|  |  |
| --- | --- |
|  |  |
| Compte rendu  Contrôleur VGA |  |
|  |  |
|  | 16/12/2025Traitement/Interfaces systèmes embarquésAdrien BONNAND Polytech Dijon - Ex Esirem |

Table des matières

[Fonctionnement chaîne vidéo 4](#_Toc216465692)

[Simulation (testbench) 6](#_Toc216465693)

[Débuggeur Hardware (ILA) 9](#_Toc216465694)

[Interfaçage AXI Stream 13](#_Toc216465695)

[Traitement flux video AXIS 15](#_Toc216465696)

[Conclusion générale 27](#_Toc216465697)

Table des figures

[Figure 1 : Architecture chaîne vidéo 4](#_Toc216465731)

[Figure 2 : Réglages IP Video Timing Controller 5](#_Toc216465732)

[Figure 3 : Fonctionnement de la chaîne vidéo 5](#_Toc216465733)

[Figure 4 : Simulation – écriture d’une image complète 6](#_Toc216465734)

[Figure 5 : Simulation – écriture du début de l'image 6](#_Toc216465735)

[Figure 6 : Simulation – écriture du début de l'image (Zoom) 7](#_Toc216465736)

[Figure 7 : Simulation – écriture d'un endroit de l'image 7](#_Toc216465737)

[Figure 8 : Simulation – écriture d'un endroit de l'image (Zoom) 7](#_Toc216465738)

[Figure 9 : Simulation – écriture de la fin de l'image 8](#_Toc216465739)

[Figure 10 : ILA - écriture du début de l'image 9](#_Toc216465740)

[Figure 11 : ILA - écriture d'un endroit de l'image 10](#_Toc216465741)

[Figure 12 : ILA - écriture d'un endroit de l'image (Zoom) 10](#_Toc216465742)

[Figure 13 : ILA - fin de l'écriture d'une image 11](#_Toc216465743)

[Figure 14 : ILA – écriture de la deuxième ligne de l’image 11](#_Toc216465744)

[Figure 15 : ILA – écriture de la quinzième ligne de l’image 12](#_Toc216465745)

[Figure 16 : Architecture interfaçage AXIS 13](#_Toc216465746)

[Figure 17 : Simulation - écriture image en AXIS 13](#_Toc216465747)

[Figure 18 : Simulation - écriture image en AXIS (Zoom) 13](#_Toc216465748)

[Figure 19 : ILA - écriture image en AXIS 14](#_Toc216465749)

[Figure 20 : Architecture IP inversion VHDL 15](#_Toc216465750)

[Figure 21 : Architecture IP inversion VHDL (data sur 32bits) 15](#_Toc216465751)

[Figure 22 : Architecture IP inversion VHDL (signaux axis connectés un à un) 16](#_Toc216465752)

[Figure 23 : IP inversion VHDL – affichage sur écran, pixels non inversés 17](#_Toc216465753)

[Figure 24 : IP inversion VHDL - affichage sur écran, pixels inversés 17](#_Toc216465754)

[Figure 25 : IP inversion VHDL - ILA 18](#_Toc216465755)

[Figure 26 : IP inversion VHDL - ILA (Zoom) 18](#_Toc216465756)

[Figure 27 : Architecture IP inversion (HLS) 18](#_Toc216465757)

[Figure 28 : IP inversion HLS – ILA 19](#_Toc216465758)

[Figure 29 : IP HLS numéro binôme 20](#_Toc216465759)

[Figure 30 : IP HLS binôme number - ILA 21](#_Toc216465760)

[Figure 31 : IP HLS numéro binôme - affichage sur écran du numéro du binôme 21](#_Toc216465761)

[Figure 32 : IP HLS numéro binôme - affichage sur écran du numéro du binôme (Zoom) 21](#_Toc216465762)

[Figure 33 : Architecture avec les trois IPs 22](#_Toc216465763)

[Figure 34 : Architecture avec les trois IPs (Zoom) 22](#_Toc216465764)

[Figure 35 : Regroupement des trois IPs - affichage sur écran 24](#_Toc216465765)

[Figure 36 : Architecture avec les trois IPs (Plus lisible) 24](#_Toc216465766)

[Figure 37 : Regroupement des trois IPs – ILA (IP inversion VHDL selectionnée) 25](#_Toc216465767)

[Figure 38 : Regroupement des trois IPs – ILA (IP HLS numéro binôme selectionnée) 25](#_Toc216465768)

# Fonctionnement chaîne vidéo

## Architecture

Une image contenant texte, diagramme, ligne, Police

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture chaîne vidéo

Premièrement, une PLL fournit une horloge de 25,2 Mhz. Cette horloge est transmise à la fois à une IP Xilinx permettant de gérer les timings de synchronisation pour l’affichage de données sur un écran. Mais aussi, elle est connectée à un compteur binaire qui permet de changer la valeur des pixels à afficher. Celui-ci possède une entrée *CE* qui autorise l’incrémentation du compteur, uniquement lors de l’écriture d’une ligne de l’image (hors périodes de blank), ainsi qu’un signal *SCLR*, actif quand *Hsync* = 1, afin de réinitialiser le compteur en début de ligne.

La sortie de ce compteur est connectée à un module développé personnellement en VHDL, afin de ne garder que les MSB du compteur. Cela permet le changement de couleur des pixels toutes les 16 incrémentations, et donc l’écriture de bandes verticales de pixels. Le signal *sel\_ActVideo* met la valeur de sortie à 0 lorsqu’il y a une écriture dans la partie blank.

Enfin, un debugger hardware (ILA) est utilisé pour enregistrer l’état des signaux importants lors du test sur cible, en s’appuyant sur des BRAMs pour le stockage.

Le fait de générer un motif vidéo (test pattern) est pertinent car cela permet de valider rapidement et simplement le bon fonctionnement d’un système d’affichage. Un motif correct valide la conformité temporelle des signaux de synchronisation (hsync/vsync), du signal active\_video ainsi que de la fréquence de changement des pixels.

Sur certains écrans, des bandes noires peuvent apparaître à cause d’une désynchronisation entre les pixels et les signaux de synchronisation (*hsync/vsync*). Ce décalage est souvent dû au traitement interne vieillissant de l’écran, qui n’aligne plus parfaitement les signaux entrants.

Une solution consiste à ajouter un léger délai soit sur les pixels, soit sur les signaux *hsync* et *vsync*, afin de réaligner correctement les informations vidéo. Cela peut être réalisé à l’aide de bascules, d’IP Xilinx comme *IOBDELAY* ou *DCM/PLL* ou tout simplement via les réglages de l’IP *Video Timing Controller* (en décalant le *Sync Start* par exemple).

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Figure : Réglages IP Video Timing Controller

Le fait de mettre les pixels à zéro quand ActVideo est égale à zéro permet d’éviter l’écriture de pixels parasites dans les zones de blanking. Cela garantit que seules les parties visibles de l’image reçoivent des valeurs de pixels correctes, stabilise l’affichage et évite l’apparition de pixels indésirables.

## En fonctionnement

Une image contenant texte, Appareils électroniques, écran, Périphérique de sortie

Le contenu généré par l’IA peut être incorrect.

Figure : Fonctionnement de la chaîne vidéo

# Simulation (testbench)

## Ecriture de l’entièreté d’une image



Figure : Simulation – écriture d’une image complète

Cette simulation, observée d’un point de vue global, montre simplement l’écriture complète d’une image. En effet, les valeurs des pixels évoluent durant toute la phase d’écriture de l’image. Le signal *Hsync* passe régulièrement à 1 pour indiquer l’écriture de chaque ligne, tandis qu’un pulse de *Vsync* en fin de trame signale la fin de l’écriture de l’image.

À la fin du testbench, on observe un « trou » dans le signal *active\_video\_out*, accompagné de pixels à 0 : cela correspond à la période de « blanking », durant laquelle aucune donnée vidéo n’est transmise.

## Début de l’écriture de l’image

Tout d’abord, il est intéressant de voir le début de l’écriture de l’image.



Figure : Simulation – écriture du début de l'image

Dès que le signal *reset* passe à 1 (état inactif), l’écriture de l’image commence. Les pulses de *Hsync* indiquent l’écriture de chaque ligne. De plus, les passages du signal *video\_active\_out* à 1 montrent que seuls les pixels utiles de l’image sont écrits, excluant ainsi la zone de blanking.

Agrandissement de la fenêtre :

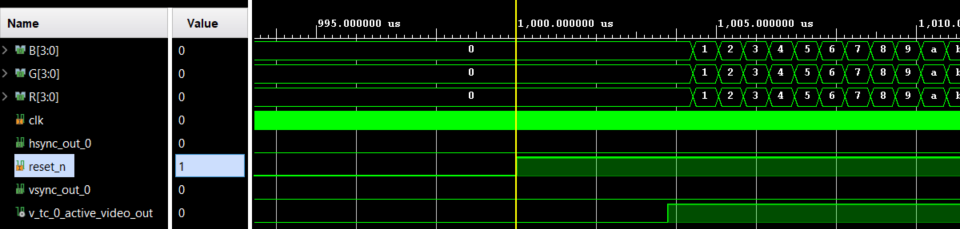


Figure : Simulation – écriture du début de l'image (Zoom)

Le compteur s’incrémente correctement, de 1 en 1, durant l’écriture de la première ligne.

## Capture pendant l’écriture de l’image

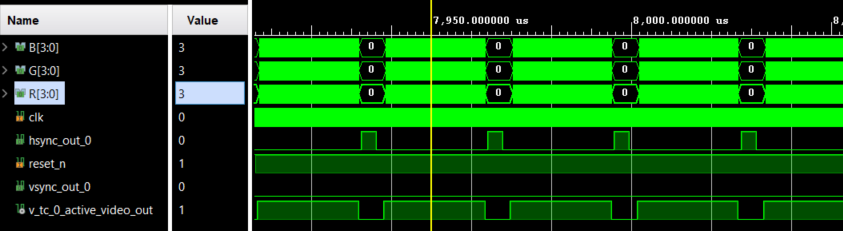


Figure : Simulation – écriture d'un endroit de l'image

Sur cette capture, il est possible de voir plusieurs écritures de lignes, grâce à chaque *Hsync* à 1.

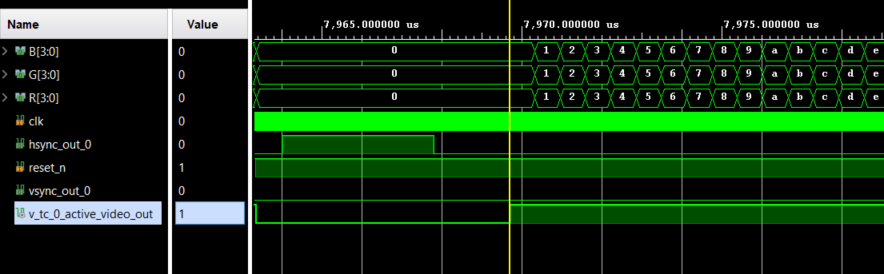


Figure : Simulation – écriture d'un endroit de l'image (Zoom)

L’agrandissement de la fenêtre permet de voir l’incrémentation des pixels. Le retour du compteur à 0 indique que l’on se trouve au début d’une nouvelle ligne d’écriture et montre qu’il est correctement réinitialisé à chaque ligne.

## Capture de la fin de l’écriture de l’image

Une image contenant capture d’écran, Logiciel multimédia, logiciel, ligne

Le contenu généré par l’IA peut être incorrect.

Figure : Simulation – écriture de la fin de l'image

Le compteur reste à 0 pendant plus de 500 µs, ce qui montre que l’écriture de l’image est terminée. Quelques pulses de *Hsync* continuent à apparaître pour générer les lignes vides de la période de « blanking », jusqu’au passage de *Vsync* à 1, signalant la fin de l’image. De plus, *video\_active\_out* reste à 0 pendant cette même durée, confirmant la présence de la phase de « blanking ».

# Débuggeur Hardware (ILA)

## Début de l’écriture de l’image

Tout d’abord, il est intéressant de voir le début de l’écriture de l’image.

### Trigger utilisé :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Cette machine à état attend un *Vsync* (fin de l’image) puis un *active\_video* à 1 (début de la nouvelle image) et enfin quelques coup d’horloge avant de lancer le trigger.

### Capture des signaux internes :

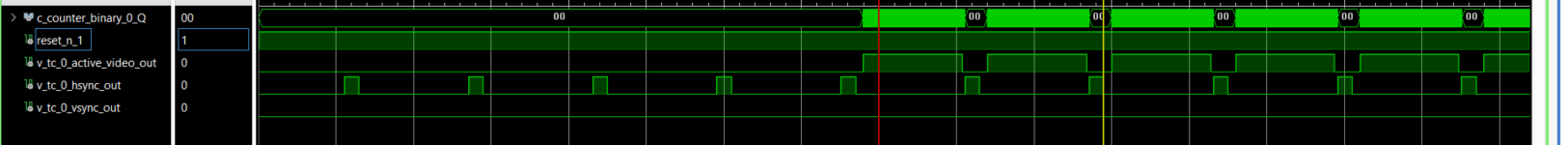


Figure : ILA - écriture du début de l'image

Sur cette capture, les pixels sont correctement à 0 pendant la période de « blanking » et reprennent leur incrémentation dès le début de l’écriture de la nouvelle image.

## Capture pendant l’écriture de l’image

### Trigger utilisé :

Front montant du signal *active\_video\_out*

### Capture des signaux internes :

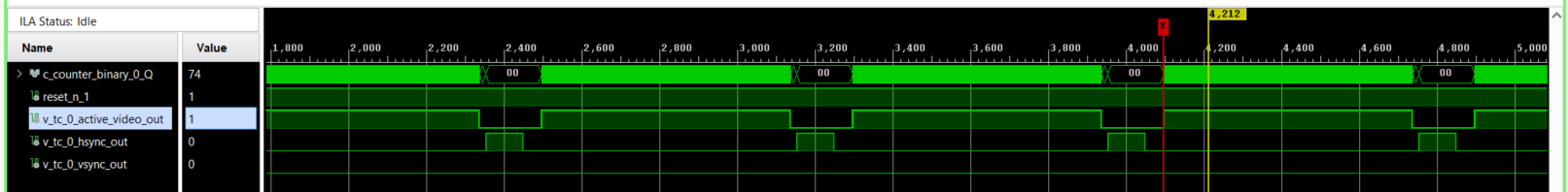


Figure : ILA - écriture d'un endroit de l'image

Sur cette capture, il est possible de voir plusieurs écritures de lignes, grâce à chaque *Hsync* à 1 et la mise à l’état bas du signal *active\_video\_out*.

Une image contenant capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure : ILA - écriture d'un endroit de l'image (Zoom)

L’agrandissement de la fenêtre permet d’observer l’incrémentation des pixels, qui recommence à 0 à chaque nouvelle ligne, confirmant que le compteur est correctement réinitialisé.

## Capture de la fin de l’écriture de l’image

### Trigger utilisé :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Cette machine à état attend un *Vsync* (fin de l’image) puis un *Hsync* et enfin quelques coup d’horloge avant de lancer le trigger

### Capture des signaux internes :

Une image contenant logiciel, Logiciel multimédia, texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure : ILA - fin de l'écriture d'une image

Le compteur reste à 0, indiquant que l’écriture de l’image est terminée. Quelques pulses de Hsync apparaissent encore pour générer les lignes vides de la zone de « blanking », jusqu’au passage de Vsync à 1, signalant la fin de l’image.

## Capture de l’écriture d’une certaine ligne de l’image

### Trigger utilisé :

Une image contenant texte, capture d’écran, menu, document

Le contenu généré par l’IA peut être incorrect.

Cette machine à état attend un *Vsync* (fin de l’image) puis compte un nombre de front montant de *video\_active\_out* (4) qui représente, à chaque fois, l’écriture d’une nouvelle ligne.

### Capture des signaux internes :

Capture des signaux quand le compteur est égal à 2 :

Une image contenant logiciel, Logiciel multimédia, ligne, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Figure : ILA – écriture de la deuxième ligne de l’image

Capture des signaux quand le compteur est égal à 15 :

Une image contenant capture d’écran, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Figure : ILA – écriture de la quinzième ligne de l’image

## Conclusion simulation et debugger Hardware

D’après l’ensemble des captures précédentes, il n’y a pas de régression entre la simulation, qui est parfaite, et le test sur cible, où des phénomènes extérieurs peuvent intervenir. Cependant, la simulation reste plus facile à analyser, car elle ne nécessite pas la mise en place de triggers et permet d’observer l’état des signaux sur toute la durée souhaitée. En comparaison, l’ILA est limitée par la taille des BRAM, ce qui restreint la capture dans le temps.

# Interfaçage AXI Stream

## Architecture avec l’interfaçage AXI STREAM

Une image contenant diagramme, texte, ligne, Plan

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture interfaçage AXIS

L’IP « AXI4-Stream to Video Out » doit être utilisée en mode slave, car elle reçoit les signaux de synchronisation provenant du module « Video Timing Controller ». C’est ensuite elle qui génère le flux vidéo en respectant les timings fournis par ce contrôleur.

## Simulation

Une image contenant capture d’écran, logiciel, Logiciel multimédia, Caractère coloré

Le contenu généré par l’IA peut être incorrect.

Figure : Simulation - écriture image en AXIS

Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Figure : Simulation - écriture image en AXIS (Zoom)

L’IP « AXI4-Stream to Video Out » met un certain temps à générer le flux vidéo car elle doit « accrocher » le signal AXI stream entrant. Cela met environ 150ms .

## Debuggeur Hardware

Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Figure : ILA - écriture image en AXIS

Sur ce chronogramme, il est possible de voir la conversion du flux video en flux AXIS. Le signal *tready* est correctement à l’état haut tout le temps, le *tvalid* indique les nouvelles transactions et le *Tdata* montre la valeur du pixel à un moment donné.

# Traitement flux video AXIS

## IP inversion VHDL

### Architecture

Une image contenant diagramme, Plan, ligne, texte

Le contenu généré par l’IA peut être incorrect.

Figure 20 : Architecture IP inversion VHDL

Architecture avec l’IP inversion en 32 bits :

Une image contenant texte, diagramme, Plan, Dessin technique

Le contenu généré par l’IA peut être incorrect.

Figure 21 : Architecture IP inversion VHDL (data sur 32bits)

Architecture avec l’IP en 8 bits mais signaux AXIS connectés un par un :

Une image contenant diagramme, Plan, Dessin technique, schématique

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture IP inversion VHDL (signaux axis connectés un à un)

Lors de l’implantation de l’IP d’inversion (première architecture), aucun pixel ne s’affichait à l’écran. J’ai alors essayé de passer l’IP en 32 bits et de supprimer l’ILA, ce qui a résolu le problème. Cependant, cela me semblait étrange que la taille des données ou l’ILA soient à l’origine du dysfonctionnement. Je suis donc revenu à l’IP en 8 bits, mais cette fois en connectant les signaux AXIS un par un. Après ce correctif, tout a fonctionné correctement et j’ai pu afficher l’inversion sur l’écran.

### Code de l’IP inversion VHDL

Une image contenant texte, capture d’écran, Police, document

Le contenu généré par l’IA peut être incorrect.

Le premier process, synchronisé sur l’horloge *s00\_axis\_aclk*, initialise le signal *tready* de l’interface esclave. Si le reset (*s00\_axis\_aresetn*) est actif, *tready* est mis à 0. Sinon, il prend la valeur de *m00\_axis\_tready*, ce qui permet de signaler que le module maître est prêt à recevoir des données.

Ensuite, la ligne suivante réalise l’inversion des données entrantes. Si le signal *VidOrig\_nVideoInv* vaut 1, les données tdata sont inversées bit à bit, sinon elles sont transmises telles quelles.

Le second process, synchronisé sur l’horloge *m00\_axis\_aclk*, gère l’interface maître. Lorsque le reset (*m00\_axis\_aresetn*) est actif, les signaux *tvalid, tlast* et *tdata* sont initialisés à 0. Sinon, *tlast* est copié depuis l’esclave et *tdata* reçoit les données qui peuvent être inversées. Le signal *tvalid* est activé uniquement si le maître est prêt (*m00\_axis\_tready* = 1) et que les données de l’esclave sont valides (*s00\_axis\_tvalid* = 1), sinon il reste à 0.

### En fonctionnement

Le deuxième interrupteur, SW1, est à l’état bas (Pas d’inversion) :

Une image contenant texte, intérieur, ordinateur, mur

Le contenu généré par l’IA peut être incorrect.

Figure : IP inversion VHDL – affichage sur écran, pixels non inversés

Le deuxième interrupteur, SW1, est à l’état haut (inversion) :

Une image contenant texte, intérieur, ordinateur, ordinateur portable

Le contenu généré par l’IA peut être incorrect.

Figure : IP inversion VHDL - affichage sur écran, pixels inversés

### Debuggeur Hardware

Une image contenant capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Figure : IP inversion VHDL - ILA

Une image contenant capture d’écran, logiciel, Logiciel multimédia, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Figure : IP inversion VHDL - ILA (Zoom)

D’après le chronogramme ci-dessus, on observe qu’une transaction AXIS entrante sur S00 est immédiatement suivie, au cycle d’horloge suivant, par une transaction sortante sur M00 depuis l’IP d’inversion, avec les données inversées. Les signaux *tready* du maître et de l’esclave restent constamment à 1, ce qui confirme la disponibilité continue du bloc. De plus, les signaux *tvalid* et *tlast* du maître sont correctement décalés d’un cycle d’horloge par rapport à ceux de l’esclave, assurant une transmission synchronisée des données.

## IP inversion HLS

### Architecture

Une image contenant diagramme, texte, Plan, ligne

Le contenu généré par l’IA peut être incorrect.

Figure 27 : Architecture IP inversion (HLS)

### Code de l’IP inversion

### Une image contenant texte, capture d’écran, Police, document Le contenu généré par l’IA peut être incorrect.

### Debuggeur Hardware

Le deuxième interrupteur, SW1, est à l’état haut (inversion) :

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Figure : IP inversion HLS – ILA

Contrairement à l’IP inversion VHDL, celle-ci met deux cycles d’horloge pour faire l’inversion.

## IP HLS numéro binôme

### Architecture

Une image contenant diagramme, Plan, ligne, Dessin technique

Le contenu généré par l’IA peut être incorrect.

Figure : IP HLS numéro binôme

### Code de l’IP inversion

Une image contenant texte, capture d’écran, Police, document

Le contenu généré par l’IA peut être incorrect.

### Debuggeur Hardware

Le deuxième interrupteur, SW1, est à l’état bas (Pas d’affichage du numéro du binôme) :

Une image contenant capture d’écran, Logiciel multimédia, logiciel, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Figure 30 : IP HLS binôme number - ILA

Comme l’IP inversion HLS, celle-ci met deux cycle d’horloge pour faire le traitement. L’ILA a conservé l’ancien nom « axi\_inverter », mais c’est bien mon IP Binôme Number qui est connectée.

### En fonctionnement

Le deuxième interrupteur, SW1, est à l’état haut (Affichage du numéro du binôme) :

Une image contenant texte, intérieur, ordinateur, mur

Le contenu généré par l’IA peut être incorrect.

Figure : IP HLS numéro binôme - affichage sur écran du numéro du binôme

Une image contenant mur, intérieur, Appareil de présentation, texte

Le contenu généré par l’IA peut être incorrect.

Figure : IP HLS numéro binôme - affichage sur écran du numéro du binôme (Zoom)

## Regroupement des trois IPs

### Architecture (1ère version)

Une image contenant diagramme, Plan, croquis, Dessin technique

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture avec les trois IPs

Une image contenant texte, diagramme, Plan, Parallèle

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture avec les trois IPs (Zoom)

Pour relier les modules entre eux, le flux vidéo AXIS est connecté aux trois modules de traitement. Leurs signaux tready sont combinés à l’aide d’un opérateur AND, ce qui permet d’autoriser l’écriture AXIS uniquement lorsque les trois modules sont prêts.

La sortie AXIS vidéo est choisie à l’aide d’un signal sel, qui est contrôlé par deux switchs permettant de sélectionner le flux vidéo souhaité.

### Code du multiplexeur

Une image contenant texte, capture d’écran, menu, document

Le contenu généré par l’IA peut être incorrect.

Les modules de traitement deviennent prêts lorsque l’esclave en sortie du multiplexeur est prêt.

### En fonctionnement

L’ensemble d’interrupteurs est configuré avec SW1 à l’état haut et SW2–SW3 positionnés sur 10 (soit la valeur 2) :

Une image contenant texte, intérieur, mur, Appareil électronique

Le contenu généré par l’IA peut être incorrect.

Figure : Regroupement des trois IPs - affichage sur écran

L’ensemble des fonctionnalités fonctionne correctement. Il est possible de sélectionner chacun des modules de traitement grâce aux switchs SW2 et SW3, tandis que le switch SW1 permet d’activer ou non le traitement choisi.

### Architecture (2ème version)

Ci-dessous, la version plus propre :

Une image contenant diagramme, Plan, Dessin technique, schématique

Le contenu généré par l’IA peut être incorrect.

Figure : Architecture avec les trois IPs (Plus lisible)

Dans cette architecture, j’ai ajouté le module « AXIS Broadcaster » afin de connecter facilement le flux vidéo AXIS aux trois modules de traitement. Le multiplexage des sorties est toujours géré par l’IP décrite précédemment.

### Debuggeur Hardware

Les switchs **SW2** et **SW3** sont réglés sur 0 :

Une image contenant texte, capture d’écran, affichage, logiciel

Le contenu généré par l’IA peut être incorrect.

Figure : Regroupement des trois IPs – ILA (IP inversion VHDL selectionnée)

Sur ce chronogramme, le flux AXIS sort d’abord de l’IP « Video\_In\_to\_AXIS » (slot 4), puis entre dans l’IP d’inversion en VHDL (slot 2), où il subit un décalage d’un cycle d’horloge dû au temps de traitement. Le multiplexeur AXIS montre que c’est le flux provenant du slot 2 qui est sélectionné en sortie de l’interface maître, et qui est ensuite transmis à l’IP « AXIS\_to\_Video\_Out ».

Les switchs **SW2** et **SW3** sont réglés sur 10 :

Une image contenant capture d’écran, texte, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Figure : Regroupement des trois IPs – ILA (IP HLS numéro binôme selectionnée)

Lorsque la combinaison des switches **SW2** et **SW3** vaut 10 (soit la valeur 2), l’IP Binôme Number est sélectionnée par le multiplexeur. Dans ce cas, le signal AXIS est décalé de deux cycles d’horloge avant d’entrer dans l’IP « AXIS\_to\_Video\_Out ».

### En fonctionnement

Cependant, cette version n’est pas fonctionnelle pour une raison encore inconnue. D’après l’ILA, les flux **AXIS** semblent corrects : les transactions sont régulières, tous les signaux *tready* sont à 1, et les *tlast* s’activent correctement à la fin de chaque image.

Le problème pourrait provenir de la gestion des ports **AXIS** dans Vivado. En effet, lorsque je connecte les signaux **AXIS** un par un, tout fonctionne correctement. En revanche, lorsque je relie directement l’ensemble d’un port **AXIS** à un autre, cela ne fonctionne plus.

# Conclusion générale

Ce TP nous a permis de renforcer nos compétences en VHDL sur FPGA, en travaillant sur la génération et la gestion de flux vidéo. Nous avons utilisé une PLL pour fournir l’horloge, un compteur pour incrémenter les pixels et un module VHDL pour créer des bandes verticales, tout en gérant correctement les périodes de blank.

L’utilisation de l’ILA a permis de vérifier le comportement des signaux sur cible, et la génération d’un motif vidéo a confirmé le bon fonctionnement du système et la conformité des signaux *Hsync*, *Vsync* et *active\_video*. Nous avons également appris à développer plusieurs modules avec une liaison AXIS que ce soit en VHDL ou à l’aide de Vitis HLS.