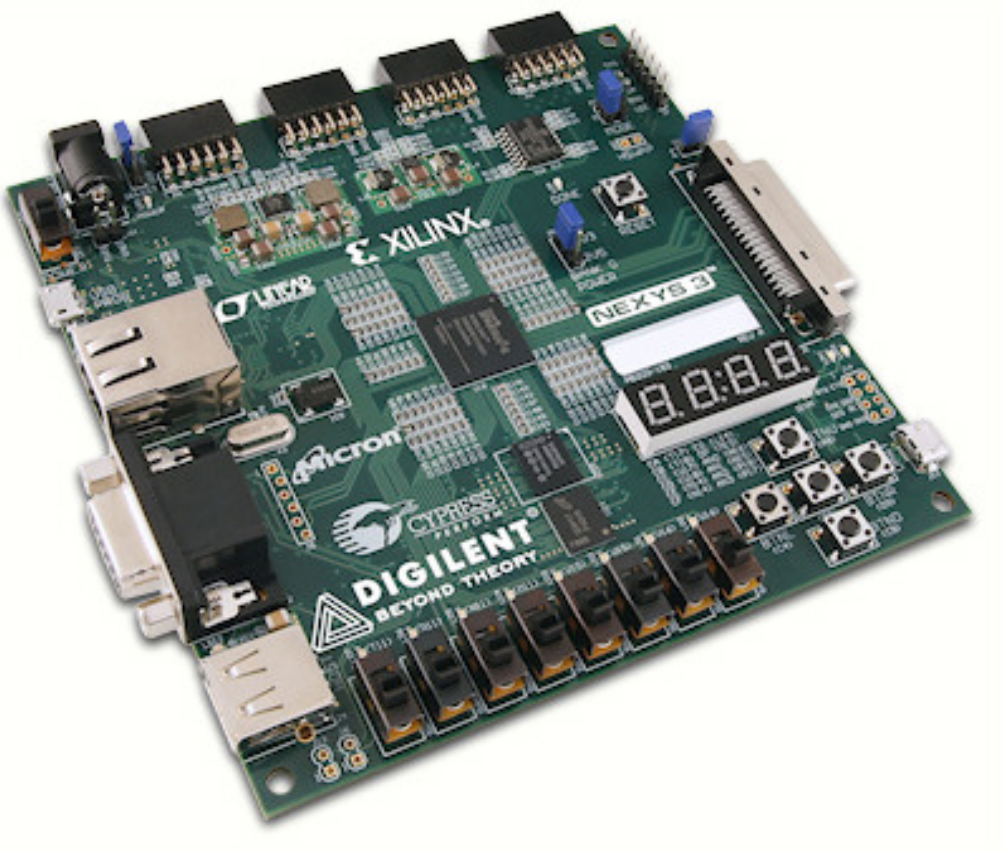
Rapport du BE MAC

Conception d'une carte Ethernet en VHDL



4IR-RT
Année Universitaire 2014-2015

MABILLE Gabriel
VENDOME Jean-Christophe

Sommaire

MABILLE Gabriel
VENDOME Jean-Christophe

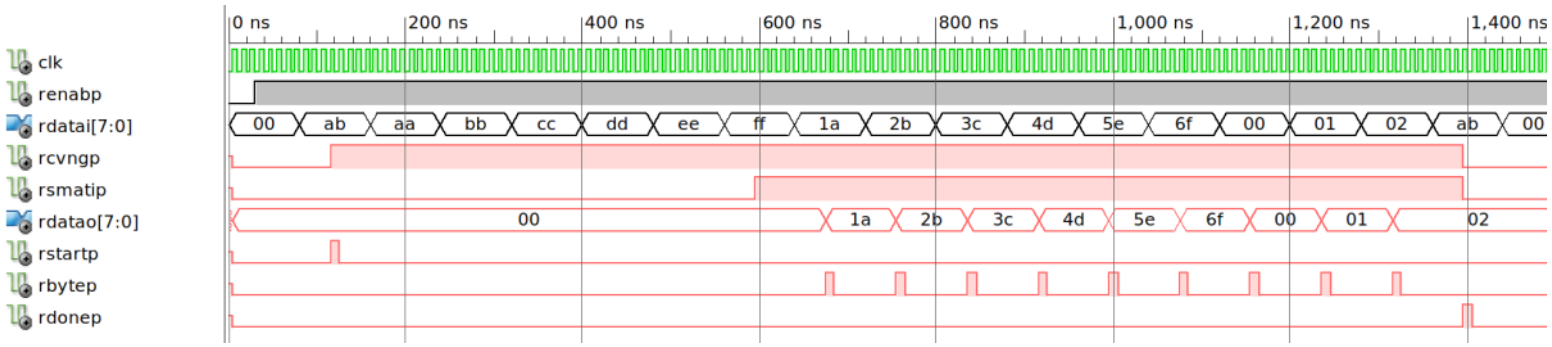
Introduction

Durant le BE MAC nous avons décrit en VDHL une carte Ethernet simplifiée faisant l'interface entre la couche physique et la couche réseau. Pour se faire, nous avons suivi le manuel technique de l'Ethernet-10 core, en éliminant toutefois des signaux.
Notre carte est capable de recevoir une trame Ethernet et de la faire remonter. Elle peut aussi transmettre les données fournies par la couche supérieure en les encapsulant. Notre carte est de plus capable de détecter une collision et de mettre en place un système de Backoff, c'est-à-dire un temps d'attente avant de permettre la ré-émission d'une trame en cas de collision. Enfin, nous lui faisons lever un flag pour indiquer qu'il y a eu plusieurs collisions.
Dans ce rapport, nous allons inclure des chronogrammes montrant ces différents points. Nous commencerons par la partie réception, nous continuerons sur la partie transmission et finirons sur la partie détection de collision et algorithme de Backoff.

I - Réception

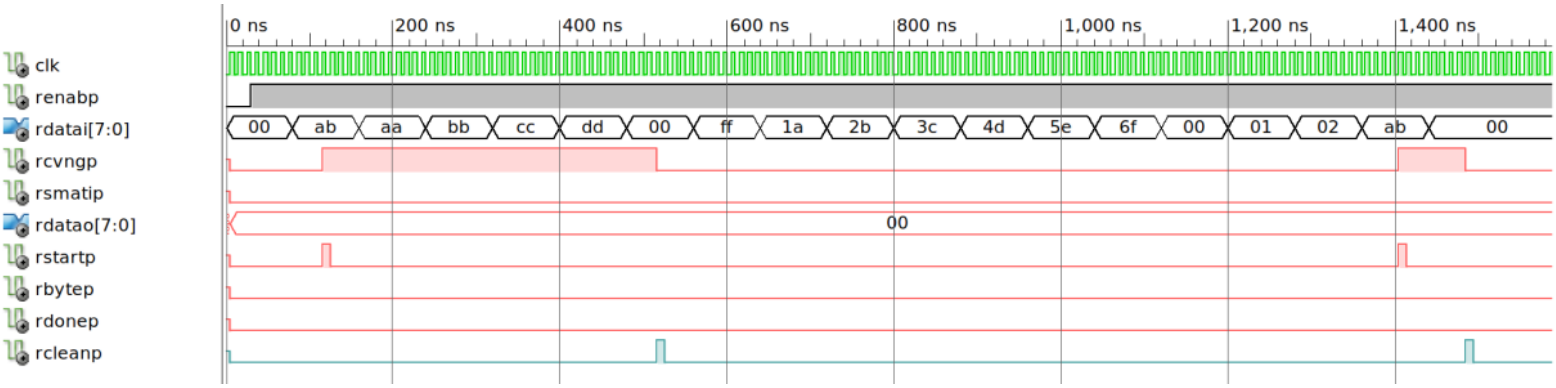
A. Description de la réception

Nous avons commencé par nous attaquer à la réception, cette partie étant plus simple à décrire que la transmission. Nous supposerons que l'adresse MAC de notre carte Ethernet est AA:BB:CC:DD:EE:FF.



Réception d'une trame qui nous est adressée

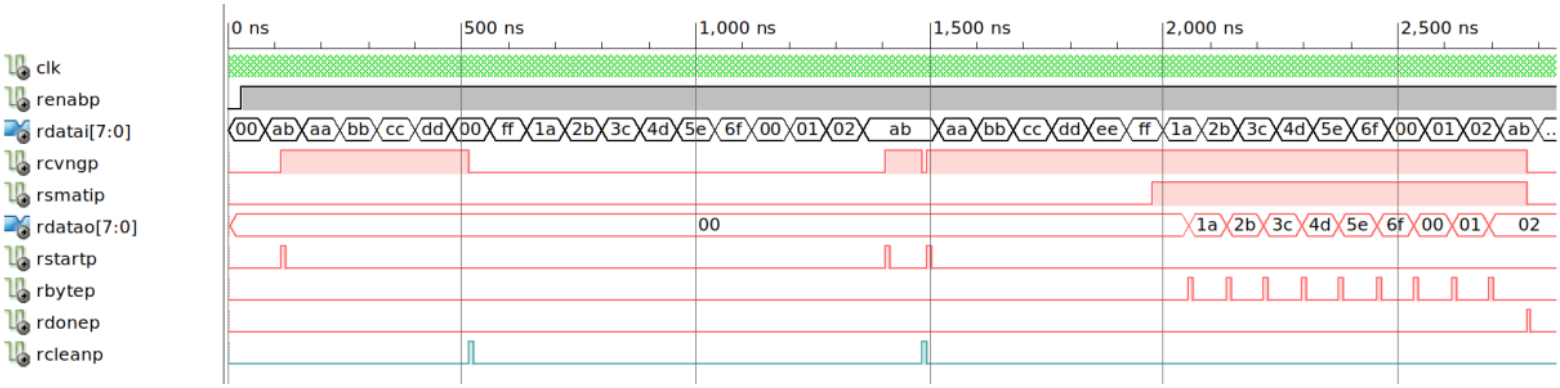
Le chronogramme ci-dessus montre la réception d'une trame qui nous est adressée. Que se passe-t-il concrètement ?
Tout d'abord la couche supérieure permet la réception en mettant RENABP à l'état HIGH. Lorsqu'on lit le Start Frame Delimiter (SFD), on indique à la couche supérieure qu'une trame arrive avec RCVNGP à l'état HIGH et on pulse RSTARTP. Comme l'adresse lue, ici AA:BB:CC:DD:EE:FF, est la nôtre, on met RSMATIP à HIGH, pour indiquer à la couche supérieure que la trame nous est destinée. Ensuite, pour chacun des octets lus, on le place dans le buffer de sortie RDATAO et on pulse RBYTEP pour indiquer à la couche supérieure qu'elle peut lire ce nouvel octet. Lorsqu'on reçoit le End Frame Delimiter (EFD), on positionne RSMATIP et RCVNGP à LOW et on pulse RDONEP pour indiquer que la trame a été entièrement reçue.



Réception d'une trame qui ne nous est pas destinée

MABILLE Gabriel
VENDOME Jean-Christophe

Le chronogramme ci-dessus montre la réception d'une trame qui ne nous est pas adressée. Voici la description des événements :
Ici aussi, la couche supérieure permet la réception en mettant RENABP à l'état HIGH. Lorsqu'on lit le SFD, on indique à la couche supérieure qu'une trame arrive avec RCVNGP à l'état HIGH et on pulse RSTARTP. Comme on se rend compte, à partir de l'octet '00', que l'adresse lue, ici AA:BB:CC:DD:00:FF, n'est pas la notre, on pulse RCLEANP pour dire à la couche supérieure qu'elle peut libérer les buffers qu'elle avait peut-être alloués en prévision de la trame actuelle et on repositionne RCVNGP à LOW. On constate toutefois un petit inconvénient, comme notre EFD correspond à la même séquence que notre SFD, notre carte va considérer le EFD de la trame précédente comme le SFD de la prochaine trame. On place donc RCVNGP à HIGH et on pulse RSTARTP, pour constater tout de suite, en lisant le premier octet '00', que l'adresse MAC ne correspond pas et indiquer, en pulsant RCLEANP, que la trame ne nous est pas destinée.



Réception d'une trame qui ne nous est pas destinée
puis d'une autre qui nous est destinée

Enfin, nous nous sommes demandé, si, malgré la détection du EFD comme un SFD, on va bien récupérer une trame qui nous est destinée si elle succède à une trame qui ne nous est pas destinée. C'est ce cas de figure qui est traitée dans le chronogramme précédent. Comme on peut le voir la réception de la seconde trame se passe correctement.

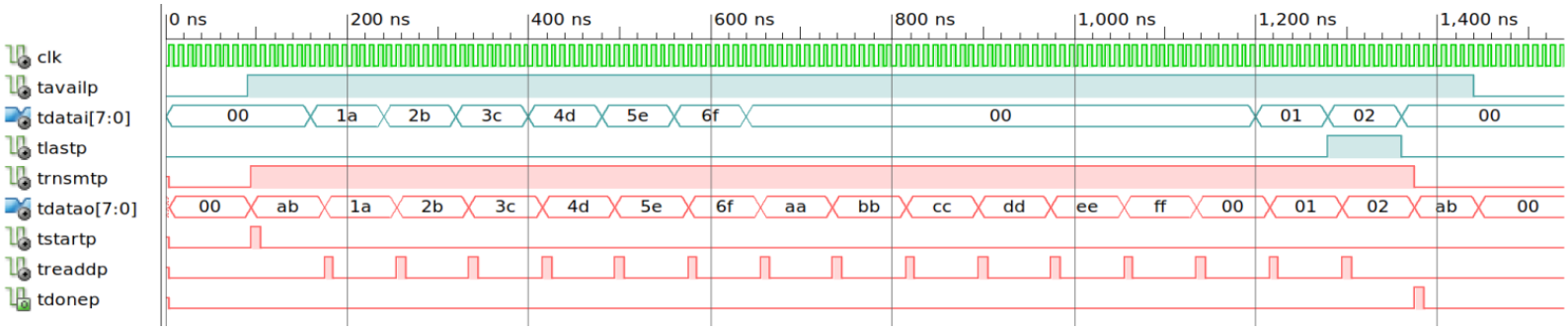
B. Problèmes rencontrés

Lors de cette étape, nous n'avons pas rencontré de problèmes insurmontables, l'essentiel de nos incompréhensions étaient dues au VHDL. Les cours nous ont donné de bonnes bases, mais, comme c'est souvent le cas, ce n'est pas parce que nous avons bien compris en classe, que nous ne rencontrons pas de problèmes lors de la mise en pratique. L'autre problème auquel nous nous sommes confrontés, a été de bien comprendre l'enchaînement des signaux que nous devions manipuler.

II - Transmission

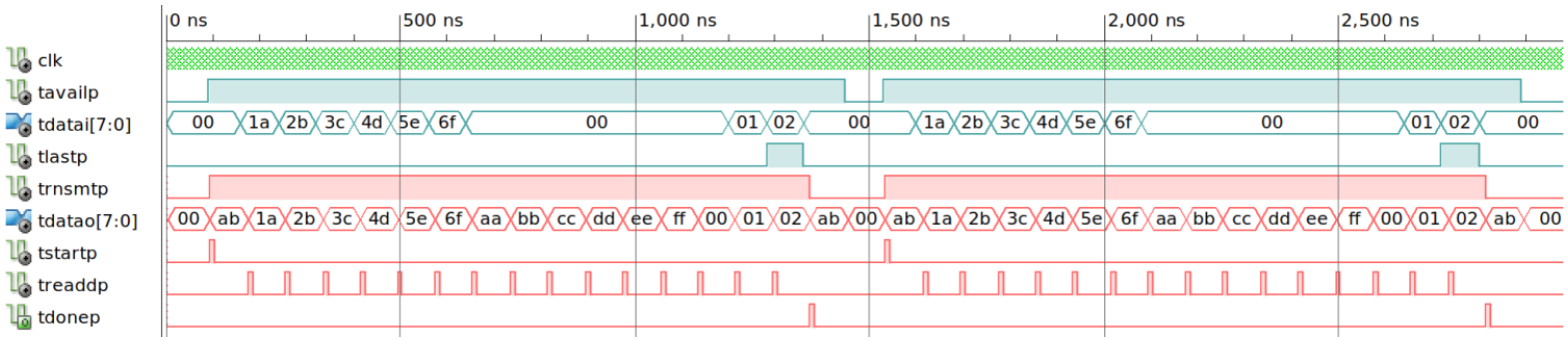
A. Description de la transmission

Nous nous sommes ensuite attaqués à la transmission d'une trame indépendemment de la réception.



Transmission Classique

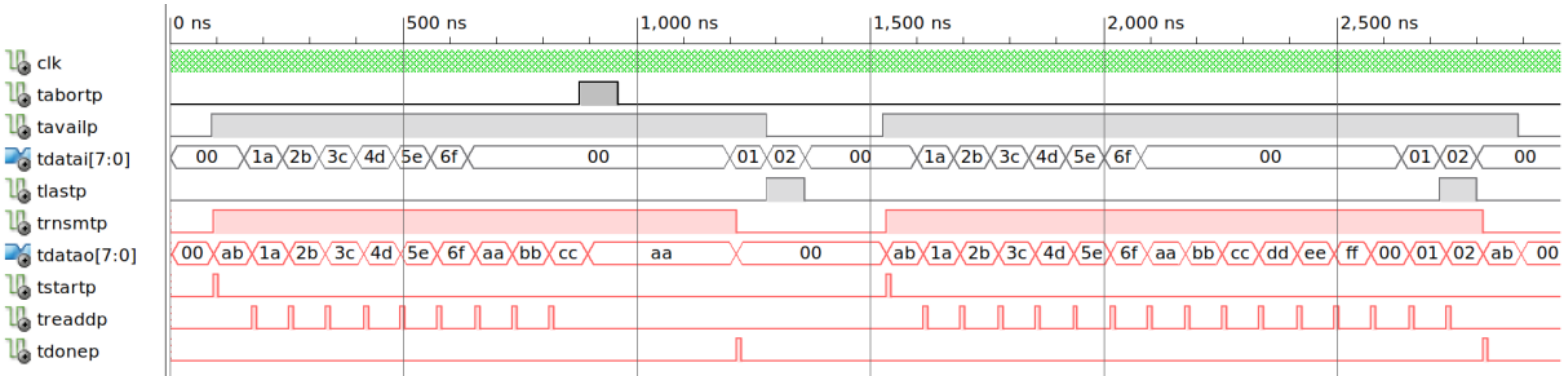
Nous avons donc commencé par décrire en VHDL une réception classique. Une fois de plus, que se passe-t-il concrètement ?
La couche supérieure indique qu'elle souhaite envoyer une trame en mettant TAVAILP à HIGH. On envoie donc le SFD sur TDATAO et indiquons à la couche supérieure que la transmission a commencé en pulsant TSTARTP et en positionnant TRNSMTP à HIGH. Ensuite on récupère l'adresse MAC de destination que la couche supérieure a positionnée dans TDATAI et on la transmet dans TDATAO octets par octets. On indique à la couche supérieure chaque fois qu'on lit et envoie un octet en pulsant TREADDP. Une fois les 6 octets de l'adresse MAC de destination envoyés, on transmet la notre. On indique chaque octet transmis en pulsant TREADDP, même si ça détourne un peu la fonction de ce pin puisqu'on ne lit pas le buffer d'entrée lors de cette étape. Une fois les 6 octets de notre adresse transmis, on lit les datas qui correspondent à la payload de la trame, ici, on a simulé la payload par 3 octets, '00', '01', '02'. Enfin, comme la couche supérieure nous indique que l'octet '02' est le dernier par TLASP à HIGH, on clôt la transmission en envoyant le EFD et en positionnant TRNSMTP à LOW et en pulsant TDONEP.



Multiples Transmissions Classiques

MABILLE Gabriel
VENDOME Jean-Christophe

Par acquis de conscience, nous avons voulu vérifier que plusieurs transmissions consécutives ne posaient pas de problème à notre carte, c'est à dire que le système était bien réinitialisé après une transmission. Tout à l'air de fonctionner comme on peut le voir dans le chronogramme précédent.



Multiples Transmissions - Première Transmission Avortée

Nous avons donc pu nous attaquer au problème suivant, permettre à l'utilisateur d'avorter une transmission. Ce cas de figure peut arriver, si, par exemple, l'hôte n'arrive pas à fournir les données à la vitesse à laquelle la carte les envoie. Dans ce cas, l'hôte, constatant qu'il ne suit pas, peut décider d'arrêter une transmission.
Le début de la transmission est classique. Par contre, on constate que cette fois ci, l'hôte positionne TABORTP à HIGH, ce qui entraîne que notre carte arrête de transmettre ce qu'elle transmettait, il s'agit en l'occurrence ici de notre adresse source, pour transmettre à la place une série de 32 bits alternés de 0 et de 1. On voit donc que notre carte transmet 4 "AA". Pour rappel, '1010 1010' est égal à "AA" en hexadécimal. Comme d'habitude, elle indique à la couche supérieure la fin de la transmission en positionnant TRNSMTP à 0 et en pulsant TDONEP.
Cette fois encore, pour voir si le système se réinitialisait bien, nous avons fait suivre la première transmission avortée par une transmission complète et on constate que tout se passe bien.

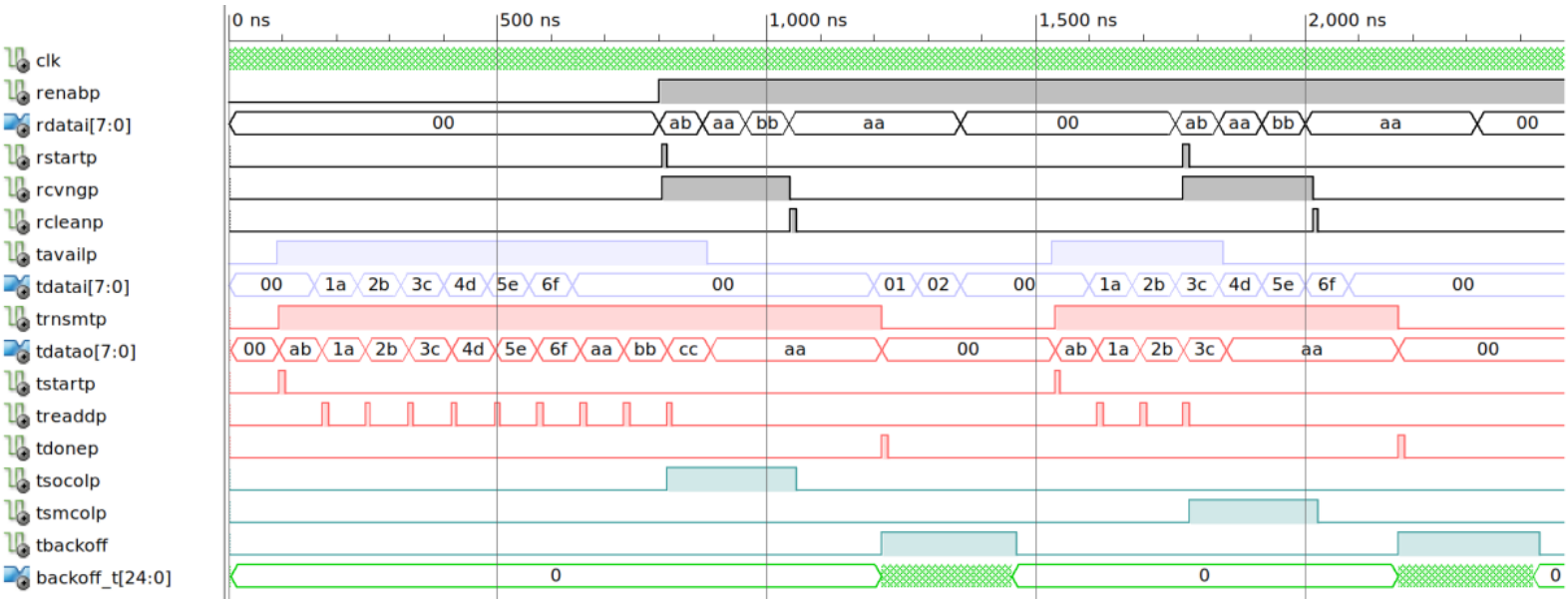
B. Problèmes rencontrés

Nous avons aussi rencontré des problèmes pour savoir quels étaient les signaux à lever et à quel moment. À ce jour, nous avons toujours un doute quant au positionnement du signal TSOCOLP (respectivement TSMCOLP) à HIGH lorsque l'utilisateur positionne à HIGH TABORTP. En effet, il ne s'agit pas à proprement parler d'une collision et ce serait donc dériver la fonction du signal TSOCOLP. Toutefois, nous n'en sommes pas sûr, mais il semble que le manuel souhaite que l'on mette TSOCOLP (respectivement TSMCOLP) à HIGH en cas d'avortement de la transmission. Il n'est pas clair non plus, si le signal doit être maintenu à HIGH pendant la durée d'envoi de la séquence de 0 et 1 alternés, ou s'il doit être pulsé car il est écrit : "TSMCOLP is valid when the TDONEP pin pulses".

III - Collision et Backoff

Enfin, nous voulions aller un peu plus loin et nous nous sommes donc penchés sur la détection des collisions multiples et la mise en place d'un backoff. Nous avons donc implémenté ces fonctionnalités.

A. Description du processus de Backoff et de la détection de collisions



Collisions Multiples avec Backoff

Nous allons voir comment nous détectons les collisions. Mais avant de commencer, nous tenons à préciser que le signal backoff_t qui correspond au décompte des clocks lors de l'attente du temps de backoff est en fait un signal interne, il n'est présent dans le chronogramme que pour montrer qu'il y a bien un décompte.

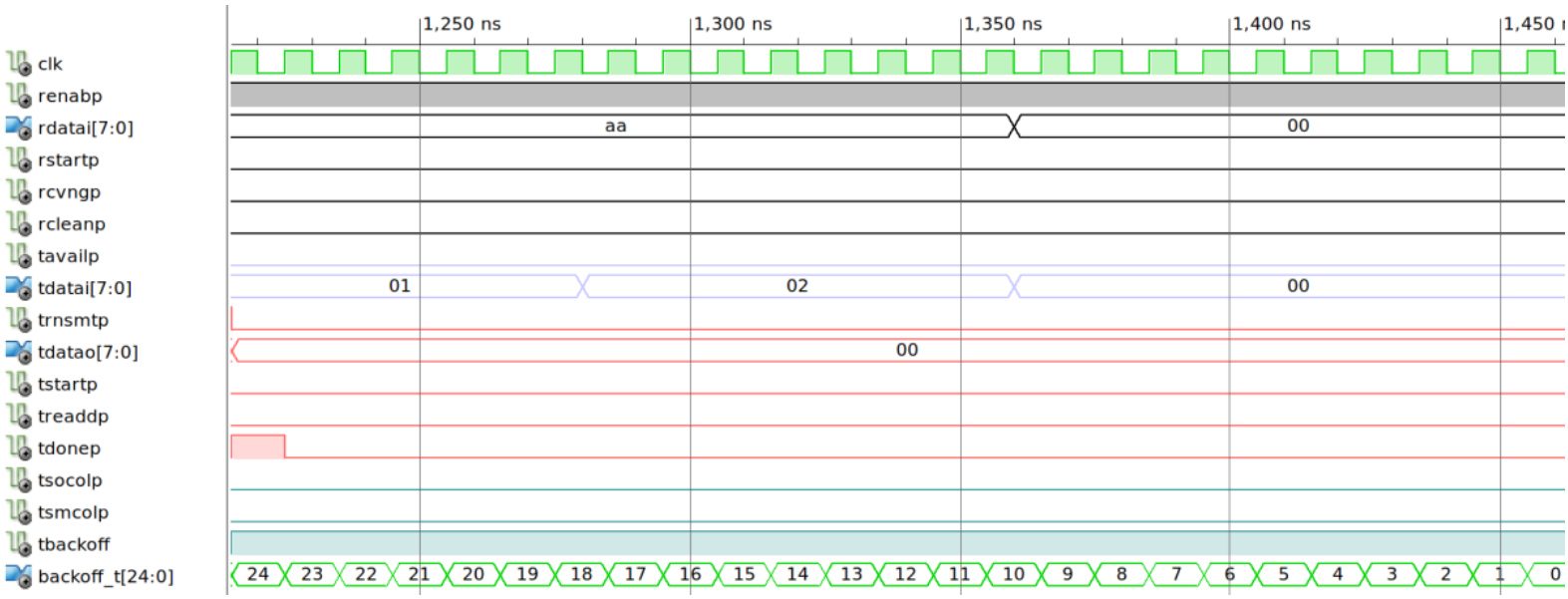
Que se passe-t-il dans la figure précédente, montrant deux collisions successives ?

Tout d'abord, on constate que jusqu'à l'émission de l'octet "BB", il s'agit d'une émission normale. Seulement, la carte est positionnée en réception et reçoit une trame. On a donc positionné RCVNGP à l'état HIGH. Comme RCVNGP et TRNSMTP sont tous deux à l'état HIGH, on en déduit qu'il y a une collision. En effet, on est en train de recevoir une trame et d'en émettre une autre en même temps. Il s'agit de la première collision, on positionne donc TSOCOLP à l'état HIGH. Après l'émission des 32 bits alternatifs, on pulse TDONEP. On force ensuite l'attente d'un temps aléatoire, nous traiterons ce sujet plus en profondeur plus loin.

Après ce temps d'attente, nous commençons une nouvelle émission, mais manque de chance, nous recevons une nouvelle trame. Il s'agit ici de la seconde collision, on positionne donc TSMCOLP à l'état HIGH, puisque nous avons subi au moins deux collisions consécutives. Après avoir envoyé la séquence de bits alternés, on ré-attend un nouveau temps de backoff.

MABILLE Gabriel
VENDOME Jean-Christophe

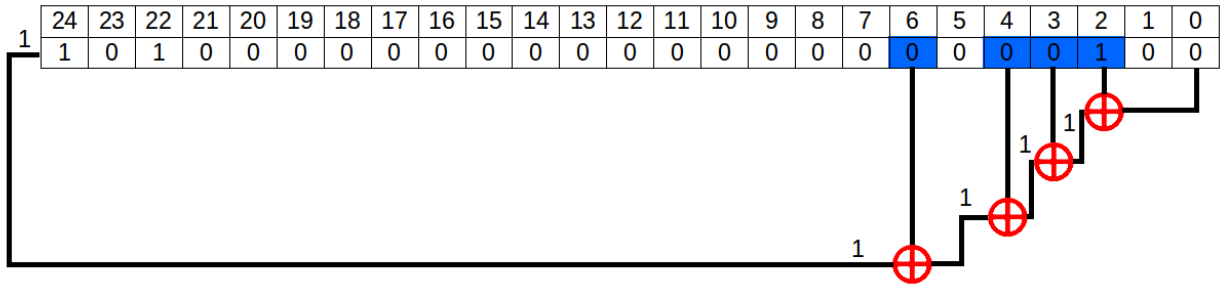
Nous allons maintenant pouvoir parler plus précisément de notre algorithme de backoff.
Tout d'abord quel est le principe ?
Comme on a pu le voir dans la figure précédente, lorsqu'une collision est détectée, il est vivement conseillé d'attendre avant de réémettre. Il faut attendre que l'hôte distant constate lui aussi la collision, qu'il envoie lui aussi la séquence de 32 bits alternés et que le canal d'émission soit libre. Toutefois, si les hôtes réémettent au bout du même temps fixe, ils se retrouveront dans la même configuration et l'on va ainsi enchaîner indéfiniment les collisions. Il faut donc attendre un temps aléatoire. Ce temps d'attente aléatoire correspond au temps de backoff. Ainsi, comme on peut le voir dans la figure suivante, une carte Ethernet tire un temps pseudo aléatoire (ici on voit que le backoff_t est de 24 clocks) et décompte les coups d'horloge. Tant que le temps de backoff n'est pas annulé, l'hôte ne peut pas émettre.



Zoom sur le décompte

Maintenant, il est légitime de se demander : Comment peut on générer un nombre pseudo aléatoire matériellement ?
Un dispositif peu onéreux permettant de réaliser cette opération est le registre à décalage à rétroaction linéaire (LFSR : Linear feedback shift register). Pour décrire de manière très superficielle son fonctionnement, disons qu'il s'agit de faire un décalage circulaire droit du registre mais que le bit sortant subit une série de ou exclusif avec d'autres bits du registre avant d'être réinjecté en début de registre.

Voici un schéma de notre LFSR :



LFSR que nous avons décrit en VHDL

Grâce à cette figure on comprend bien le fonctionnement du LFSR, le premier chiffre est 0b101000...100 soit 20971524, le second est 0b110100...010 soit 27262978, le troisième est 0b011010...001 soit 13631489, le quatrième est 0b101101...000 soit 23592960, etc...
Nous avons initialisé le registre de cette manière pour voir s'il fonctionnait bien pendant la période de débogage, on voit bien que pendant encore quelques coups d'horloge, on va juste diviser par deux mais ça redevient intéressant par la suite. On passera par exemple de 2880 à 16778656, 8389328, 4194664, 2097332, 1048666, 17301549, etc...

Après avoir transmis les 32 bits alternatifs, on retient dans backoff_t la valeur du LFSR. Pour que nos simulations ne s'éternisent pas, nous n'avons pas pris tous les 25 bits du registre, comme indiqué dans le manuel, mais seulement les 5 derniers bits. De cette manière, on a pu majorer le temps d'attente à 31s.

B. Problèmes rencontrés

Le plus gros problème que nous ayons rencontré a été dû à une erreur VHDL. Nous avions codé une boucle for allant de 24 à 1 en utilisant le mot clé TO au lieu du mots clés DOWNTO. Cette erreur nous a fait perdre une demi-heure de TP, mais nous ne sommes pas prêts de la reproduire.
L'autre soucis que nous avons eu a été de comprendre quand mettre TSMCOLP à l'état HIGH. Il nous a fallu du temps pour comprendre qu'il remplaçait TSOCOLP à partir de deux collisions consécutives.

MABILLE Gabriel
VENDOME Jean-Christophe

Conclusion

Sans nous proclamer professionnels du hardware, nous sommes assez fier de ce que nous avons réussi à réaliser. Il nous a fallu comprendre une partie du manuel technique d'Ethernet-10 Core, prendre en main le logiciel ISE-Webpack de XILINX, coder en VHDL, réaliser des simulations pertinentes. Ce n'était pas facile mais nous avons su relever le défi et en avons retiré une bonne expérience. Il nous manquait des connaissances matérielles, que ce BE nous a permis d'acquérir (nous réalisons, bien entendu, qu'il nous reste beaucoup à apprendre).

Le seul petit regret que nous avons est celui de n'avoir pas été assez rapides pour tester notre description matérielle en réel sur la carte FPGA.

Annexe

- Le schematic de notre carte Ethernet
- Le code VHDL de notre carte Ethernet