Hes·so // VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Domaine Sciences de l'ingénieur
Rte du Rawyl 47
CH- 1950 Sion 2
Tél. +41 27 606 85 11
Fax +41 27 606 85 75
info@hevs.ch

www.hevs.ch

Filière Systèmes industriels

Orientation Infotronics

Diplôme 2011

François Héritier

VLAN en FPGA

Professeur	François Corthay
Expert	Claude Magliocco

Sion, le 4 août 2011

HES-SO Valais

SI	TV
X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.AB
pof/31/01/2009

☒ FSI ☐ FTV	Année académique / *Studienjahr* **2010/2011**	No TD / *Nr. DA* **it/2011/48**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **François Héritier** Professeur / *Dozent* **François Corthay**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte* *(données complètes)* **Claude Magliocco**	

Titre / *Titel*

VLAN en FPGA

Description et Objectifs / *Beschreibung und Ziele*

Le but de ce projet est d'implémenter le standard Ethernet VLAN de réseau local virtuel sur FPGA.

Un réseau local virtuel (Virtual LAN, VLAN) est un réseau informatique logique indépendant. Une partie de la trame Ethernet code l'adresse de réseau virtuel et des appareils connectés physiquement sur le même réseau physique ne voient que les autres appareils du même réseau virtuel. Ce type de réseau virtuel est notamment utilisé par les fournisseurs d'énergie pour le contrôle des stations et de leurs équipements.

Un bloc d'interface Ethernet en FPGA a déjà été développé : il est capable de recevoir des trames Ethernet et d'en envoyer. Ce circuit est modulable : il permet d'implémenter plusieurs protocoles basés sur Ethernet (DHCP, ICMP, UDP). Le but de ce projet est d'ajouter au système le décodage de l'information de réseau local virtuel VLAN pour permettre de limiter son échange d'informations à un réseau virtuel.

Les objectifs du projet sont de :

— réaliser un bloc implémentant le protocole VLAN, l'ajouter au système déjà existant et le tester

— réaliser une application de démonstration du système.

Délais / *Termine*

Attribution du thème / *Ausgabe des Auftrags:* 06.06.2011	Exposition publique / *Ausstellung Diplomarbeiten:* 02.09.2011
Remise du rapport / *Abgabe des Schlussberichts:* 04.08.2011 | 12h00	Défense orale / *Mündliche Verteidigung:* dès la semaine 36 / *ab Woche 36*

Signature ou visa / *Unterschrift oder Visum*

Responsable de l'orientation
Leiter der Vertiefungsrichtung: .....

[1] Etudiant/*Student*: .....

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive et le caractère confidentiel du travail de diplôme qui lui est confié et des informations mises à sa disposition.*
Durch seine Unterschrift verpflichtet sich der Student, die Richtlinie einzuhalten sowie die Vertraulichkeit der Diplomarbeit und der dafür zur Verfügung gestellten Informationen zu wahren.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

VLAN en FPGA

Travail de diplôme
| édition 2011 |

Diplômant/e François Héritier

Objectif du projet

Le but de ce projet est d'implémenter le standard Ethernet VLAN de réseau local virtuel sur FPGA. Une application de démonstration du système est réalisée.

Méthodes | Expériences | Résultats

La carte peut recevoir des paquets et y répondre. Si elle reçoit un paquet VLAN, elle y répond par un paquet VLAN. Si le paquet n'est pas de type VLAN, la réponse ne l'est pas non plus. Toutefois le mode VLAN à l'émission peut être forcé sur simple pression d'un bouton.

Pour l'application de démonstration, un contrôleur est ajouté. Il permet l'envoi de trames ''Notify'' selon le protocole SSDP. Il permet donc à la carte de s'annoncer sur un réseau en fournissant son adresse IP et un numéro de port.
Un deuxième contrôleur attend une requête sur le port donné dans le ''Notify''. Une fois la première requête reçue il envoie les valeurs données par un convertisseur analogique/numérique annexe. Ces valeurs peuvent être lues sur une page web.
La carte FGPA et le bloc d'interface Ethernet ont été fournis en début de projet. Ce circuit supporte déjà les protocoles DHCP, ICMP, UDP et ARP.

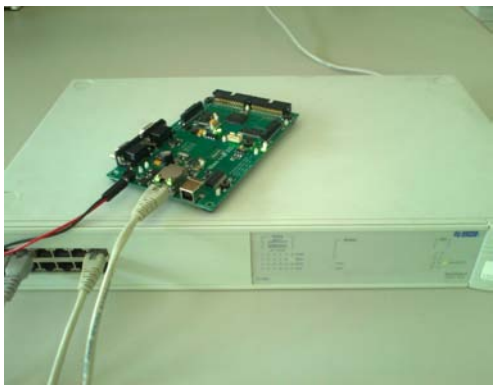
Le circuit a été testé en mode VLAN et non VLAN et se comporte de manière correcte. Il peut aussi émettre des ''Notify'' dans un intervalle de temps défini. Il peut aussi lire des données extérieures et les envoyer via le protocole UDP.

Filière
Systèmes industriels

Domaine d'application
Infotronics

Professeur responsable
Corthay François
francois.corthay@hevs.ch

Partenaire
-



Carte FPAG et
Switch programmable.

HES-SO Valais
Route du Rawyl 47
1950 Sion

Tél. 027 606 85 11
URL www.hevs.ch

Table des matières

1. Introduction

Le projet, qui a pour but d'implémenter le standard Ethernet VLAN sur FPGA, reprend un circuit développé par différentes personnes afin de l'améliorer.

Le circuit existant comprend un bloc d'interface Ethernet. Il est programmé dans le langage VHDL et peut être implémenté dans une FPGA (Field-Programmable Gate Array ou réseau de portes programmables).

Les FPGA sont des circuits logiques programmables qui contiennent une grande quantité de blocs logiques. Ces blocs sont connectés entre eux et peuvent être facilement reconfigurés.

Le bloc Ethernet permet de gérer différents protocoles basées sur Ethernet (DHCP, ICMP, UDP, ARP). Il permet d'envoyer et recevoir des trames Ethernet.

Le protocole VLAN (Virtual LAN) est une option possible lors de l'envoi de trames Ethernet. Il permet que dans un même réseau physique, seuls certains périphériques se voient sans voir les autres, aussi présents physiquement. Et ces autres périphériques ne peuvent pas les voir non plus. Il crée donc sur un réseau physique plusieurs réseaux virtuels indépendants les uns des autres. Ces réseaux virtuels sont différenciés par un numéro.

2. Cahier des charges

Les objectifs de ce projet sont de:

1. Réaliser un bloc implémentant le protocole VLAN, l'ajouter au système déjà existant et le tester.
    - Modification du bloc Ethernet de réception
    - Modification du bloc Ethernet d'envoi
    - Ajout d'un sélecteur VLAN dans chaque contrôleur
    - Modification de la couche Memory
    - Correction des erreurs dans le DHCP
    - Ajout du bouton extérieur pour forcer en mode VLAN

2. Réaliser une application de démonstration du système.
    - Création d'un nouveau contrôleur qui émet des « Notify » à l'aide du protocole SSDP dans un certain intervalle de temps.
    - Création d'un deuxième contrôleur qui émet des données lues sur une carte a/d extérieure, dans un certain intervalle de temps, à partir du moment où une requête a été reçue, à l'adresse ayant fait la requête.
    - Utilisation du contenu du Notify pour faire une requête depuis un PC à l'adresse et au port reçu.
    - Utilisation de la requête comme start pour le deuxième contrôleur et réception des données en retour.
    - Sauvegarde de ces données sur un serveur pour les lire ensuite sur une page web.

Amélioration personnelle :
Le circuit reçu devant être modifié lors du passage de la simulation à l'implémentation et inversement. A chaque changement d'un mode à l'autre, il fallait faire deux modifications dans la partie DHCP.

- Amélioration du circuit pour éviter ces changements.

3. Protocole VLAN

Le protocole VLAN (Virtual LAN) est défini par le standard IEEE 802.1Q. Il fournit un mécanisme d'encapsulation et définit le contenu d'une balise VLAN. Cette balise appelée aussi tag sera insérée dans l'en-tête de la trame Ethernet. Elle ajoute ainsi 4 octets à la trame d'origine et doit recalculer le FCS (ou CRC).

Trame Ethernet

Adresse MAC dst.	Adresse MAC src.	Len/Etype	Data	FCS

Trame Ethernet avec balise VLAN

Adresse MAC dst.	Adresse MAC src.	**Tag (inséré)**	Len/Etype	Data	**FCS (modifié)**

Description du Tag

Le champ tag contient différentes choses : On y trouve deux champs de 16 bits, le 1er s'appelle TPID et le 2ème TCI.

TPID (16 bits)	TCI (16 bits)

3.1 TPID

Les 16 bits du champ TPID servent à identifier le protocole de la balise insérée. Pour le VLAN, il est défini à 0x8100. Ce champ est comparable au champ Ethertype d'une trame Ethernet : ils font les deux 16 bits et suivent l'adresse MAC source.

3.2 TCI

Le champ TCI contient 3 champs : un premier pour la priorité (3 bits), un deuxième appelé CFI (1 bit) et le troisième qui est le VLAN ID (12 bits).

Priority (3 bit)	CFI (1 bit)	Vlan ID, VID (12 bit)

3.2.1 Priorité

Le champ priorité sur 3 bits permet de coder 8 niveaux différents de 0 à 7. Ce champ fait référence au standard IEEE 802.1p. Selon lui, le 0 est la priorité la plus basse et le 7, la plus haute. Ces niveaux sont utilisés pour fixer les priorités d'une trame VLAN par rapport à une autre.

3.2.2 CFI

Le CFI (Canonical Format Identifier) est un champ de 1 bit qui permet d'assurer la compatibilité entre Adresse MAC Ethernet et Token Ring.
Exemple : un commutateur Ethernet le fixera toujours à 0. Si un port Ethernet reçoit la valeur 1, la trame ne sera pas propagée.

3.2.3 VLAN ID

Le champ VLAN ID sur 12 bits sert à identifier à quel Virtual Lan une trame appartient. Il permet de coder 4094 VLAN différents. L'ID 0xFFF est réservé.

4. Réalisation du bloc

4.1 Fonctionnement

Les 16 premiers bits du tag VLAN (TPID) qui sont insérés dans la trame permettent d'identifier le type du paquet. Cela correspond au champ Len/Etype d'une trame Ethernet normale. Voila quelques valeurs possibles : 0x0800 : (IPv4), 0x86DD : (IPv6), 0x0806 : (ARP), 0x8035 : (RARP), **0x8100 : (VLAN).**
Grâce à ce champ, on va pouvoir détecter la présence d'une trame VLAN ou non.

Dans le cas d'une trame non VLAN, son traitement reste identique à celui réalisé jusqu'à maintenant.

Et si la trame est un VLAN (voir annexe 1, page 50), la priorité et le CFI vont lui être extraits, et transmis à l'utilisateur. Ces 4 bits ne nous intéressent pas directement.

La vérification du VLAN ID peut ensuite se faire pour savoir si le paquet VLAN appartient bien au réseau défini sur la carte.
Si c'est le cas, il sera renvoyé à l'entrée du circuit, son type sera alors à nouveau vérifié, si le paquet est de type IPv4, ARP, etc. Celui-ci sera ensuite traité comme un paquet normal jusqu'à l'émission.

Pour l'émission (voir annexe 2, page 51), on veut évidemment retourner sur le réseau un paquet VLAN dans le cas d'un paquet reçu en VLAN.
On va donc regarder le type du paquet à envoyer. Si un tag VLAN doit être ajouté, on va commencer par ajouter le VLAN ID puis le CFI et la priorité.

Une fois les tags du VLAN ajoutés, y compris le TPID, le paquet va être émis comme un paquet non VLAN.

4.2 Principe d'implémentation

Malgré que l'identification d'une trame VLAN se fasse comme pour l'IPv4 ou l'ARP, le VLAN appartient à la couche 2 (**Figure 1**), soit Ethernet, et non pas à la couche 3, comme ces deux protocoles. C'est pourquoi aucun nouveau bloc ne sera ajouté, mais une modification du bloc Ethernet va permettre la détection des trames VLAN.

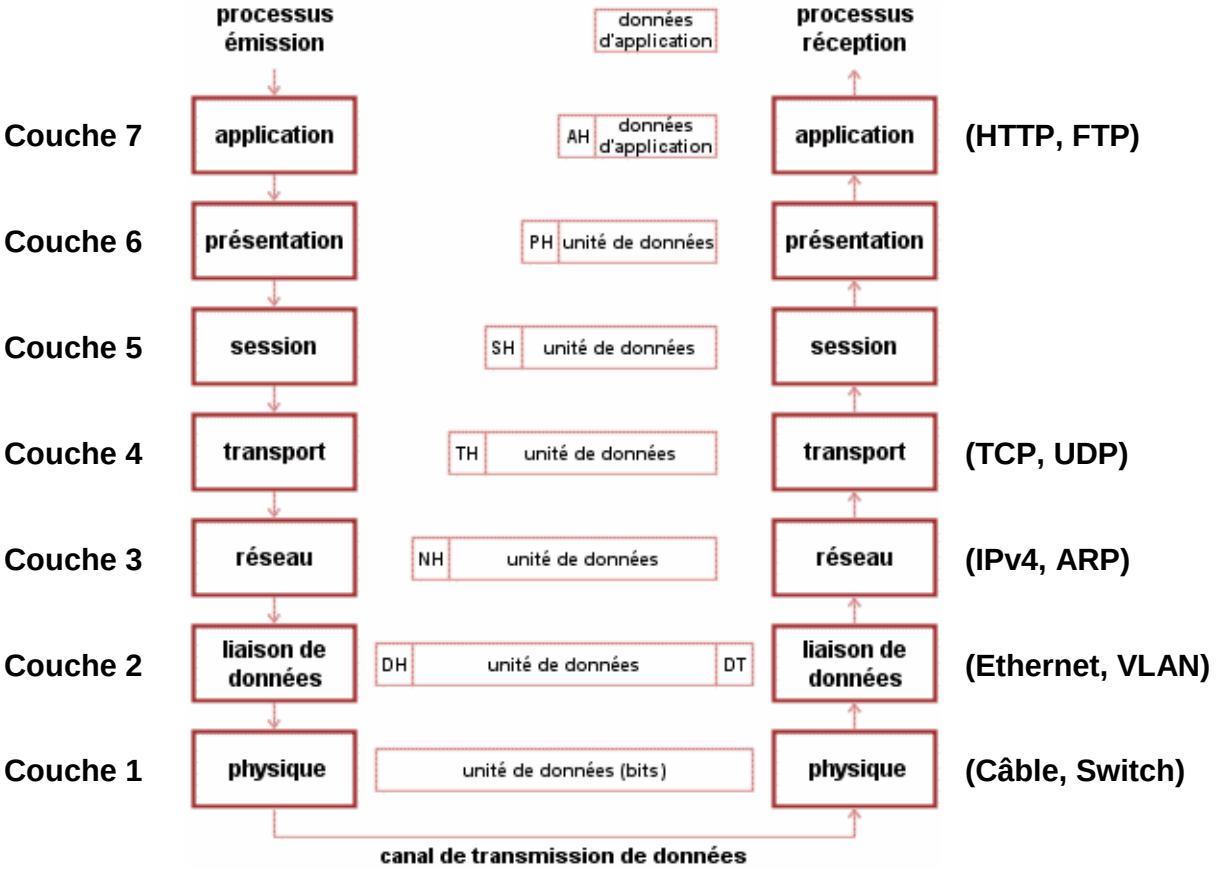


Figure 1: Couche OSI

4.3 Réception

Dans le bloc « manager_of_protocols_stack » se trouvent les différentes couches. La partie réception sera implémentée dans le bloc « ethernet_rx » (**Figure 2**). Dans le bloc « ethernet_rx_heart » se trouve la machine d'état, qui récupère les données de la trame qui arrivent, soit les adresses de source et de destination ainsi que le type de la trame (IPv4, ARP, …). C'est ici que la détection du type VLAN va être ajoutée.

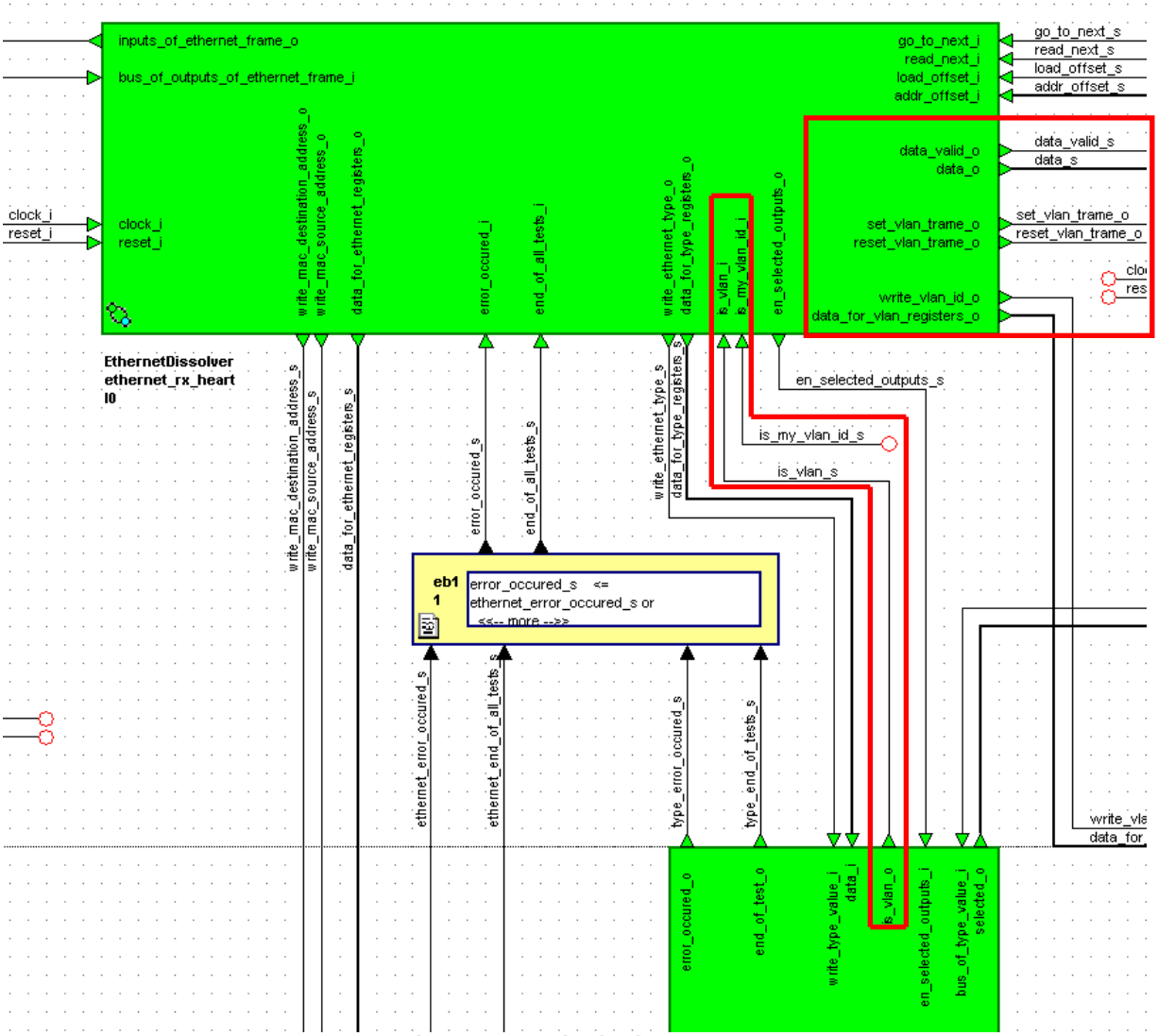


Figure 2: Partie d'ethernet_rx

Au vu de la construction d'une trame VLAN, il suffit de commencer par simplement vérifier l'Ethertype (**Figure 3**). Si le type identifié correspond au type du VLAN, soit 0x8100, les valeurs du tag TCI, comprenant la priorité, le CFI et l'ID du VLAN de la trame entrante peuvent être récupérés. L'ID est ensuite vérifié afin de savoir si la trame appartient bien à notre réseau. Si tel est le cas, l'Ethertype est revérifié et le reste de la trame se traite comme une trame normale. Sinon nous attendons la trame suivante.

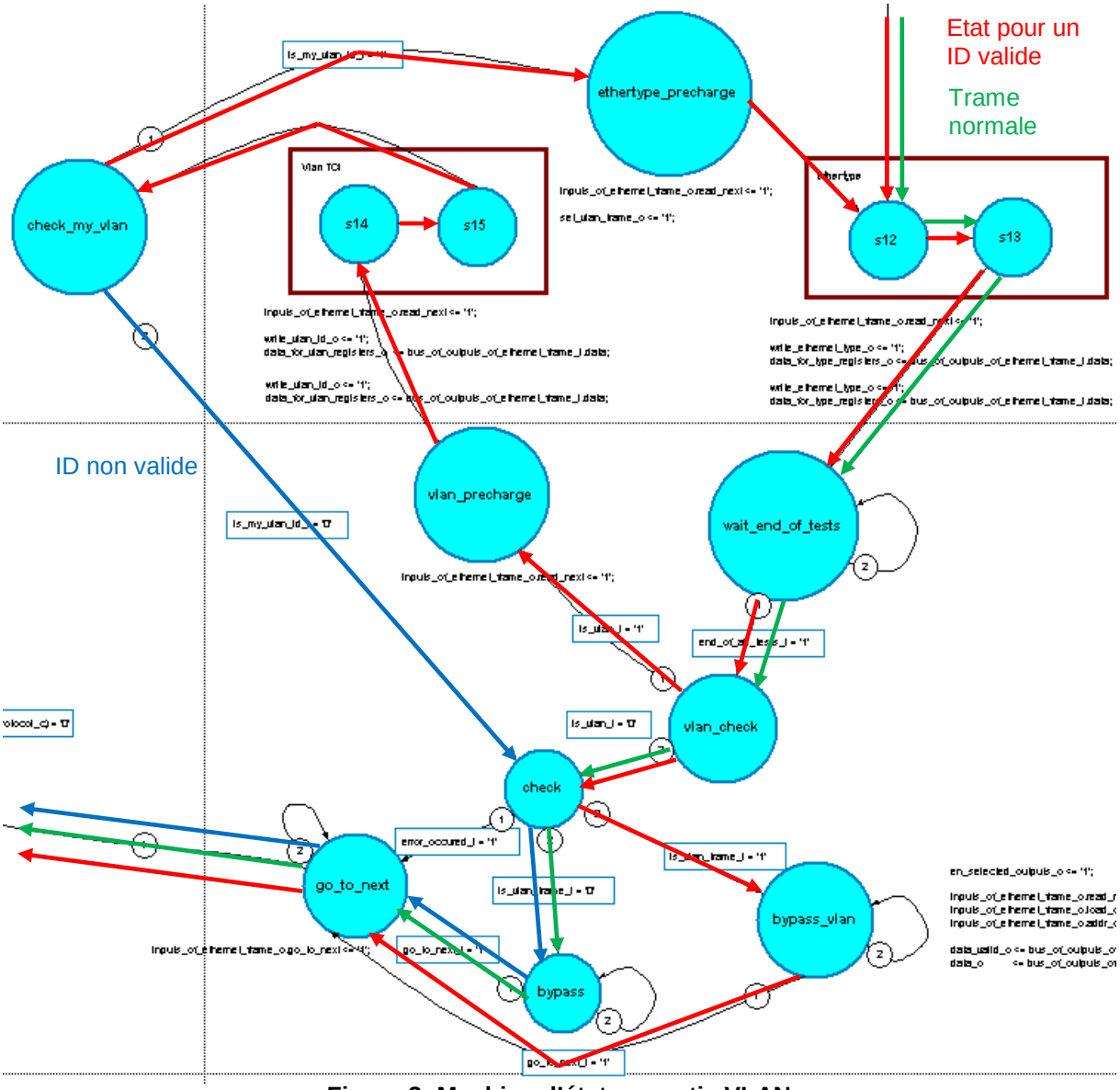


Figure 3: Machine d'état rx, partie VLAN

L'état « vlan_check » reçoit un signal « is_vlan » du bloc « type_registers » qui lui signale si la trame est un VLAN ou non. Dans le cas d'une trame VLAN, la valeur suivante de la trame est préchargée. Le tag TCI sera ensuite extrait et l'id vérifié afin de savoir si la trame appartient à notre VLAN. L'état « bypass_vlan » a été ajouté afin de décaler l'offset des datas pour les blocs suivants.

Ces valeurs sont envoyées au bloc « vlan_id » (**Figure 4**) qui, lui, va séparer la priorité et le CFi de l'ID. La priorité et le CFI sont envoyés à l'utilisateur et l'ID est envoyé au bloc « vlan_id_comparator ». Ce bloc va ensuite comparer si l'ID de la trame reçue correspond à l'ID de notre réseau. Si tel est le cas, un signal « is_my_vlan_id » est envoyé au bloc « ethernet_rx_heart ». Ce signal va permettre à la machine d'état d'aller chercher le reste de la trame si elle nous est destinée, ou d'aller à la trame suivante.

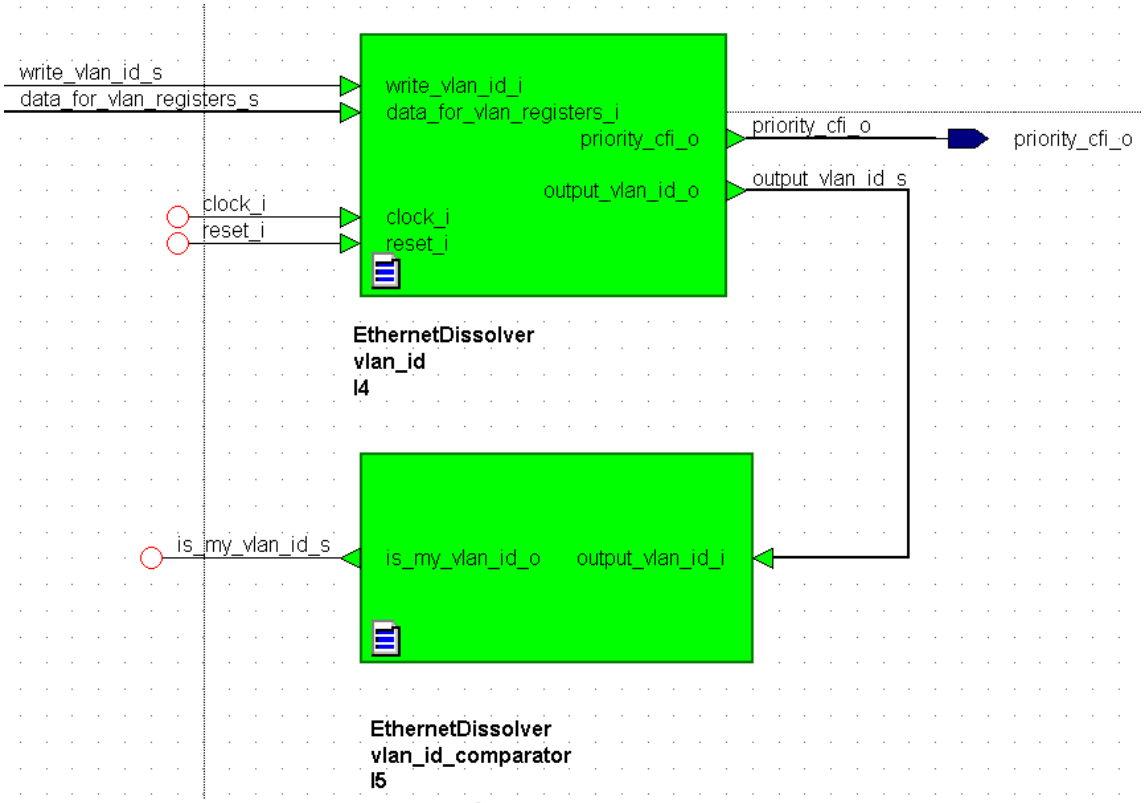


Figure 4: VLAN ID

Afin de signaler aux autres blocs à l'extérieur que la trame est un VLAN et qu'elle nous est destinée, on trouve un « set_reset » (**Figure 5**). L'état ethertype_precharge *set* ce signal et la précharge de la trame suivante le *reset*. Ce « set_reset » est aussi utilisé dans d'autres blocs.

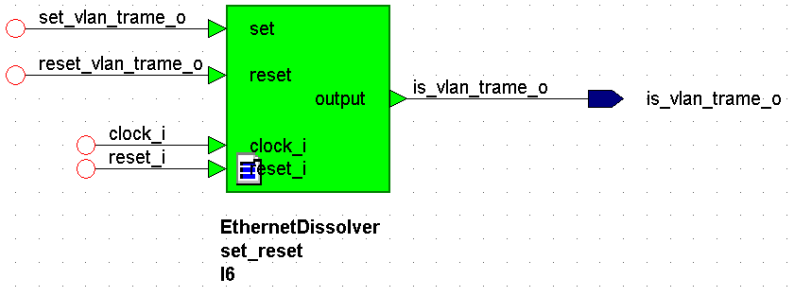


Figure 5: VLAN trame set_reset

4.4 Emission

Dans le bloc « manager_of_protocols_stack », on va implémenter la partie émission dans le bloc « ethernet_tx » (**Figure 6**).
A ce bloc sont ajoutés trois signaux : le premier pour signaler si la trame à émettre est une trame VLAN ou non, le suivant pour la priorité et le CFI définis par l'utilisateur et le dernier pour signaler si les infos sont valides.
Dans ce bloc, le bloc « ethernet_tx_heart » va être modifié. Ce bloc contient une machine d'état qui va envoyer chaque trame Byte par Byte. A ce bloc, les signaux précédents sont ajoutés, plus l'ID du réseau et le type VLAN.

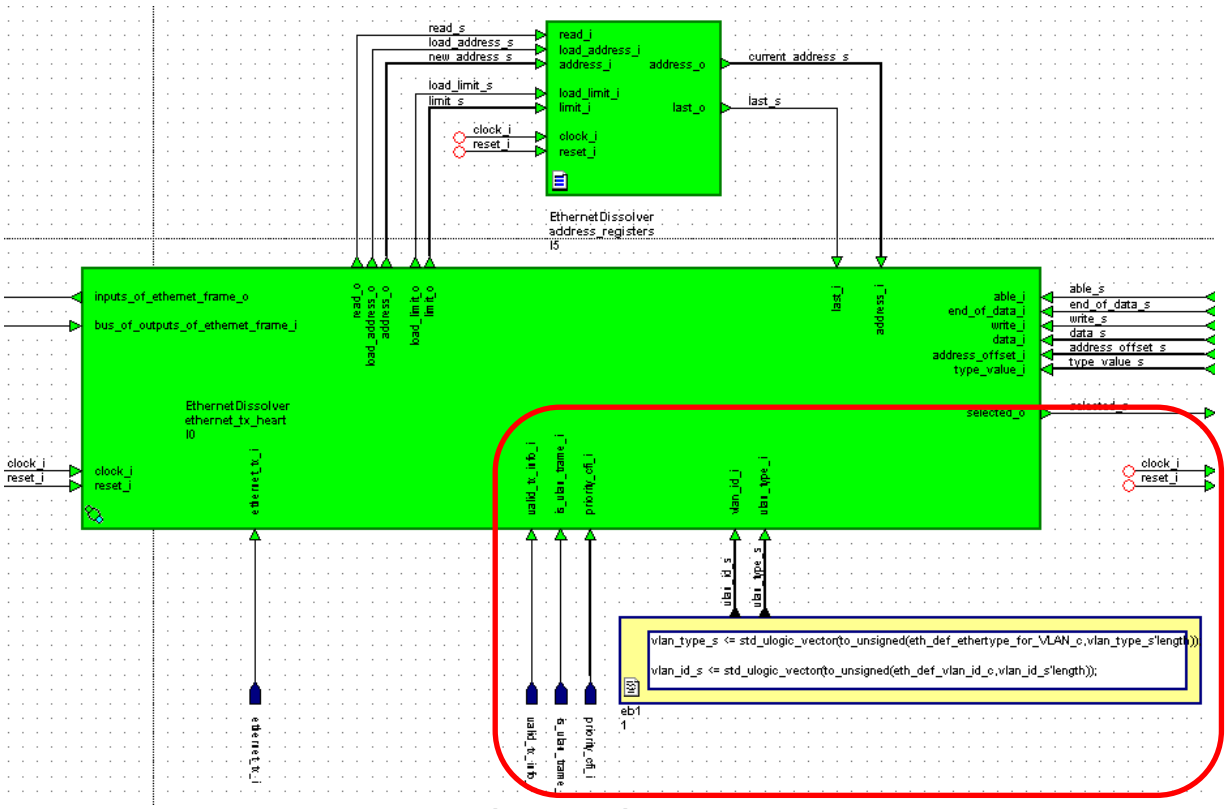


Figure 6: Ethernet_tx

La machine d'état va être modifiée afin d'émettre une trame VLAN si nécessaire. Pour cela, entre le dernier Byte de l'adresse source et l'Ethertype, les différents Bytes du tag VLAN sont ajoutés. Un signal (is_vlan_trame_i) permet de différencier ensuite si la trame à émettre est un VLAN ou non. Dans le cas d'un VLAN, la machine d'état (**Figure 7**) va émettre ces nouveaux octets, sinon, elle passera simplement à côté.

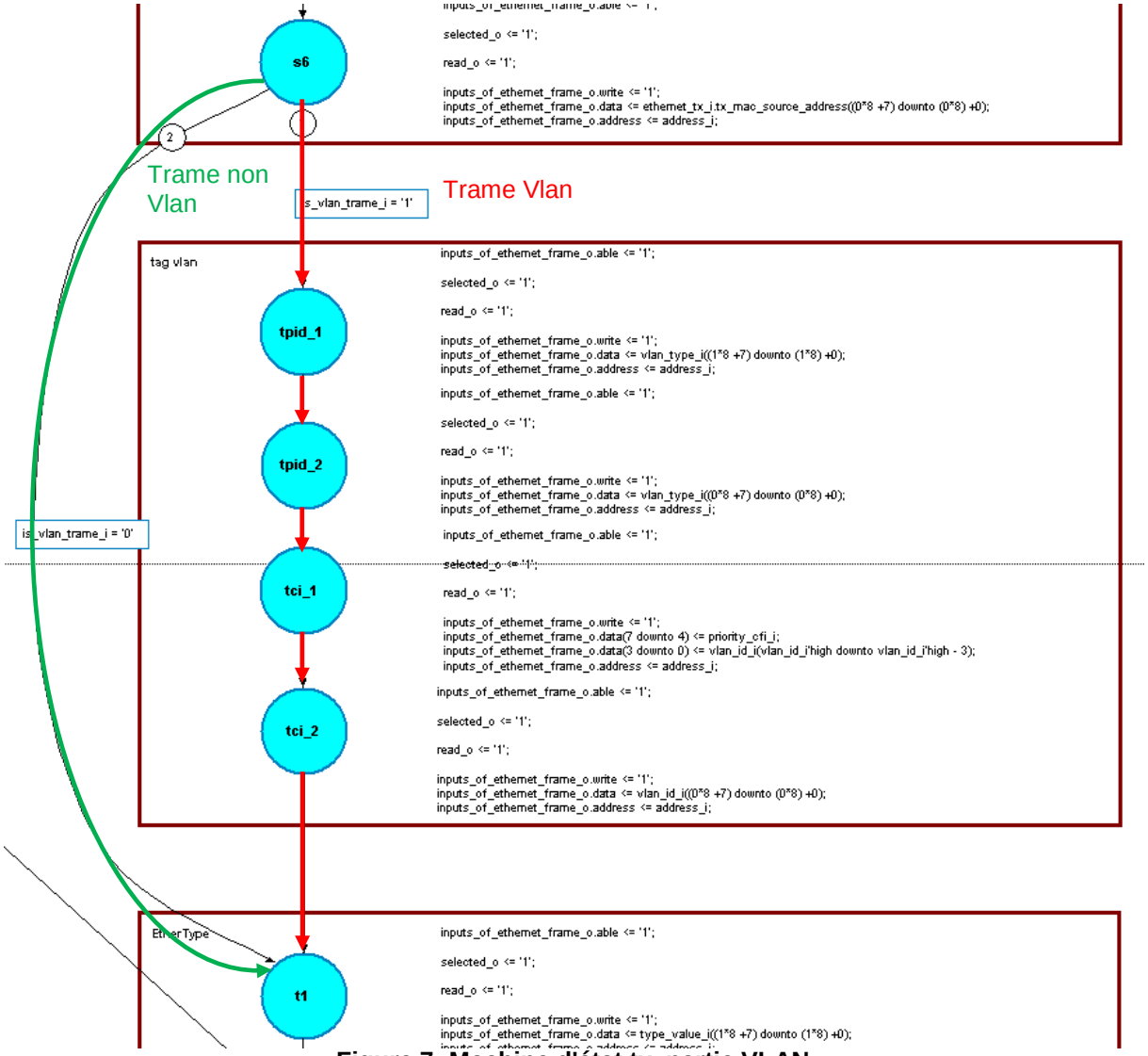


Figure 7: Machine d'état tx, partie VLAN

Lorsqu'un paquet doit être émis, le bloc «Ethernet_tx » doit donner un offset pour les données dans la mémoire, ce qui correspond à la taille du header. Cet offset est défini au début de la machine d'état du bloc «Ethernet_rx_heart » (**Figure 8**), dans l'état « bypass ». Cependant, au moment où était définie cette taille, le circuit ne connaissait pas encore si la trame à émettre était un VLAN ou non. Une trame VLAN doit avoir une taille de 4 octets supplémentaires dans le header, c'est pourquoi il faut décaler l'offset du début des données de 4 octets. Un état d'attente supplémentaire a été ajouté. Dans cet état, on attend que l'utilisateur garantisse des données valides en utilisant le signal « valid_tx_info ». Ce signal est déjà généré par chaque utilisateur, il suffit donc de le réutiliser. Le signal « is_vlan_trame » peut donc ensuite être testé.

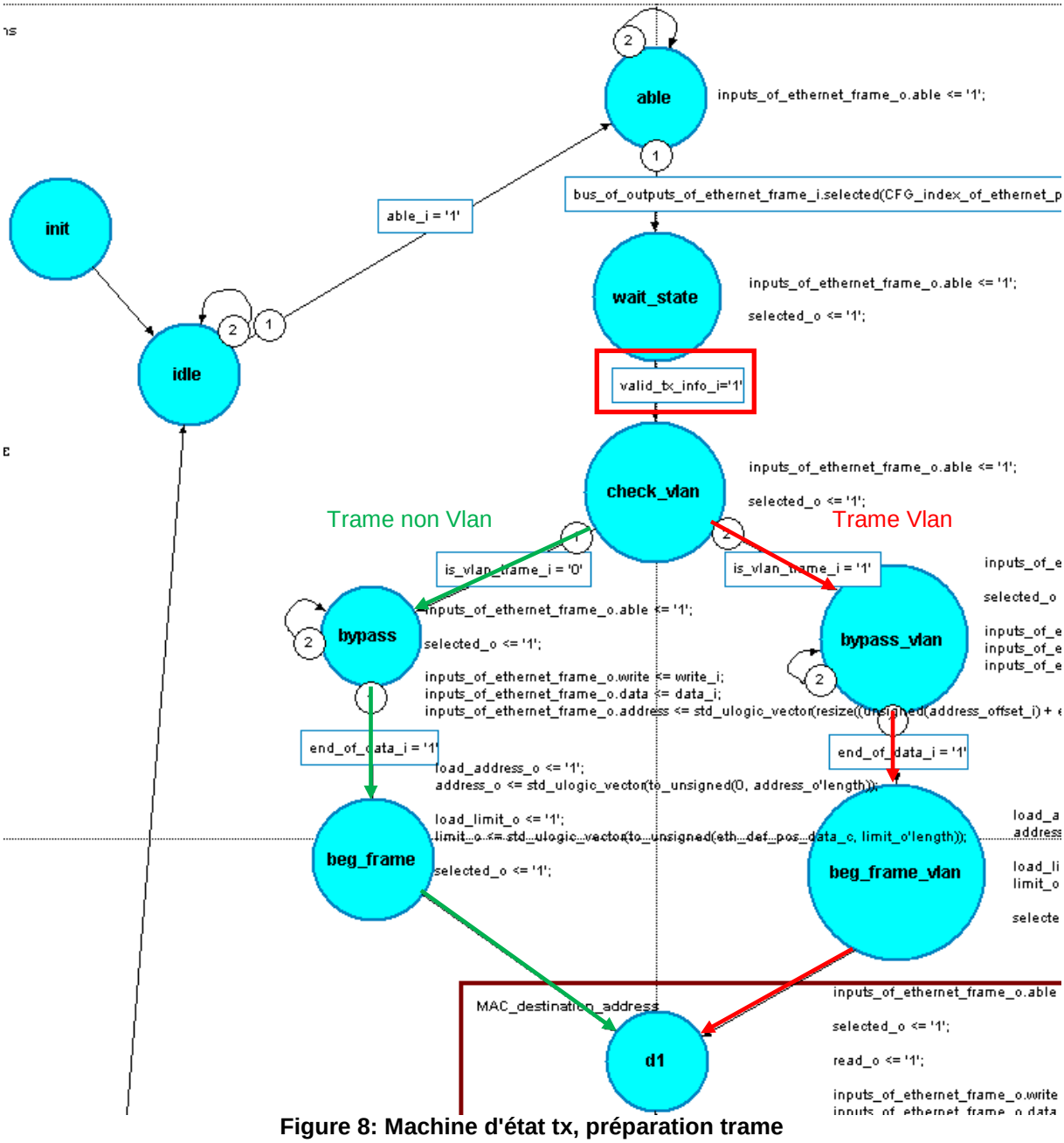


Figure 8: Machine d'état tx, préparation trame

4.5 Partie utilisateur

Chaque utilisateur qui veux émettre un paquet doit pouvoir choisir s'il émet un paquet VLAN ou non, ainsi que la priorité de son paquet.
Chaque utilisateur reçoit donc l'information qui signale si la trame reçue est un VLAN ainsi que la priorité et peut choisir ce qu'il veut émettre.
Dans un premier temps, l'objectif est de seulement faire en sorte que l'utilisateur sélectionné renvoie l'information qu'il reçoit. Deux informations sont alors nécessaires : La première est : quel utilisateur est sélectionné, et la deuxième est la fin de la trame.
L'information du bloc sélectionné est prise dans le bus « array_of_bus_of_tx_outputs_i » avec l'indice de l'utilisateur voulu. Comme cette information est utilisée lors de la transmission, elle sera prise sur le bus de sélection de transmission.
Pour l'information de fin de trame, il faut aller la chercher dans le bloc « manager_of_protocols_stack » dans le « memory_manager_tx ». Dans ce bloc, on trouve un signal appelé « load_end_s » : c'est celui que l'on va envoyer vers les différents utilisateurs pour donner l'information de fin de trame.

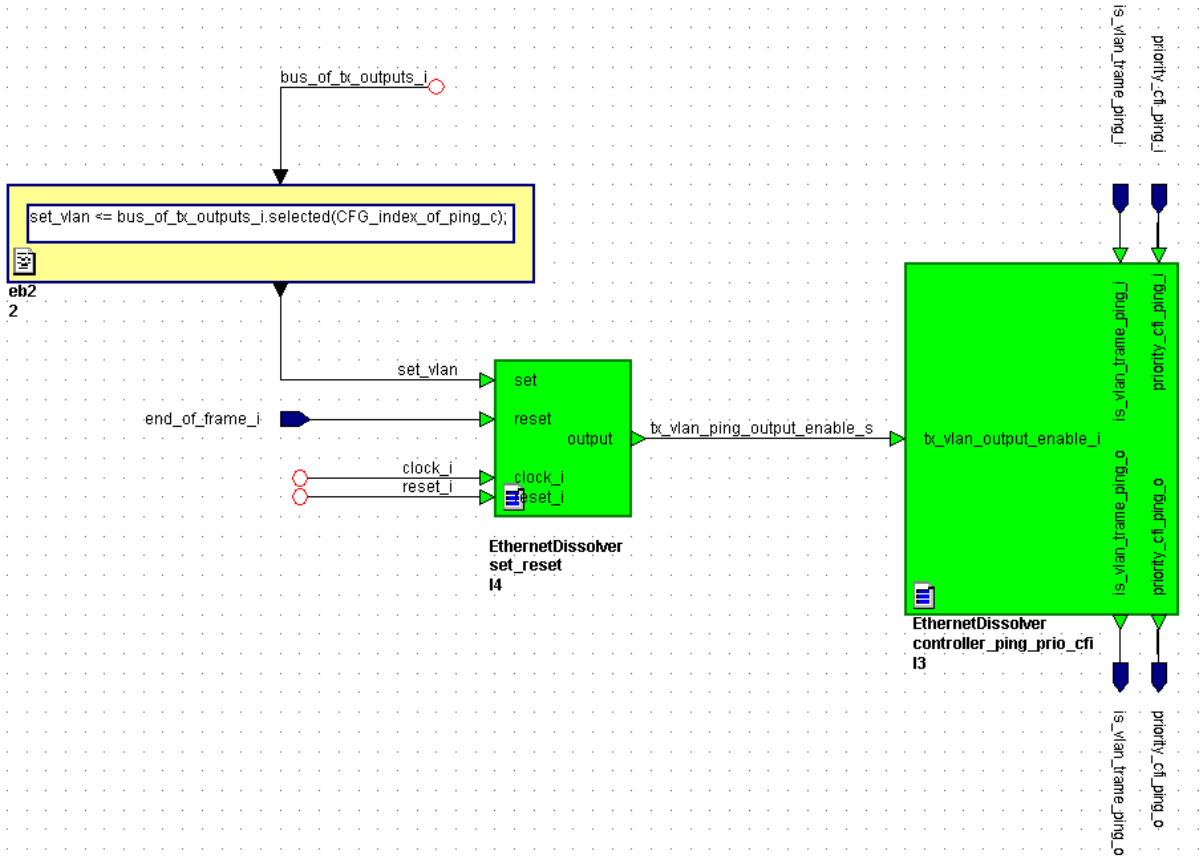


Figure 9: Sélection utilisateur

Sur la **Figure 9**, on observe une bascule set/reset, qui définit quand l'utilisateur a le droit d'écrire. Ce signal est ici utilisé pour commander le bloc « controller_ping_prio_cfi ». Ce bloc prend simplement les signaux « is_vlan_trame_XXX_i » et « prio_cfi_XXX_i » et les renvoie vers la sortie quand il

est sélectionné. Il est facile d'imaginer que ce bloc soit remplacé par un bloc qui choisit ce qu'il renvoie (VLAN et priorité).
Ces signaux émis par les différents utilisateurs sont ensuite regroupés à l'aide de « portes ou » (**Figure 10**) et renvoyés vers la couche 2 (bloc Ethernet), qui va émettre le paquet.

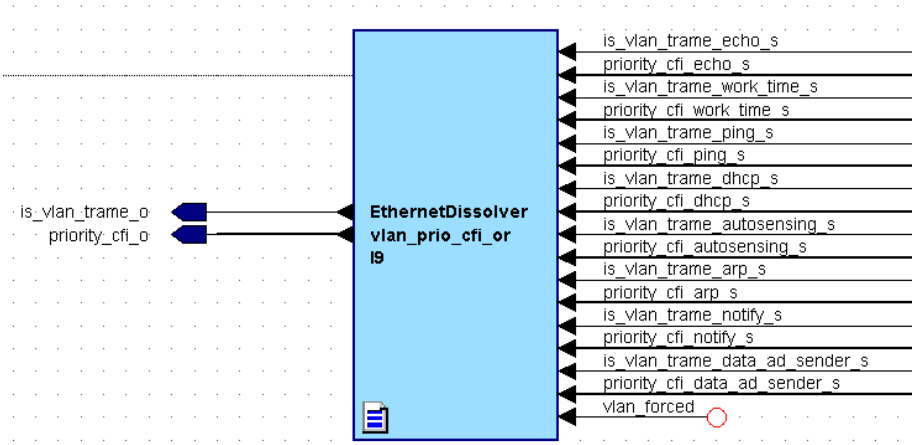


Figure 10: Porte ou

La possibilité de forcer le mode VLAN a aussi été ajoutée. Le bouton A15 de la carte permet de forcer le mode VLAN à l'émission. Dans bloc « vlan_forced_controller » (**Figure 11**) se trouve un compteur qui va lire l'état du bouton toutes les millisecondes et inverser le signal « vlan_forced » si le bouton est pressé.

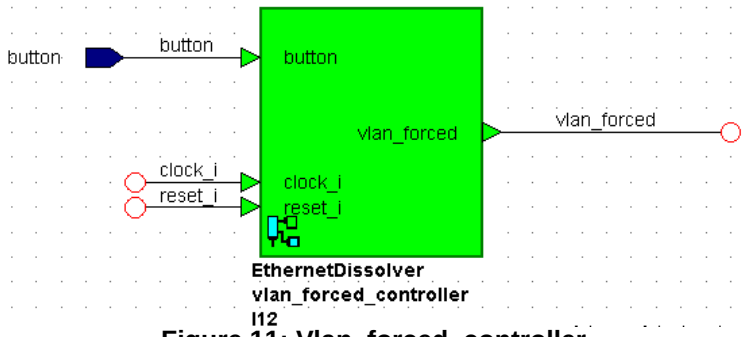


Figure 11: Vlan_forced_controller

4.6 Modification circuit original

4.6.1 DHCP

Avant de commencer à implémenter le protocole VLAN, une amélioration du circuit a été réalisée.
En effet un problème se posait en passant de la simulation à l'implémentation et inversément. A chaque changement d'un mode à l'autre, il fallait faire deux modifications dans la partie DHCP.

Afin d'améliorer le circuit existant, un signal « simul » a été ajouté. Quand ce signal est à '1', le mode simulation est en fonction, et à '0', c'est le mode implémentation qui agit.
Comme ce signal a été ajouté dans le top level, une fois dans le bloc où se trouve le test bench, et une fois du côté board, plus aucune modification n'est nécessaire.

Sur la **Figure 12**, le signal « simul » commande un sélecteur qui choisit entre le signal « random_number_s » et le signal « random_number_i ». Le choix se faisait auparavant manuellement.

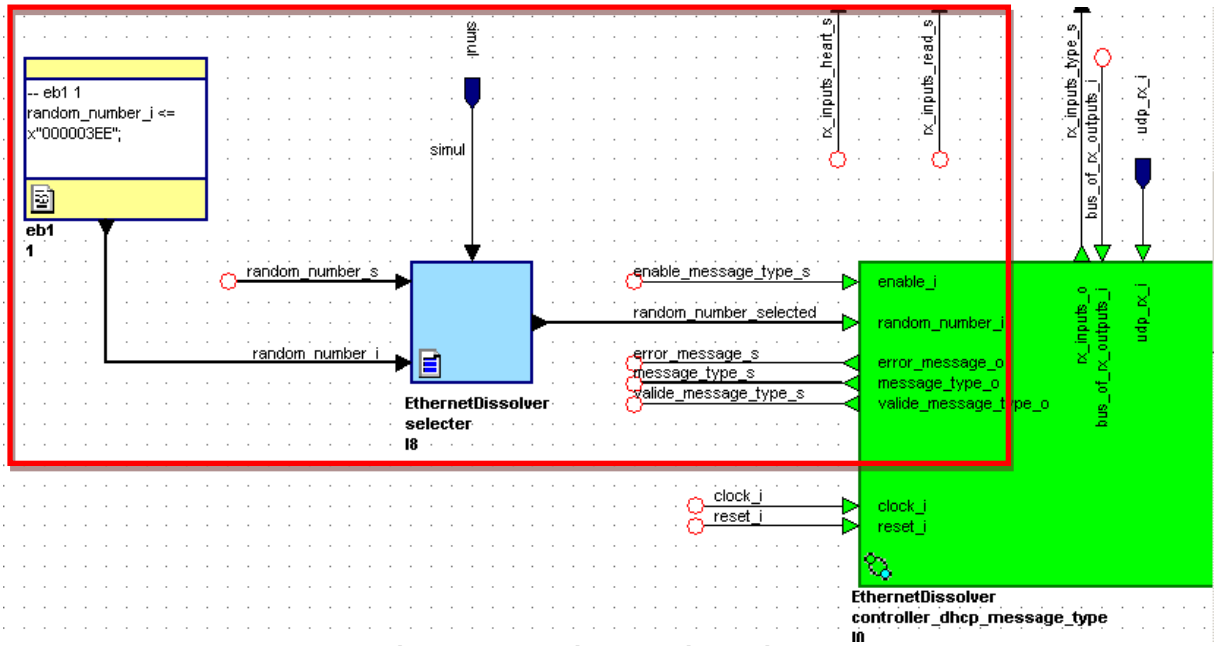


Figure 12: Random number selecter

La **Figure 13** présente une partie de la machine d'état « controller_DHCP_heart ». En mode simulation la machine d'état ne passe pas par l'état « start_timer ». Ici aussi le changement devait se faire manuellement.

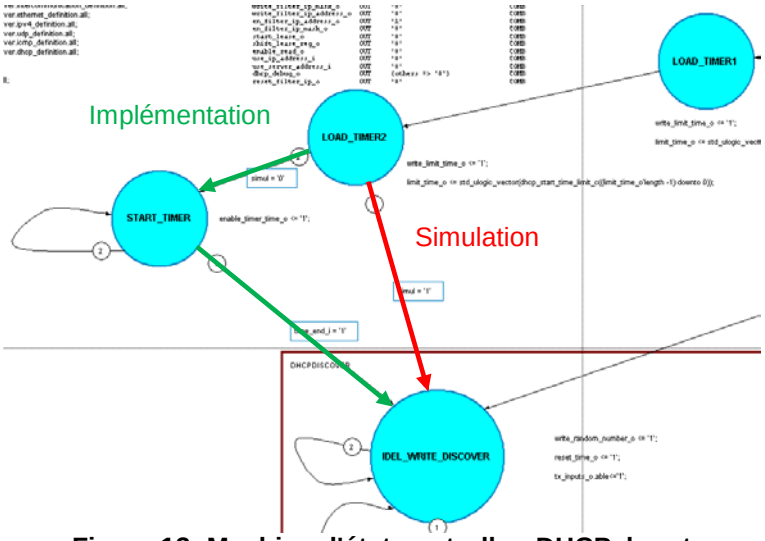


Figure 13: Machine d'état controller_DHCP_heart

Cela permettra un gain de temps et de compréhension à l'avenir.

Lors des simulations du VLAN sur le bloc DHCP, un problème a été détecté. En effet, au moment d'émettre une trame VLAN, cela ne fonctionnait pas dans certain cas. Le bloc «Ethernet_rx_heart » ayant été modifié pour attendre de connaître l'état du VLAN, il est désormais nécessaire de recevoir le signal « valid_tx_info » (voir Figure 8). Dans la machine d'état « controller_dhcp_heart » (**Figure 14**) du bloc DHCP, dans l'état « REQUEST » le signal « tx_info_output_enable_o » n'était pas défini à '1'. Ce qui impliquait que le signal « valid_tx_info » ne passait pas à '1' et le paquet n'était donc pas émis.

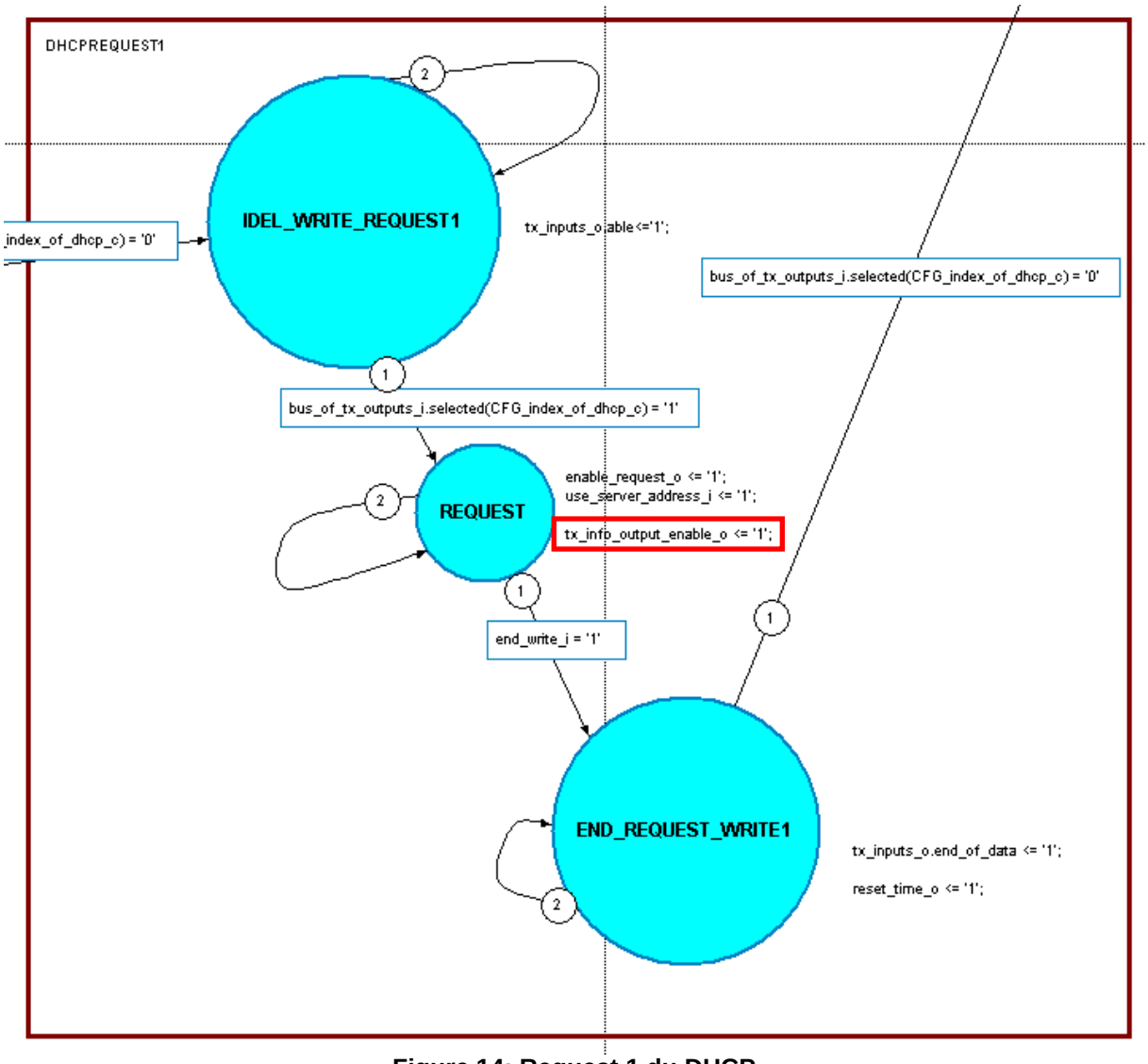


Figure 14: Request 1 du DHCP

4.6.2 Padding

Lorsque qu'un paquet n'atteint pas une certaine taille minimum, il est complété par du padding (**Figure 15**). Ceci est réalisé dans la machine d'état du « memory_manager_tx ».
Dans le cas d'une requête ARP, du padding est nécessaire. Lors de l'envoi d'une trame VLAN, la taille minimum est plus grande de 4 Bytes. Ce qui fait qu'il faut différencier l'envoi d'une trame VLAN ou non, afin de définir sa taille minimale et donc d'ajouter du padding en conséquence.

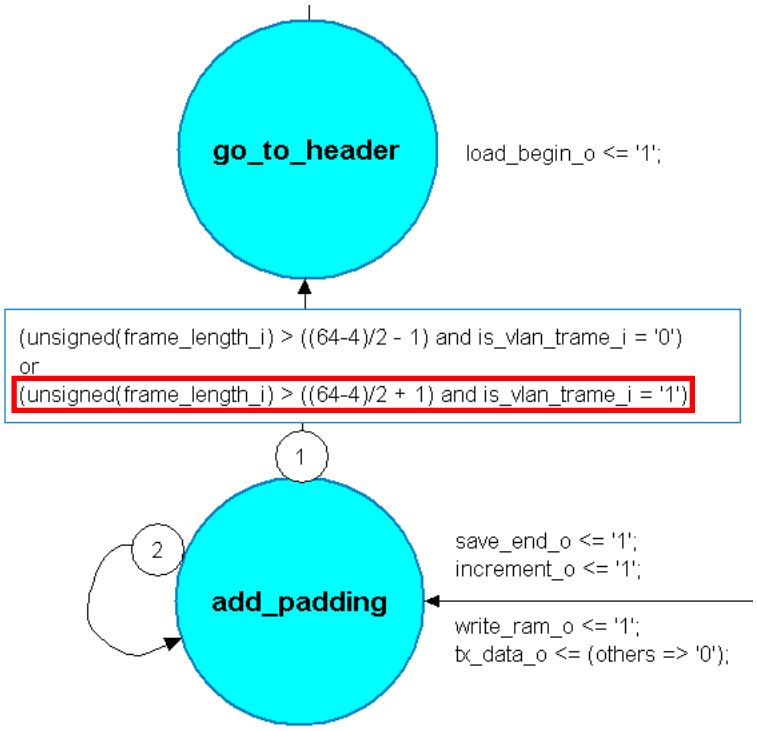


Figure 15: Padding

5. Tests en simulation

Dans les tests, plusieurs choses doivent être vérifiées. La réception et l'émission de trame VLAN doivent se faire de manière correct ; les machines d'état doivent passer dans les différents états attendus ; les différents blocs doivent réaliser les tâches attendues ; les données émises doivent être identiques avec ou sans VLAN.

Afin de tester la réception de trame VLAN, il faut préparer un fichier *.cap comprenant des trames VLAN. La **Figure 16** montre une de ces trames. On peut voir la priorité définie à 2, le CFI à 0 et l'ID à 2468, soit 0x9A4. Des fichiers comprenant des trames ARP, UDP et ICMP on été adaptés au VLAN pour les tests. Afin de les différencier, leurs noms se terminent par « _vlan.cap ».

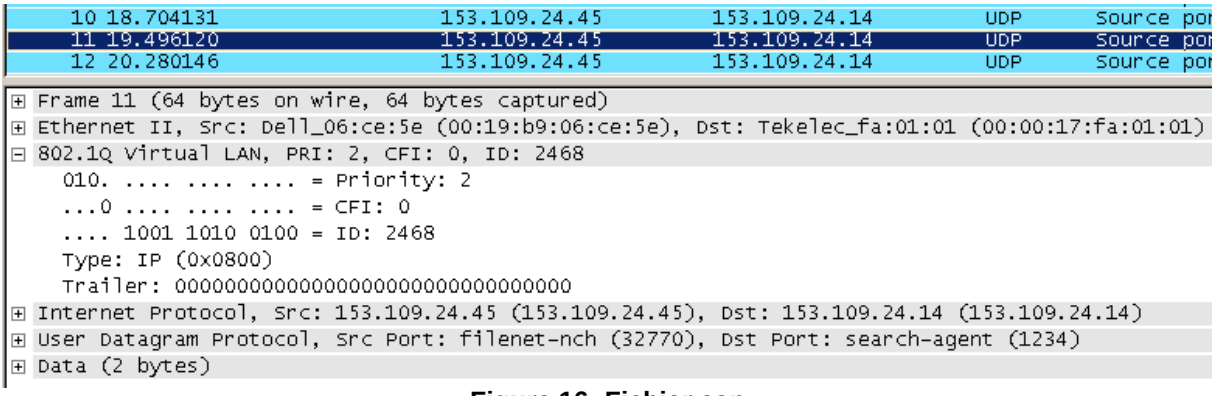


Figure 16: Fichier cap

La première chose à tester est donc la réception. Sur la **Figure 17.1**, l'observation commence dans l'état s12, qui correspond au premier octet de l'Ethertype puis continue dans l'état s13. Une fois ces 2 octets chargés, on va tester leur valeur. On voit alors que le signal « is_vlan_s » passe à '1'. Grâce à ce signal, l'état « vlan_check » dirige dans l'état de précharge du VLAN, avant d'aller chercher les 2 octets du TCI avec l'état s14 et s15 (**Figure 17.2**). Les octets se chargent dans les signaux « output_vlan_id_s » et « priority_cfi_o ». Dans l'état « check_my_vlan », on vérifie si la valeur chargée correspond à celle dans notre registre. On constate ici que le signal « is_my_vlan_id_s » passe à '1' dans l'état « check_my_vlan », ce qui fait que nous revenons dans les états s12 et s13 pour aller chercher l'Ethertype du paquet contenu dans la trame VLAN, en passant par l'état ethertype_precharge qui, lui, met le signal « set_vlan_trame_o » à '1'. La suite (**Figure 17.3**) se déroule comme un paquet normal.

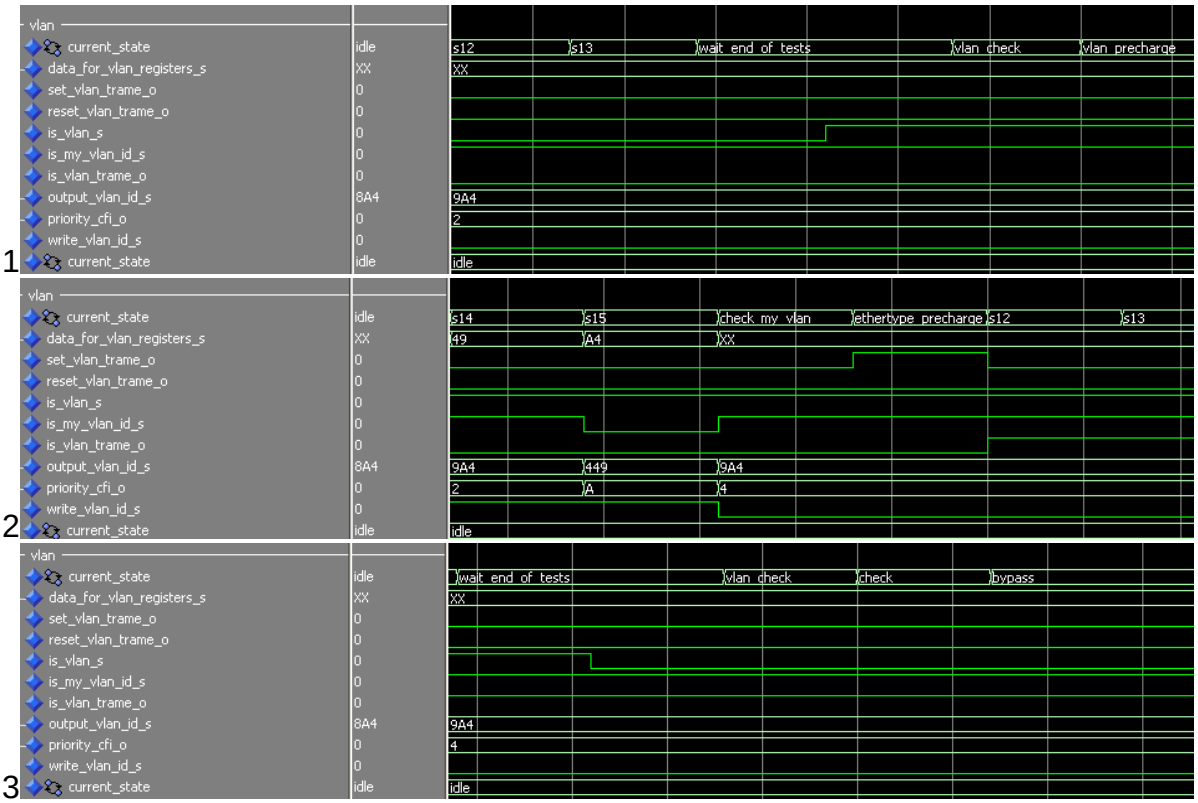


Figure 17: Machine d'état de réception du VLAN

5.1 Réception

Nous devons aussi vérifier que le circuit réagit correctement au changement à la réception de l'ID des différentes trames. Sur la **Figure 18**, on observe la réception de 4 trames VLAN. Le signal « output_vlan_id_s » charge à chaque fois la valeur 0x9A4, qui correspond à l'ID 2468 et le signal « priority_cfi_o » change à chaque trame. Les signaux « output_vlan_id_s » et « priority_cfi_o » sont mis à jour lorsque « write_vlan_id_s » est à '1'. On voit aussi les signaux « set_vlan_trame_o » et « reset_vlan_trame_o », qui marquent le début et la fin de la trame VLAN. Ces 2 signaux commandent le signal « is_vlan_trame_o ». Celui-ci informe que les trames sont des VLAN.

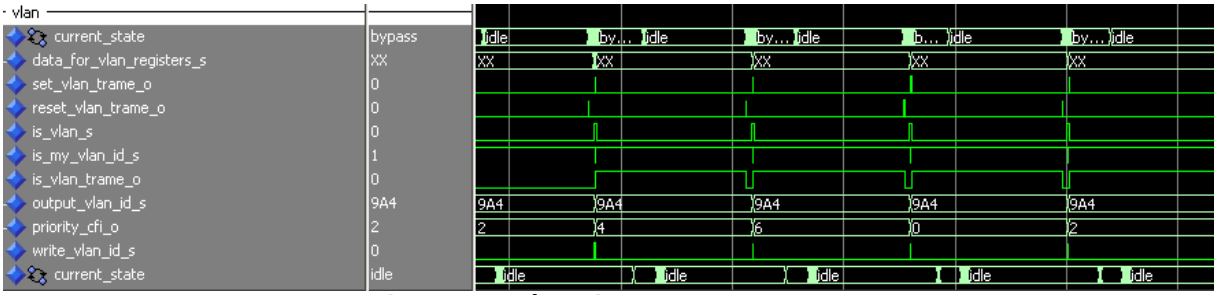


Figure 18: Réception VLAN ID correct

Dans la **Figure 19**, on remarque que lorsque la valeur de « output_vlan_id_s » n'est plus de 0x9A4, le signal « is_my_vlan_id_s » passe à '0'. Comme ce VLAN n'est pas le nôtre, le signal « set_vlan_trame_o » reste à '0'.

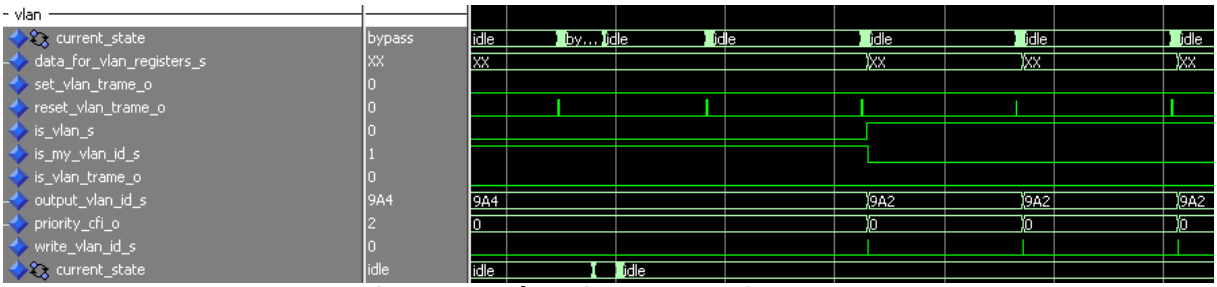


Figure 19: Réception VLAN ID incorrect

5.2 Emission

Pour l'émission, on doit vérifier que la machine d'état fasse la différence entre les trames VLAN et non VLAN. Sur la **Figure 20**, on observe l'envoi d'une trame VLAN, qui montre que les signaux « is_vlan_trame_o » et « is_my_vlan_id_s » sont à '1', une trame VLAN est donc émise. Cela est confirmé par la machine d'état, qui passe par les états tpid1 et 2, et tci1 et 2. Ces 4 états ajoutent les 32 bits du tag VLAN à la trame émise.

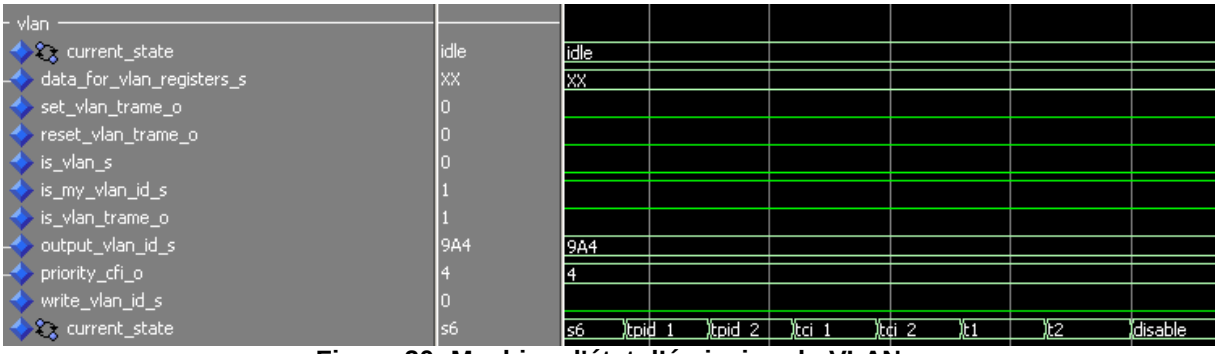


Figure 20: Machine d'état d'émission du VLAN

Sur la **Figure 21**, les données correspondantes au VLAN ont bien été chargées en mémoire. Dans la partie tx memory, sur la ligne « t_word_s », on voit 8100, qui correspond à l'Ethertype du VLAN. Dans les 2 Bytes suivants, on voit 49A4, ce qui correspond au CFI, à la priorité et à l'ID du VLAN émis. Le suivant, 0800, est l'Ipv4, soit le type du paquet émis. Ici l'utilisateur actif est l'écho, montré par le signal « tx_vlan_echo_output_enable_s » qui est à '1'. C'est donc lui qui transmet les datas du VLAN. Pour l'émission d'une trame VLAN, il est important que l'un des utilisateurs ait ce signal *enable* à '1'.

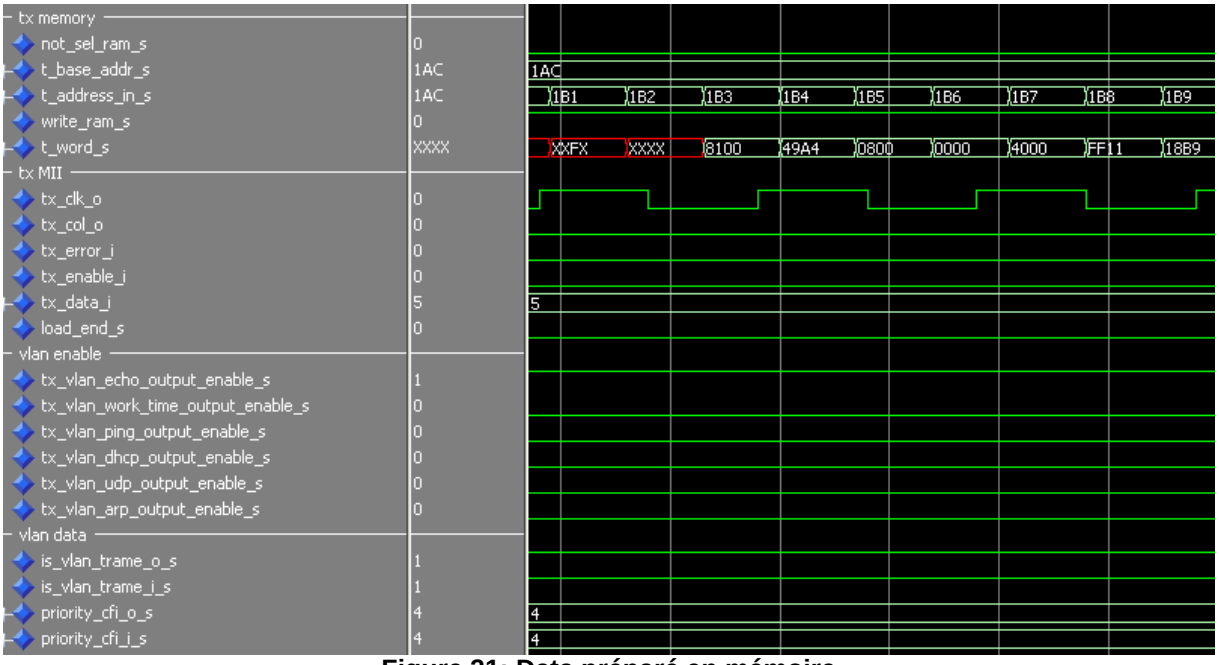


Figure 21: Data préparé en mémoire

Sur la **Figure 22**, le signal *enable* est actif lors du chargement des données en mémoire. Si ce n'était pas le cas, la machine d'état émettrait une trame normale.

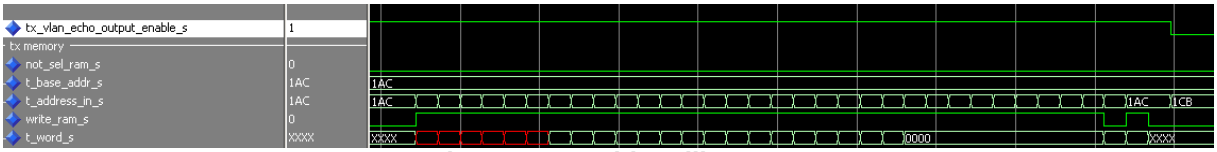


Figure 22: Enable utilisateur

Une fois les données chargées dans la mémoire, elles sont envoyées au MII. Sur la **Figure 23**, dans la partie tx MII, la lecture des données émises peut se faire sur la ligne « tx_data_i ». On remarque que pour chaque octet émis, les 4 premiers bits sont inversés avec les 4 suivants.

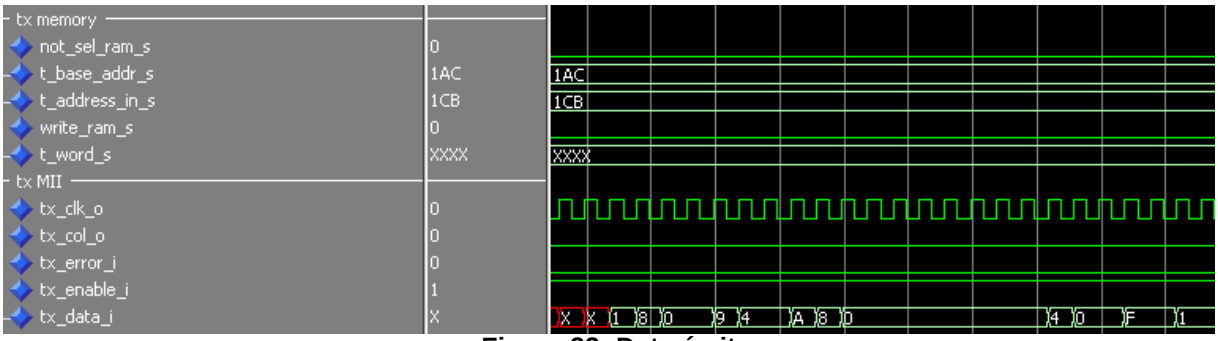


Figure 23: Data émit

Il faut encore tester que les datas émis correspondent à ceux voulus. C'est-à-dire que les trames VLAN et non VLAN soient identiques, mis à part le header et le checksum.

Une simulation d'une réponse ARP va prouver cela. Sur la **Figure 24**, on peut observer les datas émis dans le champ « t_word_s ». Le signal « is_vlan_trame_o » à '0' dit que le paquet n'est pas un VLAN. Les 12 premiers Bytes sont les adresses Mac destination et source. Ces adresses sont suivies du type ARP (0x0806). En comparant avec la **Figure 25**, on constate que les 12 premiers Bytes sont les mêmes. Cependant, ils sont suivis de 0x8100 et 0x09A4 avant le 0x0806 de l'ARP. Le 0x8100 correspond au type VLAN et le 0x09A4 à la priorité, au CFI et à l'ID. La suite est identique sur les deux trames.

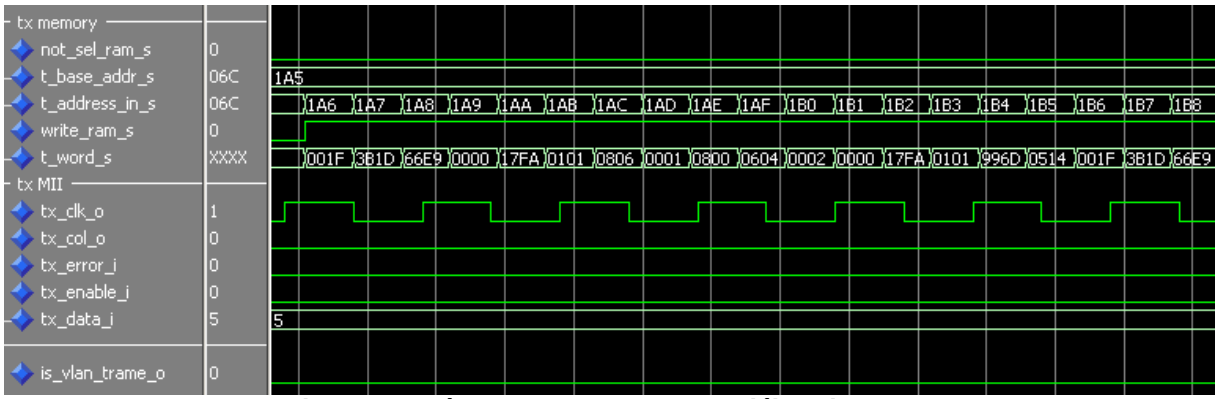


Figure 24: Réponse ARP non VLAN début de trame

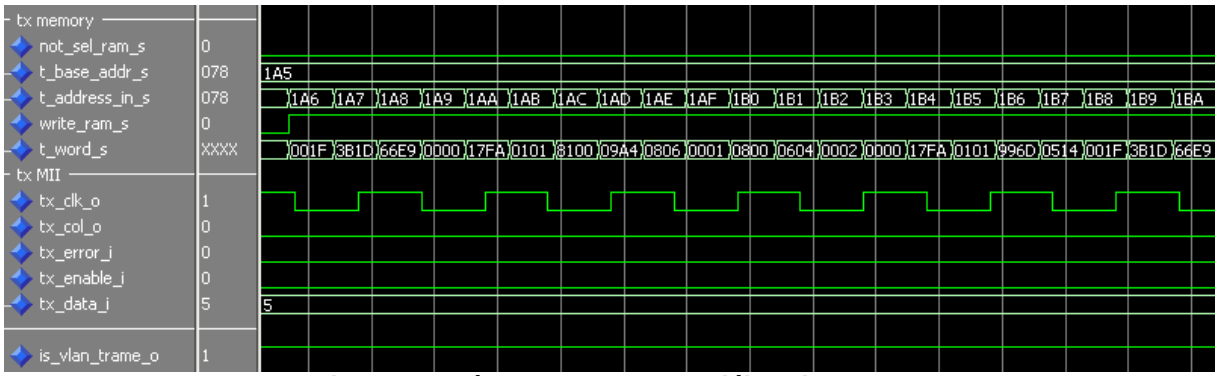


Figure 25: Réponse ARP VLAN début de trame

La deuxième chose à vérifier sur ces réponses ARP est l'ajout du padding en fin de trame. Les réponses ARP n'atteignant pas la taille minimale d'une trame, du padding doit donc être ajouté. Sur les **Figures Figure** 26 **et Figure** 27, on remarque que les deux trames ajoutent 20 Bytes de padding. Les adresses sont aussi décalées de deux, soit deux fois 16 bits, ce qui correspond aux 32 bits de la taille du tag VLAN.

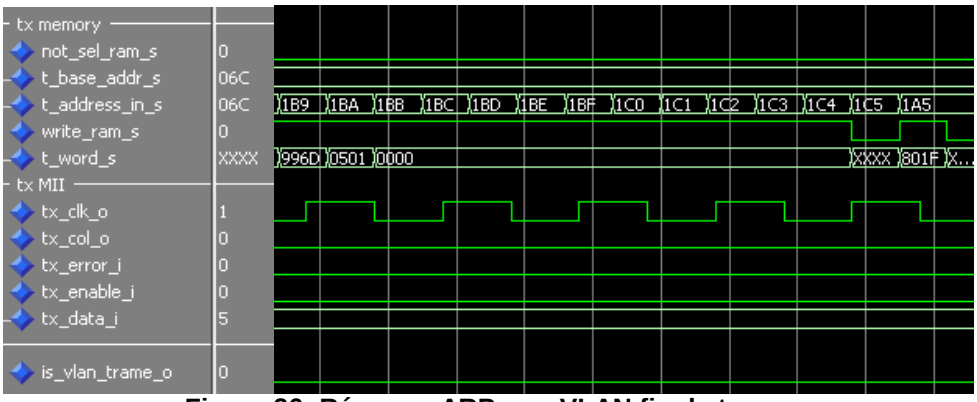


Figure 26: Réponse ARP non VLAN fin de trame

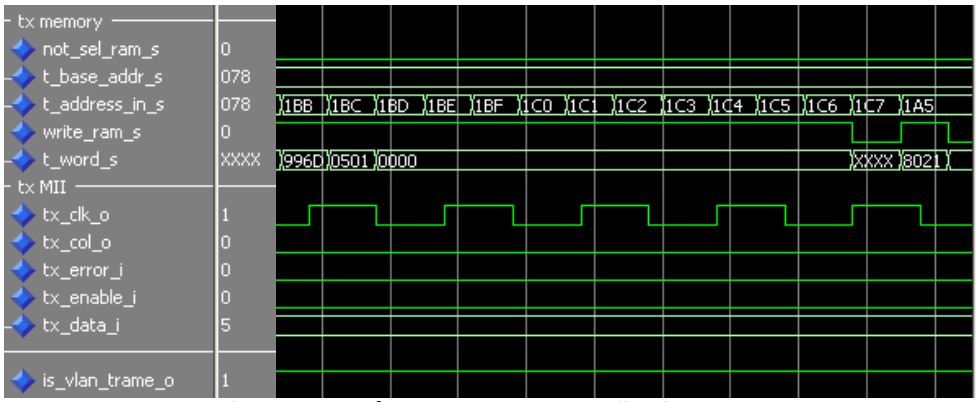


Figure 27: Réponse ARP VLAN fin de trame

Le test complet sur une réponse ARP démontre que le circuit réagit de manière correcte et envoie les trames voulues.

5.3 Partie utilisateur

Il faut encore garantir que, pour d'autres utilisateurs ou contrôleurs, la sélection se fasse de manière correcte. C'est pourquoi une simulation, avec un fichier *.cap comprenant plusieurs utilisateurs différents, a été réalisée. Sur la **Figure 28**, on constate que les VLAN *enable* de plusieurs utilisateurs s'activent. Ils sont aussi actifs pendant le chargement en mémoire des données à émettre. Comme le test a été réalisé avec des trames non VLAN, il est normal que les données VLAN soient toutes à zéro.

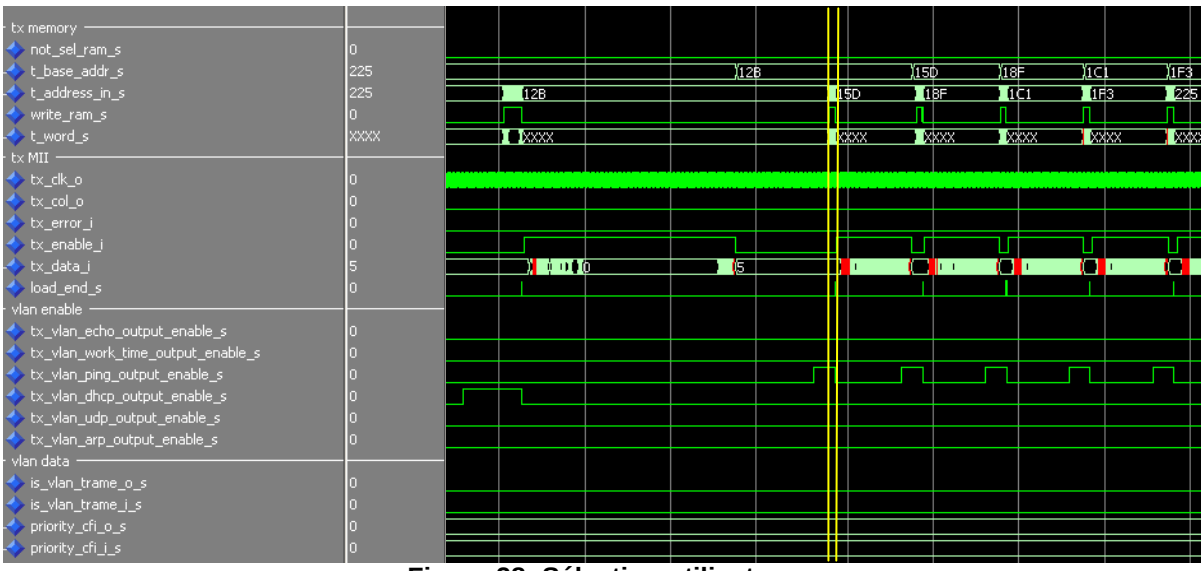


Figure 28: Sélection utilisateur

6. Tests sur carte

6.1 Réception

Le test de réception d'une trame VLAN par la carte nécessite un switch programmable. En effet, un ordinateur ne peut pas émettre de paquets VLAN. Cela rend difficile le test. Un switch programmable a donc été connecté entre l'ordinateur et la carte FPGA. Il a été configuré comme étant sur le réseau 2468, le même que sur la carte. L'entrée coté ordinateur a été réglée comme entrée non taguée, et celle vers la carte comme étant taguée. La difficulté provient du fait que l'on ne peut pas observer le paquet réellement reçu ou émis par la carte. Cependant, on peut imaginer que si le switch met un tag et que sur l'ordinateur on reçoit une réponse correcte par rapport à une requête, cela signifie que le traitement se fait correctement.
Les tests ont donc montré qui rien ne revenait en retour. La sortie taguée du switch à ensuite été testée sur Wireshark. Cette sortie ne fonctionne apparemment pas de manière correcte, les paquets reçus ne sont pas VLAN et tous ne sont pas transmis. Les statistiques du switch donnent un problème de CRC. Les tests du switch ont été faits sur plusieurs ports et le constat est le même. Les tests avec ce switch ne sont donc pas concluants. Cela peut peut-être provenir du switch ou de sa configuration. Pour garantir le bon fonctionnement de la carte, il faudrait pouvoir la tester sur un réseau dont les trames sont VLAN et dont on pourrait facilement observer les paquets.

6.2 Emission

La carte ne pouvant pas recevoir de paquet VLAN avec ce switch, la possibilité de forcer le mode VLAN avec un bouton a été ajouté. Cela a permis de comparer des trames VLAN et non VLAN émises et ainsi s'assurer le bon fonctionnement du circuit.
Evidemment les trames VLAN ne sont pas reconnues par l'ordinateur, il ne les considère donc pas comme des réponses à ses requêtes.

Le premier test est la réception du DHCP Discovert. En comparant les **Figures Figure** 29 **et Figure** 30, on peut voir que la première contient le tag et pas la suivante. Le serveur a donc répondu à cette requête sans le tag. En comparant les données des deux paquets on peut s'apercevoir qu'elles sont identiques.

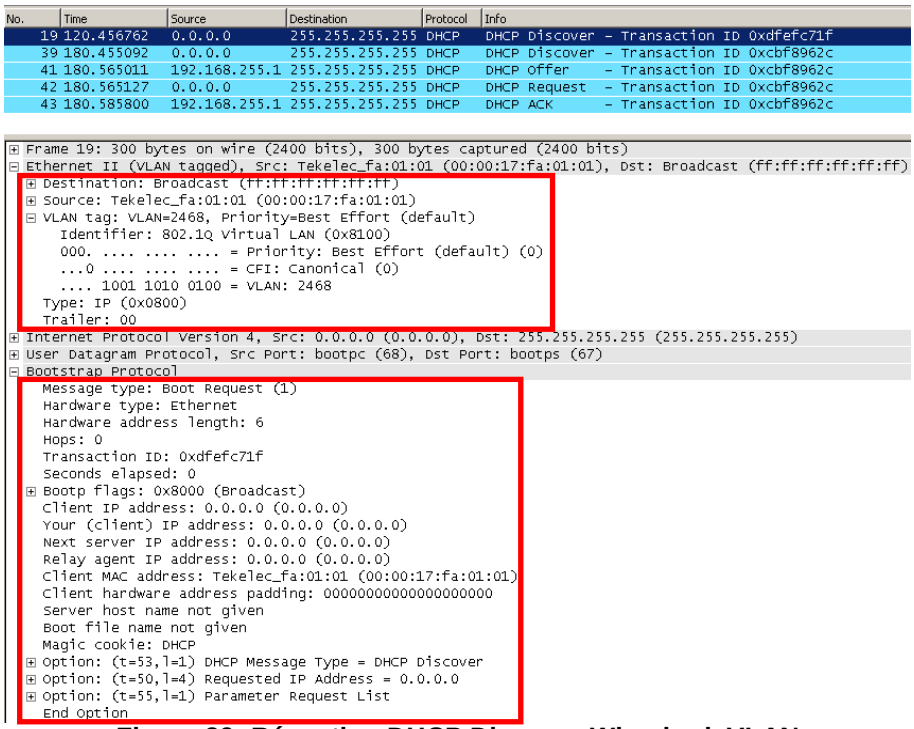


Figure 29: Réception DHCP Discover Wireshark VLAN

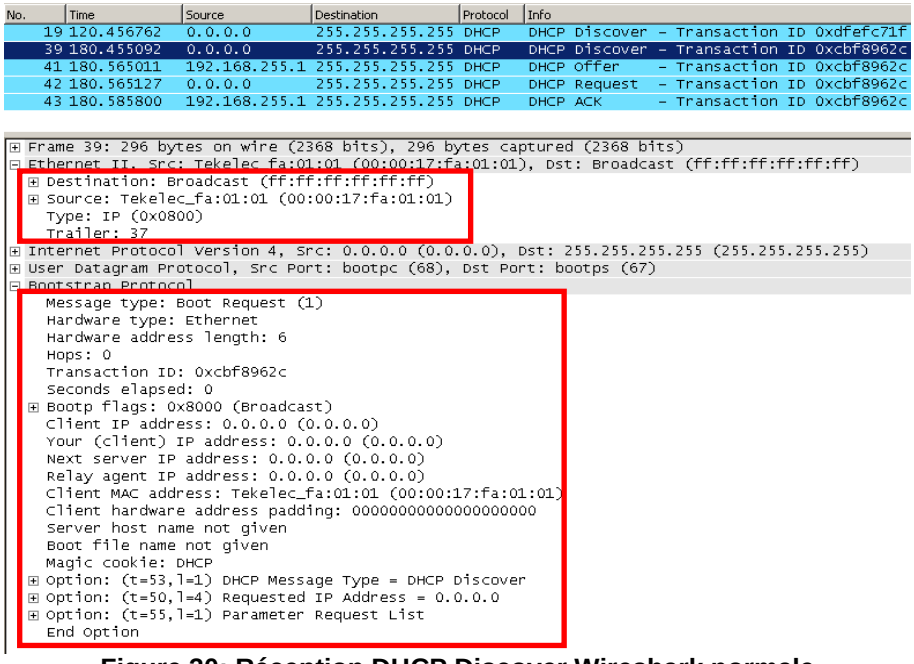


Figure 30: Réception DHCP Discover Wireshark normale

De même pour un ping, les réponses sont correctes (**Figure 31** et **Figure 32**). Les ID correspondent aux trames émises et les données sont correctes.

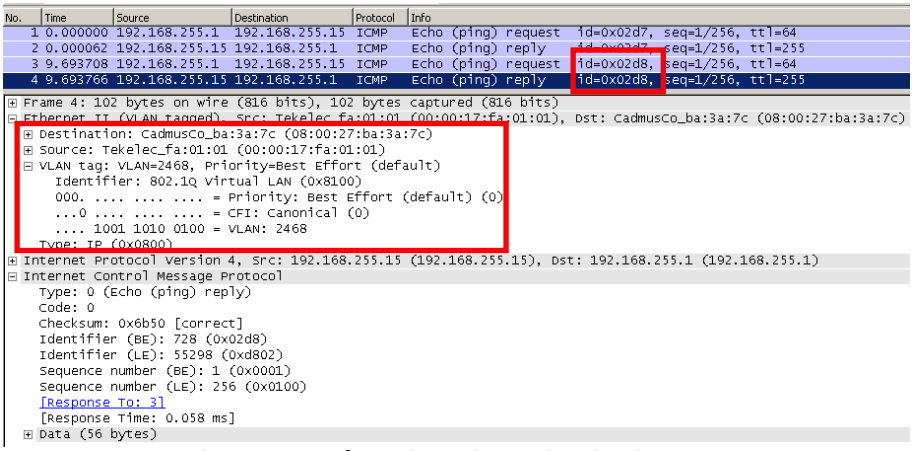


Figure 31: Réception Ping Wireshark VLAN

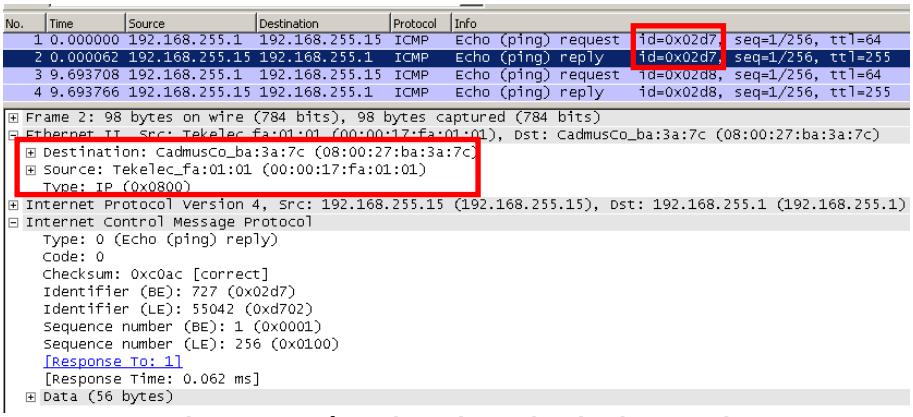


Figure 32: Réception Ping Wireshark normale

Les requêtes ARP (**Figure 33** et **Figure 34**) sont aussi correctes. Le champ Trailer comporte le même nombre de zéros, ce qui signifie que le padding est ajouté de manière correcte.

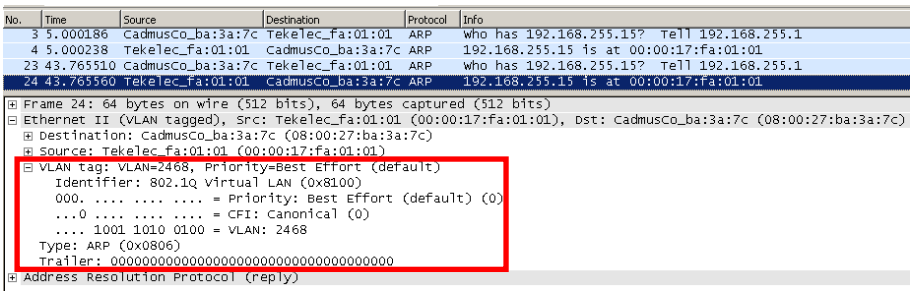


Figure 33: Réception ARP Wireshark VLAN

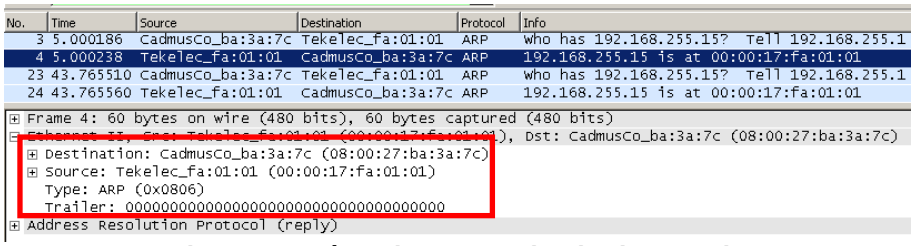


Figure 34: Réception ARP Wireshark normale

Les tests avec le switch ont par contre montré, qu'une trame reçue par le switch sur un port non tagué, mais appartenant au réseau, était transmise, mais sans le tag.

6.3 Remarque

Le test de la carte sur Windows server 2003 a démontré un problème. En effet, le serveur DHCP n'a pas réussi à attribuer une adresse IP, alors qu'Ubuntu le fait sans problème. La carte émet un « DHCP DISCOVER », le serveur lui répond par un « DHCP OFFER », la carte renvoie ensuite un « DHCP REQUEST » qui est reçu par le Windows server. Mais le serveur ne renvoie pas de « DHCP ACK ».
En comparant les **Figures Figure** 35 **et Figure** 36, des différences apparaissent dans les données, soit les adresses et les options.

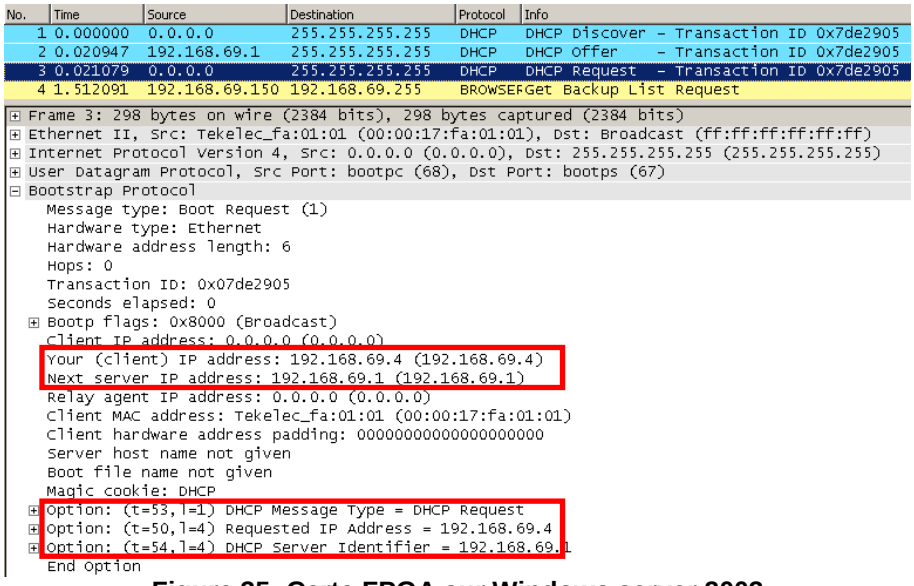


Figure 35: Carte FPGA sur Windows server 2003

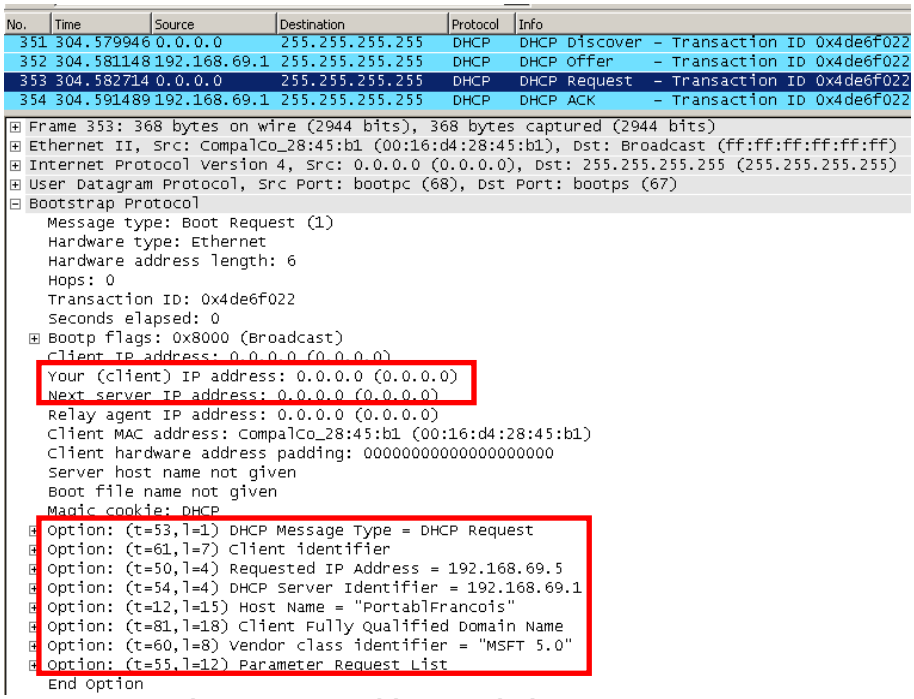


Figure 36: Portable sur Windows server 2003

Un test a été fait en mettant aussi les adresses « Your IP » et « Next server » à zéro, mais cela n'a rien changé. Un problème persiste donc lorsque la carte est branchée sur un Windows server 2003.

7. Démonstrateur

7.1 Objectif

L'objectif de la démonstration est d'acquérir des données d'un circuit analogique/digital, de les envoyer sur un serveur via la carte Ethernet, de les enregistrer dans un fichier puis de les lire sur une page web.
Dans le principe, la carte s'annonce avec un « Notify ». Le programme envoie ensuite une requête sur l'adresse IP et le port contenu dans le « Notify ». Puis la carte commence ensuite à émettre les données lues sur le circuit a/d. Le programme reçoit ces données et les enregistre dans un fichier sur un serveur web. A l'aide d'une page web en PHP, les données vont être lues et affichées à l'écran.
Donc dans une vue plus large, on pourra visualiser sur une page web, la tension et le courant mesurés sur une résistance par exemple.

7.2 Notify

Dans un premier temps, la carte doit pouvoir s'annoncer sur un réseau. Pour cela le protocole SSDP va être implémenté. Ce protocole permet à un périphérique ou un service de s'annoncer sur un réseau en fournissant son adresse IP et un numéro de port. L'émission de ce paquet se fait en multicast à l'adresse : 239.255.255.250, sur le port 1900.

7.2.1 Conception

Ce contrôleur doit donc pouvoir envoyer un message bien précis et défini par le protocole SSDP (Simple Service Discovery Protocol). L'envoi d'un Notify correspond à une trame précise ayant des champs et un contenu précis. Il faut donc faire en sorte d'émettre une trame susceptible d'être reconnue et interprétée de manière correcte par les autres périphériques du réseau. Cela ne doit évidemment pas risquer de perturber le réseau ou d'être identique à une trame envoyée par un autre périphérique.

Voilà le message qui va être envoyé en ASCII :

NOTIFY * HTTP/1.1
Host:239.255.255.250:1900
NT:urn:hes-so:device:FPGA:2.1
NTS:SSDP:Alive
Location:000.000.000.000 :0000
USN:uuid:08b3c5d1-2ba7-2011-ef34-cf13ea89b6ba::urn:hes-so:device:FPGA:2.1
Cache-Control:max-age=60

Le champ Host contient l'adresse multicast et son port. Le champ NT (Notification Type) contient un URN (Uniform Ressource Name) comprenant : le nom de l'entreprise, suivi du type de périphérique (p.ex. device ou service), suivi de son nom et de sa version. Le champ NTS définit si on active ou désactive le service. Le champ Location, qui est celui qui nous intéresse le plus, donne l'adresse IP et le port duquel on accède au service. L'USN (Unique Service Name) contient UUID

(Universally Unique IDentifier). Cet UUID devrait être formé d'un numéro aléatoire, d'un Timestamp et de l'adresse Mac de la carte, cela, afin de garantir que ce périphérique soit unique. Ici, il a été défini au hasard en ayant tout de même une structure correcte. Le champ USN contient aussi l'URN. Dans le champ Cache-Control, la variable « max-age » définit la durée de la validité des données en secondes (ici 60).

Ce message est préparé en ASCII et en binaire dans un fichier texte. Dans ce fichier, chaque ligne contient uniquement un caractère ASCII, soit 8 bits. Ce fichier va servir à initialiser une mémoire double port (**Figure 37**) dans le circuit. Comme il contient 227 lignes de 8 bits, une mémoire de 256 bits sera suffisante. Il faut donc compléter le fichier d'initialisation jusqu'à la ligne 256.

Ce message ne comprend pas l'adresse IP et le numéro de port. Ceux-ci sont ajoutés plus tard. L'adresse IP est prise directement sur le contrôleur DHCP et le numéro de port est dans un registre. Ces valeurs doivent être transformées de valeur binaire à BCD, puis de BCD à ASCII.

Pour cela, dans le bloc « controller_notify », il y a un bloc appelé « write_ip_controller » (**Figure 37**). Son rôle est de mettre à jour dans la mémoire l'adresse IP, quand elle change, ainsi que le port.

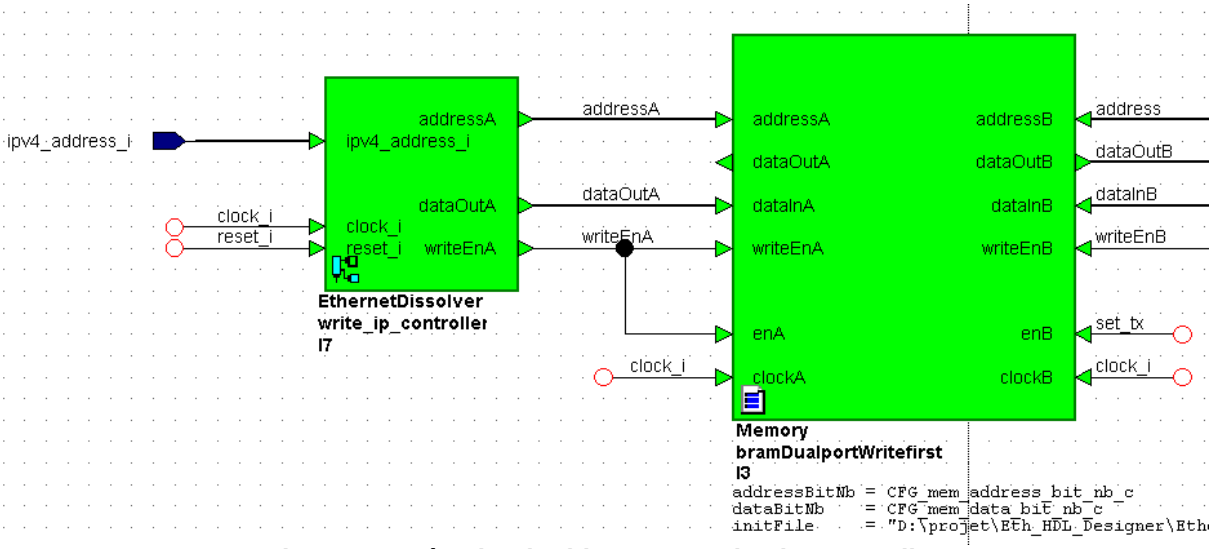


Figure 37: Mémoire double port et write_ip_controller

Dans ce bloc (**Figure 38**), l'adresse IP est séparée en 4 fois 8 bits avant d'être envoyée sur les blocs qui vont les séparer « digit » par « digit ». Cette séparation en 4 est nécessaire étant donné le format d'une adresse IP (p.ex. 153.109.5.22).

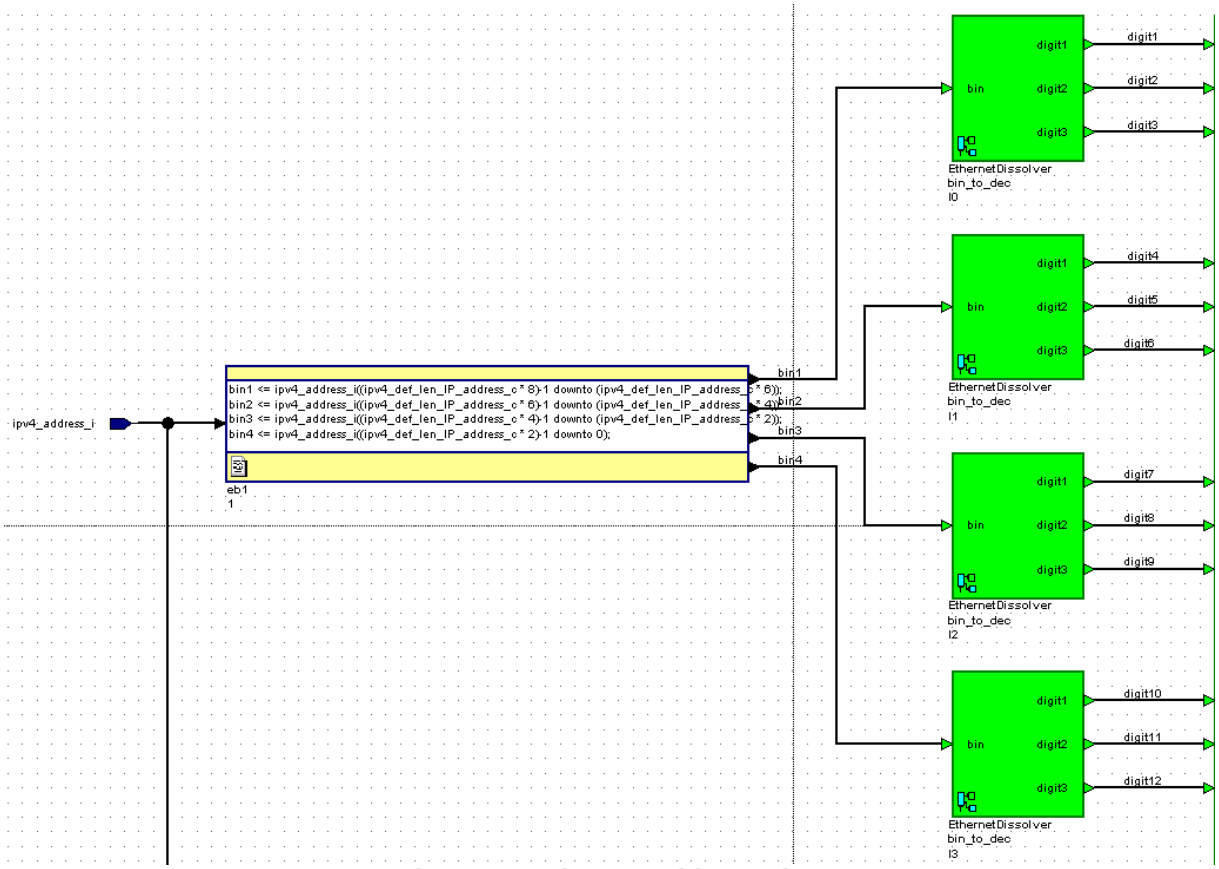


Figure 38: Adresse IP bin_to_dec

Le port va aussi être séparé digit par digit (**Figure 39**).
Afin de ne pas mettre à jour inutilement la mémoire, chaque nouvelle adresse IP est détectée (**Figure 39**). Ce bloc commande un « set_reset » qui, lui, va commander la machine d'état.

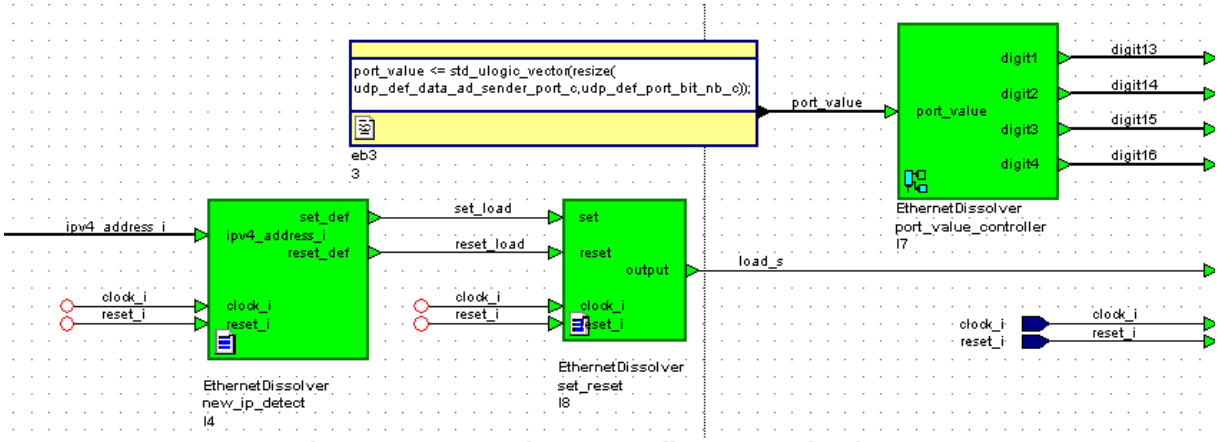


Figure 39: Port_value_controller et new_ip_detect

Cette machine d'état (**Figure 40**) va répartir chaque « digit » à la bonne adresse de la mémoire. Les données sortant de cette machine d'état sont ensuite transformées en caractères ASCII. Elle génère aussi un *enable* pour activer la mémoire et lui dire de faire une écriture.

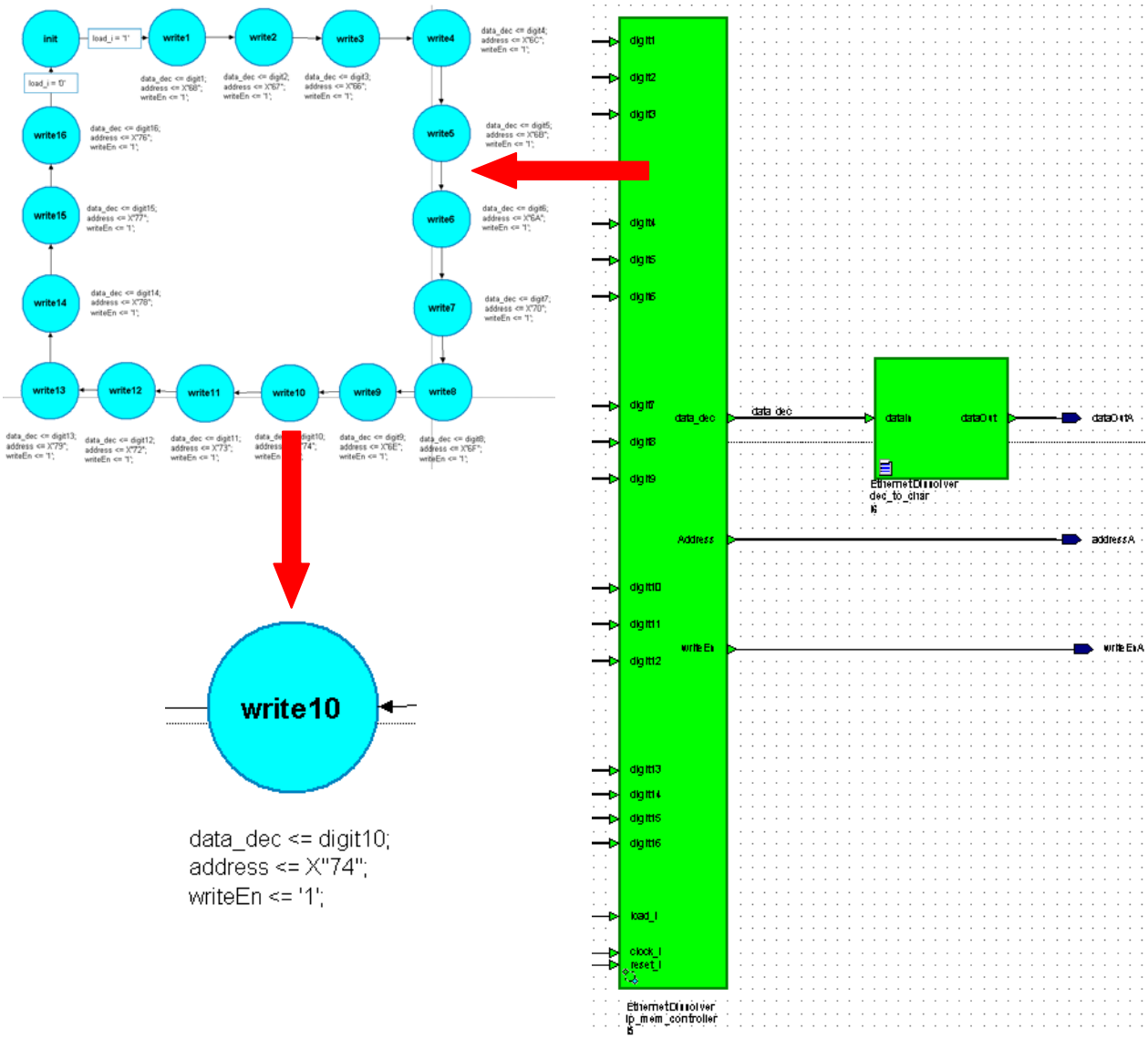


Figure 40: Machine d'état et convertisseur ASCII

Afin d'émettre le Notify chaque tant de temps, un compteur est utilisé. En changeant la taille du compteur, l'intervalle de temps entre deux trames peut être modifié. L'intervalle est défini comme suit :
Avec un clock de 66MHz, pour un envoi toutes les 15 secondes, il faut compter 15 * $66*10^6$ soit $990*10^6$. En binaire cela représente un nombre sur 30 bits. Comme la précision du temps n'est pas essentielle, le compteur ne sera pas arrêté à la valeur $990*10^6$ précise. Il comptera donc jusqu'à 2^{30}-1 soit 1073741823 ou environ 16.2688 secondes ; donc 31 bits = ~32 sec, 32 bits = ~65 sec, 29 bits = ~8 sec, etc.
Un bit de plus ou de moins influence d'un facteur 2 le temps. La constante du nombre de bit est défini dans le package « udp_definition_pkg ».

Ce compteur (**Figure 41**) est mis à zéro et démarré lorsque le contrôleur Notify est sélectionné. Il est stoppé automatiquement lorsqu'il a fait le tour et revient à zéro. Sa valeur de sortie sert à piloter les adresses de la mémoire, seuls les 8 bits de poids faible sont utilisés comme la mémoire est sur 8 bits. Pour l'envoi, il faut signaler à quel moment l'adresse est valide. Comme le fichier comporte 227 lignes, le signal « max_data_address » avertit lorsque le compteur est plus petit que 227, ce qui signifie que les données sont valides.

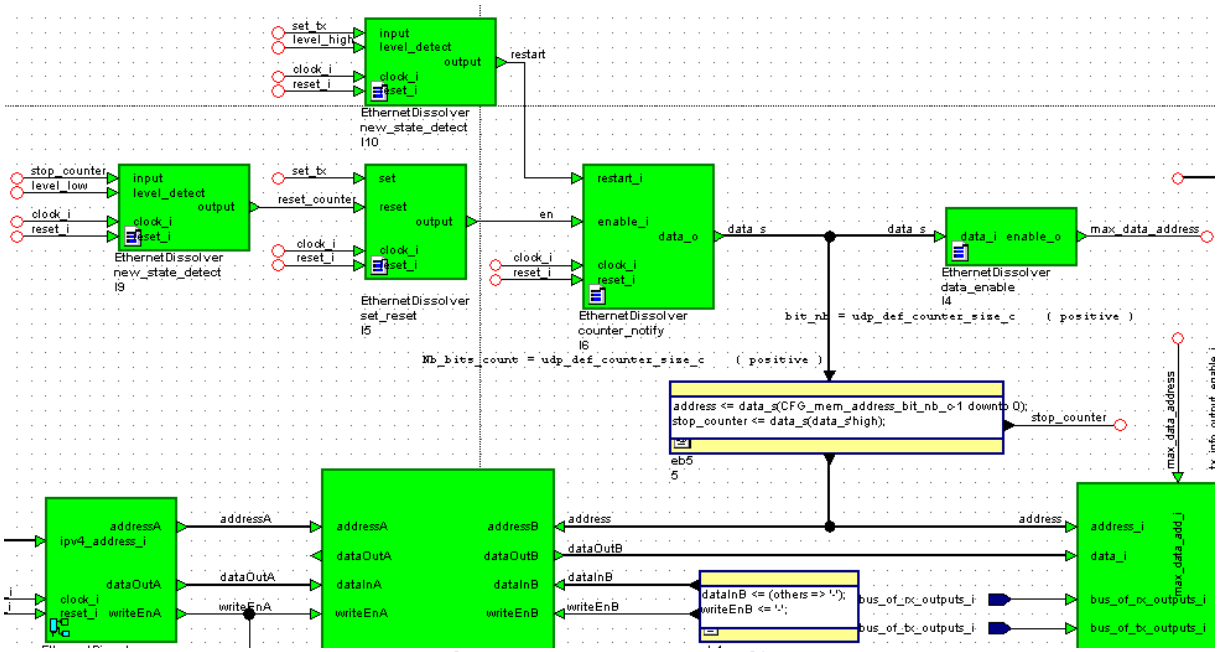


Figure 41: Compteur Notify

Lorsque le contrôleur est activé, les données en mémoire sont envoyées. La machine d'état « controller_notify_heart » (**Figure 42**) a pour rôle de charger les données à envoyer sur le bus « tx_input_o ».

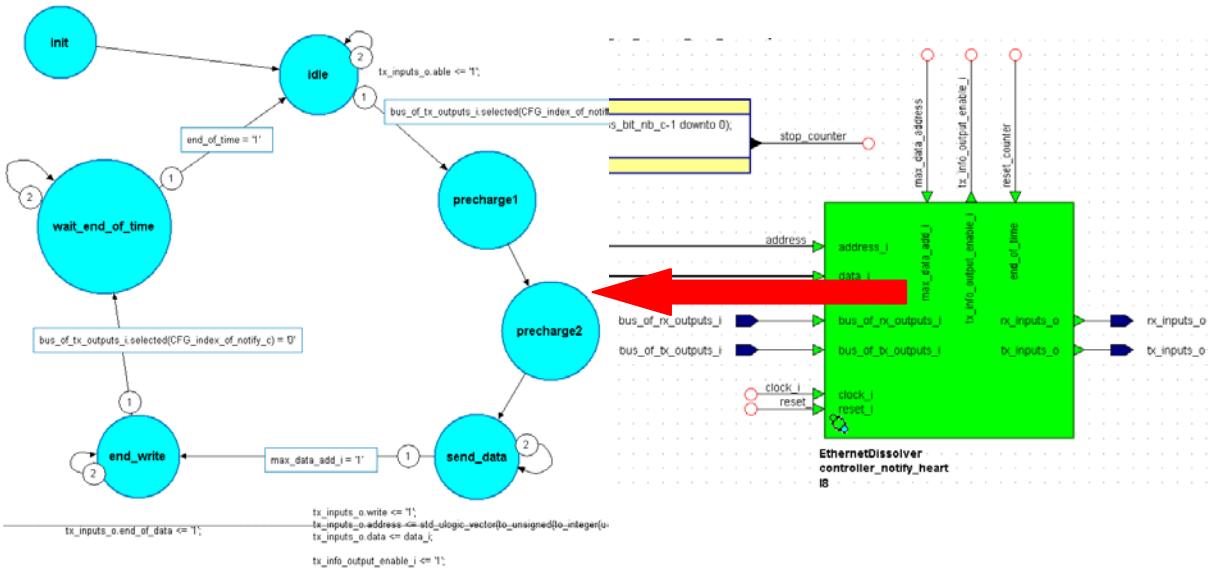


Figure 42: Machine d'état du controller_notify_heart

Le fonctionnement de cette machine d'état est relativement simple. Dans l'état « idle », elle signale qu'elle veut écrire des données. Dès que le contrôleur est sélectionné, elle passe dans les deux états de précharge, qui synchronise le compteur, les adresses et les données. L'état « send_data » met les données et les adresses correspondantes sur le bus. Elle le fait jusqu'à ce que toutes les données soient envoyées. Elle signale ensuite la fin de l'envoi et attend d'être désélectionnée. Elle attend ensuite que le compteur repasse à zéro avant d'envoyer une nouvelle trame.

Le bloc « controller_notify _tx_info_manager » (**Figure 43**) charge les adresses IP et MAC ainsi que le port UDP, et signale aux autres blocs que les données sont valides.

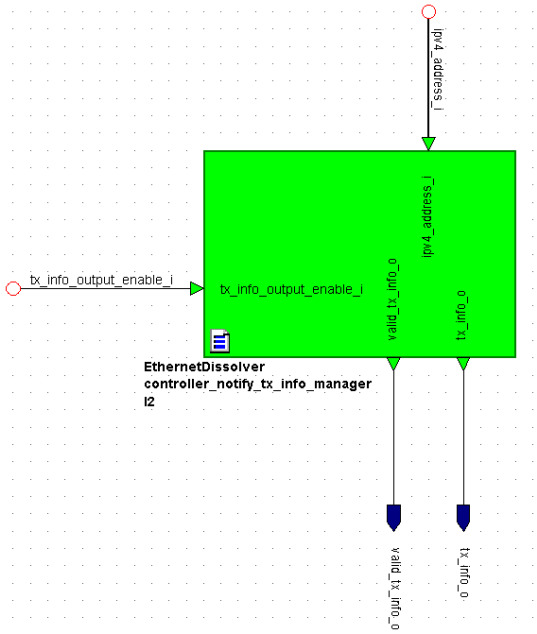


Figure 43: Controller_notify_tx_info_manager

Pour le contrôle du VLAN, le bloc est similaire aux autres contrôleurs.

7.2.2 Simulation

En simulation, différents points doivent être vérifiés, tels que : le bon comportement du compteur, le chargement de l'adresse IP et du port en mémoire, et l'envoi de la trame « Notify ».

Pour commencer, le chargement de l'adresse IP et du port va être vérifié (**Figure 44**). Une nouvelle adresse est envoyée depuis le DHCP. Le signal « load_s » informe la machine d'état qu'il faut mettre à jour la mémoire. La ligne « ipv4_address_i » montre la nouvelle adresse en hexadécimal. Les signaux « bin » montre la même adresse séparée par nombres. Ces nombres sont ensuite transformés et séparés chiffre par chiffre (p.ex. 153 devient 1 5 3). La machine d'état va mettre ces chiffres dans la mémoire en commençant par bin1 jusqu'à bin4 puis le port. Elle envoie en premier le chiffre de poids faible. Par exemple pour le 153, le « write 1 » envoie le 3, le « write 2 » le 5 et le « write 3 » le 1. Pendant toute la durée des « write », le signal « writeEnA » est à '1'. Il active la mémoire en écriture.

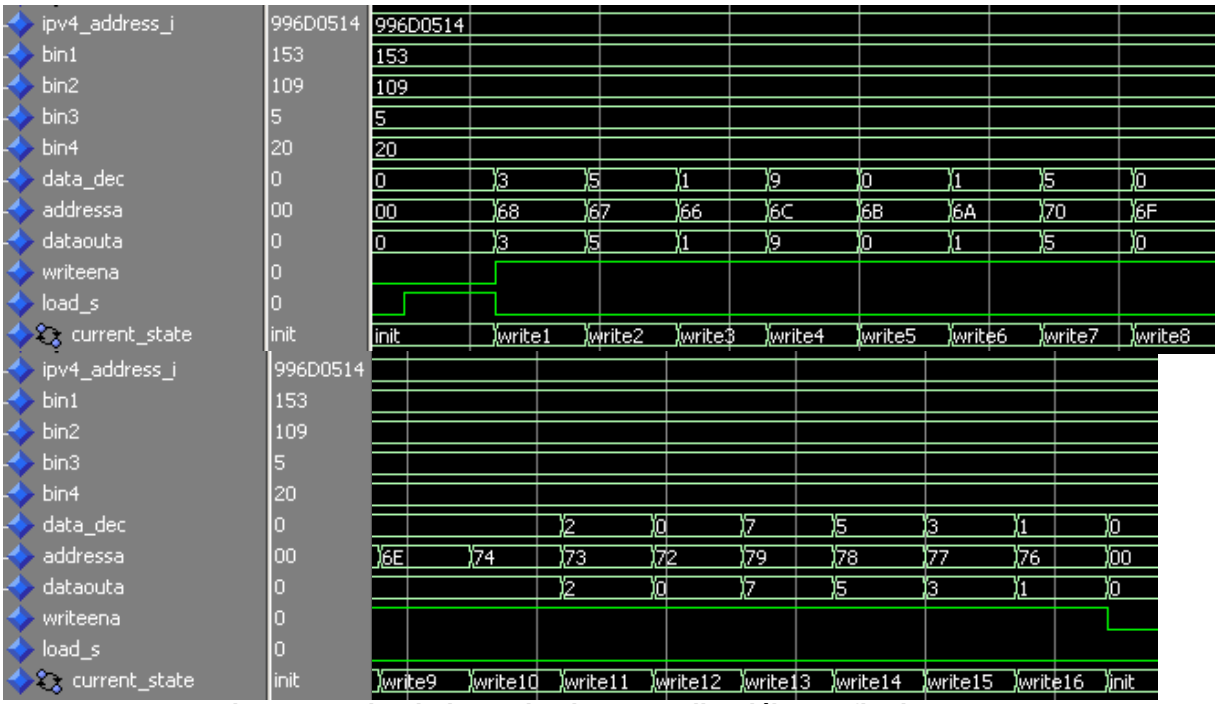


Figure 44: Simulation write_ip_controller début et fin de trame

Sur la **Figure 45**, le signal « set_tx » informe que le bloc peut envoyer un « Notify ». Il génère aussi le signal « restart » qui remet à zéro le compteur et active le « en » qui fait tourner le compteur. La valeur du compteur est affichée sur « data_s ». La machine d'état se met en mode « send_data ». Elle génère le signal « tx_info_output_enable_i ». Elle envoie aussi les données et les adresses sur le bus « tx_inputs_o ». Sur ce bus, on trouve donc dans les données les datas provenant de la mémoire et sur l'adresse, la valeur du compteur - 1.

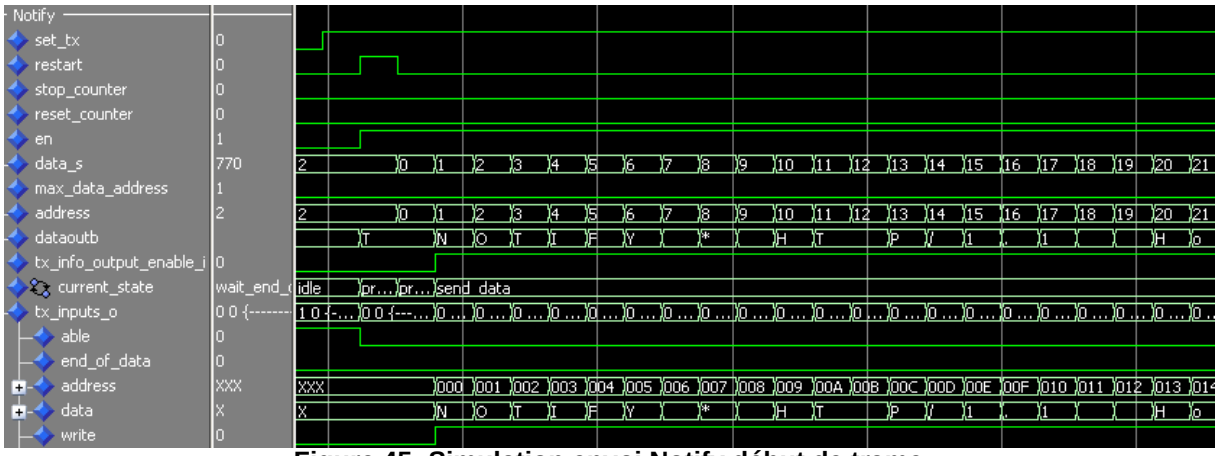


Figure 45: Simulation envoi Notify début de trame

Sur la **Figure 46**, on peut lire dans les datas « Location :153.109.005.020 :1357 », ce qui signifie que la mise à jour de la mémoire fonctionne correctement et que la position des chiffres est correcte.

Figure 46: Data location

Sur la **Figure 47**, on voit que la fin des données émises est correcte avec le « max_age=60 » suivi de deux retours ligne. La valeur 227 atteinte, le signal « max_data_address » s'active et fait changer d'état la machine qui passe alors en mode « end write ». Cet état avertit les blocs supérieurs que l'émission est terminée par le signal « end_of_data ». Il met aussi à '0' les différents *enable*.

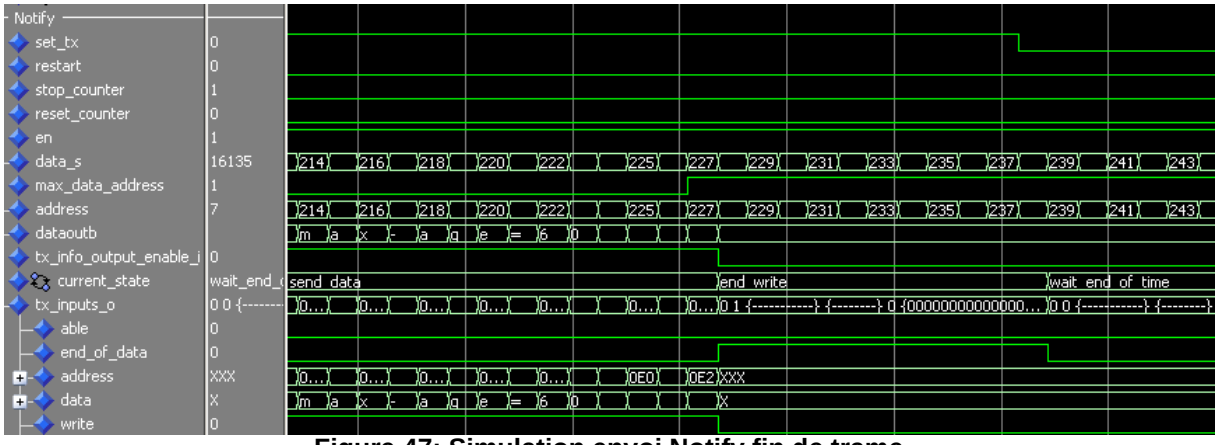


Figure 47: Simulation envoi Notify fin de trame

La machine d'état attend ensuite que le signal « set_tx » passe à '0'. Il se met alors en mode « wait_end_of_time » et attend que le compteur repasse à zéro.

Il faut encore vérifier que le bus « tx_info_o » charge les bonnes adresses UDP, IP et MAC. La **Figure 48** prouve cela. Les 2 ports sont à 1900, l'adresse IP source est celle de la carte et la destination est la multicast (EF FF FF FA ou 239.255.255.250). L'adresse MAC source est aussi celle de la carte, et la destination est celle multicast, définit par la norme (01 00 5E 7F FF FA).

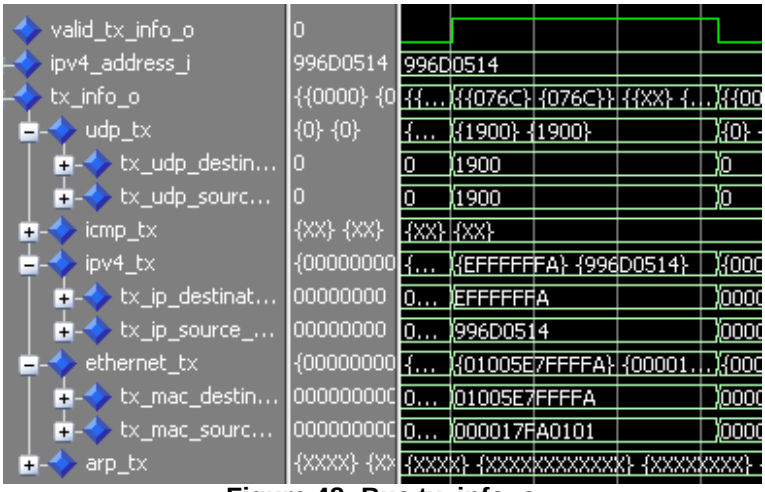


Figure 48: Bus tx_info_o

7.2.3 Test sur carte

Pour tester le « Notify », il suffit de brancher la carte, d'ouvrir le programme Wireshark et de lancer une capture sur la carte réseau où est branchée la carte. Il faut vérifier qu'un « Notify » est bien reçu, que son contenu soit correct et que le temps entre deux « Notify » soit le bon.

La **Figure 49** est une capture Wireshark des trames reçues par la carte. Le contenu correspond exactement à celui défini plus haut au point 7.2.1. L'adresse IP est encore à zéro, la carte n'en a pas encore reçu.
Pour ce qui est du temps entre 2 trames, il suffit de soustraire deux des valeurs dans le champ Time. Par exemple : 87.898323- 71.629908 = 16.268415 qui est très proche de la valeur calculée de 16.2688155, soit une différence de 0.4005msec. L'intervalle de temps est donc correct.

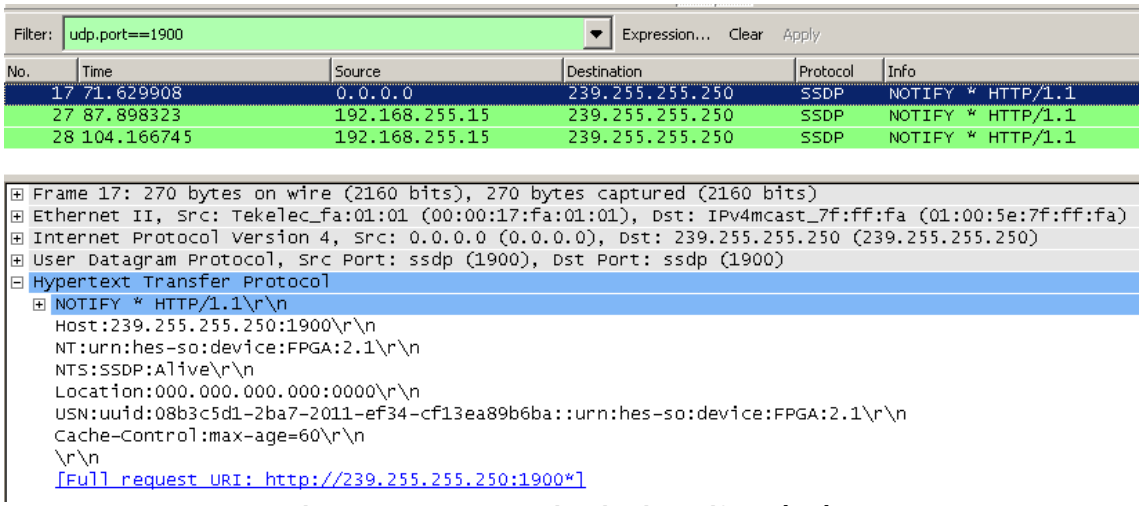


Figure 49: Capture Wireshark Notify IP à zéro

Sur la **Figure 50**, l'adresse IP a été mise à jour par le serveur DHCP. Le « Notify » envoie donc son adresse IP et le port dans le champ « Location ».

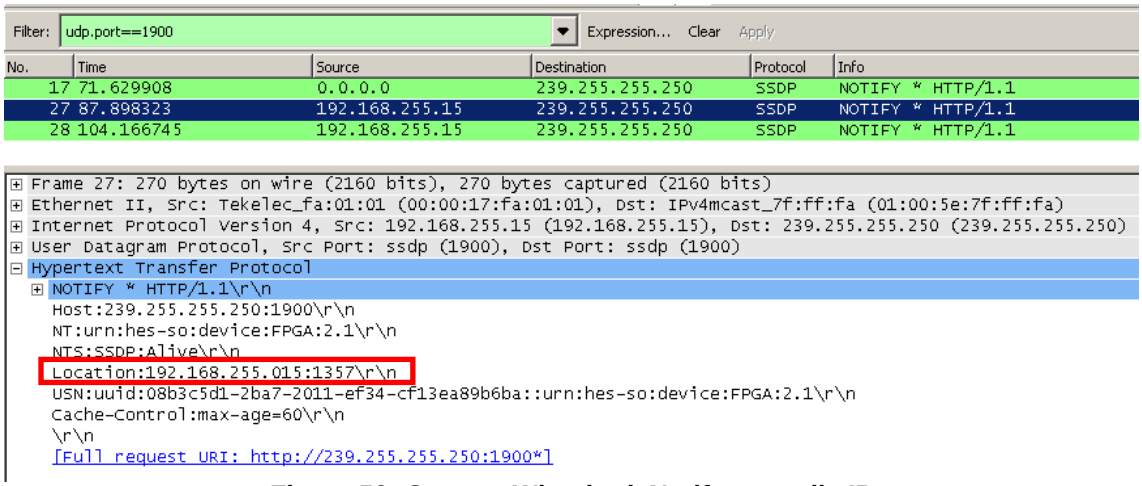


Figure 50: Capture Wireshark Notify nouvelle IP

Sur la **Figure 51**, le mode VLAN a été forcé. Wireshark reconnaît donc une trame taguée avec un ID de 2468 et une priorité à zéro soit « Best Effort ».

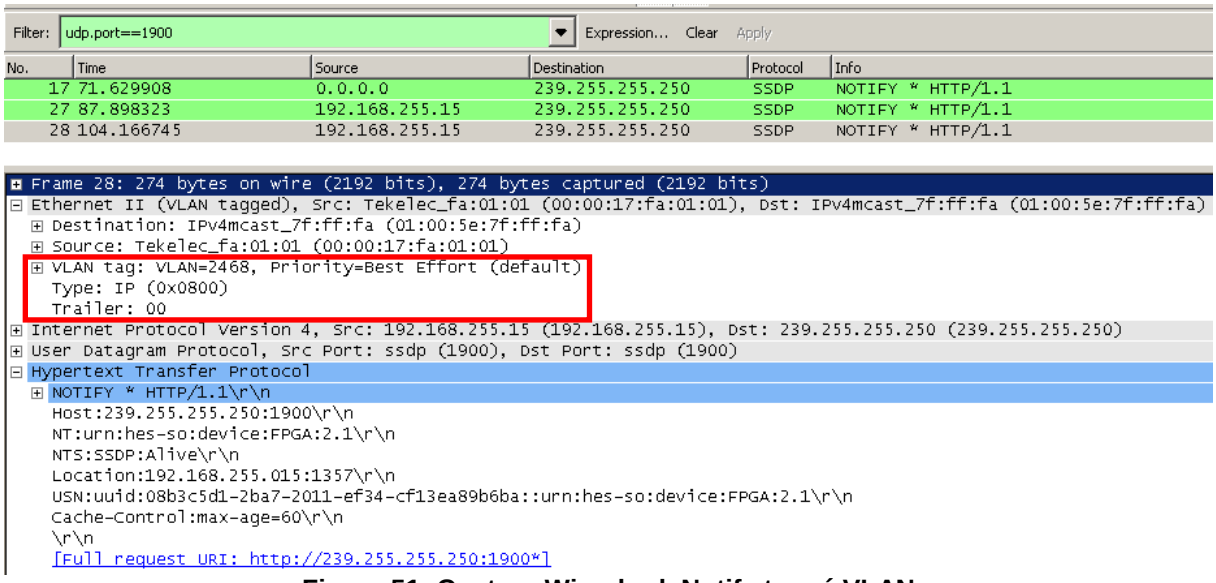


Figure 51: Capture Wireshark Notify tagué VLAN

7.3 Data a/d sender

Ce contrôleur doit permettre de récupérer des données venant de l'extérieur et de les transmettre sur le réseau via le protocole UDP. Dans un premier temps, il attendra une requête extérieure avant d'émettre quoi que ce soit. Une fois cette demande reçue, le contrôleur enverra des paquets avec le contenu des valeurs récupérées d'un convertisseur a/d. Ces valeurs seront stockées dans une mémoire double port en attendant d'être émises régulièrement (p.ex. toutes les secondes).

7.4 Software

Sur le pc, un serveur DHCP et un serveur web seront installés sur une machine virtuelle. Un petit programme récupèrera les données émises pour les mettre dans un fichier. Une page web actualisée régulièrement permettra la lecture des données du fichier et les affichera.

8. Conclusion

En conclusion, le circuit actuel permet, en plus, l'envoi et la réception de paquets VLAN ou non.
Il peut aussi s'annoncer sur le réseau selon le protocole SSDP. Il émet donc des « Notify » régulièrement dans un temps défini au préalable, en multicast à l'adresse : 239.255.255.250, sur le port 1900. Dans ce « Notify », sont envoyés l'adresse IP de la carte ainsi que le port d'où on peut y accéder.
Il peut encore lire des données extérieures et les envoyer en UDP lorsqu'on lui en fait la requête sur un port défini. Ce port est celui donné dans le « Notify ».

Il y a toutefois un problème avec le DHCP, il ne fonctionne apparemment pas comme il faut sur Windows server 2003. Windows server ne donne pas l'impression de recevoir la « Request » ou l'interprète mal. Cela pourrait venir des options choisies dans la trame de réponses, à vérifier !

Cet apport permettra à l'avenir une meilleure compatibilité avec les différents réseaux que l'on peut trouver. Il suffit donc ainsi de brancher la carte sur un réseau pour connaître immédiatement son adresse, depuis n'importe quel ordinateur du réseau, sans avoir accès à au serveur.

Figures

Annexes

1. Diagramme de réception d'une trame VLAN
2. Diagramme d'émission d'une trame VLAN

Sources

1. IEEE 802.1Q (VLAN) : http://fr.wikipedia.org/wiki/802.1Q
2. IEEE 802.1P (Priorité) : http://fr.wikipedia.org/wiki/IEEE_802.1p
3. DHCP : http://fr.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol
4. Ethernet : http://fr.wikipedia.org/wiki/Ethernet
5. OSI : http://www.frameip.com/osi/
6. SSDP : http://en.wikipedia.org/wiki/Simple_Service_Discovery_Protocol
7. UPNP: http://fr.wikipedia.org/wiki/Universal_Plug_and_Play

Matériel

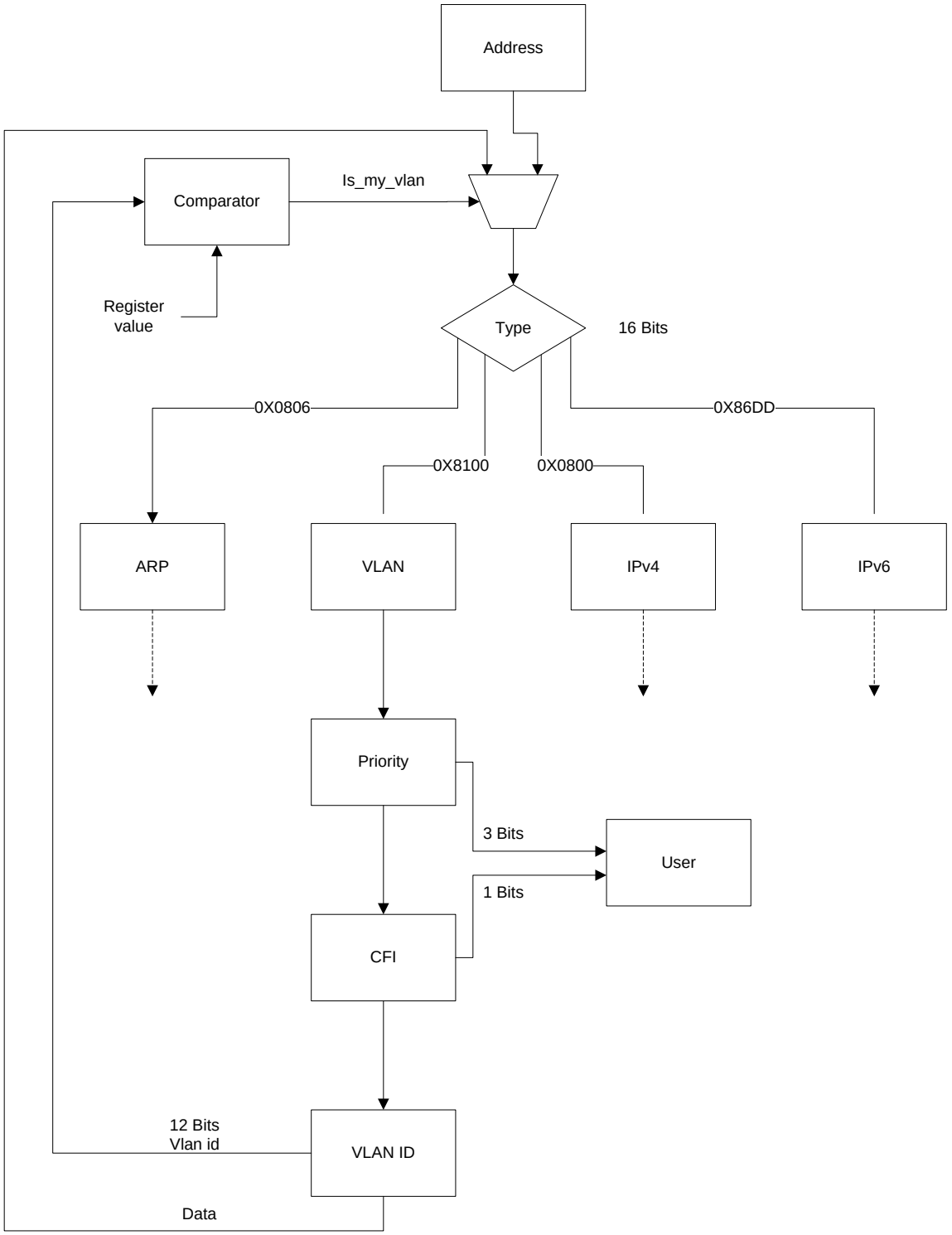
1. Pc Dell
2. Carte FPGA V2.1 board 34
3. Carte réseau PCI
4. Switch 3300

Logiciels

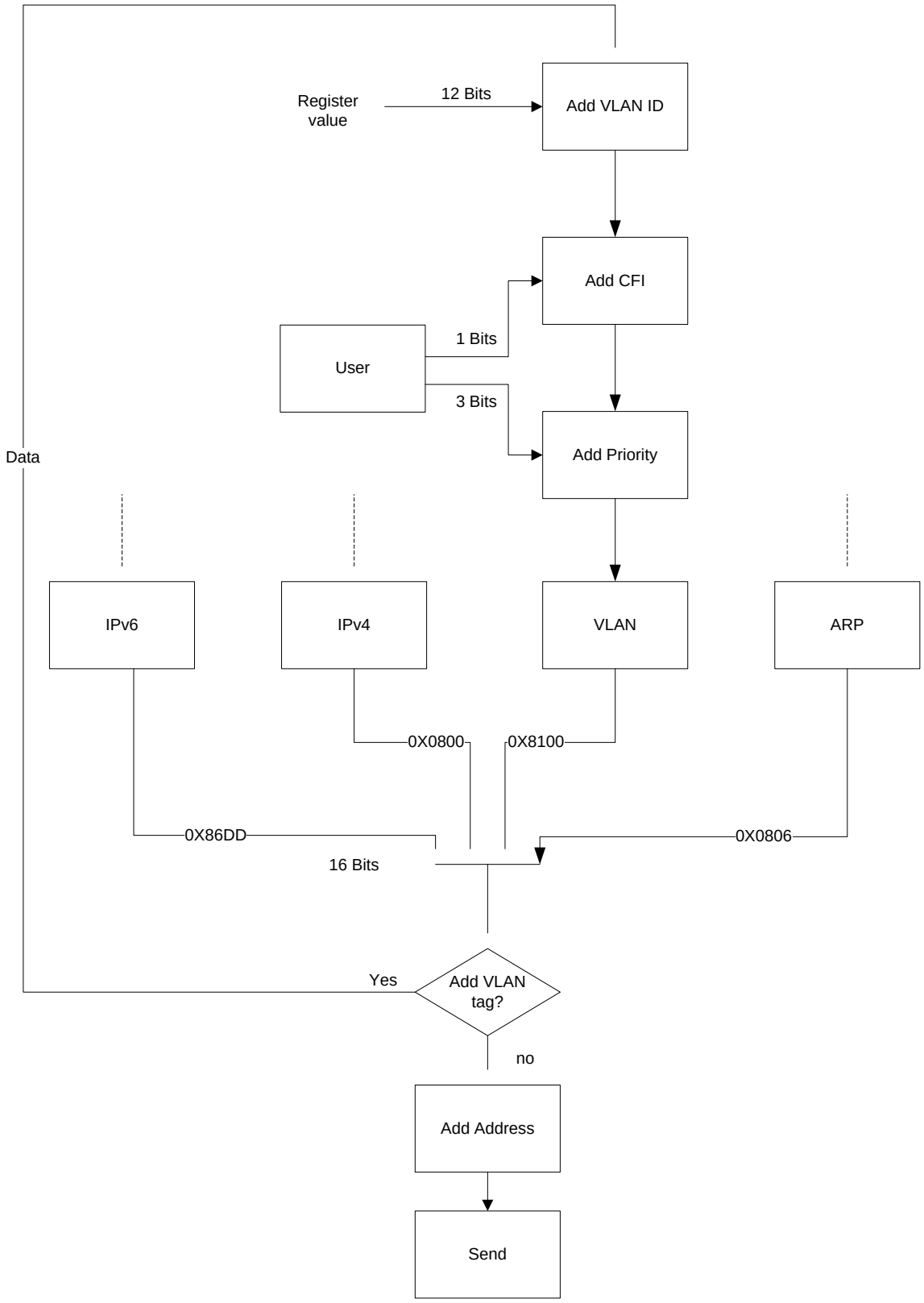
1. HDL Designer 2009.2
2. Modelsim SE 6.6a
3. Xilinx ISE Design Suite 12.1
4. Oracle VM VirtualBox 4.0.4
5. Windows server 2003 et XP
6. Wireshark 1.2.1
7. Ubuntu

Sion, le 29 juillet 2011 Héritier François

Annexe 1


Address
Comparator
Is_my_vlan
Register value
Type
16 Bits
0X0806
0X86DD
0X8100
0X0800
ARP
VLAN
IPv4
IPv6
Priority
3 Bits
User
1 Bits
CFI
12 Bits
Vlan id
VLAN ID
Data

Annexe 2


Register value
12 Bits
Add VLAN ID
Add CFI
User
1 Bits
3 Bits
Add Priority
Data
IPv6
IPv4
VLAN
ARP
0X0800
0X8100
0X86DD
0X0806
16 Bits
Yes
Add VLAN tag?
no
Add Address
Send