الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DR. YAHIA FARES DE MEDEA

FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département du Génie Électrique et de l'Informatique



Projet de Fin d'Etudes

Présenté pour l'obtention du Diplôme
de Master en Génie Électrique
Spécialité : *Systèmes Intelligents et Communicants*

THEME

ETUDE ET IMPLEMENTATION D'UN CONTROLEUR VGA SUR CARTE FPGA

Proposé et dirigé par,

Dr. BOUHEDDA Mounir

Préparés et présentés par,

Melle. BENAYAD Sihem

Melle. BOUDERBALA Houria

Dédicace

Merci Allah (mon dieu) de m'avoir donné la capacité d'écrire et de réfléchir bien , la force d'y croire,
la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire
«الحمد الله»

Je dédie ce modeste travail à celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère

A mon père, école de mon enfance, qui a été mon ombre durant toutes les années des études,
et qui a veillé tout au long de ma vie
à m'encourager, à me donner l'aide et à me protéger.
Que dieu les gardes et les protège.

A mes sœurs : Farida, Amina , Fatima et Sarah,

A mon seule et unique frère : Mohamed.

A ma tante Ratiba à qui je souhaite une longue et heureuse vie.

A toute ma famille et surtout mes cousines et mes tantes

A toutes mes ami(e)s surtout Khadidja et Houria
Que je considère comme des sœurs.

merci a tuos

BENAYAD Sihem

Dédicace

A ceux qui m'ont tout donné sans rien en retour

A ceux qui m'ont encouragé et soutenu dans mes

Moments les plus difficiles.

A mon grand-père ma grand-mère, mon père et ma mère

Pour leur amour Et leur support continu

Que ce modeste travail, soit l'exaucement de vos veux tant formulés et de

Vos prières quotidiennes.

Que dieu, le tout puissant, vous préserve et vous procure santé

et longue vie afin que je puisse à mon tour vous combler.

A mon seul frère Elhachemi. Mes chères sœurs Zahira, Zehour, ma

petite sœur Marcua et Meriem

Aucune dédicace ne serait exprimer assez profondément ce que je ressens

envers vous.

Vous occupez une place particulière dans mon cœur. Je vous dédie ce

vous souhaitant un avenir radieux, plein de

bonheur et de succès.

A mes très chères Sihem et Khadidja

En témoignage de l'amitié sincère qui nous a liée et des bons moments passés

ensemble. En souvenir de nos éclats de rire.

En souvenir de tout ce qu'on a vécu ensemble.

J'espère de tout mon cœur que notre amitié durera éternellement.

A mes oncles, mes tantes, mes cousins et mes cousines

A tous ma famille, petits et grands merci pour vos encouragements.

A tous ceux qui me sont chers.

Houria

Remerciement

En premier lieu, nous tenons d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier et à exprimer nos gratitudes à notre encadreur Mr.BOUHEDDA Mounir qui nous a orienté et nous a soutenu durant notre projet de fin d'études.

Nous témoignons reconnaissance et gratitudes aux aimable crut personnes, quel que soit de près ou de loin, qui nous ont soutenu, acceptent et orientant durant notre projet de fin d'étude.

Nous Remercions tous les ingénieurs de laboratoire P3 pour leur disponibilité qui nous a infiniment aidés dans notre travail.

Nous remercions Melle TLEMSANI pour son supporte et son aide.

Enfin, nous remercions vivement les membres de jury.

ملخص:

الهدف من هذا العمل هو تصميم وتنفيذ وحدة تحكم VGA على الدارة الكهربائية FPGA. ويركز العمل على إتقان عرض أجسام ثابتة, متحركة و كذلك عرض نصوص على شاشة VGA. لاختبارو التحقق من صحة عملنا, تم الانتهاء من اللعبة وتنفيذها على لوحة SPARTAN-3A DSP 1800A ل XILINX باستعمال لغة البرمجة VHDL.

الكلمات المفتاحية: وحدة تحكم VGA, FPGA, VHDL.

Résumé:

L'objectif de ce travail est la conception et la réalisation d'un contrôleur VGA sur carte FPGA. Le travail est focalisé sur la maitrise de l'affichage d'objets fixes, animées ainsi que des textes sur un écran VGA. Pour tester et valider notre travail, un jeu a été réalisé et implémenté sur la carte SPARTAN-3A DSP 1800A de XILINX en utilisant le langage VHDL.

Mots clés: Contrôleur VGA, FPGA, VHDL.

Abstract:

The objective of this work is the design and the realization of a controller VGA on card FPGA. The work is focused on the maitrise posting of fixed objects, animated as well as texts on a screen VGA. To test and validate our work, a play was carried out and implemented on chart SPARTAN-3A DSP 1800A of XILINX by using VHDL language.

Key words: Controller VGA, FPGA, VHDL.

Introduction générale

VGA Video graphic array est un standard d'affichage, successeur du MDA, CGA et EGA. Se rapporte à n'importe quel dispositif ou programme machine qui rendent un ordinateur capable de montrer et de manœuvrer des images.

Beaucoup d'adapteurs visuels soutiennent plusieurs différents modes de résolution, qui sont divisés en deux catégories générales : mode caractère et mode de graphiques.

Des deux modes, le mode de graphiques est le plus sophistiqué. Les programmes qui fonctionnent en mode de graphiques peuvent montrer une variété illimitée de forme et de polices, tandis que les programmes fonctionnant en mode caractère sont sévèrement limités. Des programmes qui fonctionnent entièrement en mode de graphiques s'appellent les programmes graphiques (graphics-based).

VGA est un adaptateur mode vidéo ou adapteur visuel, La plupart des adapteurs visuels peuvent fonctionner en mode des textes ou mode de graphiques. En mode des textes, un moniteur peut montrer seulement des caractères d'ASCII. En mode de graphiques, un moniteur peut montrer n'importe quelle image bit-mapped. En plus du texte et des modes de graphiques, les adapteurs visuels offrent différents modes de résolution et d'intensité de la couleur [1].

Motivé par l'utilisation des avantages des circuits numériques ainsi le standard d'affichage VGA, dans notre projet une nouvelle technique qui a nous permet de réaliser un contrôleur VGA est modélisé et optimisé en vue d'une implémentation FPGA.

Notre objectif consiste à étudier et implémenter un contrôleur VGA en carte FPGA Spartan-3A DSP 1800A. Comment peut-on afficher sur l'écran des textes et des objets colorés, dynamiques et fixes ?

Tout au long de notre travail, on a été amené à utiliser le logiciel ISE Design Suite 14.1 de Xilinx.

Notre travail s'inscrit dans le cadre de l'étude et l'implémentation d'un contrôleur VGA sur une carte FPGA. Le mémoire se divise en trois chapitres organisés de la manière qui suit:

Le premier chapitre nous présentons des généralités sur le principe du contrôleur VGA et on a parlée aussi sur le fonctionnement de la carte VGA; on a précisé les valeurs de

synchronisation de l'affichage sur écran à tube cathodique pour une résolution de 640 x 480; on a donné une petite définition sur le codage vidéo.

Le deuxième chapitre est consacré pour les résultats des simulations et l'affichage des différentes couleurs en plusieurs manières sur un écran VGA et on a parlé brièvement sur la carte FPGA.

Le troisième chapitre nous présentons la réalisation d'un jeu de PONG, et nous présentons tous les étapes de notre code VHDL avec les circuits inclus.

A la fin notre mémoire sera achevé par une conclusion générale et des perspectives.

Table des matières

LISTE DES FIGURES ..... **I**

LISTE DES TABLEAUX ..... **III**

LISTE DES ACRONYMES ..... **IV**

CHAPITRE I : LE PRINCIPE DU CONTROLEUR VGA **1**

I. Introduction **2**

II. Historique **2**

III. Fonctionnement des cartes VGA **2**

IV. Définition de la zone d'affichage avec compteurs **4**

V. Le Connectique du VGA **5**

VI. Les modes d'affichages **7**

VII. Les composants d'une carte VGA **7**

VII.1. *Cathod Ray Tube Controller ou contrôleur de tube cathodique* 8

VII.2. *La mémoire vidéo (ou mémoire d'écran)* 8

VII.3. *Graphics Data Controller ou contrôleur graphique* 8

VII.4. *Digital to Analog Converter ou convertisseur numérique analogique* 8

VII.5. *Attribute Controller ou contrôleur d'attributs* 9

VII.6. *Timing Sequencer ou séquenceur de timing* 9

VII.7. *Vue d'ensemble* 10

VIII. Le moniteur à tube cathodique **11**

IX. Le codage vidéo **12**

X. Conclusion **12**

CHAPITRE II : Conception et implémentation du contrôleur VGA sur FPGA **13**

I. Introduction **14**

II. Xilinx Spartan-3A DSP FPGA **14**

II.1. *Les atouts des circuits FPGA* 14

III. La synchronisation VGA 17

III.1. *Affichage d'une seule couleur* 18

III.2. *Affichage deux bandes verticales* 20

III.3. *Affichage deux bandes horizontal à dégradés verticales* 21

III.4. *Affichage de deux bandes horizontales dégradées* 22

III.5. *Affichage de trois bandes sans dégradation* 23

III.6. *Affichage de trois bandes Avec dégradation* 24

III.7. *Affichage de quatre zones* 24

IV. conclusion 26

CHAPITRE III: REALISATION DU JEU DE PONG 27

I. Introduction 28

II. Cahier des charges 28

III. Jeu de PONG: mode un joueur 28

III.1. *Image Statique (fixe)* 28

III.2. *Les objets dynamiques* 29

III.3. *Affichage de texte* 33

III.4. *Circuit général du jeu PONG mode un* 35

III.5. *Implémentation* 50

IV. Jeu de PONG : mode deux joueurs 51

IV.1. *Image fixe* 51

IV.2. *les objets dynamique* 51

IV.3. *Circuit général du jeu PONG (mode deux)* 51

IV.4. *Les modification des circuits* 52

IV.5. *Implémentation* 60

V. Conclusion 60

LISTE DES FIGURES

Figure I. 1. *Connectique VGA d'une FPGA Spartan-3 DSP.* **2**
Figure I. 2. *Exemple de timing d'un écran à tube cathodique.* **3**
Figure I. 3. *Durée des temps de synchro pour l'affichage VGA.* **5**
Figure I. 4. *Connecteurs VGA mâle et femelle.* **6**
Figure I.5. *Vue d'ensemble des composants d'une carte VGA.* **10**
Figure I. 6. *Schéma conceptuel d'un CRT moniteur.* **11**
Figure II. 1. *Sortie VGA [10]* **16**
Figure II. 2. *Les entres/sorties de circuit de synchronisation.* **17**
Figure II.3. *L'organigramme général du circuit de synchronisation.* **18**
Figure II. 4. *L'organigramme d'affichage de la même couleur.* **19**
Figure II. 5. *Simulation du même coller.* **19**
Figure II.6. *Affichage d'une couleur sur tout l'écran.* **19**
Figure II.7. *L'organigramme d'affichage de deux bandes verticales.* **20**
Figure II. 8. *Simulation de deux bandes verticales.* **20**
Figure II. 9. *Affichage de deux bandes verticales sur tout l'écran.* **21**
Figure II. 10. *Simulation horizontale dégradée pour R* **21**
Figure II. 11. *Simulation horizontale dégradée pour V* **21**
Figure II. 12. *Affichage de deux bandes horizontales à dégradation verticales.* **22**
Figure II.13. *Affichage de deux bandes horizontales dégradées.* **22**
Figure II.14. *L'oganigramme d'affichage de trois bandes.* **23**
Figure II.15. *Affichage de trois bandes colorées.* **23**
Figure II.16. *La Simulation de quelques exemples.* **24**
Figure II.17. *Affichage de trois bandes dégradées verticalement.* **24**
Figure II.18. *L'organigramme d'affichage de quatre zones.* **25**
Figure II.19. *Simulation de quatre zones colorées.* **25**
Figure II.20. *Affichage de division de la zone d'affichage sur quatre.* **26**
Figure III. 1. *L'image initiale.* **29**
Figure III.2. *Les deux cas de contact de palette avec les bords.* **29**
Figure III.3. *Conversion de la balle.* **30**
Figure III.4. *Balle colorée.* **31**
Figure III.5. *Le contact avec le haut, le bas et le mur.* **32**

Figure III.6. *Les principes de rebondissement.* **32**
Figure III.7. *Les entrées /sorties de circuit ROM.* **33**
Figure III.8. *Exemple du caractère « S » dans la Rom.* **35**
Figure III. 9. *Le circuit général du jeu du premier mode un PONG.* **35**
Figure III.10. *Les entrées/sorties de texte.vhd.* **36**
Figure III.11. *Les entres/sorties de ROM.vhd.* **37**
Figure III.12. *Affichage de la règle.* **39**
Figure III.13. *Message de la règle affiché du 1er mode* **39**
Figure III.14. *L'affichage sur l'écran du score et duré de la vie.* **41**
Figure III. 15. *Zone d'affichage du score et la durée de vie.* **41**
Figure III.16. *Zone d'affichage de du slogan.* **42**
Figure III.17. *Zone d'affichage du game over.* **43**
Figure III.18. *Les entrées/sorties du circuit graphe.* **44**
Figure III.19. *L'état initial de la belle et la palette.* **44**
Figure III.20. *Les entrées/sorties du circuit temps.* **45**
Figure III.21. *Les entrées/sorties de circuit compteur.* **46**
Figure III.22. *Les entrées/sorties de pong.* **46**
Figure III.23. *Hiérarchie de jeu de PONG.* **47**
Figure III.24. *La machine d'état.* **48**
Figure III.25. *L'organigramme général.* **49**
Figure III.26. *Processus d'exécution.* **50**
Figure III.27. *L'image initiale.* **51**
Figure III.28. *Circuit général mode deux.* **52**
Figure III.29. *Les entrées/sorties du circuit de pong.* **53**
Figure III.30. *La hiérarchie de jeu de pong.* **53**
Figure III.31. *Les entrées/sorties de circuits de graphe.* **54**
Figure III.32. *Les boutons poussoirs sur la carte FPGA.* **54**
Figure III.33. *Les entrées/sorties du circuit texte.vhd.* **55**
Figure III.34. *Position d'affichage des scores et durée de vie.* **57**
Figure III.35. *Message de la règle affiché du 2éme mode.* **57**
Figure III.36. *Affichage du résultat cas joueur 2 gagné le jeu.* **58**
Figure III.37. *Départ du jeu deuxième mode.* **59**
Figure III.38. *Fin du jeu deuxième mode.* **59**
Figure III.39. *Hiérarchie de jeu de PONG mode deux.* **60**

LISTE DES TABLEAUX

Tableau I. 2. *Tableau des durées de synchronisation.* **.....4**
Tableau I. 3. *Signaux sur connecteur VGA.* **.....6**
Tableau III.1. *Code ASCII de 128 premiers caractères.* **.....34**
Tableau III.2. *Rapport d'implémentation.* **.....50**
Tableau III.3. *Rapport d'implémentation du mode deux.* **.....60**

LISTE DES ACRONYMES

ASIC: *Application Spécific Integrated Circuit.*

ATC: *Attribute Controller.*

CGA: *Color Graphics Adapter.*

CRTC: *Cathode Ray Tube Controller.*

CLK: *Clock.*

DDR: *Double Data Rate.*

DDC: *Display Data Channel.*

DSP: *Digital Signal Processor.*

DAC: *Digital to Analog Converter.*

EGA: *Enhanced Graphics Adapter.*

FPGA: *Field Programmable Gate Arrays.*

FPS: *Frame Per Second.*

GDC: *Graphics Data Controller.*

I/O: *In/Out (Entrée/ sortie).*

IBM: *International Business Machines.*

LUT*: Look up Table.*

MDA: *Monochrome Display Adapter.*

MIPS: *Million instruction per second.*

MMU: *Memory Management Unit.*

PLD**:** *Programmable Logic Device.*

PS/2: *Personnal System/2.*

PE: *processing éléments.*

RGB: *red, green, blue.*

RAM: *Random Access Memory.*

ROM: *Read Only Memory.*

SPI: *Serial Flash Pinout Interface.*

TS: *Timing Sequencer.*

USB: *Universel Serial Bus.*

VGA: *Video Graphics Array.*

VHSIC: *Very High Speed Integrated Circuit.*

SDRAM: *Synchronous Dynamic Random Access Memory.*

CHAPITRE I : LE PRINCIPE DU CONTROLEUR VGA

I. Introduction

Dans ce chapitre nous nous intéressons à connaitre le principe du contrôleur VGA. VGA (Video Graphics Adapter) est une norme d'affichage visuel. Il fournit une méthode simple pour relier un système à un moniteur pour montrer l'information ou des images. Comme norme d'affichage standard, le VGA a été largement répandue.

II. Historique

Video Graphics Adapter, ou VGA, est une ancienne technologie sur laquelle reposent toutes les cartes vidéo modernes. Chaque carte vidéo produite dans les deux dernières décennies peut émettre une résolution d'affichage VGA standard de 640 x 480 - même si la plupart d'entre eux peuvent aller plus haut.

VGA a été introduit par IBM en 1987, quand ils ont introduit la gamme d'ordinateurs PS/2. Il pouvait afficher 16 couleurs différentes à la fois à la résolution standard, et jusqu'à 256 couleurs à une résolution de moitié. C'était aussi la première série de graphiques qui pourrait être intégré directement à la carte mère [2].

III. Fonctionnement des cartes VGA

Le signal VGA se décompose en 5 signaux comme montre la Figure I.1.Trois d'entre eux concernent les couleurs primaires et les deux autres servent à effectuer le balayage horizontal et vertical comme pour un écran analogique [3].

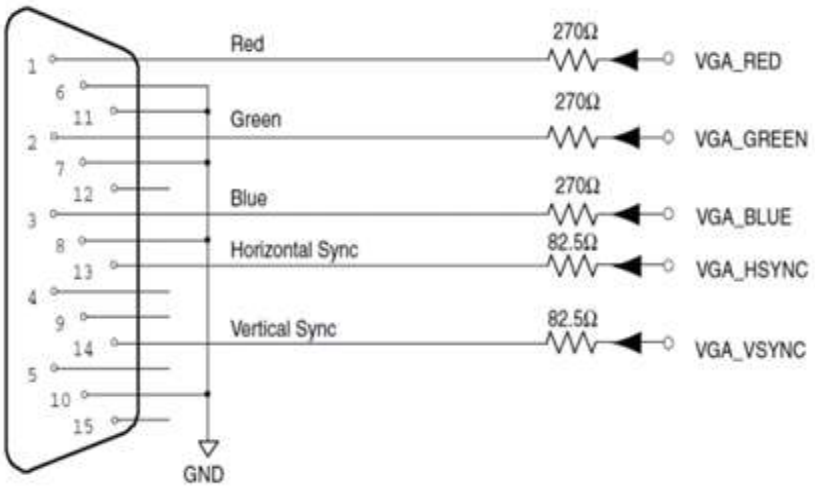


Figure I. 1. *Connectique VGA d'une FPGA Spartan-3 DSP.*

Les signaux VGA_HSYNC et VGA_VSYNC sont les deux signaux qui vont définir le balayage de l'écran à une fréquence de 60Hz. Le premier signal correspond à la synchronisation horizontal et le deuxième à la synchronisation verticale. Pour comprendre cela il suffit de regarder comment se déplace le « faisceau ». Le faisceau de balayage plus large que l'écran et va de gauche à droite sans arrêt. Il ne revient pas spontanément à gauche et sur la ligne du dessous. Pour cela nous utilisons le signal de synchronisation horizontal qui va faire revenir le faisceau au début de la ligne suivante (Figure I.2) [3].

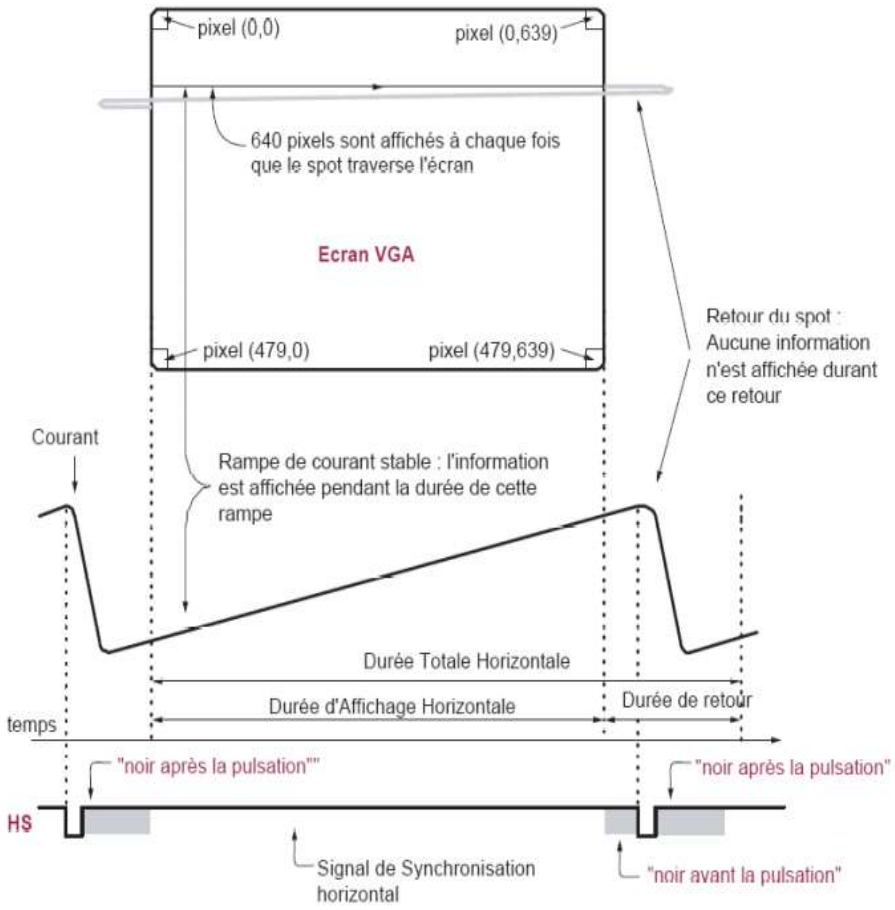


Figure I. 2. *Exemple de timing d'un écran à tube cathodique.*

On remarque alors que le retour à la ligne suivante se fait lors du passage du signal de synchronisation horizontal du niveau haut au niveau bas pendant une certaine durée. Ainsi lorsqu'un certain nombre de ligne auront été passé le faisceau se situera en bas à droite de l'écran. Afin de la ramener en haut à gauche et de recommencer il faut cette fois utiliser le signal de synchronisation vertical, en le faisant passer d'un niveau haut à un niveau bas [3].

Le rôle principal d'une carte graphique est de permettre l'affichage d'informations sur un écran d'ordinateur. Ces informations peuvent être de plusieurs types : caractères alphanumériques, figures géométriques... Une carte graphique dite VGA doit simplement être compatible avec le standard d'affichage VGA. Ce standard définit un certain nombre de modes d'affichage que la carte doit prendre en charge [3].

Le fonctionnement interne d'une carte VGA peut varier d'un constructeur à l'autre. Cependant, toute carte compatible avec la norme possède un certain nombre de composants standard pour assurer son fonctionnement. Ces composants vont permettre d'associer une couleur à chaque pixel de l'écran et de gérer le déplacement du spot du tube cathodique [3].

Les cartes graphiques modernes répondent à des standards beaucoup plus évolués que le standard VGA. Elles permettent des résolutions plus fines sur du matériel différent .Toutes ces nouvelles cartes émulent cependant les composants des premières cartes VGA, afin de permettre aux anciens programmes exploitant les modes d'affichage VGA de toujours fonctionner correctement [3].

IV. Définition de la zone d'affichage avec compteurs

Pour pouvoir afficher dans la zone d'écran possible, nous utilisons deux compteurs (hcount et vcount) pour gérer les signaux de synchronisation horizontale et verticale. Ces derniers sont initialisés à 0, à chaque coup d'horloge à 25MHz le compteur horizontal et vertical sont incrémenté pour obtenir les cycles d'horloges à fin de définir la zone d'affichage, les cycles présentés dans le Tableau I.1 tableau des durées de synchronisation.

Tableau I. 1. *Tableau des durées de synchronisation.*

Symbole	paramètre	Synchro Verticale			Synchro Horizontale	
		Temps	Horloge	Lignes	Temps	Horloge
T_S	Durée synchro	16.7ms	420000 cycles	525	32 µs	800 cycles
T_{DISP}	Durée affichage	15.36ms	384000 cycles	480	25.6µs	640 cycles
T_{PW}	Largeur pulsation	64µs	1600 cycles	2	3.84µs	96 cycles
T_{FP}	Noir avant pulsation	320µs	8000 cycles	10	640ns	16 cycles
T_{BP}	Noir après pulsation	928µs	23200 cycles	29	1.92µs	48 cycles

Ces valeurs correspondent à un signal de synchronisation, qui a la même forme pour la synchronisation horizontale et la synchronisation verticale, et qui est représenté sur la FigureI.3 suivante [4].

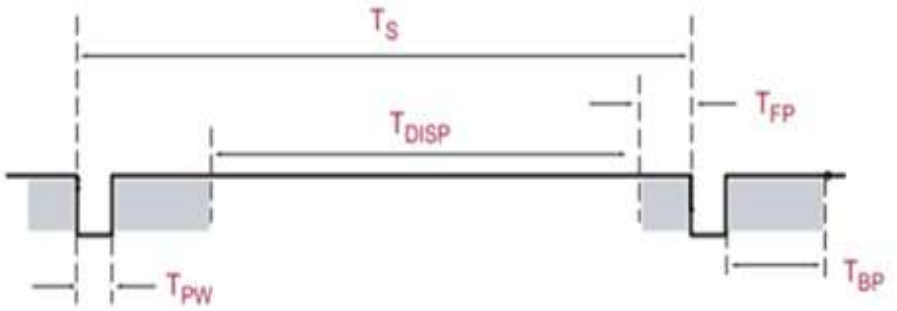


Figure I. 3. *Durée des temps de synchro pour l'affichage VGA.*

Les valeurs des temps (T_S, T_{DISP}, T_{PW}, T_{FP}, T_{BP}) de synchronisation et d'affichage sont données par le Tableau I.1.

A l'aide du schéma de la Figure I.3 précédent nous voyons qu'il faut attendre $T_{PW} + T_{BP}$ avant d'être dans une zone d'affichage.

En l'occurrence ceci correspond à 96 + 48 = 144 coups d'horloge ou une valeur de 144 pour le compteur horizontal. Le signal n'affiche plus au-delà de 784 coups d'horloge, ce qui correspond bien à 640 pixels affichés. Quant au compteur vertical, il est incrémenté lorsque le compteur horizontal atteint sa valeur maximale, soit 800. Cette fois, l'affichage n'est possible qu'entre des valeurs du compteur vertical comprises entre 34 et 514. Ce qui représente bien à une hauteur de 480 pixels.

Afin que l'affichage soit bien synchronisé, il faut absolument que les signaux R, G et B soit à 0 quand le spot lumineux est en dehors de l'écran (période de retour du spot).

V. Le Connectique du VGA

Un connecteur VGA montre que l'interface est complexe puisqu'elle gère 15 signaux différents. Un connecteur VGA est un connecteur de type D-sub (D-subminiature ou D-sub est un type de connecteurs électriques très répandus, notamment pour la connexion de matériel informatique), plus précisément DE-15. Il est utilisé pour connecter la carte d'évolution FPGA SPARTAN-3 (DSP 1800A de Xilinx) à un écran d'ordinateur en analogique. Ce connecteur est parfois appelé connecteur RGB, HD-15 ou D-sub 15. Il possède 15 broches organisées en trois rangées (Figure I.4).Chaque broche génère un signal de rôle différent que l'autre (Tableau I.2) [5].

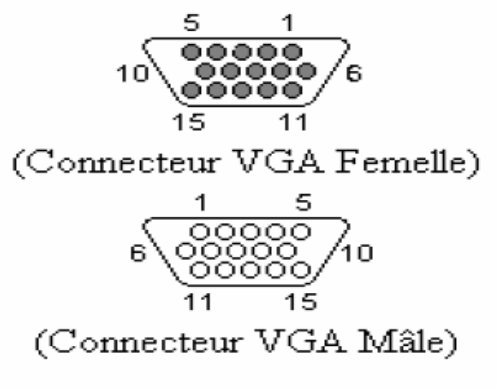


Figure I. 4. *Connecteurs VGA mâle et femelle.*

Tableau I. 2. *Signaux sur connecteur VGA.*

Broche	Nom	Direction	Description
1	Red (R)	→	Vidéo rouge (75ohms, 0.7 v)
2	Green (G)	→	Vidéo verte (75ohms,0.7 v)
3	Blue (B)	→	Vidéo bleu (75ohms,0.7v)
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)

VI. Les modes d'affichages

Le standard VGA supporte également 4 plans vidéo, le scrolling matériel, la division de l'écran en zones indépendantes et des polices de caractères définies par logiciel.

Les modes graphiques standards sont:

- 640 x 480 x 4 (16 couleurs)
- 640 x 400 x 4 (16 couleurs)
- 320 x 200 x 4 (16 couleurs)
- 320 x 200 x 8 (256 couleurs)

Il supporte également les précédents standards : EGA, CGA et MDA, et d'autres non documentés. Un autre mode graphique appelé Mode X permet d'autres solutions techniques et définitions qui ne sont pas disponibles en Mode 13h [4].

Le mode texte alphanumérique est de 80 x 25 ou 43 x 25 caractères. Chaque cellule peut être définie en utilisant 16 couleurs de texte qui peut être rendu clignotant. Dans ce cas, seules 8 couleurs sont disponibles pour le fond contre 16 si le texte n'est pas clignotant. Bien qu'un mode monochrome soit disponible, la plupart des programmeurs utilisent le mode couleur avec un texte gris et un fond noir [4].

Matériellement, l'espace vidéo monochrome se situe dans la plage d'adresse mémoire 0xB0000-0xB7FFF, et l'espace en couleur dans la plage 0xB8000-0xBFFFF. En mode couleur, chaque cellule de l'écran est associée à deux octets l'un pour le caractère et l'autre utilisé pour ses attributs (clignotement, police, couleur ...) [4].

VII. Les composants d'une carte VGA

Chacun des composants de la carte VGA réalise une opération bien précise. Tous ces composants sont paramétrables par le programmeur qui souhaite créer un mode vidéo. La configuration se fait en accédant aux registres de la carte graphique. Le mode 13h du BIOS, par exemple, correspond à une configuration précise de tous les registres. Il y a en tout plus d'une soixantaine de registres 8 bits sur la carte. Le nombre de configurations possibles (et donc de modes vidéo) est donc très grand. En pratique, seul un petit nombre d'entre eux sont effectivement utilisés [5].

VII.1. *Cathod Ray Tube Controller ou contrôleur de tube cathodique*

Une des fonctions de la carte est de commander le déplacement du spot sur l'écran. C'est le rôle dévolu au CRTC. Celui-ci possède un grand nombre de registres configurables par le programmeur qui influent directement sur les mouvements du spot. C'est grâce au CRTC qu'on peut, par exemple, modifier la taille et la résolution de l'image [5].

VII.2. *La mémoire vidéo (ou mémoire d'écran)*

L'image affichée à l'écran est stockée dans la mémoire vidéo. Lorsqu'on est en mode texte, ce sont les caractères ASCII et leur couleur qui sont stockées. En mode graphique, la couleur de chaque pixel est stockée dans la mémoire d'écran.

Il existe un lien fort entre la mémoire vidéo et le CRTC. En effet, le CRTC connaît en permanence la position du spot. Il va donc pouvoir sélectionner l'adresse dans la mémoire du pixel (en mode graphique) ou du caractère (en mode texte) que le spot doit afficher à l'écran. Une deuxième fonction du CRTC est donc de calculer l'adresse dans la mémoire vidéo qui contient l'information à afficher sur le moniteur [5].

VII.3. *Graphics Data Controller ou contrôleur graphique*

Le processeur doit pouvoir écrire dans la mémoire vidéo, pour modifier le texte ou changer l'image affichée à l'écran. Le contrôleur graphique propose plusieurs modes d'écriture et de lecture de la mémoire vidéo. Ces différents modes servent à accélérer les modifications de l'affichage lors de remplissage de zones avec une même couleur par exemple, ou des copies de portions d'écran. Le contrôleur graphique est donc l'intermédiaire entre le processeur et la mémoire graphique [5].

VII.4. *Digital to Analog Converter ou convertisseur numérique analogique*

Le moniteur doit commander l'intensité des trois faisceaux d'électrons (rouge, vert, bleu) qui vont permettre l'affichage des différentes couleurs. Il doit recevoir des signaux analogiques pour commander ces faisceaux, et non des signaux numériques. Le DAC possède une palette qui à un index de couleur (choisi parmi 256), associe les proportions des trois signaux de base rouge, vert et bleu. Ces proportions peuvent être modifiées par le programmeur pour chacune des 256 couleurs. Le DAC se charge alors de traduire les

proportions associées à une couleur en signaux analogiques parfaitement compréhensibles par le moniteur [5].

VII.5. *Attribute Controller ou contrôleur d'attributs*

Périodiquement, les données de la mémoire vidéo doivent être analysées pour en déduire l'image affichée à l'écran. Le contrôleur d'attributs va être chargé, durant l'affichage, de déduire à partir des informations en mémoire vidéo (caractères en mode texte ou couleur de pixel en mode graphique) l'index dans la palette du DAC [5].

VII.6. *Timing Sequencer ou séquenceur de timing*

Le séquenceur synchronise tous les composants de la carte cités jusqu'ici. Il est chargé de la gestion de l'horloge. Il s'occupe du rafraîchissement de la mémoire vidéo et de la gestion des accès concurrents à cette mémoire.

On affecte aussi au séquenceur la gestion du mode d'adressage de la mémoire vidéo. Cette mémoire à une capacité de 256 ko. Pourtant, le processeur ne dispose que d'un espace d'adressage de 64 ko pour y accéder. Ceci est dû au fait que la mémoire vidéo est composée de 4 "plans" de 64 ko. Le choix du plan lorsque le processeur accède à la mémoire dépend du mode d'adressage utilisé [5].

VII.7. *Vue d'ensemble*

Le schéma ci-dessous (Figure I.5) représente une vue d'ensemble sur l'interconnexion entre les composants de la carte VGA [5].

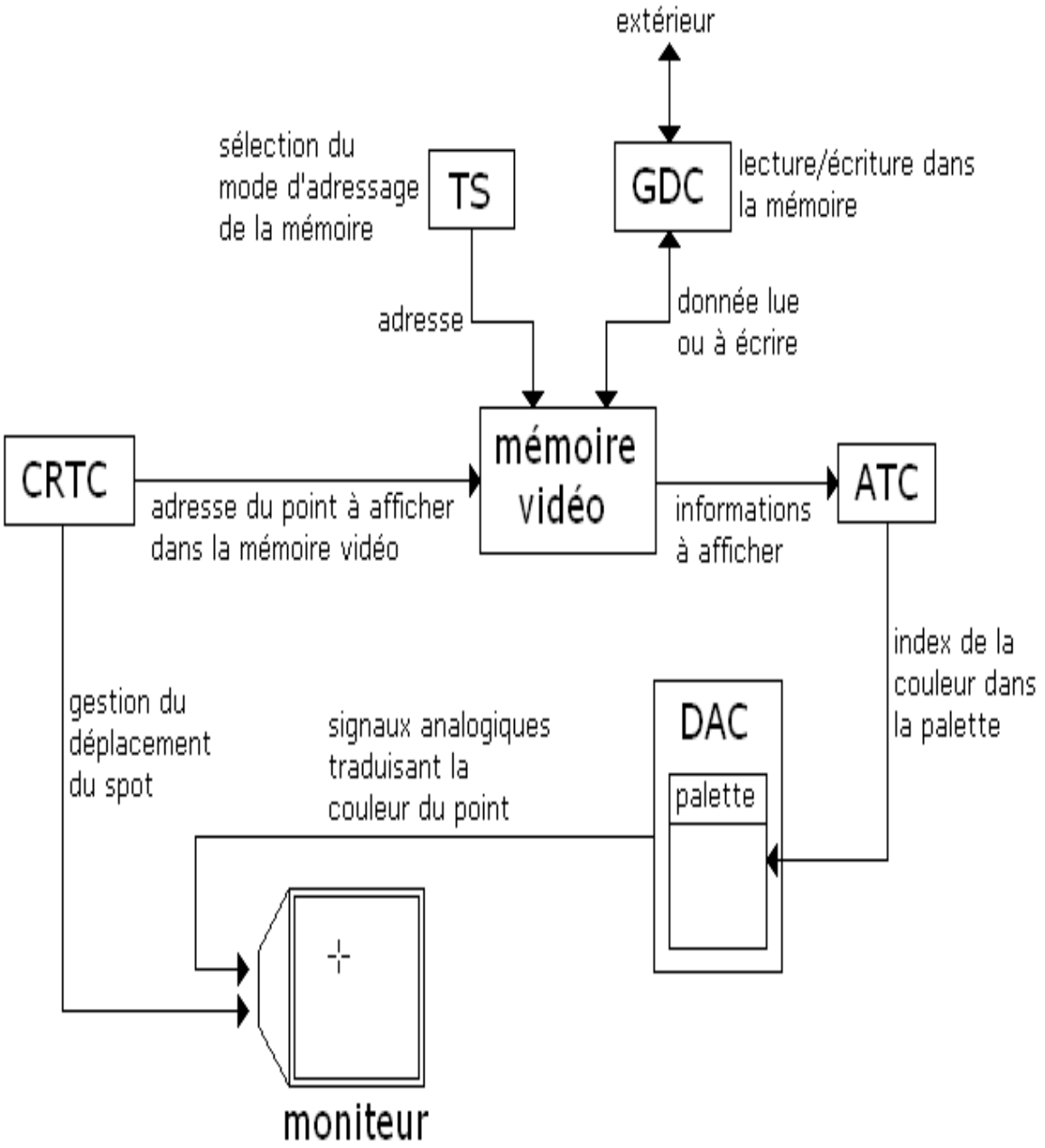


Figure I.5.*Vue d'ensemble des composants d'une carte VGA.*

VIII. Le moniteur à tube cathodique

Les moniteurs (écrans d'ordinateurs) sont la plupart du temps des tubes cathodiques (notés CRT, soit cathode ray tube ou en français tube à rayonnement cathodique), c'est à dire un tube en verre sous vide dans lequel un canon à électrons émet un flux d'électrons dirigés par un champ électrique vers un écran couvert de petits éléments phosphorescents (Figure I.6) [6].

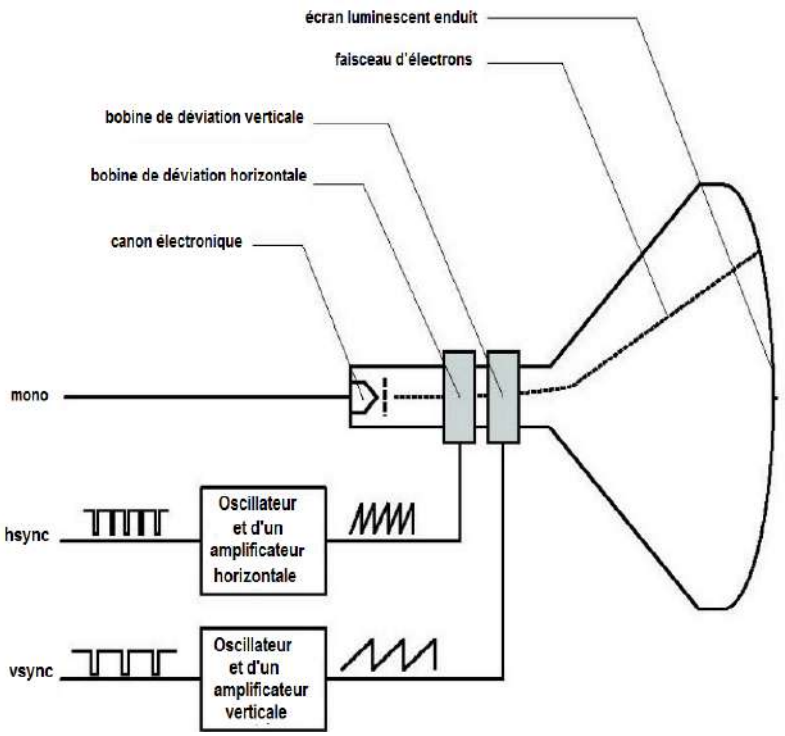


Figure I. 6. *Schéma conceptuel d'un CRT moniteur.*

Le canon à électrons est constitué d'une cathode, c'est-à-dire une électrode métallique chargée négativement, d'une ou plusieurs anodes (électrodes chargées positivement). La cathode émet des électrons attirés par l'anode. L'anode agit ainsi comme un accélérateur et un concentrateur pour les électrons afin de constituer un flux d'électrons dirigé vers l'écran. Un champ magnétique est chargé de dévier les électrons de gauche à droite et de bas en haut. Il est créé grâce à deux bobines X et Y sous tension (appelées *déflecteurs*) servant respectivement à dévier le flux horizontalement et verticalement [6].

L'écran est recouvert d'une fine couche d'éléments phosphorescents, appelés luminophores, émettant de la lumière par excitation lorsque les électrons viennent les heurter, ce qui constitue un point lumineux appelé pixel [6].

IX. Le codage vidéo

Une image en physique est une représentation d'un objet, produite par la réunion des rayons ou des faisceaux lumineux qui en émanent et se reconstituent sur un miroir, sur un écran ou sur l'œil qui perçoit cette image. Une image numérique est une image échantillonnée selon les deux axes spatiaux. Donc, une image numérique est composée d'une grille de points élémentaires appelés pixels. Le nombre de pixels par image, le nombre de bits pour le codage de chaque pixel et la nature de ces pixels constituent les principales caractéristiques d'une image numérique [6].

Une vidéo numérique est une séquence d'images numériques. L'œil humain a comme caractéristique d'être capable de distinguer environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper l'œil et de lui faire croire à une image animée. En plus des caractéristiques d'une image numérique fixe, une vidéo numérique est caractérisée par le nombre d'images par seconde (FPS = Frame Per Second). Le codage vidéo consiste à l'opération d'identification et de représentation d'une vidéo à l'aide d'un code informatique [6].

X. Conclusion

Dans ce chapitre, nous avons donné une présentation des différents composants qui peuvent exister sur un contrôleur VGA. Tout en expliquant le principe général en insistant sur les parties qui fait partie de notre travail.

CHAPITRE II : CONCEPTION ET IMPLEMENTATION DU CONTROLEUR VGA SUR FPGA

I. Introduction

Ce chapitre est consacré à l'étude et à l'implantation d'un contrôleur VGA dans un environnement logiciel/matériel (VHDL/FPGA).

II. Xilinx Spartan-3A DSP FPGA

Un FPGA, ou *field-programmable gate array*, est un composant constitué pour l'essentiel de blocs logiques configurables. Une fois 'programmé', il deviendra l'équivalent d'un processeur, pourvu de ses divers périphériques. Ce modèle est livré avec un cœur logiciel RISC 32 bits MicroBlaze (avec MMU optionnelle), utilisable sans versement de royalties. Il déploie ici une puissance d'environ 115 dhrystone[1] MIPS.

Le FPGA prend place sur une carte de développement Spartan-3A DSP 1800A, comprenant 128 Mo de DDR2 SDRAM, 16 Mo de flash parallèle et 8 Mo de flash SPI. La connectique comprend un port Ethernet à 10/100/1000 Mbps, un port série et une interface VGA [7].

II.1. Les atouts des circuits FPGA

II.1.1. Performance

Comme ils tirent parti du parallélisme matériel, les FPGA offrent une puissance de calcul supérieure à celle des processeurs de signaux numériques (DSP), car ils s'affranchissent du modèle d'exécution séquentielle et exécutent plus d'opérations par cycle d'horloge. BDTI, une importante société d'analyse et de « benchmarking », a publié des études montrant que les FPGA peuvent offrir une puissance de traitement par dollar plusieurs fois supérieure à celle d'une solution DSP dans certaines applications. Contrôler les entrées et sorties (E/S) au niveau matériel permet d'obtenir des temps de réponse plus courts ainsi que des fonctionnalités spécifiques, qui répondent mieux aux besoins de l'application [8].

II.1.2. Temps de mise sur le marché

Face à des préoccupations croissantes concernant les temps de mise sur le marché, la technologie FPGA représente une solution souple offrant des capacités de prototypage rapide. Ainsi, on peut tester une idée ou un concept, puis le vérifier sur du matériel sans avoir à

[1] Mesuré la performance du CPU.

passer par le long processus de fabrication d'un ASIC personnalisé. Par la suite, on peut apporter les éventuelles modifications nécessaires au circuit FPGA, en quelques heures au lieu de quelques semaines. Le matériel actuellement commercialisé propose également différents types d'E/S déjà connectées à un circuit FPGA programmable par l'utilisateur.
La multiplication des outils logiciels de haut niveau disponibles sur le marché permet de réduire le temps d'apprentissage avec les couches d'abstraction. Ces outils comprennent souvent des cœurs de propriété intellectuelle (fonctions précompilées) utiles pour le contrôle avancé et le traitement de signaux [8].

II.1.3. *Coût*

Les coûts de développement des ASIC personnalisés sont bien supérieurs à ceux des solutions matérielles basées sur du FPGA. L'important investissement de départ que requièrent les ASIC se justifie largement pour les grands fabricants, par exemple, qui peuvent livrer des circuits par milliers. Cependant, la plupart des utilisateurs finaux ont besoin de matériels personnalisés pour quelques dizaines ou quelques centaines de systèmes en développement. Par nature, les circuits programmables n'impliquent ni coût de fabrication, ni longs délais d'assemblage. Les besoins de la plupart des systèmes évoluent avec le temps ; or la modification progressive d'un FPGA représente un coût négligeable comparé à la dépense considérable qu'exige la préconception d'un ASIC [8].

II.1.4. Fiabilité

Tandis que les outils logiciels fournissent l'environnement de programmation, les circuits FPGA sont une véritable implémentation matérielle de l'exécution logicielle. Les systèmes basés processeur comprennent souvent plusieurs couches d'abstraction, pour aider à la planification des tâches et à la répartition des ressources entre les différents processus. La couche de driver contrôle les ressources matérielles et le système d'exploitation gère la mémoire et la bande passante du processeur. Sur chaque cœur de processeur, une seule instruction peut s'exécuter à la fois ; c'est pourquoi les systèmes basés processeur risquent toujours de voir des tâches prioritaires entrer en conflit. Les FPGA, qui n'utilisent pas de système d'exploitation, minimisent les problèmes de fiabilité car ils assurent une exécution véritablement parallèle et un matériel déterministe dédié à chaque tâche [8].

II.1.5. Maintenance à long terme

Comme nous l'avons vu, les circuits FPGA sont évolutifs et nous épargnent donc la dépense de temps et d'argent qu'implique la préconception des ASIC. Les spécifications des protocoles de communication numériques, par exemple, évoluent avec le temps. Or les interfaces basées sur ASIC peuvent poser des problèmes de maintenance et de compatibilité. Comme ils sont reconfigurables, les circuits FPGA sont capables de s'adapter aux modifications éventuellement nécessaires. À mesure qu'un produit ou qu'un système évolue, on peut y intégrer des améliorations fonctionnelles sans perdre de temps à reconcevoir le matériel ou à modifier l'implantation du circuit [8].

II.2. sortie VGA

Le DSP plate-forme de démarrage Spartan-3A comprend une sortie vidéo VGA à l'aide d'un réseau diviseur à résistances et 4 bits par couleur RGB comme le montre la Figure II.1. Ce réseau diviseur à résistances est de 510, 1K, 2K, et 4K ohms pour chaque couleur. Les sorties des trois réseaux diviseur à résistances sont présentés au connecteur DB15 P1. Signaux de synchronisation horizontale et verticale sont également générés par le FPGA et fournis à P1. En outre, une horloge 25,175 MHz (résolution VGA) est ajoutée à la carte, l'alimentation directement au FPGA à RHCLK2 (P26) sur la banque 1. Cette horloge doit être utilisée dans le contrôleur de FPGA pour synchroniser la sortie et générer l'image et se synchronise [9].

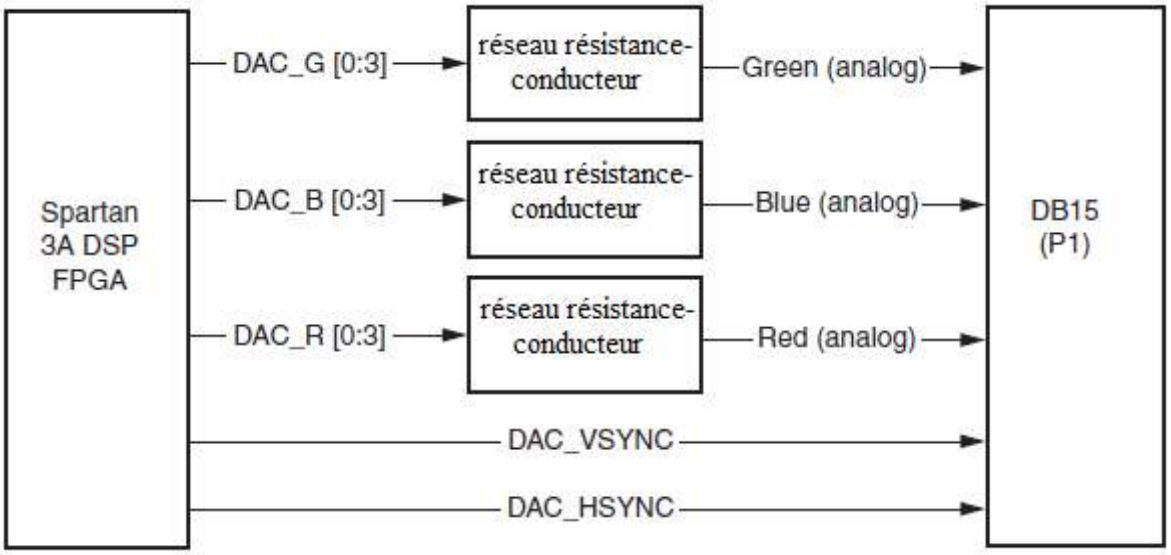


Figure II. 1. *Sortie VGA.*

III. La synchronisation VGA

Nous avons commencé notre travail par comprendre le principe VGA, comment y afficher sur un écran à tube cathodique ?, Cette étape consiste à piloter un écran VGA par des réalisations simples. Afficher des couleurs sous différentes formes : même couleur, bandes verticales, bandes horizontales, diviser l'écran sur quatre.

La Figure II.2 definie le circuit qui génère la synchronisation horizontale et verticale et donne des couleurs à chaque pixel.

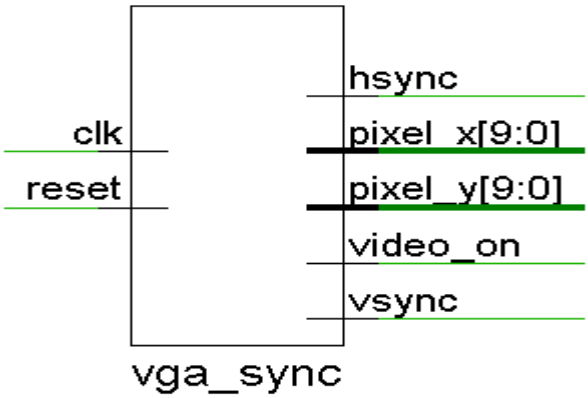


Figure II. 2.*Les entres/sorties de circuit de synchronisation.*

L'entite correspondante a ce circuit (Figure II.2)

```
entity vga_sync is
 port(clk, reset : in std_logic;
       sync_H ,sync_V :out std_logic;
       r,g,b:out std_logic_vector(3 downto 0)
      );
end vga_sync;
```

L'organigramme générale est donner dans la Figure II.3. tell que

Vcount: compteur verticale. ***Hcount:*** compteur horizontale.

Vsync: signal de synchronisation verticale. ***Hsync:*** signal de synchronisation horizontale

Vcount_y : compteur zone d'affichage Y. ***hcount_x :*** compteur zone d'affichage X.

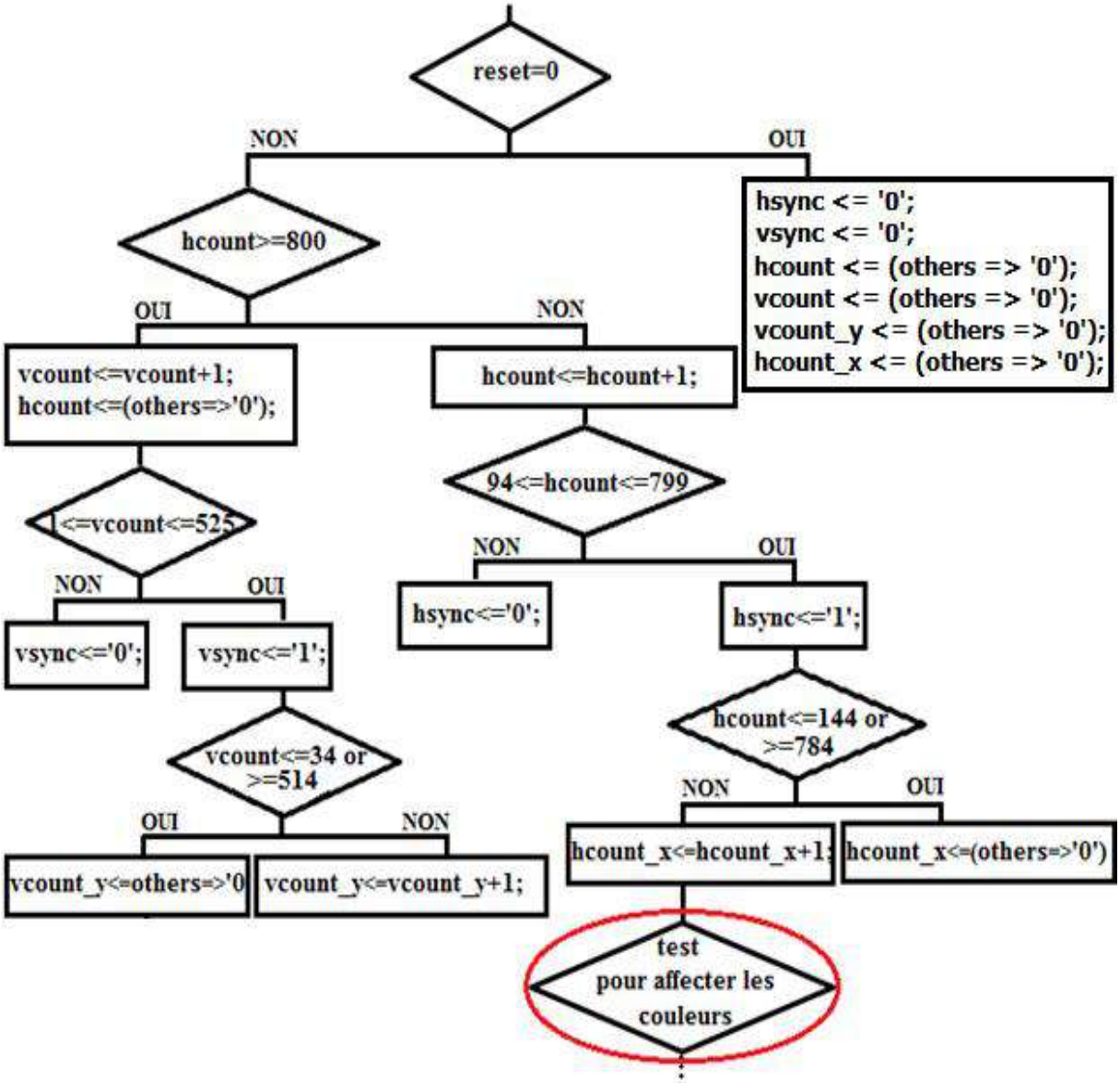


Figure II.3. *L'organigramme général du circuit de synchronisation.*

III.1. *Affichage d'une seule couleur*

La première étape est d'afficher la même couleur sur tout l'écran (on a choisi la couleur rouge).

L'organigramme est présent dans la (FigureII.4) et leur simulation est dans (Figure II.5).

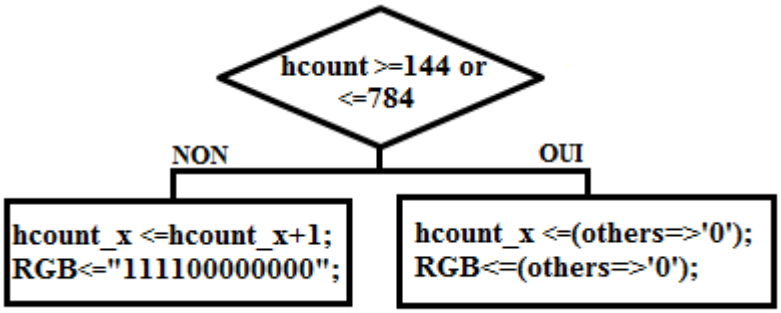


Figure II. 4. *L'organigramme d'affichage de la même couleur.*

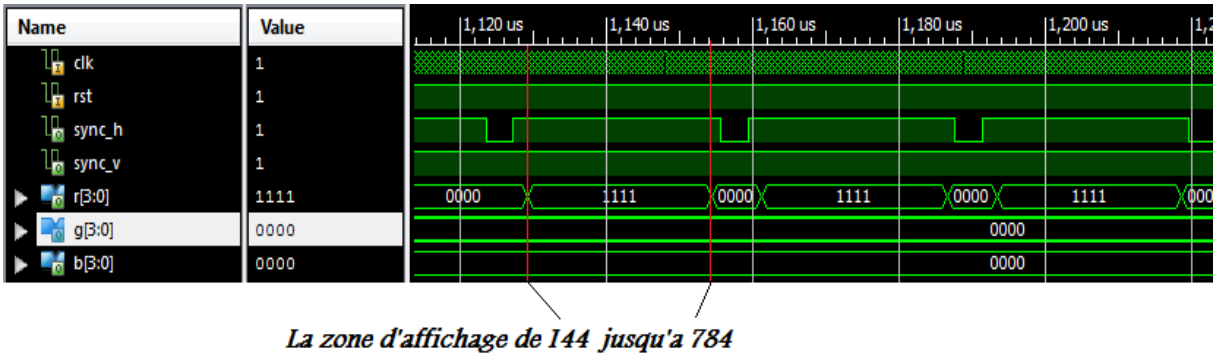


Figure II. 5. *Simulation du même coller.*

Le resultat de cette étape est représenté par (la Figure II.6).



Figure II.6. *Affichage d'une couleur sur tout l'écran.*

III.2. *Affichage deux bandes verticales*

Ensuite on a divisé l'écran sur deux bandes verticales de différentes couleurs. Le résultat de simulation (La Figure II.8) montre que lorsque le hcount_x arrive à 319 (0100111111 en binaire) le RGB change sa couleur comme montre l'organigramme suivant (Figure II.7), et l'affichage sur l'écran dans (la Figure II.9).

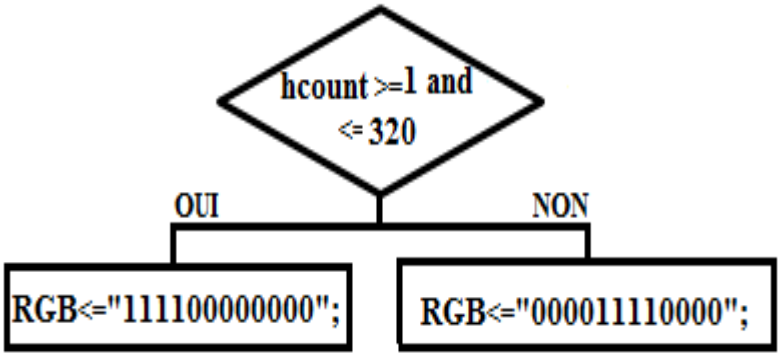


Figure II.7. *L'organigramme d'affichage de deux bandes verticales.*

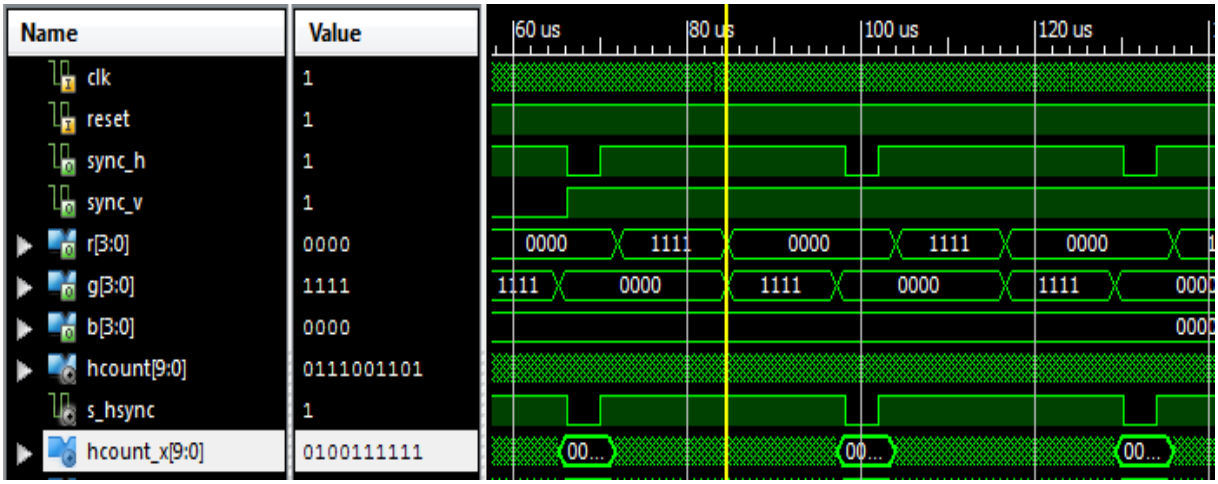


Figure II. 8. *Simulation de deux bandes verticales.*

L'affichage est le suivant (Figure II.9).



Figure II. 9. *Affichage de deux bandes verticales sur tout l'écran.*

III.3. *Affichage deux bandes horizontal à dégradés verticales*

Dans cette étape nous divisons la zone affichable sur deux bandes horizontales, chaque bande est divisée sur 16 pour obtenir des niveaux de 0 à 15 (de R (rouge)=1111 à 0000) verticales, même chose pour le vert G. Les figures suivantes (Figure II.10 et Figure II.11) représentent la simulation.

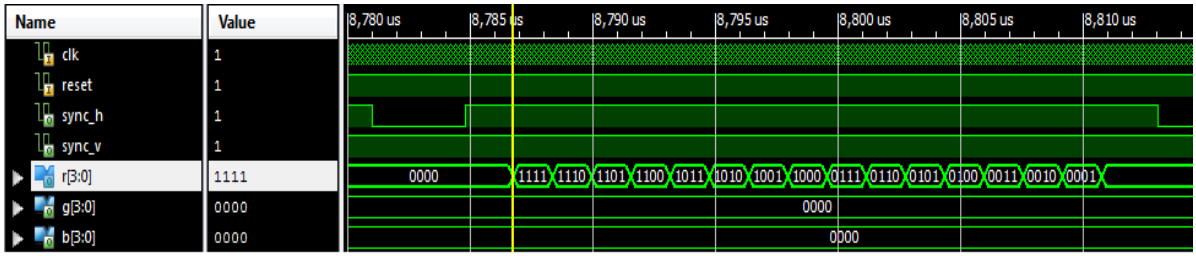


Figure II. 10. *Simulation horizontale dégradée pour R*

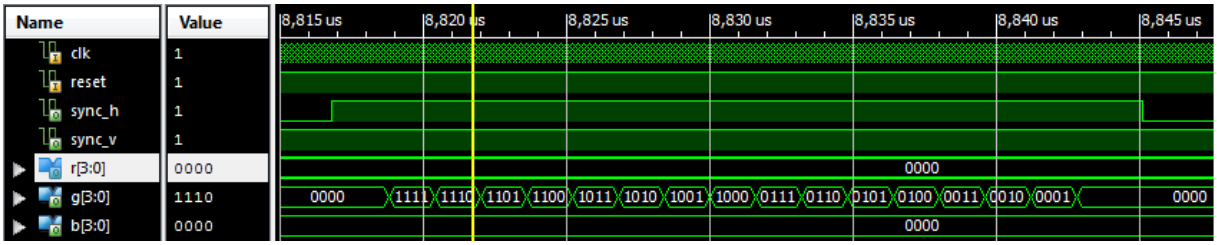


Figure II. 11. *Simulation horizontale dégradée pour V*

L'affichage est le suivant (Figure II.12).



Figure II. 12. *Affichage de deux bandes horizontales à dégradation verticales.*

III.4. *Affichage de deux bandes horizontales dégradées*

Dans cette étape nous divisons la zone affichable sur deux bandes horizontales, chaque bande est divisée sur 16 pour obtenir des niveaux de 0 à 15 (de R (rouge)=1111 à 0000) horizontale, même chose pour le vert G. La Figure II.13 représente l'affichage obtenu.

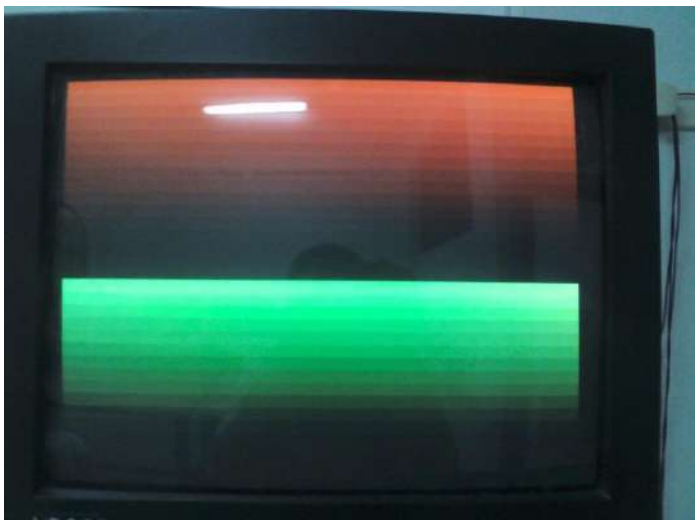


Figure II.13. *Affichage de deux bandes horizontales dégradées.*

III.5. *Affichage de trois bandes sans dégradation*

Nous avons divisé la zone affichable sur trois bandes et donnons une couleur pour chaque bande. La première bande de hcount_x >= 0 jusqu'au hcount_x <= 214 est de couleur rouge et la deuxième bande de hcount_x>= 214 à hcount_x <= 427 est le vert, la troisième bande de hcount_x = 427 à hcount_x<= 640 est le bleu. L'organigramme montre toutes ses valeurs (Figure II.14), et l'affichage est donne (Figure II.15).

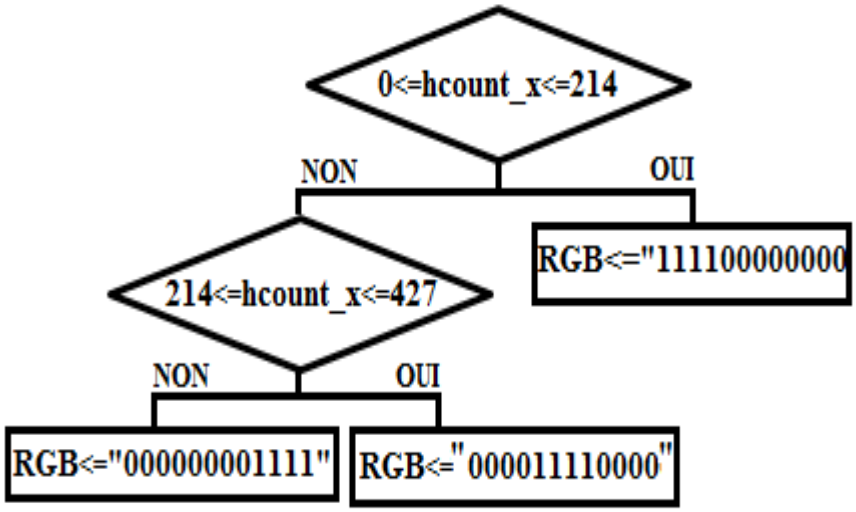


Figure II.14. *L'oganigramme d'affichage de trois bandes.*



Figure II.15. *Affichage de trois bandes colorées.*

III.6. *Affichage de trois bandes Avec dégradation*

Nous avons dégradé les couleurs de chaque bande verticalement sur 16 colonnes le résultat de simulation figuré dans la Figure II.16.

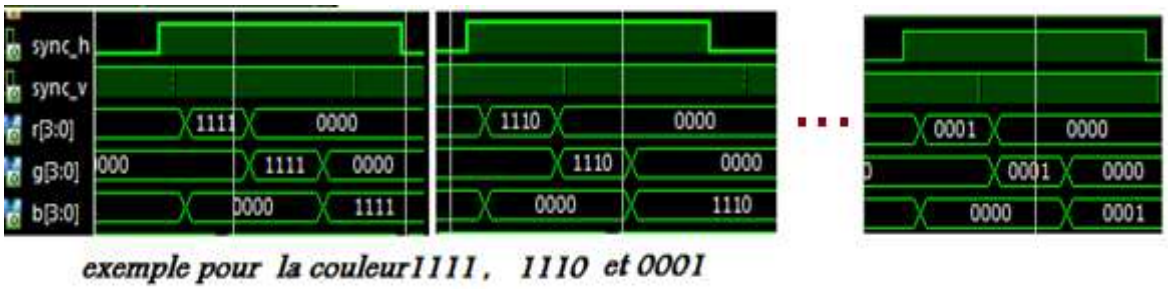


Figure II.16. *La Simulation de quelques exemples.*

L'affichage est représenté dans Figure II.17



Figure II.17.*Affichage de trois bandes dégradées verticalement.*

III.7. *Affichage de quatre zones*

La division de l'écran sur quatre zones, à chaque zone une couleur :

- La première partie à gauche de 0 à 320 (hcount_x) et de 0 à 240 (vcount_y) couleur rouge
- La deuxième partie à droite de 320 à 640 (hcount_x) et de 0 à 240 (vcount_y) couleur vert,
- La troisième partie gauche de 0 à 320 (hcount_x) et de 240 à 480 (vcount_y) couleur bleu,
- La Dernière partie couleur violet.

L'organigramme résume ses valeurs dans la figure suivante (Figure II.18), et la simulation se trouve dans la Figure (Figure II.19).

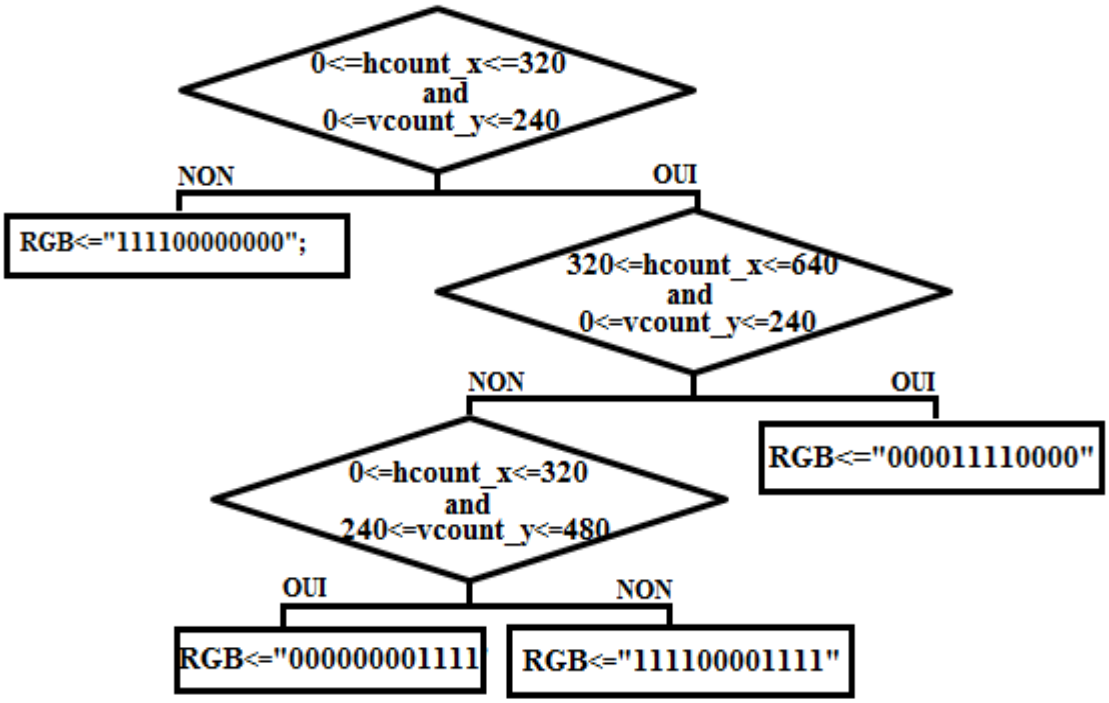


Figure II.18. *L'organigramme d'affichage de quatre zones.*

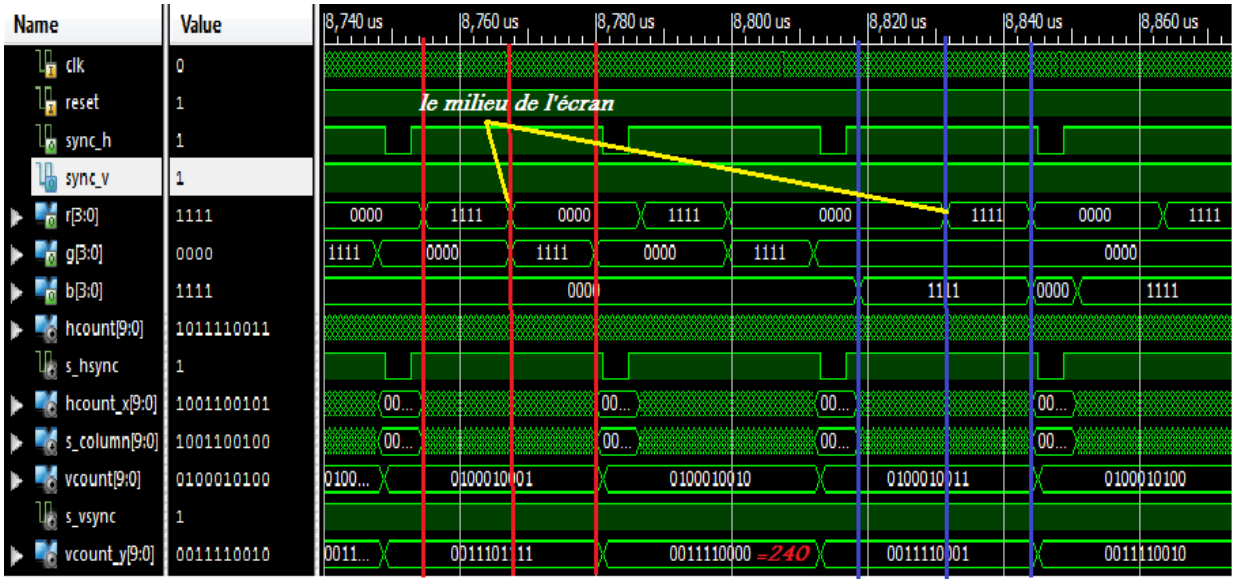


Figure II.19. *Simulation de quatre zones colorées.*

L'affichage sur l'écran dans la (Figure II.20).



Figure II.20. *Affichage de division de la zone d'affichage sur quatre.*

IV. conclusion

Dans ce chapitre nous avons parlé sur la génération des signaux des synchronisations ainsi les signaux RGB comme nous avons fait est l'affichage des couleurs défirent selon des manières différent avec le langage de description matériel VHDL.

CHAPITRE III: REALISATION DU JEU DE PONG

I. Introduction

Afin de bien comprendre et maitriser le contrôleur VGA par FPGA, on propose de réaliser un jeu implémenté entièrement sur une carte FPGA, on l'on peut jouer en visualisant sur un écran relié à la carte FPGA.

II. Cahier des charges

On a réalisé deux version du jeu de PONG en deux modes, mode d'un seul joueur et l'autre de deux joueurs le déroulement de première mode est commencé par l'affichage de la règle du jeu, la balle positionné au centre de l'écran, la palette est disposée à droite de l'écran et la durée de vie initialisé à trois. Le joueur doit renvoyer la balle avec la palette dès qu'il frappe la balle le score est incrément de un point, si le joueur rate la balle la durée de la vie est décrémenté de un jusqu'au épuisent des trois balles. La balle rebondit sur les bords supérieurs et inférieurs de la zone de jeu avec une certaine vitesse et à un certain angle.

Le déroulement du deuxième mode consiste à reenvoyer la balle avec la palette. Les palettes sont disposées à gauche et à droite de l'écran, la balle rebondit sur les bords supérieurs et inférieurs de la zone de jeu.

III. Jeu de PONG: mode un joueur

III.1. *Image Statique (fixe)*

On a débuté notre réalisation du jeu par fixer les objets de l'image (le mur, la balle et la palette) tell que la position du mur entre 32 et 35 horizontalement et la palette entre 600 et 603 horizontalement de taille 72 pixels centré au milieu d'écran et une balle carrée de taille 8 par 8 de pixels et colorer chaque objet avec une certaine couleur. La Figure III.1 représente l'image constituée.

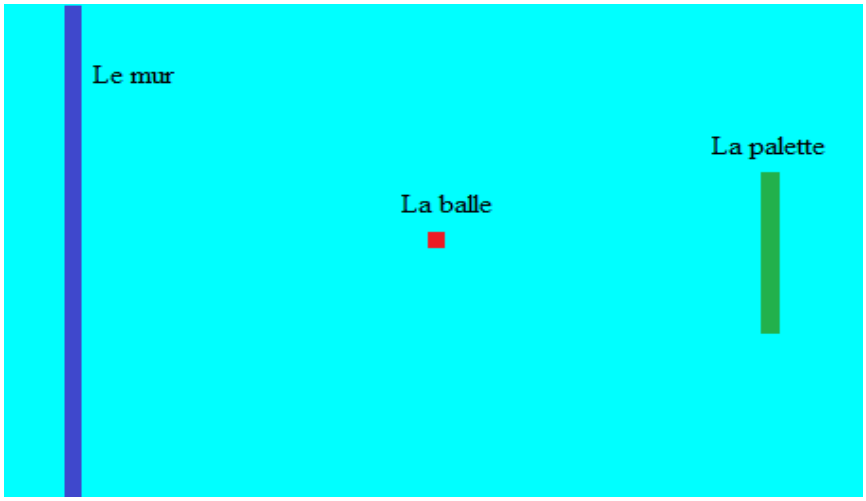


Figure III. 1. *L'image initiale.*

III.2. *Les objets dynamiques*

III.2.1. Gestion du déplacement de la palette

La palette est gérée par deux boutons poussoirs, son déplacement est réalisé selon l'axe des Y. un bouton pour monter la palette et l'autre pour descendre. En appuyant sur le bouton poussoir une seule fois la palette se déplace de 5 pixels c'est valeur qui gère la vitesse de la palette. La largeur de la palette est fixée à 4. Lorsque la limite supérieure de la palette atteint la limite supérieure de la zone d'affichage ou la limite inférieure de la palette atteint la limite inférieure de la zone d'affichage, la palette arrête l'incrémentation ou la décrémentation au bord de contact jusqu'à ce que l'utilisateur actionne le bouton poussoir dans le sens opposé, la Figure III.2 représente les deux cas.

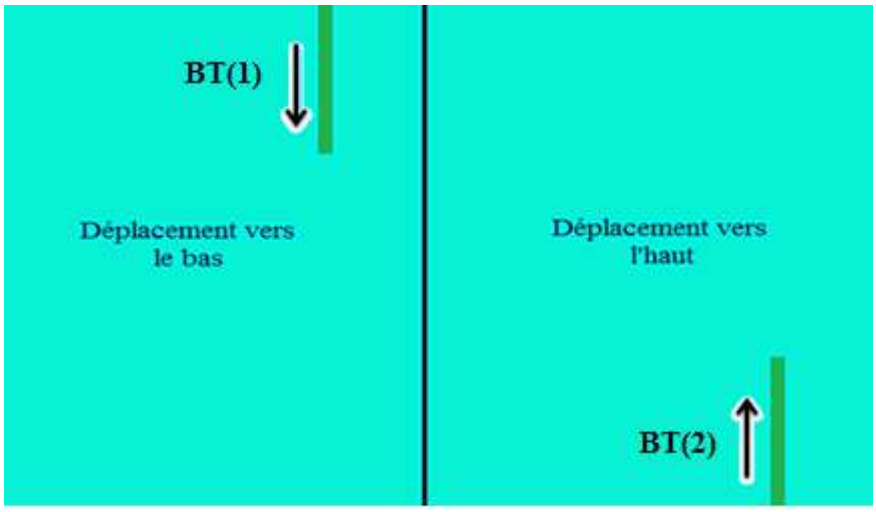


Figure III.2. *Les deux cas de contact de palette avec les bords.*

III.2.2. Gestion de déplacement de la balle

III.2.2.1. Image de la balle

Pour le stockage d'une image monochrome simple dans une ROM de taille 8 x 8, une rangée bidimensionnelle est suffisante. Les lignes de rangée représentent les lignes de l'image, et les index de ligne représentent différents pixel. En langage de description VHDL, cette rangée peut être décrite par une ROM comme suit:

```
type rom_type is array (0 to 7) of std_logic_vector (7 downto 0);
constant BALLE_ROM: rom_type :=( "00111100", --   ****
                                 "01111110", --  ******
                                 "11111111", -- ********
                                 "11111111", -- ********
                                 "11111111", -- ********
                                 "11111111", -- ********
                                 "01111110", --  ******
                                 "00111100"  --   ****
                               );
```

Si on veut une boule ronde, le processus devient alors:

- ✓ Vérifie si les coordonnées de balayage sont dans la place les 8 par 8 pixels.
- ✓ Si oui, on obtient le pixel correspondant de la table binaire.
- ✓ Si la valeur de table est égale à 1 alors on donne une couleur à notre choix, sinon on affecte la couleur du font du l'image.

La Figure III.3 représente la différence avant et après la conversion.

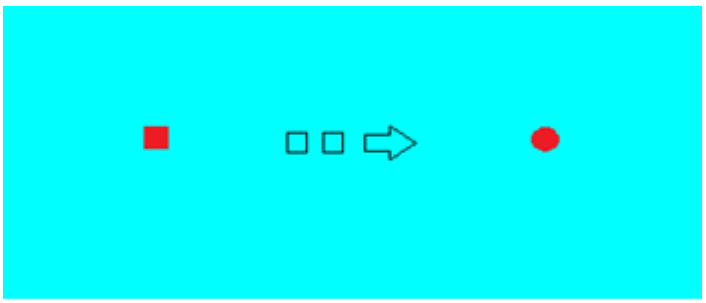


Figure III.3. *Conversion de la balle.*

III.2.2.2. Coloration de la balle

Après la lecture de la mémoire de la balle (BALLE_ROM), On a choisi de diviser la balle sur quatre et donner à chaque quart une couleur de notre choix comme suit en VHDL.

```
ball_rgb <="001" when rom_addr <=3 and rom_col<=3 else
           "100" when rom_addr <=3 and rom_col>=3 else
           "011" when rom_addr >=3 and rom_col<=3 else
           "101";
```

La Figure III.4 représente le résultat obtenu.

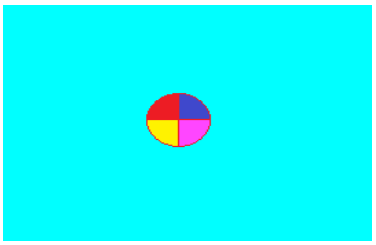


Figure III.4. *Balle colorée.*

III.2.2.3. Déplacement de la balle

La balle est placée au centre, elle est ensuite lancée lorsque le joueur appuie sur le bouton poussoir.

La balle peut changer la direction en frappant soit le mur, la palette, le fond, le dessus de l'écran. Nous décomposons la vitesse en x-composant et un y-composant.

Chaque composant peut avoir une valeur positive BALL_V_P ou une valeur négative BALL_V_N. Le déplacement est selon l'inégalité suivante :

Future position de la balle <=la position ancienne + la pente ;

La valeur courante de chaque composant est gardée dans des registres.

III.2.2.4. Contact avec les bordures

Le contact de la balle avec un bord de la zone affichable est repéré lorsque l'inégalité suivante n'est plus vérifiée :

Bord Droit d'Affichage(Mur) + taille_balle < position X < Bord Droit d'Affichage (palette) – taille_balle.

On détermine ensuite quel bord est responsable du contact avec la balle et on effectue une inversion de coordonnée du vecteur directeur afin de renvoyer la balle, tout en conservant l'angle d'incidence. Lorsque la balle est arrivée à la bordure haute de la zone affichable est repéré selon l'inégalité suivante:

La future position de la balle <= la position ancienne + la pente négative.

Lorsque la balle est arrivée à la bordure basse de la zone affichable ou elle est repérée selon l'inégalité suivante:

La future position de la balle <= la position ancienne + la pente positive.

La balle peut changer la direction en frappant le mur, la palette, ou le haut ou le bas de l'écran La Figure III.5 représente ces trois cas (le mur, le haut, le bas). (Même chose pour les sens inverses des cas présentés ci-après).

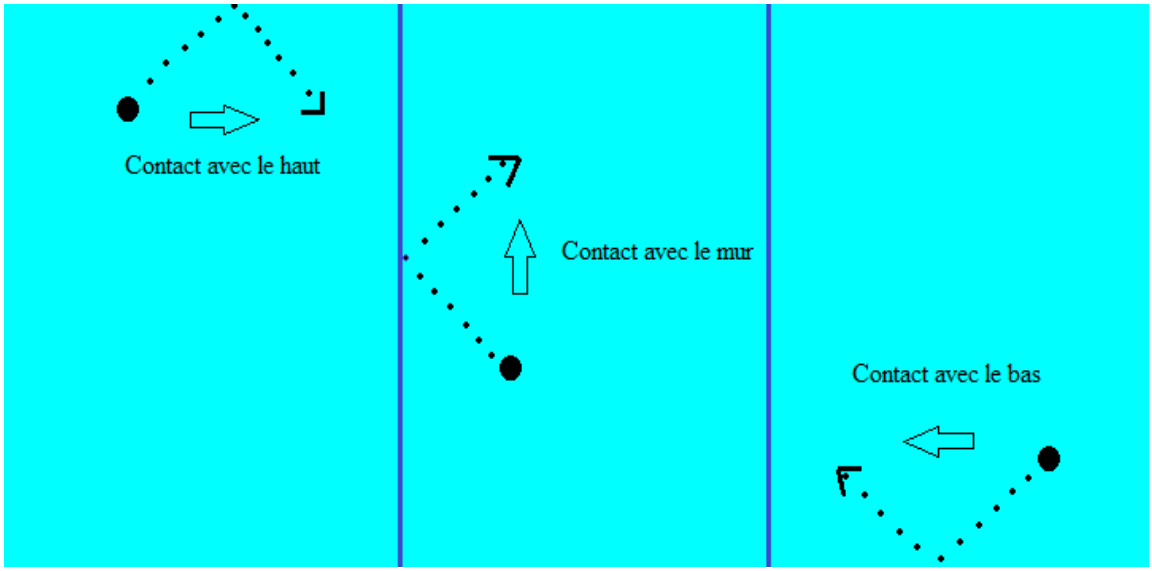


Figure III.5. *Le contact avec le haut, le bas et le mur.*

Le rebond de la balle sur la palette selon les principes donnés par la Figure III.6.

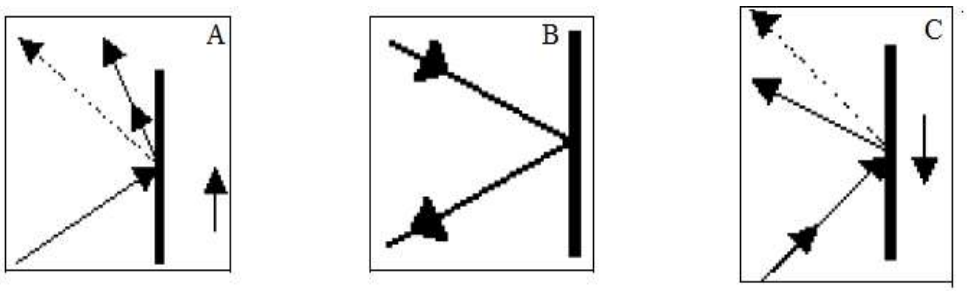


Figure III.6. *Les principes de rebondissement.*

A – BROSSAGE : Balle accélérée à trajectoire plus basse que la normale

B – NORMALE: Vitesse constante à trajectoire de réflexion normale.

C – LIFT: Balle ralentie à trajectoire plus haute que la normale.

Dans notre application on a choisi la vitesse constante (cas B).

D'une manière générale, lorsque l'on arrive sur les bords gauche et droit on inverse la composante en X de la pente et lorsque l'on arrive sur le bord haut ou bas on inverse la composante en y.

III.3. *Affichage de texte*

Il s'agit d'une ROM contenant le code ASCII de 128 caractères (Tableau III.1). Ce qui fait qu'on a besoin d'une ROM de taille 128 x 16 x 8 = 2k x 8. La Figure III.7 représente le circuit ROM qu'on a réalisé.

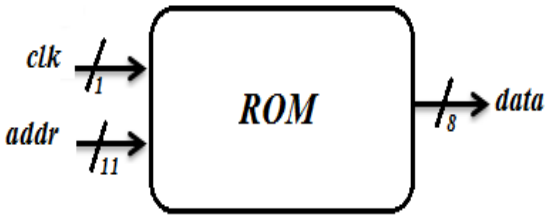


Figure III.7. *Les entrées /sorties de circuit ROM.*

L'entité correspondante à ce circuit (Figure III.7) est comme suite:

```
entity ROM is
   port(clk: in std_logic;
      addr: in std_logic_vector(10 downto 0);
      data: out std_logic_vector(7 downto 0)
   );
end ROM;
```

On a besoin d'un bus d'adresse à 11 bits car notre mémoire est de taille 2048 adresses ligne, et la taille totale 128 x 16 x 8 = 16384 bits (2k x 8).

Tableau III.1. *Code ASCII de 128 premiers caractères.*

Déc	Hex	Char	Déc.	Hex	Char	Déc.	Hex	Char	Déc.	Hex	Char
0	00	nulle	32	20	espace	64	40	@	96	60	`
1	01	début du titre	33	21	!	65	41	A	97	61	a
2	02	début de texte	34	22	"	66	42	B	98	62	b
3	03	Fin de texte	35	23	#	67	43	C	99	63	c
4	04	la fin de transmettent	36	24	$	68	44	D	100	64	d
5	05	enquête	37	25	%	69	45	E	101	65	e
6	06	Reconnaître	38	26	&	70	46	F	102	66	f
7	07	Cloche audible	39	27	‘	71	47	G	103	67	g
8	08	Espacement arrière	40	28	(	72	48	H	104	68	h
9	09	Tabulation horizontale	41	29	)	73	49	I	105	69	i
10	0A	Retour à la ligne	42	2A	*	74	4A	J	106	6A	j
11	0B	Tabulation verticale	43	2B	+	75	4B	K	107	6B	k
12	0C	Saut page	44	2C	,	76	4C	L	108	6C	l
13	0D	Retour de chariot	45	2D	-	77	4D	M	109	6D	m
14	0E	Décalage dehors	46	2E	.	78	4E	N	110	6E	n
15	0F	Décalez dedans	47	2F	/	79	4F	O	111	6F	o
16	10	Espace de liaison de données	48	30	0	80	50	P	112	70	p
17	11	Contrôle de l'appareil	49	31	1	81	51	Q	113	71	q
18	12	Contrôle de l'appareil	50	32	2	82	52	R	114	72	r
19	13	Contrôle de l'appareil	51	33	3	83	53	S	115	73	s
20	14	Contrôle de l'appareil	52	34	4	84	54	T	116	74	t
21	15	accusé de réception négatif	53	35	5	85	55	U	117	75	u
22	16	Lien de données évasion	54	36	6	86	56	V	118	76	v
23	17	fin du bloc de transmission	55	37	7	87	57	W	119	77	w
24	18	annulation	56	38	8	88	58	X	120	78	x
25	19	fin de support	57	39	9	89	59	Y	121	79	y
26	1A	substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	échappement	59	3B	;	91	5B	[	123	7B	{
28	1C	séparateur de fichier	60	3C	<	92	5C	\	124	7C	|
29	1D	séparateur de groupe	61	3D	=	93	5D	]	125	7D	}
30	1E	séparateur d'enregistrement	62	3E	>	94	5E	^	126	7E	~
31	1F	séparateur d'unité	63	3F	?	95	5F	_	127	7F	DEL

Chaque caractère est de taille 16 x 8, la Figure III.8 montre la forme de caractère « S ». Pour lire ce caractère il suffit d'écrire son adresse hexadécimale « x53 » en binaire dans le signal char_addr_sc de taille 7 bits comme montre l'instruction suivante.

```
char_addr_sc <=
   "1010011" when "0000", -- S x53
```

```
-- code x53
"00000000",  -- 0
"00000000",  -- 1
"01111100",  -- 2   *****
"11000110",  -- 3  **   **
"11000110",  -- 4  **   **
"01100000",  -- 5   **
"00111000",  -- 6    ***
"00001100",  -- 7      **
"00000110",  -- 8      **
"11000110",  -- 9  **   **
"11000110",  -- a  **   **
"01111100",  -- b   *****
"00000000",  -- c
"00000000",  -- d
"00000000",  -- e
"00000000",  -- f
```

Figure III.8. *Exemple du caractère « S » dans la Rom.*

III.4. *Circuit général du jeu PONG mode un*

La Figure III.9 représente le schéma du jeu PONG. Il composé de six circuits principaux.

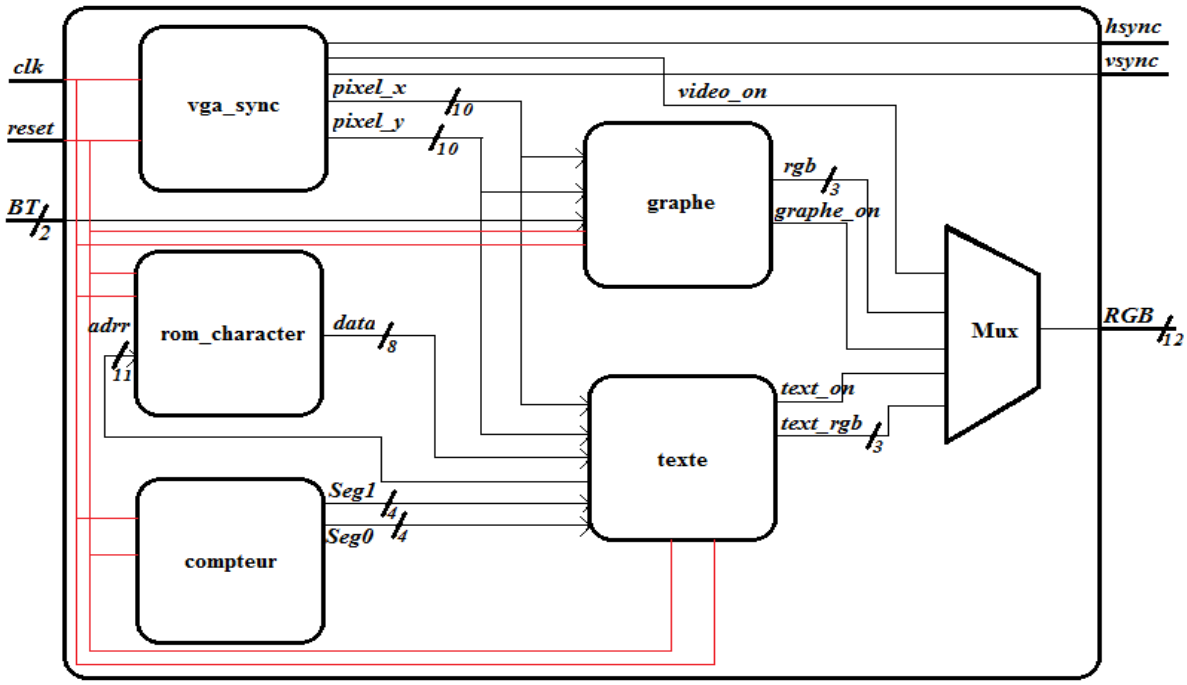


Figure III. 9. *Le circuit général du jeu du premier mode un PONG.*

III.4.1. Les circuits

III.4.1.1. Le circuit de La synchronisation

On a déjà parlé sur la synchronisation dans le chapitre II.

III.4.1.2. Le circuit de Texte

Afficher un texte à l'écran implique une mémorisation du message à afficher. Celui-ci sera constitué d'un tableau d'adresses pour les caractères. Il faudra aussi décider à priori de l'emplacement du ou des messages à afficher, éventuellement de la taille des caractères (on peut facilement multiplier par 2 ou par 4 …).

Le balayage VGA doit ensuite prendre en compte horizontalement (compteur colonne) et verticalement (compteur ligne) chaque pixel utile extrait du message relié au composant ROM. La Figure III.10 représente un schéma synoptique du circuit texte.

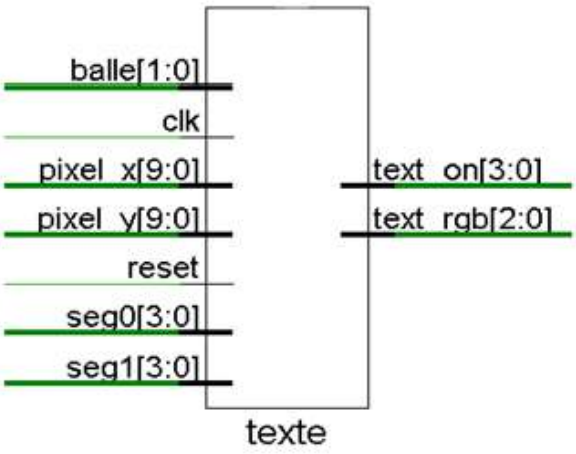


Figure III.10. *Les entrées/sorties de texte.*

Et l'entité correspondante à ce circuit est la suivante :

```
entity texte is
   port(clk, reset: in std_logic;
      pixel_x, pixel_y: in std_logic_vector(9 downto 0);
      seg0, seg1: in std_logic_vector(3 downto 0);
      balle: in std_logic_vector(1 downto 0);
      text_on: out std_logic_vector(3 downto 0);
      text_rgb: out std_logic_vector(2 downto 0)
   );
end texte;
```

Le signal d'entré « balle » est utilisé pour montrer le nombre de vie du jeu sur l'écran.

Le signal de sortie text_on est de quatre bits de 0 jusqu'à 3:

- ✓ Le bit « 0 » pour afficher le message de fin;
- ✓ Le bit « 1 » pour afficher la règle du jeu ;
- ✓ Le bit « 2 » pour afficher le slogan;
- ✓ Le dernier bit pour afficher le score et la durée de vie.

Le signal de sortie « text_rgb » donne un couleur à chaque texte affiché sur l'écran.

L'instruction qui va afficher les textes est :

```
text_on <= zone_score_on & slogan_on & regle_on & fin_on;
```

Le circuit "texte" a besoin d'un autre circuit il s'agit du circuit ROM donné dans la Figure III.11 suivante:

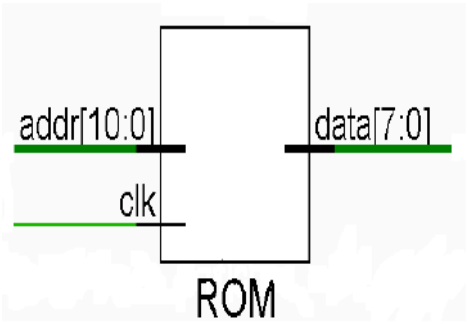


Figure III.11. *Les entres/sorties de ROM.*

La déclaration de component dans le circuit de texte pour hériter les données nécessaires est donnée comme suite:

```
U: entity work.ROM
   port map(clk=>clk, addr=>rom_addr,data=>font_word);
```

III.4.1.2.1. Les messages à afficher

En général, on peut écrire une expression qui a nous donnée la taille du chaque texte écrit sur l'écran comme suit :

Pix_y (N downto M) = 2^{N+1} = Valeur = 16 x facteur.

Pix_x (n downto m) = 2^{n+1} = valeur = 8 x facteur.

Tell que N et n représente le bit fort des compteurs (horizontale et verticale) du balayage.

Valeur et valeur représente le résultat du 2^{N+1} et 2^{n+1} respectivement.

Facteur est représenté la taille du texte à afficher, on a obtenu le par la division du Valeur/valeur sur la taille du caractère dans la mémoire (Valeur ÷ 16 et valeur ÷ 8).

III.4.1.2.1.1. La règle de jeu

Elle a été positionnée au centre de l'écran écrite sur quatre lignes comme le montre l'instruction suivante

```
regle_on <= '1' when pix_x(9 downto 7) = "010" and
                pix_y(9 downto 6)="0010"  else '0';
```

Le message est positionné entre la ligne 256 et 384 horizontalement et entre 128 et 192 verticalement.

La taille des caractères peut être précisée par l'instruction suivante qui lu les caractères du chaine sans modifier leurs taille (16 x 8).

```
Ligne_addr_r <= std_logic_vector(pix_y(3 downto 0));
bit_addr_r <= std_logic_vector(pix_x(2 downto 0));
```

Pix_y (3 downto 0) = $2^4 = 16$

Pix_x (2 downto 0) = $2^3 = 8$

On a lu les caractères directement de la mémoire sans modifier leurs tailles.

La règle est : « Utilisez deux boutons BT (1/2) pour déplacer la palette », et on l'a stocké dans REGLE_ROM comme suite:

```
type regle_rom_type is array (0 to 63) of
 std_logic_vector (6 downto 0);
constant REGLE_ROM: regle_rom_type:=
  ("1010101", -- U
   "1110100", -- t
   "1101001", -- i
   "1001100", -- L
   "1101001", -- i
   "1110011", -- s
   "1100101", -- e
   "1111010", -- z
   "0000000", --
   "1100100", -- d
   "1100101", -- e
   "1110101", -- u
   "1111000", -- x
   "0000000", --
   "0000000", --
   "0000000", --
       :
       :
       :
```

La Figure III.12 montre le résultat affiché sur l'écran.

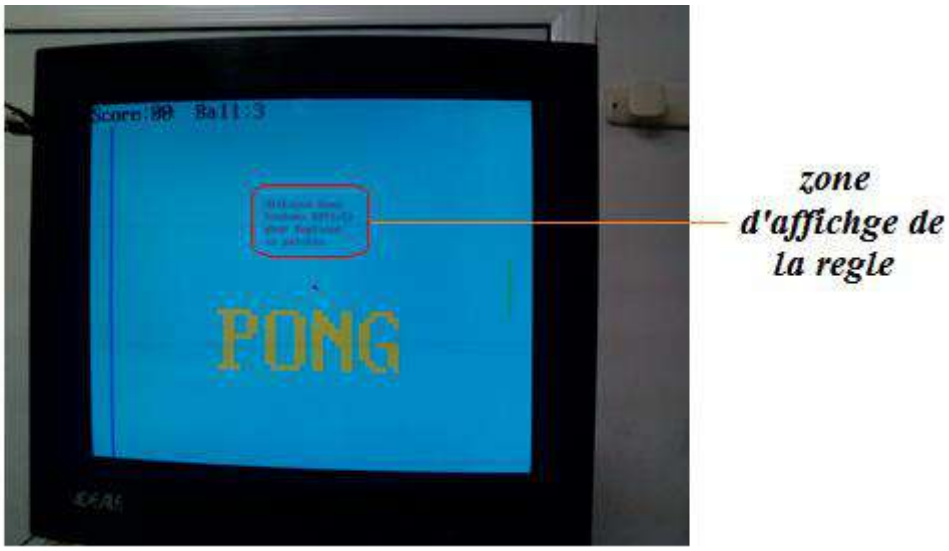


Figure III.12. *Affichage de la règle.*

La Figure III.13 montre la règle mode 1

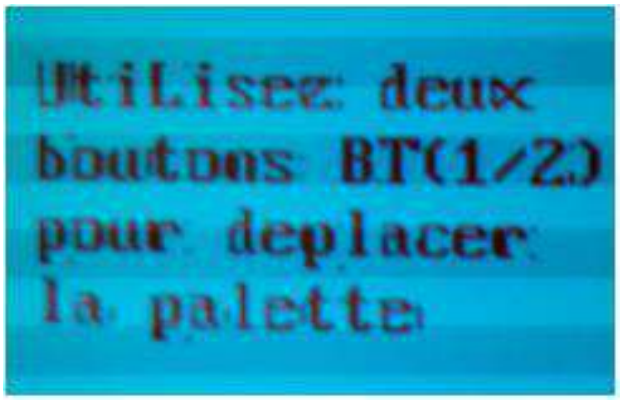


Figure III.13. *Message de la règle affiché du 1er mode*

III.4.1.2.1.2. Score et durée de vie

Le score et la durée de vie ont été positionnés en haut gauche de l'écran, à l'aide l'instruction VHDL comme suit

```
zone_score_on <=
   '1' when pix_y(9 downto 5)=0 and
            pix_x(9 downto 4)<16 else '0';
```

La position des caractéres est entre 0 et 256 horizontalement et entre 0 et 15 verticalement.

La taille choisie est comme suite :

```
Ligne_addr_sc <= std_logic_vector(pix_y(4 downto 1));
bit_addr_sc <= std_logic_vector(pix_x(3 downto 1));
```

Pix_y (4 downto 1) = 2^5 = 32 = 16 x 2.

Pix_x (3 downto 1) = 2^4 = 16 = 8 x 2.

La taille des caractères lu est multiplié fois deux.

Les adresses de cette chaine des caractères « Score : 00 Ball : 3 » affichée est:

```
with pix_x(7 downto 4) select
  char_addr_score <=
     "1010011" when "0000", -- S x53
     "1100011" when "0001", -- c x63
     "1101111" when "0010", -- o x6f
     "1110010" when "0011", -- r x72
     "1100101" when "0100", -- e x65
     "0111010" when "0101", -- : x3a
     "011" & seg1 when "0110",
     "011" & seg0 when "0111",
     "0000000" when "1000",
     "0000000" when "1001",
     "1000010" when "1010", -- B x42
     "1100001" when "1011", -- a x61
     "1101100" when "1100", -- l x6c
     "1101100" when "1101", -- l x6c
     "0111010" when "1110", -- :
     "01100" & balle when others;
```

L'affichage du "score" et "balle" est fixe mais l'affichage seg1, seg0 et balle est dynamique.

Seg1, ***seg0***: géré par le circuit compteur, les deux signaux est de type std_logic_vector de 4 bits pour montrer le nombre des points gagnés.

Balle: est un signal de type std_logic_vector de 2 bits pour montrer la durée de la vie, initialement égale à 3 ("11"). Il décrémente à chaque perte de la balle par un. Dès que la durée de vie est égale à zéro, le jeu il s'arrête.

L'affichage de ces caractères est donné par la Figure III.14.

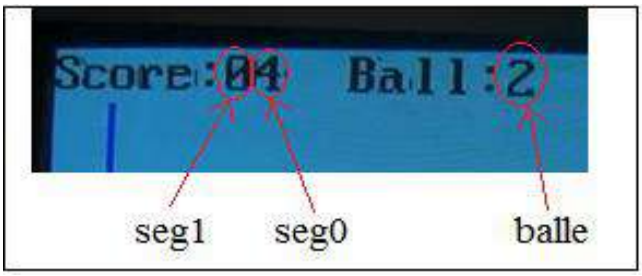


Figure III.14. *L'affichage sur l'écran du score et duré de la vie.*

La Figure III.15 montre le résultat obtenu sur un écran:

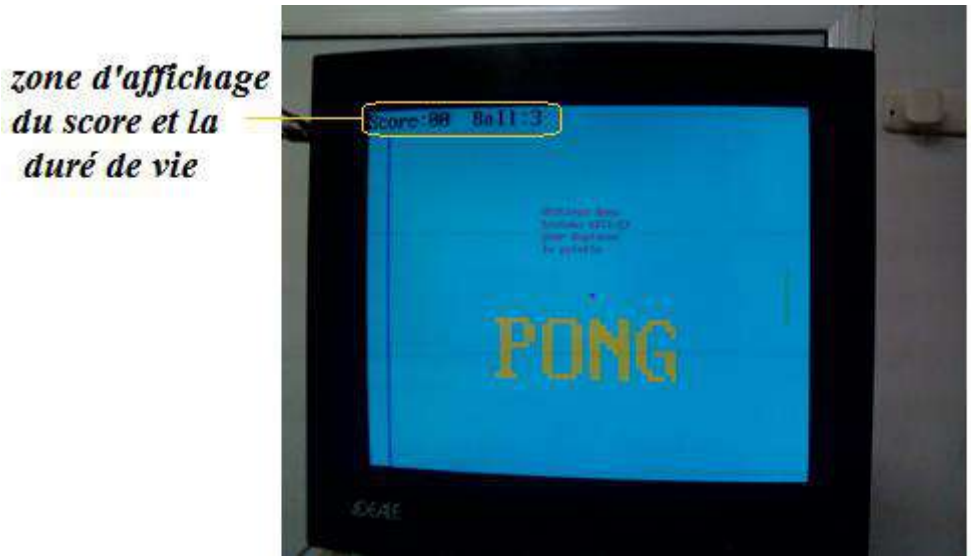


Figure III. 15. *Zone d'affichage du score et la durée de vie.*

III.4.1.2.1.3. Slogan

Il est affiché au milieu bas de l'écran entre 256 et 320 (pix_x (9 downto 7) = 2 => 256) plus la taille du caractère qui égale à 64 montré comme suite

```
slogan_on <=
   '1' when pix_y(9 downto 7)=2 and (3<= pix_x(9 downto 6)
            and pix_x(9 downto 6)<=6) else '0';
```

On a agrandi la taille des caractères huit fois la taille des caractères dans la ROM.

Pix_y (6 downto 3) = 2^7 = 128 = 16 x 8.

Pix_x (5 downto 3) = 2^6 = 64 = 8 x 8.

Comme montre l'instruction suivante,

```
Ligne_addr_sl <= std_logic_vector(pix_y(6 downto 3));
bit_addr_sl <= std_logic_vector(pix_x(5 downto 3));
```

Le slogan est « PONG », présente comme suit :

```
with pix_x(8 downto 6) select
  char_addr_slogan <=
     "1010000" when "011", -- P x50
     "1001111" when "100", -- O x4f
     "1001110" when "101", -- N x4e
     "1000111" when others; --G x47
```

La FigureIII.16 illustre le résultat obtenu

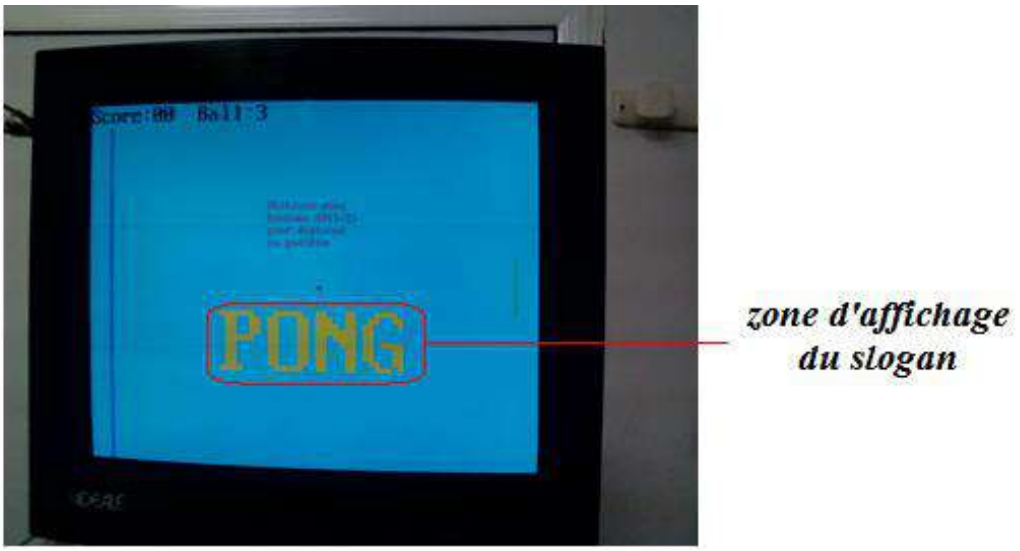


Figure III.16. *Zone d'affichage de du slogan.*

III.4.1.2.1.4. La fin de jeu

Lorsque le jeu est terminé ou bien la durée de vie est égale à zéro le message « Game Over» s'affiche au centre de l'écran entre 192 et 258 verticalement, et sa position est donnée par l'instruction suivante:

```
fin_on <='1' when pix_y(9 downto 6)=3 and 5<= pix_x(9 downto 5)
             and pix_x(9 downto 5)<=13 else '0';
```

On a agrandie la taille des caractères quarte fois la taille des caractères dans la ROM.

Pix_y (5 downto 2) = 2^6 = 64 = 16 x 4.

Pix_x (4 downto 2) = 2^5 = 32 = 8 x 4.

Comme montre l'instruction suivante,

```
Ligne_addr_F <= std_logic_vector(pix_y(5 downto 2));
bit_addr_F <= std_logic_vector(pix_x(4 downto 2));
```

Les adresses de ces caractères sont :

```
with pix_x(8 downto 5) select
  char_addr_F <=
     "1000111" when "0101", -- G x47
     "1100001" when "0110", -- a x61
     "1101101" when "0111", -- m x6d
     "1100101" when "1000", -- e x65
     "0000000" when "1001", --
     "1001111" when "1010", -- O x4f
     "1110110" when "1011", -- v x76
     "1100101" when "1100", -- e x65
     "1110010" when "1101", -- r x72
     "0000000" when others;
```

La Figure III.17 représente le message aussi que sa position.

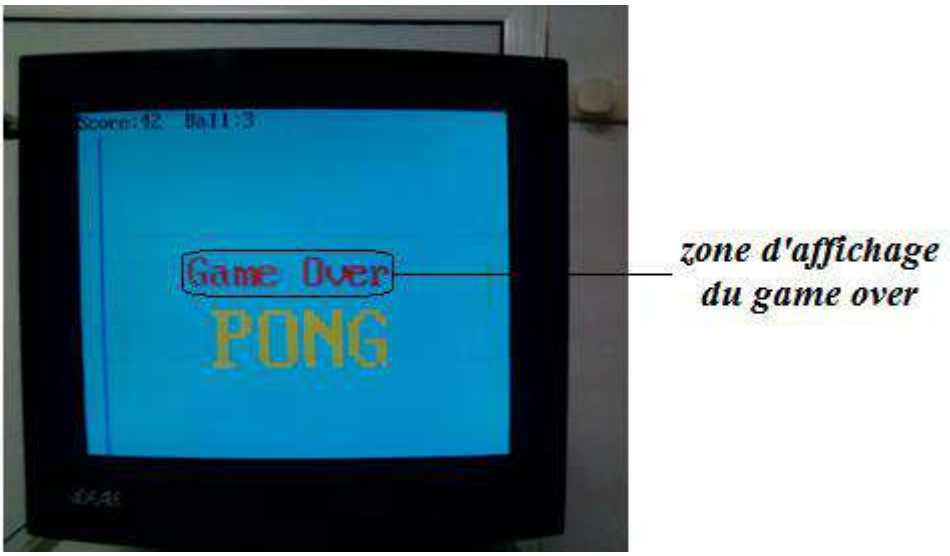


Figure III.17. *Zone d'affichage du game over.*

III.4.1.3. Le circuit de graphe

Le circuit "graphe" est responsable des objets affichées sur l'écran(le mur, la balle, et la palette) et leurs mouvements (la balle et la palette) la Figure III.18 modélise les entrées/sorties de graphe.

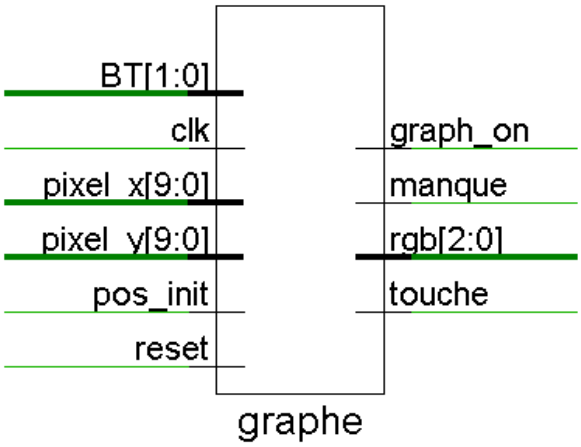


Figure III.18. *Les entrées/sorties du circuit graphe*.

L'entité peut être donnée comme suit,

```
entity graphe is
   port(clk, reset:in std_logic;
      BT:in std_logic_vector(1 downto 0);
      pixel_x,pixel_y: in std_logic_vector(9 downto 0);
      pos_init: in std_logic;
      graph_on,touche,manque: out std_logic;
      rgb: out std_logic_vector(2 downto 0)
   );
end graphe;
```

Pos_init: C'est un signal de commande .Quand on l'affirme, la barre verticale est placée au milieu de l'écran sans mouvement et la balle au centre de l'écran. La Figure III.18 donne l'état initial du jeu (lorsque le signal pos_init = 1).

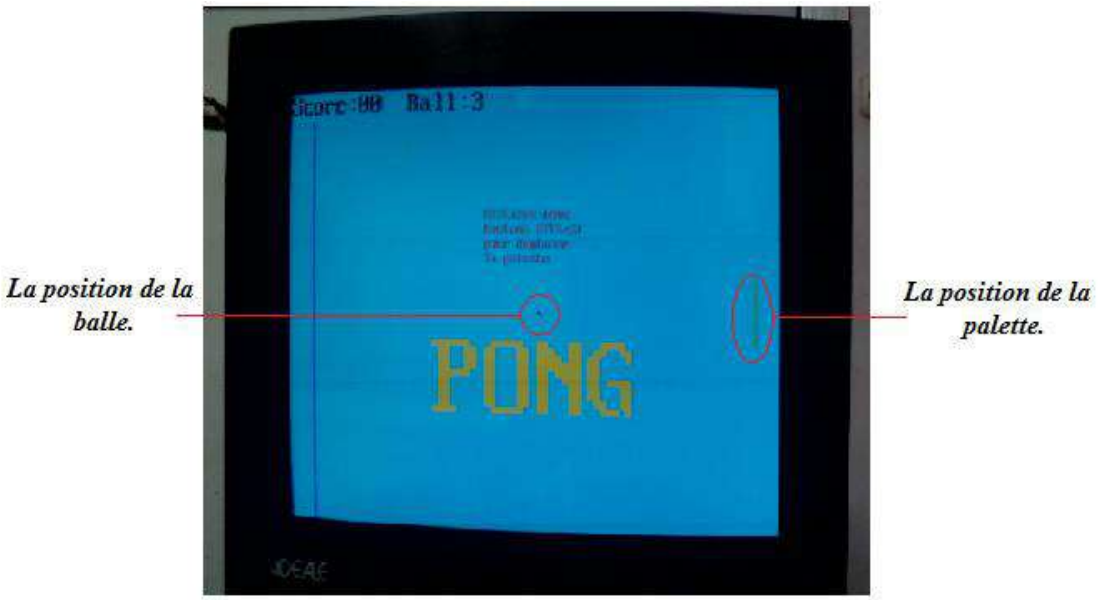


Figure III.19. *L'état initial de la belle et la palette.*

Touche et *manque* les signaux de statut:

- ✓Le signal *touche* est affirmé pour un rythme quand la palette frappe la balle.
- ✓Le signal *manque* est affirmé quand la balle dépasse la palette sans la toucher.

Le signal de sortie *graph_on* est utilisé pour indiquer un état actif du sous ensemble graphique (Le mur, la balle et la palette).

III.4.1.4. Le circuit de temps

Le module de temporisateur (circuit de temps) emploie une fréquence de 60hz, pour produire un certain intervalle de temps temps_j.

Son but est de faire une pause vidéo pour un petit intervalle entre les transitions de l'image.

La Figure III.20 montre les entres/sorties de circuit temps.

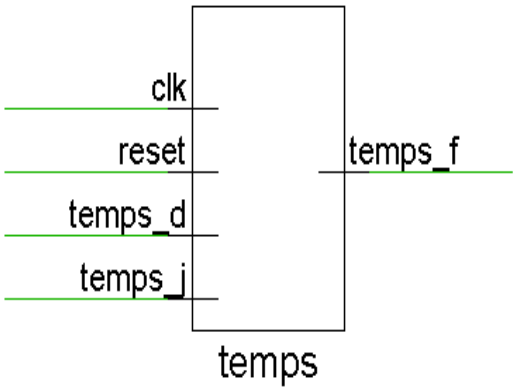


Figure III.20. *Les entrées/sorties du circuit temps.*

L'entité peut être donnée comme suit,

```
entity temps is
   port(clk, reset:in std_logic;
      temps_d,temps_j:in std_logic;
      temps_f: out std_logic
   );
end temps;
```

Temps_d, temps_j, temps_f : les trois signaux pour contrôler la pause entre les transitions du l'écran talle que :

- ✓ *Temps_d* : signal d'entré représente le début de pause;
- ✓ *Temps_j* : signal d'entré représente la durée de pause;
- ✓ *Temps_d* : signal de sortie représente la fin de pause.

III.4.1.5. Le circuit de compteur

Ce circuit est un compteur à deux chiffres de BCD comptant de 00 à 99, utilisé pour garder la trace des points du jeu lorsque la palette touche la balle, le compteur est incrémenté par 1. La Figure III.21 montre les entres/sorties de circuit compteur.

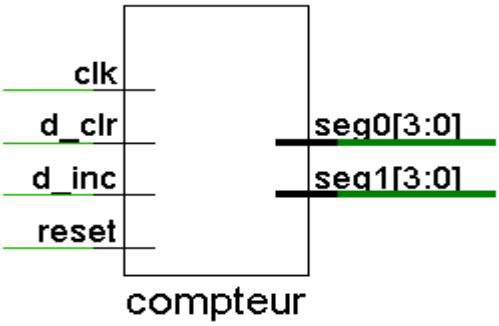


Figure III.21. *Les entrées/sorties de circuit compteur.*

L'entité correspondante est donnée comme suit :

```
entity compteur is
   port(clk, reset: in std_logic;
      d_inc, d_clr: in std_logic;
      seg0,seg1: out std_logic_vector (3 downto 0)
   );
end compteur;
```

d_clr : est un signal pour la rénitialiser le compteur à zéro.

d_inc : est un signal qui a chaque front montant de ce signal le compteur BCD incrément.

III.4.1.6. Le circuit de PONG

Le circuit de "pong" regroupe tous les circuits précédents, la Figure III.22 montre les entrées sorties de ce circuit.

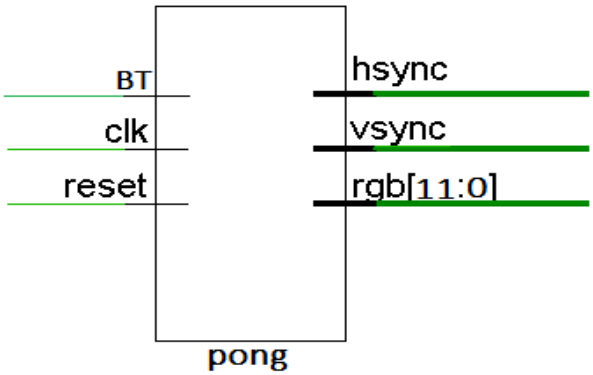


Figure III.22. *Les entrées/sorties de pong.*

L'entité correspondante à ce circuit est donnée comme suit :

```
entity pong is
   port(clk, reset: in std_logic;
      BT: in std_logic_vector(1 downto 0);
      hsync, vsync: out std_logic;
      rgb: out std_logic_vector (11 downto 0)
   );
end pong;
```

L'hiérarchie de tous les circuits est montré par la Figure III.23.

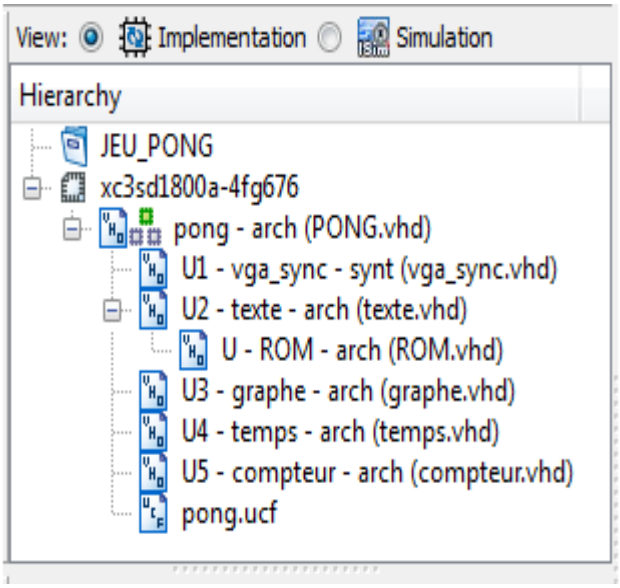


Figure III.23. *Hiérarchie de jeu de PONG.*

III.4.1.6.1. La machine d'état

Ce circuit contient une machine d'état représente les états de jeu de PONG :

III.4.1.6.1.1. L'état de démarrage

Cet état est nommée NVpartie elle permet le démarrage d'une nouvelle partie du jeu, elle permet d'initialiser le jeu par trois balles et le score par zéro lorsque le bouton (BT) est touché on bascule vers l'état suivante.

III.4.1.6.1.2. L'état jouer

Lorsque le nombre de balle est différant de zéro le jeu se déroule, si la balle frappe la palette le score s'incrémente de 1. Si le joueur rate la balle, la durée de vie décrémente et le jeu prend un pause, on bascule vers l'état NVballe tant que le nombre des balles est différant de zéro sinon vers l'état fin.

III.4.1.6.1.3. L'état de nouvelle balle

Dans cet état on attend un certain temps jusqu'à le joueur touche le bouton du jeu pour recommencer la partie.

III.4.1.6.1.4. L'état fin du jeu

Lorsque le nombre de balle est égale à zéro, on est à l'état fin du jeu et on passe à l'état NVpartie en appuyant sur un BT.

La Figure III.24 montre la machine d'état décrite.

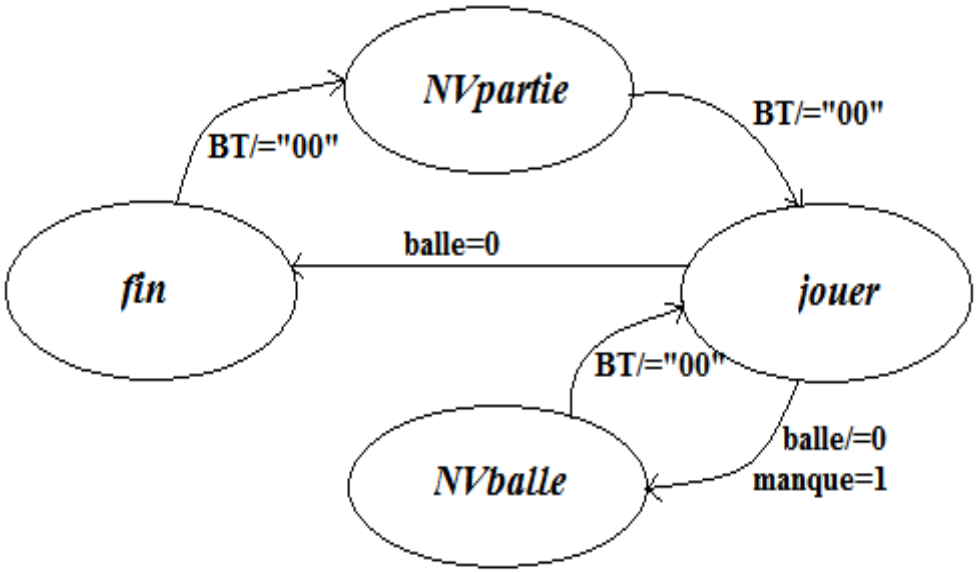


Figure III.24. *La machine d'état.*

La Figure III.25 donne l'organigramme qui résume le fonctionnement interne entre les circuits lorsque le signal pos_init = 1, on est dans l'état initial, si on appuie sur un bouton la valeur de pos_init devienne 0, et le jeu commence. Si le nombre des balles est égal à zéro on arrive à l'état fin sinon le jeu recommence avec une nouvelle balle.

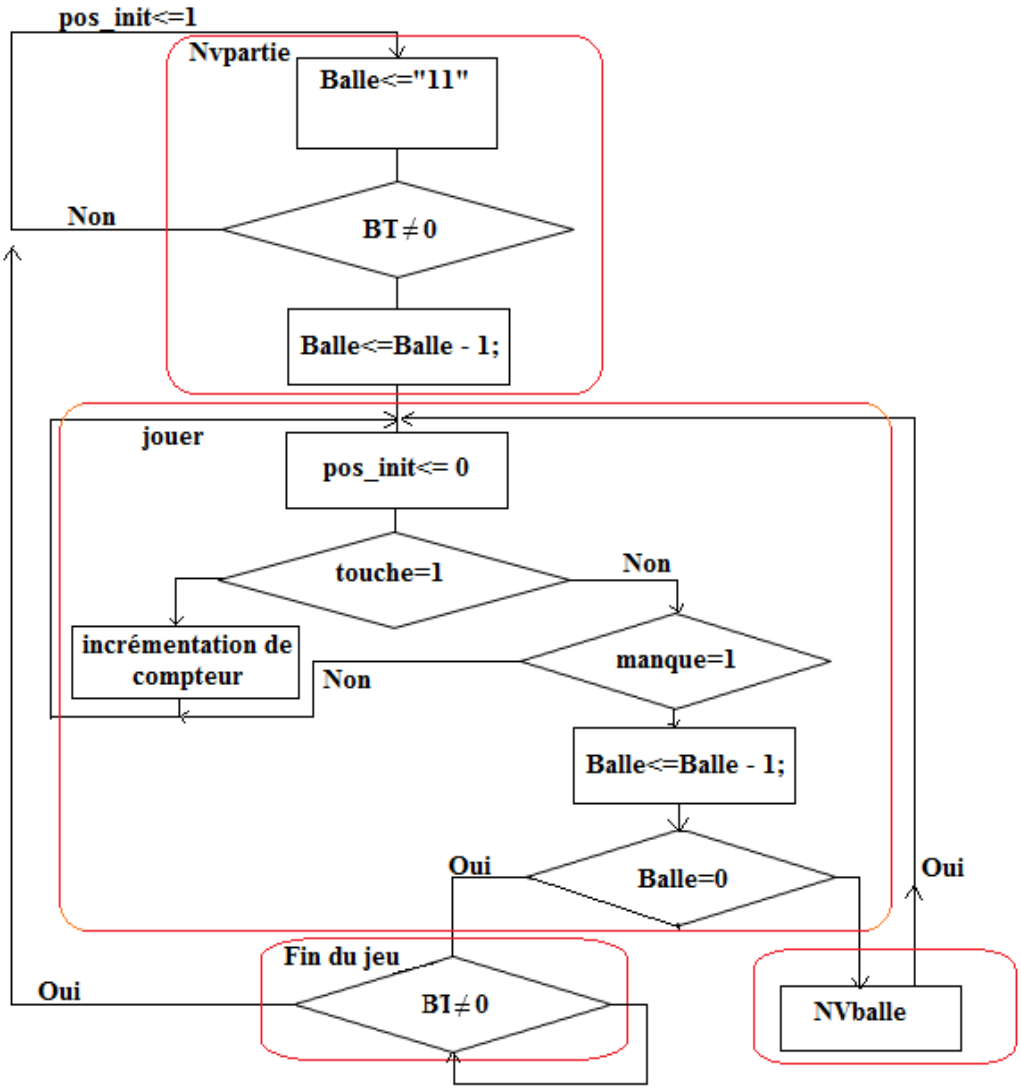


Figure III.25. *L'organigramme général.*

III.4.2. La résumé du déroulement du jeu

Le principe du jeu de PONG est résumé par les points suivants:

- ✓Quand le jeu commence il montre le texte ainsi que la règle sont affichés;
- ✓Après qu'un joueur appuie sur un bouton le jeu commence;
- ✓Le joueur marque un point du frappant chaque fois la balle avec la palette;
- ✓Quand le joueur manque la balle, le jeu fait une pause et la nouvelle balle est fournie avec décrémentation du nombre de vie par 1;
- ✓Trois balles sont fournies en chaque session;
- ✓Les points et le nombre de balles restantes sont montrés sur le dessus de l'écran;
- ✓Après que le manque 3, le jeu se terminé avec message « Game Over ».

III.5. *Implémentation*

On a synthétisé notre travail par le logiciel ISE de Xilinx on utilisant la carte 3SD1800A-FG676 FPGA qui possède une fréquence de 125 MHz.

Voici le processus d'implémentation présenté dans la Figure III.26 suivante

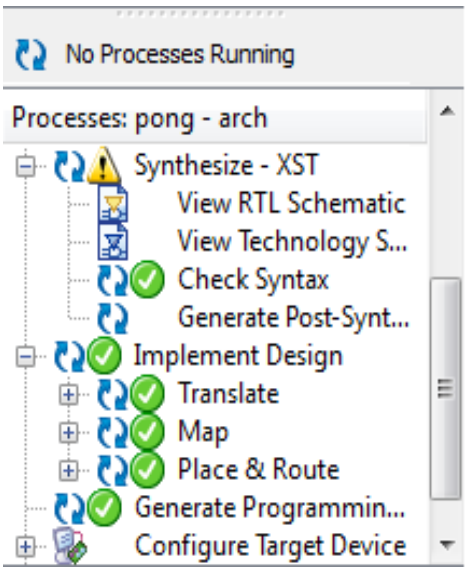


Figure III.26. *Processus d'exécution.*

Et le rapport d'implémentation est donné par le (Tableau III.2).

Tableau III.2. *Rapport d'implémentation.*

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	135	33,280	1%	
Number of 4 input LUTs	519	33,280	1%	
Number of occupied Slices	287	16,640	1%	
Total Number of 4 input LUTs	535	33,280	1%	
Number of bonded IOBs	18	519	3%	
Number of BUFGMUXs	1	24	4%	
Number of RAMB16BWERs	1	84	1%	

IV. Jeu de PONG : mode deux joueurs

IV.1. *Image fixe*

Dans cette partie on a essayé de compliquer un peu le jeu, dans ce mode il y'a deux palettes, une pour chaque joueur. La première palette est à gauche entre 32 et 35 horizontalement et l'autre entre 600 et 603 et les deux de taille 72, la position de la balle est au centre de l'écran (Figure III.27).

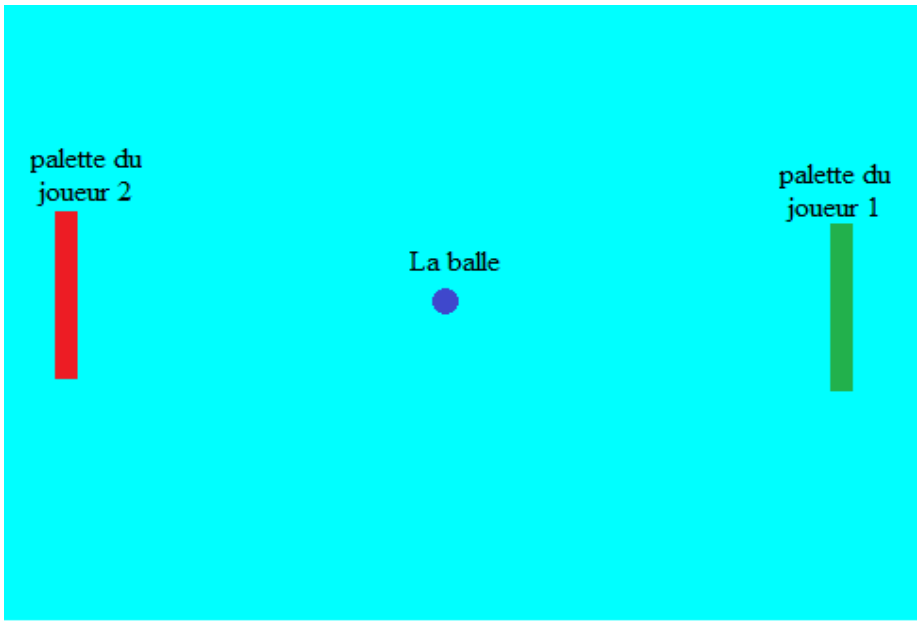


Figure III.27. *L'image initiale.*

IV.2. *les objets dynamique*

Le principe de déplacement est le même avec le jeu précédant, soit la balle ou la palette sauf dans ce jeu on a deux palettes qui bougent selon les boutons appuyés.

IV.3. *Circuit général du jeu PONG (mode deux)*

Ce circuit représente le schéma général du jeu PONG comme le montre la Figure III.28 suivante:

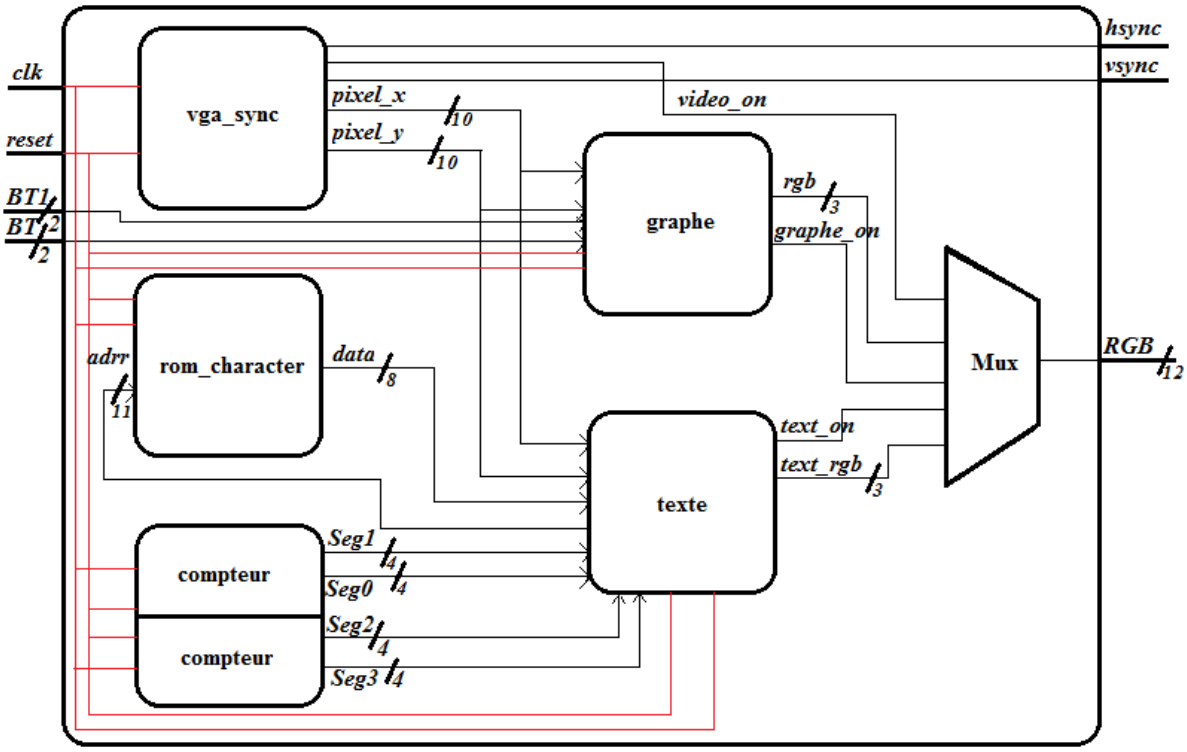


Figure III.28. *Circuit général mode deux.*

IV.4. *Les modification des circuits*

Tous les circuits de ce jeu sont les même avec le jeu précédant et le seul changement réside dans les circuits PONG, graphe et texte.

IV.4.1. Le circuit de PONG

Dans ce circuit on a ajouté le BT1 comme une entrée à deux bit dans l'entité qui gère le déplacement de la deuxième palette (Figure III.29), et une autre composante du circuit compteur pour les calculs du deuxième score de l'autre joueur comme le présente la hiérarchie dans la Figure III.30.

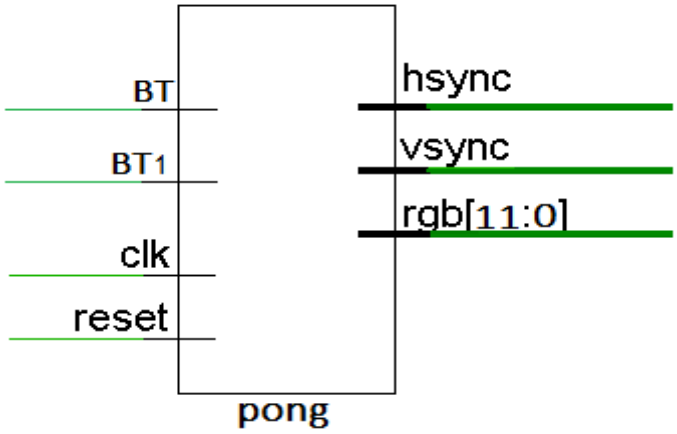


Figure III.29. *Les entrées/sorties du circuit de pong.*

La hiérarchie de ce circuit est donnée par la Figure III.30.

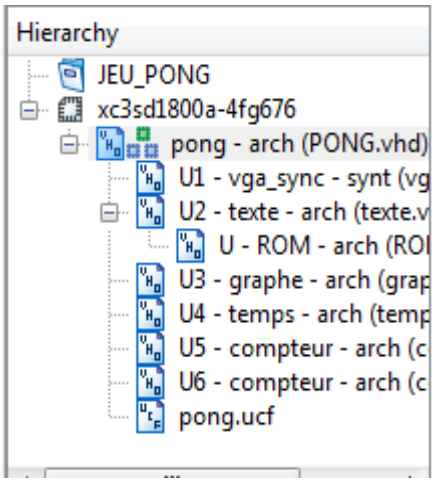


Figure III.30. *La hiérarchie de jeu de pong.*

Dans l'entité suivante on a quatre boutons d'entrées, deux pour la palette gauche et deux pour l'autre palette (BT et BT1).

```
entity pong is
   port(clk, reset: in std_logic;
      BT: in std_logic_vector(1 downto 0);
      BT1:in std_logic_vector(1 downto 0);
      hsync, vsync: out std_logic;
      rgb: out std_logic_vector (11 downto 0)
   );
end pong;
```

IV.4.2. Le circuit de graphe

Dans le circuit graphe on a remplacé le mur par la palette, cette palette est de la même caractéristique avec l'autre palette. La Figure III.31 présente les entrées/sorties du circuit graphe.

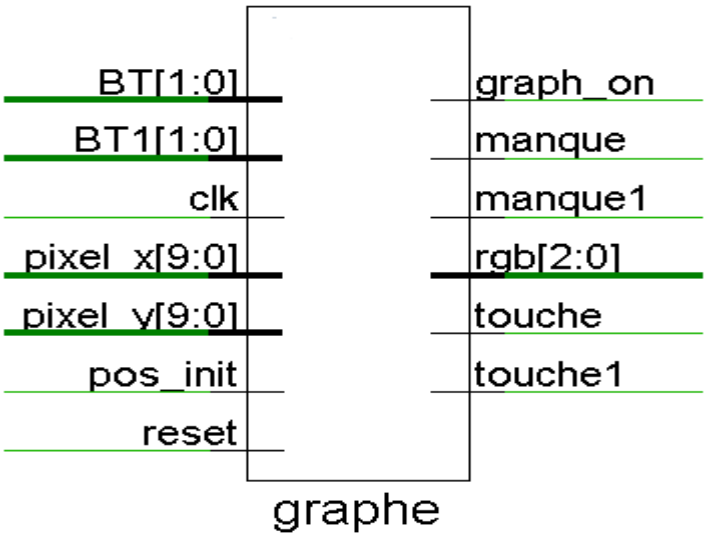


Figure III.31. *Les entrées/sorties de circuits de graphe.*

Dès que l'un des joueurs appuie sur un bouton poussoir, l'une des palettes bouge en haut ou en bas. S'il appuie sur le BT la palette à droite bouge et si BT1 est appuyé la palette gauche bouge (Figure III.32)

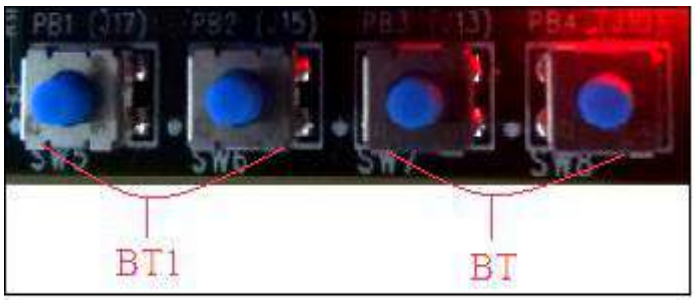


Figure III.32. *Les boutons poussoirs sur la carte FPGA.*

IV.4.3. Le circuit du texte

On a amélioré ce circuit par l'ajout de deux segments (seg2 et seg3) qui définissent le score du deuxième joueur (Figure III.33).

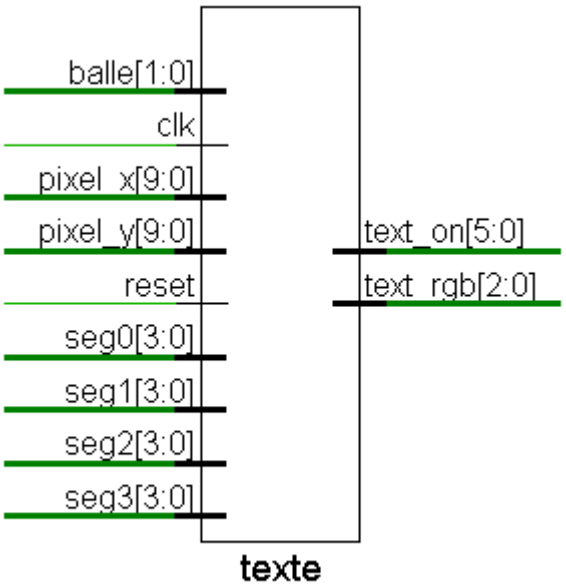


Figure III.33. *Les entrées/sorties du circuit texte.*

L'entité correspondante à ce circuit est comme suit

```
entity texte is
   port(clk, reset: in std_logic;
      pixel_x, pixel_y: in std_logic_vector(9 downto 0);
      seg0, seg1: in std_logic_vector(3 downto 0);
      seg2, seg3: in std_logic_vector(3 downto 0);
      balle: in std_logic_vector(1 downto 0);
      text_on: out std_logic_vector(5 downto 0);
      text_rgb: out std_logic_vector(2 downto 0)
   );
end texte;
```

Le signal de sortie text_on est de quatre bits de 0 jusqu'à 5:

- ✓ Le bit « 0 » pour afficher le message du résultat du jeu;
- ✓ Le bit « 1 » pour afficher la fin du jeu;
- ✓ Le bit « 2 » pour afficher la règle;
- ✓ Le bit « 3 » pour afficher le slogan;
- ✓ Le bit « 4 » pour afficher le score du joueur de droite;

✓ Le dernier bit pour afficher le score du joueur de gauche avec la durée de vie.

L'instruction regroupe les six bits donné par:

```
text_on <= zone_score_on & zone_score1_on & slogan_on & regle_on & fin_on & zone_aff_on ;
```

Le signal text_rgb est le responsable sur la couleur de chaque texte affiché.

IV.4.3.1. Les messages à affichées

IV.4.3.1.1. Le score et la durée de la vie

On a ajouté avec le premier score un autre score concernant le deuxième joueur qui est situé à droite en haut entre 512 et 640 verticalement et entre 0 et 31 horizontalement (FigureIII.34). Dès que le deuxième joueur frappe la balle il est incrémenté par 1.

Il est positionné par l'instruction suivante,

```
zone_score1_on <='1' when pix_y(9 downto 5)=0 and
            pix_x(9 downto 0)>="1000000000" and
            pix_x(9 downto 0)<="1010000000" else '0';
```

La taille choisie est comme suit,

```
Ligne_addr_sc1 <= std_logic_vector(pix_y(4 downto 1));
bit_addr_sc1 <= std_logic_vector(pix_x(3 downto 1));
```

Les adresses des caractères qui concernent le joueur de droite sont regroupées par l'instruction ci-après,

```
with pix_x(6 downto 4) select
char_addr_sc1 <=
   "1010011" when "000", -- S x53
   "1100011" when "001", -- c x63
   "1101111" when "010", -- o x6f
   "1110010" when "011", -- r x72
   "1100101" when "100", -- e x65
   "0111010" when "101", -- : x3a
   "011" & seg1 when "110",
   "011" & seg0 when others;
```

Les adresses des caractères concernant le joueur de gauche sont regroupées par l'instruction ci-après,

```
with pix_x(7 downto 4) select--7,4
  char_addr_sc <=
     "1010011" when "0000", -- S x53
     "1100011" when "0001", -- c x63
     "1101111" when "0010", -- o x6f
     "1110010" when "0011", -- r x72
     "1100101" when "0100", -- e x65
     "0111010" when "0101", -- : x3a
     "011" & seg3 when "0110",
     "011" & seg2 when "0111",
     "0000000" when "1000",
     "1000010" when "1001", -- B x42
     "1100001" when "1010", -- a x61
     "1101100" when "1011", -- l x6c
     "1101100" when "1100", -- l x6c
     "0111010" when "1101", -- :
     "01100" & balle when "1110",
     "0000000" when others;
```



Figure III.34. *Position d'affichage des scores et durée de vie.*

IV.4.3.1.2. la règle

L'emplacement et la taille de la règle sont les même que celui du mode 2 (Figure III.35). Le message affiché est « utilisez les BT et BT1 pour déplacer les deux palettes ».

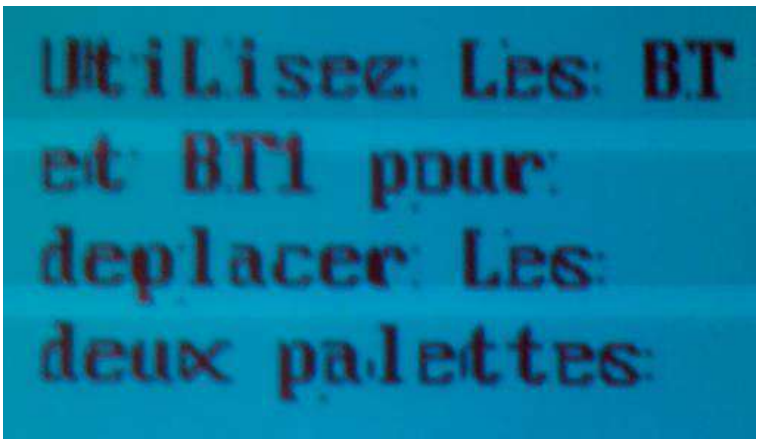


Figure III.35. *Message de la règle affiché du 2éme mode.*

IV.4.3.1.3. Résultats de jeu

On a inséré un message à la fin du jeu, qui donne numéro du joueur gagnant de la partie, identifié par le score supérieur, si le score à droite est supérieur on affiche « joueur 1 a gagné » sinon « joueur 2 a gagné », si les deux sont égaux alors on affiche « joueur ? a gagné ». La Figure III.36 donne le résultat du jeu qui montre le score des deux joueurs et affiche que le joueur 2 est gagnant.



Figure III.36. *Affichage du résultat cas joueur 2 gagné le jeu.*

On a choisi de positionner les messages du résultat entre 256 et 512 verticalement et entre 32 et 64 horizontalement par l'instruction suivante;

```
zone_aff_on <='1' when pix_y(9 downto 5)=1 and
               pix_x(9 downto 0)>="0100000000" and
               pix_x(9 downto 0)<="1000000000" else '0';
```

La taille des caractères est multipliée deux fois ((8 x 16) x 2 = (16 x 32)) l'instruction donnée comme suite

```
Ligne_addr_aff <= std_logic_vector(pix_y(4 downto 1));
bit_addr_aff <= std_logic_vector(pix_x(3 downto 1));
```

Les adresses de chaine des caractères sont les suivantes, "seg" change selon le résultat du jeu;

```
 with pix_x(7 downto 4) select
char_addr_aff <="1001010" when "0000", -- J
                "1101111" when "0001", -- o
                "1110101" when "0010", -- u
                "1100101" when "0011", -- e
                "1110101" when "0100", -- u
                "1110010" when "0101", -- r
                "011" & seg when "0110", -- 2/1/?
                "1100101" when "0111", -- e
                "1110011" when "1000", -- s
                "1110100" when "1001", -- t
                "0000000" when "1010", --
                "1100111" when "1011", -- g
                "1100001" when "1100", -- a
                "1100111" when "1101", -- g
                "1101110" when "1110", -- n
                "1100101" when others; -- e
```

La Figure III.37 représente le démarrage du jeu de deuxième mode.



Figure III.37. *Départ du jeu deuxième mode.*

La Figure III.38 représente la fin du jeu de deuxième mode.



Figure III.38. *Fin du jeu deuxième mode.*

La hiérarchie du circuit jeu PONG (mode 2) est donnée par la Figure III.39

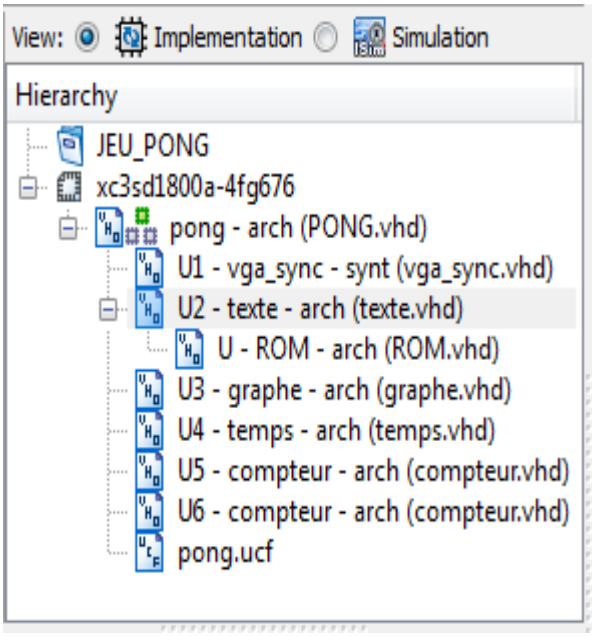


Figure III.39. *Hiérarchie de jeu de PONG mode deux.*

IV.5. *Implémentation*

Le rapport d'implémentation est donné par le Tableau III.3

Tableau III.3. *Rapport d'implémentation du mode deux.*

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	153	33,280	1%	
Number of 4 input LUTs	733	33,280	2%	
Number of occupied Slices	396	16,640	2%	
Total Number of 4 input LUTs	752	33,280	2%	
Number of bonded IOBs	20	519	3%	
Number of BUFGMUXs	1	24	4%	
Number of RAMB16BWERs	1	84	1%	

V. Conclusion

Dans ce chapitre nous avons donné une explication détaillée avec tous les circuits utilisés dans notre application pour la réalisation du jeu de PONG en deux modes, et comme nous avons vu dans les rapports d'implémentations de deux modes que les circuits des jeux a consommé 1% ce qui montre l'optimisation la bonne hiérarchie de conception adoptés.

Conclusion Générale

Un contrôleur VGA est réalisé et implémente sur carte FPGA pour la gestion d'affichage et la synchronisation et le rafraichissement de l'écran. A partir de l'étude et la conception des résultats de notre travail, on a conclu que la synchronisation basée sur des compteurs qui est assuré de figurer la zone d'affichage selon la résolution voulu, dans notre projet elle a été (640 x 480). La coloration RGB (rouge, vert et bleu) généré à la sortie du port VGA de la carte FPGA, chaque couleur est codée sur 4 bits, chaque bit donne une tension d'un couleur, La génération des objets fixes ensuite le fait bouger sous certaines conditions. Ce travail a touché plusieurs aspects dans l'affichage VGA

1. Affichage d'objets fixes.
2. Affichage d'objets animés on contrôlant le trajet et la vitesse.
3. Coloration d'objets que le sont fixes ou animés.
4. Affichage de texte statique.
5. Affichage de texte dynamique selon le mouvement d'une image.

En plus, grâce à la description matérielle à l'aide du langage VHDL, on a pu mettre en place un circuit structuré modulable par l'affichage VGA.

Le circuit réalisé peut être très utile pour le contrôle des panneaux d'affichage et panneaux publicitaire ou la taille des panneaux peut être facilement contrôlable, quelque modification sont à faire facilement.

Comme perspective à notre travail, on propose d'introduire en plus un contrôleur clavier et souris, utilisation de mémoire externe pour le stockage des images, texte et couleurs.

Bibliographies

[1] webopedia. VGA – video graphics array. Disponible : http://www.webopedia.com/ TERM/ V/VGA.html. [Consulté le 2 juin 2014].

[2] DOUMENC Rémi et BERHAULT Guillaume, "Projet VHDLCasse-briques", 10 janvier 2011. [En ligne]. Disponible: http://uuu.enseirb-matmeca.fr/~gberhault/docs/S7/VHDL/2A-VHDL_Projet_Casse_briques.pdf. [Consulté le 25 mars février 2014].

[3] Action AV & IT services, " Connecteur VGA, Disponible: http://www.action.mc/pdf /index 88f9.pdf. [Consulté le 5 mars 2014].

[4] " Mini-projet Electronique Numérique LE201 Pilotage d'un écran VGA Digicode en couleurs et Réalisation du jeu Simon", 2011-2012. [En ligne]. Disponible: http://www.licence.elec.upmc.fr/.../TP201-2012-2.pdf **.** **[**Consulté le 20 février 2014**].**

[5] max-k, Video Graphics Array (VGA), " *Les cartes graphiques*, 2006. Disponible: http://maxk69.free.fr/site-sepr-cg/index.php?page=vga. [Consulté le 11 février 2014].

[6] Ahmed BEN ATITALLAH. " Etude et Implantation d'Algorithmes de Compression d'Images dans un Environnement Mixte Matériel et Logiciel". 11 Juillet 2007 [En ligne]. Disponible: grenet.drimm.u-bordeaux1.fr / 2007/ BEN_ATITALLAH_AHMED-_2007.pdf. [Consulté le 20 mars 2014].

[7] David Feugey, Un kit FPGA pour les applications réseau," *silicon.fr*. 29 février 2008. Disponible:http://www.silicon.fr/un-kit-fpga-pour-les-applications-reseau-28435.html. [Consulté le 19 mai 2014].

[8] national instrument, 1 avril 2014. « *Les cinq principaux atouts de la technologie FPGA* ».

[9] Spartan-3A DSP Starter Platform User Guide,"*www.xilinx.com*. Disponible: [http://www.xilinx.com/products/boards-and-kits/HW-SD1800A-DSP-SB-UNI-G.htm]. [Consulté le 20 avril 2014].

Annexe

Assignement des pins

Il faut définir les entrées/sorties du FPGA à utiliser. Comme le FPGA est intégré sur une carte comportant des boutons poussoirs, une horloge, un port VGA, etc... La documentation de la carte fournit le nom des pins du FPGA reliées à ces entrées et sorties.

CLK: P26.

Reset: A7.

DAC_vsync: K25.

DAC_hsync: K26.

DAC_R0: L20.

DAC_R1: K20.

DAC_R2: F25.

DAC_R3: F24.

DAC_G0: M19.

DAC_G1: M18.

DAC_G2: J23.

DAC_G3: J22.

DAC_B0: L22.

DAC_B1: K21.

DAC_B2: G23.

DAC_B3: G24.

Les pins des boutons poussoirs utilisé:

BT [0]: J10.

BT [1]: J13.

BT1 [0]: J15.

BT1 [1] = J17.

Ecran utilisé

Videale numéro de modèle : 775CE



La carte utilisée:

