Palaiseau. De 1974 à 2001, il a été connu sous le nom de Laboratoire de Physique Nucléaire des Hautes Energies (LPNHE-X). Ensuite, le physicien Louis Leprince Ringuet, chercheur et professeur à l'Ecole Polytechnique lui céda son nom. Le programme de recherche du laboratoire porte en premier lieu, sur la physique des particules auprès des accélérateurs du CERN (Suisse), de DESY (Allemagne), de SLAC (Californie, USA) et de PHENIX (Brookhaven, USA). De plus, depuis 1990, un programme d'astrophysique a été ouvert, d'abord au Mont Whipple (Arizona, USA), puis sur le site de Thémis dans les Pyrénées. En parallèle, des activités pluridisciplinaires se créent en fonction des synergies associées aux divers programmes.

Ses activités :

Le domaine d'étude du LLR-Polytechnique et les applications qui en découlent appartiennent à la physique des particules et à l'astrophysique. En parallèle, des travaux d'envergure pluridisciplinaires (souvent liés aux compétences des laboratoires voisins sur l'Ecole Polytechnique) sont traités. Chaque domaine est associés un groupe de chercheurs, des techniciens et des ingénieurs qui s'unissent pour travailler autour d'un même projet. Ainsi, actuellement, que ce soit en physique des particules ou en astrophysique, plusieurs projets sont menés conjointement.

Le groupe électronique :

Le groupe d'électronique comprend 11 ingénieurs et techniciens permanents. Deux spécialistes sont chargés de la conception des cartes et deux électroniciens assurent le câblage des prototypes et des petites séries produites au laboratoire. La première mission du service est de répondre aux demandes des physiciens que ce soit pour la préparation de nouvelles expériences ou la modification d'expériences en cours. le service assure également la conception, la réalisation et la maintenance des systèmes électroniques nécessaires aux expériences Parmi les Projet que le service électronique est en phase de réalisation en cite :

– R&D de système générique d'acquisition de données pour les futures expériences de Physique.

– Électronique d'acquisition et contrôle pour l'expérience ballon HARPO.

– Cartes TCC68 de déclenchement du sous détecteur ECAL (tonneau) de l'expérience CMS au CERN.

– Cartes TCC48 de déclenchement du sous détecteur ECAL (bouchons) de l'expérience CMS au CERN.

La seconde mission du groupe est de se maintenir au plus haut niveau de compétence possible, face à ce qui se fait dans les autres laboratoires et dans l'industrie. Cette mission difficile est accomplie au contact de collaborateurs et concurrents internationaux.

Pour remplir ses missions, le groupe dispose de puissants moyens de conception électronique, principalement basés sur un réseau de stations de travail SUN, mais aussi sur PC. Les logiciels de conception utilisés dans le service proviennent des sociétés **Cadence**, INCESIVE, Compass, Vantage, **Xilinx** et Altera.

Groupe mécanique :

Dispose de 15 ingénieurs et techniciens permanents. Sa mission intervient dans la conception, le calcul, mise au point, commandes et suivis dans l'industrie. Il effectue également les montages sur site.

Groupe informatique :

Actuellement, tous les secteurs d'activité du laboratoire utilisent quotidiennement des moyens informatiques : les chercheurs et informaticiens pour les développements de logiciels généraux ou liés aux expériences, les groupes d'électronique et de mécanique avec les techniques variées de CAO ainsi que les services administratifs avec les outils de bureautique sur micro-ordinateurs.

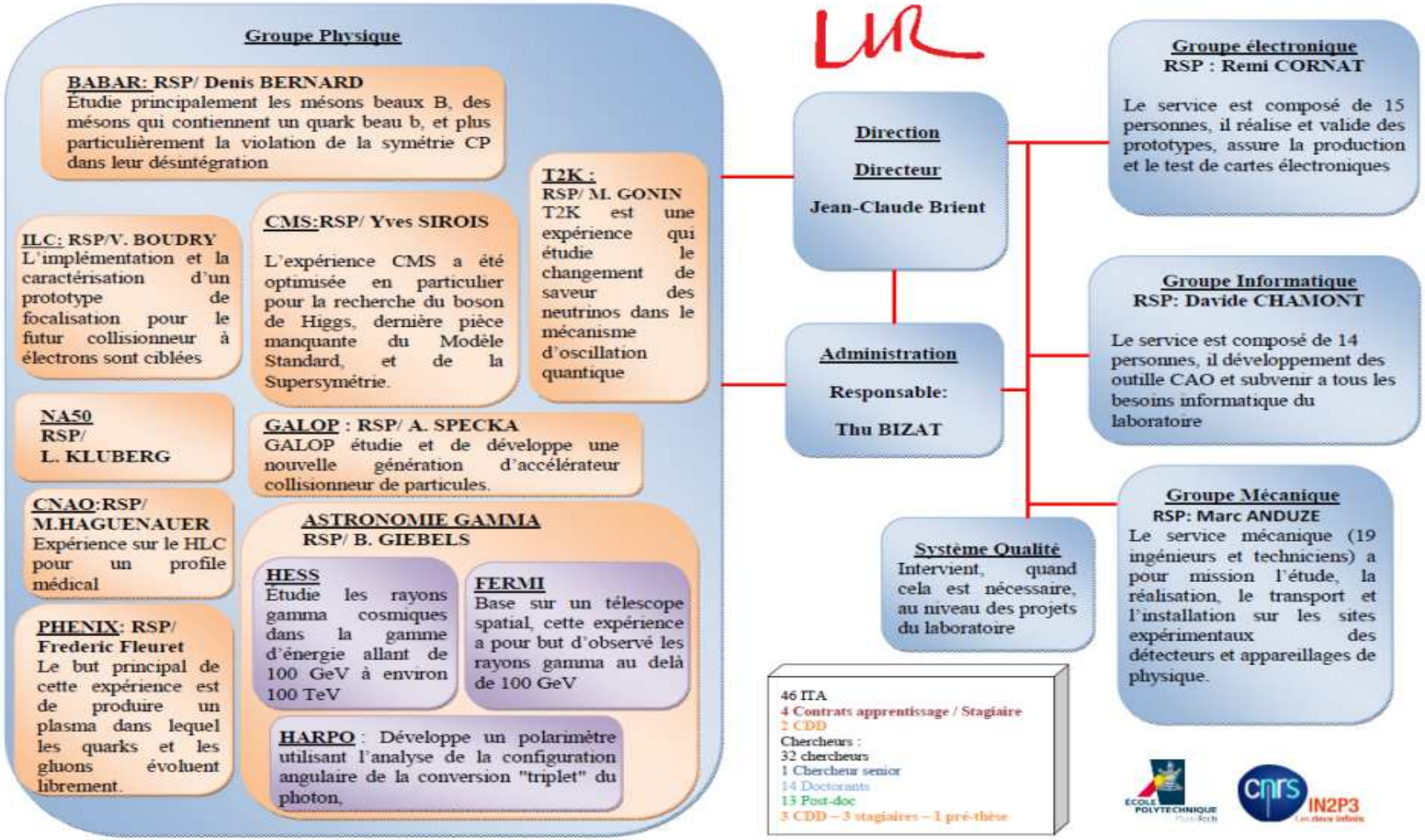


Figure 2: Les différentes activités du laboratoire LLR

III. LE PROJET

1. PRESENTATION DU PROJET

Ce sujet de stage répond à une demande de différentes manipulations. Il est rattaché à un projet nommé CALICE mais doit pouvoir être utilisé par les différents sujets du LLR.

1.1 Présentation : ILC - CALICE

ILC

L'ILC (International Linear Collider) est un projet de collisionneur d'électrons et de positrons à une énergie totale comprise entre 90 GeV et 1 TeV. Une telle machine sera en mesure de couvrir un large éventail de mesures de haute précision des particules et leurs interactions, telles le boson de Higgs. Elle apportera une contribution très significative à la compréhension de la physique des particules.

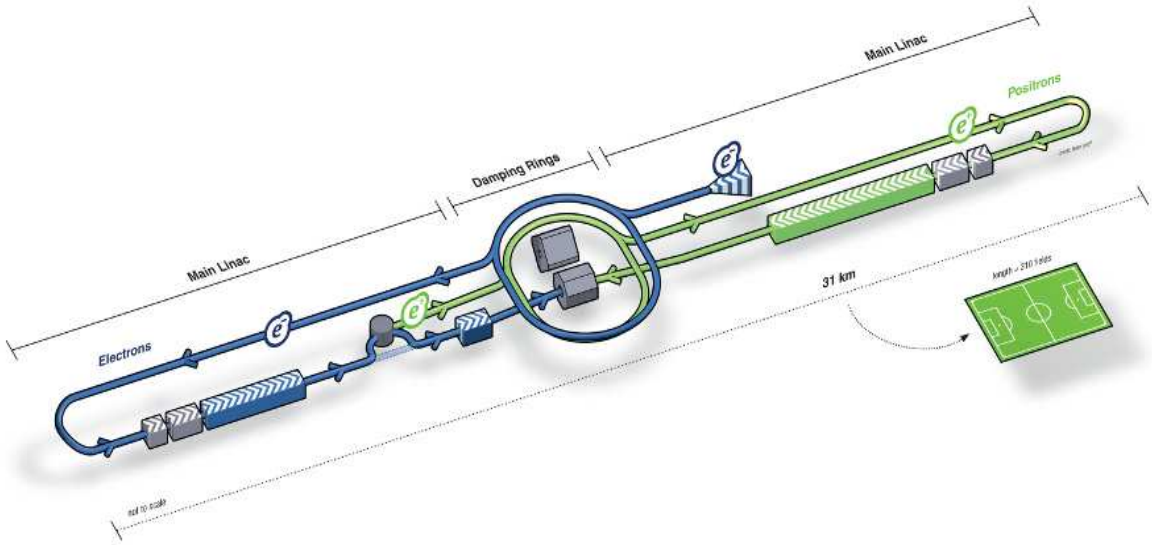


Figure 3: Conception d'ILC

Dans la configuration actuelle, l'ILC comprend deux bras : l'un pour accélérer les paquets d'électrons, l'autre pour les paquets de positrons. Chaque paquet contient environ 20 milliards de particules, avec une taille verticale de quelques nanomètres . Les paquets accélérés se rencontrent au centre du complexe, où les collisions se produisent 14 000 fois par seconde. La longueur totale de l'accélérateur sera d'environ 30 km et les particules accélérées par 16 000 cavités supraconductrices en Niobium fonctionnant à une température de 2ºK (-271 degrés Celsius). Le site d'installation de la machine sera décidé dans les prochaines années.

Figure 4 : Une structure de neuf cavités accélératrices

Détecteurs

Deux détecteurs sont prévus pour enregistrer ce qui est produit dans les collisions de particules. Ils profiteront des avancées technologiques récentes (en particulier de l'industrie électronique) pour améliorer significativement les performances par rapport aux expériences actuelles de physique des particules.

Ces détecteurs sont constitués de plusieurs sous-systèmes qui mesurent les différentes propriétés des particules produites lors des collisions.

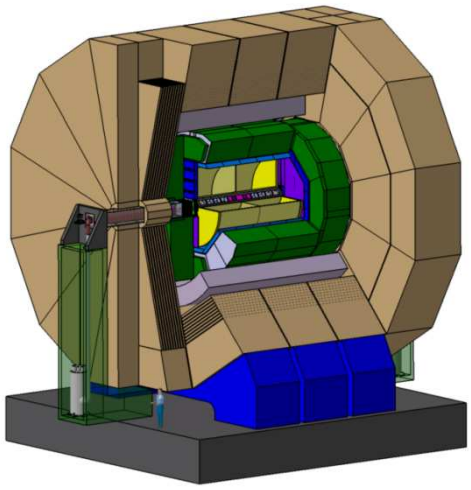


Figure 5: Conception du détecteur ILD

Calorimétrie

Le calorimètre est un sous-système d'un tel détecteur. Son rôle est de mesurer l'énergie des particules produites lors des collisions. Les calorimètres peuvent être considérés comme des appareils photo numériques, en 3 dimensions, enregistrant l'empreinte des interactions des particules et mesurant la quantité d'énergie qu'elles déposent. Les calorimètres de l'ILC sont conçus pour avoir 1000 fois plus de pixels que les appareils actuels. Ce seront des calorimètres "par imagerie" ultra-granulaires, avec plus de 100 millions de pixels.

La recherche sur ces calorimètres est effectuée au sein de la collaboration internationale CALICE.

Cette approche de la calorimétrie peut avoir des applications à d'autres expériences en physique des particules, d'autres domaines de la physique et au-delà, par exemple pour l'imagerie médicale.

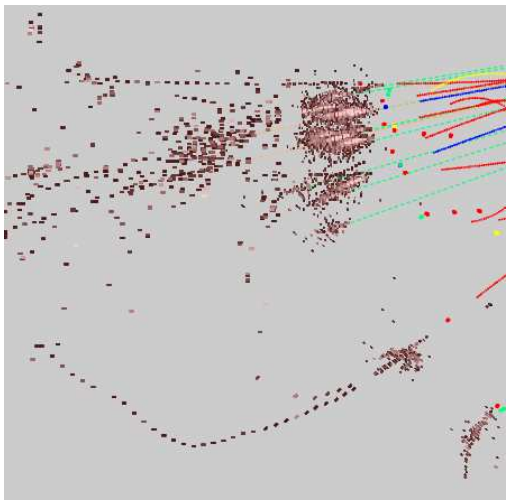


Figure 6 : Une gerbe hadronique vu avec un calorimètre à l'imagerie

Activités au LLR

Le groupe CALICE-ILC au LLR effectue des recherches dans 3 domaines principaux :

- Calorimétrie par imagerie
- Études d'un détecteur pour un collisionneur linéaire
- Études de focalisation finale des faisceaux d'électrons pour ILC

1.2 Réunion de lancement du projet

Ce stage a commencé par une réunion qui a permis de cibler les attentes de chacun et d'établir un cahier des charges. Le compte rendu de réunion se trouve en annexes page E. Au cours de cet échange, chaque représentant des différents services concernés (CALICE, HARPO, GALO, ILM, CMS, TLB) par cette interface Giga Ethernet a pu présenter son projet et s'exprimer à propos de ses

besoins. D'autre part cette rencontre a permis une immersion efficace et pertinente, dans ce sujet (familiarisation avec le vocabulaire, les intervenants, le contexte, ...)

1.3 Cahier des charges

Dans le cadre de ce stage, l'objectif est de prendre en main une carte d'évaluation Xilinx SP601, de comprendre et identifier le code d'exemple d'une liaison giga Ethernet fourni par la société et enfin, de modifier ce système pour lui permettre de réaliser des entrées/sorties Ethernet. Un PING entre un ordinateur et la carte peut être une solution pour valider la réception et l'émission.

PING : Une demande de PING permet de tester l'accessibilité d'une autre machine à travers un réseau. Cette requête est en fait une question :

L'ordinateur dit : Je suis *MACHINE_PC* et je souhaite parler à *CARTE_XILINX* voici quelques données *05210521450023*...

La Carte Xilinx concernée répond : Je suis *CARTE_XILINX* et je souhaite parler à *MACHINE_PC*, voici vos données *05210521450023*...

Un réseau Ethernet pouvant être composé de multitude de machines interconnectée le PING permet de savoir s'il est possible de dialoguer avec une machine spécifique.

CAHIER DES CHARGES

Contexte

L'un des projets actuellement développé au LLR s'appelle **CALICE**. Il est lié à une expérience en cours d'élaboration : l'ILC (International Linear Collider). Le département électronique du laboratoire doit fournir une carte capable de rassembler, mettre en forme et envoyer des données reçues de capteurs directement situés sur le collisionneur linéaire. Elles doivent donc être gérées de façon **fiable**, pour n'avoir aucune perte, et de manière très **rapide** puisque le flot d'information est conséquent.

Objectif

L'objectif principal est d'envoyer des données recueillies par des capteurs. Pour cela, il est nécessaire de les recevoir, les mettre en forme et les **transmettre** vers des cellules de stockage (gérées par un autre organisme). Le sujet de stage qui fait partie ce projet Calice ne concerne que la gestion de l'**envoi**. Les données sont donc déjà recueillies et formatées.

De plus, ce type de communication intéresse plusieurs autres projets du LLR. Il doit donc répondre à différentes demandes.

Moyens mis en œuvre et calendrier :

- Prendre en main une carte d'évaluation Xilinx (SP601) qui permettra d'avoir un banc de test.
- Tester cette carte afin de valider ce banc d'essai.
- Développer un circuit d'après une base (fournie par Xilinx) capable de réaliser une communication de base entre la carte et un PC via une liaison Giga Ethernet selon le protocole UDP.
- Améliorer ce circuit en ajoutant différentes fonctions.
- Durée de l'étude : 5 mois.

Contraintes

- Utiliser le kit de développement de Xilinx SP601 basé sur un FPGA de type Spartan 6.
- Utiliser le Verilog.
- Programmer à l'aide des outils Xilinx (ISE).

Intervenant :

François BRENOT : Stagiaire

Permanant : Rémi CORNAT : Ingénieur de Recherche.

1.4 Déroulement du projet

Ce projet s'est déroulé de façon spontanée, suivant un ordre chronologique, logique, par rapport au un développement d'un produit. Tout d'abord, le projet a débuté par une phase d'étude d'un système existant, puis par le développement et s'est terminé par la restitution.

En voici une représentation à l'aide du diagramme de Gantt :

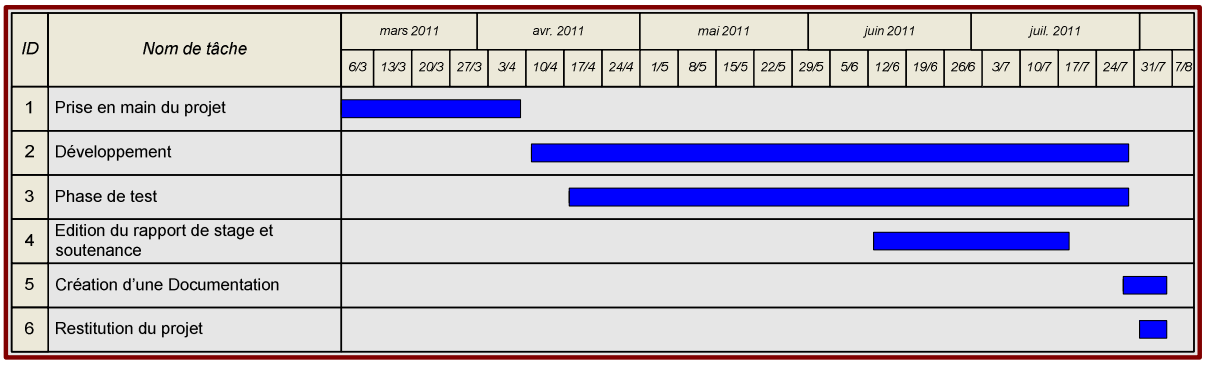


Figure 7 : Diagramme de Gantt du projet

Cette figure met en valeur les principaux axes du stage en fonction du temps :

- La prise en main du projet : Cette étape est tout d'abord, une **découverte** de l'étude Calice, puis, une familiarisation avec le langage Verilog et la **compréhension** du système existant proposé par la société Xilinx (sur lequel on reviendra dans la suite de ce rapport). Cette dernière partie est en fait du **reverse ingeniering**.

- La phase de développement et de simulation : Intervenant après s'être approprié le système de Xilinx, elle débute la partie R&D du projet. Cette dernière se déroule en 3 phases qui sont : la **modification** et **adaptation** du code initial, la **simulation** et enfin l'**implémentation** sur le banc de test.

- La restitution du projet : Cette dernière étape est un **retour d'expérience** avec l'état d'avancement du projet ainsi que l'avenir de l'étude. Une documentation doit être créée pour permettre aux personnes voulant utiliser ou continuer le développement, de pouvoir prendre rapidement et facilement en main le projet.

1.5 Le protocole de communication Ethernet :

Le protocole Ethernet permet de communiquer dans un réseau de machines connectées entre elles. Il est très vaste car l'avancée technologique ajoute toujours de nouvelles normes. Une présentation de la norme utilisée (IEEE 802.3) est donc indispensable avant de présenter ce projet plus en détails.

Dans un réseau Ethernet, le câble diffuse les données à toutes les machines connectées, de la même manière que les ondes radiofréquences parviennent à tous les récepteurs. Le nom Ethernet dérive de cette analogie1 : Avant le XXe siècle, on imaginait que les ondes se propageaient dans l'éther, milieu hypothétique censé baigner l'Univers. Quant au suffixe net, il s'agit de l'abréviation du mot network (réseau) en anglais.

Structure de l'entête

Voici un exemple d'une trame Ethernet :

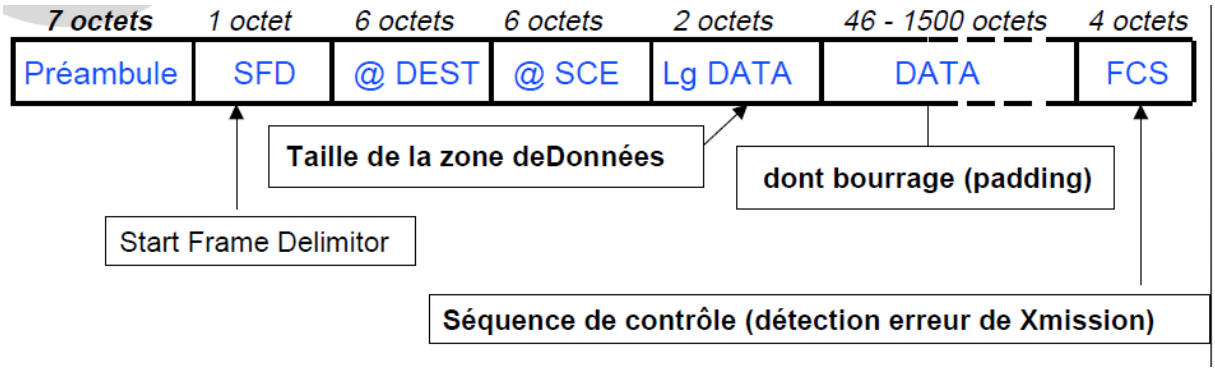


Figure 8 : Trame Ethernet

Définition des différents champs :

Préambule

Ce champ est codé sur 7 octets et permet de **synchroniser** l'envoi. Chacun des octets vaut 10101010 et cette série permet à la carte réceptrice de synchroniser son horloge.

SFD

Ce champ est codé sur 1 octet et indique à la carte réceptrice que le **début de la trame** va commencer. La valeur de SFD (Starting Frame Delimiter) est 10101011.

Adresse destination

Ce champ est codé sur 6 octets et représente l'**adresse** MAC (Medium Access Control) de l'adaptateur **destinataire**. Dans le cadre d'un broadcast[1], l'adresse utilisée est FF-FF-FF-FF-FF-FF.

Cette adresse est ce que l'on appelle l'adresse physique d'une carte Ethernet (Hardware address). Elle est divisée en deux parties égales :

- Les trois premiers octets désignent le constructeur. C'est l'organisation OUI (Organizationally Unique Identifier) gérée par l'IEEE, qui référence ces correspondances.
- Les trois derniers octets désignent le numéro d'identifiant de la carte dont la valeur est laissée à l'initiative du constructeur qui possède le préfixe.

L'association de l'IEEE et du constructeur assure ainsi l'unicité de l'attribution des numéros d'adresse MAC.

Adresse source

Tout comme l'adresse destination, ce champ est codé sur 6 octets et représente l'**adresse** MAC (Medium Access Control) de l'adaptateur **émetteur**.

[1] Broadcast : information(s) adressée(s) à l'ensemble du réseau.

Ether Type

Ce champ est codé sur 2 octets et indique **le type de protocole** inséré dans le champ « donnée » ou la **taille de la trame** :

- De 0 à 1500 (valeur décimale), il est interprété comme le champ "**longueur**" et indique la longueur du champ "donnée".
- Au-delà de 1500 (ou 05DC en hexadécimal), il est interprété comme le champ « **type »** et indique la nature du protocole de niveau supérieur. Voici un extrait des différentes correspondances :

0x6000 - DEC
0x0609 - DEC
0x0600 - XNS
0x0800 - **IPv4**
0x0806 - ARP
0x8019 - Domain
0x8035 - RARP
0x809B - AppleTalk
0x8100 - 802.1Q
0x86DD **- IPv6**

...

Ce niveau de protocole correspond à la couche supérieure (la couche réseau) du modèle OSI (cf. page 21).

Dans le cadre de ce projet, il est important de construire une base simple. Ce champ est donc utilisé comme indicateur de taille de trame (sa valeur sera inférieure à 1500). Une fois cette étape validée, on pourra facilement, dans une future étude, utiliser une de ces surcouches en utilisant cette base MAC comme cela :

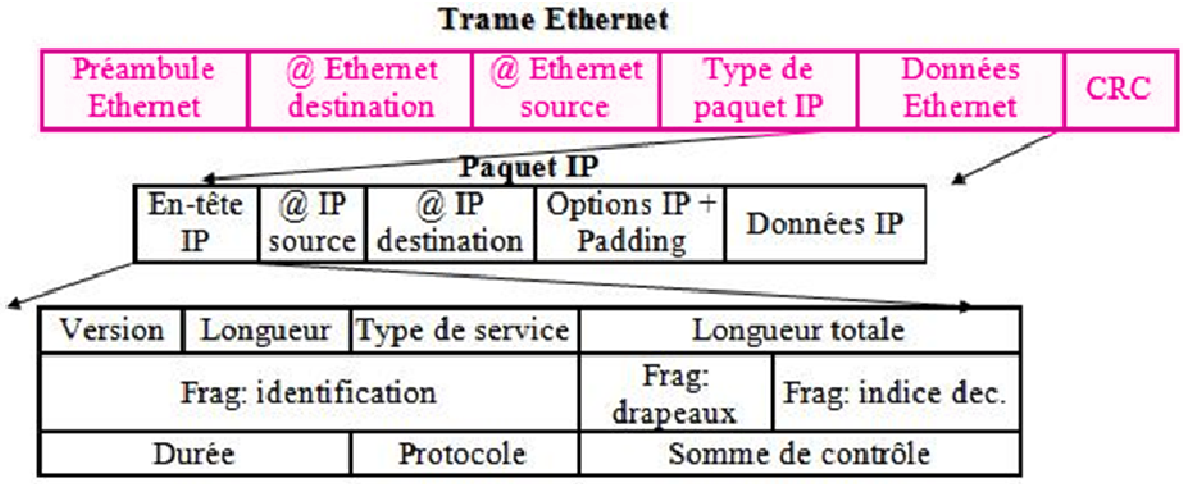


Figure 9 : Trame IPv6

FCS

Ce champ est codé sur 4 octets et représente **la séquence de contrôle** de trame. Il permet à l'adaptateur qui réceptionnera cette trame de détecter toute erreur pouvant s'être glissée au sein de la trame.

Les erreurs binaires sont principalement créées par les variations d'affaiblissement du signal et l'induction électromagnétique parasite dans les câbles Ethernet ou les cartes d'interface. La valeur de FCS (Frame Check Sequence) est le résultat d'un calcul polynomial appelé CRC (Cyclic Redundancy Code). A la réception de la trame, la couche liaison effectue le même calcul et **compare** les deux résultats qui doivent être égaux afin de valider la conformité de la trame reçue.

Ethernet peut être résumé grâce à un modèle O.S.I. à 7 couches (Open System Interconnection) :

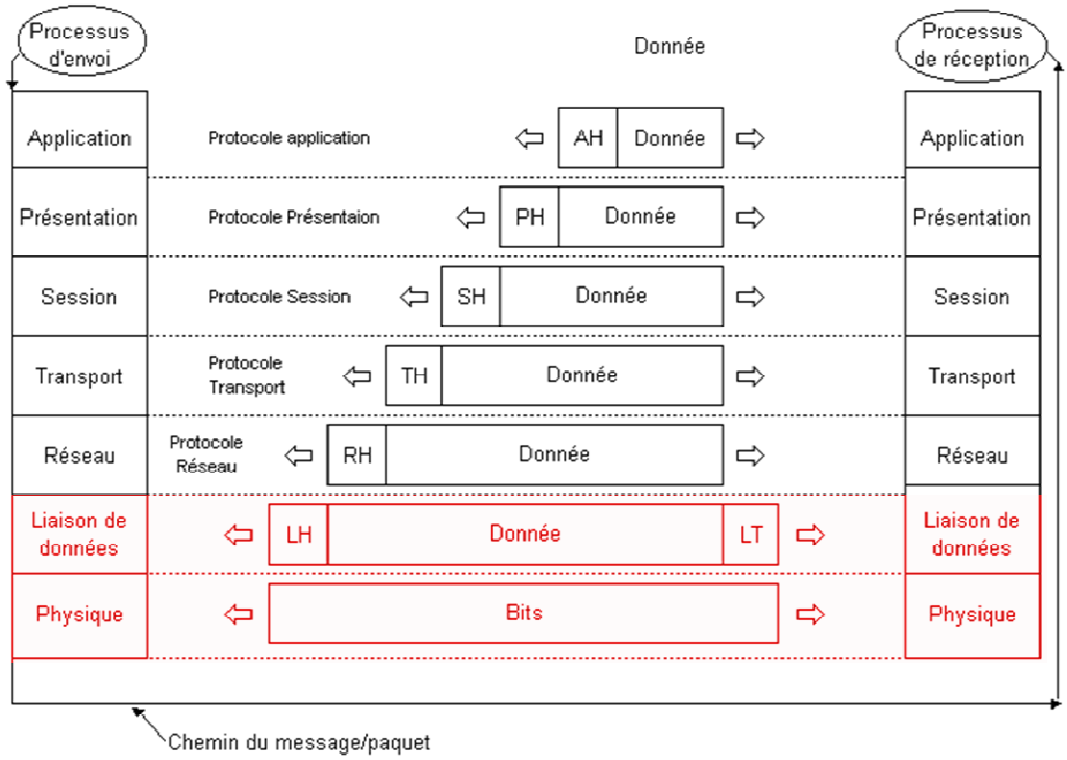


Figure 10 : Modèle OSI

AH :Application Header	NH : Network Header
PH : Presentation Header	LH : Link Header
SH : Session Header	LH : Link Header
TH : Transport Header	SDU : Service Data Unit

Dans ce projet, on utilisera uniquement les 2 dernières étapes. La couche **PHY** : la couche physique est réalisée par un composant externe au FPGA (mais présent sur la carte d'évaluation Xilinx) : Marvell (88E1111) (cf. présentation de la carte page : 23). La couche **Liaison de données** est assurée par le FPGA. La première est en réalité la transformation de la trame en une forme dite physique

donc de '0' et de '1'. La seconde est la couche basique présentée page 19 . Comme écrit précédemment, pour faire l'analogie avec le champ Ether type, la troisième couche (la couche réseau) pourrait être une surcouche IPv4 , IPv6 , ou AppleTalk...

Ethernet est un protocole de réseau local à commutation de **paquets**. Bien qu'il implémente la couche physique (PHY) et la sous-couche Media Access Control (MAC) du modèle OSI, le protocole Ethernet est classé dans la couche de liaison car les formats de trames que le standard définit sont normalisés et peuvent être **encapsulés** dans des protocoles autres que ses propres couches physiques MAC et PHY. Ces couches physiques font l'objet de normes séparées en fonction des débits, du support de transmission, de la longueur des liaisons et des conditions environnementales.

Exemple des versions de la norme 802.3 de la couche PHY :

- 802.3 10 base 5 ®Ethernet « épais » (Ethernet jaune)

Câble coaxial épais de 50 W, longueur max d ’un segment : 500 m

- 802.3 10 base 2 : Ethernet « fin » (Thin Ethernet)

Câble coaxial souple, longueur max d ’un segment : 180 m

- 802.3 10 base T : Ethernet 10 base T

Paire torsadée, longueur max d ’un segment : 100 m

- **802.3 1000 base T** : 1 Gbit/s

Paire torsadée, longueur max d ’un segment : 100 m

2. Le projet en détails :

2.1. Exemple de Xilinx :

Cet exemple est associé à la carte d'évaluation de Xilinx SP601 :

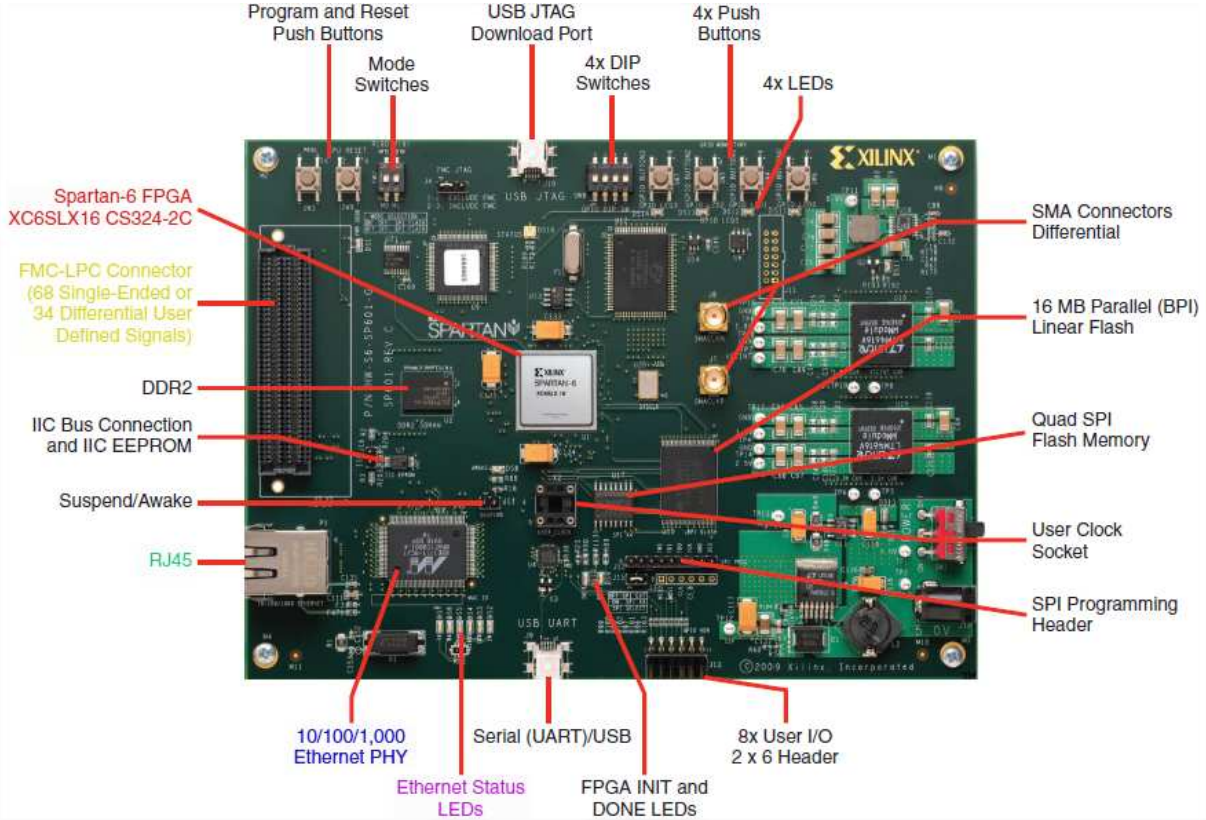


Figure 11 : Carte d'évaluation Xilinx SP601

Sur cette photo, nous pouvons voir différentes parties importantes :

- Le FPGA (en rouge) : Spartan-6 XC6SLX16 in CS324 package.
- 10/100/1000 Tri-Speed Ethernet PHY (en bleu) : Marvell Alaska PHY (88E1111).
- Port RJ45 (en vert).
- Un connecteur LPC (en jaune) : permettant notamment de connecter une mezzanine pour faciliter l'accès aux pins du FPGA pour des mesures
- Des LED (en violet) pour indiquer le bon établissement de la communication ainsi que la vitesse de transmission.
- USB JTAG : pour programmer le FPGA.

L'exemple est un outil servant de démonstration pour la carte. Voici la présentation de ce système.

Ce module permet de faire du traitement d'image. Xilinx fournit deux parties dans cet exemple. Une partie de code électronique en Verilog et une autre software. La partie software permet d'envoyer une image choisie vers la carte d'évaluation. C'est une interface graphique pour PC :

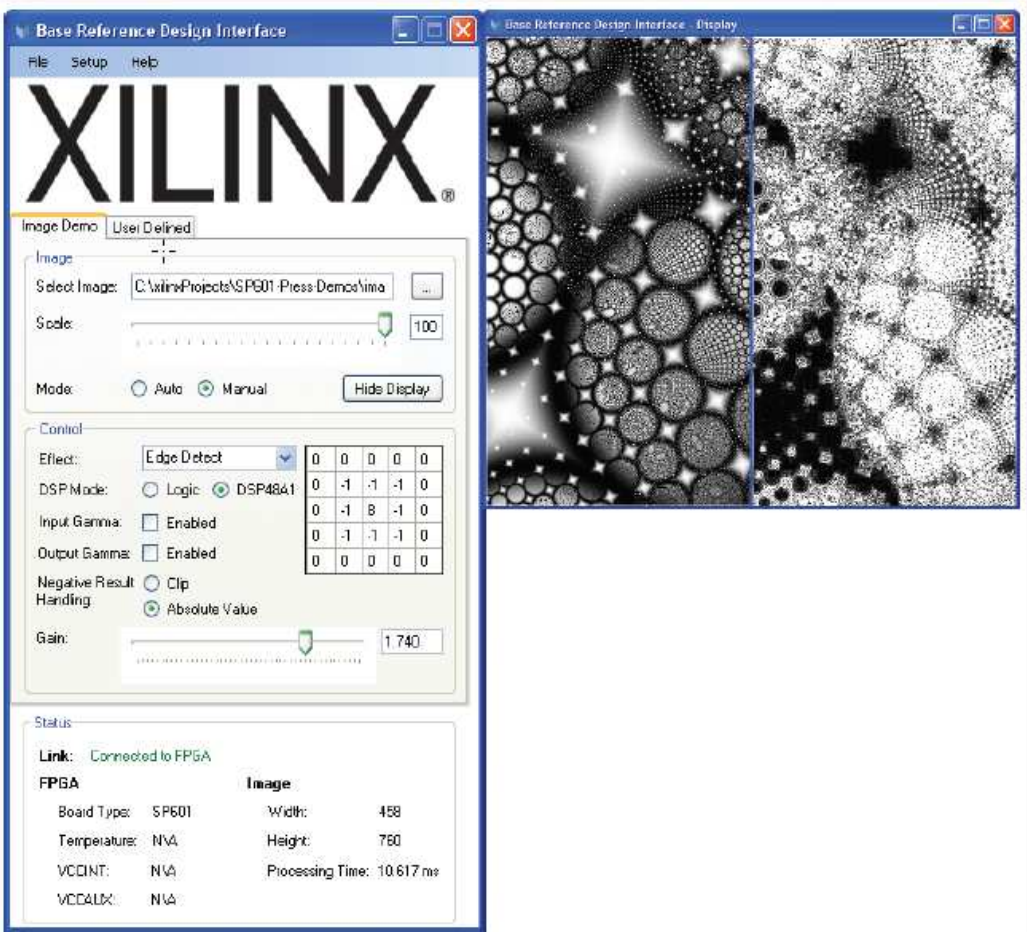
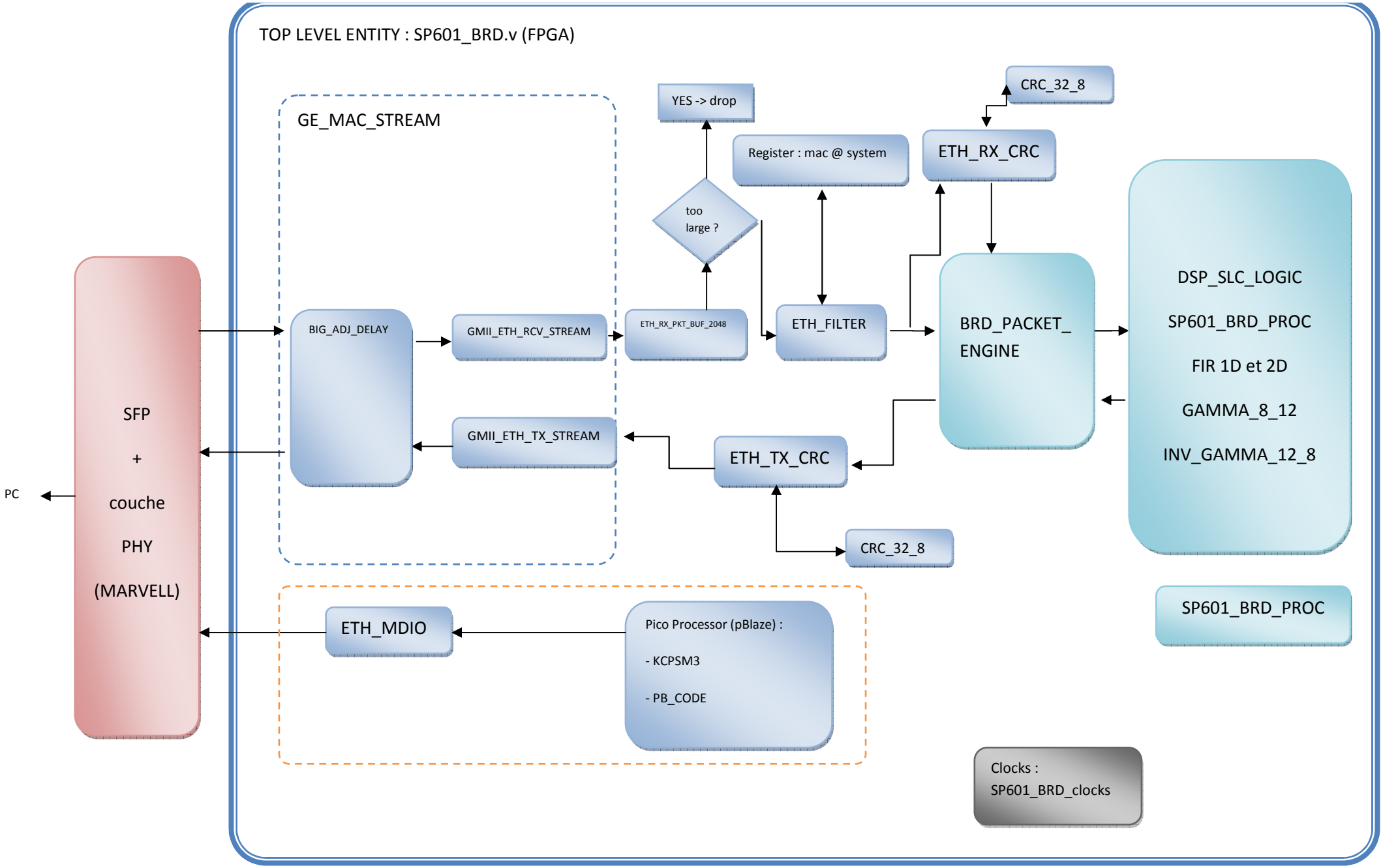


Figure 12 : Interface graphique de l'exemple de Xilinx

Le FPGA (situé sur la carte Xilinx) est donc le **calculateur** permettant d'appliquer des filtres sur l'image. Une fois l'image traitée par l'électronique, elle est **renvoyée** vers le software pour l'affichage du résultat. L'intérêt de cet exemple réside dans la **communication** entre le PC et le FPGA : L'image est envoyée et réceptionnée via une transmission Giga-Ethernet (filaire : câble RJ45).

Voici un schéma synoptique du programme :

Figure 13 : Schéma synoptique de l'exemple de Xilinx



Sur ce schéma synoptique, on peut remarquer en rouge **l'interface physique** (PHY) et dans l'encadré bleu (Le FPGA), la **couche liaison de données** du modèle OSI. Dans ce même cadre bleu, nous apercevons les **blocs en Verilog** implémentés dans le FPGA. De plus, on peut trouver 4 parties:

- GE_MAC_STREAM (pointillés bleu) : Ces modules permettent de faire l'**interfaçage** entre la couche PHY (Marvell) et ce système. Nous expliquerons sa fonction en détails par la suite.
- Clocks (bloc gris) : Ce module génère les **horloges** qui sont diffusées dans la quasi-totalité des modules.
- Pico_blaze (pointillés orange) : Ce bloc est un **picoprocesseur**, intervenant à la mise sous tension de la carte, il **configure les registres** du Marvell. Il est composé d'un petit ALU (contenant des opérations très basiques) et d'une mémoire incluant les opérations à faire : lecture/écriture de registre.
- Le reste des modules Verilog constituent la **création/vérification** (blocs bleu ciel) des trames ainsi que le **stockage** et **traitement** d'images (blocs bleu azur).

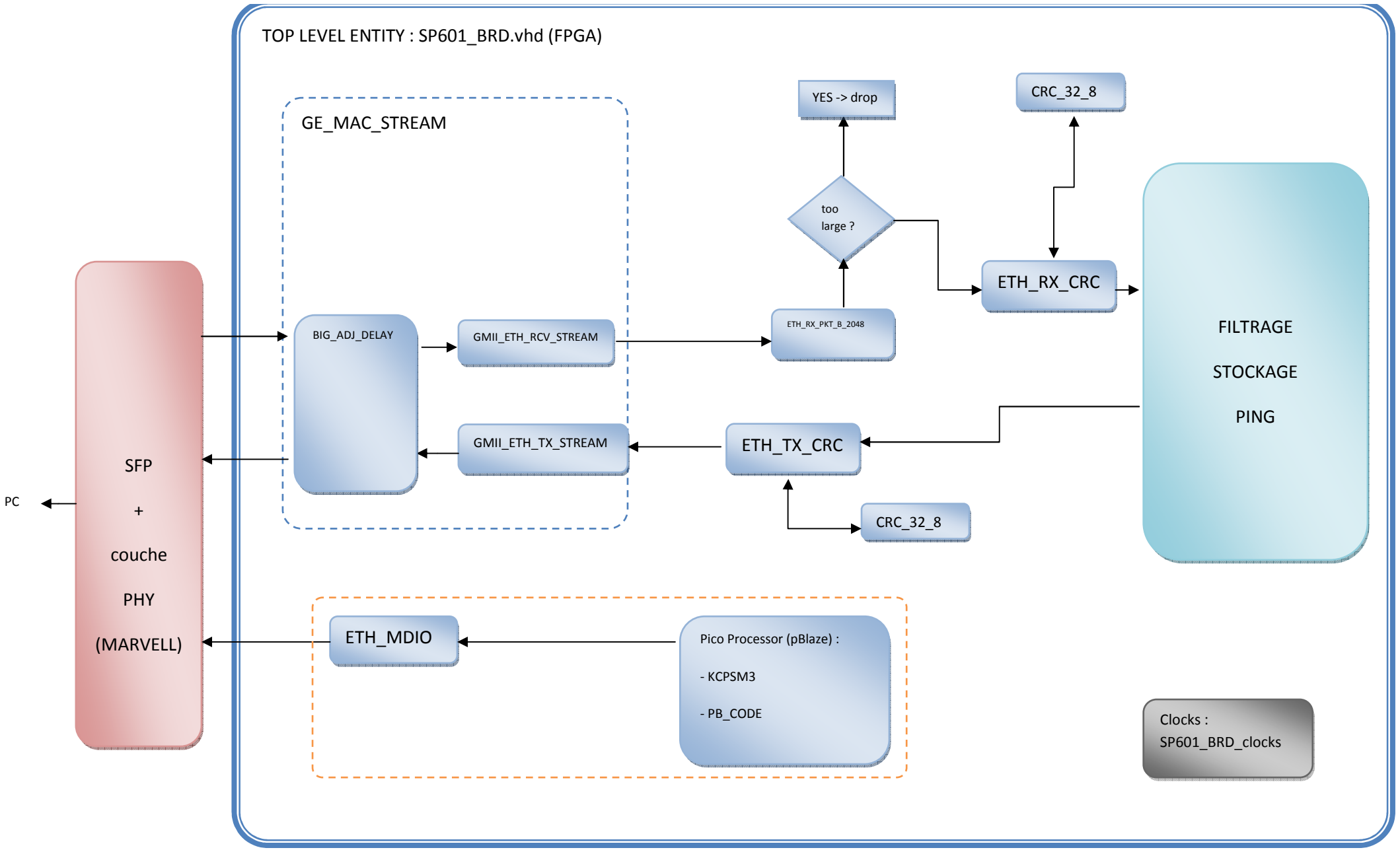
2.2. Travail réalisé

Le premier objectif était de **prendre en main** l'exemple de Xilinx, le second est de **générer un Ping** pour tester la carte et établir une transmission.

Afin de n'utiliser que les parties nécessaires à la transmission Giga Ethernet, les blocs bleu azur, correspondants à la gestion de l'image et à son traitement, doivent être remplacés dans un premier temps par un module qui génère un Ping. Ceci permet de valider la réception ainsi que la transmission de données. Pour cela, il faut d'abord créer ces modules et modifier le fichier de Top-level pour instancier ces nouveaux blocs.

Pour modifier l'entité de plus haut niveau appelée SP601_BRD il a fallu tout d'abord la **traduire** du Verilog vers le VHDL avant d'**instancier** les nouveaux blocs. Voici la nouvelle structure de la nouvelle entité :

Figure 14 : Schéma synoptique du projet



2.3. Fonctionnement du circuit : Gestion de la trame Ethernet

On suppose une trame envoyée par un ordinateur et réceptionnée par la carte Xilinx. Cette trame répond au protocole **Ethernet**. Cette dernière est transmise via un câble Ethernet donc sur une liaison série. Elle est réceptionnée par la couche Physique (voir modèle OSI page 21) qui est assurée par un circuit dédié : Marvell (88E1111).

- Marvell (88E1111)

Ce composant est un **désérialiseur/sérialiseur** qui transforme cette trame série pure (codé en manchester sur 1 bit) en une trame série et **parallèle** codée sur un bus de 8 bits. Il ajoute des **signaux de contrôle** qui seront détaillés dans la suite de ce rapport. Cette trame est donc de la forme suivante :

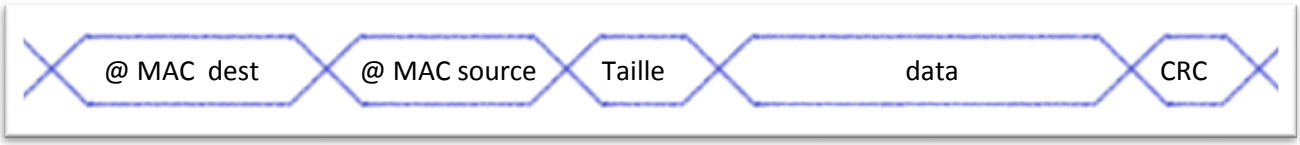


Figure 15 : Trame Ethernet réceptionnée et désérialiser (sortie du Marvell)

Cette information codée sur 8 bits va être transmise au FPGA. Elle est traitée par un premier module appelé GE_MAC_STREAM (voir page précédente).

- GE_MAC_STREAM (partie réception)

Ce module permet de faire une première gestion des informations comme le **contrôle** de la **vitesse** de transmission (100Mb/s ou 1000Mb/s).Cependant, dans notre cas, nous n'utiliserons que du Giga-Ethernet : cette fonction n'a donc que peu d'intérêt hormis un test pour être sûr d'être dans les bonnes conditions de transmission.

Il permet aussi de créer une **file d'attente** de trame pour ne perdre aucune information car le Marvell fournit les trames à la vitesse où il les reçoit.

- ETH_RX_PKT_B_2048

La trame Ethernet est ensuite transmise à un bloc appelé ETH_RX_PKT_B_2048 qui **vérifie la longueur** de la trame. En effet, pour respecter la norme Ethernet cette dernière doit avoir une taille **maximum** de 1526 octets. Si elle ne remplit pas ce critère, elle est immédiatement rejetée.

- ETH_RX_CRC

Ce module de contrôle de CRC (Cyclic Redundancy Check**)** va effectuer un calcul sur la trame pour le tester avec celui déjà présent sur cette dernière et ainsi **vérifier son intégrité**. Ce test peut être apparenté à un contrôle de parité.

- Filtrage - stockage - Ping

La trame Ethernet n'a pas changé de format puisqu'elle n'a subi que des contrôles. Le schéma synoptique ci-dessous montre comment fonctionne ce module.

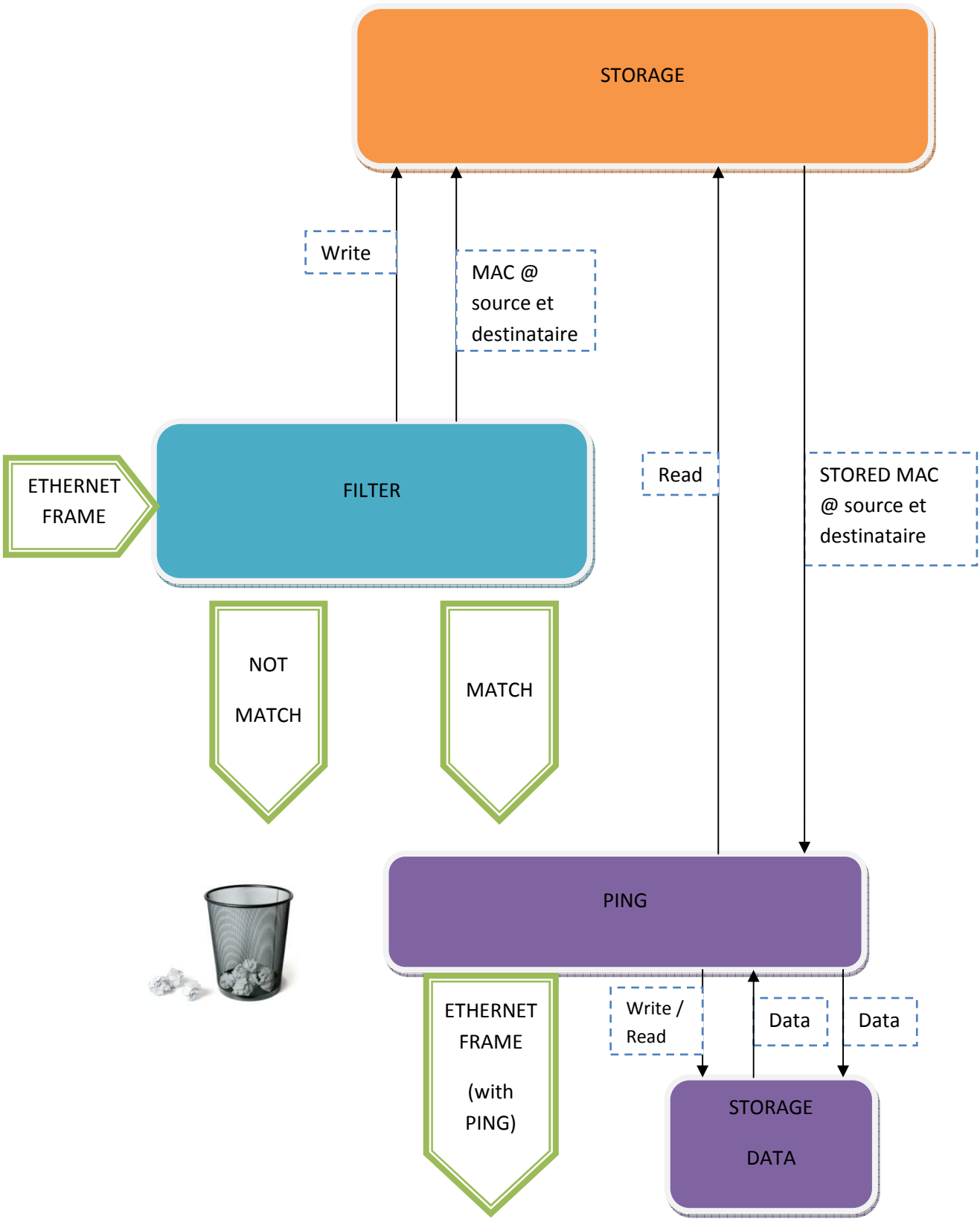


Figure 16 : Schéma synoptique Filtrage - Stockage - Ping

- La trame est tout d'abord **filtrée** (bloc bleu) à l'aide d'une comparaison de la MAC adresse de destinataire inclue dans la trame et une MAC adresse définie dans le code.

- Les adresses MAC de source et destination sont **stockées** dans un registre (bloc orange). Cette opération est effectuée en même temps que l'opération de filtrage.

-La trame Ethernet est **reconstruite** par le bloc appelé PING (en violet). Ayant rencontré des problèmes pour valider la partie émission de données, il a fallu créer **différentes versions** de ce bloc. Une première version utilise une machine d'état, une deuxième utilise une registre à décalage et enfin, une troisième, un registre préprogrammé (trame figée dans le fichier VHDL). Les deux premières versions sont sensiblement identiques (mis à part l'efficacité du code). En revanche, la dernière permet de simplifier le programme pour ne pas faire de Ping, elle permet de se focaliser sur la partie envoi de données.

- Une fois la trame **reconstruite**, elle est transmise au circuit d'envoi de trame pour être formatée aux normes Ethernet.

- ETH_TX_CRC

Ce module reçoit la trame reconstruite, il **calcule** alors le CRC et l'ajoute à cette dernière.

- GMII_ETH_TX_STREAM

GMII: Gigabit Media Independent Interface.

Ce dernier circuit permet de faire l'**interfaçage** entre la couche **MAC** et la couche **PHY** (Marvell).

2.4. Fonctionnement du circuit : Les signaux de contrôles

Pour gérer cette trame Ethernet il faut différents signaux de contrôle. Deux signaux sont créés au début de cette chaine par le Marvell. Ci-dessous la forme de ces signaux :

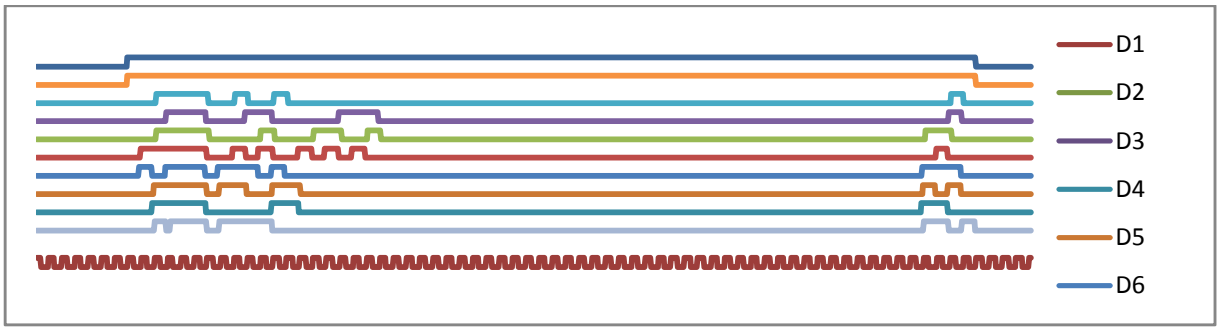


Figure 17 : Chronogramme de la trame Ethernet

Ce chronogramme représente la trame Ethernet obtenue à la sortie du Marvell. Deux signaux de **contrôle** sont situés en haut de ce schéma (en bleu D6 et orange D5). En dessous, on remarque 8 bits de **données** (La trame Ethernet). Et enfin, en rouge l'horloge principale du système D1.

Ces deux signaux de contrôle permettent à tous les modules de connaitre le **début** et la **fin** de la trame. Ils permettent une parfaite synchronisation pour la réception et le traitement de données. De plus, ils servent de "**handcheck**" entre le circuit intégré au FPGA et le Marvell.

2.4 La couche PHYsique : Marvell 88E1111

Le composant Marvell 88E1111 assure la partie **Physique** du modèle OSI, appelé aussi transceiver[2] Giga Ethernet. Il est capable de travailler sur les classifications : **1000BASE-T**, 100BASE-T et 10BASE-T de la norme 802.3 (Ethernet). Ce sont en fait les vitesses de transmission. Exemple : 1000BASE-T pour le Giga Ethernet.

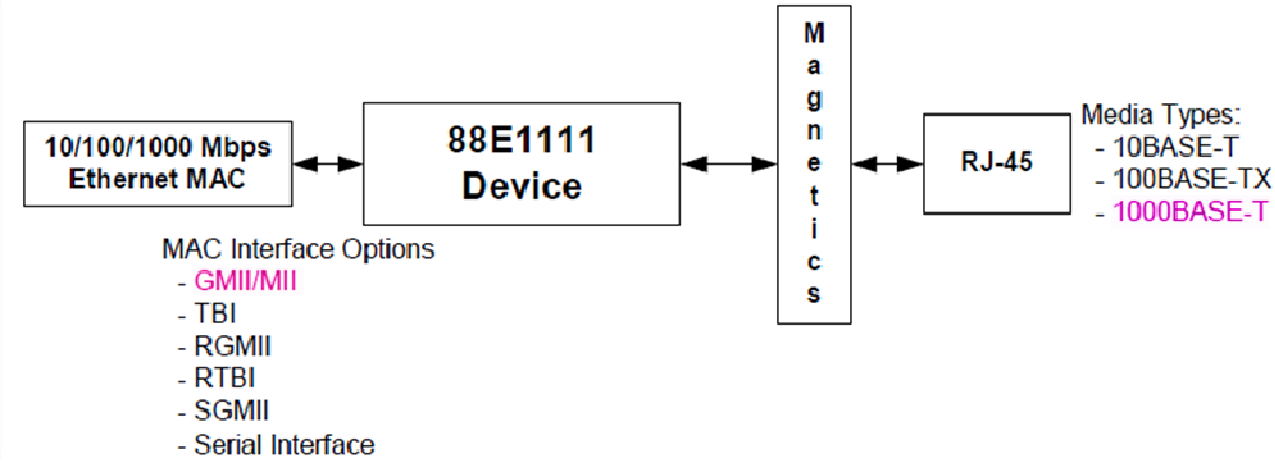


Figure 18 : Application possible avec le Marvell 88E1111

Le Marvell est **paramétrable** via ses registres.

Il est en fait un sérializer/desérializer. Lors de la réception il récupère les données en **séries** (codées en Manchester) qu'il transforme en données **parallèles**. Exemple :

Données sur le câble Ethernet : ...0101 0000 0111 1010 0110 1111...

Données en sortie du composant : ...5 0 7 A 6 F... (Sur un BUS : 4 bit) codé en hexadécimal.

3. Les Tests

Après une période de programmation, une période de test a permis de valider le circuit. Voici la méthode utilisée pour valider le circuit :

- Validation par simulation.
- Synthèse.
- Insertion des probes (cf : 3.3 L'implémentation).
- Mise en place du banc de test (Oscilloscope, PC, sondes).
- Mesure les signaux.

[2] Transceiver: TRANSmitter and a reCEIVER. Gestion de l'émission et de la réception dans le même boitier.

Plusieurs mesures ont étés effectuées sur le banc de test pour **valider** différents blocs. Ce protocole a donc été suivi à de nombreuses reprises.

3.1. La simulation

Dans un premier temps, pour être cohérent avec le logiciel **Xilinx ISE** (logiciel qui implémentera le circuit dans le FPGA par la suite) nous avons utilisé le logiciel **ModelSim**. Puis, rapidement, des problèmes de compatibilité de librairies sont apparus. Il a fallu migrer le projet sous environnement Unix pour travailler avec le logiciel de simulation de **cadence**.

Pour réaliser cette simulation sur ordinateur, il a fallu mettre en place un **banc de test** comprenant le circuit à simuler et un deuxième circuit permettant d'exciter des signaux pour **émuler** la couche physique (le Marvell).

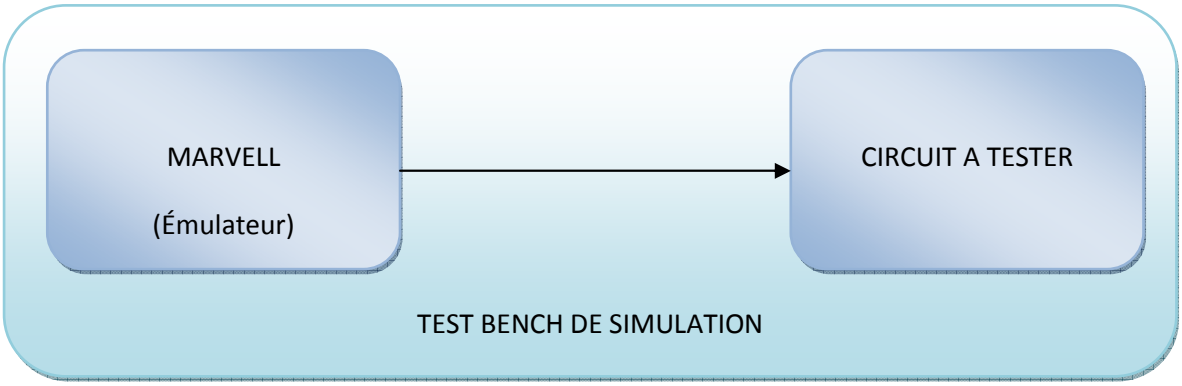


Figure 19 : Synoptique du test bench de simulation

Emulation du Marvell

Ce module se présente de cette façon :

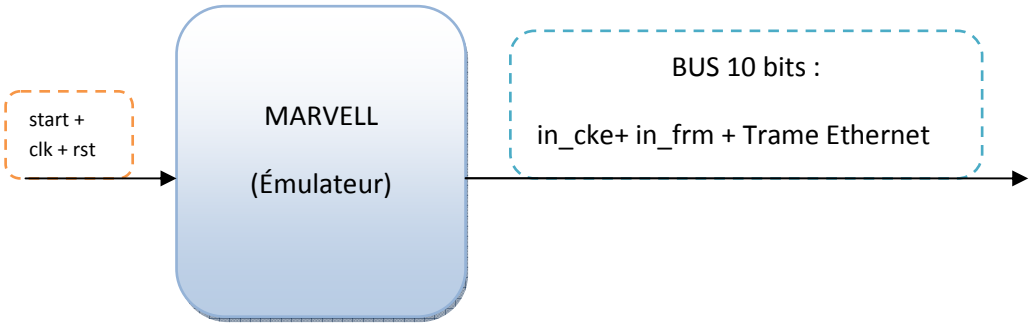


Figure 20 : Synoptique du module d'émulation du test bench

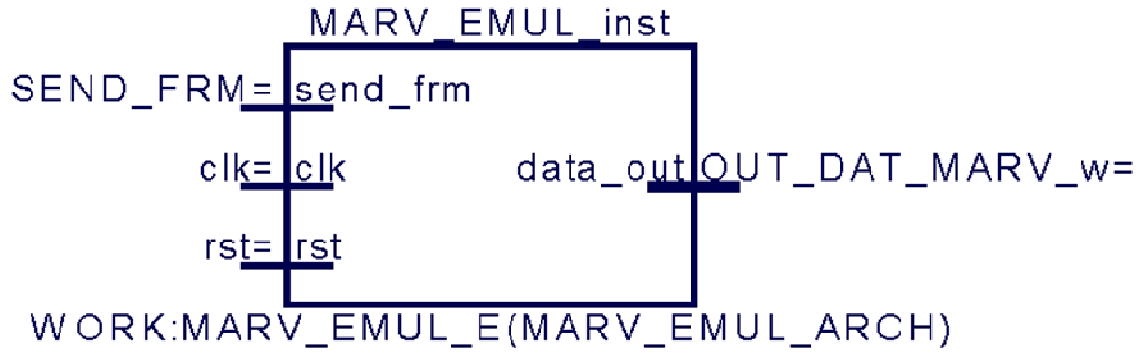


Figure 21 : Synoptique des E/S de l'émulateur du Marvell interprété par le simulateur

Cet émulateur est réalisé avec une machine d'état (voir annexe page : F). Elle fonctionne de la manière suivante :

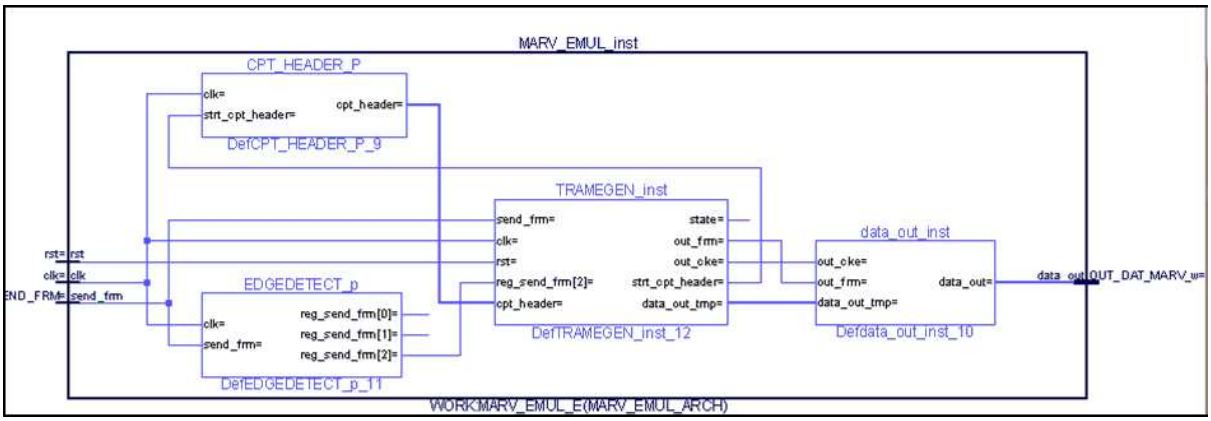


Figure 22 : Synoptique du module d'émulation du Marvell interprété par le simulateur

Quatre modules peuvent être distingués :

- EDGEDETECT_p : C'est un process qui permet de détecter un front montant sur le bit de Start appelé SEND_FRM (entrée sur 1 bit). Ce module est nécessaire puisque l'instruction rising_edge (détection de front montant) ne peut être sensible qu'à une horloge. Des tests ont d'ailleurs été effectués pour vérifier cela. Le logiciel synplify renvoit alors l'erreur suivant :

E: CL123 :"C:\Users\User10062\Desktop\Test detection fronts\detect_with_rising_edge_a.vhdl":

29:1:29:2|The logic for out_data_event does not match a standard flip-flop.

On est donc obligé d'utiliser deux bascules D :

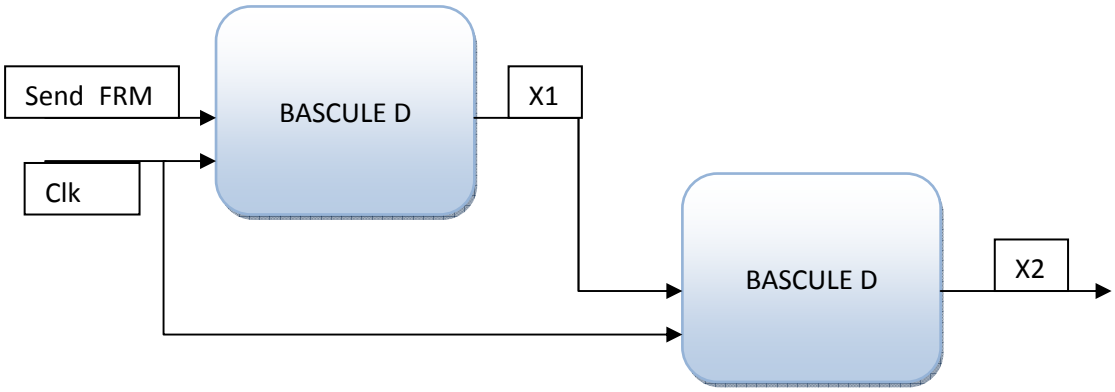


Figure 23 : Synoptique du module de détection de front

Ainsi, si X2='0' et Send_FRM ='1', nous sommes en présence d'un **front montant**.

- CPT_HEADER_P : C'est un process permettant de compter jusqu'à 7 pour l'envoi de 7 bits de **préambule** "55" (header_1 sur le chronogramme suivant).
- TRAME_GEN_inst : C'est le **générateur de trame**, ici une machine d'état. Ces états sont visibles dans le chronogramme ci-après.
- data_out_inst : C'est un simple **registre**. Il permet de réunir le bus d'information (sur 8 bits) et les deux signaux de contrôles (in_frm et in_cke sur 1 bit chacun) en un seul BUS de 10 bits.

Sur le chronogramme suivant, on voit une trame simplifiée (trop courte et sans CRC). La partie haute est le signal de l'état de la machine. On peut trouver header_1 et header_2 pour les signaux de synchronisation et start (5555555 5D), l'adresse MAC destinataire, l'adresse MAC source, length pour la taille de la trame et enfin data_1, _2, _3 pour les données. La partie basse de cette mesure est la trame construite que l'on injecte dans le circuit à tester.

Figure 24 : Trame Ethernet sortante de l'émulateur de Marvell

Une fois les deux modules assemblés, l'émulateur et le circuit à tester, on réalise des mesures à différents points du circuit. Par exemple, voici un chronogramme avec 7 points de tests. Parmi ces points, en haut, nous pouvons voir la mesure évoquée précédemment en sortie d'émulateur de Marvell et la trame de sortie en fin de chaine de traitement.

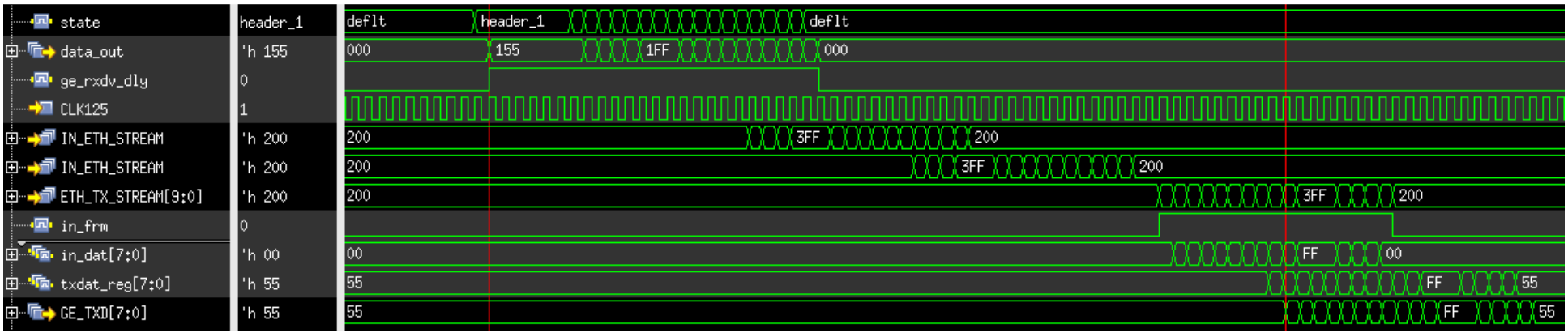


Figure 25 : Propagation de la trame dans le circuit

A l'aide de cette simulation on peut calculer le **delta** entre le moment où l'on reçoit la trame et le moment où l'on est capable de la renvoyer delta = 465 ns (2 droites rouges sur le chronogramme). Ce temps de latence est dû aux registres et dépend de la longueur de la trame reçue. Ceci est dû au nombre de données à stocker en mémoire avant de reconstituer la trame. Cette mesure est donc faite avec une trame de test incomplète de 128 ns (n'ayant pas la longueur minimale d'une trame Ethernet et pas de CRC). On peut calculer facilement le temps de latence en fonction du nombre d'octets. En effet, si la trame fait un octet de plus il faudra un coup d'horloge de plus pour stocker l'information. Donc pour un octet = 8 ns de retard. On sait que la longueur minimum d'une trame Ethernet est de 72 octets (dont 46 octets de data) donc delta **minimum** = 809 ns. De plus, la longueur maximale est de 1526 octets (dont 1500 octets de data) donc delta **maximum** = 12 441 ns.

Sur les deux relevés ci-dessous, on peut voir un zoom sur la trame entrante dans le circuit et un zoom sur la trame sortante qui a été traitée.

Figure 26 : Trame entrante dans le circuit à tester

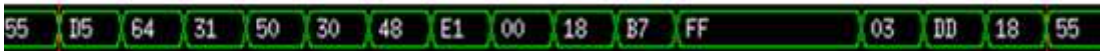


Figure 27 : Trame sortante après traitement par le circuit

On peut voir que le circuit de réception fonctionne parfaitement puisqu'il a bien modifié la trame pour exécuter un PING. Ce dernier est réalisé en prenant la trame et en inversant les adresses MAC de source et de destination. En résumé un PING est la réponse à une question :

- On reçoit l'information : Je suis 64-31-50-30-48-E1, je souhaite parler à 00-18-B7-FF-FF-FF, voici la taille des données envoyées 03 et voici les données DD-18-55.
- On répond : Je suis 00-18-B7-FF-FF-FF, je souhaite parler à 64-31-50-30-48-E1, voici la taille des données envoyées 03 et voici les données DD-18-55.

A l'aide de ces mesures, on sait alors que le circuit est capable de recevoir une information et de la traiter. Il peut donc être synthétisé et ensuite testé sur le banc de test afin de valider la partie réception.

Une fois que tous les blocs ont étés simulés d'après ce même schéma, ils sont transférés vers le logiciel Synplify pour en faire la synthèse.

3.2. La synthèse

Pour synthétiser ce circuit, on a utilisé le logiciel **Synplify** de la société **Synopsys**, un simulateur externe au simulateur de Xilinx ISE. Ce choix a été fait pour des raisons de **performances**. Voici le synoptique du flot de conception choisi :

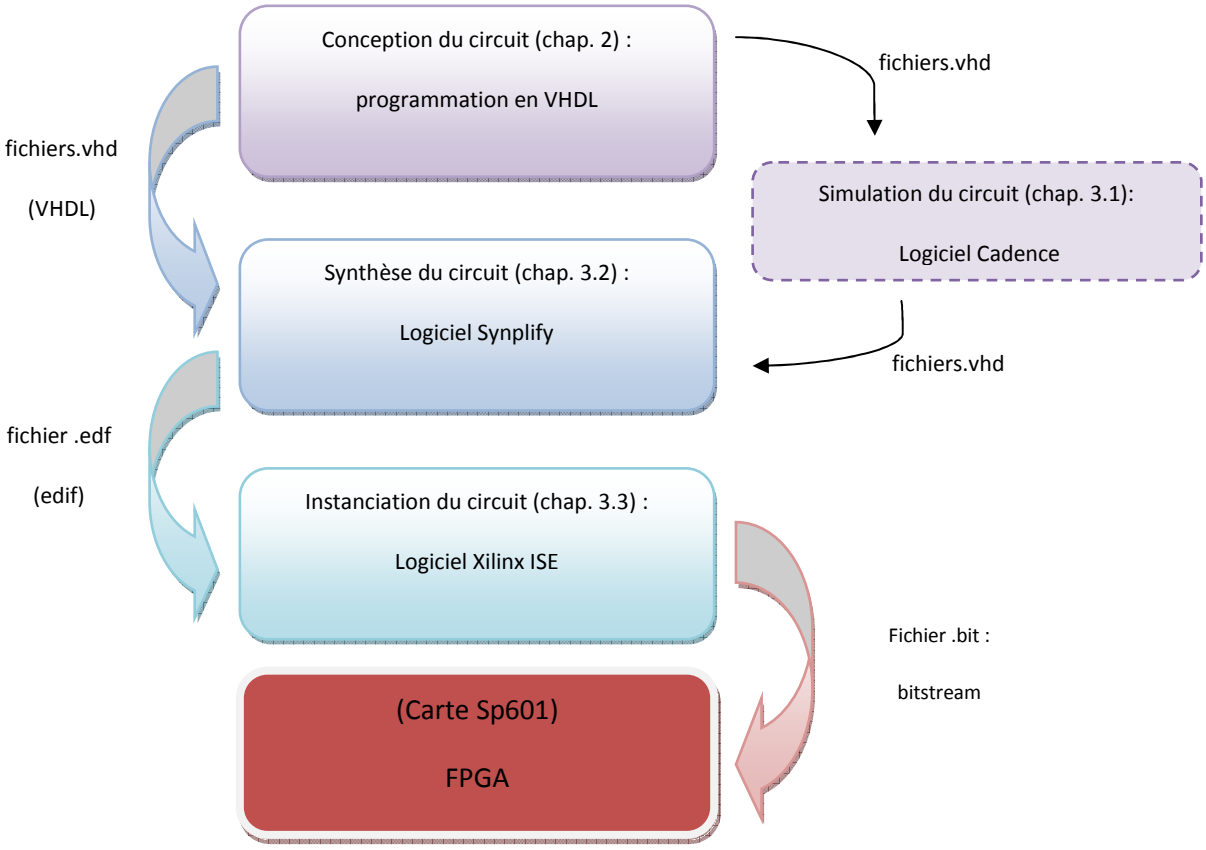


Figure 28 : Schéma synoptique du flot de conception

Ce flot de conception permet de Synthétiser les fichiers VHDL et de **générer** un fichier **edif**. Ce dernier est alors inséré dans **Xilinx ISE**. Cela permet d'éviter qu'ISE ne synthétise une nouvelle fois les fichiers (nous y reviendrons dans la suite de ce dossier).

Voici l'impression d'écran de la liste des fichiers utiles au logiciel synplify :

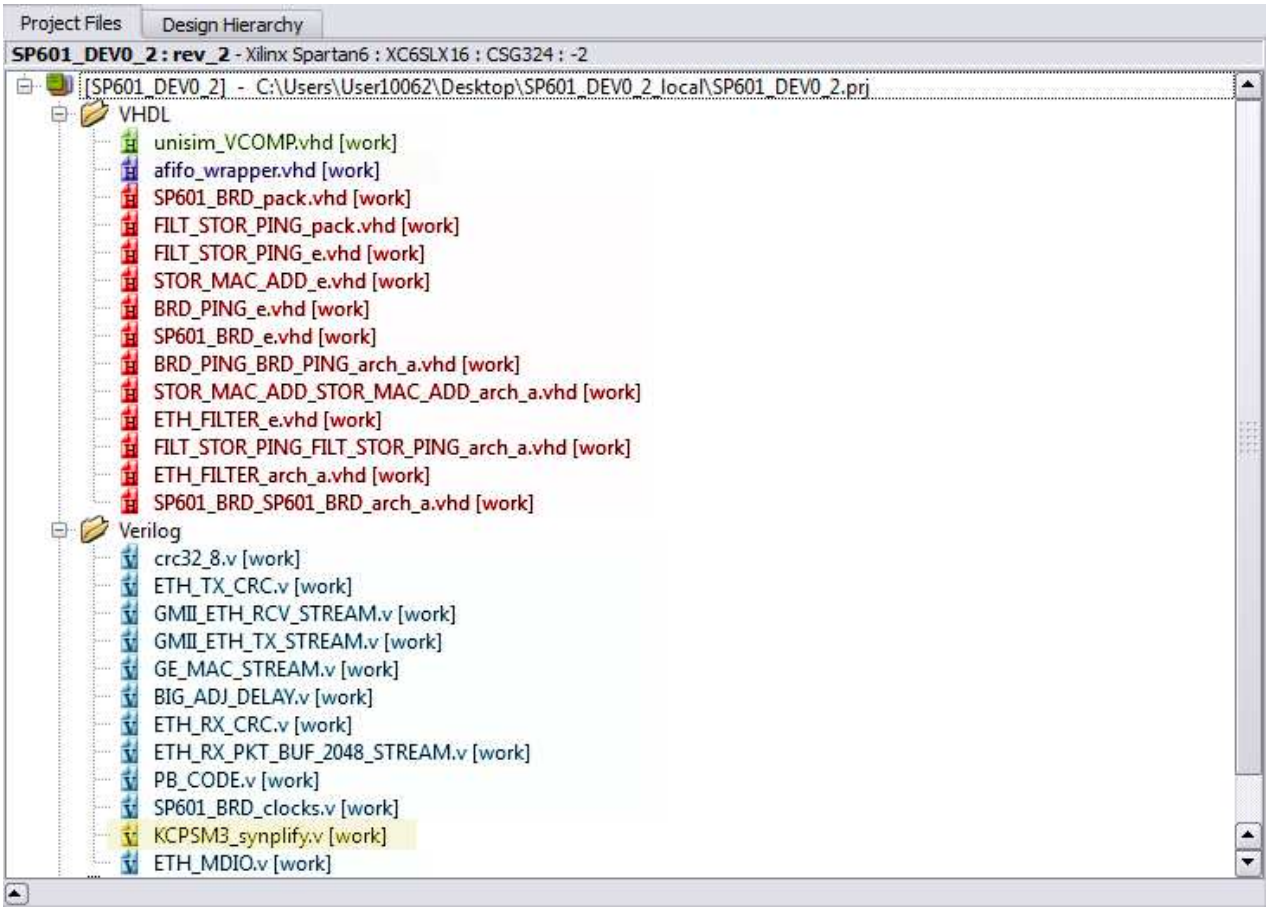


Figure 29 : Fichier de projet pour la synthèse

- Les fichiers en rouge et bleu ciel : Fichiers de projet, en langage VHDL et Verilog. D'une part les fichiers **Verilog**, appartenant à la société Xilinx et d'autre part les fichiers **VHDL** créés pour ce projet.
- Le fichier en vert : Librairie contenant toutes les **fonctions primitives logiques** comme des portes ET à 2 entrées, 3 entrées etc.
- Le fichier en jaune : Seconde librairie contenant des fonctions primitives logiques **spécifiques** aux FPGA de la société Xilinx.
- Le fichier en violet : **Fichier vide**. C'est en réalité une boite noire avec des entrées et des sorties. Il permet de faire comprendre au synthétiseur qu'il doit créer une entité vide. Elle sera **complétée** par le **générateur de fonction** de Xilinx ISE au moment de l'implémentation.

Un fois, les fichiers utiles réunis dans ce projet, on peut lancer la **synthèse**. Le logiciel va, avec des algorithmes, interpréter le code VHDL et Verilog puis le simplifier au besoin et enfin le synthétiser.

Une fois la synthèse réussie, Synplify génère un fichier de "**Port Map**" avec l'extension .edf.

Voici un exemple de fichier edif :

```
(cell LUT3_L (cellType GENERIC)
  (view PRIM (viewType NETLIST)
   (interface
    (port I0 (direction INPUT))
    (port I1 (direction INPUT))
    (port I2 (direction INPUT))
    (port LO (direction OUTPUT))
   )
  )
)
(cell LUT3 (cellType GENERIC)
  (view PRIM (viewType NETLIST)
   (interface
    (port I0 (direction INPUT))
    (port I1 (direction INPUT))
    (port I2 (direction INPUT))
    (port O (direction OUTPUT))
   )
  )
)
```

Figure 30 : Fichier edif

Dans ce fichier, on peut remarquer que les blocs sont déjà répartis en **Unités Arithmétiques et Logique** (LUT3 par exemple). Le travail de synthèse est donc bien terminé et le logiciel Xilinx ISE pourra implémenter ces données dans les LUT du FPGA et n'aura qu'à les **interconnecter**.

3.3 L'implémentation

Le logiciel ISE de Xilinx intègre différentes fonctionnalités comme la synthèse ou le **placement routage**. C'est cette dernière qui nous intéresse puisque synplify génère un fichier edif déjà synthétisé. ISE doit donc utiliser ce fichier pour placer les différentes informations dans les LUT du FPGA et les interconnecter. Grâce à un autre outil, on peut aussi créer des **probes** (signaux routés vers l'extérieur du FPGA) pour mesurer des signaux voulu. Voici un exemple simple de script :

```
# probe script file: probes.scr
probe script begin
probe add FILT_STOR_PING_INST/DAT_HANDCHECK_STOR_W -targetpins A16  -usedpin A16
probe add CLK_W -targetpins F9  -usedpin F9
probe script end
```

Figure 31 : Fichier de projet pour la synthèse

Dans ce script on indique par exemple au logiciel de **router** le signal CLK_W vers la pin F9 du FPGA ou la pin F9 du FPGA (on force vers F9).

Attention cependant car on ne peut pas router vers l'extérieur tous les signaux. En effet, comme le fichier est déjà synthétisé, certains peuvent avoir été **supprimés** ou **remplacés** par des équivalents. Dans le cas où l'on voudrait obtenir un signal habituellement supprimé par le synthétiseur, il faut forcer ce dernier dans le code VHDL grâce à une commande. Chaque synthétiseur à son propre langage pour cela.

A la fin de ce placement & routage, ISE **génère** un fichier "**.bit**" prêt à être transféré dans le FPGA.

3.4. Les mesures

Voici le banc de test monté pour les mesures (ou test bench) :

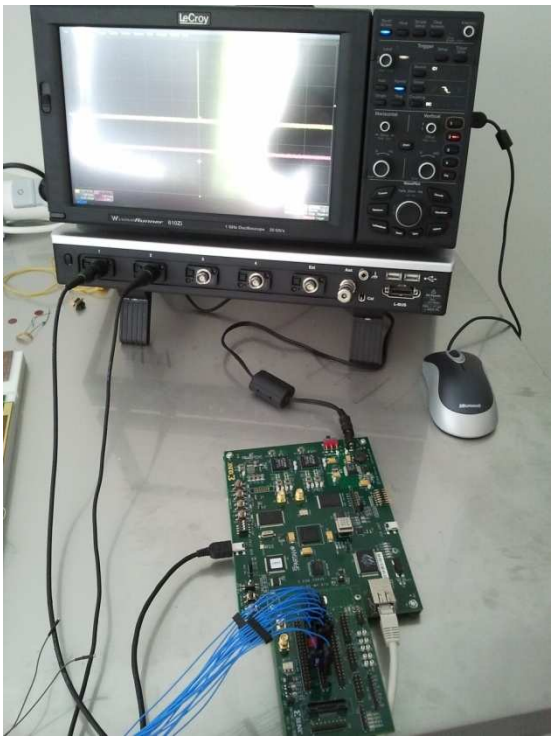


Figure 32 : Banc de test

Banc de test :

- Oscilloscope numérique/analogique : LeCroy, Waverunner 610 Zi
- Carte d'évaluation : Xilinx SP601
- PC avec une configuration linux
- Logiciels :

- Module de création de trame (programmé sous langage Python)

- Outil d'analyse de protocoles réseau : Wireshark

Carte d'évaluation :

- USB JTAG : connexion pour la programmation du FPGA depuis le PC
- FPGA
- MARVELL
- Mezzanine de mesure : permet de router des pins du FPGA vers une barrette de connecteur pour un meilleur accès
- Sondes d'oscilloscope numérique : permettent de visualiser des signaux numériques (sous forme de BUS par exemple)
- Câble Giga Ethernet

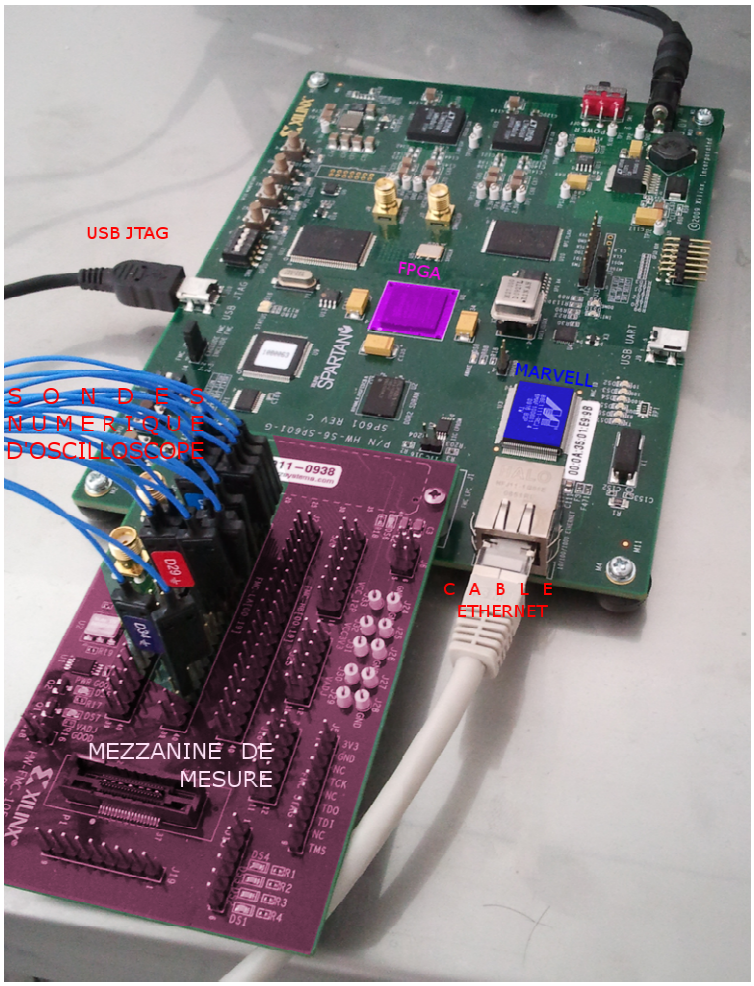


Figure 33 : Carte SP601

Une fois le FPGA programmé via l'interface USB JTAG de la carte, une trame de test est envoyée depuis l'ordinateur. La carte la réceptionne et effectue un PING. Certains signaux ont été routés vers l'extérieur et visualisés sur l'oscilloscope. Cette démarche permet de tester les blocs, les uns après les autres. Par exemple voici la mesure du signal en sortie de filtre (avant l'exécution du PING) :

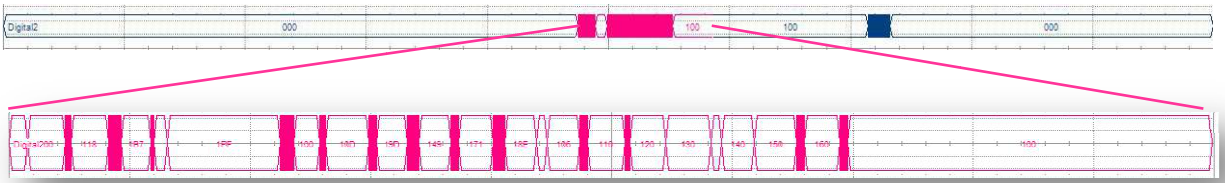


Figure 34 : Mesure de l'entête de la trame sur l'oscilloscope

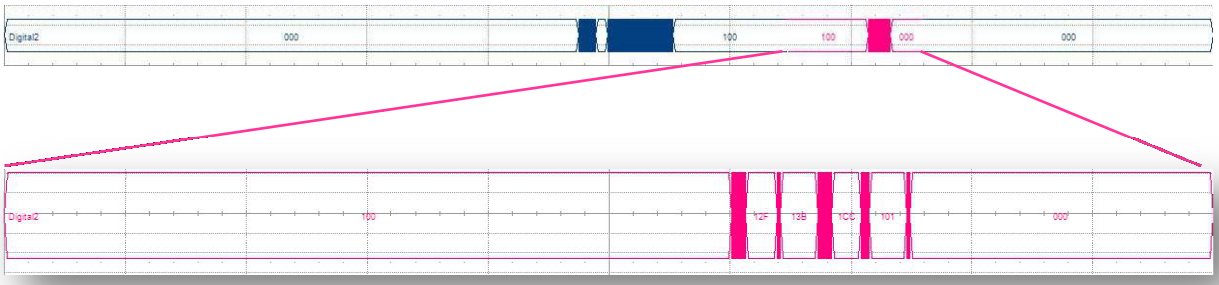


Figure 35 : Mesure de la fin de trame sur l'oscilloscope

On peut voir que toutes les informations sont complètes et que la carte a "compris" que cette trame **lui était bien adressée**. De plus le CRC a été validé, il n'y a donc **pas d'erreur de transmission** (Les conditions sont très favorables sur un banc de test : pas de parasite, pas de longueur câble excessive, pas de réseau...).

Ces mesures montrent que la trame a bien passé tous les blocs situés en amont du bloc PING et valide donc toute la partie réception de trame.

La partie d'émission est actuellement en phase de validation, voici la simulation cette dernière à l'entrée du Marvell :

Figure 36 : Trame en sortie du FPGA

Certains les éléments sont bien présents mis à part le CRC:

- Le préambule, les adresses et la taille.
- Le **bourrage**[3] : La taille **minimum** d'une trame Ethernet est donc atteinte.

Une recherche est donc en cours puisque le CRC doit normalement être calculé et intégré par un module Verilog. Cela pourrait donc provenir d'une mauvaise instanciation.

[3] Bourrage : Ajout de bit(s) de valeur « 0 » dans une trame pour la faire atteindre la taille minimum requis

IV. CONCLUSION

1. Conclusion sur le projet

Ce projet m'a permis de me confronter à une réelle recherche en laboratoire. J'ai pu mettre en application et approfondir les connaissances acquises au cours de mon cursus secondaire. En effet, je me suis formé, notamment, à utiliser différents **logiciels** comme Xilinx ISE, ModelSim , Synplify, Cadence SimVision, ...

De plus, j'ai acquis, une expérience non négligeable en **programmation** de circuit électronique, particulièrement au niveau de la **gestion** de gros projets mixtes (VHDL-Verilog) et dans l'**optimisation** de la programmation en VHDL. Ce projet m'a aussi permis de m'initier au langage **Verilog**.

N'ayant que côtoyé le monde de l'industrie dans mes précédentes expériences, ce stage m'a permis de découvrir le milieu de la recherche. Je retiendrai donc quelques axes principaux qui caractérisent la recherche en laboratoire : le travail en **autonomie**, une certaine **liberté** dans la conception de projet, une **diversité** dans les taches effectuées (j'ai pu faire de la simulation, de la synthèse, de l'implémentation et des tests sur banc d'essai), des **projets à long terme** et une recherche d'**innovation**.

2. Le futur

- Au niveau personnel :

Je souhaite, si j'en ai la possibilité, réaliser une petite étude des différentes versions de circuit en vérifiant tout d'abord les temps de propagations dans chacune d'entre elles puis les ressources utilisées (Place occupée dans le FPGA, ...).

- Au niveau du projet :

- Finaliser la validation de l'émission de données, à l'aide du PING. Effectuer une simulation et mesurer les résultats sur la carte.

- Utiliser un bloc mémoire interne au FPGA pour stocker toutes les informations. Ce dernier étant programmé par un générateur de fonction (Coregen).

-Créer une base de données des différentes versions existantes ainsi que des documents de références pour chacune d'entre elles. Elaborer un dossier technique pour le laboratoire.

ANNEXES

SOMMAIRE DES ANNEXES

TABLE DES ILLUSTRATIONS

SOURCES

http://cisco.goffinet.org/s1/principe_encapsulation

http://www.frameip.com/entete-ethernet/

http://fr.wikipedia.org/wiki/Ethernet

http://www.technologuepro.com/reseaux/Chapitre5-reseaux-locaux.htm

Manuels d'utilisateur des logiciels

Documentation technique du Marvell 88E1111 (la partie libre d'accès)

COMPTE RENDU DE REUNION DU 07/03/2011

Intervenants : chercheurs sur différents projets, susceptibles d'utiliser cette liaison Giga-Ethernet : CALICE, HARPO, GALO, ILM, CMS, TLB.

BUT : Le projet que je dois mettre en œuvre est une demande de CALICE. Le but est de créer cette liaison Giga-Ethernet pour l'appliquer à CALICE. Or plusieurs projets en ont besoin. Il est donc préférable d'élaborer un système "standard".

Projet CALICE : le but est de supprimer une carte LDA créée par un autre laboratoire (donc moins maitrisée), et peu pratique pour être mise dans des racks. De plus elle utilise des blocs IP (intellectual property) Xilinx qui sont obsolètes : GEMAC. Ces blocs doivent être changés pour utiliser des TEMAC.

Le choix du FPGA Spartan est fixé car DCC déjà en hardware. Pour créer ce lien, il faut un serialiseur. 4 possibilités sont donc envisageables : soit Marvell 88E1111 soit TLK 2201 avec TEMAC en software ou en hardware.

Le Marvell oblige de faire une liaison cuivre et le TLK2201 donne le choix entre cuivre et fibre.

DCC : Data Concentrator Card.

Projet HARPO : Projet basé actuellement sur des FPGA Virtex 5.

Projet GALO : Caméra donc fort débit et ne souhaite pas de switch.

projet ILM : Ethernet sécurisé entre automate avec une demande en débit pas forcément élevé ni de Ethernet en temps réel.

Projet CMS : TLB : Choix du FPGA 6 car problème sur Spartan 6 (avec les liaisons séries quand on utilise les IO juste à côté).

Démarche : Test sur banc de test avec la carte SP601 avec l'exemple fourni par Xilinx puis essais avec UDP puis mezzanine UDP et enfin TCP.

```
--FSM TEST TRAME ETHERNET :



architecture MARV_EMUL_arch of MARV_EMUL_e is

type state_value is (deflt, header_1, header_2, mac_dest_1, mac_dest_2, mac_dest_3,
mac_dest_4, mac_dest_5, mac_dest_6, mac_src_1, mac_src_2, mac_src_3, mac_src_4, mac_src_5,
mac_src_6, length_dat, dat1, dat2,dat3);
signal state : state_value :=  deflt;
signal cpt_header : std_logic_vector(2 downto 0);
signal data_out_tmp : std_logic_vector(7 downto 0);
signal out_cke, out_frm  : std_logic;
signal reg_send_frm : std_logic_vector(2 downto 0):="000";
signal strt_cpt_header  : std_logic;



begin

EDGEDETECT_p : process(clk)
        begin
                if(clk'event and clk='1') then
                        reg_send_frm (0) <= send_frm;
                        reg_send_frm (1) <= reg_send_frm (0);
                elsif(send_frm='1' and reg_send_frm(1)='0') then
                        reg_send_frm(2)<='1';
                else reg_send_frm(2)<='0';
                end if;
        end process;


TRAMEGEN_inst :         process (state,send_frm,clk, rst)
        begin

                if (rst = '1') then
                        data_out_tmp <= (others =>'0');
                        state <= deflt;
                        out_frm<='0';
                        out_cke<='0';

                elsif(clk'event and clk='1' and rst='0') then


                        case state is

                        when deflt =>
                                                        if(reg_send_frm(2)='0' ) then
                                                                data_out_tmp <= (others =>'0');
                                                                state <= deflt;
                                                                out_frm<='0';
                                                                out_cke<='0';
                                                                strt_cpt_header<='0';
                                                        else

                                                                data_out_tmp <= x"00";
                                                                out_frm<='0';
                                                                out_cke<='0';
                                                                strt_cpt_header<='1';
                                                                state <= header_1;
                                                        end if;
                        when header_1 =>
                                                        out_frm<='1';
                                                        out_cke<='1';
                                                        if(cpt_header="110") then
                                                        state <= header_2;
                                                        strt_cpt_header<='0';
                                                        data_out_tmp <= x"55";
                                                        else
                                                        state <= header_1;
                                                        data_out_tmp <= x"55";
                                                        strt_cpt_header<='1';
                                                        end if;


                        when header_2 =>
                                                        data_out_tmp <= x"D5";
```

```
                                                state <= mac_dest_1;
                                                out_frm<='1';
                                                out_cke<='1';


                        when mac_dest_1 =>
                                                data_out_tmp <= x"64";-- source MAC
address (carte gigabit ethernet : 64-31-50-30-48-E1)

                                                state <= mac_dest_2;
                                                out_frm<='1';
                                                out_cke<='1';

                        when mac_dest_2 =>
                                                data_out_tmp <= x"31";

                                                state <= mac_dest_3;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_dest_3 =>
                                                data_out_tmp <= x"50";

                                                state <= mac_dest_4;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_dest_4 =>
                                                data_out_tmp <= x"30";

                                                state <= mac_dest_5;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_dest_5 =>
                                                data_out_tmp <= x"48";

                                                state <= mac_dest_6;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_dest_6 =>
                                                data_out_tmp <= x"E1";

                                                state <= mac_src_1;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_1 =>
                                                data_out_tmp <= x"00";-- destinataire MAC
address (carte SP601 : 00_18_B7_FF_FF_FF)

                                                state <= mac_src_2;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_2 =>
                                                data_out_tmp <= x"18";

                                                state <= mac_src_3;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_3 =>
                                                data_out_tmp <= x"B7";

                                                state <= mac_src_4;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_4 =>
                                                data_out_tmp <= x"Ff";

                                                state <= mac_src_5;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_5 =>
                                                data_out_tmp <= x"FF";

                                                state <= mac_src_6;
                                                out_frm<='1';
                                                out_cke<='1';
                        when mac_src_6 =>
                                                data_out_tmp <= x"FF";
```

```
                                                state <= length_dat;
                                                out_frm<='1';
                                                out_cke<='1';
                        when length_dat =>
                                                data_out_tmp <= x"03";-- lenght of the
frame

                                                state <= dat1;
                                                out_frm<='1';
                                                out_cke<='1';
                        when dat1 =>
                                                data_out_tmp <= x"55";
                                                state <= dat2;
                                                out_frm<='1';
                                                out_cke<='1';
                        when dat2 =>
                                                data_out_tmp <= x"DD";
                                                state <= dat3;
                                                out_frm<='1';
                                                out_cke<='1';
                        when dat3 =>
                                                data_out_tmp <= x"18";
                                                state <= deflt;
                                                out_frm<='1';
                                                out_cke<='1';

                        when others => state <= deflt;
                        end case;
                else null;
                end if;

        end process;

data_out_inst :         data_out<= out_cke & out_frm & data_out_tmp;


        CPT_HEADER_P : process(clk, strt_cpt_header)
        begin
                if(clk'event and clk='1') then
                        if(strt_cpt_header='1') then
                        cpt_header<=cpt_header+1;
                        else
                        cpt_header<="000";
                        end if;
                else
                cpt_header<=cpt_header;
                end if;
        end process CPT_HEADER_P;

end MARV_EMUL_arch;
```