Polytech Dijon

FISA IOT 5A

VHDL - TP n°1

FPGA - Chaîne video

Auteur :
CAMUS Pierre-Marie

Tuteur :
DUBOIS Julien

2024-2025

Sommaire

Table des Figures

1 Compte Rendu

1.1 But

Le projet vise à mettre en œuvre une chaîne vidéo complète et à effectuer un traitement sur le flux vidéo. Le traitement serait décrit à un certain stade à l'aide d'outils de prototypage de synthèse de haut niveau. L'objectif principal est de configurer une chaîne vidéo de base et d'appliquer un traitement au flux vidéo. Le traitement sera d'abord effectué manuellement, puis à l'aide de la synthèse de haut niveau. Tout d'abord, un flux vidéo est configuré à partir d'un compteur binaire, un motif de niveaux de gris est généré, puis affiché sur l'interface VGA. Différentes stratégies sont présentées dans cette partie du tutoriel afin d'obtenir la chaîne complète. Notez que pour tous les composants de la bibliothèque, vous pouvez consulter sa documentation en cliquant sur l'icône de documentation dans la fenêtre de configuration.

1.2 Méthodologie

1. Configurer une simple chaîne video.
2. Chaîne vidéo comprenant une interface AXI4-stream.
3. Traitement vidéo et mise en œuvre.

1.3 Schéma

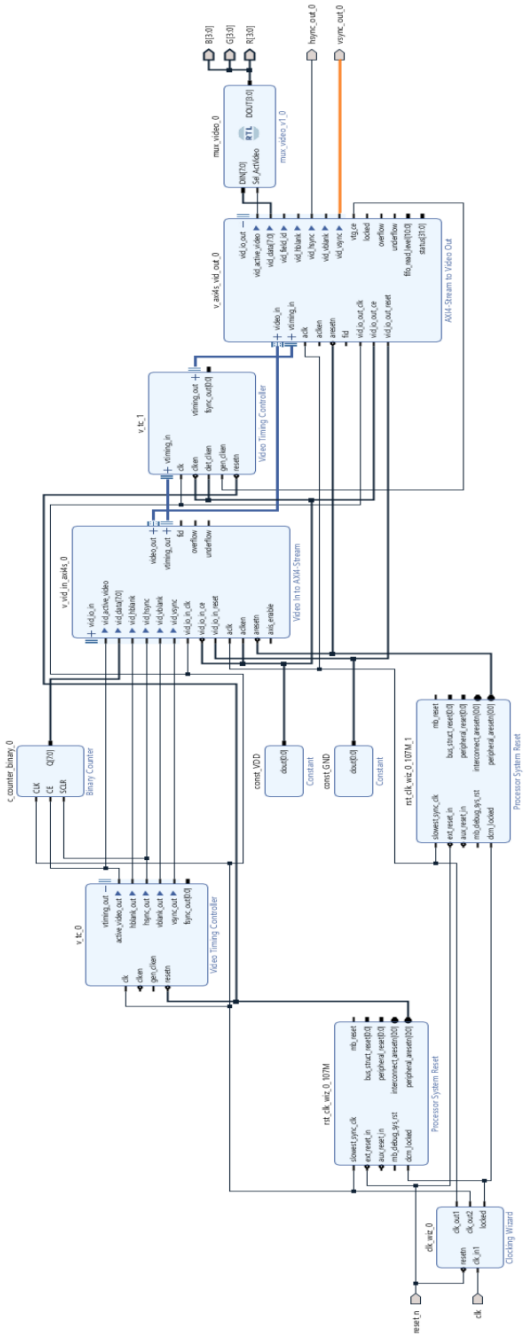

FIGURE 1.1 – Chaine video

1.4 Part 1 : Configuration d'une chaîne video

Dans un premier temps, un bloc Clock Wizard est configuré afin de générer deux horloges distinctes : une horloge à 108 MHz destinée au traitement du flux de pixels, et une horloge à 25,2 MHz utilisée pour la génération temporelle du signal vidéo. L'entrée de réinitialisation du bloc est activée en mode actif bas. Un compteur binaire 8 bits est ensuite ajouté avec validation par CE et remise à zéro synchrone (SCLR). Le compteur ainsi que le bloc *Processor System Reset* sont synchronisés par l'horloge à 25.2 MHz. Le signal *locked* du *Clock Wizard* est connecté à l'entrée *dcm_locked*. Les entrées externes sont renommées *clk* et *reset_n*, et le reset externe est connecté à l'entrée *ext_reset_in*.

Un *Video Timing Controller (VTC)* est ensuite intégré et configuré en 480p (640×480) sans interface *AXI* ni détection automatique. Son horloge est reliée à *clk_out2* et son signal de reset à *reset_n*. Les signaux de synchronisation *HSYNC* et *VSYNC* sont exportés vers les sorties *hsync_out_0* et *vsync_out_0*. Le signal *active_video_out* est utilisé pour piloter l'entrée *CE* du compteur, ce qui permet de faire évoluer le motif uniquement pendant les périodes vidéo actives.

Les 4 bits de poids fort (*MSB*) du compteur sont extraits à l'aide d'un bloc *Slice* et connectés aux trois sorties *R*, *G* et *B* (4 bits) afin de générer l'affichage couleur. Un motif périodique apparaît ainsi à l'écran.

FIGURE 1.2 – Niveau de gris

Après génération du *wrapper HDL*, le fichier de contraintes *Base_line_top.xdc* est ajouté et vérifié conformément au brochage VGA de la *ZedBoard*. La synthèse, le placement, le routage puis la génération du *bitstream* sont réalisés avant la configuration de la carte. À l'affichage, des bandes avec un niveau de gris different apparaissent. Le niveau de gris peut prendre 256 valeurs de 0 à 255. On utilise les 4 bits de points fort sur le mot de 8 bits a cause du *xlslice* ceux qui donne :

$$\text{Nb de bande} = \frac{256}{2^4} = 16$$

Lien github Step14.

Le bloc d'extraction des bits est ensuite modifié en *VHDL* afin que la sortie soit égale aux 4 *MSB* uniquement lorsque le signal *active_video* est à l'état haut, et nulle sinon. Ce bloc est ajouté au design via *Add Module to Block Design*. Cette modification permet d'éliminer les artefacts vidéo pour s'adapter au mieux à l'écran non présent sur le screenshot 1.2

Enfin, une phase de validation et de débogage matériel est menée. Une simulation avec *ILA* permet d'observer les signaux *HSYNC*, *VSYNC*, *blank* et *active_video*. Une vérification à l'oscilloscope confirme la cohérence temporelle des signaux. L'outil *Integrated Logic Analyzer (ILA)* est ensuite utilisé pour le débogage matériel, avec une machine d'état

de déclenchement basée successivement sur *VSYNC*, *HSYNC*, puis un retard de quelques cycles avant le déclenchement.

Lien github Step15.

1.5 Part 2 : Ajout d'une AXI4-stream

Tous les composants sont interconnectés conformément au schéma global présenté sur la figure 1.1.

Un **testbench** est proposé pour valider le fonctionnement de la chaîne. Il est important de noter que la simulation peut être très longue, et il est conseillé de l'exécuter en parallèle avec d'autres tâches.

Enfin, il convient de mesurer le temps nécessaire avant que le système soit **verrouillé (locked)**, afin de s'assurer de la stabilité du flux vidéo et de l'interface AXI4-Stream.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity bit_extraction is
    Port ( DIN : in STD_LOGIC_VECTOR (7 downto 0);
           Sel_ActVideo : in STD_LOGIC;
           DOUT : out STD_LOGIC_VECTOR (3 downto 0));
end bit_extraction;

architecture Behavioral of bit_extraction is

begin

select_entry: process(Sel_ActVideo, DIN)
    begin
        if (Sel_ActVideo = '0') then
            DOUT <= "0000";
        else
            DOUT <= DIN (7 downto 4);
        end if;

end process;


end Behavioral;
```

En mode SLAVE, le composant respecte le protocole AXI4-Stream en n'acceptant les données que lorsque le flux est disponible, garantissant ainsi une synchronisation correcte entre les données vidéo, les signaux de contrôle (HSYNC, VSYNC) et l'horloge vidéo. Ce mode est donc indispensable pour assurer un affichage stable et cohérent.

Lien github Part2

1.6 Part 3 : Mise en oeuvre

Après validation de l'IP, celui-ci est intégré dans la chaîne vidéo existante validée dans la Partie II. Le module est ajouté au design en sélectionnant *Add Module to Block Design* sur le fichier VHDL, puis les connexions sont effectuées et la nouvelle architecture générée et validée telle que l'on peut le voir figure 1.3 ci-dessous.

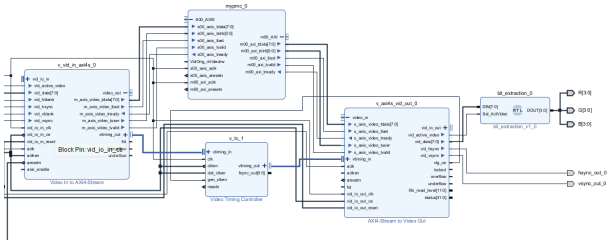

FIGURE 1.3 – Second timer controller

Deux modules supplémentaires sont générés en C/C++ via **Vivado HLS** pour être intégrés dans la chaîne vidéo finale :

1. Le premier module réalise l'inversion des niveaux de gris, identique à l'IP manuel décrit précédemment.
2. Le second module insère le numéro de *binôme* (fourni par l'enseignant) dans chaque trame vidéo.

Pour la validation, les modules sont interconnectés et des interrupteurs externes permettent de sélectionner en ligne le mode souhaité :

— Inversion manuelle des niveaux de gris.
— Inversion par module HLS.
— Incrustation du numéro de binôme par module HLS.

Cette configuration permet de tester et comparer facilement les différents modes de traitement en temps réel.

Lien github Part3