MINI PROJET SOC/FPGA

Ahmed AOUCHAR

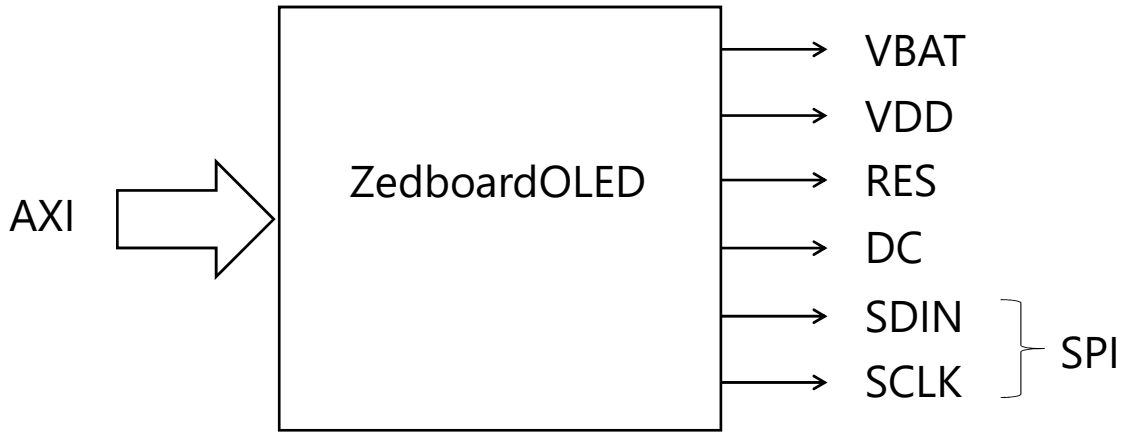
ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

L'École des INGÉNIEURS Scientifiques

1. OBJECTIF

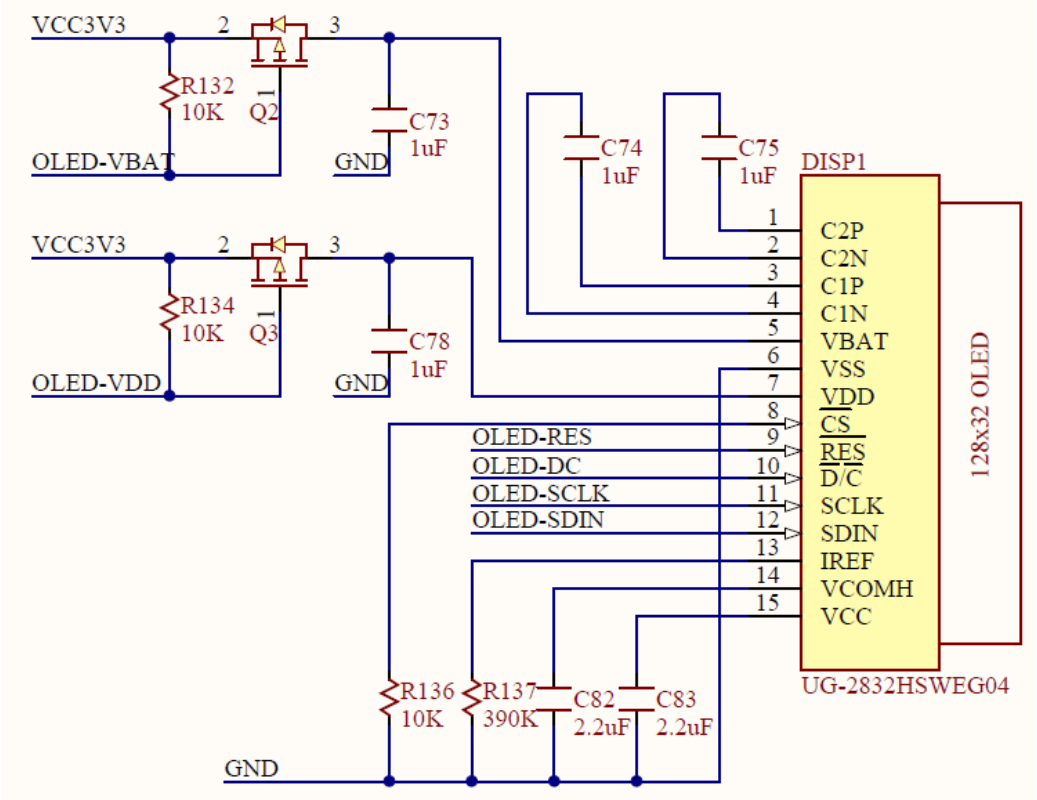
- Créer une propriété intellectuelle « IP » permettant d'ajouter un nouveau périphérique au processeur
- Ce périphérique est un afficheur matriciel OLED
- La communication avec l'afficheur se fera à travers l'interface AXI
- Un driver sera écrit pour faciliter la commande de l'afficheur dans les programmes d'application
- La partie proche du matériel sera codée en VHDL
- La couche supérieure sera codée en C
    1. Fonction « print_char »
    2. Fonction « print_message »
    3. Fonction « clear_OLED »

2. SYNOPTIQUE

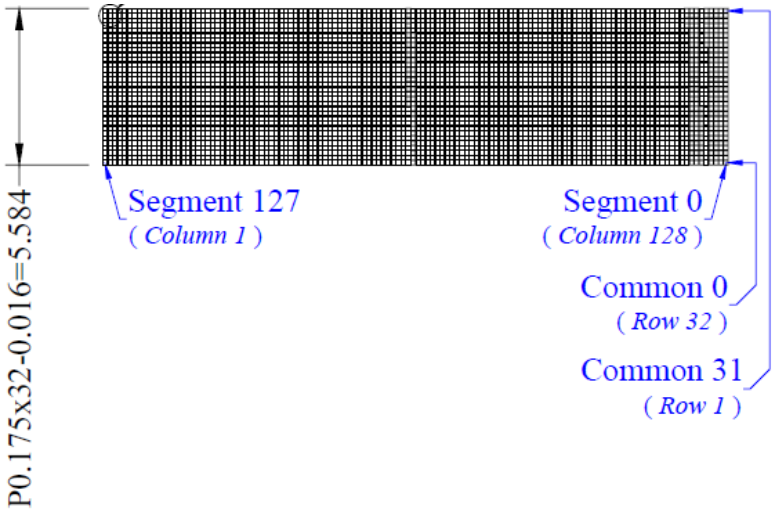


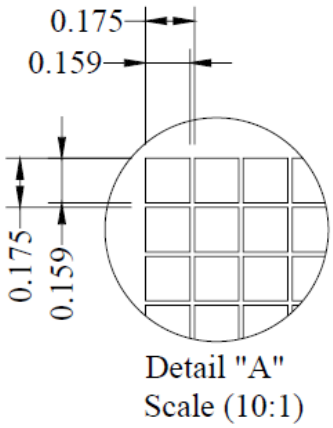
- VBAT : alimentation des segments
- VDD : alimentation de la logique
- RES : remise à zéro
- DC : commande / data
- SDIN : data SPI
- SCLK : horloge SPI

3. SCHÉMA DE BRANCHEMENT

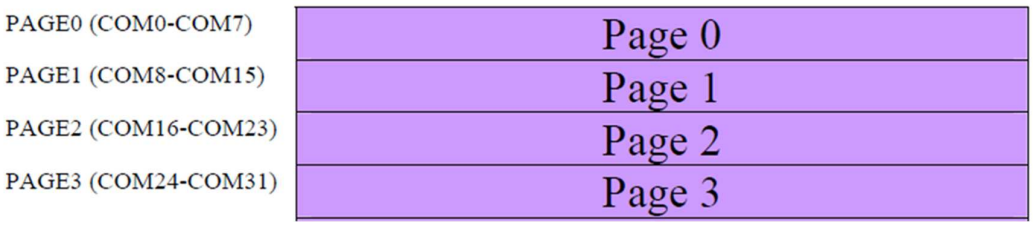


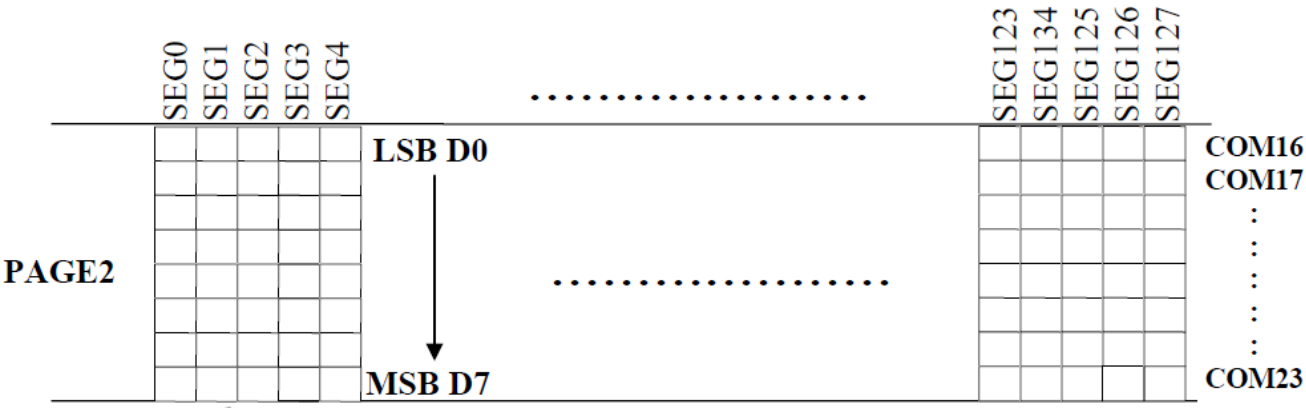
4. STRUCTURE DE L'AFFICHEUR





5. DÉCOUPAGE EN PAGES

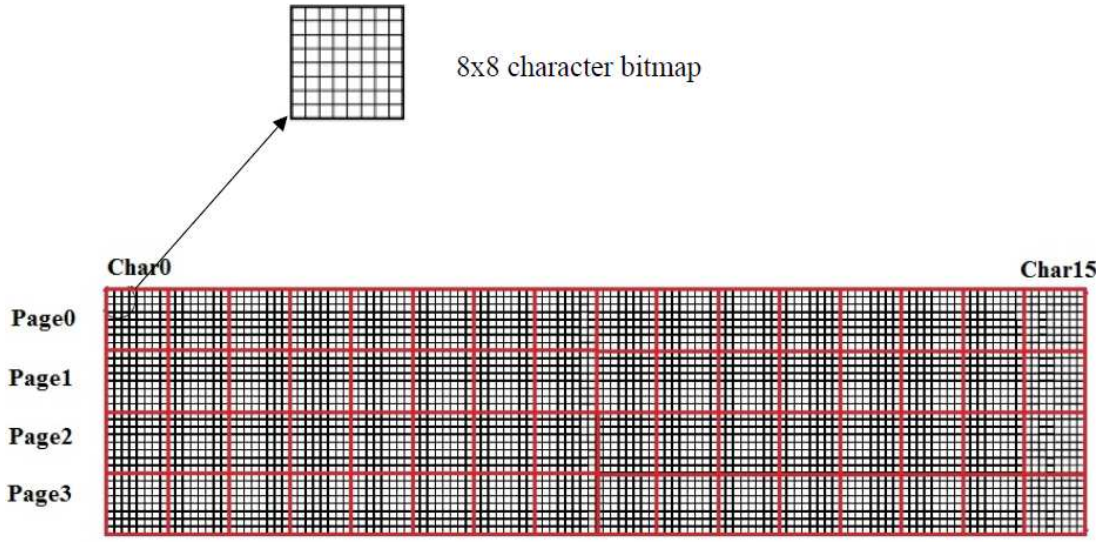




- Un caractère occupe une matrice 8x8 pixels

6. AFFICHAGE MATRICIEL DES CARACTÈRES

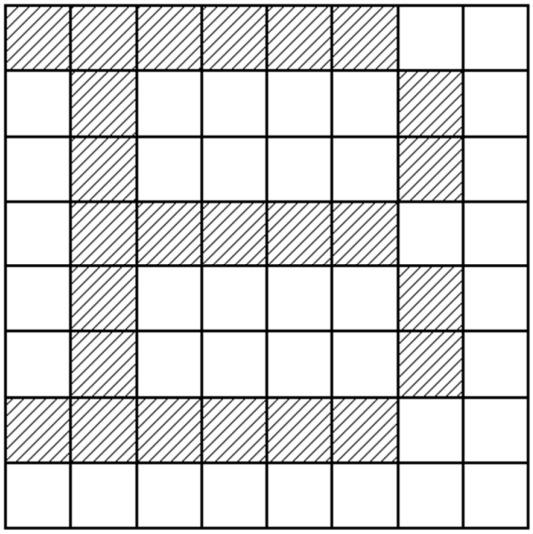
- Chaque page peut représenter 16 caractères avec une résolution de 8x8 pixels.



7. DESSIN DES MOTIFS OCTET PAR OCTET

- Un caractère occupe une matrice 8x8 pixels
- Il faut envoyer 8 octets par caractère (ex : lettre B)

41	7F	49	49	49	49	36	00

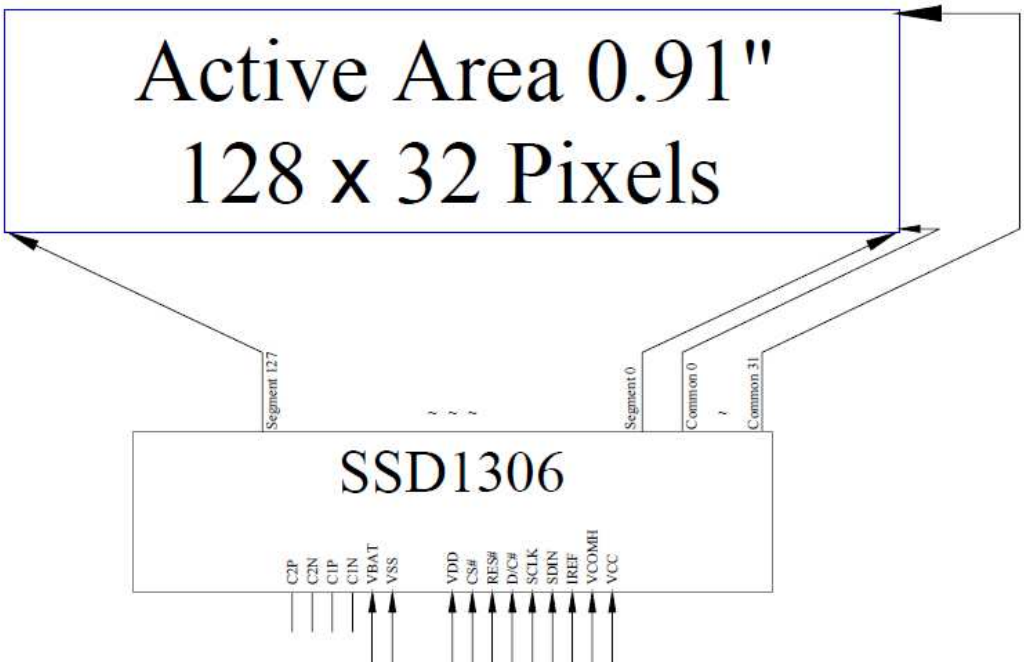


8. STOCKAGE DES EMPREINTES EN MÉMOIRE

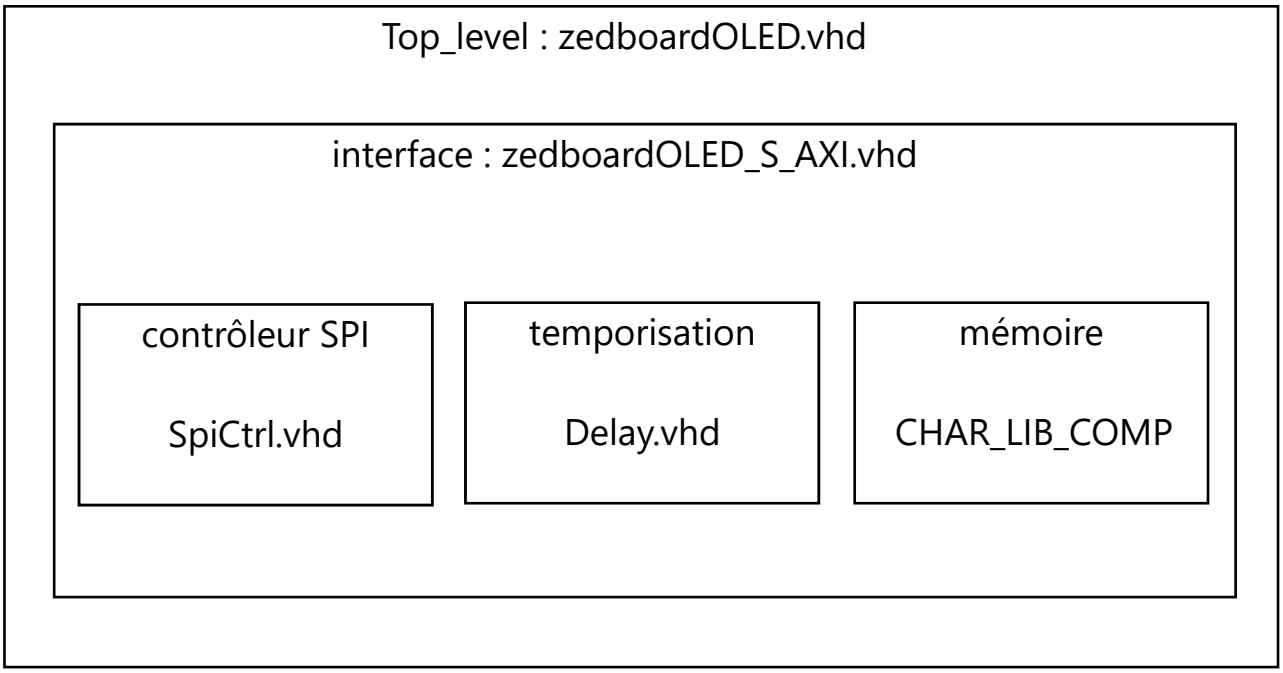
- Les empreintes des caractères sont stockées en mémoire
- Comme le titre occupe 3 octets, il faut ajouter un offset de 3

```
54      00,22,41,49,49,36,00,00,
55      00,18,14,12,7f,10,00,00,
56      00,27,49,49,49,71,00,00,
57      00,3c,4a,49,48,70,00,00,
58      00,43,21,11,0d,03,00,00,
59      00,36,49,49,49,36,00,00,
60      00,06,09,49,29,1e,00,00,
61      00,00,00,12,00,00,00,00,
62      00,00,00,52,30,00,00,00,
63      00,00,08,14,14,22,00,00,
64      00,14,14,14,14,14,14,00,
65      00,00,22,14,14,08,00,00,
66      00,02,01,59,05,02,00,00,
67      3e,41,5d,55,4d,51,2e,00,
68      40,7c,4a,09,4a,7c,40,00,
69      41,7f,49,49,49,49,36,00,   ←——  Lettre 'B' = 0x42
70      1c,22,41,41,41,41,22,00,
71      41,7f,41,41,41,22,1c,00,
72      41,7f,49,49,5d,41,63,00,
73      41,7f,49,09,1d,01,03,00,
```


9. LE CONTRÔLEUR EMBARQUÉ



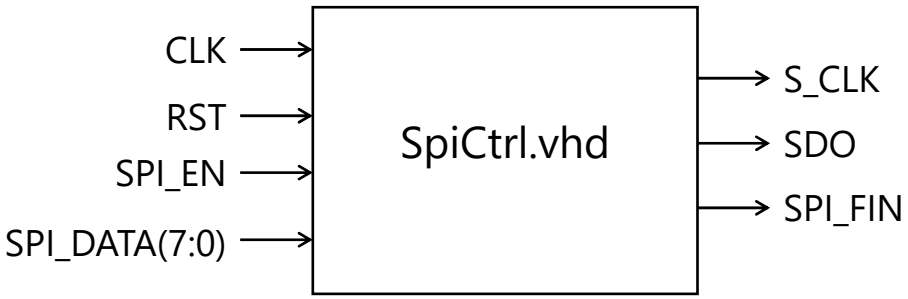
10. SYNOPTIQUE DU DRIVER



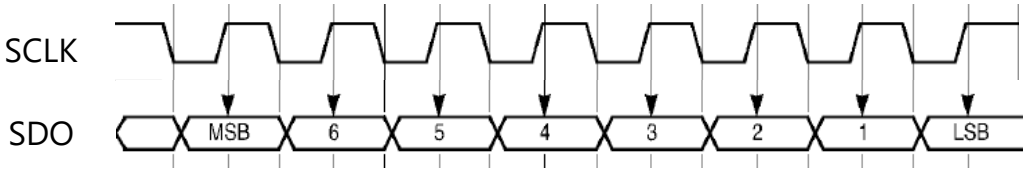
11. LE MODULE « DELAY »

- Clk : 100 MHz
- Le contrôleur zedboardOLED fixe la durée (en ms) sur le bus DELAY_MS et active le signal DLAY_EN pour lancer la temporisation
- Le signal DELAY_FIN signale au contrôleur la fin de la temporisation

12. LE CONTRÔLEUR SPI



- CLK = 100 MHz et S_CLK = 3,125 MHz
- Le contrôleur zedboardOLED positionne l'octet à transmettre sur le bus SPI_DATA et active le signal SPI_EN pour démarrer la transmission série
- SPI_FIN signale au contrôleur la fin de la transmission



13. LE MODULE ZEDBOARDOLED_S_AXI.VHD

- Ce module gère la plus grosse partie du travail du contrôleur
  1. L'interfaçage avec AXI
  2. Les registres de données slv_reg0 à slv_reg15 (chaque registre contient 4 codes ASCII « 4 caractères »)
  3. Le registre de contrôle slv_reg16
  4. L'initialisation de l'afficheur et l'envoi de données ou commandes

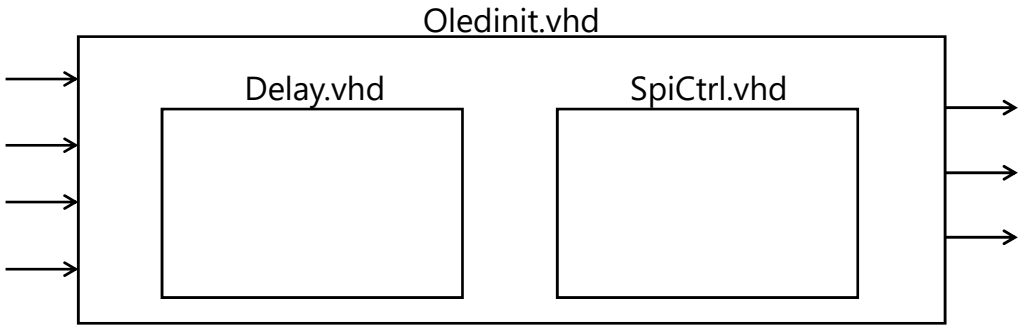
page0	Slv_reg0	Slv_reg1	Slv_reg2	Slv_reg3
page1	Slv_reg4	Slv_reg5	Slv_reg6	Slv_reg7
page2	Slv_reg8	Slv_reg9	Slv_reg10	Slv_reg11
page3	Slv_reg12	Slv_reg13	Slv_reg14	Slv_reg15

- slv_reg16(0) = 1 -> rafraichit l'affichage
- slv_reg16(1) = 1 -> efface l'afficheur
- Les autres bits du registre slv_reg16 sont inutilisés

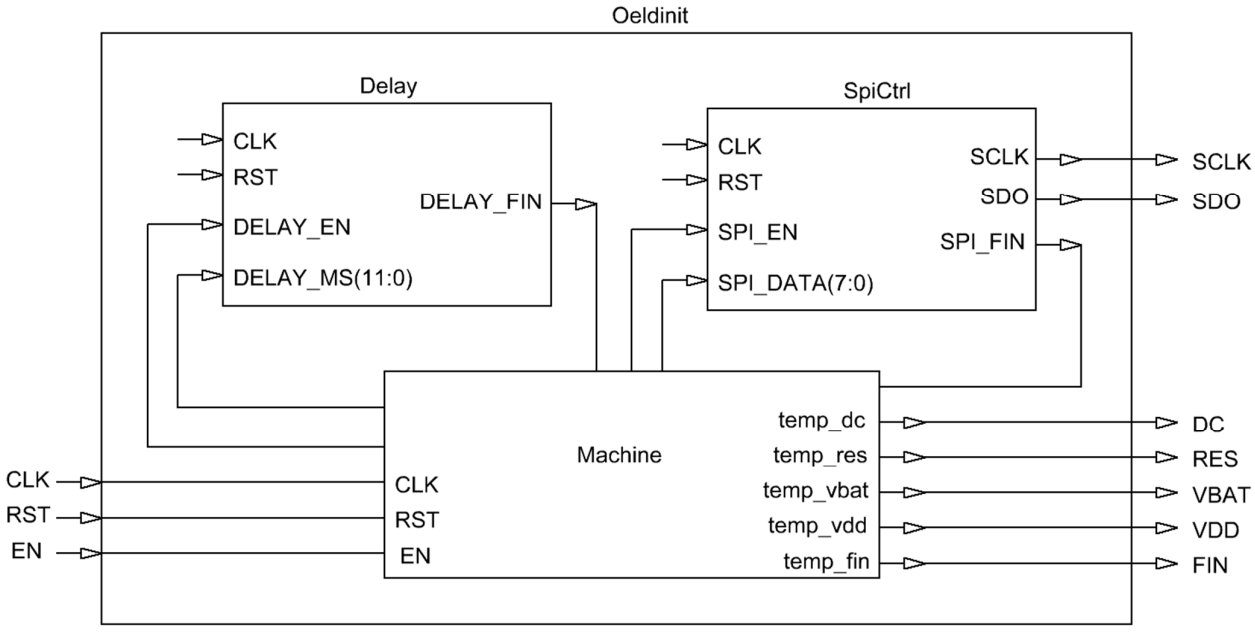
14. TRAVAIL À EFFECTUER : PARTIE 1

- Créer un nouveau projet nommé **OLED1** dans lequel seront testés les modules **Delay** et **SpiCtrl**, ainsi que l'initialisation de l'afficheur.

1. Coder le module Delay et le valider en simulation.
2. Faire le même travail avec le module SpiCtrl
3. Maintenant que les briques de base sont fonctionnelles, elles seront utilisées pour l'initialisation de l'afficheur et l'allumage de tous les pixels. Ce travail sera fait dans un nouveau programme vhdl nommé **Oledinit**. Il s'agit de coder la machine la machine d'états **oledinit_machine** disponible sur MOODLE.

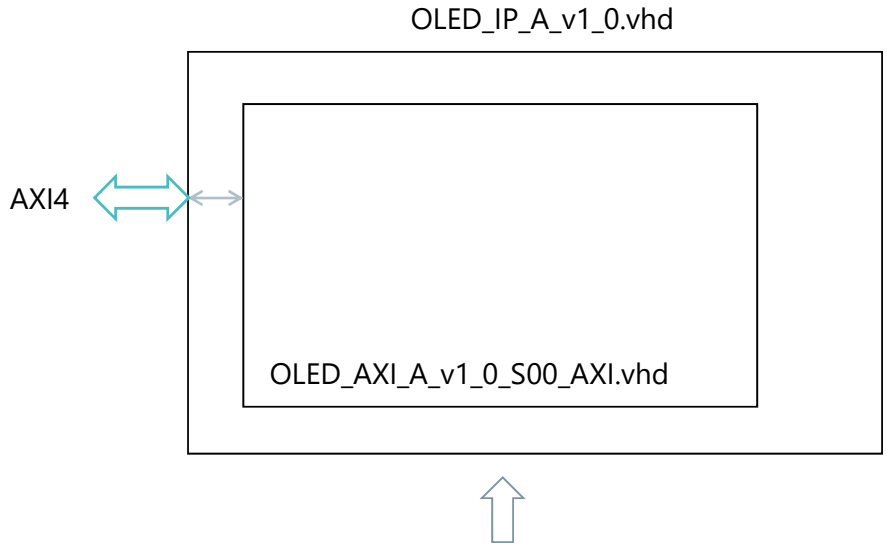


SYNOPTIQUE DU MODULE « OLEDINIT »



15. TRAVAIL À EFFECTUER : PARTIE 2

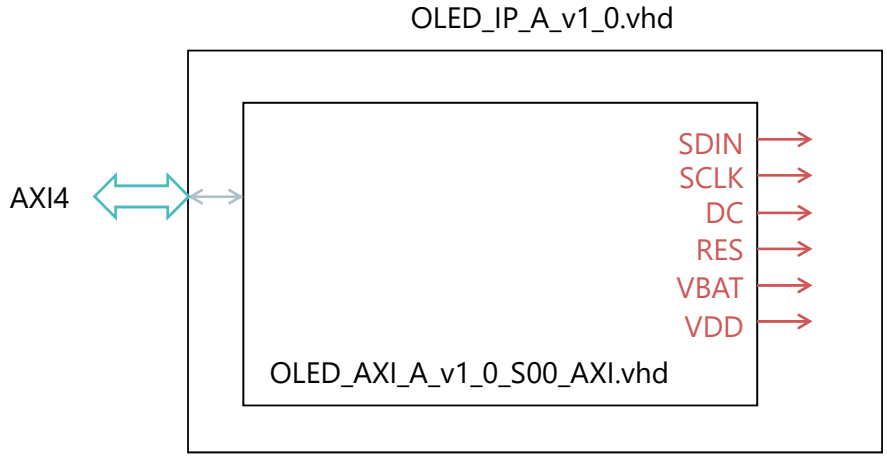
- Créer une nouvelle IP OLED (voir le fichier **création_OLED_IP.pdf**)



Ces deux fichiers sont créés automatiquement par l'outil « MANAGE IP »
La connexion AXI est faite mais pas les connexions avec le périphérique (???)
Il faut le ajouter à la main !

PARTIE 2-2

- Ajouter des sorties au fichier qui contrôle l'afficheur OLED :

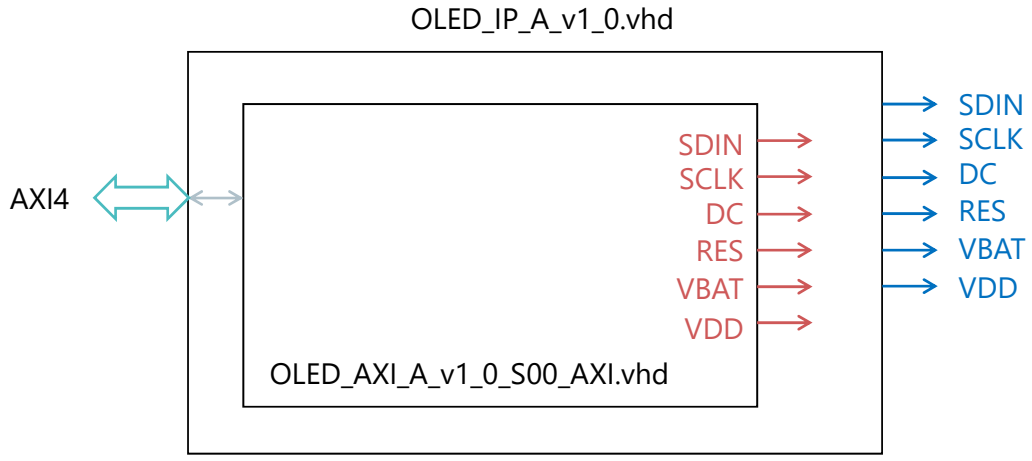



```
port (
    -- Users to add ports here
    SDIN : out std_logic;
    SCLK : out std_logic;
    DC : out std_logic;
    RES : out std_logic;
    VBAT : out std_logic;
    VDD : out std_logic;
```

Ajouté au fichier :
OLED_AXI_A_v1_0_S00_AXI.vhd

PARTIE 2-3

- Ajouter les mêmes sorties au fichier enveloppe (wrapper) :

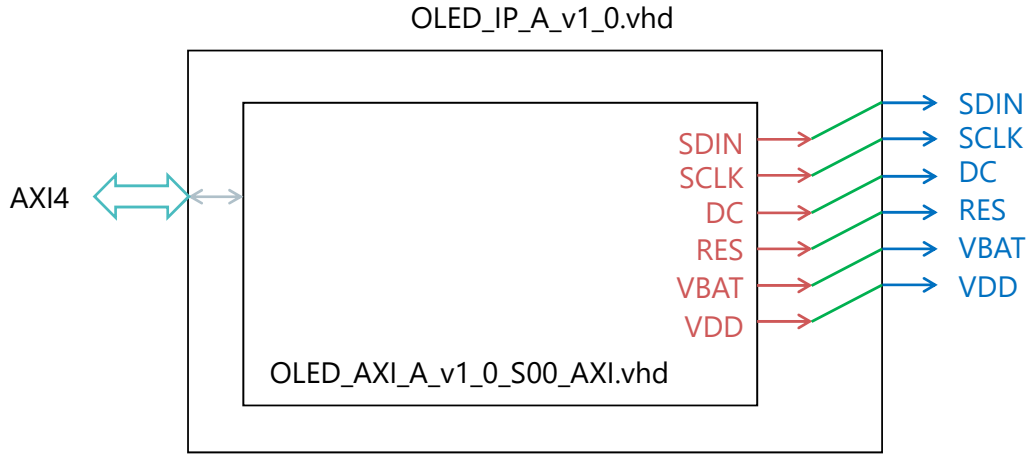



```
port (
    -- Users to add ports here
    SDIN : out std_logic;
    SCLK : out std_logic;
    DC : out std_logic;
    RES : out std_logic;
    VBAT : out std_logic;
    VDD : out std_logic;
```

Ajouté au fichier :
OLED_IP_A_v1_0.vhd

PARTIE 2-4

- Connecter les nouvelles sorties après les avoir déclarées dans le composant :



- Pour cela, il faut ajouter quelques compléments d'information au fichier enveloppe **OLED_IP_A_v1_0.vhd**

PARTIE 2-4 (SUITE)

```
component OLED_IP_A_v1_0_S00_AXI is
    generic (
    C_S_AXI_DATA_WIDTH  : integer    := 32;
    C_S_AXI_ADDR_WIDTH  : integer    := 7
    );
    port (
    SDIN : out std_logic;
    SCLK : out std_logic;
    DC : out std_logic;
    RES : out std_logic;
    VBAT : out std_logic;
    VDD : out std_logic;
```

⇦ Ajouté au fichier : OLED_IP_A_v1_0.vhd

Ajouté au fichier : OLED_IP_A_v1_0.vhd ⇨

```
OLED_IP_A_v1_0_S00_AXI_inst : OLED_IP_A_v1_0_S00_AXI
    generic map (
        C_S_AXI_DATA_WIDTH  => C_S00_AXI_DATA_WIDTH,
        C_S_AXI_ADDR_WIDTH  => C_S00_AXI_ADDR_WIDTH
    )
    port map (
        SDIN => SDIN,
        SCLK => SCLK,
        DC => DC,
        RES => RES,
        VBAT => VBAT,
        VDD => VDD
```


PARTIE 2-5

- Copier la logique de contrôle de l'afficheur « user logic » :

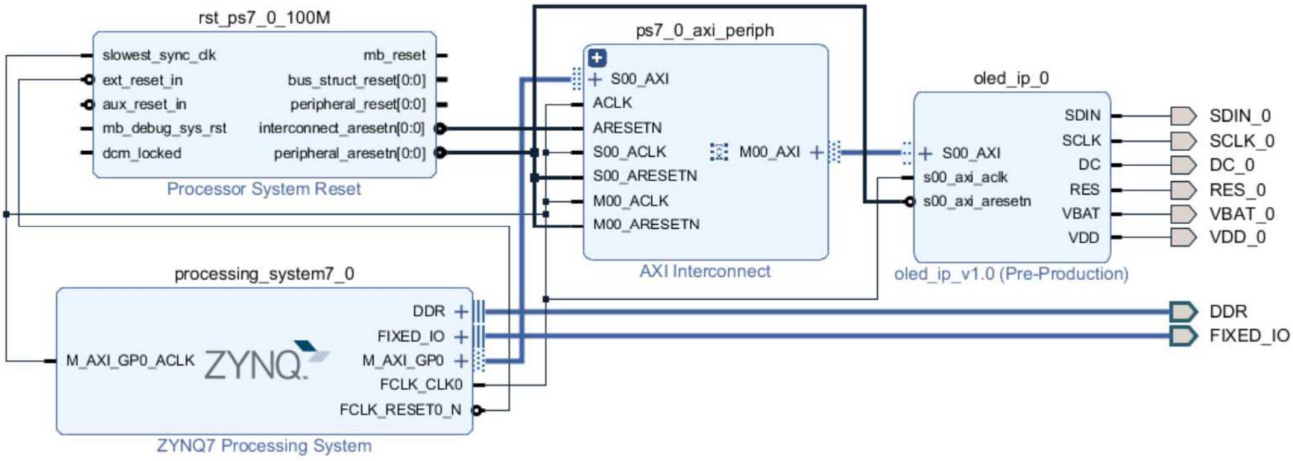
Ajouté au fichier : OLED_IP_A_v1_0.vhd ⇨

```
-- Add user logic here
--=============================================
--          début d'ajout
--=============================================

process (s_axi_aclk)
begin
if rising_edge(s_axi_aclk) then
    if (rst_in = '1') then
        current_state <= idle;
        temp_res <= '0';
    else
        temp_res <= '1';
        case (current_state) is
            when idle    =>
                if (init_first_r = '1') then
```

16. TRAVAIL À EFFECTUER : PARTIE 3

- Créer un nouveau projet qui va utiliser l'IP OLED :



- Générer le bitsteam et exporter la plateforme matérielle vers SDK.

PARTIE 3-2

Tester ce programme :
Quel message est affiché ?

```
#include "xparameters.h"
#include "xil_io.h"
#include <string.h>
#define DELAY 10000

/*  Base memory address of OLED_IP */
#define OLED_BASE XPAR_OLED_IP_0_S00_AXI_BASEADDR

int main(void)
{
    int i=0;

    Xil_Out32(OLED_BASE,0x44434241);
    Xil_Out32(OLED_BASE+4,0x48474645);
    Xil_Out32(OLED_BASE+8,0x4C4B4A49);
    Xil_Out32(OLED_BASE+12,0x504F4E4D);

    Xil_Out32(OLED_BASE+16,0x34333231);
    Xil_Out32(OLED_BASE+20,0x38373635);
    Xil_Out32(OLED_BASE+24,0x00003039);
    Xil_Out32(OLED_BASE+28,0x00000000);

    Xil_Out32(OLED_BASE+32,0x00000000);
    Xil_Out32(OLED_BASE+36,0x49534E45);
    Xil_Out32(OLED_BASE+40,0x4E454143);
    Xil_Out32(OLED_BASE+44,0x00000000);

    Xil_Out32(OLED_BASE+48,0x00000000);
    Xil_Out32(OLED_BASE+52,0x00000000);
    Xil_Out32(OLED_BASE+56,0x00000000);
    Xil_Out32(OLED_BASE+60,0x00000000);

    for (i=0;i<=DELAY ;i++)
        Xil_Out32(OLED_BASE+(64), 1);
    for (i=0;i<=DELAY ;i++)
        Xil_Out32(OLED_BASE+(64), 0);
    return 1;
}
```

- Ecrire de driver de l'afficheur OLED « zed_oled.c »

```
int print_char( char char_seq, unsigned int page ,unsigned int position);

int print_message(char *start , unsigned int page);

void clear(void);
```

- Développer une petite application basée sur le jeu du serpent (**snake**).