01/06/2023

Interface VGA

Plan de Validation (Solution finale)

Atmani Hicham & Kamal Kherchouch
FORMATION SAFRAN CHEZ AJC

Date	Version	Remarque	Auteur
05/06/2023	A0	Création du Document de plan de validation pour la solution intermédiaire	AHI & KKH
12/06/2023	A1	Ajout de l'analyse hardware	AHI & KKH
13/06/2023	A1	Ajout du module Pattern_VGA Retrait des résultats de simulation pour les placer dans le rapport de validation version A0.	AHI & KKH
15/06/2023	A2	Ajout des objectifs de validation (analyse, test et démonstration)	AHI & KKH
29/06/2023	A3	Mise à jour de la solution intermédiaire (v. A2) en y ajoutant le filtre Gaussien (solution finale)	AHI & KKH

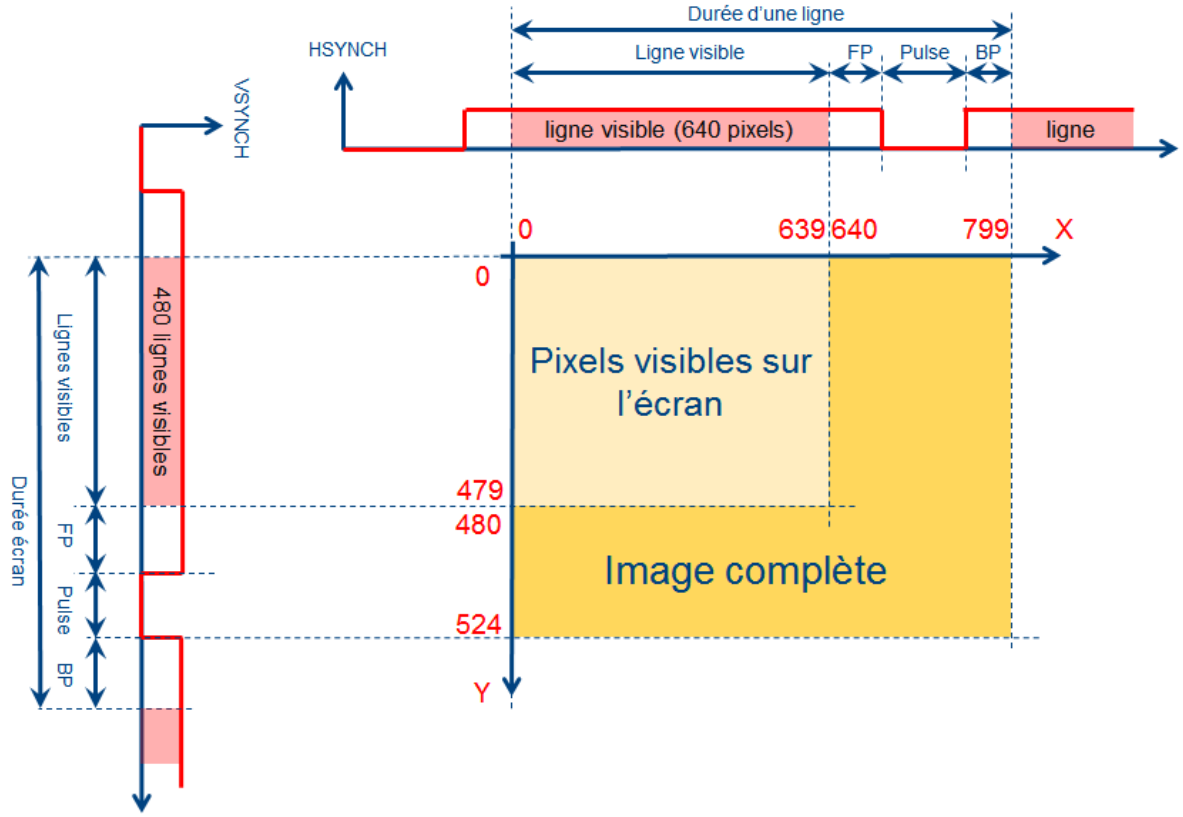
Table des matières

1. Présentation de la norme VGA et de la solution intermédiaire

1.1. Norme VGA

La norme VGA 640 x 480 pixels (avec 16 couleurs) à 60 Hz impose une horloge de 25.175 MHz par pixel avec un rafraichissement vertical de 31.46875 KHz.

La norme VGA est représentée via l'image ci-dessous :



On y retrouve notre partie visible de l'image (640 x 480 pixels) plus des pixels supplémentaires qui nous permettrons d'avoir une marge lors de la synchronisation de l'image.

Les pixels sont composés de la manière suivante (selon la norme VGA):

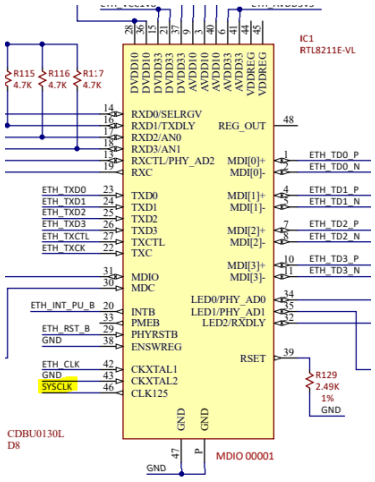
Nom	Horizontal	Vertical
Image visible	640	480
Front Porch	16	10
Back Porch	48	33
Largeur de synchronisation	96	2
Nb total de pixel	**800**	**525**

1.2. Analyse Hardware

1.2.1. Carte Cora Z7

L'analyse hardware de la carte Cora Z7 (XC7Z010-1CLG400C) va nous permettre de déterminer l'horloge d'entrée (pour notre système) que nous délivre cette dernière.

Sur le schématique de la carte Cora, nous pouvons voir que l'horloge système (SYSCLK) est fournie par le composant IC1 via la pin 46.



En cherchant sur la datasheet du composant, on peut voir que la pin 46 (version à 48 pins) délivre une horloge cadencée à 125MHz.

6.2. Clock

Table 2. Clock

Pin No. (48-pin)	Pin No. (64-pin)	Pin Name	Type	Description
42	61	CKXTAL1	I	25/50MHz Crystal Input. If a 25/50MHz oscillator is used, connect CKXTAL1 to the oscillator's output (see section 10.3, page 57 for clock source specifications).
43	62	CKXTAL2	O	25/50MHz Crystal Output. Connect to GND if an external 25/50MHz oscillator drives CKXTAL1.
46	1	CLK125	O/PD	125MHz Reference Clock Generated from Internal PLL. This pin should be kept floating if the 125MHz clock is not used by MAC.

Note: For those who conduct crystal ppm measurement for the models transited from RTL8211D to RTL8211E, note that the design of external circuit shall be modified, i.e. the output resistor should be connected to CKXTAL1 rather than CKXTAL2.

On sait donc, de source sûre, que notre carte Cora Z7, nous fournit une horloge cadencée à 125 MHz, et qu'il va falloir prendre en compte cette horloge pour notre projet.

1.2.2. Carte Pmod

L'objectif de cette partie est de comprendre le fonctionnement de la carte Pmod.

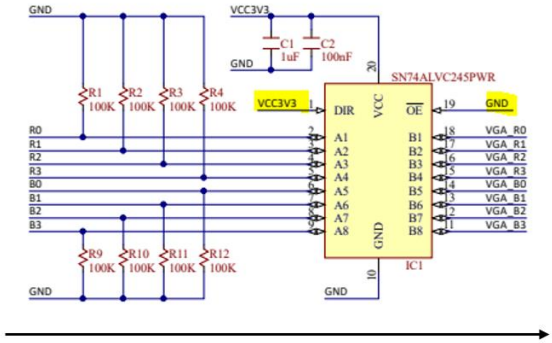
D'après la datasheet, on peut voir que la carte est composée de :

- Deux connecteurs Pmod
- Deux composants tranceivers
- Une partie de conversion numérique / analogique
- Un connecteur VGA

Les deux connecteurs Pmod permettent de localiser les signaux qui vont communiquer avec la carte Cora z7. L'emplacement des signaux sur les connecteurs Pmod seront décrit plus tard dans le fichier. (cf. Plan de validation de la solution intermédiaire)

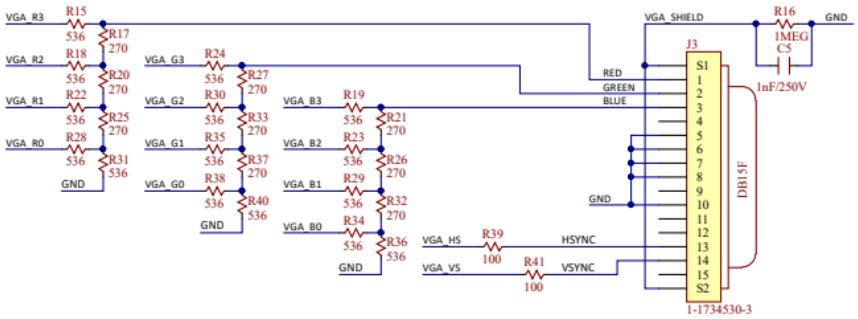
D'après la datasheet des tranceivers SN74AVC4T245 on peut voir que les deux composants sont configuré pour avoir une communication allant du port A vers le port B, dans notre cas de la carte Cora Z7 vers notre écran où sera projeter notre image.

CONTROL INPUTS		OUTPUT CIRCUITS		OPERATION
$\overline{OE}$	DIR	A PORT	B PORT	
L	L	Enabled	Hi-Z	B data to A bus
L	H	Hi-Z	Enabled	A data to B bus
H	X	Hi-Z	Hi-Z	Isolation



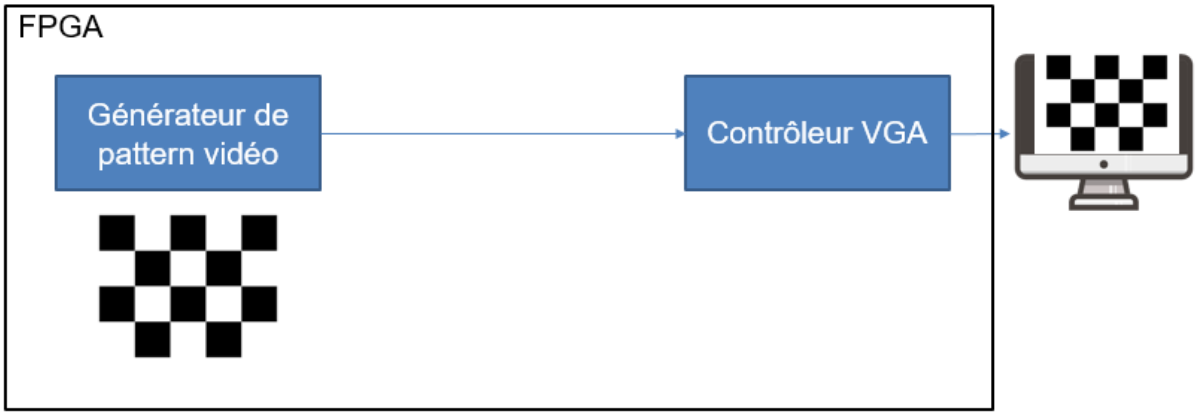
Sens de communication

Pour finir, les signaux sortant des tranceivers vont se retrouver dans un bloc de pont diviseur de tension afin de permettre une conversion des numérique en analogique (4 vers 1) puis sortie à l'aide du connecteur VGA.
On passe donc de 4 signaux d'intensité par couleurs à un signal qui sera variable entre 0V et 3.3V. (0V étant pour la couleur Noir et 3.3V pour la couleur Blanc.)
Le niveau de tension du signal de couleur permettra de définir l'intensité de la couleur.



1.3. Objectif de la solution intermédiaire

L'objectif de la solution intermédiaire est de générer un « pattern » vidéo (type damier noir et blanc) et de projeter ce dernier sur un écran à l'aide d'un contrôleur VGA (Cf. image ci-dessous).



<u>Remarque</u> : Le pattern que nous allons réaliser est un damier de couleur noir et blanc de cinq colonnes pour 4 lignes. Ce damier aura l'avantage de :

- Nous permettre de valider l'utilisation de l'intégralité des couleurs (car pour avoir du blanc, il faut utiliser 100% de l'intensité des couleurs rouge, vert et bleu)
- Le damier nous permettra de valider la gestion des lignes et des colonnes car ça nous permettra de valider qu'un carré n'est dévier et que l'on a donc bien des lignes horizontales et verticales
- Le damier noir et blanc nous permettre de mieux visualiser la mise en place de notre filtre Gaussien avec les nuances entre les hautes et les basses fréquences

2. Modules

Description des modules que nous aurons besoin de réaliser et de valider unitairement avant de les intégrer à notre solution intermédiaire finale.

2.1. Module d'horloge : Phase-Locked Loop (PLL)

Nous devons réaliser une horloge cadencée à la fréquence de fonctionnement de la norme VGA soit 25.175 MHz à partir de notre horloge d'entrée, cadencée à 125 MHz. Pour ce faire nous allons créer un module PLL.

Pour cela nous aurons les signaux suivants :

En entrée

- Clk : signal d'horloge de 125 MHz.
- Reset : signal permettant de réinitialiser notre module.

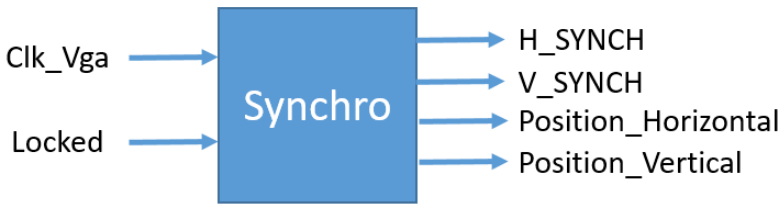
En Sortie

- Clk_Vga : signal d'horloge cadencé à 25.175 MHz.
- Locked : signal qui passe un '1' lorsque la clock de sortie est validée à la consigne demandée.

<u>Remarque</u> : pour la suite de notre projet nous utiliserons le signal Locked comme signal de reset des autres modules. Cela nous permettra de toujours travailler avec une horloge VGA stabilisée à 25.175MHz.

2.2. Module de synchronisation (Synchro)

Le module de synchronisation va nous permettre de réaliser le dimensionnement de notre afficheur, en fonction des informations données dans la partie Norme VGA.



Pour cela nous aurons les signaux suivants :

En entrée

- Clk_Vga : signal d'horloge de 25.175 MHz.
- Locked : signal permettant de réinitialiser notre module (Reset).

En Sortie

- H_SYNCH : signal de synchronisation des lignes
- V_SYNCH : signal de synchronisation des colonnes
- Position_Horizontal : compteur de pixel sur l'axe horizontal
- Position_Vertical : compteur de pixel sur l'axe vertical

2.3. Module de réalisation d'image (Pattern_VGA)

Le module pattern_VGA va nous permettre de :

- Réaliser notre damier sur la partie visible de notre image
- Forcée la couleur noire sur la partie invisible de notre image

Pour ce faire, nous allons créer le module suivant :

Pour cela nous aurons les signaux suivants :

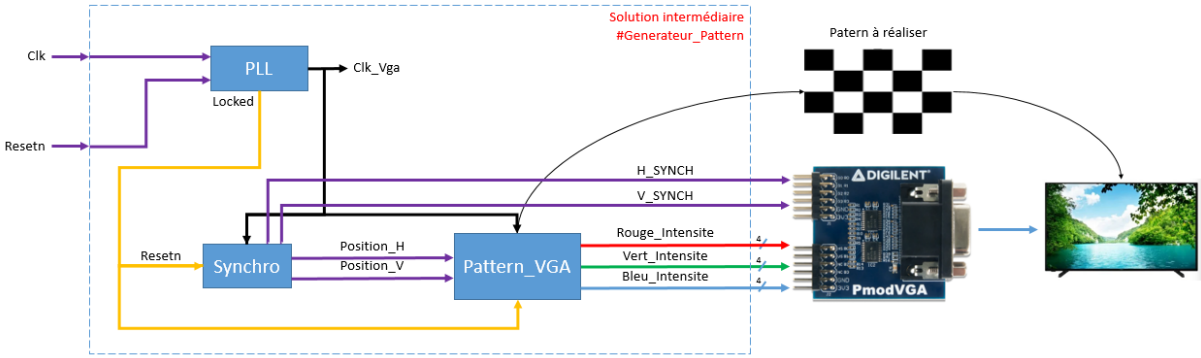
En entrée

- Clk_Vga : signal d'horloge de 25.175 MHz.
- Locked : signal permettant de réinitialiser notre module (Reset).
- Position_H : compteur de pixel sur l'axe horizontal
- Position_V : compteur de pixel sur l'axe vertical

En Sortie

- Rouge_Intensite : intensité de la couleur rouge sur 4 bits.
- Vert_Intensite : intensité de la couleur vert sur 4 bits.
- Bleu_Intensite : intensité de la couleur bleu sur 4 bits.

3. Réalisation de la solution Intermédiaire



La solution intermédiaire devra donner le résultat suivant :

Le module PLL nous délivre notre signal d'horloge permettant de faire fonctionner le VGA et un signal permettant de gérer nos reset système.

Le module synchro lui va nous fournir les signaux (H_Synch et V_Synch) qui iront directement sur la carte Pmod. Et les deux compteurs de pixel (horizontal et vertical).

La partie pattern_VGA va nous permettre de générer notre pattern vidéo souhaité. Ici, nous voulons créer un damier noir et blanc avec 5 colonnes et 4 lignes.

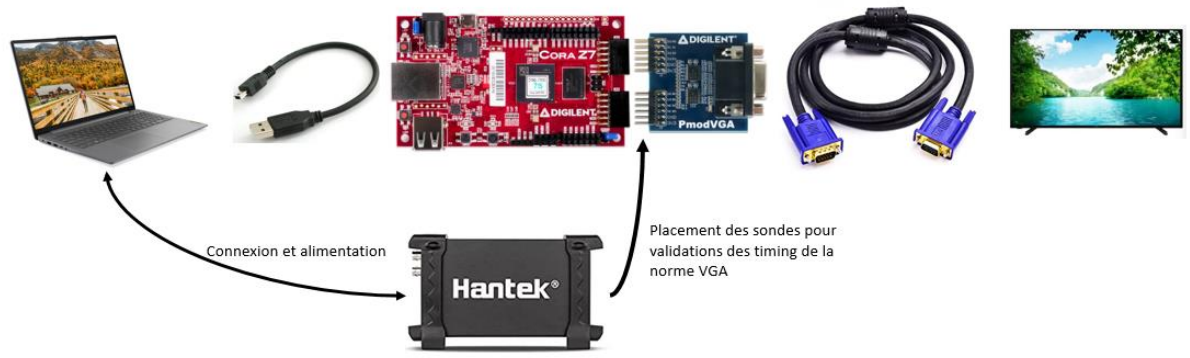
Cela implique que pour le format VGA (640 x 480 pixels) chaque rectangle du damier devra mesurer 128 x 120 pixels.

4. Plan de validation de la solution intermédiaire

L'objectif de ce plan de validation est de proposer une solution afin de valider le bon fonctionnement de notre proposition de solution.
Si on reprend le tableau de décomposition des pixels de la norme VGA présenté dans la partie Norme VGA.
Sachant que la résolution d'un pixel est de 25.175 MHz soit 39,72 ns et que la fréquence de rafraichissement verticale est de 31.46875 KHz soit de 31,78 µs (ce qui correspond à 480 lignes x 800 pixels), on obtient les timings suivant :

Nom	Horizontal (Pixel)	Horizontal (Timing µs)	Vertical (Pixel)	Vertical (Timing ms)
Image visible	640	25.422	480	15.253
Front Porch	16	0.636	10	0.318
Back Porch	48	1.907	33	1.049
Largeur de synchronisation	96	3.813	2	0.064
Nb total de pixel	**800**	**31.778**	**525**	**16.683**

Pour valider ces différents timings, nous utiliserons un oscilloscope Hantek et des sondes que nous placerons sur les broches à valider :



Le pinout de la carte Pmod VGA est le suivant :

Header J1						Header J2					
Pin	Signal	Description	Pin	Signal	Description	Pin	Signal	Description	Pin	Signal	Description
1	R0	Red 0	7	B0	Blue 0	1	G0	Green 0	7	HS	Horizontal Sync
2	R1	Red 1	8	B1	Blue 1	2	G1	Green 1	8	VS	Vertical Sync
3	R2	Red 2	9	B2	Blue 2	3	G2	Green 2	9	NC	Not Connected
4	R3	Red 3	10	B3	Blue 3	4	G3	Green 3	10	NC	Not Connected
5	GND	Power Supply Ground	11	GND	Power Supply Ground	5	GND	Power Supply Ground	11	GND	Power Supply Ground
6	VCC3V3	Positive Power Supply	12	VCC3V3	Positive Power Supply	6	VCC3V3	Positive Power Supply	12	VCC3V3	Positive Power Supply

4.1. Analyse de la version intermédiaire du projet

Cette partie va permettre de faire l'analyse de la partie intermédiaire de notre projet. Pour ce faire, nous utiliserons la partie simulation du logiciel Vivado.

4.1.1. Validation du module de synchronisation

Cette partie va permettre de réaliser la validation en simulation du module synchronisation présenté dans la partie Module de synchronisation (Synchro).
Pour ce faire, nous devrons commencer par réaliser le code VHDL du module, puis un test Bench qui permettra de faire fonctionner ce dernier.
Pour ce test bench, et comme la PLL n'est toujours pas mis en place, nous créerons directement une horloge cadencé à 25.175MHz (fréquence de fonctionnement du VGA).
La validation du module suivra les étapes suivantes :

- Validation du reset.
    - Si le reset est à 1, le module ne doit pas fonctionner
    - Si le reset est à 0, le module est en mode de fonctionnement
- Validation de la création de l'horloge du test Bench.
    - L'horloge doit être cadencée à 25.175MHz.
- Validation du signal de synchronisation horizontal (H_SYNCH).
    - L'impulsion doit être de 3.8µs
    - La période doit être de 31.778µs
- Validation du signal de synchronisation vertical (V_SYNCH).
    - L'impulsion doit être de 64ms
    - La période doit être de 16.68ms

4.1.2. Validation de l'ajout du module PLL

Après avoir ajouté le module PLL à notre projet, nous modifieront notre test Bench pour avoir une horloge d'entrée cadencée à 125MHz (horloge qui sera plus tard fournis par la carte Cora Z7).

La validation suivra les étapes suivantes :

- Validation de la création de l'horloge du test Bench.
    - L'horloge doit être cadencée à 125MHz
- Validation de l'horloge de sortie de la PLL.
    - Le signal Locked doit être à 1 pour indiquer que l'horloge de sortie soit stabilisée
    - L'horloge doit être cadencée à 25.175MHz

4.1.3. Validation de l'ajout du module Pattern_VGA

Une fois le module ajouté à notre projet, nous n'aurons pas besoin de modifier le test Bench pour réaliser la simulation.

La validation suivra les étapes suivantes :

- Valider que les sorties d'intensité de couleur sont à 0 lorsque l'on est dans les zones front porsh, back porsh et largeur de synchronisation.
- Visualiser la représentation d'une ligne avec 2 parties blanches
- Visualiser la représentation d'une ligne avec 3 parties blanches

4.2. Test de la version intermédiaire du projet

Le test de la version intermédiaire du projet passe par :

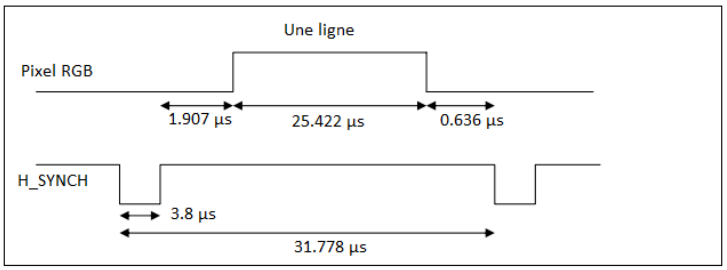
- La réalisation de la synthèse du projet
  - Vérifier que le schéma obtenu est bien en accord avec la solution proposée
- La réalisation de l'implémentation du projet
  - Vérifier qu'il n'y a pas de déviation de timing
- La génération du Bitstream et l'envoie du projet sur la carte Cora.
  - Validation à l'oscilloscope du l'impulsion et de la période de H_SYNCH
  - Validation à l'oscilloscope du l'impulsion et de la période de V_SYNCH
  - Validation de la représentation d'une ligne avec 2 parties blanches
  - Validation de la représentation d'une ligne avec 3 parties blanches

Pour valider les signaux à l'aide de l'oscilloscope, le bitstream doit être envoyé à la carte cora. Et les sondes placées aux différents endroits du connecteur Pmod de la carte Cora, comme expliqué dans la partie : Plan de validation de la solution intermédiaire.

4.2.1. Signal H_SYNCH

Pour valider le signal H_SYNCH, on va connecter une sonde d'oscilloscope à la broche 7 du connecteur J2 et connecter la masse de cette sonde à la masse de notre système.
On connecte également une seconde sonde à l'une de nos broche de pixel afin de faire valider la taille de l'image (exemple broche 1 du connecteur J2 pour visualiser la couleur vert0).

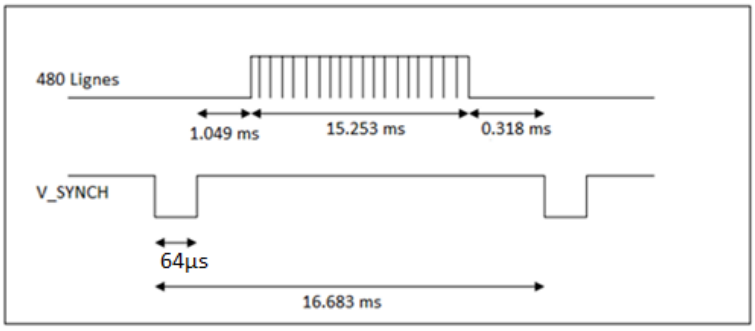
Pour réaliser la capture, on synchronise notre oscilloscope pour détecter un front descendant de notre signal H_SYNCH. Et on valide les timings suivant :



4.2.2. Signal V_SYNCH

Pour valider le signal V_SYNCH, on va connecter une sonde d'oscilloscope à la broche 8 du connecteur J2 et connecter la masse de cette sonde à la masse de notre système.
On connecte également une seconde sonde à l'une de nos broche de pixel afin de faire valider la taille de l'image (exemple broche 1 du connecteur J2 pour visualiser la couleur vert0).

Pour réaliser la capture, on synchronise notre oscilloscope pour détecter un front descendant de notre signal H_SYNCH. Et on valide les timings suivant :



4.3. Démonstration de la version intermédiaire du projet

La démonstration de la solution intermédiaire va nous permettre de voir que l'on arrive bien à projeter sur un écran notre pattern. Pour ce faire nous allons réaliser le montage présenté dans la partie Plan de validation de la solution intermédiaire.

Il y a deux cas possibles :

- Lorsque le bouton reset est activé, le damier ne s'affiche pas
- Lorsque le bouton reset n'est pas activé, le damier est affiché.

5. Présentation et réalisation de la solution finale

Une fois la solution intermédiaire validée, nous obtenons un pattern vidéo (damier) qui est projeté à l'écran via le VGA. L'objectif de la solution finale est d'ajouter un filtre entre la génération du pattern vidéo et la sortie projetée. Nous utiliserons dans notre cas un filtre Gaussien.

5.1. Présentation du filtre de gauss

Le filtre Gaussien fait partie de la famille des filtres d'atténuations. Ce filtre permet d'uniformiser les images en donnant un côté flouté aux détails présents sur l'image. Ce qui revient à dire en donnant un côté flouté et harmonieux à l'image.

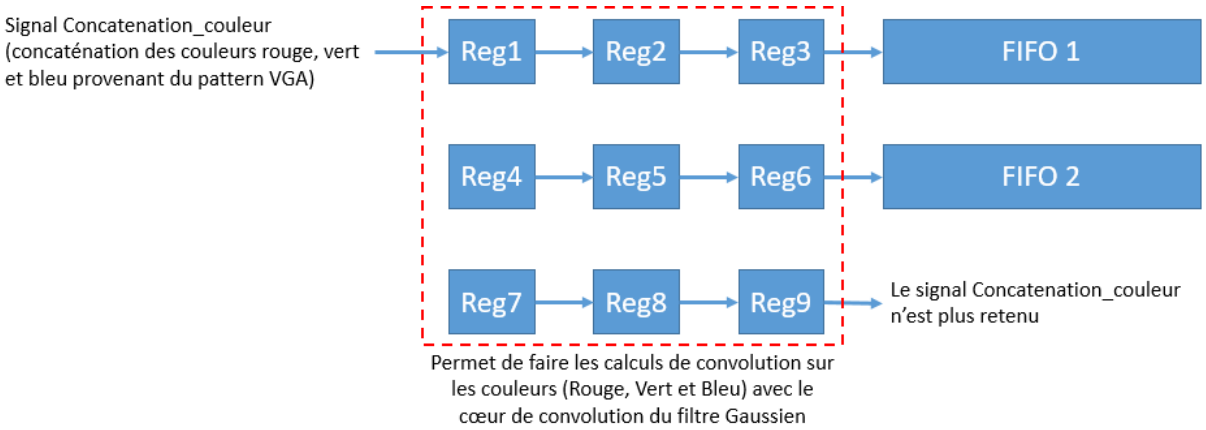
Dans notre cas, notre damier ne sera donc plus net dans les zones de transitions blanc/noir ou noir/blanc mais plutôt flouté dans ces mêmes zones de transitions.

Pour ce faire, le filtre gaussien va (pixel par pixel) réaliser une convolution avec tous les pixels avoisinants le pixel à traiter. Dans notre cas, nous utiliserons donc le cœur de convolution suivant :

1	2	1	K1	K2	K3
2	4	2	K4	K5	K6
1	2	1	K7	K8	K9

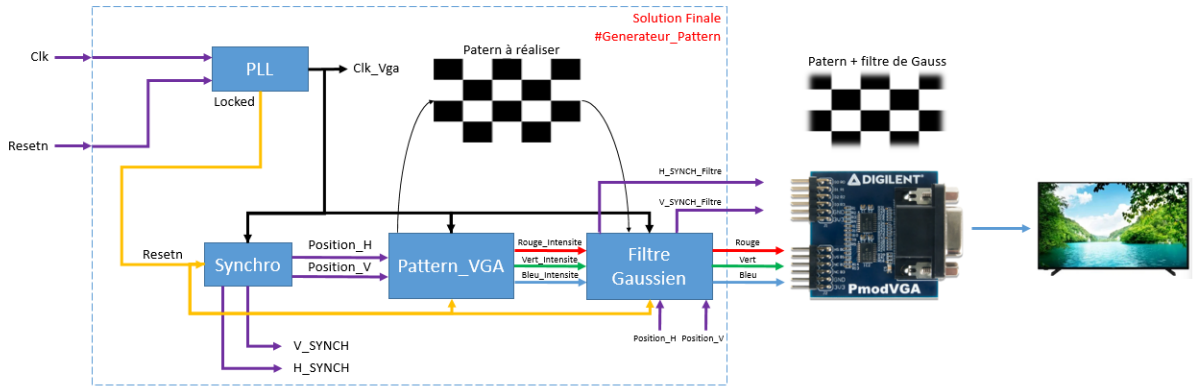
5.2. Stockage des pixels dans les registres

Pour pouvoir réaliser les calculs de convolution, il nous faut stocker les valeurs des pixels d'entrés (sortie du module pattern_VGA). L'enregistrement des pixels se fait ligne par ligne avec des passages par des registres afin de pouvoir réaliser le calcul de convolution du filtre.
Pour ce faire, nous utiliserons deux Fifos ainsi que 9 registres comme le montre le schéma suivant :



5.3. Réalisation de la solution finale

L'architecture de la solution finale sera la suivante :



La solution finale sera donc légèrement modifiée :

Le module PLL nous délivre notre signal d'horloge permettant de faire fonctionner le VGA et un signal permettant de gérer nos reset système lorsque l'horloge VGA sera stabilisée.

Le module synchro lui va nous fournir les signaux (H_Synch et V_Synch) qui n'iront plus directement sur la carte Pmod. On y trouvera également les deux compteurs de pixel (horizontal et vertical) qui seront utilisés pour générer le pattern vidéo ainsi que la création du filtre.

Le module pattern_VGA va nous permettre de générer notre pattern vidéo souhaité. Ici, nous voulons créer un damier noir et blanc avec 5 colonnes et 4 lignes à l'aide des compteurs de pixel. Cela implique que pour le format VGA (640 x 480 pixels) chaque rectangle du damier devra mesurer 128 x 120 pixels.
Ce module va nous fournir en sortie 3 vecteurs de signaux qui nous permettrons de gérer les couleurs (rouge, vert et bleu).

Le module filtre Gaussien recevra le pattern vidéo créé en amont. Son rôle sera d'y ajouter le filtre par-dessus, de redéfinir les signaux de synchronisation horizontal et vertical afin de pouvoir projeter le tout à l'aide de la carte Pmod.

5.4. Ajout du module filtre Gaussien

Le module Filtre_Gaussien va nous permettre de :

- Mettre à jour les signaux de synchronisation Horizontal et Vertical
- Ajouter le filtre sur la partie visible de l'image
    - Remplir les différents registres
    - Activer ou désactiver la lecture ou l'écriture des FIFOs
    - Réaliser les différents calculs de convolution en fonction des couleurs d'entrées
    - Sortir les nouveaux résultats des pixels

Pour ce faire, nous allons créer le module suivant :

Pour cela nous aurons les signaux suivants :

En entrée

- Clk_Vga : signal d'horloge de 25.175 MHz.
- Locked : signal permettant de réinitialiser notre module (Reset).
- Position_H : compteur de pixel sur l'axe horizontal.
- Position_V : compteur de pixel sur l'axe vertical.
- Rouge_Intensite : intensité de la couleur rouge sur 4 bits provenant du module Pattern_VGA (damier).
- Vert_Intensite : intensité de la couleur rouge sur 4 bits provenant du module Pattern_VGA (damier).
- Bleu_Intensite : intensité de la couleur rouge sur 4 bits provenant du module Pattern_VGA (damier).

En Sortie

- Rouge : intensité de la couleur rouge sur 4 bits après l'ajout du filtre.
- Vert : intensité de la couleur vert sur 4 bits après l'ajout du filtre.
- Bleu : intensité de la couleur bleu sur 4 bits après l'ajout du filtre.
- H_SYNCH_filtre : signal de synchronisation des lignes en sortie du filtre.
- V_SYNCH_filtre : signal de synchronisation des colonnes en sortie du filtre.

6. Plan de validation de la solution finale

Le plan de validation de la solution finale sera sensiblement le même que celui utilisé pour faire valider la solution intermédiaire.
La seule différence sera portée sur la validation du nouveau module ajouté : Module filtre Gaussien.

6.1. Analyse de la version finale du projet

Cette partie va permettre de faire l'analyse de la partie finale de notre projet. Pour ce faire, nous utiliserons la partie simulation du logiciel Vivado.

6.1.1. Validation de l'ajout du module Filtre_Gauss

Une fois le module ajouté à notre projet, nous n'aurons pas besoin de modifier le test Bench pour réaliser la simulation.
La validation suivra les étapes suivantes :

- Validation du remplissage des registres
- Validation de l'écriture et de la lecture des registres
- Validation de la convolution gaussienne
- Validation (à l'aide d'un signal d'activation vidéo provisoire) de l'activation de l'image (afin de ne pas avoir de décalage d'image).

6.2. Test de la version finale du projet

Le test de la version finale du projet passe par :

- La réalisation de la synthèse du projet
  - Vérifier que le schéma obtenu est bien en accord avec la solution proposée
- La réalisation de l'implémentation du projet
  - Vérifier qu'il n'y a pas de déviation de timing
- La génération du Bitstream et l'envoie du projet sur la carte Cora.
  - Validation à l'oscilloscope du l'impulsion et de la période de H_SYNCH
  - Validation à l'oscilloscope du l'impulsion et de la période de V_SYNCH

Pour valider les signaux à l'aide de l'oscilloscope, le bitstream doit être envoyé à la carte cora. Et les sondes placées aux différents endroits du connecteur Pmod de la carte Cora, comme expliqué dans la partie : Plan de validation de la solution intermédiaire.

6.3. Démonstration de la version finale du projet

La démonstration de la solution finale va nous permettre de voir que l'on arrive bien à projeter sur un écran notre pattern flouté. Pour ce faire nous allons réaliser le montage présenté dans la partie Plan de validation de la solution intermédiaire.