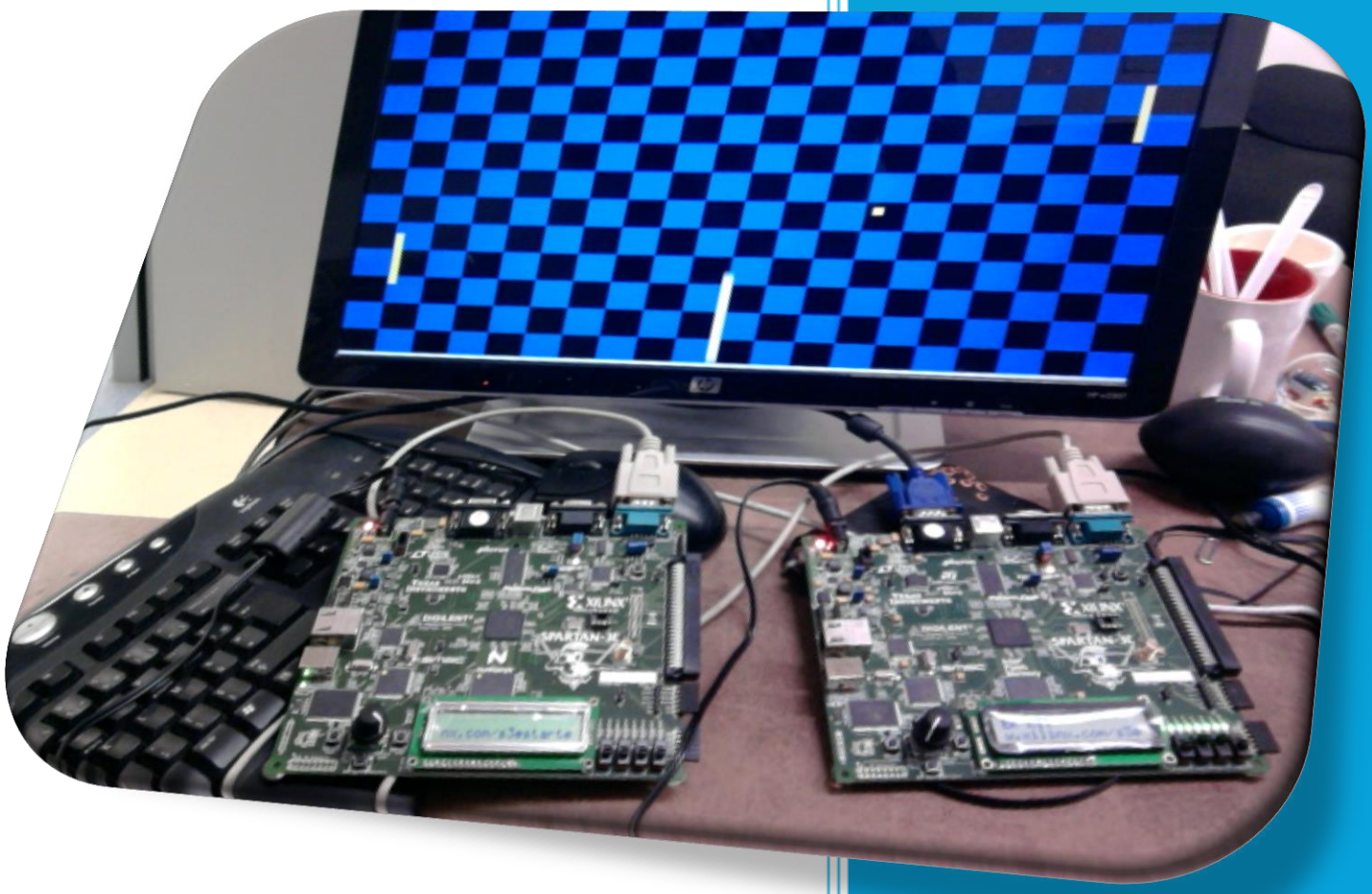
SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

LU3EE100

Console SU-EE100 sur FPGA



Documentation Technique (Etudiants)

Julien Denoulet

SU-EE100 – Mini Projet

L3 EEA

PRESENTATION GENERALE DE LA CONSOLE SU-EE100

1) Introduction

La console SU-EE100 contient deux jeux comptant parmi les plus célèbres de l'histoire des jeux vidéo

- Casse-Briques (pour 1 joueur)

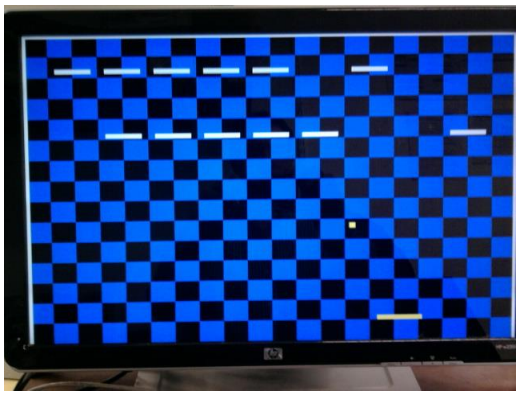


Figure 1 – Casse Briques

- Pong (pour 2 joueurs)

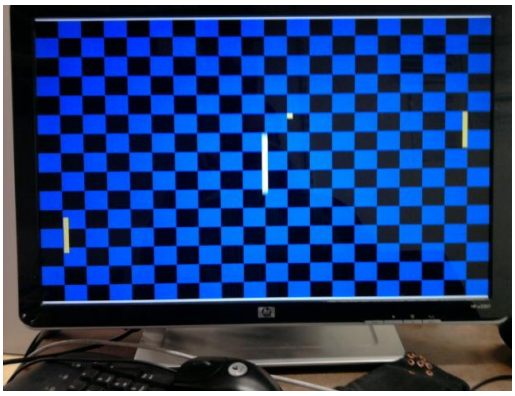


Figure 2 – Pong

L'utilisateur peut à tout moment choisir le jeu auquel il a envie de jouer, fixer un certain nombre d'options (vitesse de la balle, taille des raquettes…) et également configurer la SU-EE100 en mode console ou en mode manette.

Pour déplacer la raquette du Casse-Briques, le joueur peut utiliser l'accéléromètre présent sur la carte ou bien un encodeur rotatif connecté à la console. Pour le jeu Pong, il faut utiliser deux SU-EE100 : l'une en mode console, et l'autre en mode manette. La manette se connecte à la console via un câble.

L'affichage se fait sur un écran grâce à la sortie VGA de la console.

Figure 3 : Borne d'arcade originale du jeu Pong (1972)

2) Présentation des jeux

Figure 4 : Borne d'arcade jeu Casse Briques (1975)

- ***Casse Briques***
  - Le principe général de ce jeu est de détruire, à l'aide d'une balle, un ensemble de briques se trouvant dans la partie supérieure de l'écran.
  - Pour cela, le joueur contrôle une raquette qu'il peut seulement déplacer sur un axe horizontal au bas de l'écran.
  - Le but est d'empêcher la balle de franchir cette ligne en la frappant avec la raquette.
  - Si le joueur ne parvient pas à rattraper la balle :
    - il perd la partie. Cela se traduit par un écran rouge pendant quelques secondes.
    - La balle est alors remise en jeu pour donner une nouvelle chance au joueur.
  - Si le joueur parvient à casser toutes les briques,
    - La partie est gagnée. L'écran devient entièrement vert.
    - Pour recommencer une partie, il faut réinitialiser le jeu en faisant un Reset de la console
  - Il y a par ailleurs plusieurs options de jeu :
    - Réglage de la taille de la raquette
    - Réglage de la vitesse de la balle
    - Mise en pause du jeu

- ***Pong***
  - Pong est une simulation simpliste de tennis de table (ping-pong) qui se joue à deux.
    - Le 1er joueur utilise l'encodeur rotatif ou l'accéléromètre de la SU-EE100 configurée en mode Console. Il contrôle ainsi la raquette située à gauche de l'écran
    - Le 2ème joueur utilise l'encodeur rotatif d'une autre SU-EE100, configurée en mode Manette, et reliée à l'autre SU-EE100 via un câble. Le 2ème joueur contrôle ainsi la raquette située à droite de l'écran. A noter qu'il n'est pas possible d'utiliser l'accéléromètre en mode Manette.
  - Une balle se déplace à travers l'écran, rebondissant sur les rebords du haut et du bas
  - Les deux joueurs commandent chacun une raquette, la faisant glisser verticalement dans l'écran.
  - Si la balle frappe la raquette, elle rebondit vers l'autre joueur.
  - Si le joueur manque la balle avec sa raquette, il perd la partie. Cela se traduit par un écran rouge pendant quelques secondes.
  - Il y a par ailleurs plusieurs options de jeu :
    - Réglage de la taille de la raquette
    - Réglage de la vitesse de la balle
    - Insertion d'un obstacle mobile au milieu de l'écran. Si la balle rebondit contre cet obstacle, elle repart dans le sens inverse.
    - Mise en pause du jeu

3) Interface joueur

Pour déplacer les raquettes des jeux, le joueur peut utiliser l'accéléromètre présent sur la carte ou bien un encodeur rotatif (Figure 5.1 ci-contre).

- L'encodeur tourne vers la gauche ou la droite.
- Il est relié à la carte via le connecteur ***PMOD B*** ou ***PMOD C*** selon que la carte est configurée en mode Console (PMOD B) ou Manette (PMOD C).

L'interface joueur de la SU-EE100 se compose également de (Figure 5.2):

- 16 interrupteurs dont 5 sont utilisés :
  - ***S15, S3, S2, S1*** et ***S0***.
  - ***S15*** est l'interrupteur le plus à gauche.
- 5 boutons poussoirs
  - ***Nord, Sud, Est, Ouest*** et ***Centre***

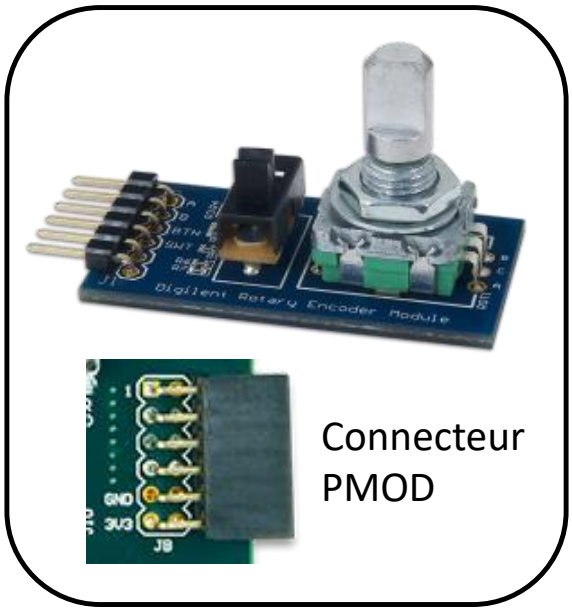


Figure 5.1 : Encodeur rotatif et connecteur PMOD

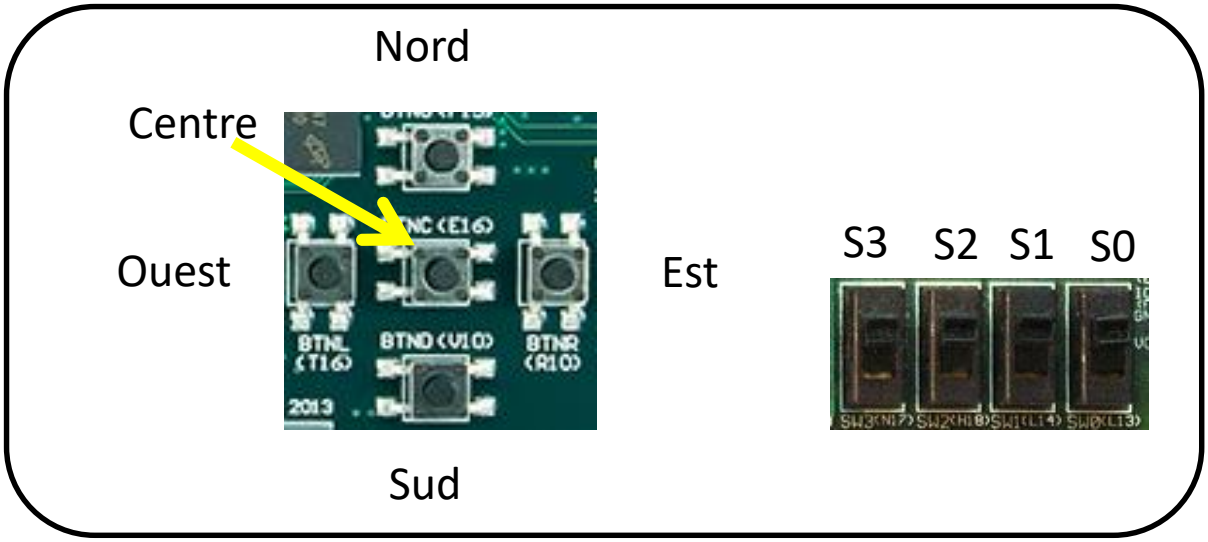


Figure 5.2 : Interface joueur de la SU-EE100

- ***Commandes générales de la SU-EE100***
  - ***S0*** : Commande Marche/Arrêt de la console
    - Interrupteur vers le bas → Arrêt
    - Interrupteur vers le haut → Marche
  - ***S15*** : Choix du composant pour le déplacement des raquettes
    - Interrupteur vers le bas → Encodeur rotatif
    - Interrupteur vers le haut → Accéléromètre

- ***Ouest*** : Reset système
  - Actif si on appuie sur le bouton
  - Permet de redémarrer une nouvelle partie de Casse Briques après une partie gagnée.
- ***Sud***: Sélection du mode de la SU-EE100
  - L'appui sur le bouton permet de passer du mode Console au mode Manette et inversement.
- ***Nord*** ou ***Est***: Sélection du jeu actif
  - L'appui sur l'un des boutons permet de passer du jeu Casse Briques au jeu Pong et inversement.
- ***Centre***: Mise en pause du jeu.
  - L'appui sur le bouton permet de mettre le jeu en pause.
  - Un nouvel appui permet de reprendre le jeu.

- ***Commandes communes aux deux jeux***
  - ***S3*** : Commande de la taille des raquettes
    - Interrupteur vers le bas → Raquettes courtes
    - Interrupteur vers le haut → Raquettes longues
  - ***S2*** : Commande de la vitesse de la balle
    - Interrupteur vers le bas → Vitesse lente
    - Interrupteur vers le haut → Vitesse rapide

- ***Commande spécifique au jeu Casse Briques***
  - ***Encodeur rotatif ou accéléromètre***:
    - Déplacement horizontal de la raquette

- ***Commandes spécifiques au jeu Pong***
  - ***Accéléromètre (carte console seulement) ou encodeur rotatif (pour les deux joueurs)***:
    - Déplacement vertical de la raquette selon le
  - ***S1*** : Ajout d'un obstacle mobile
    - Interrupteur vers le bas → Pas d'obstacle
    - Interrupteur vers le haut → Présence d'un obstacle

Figure 6 – La Magnavox Odyssey, 1ère console vidéo (1972)

4) Plate-forme matérielle

La console de jeux SU-EE100 est implémentée sur une carte Xilinx Nexys 4 (Figure 7).

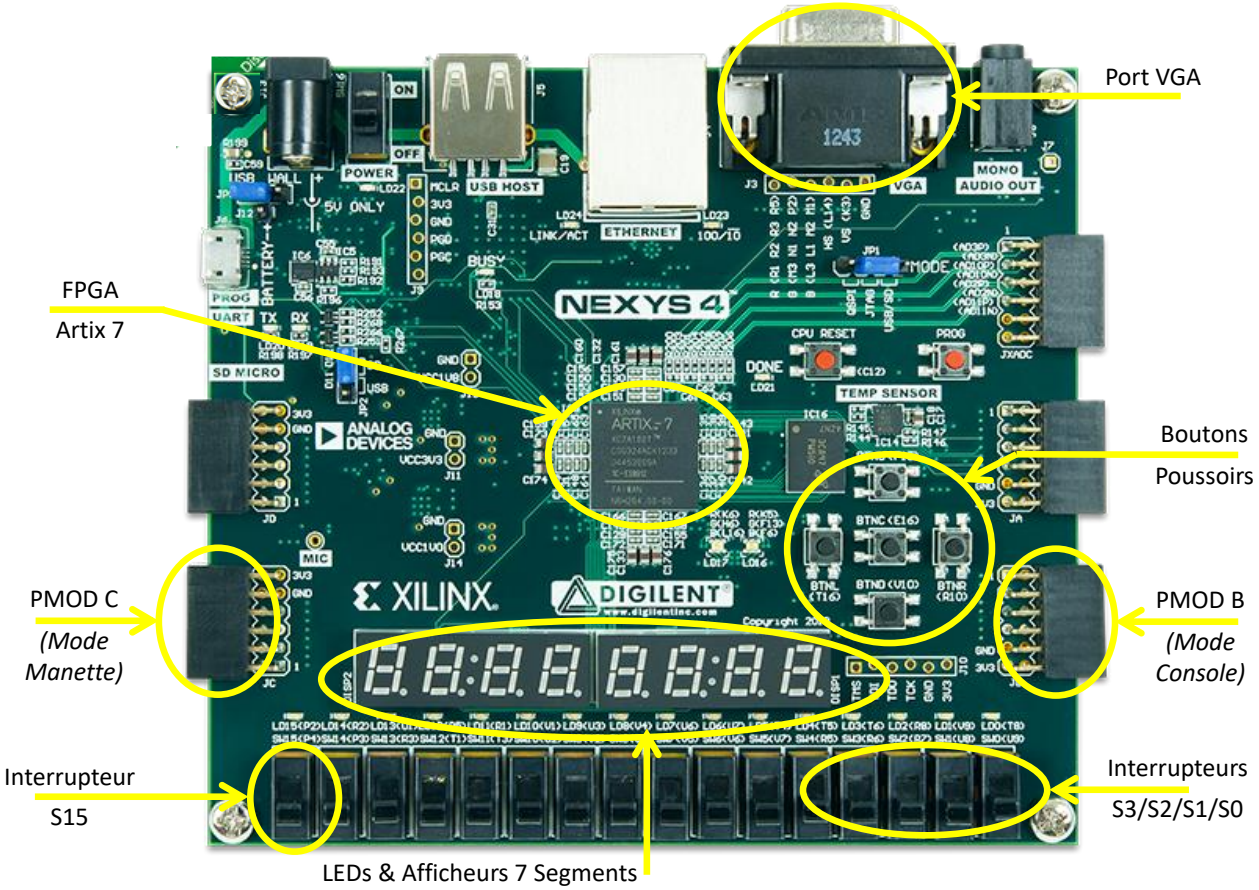


Figure 7 – Carte Nexys 4

Cette carte dispose des éléments suivants

- **FPGA Artix 7**
  - Ce composant programmable contient toute l'architecture numérique de la console
  - Il est cadencé par une horloge interne de fréquence **100 MHz**
- **3 Connecteurs PMOD**
  - PMOD B : On y connecte l'encodeur rotatif en mode Console
  - PMOD D : On y connecte l'encodeur rotatif en mode Manette
  - Pour jouer à 2 joueurs, les 2 PMOD doivent également être reliées par un câble RJ45.
- **Boutons Poussoirs**
  - Les 5 boutons poussoirs (Nord, Sud, Est, Ouest et Centre), permettent de paramétrer la console
- **Interrupteurs**
  - Ces 5 interrupteurs (S15 et S3-S0) permettent également de paramétrer l'activité de la console
- **Afficheurs 7 segments**
  - Fournissent des messages (mise en pause, jeu activé…)
- **LED**
  - Les 16 LED indiquent si la SU-EE100 est en mode console (allumées) ou manette (éteintes).
- **Port VGA**
  - Ce connecteur permet à la carte d'envoyer des signaux vidéo vers un écran VGA.
-

- **Accéléromètre**
    - Ce composant se trouve sur la face arrière de la carte Nexys 4.

5) Architecture générale du FPGA

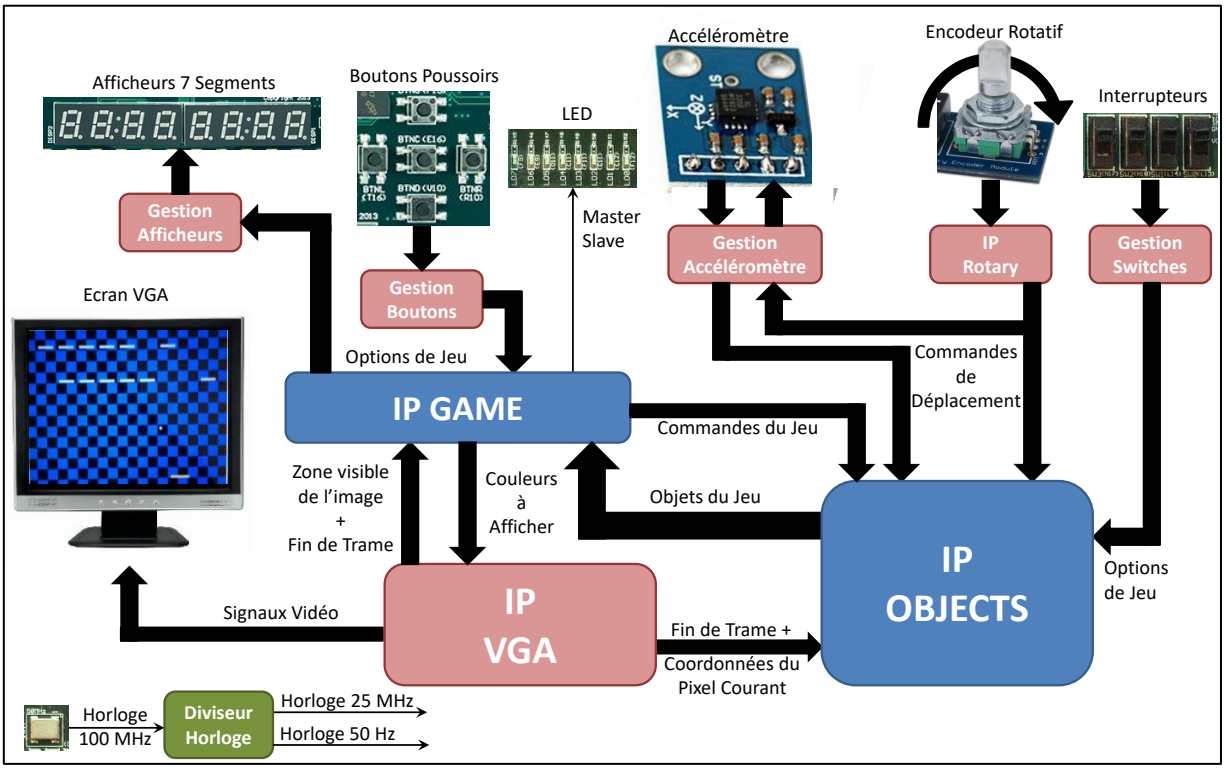


Figure 8 – Architecture interne du FPGA

L'architecture de la console implémentée dans le FPGA est donnée en Figure 8. On distingue dans ce schéma-blocs simplifié 8 modules.

1) Diviseur Horloge

- Ce module comprend en fait 2 blocs qui génèrent 2 horloges à partir de l'horloge référence de 100 MHz. Une horloge 25 MHz pour cadencer la quasi-totalité des modules de l'architecture, et une horloge à 25 Hz pour traiter les signaux issus de l'accéléromètre.

2) Gestion Boutons

- Ce module détecte les événements survenant sur les 5 boutons poussoirs et génère en conséquence plusieurs signaux pour modifier le comportement de la console (reset système, sélection mode console/manette, choix du jeu, demande de pause…)

3) Gestion Switches

- Ce module détecte les événements survenant sur les 5 interrupteurs et génère en conséquence plusieurs signaux pour modifier le comportement de la console (marche /arrêt du système, vitesse de la balle, taille des raquettes, choix entre l'encodeur rotatif et l'accéléromètre…)

4) IP Rotary

- Ce module détecte les événements survenant sur l'encodeur rotatif (rotary encoder en Anglais) connecté à la carte et génère en conséquence des commandes de déplacement pour les objets du jeu (les raquettes) qui sont pilotés par cet encodeur.

5) Gestion Accéléromètre

- Ce module détecte les déplacements de la carte mesurés par l'accéléromètre et génère en conséquence des commandes de déplacement pour les objets du jeu (les raquettes) qui sont pilotés par cet accéléromètre.

6) IP VGA

- Ce module envoie à l'écran VGA les signaux vidéo permettant d'afficher les images du jeu. La couleur des pixels est donnée en entrée par le module **IP Game**.
- Le module transmet également aux autres blocs de l'architecture des informations sur l'image qui est en train d'être affichée (coordonnées du pixel courant, fin de l'image…)

7) Gestion Afficheurs

- Ce module contrôle les 8 afficheurs 7 segments de la carte pour afficher des messages d'information au joueur.

8) IP Objects

- Ce module a pour but de générer tous les objets nécessaires au jeu Pong ou Casse-Briques (Raquettes, balle, murs, briques…). Pour cela, il prend en entrée :
  - Des commandes de jeu transmises par **IP Game** et **Gestion Switches**
  - Des commandes de déplacement transmises par **IP Rotary** et **Gestion Accéléromètre**
  - Des informations sur l'image à afficher transmises par **IP VGA**.

9) IP Game

- Ce module génère :
  - Des commandes vers **IP Objects**, **IP RS232** et les **LED** donnant le comportement général de la SU-EE100 (mode console/manette, choix du jeu, mode pause…)
  - La couleur du pixel à afficher. Cette information est transmise à **IP VGA**.
- Pour cela, il reçoit en entrée des informations provenant des autres modules du système.

ARCHITECTURE DETAILLEE DE LA CONSOLE SU-EE100

IMPORTANT - Il est à noter que

1) ***Toutes les horloges sont actives sur leur front montant***
2) ***Tous les resets asynchrones sont actifs au niveau bas***
3) ***Sauf mention contraire, les autres commandes sont actives au niveau haut.***

1) Diviseur Horloge

Rôle : Génération des horloges 25 MHz et 25 Hz nécessaires au fonctionnement de la console.

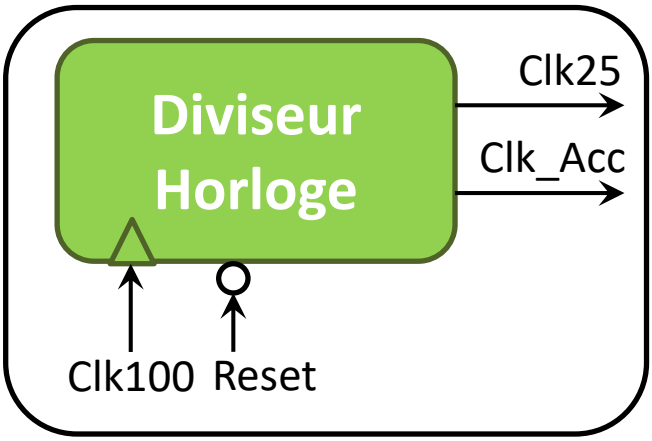


Figure 9 – Vue externe du module Diviseur Horloge

Entrées :

- ***Clk100*** : Horloge 100 MHz fournie par la carte
- ***Reset*** : Reset asynchrone

Sortie :

- ***Clk25*** : Horloge de sortie à 25 MHz
- ***Clk_Acc*** : Horloge de sortie à 25 Hz

Fonctionnement :

- Des compteurs cadencés par l'horloge ***Clk100*** comptent le nombre de cycles pendant lesquels les horloges de sortie doivent être au niveau haut ou bas.

NB : Dans les fichiers source de la console, L'horloge Clk25 est fournie par le module d'entité ClkDiv. L'horloge Clk_Acc est fournie par le module d'entité ClkAcc.

2) Gestion Boutons

Rôle : Ce bloc permet la gestion des 5 boutons poussoirs de la carte. Selon les cas, il va indiquer le niveau logique ou bien si l'on a appuyé sur le bouton. La disposition des boutons et interrupteurs est donnée en Fig.10.

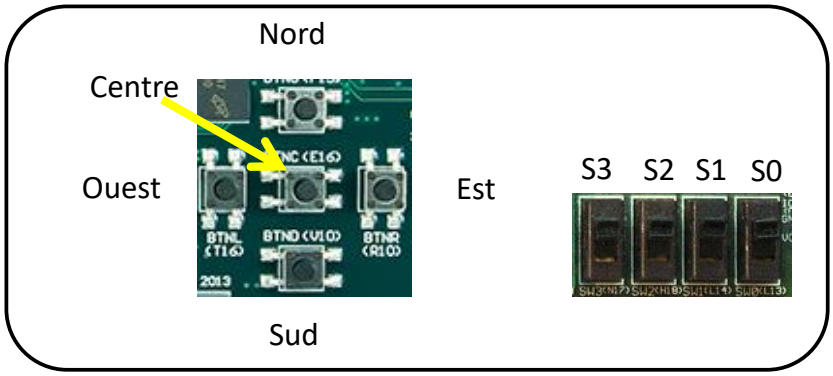


Figure 10 - Disposition des boutons poussoirs et des interrupteurs (S15 est tout à gauche de la rangée d'interrupteurs)

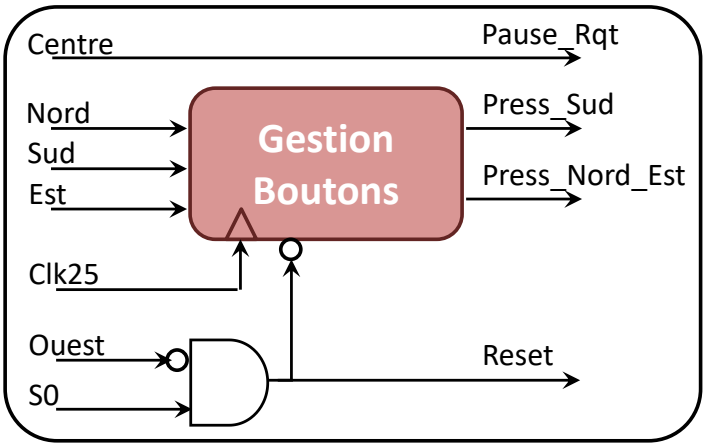


Figure 11 – Vue externe du module Gestion Boutons

Entrées :

- ***Clk25*** : Horloge 25MHz
- ***Centre, Nord, Sud, Est, Ouest***: Boutons
- ***S0***: Interrupteur

Sorties :

- ***Reset***: Reset asynchrone du système
- ***Pause_Rqt***: Demande de pause du jeu
- ***Press_Sud***: Détection de l'appui sur le bouton Sud
- ***Press_Nord_Est***: Détection de l'appui sur le bouton Nord ou le bouton Est

Fonctionnement :

- Le ***Reset*** (actif au niveau bas) est activé si l'une de ces deux conditions est remplie :
  - L'interrupteur ***S0*** est tiré vers le bas (niveau logique 0)
  - L'interrupteur ***S0*** est tiré vers le haut et le bouton poussoir ***Ouest*** est appuyé (niveau logique haut)
  - ➢ L'interrupteur ***S0*** sert ainsi de commande Marche/Arrêt de la console SU-EE100

- La sortie ***Pause_Rqt*** est directement connectée ay bouton ***Centre.***
- La commande ***Press_Sud*** est activée pendant un cycle d'horloge dès que l'on détecte un appui sur le bouton ***Sud***. La commande repasse ensuite automatiquement à 0
- La commande ***Press_Nord_Est*** est activée pendant un cycle d'horloge dès que l'on détecte un appui sur l'un des deux boutons ***Nord*** ou ***Est***. La commande repasse ensuite automatiquement à 0

- Gestion de l'anti-rebond des boutons ***Nord***, ***Sud*** et ***Est***.
  - Dès que l'on détecte que l'un des 3 boutons est appuyé, une temporisation est déclenchée. Pendant cette temporisation, on ne peut plus détecter d'autre action sur les boutons poussoirs.
  - A la fin de cette temporisation, le mécanisme de détection est réamorcé dès que le bouton sur lequel on a appuyé a été relâché.

➢ ***Press_Sud*** servira à faire un changement pour passer la SU-EE100 du mode Console au mode Manette.
➢ ***Press_Nord_Est*** servira à demander un changement du jeu sur la console (Casse Briques ou Pong)

3) Gestion Switches

Rôle : Ce bloc permet la gestion des interrupteurs (à part S0, utilisé dans l'IP Gestion Boutons). Il permet de fixer des options pour les jeux Casse-Briques et Pong

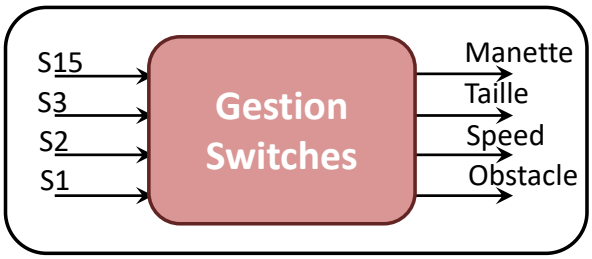


Figure 12 – Vue externe du module Gestion Switches

Entrées :

- ***S15, S3, S2, S1***: Interrupteurs de la carte Nexys4

Sorties :

- ***Manette***: Sélection encodeur / accéléromètre
- ***Taille***: Fixe la taille des raquettes
- ***Speed***: Fixe la vitesse de la balle
- ***Obstacle***: Ajoute un obstacle mobile au milieu de l'écran (Jeu Pong uniquement)

Fonctionnement :

- Chacune des trois sorties est directement connectée à l'un des interrupteurs de la carte. Le couplage Interrupteur ↔ Sortie ainsi que leur rôle respectif est donné dans le tableau ci-dessous.

Interrupteur	Sortie Associée	Niveau Logique	Action	Valable pour les jeux
S15	Manette	0	Déplacement par Encodeur Rotatif	Casse-Brique + Pong
		1	Déplacement par Accéléromètre	
S3	Taille	0	Raquette Courte	Casse-Brique + Pong
		1	Raquette Longue	
S2	Speed	0	Balle Lente	Casse-Brique + Pong
		1	Balle Rapide	
S1	Obstacle	0	Pas d'obstacle	Pong
		1	Ajout d'un obstacle mobile	

Table 1 : Fonctionnalité des interrupteurs S15, S3, S2 et S1

4) IP Rotary

Rôle : Ce bloc permet la gestion de l'encodeur rotatif de la carte Nexys 4. Il génère les commandes de déplacement (rotation vers la gauche ou vers la droite). Ces signaux sont utilisés par la suite pour piloter la raquette du jeu Casse-Briques, ainsi que la raquette gauche du jeu Pong.

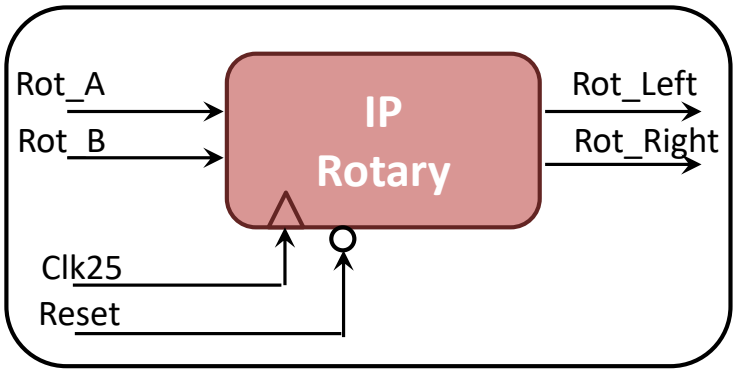


Figure 13 – Vue externe du module IP Rotary

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Reset*** : Reset asynchrone
- ***Rot_A***: 1er Interrupteur de l'encodeur
- ***Rot_B***: 2ème Interrupteur de l'encodeur

Sorties :

- ***Rot_Left***: Commande de rotation à gauche
- ***Rot_Right***: Commande de rotation à droite

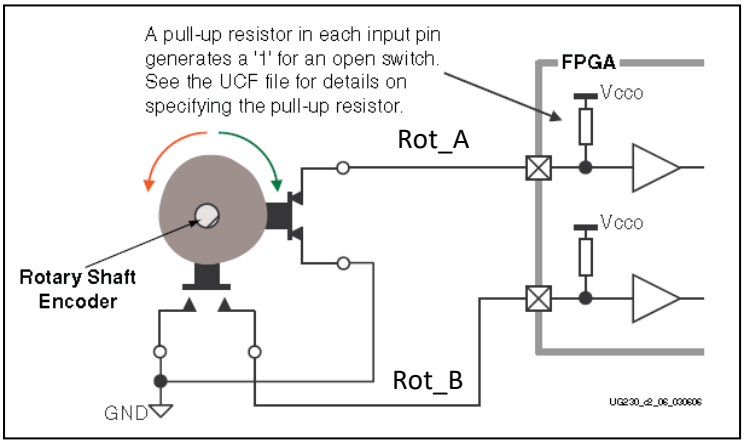


Figure 14 – Fonctionnement de l'encodeur rotatif

Les signaux ***Rot_A*** et ***Rot_B*** sont générés par des interrupteurs situés sur l'encodeur. Leur état dépend des rotations de l'encodeur (voir Figure 14). Selon l'évolution des niveaux logiques de ***Rot_A*** et ***Rot_B*** (Par exemple, est-ce que ***Rot_A*** passe au niveau haut avant ou après ***Rot_B*** ?), il est possible de déterminer le sens de rotation de l'encodeur.

Fonctionnement

- Le module **IP Rotary** est composé de deux sous-blocs
  - **Rotary** gère les signaux issus de l'encodeur rotatif
  - **Move** est une Machine à Etats qui génère les commandes de déplacement ***Rot_Left*** et ***Rot_Right***

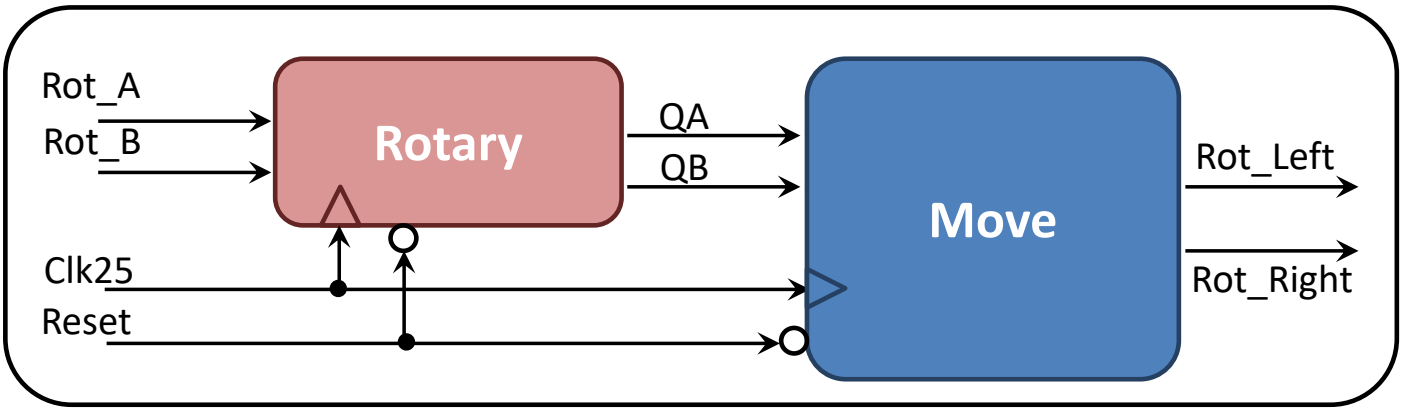


Figure 15 – Architecture interne du module IP Rotary

Fonctionnement de Rotary

- Ce sous-module a deux fonctions
  - Echantillonnage des trois signaux de l'encodeur rotatif
  - Génération des signaux ***QA*** et ***QB***, qui filtrent les signaux ***Rot_A*** et ***Rot_B*** de la façon suivante
    - ***QA*** passe à 1 si les deux signaux ***Rot_A*** et ***Rot_B*** sont à 1
    - ***QA*** passe à 0 si les deux signaux ***Rot_A*** et ***Rot_B*** sont à 0
    - ***QB*** passe à 1 si le signal ***Rot_A*** est à 0 et le signal ***Rot_B*** est à 1
    - ***QB*** passe à 0 si le signal ***Rot_A*** est à 1 et le signal ***Rot_B*** est à 0

Fonctionnement de Move

- Ce sous-module est une machine à états.
  - Elle prend en entrée les signaux ***QA*** et ***QB*** et génère les sorties ***Rot_Left*** et ***Rot_Right***.
- Le graphe d'états est établi de la façon suivante.
  - On prend ***QA*** comme signal de référence. Une commande ***Rot_Left*** ou ***Rot_Right*** est générée uniquement si on a détecté un changement d'état sur ***QA.***
  - Le sens de rotation va alors dépendre de ***QB***
    - Si ***QA*** change d'état AVANT ***QB***, cela correspond à une rotation à gauche du codeur
      - On met ***Rot_Left*** à 1 pendant un cycle d'horloge
    - Si ***QA*** change d'état APRES ***QB***, cela correspond à une rotation à droite du codeur
      - On met ***Rot_Right*** à 1 pendant un cycle d'horloge

5) Gestion Accéléromètre

Rôle : Ce bloc assure la gestion de l'accéléromètre, pour générer les commandes de déplacement de la raquette en fonction de l'inclinaison de la carte par le joueur.

Le capteur est relié au FPGA par l'intermédiaire d'un bus SPI. Le module contient donc un contrôleur de bus SPI afin de pouvoir communiquer avec l'accéléromètre..

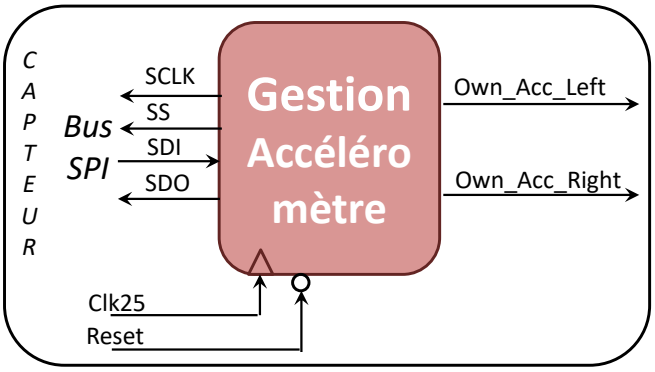


Figure 16 – Vue externe du module Gestion Accéléromètre

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Reset*** : Reset asynchrone

Interface SPI:

- ***SCLK***: Horloge
- ***SS***: Validation du capteur
- ***SDI, SDO***: Serial Data Input/Output

Sorties :

- ***Own_Acc_Left, Own_Acc_Right***: Commandes de déplacement à gauche et à droite.

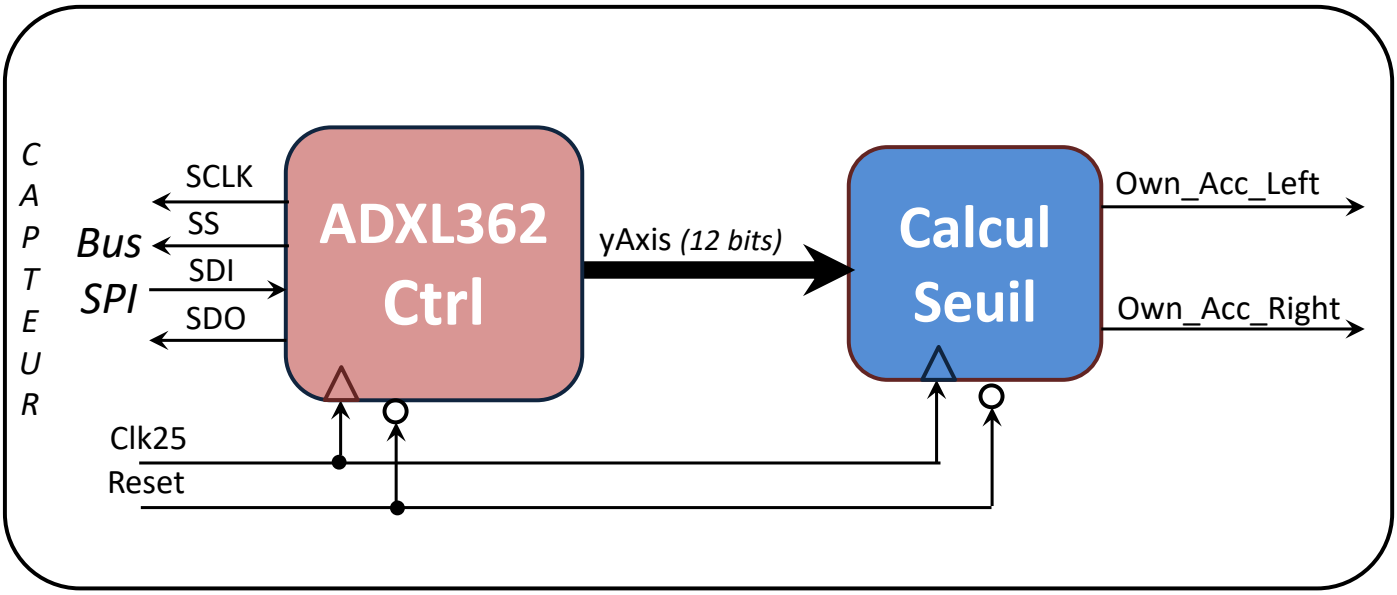


Figure 17 – Architecture interne du module Gestion Accéléromètre

Fonctionnement

- Le module **Gestion Accéléromètre** est composé de deux sous-blocs
  - **ADXL362** gère la communication avec l'accéléromètre via le bus SPI.
    - Il transmet notamment en permanence la valeur de l'accélération sur l'axe Y du capteur. C'est cette composante qui est utilisée pour générer les commande de déplacement.
    - La valeur transmise par le capteur est codée sur **12 bits en complément à 2**.

- **Calcul Seuil** génère les commande de déplacement à gauche et à droite si la valeur sur l'axe Y du capteur dépasse un certain seuil.
  - A la différence des commandes de déplacement issues de l'encodeur rotatif, qui ne restent actives que pendant un cycle d'horloge, les sorties ***Own_ACc_Left*** et ***Own_Acc_Right*** restent actives tant que la valeur de ***yAxis*** dépasse le seuil.

6) IP VGA

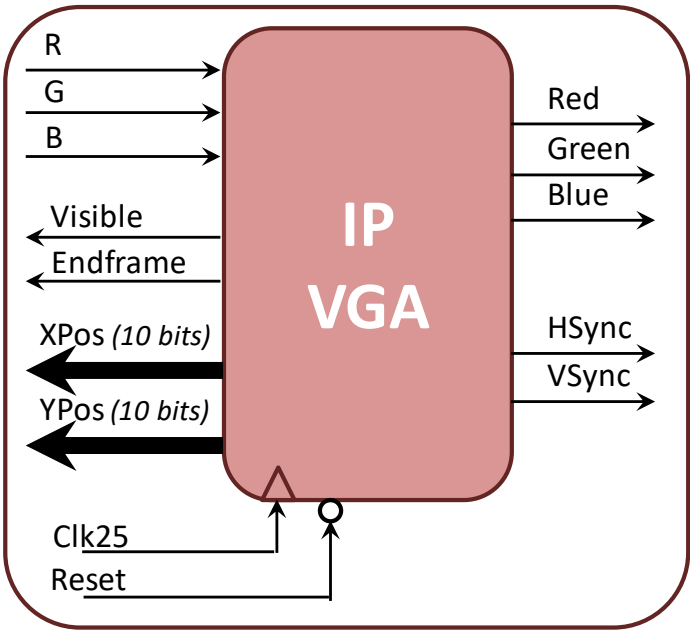


Figure 18 – Vue externe du module IP VGA

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Reset*** : Reset asynchrone
- ***R,G,B***: Niveaux de Rouge (R), Vert (G) et de Bleu (B) souhaités pour le pixel à afficher

Sorties :

- ***Red, Green, Blue***: Couleur du pixel. Signaux transmis à l'écran VGA.
- ***HSync, VSync***: Synchronisation horizontale et verticale. Transmis à l'écran VGA. Signaux actifs à l'état bas.
- ***XPos, YPos***: Coordonnées X,Y du pixel courant
- ***Visible***: Signal indiquant que le pixel courant appartient à la zone visible de l'image
- ***Endframe***: Signal indiquant que le pixel courant est le dernier de l'image visible

Rôle : Ce bloc envoie à l'écran VGA les signaux vidéo permettant d'afficher les images du jeu. Il fournit également aux autres blocs du système des informations telles que les coordonnées du pixel courant, la fin de l'image…

Fonctionnement

- L'image VGA est composée d'une matrice de pixels organisés en 521 lignes et 800 colonnes.
- Une partie de cette matrice (480 lignes et 640 colonnes) constitue la zone visible de l'image. Le reste de l'image correspond à une zone de synchronisation.
- Le module **IP VGA** est organisé autour d'un double compteur qui calcule le numéro de ligne et de colonne du pixel affiché à l'écran. La valeur de ces deux compteurs est disponible en sortie dans ***XPos*** et ***YPos***. Les tableaux ci-dessous donnent la correspondance entre les numéros de ligne/colonne et la zone de l'image.
  - Le pixel de coordonnées (0,0) est situé en haut à gauche de l'écran.

Numéro de Colonne	Zone de l'Image	Valeur de HSync
0 → 639	Visible	1
640 →664	Pré-Synchronisation	1
665 → 759	Synchronisation	0
760 → 799	Post-Synchronisation	1

Tab.2 : Zone de l'image en fonction des colonnes

Numéro de Ligne	Zone de l'Image	Valeur de VSync
0 → 479	Visible	1
480 →489	Pré-Synchronisation	1
490 → 491	Synchronisation	0
492 → 520	Post-Synchronisation	1

Tab.3 : Zone de l'image en fonction des lignes

- Le signal ***Visible*** passe à 1 lorsque le pixel courant appartient à la zone visible de l'image
- Le signal ***Endframe*** est activé lorsque l'on affiche le dernier pixel de la zone visible de l'image
- La couleur de chaque pixel (sorties ***Red***, ***Green*** et ***Blue***) est fixée directement à partir des entrées ***R***, ***G*** et ***B*** (voir Table 4).

Red	Green	Blue	Couleur Affichée
0	0	0	Noir
0	0	1	Bleu
0	1	0	Vert
0	1	1	Cyan
1	0	0	Rouge
1	0	1	Magenta
1	1	0	Jaune
1	1	1	Blanc

Tab.4 : Code couleur de l'IP VGA

A noter

La sortie VGA proposée sur la carte Nexys 4 offre 4 bits par composante (rouge, verte ou bleue), permettant d'avoir 16 teintes pour chaque couleur, soit un total de 4096 couleurs affichables.

Les sorties ***Red***, ***Green*** et ***Blue*** sont connectées sur le MSB de chaque composante.

7) Gestion Afficheurs

Rôle : Ce module a pour objectif de piloter les 8 afficheurs 7 semgnets présents sur la carte Nexys 4 afin d'y inscrire des messages d'information pour le joueur.

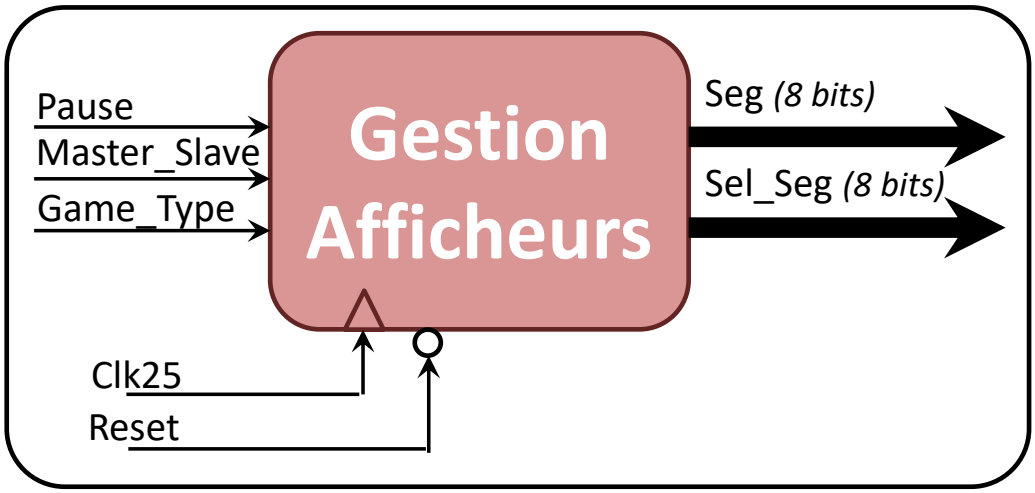


Figure 19 – Vue externe du module Gestion Afficheurs

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Reset*** : Reset asynchrone

- ***Pause*** : Indicateur de mise en pause du jeu
- ***Master_Slave***: Indique si la SU-EE100 est en mode Console ou en mode Manette
- ***Game_Type***: Type de Jeu actif (Casse Briques ou Pong)

Sorties :

- ***Sel_Seg***: Bus de sélection des afficheurs pour y modifier la valeur des segments selon la valeur du bus ***Seg***.
- ***Seg***: Bus d'état des segments des afficheurs.

Fonctionnement des afficheurs 7 segments

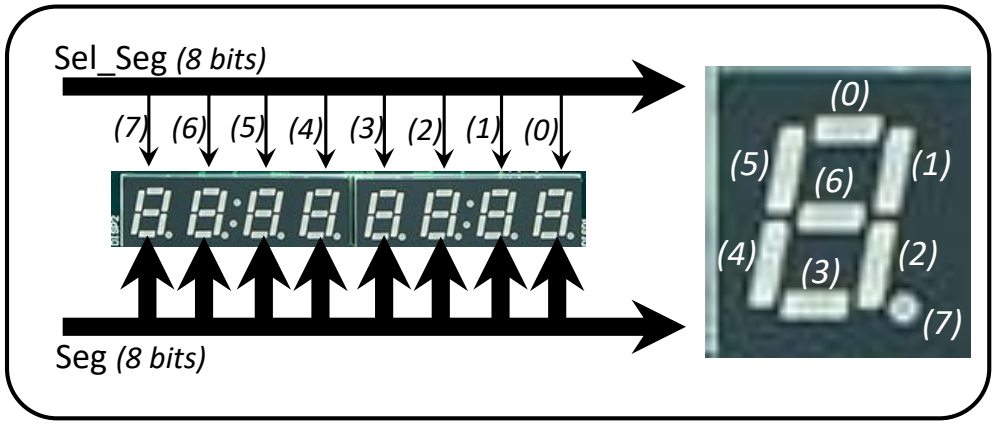


Figure 20 – Fonctionnement des Afficheurs et Numérotation des Segments

- Les 8 afficheurs ont un bus commun ***Seg*** pour déterminer l'état des 7 segments et du point.
  - ***Seg(i)*** = 0 → Segment i allumé.
  - ***Seg(i)*** = 1 → Segment i éteint.

- Le bus ***Sel_Seg*** permet de modifier individuellement l'état d'un afficheur
  - ***Sel_Seg(i)*** = 0 → Les segments de l'afficheur i prennent la valeur fournie sur le bus ***Seg***.
  - ***Sel_Seg(i)*** = 1 → Tous les segments de l'afficheur i sont éteints.

➢ Si l'on veut afficher des caractères différents sur chaque afficheur, il est donc nécessaire de rafraichir en permanence le contenu de chaque afficheur puisque d'une part le bus de données des afficheurs est commun, et que d'autre part la désélection d'un afficheur éteint tous ses segments.
➢ La fréquence de rafraichissement doit être compatible avec la persistance rétinienne de l'œil humain.

Fonctionnement du module Gestion Afficheurs

- Le module est structuré autour d'un compteur qui tous les 10000 cycles d'horloge sélectionne un des afficheurs pour mettre à jour l'état de ses segments.

- En fonction de l'état des commandes ***Pause***, ***Master_Slave*** ou ***Game_Type***, on affiche alors un message différent.

  - Si ***Master_Slave*** est actif (SU-EE100 en mode manette), on affiche le message "**MANETTE**"
  - Sinon, si ***Pause*** est actif, on affiche le message "**PAUSE**"
  - Sinon, on affiche le message "**PONG**" ou "**CASSEBRI**" selon la valeur de ***Game_Type***.

8) IP Objects

Rôle : Ce module a pour objectif d'indiquer si le pixel courant de l'image appartient à l'un des objets affichés dans le jeu Casse Briques ou Pong (Raquettes, balle, murs, briques…)

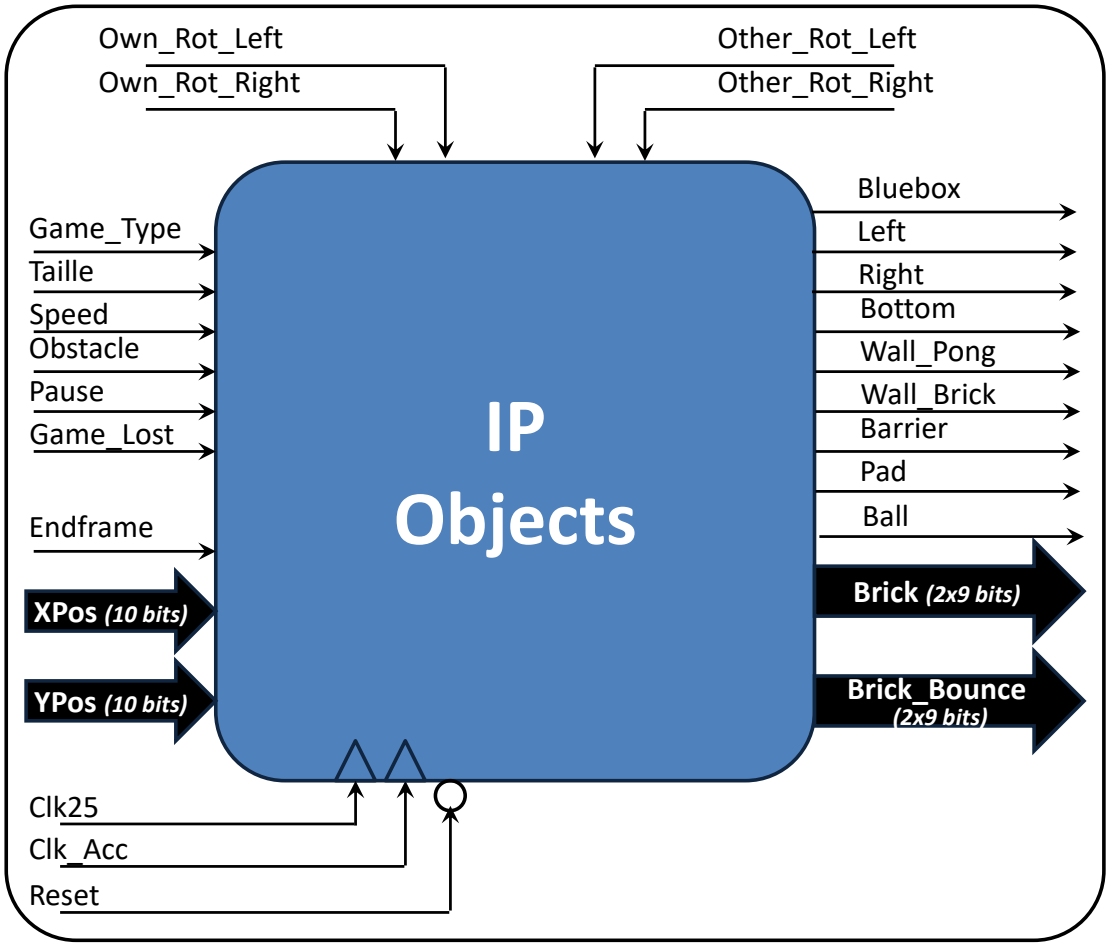


Figure 21 – Vue externe du module IP Objects

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Clk_Acc*** : Horloge de 25 Hz
- ***Reset*** : Reset asynchrone
- ***Signaux de l'IP VGA***
  - ***XPos, Ypos***: Coordonnées en X et en Y du pixel courant
  - ***Endframe***: Signal de fin de la zone visible de l'image
- ***Commandes de déplacement des encodeurs rotatifs***
  - ***Own_Left, Own_Right***: Commandes de déplacement (par encodeur rotatif ou accéléromètre)
  - ***Other_Left, Other_Right***: Commandes de déplacement venant d'une autre SU-EE100 en mode Manette.
- ***Paramètres de Jeu***
  - ***Manette***: Indique si les commandes de déplacement viennent de l'accéléromètre ou l'encodeur rotatif.
  - ***Game_Type***: Type de Jeu actif (Casse Briques ou Pong)
  - ***Taille***: Taille des raquettes du jeu (Raquettes courtes ou longues)
  - ***Speed***: Vitesse de la balle (Lente ou rapide)
  - ***Obstacle***: Présence d'un obstacle mobile au milieu de l'écran (Jeu Pong uniquement)
  - ***Pause***: Jeu en mode pause
  - ***Game_Lost***: Signal indiquant que la partie est perdue.

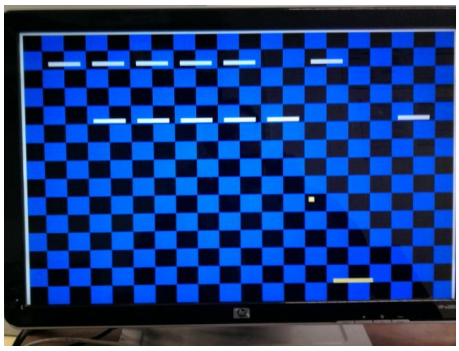
Sorties :

- ***Bluebox***: Indique que le pixel appartient à une case bleue du décor
- ***Left, Right, Bottom***: Indique que le pixel se trouve sur le bord gauche, droit ou bas de l'écran
- ***Wall_Pong, Wall_Brick***: Indique que le pixel appartient à l'un des murs du jeu Pong ou Casse Briques
- ***Barrier***: Indique que le pixel appartient à l'obstacle mobile du jeu Pong
- ***Pad***: Indique que le pixel appartient à une raquette
- ***Brick***: Indique que le pixel appartient à une des briques du jeu Casse Briques
- ***Brick_Bounce***: Indique les briques qui ont été percutées par la balle (jeu Casse Briques)
- ***Ball***: Indique que le pixel appartient à la balle

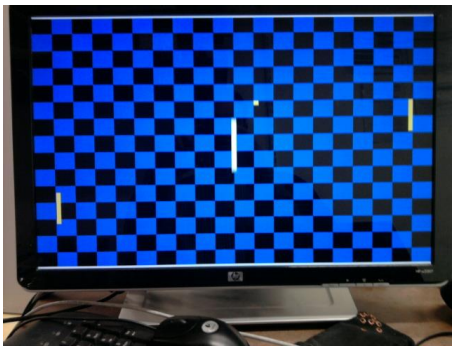
Fonctionnement général

- Le décor de fond est un damier bleu et noir, où chaque case a une dimension de 32x32 pixels.
- Chaque jeu dispose ensuite d'objets spécifiques

- Les objets du jeu Casse Briques sont les suivants
    - Trois murs blancs, situés en haut, à gauche et à droite de l'écran
    - Une raquette jaune, située en bas de l'écran et pouvant se déplacer horizontalement
    - Une balle jaune
    - Deux rangées de 9 briques blanches



- Les objets du jeu Pong sont les suivants
    - Deux murs blancs, situés en haut et en bas de l'écran
    - Deux raquettes jaunes, située à gauche et à droite de l'écran et pouvant se déplacer verticalement.
    - Une balle jaune
    - Un obstacle blanc situé au milieu de l'écran et se déplaçant verticalement (si l'option ***Obstacle*** est activée)



- La gestion de tous ces objets par le module **IP Objects** est réalisée grâce à 5 sous-modules.

1) **Decor :**
    - Gère le damier de fond d'écran (cases bleues)
    - Gère les rebords d'écran
    - Gère les murs
    - Gère l'obstacle mobile du jeu Pong
2) **Pad_Ctrl**
    - Gère les raquettes
3) **Brick_Ctrl**
    - Gère les briques du jeu Casse Briques
4) **Ball_Ctrl**
    - Gère la balle
5) **Bounce_Ctrl**
    - Gère les rebonds de la balle contre tous les autres objets

8.1) Module Decor

- Le sous-module **Décor** va activer ses sorties si le pixel dont les coordonnées sont données par ***XPos*** et ***YPos*** appartient à l'un des objets gérés par le sous-module.

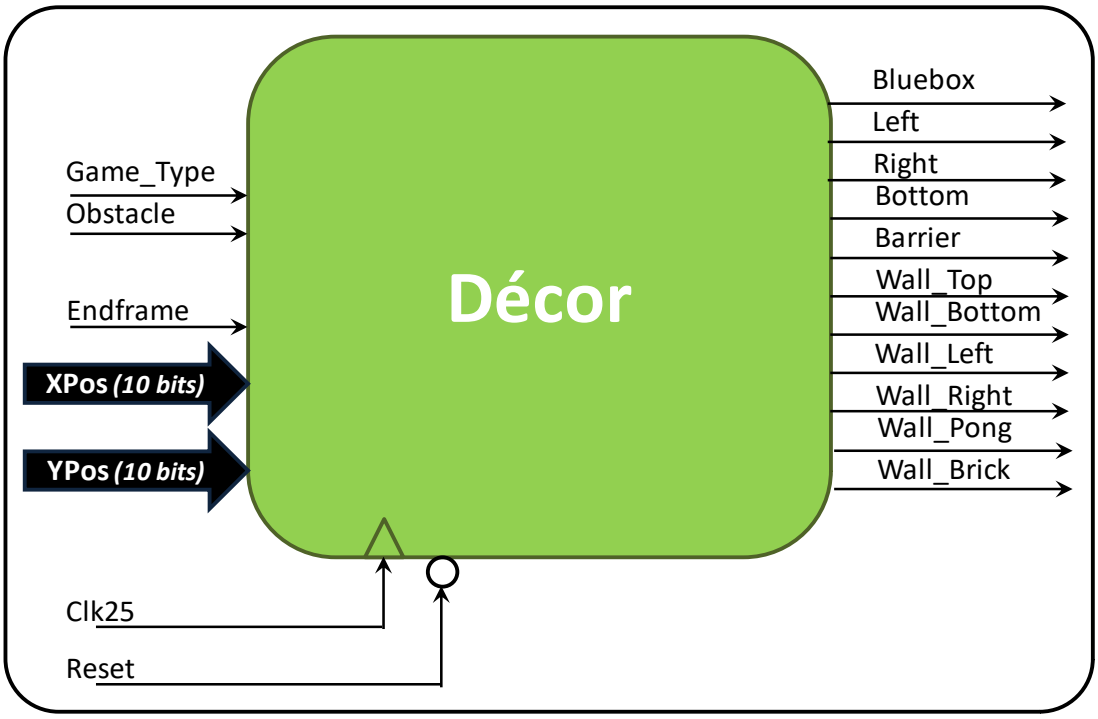


Figure 22 – Vue externe du sous-module Décor

- Les entrées/sorties de ce module sont en partie celles du module **IP Objects**. On trouve en plus 4 sorties spécifiques (***Wall_Top, Wall_Bottom, Wall_Left***, ***Wall_Right***).

- Le comportement des sorties est le suivant :
  - ***Bluebox*** est activée si l'on est dans une case de couleur bleue. Elle est désactivée si l'on est dans une case de couleur noire.
  - ***Left*** est activée si le jeu actif est Pong et que le pixel est sur le bord gauche de l'écran
  - ***Right*** est activée si le jeu actif est Pong et que le pixel est sur le bord droit de l'écran
  - ***Bottom*** est activée si le jeu actif est Casse Briques et que le pixel est sur la dernière ligne de l'écran
  - ***Wall_Top*** est activée si le pixel appartient au mur du haut.
  - ***Wall_Bottom*** est activée si le jeu actif est Pong et que le pixel appartient au mur du bas
  - ***Wall_Left*** est activée si le jeu actif est Casse Briques et que le pixel appartient au mur de gauche
  - ***Wall_Right*** est activée si le jeu actif est Casse Briques et que le pixel appartient au mur de droite
  - ***Wall_Pong*** est activée si le jeu actif est Pong et que le pixel appartient à l'un des murs de ce jeu.
  - ***Wall_Brick*** est activée si le jeu est Casse Briques et e le pixel appartient à l'un des murs de ce jeu
  - ***Barrier*** est activée si :
    - Le jeu actif est Pong
    - La commande ***Obstacle*** est activée
    - Le pixel appartient à l'obstacle mobile.
  - Les sorties sont automatiquement désactivées si elles correspondent à des objets qui n'appartiennent pas au jeu actif (Par exemple, ***Wall_Right*** est désactivé si le jeu est Pong).
  - L'obstacle se déplace verticalement de deux pixels à chaque image. Lorsqu'il atteint l'un des murs du haut ou du bas, son sens de déplacement est inversé.

8.2) Module Pad Ctrl

- Ce sous-module indique si le pixel courant de l'image appartient à l'une des raquettes des jeux de la console.

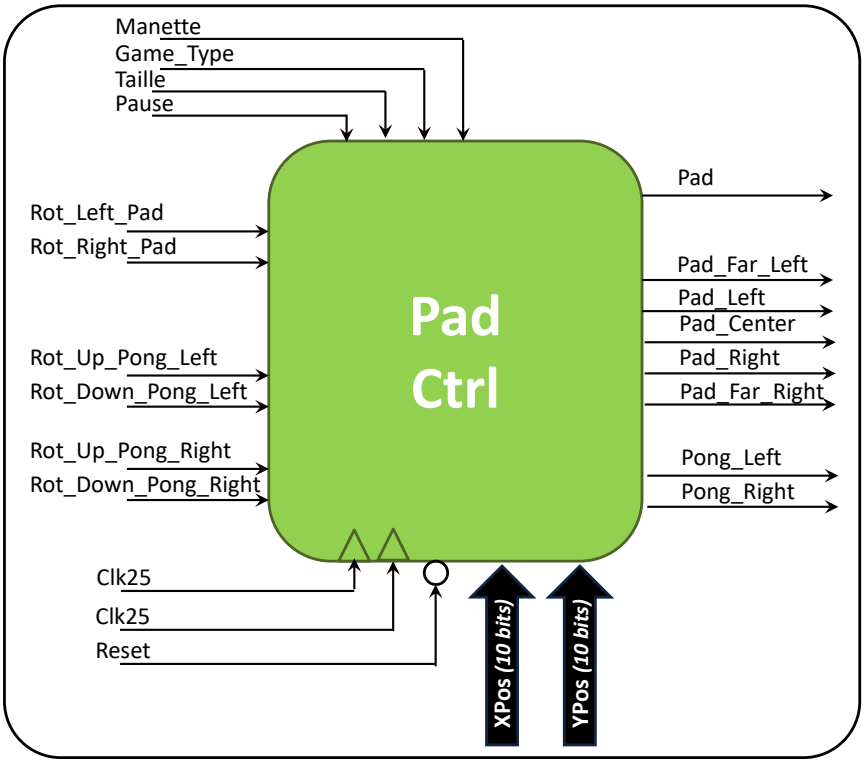


Figure 23 – Vue externe du sous-module Pad_Ctrl

- Les entrées/sorties de ce module sont en partie celles du module **IP Objects**.
- Les entrées ***Rot_Left_Pad*** et ***Rot_Right_Pad*** permettent de piloter la raquette du Jeu Casse Briques
  - Elles sont connectées aux signaux ***Own_Rot_Left*** et ***Own_Rot_Right*** du module **IP Objects**
  - Ces signaux ne sont pas pris en compte si le jeu actif est Pong
- Les entrées ***Rot_Up_Pong_Left*** et ***Rot_Down_Pong_Left*** commandent la raquette gauche du Jeu Pong
  - Elles sont connectées aux signaux ***Own_Rot_Left*** et ***Own_Rot_Right*** du module **IP Objects**
  - Ces signaux ne sont pas pris en compte si le jeu actif est Casse Briques
- Les entrées ***Rot_Up_Pong_Right*** et ***Rot_Down_Pong_Right*** commandent la raquette droite du Jeu Pong
  - Elles sont connectées aux signaux ***Other_Rot_Left*** et ***Other_Rot_Right*** du module **IP Objects**
  - Ces signaux ne sont pas pris en compte si le jeu actif est Casse Briques

- Le module **Pad_Ctrl** calcule en interne les coordonnées en X et en Y du premier pixel (en haut et à gauche de l'objet) des raquettes des différents jeux.
  - A partir de ce premier pixel, on délimite une zone rectangulaire.
  - La surface de la zone dépend du paramètre ***Taille***.
  - Si le pixel appartient à cette zone, on active l'une des sorties (***Pad_XXX*** ou ***Pong_XXX***).
- Le module gère également le déplacement des raquettes.
  - Si on reçoit en entrée une commande de déplacement, on modifie la coordonnée correspondante de la raquette concernée.
  - Le déplacement des raquettes est gelé si le paramètre ***Pause*** est activé.

- La sortie ***Pad*** indique que le pixel courant appartient à une raquette (peu importe le jeu).
- Les sorties ***Pong_Left*** et ***Pong_Right*** indiquent que le pixel courant appartient à l'une des raquettes de Pong.
- La raquette du jeu Casse Briques est divisée en 5 zones :
    - Extrême gauche, gauche, centre, droite, extrême droite.
    - Les sorties ***Pad_Far_Left***, ***Pad_Left***, ***Pad_Center***, ***Pad_Right*** et ***Pad_Far_Right*** indiquent que le pixel appartient à une des 5 zones.
    - Ces sorties ne sont pas actives si l'on joue à Pong

- Selon que le joueur utilise l'accéléromètre ou l'encodeur rotatif pour déplacer sa raquette, une horloge différente est utilisée pour mettre à jour les coordonnées du pad.
    - Si le joueur utilise l'encodeur rotatif, on utilise l'horloge 25 MHz.
    - Si le joueur utilise l'accéléromètre, on utilise l'horloge 25 Hz afin que le nombre de requêtes de déplacement soit équivalent à celui de l'encodeur rotatif.
    - A noter que pour la raquette contrôlée depuis une autre carte SU-EE100, on considère que seul l'encodeur rotatif est utilisé, l'horloge contrôlant les déplacements de cette raquette est donc celle à 25 MHz.

8.3) Module Brick_Ctrl

- Ce sous-module indique si le pixel courant de l'image appartient à l'une des briques du jeu Casse Briques.

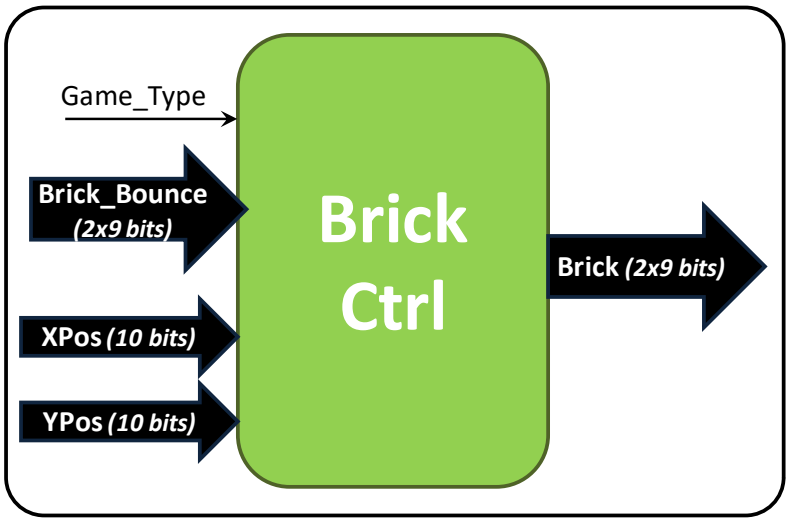


Figure 24 – Vue externe du sous-module Brick_Ctrl

- Les briques ont une position prédéfinie dans l'image. Elles sont réparties en 2 rangées de 9 briques.
    - La sortie ***Brick*** est donc organisée sous la forme d'un tableau de 2x9 bits.
    - Le bit ***Brick(i)(j)*** indique si le pixel courant appartient à la brique (i)(j).

- La sortie ***Brick(i)(j)*** est activée si :
    - Le jeu actif est Casse Briques,
    - Le pixel courant appartient à la zone de la brique (i)(j)
    - La balle n'a pas déjà rebondi contre la brique (l'entrée ***Brick_Bounce(i)(j)*** n'est pas activée)

8.4) Module Ball Ctrl

- Ce sous-module calcule les coordonnées de la balle et gère son déplacement en fonction de :
  - La vitesse de la balle fixée par le paramètre ***Speed***
  - Les rebonds contre les différents objets du jeu représentés par les entrées ***XXX_Bounce***.

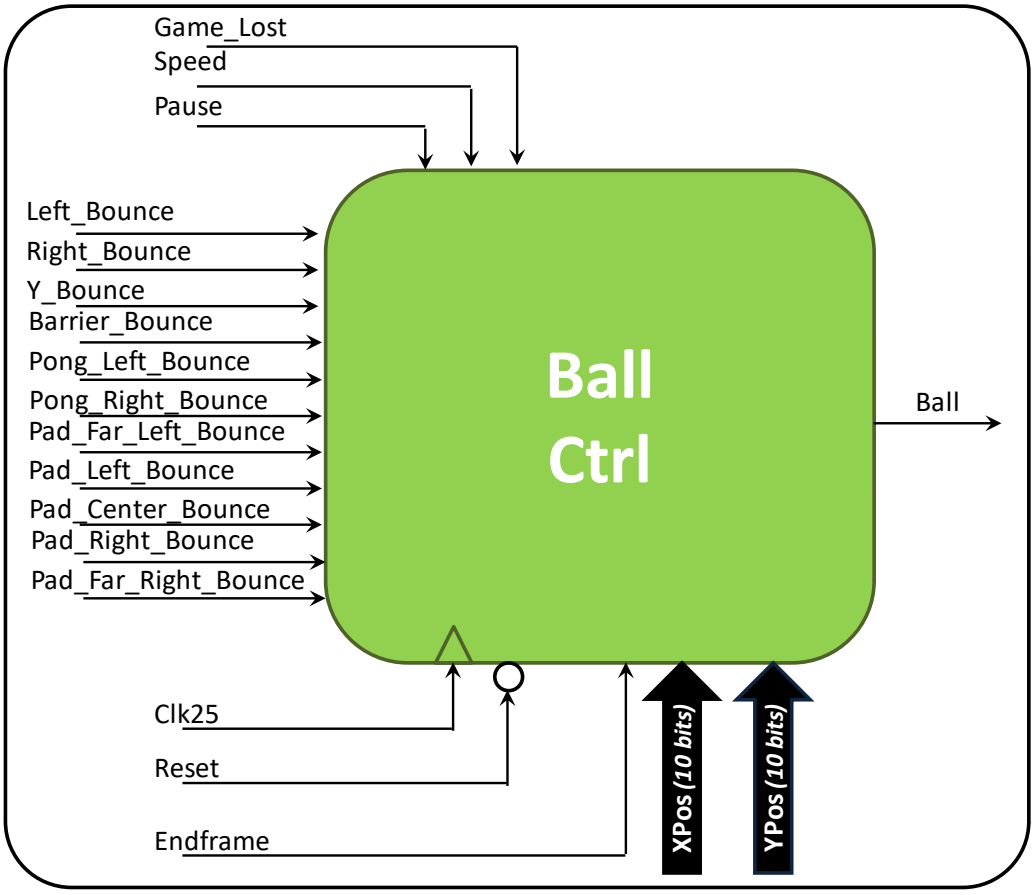


Figure 25 – Vue externe du sous-module Ball_Ctrl

- On calcule en interne les coordonnées en X et en Y du premier pixel (en haut et à gauche) de la balle.
  - A partir de ce 1er pixel, on détermine une zone carrée de 8x8 pixels
  - Si le pixel courant appartient à cette zone, on active la sortie ***Ball***.

- Les coordonnées de la balle sont mises à jour à chaque fin d'image (***Endframe***), sauf si le mode ***Pause*** est activé.
  - La mise à jour se fait en fonction de la trajectoire de la balle
  - Il y a 12 trajectoires possibles
  - La balle garde la même trajectoire tant qu'elle ne rebondit pas contre un autre objet.
  - En cas de rebond, on change la trajectoire de la balle
  - Si la partie est perdue (***Game_Lost***), alors on replace la balle à une position prédéterminée.
  - Pour plus d'informations sur les trajectoires de balle, voir le code VHDL…

8.5) Module Bounce_Ctrl

- Ce sous-module indique si la balle rebondit contre l'un des objets des jeux.

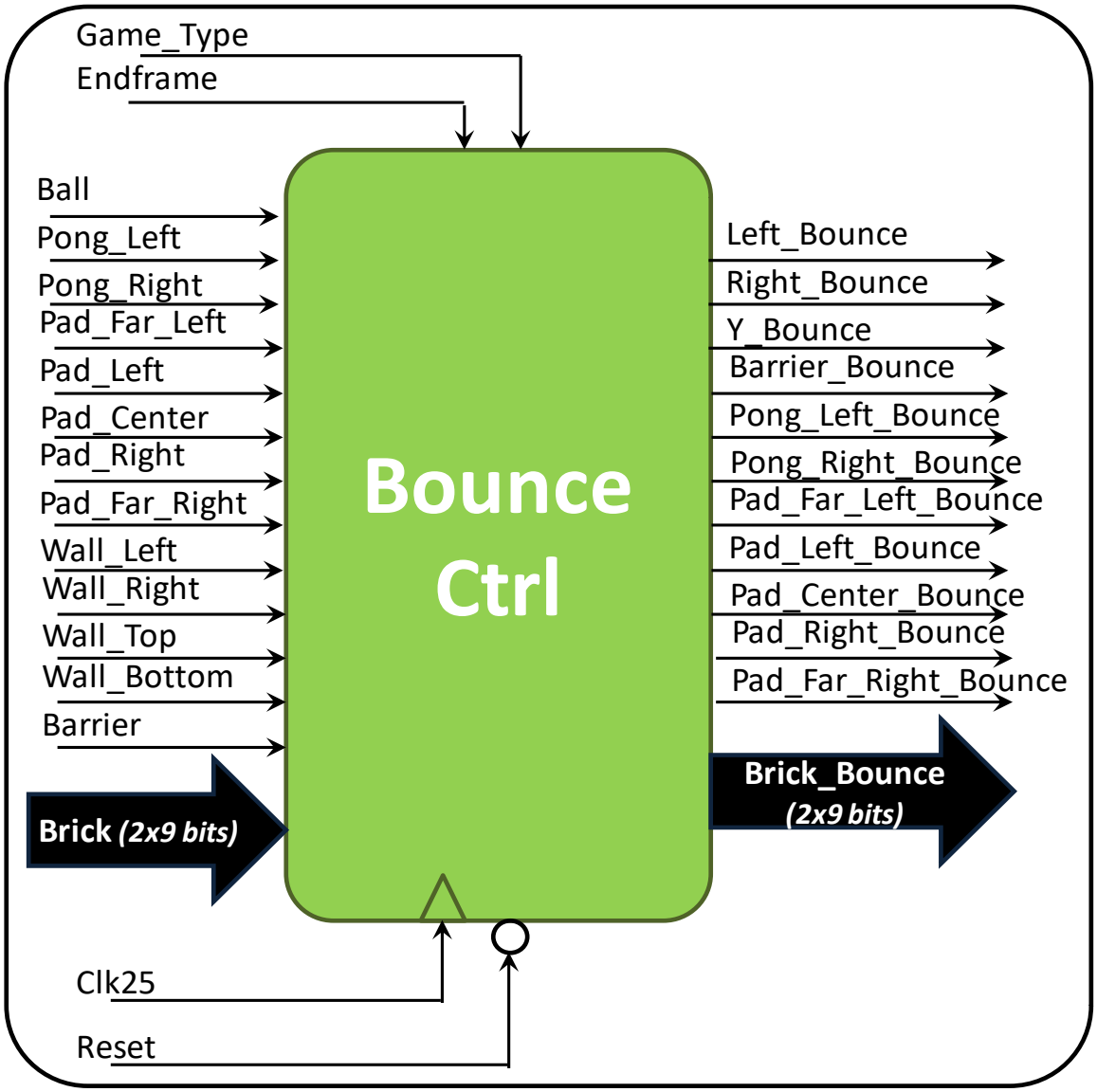


Figure 26 – Vue externe du sous-module Bounce_Ctrl

- Pendant que l'on balaie l'image (***Endframe*** n'est pas actif)
  - Si le pixel courant appartient en même temps à la balle et un objet XXX, on active la sortie ***XXX_Bounce***.
  - La sortie ***Brick_Bounce*** ne peut être activée que si le jeu actif est Casse Briques.

- Quand on arrive à la fin de l'image (***Endframe***)
  - Les sorties ***XXX_Bounce*** sont remises à zéro

9) IP Game

Rôle : Ce bloc contrôle le fonctionnement de la console ce qui consiste principalement à :

- Sélectionner le mode Console ou Manette pour la SU-EE100
- Sélectionner le jeu actif (Pong ou Casse Briques)
- Activer ou désactiver le mode Pause
- Indiquer si la partie est gagnée ou perdue

Il transmet également à l'IP VGA la couleur des objets que l'on souhaite afficher

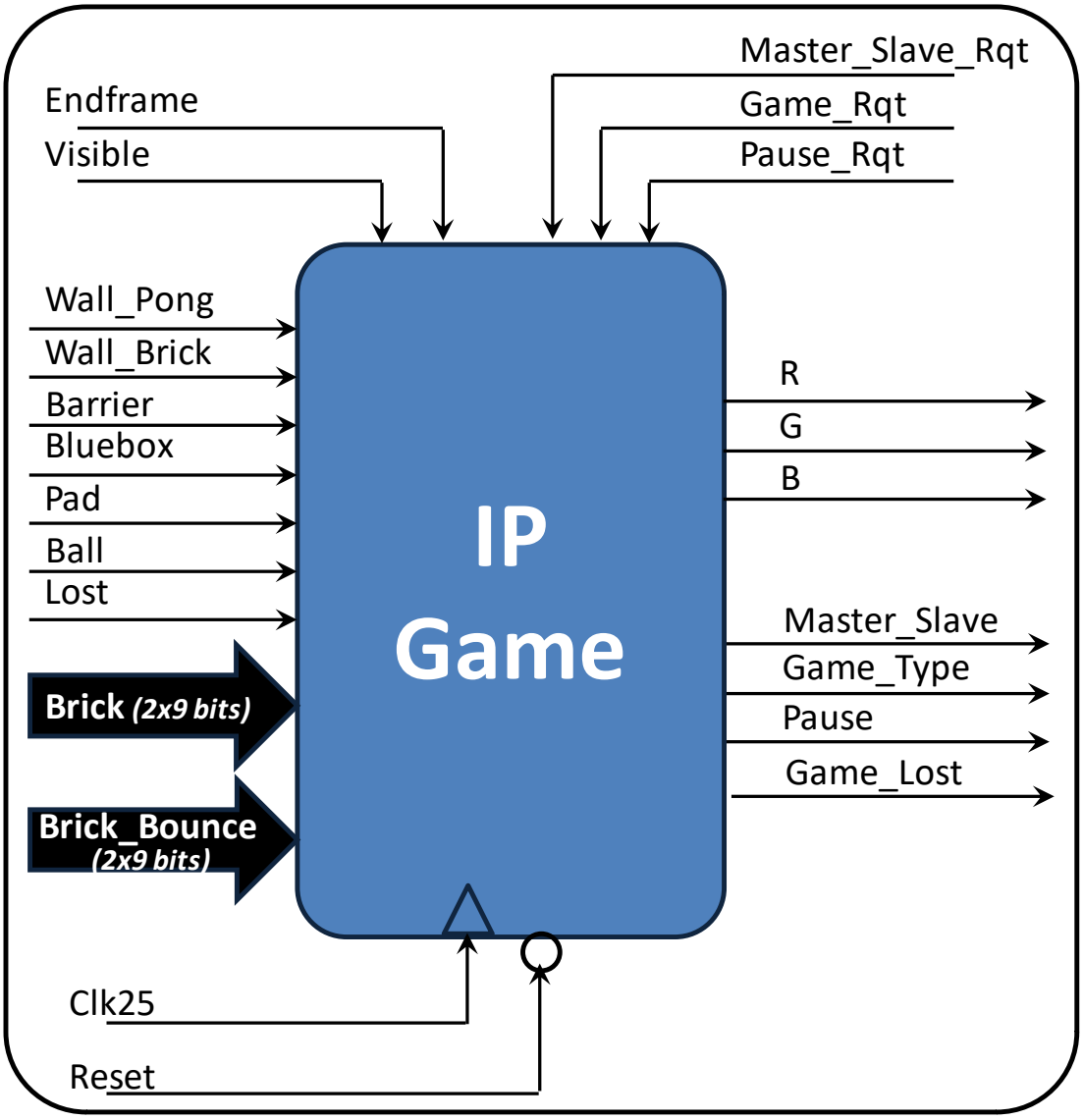


Figure 27 – Vue externe du module IP Game

Entrées :

- ***Clk25*** : Horloge 25 MHz
- ***Reset*** : Reset asynchrone
- ***Signaux de l'IP VGA***
  - ***Endframe***: Signal de fin de la zone visible de l'image
  - ***Visible***: Indique que le pixel courant appartient à la zone visible de l'image

- ***Paramètres de jeu de la SU-EE100***
  - ***Master_Slave_Rqt***: Demande de changement de mode de la SU-EE100 (Console / Manette)
  - ***Game_Rqt***: Demande de changement de jeu (Casse Briques / Pong)
  - ***Pause_Rqt***: Demande de mise en pause du jeu ou de sortie de pause.
- ***Objets des jeux***
  - ***Wall_Pong, Wall_Brick***: Indique que le pixel appartient à l'un des murs du jeu Pong ou Casse Briques
  - ***Barrier***: Indique que le pixel appartient à l'obstacle mobile du jeu Pong
  - ***Bluebox***: Indique que le pixel appartient à une case bleue du décor
  - ***Pad***: Indique que le pixel appartient à une raquette
  - ***Ball***: Indique que le pixel appartient à la balle
  - ***Lost***: Indique que le pixel se trouve sur un des bords de l'écran
  - ***Brick***: Indique que le pixel appartient à une des briques du jeu Casse Briques
  - ***Brick_Bounce***: Indique les briques qui ont été percutées par la balle (jeu Casse Briques)

Sorties :

- ***R,G,B***: Niveaux de Rouge (R), Vert (G) et de Bleu (B) souhaités pour le pixel à afficher
- ***Master_Slave***: Indique si la SU-EE100 est en mode Console ou en mode Manette
- ***Game_Type***: Indique si le jeu actif est Casse Briques ou Pong.
- ***Pause***: Indique si le jeu est en mode pause
- ***Game_Lost***: Signal indiquant que la partie est perdue (quand la balle est sortie de l'écran).

Fonctionnement général

- La couleur par défaut transmise à l'IP VGA est le noir.
- Si le pixel courant appartient à l'un des objets du jeu, on modifie ***R***, ***G*** et ***B*** pour indiquer la couleur souhaitée

- ***Game_Lost***
  - La sortie est activée quand la partie est perdue, c'est-à-dire quand la balle est sortie de l'écran
- ***Master_Slave***
  - En mode Console, la sortie est mise à 1.
  - En mode Manette, la sortie est mise à 0.
  - On inverse le niveau logique si on détecte une requête sur ***Master_Slave_Rqt***
- ***Game_Type***
  - Si le jeu actif est Pong, la sortie est mise à 1.
  - Sinon (Casse Briques), la sortie est mise à 0
  - On inverse le niveau logique si on détecte une requête sur ***Game_Rqt***
- ***Pause***
  - En mode Pause, la sortie est mise à 1
  - Sinon, la sortie est au niveau bas
  - On inverse le niveau logique si on détecte une requête sur ***Pause_Rqt***

- La gestion de toutes ces tâches est réalisée grâce à 4 sous-modules.

1) **Display:**
   - Sélection de la couleur à afficher
2) **Mode**
   - Gestion de l'état de la partie (En cours, gagnée, perdue…)
   - Gestion du mode Pause
3) **Master Slave Manager**
   - Gestion du mode de la SU-EE100 (Console / Manette)
4) **Game Manager**
   - Sélection du jeu actif (Casse Briques / Pong)

9.1) Module Display

- Ce sous-module détermine la couleur du pixel à afficher. Cette information est ensuite transmise à **IP VGA**

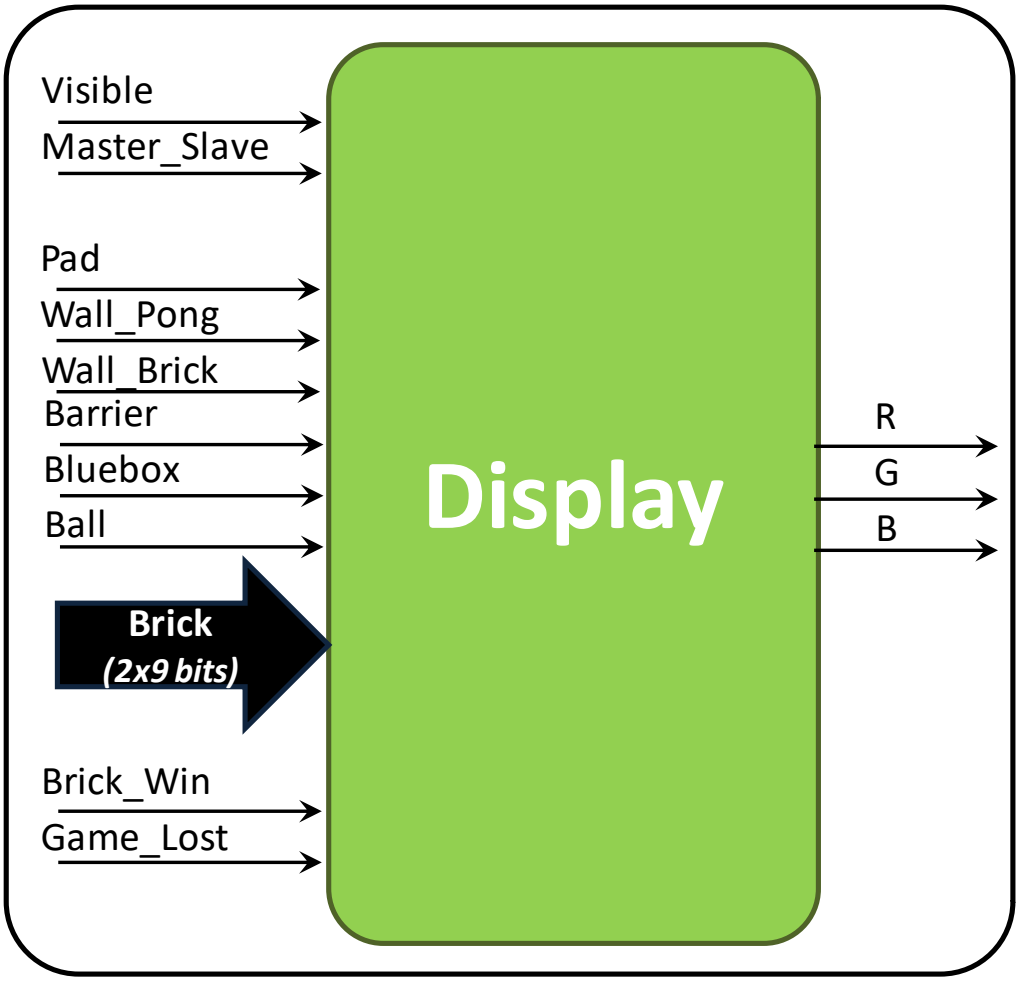


Figure 28 – Vue externe du sous-module Display

- Si l'on n'est pas dans la zone visible de l'image, ou que la SU-EE100 est en mode Manette (***Master_Slave*** = 0), alors la couleur affichée est le noir.
- Sinon :
  - Si le pixel courant appartient à une raquette ou à la balle, la couleur à afficher est le jaune
  - Si le pixel courant appartient à une brique, un mur ou à l'obstacle, la couleur à afficher est le blanc
  - Si le pixel courant appartient à une case bleue, la couleur à afficher est le bleu.
  - Si la partie est perdue (***Game_Lost***), on affiche du rouge
  - Si la partie de Casse Briques est gagnée (***Brick_Win***), on affiche du vert.

9.2) Module Mode

- Ce sous-module détermine l'état de la partie en cours et gère le mode Pause de la console.

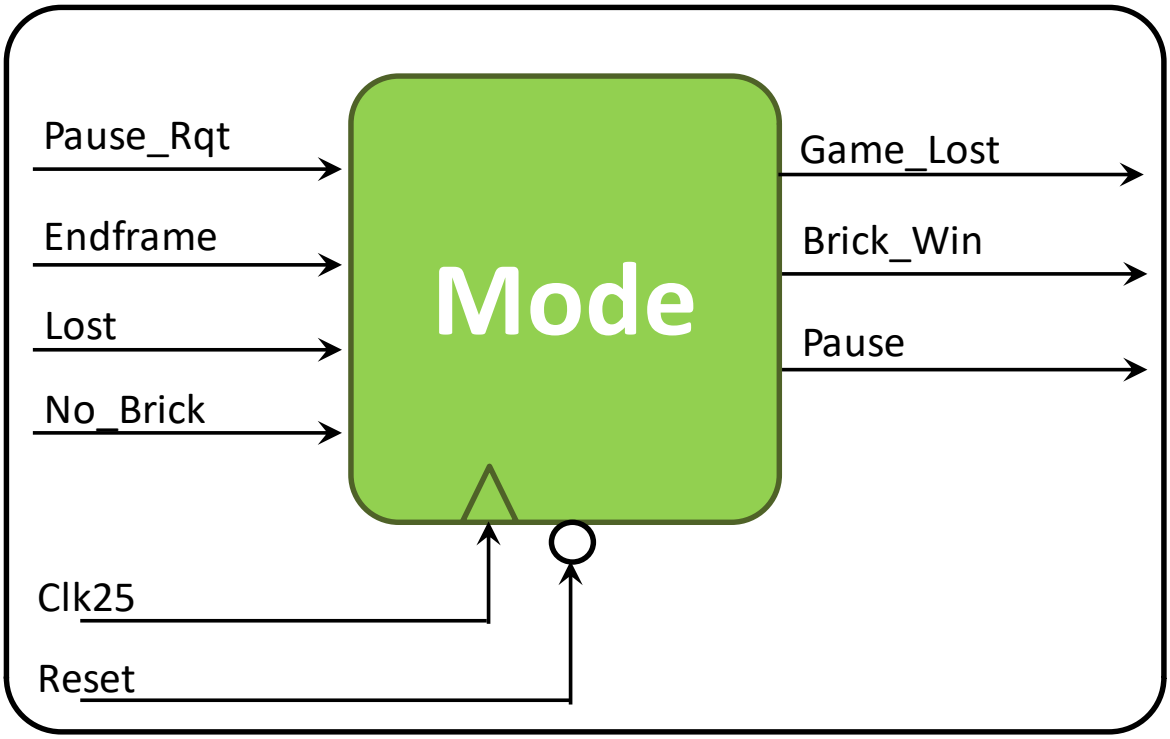


Figure 29 – Vue externe du sous-module Mode

- Ses entrées/sorties reprennent celles du module IP Game. Une sortie supplémentaire, ***Brick_Win*** permet d'indiquer qu'une partie de Casse Briques est gagnée.
- Ce sous-module est lui-même composé de 3 blocs

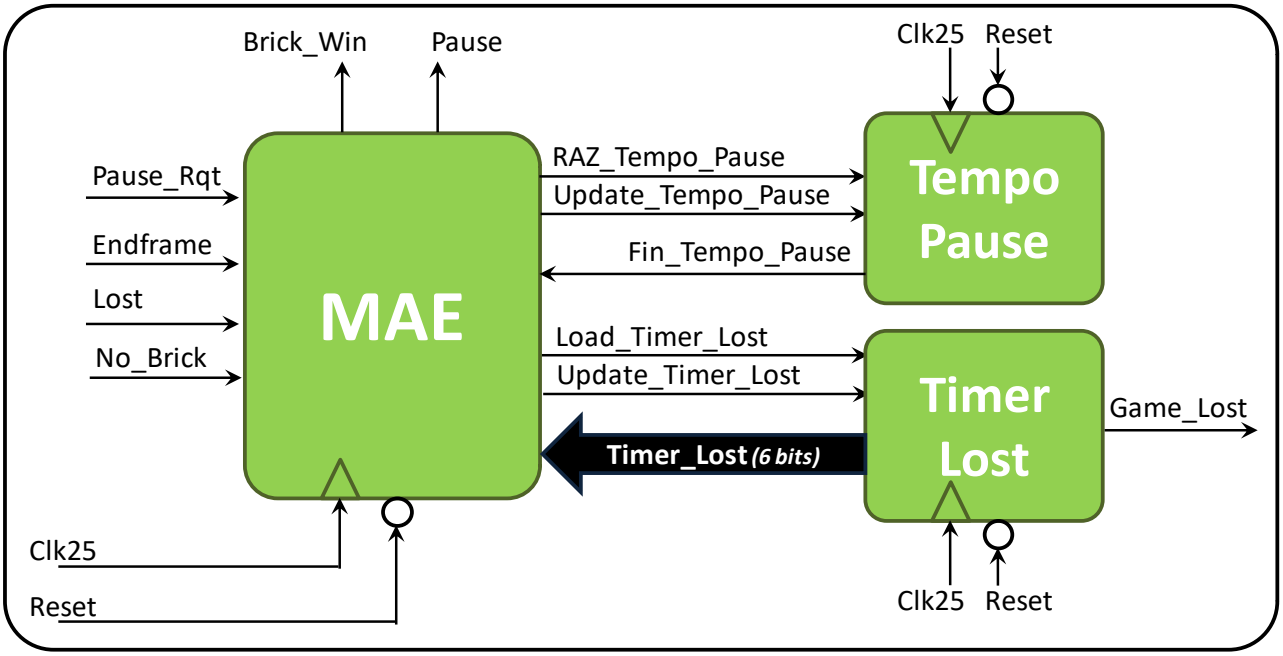


Figure 30 – Vue interne du sous-module Mode

1) **Tempo Pause:**
   - Compteur permettant de gérer l'anti-rebond du bouton central (qui fait les demandes de pause).
   - Utilisé pour la gestion du mode Pause de la SU-EE100
2) **Timer Lost**
   - Compteur permettant de générer la sortie ***Game_Lost***
3) **MAE**
   - Machine à états pour gérer
     - Les sorties ***Brick_Win*** et ***Pause***
     - Les deux compteurs **Tempo Pause** et **Timer Lost**

- Le signal ***No_Brick*** en entrée de la MAE est la sortie d'une porte ET prenant en entrée tous les bits de ***Brick_Bounce***.
  - ***No_Brick*** est activé si la balle a rebondi contre toutes les briques (c'est-à-dire que toutes les briques ont été détruites)

9.2.1) Tempo Pause

- **Tempo Pause** est un compteur 10 bits possédant les fonctionnalités suivantes
  - RAZ synchrone (si ***RAZ_Tempo_Pause*** est activée)
  - Incrémentation (si ***Update_Tempo_Pause*** est activée)
  - Maintien de la valeur précédente (si aucune commande n'est activée)

  - La sortie du compteur est comparée à la valeur maximale (tous les bits à 1). Si tel est le cas, la sortie ***Fin_Tempo_Pause*** est activée.

9.2.2) Timer Lost

- **Timer Lost** est un compteur 6 bits possédant les fonctionnalités suivantes
  - Chargement parallèle à la valeur 63 (tous les bits à 1) (si ***Load_Timer_Lost*** est activée)
  - Décrémentation (si ***Update_Timer_Lost*** est activée)
  - Maintien de la valeur précédente (si aucune commande n'est activée)

  - Si la sortie du compteur est supérieure à 0, la sortie ***Game_Lost*** est activée.

9.2.3) MAE

- La **MAE** a pour but de gérer :
  - Les sorties ***Brick_Win*** et ***Pause***
  - Les deux compteurs **Tempo Pause** et **Timer Lost**

- A l''état initial, le jeu est en Pause.
  - Le compteur de temporisation est initialisé à 0.

- On reste en pause tant que l'on a pas de requête de changement du mode Pause
  - Si tel est le cas, on sort du mode Pause et on démarre la temporisation
    - Le jeu est alors en mode actif.

- On reste dans le mode actif tant que l'on a pas de requête de changement du mode Pause
  - Si tel est le cas, on rentre en mode Pause et on démarre la temporisation
    - Le jeu est de nouveau stoppé.

- Il faut veiller à ce qu'UN ET UN SEUL changement de mode soit effectué POUR CHAQUE APPUI sur le bouton.
  - Si l'utilisateur garde le bouton poussoir appuyé, il ne faut pas changer plusieurs fois le mode Pause.
- Quand le jeu est actif
  - Si on détecte que toutes les briques ont été détruites, on génère le signal ***Brick_Win*** indéfiniment.
  - SI on détecte que la partie est perdue,
    - On fait un chargement de **Timer Lost** puis on passe en mode Pause

- Quand **Timer Lost** a été chargé,
  - A chaque fin d'image (***Endframe***) et tant que la valeur du compteur **Timer Lost** est supérieure à 0,
    - On décrémente la valeur de **Timer_Lost**
    - Cela permettra au signal ***Game_Lost*** (voir descriptif de **Timer Lost**) de rester activée pour une durée de 64 images.

9.3) Module Master/Slave Manager

- Ce sous-module, structuré en Machine à Etats, fixe le mode de fonctionnement de la SU-EE100
  - Mode Console → Master_Slave = 1
  - Mode Manette → Master_Slave = 0

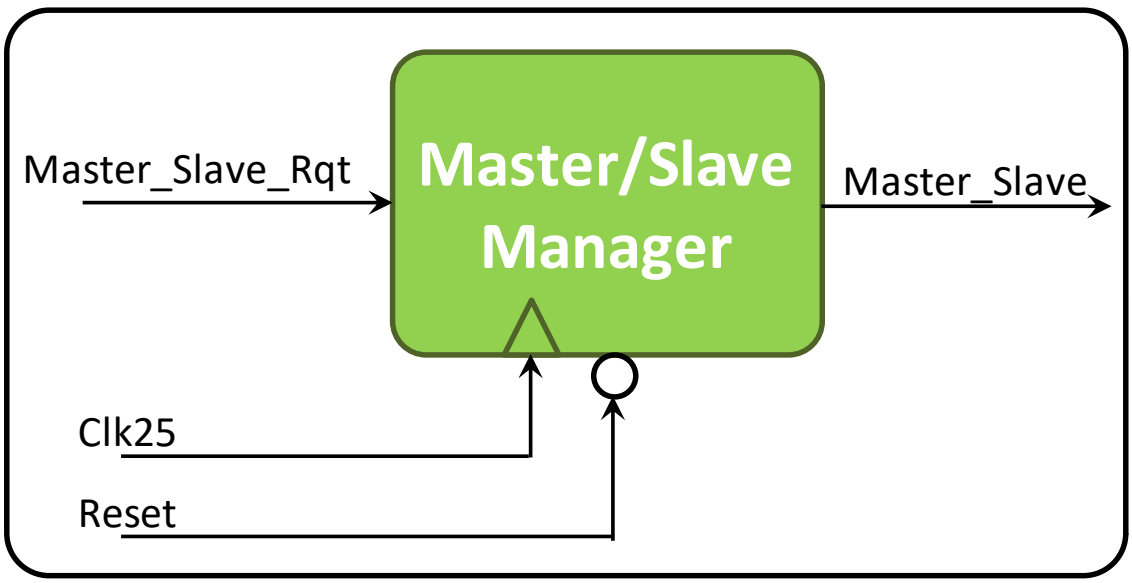


Figure 31 – Vue externe du sous-module Master/Slave Manager

- A l'état initial, la SU-EE100 est en mode Console
  - En cas de demande de changement de mode (***Master_Slave_Rqt***), la SU-EE100 change de mode
  - On va passer ainsi du mode Console au mode Manette, ou inversement.
  - Il faut veiller à ce qu'UN ET UN SEUL changement de mode soit effectué POUR CHAQUE REQUETE.
    - Une requête est terminée lorsque ***Master_Slave_Rqt*** est désactivée.

9.4) Module GameManager

- Ce sous-module, structuré en Machine à Etats, fixe le jeu actif de la SU-EE100
  - Casse Briques → ***Game_Type*** = 0
  - Pong → ***Game_Type*** = 1

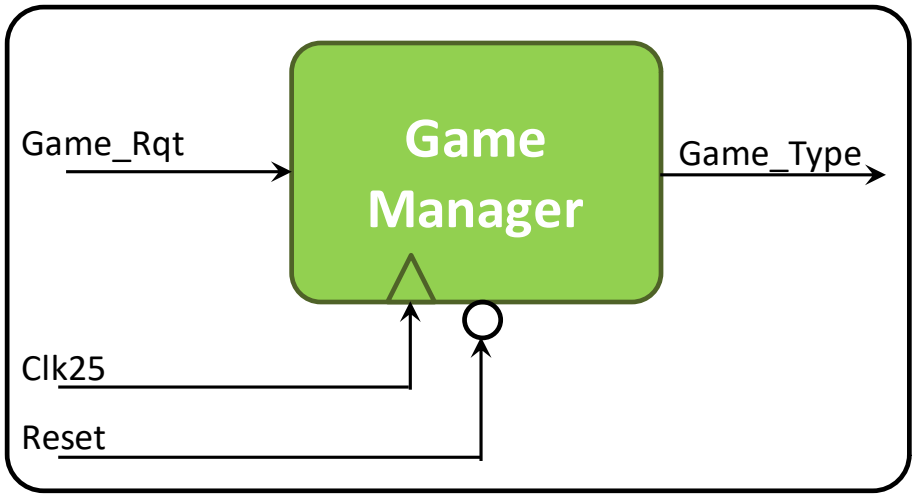


Figure 32 – Vue externe du sous-module Master/Slave Manager

- La MAE suit exactement le même principe de fonctionnement que celle de **Master/Slave Manager** (voir plus haut)
- A l'état initial, le jeu actif est le Casse Briques
  - SI on détecte une demande de changement de jeu (***Game_Rqt***), la SU-EE100 change de jeu
  - On va passer ainsi du Casse Briques à Pong, ou inversement.
  - Il faut veiller à ce qu'UN ET UN SEUL changement de mode soit effectué POUR CHAQUE REQUETE.
    - Une requête est terminée lorsque ***Game_Rqt*** est désactivée.