

## Télécran (Etch a sketch en V.O.)

Il est conseillé de lire l'ensemble de l'introduction avant de se lancer dans les questions. Les 3 parties du problème sont indépendantes et peuvent se traiter dans n'importe quel ordre. Au sein de la partie A, les questions A5 à A8 ne dépendent pas des questions A1 à A4.

Toutes les réalisations séquentielles doivent être synchrones.

On vous demande de numéroté **très clairement** les questions et les parties.

Barème (pour information, susceptible d'être modifié) :

<b>Partie A</b>	<b>15 points</b>
Partie B	2 points
Partie C	3 points
Partie D	5 points

## Présentation du projet

Qui a dit que pour devenir ingénieur, il fallait perdre son âme d'enfant ? Personne de l'ENSEA, en tout cas ! Nous allons donc dans cet examen faire ressortir l'enfant qui sommeille en nous (pas très loin, dans bien des cas) pour fabriquer un télécran numérique. Pour cela, nous utiliserons quatre boutons poussoirs, un convertisseur numérique / analogique basique, un écran et, of course, un composant logique programmable de la famille des FPGA (dont la connaissance intime n'est pas indispensable pour réussir l'examen).

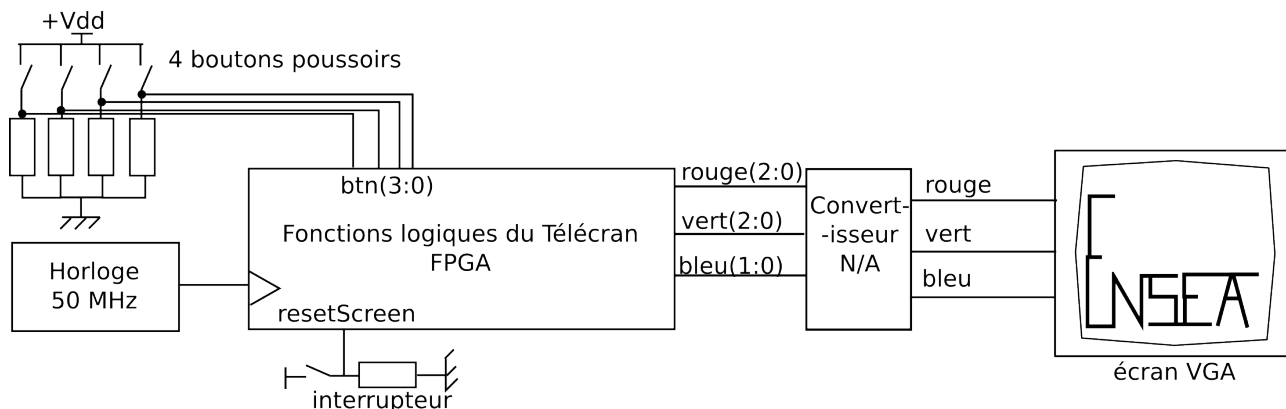


Figure 1: Le projet dans son environnement.

**Qu'est-ce qu'un télécran ?** C'est un jouet servant à tracer des dessins sur un écran. Le

dessin est constitué d'une unique ligne noire continue horizontale ou verticale, éventuellement à 45 degré si l'on est un peu habile. L'originale est mécanique (un stylet ne pouvant pas être remonté vient écrire sur un écran), le notre sera évidemment numérique.

Le comportement du système est le suivant : à l'allumage, l'écran affiche une couleur uniforme (fond d'écran), un pointeur est situé au centre.

L'appui bref sur l'un des boutons (btn(3:0)) « haut », « bas », « gauche », « droite » fait avancer le pointeur dans la direction indiquée d'un seul pixel. Le pointeur laisse une trace noire derrière lui.

Un appui modérément long provoque une mise en mouvement modérément rapide. Un appui très long augmente la vitesse de déplacement.

L'activation du bouton de reset remet le système dans son état d'origine.

Pour réaliser cette fonction, on décompose le projet en trois blocs fonctionnels : gestion des boutons, gestion du pointeur, gestion de l'écran. La figure 2 ci-dessous indique cette décomposition fonctionnelle et les signaux intermédiaires générés.

Compte tenu du temps imparti à l'épreuve, tous les aspects ne seront pas abordés, mais l'ensemble fournira un sujet de mini-projet aux étudiants de 1A.

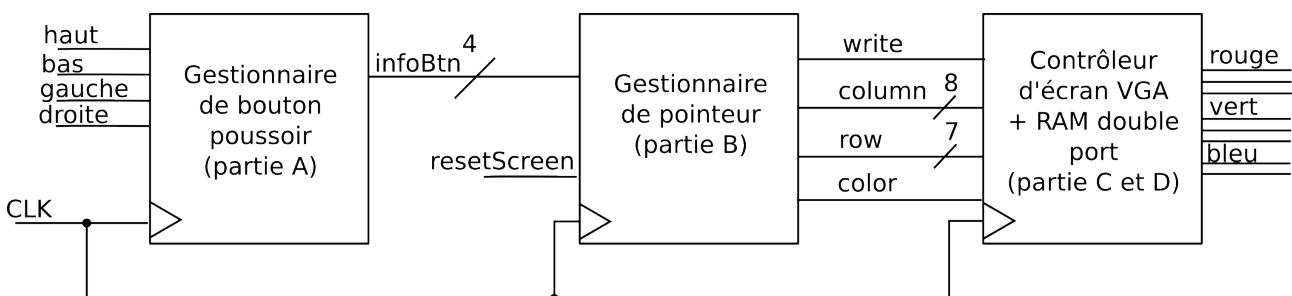


Figure 2: Décomposition fonctionnelle du projet.

## Partie A : étude de la gestion des boutons (1A et 1B : 14 points).

Nous commençons par le plus facile... Ce bloc fonctionnel est constitué de 4 blocs identiques transformant pour chacun des quatre interrupteurs le signal d'entrée en un signal de sortie exploitable par le gestionnaire de pointeur (partie B). Pour la partie A, on se concentrera sur la gestion de l'interrupteur « haut », sachant qu'il faudra quatre séquenceurs identiques pour chacun des quatre boutons.

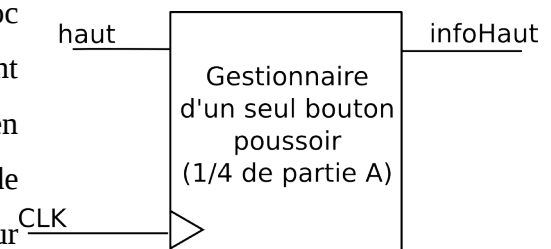


Figure 3: Entrée / sortie du séquenceur de gestion de BP (Partie A).

L'évolution du signal de sortie se fait de la manière suivante :

- Lors d'un appui court (inférieur à 200 ms), on génère un « top » sur le signal

« infoBtn » (un « top » signifie que le signal passe à « 1 » durant un front uniquement ou pendant une période d'horloge complète). L'état non appuyé se nomme par la suite « released ».

- Lors d'un appui modérément long (inférieur à 1 seconde), on génère un top toutes les 200 ms. Cet état se nomme par la suite « short ».
- Lors d'un appui long (supérieur à 1 seconde), on génère un top toutes les 100 ms. L'état se nomme « long ».
- A l'issue d'un appui, quelle que soit sa durée, on rentre dans un état anti-rebond (nommé « Bounce ») qui dure 100 ms.

L'architecture interne de ce bloc utilise un compteur pour générer les temporisations. On la précise en figure 4.

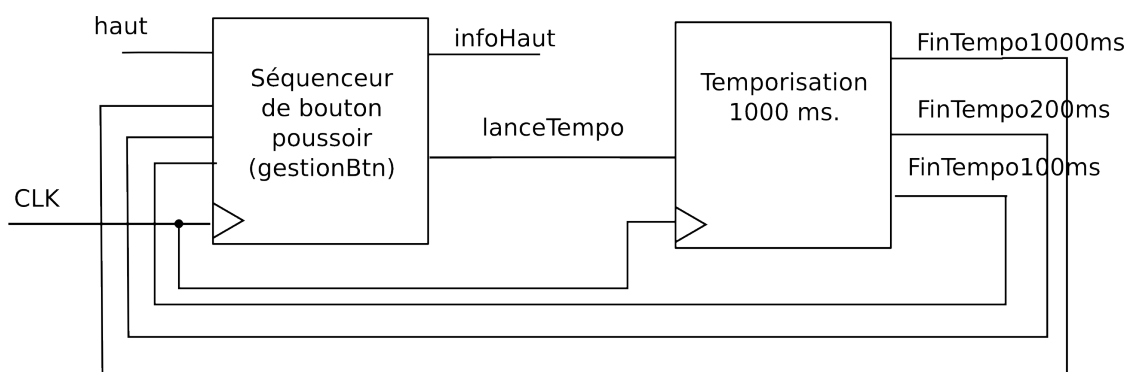


Figure 4: Gestion du bouton poussoir "haut". Interconnexion avec le compteur

Enfin, afin que la description du séquenceur soit sans équivoque, on vous fournit le code

VHDL correspondant (*attention, ce code se poursuit sur la page 4*).

```

1 architecture Behavioral of gestionBtn is
2 type etat is (released,short,long,bounce);
3 signal present, futur : etat;
4 begin
5     process (clk)
6     begin
7         if (clk'event and clk='1') then
8             present <= futur;
9         end if;
10    end process;
11
12    process (present, btn, fin100, fin200, fin1000)
13    begin
14        case present is
15            when released =>
16                if btn = '1' then futur <= short;
17                else futur <= released;
18                end if;
19            when short =>
20                if btn = '0' then futur <= bounce;
21                elsif fin100='1' then futur <= long;
22                else futur <= short;

```

```

23         end if;
24     when long =>
25         if btn = '0' then futur <= bounce;
26         else futur <= long;
27         end if;
28     when bounce =>
29         if fin100='1' then futur <= released;
30         else futur <= bounce;
31         end if;
32 end case;
33 end process;
34
35 infoBtn <= '1' when (present=released and btn='1') or
36 (present=short and fin200='1') or (present=long and fin100='1') else
37 '0';
38
39 lanceTempo <= '1' when (present=released and btn='1') or (present=short
40 and btn='0') or (present=long and btn='0') else '0';

```

**Question A-1 :** à l'aide de la description VHDL et en vous aidant éventuellement de la description textuelle de la page 2, tracez le diagramme d'état du séquenceur de gestion de bouton.

On vous propose le codage suivant pour les états :

Etat	$Q_1Q_0$
Released	« 00 »
Short	« 01 »
Long	« 10 »
Bounce	« 11 »

**Question A-2 :** *sur le document-réponse A-2*, dessinez l'architecture du séquenceur gestionBtn en faisant apparaître deux blocs combinatoires et les bascules. *Sur le document réponse A-2*, encadrez les lignes du codes VHDL décrivant la fonctionnalité de chacun des trois blocs (deux combinatoires et un bloc de bascule).

**Question A-3 :** déterminez par la méthode de votre choix les équations de vos deux blocs combinatoires.

**Question A-4 :** dessinez une réalisation de l'équation de D1 en transistors MOS. On cherchera à minimiser le nombre de transistors.

Nous nous intéressons désormais au compteur générant les signaux fin100, fin200 et fin1000. On rappelle que l'horloge du système est à 50 MHz. Le compteur a un cycle complet qui dure une seconde. La fonctionnalité complète est représentée page suivante.

Pour que la sortie fin100 soit active *toutes les 100 millisecondes* pendant une période d'horloge (à 50 MHz), on la génère à l'aide d'un compteur, qui va entraîner à la fin de chaque cycle (sortie de type « ripple » ou « recyclage » ou « retenue ») l'avancé d'un autre compteur. Le second

compteur est modulo 2, afin de générer le signal *fin200*. Le signal *fin200* fait lui même office d'*enable* pour un troisième compteur, modulo 5, qui permet de marquer une seconde ( $5 \times 200 \text{ ms} = 1\text{s}$ ).

Chaque compteur possède une entrée d'enable et une entrée de reset.

Le détail de l'architecture de cette sous partie est donnée sur la figure 5.

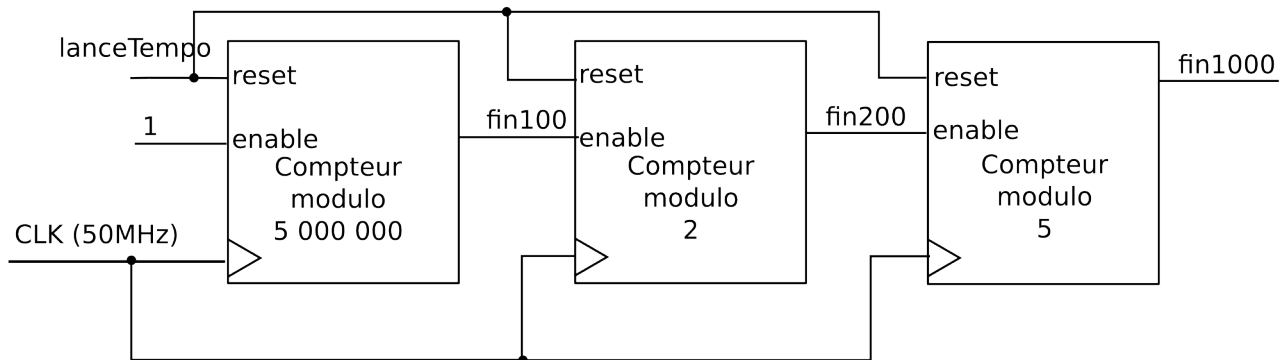


Figure 5: Fonctionnalité de temporisation de la partie A.

**Question A-5 :** combien de bascules sont nécessaires à la synthèse de cette fonctionnalité ?

**Question A-6 :** déterminez complètement l'architecture interne du compteur modulo 2<sup>1</sup>. Ce compteur compte de 0 à 1, avec une entrée d'enable, une entrée de reset, et une sortie de type « ripple » vraie lorsque le compteur est à 1 et que enable vaut 1.

**Question A-7 :** sur le document réponse A-7 (*attention, ce document réponse est différent en fonction de votre promotion d'origine : 1A ou 1B*), tracez l'implantation dans un FPGA simplifié du compteur modulo 2.

## Partie B : étude du gestionnaire de pointeur (1A et 1B : 2 points).

Une image est constituée de 160 par 120 pixels (version peu gourmande en ressource par rapport à la résolution classique 640 par 480 – cf partie C). La RAM contiendra une image avec le codage suivant :

- Si dans la RAM un pixel est à « 0 », il s'affichera coloré (d'une couleur à définir, ce sera pour nous le fond).
- Si dans la RAM un pixel est à « 1 », il s'affichera noir (ce sera pour nous le trait).

Le schéma de la figure 6 indique les numéros de pixels.

<sup>1</sup> Avouez qu'on est plutôt sympa...

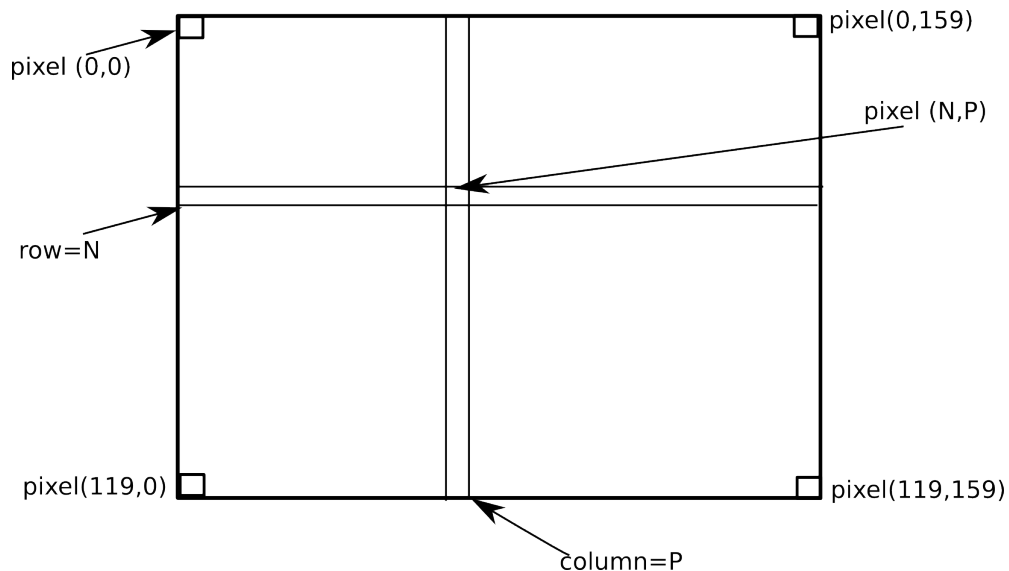


Figure 6: Numérotation des pixels dans une image VGA.

Pour tracer notre image, on va donc avoir deux pointeurs, *row* et *column* permettant de nous repérer dans l'image. Ils se codent respectivement sur 7 et 8 bits.

Le but de ce bloc est :

- si *reset Screen* est à '1' de lancer l'écriture de '0' sur la RAM pour chaque adresse possible de (*row*, *column*).
- si *reset Screen* est à '0', d'écrire un seul « 1 » à la position courante (*row*, *column*), et de faire évoluer (*row*, *column*) en fonction des informations du signal *infoBtn(3:0)*.

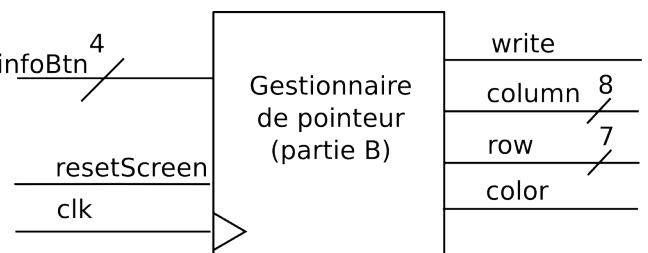


Figure 7: Entrées / sortie de la partie B

Dans le premier cas (*resetScreen*), les deux compteurs *row* et *column* balayent toutes les valeurs possibles de (*row*, *column*). Dans le second cas, *column* et *row* ne bougent que selon les informations venant du signal *infoBtn* (« haut » => *row*--, « bas » => *row*++...)

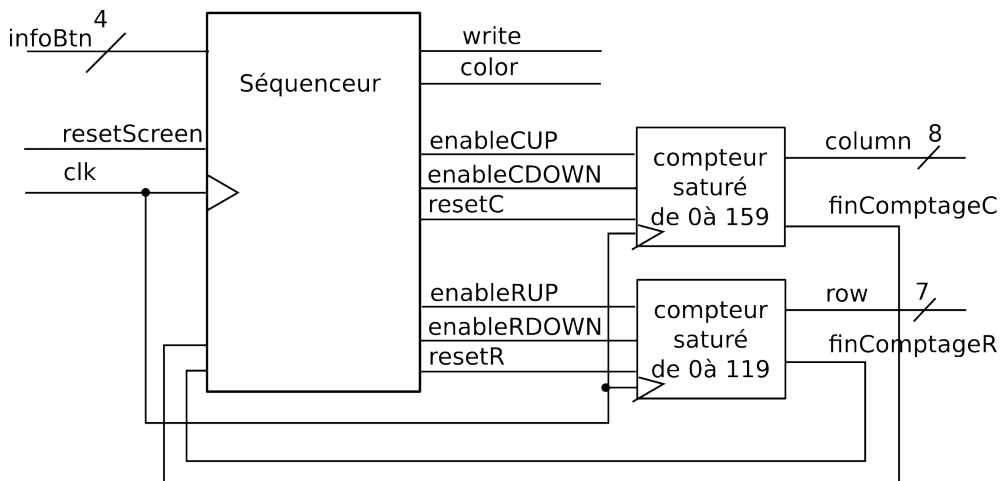


Figure 8: Détail du contenu de la partie B

Cette partie B se compose d'un séquenceur et de compteurs comme il est précisé sur la figure 8 page précédente.

Un compteur saturé est un compteur qui ne « recycle » pas. Par exemple, pour le compteur row, parvenu à 119, si un ordre enableRUP a lieu, le compteur reste à 119. La question porte sur ce compteur.

```

1 architecture Behavioral of compteurSat is
2 signal count : std_logic_vector (6 downto 0) := " 0000000 ";
3 begin
4     process (clk)
5     begin
6         if (clk'event and clk='1') then
7             if condition1
8                 then action1
9             elsif condition2
10                then action2
11             elsif condition3
12                then action3
13             end if;
14         end if;
15     end process ;
16
17     sortie <= count;
18
19 end Behavioral;

```

**Question B-1 :** écrivez les lignes VHDL correspondant à condition1, action1, condition2, action2, condition3 et action3. Si nécessaire, vous pouvez ajouter des tests.

## Partie C : étude de l'écran VGA (1A et 1B : 3 points).

Notre écran est compatible avec la norme VGA (Video Graphic Array), comme n'importe quel écran d'ordinateur. Cette norme est analogique et nécessite que la couleur de chaque pixel soit décomposée selon les couleurs primaires Rouge, Vert, Bleu.

Pour « fabriquer » le signal correspondant à la couleur à partir de trois bits (rgb(7:5)), voici le schéma mis en oeuvre (la résistance de 75 ohms correspond à la résistance d'entrée du moniteur, et n' est donc pas physiquement présente sur la carte numérique). Chaque bit à « 1 » impose une tension de 3.3 Volts.

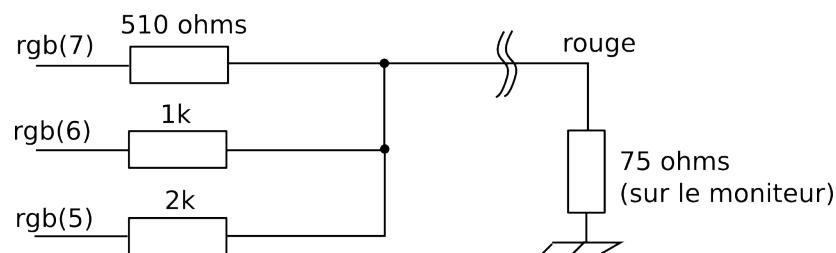


Figure 9: Réseaux R-2R pour une conversion numérique analogique

Le signal vert est fabriqué par un schéma totalement analogue.

**Question C-1 :** combien de niveaux peuvent prendre les signaux codant les couleurs rouge et verte ? Quelle est la tension maximale que l'on peut obtenir sur les signaux codant les couleurs rouge et verte ?

Lorsque la tension est au maximum calculé question C-1, la couleur est « à fond ».

Voici le schéma global donnant l'interconnexion du composant logique programmable avec l'écran. Le bleu a moins de bits réservés car l'œil humain est moins sensible à la couleur bleue.

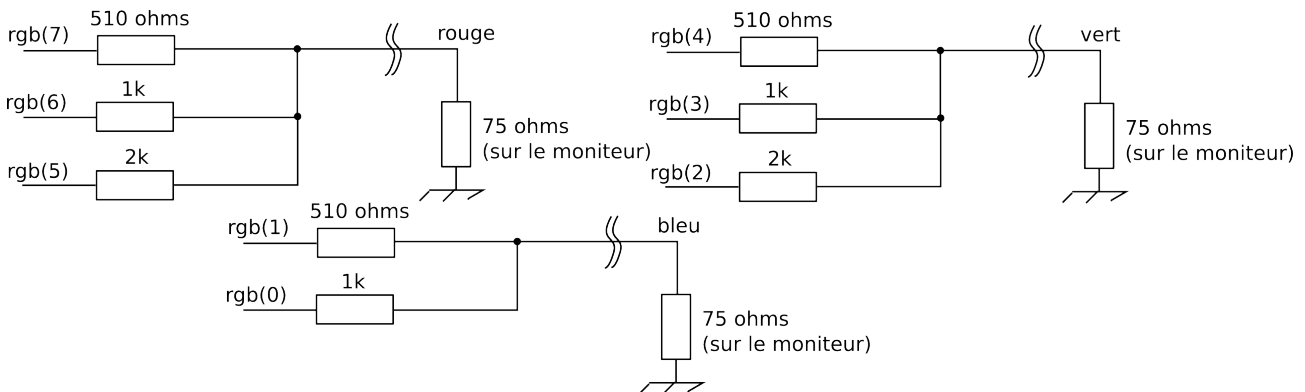


Figure 10: convertisseur R-2R complet pour les trois sorties RGB.

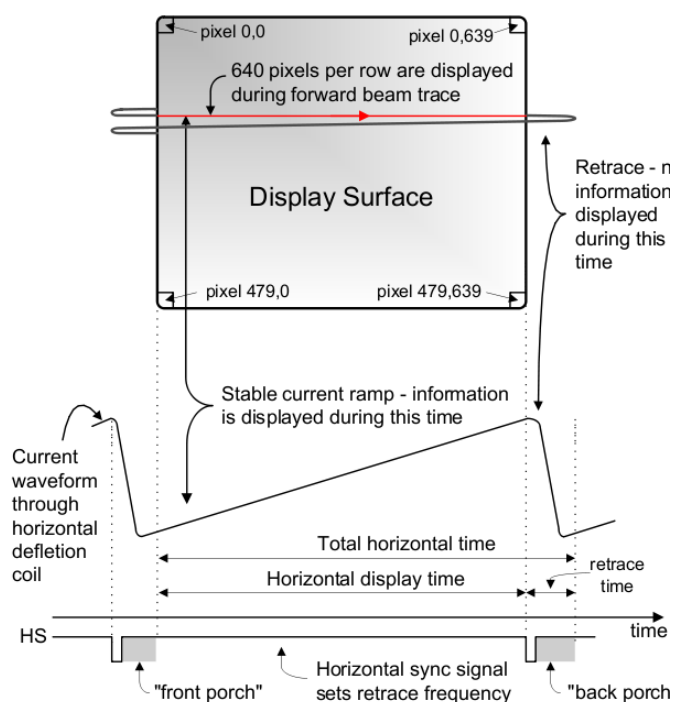
**Question C-2 :** combien de niveaux de couleurs peut-on générer avec notre composant logique programmable ?

## Etude de la synchronisation

La norme VGA définit en plus des trois signaux de couleurs deux signaux de synchronisation, l'un vertical et l'autre horizontal. En effet, une image est réalisée par un balayage constant d'une surface phosphorescente par un faisceau d'électrons. Les signaux de synchronisations correspondent au temps nécessaire au faisceau pour revenir soit en haut, arrivé à la dernière ligne (synchronisation verticale), soit à gauche, après le dernier pixel (synchronisation horizontale).

De plus, avant et après ces signaux se trouvent un « front porch » et un « back porch » correspondant à des temps pendant lesquels le faisceau est en dehors de l'écran. Selon la norme, il faut alors que l'information de couleur corresponde à noir (c'est à dire un faisceau à son intensité minimum).

Il est à noter que sur les écrans plus modernes (de type LCD), les phénomènes physiques



- 8 - Figure 11: Principe de la synchronisation VGA



en jeu ne sont plus du tout les mêmes, mais que, pour des raisons de compatibilité, les signaux restent identiques.

La figure 11, extraite du mode d'emploi de la Basys 2, reprend cette explication.

On s'intéresse dans l'examen uniquement au signal de synchronisation horizontale. Son chronogramme est indiqué sur la figure 12, ainsi que le recyclage du compteur qui génère ce signal.

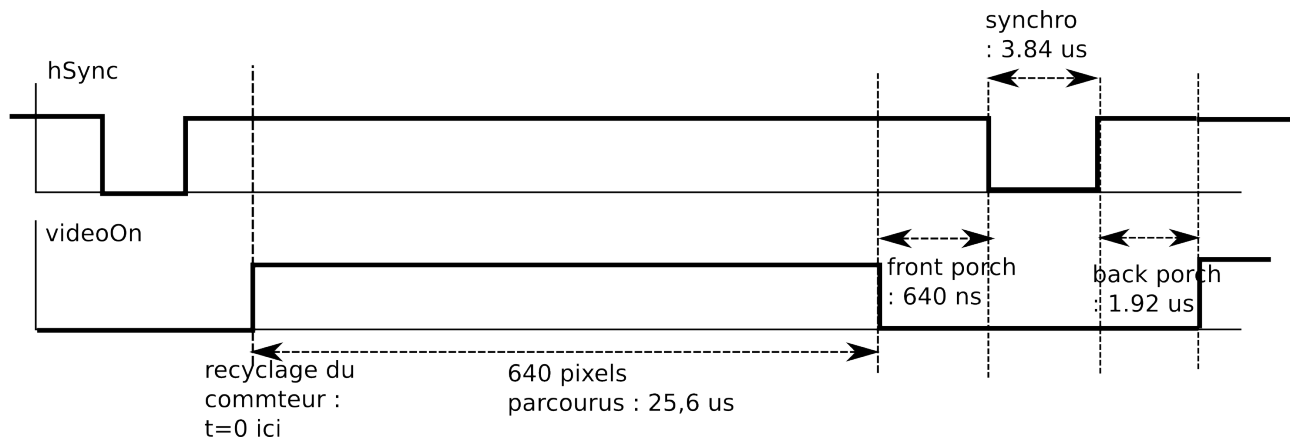


Figure 12: Timing de la synchronisation horizontale VGA 640 x 480 à 60 Hz

On va générer ces signaux à l'aide de compteurs.

On décide de coder en VHDL la synthèse de la synchronisation horizontale par le code suivant (attention code sur deux pages) :

```

1 architecture Behavioral of vgaControler is
2   signal hCounter : std_logic_vector (10 downto 0) := (others=>'0');
3   signal vCounter : std_logic_vector (9 downto 0) := (others=>'0');
4   signal hS, vS : std_logic := '0';
5 begin
6   process (clk)
7     begin
8       if (clk'event and clk='1') then
9         if hCounter = TEMPS1 then
10           hCounter <= (others=>'0');
11         else hCounter <= hCounter + '1';
12         end if;
13       end if;
14     end process ;
15
16   hS <= '0' when (hCounter > TEMPS2) and (hCounter < TEMPS3) else
17   '1';
18   [...]
```

**Question C-3 :** TEMPS1, TEMPS2 et TEMPS3 sont trois constantes symboliques.

Que valent les valeurs de TEMPS1, TEMPS2 et TEMPS3 en décimal dans le code VHDL (on rappelle que l'horloge interne fonctionne à 50 MHz) ?

## Partie D : drôle de RAM<sup>2</sup> (1A et 1B : 5 points).

Après avoir généré la synchronisation, qui est une partie indépendante, on s'intéresse à la RAM vidéo contenant l'image. L'interconnexion entre la RAM vidéo et la synchronisation est précisée sur la figure 13 qui détaille l'intérieur des parties C et D.

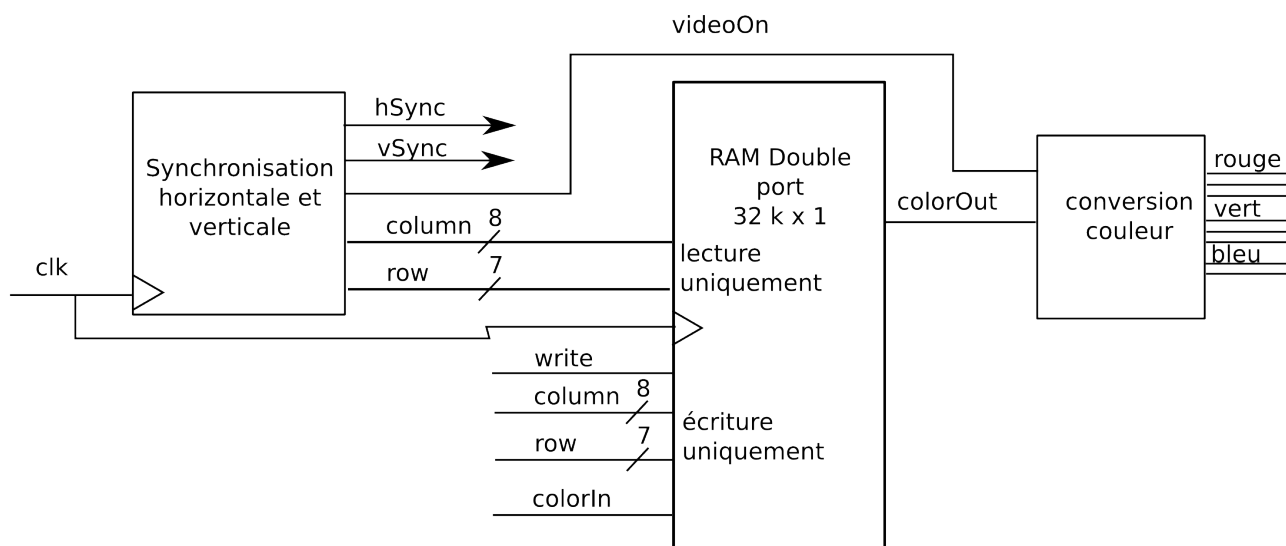


Figure 13: détail du contrôleur VGA complet.

Le bloc de conversion de couleur n'est qu'un simple bloc combinatoire transformant les « 0 » en une couleur déterminée, et le « 1 » en noir, c'est à dire que `rgb="00000000"`.

**Question D-1 :** sachant que l'écran sera géré selon la norme 640 par 480 (une image entière comporte 640 lignes et 480 colonnes), vu qu'un pixel est codé sur 8 bits (voir plus haut question C-2), quelle serait la quantité de RAM (en bit et en octets) nécessaire pour mémoriser une image ?

**Question D-2 :** les FPGA intègrent tous une certaine quantité de mémoire sous forme de RAM simple ou double port. Existe-t-il un composant de la gamme Xilinx (gamme précisé sur le tableau figure 14) capable de faire tenir un tel contrôleur d'écran complet ? On ne tiendra compte que de la RAM par Bloc disponible (donnée en kbits).

Device	System Gate	Equivalent Logic Cell	Block RAM bits	Dedicated Multiplier	Maximum user I/O
XC3S100E	100k	2160	72k	4	108
XC3S250E	250k	5508	216k	12	172
XC3S500E	500k	10476	360k	20	232
XC3S1200E	1200k	19512	504k	28	304
XC3S1600E	1600k	33192	648k	36	376

Figure 14: Tableau de choix des FPGA Spartan 3, extrait simplifié des data sheet Xilinx

**Question D-3 :** Ne disposant « que » de composants XC3S100E, on va limiter

<sup>2</sup> Toute ma considération ira à qui voit ici un jeu de mot culturel. J'aurais aussi pu écrire : « Aux RAMs, galériens »...

notre utilisation à une image 160 par 120 pixels, image monochrome. Justifiez qu'une telle utilisation est possible en calculant la quantité de RAM restant disponible après l'implémentation du contrôleur.

La RAM disponible est organisée en quatre blocs proposant 14 bits d'adresses.

**Question D-4 :** Comment interconnecter deux blocs de RAM de 14 bits pour en faire un bloc de 15 bits d'adresses ? (**attention** : ce n'est pas la même question qu'en cours – slide vue en 1A et 1B – car la sortie n'est pas trois états. Il est donc nécessaire de choisir une des deux sorties).

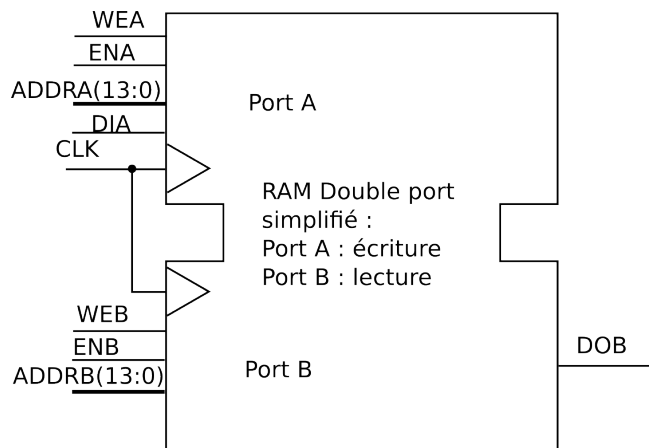


Figure 15: RAM Double port

- ENx permet d'activer le port X.
- WEx : S'il vaut '1' : écriture. S'il vaut '0' : lecture.
- DIx : Donnée entrée sur le port X.
- DOx : Donnée lue sur le port X.
- ADDR<sub>x</sub> : Adresse de travail du port X.

## Document-réponse A-2 :

Nom, prénom, groupe :

Indiquez ci-dessous le détail de l'architecture du séquenceur en utilisant des bascules et des blocs combinatoires identifiés :

Encadrez dans le code ci-dessous chaque partie correspondant aux blocs ci-dessus :

```
1 architecture Behavioral of gestionBtn is
2 type etat is (released,short,long,bounce);
3 signal present, futur : etat;
4 begin
5     process (clk)
6     begin
7         if (clk'event and clk='1') then
8             present <= futur;
9         end if;
10    end process;
11
12    process (present, btn, fin100, fin200, fin1000)
13    begin
14        case present is
15            when released =>
16                if btn = '1' then futur <= short;
17                else futur <= released;
18                end if;
19            when short =>
20                if btn = '0' then futur <= bounce;
21                elsif fin1000='1' then futur <= long;
22                else futur <= short;
23                end if;
24            when long =>
25                if btn = '0' then futur <= bounce;
26                else futur <= long;
27                end if;
28            when bounce =>
29                if fin100='1' then futur <= released;
30                else futur <= bounce;
31                end if;
32        end case;
33    end process;
34
35    infoBtn <= '1' when (present=released and btn='1') or
36    (present=short and fin200='1') or (present=long and fin100='1') else
37    '0';
38
39    lanceTempo <= '1' when (present=released and btn='1') or
40    (present=short and btn='0') or (present=long and btn='0') else '0';
```

Nom, prénom, groupe :

