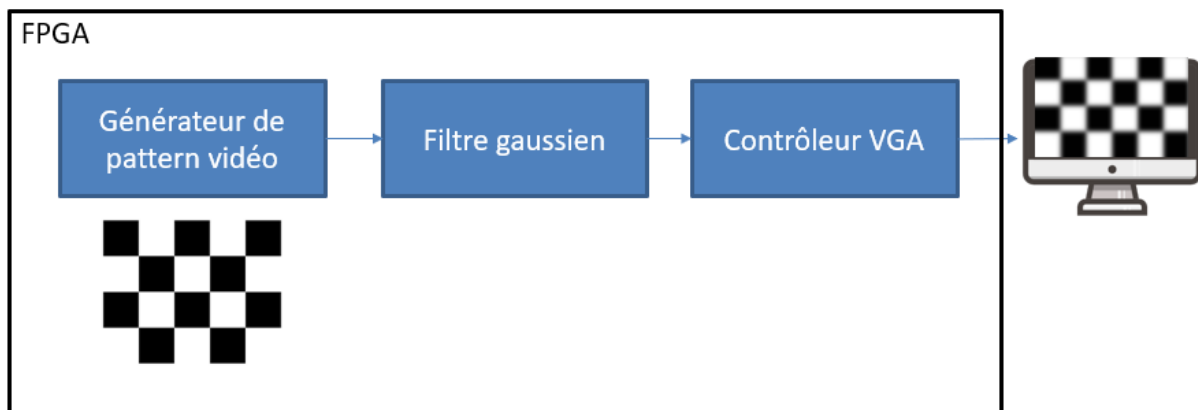


# Réalisation d'une IP de traitement d'image sur cible Zynq7020 et affichage VGA



## Table des matières

|      |  |    |
|------|--|----|
| I.   | LE FORMAT VGA .....                      | 3  |
| a.   | Qu'est-ce que le format VGA ? .....      | 3  |
| b.   | Comment fonctionne le format VGA ? ..... | 4  |
| c.   | Les signaux de synchronisation .....     | 4  |
| d.   | Les spécifications VGA .....             | 5  |
| II.  | LE MATERIEL .....                        | 6  |
| a.   | Matériel à disposition .....             | 6  |
| b.   | La carte CoraZ7 .....                    | 6  |
| c.   | La carte Pmod VGA .....                  | 8  |
| d.   | Le câble VGA .....                       | 10 |
| III. | LA PHASE INTERMEDIAIRE .....             | 11 |
| a.   | Description .....                        | 11 |
| b.   | Architecture globale .....               | 12 |
| c.   | Module VGA_sync .....                    | 13 |
| IV.  | PLAN DE VALIDATION .....                 | 15 |
| a.   | Analyse .....                            | 15 |

## I. LE FORMAT VGA

### a. Qu'est-ce que le format VGA ?

#### CARACTERISTIQUES :

- La résolution standard pour l'image visible est de 640x480 pixels. Cette image est représentée en bleu dans l'illustration ci-dessous.
- Chaque pixel possède 3 composantes (RGB : Red, Green, Blue), codée sur 4 bits.
- Des pixels additionnels sont utilisés pour l'image virtuelle (zone utilisée pour les signaux de synchronisation notamment), ce qui agrandit la zone à 800x525 pixels.
- Le rafraichissement de l'écran se fait 60 fois par seconde, soit une fréquence de 60Hz.
- Le format VGA est un format analogique.

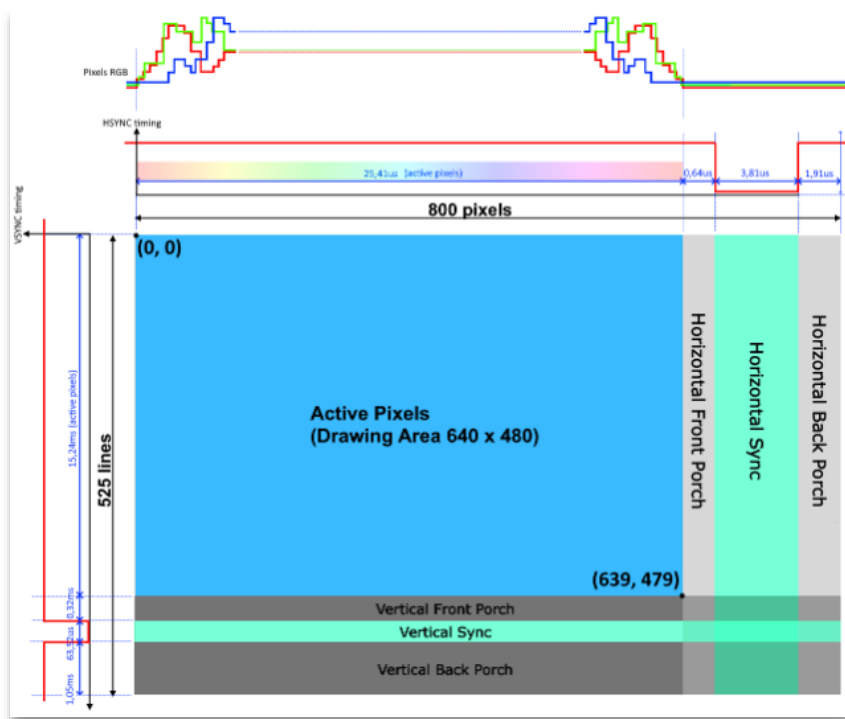
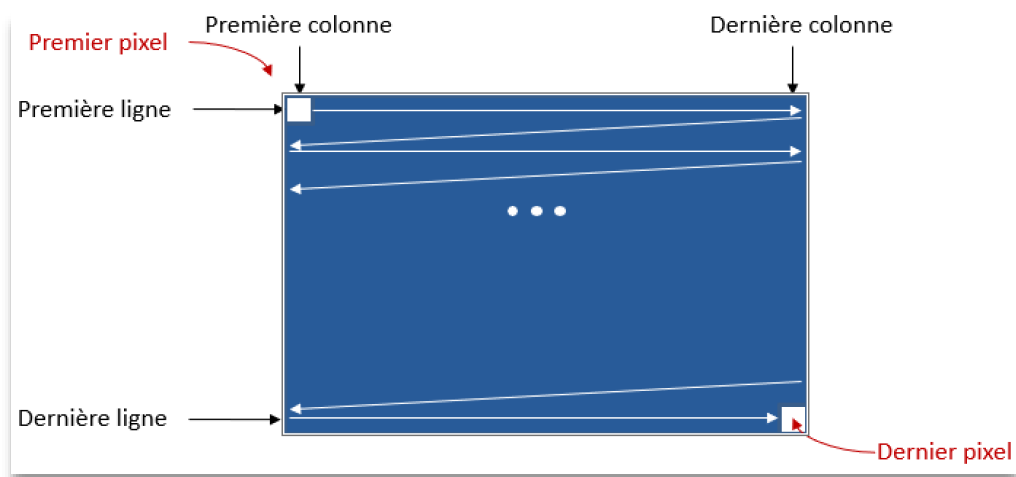


Image au format VGA

## b. Comment fonctionne le format VGA ?

### PRINCIPE :

- Balayage de la première ligne de gauche à droite,
- Balayage de la seconde ligne de gauche à droite,
- ...,
- Balayage de la première ligne à la dernière ligne,
- Puis, retour à la première ligne.



Balayage de l'image

## c. Les signaux de synchronisation

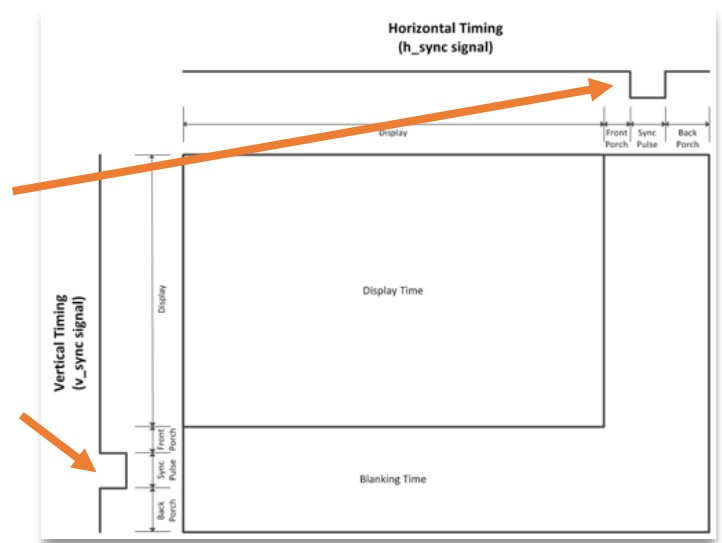
Afin de réaliser le balayage de l'image, des signaux de synchronisation sont nécessaires pour se repérer sur les lignes et les colonnes.

### SIGNAL DE SYNCHRO HORIZONTAL : [hsync](#)

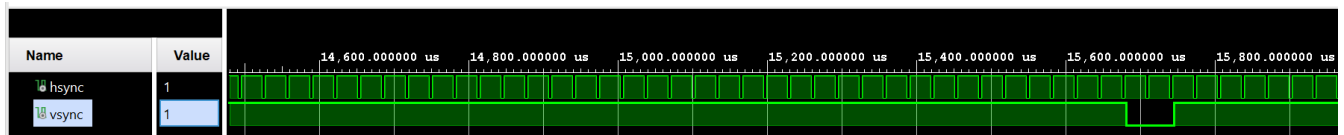
- ❖ Actif à l'état bas dans la zone virtuelle
- ❖ A pour objectif de savoir où on se situe sur la ligne

### SIGNAL DE SYNCHRO VERTICAL : [vsync](#)

- ❖ Actif à l'état bas dans la zone virtuelle
- ❖ A pour objectif de savoir où on se situe sur l'image



Les signaux **hsync** et **vsync** étant actifs à l'état bas, une simulation de ces signaux pourrait ressembler à ceci :



#### d. Les spécifications VGA

Le tableau ci-dessous est issu des spécifications :

##### General timing

|                     |              |
|---------------------|--------------|
| Screen refresh rate | 60 Hz        |
| Vertical refresh    | 31.46875 kHz |
| Pixel freq.         | 25.175 MHz   |

##### Horizontal timing (line)

Polarity of horizontal sync pulse is negative.

| Scanline part | Pixels | Time [μs]        |
|---------------|--------|------------------|
| Visible area  | 640    | 25.422045680238  |
| Front porch   | 16     | 0.63555114200596 |
| Sync pulse    | 96     | 3.8133068520357  |
| Back porch    | 48     | 1.9066534260179  |
| Whole line    | 800    | 31.777557100298  |

##### Vertical timing (frame)

Polarity of vertical sync pulse is negative.

| Frame part   | Lines | Time [ms]         |
|--------------|-------|-------------------|
| Visible area | 480   | 15.253227408143   |
| Front porch  | 10    | 0.31777557100298  |
| Sync pulse   | 2     | 0.063555114200596 |
| Back porch   | 33    | 1.0486593843098   |
| Whole frame  | 525   | 16.683217477656   |

##### Compréhension des données :

###### ➤ General timing :

Nous avons 800x525 pixels par image (visible + virtuelle).  
 800 x 525 x 60 pixels traités par seconde  
 Soit 25.2 millions de pixels traités par seconde  
 Ce qui ne correspond pas tout à fait à 25.175MHz.  
 De fait, la fréquence de rafraichissement de l'écran est plutôt de 59.94Hz.

###### ➤ Horizontal timing :

$tp = 1/25.175 = 0.039722 \text{ us} = 39.722 \text{ ns}$  pour un pixel  
 En multipliant le nombre de pixels par  $tp$ , nous retrouvons bien les valeurs de la colonne « Time ».  
 Ex. :  
 $640 \times tp = 25.422 \text{ microsecondes}$

###### ➤ Vertical timing :

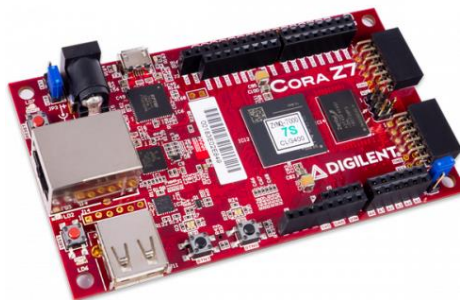


Une ligne comprend 800 pixels, soit  $tl$  environ 31.778 microsecondes par ligne.

En multipliant le nombre de lignes par  $tl$ , nous retrouvons bien les valeurs de la colonne « Time ».

Ex. :  
 $480 \times tl = 15.253 \text{ ms}$

## II. LE MATERIEL

### a. Matériel à disposition

| Carte Xilinx CoraZ7 avec un câble USB pour connexion à l'ordinateur               | Carte Pmod VGA (Digilent)  | Câble VGA mâle-mâle   |
|---|--|---|
|  |  |  |

La carte CoraZ7 va nous permettre de générer les signaux de synchronisation au format **numérique**, ainsi que les signaux RGB avec les niveaux d'intensité de chaque couleur. Cependant, comme nous l'avons vu en introduction, les signaux VGA sont **analogiques**. Par conséquent, nous allons avoir besoin d'un convertisseur numérique/analogique. Ce sera la fonction de la carte Pmod VGA.

### b. La carte CoraZ7

L'horloge disponible en utilisant un composant PLL sur la carte coraZ7 est de 125 MHz comme le montre l'extrait ci-dessous issu du manuel de référence :

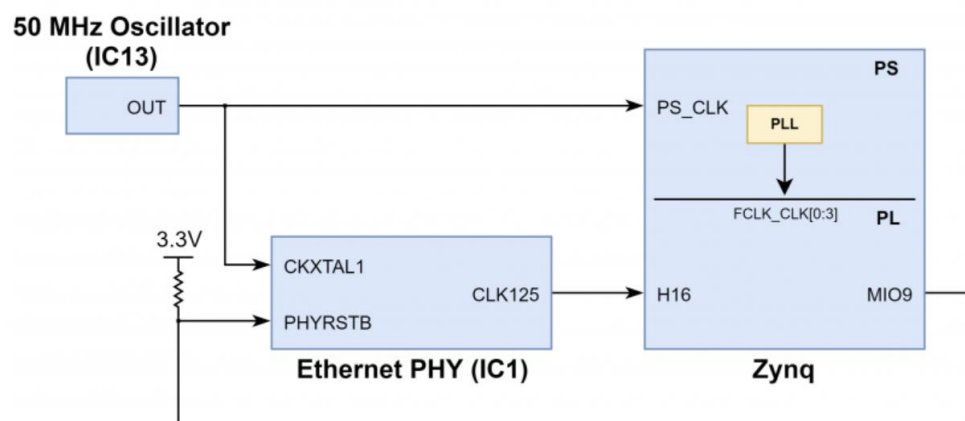
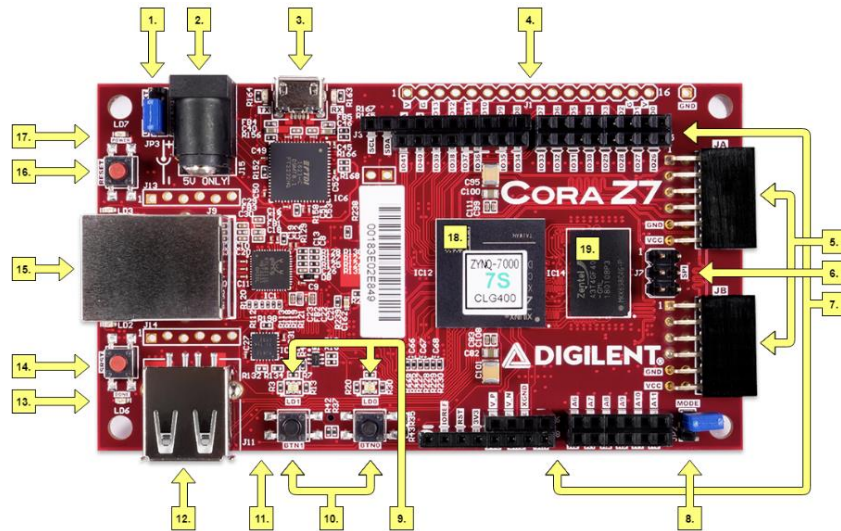


Figure 9.1. Cora Z7 Clocking

Cette donnée sera importante pour la suite car nous ne devons pas perdre de vue que le format VGA demande une fréquence de fonctionnement de 25.175 MHz.



| Callout | Description                              | Callout | Description                            |
|---------|--|---------|--|
| 1       | Power select jumper (Ext. supply / USB)  | 11      | microSD card slot (underside of board) |
| 2       | Power jack (for optional ext. supply)    | 12      | USB host port                          |
| 3       | Shared USB JTAG / UART port              | 13      | FPGA programming DONE LED              |
| 4       | Unloaded expansion header                | 14      | Processor subsystem reset button       |
| 5       | Pmod connectors                          | 15      | Ethernet port                          |
| 6       | SPI header (Arduino/ChipKIT compatible)  | 16      | Power on reset button                  |
| 7       | Arduino/ChipKIT shield connectors        | 17      | Power good LED                         |
| 8       | Programming mode jumper (JTAG / microSD) | 18      | Zynq-7000                              |
| 9       | User tri-color LEDs                      | 19      | DDR3L memory                           |
| 10      | User push buttons                        |         |  |

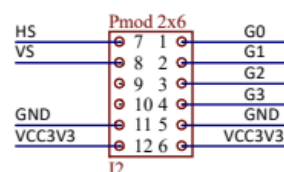
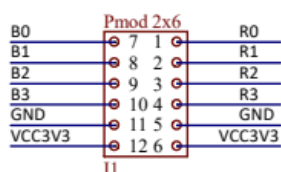
Les signaux numériques générés par la carte coraZ7 seront disponibles en position 5 sur le schéma ci-dessus (Pmod connectors JA et JB). C'est donc à cet endroit que nous devons connecter la carte Pmod VGA.

### c. La carte Pmod VGA

La carte Pmod VGA nous servira de convertisseur numérique/analogique. Pour comprendre comment elle fonctionne, nous avons extrait du manuel de référence les schémas ci-dessous :

Nous retrouvons sur le premier schéma le brochage attendu par la carte coraZ7 :

- HS (synchro horizontal)
- VS (synchro vertical)
- Bx, Rx, Gx, les signaux RGB
- GND, la masse
- VCC3V3, tension de 3.3V



Dans notre configuration :

- /OE est à l'état bas,
- Et DIR est à l'état haut.

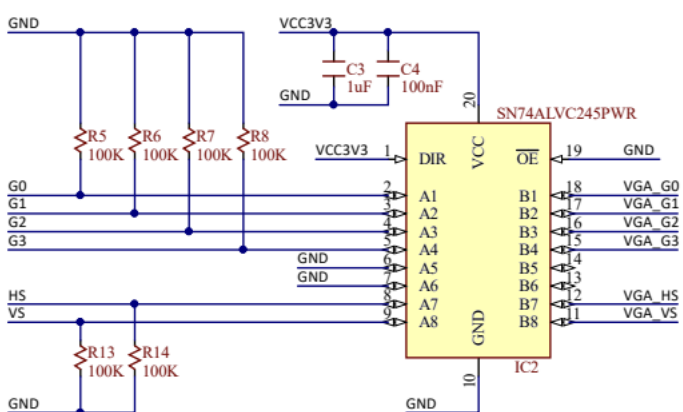
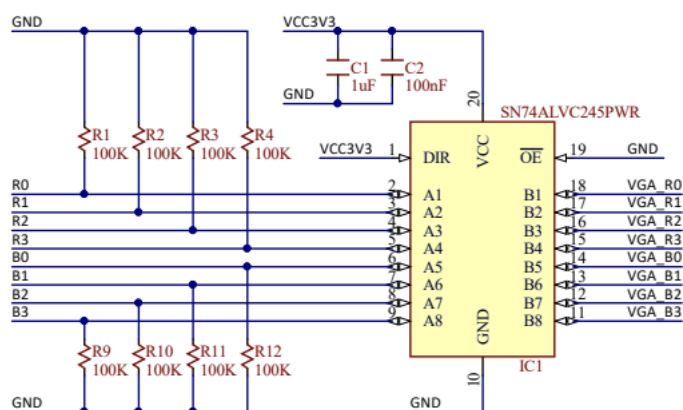
Nous sommes dans la configuration « A data to B bus ».

FUNCTION TABLE

| INPUTS |     | OPERATION       |
|--------|-----|-----------------|
| OE     | DIR |                 |
| L      | L   | B data to A bus |
| L      | H   | A data to B bus |
| H      | X   | Isolation       |

Fonctions du SN74ALVC245PWR :

- Conçu pour convertir les signaux logiques entre des niveaux de tension basse tension (LVCMOS) et des niveaux de tension plus élevés compatibles avec la technologie TTL.
- Protection contre les courts-circuits et les surintensités.

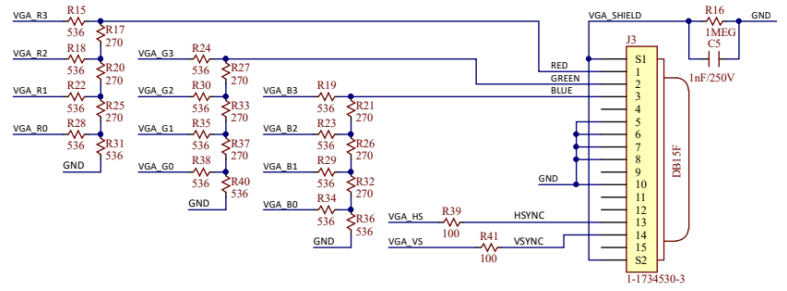




Circuit permettant de combiner les tensions :

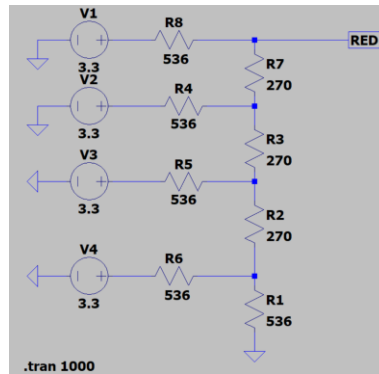
- VGA\_Rx en un seul signal,
- VGA\_Bx en un seul signal,
- VGA\_Gx en un seul signal.

Les signaux RGB ont un niveau entre 0V et 3.1V suivant les niveaux d'entrées.



Une simulation sous LT-spice nous a permis d'obtenir le résultat ci-dessous :

|        |        | Tension (en V) |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|--------|--------|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Entrée | VGA R3 | 0              | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 1   | 1   | 1   | 1   | 1   | 1   | 1   |
|        | VGA R2 | 0              | 0   | 0   | 0   | 1   | 1   | 1   | 1   | 0   | 0   | 0   | 0   | 1   | 1   | 1   | 1   |
|        | VGA R1 | 0              | 0   | 1   | 1   | 0   | 0   | 1   | 1   | 0   | 0   | 1   | 1   | 0   | 0   | 1   | 1   |
|        | VGA R0 | 0              | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   | 0   | 1   |
| Sortie | RED    | 0.0            | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 | 1.2 | 1.4 | 1.6 | 1.8 | 2.1 | 2.3 | 2.5 | 2.7 | 2.9 | 3.1 |



d. Le câble VGA

Le câblage du câble femelle VGA est détaillé ci-dessous :

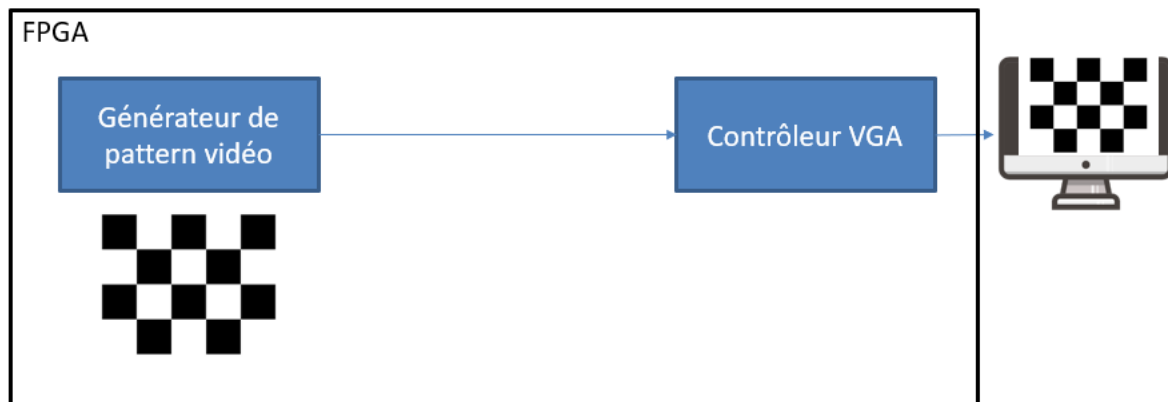
Câble et connecteur femelle VGA (DE-15)

| Pin | Name                 | Dir | Description  |
|-----|----------------------|-----|--|
| 1   | RED                  | →   | Red Video (75 ohm, 0.7 V p-p)  |
| 2   | GREEN                | →   | Green Video (75 ohm, 0.7 V p-p)  |
| 3   | BLUE                 | →   | Blue Video (75 ohm, 0.7 V p-p)   |
| 4   | RES                  |     | RESERVED   |
| 5   | SGND                 | —   | Ground   |
| 6   | RGND                 | —   | Red Ground   |
| 7   | GGND                 | —   | Green Ground   |
| 8   | BGND                 | —   | Blue Ground  |
| 9   | KEY                  | -   | Key (No pin) / Optional +5V output from graphics card  |
| 10  | GND                  | —   | Sync Ground  |
| 11  | ID0                  | ←   | Monitor ID Bit 0 (optional)  |
| 12  | SDA                  | ↔   | I2C bidirectional data line  |
| 13  | HSYNC<br>or<br>CSYNC | →   | Horizontal Sync (or Composite Sync)  |
| 14  | VSYNC                | →   | Vertical Sync which works also as data clock   |
| 15  | SCL                  | ↔   | I2C data clock in DDC2, Monitor ID3 in DDC1. A slave can pull SCL low to make the master wait. |

### III. LA PHASE INTERMEDIAIRE

#### a. Description

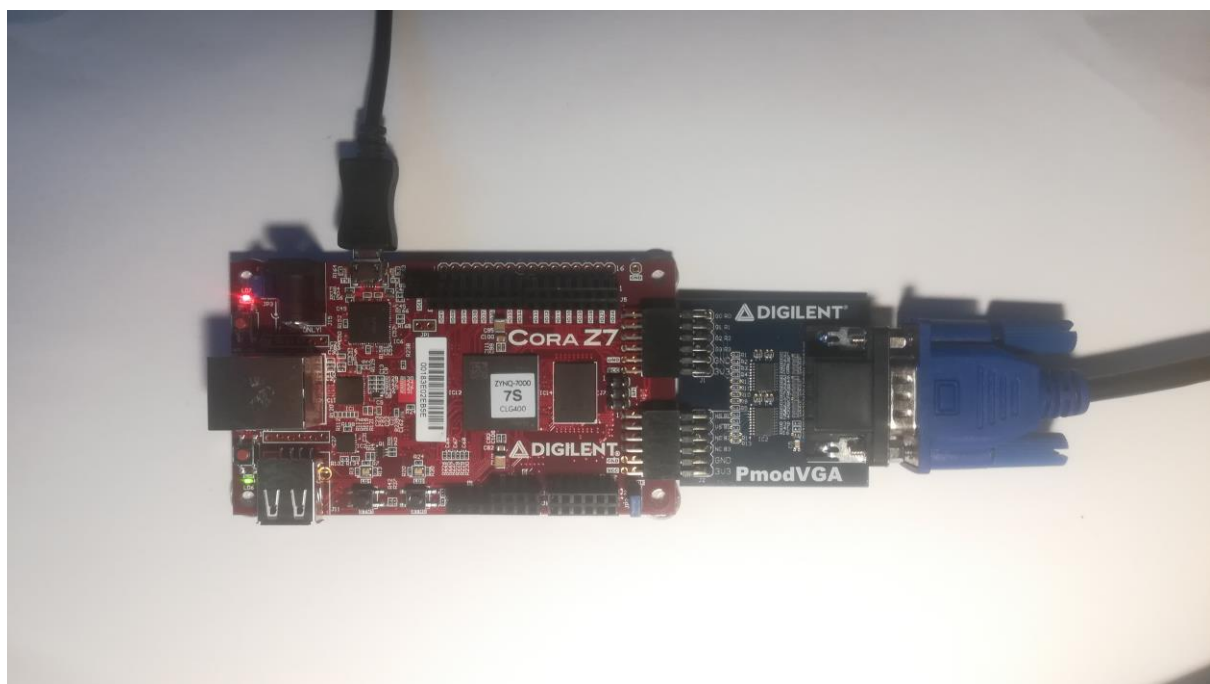
Dans un premier temps, nous allons chercher à afficher sur un écran un damier.



Ainsi, nous allons devoir concevoir une architecture sur la carte coraZ7 permettant :

- De générer les signaux de synchronisation (hsync et vsync),
- De générer les signaux RGV.


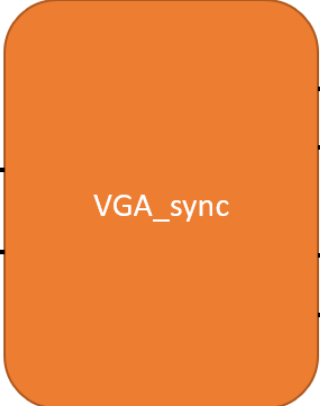
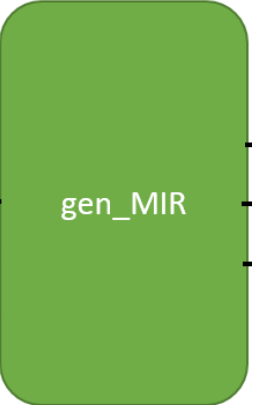
La carte Pmod VGA sera connectée sur les connecteurs Pmod JA et JB de la carte Xilinx afin de réaliser la conversion numérique/ analogique. Enfin, un câble VGA reliera la carte Pmod VGA à un écran.

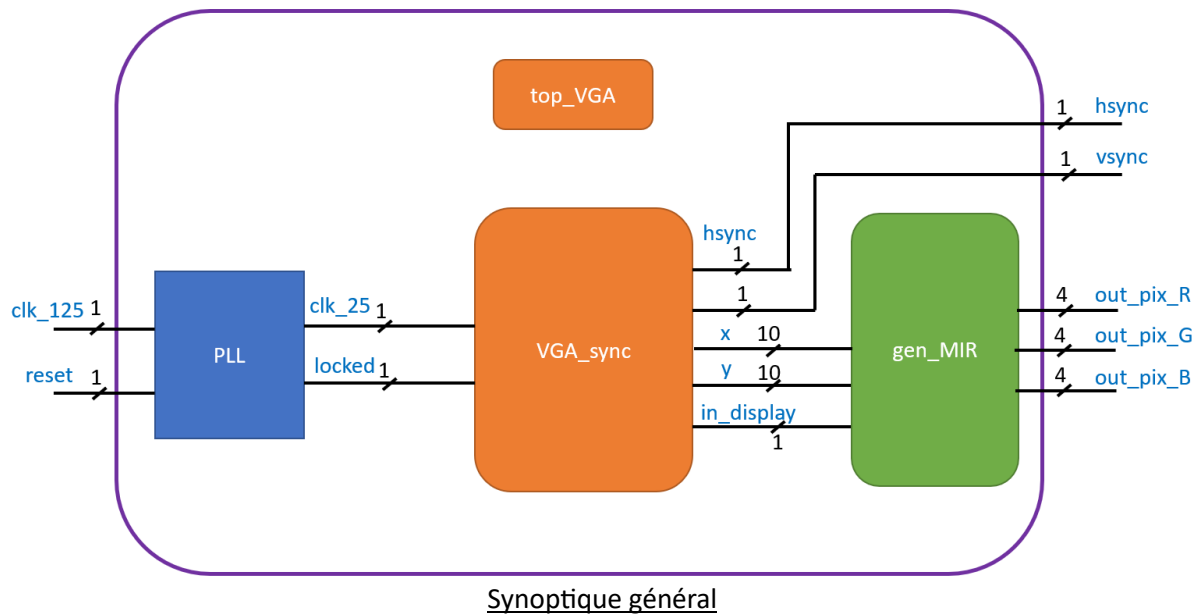


## b. Architecture globale

Nous avons choisi d'utiliser un composant pour générer les signaux de synchronisation (VGA\_sync), et un autre pour créer l'image (gen\_MIR).

Ainsi, nous pouvons décomposer l'architecture en 3 parties :

|   |   |
|---|---|
|    | <p>Le composant PLL nous permet de fournir l'horloge à 25.175MHz nécessaire au format VGA.</p> <p>Entrées :</p> <ul style="list-style-type: none"> <li>- Bouton reset,</li> <li>- Horloge à 125MHz de la carte coraZ7.</li> </ul> <p>Sorties :</p> <ul style="list-style-type: none"> <li>- Signal locked qui servira de reset au composant VGA_sync,</li> <li>- Horloge à 25.175MHz.</li> </ul>  |
| <p>Entrées :</p> <ul style="list-style-type: none"> <li>- Signal locked qui sert de reset au composant VGA_sync,</li> <li>- Horloge à 25.175MHz.</li> </ul> <p>Sorties :</p> <ul style="list-style-type: none"> <li>- Signaux de synchronisation nécessaires au balayage de l'écran (hsync et vsync),</li> <li>- Coordonnées des pixels au fur et à mesure du balayage (x et y),</li> <li>- Signal indiquant si on se situe dans l'image visible (in_display).</li> </ul> |   |
|    | <p>Entrées :</p> <ul style="list-style-type: none"> <li>- Coordonnées des pixels au fur et à mesure du balayage (x et y),</li> <li>- Signal indiquant si on se situe dans l'image visible (in_display).</li> </ul> <p>Sorties :</p> <ul style="list-style-type: none"> <li>- out_pix_R = niveau d'intensité du pixel en rouge,</li> <li>- out_pix_G = niveau d'intensité du pixel en vert,</li> <li>- out_pix_B = niveau d'intensité du pixel en bleu.</li> </ul> |



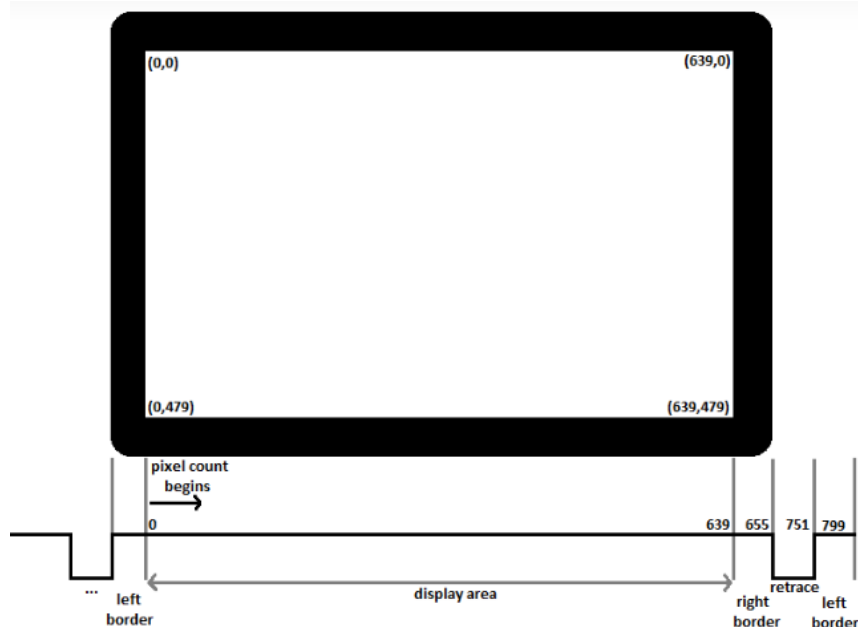
### c. Module `VGA_sync`

#### 1) Traitement des abscisses x :

Un premier compteur « `count_x` » nous permet de compter les fronts montants de l'horloge à 25.175MHz et ainsi de calculer la valeur de `x`.

Lorsque le compteur « `count_x` » a compté 800 périodes, le signal `end_count_x` se positionne à 1 le temps d'une période et le compteur est réinitialisé.

La valeur de `x` est récupérée par un module « `calculate_hsync` » qui va créer le signal `hsync`.



## 2) Traitement des ordonnées y :

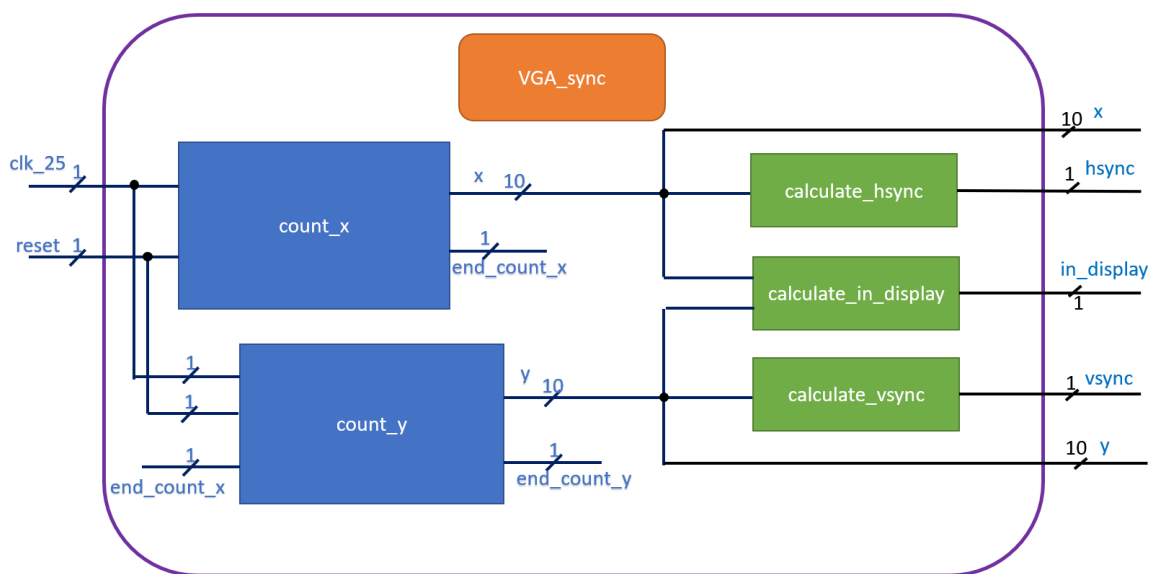
Un second compteur « count\_y » compte les signaux end\_count\_x reçus. Ceci lui permet de définir l'ordonnée y.

La valeur de y est récupérée par un module « calculate\_vsync » qui va créer le signal vsync.



## 3) Module in\_display :

Suivant les valeurs de x et de y reçues, le module in\_display va déterminer si le pixel est dans l'image visible ou virtuelle.



Synoptique du module VGA\_sync

## IV. PLAN DE VALIDATION

Afin de valider le design, 3 étapes sont nécessaires :

- Analyse : résultat obtenu en mettant en œuvre une modélisation ou une simulation d'une partie du système,
- Test : Résultat obtenu par mesure physique (Scope ou ILA) du système soumis à un pattern d'entrée connu,
- Démonstration : Résultat obtenu en mettant en œuvre le système dans les conditions de fonctionnement finales (Vidéo du système en fonctionnement, autre acquisition vidéo).

### a. Analyse