heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Hes·so

Filière Télécommunication

Travail de diplôme de Bachelor 2009

Conception d'une unité de stockage avec NetFPGA

HEIG-VD
Yverdon-les-Bains
vendredi 17 juillet 2009

Etudiant: Evgenij Sviridovitch
Professeur: Dr. Stephan Robert
Département: TIC

LE RESUME

Pour résoudre les problèmes de stockage, on propose dans ce document d'utiliser une plateforme NetFPGA pour implémenter un Network Attached Storage. Le but est d'étudier le principe de stockage avec les disques directement attachés au réseau (NAS) ainsi qu'étudier la plateforme NetFPGA et implémenter en matériel un dispositif du NAS fonctionnel.

CAHIER DES CHARGES

Les NAS (Network Attached Storage) ne sont pas de nouveaux appareils. Il est cependant intéressant de savoir comment ils fonctionnent. De plus qu'est-ce qui défini leurs prix ? Pourquoi certaines unités sont chères et d'autres bon marché ? Le but ici est de construire une unité NAS simple en utilisant une plateforme de développement dédiée. Les outils de développement ainsi que la plateforme sont données dans le cadre du projet. L'étudiant bénéficie d'une certaine flexibilité quant aux choix de l'architecture. L'unité de stockage peut par exemple être un petit PC muni d'un disque dur. Le choix du protocole de communication est aussi laissé ouvert mais il faut que l'étudiant assume ses choix et qu'il les justifie. Il s'agira essentiellement de démontrer que le système est performant et de trouver des techniques pour le rendre performant. Il faudra en tout cas implémenter un système qui doit être fonctionnel à la fin du projet.

TABLES DES MATIERES

INTRODUCTION

La quantité de données informatiques augmente chaque jour. Pour une entreprise, ces données ont une valeur importante et c'est évident qu'il faut les sauvegarder. Lorsque le volume des données croît, le temps qu'il faut pour les transmettre et les stocker augmente aussi mais de manière non linéaire. Il est donc nécessaire d'augmenter les performances des unités de stockage.

Ce travail vise à étudier le (NAS) Network Attached Storage et son fonctionnement ainsi qu'à étudier la plateforme netFPGA, laquelle nous permettra d'implémenter un serveur de sauvegarde (backup). Le sujet traité fait intervenir plusieurs niveaux de Télécommunications. Le choix de la plateforme netFPGA avec ses outils de développement a été rendu nécessaire par la complexité du problème, en termes de performances de stockage

Ce projet est divisé en deux parties de deux chapitres chacun. Le chapitre cinq contient l'analyse du travail effectué avec une conclusion. Dans la première partie (Chapitres 1 et 2), on va parler de différents types de stockage sur le réseau et des avantages et inconvénients des différentes solutions existantes. On arrivera à la conclusion qu'une solution hardware est préférable à une solution purement software. Cette partie est entièrement consacrée aux études théoriques.

La deuxième partie (Chapitres 3 et 4) du projet est consacrée à l'étude du fonctionnement de la plateforme netFPGA ainsi que de sa programmation. Cette partie comprend également l'étude des protocoles à implémenter via leur RFC ainsi que le test de la plateforme NetFPGA et l'analyse des résultats obtenus.

Le développement suit une approche en cycles, avec d'abord une étude théorique, suivie par un développement de matériel.

1. FAMILIARISATION AVEC LES SYSTEMES NAS ET SAN

1.1 INTRODUCTION A LA PROBLEMATIQUE[1]

L'évolution de la masse des données à garder a inévitablement un impact sur la capacité des disques durs. Le premier changement d'échelle intervient à ce niveau. En effet, à force d'accroître la capacité des disques durs, les solutions de stockage externe, chargées de consolider les informations dans l'entreprise, croulent littéralement sous les données.

Au début des années quatre-vingt-dix, les constructeurs vendaient des solutions à même de supporter plusieurs centaines de giga-octets de données. Maintenant, l'unité de mesure s'exprime en téraoctet, soit 1000 Go. Cette évolution est tellement forte que plusieurs millions de téraoctets d'espace de stockage (pour les unités externes – sans prendre en compte les disques durs des ordinateurs, des serveurs, etc.) sont vendus chaque année. Pour rappel, il ne s'en vendait que quelques milliers au début des années quatre-vingt-dix.

Toujours indissociable de l'explosion des données : l'adoption d'Internet. Ce média de communication ne cesse de séduire les utilisateurs. Les courbes de croissance d'Internet reposent sur un scénario simple : en période de croisière, on estime que le nombre d'internautes progresse de 10% par mois. Cependant, même si les analystes savent bien que ce taux de croissance va décroitre, ils ne sont pas encore à même de déterminer le moment de l'inflexion. Le cabinet ComScore[2] annonce qu'au cours du mois de décembre 2008, la barre symbolique du milliard d'utilisateurs d'Internet a été atteinte. Son étude prend en compte les personnes de plus de 15 ans surfant sur leur lieu de travail ou à domicile. Seulement, avec l'avènement d'Internet, les utilisateurs sont encore plus critiques qu'auparavant. Ils souhaitent disposer immédiatement de l'information.

L'utilisateur manipule de plus en plus de données. Les logiciels bureautiques toujours plus sophistiqués sont de plus en plus gourmands en capacité. Internet et messageries se généralisent et échangent des objets multimédias de plus en plus volumineux. Il n'est pas rare de voir circuler des documents dépassant la centaine de mégaoctets !

[1] - Bibliographie [1]
[2] - http://www.comscore.com/

L'utilisateur final vit dans l'inquiétude de voir ses quotas de volumes dépassés. Chaque jour il reçoit, de la part des serveurs, des avertissements lui signifiant le dépassement de l'espace de stockage qui lui a été alloué. Souvent, le réseau se sature, induisant des temps d'attente de plusieurs dizaines de secondes.

1.2 DES PERTES DE PRODUCTIVITÉ LATENTES

L'entreprise a-t-elle vraiment conscience des pertes de productivité engendrées par l'indisponibilité ou le ralentissement de son outil informatique ? Sait-on que, dans le monde ouvert, les pertes de productivité dues à l'indisponibilité ou au mauvais dimensionnement d'un système de stockage peuvent représenter jusqu'à 60% des coûts opérationnels, qui eux-mêmes représentent jusqu'à quatre fois les coûts d'acquisition du matériel !

Traditionnellement, l'entreprise utilisait la notion de téraoctets par administrateur pour quantifier les coûts d'administration d'un infocentre. Aujourd'hui, avec le déploiement des réseaux, des applications et des serveurs du monde ouvert, cette notion est dépassée et l'administrateur se heurte à de nouveaux problèmes.

1.3 SAUVEGARDER LES DONNÉES DE PLUS EN PLUS VITE[1]

Examinons le cas d'une entreprise qui dispose de plusieurs serveurs d'applications répartis sur ses réseaux et qui doit rester opérationnelle de six heures du matin à huit heures du soir. Après huit heures, elle procède à des opérations de *batch* qui peuvent durer jusqu'à trois heures du matin. Il lui reste une fenêtre de trois heures pour exécuter la sauvegarde journalière de ses données. La durée de la sauvegarde se heurte à deux contraintes :

- Les données passent par le réseau d'entreprise où elles sont freinées lors du passage dans les couches logicielles de communication. Même à l'heure de Gigabit-Ethernet, il est difficile de sauvegarder des dizaines de giga-octets à l'heure ;

[1] - Bibliographie [1]

- La quantité de données à sauvegarder augmente continuellement. Le nombre de serveurs d'applications livrés dans le monde augmente de 20% par an et les entreprises s'intéressent de plus en plus aux détails de l'information. Cela leur permet de mieux connaître les besoins de leurs clients mais cela entraînera surtout, selon les analystes, une inflation du stockage de plus de 500% ! Bientôt, certaines entreprises, par exemple dans le secteur de la grande distribution, auront besoin de sauvegarder des centaines de giga-octets, voire des téraoctets !

1.4 ADMINISTRER DES RESSOURCES HÉTÉROGÈNES

La situation se complique quand on observe que les ressources de stockage attachées à chaque serveur sont dépendantes de la marque du serveur. En particulier, elles sont administrées à travers le serveur. Comme une entreprise dispose en moyenne de deux à trois types de plates-formes (Sun, HP, Bull etc.) et d'environnements d'exploitation (AIX, NT, etc.) l'administrateur doit s'appuyer sur plusieurs standards d'administration pour gérer des ressources de stockage hétérogènes. Devant une telle complexité, les compétences de l'administrateur trouvent vite leurs limites. Administrer des ressources de stockages dispersées et hétérogènes peut coûter trois à quatre fois plus cher qu'administrer un ensemble de stockage homogène partagé entre plusieurs serveurs hétérogènes !

Un réseau complexe soumis à de nombreuses contraintes est délicat à administrer (Figure 1.1)

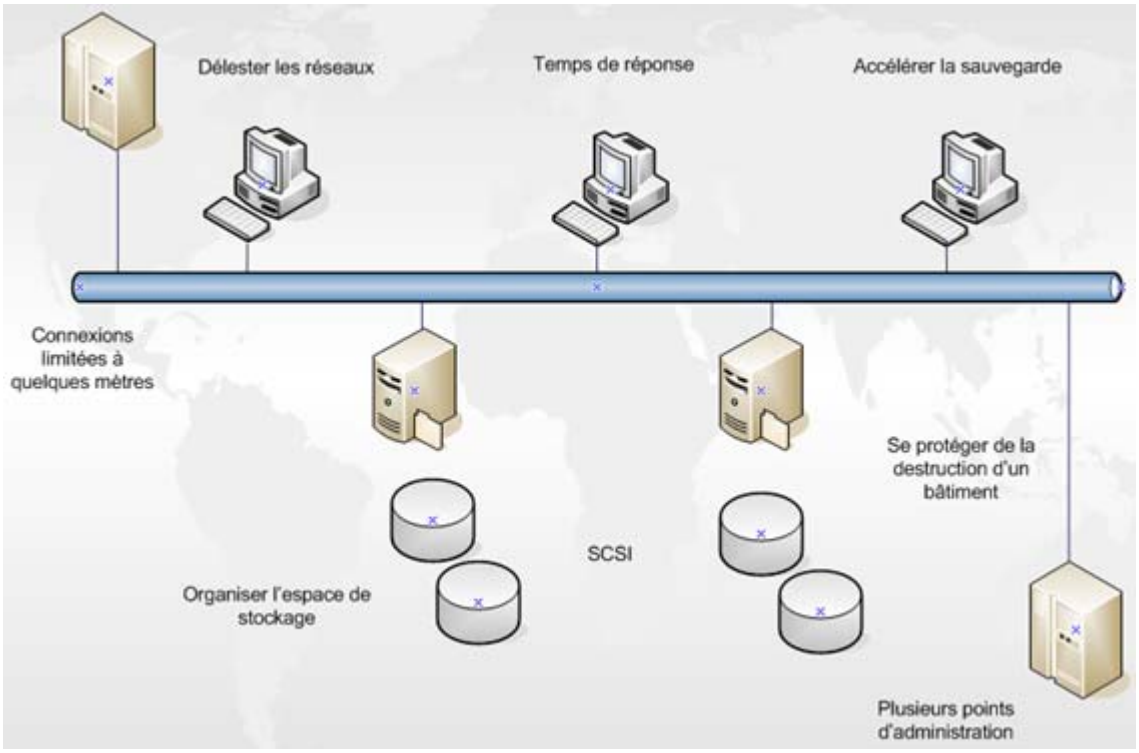


Figure 1.1 – Les réseaux d'entreprise se sont considérablement complexifiés et doivent faire face, parallèlement, à la croissance exponentielle des données[1]

1.5 LE STOCKAGE EN RÉSEAU

1.5.1 LA CONCEPTION DE STOCKAGE SAN

La mise en réseau du stockage est un mouvement irréversible. Le marché du stockage en réseau représente près de la moitié du stockage externe estimé alors à quarante milliards de dollars. Il existe deux conceptions du stockage en réseau : le NAS et le SAN. Faisons la description du SAN (Storage Area Network) et ensuite du NAS (Network Attached Storage).

Storage Area Network - est un réseau spécialisé permettant de mutualiser des ressources de stockage (Figure 1.2 ci-dessous).

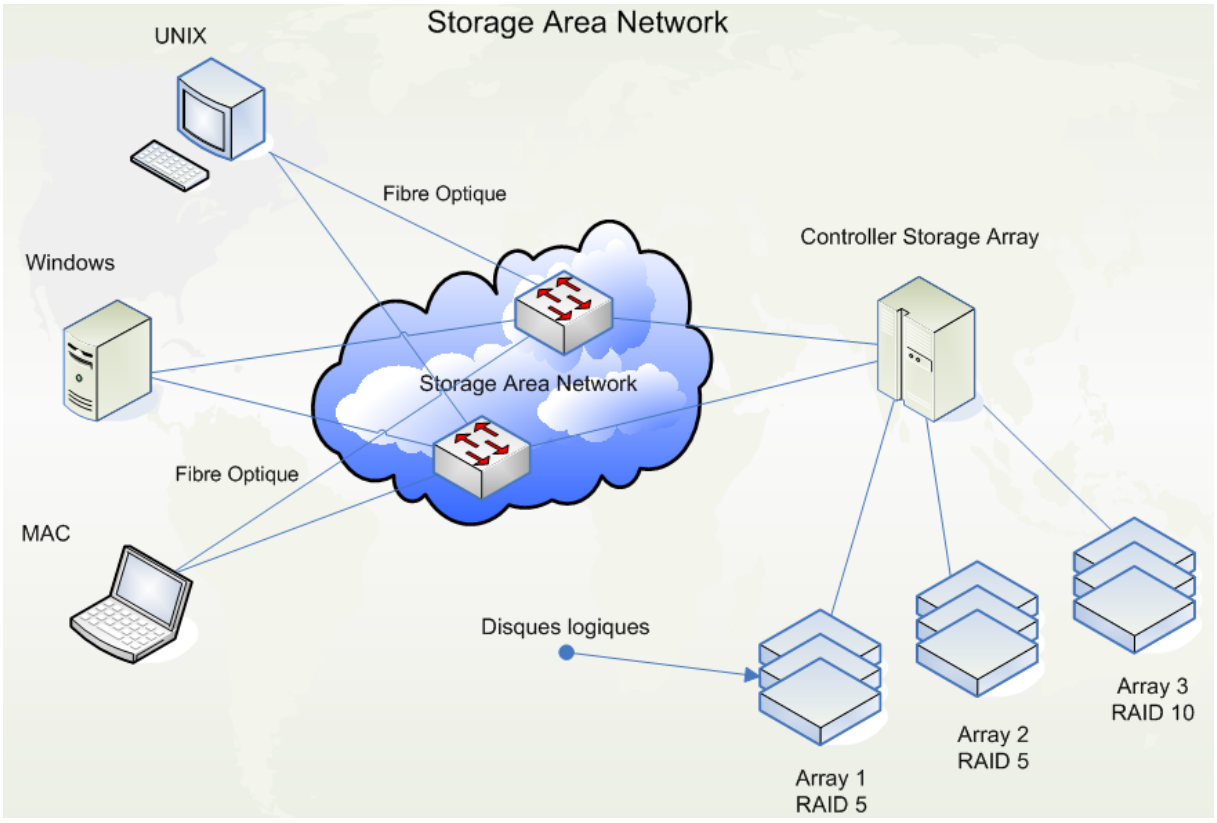


Figure 1.2 - Simple architecture SAN

Le terme SAN désigne une nouvelle architecture de stockage selon laquelle les systèmes de stockage sont attachés à un réseau rapide entièrement dédié au stockage. Il s'agit bien d'un nouveau réseau qui vient en complément des réseaux de communication existants. Les serveurs d'applications qui accèdent aux systèmes de stockage sont également connectés au réseau rapide et échangent les données en protocole de type entrée/sortie. La technologie de choix pour ce réseau rapide est le Fibre Optique. L'administration est dédiée à l'ensemble des systèmes de stockage et à l'infrastructure de commutation Fibre Channel.

L'architecture SAN s'impose rapidement. Le concept de SAN apporte une vraie rupture technologique dans la façon d'appréhender le stockage d'entreprise. Avec le SAN, le stockage dispose enfin d'un réseau adapté aux besoins d'accès et d'échange de l'information.

1.5.2 LA CONCEPTION DE STOCKAGE NAS

Le NAS est au sens propre une solution de stockage réseau (Figure 1.3).

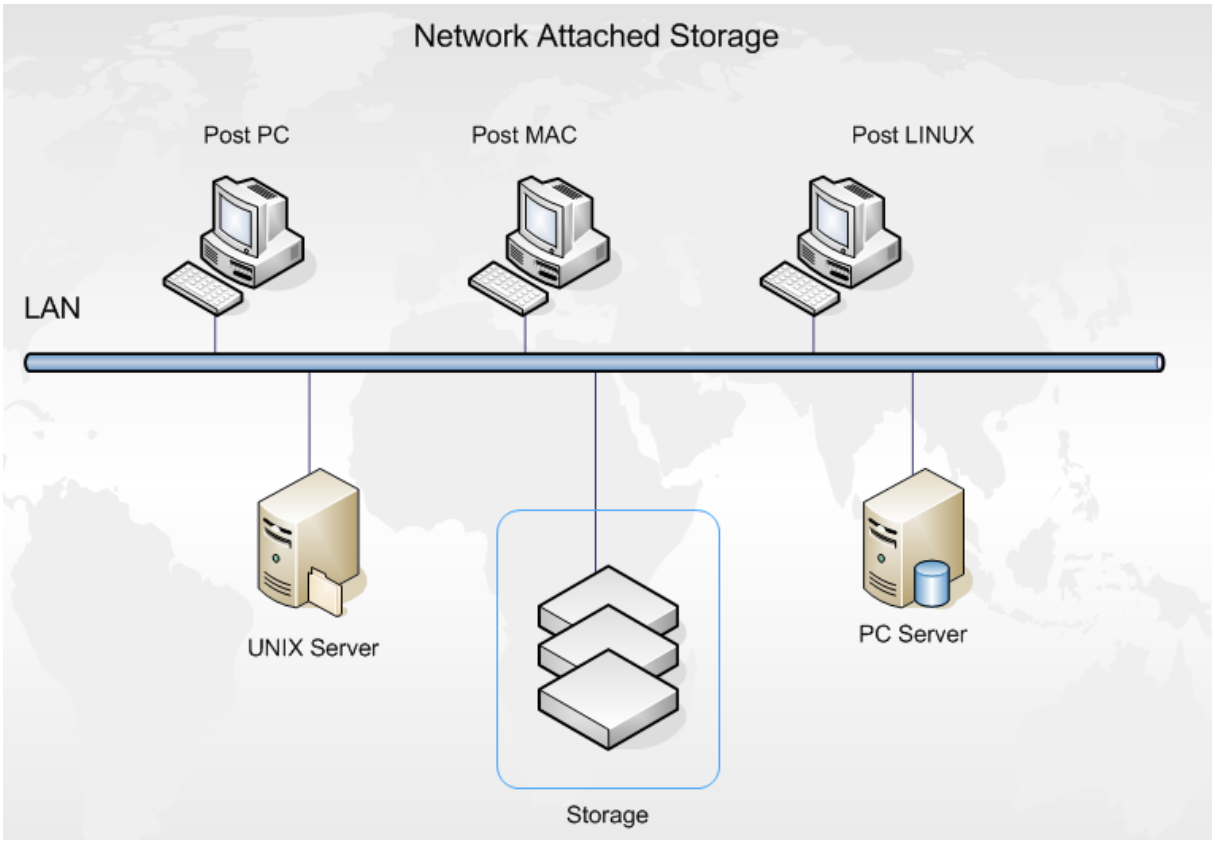


Figure 1.3 - Simple architecture NAS

Autrement dit, il s'agit de manière très schématique d'une sorte de disque dur externe que l'on ne connecte pas au port USB/FireWire de sa machine, mais directement sur un port réseau RJ45. Cette plateforme (disque dur) dispose son propre adresse réseau. Les utilisateurs accèdent directement aux fichiers qui y sont enregistrés ou passent par le serveur d'application. Ce dernier se charge de la redirection de requêtes. Les plates-formes fonctionnent sur le même principe qu'un serveur de réseau traditionnel. Elles offrent l'avantage d'une réelle autonomie fonctionnelle, par opposition au stockage habituel qui réside physiquement à l'intérieur du serveur de réseau ou au réseau SAN qui nécessite la mise en place d'une infrastructure spécifique. Concrètement, une plate-forme NAS est vue par le serveur comme un poste client parmi d'autres, ce qui facilite considérablement son installation. En effet, pour que celle-ci soit reconnue sur le réseau local et donc devienne accessible, il suffit de lui affecter une adresse IP. Cette tâche revient à

l'administrateur réseau, lequel maîtrise le plan d'adressage. En effet, les réseaux IP disposent de deux modes d'adressage : l'adressage fixe ou l'adressage dynamique (via un serveur DHCP). Dans le premier cas, les équipements se voient affecter une adresse IP fixe. Elle est invariable. Seulement, cette approche nécessite un réservoir d'adresses suffisant pour pouvoir connecter tous les équipements. Afin de gérer la pénurie, les "gourous des réseaux locaux" ont mis au point le mécanisme DHCP. Ce dernier offre une grande souplesse dans la gestion des adresses IP, car outre la possibilité d'affecter des adresses de manière dynamique, DHCP dispose aussi d'une option lui permettant d'affecter des adresses de manière fixe. Si le réseau local est supervisé par un service DHCP, il faut absolument affecter une adresse fixe à un serveur de fichier NAS. Ces derniers offrent également la possibilité de multiplier simplement la capacité de stockage en lui ajoutant des disques durs, voire d'adjoindre une seconde baie de disques à la première. Elle sera alors considérée comme faisant partie intégrante de la première. Si jamais cela ne suffit pas, il est possible d'y ajouter un autre système de fichiers indépendant. À cela s'ajoute un second avantage : la rapidité. En rendant l'accès aux données et son administration autonomes par rapport au serveur du réseau, les applications et les fichiers peuvent être disponibles aux utilisateurs plus rapidement. Pourquoi ? Parce qu'elles ne se disputent pas les ressources processeur du serveur, par ailleurs constamment occupé à d'autres tâches de gestion du réseau.

Une solution de NAS regroupe plusieurs éléments. Outre la batterie de disques durs et une architecture matérielle généralement comparable à un serveur renforcé (processeur, mémoire, alimentation redondante…), elle inclut son propre système de gestion de fichiers, un logiciel spécifiquement conçu pour le stockage distribué (déploiement, gestion, administration) et, le plus souvent, un sous-système RAID pour la sécurisation des fichiers en temps réel.

1.6 AVANTAGE DU STOCKAGE NAS

Faisons ce parcours.

- **Facilité d'installation**

Nous pouvons ajouter des serveurs NAS à notre réseau local en quelques minutes, sans avoir à immobiliser ce dernier. Ces serveurs sont particulièrement adaptés aux applications qui impliquent de nombreux accès en lecture/écriture.

- **Allégement de notre serveur réseau**

Les serveurs NAS contribuent à accroître les capacités de stockage "à la volée", ce qui nous permet de rediriger le trafic réseau et évite d'avoir à ajouter des nœuds réseau supplémentaires. Les responsables d'entreprise peuvent délester le serveur réseau des tâches de services de fichiers qui requièrent une largeur de bande importante. Le temps de latence est alors diminué et le risque de perturbation des tâches cruciales, telles que la gestion des applications ou la messagerie électronique, est réduit. La souplesse du stockage NAS permet d'ajouter des capacités de stockage où nécessaire, y compris sur des sites distants. Enfin, il permet d'effectuer des sauvegardes sans que les performances du serveur réseau s'en trouvent affectées.

- **Simplification du partage de données**

Les réseaux modernes sont des environnements hétérogènes. Or, le stockage NAS nous permet de nous connecter à plusieurs systèmes d'exploitation et de partager des données entre des clients et des serveurs disparates. Pour faciliter ce partage de données entre plates-formes, le stockage NAS prend en charge à la fois le protocole NFS (Network File System) pour les systèmes UNIX et le protocole CIFS (Common Internet File System) pour les systèmes Microsoft.

1.7 AVANTAGE DU RÉSEAU SAN

- **Consolidation des informations**

Principal avantage d'un réseau SAN: il permet de consolider une grande quantité d'informations au sein d'un réseau de stockage centralisé. Il connecte l'ensemble des ressources de stockage et soulage le trafic réseau associé à l'accès à ces ressources sur un réseau distinct. Cela se traduit par une réduction du temps de latence et une utilisation plus efficace des ressources.

- **Accélération de l'extraction des données**

La technologie Fibre Channel sur laquelle repose le réseau SAN utilise une boucle arbitrée qui offre des vitesses de transfert de données réelles de 100 Mbps. Si l'on compare ce débit à celui qu'offre aujourd'hui la technologie SCSI, soit entre 40 et 80 Mbps, l'écart est considérable. Toutefois, une nouvelle technologie SCSI est actuellement en cours de développement. Elle promet d'amener les vitesses de transfert SCSI à un niveau proche de celui qu'offre aujourd'hui la technologie Fibre Channel. Les réseaux SAN peuvent également prendre en charge un nombre quasiment illimité de matériels, si l'entreprise est prête à investir dans l'infrastructure (serveurs, multiplexeurs, passerelles et unités de stockage).

- **Simplification des sauvegardes et restaurations**

Les réseaux SAN facilitent les opérations de sauvegarde et de reprise après incident. Les données peuvent ainsi être mises en miroir sur un site distant en vue d'une reprise transparente après incident, ou être sauvegardées rapidement sur un autre site sans que cela n'affecte les performances réseau. Un réseau SAN permet de sauvegarder plusieurs gigaoctets de données en quelques heures seulement. De plus, il prend en charge un large éventail de techniques réseaux (déroutement, clusterisation, reprise à chaud, mise en miroir et réplication, par exemple). Ces techniques assurent une protection contre la perte de données, et améliorent la disponibilité des informations.

- **Évolutivité exceptionnelle**

Avec son évolutivité intrinsèque quasiment illimitée, le réseau SAN constitue un choix idéal pour les réseaux qui connaissent une croissance rapide, ou qui ont besoin d'augmenter leurs capacités de stockage de façon sporadique. Les outils de repartitionnement et de gestion permettent aux administrateurs réseau de réallouer l'espace de stockage entre les serveurs en repartitionnant simplement le réseau SAN. Ce processus consiste à allouer un certain espace de stockage à un serveur réseau au lieu de connecter directement cet espace au serveur réseau.

Cependant, les réseaux SAN ont leurs limites. Le prix d'entrée est élevé, un réseau SAN de taille moyenne ne coûtant pas moins de 283 000 euros. De plus, le manque

de standardisation pose d'énormes problèmes d'interopérabilité que les fournisseurs commencent tout juste à résoudre.
Présentons les deux tableaux comparatifs ci-dessous:

Tableau 1.1 – La différence entre SAN et NAS

	NAS	SAN
Réseau	Réseau d'entreprise existant (Ethernet, FDDI, etc)	Réseau dédié et spécialisé (Fibre Channel)
Fonction des unités de stockage	Serveur de fichiers Gestion multiprotocole	Serveur de ressources de stockage Aide à la protection, au partage et au mouvement des données
Protocole d'échange	Type message (TCP/IP)	Type d'éntrée/sortie (SCSI, etc.)
Bande passante	Directement liée au réseau local mis en place: Ethernet haut débit (100 Mbit/s, 1 Gbit/s...), FDDI (100 Mbit/s) ou encore ATM (155/622 Mbit/s)	Dépend directement des possibilités du Fibre Channel (10 Gbit/s) et de l'architecture réseau mise en place (Fabric, FC-AL)
Administration de stockage	Le stockage est administré via le serveur	L'administration est directe, incluant toute l'infrastructure SAN

Tableau 1.2 – Avantages et inconvénients

	NAS	SAN
Disponibilité	Données accessbiles en permanence	Données accessibles en permanance par n'import quel serveur du réseau
Intéropérabilité	Système support plusieurs OS	Partage de données en environnement hétérogènes
Fiabilité	Architecture interne équipée de dispositifs rédondants	Architecture interne conçu contre les risques de panne totale ou partielle
Evolutivité	Capacité pouvant atteindre plusieurs téra-octets	Capacité pouvant atteindre plusieurs téra-octets
Simplicité	Mis en œuvre rapide	Conception et installation nécessitant souvent la présence d'un expert
Administration	Centralisation sur le poste de l'administrateur	Centralisation sur le poste de l'administrateur
Sécurité	Accès par mot de passe	Réplication locale ou distante
Performance	Vitesse suivant le débit du réseau local en place	Risque de saturation faible (débit de 200 Mo/s)

Nous avons vu la différence entre deux solutions de stockage sur le réseau, et bien chacune a ses avantages et inconvénients. Nous ne pouvons pas dire qu'on préfère un type de stockage à un autre. Tout dépend de vos besoins.

1.8 ETUDE DU MARCHÉ. SYSTÈMES EXISTANTS

Il existe le grand choix de système de stockage NAS sur le marché quelque soit le type de stockage soit des fichiers media soit le backup. Dans cette partie nous étudierons les différentes solutions.

Bien qu'il existe des solutions presque gratuites basées sur des PC recyclés ou des serveurs existants, les serveurs de fichiers dédiés ou NAS rencontrent un succès croissant, essentiellement dû à l'effondrement de leur prix.

L'offre comprend, aujourd'hui, des produits très disparates ciblant aussi bien les particuliers que les entreprises. Presque tous reposent sur les protocoles de partage de fichiers des mondes Unix et Windows (NFS et CIFS). En entrée de gamme, de simples boîtiers intégrant un ou deux disques et une prise Ethernet sont proposés à partir de 200 euros, par des constructeurs comme l'américain Iomega ou le taïwanais D-Link (les produits génériques coûtent encore moins cher). Malgré leur interface gigabit Ethernet, leurs performances sont pénalisées par un processeur et une mémoire basiques, ce qui en limite l'usage à quelques utilisateurs simultanés. Une architecture Raid 1 les rend éventuellement tolérants aux pannes mais le remplacement d'un disque nécessite l'arrêt de l'exploitation.

Les serveurs NAS vraiment conçus pour les petites entreprises sont basés soit sur un système Windows, soit sur Linux. Les seconds, dont le prix se situe entre 700 et 2 000 euros, sont généralement les moins chers mais intègrent déjà le hot swap. Cette fonction, naguère réservée aux grands comptes, permet de remplacer à chaud un disque déficient, donc sans aucune interruption de service.

Les serveurs de fichiers facturés entre 1 000 et 5 000 euros sont en majorité basés sur Windows Storage Server (WSS). Le matériel est alors proche de celui d'un PC, à tel point qu'il peut, en plus du service de fichiers, exécuter un moteur de base de données SQL ou un antivirus. Par rapport au Windows classique, WSS ajoute des fonctions liées au stockage, comme la sauvegarde automatique, l'élimination des doublons, l'indexation ou la réplication sur un serveur distant.

Ces produits se montrent particulièrement évolutifs, avec une capacité pouvant être étendue à plusieurs téraoctets (To). Ils peuvent même intégrer un système de sauvegarde à bandes. Autre avantage, la gestion des droits s'appuie sur l'Active Directory[1]. Dans le tableau 1.3 ci-après on donne quelques exemples de servers de stockage avec leur prix et la capacité de stockage.

Tableau 1.3 – Quelques exemples de serveurs de stockage[2]

Produit	Constructeur	Capacité	Interface	Prix
Produits grand public (système propriétaire)				
DNS-323	D-Link	250 Go (Raid 1)	Gigabit Ethernet	300 €
SC 101 T	Netgear	Selon disques (Raid 1)	Gigabit Ethernet	300 €
StorCenter	Iomega	250 Go à 1 To (Raid 0 ou 1)	Gigabit Ethernet (version Wi-Fi 802.11g)	200 à 400 € (700 € pour la version Wi-Fi)
Produits entreprises (système propriétaire à noyau Linux)				
Ready Nas	Infrant Technologies (racheté par Netgear)	1 à 3 To (Raid 1 ou 5)	Gigabit Ethernet	1400 à 3800 €
StorCenter Pro 150 D	Iomega	1 ou 2 To (Raid 1 ou 5)	Gigabit Ethernet	795 ou 1150 €
Produits entreprises (système Windows Storage Server)				
Dell PowerVault (basé sur Dell 840)	Dell	500 Go à 2 To (Raid 1 à 5)	Gigabit Ethernet	A partir de 790 €
HP Proliant Storage Server	HP	1 à 2 To (Raid 1 à 5)	Gigabit Ethernet	1700 à 5000 €
StorCenter Pro 250 D	Iomega	500 Go à 1,5 To (Raid 1 ou 5). Sauvegarde intégrée sur disques amovible de 35 ou 70 Go	Gigabit Ethernet	1700 à 3500 €

[1] - voir annexe 3
[2] - http://www.hardware.fr/medias/photos_news/00/22/IMG0022187_1.gif

Il existe des solutions beaucoup plus performantes mais aussi beaucoup plus chères. Toutes ces solutions appartiennent aux géants comme IBM, HP, Dell etc. BlueArc[1] propose ainsi son SiliconServer (FPGA) avec du Fibre Optique entre les nœuds de stockage et du GigabitEthernet en entrée-sortie du serveur. La capacité de stockage pour ces serveurs est de l'ordre 200-300 To pour le prix 500000$ environs. La vitesse de transmission est de l'ordre 1-10 Gb/s. Le serveur de la plus basse gamme coute 50000$ environs.

Une autre solution pour le stockage représente les systèmes open source, comme par exemple une minuscule distribution linux NASLite[2], destinée à transformer un ordinateur en serveur de fichiers. Il existe 3 versions gratuites, chacune ayant la taille de 1722Ko. Elles permettent respectivement de créer un serveur Samba, FTP ou NFS. D'autre versions, payantes, permettent des fonctionnalités plus avancées.

Si on compare la carte NetFPGA avec les produits présentés dans le tableau 1.2 ci-dessus on peut constater que pour le prix de netFPGA à 1000$ nous pouvons gagner en performances tout simplement parce que la plateforme NetFPGA est un dispositif purement matériel.

[1] - http://www.bluearc.com/html/products/hardware.shtml
[2] - http://www.serverelements.com/

2. CHOIX DE L'ARCHITECTURE DU SYSTEME POUR QU'IL SOIT PERFORMANT

2.1 LES POINTS À VÉRIFIER SUR UN SERVEUR NAS

Pour rendre le système de stockage le plus performant on commence ce chapitre par montrer les différents points à vérifier sur un serveur NAS. Ces points sont les suivants:

1. La sécurité des données

On ne peut plus se permettre aujourd'hui d'avoir des données enregistrées sur un simple disque dur. Les solutions RAID (Miroir ou RAID 5 ou 6) qui sont tolérantes à la panne et permettent de continuer à travailler même si un disque dur est tombé en panne.

2. Les protocoles réseaux

Attention tous les serveurs NAS ne savent pas forcement gérer tous les protocoles réseaux (Windows®, Unix®/Linux®, Mac®, Novell®). Ceux-ci sont indispensables pour une bonne interopérabilité dans l'environnement informatique d'une entreprise.

3. Le backup

Voici un point extrêmement important qui reprend les points sur la sécurité des données. Un serveur NAS gère en général une grande quantité de données, qui peut aller de quelques Giga octets à plusieurs centaines voir des Téra Octets.

On imagine un plantage total de la solution et une perte totale ou partielle des données. S'il n'y a pas eu de sauvegarde faite sur bande, ce sont des journées de travail perdues et bien d'autres ennuis en perspective.

La solution NAS doit être capable de gérer sa propre sauvegarde (Lecteur seul et également des librairies ou VTL (Virtual Tape Library) de façon à pouvoir adopter des politiques de sauvegarde efficaces) ou pouvoir être sauvegardée par un serveur de sauvegarde distant (réplication). Ce n'est pas le cas de tous les serveurs NAS. En général, on peut les sauvegarder à distance mais ils ne savent pas forcement gérer les fichiers ouverts et les bases de données.

4. Les performances

Tout dépend de l'utilisation du NAS. Les performances peuvent être un point déterminant. Il faudra regarder alors :

- S'il y a la possibilité d'avoir des liaisons Gigabit Ethernet, (simples ou multiples).
- La gestion du RAID (Matériel ou logiciel). Il faudra privilégier le RAID matériel pour plus de performances.
- Les disques durs qui sont généralement SATA pour les solutions entrée de gamme et SCSI ou SAS pour le haut de gamme. Il faut relativiser ses besoins par rapport au prix. Le SATA est beaucoup moins cher que le SCSI et le SAS.
- Les fonctionnalités réseaux, comme le Load Balancing et l'équilibrage des charges, permettent de répartir la charge réseau sur deux cartes Ethernet distinctes afin d'éviter un goulot d'étranglement et une tolérance à la panne.

5. La disponibilité

En fonction des besoins le NAS devra être plus ou moins tolérant à la panne. Dans certains cas un arrêt de plusieurs heures ne sera pas contraignant alors que dans d'autres il ne devra pas s'arrêter du tout. Des fonctions peuvent répondre à ces besoins. On retrouvera par exemple :

- La Notion de 'Fail Over'[1] pour les réseaux : Elle consiste en l'association de deux (minimum) cartes réseaux (Teaming), qui sont capables de gérer la défaillance d'une carte sans intervention humaine.
- La notion de cluster : Certaines solutions ne doivent pas tomber en panne : A cette fin les fonctionnalités cluster peuvent être utilisées. Deux NAS se partageront les mêmes données : lors de la défaillance de l'un d'eux l'autre prendra le relai.
- La notion de disques HOT SWAP : La plupart des solutions NAS proposent du RAID 5, RAID 6 ou RAID 1 sur deux ou plusieurs disques durs internes sans la

[1] - voir l'annexe 3

fonctionnalité d'extraction à chaud. Effectivement, il n'y a pas de perte de données mais l'arrêt du serveur est obligatoire pour changer le disque dur. C'est une opération qui nécessite l'ouverture du serveur et la manipulation de tournevis, alors qu'il existe aujourd'hui des NAS avec des tiroirs hot swap, ce qui permet de ne pas arrêter le serveur. Le changement est simple et, dans certains cas, on verra des disques paramétrés en secours (HOT SPARE).

- La redondance : Le cluster en fait partie, mais elle existe aussi pour les alimentations, les contrôleurs RAID et les ventilateurs.

2.2 SYSTÈME RAID

Nous avons utilisé plusieurs fois l'abréviation RAID et il ne sera pas superflu pour parler de cette technologie car il est forcement rattaché au NAS. La description des différents niveaux de RAID est importante pour comprendre l'utilité de chacun pour la future utilisation avec la plateforme NetFPGA.

RAID est un acronyme signifiant "Redundant Array of Inexpensive (ou Independent) Disk" ou ensemble redondant de disques indépendants.

Un ensemble ("Array") de type RAID est une collection de disques agissant comme une unité unique de stockage qui supporte la tolérance de panne de disque sans perte de données et qui fonctionne de façon indépendante des autres sous-systèmes.

2.2.1 LES DIFFÉRENTS NIVEAUX DE RAID

Un groupe de recherche de l'université Californienne de BERKELEY a défini sept niveaux de RAID. Chaque niveau correspond à la manière dont les données sont stockées sur les différents disques (un compromis entre coût, sécurité et vitesse). La compréhension de ces niveaux est importante car chaque niveau est optimisé pour une utilisation différente.

1. RAID niveau 0 : Mode "STRIPING"

En mode RAID 0, la donnée à stocker est répartie sur différents disques synchronisés. Aucune information de redondance n'est stockée, ce qui en résulte une vitesse de transfert importante. Néanmoins, la moindre panne de disque entraîne la perte irrémédiable de données. Le nombre minimum de disques requis, pour ce mode est de 2. Ce niveau de RAID est aussi appelé mode " **STRIPING".**

2. RAID niveau 1 : Mode "Mirroring" et "Duplexing"

En mode RAID 1, la donnée est intégralement dupliquée d'un disque sur un autre, d'où une redondance importante en cas de panne d'un disque. Néanmoins ce mode n'admet pas le "HOT SWAP" des disques. Les performances sont supérieures à un disque seul mais le coût du Méga octet est très élevé. Généralement ce mode est réalisé directement par le système d'exploitation de la machine "hôte" ou serveur, entraînant ainsi une charge non négligeable de la CPU. Ce mode requiert au minimum 2 disques et est communément appelé "**MIRRORING**" lorsqu'il est réalisé sur le même canal SCSI, et "**DUPLEXING**" lorsque 2 canaux SCSI sont utilisés.

3. RAID niveau 2 : Code ECC de HAMMING

En mode RAID 2, chaque bit du mot de la donnée est stocké sur un ou plusieurs disques de données et le système génère et stocke sur un ou plusieurs disques ECC un code de correction d'erreur selon l'algorithme de Hamming. L'avantage de ce système est un taux de transfert possible très élevé et une correction d'erreur à la volée sans dégradation des performances. Néanmoins ce système à 'un coût élevé par rapport aux performances obtenues. D'autre part, aujourd'hui tous les périphériques SCSI possèdent en interne leur propre gestion de correction d'erreur. De ce fait le RAID 2 n'est pratiquement plus utilisé.

4. RAID niveau 3 : Transfert parallèle avec gestion de parité

En mode RAID 3, la donnée à stocker est répartie ("STRIPED") en octet sur différents disques synchronisés de données. Le système génère et stocke une parité sur un seul

disque de parité. Il en résulte des taux de transfert très importants en lecture mais aussi en écriture. La panne d'un disque de données a très peu d'impact sur la performance et le nombre minimum de disques pour l'utilisation de ce niveau 3 est de 3. La complexité du contrôleur est moyenne en mode RAID 3. Néanmoins, il est très difficile de le réaliser uniquement par logiciel. En réalité, très peu de constructeurs implémentent un réel RAID niveau 3. En général la taille minimum d'une information stockée est de la taille d'un secteur typiquement 512 octets.

5. RAID niveau 4 : Disques de données indépendants avec gestion de parité partagée

En mode RAID 4, un bloc entier de données est stocké sur un seul disque. Le système génère et stocke une parité de plusieurs blocs provenant de différents disques de données sur un seul disque de parité. Les taux de transferts sont très importants en lecture de larges fichiers mais ce mode est peu performant en écriture. D'autre part, du fait du partage de la parité sur des blocs entiers de données, la reconstruction des données peut s'avérer difficile en cas de panne d'un disque. La panne d'un disque a un impact moyen sur les performances du système. Dans ce mode, les disques de données sont indépendants et ne sont pas lus en parallèle sauf dans le cas où les données proviennent de disques différents. La taille d'un bloc peut varier en fonction de la taille d'un secteur et plusieurs Méga octets suivant le contrôleur utilisé. Le coût du méga octet est relativement faible en mode 4.

6. RAID niveau 5 : Disques de données indépendants avec gestion de parité partagée et distribuée

En mode RAID 5, les informations sont stockées de manière similaire au mode 4, par contre la parité est générée et stockée de façon distribuée sur les disques de données. Les performances sont importantes pour la lecture de larges fichiers, bonnes pour la lecture de petits fichiers et en mode écriture. D'autre part, du fait du partage distribué de la parité sur des blocs entiers de données, la reconstruction des données peut s'avérer difficile en cas de panne d'un disque. La panne d'un disque a un impact moyen sur les performances du système. Dans ce mode, les disques de données sont indépendants et ne sont pas lus en parallèle sauf dans le cas où les données proviennent de disques différents. La taille d'un bloc peut varier en fonction

de la taille d'un secteur et plusieurs Méga octets suivant le contrôleur utilisé.3 Le coût du Méga octet est relativement faible en mode 5. Ce mode requiert au minimum 3 disques.

7. RAID niveau 6 : Disques de données indépendants avec gestion de parité partagée, distribuée et répartie

Le mode RAID 6 est similaire au mode 5 mais utilise plusieurs disques de parité. De ce fait, le mode 6 admet de perdre plus d'un disque de données et de continuer à fonctionner en mode dégradé.

8. RAID niveau 7 : Transferts asynchrones optimisés pour importants débits de données

En mode RAID 7, les informations à stocker sont réparties sur plusieurs disques de données avec un ou plusieurs disques de parité. Le mode 7 met en jeu une carte microprocesseur fonctionnant sous un noyau temps-réel qui contrôle toutes les opérations des canaux SCSI hôtes et des disques, le calcul de la parité, la gestion du cache ainsi que la surveillance des disques. Tous les transferts sont en mode asynchrone ce qui accroît de 1,5 à 6 fois les performances en écriture par rapport aux autres niveaux RAID. Ce mode supporte la perte de plusieurs disques suivant le nombre de disques de parité assignés, et d'autre part peut supporter théoriquement 12 canaux hôtes et 48 disques connectés. La plupart des contrôleurs de RAID 7 peuvent être paramétrés et administrés sur une liaison Ethernet à travers un agent SNMP.

Certains constructeurs de contrôleur RAID propose des combinaisons de plusieurs niveaux de RAID, en voici quelques exemples : RAID 5+1ou RAID niveau 1+0 : Combinaison du mode 0 ("STRIPING") et du mode 1 ("MIRRORING") Ce mode à l'avantage de combiner les performances avec la sécurité. RAID niveau 53 : Combinaison du mode 0 ("STRIPING") et du mode 3 Ce mode accroît davantage encore la combinaison des performances avec la sécurité.

2.3 CHOIX DE LA PLATEFORME

Pour faire un bon choix de la plateforme de stockage il faut tenir compte des points présentés ci-dessous:

- le prix;
- la capacité totale;
- vitesse d'accès, de lecture/écriture sur les disques: port Gigabit Ethernet obligatoire, mémoire vive, processeur, load balancing (répartition de charge)
- prise en charge du RAID (5);
- nombre de connexions simultanées;
- accès et quota;
- prise en charge des disques de très grosse capacité (1 To);
- consommation;
- bruit et ergonomie;

2.4 ARCHITECTURE NETFPGA

La plateforme NetFPGA a été développée à Stanford Université (Figure 2.3). Cette plateforme est destinée à l'utilisation dans la recherche aussi bien qu'à l'enseignement. Elle permet pour les étudiants et les développeurs d'implémenter en matériel des systèmes de réseau très performants.

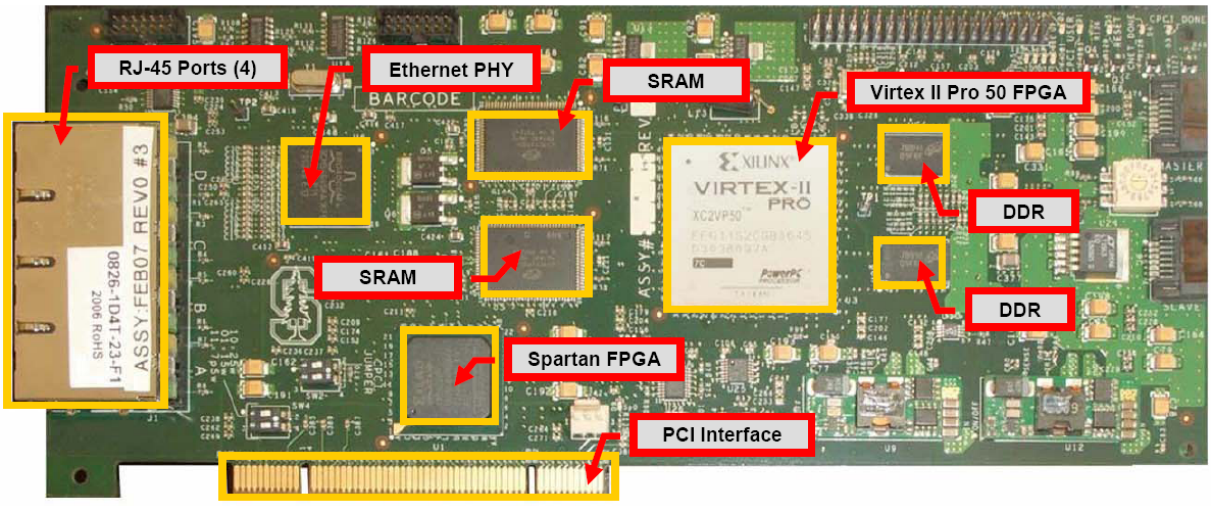


Figure 2.3 – La carte NetFPGA

réf: http://wwwtlc.iet.unipi.it/NP/NetFPGA_upright.jpg

Ils existent déjà plusieurs projets qui ont été développés à Stanford Université et dans les autres institutions dans le monde entier comme par exemple:

- Ethernet NIC (Network Interface Card) à 4 ports avec l'accélération matériel;
- Self-learning[1] Ethernet switch à 4 ports;
- Le routeur IP à 4 ports avec un CPU embarqué;
- Système de Détection d'intrusion de réseau.

Pour le but pédagogique le NetFPGA permet aux étudiants d'avoir l'expérience en implémentation des systèmes de réseaux réels tels que des routeurs et des switchs.

Cette plateforme se compose d'une carte de développement reprogrammable ainsi que d'un didacticiel de courseware[2] au but d'étudier le fonctionnement d'une telle plateforme. La plateforme de développement elle-même est une carte PCI qui peut être installée dans n'importe quel ordinateur avec une interface PCI-X. La Figure 2.4 ci-dessous nous montre un schéma détaillé avec des composants majeurs de la plateforme NetFPGA.

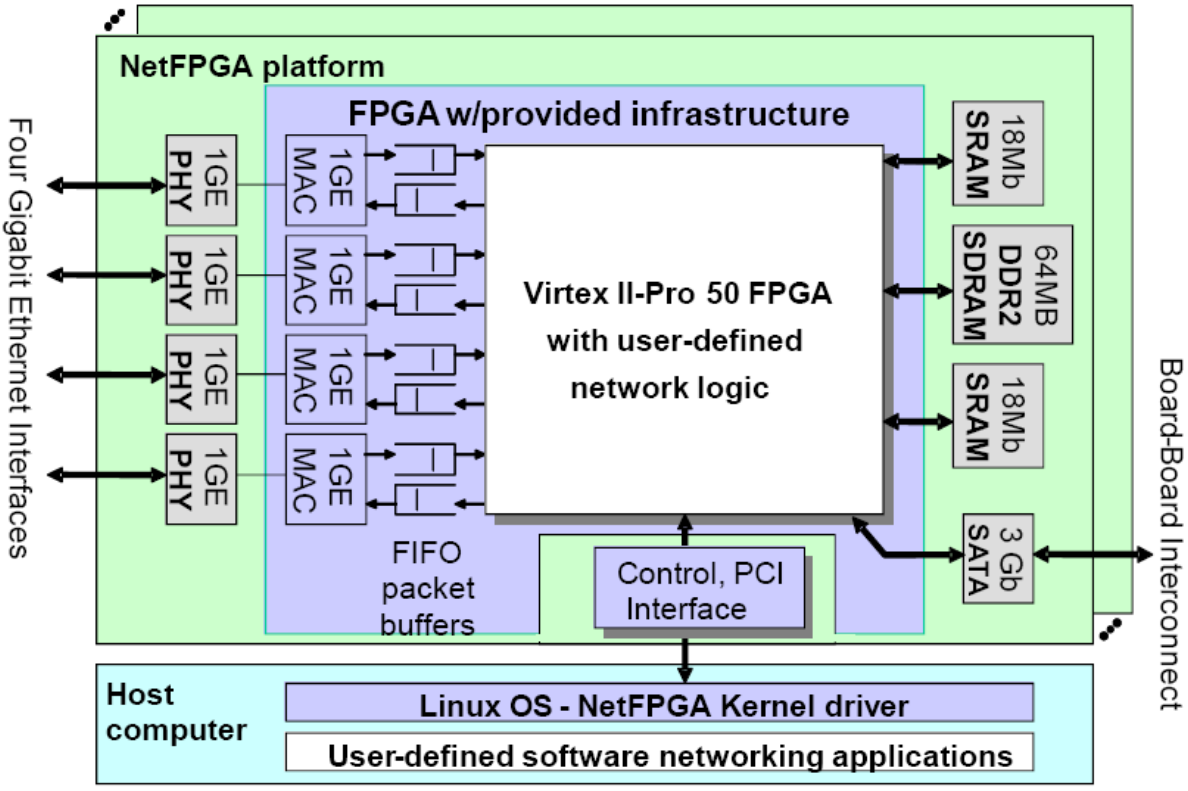


Figure 2.4 - Les composants majeurs de NetFPGA plateforme

réf: http://www.arl.wustl.edu/~lockwood/publications/NetFPGA-MSE_Conf-Lockwood-2007_05_30.pdf

[1,2] - voir annexe 3

La carte contient user-programmable FPGA Virtex II pro 50 (avec deux processeurs PowerPC embarqués), SRAM - 4.5 Mb, DRAM - 64 Mb, quatre interfaces 1Gb/s full-duplex et deux interfaces SATA. Ces interfaces sont conçues pour l'interconnexion entre les deux ou plusieurs cartes NetFPGA. Mais elles ne sont pas utilisables pour connecter un dispositif de stockage. La spécification d'IPCore[1] [14] nous montre que seulement les FPGA Virtex 4 et Virtex 5 sont capables d'implémenter le protocole SATA.

La programmation et l'administration de NetFPGA se fait par l'ordinateur hôte au travers du bus PCI. Cela permet aux utilisateurs de développer et de déployer ses projets sans avoir l'accès physique à la carte. Accent Technology[2] offre un système NetFPGA déjà préinstallé et configuré comme approuvé par l'Université de Stanford (Figure 2.5)

Figure 2.5 – Le système NetFPGA préinstallé.
(réf: http://www.accenttechnologyinc.com/images/netfpga_cube_400x300.jpg)

Un certain nombre d'outils commerciaux sont utilisés pour le développement tandis que des autres outils de développement sont libre[3] (spécifiquement Xilinx[4]). Le simulateur ModelSim est disponible au laboratoire REDS est couvert par la licence.

[1] - voir annexe 3
[2] - http://www.accent-technology.com/
[3] - voir annexe 5
[4] - http://www.xilinx.com/tools/webpack.htm

2.5 NETFPGA EST UNE PLATEFORME DE RECHERCHE

NetFPGA est une plateforme de recherche très efficace car elle est simple à utiliser (mais il faut bonne connaissance du Verilog) et fournit les bonnes performances (en Gbps).

Plusieurs aspects rendent ce système facile à utiliser:

Le "squelette" Verilog[1] proposé par Stanford fournit les interfaces avec les fonctionnes principales telles que: Ethernet MAC, la gestion de la SRAM, la gestion du bus PCI. Ces interfaces utilisent les protocoles de la "demande simple" (request-grant protocols). Cela simplifie le développement car les utilisateurs ne doivent pas s'occuper des détails de MAC ou de SRAM au bas niveau.

L'aspect suivant consiste en ce que les pilotes qui gèrent la carte NetFPGA sont déjà installés sur le système Linux de NetFPGA Cube et donc la carte est prête à l'utilisation. Le système NetFPGA supporte le contrôleur DMA (Direct Memory Access[2]). Ce contrôleur est implanté dans FPGA Spartan (Figure 2.3 ci-dessus). Le design (circuit) implémenté par l'utilisateur peut avoir l'accès au DMA via un simple mécanisme FIFO[3]. Toutes les manipulations avec les interruptions et du transfert des paquets à travers le bus PCI sont gérées par le circuit Spartan.

2.6 NETFPGA - UNE PLATE-FORME D'ÉTUDE

L'intention consiste en ce qu'un étudiant ou un groupe d'étudiants devraient développer un Network System. Par le système on aimerait spécifiquement dire tant matériel que le logiciel. Comme chaque pièce de l'équipement contient tous les deux c'est donc à l'étudiant de décider qu'est-ce qu'il devra implémenter en matériel, en logiciel et comment faire le système complet et fonctionnel.

Voici le schéma de la carte NetFPGA vu par le développeur (Figure 2.5 ci-après)

[1] - voir le repository de ce projet de diplôme
[2,3] - voir annexe 3

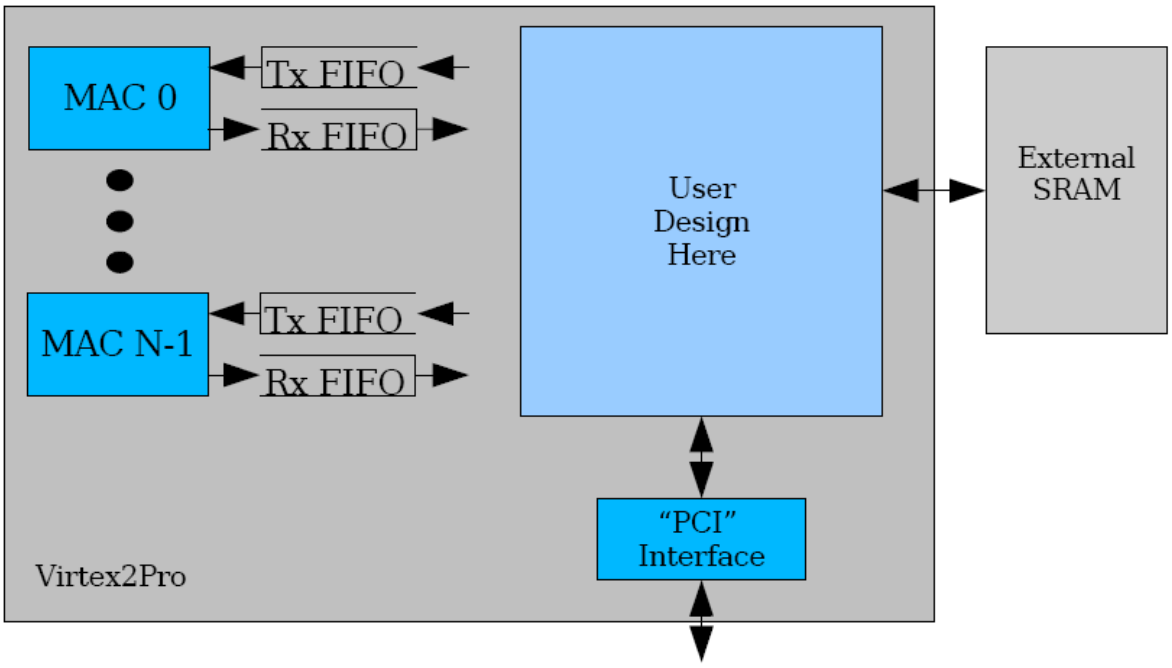


Figure 2.5 – La carte NetFPGA, vue par le développeur

(réf: www.netfpga.org)

Le développeur voit les 'N' interfaces de réseau. Chaque carte contient seulement 4 interfaces physiques et pour chaque interface physique on peut utiliser VLAN[1] avec 256 interfaces virtuelles possibles. Le User Design (le circuit fait par l'utilisateur) implémenté en matériel doit accepter des paquets entrants et décider qu'en faire et au besoin générer des paquets qui partent vers le réseau (par exemple vers le next-hop)

Le développeur dispose d'un simple PCI interface qui permet d'avoir les accès en lecture et en écriture pour configurer, contrôler ou déboguer le système implémenté. Un exemple typique est un projet "référence routeur"[2]. Ce projet permet à un étudiant non seulement de tester le routeur mais aussi d'écrire ces propres programmes et les exécuter sur un poste Linux. Au travers du bus PCI ces programmes peuvent accéder aux tables de routage implémentés en matériel.

Dans ce chapitre j'ai montré les certains fonctionnalités de la plateforme NetFPGA qui prouvent que ce dispositif est prétend d'être une architecture capable à réaliser un système performant.

[1] - voir annexe 3
[2] - voir le repository de ce travail de diplôme

3. IMPLEMENTATION

La première question qu'on peut poser, quel est le protocole qu'on va utiliser pour transférer les données au travers du réseau? Ce protocole doit être léger et simple à implémenter. Bien qu'il existe plusieurs protocoles tels que AFP comme Apple Filing Protocol[1], NFS comme Network File System[1], CIFS comme Common Internet File System[1], NCP comme NetWare Core Protocol[1] etc. On va s'arrêter avec un protocole de transfert de fichiers très simple qui est le TFTP. Une fois le système est fonctionnel on peut alors changer ce protocole par un autre qui pourra implémenter les fonctions de sécurisation et d'authentification mais qui soit aussi plus difficile à réaliser en Verilog.

3.1 STATE MACHINE DU NAS

Avant de commencer la réalisation des différents modules de notre application faisons le graphe d'état de fonctionnement de notre serveur hardware. Son schéma est représenté sur la Figure 3.1 ci-dessous.

Voici la description des états par lesquels passe le NAS ainsi que les conditions de transition d'un état à un autre.

L'état initiale du serveur est "PRÊT", le démarrage du serveur est équivalent de son initialisation - "RESET". Le serveur boucle dans cet état tant qu'il n'y a pas de requête soit "WRQ" (demande d'écriture) soit "RRQ" (demande de lecture). Lorsqu'il reçoit une requête "WRQ" ou "RRQ" il passe à un autre état "ETAB_CONNEXION" (établissement de connexion). Dans cet état le serveur envoie au client l'accusé de réception et passe à l'état suivant "RECEPTION". Dans le cas où le serveur ne peut pas établir la connexion avec le client il passe à l'état "PRÊT". Pendant "RECEPTION" le serveur peut soit fournir un fichier soit le stocker, mais dans les deux cas on appelle cet état "RECEPTION". Le serveur reste dans cet état tant qu'il ne reçoit pas le dernier paquet qui a la taille moins que 512 octets (voir le sous-chapitre suivant "Etude et implémentation de TFTP"). Si tout bien passé il passe à l'état "FIN" et ensuite "PRÊT" et il se met en écoute d'une nouvelle connexion. Sinon, dans le cas d'une erreur il passe à un état "INITIALISATON".

[1] - voir annexe 3

Dans cette phase il efface tous les données reçus, ensuite il initialise les registres et vide la mémoire vive et finalement se met en écoute - état "PRÊT".

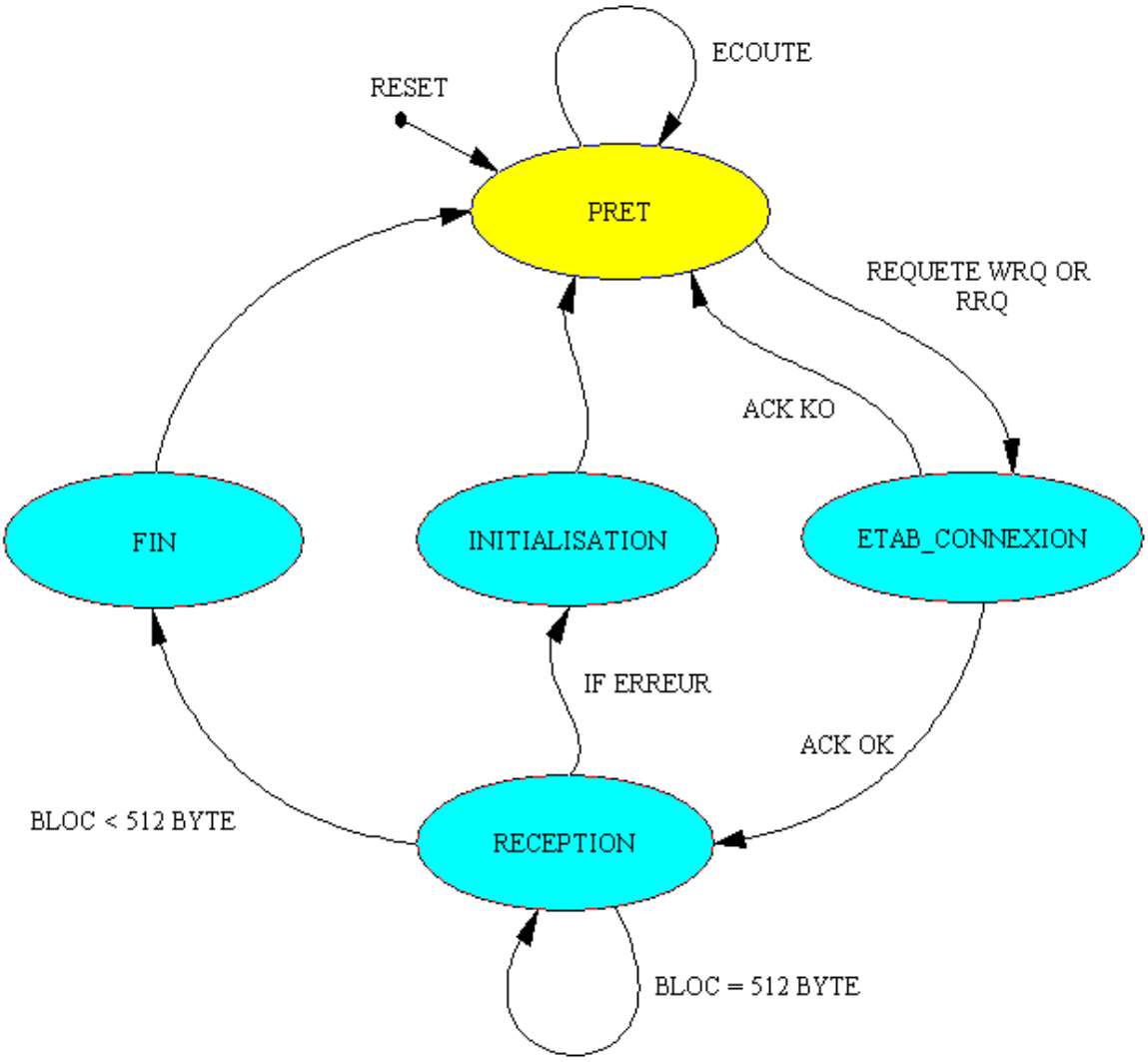


Figure 3.1 - La "state-machine" de fonctionnement de NAS

3.2 ETUDE DE FONCTIONNEMENT DU NAS

L'étape suivante est consiste d'écrire le fonctionnement du NAS représenté sur la figure 3.2 ci-dessous. Le modèle qu'on va appliquer tout au long du projet pour

réaliser le NAS est celui TOP-DOWN. C'est une technique dite descendante ou approche top-down (de haut en bas). Cette approche consiste à concevoir le sujet d'études ou le produit dans les grandes lignes, puis, itérativement, à s'intéresser à des détails de plus en plus fins. L'élaboration d'un circuit intégré repose sur une technique top-down et c'est pour cela on l'appliquera pour implémenter le NAS. Et ce n'est pas la seule raison il y en a encore une. C'est que cette approche permet de délimiter et de conceptualiser rapidement le projet et de le diviser en sous-parties aisément manipulable. Elle permet donc d'avoir une vue globale du projet final et de donner une estimation rapide, bien qu'approximative, de sa complexité et de son coût.

Revenons à la figure 3.2 et faisons quelques commentaires pour ce schéma. Tout le traitement des paquets se fait dans le module NAS, qui représente pour l'instant une "boite noir". On ne sait pas tout ce qu'il y a à l'intérieur mais on connait ses entrées et sorties. Ce module sera implémenté dans un circuit Vritex II qui est relié aux autres périphériques sur la carte NetFPGA tels que la mémoire SRAM, SDRAM, contrôleur GigabitEthernet, interfaces PCI et SATA.

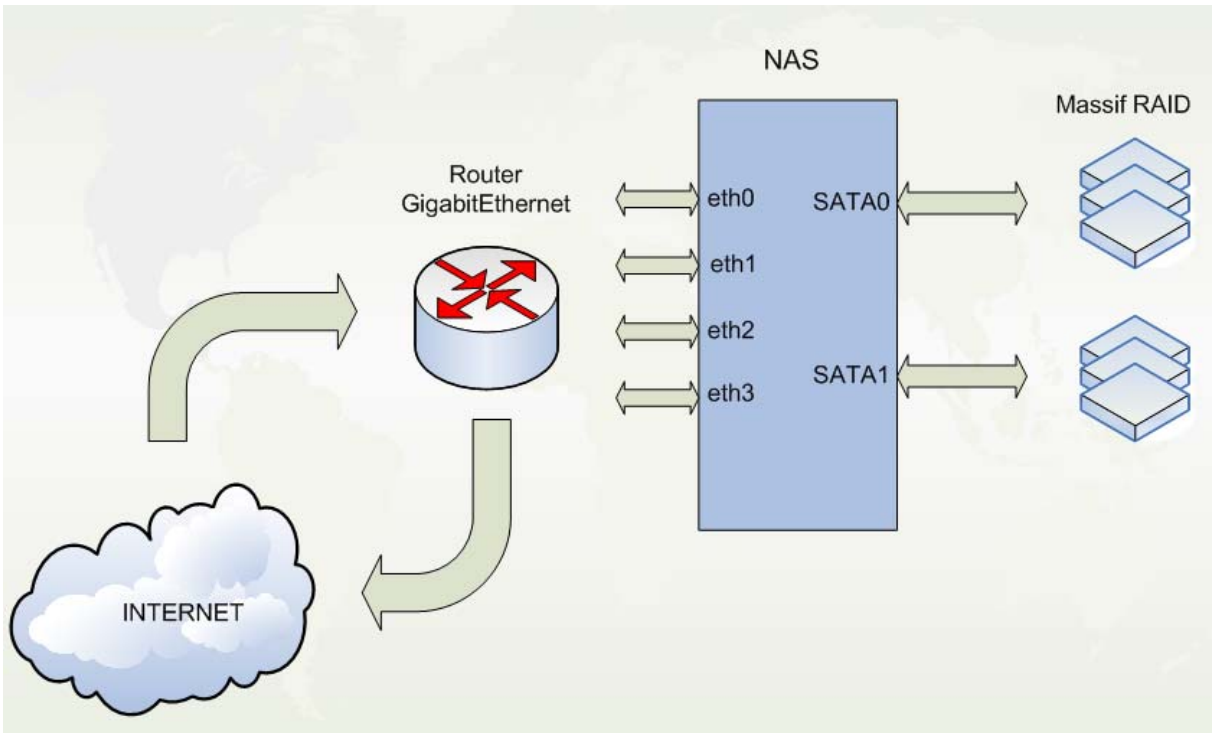


Figure 3.2 - Le schéma principal de NAS

Traitons les deux cas. Dans le premier cas d'une réception des données, elles seront extraites et sauvegardées sur les disques durs connectés en massif RAID 5 (Figure 3.3 ci-dessous). Je propose d'utiliser RAID 5 pour avoir le meilleur débit de stockage et la redondance de données perdues. Une trame reçue sera découpée en 4 morceaux. Dans le deuxième cas d'une envoie des données, elles seront découpées en marceaux de taille fixe (512 octets) sauf le dernier paquet, chaque morceau sera intégré dans le paquet TFTP qui sera inséré sur la pile d'un datagramme qui va le transporter vers la destination (Figure 3.4 ci-dessous)

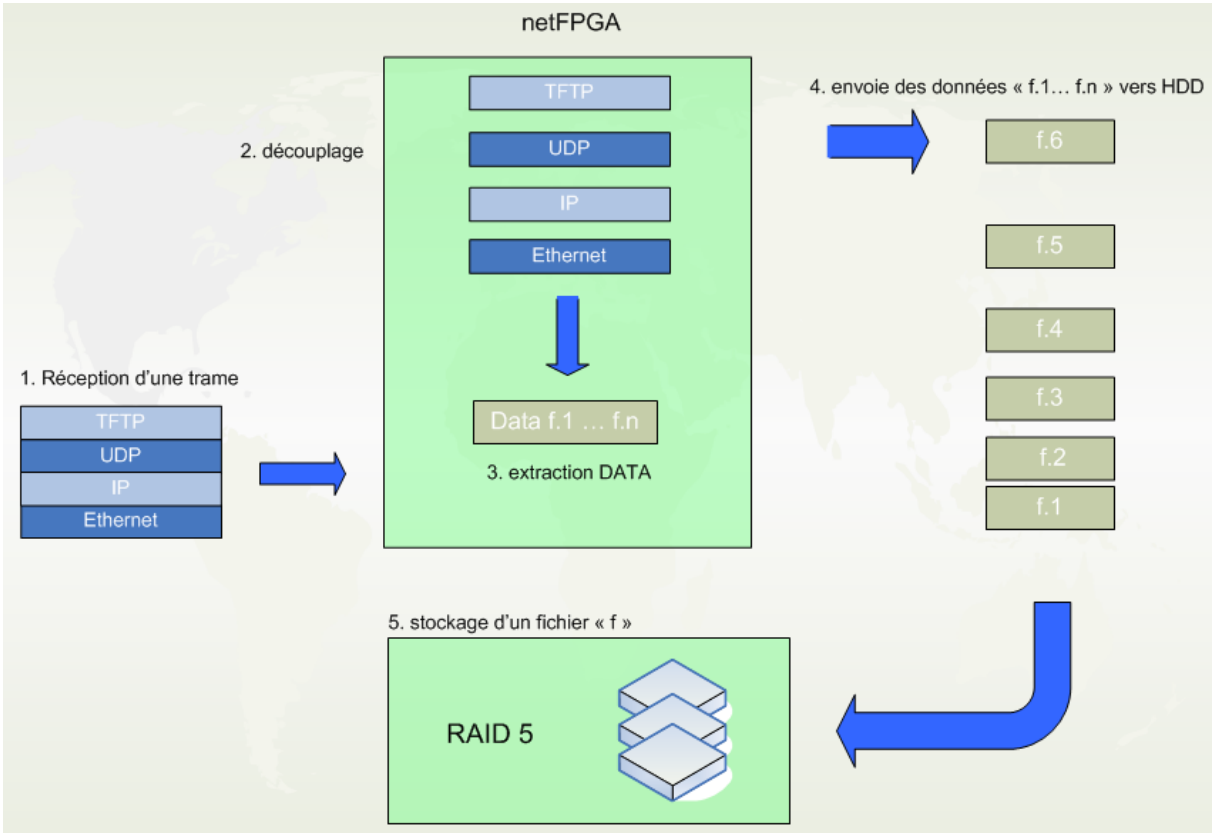


Figure 3.3 - La réception et stockage des données.

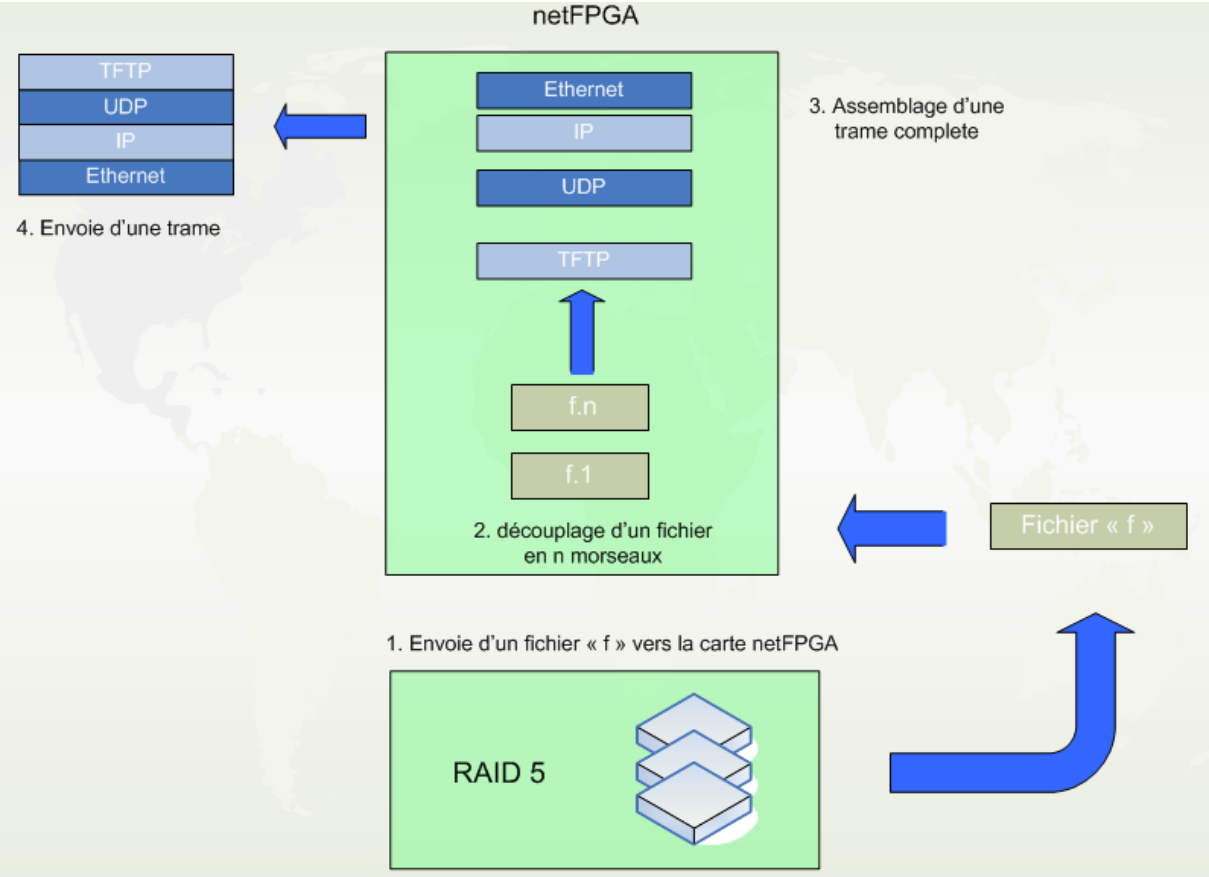


Figure 3.4 - L'envoie des données.

3.3 ETUDE ET IMPLEMENTATION DE PROTOCOLE TFTP

3.3.1 ETUDE DU TFTP

L'étape suivant est d'étudier le protocole TFTP en détail afin de pouvoir le réaliser dans un module hardware.

La dernière version du protocole est le 2, définie dans RFC 1350. Donc TFTP est un protocole très simple utilisé pour transférer des fichiers. Chaque paquet est validé séparément.

Ce protocole a été implanté au dessus du protocole Internet UDP (Figure 3.1 et 3.2 ci-dessous), il lui manque donc la plupart des fonctionnalités d'un FTP ordinaire. Il nous faudra étudier chaque couche pour pouvoir implémenter chaque une. Continuons avec le protocole TFTP, et étudions le plus en détaille.

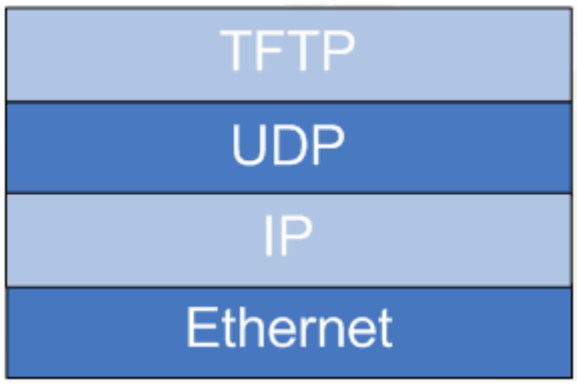


Figure 3.1 - La représentation des différentes couches d'un datagramme TFTP

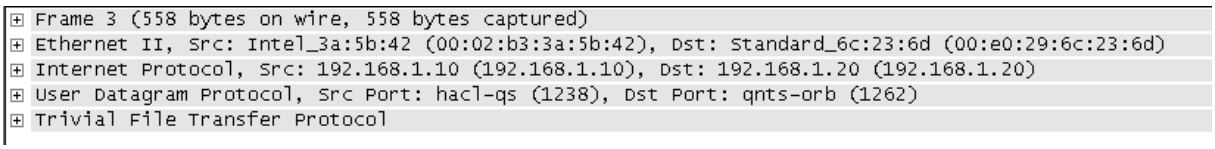


Figure 3.2 - La même représentation, mais faite avec Wireshark (on voit les en-têtes de 4 couches)

La seule chose qu'il peut réaliser est lire et écrire des fichiers depuis ou vers un serveur distant. Il ne peut pas afficher le contenu d'un répertoire, et actuellement l'authentification des utilisateurs n'est pas prévue.

N'import quel transfert démarre par une demande de lecture ou d'écriture de fichier, qui aussi sert de demander de connexion. Si le serveur autorise la requête, la connexion est ouverte et le fichier est envoyé par blocs d'une taille fixe de 512 octets (Figures 3.3 - 3.5)

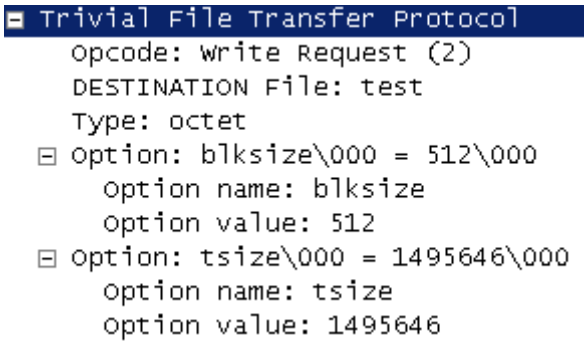


Figure 3.3 - La requête "Write"

Ce qui est important c'est de voir la taille de cette requête et la taille de ses différents champs pour qu'on puisse réaliser son model en Verilog.

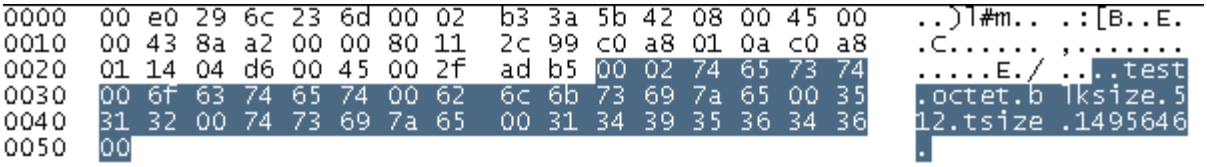


Figure 3.4 - La taille de la requête TFTP "write" vaut 39 octets

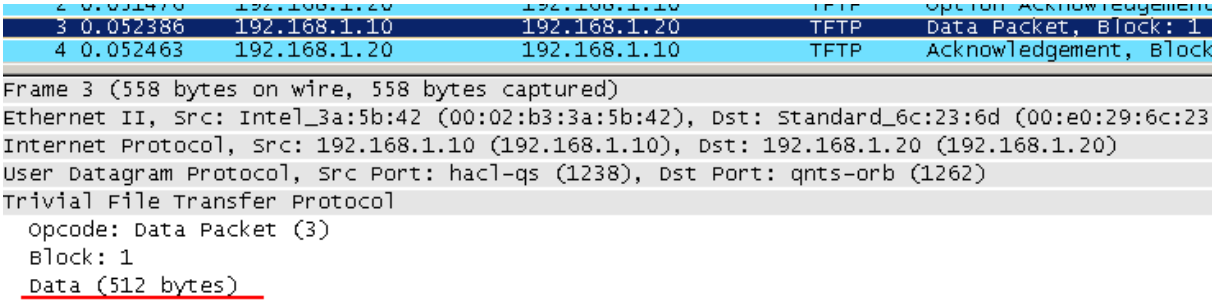


Figure 3.5 - La taille de bloc "data" est de 512 octets

Chaque paquet de données contient un bloc de données, et doit être acquitté par un paquet "accusé de réception" avant que le paquet suivant ne puisse être émis. Un paquet de données de moins de 512 octets signale la terminaison du transfert. Si un paquet se perd su le réseau, une fin d'attente se déclenchera chez le destinataire et il pourra retransmettre son dernier paquet (qui peut être de données ou un accusé de réception), provoquant ainsi la retransmission du paquet perdu par l'émetteur. L'émetteur, depuis l'étape garantissant que tous les anciens paquets ont bien été reçus, ne doit conserver qu'un paquet en mémoire pour la retransmission. Observons que les deux machines concernées par un transfert sont considérées comme émettrice et réceptrice. L'une envoie des données et reçoit des accusés de réception, l'autre envoie des accusés de réception et reçoit des données. Notre NAS doit contenir les deux blocs en gros qui vont accomplir ses fonctions d'émission et de réception des paquets.

La plupart des erreurs provoquent la rupture de la connexion. Une erreur est signalée par l'émission d'un paquet "erreur" (Figure 3.6)

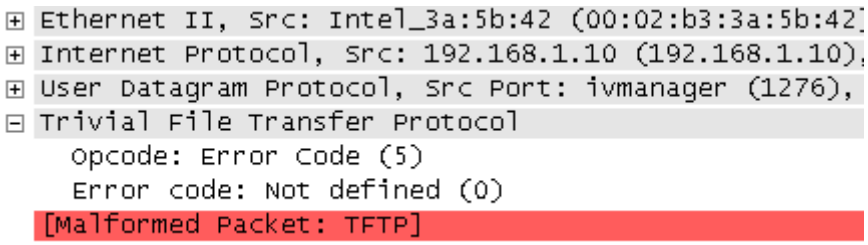


Figure 3.6 - Ce paquet "Erreur" provoque la rupture de connexion

Ce paquet n'est ni acquitté ni retransmis (par exemple, un serveur TFTP ou un utilisateur peut rompre sa connexion après l'envoie d'un message d'erreur) alors l'autre extrémité de la connexion peut ne pas l'avoir reçu. Par conséquent des délais d'attente sont utilisés quand le paquet "erreur" a été perdu pour détecter une telle terminaison. Les erreurs sont provoquées par trois types d'événements: incapacité à satisfaire la demande (par exemple, fichier non trouvé, violation d'accès, ou utilisateur inexistant), réception d'un paquet qui ne peut s'expliquer par un délai ou une duplication sur le réseau (par exemple, un paquet mal formé), ou la perte de l'accès à une ressource nécessaire (par exemple, disque plein ou accès refusé pendant le transfert). TFTP ne reconnaît qu'une seule condition d'erreur qui ne provoque pas la rupture: le port source d'un paquet reçu est incorrect. Dans ce cas, un paquet d'erreur est émis vers la machine d'origine.

Ce protocole est très restrictif afin de simplifier l'implémentation. Par exemple, la taille fixe des blocs prépare franchement l'allocation future, et l'étape d'attente d'accusé de réception fournit un mécanisme de contrôle de flux et évite le besoin de remettre dans l'ordre les paquets entrants.

Comme mentionné, TFTP est conçu pour être implanté au dessus d'un protocole datagramme (UDP). Puisque le Datagramme est implanté sur le protocole Internet, les paquets possèdent un en-tête Internet un en-tête de datagramme, et un en-tête TFTP. En plus, les paquets peuvent avoir un en-tête (LNI, ARPA et dans notre cas Ethernet) pour leur permettre de circuler sur le support de transmission local. Comme indiqué dans la figure 3.1 ci-dessus, l'ordre des contenus d'un paquet sera: en-tête support local, en-tête Internet, en-tête Datagramme, en-tête TFTP, suivis par le reste du paquet TFTP. TFTP ne précise directement aucune valeur dans l'en-tête Internet. Par contre, le port source et le port destination de l'en-tête du Datagramme sont utilisés par TFTP et le champ longueur reflète la taille du paquet TFTP. Les

identificateurs de transfert (TID) utilisé par TFTP sont transmis à la couche Datagramme pour être utilisés comme ports; donc ils doivent être compris entre 0 et 65535. L'en-tête TFTP est constitué d'un champ de 2 octets qui indique le type du paquet (par exemple DONNEE, ERREUR, etc.) Figure 3.7

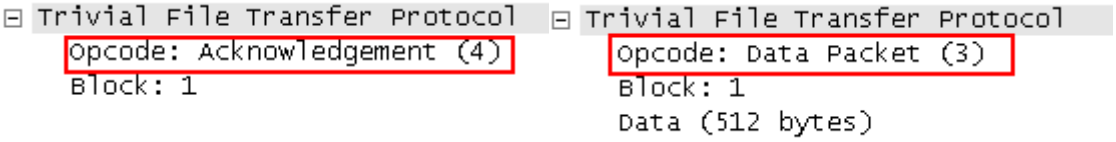


Figure 3.7 - L'en-tête TFTP (indiqué le type de paquet ACK et DATA)

Un transfert est établi par l'émission d'une requête (rappel WRQ pour écrire vers un système de fichier, ou RRQ pour le lire), et la réception d'une réponse positive, un accusé de réception pour écrire, ou le premier paquet de données à lire. En général un paquet "accusé de réception" doit contenir le numéro de bloc du paquet de données qui doit être acquitté (Figure 3.8)

Source	Destination	Protocol	Info
192.168.1.10	192.168.1.20	TFTP	Read Request, File: test_1
192.168.1.20	192.168.1.10	TFTP	Option Acknowledgement, bl
192.168.1.10	192.168.1.20	TFTP	Acknowledgement, Block: 0
192.168.1.20	192.168.1.10	TFTP	Data Packet, Block: 1
192.168.1.10	192.168.1.20	TFTP	Acknowledgement, Block: 1
192.168.1.20	192.168.1.10	TFTP	Data Packet, Block: 2
192.168.1.10	192.168.1.20	TFTP	Acknowledgement, Block: 2
192.168.1.20	192.168.1.10	TFTP	Data Packet, Block: 3
192.168.1.10	192.168.1.20	TFTP	Acknowledgement, Block: 3

Figure 3.8 - chaque paquet ACK contient le numéro du paquet reçu/envoyé

A chaque paquet de données est associé un numéro de bloc; les numéros de blocs sont consécutifs et démarrent à 1. Puisque la réponse positive à une demande d'écriture est un paquet "accusé de réception", dans ce cas particulier le numéro de bloc sera zéro. Normalement, puisqu'un paquet "accusé de réception" valide un paquet de données, le paquet "accusé de réception" doit contenir le numéro de bloc du paquet de données à valider. Si la réponse est un paquet "erreur", alors la requête est rejetée.

Chaque extrémité choisit en priorité un TID pour elle même afin d'établir la connexion. Il sera utilisé durant cette connexion. Le TID d'une connexion sera choisit aléatoirement, ainsi la probabilité que le même nombre soit choisit deux fois de suite est très faible. A chaque paquet sont associés les deux TID de la phase de connexion, le TID source et TID destination. Ces TID sont remis au support UDP comme ports source et destination. Une machine effectuant une demande choisit son TID source comme décrit ci-dessus, et émet sa requête initiale avec le TID réservé 69 en décimal pour la machine destinataire. La réponse à la demande, en fonctionnement normal, utilise le TID choisi par le serveur comme TID source et le TID choisi par le requérant dans son message préalable comme TID destination. Les deux TID choisis sont alors utilisés pour le reste du transfert.

Par exemple, la suite montre les étapes utilisées pour établir une connexion dans le but d'écrire un fichier. Notons que WRQ, ACK et DATA sont les noms respectifs des types de paquets demande d'écriture, accusé de réception et données.

1. La machine 192.168.1.10 émet un "WRQ" vers la machine 192.168.1.20 (serveur) avec source = TID de 192.168.1.10 qui vaut 1238, destination = 69 (Figure 3.9)
2. La machine 192.168.1.20 émet un "ACK" (avec numéro de bloc = 0) vers la machine 192.168.1.10 avec source = TID de 192.168.1.20 qui vaut 1262, destination = TID de 192.168.1.10 qui vaut 1238 défini préalablement lors de la requête "WRQ" (Figure 3.10 ci-dessous)

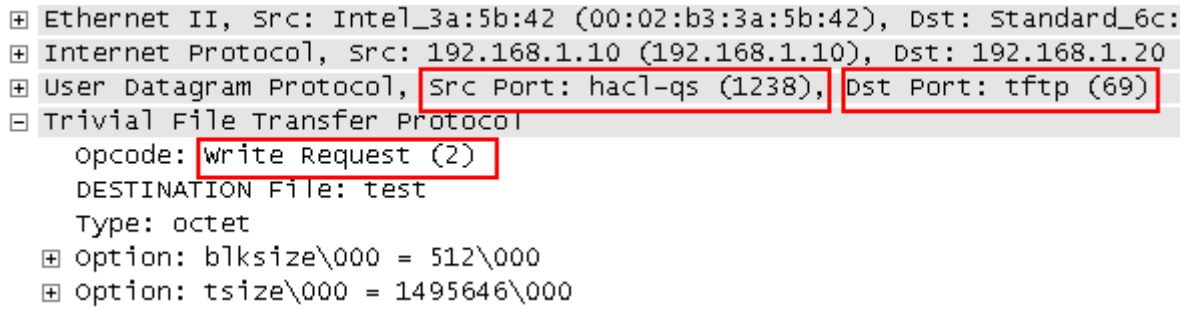


Figure 3.9 - la requête "WRQ" avec le TID de source = 1238

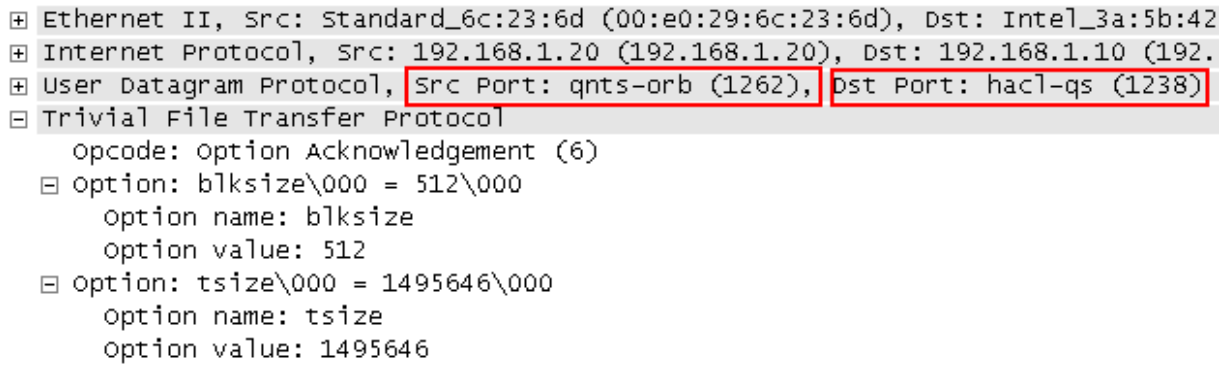


Figure 3.10 - la requête "ACK" avec le TID de source = 1262, destination = 1238

A cet instant la connexion a été établie et le premier paquet de données peut être émis par la machine 192.168.1.10 avec un numéro de séquence à 1. Dans la prochaine étape, et dans toutes les suivantes, les machines doivent s'assurer que le TID source est égal à la valeur contenue lors des étapes 1 et 2. Si un TID source n'est pas le même, le paquet doit être considéré comme provenant par erreur d'un autre endroit. Un paquet "erreur" doit être envoyé à la source du mauvais paquet, tandis que le transfert n'est pas perturbé. Ceci ne peut être réalisé que si le TFTP reçoit effectivement un paquet avec un TID incorrect. Si le protocole de transport ne le permet pas, cette condition d'erreur particulière n'arrivera pas.

TFTP reconnaît cinq types de paquets, tous ceux-ci sont mentionnés ci-dessous:

Code opération	Opération
1	Demande de lecture (RRQ)
2	Demande d'écriture (WRQ)
3	Données (DATA)
4	Accusé de réception (ACK)
5	Erreur (ERROR)

Figure 3.11 - cinq types de paquets TFTP

Pour la description du protocole TFTP on s'arrête à ce point là et passons à l'étape suivant qui est l'implémentation.

3.3.2 IMPLEMENTATION DU TFTP

Après avoir vu les détails sur le protocole TFTP nous descendons dans notre model top-down au niveau plus bas. C'est-à-dire nous allons réaliser un module qui implémente le protocole TFTP. Dans notre projet il y aura deux modules TFTP différents. Un bloc qui est pour le but de traiter les trames reçues (Figure 3.12) et un autre pour traiter les trames qui seront envoyées (Figure 3.13). Il faut préciser que ces blocs sont simplifiés et le schéma final du NAS est présenté en Annexe 1.

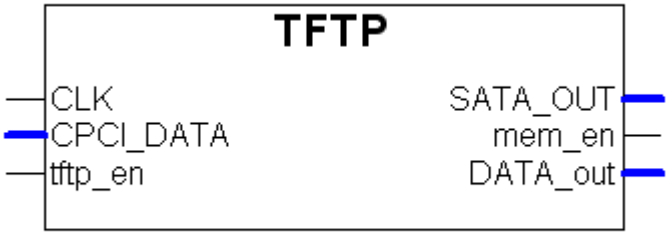


Figure 3.12 - bloc TFTP qui traite les trames reçues

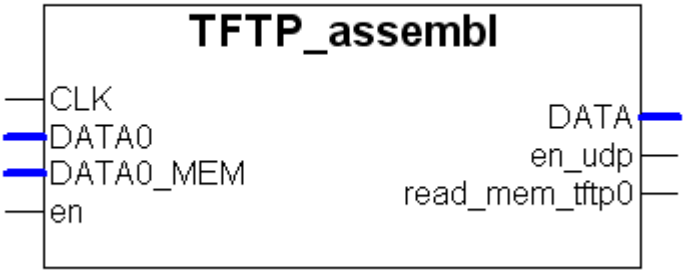


Figure 3.13 - bloc "TFTP_assembl" qui construit un paquet TFTP

Le map des entrées-sorties est présenté en Annexe 2. Par contre il est nécessaire de faire la description détaillée de certaine fonctionnalité de chacun des blocs.

On commence par bloc **TFTP**. Lorsqu'il Il est activé par le signal "tftp_en" Il reçoit sur son entrée "CPCI_DATA" le paquet TFTP extrait dans le bloc précédent "analyseur"[1]. Ensuite le bloc **TFTP** extrait la partie DATA (figure 3.14 ci-dessous) de ce paquet TFTP est l'envoie vers la sortie "SATA_OUT". Cette sortie est connectée au contrôleur

[1] - voir annexe 1

RocketIO (SATA). Ce contrôleur se préoccupe d'envoyer ces données sur le disque dur au travers de protocole SATA.

La sortie "mem_en" sert à activer le gestionnaire de mémoire. Le gestionnaire de mémoire est activé par le bloc **TFTP** qui à son tour envoie le flux des données au travers de la sortie "DATA_out" pour les mémoriser dans la SRAM. Pour le protocole TFTP on va mémoriser seulement le numéro de bloc reçu car ce même numéro sera utilisé pour acquitter le bloc reçu.

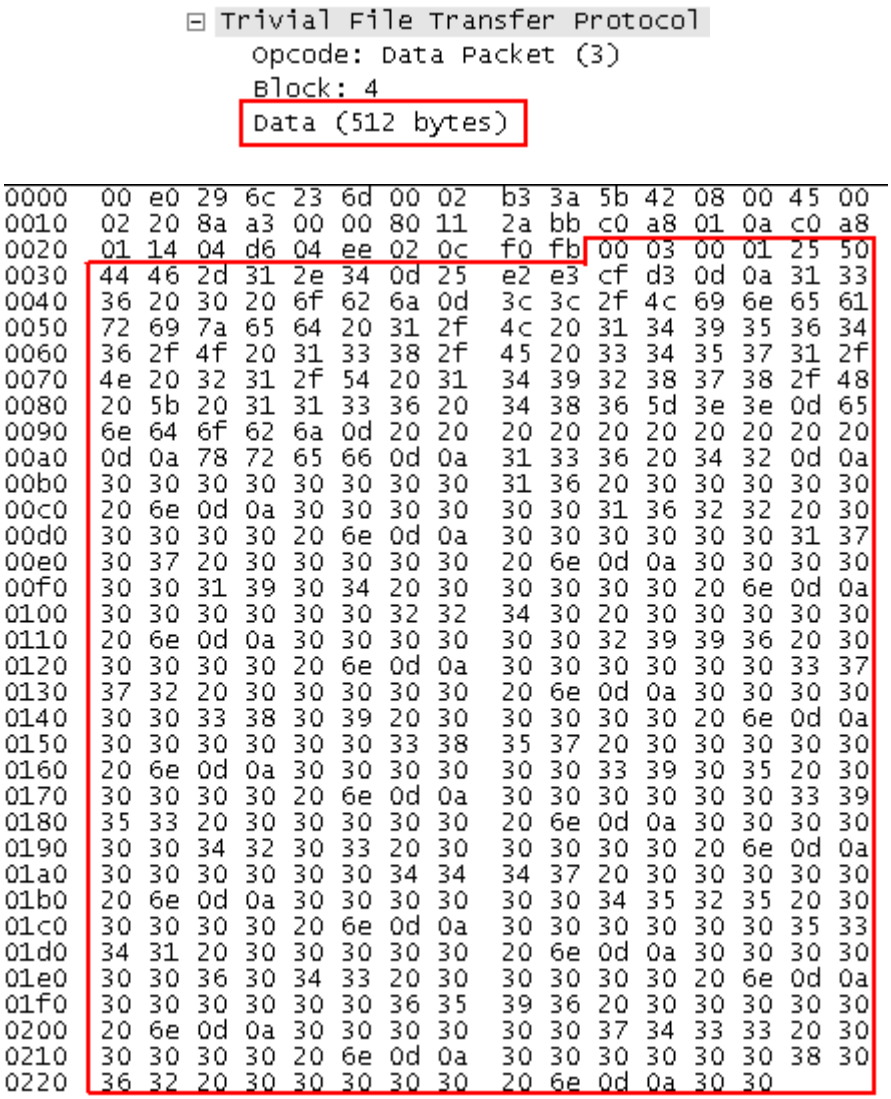


Figure 3.14 - DATA extraits par le bloc TFTP

Maintenant on suppose qu'on envoie les données au même destinataire pour acquitté la trame reçue. Dans ce cas là on utilisera un autre bloc **TFTP_assembl**. Il reçoit les données à l'entrée DATA0 depuis le contrôleur SATA (Figure 3.13 ci-dessus) ainsi que les données qui ont étaient mémorisées dans la SRAM préalablement. Toutes ces données sont assemblées par ce bloc en un paquet TFTP et ce paquet est envoyé sur la sortie DATA vers un bloc suivant **UDP_assembl**. Avant d'envoyer ce paquet TFTP on l'active par un signal "en_udp". On termine avec les blocs TFTP et passe à une autre couche qui est la couche d'UDP.

3.4 ETUDE ET IMPLEMENTATION DE PROTOCOLE UDP

3.4.1 ETUDE DU PROTOCOLE UDP

Le protocole User Datagram Protocol est définit dans RFC768. Il est pour le but de fournir une communication par paquet unique entre deux processus dans un environnement réseau étendu. Ce protocole suppose l'utilisation du protocole IP comme support de base à la communication.
Ce protocole définit une procédure permettant à une application d'envoyer un message court à une autre application, selon un mécanisme minimaliste. Ce protocole est transactionnel, et ne garantit ni la délivrance du message, ni son éventuelle duplication.

La structure de l'entête est représentée sur la figure 3.15.

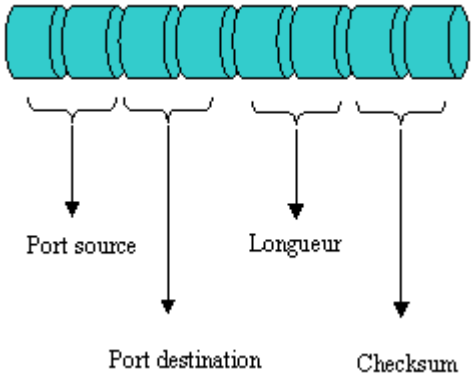


Figure 3.15 - Entête UDP - 8 octets

3.4.1.1 DEFINITION DES DIFFERENTS CHAMPS

1. Port source.

Le champ port source est codé sur 16 bits et correspond au port relatif à l'application en cours sur la machine source. Comme c'était déjà montré pour le protocole TFTP qui est transporté sur UDP les numéros de ports sont les TID utilisés par TFTP qui sont transmis à la couche Datagramme.

2. Port destination

Le champ Port destination est codé sur 16 bits et il correspond au port relatif à l'application en cours sur la machine de destination. Comme dans le cas de port source les numéros de port destination sont les TID utilisés par TFTP.

3. Longueur

Le champ Longueur est codé sur 16 bits et il représente la taille de l'entête et des données. Sont unité est l'octet et sa valeur maximale est 64 Koctets (2^16)

4. Checksum

Le champ Checksum est codé sur 16 bits et représente la validité du paquet de la couche 4 UDP. Dans notre module on ne va pas utiliser le Checksum pour simplifier la tache. C'est tout à fait possible car un checksum transmis avec une valeur zéro indique qu'aucun Checksum n'a été calculé.

Ce protocole est simple à implémenter. Nous avons seulement le port de source, le port de destination et la longueur qui se calcule en prenant la taille d'un paquet TFTP et en y ajoutant la taille d'entête du datagramme UDP.

3.4.2 IMPLEMENTATION DU PROTOCOLE UDP

Les deux modules qui implémentent ce protocole sont représentés sur les Figures 3.16 et 3.17

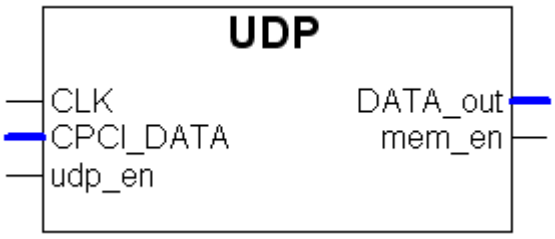


Figure 3.16 - bloc UDP qui traite les trames reçues

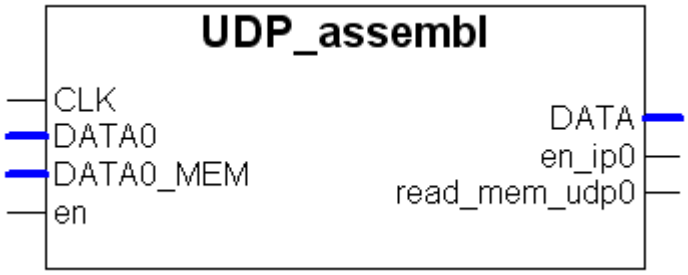


Figure 3.17 - bloc "UDP_assembl" qui construit un paquet UDP

La signification des entrées-sorties est similaire à celle des modules **TFTP** et **TFTP_assembl**. Le map des entrées-sorties est présenté en Annexe 2. Précédemment nous avons vu que le module **TFTP** extrait du datagramme reçu les données ainsi que les valeurs de certain champs. La même fonctionnalité a pour le bloc **UDP** qui extrait les valeurs des ports source et destination et les place dans la mémoire vive. Le bloc **UDP_assembl** sert à construire le paquet UDP et à insérer au dessous de la couche TFTP. Pour que ça soit plus clair on s'adresse au bloc-schéma sur la Figure 3.18

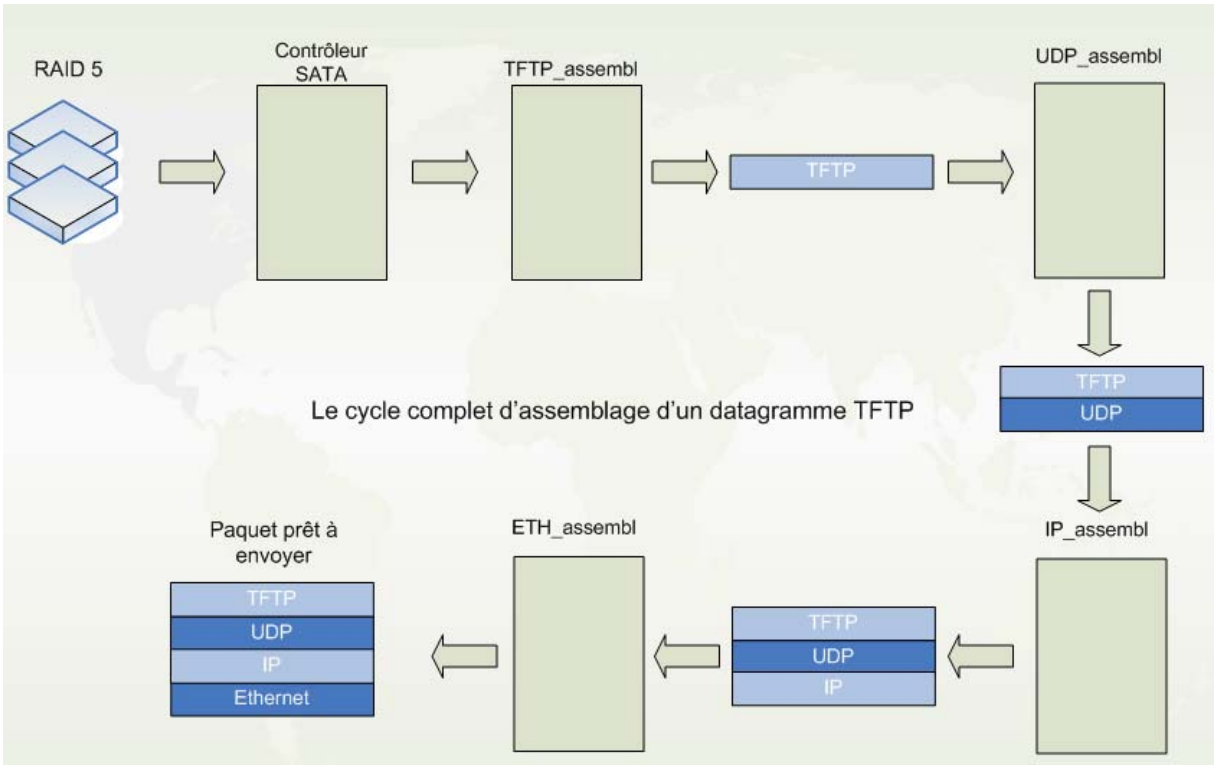


Figure 3.18 - bloc-schéma d'assemblage d'un datagramme TFTP complet

Dans ce bloc-schéma il y a les différents blocs qui sont connectés en série. Chaque bloc crée une couche et l'insère dans un datagramme. On commence par TFTP et on termine par la couche Ethernet II. Le contrôleur RocketIP (SATA) convertit les données codées avec un protocole SATA en données codées sur 64 bits. A la sortie de cette série des blocs on obtient un datagramme prêt à l'envoie vers sa destination.

3.5 ETUDE ET IMPLEMENTATION DU PROTOCOLE IP

3.5.1 ETUDE DU PROTOCOLE IP

Le protocole Internet est définit dans RFC791. Il est conçu pour supporter l'intercommunication de systèmes informatiques sur une base de réseau par commutation de paquets. Le rôle du protocole internet est la transmission de blocs de données, appelés datagrammes, d'une source vers une destination, la source et la destination étant des ordinateurs hôtes identifiés par une adresse de longueur fixe. Le protocole Internet dispose des mécanismes permettant la fragmentation de longs datagrammes et leur réassemblage, lors de leur transmission à travers des réseaux de "dimension" inférieur.

Faisons la brève description du protocole Internet avant de pouvoir poursuivre son réalisation.

Le protocole Internet implémente deux fonctions de base: l'adressage et la fragmentation. Les modules Internet exploiteront les adresses inscrites dans l'entête Internet pour acheminer le datagramme vers sa destination. La sélection d'un chemin partant de la source vers la destination est appelée le routage. Le protocole Internet considère chaque datagramme Internet comme une entité indépendante et sans aucune relation avec d'autres datagrammes.

Le protocole Internet utilise quatre mécanismes clefs pour procurer le service promis: **Type de Service**, **Durée de Vie**, **Options** et **Checksum** d'entête.

- Le Type de Service indique la qualité du service désiré. Le type de service est un ensemble générique de paramètres qui caractérisent les choix de service disponible sur le réseau qui support la communication Internet. Cette indication de type de service sera utilisée par les routeurs pour commuter les paramètres de transmission actuels d'un réseau particulier, le réseau à utiliser sur le segment suivant, ou pour spécifier le routeur suivant lors du routage d'un datagramme.

- La Durée de Vie indique une limite pour la durée d'existence d'un datagramme dans le réseau. Elle est initialisée par l'émetteur du datagramme et décrémentée par les éléments actifs situés sur le chemin que parcourt le datagramme. Si cette durée de vie atteint la valeur zéro avant que le datagramme Internet n'atteigne sa destination, ce dernier sera détruit. Cette durée de vie peut être vue comme une temporisation d'autodestruction du datagramme.
- Les Options permettent le transport de signaux de contrôle utiles voir nécessaires dans certaines situations particulières mais secondaire pour la fonction essentielle de la communication. Les options permettent par exemple le marquage temporel, le codage de sécurité, et des commandes de routage spéciales.
- Le Checksum d'entête permet de vérifier que les informations ajoutées par un module Internet ont été correctement transmises. Les données peuvent néanmoins contenir des erreurs. Si le Checksum échoue, le datagramme Internet est immédiatement rejeté par l'entité qui l'a reçu.

Pour avoir plus de détails sur les fonctionnalités de protocole Internet telles que la Fragmentation et l'Adressage, voir la RFC 791

3.5.1.1 STRUCTURE D'ENTETE IP

Voici la structure d'entête IP basé sur 20 octets (Figure 3.19) et le complément de l'entête (Figure 3.20)

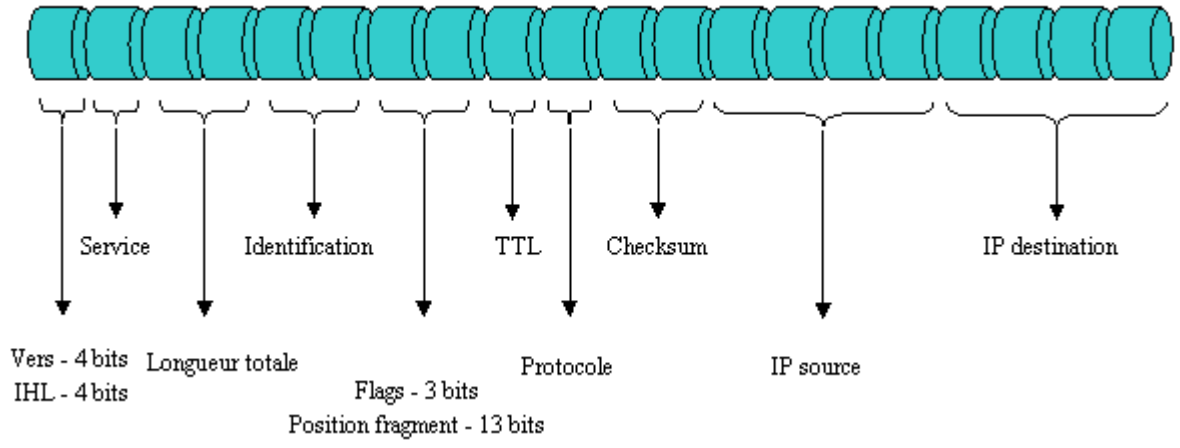


Figure 3.19 l'entête IP sur 20 octets

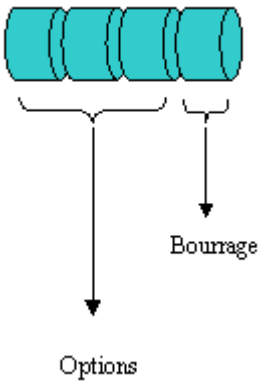


Figure 3.20 le complément optionnel de l'entête IP basé sur 4 octets

Analysons les champs d'entête pour définir comment les gérer dans le bloc IP qu'on va créer.

1. Le champ **Vers**

Le champ version est codé sur 4bits. Il représente le numéro de version du protocole IP. Pour notre module on peut simplifier l'assignation à ce champ et on lui donne une valeur 0x4 car on utilise IPv4. Donc ce champ est une constante.

2. **IHL**

IHL signifie "Internet header length". Ce champ est codé sur 4 bits et représente la longueur en mots de 32 bits de l'entête IP. Par défaut il est égal à 0x5 (20 octets). Donc ce champ est une constante.

3. **Service**

Le champ Service "Type of Service" est codé sur 8 bits, il permet la gestion d'une qualité de service traitée directement en couche 3 du modèle OSI. Cependant, la plupart des équipements de Backbone ne tiennent pas compte de ce champ et même certain le réinitialise à 0. Donc on lui attribue la valeur 0.

4. **Longueur totale**

Ce champ est codé sur 16 bits et représente la longueur du paquet incluant l'entête IP et les Data associées. La longueur totale est exprimé en octets, ceci permettant de spécifier une taille maximum de 2^{16} = 65535 octets. La longueur des Data est obtenue par la combinaison des champs IHL et Longueur totale:
Longueur_des_data = Longueur_totale - (IHL * 4);

5. **Identification**

Ce champ est codé sur 16 bits et constitue l'identification utilisée pour reconstituer les différents fragments. Chaque fragment possède le même numéro d'identification, les entêtes IP des fragments sont identiques à l'exception des champs Longueur Totale, Checksum et position Fragment.
Pour tous les détails des mécanismes de fragmentation et de réassemblage voir la RFC 815.

6. **Flags**

Le champ Flags est codé sur 3 bits et indique l'état de la fragmentation. Voici les détails des différents bits constituant ce champ:

a. **Reserved.** Le premier bit est réservé et positionné à 0, donc dans le module IP c'est une constante.
b. **DF**. Appelé DF "Don't Fragment", le second bit permet d'indiqué si la fragmentation est autorisé. Si un Datagramme devant être fragmenté possède le flag DF à 1, alors, il sera alors détruit. Dans le module IP ce bit doit être géré.
c. **MF**. Appelé MF "More Fragments", le troisième bit indique s'il est à 1 que le fragment n'est pas le dernier. Ce bit doit être géré par le module IP.

7. **Position Fragment**

Le champ Position Fragment est codé sur 13 bits est indique la position du fragment par rapport à la première trame. Le premier fragment possède donc le champ Position Fragment à 0. Ce bit sera géré par le module IP.

8. **TTL**

Le champ TTL (Time To Live) est codé sur 8 bits et indique la durée de vie maximale du paquet. Il représente la durée de vie en seconde du paquet. Si le TTL arrive à 0, alors l'équipement qui possède le paquet, le détruira. Le but de champ TTL est d'éviter de faire circuler des trames en boucle infinie.

9. **Protocole**

Ce champ est codé sur 8 bits es représente le type de Data qui se trouve derrière l'entête IP.
Pour tous les détails des types de protocole voir la RFC 1700 qui remplace désormais la RFC 1340.
Voici la liste des protocoles les plus connu[1]:

- 01 - 00001 - ICMP
- 02 - 00010 - IGMP
- 06 - 00110 - TCP
- 17 - 10001 - UDP

10. **Checksum**

Ce champ est codé sur 16 bits et représente la validité du paquet de la couche 3. Pour pouvoir calculer le Checksum, il faut positionner le champ du Checksum à 0 et ne considérer que l'entête IP. Donc par exemple, si deux trames ont la même entête IP et deux entête ICMP et Data différentes, le checksum IP sera alors le même.

Un exemple de fonction écrit en Perl[1] permettant la vérification du Checksum IP est présenté en Annexe 4. Cette fonction permet de tester la Checksum implémenté en hardware.

Pour tous les détails du Checksum IP voir la RFC 1071.

11. **Adresse IP source**

Le champ IP source est codé sur 32 bits et représente l'adresse IP source ou de réponse. Ce champ est tout simplement mémorisé dans la SRAM pour chaque interface: eth0 .. eth3 soit construit par le module IP.

12. **Adresse IP destination**

Le champ IP destination est codé sur 32 bits et représente l'adresse IP destination. Ce champ sera implémenté comme pour l'Adresse IP source.

13. **Option**

Ce champ est codé entre 0 et 40 octets. Il n'est pas obligatoire et donc on l'omet.

[1] - voir annexe 3

14. **Bourrage**

Ce champ est de taille variable comprise entre 0 et 7 bits. Il permet de combler le champ Option afin d'obtenir un entête IP multiple de 32 bits. La valeur des bits de bourrage est 0.

Les deux modules qui réalisent le protocole Internet sont représentés sur les Figures 3.21 - 3.22

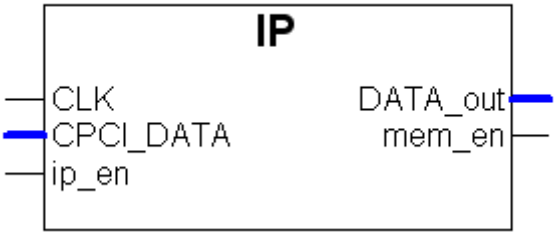


Figure 3.21 - bloc IP qui traite les trames reçues

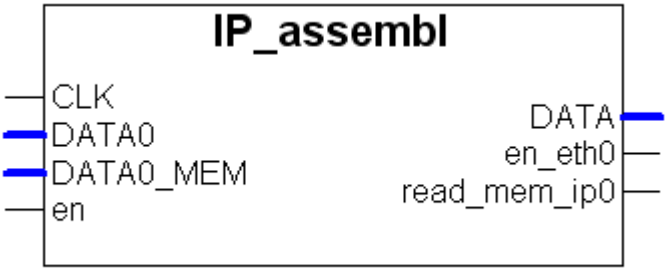


Figure 3.22 - bloc "IP_assembl" qui construit un paquet IP

La signification des entrées-sorties est similaire à celle des modules **TFTP** et **UDP.** Le map des entrées-sorties est présenté en Annexe 2. Vu la complexité de ce protocole on utilisera IPCores (Intellectual Property Core) TCP/IP tout prêt qui implémente toutes les fonctionnalités.

Au cadre de ce travail de diplôme nous n'allons pas implémenter ce protocole vu sa complexité. Il existe plusieurs IPCores (Intellectual Property Core) déjà implémentés. Xilinx qui est producteur des circuits Virtex II propose d'utiliser un IPCore pour la couche IP/TCP.

3.6 ETUDE ET IMPLEMENTATION DU PROTOCOLE ETHERNET II

La dernière couche à implémenter est la couche de protocole Ethernet II qui est d'ailleurs le plus simple.

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.

L'Ethernet est basé sur un principe de dialogue sans connexion et donc sans fiabilité. Les trames sont envoyées par l'adaptateur sans aucune procédure de type «handshake»[1] avec l'adaptateur destinataire. Le service sans connexion d'Ethernet est également non fiable, ce qui signifie qu'aucun acquittement, positif ou négatif, n'est émis lorsqu'une trame passe le contrôle CRC avec succès ou lorsque celle-ci échoue. Cette absence de fiabilité constitue sans doute la clé de la simplicité et des coûts modérés des systèmes Ethernet.

L'entête de protocole Ethernet II est présenté sur la Figure 3.23

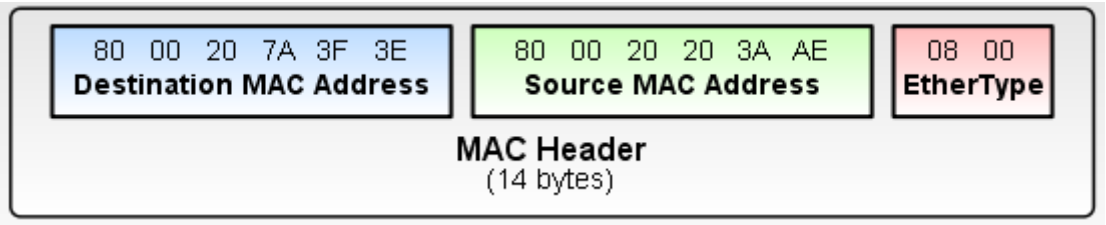


Figure 3.23 - l'entête du protocole Ethernet II[2]

De cette figure on peut tirer qu'on a deux champs à mémoriser, le champ Destination MAC Adresse et le champ Source MAC Adresse et un champ EtherType qui est une constante. EtherType est un champ d'une trame Ethernet indiquant quel est le

[1] - voir annexe 3
[2] - http://upload.wikimedia.org/wikipedia/commons/thumb/1/13/Ethernet_Type_II_Frame_format.svg/700px-Ethernet_Type_II_Frame_format.svg.png

protocole de niveau supérieur utilisé dans le champ "donnée" de cette trame. Il fait 2 octets. Sa valeur 0x0800 indique que le protocole de niveau supérieur est Internet Protocol, Version 4 (IPv4)

Les deux modules à implémenter sont semblables à ceux des protocoles TFTP, UDP et IP et sont présentés sur les figures 3.24 et 3.25

Figure 3.21 - bloc "ETH" qui traite les trames reçues

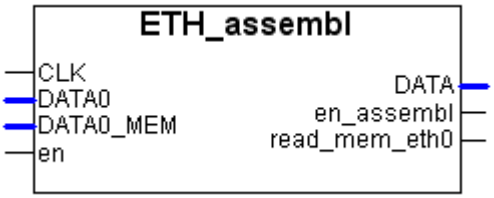


Figure 3.22 - bloc "ETH_assembl" qui construit un paquet Ethernet

Maintenant il nous reste que de connecter tous les modules ensemble. Le schéma du NAS obtenu est présenté en annexe 1. Plusieurs signaux qui n'ont pas la signification importante à ce niveau d'implémentation ont été omis.

L'étude et l'implémentation du NAS est un processus assez complexe et demande beaucoup de temps. Comme ce travail de diplôme est consacré à l'introduction à la plateforme netFPGA on s'arrête pour l'implémentation à ce niveau là. L'analyse du travail qui a été effectué ainsi que les différentes raisons pour lesquelles il n'a pas été terminé on va voir dans le chapitre 5 "Analyse du travail effectué".

4. TEST DE "GENERATEUR DES PAQUETS"

Ce projet nous fournit un simple générateur de paquets ainsi qu'un outil de capture. Il est conçu pour le but d'injecter des paquets dans un réseau et/ou d'observer des paquets provenant du réseau. La fonction de génération dans cette implémentation est un mécanisme de "playback". La séquence des datagrammes Ethernet est chargée dans le générateur depuis le fichier *.pcap le générateur à son tour continue alors à transmettre cette séquence des datagrammes. Le temps d'émission entre les datagrammes individuels peut-être fermement contrôlé; par défaut le générateur utilise le retard indiqué dans le fichier de source *.pcap. Un utilisateur peut spécifier facultativement un débit maximal des données et le temps d'émission entre les datagrammes.

La suite des étapes ci-dessous permettra de tester le générateur de paquets.

4.1 INSTALLATION

Installation depuis NetFPGA Yum repository

1. Installez le NetFPGA Base Package:
   http://netfpga.org/netfpgawiki/index.php/Guide#Install_NetFPGA_Base_Package
2. Installez le générateur des paquets

```
yum install netfpga-packet_generator
```

Après avoir installé le générateur des paquets il faut télécharger depuis le site NetFPGA la distribution beta du projet:
http://netfpga.org/beta/distributions/netfpga_packet_generator_1_0_2.tar.gz

4.2 USAGE

Exécutez la commande suivante:

```
packet_generator_setup.pl
```

1. Il faut s'assurer que le pilote de noyau de NetFPGA a été chargé et que le CPCI (Virtex II) a été programmé.
2. Charger le fichier binaire de générateur de paquets dans la FPGA:

```
nf2_download packet_generator.bit
```

4.3 GENERATION DES PAQUETS/CAPTURES

Le générateur est contrôlé via la commande *packet_generator.pl*

```
packet_generator.pl
      -q<queue number> <pcap file>
      [-r<queue number> <rate>] (Kbps)
      [-i<queue number> <number of iterations>]
      [-d<queue number> <delay between packets>] (ns)
      [-c<queue number> <capture file>

packet_generator.pl --help  - show detailed help
```

4.4 OPTIONS

Notez: le nombre de queue devrait être un nombre entre 0 et 3 inclus, conforme à nf2c0, nf2c1, nf2c2 et nf2c3.

- q <queue number> <pcap file>

spécifie le fichier *.pcap et l'envoie d'une queue;

-r <queue number> <rate>

spécifie le taux pour chaque queue en Kbps;

-i <queue number> <number of iteration>

spécifie le nombre d'itération pour chaque queue;

-d <queue number> <delay between packets>

spécifie le retard entre les paquets, en *ns*. Sinon on utilise le délai enregistré dans le fichier *.pcap utilisé.

-c <port number> <capture file>

spécifie le fichier pour la capture.

4.5 RESULTATS

L'exemple ci-dessous montre la génération et l'envoie les données de l'interface MAC2 à l'interface MAC3 directement connecté. Dans cet exemple le taux est limité à 1.000 Mbps:

```
Limiting MAC Queue 2 to 1.000 Mbps (tokens = 1, clks = 1000)
Loaded 1 packet(s) into MAC Queue 2
Limiting CPU Queue 0 to 200.000 Mbps (tokens = 1, clks = 5)
Limiting CPU Queue 1 to 200.000 Mbps (tokens = 1, clks = 5)
Limiting CPU Queue 2 to 200.000 Mbps (tokens = 1, clks = 5)
Limiting CPU Queue 3 to 200.000 Mbps (tokens = 1, clks = 5)
Sending packets...
Last packet scheduled for transmission at 12.132 seconds
12 seconds elapsed...

Transmit statistics:
====================

MAC Queue 2:
        Packets: 1000
        Completed iterations: 1000


Receive statistics:
===================

MAC Queue 0:
        Packets: 0
MAC Queue 1:
        Packets: 0
MAC Queue 2:
        Packets: 0
MAC Queue 3:
        Packets: 1000
        Bytes: 1518000
        Time: 12.131868144 s
        Rate: 1.001 Mbps
```

Par les calculs on estime que le dernier paquet est transmis au temps 12.144 secondes, en sachant la taille de paquet qui vaut 1518 octets et le nombre d'itération.

La statistique pour la queue MAC 2 (nf2c2) nous indique que 1000 paquets ont été envoyés et 1000 itérations ont été accomplies. La statistique de réception pour la queue MAC 3 (nf2c3) indique que 1000 paquets ont été reçu que nous donne au total

1518000 octets. Le temps entre la réception du premier octet et celui le denier est de 12.132 secondes environs. Cela correspond à un taux de 1.001 Mbps.

4.6 EXEMPLE

1. Pour envoyer les paquets contenus dans un fichier *http.pcap* (que vous trouverez dans le projet fourni avec la mémoire: /Projects/netfpga_packet_generator/NF2/projects/packet_generator/sw) vers l'interface *nf2c0* il faut exécuter la commande suivante:

```
packet_generator.pl -q0 http.pcap
```

2. Faire 10 itérations d'envoie des paquets contenus dans un fichier http.pcap sur l'interface *nf2c0*. Capturer les données reçus sur le port *nf2c1* dans un fichier *nf2c1.pcap* :

```
packet_generator.pl -q0 http.pcap -i0 10 -c1 nf2c1.pcap
```

3. Envoyer des paquets contenus dans un fichier *http.pcap* via l'interface *nf2c0*. Limiter le taux à 1 Mbps (1000 Kbps):

```
packet_generator.pl -q0 http.pcap -r0 1000
```

4. Envoyer les paquets contenus dans un fichier *http.pcap* via l'interface *nf2c0*. Mettre le délai entre l'envoie de deux paquets consécutives à 0 *ns*.

```
packet_generator.pl -q0 http.pcap -d0 0
```

5. Envoyer les paquets contenus dans les fichiers *http0.pcap*, *http1.pcap*, *udp2.pcap*, *udp3.pcap* sur les interfaces de sorties *nf2c0*, *nf2c1*, *nf2c2*, *nf2c3* respectivement. Capturer les données reçues sur les mêmes quatre ports dans un fichier *nf2c0.pcap*, *nf2c1.pcap*, *nf2c2.pcap*, *nf2c3.pcap*:

```
packet_generator.pl -q0 http0.pcap -q1 http1.pcap
                    -q2 udp2.pcap -q3 udp3.pcap
                    -c0 nf2c0.pcap -c1 nf2c1.pcap
                    -c2 nf2c2.pcap -c3 nf2c3.pcap
```

4.7 LES COMMENTAIRES

Toutes les mesures sont exécutées en matériel. Le matériel envoie tous les paquets reçus au niveau supérieur qui est un logiciel qui reçoit ces paquet, les analyse et enfin les sauvegarde dans un fichier de capture *.pcap Comme la bande passante du bus PCI est moins performante que celle des quatre ports GigabitEthernet il est donc possible que quelque paquets reçus soient perdus. La perte des paquets se produise lorsque les tampons sur la carte FPGA débordent.

5. ANALYSE DU TRAVAIL EFFECTUE DURANT LA PERIODE SEMESTRIELLE

Le début de mon projet a été consacré à l'apprentissage du langage de programmation hardware Verilog. Pour cela j'ai utilisé un livre "Verilog HDL: Digital Design and Modeling" [2]. Après avoir acquis ce langage et rédigé un manuel[1] j'ai poursuivi par l'étude et la compréhension du projet "Référence Routeur" qui est d'ailleurs fourni avec la plateforme NetFPGA. J'ai étudié plusieurs librairies Verilog pour pouvoir les utiliser plus tard. Le projet "Référence Routeur" contient 30000 lignes de code Verilog environ et en plus environ 10000 lignes de code en C, ceci sans avoir tenu compte des fichiers écrits en Perl5, Python, Java et scripts bash qui servent à contrôler la carte, exécuter les différents tests et obtenir les résultats via interface graphique.

Deuxièmement il a fallu préparer un poste avec le système Linux comprenant des pilotes installés ainsi que les outils de développement[2].

Le 2 juin j'ai reçu la plateforme NetFPGA et à partir de ce moment là j'ai pu commencer à tester cette plateforme et travailler avec le projet "Générateur des paquets" qui a été présenté au Chapitre 4.

Voici quatre remarques importantes par rapport à la réalisation d'un dispositif complet et fonctionnel:

1. Comme déjà mentionné au Chapitre 3, le circuit FPGA Virtex II qui est le cœur de notre plateforme, contient un contrôleur RocketIO. Ce contrôleur sert à gérer les deux interfaces SATA. Après avoir étudié les datasheet de l'IPCore SATA, j'ai découvert que ce denier est conçu pour être utilisé seulement avec les circuits programmables Virtex 4 et Virtex 5. Donc le concept de stockage sur les disques durs directement attachés à la carte a été exclu.
2. Je suis passé à la deuxième solution qui me semblait plausible, consistant à stocker les données reçues via 4 interfaces GigabitEthernet sur le disque dur de notre poste Linux. J'étais prêt à admettre une réduction du débit de transmission des données via un bus PCI. Cependant, cette solution s'est révélée impraticable puisque le FPGA Spartan qui est programmé par le

[1] - voir annexe 6
[2] - voir annexe 5

producteur de la carte NetFPGA cache sa structure interne qui implémente le contrôleur DMA. Et par conséquent je ne pouvais pas gérer les entrées-sorties et les interruptions via ce bus.

3. Depuis le "répository" netFPGA de Stanford Université, j'ai pu obtenir 4 projets fonctionnels avec leur code source. Cela a été fait pour pouvoir étudier ce code source et le recompiler pour s'assurer du bon fonctionnement des outils de développement ainsi que de la faisabilité du projet. Mais ici j'ai aussi essuyé un échec. Il se trouve que les répertoires de code source Verilog ne contiennent pas tous les fichiers qu'il faut pour la compilation.
4. Le dernier point ayant influencé la réalisation du projet a été le retard avec lequel j'ai reçu la plateforme.

Malgré tous ces problèmes, le projet était très intéressant et m'a permis d'approfondir les connaissances du langage Verilog, des systèmes de stockage SAN et NAS et, enfin, de découvrir la plateforme NetFPGA.

Après avoir travaillé sur ce projet et avoir acquis une certaine expérience sur un projet d'une telle complexité, j'aimerais proposer des améliorations au niveau de l'organisation du travail pour permettre, dans le futur, d'obtenir de meilleurs résultats au terme fixé. Les connaissances en Verilog, du C, Perl, système Linux doivent être préalablement acquises. Toute la documentation sur la plateforme netFPGA ainsi que le code source complet doivent être présentes. Le poste Linux doit être préalablement installé et configuré.

1. Commencer le travail par l'étude du projet "Référence routeur" qui comprend l'exécution des différents tests[1]
2. Installer les outils de développement présents sur la liste en Annexe 5
3. Essayer de crée un nouveau projet Xilinx en utilisant les manuels présents dans le répository de ce projet.
4. L'étape suivant consiste à compiler ce projet et de faire le placement et du routage pour le circuit Virtex II
5. Après avoir obtenu le fichier binaire essayer de le charger sur la carte NetFGPA
6. Passer les "Selftests" avant de pouvoir obtenir les résultats cohérents à ceux représentés sur le site de NetFPGA
7. Etudier et maitriser tous les protocoles à implémenter.

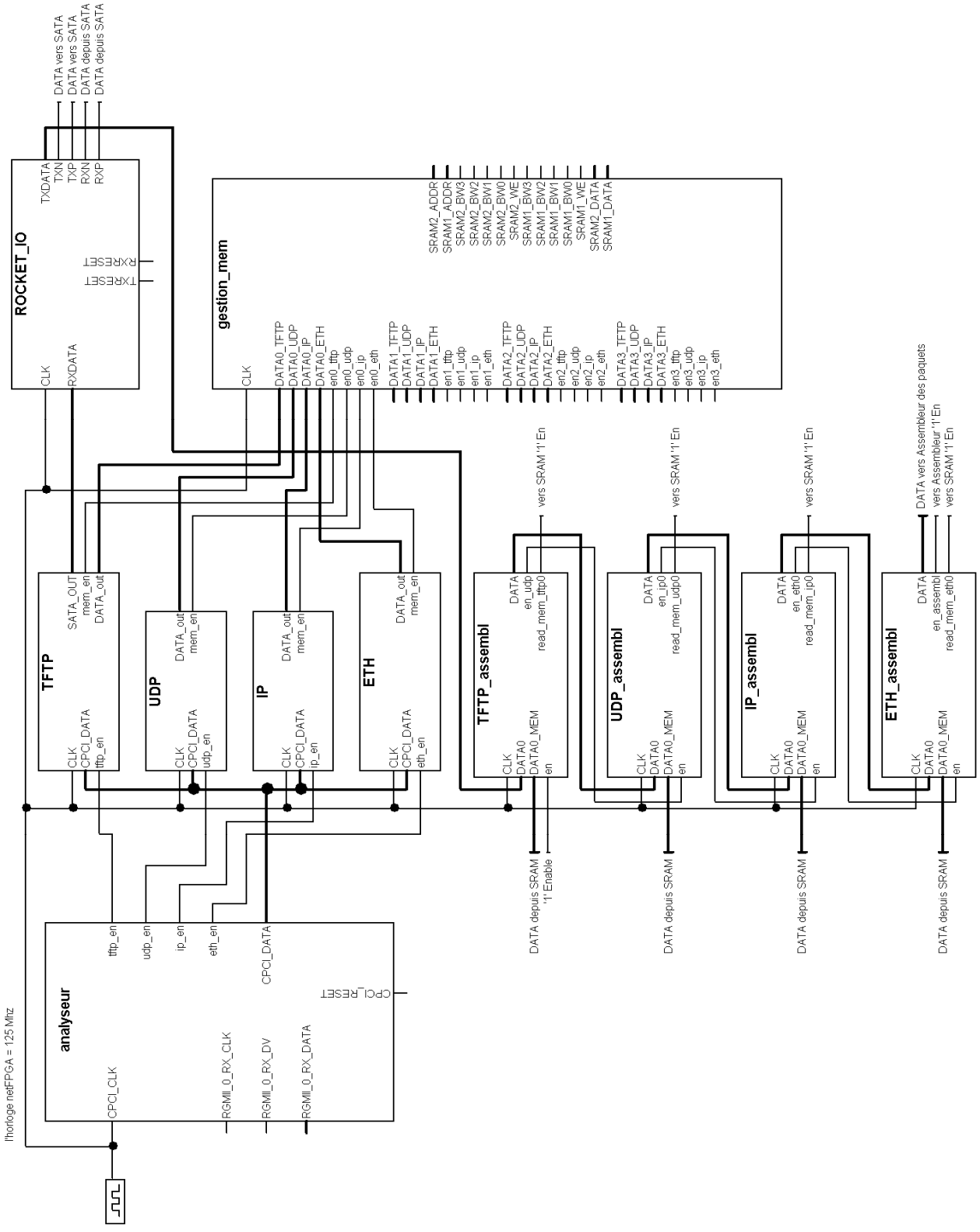
[1] - www.netfpga.org

8. Développer la conception du NAS
9. En utilisant une technique "top-down" proposer sa solution sous forme d'une "boite noir" avec les entrées et sorties bien définies
10. Définir les différents blocs qui sont inclus dans "boite noir" ainsi que les fonctionnes qu'ils réalisent
11. Décrire en Verilog le comportement de chaque bloc défini préalablement ainsi que les Testbench[2] qui permettent de vérifier le bon fonctionnement de chaque bloc
12. Fusionner tous les blocs en un module qu'on appelle Hardware Network Area Storage
13. Ce point est facultatif et propose d'écrire les programmes en C permettant de tester toutes les fonctions qui sont implémentées dans NAS. Tous les tests s'exécutent à l'aide des scripts fait en Perl5

6. CONCLUSION

La conclusion la plus évidente que l'on peut émettre à propos de ce travail de diplôme est que malgré le travail effectué, il reste encore beaucoup de travail à faire pour rendre le système NAS fonctionnel et efficace. Ce projet est très large au point de vu technique. Il touche les domaines de télécommunication, la programmation, les systèmes numériques et les systèmes de stockage. La technologie FPGA pourrait représenter une opportunité sur le marché des NAS en offrant de nouvelles capacités et de meilleures performances en taux de transfert et de stockage.

ANNEXE 1 - SCHEMA DU NAS



ANNEXE 2 - MAP DES ENTREES-SORTIES

Table des entrées-sorties		
Module	Entrées-sorties	Description
TFTP	CLK	l'horloge de netFPGA = 125 Mhz
	CPCI_DATA	données reçues de decoupleur des paquets, 64 bits
	tftp_en	l'activation de bloc TFTP = '1' est actif
	SATA_OUT	les données extraites pour sauvegarder dans SRAM
	mem_en	l'activation de gestionnaire de mémoire
	DATA_OUT	données à mémoriser dans SRAM, 64 bits
TFTP_assembl	DATA0 - DATA3	les données, provenant de RocketIO, 64 bits
	DATA0_MEM - DATA3_MEM	les données provenant de SRAM, 64 bits
	en	l'activation du bloc TFTP_assembl, '1' est actif
	DATA	paquet TFTP assemblé, 64 bits
	en_udp0	l'activation d'un bloc UDP_assembl, '1' est actif
	read_mem_tftp0 - read_mem_tftp3	l'activation de la lecture des données depuis SRAM, '1' est actif
UDP	CPCI_DATA	données reçues de decoupleur des paquets, 64 bits
	udp_en	l'activation du bloc UDP, '1' est actif
	DATA_OUT	les données extraites pour sauvegarder dans SRAM
	mem_en	l'activation de gestionnaire de mémoire
UDP_assembl	DATA0 - DATA3	les données, provenant du bloc TFTP_assembl, 64 bits
	DATA0_MEM - DATA3_MEM	les données provenant de SRAM, 64 bits
	en	l'activation du bloc UDP_assembl, '1' est actif
	DATA	paquet UDP assemblé, 64 bits
	en_ip0	l'activation d'un bloc IP_assembl, '1' est actif
	read_mem_udp0 - read_mem_udp3	l'activation de la lecture des données depuis SRAM, '1' est actif
IP	CPCI_DATA	données reçues de decoupleur des paquets, 64 bits
	ip_en	l'activation du bloc IP, '1' est actif
	DATA_OUT	les données extraites pour sauvegarder dans SRAM
	mem_en	l'activation de gestionnaire de mémoire
IP_assembl	DATA0 - DATA3	les données, provenant du bloc UDP_assembl, 64 bits
	DATA0_MEM - DATA3_MEM	les données provenant de SRAM, 64 bits
	en	l'activation du bloc IP_assembl, '1' est actif
	DATA	paquet IP assemblé, 64 bits
	en_eth0	l'activation d'un bloc ETH_assembl, '1' est actif
	read_mem_ip0 - read_mem_ip3	l'activation de la lecture des données depuis SRAM, '1' est actif
ETH	CPCI_DATA	données reçues de decoupleur des paquets, 64 bits
	eth_en	l'activation du bloc ETH, '1' est actif
	DATA_OUT	les données extraites pour sauvegarder dans SRAM
	mem_en	l'activation de gestionnaire de mémoire

La suite du tableau MAP des entrées-sortie

Table des entrées-sorties		
Module	Entrées-sorties	Description
ETH_assembl	DATA0 - DATA3	les données, provenant du bloc IP_assembl, 64 bits
	DATA0_MEM - DATA3_MEM	les données provenant de SRAM, 64 bits
	en	l'activation du bloc ETH_assembl, '1' est actif
	DATA	paquet IP assemblé, 64 bits
	en_assembl	l'activation d'un bloc d'asseblage, '1' est actif
RocketIO	RXDATA	bus de données reçues, 64 bits
	TXDATA	bus de données pour envoyer, 64 bits
	TXN, TXP	données envoyées vers les disques durs, codées en SATA
	RXN, RXP	données provenant des disques durs, codées en SATA
Analyseur	RGMII_0_RX_CLK	signal de syncronisation vers le disque durs
	RGMII_0_RX_DV	le bit de validité pour les données reçues
	RGMII_0_RX_DATA	données reçues provenant depuis le contrôleur PHY, 4 bits
	CPCI_DATA	données extraite du datagramme reçu, 64 bits
	tftp_en	activation d'un module TFTP, '1' est actif
	udp_en	activation d'un module UDP, '1' est actif
	ip_en	activation d'un module IP, '1' est actif
	eth_en	activation d'un module ETH, '1' est actif
gestion_mem	DATA0_TFTP - DATA3_TFTP	données à mémoriser, provenant d'une couche TFTP
	DATA0_UDP - DATA3_UDP	données à mémoriser, provenant d'une couche UDP
	DATA0_IP - DATA3_IP	données à mémoriser, provenant d'une couche IP
	DATA0_ETH - DATA3_ETH	données à mémoriser, provenant d'une couche ETHERNET
	SRAM1_ADDR - SRAM2_ADDR	bus d'adressage, 20 bits
	SRAM1_DATA - SRAM2_DATA	bus de données, 36 bits
	SRAM1_WE - SRAM2_WE	Write Enable Input
	SRAM1_BW0 - SRAM1_BW3	les bits de synchronisation pour le module 1 SRAM
	SRAM2_BW0 - SRAM2_BW3	les bits de synchronisation pour le module 2 SRAM

ANNEXE 3 - DICTIONNAIRE INFORMATIQUE

Active Directory - est la mise en œuvre par Microsoft des services d'annuaire pour une utilisation principalement destinée aux environnements Windows. L'objectif principal d'Active Directory est de fournir des services centralisés d'identification et d'authentification à un réseau d'ordinateurs utilisant le système Windows. Il permet également l'attribution et l'application de stratégies, la distribution de logiciels, et l'installation de mises à jour critiques par les administrateurs.

Apple Filing Protocol (AFP) - AppleShare est un protocole de partage de fichier utilisé sur Macintosh. Ce protocole fait partie de la couche présentation (ou sixième couche) du modèle OSI, permettant un échange de fichiers pour les ordinateurs équipés des systèmes d'exploitation Apple Mac OS X ou Classic, tout comme les protocoles SMB, NFS, FTP ou WebDAV.Il supporte actuellement l'encodage Unicode pour les noms de fichiers, les permissions POSIX et les listes de contrôles d'accès, les quotas utilisateurs d'UNIX. Ce fut le seul protocole d'échange de fichiers natif sur Mac OS 9, alors que

Common Internet File System - anciennement Server Message Block (SMB) est un protocole réseau permettant principalement de partager des fichiers, mais aussi des imprimantes, des ports séries et d'autres types de communications que l'on peut avoir entre différents ordinateurs d'un réseau. Il est principalement utilisé par les ordinateurs équipés de Microsoft Windows. Dans la suite de l'article, on parlera de SMB, plus utilisé que CIFS.

Courseware - Logiciel interactif destiné à l'enseignement ou à l'apprentissage, et pouvant inclure un contrôle de connaissance

Direct Memory Access - L'accès direct à la mémoire ou DMA (sigle anglais de Direct Memory Access) est un procédé informatique où des données circulant de ou vers un périphérique (port de communication, disque dur) sont transférées directement par un contrôleur adapté vers la mémoire principale de la machine, sans intervention du microprocesseur si ce n'est pour initier et conclure le transfert. La conclusion du transfert ou la disponibilité du périphérique peuvent être signalés par interruption.

Fail-over - Le basculement (en anglais, fail-over qui se traduit par passer outre à la panne) est la capacité d'un équipement à basculer automatiquement vers un chemin réseau alternatif ou en veille.

First In, first Out - L'acronyme FIFO est l'abréviation de l'expression anglaise First In, first Out, que l'on peut traduire par « premier arrivé, premier servi » (littéralement « premier entré, premier sorti »). Cette méthode correspond à une méthode de traitement des éléments d'une file (calculs d'un ordinateur, stocks).

Handshake - Séquence de messages ou de signaux échangés entre deux ou plusieurs appareils afin d'assurer la synchronisation de la transmission des données.

IP Core - IP (la propriété intellectuelle). Core - est un bloc de données logiques (numériques) qui est utilisé pour programmer un circuit FPGA ou ASIC au but de réaliser un produit spécifique.

Mac OS X permet de base d'utiliser ses concurrents SMB et FTP.

NetWare Core Protocol - Protocole propriétaire conçu par Novell dans le but de simplifier l'accès aux fonctions des services d'impression et de fichiers de NetWare. Il utilise les couches réseaux IPX ou IP.

Network File System - Le système de fichiers en réseau (Network File System ou NFS) est un protocole développé par Sun Microsystems qui permet à un ordinateur d'accéder à des fichiers via un réseau. Ce système de fichiers en réseau permet de partager des données principalement entre systèmes UNIX. Des implémentations existent pour Macintosh ou Microsoft Windows.NFS est compatible avec IPv6 sur la plupart des systèmes.

Perl - Perl est un langage de programmation créé par Larry Wall en 1987 et reprenant des fonctionnalités du langage C et des langages de scripts shell (sh). Dans ce projet il est largement utilisé pour exécuter les tests de différentes fonctions implémenté dans NetFPGA

self-learning - pour un switch par exemple, c'est le mode de fonctionnement en auto-apprentissage.

Timestamping - l'horodatage électronique est une technique qui permet de prouver l'existence d'un document avant un instant précis.

Virtual LAN - Un réseau virtuel, communément appelé VLAN (pour Virtual LAN), est un réseau informatique logique indépendant. De nombreux VLAN peuvent coexister sur un même commutateur réseau (switch).

ANNEXE 4 - EXEMPLE DU CODE PERL POUR TESTER LA CHECHKSUM

```
#!/usr/bin/perl
# Author: Evgenij Sviridovitch
# Date: 11/06/2009

# Objective:
# Garantire que les paquets routés sont un checksum correcte

use Error qw(:try);
use IO::Socket;
use NF2::TestLib;
use NF2::PacketLib;
use strict;

chdir '../../sw' or die "Can't cd: $!\n";

my $pid;

# Fork d'un processus Checksum
if ( !( $pid = fork ) ) {
    # Run checksum dans ce processus
    exec "./scone", "-r", "rtable.netfpga";
    die "Erreur d'execution: $!";
} else {
    my $exitCode = 1;
    try {
        # Execution du contrôle dans ce processus

        # Attente que le routeur s'initialise
        sleep(1);

        # execution de PCAP sur eth1, eth2
        my @interfaces = ( "eth1", "eth2" );
        nftest_init( \@ARGV, \@interfaces );
        nftest_start_vhosts( \@interfaces );

        # Registration des IP adresses
        nftest_register_router('eth1', '00:00:00:00:00:01', '192.168.0.2');
        nftest_register_router('eth2', '00:00:00:00:00:02', '192.168.1.2');

        # Registration des hôtes
        nftest_create_host('eth1', 'aa:bb:cc:dd:ee:f0', '192.168.0.100');
        nftest_create_host('eth2', 'ca:fe:f0:0d:00:00', '192.168.1.100');
```

```
        # Envoie des paquets
        my $pkt = nftest_send_IP('192.168.0.100', '192.168.1.100', len => 100);

        # Attante des paquets
        nftest_expect_ARP_exchange('192.168.0.100', '192.168.0.2');
        nftest_expect_ARP_exchange('192.168.1.2', '192.168.1.100');

        $pkt->set(
            SA => nftest_get_vhost_mac('192.168.1.2'),
            DA => nftest_get_vhost_mac('192.168.1.100')
            );
        $pkt->decrement_ttl;
        nftest_vhost_expect('192.168.1.100', $pkt->packed);

        sleep 5;

        # La fin et l'impréssion des erreurs
        nftest_finish();
        my $total_errors = nftest_print_vhost_errors();

        if ( $total_errors == 0 ) {
            print "SUCCESS!\n";
            $exitCode = 0;
        } else {
            print "FAIL: $total_errors errors\n";
            $exitCode = 1;
        }
    } catch Error with {
        # Le catch de n'importe quelle erreur et son impréssion
        my $ex = shift;
        if ($ex) {
            print $ex->stringify();
        }
    } finally {
        # Assurer que le processuis est tué même si'il y a des erreurs
        kill 9, $pid;
        # La sortie avec des résultats
        exit($exitCode);
    };
}
```

ANNEXE 5 - OUTILS DE DEVELLOPEMENT

1. Xilinx: ISE Foundation, Version: 9.2i SP4. Service Pack 4. IP Update 2

Permet de crée les projets Verilog, VHDL

2. Mentor Graphics: ModelSim. Version SE 6.2G

La simulation numérique.

3. Xilinx: ChipScope Pro. Version 9.1.02i

Logiciel de débogage.

4. A votre choix, un outil de développement en JAVA

5. A votre choix, un outil de développent Perl5

REFERENCES

1. NAS et SAN. Solution de stockage, sécurité, infrastructures.
Xavier Bouchet, Henri Gillarès-Caillat, Dunod, InfoPro, Broché, 2003, 210 pages

2. Verilog HDL: Digital Design and Modeling.
Cavanagh Joseph, CRC Press. 2003, 900 pages

3. Digital Design An Embedded Systems Approach. Using Verilog
Peter J. Ashenden, 2008, 541 pages

4. http://monge.univ-mlv.fr/~duris/NTREZO/20022003/SAN_NAS.pdf
"LES SOLUTIONS DE STOCKAGE EN RESEAU LE SAN CONTRE LE NAS"

5. http://monge.univ-mlv.fr/~duris/NTREZO/20052006/KomarLeCamMancel-SAN-NAS-xpose.pdf "NAS – SAN Les nouvelles solutions de stockage"

6. http://www.zdnet.fr/actualites/informatique/0,39040745,2108230,00.htm
Comparatif NAS et SAN, Ch. 1 de rapport

7. http://www.clubic.com/article-151126-1-stockage-reseau-nas.html
Stockage Réseau NAS. Ch. 1 de rapport

8. http://www.nasfr.com/

9. http://www.digital-storage.fr/stockage-nas.php

10.http://www.nasstor.net/?name=raid
(description RAID)

11. http://www.supinfo-projects.com/fr/2006/rapport_fr_2005-2006/1/
NAS et SAN

12. RFC1350 protocole TFTP

13. RFC768 - protocole UDP

14. RFC791 - protocole IP

15.http://www.digilentinc.com/Products/Detail.cfm?Prod=NETFPGA&Nav1=Products&Nav2=Programmable

16. http://www.xilinx.com/tools/webpack.htm

17. http://www.xilinx.co/publications/3rd_party/products/ASICSWS_SATA_H1.pdf
Alliance Core. ASICS Worl Services, LTD