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

Implémentation d'un contrôleur d'un clavier sur une carte FPGA

Proposé et dirigé par,

Dr. BOUHEDDA Mounir

Co-encadré par,

Melle. BENAYAD Sihem

Préparé et présenté par,

Melle. KERMEZLI Zineb

Mémoire préparé au sein du Laboratoire des Systèmes Électroniques Avancés (LSEA) de l'Université de Médéa

Année Universitaire 2014-2015


بسم الله الرحمن الرحيم

Dédicace :

بسم الله الرحمان الرحيم والصلاة و السلام على اشرف المرسلين سيدنا محمد عليه ازكى الصلاة و السلام .

بعد سنين من الجد و الاجتهاد في طلب العلم و بتوفيق من الله عز و جل و سهر و رعاية

الوالدين تمكنت من الوصول الى مستوى الثانية ماستر و عمل مذكرة تخرج , التي اعتبرها

ثمار جهدي و تعبي, و بدوري اهدي هاته المذكرة الى ابي و أمي اللذان كانا عونا و دافعا لي خلال مساري التعليمي .

الى كل أفراد عائلتي و ابنتي اسراء , و الى كل الاساتذة الكرام الذين تعبو في تعليمي و لا انسى زميلاتي سهام و حورية اللتان كانت لهما دور هام في عمل هاته المذكرة .

و في الاخير أطلب من الله عز و جل ان يوفقني في طلب المزيد من العلم لي و لجميع الطلبة .

Remerciements :

En premier lieu, je tiens d'abord à remercier Dieu le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir

ce modeste travail.

En second lieu, je tiens à remercier et à exprimer ma gratitudes à mon encadreur Dr. BOUHEDDA Mounir qui m'a orienté et m'a soutenu durant mon projet de fin d'études.

Je témoigne reconnaissance

et gratitudes aux aimable crut

personnes, quel que soit de près ou de loin,

qui m'ont soutenu, acceptent et orientant durant mon projet

de fin d'étude.

ملخص :

الهدف من هذا العمل هو تصميم وتنفيذ وحدة تحكم لوحة المفاتيح على الدارة الكهربائية FPGA. ويتركز العمل على فهم طريقة نقل البيانات بواسطة المنفذ PS/2, وقمنا باستخراج المعلومات من سلسلة البيانات الآتية من لوحة المفاتيح, وقد تم التنفيذ و التحقق من هذا العمل عن طريق ثلاثة دارات طبقت على لوحة SPARTAN-3e DSP 1600e ل XILINX باستعمال لغة البرمجة VHDL.

الكلمات المفتاحية: وحدة تحكم لوحة المفاتيح ,PS/2, FPGA, VHDL.

Résumé:

L'objectif de ce travail est la conception et la réalisation d'un contrôleur d'un clavier avec l'utilisation de la carte FPGA. Le travail est focalisé sur la transmission de données à partir d'un porte PS / 2, l'information y extraite à partir de la trame qui vient du clavier. Le pilote réalisé a été testé en proposant trois circuit qui sont implémentés sur la carte SPARTAN-3E DSP 1600E de XILINX en utilisant le langage VHDL.

Mots clés: PS / 2, Contrôleur clavier, FPGA, VHDL.

Abstract:

The aim of this work is the design and implementation of a keyboard controller with the use of the FPGA board. The work is focused on the transmission of data from a PS / 2 door, the data are extracted from the keyboard. The realized pilot is tested with three proposed examples implemented on Spartan-3E DSP 1600E XILINX development board using VHDL language.

Key words: keyboard controller, PS / 2, FPGA, VHDL.

Table des matières

REMERCIMENTS.....I
RESUMÉ.....II
TABLE DES MATIÉRE.....IV
LISTE DES FIGURES.....V
LISTE DES TABLEAUX.....VII
LISTE DES ACRONYMES.....VIII
INTRODUCTION GENERALE.....1
CHAPITRE I :*LE PRINCIPE DE CONTROLEUR VGA ET PS2*.....2
I. Introduction.....2
II. VGA.....2
II.1. Historique.....2
II.2. Fonctionnement des cartes VGA.....2
II.3. Définition de la zone d'affichage avec compteurs.....4
II.4. Le Connectique du VGA.....5
II.5. Les modes d'affichages.....7
II.6. Les composants d'une carte VGA.....7
II.7. Contrôleur de tube cathodique.....8
II.8. La mémoire vidéo.....8
II.9. Contrôleur graphique.....8
II.10. Convertisseur numérique analogique.....8
II.11. Contrôleur d'attributs.....9
II.12. Séquenceur de timing.....9
II.13. Vue d'ensemble.....10
II.14. Le moniteur à tube cathodique.....10
II.15. Le codage vidéo.....12
III. Le protocol PS/2.....12
III.1. Historique.....12
III.2. Brochage.....13
III.3. La communication système périphérique PS2.....14
IV.Conclusion.....15
CHAPITRE II : *PRÉSENTATION DU CONTROLEUR DU CLAVIER*.....16
I. Introduction.....**16**
II. Le protocole PS/2.....**16**
II.1. Le connecteur PS/2.....16
II.2 .La transmission d'octets.....16

III. Présentation du mode de fonctionnement du clavier 17
III.1. Principe 17
III.2 .Fonctionnement courant 18
III.3 .Vue d'ensemble du code de balayage 20
IV. Conclusion **21**
CHAPITRE III : *IMPLÉMENTATION FPGA DU CONTROLEUR DU CLAVIER* 22
I .Introduction **22**
II .Cahier des charges **22**
III. Circuit général du clavier **22**
III.1. La machine d'état ***24***
III.2 . Exemple d'implémentation de circuit clavier ***26***
III.3 .Implémentation de circuit clavier ***28***
IV. Interdépendance de clavier et l'écran 29
IV.1. Affichage de lettre avec Mini clavier. ***29***
IV.*1.1. Circuit général de l'affichage.* 30
IV.2. Les circuits 31
IV.2.1. *Le circuit texte* 31
IV.2.2 . *Le circuit de synchronisation VGA* 34
IV.2.3. *Le circuit du diviseur de fréquence* 36
IV.3. Exemple d'implémentation de circuit affiche ***38***
IV.4.Implémentation de circuit affiche 40
V. Méthodes d'affichage avec clavier **41**
V.*1. Circuit général de jeu de PONG* 41
V.2. Gestion du déplacement de la palette ***42***
V.3. Les touches qui gérent chaque palette ***43***
V.4. La barre de vie ***44***
V.5. l'affichage des lettres par la coloration d'écran ***45***
V.5.1. Méthode de coloration ***45***
V.6. Le circuit clavier 47
V.7. Implémentation de circuit pong 49
VI . Conclusion 50
CONCLUSION GÉNÉRALE 51
BIBLIOGRAPHIES 52
ANNEXES 54

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 I. 7. Les bronchement de PS2 13
Figure I. 8. *Transmission entre système et périphérique* 14
Figure II. 2. *Exemple de trame transmis (un octet)* 16
Figure II. 3. *Structure d'un clavier* 18
Figure II. 4. *Codes de scan d'un clavier PS2 QWERTY* 20
Figure III. 1. *Les entrées/sorties de circuit clavier.* 22
Figure III. 2. *Hiérarchie du code clavier.* 23
Figure III. 3. *La machine d'état de circuit clavier.* 24
Figure III. 4. *Les touches utilisées.* 26
Figure III. 5. *Code ASCII de la touche Q apparaissant sur les LEDs de la Spartan 3E-1600* 26
Figure III. 6. *Code ASCII de la touche S apparaissant sur les LEDs de la Spartan 3E-1600.* 27
Figure III. 7. *Code ASCII de la touche 7 apparaissant sur les LEDs de la Spartan 3E-1600.* 27
Figure III. 8. *Code ASCII de la touche* « espace » *apparaissant sur les LEDs de la Spartan 3E-160028*
Figure III. 9. *Processus d'exécution.de cicuit clavier* 28
Figure III. 10. *Le circuit général de la source affiche.* 30
Figure III. 11. *Représentation de la hiérarchie du contrôleur VGA.* 30
Figure III. 12. R*eprésentation du circuit de texte* 31
Figure III. 13. R*eprésentation du circuit de ROM de caractères* 32
Figure III. 14. Exemple du carectére « s »dans la ROM 323
Figure III. 15. *Représentation de circuit de VGA_sync* 34
Figure III. 16. *L'organigramme général du circuit de synchronisation.* 35
Figure III. 17. *Représentation du schéma synoptique du circuit div_clk.* 36
Figure III. 18. *Représentation des boutons poussoirs à utilises.* 36
Figure III. 19. *Représentation du* Switch. 37
Figure III. 20. *Représentation de la position de la lettre « i ».* 37
Figure III. 21. *L'affichage de la lettre « i ».* 38

Figure III. 22. *L'affichage de la lettre « s ».* 38
Figure III. 23. *L'affichage de la lettre « r ».* 39
Figure III. 24. *L'affichage de la lettre « a ».* 39
Figure III. 25 . *Mini clavier* 40
Figure III. 26. *Processus d'exécution.de circuit affiche* 40
Figure III. 27. L*e circuit général du jeu de PONG.* 41
Figure III. 28. *L'hiérarchie du circuit général.* 42
Figure III. 29. *L'image qui montre la position initiale des palettes.* 43
Figure III. 30. *Le déplacement des palettes.* 44
Figure III. 31. *Les deux barres de vie.* 44
Figure III. 32. *L'image à la fin du jeu.* 45
Figure III. 33. *Coloration de la lettre « G ».* 46
Figure III. 34. *Coloration de chaque partie de la lettre « G ».* 46
Figure III. 35. *Le circuit global de pong.* 47
Figure III. 36. L*e circuit clavier.* 47
Figure III. 37. *Les touches de clavier utilisées.* 48
Figure III. 38. *Processus d'exécution.de circuit pong* 49

LISTE DES TABLEAUX

Tableau I. 1. *Tableau des durées de synchronisation.* 4
Tableau I. 2. *Signaux sur connecteur VGA.* 6
Tableau I. 3. *Brochement de PS2.* 13
Tableau II. 1. *Table des caractères ASCII de base* 19
Tableau II. 2. *Table des caractères ASCII étendus* 19
Tableau III. 1. *Rapport d'implémentation de circuit clavier* 29
Tableau III. 2. *Rapport d'implémentation de circuit affiche .* 41
Tableau III. 3. *Rapport d'implémentation de circuit pong.* 50

LISTE DES ACRONYMES

ASIC: *Application Spécific Integrated Circuit.*

ATC: *Attribute Controller.*

CGA: *Color Graphics Adapter.*

CRTC: *Cathode Ray Tube Controller.*

CLK: *Clock.*

DAC: *Digital to Analog Converter.*

DDC: *Display Data Channel.*

DDR: *Double Data Rate.*

DSP: *Digital Signal Processor.*

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.*

LSB: *Least Significant Bit.*

MDA: *Monochrome Display Adapter.*

MIPS: *Million Instruction Per Second.*

MMU: *Memory Management Unit.*

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

PS/2: *Personnal System/2.*

PE: *Processing Elements.*

PS2D: *Personnal System/2 Data.*

PS2C: *Personnal System/2 Clock.*

RGB: *Red, Green, Blue.*

RAM: *Random Access Memory.*

ROM: *Read Only Memory.*

SPI: *Serial Flash Pinout Interface.*

SDRAM: *Synchronous Dynamic Random Access Memory.*

TS: *Timing Sequencer.*

USB: *Universel Serial Bus.*

VGA: *Video Graphics Array.*

VHDL : *Very high speed integrated circuit Hardware Description Language.*

Introduction générale

Le travail qu'il m'a été proposé dans le cadre de ce projet de fin d'études de Master a pour but de réaliser un pilote hardware d'un clavier. Il a été donc nécessaire de programmer en VHDL deux circuits FPGA, l'une pour la réception des données à partir du clavier et l'autre pour le contrôle de l'écran VGA afin de visualiser les résultats de la réception des données à partir du clavier.

Le langage VHDL est un langage de description de haut niveau pouvant être utilisé pour programmer des composants programmables, en particulier de type FPGA.

Le document présenté est scindé en trois parties,

Dans le premier chapitre, nous présentons des généralités sur le principe de fonctionnement de la carte VGA, aussi que du contrôleur VGA et port PS/2 .

Le deuxième chapitre est consacré au protocole de communication et surtout de réception du clavier.

Dans le troisième chapitre, nous présentons nos implémentations réalisées, nous avons divisé notre travail en trois parties, la première partie est d'affiché le code ASCII d'un caractère sur les LEDs de la carte FPGA, la deuxième partie nous avons utilisé les boutons poussoirs comme un Mini clavier ou chaque bouton est responsable de l'affichage d'un caractère à partir d'une mémoire interne, la dernière partie est de réalisé un jeu de PONG commandé par les touches du clavier, 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.

CHAPITRE I : LE PRINCIPE DU CONTROLEUR VGA ET PS2.

-

I. Introduction

Dans ce chapitre on s'intéresse à l'étude du principe du contrôleur VGA et PS2. Le concept général ainsi que le protocole de communication sont présentés en détails en vue d'un codage VHDL et implémentation sur FPGA.

II. VGA

II.1. Historique

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 [1].

II.2. Fonctionnement des cartes VGA

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

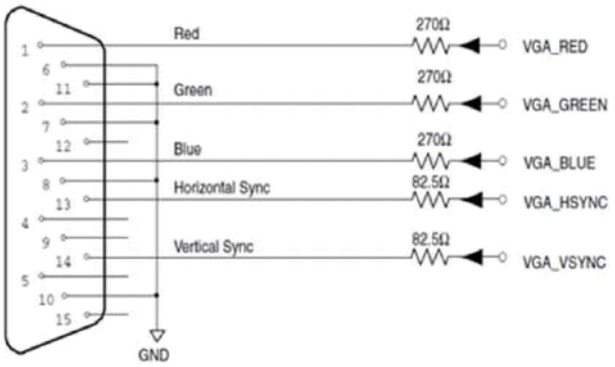


Figure I.1. *Connectique VGA d'une FPGA Spartan-3 DSP [3].*

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 horizontale 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].

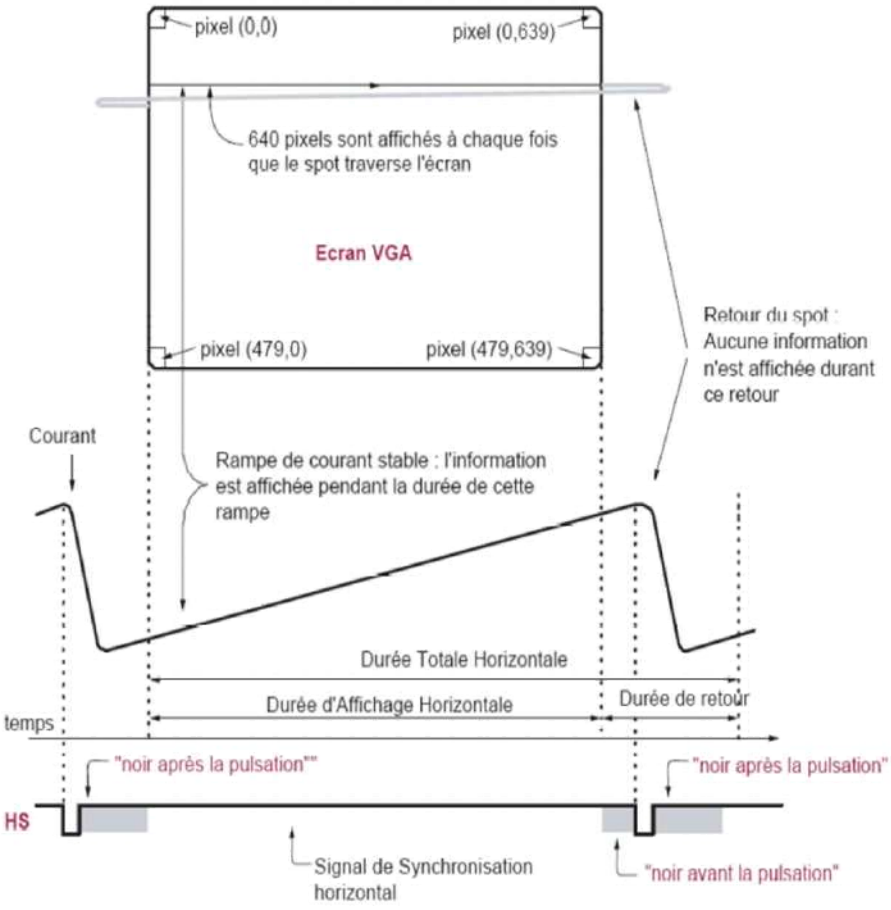


Figure I.2. *Exemple de timing d'un écran à tube cathodique [3].*

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 certains nombres de lignes auront été passé, le faisceau se situera en bas à droite de l'écran. Afin de le 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 certains nombres 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].

II.3. 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 incrémentés pour obtenir les cycles d'horloges afin de définir la zone d'affichage, les cycles sont présentés dans le Tableau I.1.

Tableau I. 1. *Tableau des durées de synchronisation [4].*

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 ont la même forme pour la synchronisation horizontale et la synchronisation verticale, et qui sont représentées sur la FigureI.3 suivante [4].

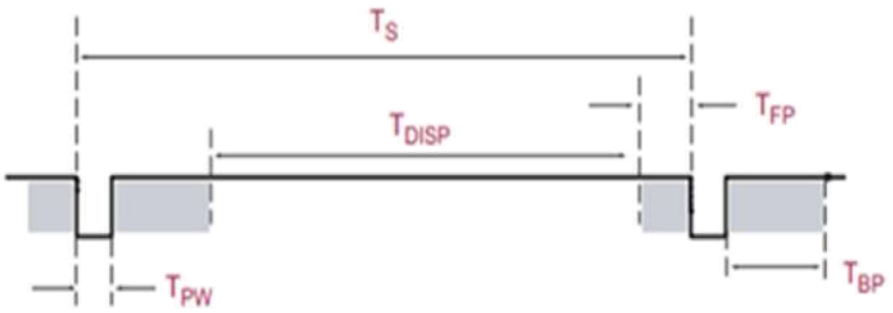


Figure I.3. *Durée des temps de synchro pour l'affichage VGA [4].*

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).

II.4. 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 ce dernier 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].

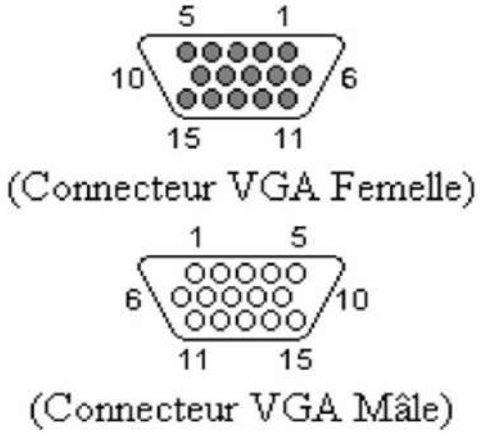


Figure I.4. Connecteurs VGA mâle et femelle [5].

Tableau I. 2. *Signaux sur connecteur VGA [5].*

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)

II.5. 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].

II.6. 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'entres eux sont effectivement utilisés [5].

II.7. 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].

II.8. *La mémoire vidéo*

L'image affichée à l'écran est stockée dans la mémoire vidéo. Lorsqu'on est en mode texte, ce sont les caractères ASCII et leurs couleurs 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].

II.9. *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].

II.10. 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 à

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

II.11. 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].

II.12. 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].

II.13. *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].

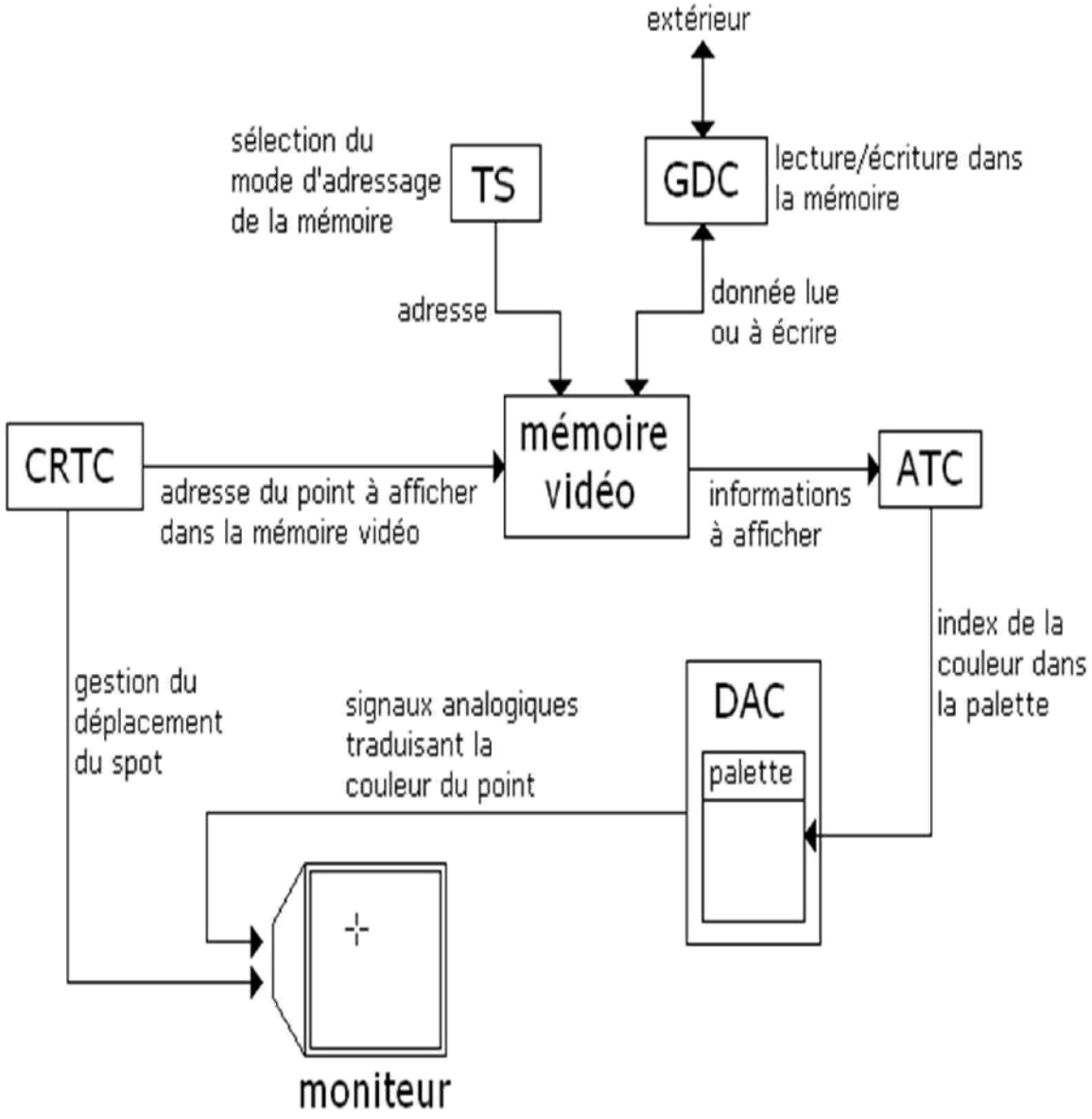


Figure I.5. *Vue d'ensemble des composants d'une carte VGA.*

II.14. 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].

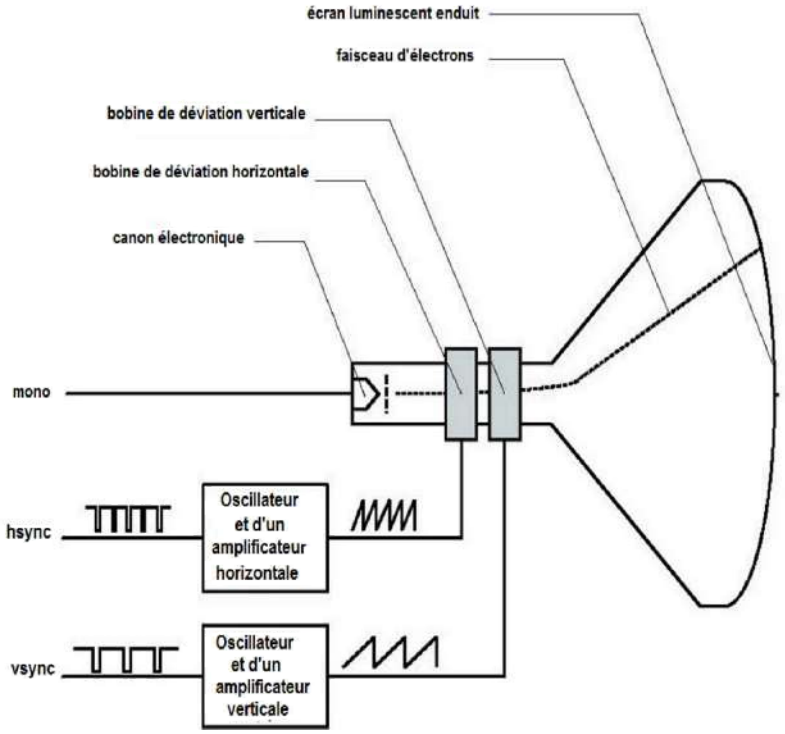


Figure I.6. Schéma conceptuel d'un CRT moniteur [6].

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].

II.15. 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].

III. Le Protocol PS/2

Le port **PS/2** ou **port Mini-din** est un port de connexion de dimensions réduites pour souris ou clavier.

III.1. Historique

Il est apparu avec les ordinateurs IBM PS/2 vers 1987. Cependant, le port PS/2 s'est généralisé plus tard avec l'introduction de la norme ATX pour carte mère en 1995.

Auparavant le clavier se branchait sur un connecteur DIN tandis que la souris se branchait généralement sur un port série ; ces deux connecteurs ont disparu avec la généralisation du port PS/2 et de l'USB.

En 2013, la plupart des nouvelles cartes mères disposent encore de port PS/2 bien que de nombreux claviers et souris utilisent désormais le port USB, ils permettent néanmoins de ne pas « perdre » deux ports USB pour le clavier et la souris ; en cela des adaptateurs

USB/PS2 existent. On peut également utiliser des claviers et souris sans fil grâce à la technologie *Bluetooth* [12].

Le protocole PS/2 permet une communication synchrone, bidirectionnelle et bits en série entre un système et un périphérique (couramment clavier, souris ou touchpad de portable). Chaque coté est susceptible d'envoyer des commandes ou des données mais un seul côté ne pourra transmettre à la fois puisque la liaison se fait sur un seul fil. C'est un connecteur mini-DIN-6 qui comporte 6 broches incluant masse, alimentation, la donnée série DATA et l'horloge CLK. Deux broches sont inutilisées [8].

III.2. Brochage

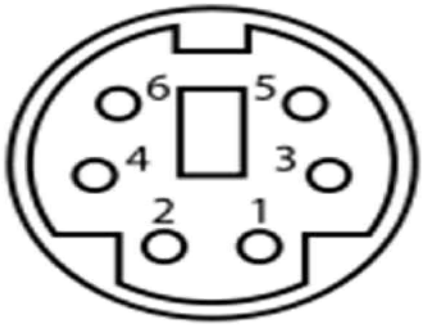


Figure I.7. *Brochage de PS2 [8]***.**

Tableau I. 3. *Brochage de PS2 [8*].

1	**ps/2 DATA**
2	**N/C**
3	**Ground 0V**
4	**Power + 5V**
5	**PS/2 CLK**
6	**N/C**

Broche 1 : Données, fil rouge ou vert

Broche 2 : Réservé, fil vert

Broche 3 : 0V (référentiel), fil blanc

Broche 4 : + 5V, fil jaune

Broche 5 : Horloge, fil noir

Broche 6 : Réservé, fil bleu

Sur certains portables, les broches 2 et 6 transportent respectivement les données et l'horloge pour la souris, dans le cas de l'utilisation d'un câble séparateur [8].

Il est fortement déconseillé de brancher ou débrancher à chaud du matériel sur un port PS/2. Il est également déconseillé de brancher la souris sur le port clavier et vice-versa. C'est pourquoi sur les cartes mère ATX (standard créé en 1995) et sur les connecteurs des périphériques il y a des codes de couleurs (violet pour le clavier, vert pour la souris) sur les prises. Avant 1995, le montage des PC était généralement réservé aux professionnels.

En cas de dysfonctionnement du port PS/2 réservé au clavier, le système d'exploitation Linux supporte la connexion du clavier et sa gestion sur le port PS/2 réservé à la souris [8].

III.3. La communication système périphérique PS2

C'est le système maître qui alimente le périphérique PS/2. Les signaux DATA et CLK sont bidirectionnels et à sortie collecteur ouvert. Une résistance de 10Kohm environ doit être placées entre ces deux signaux et l'alimentation ce qui garantit un niveau haut hors de toute transaction.

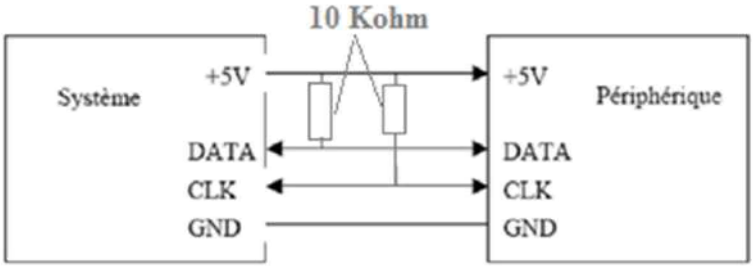


Figure I.8. *Transmission entre système et périphérique [8].*

Le système maître est susceptible d'envoyer au périphérique (clavier ou souris) des ordres de commande mais à un autre moment c'est le périphérique (clavier ou souris) qui va émettre des messages ou retourner des données. Les octets de commandes et messages sont transmis de façon synchrone et série sur 11/12bits (les 8 bits à transmettre, LSB en premier, précédés d'un bit Start ('0') et suivi d'un bit de parité et d'un bit stop ('1'). Le bit de parité impaire vaut '1' si le nombre total de '1' dans l'octet et le bit de parité lui-même est impair [8].

Durant la transmission, c'est le périphérique qui fournit l'horloge en la positionnant à un niveau bas (front descendant) tandis que l'émetteur (système ou périphérique) place DATA à un niveau bas pour un bit '0' ou le laisse inactif pour un bit '1'. La fréquence d'horloge est au maximum de 33 KHz mais le plus couramment de 15 KHz. Entre les transmissions, le bus peut être :

- Idle : CLK et DATA sont tous les deux au niveau haut. Il n'y a pas d'activité.
- Inhibit : Le système maintient CLK au niveau bas.
- Request to send : Le maître maintient DATA au niveau bas mais laisse CLK flotter. Il est prêt à émettre [8].

IV. Conclusion

Dans ce chapitre, une présentation des différents composants qui peuvent exister sur un contrôleur VGA. Tout en expliquant le principe général de synchronisation et le principe de transmission des données est illustré.

CHAPITRE II : PRESENTATION DU CONTROLEUR DU CLAVIER.

I. Introduction

Dans ce chapitre le principe de fonctionnement détaillé du clavier est présenté en insistant sur la génération et la transmission des données.

II. Le protocole PS/2

Le protocole PS/2 permet une communication synchrone et bidirectionnelle. Elle a pour particularité d'être une liaison série entre un système (l'ordinateur) et un périphérique (classiquement un clavier ou une souris). Une telle liaison n'est assurée que par un seul fil. Ainsi, si le caractère bidirectionnel de la communication permet aux deux côtés (i.e. au système ou au périphérique) d'envoyer des données, il n'y a qu'un seul des deux côtés qui ne peut transmettre à la fois [9].

II.1 La transmission d'octets

La transmission d'octet se fait de façon synchrone et série par trame de 11 bits. Chaque trame se compose de 8 bits d'information, le LSB étant positionné en premier, précédés d'un bit de start à 0 et suivis d'un bit de parité et d'un bit de stop à 1. Si le nombre total de '1' dans l'octet est pair, le bit de parité vaut '1' (Figure II.2) [9].

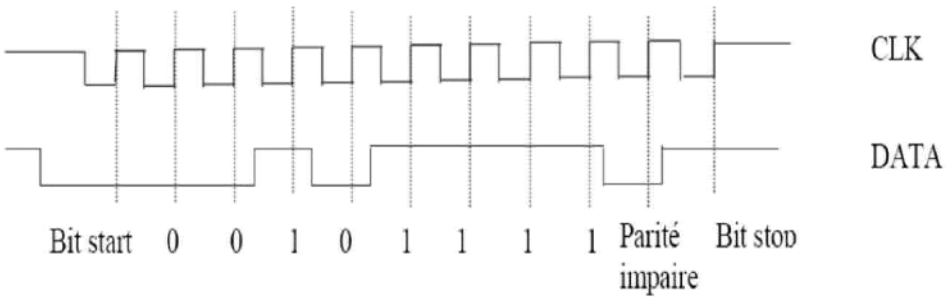


Figure II.1. *Exemple de trame transmis (un octet) [9].*

Le signal d'horloge qui permet de cadencer la transmission est fourni par le périphérique. La fréquence d'horloge est couramment de 15 KHz (elle varie cependant selon les modèles de souris et peut valoir jusqu'à 33 KHz).

Entre les transmissions de données, le bus peut être dans l'état Idle (CLK et DATA sont tous les deux au niveau haut, il n'y a pas d'activité) ou encore Request to send (le système maître maintient DATA au niveau bas et laisse le signal CLK flotter, il est alors prêt à émettre.

Pour les transmissions de données dans le sens périphérique vers système maître, le bus doit au préalable se trouver dans l'état Idle. Le périphérique transmet les données au rythme de son horloge dont la période varie selon les systèmes de 60 à 100 µs. Il change les données sur front montant de l'horloge, le système doit les lire sur front descendant.

Pour ce qui concerne les transmissions dans le sens système maître vers périphérique, le système maître indique au périphérique qu'il souhaite émettre une commande en maintenant le signal d'horloge au niveau bas pendant 100 µs puis place le bus en état Request To Send en forçant Data à 0 et en laissant le signal d'horloge. Dès que le périphérique reconnaît l'état Request To Send du bus, il se tient prêt à recevoir une commande et émet son signal d'horloge. Cette étape se fait en moins de 10ms. Le système maître change les données sur front descendant, le périphérique échantillonne chaque bit sur front montant de l'horloge [9], [12].

III. Présentation du mode de fonctionnement du clavier

III.1. Principe

Le clavier, tout comme la souris, envoie ses données au système via une connexion série bidirectionnelle. Outre la masse, cette liaison est constituée de deux fils, l'un véhicule le signal d'horloge fourni par le clavier, l'autre sert au transport des données. La liaison étant, comme dans le cas de la souris bidirectionnelle, le clavier pourra non seulement envoyer des données au système mais aussi recevoir des commandes de celui-ci.

Chaque système étant conçu pour pouvoir recevoir plusieurs types de clavier suivant le pays de commercialisation, un mécanisme de standardisation des touches est utilisé. A chaque appui sur une touche, le clavier envoie au système un code géographique appelé « scan code » qui est uniquement lié à l'emplacement de la touche sur le clavier.

Le clavier peut stocker jusqu'à 10 scan codes. De plus, pour permettre la détection de l'appui simultané sur plusieurs touches (par exemple pour écrire une lettre majuscule), il existe deux types de scan codes. En effet, un scan code appelé « make code » est envoyé au système lors de l'enfoncement d'une touche et un autre appelé « break code » est envoyé lors de son relâchement.

Ainsi, lorsque l'on appuie sur la touche « shift » toutes les lettres tapées après l'envoi du make code de shift et avant l'envoi de son break code seront écrites en majuscule.

III.2. Fonctionnement courant

Il faut maintenant s'intéresser à la structure même du clavier. En effet, pour détecter l'enfoncement d'une touche, un réseau de câbles est placé sous le clavier de l'ordinateur. Chaque touche est installée sur une intersection de câbles et ferme un contact si elle est enfoncée. Le microprocesseur du clavier utilise un courant qui balaie chaque rangée en séquence, des milliers de fois par seconde, que le clavier soit utilisé ou non (Figure II.3) [10].

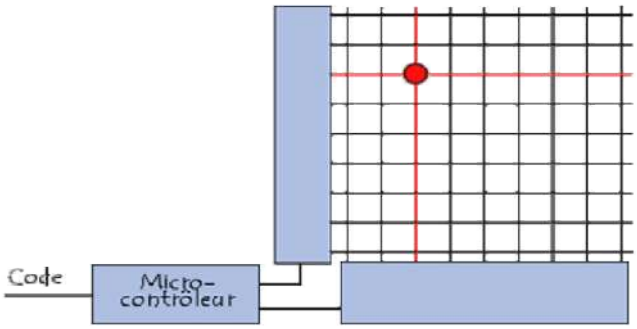


Figure II.2. *Structure d'un clavier [10].*

Quand une touche est enfoncée, le microprocesseur détecte une rangée comprenant un circuit fermé et entame le balayage des colonnes. L'identité de la touche est transmise sous la forme d'un code de touche qui est ensuite décodé et envoyé à l'écran sous forme d'un caractère dactylographié ou d'une instruction.

Chaque touche correspond donc à un code appelé code ASCII (American Standard Code for Information Interchange). Il permet le codage de caractères sur 8 bits, soit 256 caractères possibles.

Les codes 0 à 31 ne sont pas des caractères. On les appelle caractères de contrôle car ils permettent de faire des actions telles que le retour à la ligne ou l'émission d'un bip sonore.

Les codes 65 à 90 représentent les majuscules. Les codes 97 à 122 représentent les minuscules. Ainsi, il suffit de modifier le $6^{ème}$ bit pour passer de majuscules à minuscules, c'est-à-dire ajouter 32 au code ASCII en base décimale [10].

Tous ces codes peuvent être retrouvés dans les tableaux II.1 et II.2.

Tableau II. 1. *Table des caractères ASCII de base [12]*

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Tableau II. 2. *Table des caractères ASCII étendus [12]*

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▓	209	D1	╤	241	F1	±
146	92	Æ	178	B2	│	210	D2	╥	242	F2	≥
147	93	ô	179	B3	┤	211	D3	╙	243	F3	≤
148	94	ö	180	B4	╡	212	D4	Ö	244	F4	⌠
149	95	ò	181	B5	╢	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╖	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╕	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╣	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╗	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	╝	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╜	219	DB	▒	251	FB	√
156	9C	£	188	BC	╛	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

III.3 Vue d'ensemble du code de balayage

Un clavier est constitué d'une matrice de touches et un microcontrôleur embarqué qui surveille les activités des touches et envoie un code de balayage en conséquence. Trois types d'activités des touches sont observés:

- Quand une touche est pressée, le code correspondant est envoyé.
- Quand une touche est enfoncée continuellement, le code est transmis en continue à une vitesse constante. Par défaut, un clavier PS2 transmet le code 100 ms après l'appui de la touche pour 0.5 secondes.
- Quand la touche est relâchée, le code de rupture est transmis.

Le code des touches de la partie principale d'un clavier PS2 est représenté sur la figure II.4. Il est normalement 1 octet de large et représenté par deux nombres hexadécimaux. Par exemple, le code de marque de la touche A est 1C. Ce code peut être véhiculé par un paquet lors de la transmission. Les codes d'une combinaison de touches à usage spécial, qui sont connus comme les touches prolongées, peuvent avoir 2 à 4 octets. Quelques-uns de ces touches sont présentés dans la figure II.4. Par exemple, le code de marque de la flèche du haut sur la droite est EO 75. Plusieurs paquets sont nécessaires pour la transmission. Les codes de rupture des touches régulières se composent de F0 suivi du code de marque de la clé. Par exemple, le code de la rupture de la touche A est F0 1C. Le clavier PS2 transmet une séquence de codes en fonction des activités des touches. Par exemple, quand on appuie et on relâche la touche A, le clavier transmet d'abord son code de marque et puis le code de pause: F0 1C 1C [11].

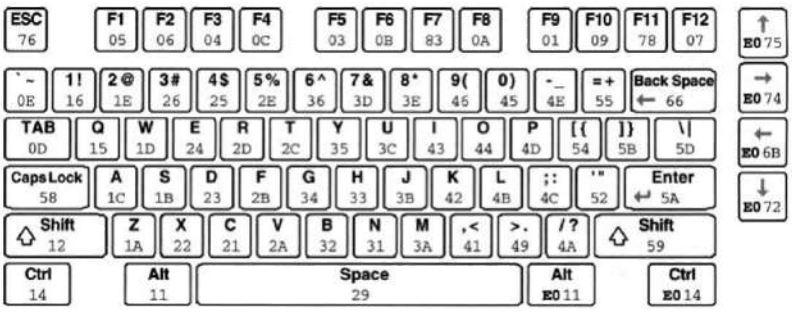


Figure II.3. *Codes de scan d'un clavier PS2 QWERTY*

Si nous maintenons la touche enfoncée pendant un certain temps avant de le relâcher, le code de marque sera transmis à plusieurs reprises:

1C 1C 1C ... 1C F0 1C.

Plusieurs clés peuvent être pressées en même temps. Par exemple, nous pouvons d'abord appuyer sur la touche shift (dont la marque est le code 12) puis sur la touche A, puis relâchez la touche A puis relâchez la touche Maj:

12 1C 1C F0 F0 12.

La séquence précédente est de savoir comment nous obtenons normalement une majuscule A. Notons qu'il n'y a pas de code spécial pour distinguer les touches minuscules et majuscules. Il est de la responsabilité de l'appareil hôte de garder une trace de savoir si la touche Maj est pressée et pour déterminer le cas en conséquence [11].

IV. Conclusion

Pour une implémentation FPGA du driver de clavier PS2, le protocole de communication a été présenté en détails dans le chapitre prochain, en présentant tous les modes de fonctionnement du clavier.

CHAPITRE III : IMPLEMENTATION SUR FPGA DU CONTROLEUR DU CLAVIER.

I. Introduction

Afin de bien comprendre et réaliser le contrôleur clavier sur FPGA, on propose de faire un mini clavier et de réaliser un jeu implémenté entièrement sur une carte FPGA commandé par le clavier.

II. Cahier des charges

On a débuté notre travail par comprendre le principe de fonctionnement du clavier à travers le port PS2. La première étape est d'arriver à afficher le code ASCII des touches sur les LEDs de la carte FPGA.

La deuxième étape est d'afficher sur l'écran des lettres à partir d'une mémoire interne par utilisation des quatre boutons poussoirs (de la cartre d'évaluation FPGA). On considère ces boutons poussoir comme une sorte d'un mini clavier chaque bouton est responsable de l'affichage d'une lettre.

La dernière étape est de réaliser un jeu de pong contrôlé par un clavier.

III. Circuit général du clavier

Nous avons commencé notre travail par comprendre le principe de fonctionnement de clavier avec le port PS2, comment transmettre les données après en appuyant sur une touche. Cette étape consiste à envoyer les données à partir du clavier vers la carte FPGA pour allumer les LEDs.

La Figure III.1. représsantation des entrées/sorties de circuit clavier.

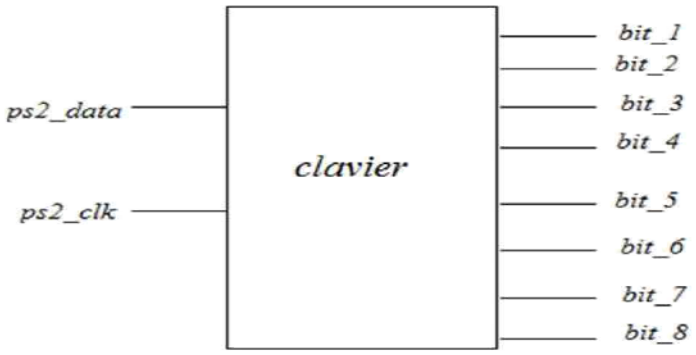


Figure III.1. Les entrées/sorties de circuit clavier.

L'entité correspondante à ce circuit (figure III.1)

```
 entity clavier is
port(           ps2_data: in std_logic;
                 ps2_clk: in std_logic;
                    bit_1  : out std_logic;
                    bit_2  : out std_logic;
                    bit_3  : out std_logic;
                    bit_4  : out std_logic;
                    bit_5  : out std_logic;
                    bit_6  : out std_logic;
                    bit_7  : out std_logic;
                    bit_8  : out std_logic);
end clavier;
```

- ✓ Ps2_data : Les informations envoyées par le clavier.
- ✓ Ps2_clk : Horloge généré par le port PS2.
- ✓ Bit_[1…8] : La suite de donnée.

L'hiérarchie du circuit est montré par la Figure III.2.

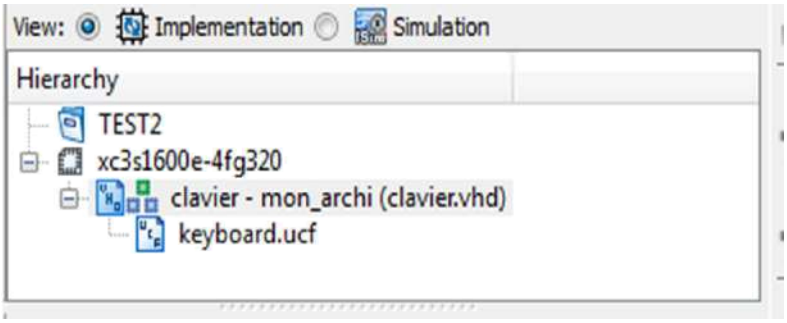


Figure III.2. *Hiérarchie du code clavier.*

III.1 La machine d'état

Ce circuit contient une machine d'état qui représente les états de transmission. La Figure III.3 montre la machine d'état décrite.

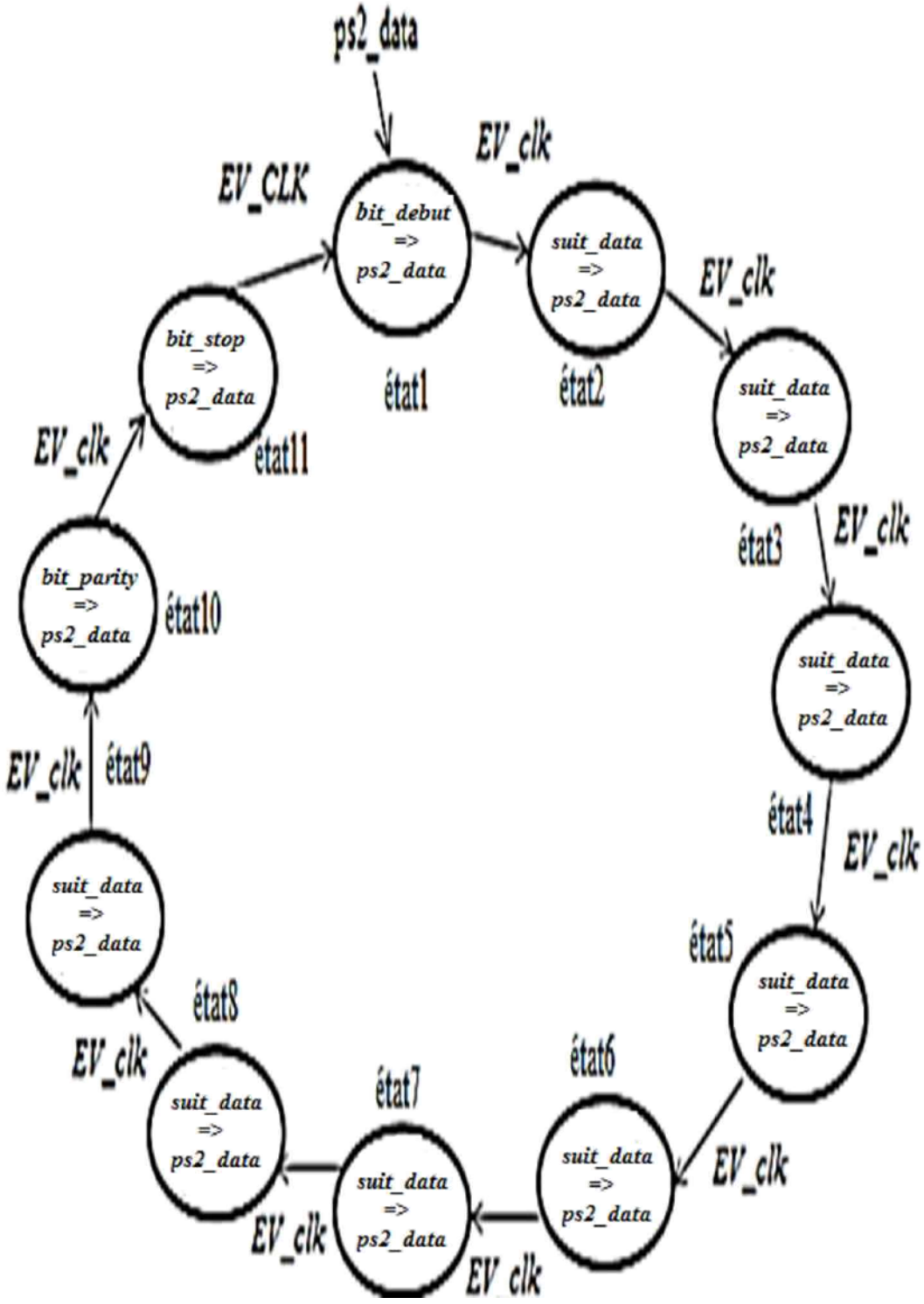


Figure III.3. *La machine d'état de circuit clavier.*

- ✓ EV_clk : Un évènement d'horloge.
- ✓ Suit_data : La suit d'information qui contient huit bits.
- ✓ Bit_début : Bit de début de transmission.
- ✓ Bit_parity : Bit de parité.
- ✓ Bit_stop : Bit de stop de transmission.

Il existe onze états, chaque état prend un bit,

- ✓ Etat 1 : Prendre le bit début de transmettre.
- ✓ Etat [2..9] : Prendre les huit bits de données.
- ✓ Etat 10 : Prendre le bit de parité.
- ✓ Etat 11 : Prendre le bit de fin de transmettre.

Lorsqu'en appuyant sur une touche, une suite de données sera transmettre pour chaque changement d'horloge, comme le montre les instructions suivantes,

```
if ps2_clk'event and ps2_clk = '1' then
    etat_present <= etat_suivant ;
end if;

if ps2_clk'event and ps2_clk = '0' then
  if etat_present = etat1 then
      bit_debut <= ps2_data;
      etat_suivant <= etat2;
   elsif etat_present = etat2 then
        suit_data (0) <= ps2_data;
        etat_suivant <= etat3;
                .
                .
                .
```

III.2 Exemple d'implémentation du circuit clavier

Juste comme exemples, On a fait le choix sur quelques touches afin de tester notre conception du circuit implémenté sur FPGA Spartan 3E- DSP 1600E de Xilinx.

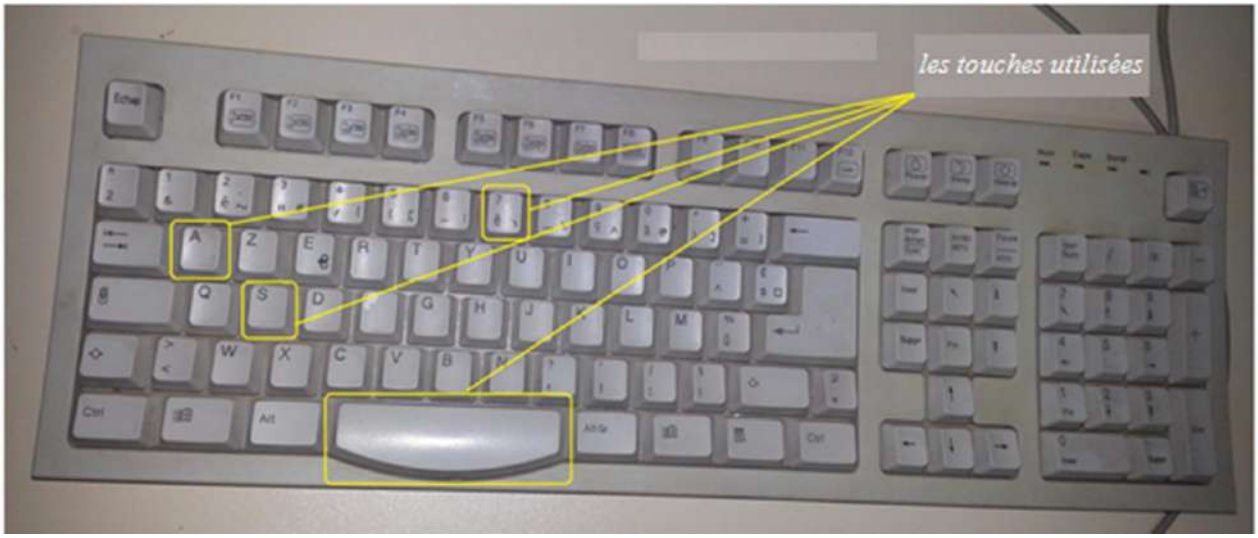


Figure III.4. *Les touches utilisées.*

- La touche « Q »

Lorsqu'on appuyant sur la touche Q le code key sera affiché sur les Leds de la carte FPGA qui est « 00010101 » comme le montre la Figure III.5,

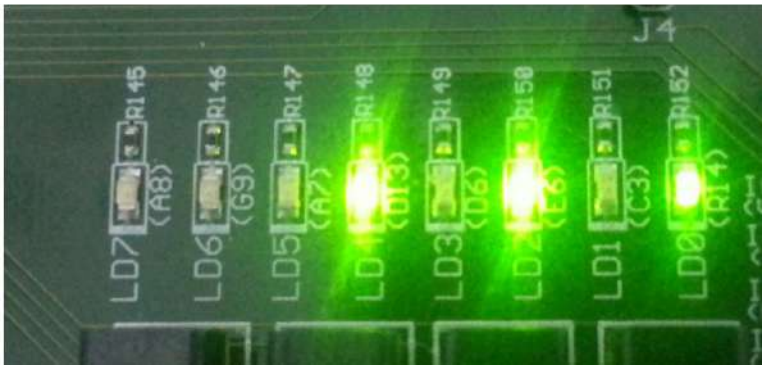


Figure III.5. *Code ASCII de la touche Q apparaissant sur les LEDs de la Spartan 3E-1600.*

- La touche « S »

Lorsqu'on appuyant sur la touche S le code key sera affiché sur les Leds de la carte FPGA qui est « 00011011 » comme le montre la Figure III.6,

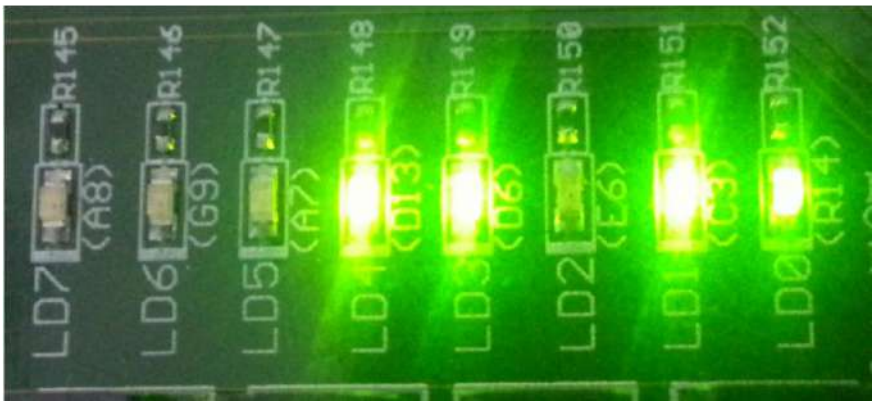


Figure III.6. *Code ASCII de la touche S apparaissant sur les LEDs de la Spartan 3E-1600.*

- La touche « 7 »

Lorsqu'on appuyant sur la touche 7 le code key sera affiché sur les Leds de la carte FPGA qui est « 00111101 » comme le montre la Figure III.7,

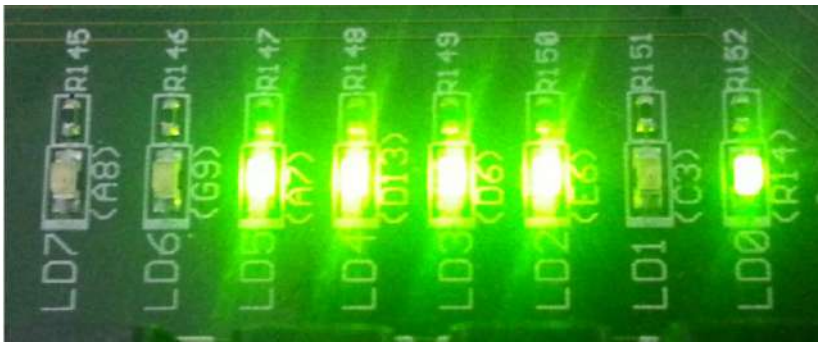


Figure III.7. *Code ASCII de la touche 7 apparaissant sur les LEDs de la Spartan 3E-1600.*

- La touche « espace »

Lorsqu'on appuyant sur la touche « espace » le code key sera affiché sur les Leds de la carte FPGA qui est « 00101001 » comme le montre la Figure III.8.

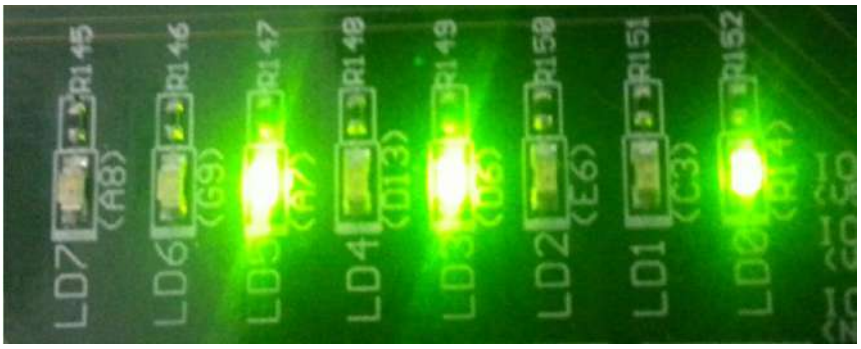


Figure III.8. *Code ASCII de la touche* « espace » *apparaissant sur les LEDs de la Spartan 3E-1600.*

III.3 Implémentation de circuit clavier

On a synthétisé le code VHDL clavier par le logiciel ISE de Xilinx, en utilisant la carte 3SD1600e-4FG320 FPGA.

Voici le processus d'implémentation présenté dans la Figure III.9 suivante :

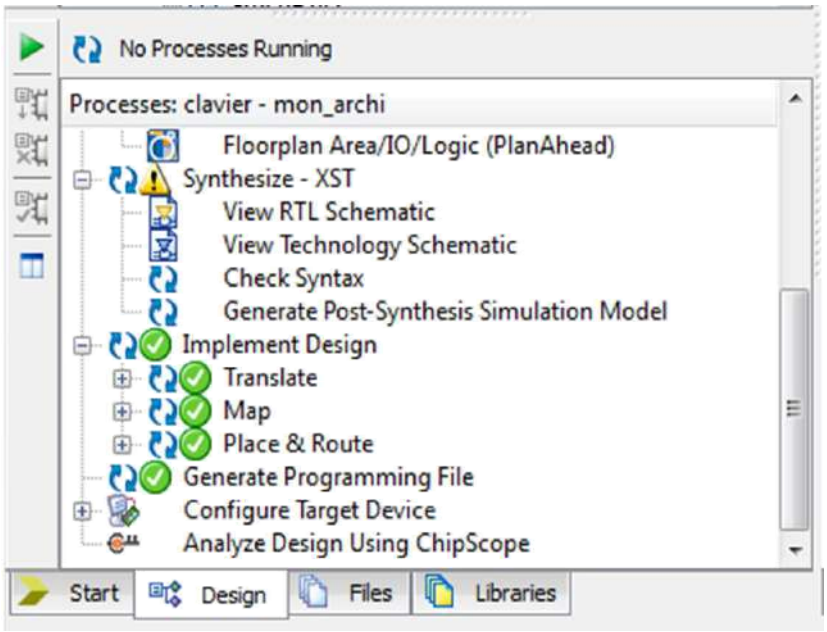


Figure III.9. *Processus d'exécution de circuit clavier.*

Et le rapport d'implémentation est donné par le Tableau III.1.

Tableau III.1. *Rapport d'implémentation de circuit clavier.*

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	30	29,504	1%	
Number of occupied Slices	20	14,752	1%	
Number of Slices containing only related logic	20	20	100%	
Number of Slices containing unrelated logic	0	20	0%	
Number of bonded IOBs	10	250	4%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	1.48			

IV. Interdépendance de clavier et l'écran

On a deux méthodes d'affichage sur l'écran **:**

- Un affichage avec l'utilisation d'une mémoire interne qui contient des caractères. Chaque caractère est défini par son adresse, en appelant ce caractère, on peut définir sa taille, sa position et sa couleur.
- Un affichage par coloration d'écran, c'est à dire on peut colorée, la zone d'affichage en spécifiant les pixels qui doit les colorés.

IV.1. Affichage de lettre avec Mini clavier.

Nous avons appliqué cette méthode sur la carte FPGA pour l'affichage des lettres sur l'écran en utilisant des boutons poussoirs comme un mini clavier, chaque bouton poussoirs représente une lettre.

IV.1.1. Circuit général de l'affichage

La Figure III.10 représente le schéma d'affichage. Il est composé de cinq circuits principaux.

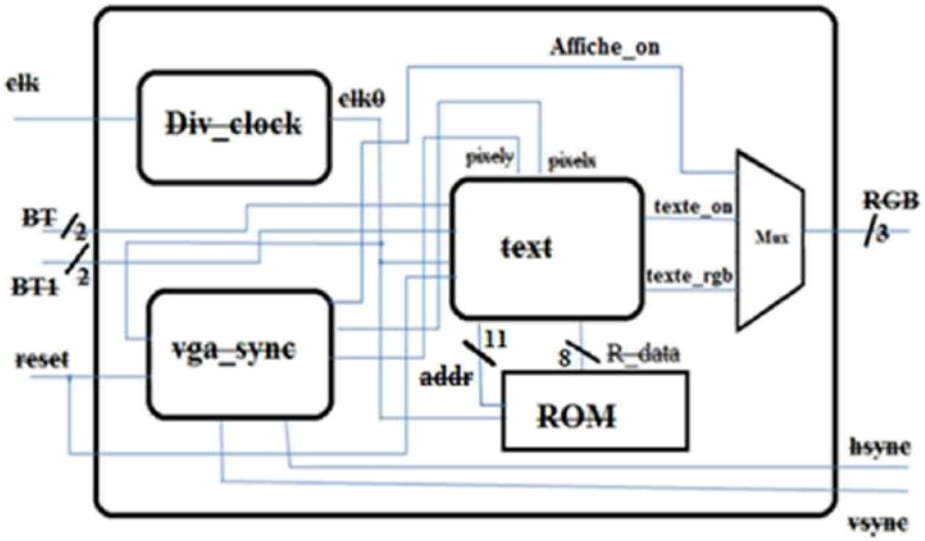


Figure III.10. *Le circuit général de la source affichée.*

Le code principal "affiche" fait appel aux ressources: vga_sync, text, ROM et div_clk, la hiérarchie est représentée par la Figure III.11.

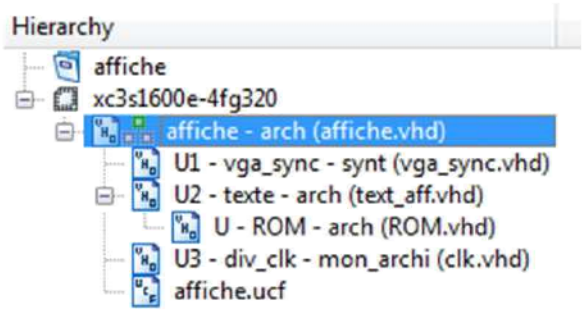


Figure III.11. *La hiérarchie du contrôleur VGA.*

L'entité correspondante est donnée comme suit :

```
entity affiche  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 (2 downto 0)
   );
end affiche;
```

Pour les boutons BT, BT1 de la carte d'évaluation FPGA, chaque bouton affiche une lettre.

- ✓ ***BT = '10'*** affiche la lettre « s »,
- ✓ ***BT = '01'*** affiche la lettre « i »,
- ✓ ***BT1 = '10'*** affiche la lettre « a »,
- ✓ ***BT1 = '01'*** affiche la lettre « r »,
- ✓ ***Vsync***: signal de synchronisation verticale.
- ✓ ***Hsync:*** signal de synchronisation horizontale
- ✓ ***Rgb :*** représente la couleur de la lettre.

IV.2. Les circuits

IV.2.1. Le circuit texte

Afficher un caractère à l'écran implique un appel à l'adresse mémoire du caractère à afficher. 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.12 représente un schéma synoptique du circuit texte.

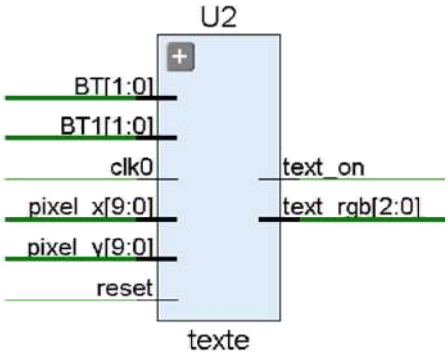


Figure III.12. *Le circuit de texte.*

L'entité correspondante à ce circuit est la suivante :

```
entity texte is
   port(clk0, reset: in std_logic;
        pixel_x, pixel_y: in std_logic_vector(9 downto 0);
        BT,BT1: in std_logic_vector(1 downto 0);
        text_on: out std_logic;
        text_rgb: out std_logic_vector(2 downto 0)
        );
end texte;
```

- ✓ Clk0 : l'horloge générée par la source DIV_clk (25Mhz).
- ✓ Text_on : signal de sortie à un bit.
- ✓ Text_rgb : la couleur de chaque lettre à afficher sur l'écran.

L'instruction qui va afficher le caractère est :

```
text_on <= zone_text_on ;
```

Si en appuyant sur l'un des quatre boutons, une lettre est affichée à l'emplacement donné.

On concatène les deux boutons BT et BT1 dans un seul vecteur BT2 de quatre bits comme suit :

```
BT2 <= BT&BT1;
```

L'affichage de la lettre se fait selon le code suivant:

```
process (BT2, zone_text_on)
begin

if  zone_text_on = '1' then
          if BT2=("1000" ) then
             char_addr_text  <= "1110011";--s

          elsif  BT2=("0100" ) then

                 char_addr_text  <=   "1101001";--i


          elsif BT2=("0010" ) then


                 char_addr_text <= "1100001";--a


          elsif  BT2=("0001" ) then

                 char_addr_text <=  "1110010";--r


          end if;
   end if;
end process;
```

Le circuit "texte" a besoin d'un autre circuit, il s'agit du circuit ROM donné dans la Figure III.13 suivante:

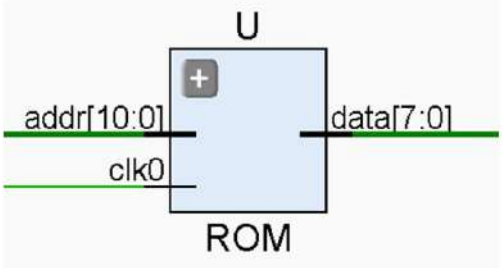


Figure III.13. *Le circuit de ROM de caractères.*

On a une mémoire interne qui contient 128 caractères.

L'entité suivante correspond au circuit ROM

```
entity ROM is
   port(clk0: in std_logic;
      addr: in std_logic_vector(10 downto 0);
      data: out std_logic_vector(7 downto 0)
   );
end ROM;
```

Chaque caractère est de taille 16 x 8, la Figure III.14 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_text de taille 7 bits comme montre l'instruction suivante.

```
char_addr_text  <= "1110011";--s
```

```
                ⋮
-- 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.14. *Exemple du caractère « S » dans la 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(clk0=>clk0, addr=>rom_addr,data=>font_word);
```

La taille des caractères peut être précisée par l'instruction suivante qui lit les caractères de la chaine sans modifier sa taille (16 x 8). Mais on a pu doubler sa taille (32 x 16) comme suit :

```
Ligne_addr_text  <= std_logic_vector(pix_y(4 downto 1));
bit_addr_text  <= 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.

IV.2.2 Le circuit de 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 lettres commandes par les quatre boutons poussoirs.

La Figure III.15 definie le circuit qui génère la synchronisation horizontale et verticale et donne des couleurs à chaque pixel.

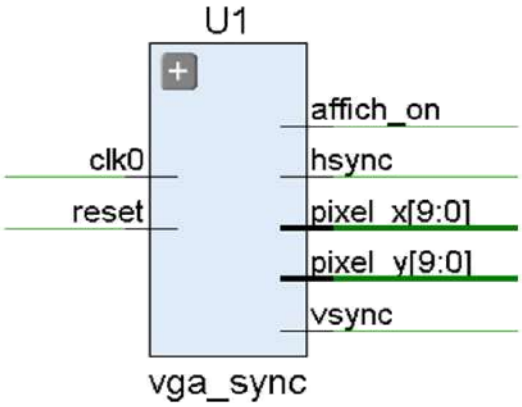


Figure III.15. *Le circuit de VGA_sync*

L'entité correspondante à ce circuit :

```
entity vga_sync is
 port(clk0, reset : in std_logic;
       hsync ,vsync : out std_logic;
      affich_on:out std_logic;
      pixel_x, pixel_y  : out std_logic_vector(9 downto 0)
     );
 end vga_sync;
```

L'organigramme générale est donné dans la Figure III.16. avec :

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.

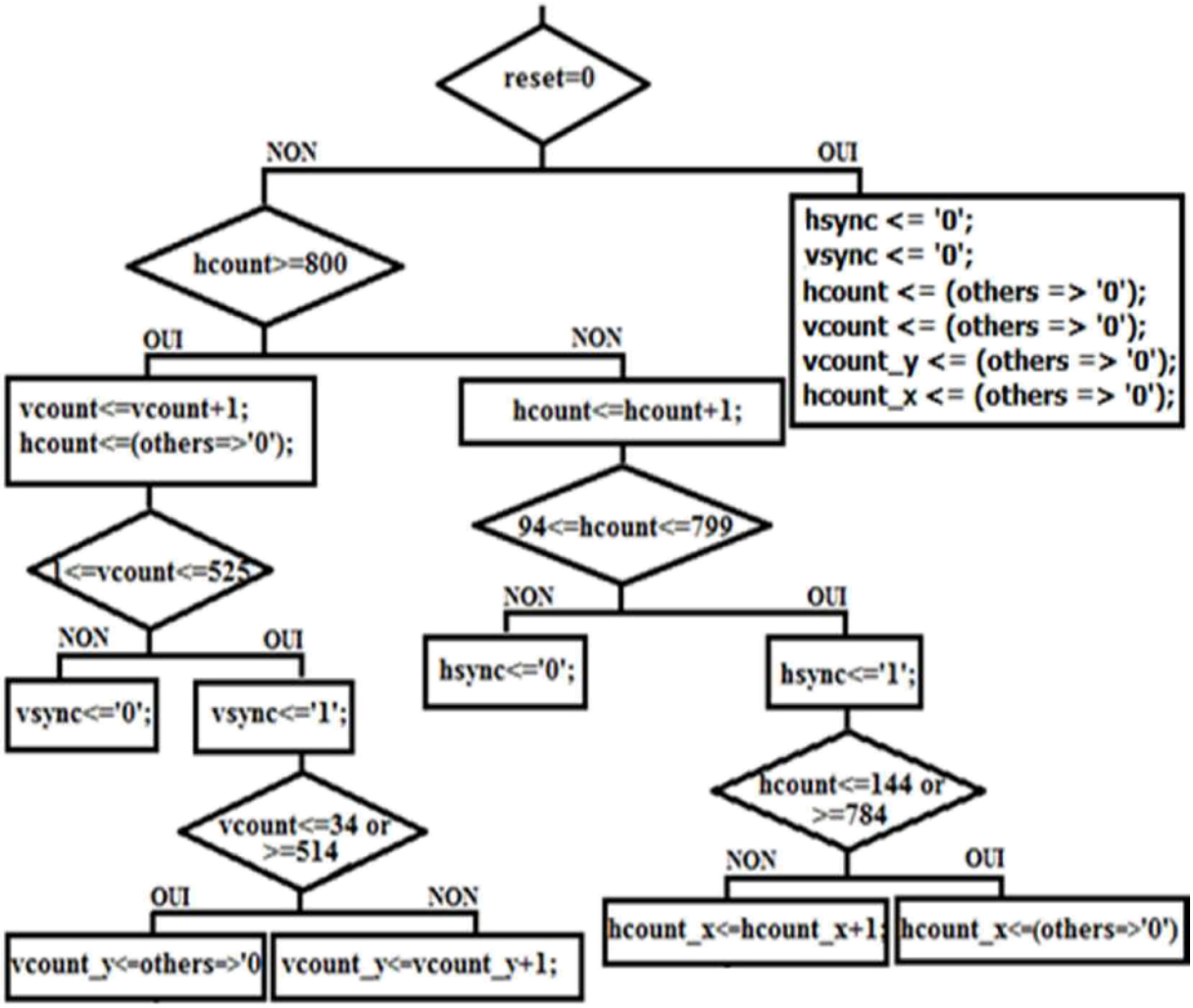


Figure III.16. *L'organigramme général du circuit de synchronisation.*

IV.2.3. Le circuit du diviseur de fréquence

La carte possède une horloge de 50 MHz et étant donné que l'écran a besoin d'une fréquence de 25 MHz, il faut diviser la fréquence par deux. La Figure III.17 représente un schéma synoptique du circuit diviseur de fréquence.

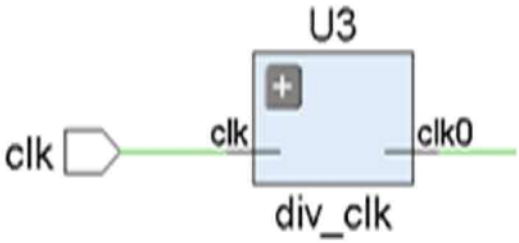


Figure III.17. *Représentation d'un schéma synoptique du circuit div_clk.*

L'entité peut être donnée comme suit,

```
entity div_clk is
    port (
        clk     : in std_logic;
        clk0    : out std_logic
    );
end entity div_clk;
```

- ✓ Clk : représente l'horloge générée par la carte FPGA qui est 50 Mhz.
- ✓ clk0 : est la division de cette clk par 2 c'est à dire 25Mhz.

Les boutons poussoirs utilisées sont montrées par la Figure III.18

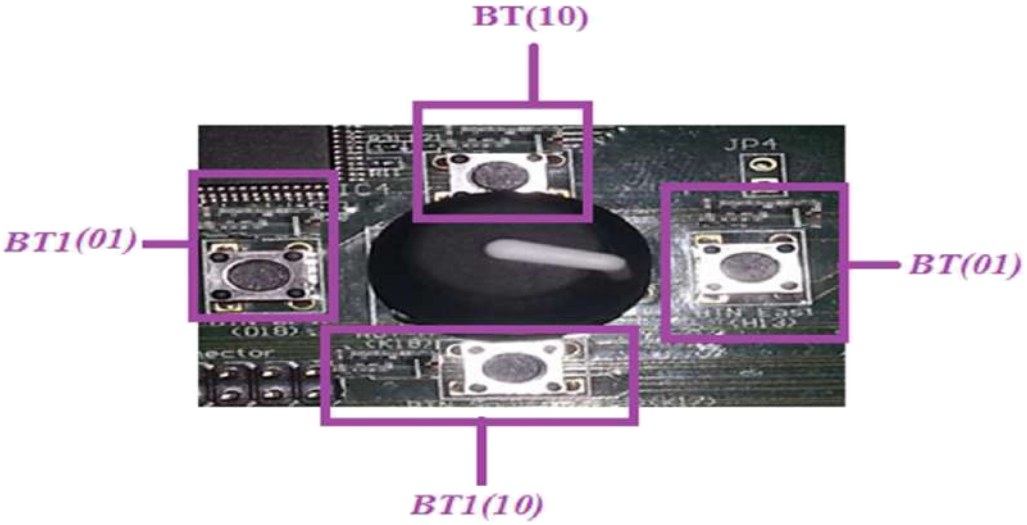


Figure III.18. *Représentation des boutons poussoirs à utilisées.*

Le Switch utilisé comme reset est montré par la Figure III.19

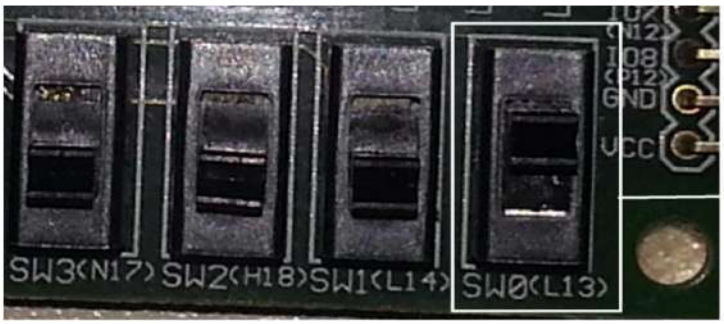


Figure III.19. *Représentation* du Switch.

- On a positionné le caractère en haut à gauche, et de multiplier la taille de la lettre par deux. la Figure III.20 montre la position de la lettre à afficher par exemple « i ».



Figure III.20. *Représentation de la position de la lettre « i ».*

IV.*3.* *Exemple d'implémentation du circuit affiche*

- La touche « i »

Lorsqu'en appuyant sur le bouton BT(01) la matrice de pixel sera affichée sur l'écran comme le montre la Figure III.21.

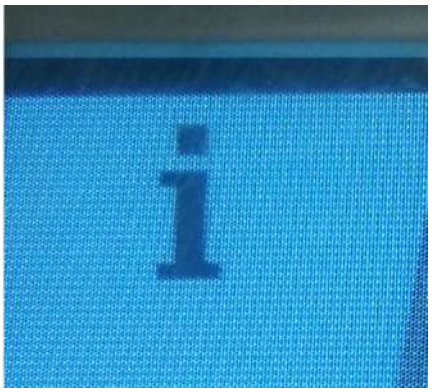


Figure III.21. *Affichage sur écran de la lettre « i ».*

- La touche « s »

Lorsqu'en appuyant sur le bouton BT(10) la matrice de pixel sera affichée sur l'écran comme le montre la Figure III.22.

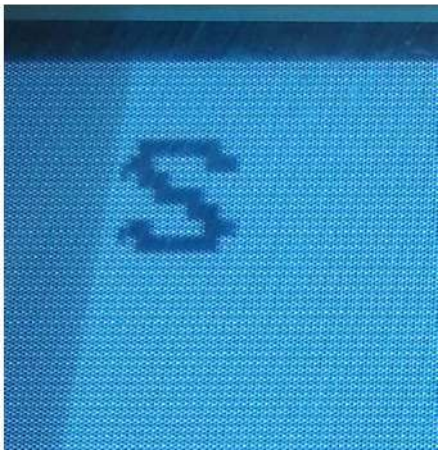


Figure III.22. *Affichage sur écran de la lettre « s ».*

- La touche « r»

Lorsqu'en appuyant sur le bouton BT1(01) la matrice de pixel sera affichée sur l'écran comme le montre la Figure III.23.

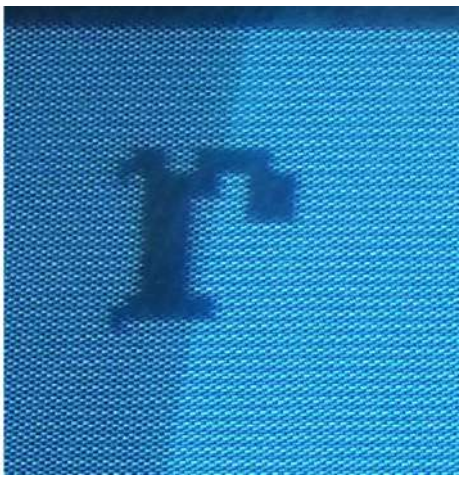


Figure III.23. *Affichage sur écran de la lettre « r ».*

- La touche « a »

Lorsqu'en appuyant sur le bouton BT1(10) la matrice de pixel sera affichée sur l'écran comme le montre la Figure III.24.



Figure III.24. *Affichage sur écran de la lettre « a ».*

La Figure III.25 suivante représente le mini clavier.

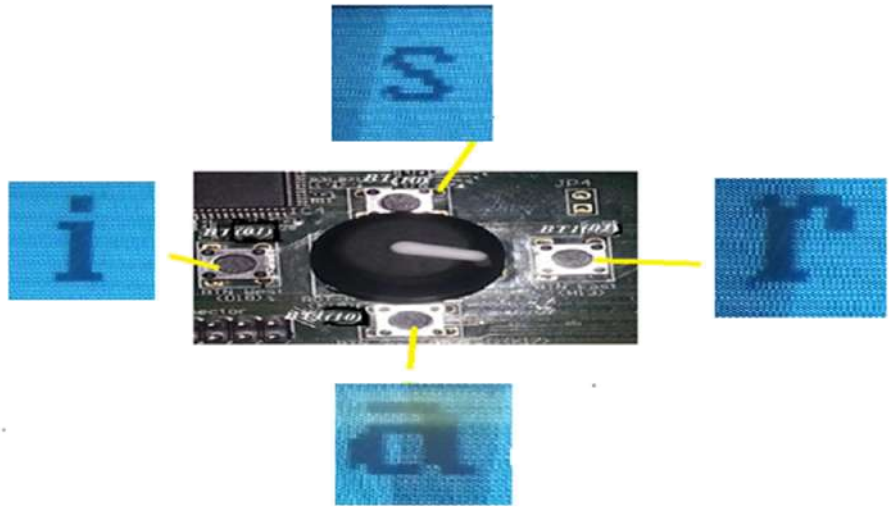


Figure III.25. *Mini clavier.*

IV.4. Implémentation de circuit affiche

On a synthétisé le code VHDL de l'affichage par le logiciel ISE de Xilinx en utilisant la carte 3SD1600e-4FG320 FPGA qui possède une fréquence de 50 Mhz.

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

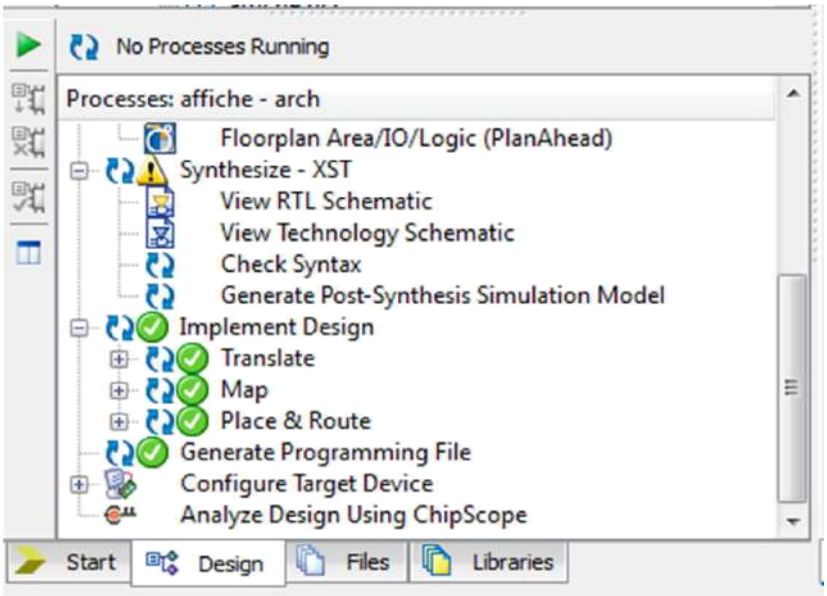


Figure III.26. *Processus d'exécution de circuit affiche.*

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

Tableau III.2. *Rapport d'implémentation de circuit affiche.*

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	84	29,504	1%	
Number of 4 input LUTs	117	29,504	1%	
Number of occupied Slices	84	14,752	1%	
Total Number of 4 input LUTs	126	29,504	1%	
Number of bonded IOBs	11	250	4%	
Number of RAMB16s	1	36	2%	
Number of BUFGMUXs	3	24	12%	

V. Méthodes d'affichages avec clavier

Comme application pour le driver de l'écran, On a réalisé un jeu de pong commandé par un clavier et implémenté sur carte FPGA, nous pouvons utiliser quatre touches du clavier pour y jouer.

V.1. Circuit général de jeu de PONG

La Figure III.27 définie le circuit général de jeu de PONG.

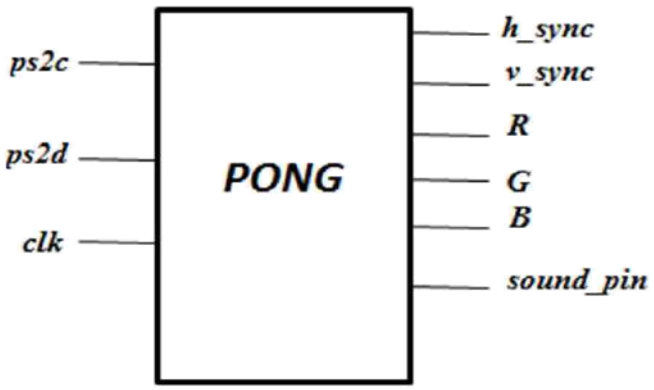


Figure III.27. *Le circuit général du jeu de PONG.*

L'entité peut être donnée comme suit,

```
entity Pong is
    Port (
      clk : in STD_LOGIC;
      h_sync : out STD_LOGIC;
      v_sync : out STD_LOGIC;
      R : out STD_LOGIC;
      G : out STD_LOGIC;
      B : out STD_LOGIC;
      sound_Pin : buffer STD_LOGIC := '0';
      ps2c : in STD_LOGIC;
      ps2d : in STD_LOGIC
   );
end Pong;
```

- ✓ R : La couleur rouge.
- ✓ G : La couleur verte.
- ✓ B : La couleur bleue.

L'hiérarchie de circuit est montré par la Figure III.28.

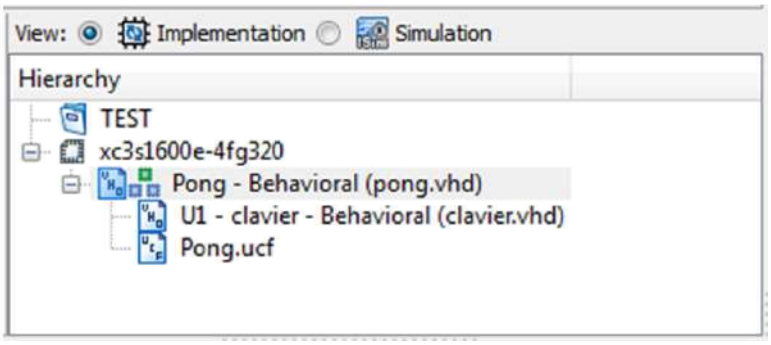


Figure III.28. *L'hiérarchie du circuit général.*

V.2 Gestion du déplacement de la palette

On a débuté notre réalisation du jeu par la coloration d'écran (la balle, les deux palettes, les deux barres de vie et la ligne discontinue) telle que la position de la ligne discontinue au centre de l'écran et la palette gauche entre 22 et 28 verticalement de taille 60 pixels centrée au milieu d'écran commandée par deux touches de clavier, et la palette droite entre 612 et 618 verticalement de taille 60 pixels centrée au milieu d'écran commandée par deux autres touches du clavier. La Figure III.29 représente l'image constituée.



Figure III.29. *L'image qui montre la position initiale des palettes.*

V.3. Les touches qui gérent chaque palette

Chaque palette est gérée par deux touches, son déplacement est réalisé selon l'axe des Y. Une touche pour monter la palette et l'autre pour descendre. En appuyant sur une touche une seule fois, la palette se déplace vers le haut ou vers le bas. La largeur de la palette est fixée à 6 pixels. 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 la touche dans le sens opposé, la Figure III.30 représente les touches responsables de la commande de déplacement des deux palettes.

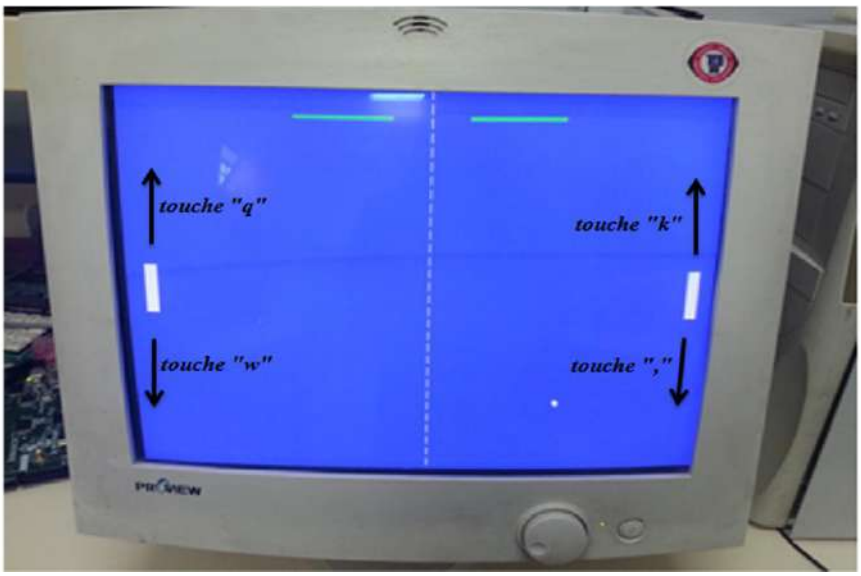


Figure III.30. *Le déplacement des palettes.*

V.4 La barre de vie

Les deux barres de vie sont de taille 100 pixels colorés en vert, telle que la position de la barres de vie droite entre 359 et 459 verticalement et la position de la barres de vie gauche entre 179 et 279 verticalement, chaque manque de balle la barre de vie sera colorée en rouge par 20 pixels à côté de la palette. En tout, on a la possibilité de rater la balle cinq fois. La Figure III.31 représente l'image constituée qui montre qu'on a manqué la balle quatre fois à droite.

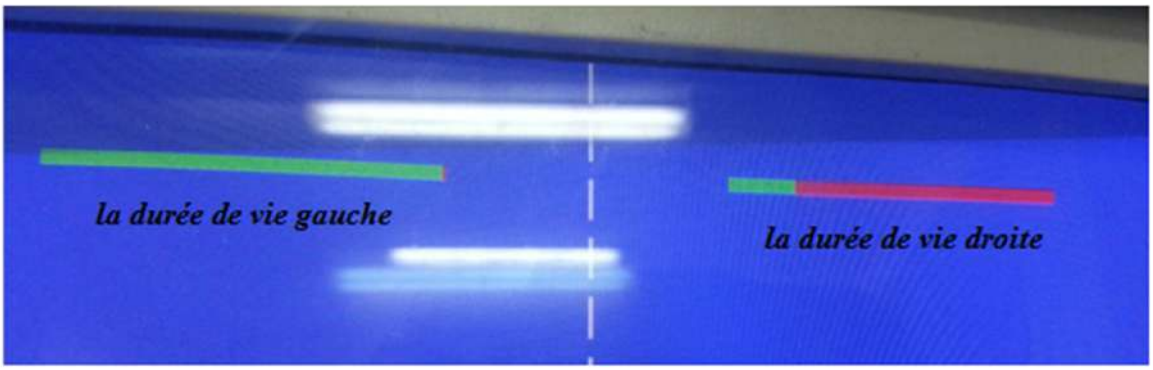


Figure III.31. *Les deux barres de vie.*

V.5 l'affichage des lettres par la coloration d'écran

Lorsque la barre de vie devient rouge un message sera affiché sur l'écran « GAME OVER », Pour affichée ce message on a colorée la zone d'affichage selon la forme des lettres. La Figure III.32 représente l'image à la fin du jeu.



Figure III.32. *L'image à la fin du jeu.*

V.5.1 Méthode de coloration

On a affiché chaque lettre par la spécification de couleur de chaque pixel, pour montrer comment ça marche pour la lettre « G » on colore chaque partie par une couleur différente, comme le montre les instructions suivantes

```
-- G
if pixel_X >= 20 and pixel_X <= 80 and pixel_Y >= 180 and pixel_Y <= 190 then
   color <= "111";
elsif pixel_X >= 20 and pixel_X <= 30 and pixel_Y >= 190 and pixel_Y <= 300 then
   color <= "100";
elsif pixel_X >= 20 and pixel_X <= 80 and pixel_Y >= 290 and pixel_Y <= 300 then
   color <= "110";
elsif pixel_X >= 70 and pixel_X <= 80 and pixel_Y >= 240 and pixel_Y <= 300 then
   color <= "001";
elsif pixel_X >= 50 and pixel_X <= 80 and pixel_Y >= 240 and pixel_Y <= 250 then
   color <= "011";
```

La Figure III.33 montre l'affichage de la lettre « G » colorée.

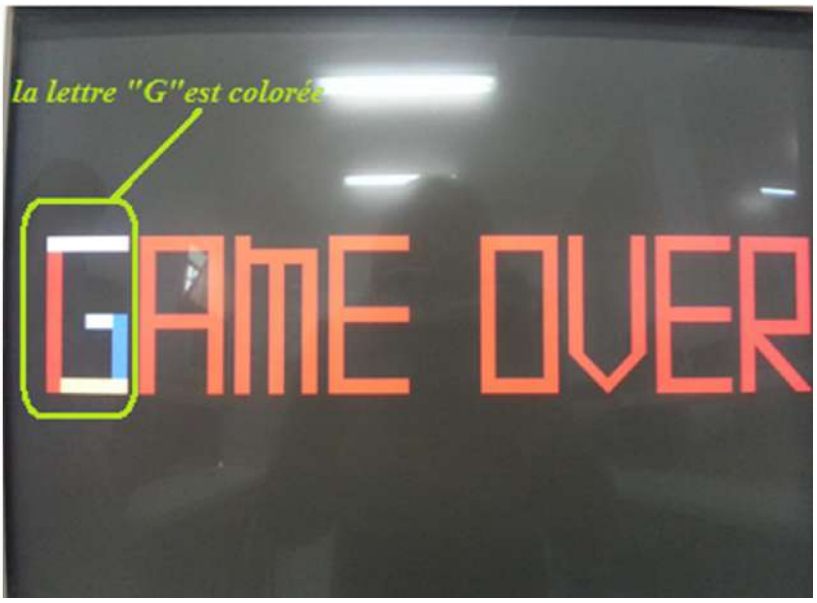


Figure III.33. *Coloration de la lettre « G ».*

La Figure III.34 représente la lettre « G » avec la coloration de chaque partie.

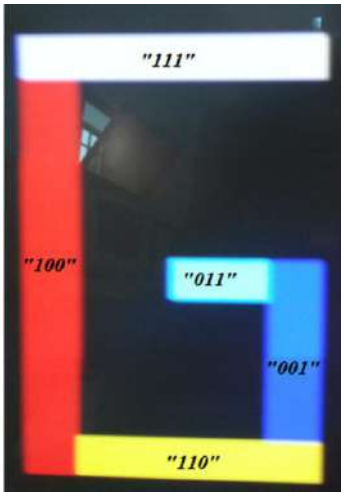


Figure III.34. *Coloration de chaque partie de la lettre « G ».*

V.6 Le circuit clavier

Comme nous avons parlé au paragraphe IV.3.3, que le circuit pong nécessite un autre circuit qui est le clavier ou la figure III.35 montre le schéma détaillé du circuit global.

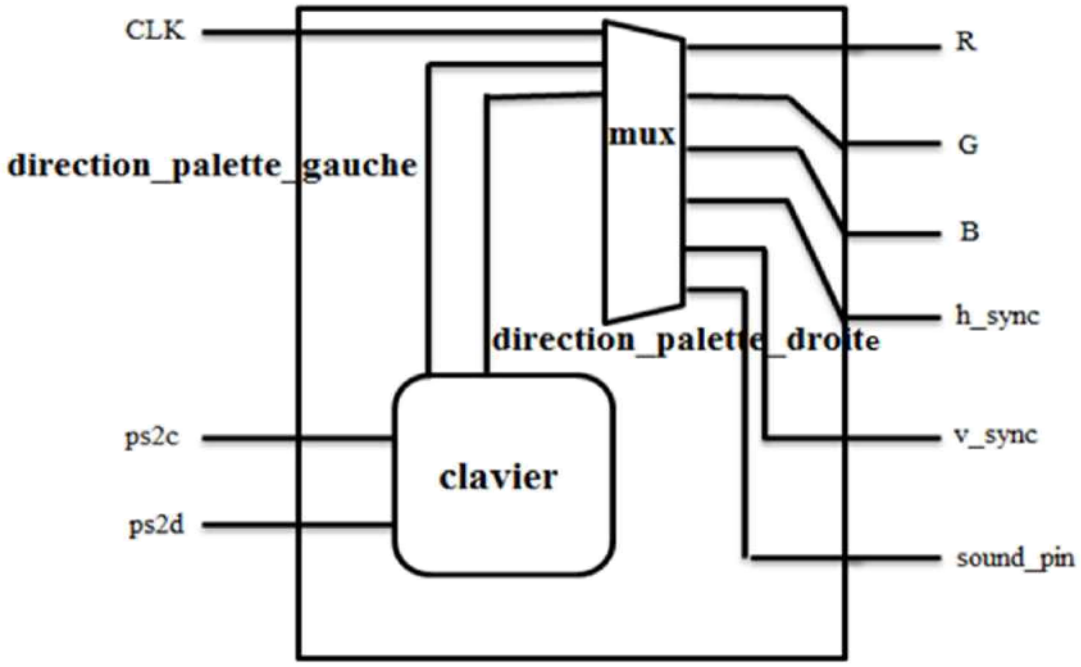


Figure III.35. *Le circuit global de pong.*

La Figure III.36 présente un schéma synoptique du pilote du clavier avec le jeu.

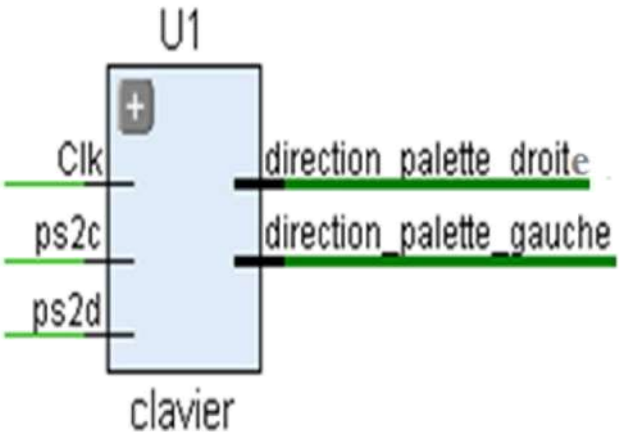


Figure III.36. *Le circuit clavier.*

L'entité peut être donnée comme suit,

```
entity clavier is
    Port ( Clk : in STD_LOGIC;
           ps2c : in  STD_LOGIC;--ps2c
           ps2d: in  STD_LOGIC;--ps2d
           direction_palette_gauche : buffer  integer;--J_gauche
           direction_palette_droite: buffer  integer--J_droit
   );
end clavier;
```

- ✓ Direction_palette_gauche: déplacement de la palette de gauche en fonction des touches «' q' et 'w' ».
- ✓ Direction_palette_droite: déplacement de la palette de droite en fonction des touches « 'k'et ',' ».

On a fait une autre méthode qui sert à prendre la suite des données par le décalage vers la gauche comme montre l'instruction suivant :

```
elsif compteur_bit > 0 and compteur_bit < 9 then -- décaler de un bit dans le code dela gauche
   code_clavier <= ps2d & code_clavier(7 downto 1);
```

La Figure III.37 représente les touches du clavier utilisé.



Figure III.37. *Les touches de clavier utilisées.*

Les instructions suivantes montrent les touches utilisées pour déplacer les deux palettes.

```
constant touch_q : STD_LOGIC_VECTOR(7 downto 0) := "00011100";--touch q
constant touch_w : STD_LOGIC_VECTOR(7 downto 0) := "00011010";--touch w
constant touch_k : STD_LOGIC_VECTOR(7 downto 0) := "01000010";--touch k
constant touch_, : STD_LOGIC_VECTOR(7 downto 0) := "00111010";--touch ,
```

Les deux palettes droite et gauche seront déplacées selon des touches comme le montre les instructions suivantes :

```
if code_clavier = touch_q then
   direction_palette_gauche <= -1;
elsif code_clavier = touch_w then
   direction_palette_gauche <= 1;
elsif code_clavier = touch_k then
   direction_palette_droite<= -1;
elsif code_clavier = touch_, then
   direction_palette_droite<= 1;
end if;
```

V.7 Implémentation de circuit pong

On a synthétisé le code VHDL du pong par le logiciel ISE de Xilinx et le processus d'implémentation est présenté par la Figure III.38 suivante

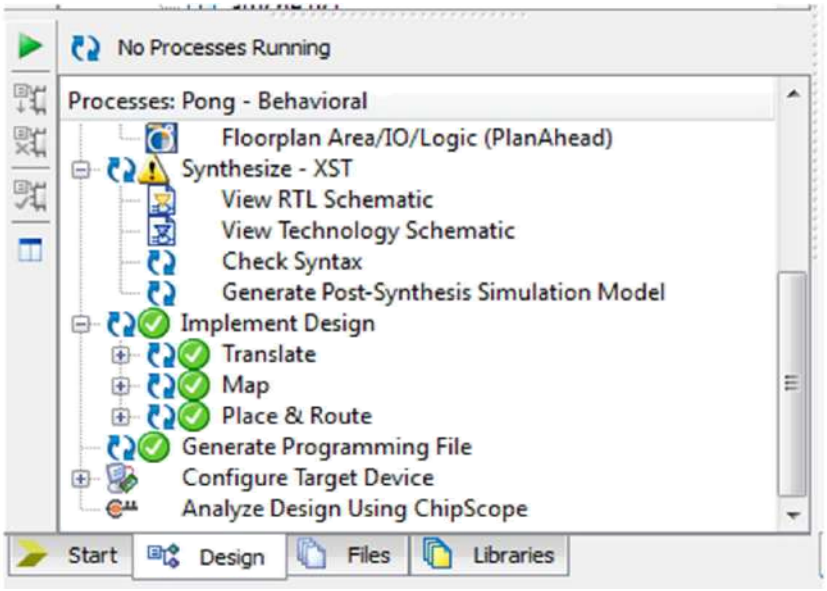


Figure III.38. *Processus d'exécution de circuit pong.*

Et le rapport d'implémentation est donné par le (Tableau III.3).

Tableau III.3. *Rapport d'implémentation de circuit pong.*

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Total Number Slice Registers	271	29,504	1%	
Number of 4 input LUTs	1,559	29,504	5%	
Number of occupied Slices	1,016	14,752	6%	
Total Number of 4 input LUTs	1,878	29,504	6%	
Number of bonded IOBs	9	250	3%	
Number of BUFGMUXs	6	24	25%	

VI. Conclusion

Dans ce chapitre, nous avons présenté en détail tous les circuits utilisés pour une implémentation dans un circuit FPGA, pour l'affichage de caractères sur écran et la gestion du clavier qui est utilisé pour contrôler un jeu simple.

Conclusion générale

Ce projet a eu pour avantage de nous apprendre à mieux appréhender la programmation de circuit FPGA en VHDL. On a pu ainsi se familiariser avec le fonctionnement de la carte Spartan-3 de Xilinx à travers un exemple concret, à savoir la réalisation d'un pilote du clavier.

Les difficultés rencontrées nous ont permis d'étudier le fonctionnement précis d'un clavier mais aussi de connaître de façon détaillée le protocole PS/2. La principale difficulté réside dans le fait de valider notre travail par un affichage VGA sur écran, ce qui nous a amené à étudier le pilote VGA et l'implémenter.

Le travail dans le cadre de ce projet est validé par l'implémentation de trois exemples qui englobe les efforts déployé dans le cadre de ce mémoire de PFE.

Le premier exemple est l'affichage sur LEDs de la carte FPGA du code ASCII de n'importes quelles touches du clavier. Dans notre deuxième exemple, on a utilisé les quatre boutons poussoirs de la carte FPGA pour afficher une lettre sur un écran VGA à une taille et couleurs spécifiées. Et enfin, un jeu de pong est réalisé ou l'on peut jouer avec les touches du clavier et la visualisation sur écran.

Tout au long de ce travail, on a adopté le principe de modularité en VHDL pour le contrôle du clavier et l'affichage.

Comme perspective à notre travail, on propose d'introduire en plus un contrôleur souris. On peut aussi envisager d'implémenter les pilotes des claviers et souris sans fils.

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] 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].

[8] PATRIE Thomas et CARLIER Julien « Projet conception de circuits le jeu de la souris ». 2006 [En ligne]. Disponible: http://joulious.free.fr/RapportProj/Rapport_vhdl.pdf

[9] « Communications Interfacing to the PS/2 Keyboard ». [En ligne]. Disponible: *proton.ucting.udg.mx/~mariocc/cap10.pdf*

[10] « le clavier ». juin 2014 [En ligne]. Disponible: *lacl.univ-paris12.fr/cegielski/bios/ch6.pdf*

[11] John Wiley & Sons. « FPGA PROTOTYPING BY VHDL EXAMPLES Xilinx SpartanTM-3 Version », Hoboken, New Jersey, 2008

[12] « interfacing to the keyboard and mouse » . [En ligne]. Disponible: ***use****rs.ece.gatech.edu/~hamblen/book/.../Chap11.ppt*

Annexes

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, un port PS/2, etc... La documentation de la carte fournit le nom des pins du FPGA reliées à ces entrées et sorties.

- ✓ Les pins de circuit clavier :

```
NET "ps2_data" LOC = G13;

NET "ps2_data" IOSTANDARD = LVCMOS33;

NET "ps2_data" DRIVE = 8;

NET "ps2_clk" LOC = G14;

NET "ps2_clk" IOSTANDARD = LVCMOS33;

NET "ps2_clk" DRIVE = 8;

NET "ps2_clk" CLOCK_DEDICATED_ROUTE = FALSE;

NET "bit_8" LOC = A8;

NET "bit_7" LOC = G9;

NET "bit_6" LOC = A7;

NET "bit_5" LOC = D13;

NET "bit_4" LOC = E6;

NET "bit_3" LOC = D6;

NET "bit_2" LOC = C3;

NET "bit_1" LOC = D4;

NET "bit_1" IOSTANDARD = LVTTL;
```

NET "bit_1" DRIVE = 8;

NET "bit_2" IOSTANDARD = LVTTL;

NET "bit_2" DRIVE = 8;

NET "bit_3" IOSTANDARD = LVTTL;

NET "bit_3" DRIVE = 8;

NET "bit_4" IOSTANDARD = LVTTL;

NET "bit_4" DRIVE = 8;

NET "bit_5" IOSTANDARD = LVTTL;

NET "bit_5" DRIVE = 8;

NET "bit_6" IOSTANDARD = LVTTL;

NET "bit_6" DRIVE = 8;

NET "bit_7" IOSTANDARD = LVTTL;

NET "bit_7" DRIVE = 8;

NET "bit_8" IOSTANDARD = LVTTL;

NET "bit_8" DRIVE = 8;

- ✓ Les pins de circuit affiche :

NET "BT[0]" LOC = D18;

NET "BT[1]" LOC = V4;

NET "BT1[0]" LOC = H13;

NET "BT1[1]" LOC = K17;

NET "reset" LOC = L13;

NET "rgb[0]" LOC = H14;

NET "rgb[1]" LOC = H15;

```
NET "rgb[2]" LOC = G15;

NET "hsync" LOC = F15;

NET "vsync" LOC = F14;

NET "rgb[0]" IOSTANDARD = LVTTL;

NET "rgb[0]" DRIVE = 8;

NET "rgb[0]" SLEW = FAST;

NET "rgb[1]" IOSTANDARD = LVTTL;

NET "rgb[2]" IOSTANDARD = LVTTL;

NET "clk" IOSTANDARD = LVCMOS33;

NET "hsync" IOSTANDARD = LVTTL;

NET "vsync" IOSTANDARD = LVTTL;

NET "rgb[1]" DRIVE = 8;

NET "rgb[2]" DRIVE = 8;

NET "hsync" DRIVE = 8;

NET "vsync" DRIVE = 8;

NET "rgb[1]" SLEW = FAST;

NET "rgb[2]" SLEW = FAST;

NET "hsync" SLEW = FAST;

NET "vsync" SLEW = FAST;

NET "BT[0]" IOSTANDARD = LVTTL;

NET "BT[1]" IOSTANDARD = LVTTL;

NET "BT1[1]" IOSTANDARD = LVTTL;

NET "BT1[0]" IOSTANDARD = LVTTL;
```

```
NET "BT[1]" PULLDOWN;

NET "BT[0]" PULLDOWN;

NET "BT1[0]" PULLDOWN;

NET "BT1[1]" PULLDOWN;

NET "clk" LOC = C9;

NET "reset" IOSTANDARD = LVTTL;

NET "reset" PULLUP;
```

- ✓ Les pins de circuit PONG :

```
NET "B" LOC = G15;

NET "clk" LOC = C9;

NET "G" LOC = H15;

NET "h_sync" LOC = F15;

NET "R" LOC = H14;

NET "v_sync" LOC = F14;

NET "ps2c" LOC = G14;

NET "sound_Pin" LOC = A8;

NET "ps2c" IOSTANDARD = LVCMOS33;

NET "ps2c" DRIVE = 8;

NET "ps2c" CLOCK_DEDICATED_ROUTE = FALSE;

NET "ps2d" LOC = G13;

NET "ps2d" IOSTANDARD = LVCMOS33;

NET "ps2d" DRIVE = 8;

NET "clk" IOSTANDARD = LVCMOS33;
```

```
NET "B" IOSTANDARD = LVTTL;

NET "B" SLEW = FAST;

NET "B" DRIVE = 8;

NET "G" IOSTANDARD = LVTTL;

NET "G" DRIVE = 8;

NET "G" SLEW = FAST;

NET "h_sync" IOSTANDARD = LVTTL;

NET "h_sync" DRIVE = 8;

NET "h_sync" SLEW = FAST;

NET "R" IOSTANDARD = LVTTL;

NET "R" DRIVE = 8;

NET "R" SLEW = FAST;

NET "v_sync" IOSTANDARD = LVTTL;

NET "v_sync" DRIVE = 8;

NET "v_sync" SLEW = FAST;

NET "sound_Pin" IOSTANDARD = LVTTL;

NET "sound_Pin" SLEW = SLOW;

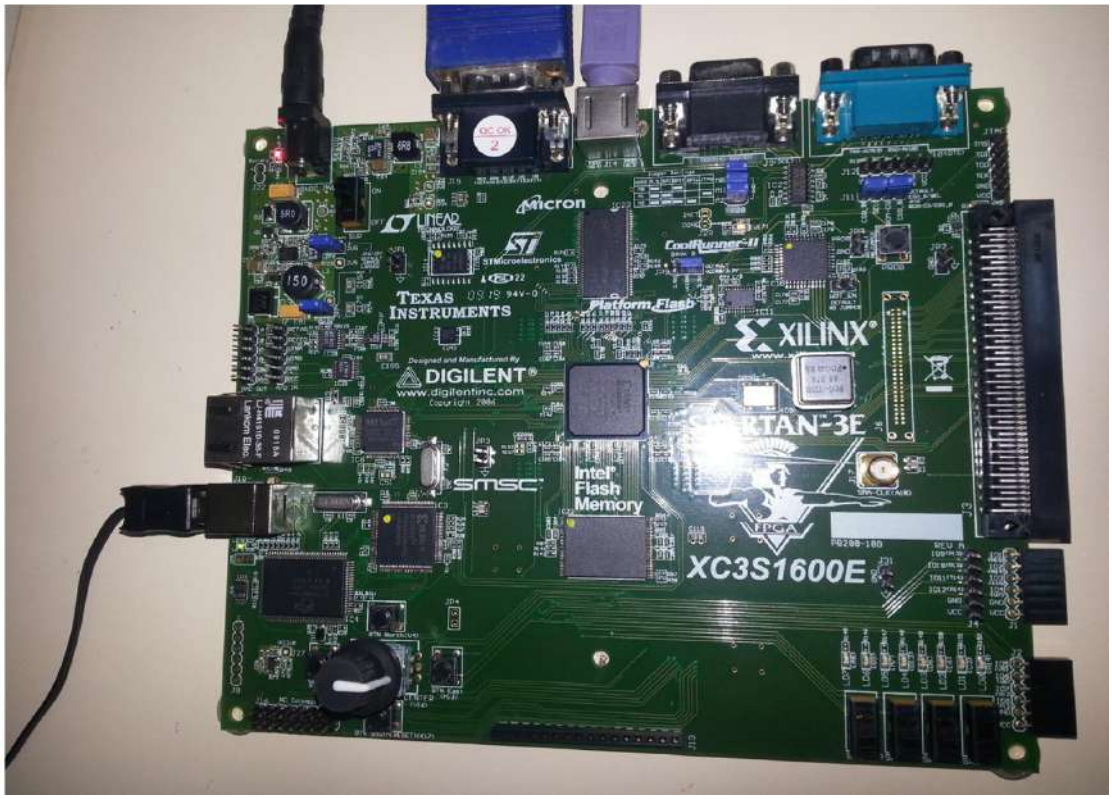
NET "sound_Pin" DRIVE = 8;
```

- ***Ecran utilisé***

Proview PX -772



- ***La carte utilisée***:



- **Les connecteurs d'entrée-sortie**

✓ **Port PS/2 et port VGA**



✓ Câble PS/2



✓ Les connecteur clavier,VGA

