TP2 VHDL
Affichage VGA

1 Le matériel

VGA (Video Graphics Adapter) est une norme de signal vidéo que l'on trouve principalement dans les ordinateurs individuels. Ce signal vidéo est constitué de 5 signaux :
3 analogiques avec des niveaux de 0.7 à 1.0 volt pour chacune des couleurs Rouge, Vert et Bleu (RGB) accompagnés de 2 signaux logiques que sont les synchros horizontale et verticale.

1.1 Le moniteur de visualisation

Le principal composant du moniteur est le tube cathodique dont l'écran est la face visible pour l'utilisateur. Le faisceau d'électron est dévié selon un parcours horizontal de gauche à droite de l'écran, puis retour à la ligne avec décalage vers le bas et ceci jusqu'à atteindre le point le plus bas à droite. Dans ce parcours, l'information RGB est utilisée pour contrôler l'intensité du faisceau d'électrons. Cela donnera selon le cas un point de couleur rouge, bleu ou vert (couleurs primaires) plus ou moins intenses, l'œil humain percevant une couleur résultante de ces trois points très proches physiquement.

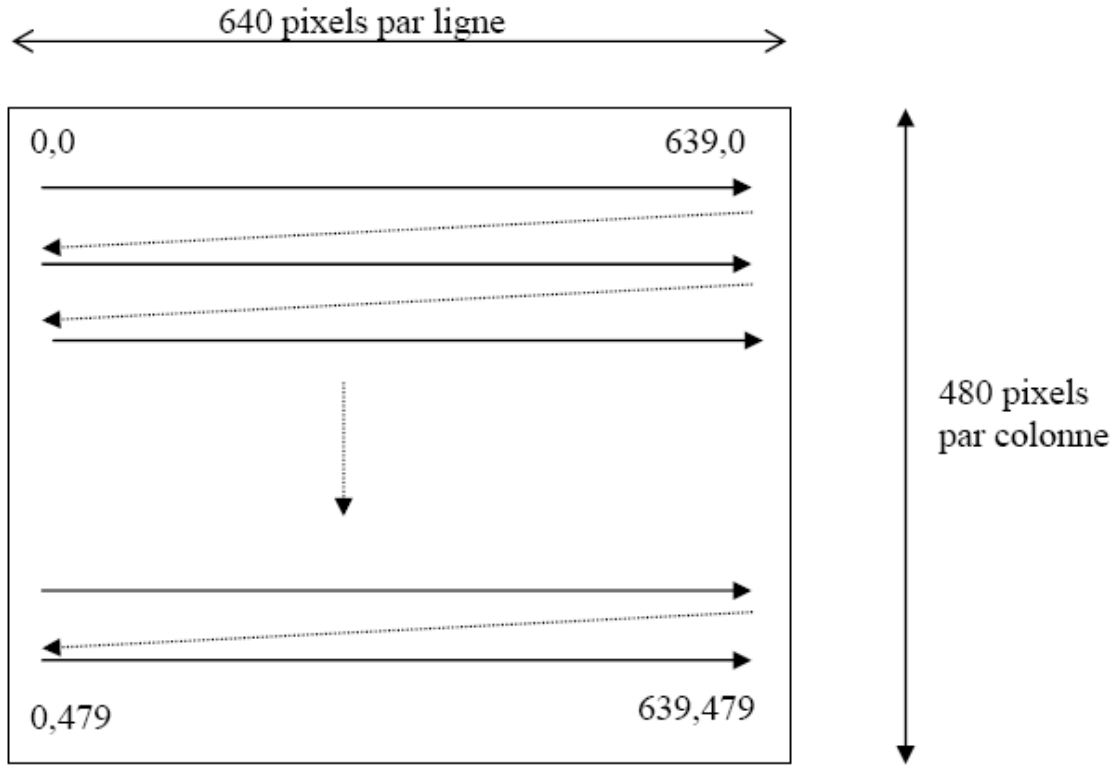


Figure 1 :Image 640 X 480

En VGA standard, l'écran contient 640 par 480 **pixels** (picture elements) qui doivent être balayés à une fréquence supérieure à 30 Hz si l'on veut éviter un effet de scintillement. On trouve en général sur les PC une fréquence de rafraîchissement de 60 Hz (60 images par seconde). Les moniteurs du commerce fonctionnent tous dans une certaine plage autour de cette fréquence de référence.

Il existe plusieurs standards VGA mais le plus utilisé est le « VGA industry standard », en voici les caractéristiques :

General characteristics

```
Clock frequency 25.175 MHz
Line  frequency 31469 Hz
Field frequency 59.94 Hz
```

One line

```
  8 pixels front porch
 96 pixels horizontal sync
 40 pixels back porch
  8 pixels left border
640 pixels video
  8 pixels right border
---
800 pixels total per line
```

One field

```
  2 lines front porch
  2 lines vertical sync
 25 lines back porch
  8 lines top border
480 lines video
  8 lines bottom border
---
525 lines total per field
```

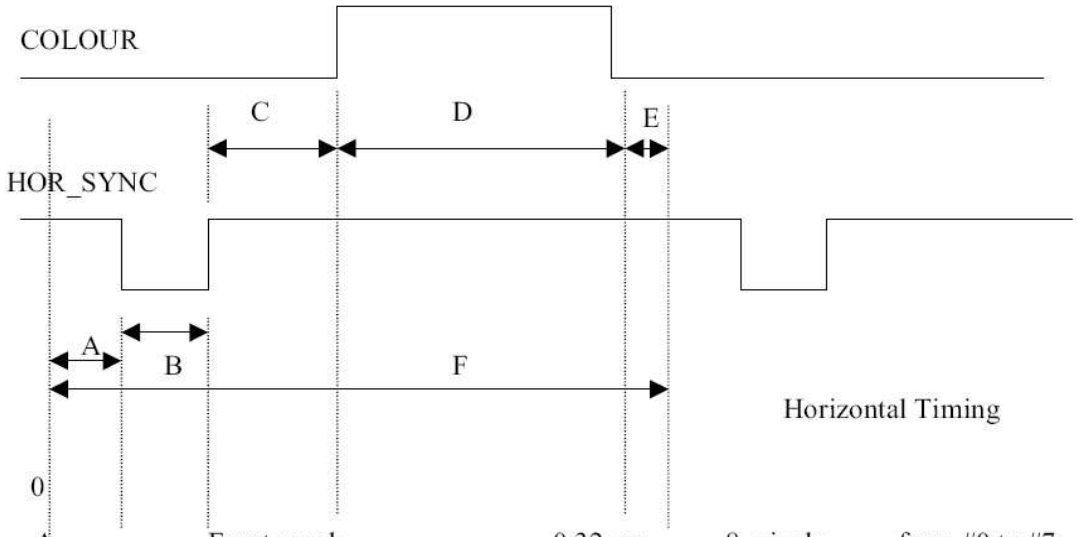
Other details

```
Sync polarity: H negative, V negative
Scan type: non interlaced.
```

D'après les caractéristiques ci-dessus, Nous pouvons relever 5 valeurs importantes :

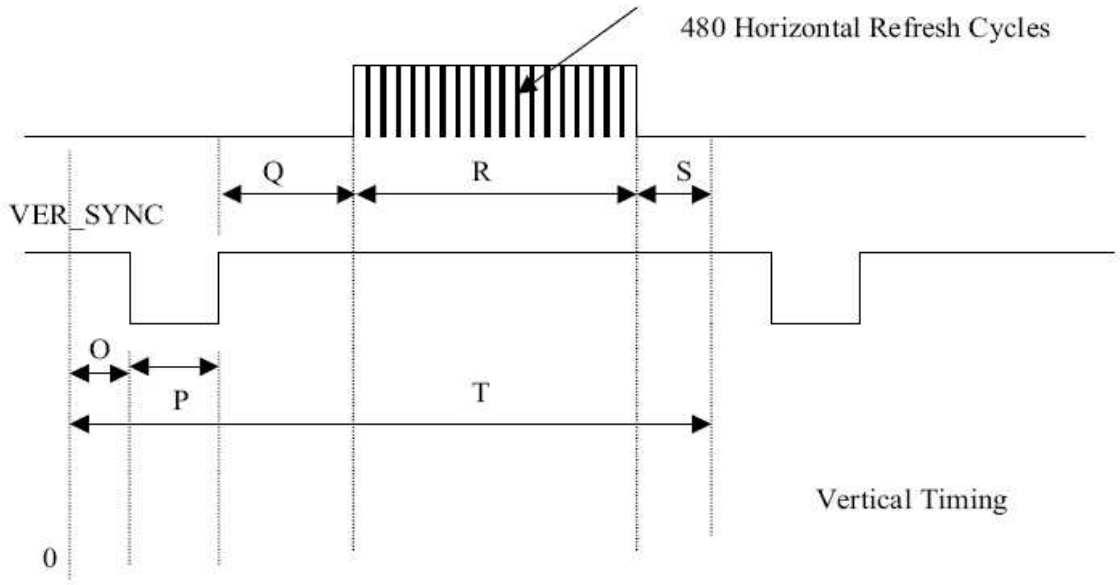
- La fréquence pixels (Clock Frequency) = 25,175 MHz
- La fréquence ligne (Line Frequency) = 31469 Hz
- La fréquence image (Field Frequency) = 59,94 Hz
- Le nombre de pixels par ligne (pixels total per line) = 800
- Le nombre de ligne par image (lines total per field) = 525

La trame VGA est découpée en 525 lignes de 800 pixels et seulement 480 lignes de 640 pixels sont visibles à l'écran. Le reste des lignes et des pixels sont utilisés pour la synchronisation comme le montre les figures 2 et 3. Le signal HOR_SYNC pour la synchronisation horizontale et le signal VER_SYNC pour la synchronisation verticale.



A	Front porch	0.32 us	8 pixels	from #0 to #7
B	Sync. pulse	3.81 us	96 pixels	from #8 to #103
C	Back porch	1.90 us	48 pixels	from #104 to #151
D	Active video time	25.42 us	640 pixels	from #152 to #791
E	Right border	0.32 us	8 pixels	from #792 to #799
F	Scanline time	31.77 us	800 pixels	

Figure 2 : signal de synchronisation horizontale



O	Front porch	62.5 us	2 lines	from #0 to #1
P	Sync. pulse	62.5 us	2 lines	from #2 to #3
Q	Back porch	1.05 ms	33 lines	from #4 to #36
R	Video	15.25 ms	480 lines	from #37 to #516
S	Bottom border	0.25 ms	8 lines	from #517 to #524
T	Whole picture	16.68 ms	525 lines	

Figure 3 : signal de synchronisation verticale

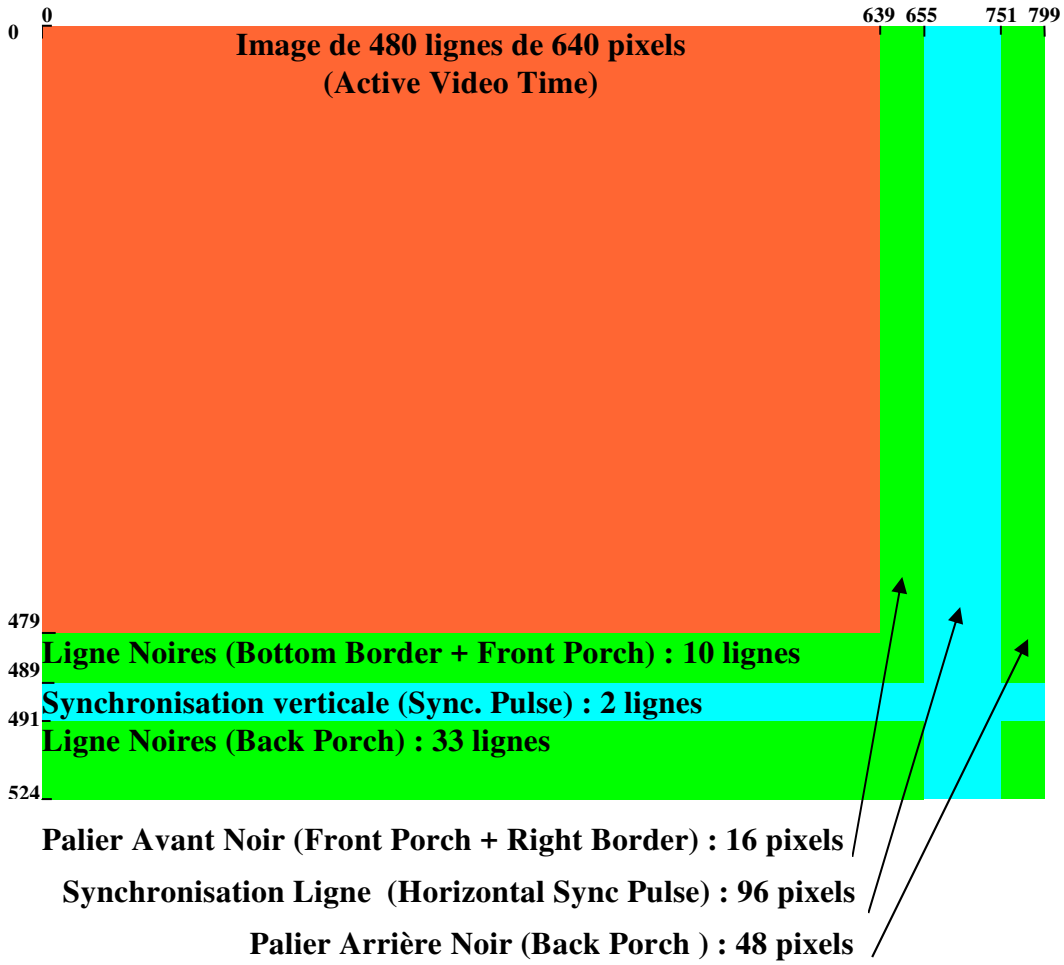


Figure 4 : trame d'une image avec une horloge de 25,125 MHz

1.2 Le connecteur

Il permet de relier le système générateur de vidéo (le plus souvent l'ordinateur) au moniteur

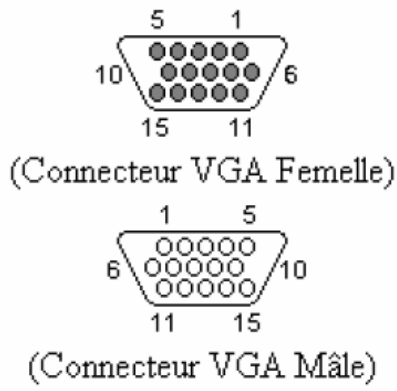


Figure 6 : Connecteurs VGA

Les signaux véhiculés sont ceux du tableau ci-après :

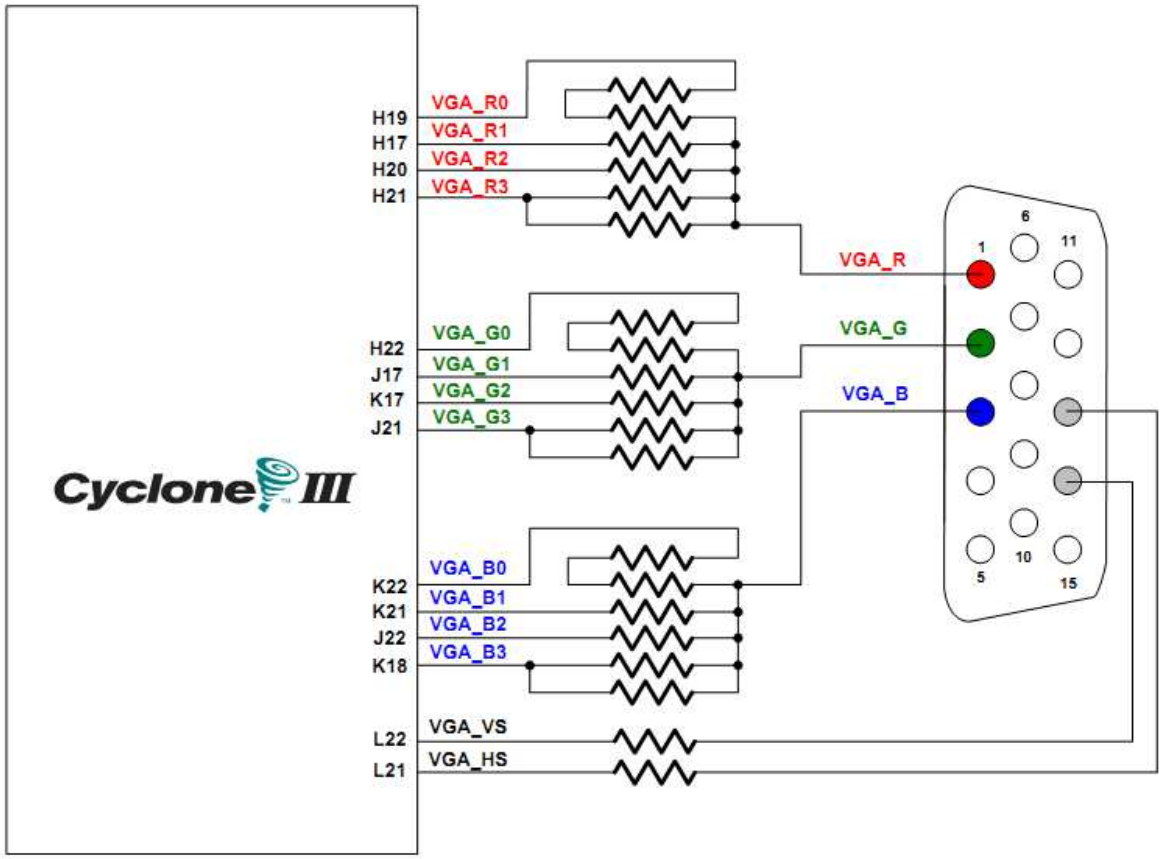
Broche	Nom	Direction	Description
1	RED	→	Vidéo rouge (75 ohms, 0.7 volt)
2	GREEN	→	Vidéo verte (75 ohms, 0.7 volt)
3	BLUE	→	Vidéo bleu (75 ohms, 0.7 volt)
4	RES		Réservé
5	GND	—	Masse
6	RGND	—	Masse rouge
7	GGND	—	Masse verte
8	BGND	—	Masse bleu
9	+5V	→	+5V continu
10	SGND	—	Masse synchro
11	ID0	←	Monitor ID Bit 0 (optionnel)
12	SDA	↔	Ligne de données série DDC
13	HSYNC ou CSYNC	→	Synchro horizontale ou composite
14	VSYNC	→	Synchro verticale
15	SCL	↔	Ligne d'horloge DDC (Display Data Channel)

1.3 La carte DE0

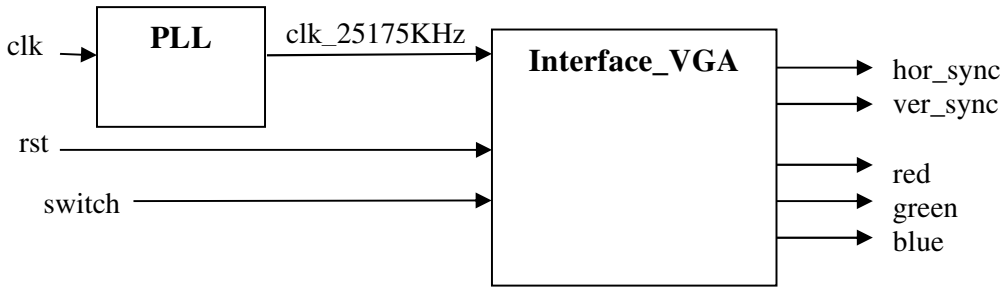
La carte DE0 est cadencée par un quartz de 50 MHz, on utilisera une PLL[1] du FPGA cyclone III pour convertir la fréquence de 50 MHz en une fréquence de 25,175 MHz nécessaire pour l'affichage VGA.

La carte DE0 contient un convertisseur analogique/numérique basique composé d'un réseau de résistances R/2R. De ce fait, les signaux RGB sont codés sur 4 bit est la palette des couleurs se limite aux 4096 (2^{12}) couleurs primaire.

[1] PLL signifie Phase Lock Loop, il s'agit d'une fonction très utilisée en électronique pour faire des opérations sur des fréquences (divisions, multiplications, modulations, etc…)


H19
H17
H20
H21
VGA_R0
VGA_R1
VGA_R2
VGA_R3
VGA_R
H22
J17
K17
J21
VGA_G0
VGA_G1
VGA_G2
VGA_G3
VGA_G
VGA_B
K22
K21
J22
K18
VGA_B0
VGA_B1
VGA_B2
VGA_B3
L22
L21
VGA_VS
VGA_HS
Cyclone III
1
6
11
5
10
15

2 Génération du signal vidéo



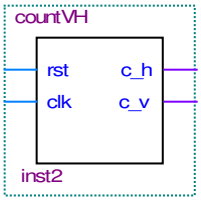
2.1 Génération des signaux de synchronisation

Le but de cette partie est de générer les signaux de synchronisation VER_SYN et HOR_SYN nécessaire pour piloter l'écran VGA. On vous fournit le composant PLL qui permet de diviser l'horloge de 50 Mhz afin d'obtenir l'horloge de 25,175 Mhz nécessaire pour générer les signaux de synchro. Cette PLL existe en dur dans le FPGA, il suffit de la configurer. Pour cela on a utilisé l'outil Megawizard de Quartus II qui nous fournit une IP synthétisable de la PLL (voir **src/pll_vga.vhd**).

2.1.1 Compteur de pixels et compteur de lignes

Deux compteurs sont nécessaires.

- Un pour compter les pixels (c_h) sur chaque ligne de 0 à 799. Ce compteur s'incrémente avec l'horloge de 25,175 MHz.
- Un pour compter les lignes (c_v) de 0 à 524. Ce compteur s'incrémente à chaque fois que le compteur pixels passe de 799 à 0.

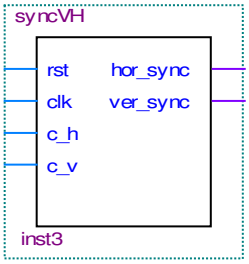


- Copier le fichier **TP2_Affichage_VGA_DE0.zip**
- Dézipper ce fichier.
- Ouvrir le fichier **src/countVH.vhd**
- Compléter le code et simuler-le.

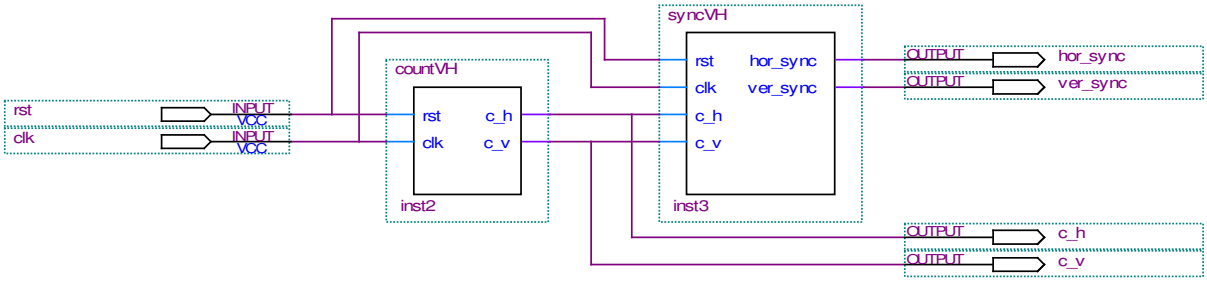
2.1.2 Signaux de synchronisations verticale et horizontale

Objectifs :

- Utiliser le composant précédemment décrit afin de générer les signaux HOR_SYNC et VER_SYNC.
- Tester les valeurs de c_h et c_v afin de générer les signaux de synchronisations horizontal et vertical comme spécifié sur les figures 2, 3 et 4.



- Ouvrir le fichier **src/syncVH.vhd**
- Compléter le code.
- Instancier les composant syncVH et countVH dans le fichier **SyncGen.vhd** comme sur le schéma ci-dessous, n'oubliez pas de déclarer les signaux internes c_h et v_h :



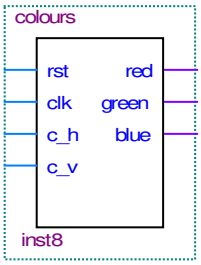
- Ouvrir est compléter le fichier **src/SyncGen.vhd**
- Simuler le composant **SyncGen.vhd** avec le script de simulation **syncsimu.do** et le banc de test **syncgen_tb.vhd** que vous trouverez dans le répertoire **simu**.
- Vérifiez la simulation. Si il n'y a pas d'erreur, vous devriez voir écrit ******Synchronisation Timing is Alright*******.
- Faites vérifier par l'enseignant.

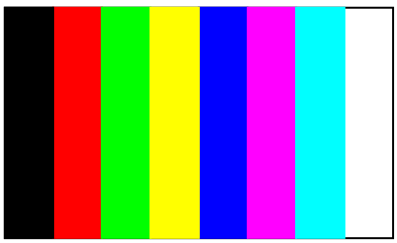
2.2 Génération d'une mire de couleur horizontale et verticale

Le but de cette partie est de générer les signaux de couleurs RGB (Red, Green, Blue) qui permettent d'afficher des images, du texte ou des formes. En général les signaux RGB sont sur plusieurs bits et convertis en signaux analogiques par un CAN (Convertisseur Numérique Analogique). Sur notre carte il n'y a pas de CAN et les signaux RGB sont codés sur 1 bit. Nous pouvons donc générer seulement 8 couleurs.

2.2.1 Mire vidéo horizontale

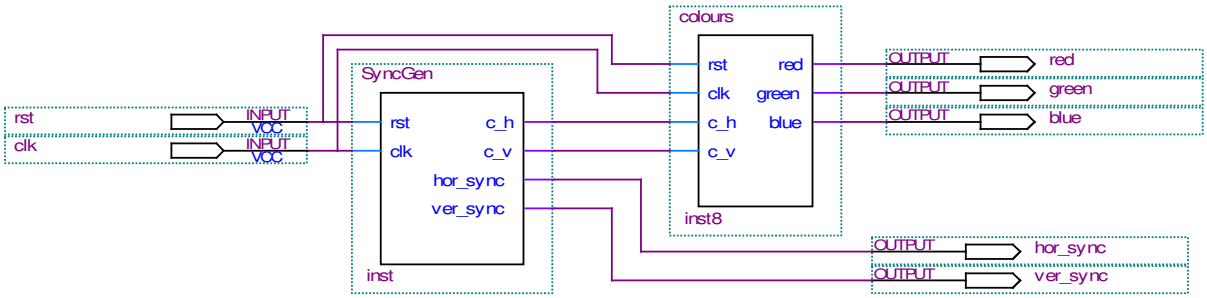
Vous allez afficher une mire horizontale de 8 couleurs sur le moniteur. Pour cela il faut tester le compteur de pixels (c_h) entre deux valeurs et choisir une valeur pour les signaux RGB. Par exemple : si c_h est entre 0 et 80 pixels on veut une couleur rouge (red <= x"1111" , green <= "0000" et blue <="0000").





Ouvrir et compléter l'architecture mire du fichier **/src/colours.vhd** afin de générer une mire horizontale de 8 couleurs.
Attention, pendant le palier avant noir (Front Porch), palier arrière noir (Back Porch) il faut que les couleurs RGB soient à zéro pour que l'affichage fonctionne correctement.

Instancier les composants **SyncGen** et **colours** dans le fichier **interface_VGA.vhd** comme le montre le schéma ci-dessous (**Attention de bien spécifier l'architecture « mire » de l'entité « colours » lors de l'instanciation de colours**) :



- Faire la synthèse et programmer la carte FPGA.
- Le fichier **top_level.vhd** est déjà complet.

Synthèse et programmation de la carte.

Créez un projet sur Quartus dans le répertoire **fit**, nommez le **TP2_VGA** sélectionnez le FPGA relatif à la carte de développement que vous utilisez (voir le « User Manuel » de cette carte). Rajouter vos sources en sélectionnant les sources qui sont dans le répertoire **src** (sans les copier dans le répertoire fit).
Sélectionnez le composant **top_level** comme entité de plus haute hiérarchie (top-level). Pour cela, sélectionnez le « File » **top_level.vhd** dans le Project Navigator, faites un clic droit avec la souris et sélectionnez « Set as top-level Entity ».
Maintenant il faut faire l'assignement des pins d'entrées/sorties :
Lancer une première fois la synthèse de votre projet (**Processing -> Start compilation**).
Ensuite assignez les pins du FPGA depuis le menu **Assignements → Pin Planner** en vous reportant au manuel de la carte pour choisir les bons numéros de pin sur la colonne **Location.**
Après l'assignement des pins, il faut relancer la synthèse de votre projet.

Programmation du FPGA :

- ➢ Connectez la carte à votre ordinateur et connecter un écran VGA sur la sortie VGA de la carte.
- ➢ Ouvrez le Programmateur qui se trouve dans Tools → Programmer.
- ➢ Connecter la carte de développement avec le cable USB (USB Blaster)
- ➢ En haut à gauche de cette fenêtre, en face de Hardware Setup, si il n'y a pas noté USB_Blaster, cliquez sur Hardware Setup et choisissez USB_Blaster et cliquez sur Close.
- ➢ Lorsque USB_Blaster est bien sélectionné, cliquez sur Start et normalement le bitstream (fichier .sof) se charge dans le FPGA.
- ➢ La carte doit alors afficher une mire de couleur sur l'écran VGA.

2.2.2 Mire vidéo verticale

Pour la mire vidéo verticale, au lieu de tester c_h il faut tester c_v. Pour cela, il suffit de modifier le fichier **colours.vhd** (mettre la génération de la mire horizontale en commentaire).

Faites les modifications dans **colours.vhd**.

Faites la synthèse et tester sur la carte DE0.

2.2.3 Mire vidéo verticale et horizontale

On veut pouvoir choisir entre la mire verticale et la mire horizontale à l'aide du bouton poussoir « SW0 ». Pour cela modifier l'architecture mire du fichier **colours.vhd** afin de prendre en compte ce nouveau paramètre.

Faites les modifications dans **colours.vhd**.

Faites la synthèse et programmer la carte DE0.

Faites vérifier par l'enseignant.

- Relever le nombre de Logic Element (LE) utilisé sur le FPGA.
- Relever également la fréquence maximum de fonctionnement de votre design.

Notez ces 2 valeurs (LE et FMAX) au tableau en face de vos noms.

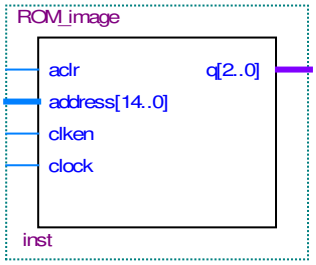
2.3 Affichage d'une image

On désire maintenant afficher une image sur l'écran VGA. Pour cela nous allons utiliser les blocs mémoire qui sont dans le FPGA (M9K RAM blocks) et un logiciel basique de dessin (Microsoft Paint).

Si nous voulons afficher une image de 640 pixels * 480 lignes * 12 bits de couleurs (RGB), il nous faut une mémoire de 3 686 400 Bits. Le FPGA EP3C16 que nous utilisons sur la carte DE0 ne contient que 516 096 Bits de mémoire. Pour cette raison, nous sommes obligés de faire un fenêtrage et donc afficher l'image que sur une partie de l'écran. Nous avons choisit de créer une image de 160 pixels * 120 lignes * 3 bits de couleur (RGB). Ce qui nous donne un total de 57600 bits et qui ne dépasse pas la capacité mémoire du FPGA.

Si nous voulons afficher une image de 640 pixels * 480 lignes il nous faudrait rajouter une mémoire externe au FPGA.

Avec l'outil Megawizard du logiciel Quartus II, nous avons générer une mémoire de type ROM qui sera initialisé avec un fichier **display.mif** que vous allez créer avec Microsoft Paint. Voici ci-dessous l'entité de la ROM ainsi créé. Vous avez une explication du fonctionnement de cette ROM dans le dossier docs.



2.3.1 Création de l'image

Tout d'abord, il faut dessiner l'image que vous voulez afficher sur l'écran VGA. Créer cette image avec le logiciel Paint de Microsoft Windows de la manière suivante :

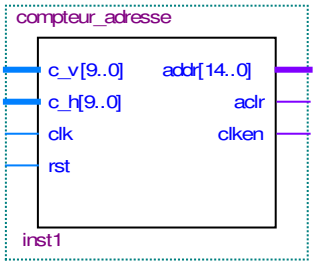
- Ouvrir Paint qui se trouve typiquement dans le menu **Démarrer → Tous les programmes → Accessoires → Paint.**
- Choisissez dans le menu principal **Propriétés**. Dans la boite de dialogue, mettez 160 pour la largeur et 120 pour la hauteur. Choisir Pixels comme unités.
- Faites un dessin de votre choix. Utiliser les couleurs basiques tels que celle que l'on retrouve dans la mire.
- Une fois que le dessin est terminé, appuyez sur Ctrl+A pour sélectionner l'image entière, faite un clic droit sur l'image et sélectionnez Faire pivoter → Retourner verticalement.
- Enregistrer le dessin en choisissant le format **Bitmap 24 bits (.bmp)** et nommez-le **image.bmp** par exemple.
- Récupérer l'exécutable **bmp2filegeneral.exe** dans le répertoire **Soft**.
- Placez cet exécutable dans le même dossier que le fichier **image.bmp**. Lancer une fenêtre DOS (**Démarrer →Rechercher des programmes et fichiers..** et lancer le programme **cmd**). Dans cette fenêtre, écrire la commande suivante : **bmp2mifgeneral image.bmp.** Ceci va créer deux fichiers : **image.colour.mif** et **image.mono.mif**. Renommer le fichier **image.colour.mif** en **display.mif** et déplacer-le dans le dossier **fit**.

2.3.2 Compteur d'adresse

Pour générer l'image, il suffit de décrire un compteur d'adresse qui va incrémenter l'adresse en fonction de c_v et c_h. En fonction de la position du pixel dans l'image on va incrémenter le compteur ou non.

Si on veut afficher l'image en haut à gauche de l'écran, il faut tester que c_h soit inférieur à 160 et que c_v soit inférieur à 120 pour que l'adresse s'incrémente. Pour être sur d'être synchrone, il faut remettre l'adresse à 0 après les 120 lignes.

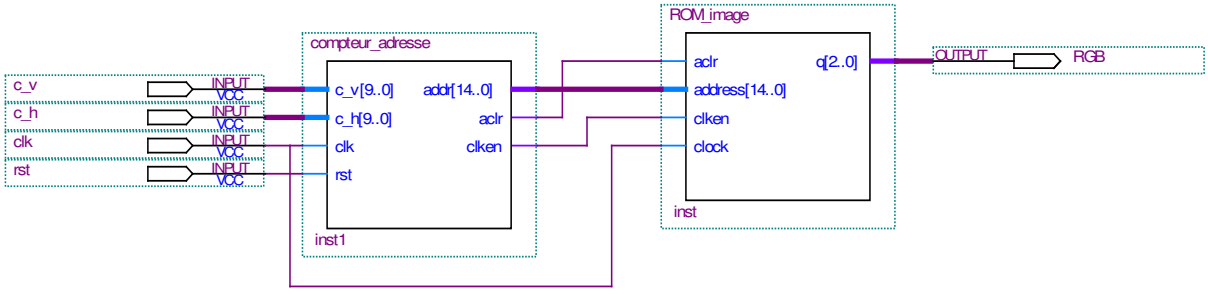
Lorsqu'on est en dehors de la fenêtre de l'image, il faut utiliser l'entrée **aclr** de la ROM pour mettre à zéro les signaux RGB.



- Ouvrir et compléter le fichier **compteur_adresse.vhd**

2.3.3 Assemblage

Pour tester l'affichage d'une image il faut instancier l'entité **compteur_adresse** et **ROM_image** dans l'architecture **struct** du fichier **colours.vhd** comme le montre le schéma ci-dessous. Attention ! Il ne faut pas oublier de changer le nom de l'architecture (struct) dans l'instanciation de **colours** dans le fichier **interface_VGA.vhd.**



- Ouvrir le fichier **colours.vhd** et compléter l'architecture **struct**. Ne pas oublier de changer l'instanciation de **colours** dans **interface_VGA.vhd**
- Faire la synthèse et programmer la carte DE0.

- Faites vérifier par l'enseignant.

2.3.4 Fenêtrage

Rendre le composant **compteur_adresse** générique en créant quatres paramètres génériques entier Ppixel, Pligne, Npixel et Nligne. Ppixel et Pligne représentant la position du pixel et la position de la ligne du coin haut gauche de l'image. Npixels et Nlignes représentant le nombre de pixel et le nombre de ligne de l'image.

- Modifier le fichier **compteur_adresse.vhd**
- Faire la synthèse et programmer la carte DE0.

- Faites vérifier par l'enseignant.

2.3.5 Commande de la position de l'image depuis les switch

Au lieu de commander la position de l'image avec des paramètres génériques, on désire maintenant commander la position de l'image à l'aide des switch de la carte DE0.

Par exemple les switch SW0 à SW4 commandent la position horizontale et les switch SW5 à SW9 la position verticale.

Modifier le code des différents fichiers impactés par cette modification afin de réaliser cette nouvelle fonctionnalité.

Faire la synthèse et programmer la carte DE0.

Faites vérifier par l'enseignant.

2.3.6 Adaptateur VGA

On désire à présent afficher l'image sur la totalité de l'écran VGA. Pour cela il faut que chaque case de la mémoire soit représentée par un **boxsel** de 4 x 4 pixels. Ainsi cela revient à étirer l'image sur la totalité de l'écran VGA.

- 4 x 160 pixels = 640 pixels
- 4 x 120 pixels = 480 pixels

Modifier le fichier **compteur_adresse.vhd** afin de réaliser cette nouvelle fonctionnalité.

Faire la synthèse et programmer la carte DE0.

Faites vérifier par l'enseignant.