Ce projet initialise l'afficheur OLED et allume tous les pixels (voir machine d'état oledinit)

```
entity oledinit is
    Port ( CLK : in STD_LOGIC;
           RST : in STD_LOGIC;  -- poussoir BTNC
           EN : in std_logic;   -- poussoir BTND
           SDO : out STD_LOGIC;     -- spi data
           SCLK : out STD_LOGIC;    -- spi clock
           DC : out STD_LOGIC;      -- Data/command pin
           RES : out STD_LOGIC;     -- reset
           VBAT : out STD_LOGIC;
           VDD : out STD_LOGIC;
           FIN : out STD_LOGIC);    --  LED0
end oledinit;

architecture Behavioral of oledinit is

component Delay is
    Port ( CLK          : in  STD_LOGIC; --System CLK
           RST          : in STD_LOGIC;  --Global RST
           DELAY_MS     : in  STD_LOGIC_VECTOR (11 downto 0);
           DELAY_EN     : in  STD_LOGIC; --Delay block enable
           DELAY_FIN    : out  STD_LOGIC); --Delay finish flag
end component;

component spictrl is
    Port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           spi_en : in STD_LOGIC;
           spi_data : in STD_LOGIC_VECTOR (7 downto 0);
           sdo : out STD_LOGIC;
           sclk : out STD_LOGIC;
           spi_fin : out STD_LOGIC);
end component;

type etat_type is (spi, spiEnd, tempo, tempoEnd, back, idle,
                   vddOn, wait1, dispOff, resetOn, wait2,
                   resetOff, chargePump1, chargePump2,
                   preCharge1, preCharge2, vbatOn, wait3,
                   dispContrast1, dispContrast2,
                   invertDisp1, invertDisp2,
                   comConfig1, comConfig2,
                   dispOn, fullDisp, done);

signal etat, etat_retour : etat_type := idle;

signal temp_dc : STD_LOGIC := '0';      -- commande (0) / data (1)
signal temp_res : STD_LOGIC := '1';     -- reset actif à 0
signal temp_vbat : STD_LOGIC := '1';    -- vbat actif à 0
signal temp_vdd : STD_LOGIC := '1';     -- vdd actif à 0
signal temp_fin : STD_LOGIC := '0';

signal temp_delay_ms : STD_LOGIC_VECTOR (11 downto 0) := (others => '0');
signal temp_delay_en : STD_LOGIC := '0';
signal temp_delay_fin : STD_LOGIC;
signal temp_spi_en : STD_LOGIC := '0';
signal temp_spi_data : STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
signal temp_spi_fin : STD_LOGIC;
```