Escola Universitària d'Enginyeria
Tècnica Industrial de Barcelona
Consorci Escola Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Annexos

"PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT"

TFG presentat per optar al títol de GRAU en ENGINYERIA ELECTRÒNICA INDUSTRIAL I AUTOMÀTICA

per **Xavier Pidas i Pagerols**

Barcelona, 8 de Juny de 2016

Director: Jordi Cosp i Vilella
Departament d'EEL
Universitat Politècnica de Catalunya (UPC)

ÍNDEX ANNEXOS

ANNEX I. CODI VHDL

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    16:59:09 01/21/2016
5   -- Module Name:    MAIN - Structural
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12
13  entity MAIN is
14     port(CLK50:    in std_logic;-- CLK INPUT 50 MHz
15          RST:      in std_logic;-- Reset
16          TRIGGER:  in std_logic;--TRIGGER per capturar una imatge
17          INTERRUPTORS: in std_logic_vector(7 downto 0); --Per escollir el procés a
    aplicar
18
19          --Per la càmera
20          CAM_DATA: in std_logic_vector(7 downto 0);
21          PCLK:     in std_logic;
22          HS_CAM:   in std_logic;
23          VS_CAM:   in std_logic;
24          SCL:    out std_logic;
25          SDA:    out std_logic;
26          XCLK:   out std_logic;
27          PDN:    out std_logic;
28
29          --Pel port VGA
30          RGB:    out std_logic_vector(7 downto 0); -- 8 bit color (256 colors
    diferents) 3-bit R, 3-bit G, 2-bit B
31          HS_VGA: out std_logic;
32          VS_VGA: out std_logic;
33
34          -- PER LA RAM
35          DATA_RAM:    inout std_logic_vector (15 downto 0);
36          ADRESS_RAM:  out std_logic_vector(22 downto 0);
37          OE:          out std_logic;
38          WE:          out std_logic;
39          ADV:         out std_logic;
40          CLK_RAM:     out std_logic;
41          UB:          out std_logic;
42          LB:          out std_logic;
43          CE:          out std_logic;
44          CRE:         out std_logic);
45  end MAIN;
46
47  architecture Structural of MAIN is
48  ----------------------------------------------------------------------
49  --COMPONENTS
50     component CAM_DRIVER
51        port(VS_CAM, HS_CAM, PCLK, TRIGGER, CLK25, RST: in std_logic;
52             XCLK, PDN, SDA, SCL: out std_logic;
53             CAM_DATA: in std_logic_vector(7 downto 0);
54
55             --PER A RAM_DRIVER
```

```
56              ENABLE_PROCESSING: out std_logic;
57              WT: out std_logic;
58              DATA_OUT: out std_logic_vector(15 downto 0);
59              ADRESS_OUT: out std_logic_vector (22 downto 0));
60      end component;
61
62      component RAM_DRIVER
63         port(WT, RD, CLK50, RST: in std_logic;
64              ADRESS_IN_CAM: in std_logic_vector(22 downto 0);
65              ADRESS_IN_VGA: in std_logic_vector(22 downto 0);
66              ADRESS_PROC: in std_logic_vector(22 downto 0);
67              ADRESS_RAM: out std_logic_vector(22 downto 0);
68              DATA_IN_CAM: in std_logic_vector (15 downto 0);
69              DATA_RAM: inout std_logic_vector (15 downto 0);
70              DATA_PROC: inout std_logic_vector (15 downto 0);
71              DATA_OUT_VGA: out std_logic_vector(15 downto 0);
72              READ_PROC: in std_logic;
73              WRITE_PROC: in std_logic;
74              OE, WE, ADV, CLK_RAM, UB, LB, CE, CRE: out std_logic);
75      end component;
76
77      component VGA_DRIVER
78         port(CLK25, CLK50, RST: in std_logic;
79               HS, VS:  out std_logic;
80               RGB:  out std_logic_vector(7 downto 0);
81               IMATGE: in std_logic_vector(2 downto 0);
82
83               -- Per a RAM_DRIVER
84               RD: out std_logic; -- Per avisar a RAM_DRIVER que es vol llegir
85               READ_EN: in std_logic; --Avís que ja es pot començar a llegir de la ram
86               DATA_IN_RAM: in std_logic_vector (15 downto 0); --Dades de la RAM
87               ADRESS_OUT_RAM: out std_logic_vector( 22 downto 0)); --Adreça on llegir
    el color
88      end component;
89
90      component PROCESSING_IMAGE
91         port(ENABLE: in std_logic;
92              CLK50: in std_logic;
93              RST: in std_logic;
94              INTERRUPTORS: in std_logic_vector(7 downto 0);
95              IMATGE: out std_logic_vector(2 downto 0);
96              READ_EN: out std_logic;
97              READ_PROC: out std_logic;
98              WRITE_PROC: out std_logic;
99              DATA_RAM: inout std_logic_vector(15 downto 0);
100             ADRESS_RAM: out std_logic_vector(22 downto 0));
101        end component;
102
103     component CLK_DIV
104        port(CLK_IN: in std_logic;
105             CLK_OUT: out std_logic);
106     end component;
107  ----------------------------------------------------------------------
108  --SIGNALS
109     signal clk_25:std_logic;
110     signal wt_ram: std_logic;
111     signal rd_ram: std_logic;
```

```
112      signal read_en_proc: std_logic;
113      signal data_ram_vga: std_logic_vector(15 downto 0);
114      signal data_ram_cam: std_logic_vector(15 downto 0);
115      signal adress_cam: std_logic_vector(22 downto 0);
116      signal adress_vga: std_logic_vector(22 downto 0);
117      signal en_processing: std_logic;
118      signal data_proc_ram: std_logic_vector(15 downto 0);
119      signal adress_proc_ram:  std_logic_vector(22 downto 0);
120      signal read_proc_ram: std_logic;
121      signal write_proc_ram: std_logic;
122      signal imatge_vga: std_logic_vector(2 downto 0);
123
124   begin
125      VGA_DRIVER_1: VGA_DRIVER port map (CLK25=>clk_25, CLK50=>CLK50, RST=>RST, HS=>
         HS_VGA, VS=>VS_VGA, RGB=>RGB, IMATGE=>imatge_vga, READ_EN=>read_en_proc, RD=>rd_ram,
         DATA_IN_RAM=>data_ram_vga, ADRESS_OUT_RAM=>adress_vga);
126      CLK_DIV_1: CLK_DIV port map(CLK_IN=>CLK50, CLK_OUT=>clk_25);
127      CAM_DRIVER_1: CAM_DRIVER port map(VS_CAM=>VS_CAM, HS_CAM=>HS_CAM, PCLK=>PCLK,
         TRIGGER=>TRIGGER, CLK25=>clk_25, RST=>RST, XCLK=>XCLK, PDN=>PDN, SDA=>SDA, SCL=>SCL,
         CAM_DATA=>CAM_DATA, ENABLE_PROCESSING=>en_processing, WT=>wt_ram, DATA_OUT=>
         data_ram_cam, ADRESS_OUT=>adress_cam);
128      RAM_DRIVER_1: RAM_DRIVER port map(WT=>wt_ram, RD=>rd_ram, CLK50=>CLK50, RST=>RST,
         DATA_IN_CAM=>data_ram_cam, ADRESS_IN_CAM=>adress_cam, ADRESS_PROC=>adress_proc_ram,
         ADRESS_IN_VGA=>adress_vga, DATA_RAM=>DATA_RAM, ADRESS_RAM=>ADRESS_RAM, DATA_PROC=>
         data_proc_ram, DATA_OUT_VGA=>data_ram_vga, READ_PROC=>read_proc_ram ,WRITE_PROC=>
         write_proc_ram ,OE=>OE, WE=>WE, ADV=>ADV, CLK_RAM=>CLK_RAM, UB=>UB, LB=>LB, CE=>CE,
         CRE=>CRE);
129      PROCESSING_IMAGE_1: PROCESSING_IMAGE port map(ENABLE=>en_processing, CLK50=>CLK50,
         RST=>RST, INTERRUPTORS=>INTERRUPTORS, IMATGE=>imatge_vga, READ_EN=>read_en_proc,
         READ_PROC=>read_proc_ram, WRITE_PROC=>write_proc_ram, DATA_RAM=>data_proc_ram,
         ADRESS_RAM=>adress_proc_ram);
130   end Structural;
131
132
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    15:58:13 01/26/2016
5   -- Module Name:    CAM_DRIVER - Structural
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10
11  library IEEE;
12  use IEEE.STD_LOGIC_1164.ALL;
13  use ieee.std_logic_unsigned.all;
14
15  entity CAM_DRIVER is
16     port(VS_CAM:   in std_logic;
17          HS_CAM:   in std_logic;
18          PCLK:     in std_logic;
19          TRIGGER:  in std_logic;
20          CLK25:    in std_logic;
21          RST:      in std_logic;
22          XCLK: out std_logic;
23          PDN:  out std_logic;
24          SDA:  out std_logic;
25          SCL:  out std_logic;
26          CAM_DATA: in std_logic_vector(7 downto 0);
27
28          --PER A RAM_DRIVER
29          WT:           out std_logic;
30          ENABLE_PROCESSING: out std_logic; -- Si '0' -> VGA DRIVER no pot llegir de la
    RAM
31          DATA_OUT:     out std_logic_vector(15 downto 0);
32          ADRESS_OUT:   out std_logic_vector (22 downto 0));
33  end CAM_DRIVER;
34
35  architecture Structural of CAM_DRIVER is
36  ------------------------------------------
37  --COMPONENTS
38     component CAM_PROCESSOR
39        port(VS_CAM, TRIGGER, BUSY_SCCB, BUSY_IMPORTER, CLK25, RST: in std_logic;
40             XCLK, PDN, ENABLE_SCCB, ENABLE_DATA_IMPORTER: out std_logic;
41             SCCB_DATA,SUBADRESS: out std_logic_vector(7 downto 0);
42             ENABLE_PROCESSING: out std_logic);
43     end component;
44
45     component CAM_DATA_IMPORTER
46        port(HS_CAM,PCLK, RST, ENABLE_DATA_IMPORTER: in std_logic;
47             CAM_DATA: in std_logic_vector(7 downto 0);
48             WT, BUSY: out std_logic;
49             DATA_OUT: out std_logic_vector(15 downto 0);
50             ADRESS_OUT: out std_logic_vector (22 downto 0));
51     end component;
52
53     component SCCB_MODULE
54        port(ENABLE_SCCB,CLK25,RST: in std_logic;
55             BUSY, SDA, SCL: out std_logic;
56             SCCB_DATA,SUBADRESS: in std_logic_vector(7 downto 0));
```

```
57      end component;
58    ------------------------------------------------
59    --SIGNALS
60    signal busy: std_logic;
61    signal busy_imp: std_logic;
62    signal en_sccb: std_logic;
63    signal sccb_data: std_logic_vector (7 downto 0);
64    signal subadress: std_logic_vector(7 downto 0);
65    signal en_data_imp: std_logic;
66
67    ------------------------------------------------
68    begin
69       CAM_PROCESSOR_1: CAM_PROCESSOR port map(VS_CAM=>VS_CAM, TRIGGER=>TRIGGER, BUSY_SCCB
      =>busy, BUSY_IMPORTER=>busy_imp, CLK25=>CLK25, RST=>RST, XCLK=>XCLK, PDN=>PDN,
      ENABLE_SCCB=>en_sccb, ENABLE_DATA_IMPORTER=>en_data_imp, SCCB_DATA=>sccb_data,
      SUBADRESS=>subadress, ENABLE_PROCESSING=>ENABLE_PROCESSING);
70       CAM_DATA_IMPORTER_1: CAM_DATA_IMPORTER port map(HS_CAM=>HS_CAM, PCLK=>PCLK, RST=>
      RST, ENABLE_DATA_IMPORTER=>en_data_imp, CAM_DATA=>CAM_DATA, WT=>WT, BUSY=>busy_imp,
      DATA_OUT=>DATA_OUT, ADRESS_OUT=>ADRESS_OUT);
71       SCCB_MODULE_1: SCCB_MODULE port map(ENABLE_SCCB=>en_SCCB, CLK25=>CLK25, RST=>RST,
      BUSY=>busy, SDA=>SDA, SCL=>SCL, SCCB_DATA=>sccb_data, SUBADRESS=>subadress);
72    end Structural;
73
74
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer: Xavier Picas Pagerols
3   --
4   -- Create Date:    16:55:08 01/27/2016
5   -- Module Name:    CAM_PROCESSOR - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity CAM_PROCESSOR is
15     port(VS_CAM:   in std_logic;
16          TRIGGER:  in std_logic;
17          BUSY_SCCB:in std_logic;
18          BUSY_IMPORTER: in std_logic;
19          CLK25:    in std_logic;
20          RST:      in std_logic;
21          XCLK:     out std_logic;
22          PDN:      out std_logic;
23          ENABLE_SCCB:    out std_logic;
24          ENABLE_DATA_IMPORTER: out std_logic;
25          SCCB_DATA: out std_logic_vector(7 downto 0);
26          SUBADRESS: out std_logic_vector(7 downto 0);
27          ENABLE_PROCESSING: out std_logic);
28     end CAM_PROCESSOR;
29
30  architecture Behavioral of CAM_PROCESSOR is
31     type TIPUS is (RESET, POWER_UP, SEND_DATA_SCCB, ESPERA_SCCB, ESPERA_TRIGGER,
    ESPERA_VSYNC, GUARDAR_DADES, ESPERA_IMPORTER);
32     type TIPUS_TRIGGER is (S0, S1, S2);
33
34     signal ESTAT:TIPUS;
35     signal ESTAT_TRIGGER: TIPUS_TRIGGER;
36
37     signal preescaler: integer range 0 to 125000;
38     signal preescaler2: integer range 0 to 125;
39     signal REGISTRE: integer range 1 to 5; --N° registre a escriure (MODIFICAR PER
    TANTS REGISTRES COM HI HAGI)
40     signal en_trig: std_logic; --Enable del TRIGGER
41     signal primera_foto: std_logic; --per fer saber al VGA DRIVER que no pot llegir
    fins que no s'hagi capturat la primera foto.
42
43  begin
44
45     XCLK<=CLK25;
46
47     process(CLK25)
48     begin
49        if (CLK25='0' and CLK25'event) then
50           if (RST='1') then
51              ESTAT <= RESET;
52              ESTAT_TRIGGER <= S0;
53              PDN<='1';
54              ENABLE_SCCB<='0';
```

```
55              ENABLE_DATA_IMPORTER<='0';
56              en_trig<='0';
57           else
58
     ---------------------------------------------------------------------------
     MÀQUINA D'ESTATS POLSADOR TRIGGER
59              case ESTAT_TRIGGER is
60                 when S0 =>
61                    en_trig<='0';
62                    if (TRIGGER='1') then
63                       ESTAT_TRIGGER<=S1;
64                    else
65                       ESTAT_TRIGGER<=S0;
66                    end if;
67                 when S1 =>
68                    en_trig<='1';
69                    ESTAT_TRIGGER<=S2;
70                 when S2 =>
71                    en_trig<='0';
72                    if (TRIGGER='0') then
73                       ESTAT_TRIGGER<=S0;
74                    else
75                       ESTAT_TRIGGER<=S2;
76                    end if;
77              end case;
78
     ---------------------------------------------------------------------------
     MÀQUINA D'ESTATS PRINCIPIAL
79              case ESTAT is
80                 ---------------------- RESET
81                 when RESET =>
82                       en_trig<='0';
83                       REGISTRE<=1;
84                       preescaler<=0;
85                       preescaler2<=0;
86                       ENABLE_PROCESSING<='0';
87                       primera_foto<='0';
88                       PDN<='1';
89                       ESTAT<=POWER_UP;
90                 ----------------------
91                 when POWER_UP =>
92                    preescaler <= preescaler + 1;
93                    PDN<='1';
94                    if (preescaler=125000) then -- 5ms
95                       preescaler<=0;
96                       PDN<='0';
97                       ESTAT<=SEND_DATA_SCCB;
98                    else
99                       ESTAT<=POWER_UP;
100                   end if;
101                ---------------------- SEND DATA SCCB
102                when SEND_DATA_SCCB =>
103                   ENABLE_SCCB<='1';
104                   ESTAT<=ESPERA_SCCB;
105                   case REGISTRE is
106                      when 1 =>  -- MIRROR AND FLIP (PER FER PROVES)
107                         SUBADRESS<=X"1E";
```

```
108                         SCCB_DATA<=X"01";  --default value 0x01  // flip 0x10 //
    mirror 0x20 // mirror and flip 0x30
109                      when 2 => -- TEST PATTERN (PER FER PROVES)
110                         SUBADRESS<=X"71";
111                         SCCB_DATA<=X"00";-- 0x00 per desactivar-ho // 0x80 per
    activar-ho
112                      when 3 => -- OUTPUT FORMAT
113                         SUBADRESS<=X"12";
114                         SCCB_DATA<=X"04";
115                      when 4 => -- RGB565
116                         SUBADRESS<=X"40";
117                         SCCB_DATA<=X"D0";
118                      when 5 => -- RGB444 ENABLE
119                         SUBADRESS<=X"8C";
120                         SCCB_DATA<=X"02";
121                   end case;
122                ---------------------- ESPERA SCCB
123                when ESPERA_SCCB=>
124                   ENABLE_SCCB<='0';
125                   preescaler<=preescaler+1;
126                   if (preescaler>=2) then -- Espera de 120ns abans de llegir el BUSY
    per evitar llegir un '0' abans d'hora
127                      if (BUSY_SCCB='1') then
128                         ESTAT<=ESPERA_SCCB;
129                      else
130                         preescaler2<=preescaler2+1;
131                         if (preescaler2=125) then -- Espera de 5us abans de tornar a
    enviar dades
132                            preescaler2<=0;
133                            if (REGISTRE<5) then --- canviar el numero per tants
    registres com es vulgui escriure (IMPORTANT CANVAIR EL NUMERO!)
134                               REGISTRE<=REGISTRE+1;
135                               ESTAT<=SEND_DATA_SCCB;
136                            else --REGISTRE=5
137                               ESTAT<=ESPERA_TRIGGER;
138                               preescaler<=0;
139                            end if;
140                         end if;
141                      end if;
142                   end if;
143                -------------------------- ESPERA EL TRIGGER PER CAPTURAR UNA IMATGE
144                when ESPERA_TRIGGER =>
145                   if (en_trig='1') then
146                      ESTAT<=ESPERA_VSYNC;
147                      ENABLE_PROCESSING<='0'; --Mentre es captura la imatge,
    PROCESSING IMAGE no pot actuar
148                   else
149                      if (primera_foto='1') then --No és la primera foto capturada
150                         ENABLE_PROCESSING<='1'; --PROCESSING IMAGE pot actuar
151                      else --És la primera foto que es tira
152                         ENABLE_PROCESSING<='0'; --PROCESSING IMAGE no pot actuar fins
    que no s'hagi tirat la primera foto
153                      end if;
154                      ESTAT<=ESPERA_TRIGGER;
155                   end if;
156                -------------------------- ESPERA LA SINCRONITZACIÓ VERTICAL
157                when ESPERA_VSYNC =>
```

```
158                     if (VS_CAM='1') then
159                        ESTAT<=GUARDAR_DADES;
160                     -- preescaler<=0;
161                     else
162                        ESTAT<=ESPERA_VSYNC;
163                     end if;
164                  ---------------------------- S'ACTIVA L'ENABLE DEL  CAM_DATA_IMPORTER
165                  when GUARDAR_DADES =>
166                     ENABLE_DATA_IMPORTER<='1';
167                     preescaler<=preescaler+1;
168                     if (preescaler=2) then --120ns
169                        preescaler<=0;
170                        ESTAT<=ESPERA_IMPORTER;
171                     end if;
172                  ---------------------------------- ESPERA QUE IMPORTER ACABI
173                  when ESPERA_IMPORTER =>
174                     ENABLE_DATA_IMPORTER<='0';
175                     if (BUSY_IMPORTER='0') then
176                        ESTAT<=ESPERA_TRIGGER;
177                        primera_foto<='1';
178                     else
179                        ESTAT<=ESPERA_IMPORTER;
180                     end if;
181                  ------------------------------------
182               end case;
183            end if;
184         end if;
185      end process;
186   end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer: Xavier Picas Pagerols
3   --
4   -- Create Date:    16:46:36 01/26/2016
5   -- Module Name:    SCCB_MODULE - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use ieee.std_logic_unsigned.all;
13
14
15  entity SCCB_MODULE is
16        port(ENABLE_SCCB: in std_logic;
17             CLK25:  in std_logic;
18             RST:    in std_logic;
19             BUSY:   out std_logic;
20             SDA:    out std_logic;
21             SCL:    out std_logic;
22
23             SCCB_DATA: in std_logic_vector(7 downto 0);
24             SUBADRESS: in std_logic_vector(7 downto 0));
25  end SCCB_MODULE;
26
27  architecture Behavioral of SCCB_MODULE is
28     type TIPUS is (ESPERA_ENABLE, START, WRT, ACK, PULSE, STOP);
29     signal ESTAT: TIPUS;
30
31     type TIPUS_FASE is (IP, ADRESS, DATA);
32     signal FASE: TIPUS_FASE; --Fase on es troba la transmissió // FASE 1: IP ADRESS //
   FASE 2: SUB-ADRESS // FASE 3: DATA //
33     signal data_to_write: std_logic_vector(7 downto 0); --Byte a escriure
34
35     signal PULSE_STATE: integer range 0 to 2; --Control del PULSE
36     signal preescaler: integer range 0 to 125; --Per tenir 100kHz
37     signal count: integer range 0 to 8; --Compador dels bits enviats
38
39     constant IPadress: std_logic_vector(7 downto 0):=X"42"; --Adreça de la càmera
   (esclau)    -WRITE=0x42- / READ=0x43
40
41  begin
42     process(CLK25)
43     begin
44        if (CLK25='0' and CLK25'event) then
45           if (RST='1') then
46              ESTAT <= ESPERA_ENABLE;
47              SDA<='1';
48              SCL<='1';
49           else
50              case ESTAT is
51                 ----------------RESET//ESPERA_ENABLE
52                 when ESPERA_ENABLE =>
53                    SDA<='1';
54                    SCL<='1';
55                    preescaler<=0;
```

```
56                      count<=0;
57                      BUSY<='0';
58                      FASE<=IP;
59                      PULSE_STATE<=0;
60                      if (ENABLE_SCCB='1') then
61                         BUSY<='1';
62                         ESTAT<=START;
63                      else
64                         ESTAT<=ESPERA_ENABLE;
65                      end if;
66                   ---------------- START
67                   when START =>
68                      data_to_write<=IPadress; --El primer byte a enviar és l'adreça de
    l'esclau
69                      SDA<='0';
70                      PREESCALER <= PREESCALER + 1;
71                      if (PREESCALER=125) then --5us
72                         PREESCALER<=0;
73                         SCL<='0';
74                         ESTAT<=WRT;
75                      else
76                         ESTAT<=START;
77                      end if;
78                   ---------------- WRITE
79                   when WRT =>
80                      if (count<=7) then
81                         SDA<=data_to_write(7);
82                         ESTAT<=PULSE;
83                      else
84                         count<=0;
85                         ESTAT<=ACK;
86                      end if;
87                   ---------------- PULSE
88                   when PULSE=>
89
90                      case PULSE_STATE is
91                         when 0 =>
92                            SCL<='0';
93                            preescaler<=preescaler+1;
94                            if (preescaler=62) then -- 2,5us
95                               preescaler<=0;
96                               PULSE_STATE<=1;
97                               ESTAT<=PULSE;
98                            end if;
99                         when 1 =>
100                           SCL<='1';
101                           preescaler<=preescaler+1;
102                           if (preescaler=125) then -- 5us
103                              preescaler<=0;
104                              PULSE_STATE<=2;
105                              ESTAT<=PULSE;
106                           end if;
107                        when 2 =>
108                           SCL<='0';
109                           preescaler<=preescaler+1;
110                           if (preescaler=62) then -- 2,5us
111                              preescaler<=0;
```

```
112                              data_to_write(7 downto 1) <= data_to_write(6 downto 0);
    --Shift left one
113                              ESTAT<=WRT;
114                              PULSE_STATE<=0;
115                              count<=count+1;
116                           end if;
117                        end case;
118                     ------------------------------ ACK
119                     when ACK =>
120                        SDA<='0';
121                        case PULSE_STATE is
122                           when 0 =>
123                              SCL<='0';
124                              preescaler<=preescaler+1;
125                              if (preescaler=62) then -- 2,5us
126                                 preescaler<=0;
127                                 PULSE_STATE<=1;
128                                 ESTAT<=ACK;
129                              end if;
130                           when 1 =>
131                              SCL<='1';
132                              preescaler<=preescaler+1;
133                              if (preescaler=125) then -- 5us
134                                 preescaler<=0;
135                                 PULSE_STATE<=2;
136                                 ESTAT<=ACK;
137                              end if;
138                           when 2 =>
139                              SCL<='0';
140                              preescaler<=preescaler+1;
141                              if (preescaler=62) then -- 2,5us
142                                 preescaler<=0;
143                                 PULSE_STATE<=0;
144                                 case FASE is
145                                    when IP =>
146                                       FASE<=ADRESS;
147                                       data_to_write<=SUBADRESS;
148                                       ESTAT<=WRT;
149                                    when ADRESS =>
150                                       FASE<=DATA;
151                                       data_to_write<=SCCB_DATA;
152                                       ESTAT<=WRT;
153                                    when DATA =>
154                                       FASE<=IP;
155                                       ESTAT<=STOP;
156                                       SDA<='0';
157                                 end case;
158                              end if;
159                        end case;
160                  ------------------------------------ STOP
161                  when STOP =>
162                     SCL<='1';
163                     PREESCALER <= PREESCALER + 1;
164                     if (PREESCALER=125) then --5us
165                        PREESCALER<=0;
166                        SDA<='1';
167                        ESTAT<=ESPERA_ENABLE;
```

```
168                         BUSY<='0';
169                     end if;
170                 end case;
171             end if;
172         end if;
173     end process;
174 end Behavioral;
175
176
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer: Xavier Picas Pagerols
3   --
4   -- Create Date:    13:59:21 02/20/2016
5   -- Module Name:    CAM_DATA_IMPORTER - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity CAM_DATA_IMPORTER is
15     port(HS_CAM: in std_logic;
16          PCLK: in std_logic;
17          RST: in std_logic;
18          ENABLE_DATA_IMPORTER: in std_logic;
19          CAM_DATA: in std_logic_vector(7 downto 0);
20          WT: out std_logic;
21          BUSY: out std_logic;
22          DATA_OUT: out std_logic_vector(15 downto 0);
23          ADRESS_OUT: out std_logic_vector (22 downto 0));
24  end CAM_DATA_IMPORTER;
25
26  architecture Behavioral of CAM_DATA_IMPORTER is
27     TYPE TIPUS is (ESPERA_ENABLE, GUARDAR_DADES);
28     signal ESTAT: TIPUS;
29
30     signal count_data: integer range 1 to 2; --Comptador per guardar els dos bytes del
    pixel
31     signal count_row: integer range 0 to 480; --Comptador per saber en quina fila es
    passa
32     signal count_pixel: integer range 0 to 640; --Comptador per saber en quin pixel es
    passa
33     signal data1, data2: std_logic_vector (7 downto 0); --On es guarden els dos bytes
    previament
34     signal adress_out_ram: std_logic_vector(22 downto 0);
35
36  begin
37
38     DATA_OUT <= data1 & data2; --DATA_OUT = [15:8]+[7:0]
39
40     ADRESS_OUT <= adress_out_ram;
41
42
43     process(PCLK)
44     begin
45        if (PCLK='1' and PCLK'event) then
46           if (RST='1') then
47              ESTAT<=ESPERA_ENABLE;
48              BUSY<='0';
49           else
50              case ESTAT is
51                 ------------------------------------ RESET//ESPERA ENABLE
52                 when ESPERA_ENABLE =>
53                       adress_out_ram<=(others=>'0');-- ES GUARDA A LA IMATGE 0
```

```
54                         count_data<=1;
55                         count_row<=0;
56                         count_pixel<=0;
57                         WT<='0';
58                         data1<=X"00";
59                         data2<=X"00";
60                         if (ENABLE_DATA_IMPORTER='1') then
61                            ESTAT<=GUARDAR_DADES;
62                            BUSY<='1';
63                         else
64                            ESTAT<=ESPERA_ENABLE;
65                         end if;
66                   ------------------------------------ GUARDAR_DADES
67                   when GUARDAR_DADES =>
68                      if (HS_CAM='1') then
69                         case count_data is
70                            when 1 => --es guarda el primer byte
71                               data1<=CAM_DATA;
72                               count_data<=2;
73                               WT<='0';
74                               if (not(count_row=0 and count_pixel=0)) then
75                                  adress_out_ram<=adress_out_ram+1;
76                               end if;
77                            when 2 => --es guarda el segon byte i s'envia a la RAM
78                               WT<='1';
79                               data2<=CAM_DATA;
80                               count_data<=1;
81                               if (count_pixel=639) then --últim pixel de la línia
82                                  count_pixel<=0;
83                                  if (count_row=479) then --Si és l'última línia
84                                     BUSY<='0';
85                                     ESTAT<=ESPERA_ENABLE;
86                                  else
87                                     count_row<=count_row+1;
88                                  end if;
89                               else
90                                  count_pixel<=count_pixel+1;
91                               end if;
92                         end case;
93                      else
94                         WT<='0';
95                      end if;
96                      ------------------------------------------------------------
97                end case;
98             end if;
99          end if;
100      end process;
101   end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer: Xavier Picas Pagerols
3   --
4   -- Create Date:    12:16:42 02/01/2016
5   -- Module Name:    RAM_DRIVER - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity RAM_DRIVER is
15     port(WT:              in std_logic;
16          CLK50:           in std_logic;
17          RST:             in std_logic;
18          RD:              in std_logic;
19          ADRESS_IN_CAM:   in std_logic_vector(22 downto 0);
20          ADRESS_IN_VGA:   in std_logic_vector(22 downto 0);
21          ADRESS_RAM:      out std_logic_vector(22 downto 0);
22          ADRESS_PROC:     in std_logic_vector(22 downto 0);
23          DATA_RAM:        inout std_logic_vector (15 downto 0);
24          DATA_IN_CAM:     in std_logic_vector (15 downto 0);
25          DATA_OUT_VGA:    out std_logic_vector(15 downto 0);
26          DATA_PROC:       inout std_logic_vector (15 downto 0);
27          READ_PROC:       in std_logic;
28          WRITE_PROC:      in std_logic;
29          OE:              out std_logic;
30          WE:              out std_logic;
31          ADV:             out std_logic;
32          CLK_RAM:         out std_logic;
33          UB:              out std_logic;
34          LB:              out std_logic;
35          CE:              out std_logic;
36          CRE:             out std_logic);
37  end RAM_DRIVER;
38
39  architecture Behavioral of RAM_DRIVER is
40     type TIPUS is (RESET, CONFIGURE, READ_ARRAY, WRITE_RAM, READ_RAM_BURST,
    READ_RAM_ASYNC, STANDBY, SEND_DATA_VGA);
41     signal ESTAT:TIPUS;
42
43     signal DATA_RAM_IN, DATA_PROC_IN: std_logic_vector (15 downto 0);
44     signal DATA_RAM_OUT, DATA_PROC_OUT: std_logic_vector (15 downto 0);
45     signal OE_inout, OE_inout_proc: std_logic; --OUTPUT ENABLE (inout)
46     signal preescaler: integer range 0 to 4; --preescaler per comptar 80 ns
47     signal clk_ram_enable: std_logic;
48     signal clk_ram_counter: integer range 0 to 12; --contador de bytes llegits
49
50  begin
51
52     with clk_ram_enable select   --Enable per activar CLK RAM
53        CLK_RAM <= CLK50   when '1',
54                   '0'     when others;
55
56     ----------------------------------------------------------------------------------
```

```
57       DATA_RAM_IN<=DATA_RAM;--Sempre es llegeix DATA_RAM
58
59       process(OE_inout, DATA_RAM_OUT) --PROCESS per controlar DATA_RAM inout
60       begin
61          if (OE_inout='1') then
62             DATA_RAM<=DATA_RAM_OUT;
63          else
64             DATA_RAM<=(others=>'Z');
65          end if;
66       end process;
67
68       ---------------------------------------------------------------------------
69       DATA_PROC_IN<=DATA_PROC;--Sempre es llegeix DATA_PROC
70
71       process(OE_inout_proc, DATA_PROC_OUT) --PROCESS per controlar DATA_PROC inout de
     PROCESSING IMAGE
72       begin
73          if (OE_inout_proc='1') then
74             DATA_PROC<=DATA_PROC_OUT;
75          else
76             DATA_PROC<=(others=>'Z');
77          end if;
78       end process;
79
80       ---------------------------------------------------------------------------
81       process(CLK50) --PROCESS PRINCIPAL
82       begin
83          if (CLK50='1' and CLK50'event) then
84             if (RST='1') then
85                ESTAT<=RESET;
86                DATA_OUT_VGA<=(others=>'0');
87                OE<='0';
88                WE<='1';
89                ADV<='1';
90                UB<='0';
91                LB<='0';
92                CE<='1';
93                CRE<='0';
94                preescaler<=0;
95                OE_inout<='0';
96                OE_inout_proc<='0';
97                clk_ram_enable<='0';
98                clk_ram_counter<=0;
99             else
100               case ESTAT is
101                  ------------------------- RESET
102                  when RESET =>
103                     ADRESS_RAM<="000100000101110000011010"; --Canviar el valor en cas
     d'una configuracio diferent
104                     ESTAT<=CONFIGURE;
105                  ------------------------- Configuració de la RAM
106                  when CONFIGURE =>
107                     CRE<='1';
108                     ADV<='0';
109                     CE<='0'; --Chip enabled
110                     WE<='0'; --WRITE (max 4us)
111                     if (preescaler<4) then --ESPERA DE 80ns (min 70 ns)
```

```
112                        preescaler<=preescaler+1;
113                     else
114                        preescaler<=0;
115                        ADRESS_RAM<=(others=>'0');
116                        ADV<='1';
117                        CE<='1'; --Chip disabled
118                        WE<='1';
119                        CRE<='0';
120                        ESTAT<=READ_ARRAY;
121                     end if;
122                  -------------------------- READ ARRAY AFTER CONFIGURE
123                  when READ_ARRAY =>
124                     ADV<='0';
125                     CE<='0'; --Chip enabled
126                     if (preescaler<4) then --READ cycle time 70ns min
127                        preescaler<=preescaler+1;
128                     else
129                        preescaler<=0;
130                        ADV<='1';
131                        CE<='1'; --Chip disabled
132                        WE<='1';
133                        ESTAT<=STANDBY;
134                     end if;
135                  ------------------------- STAND BY
136                  when STANDBY =>
137                     ADRESS_RAM<=(others=>'0');
138                     ADV<='1';
139                     CE<='1'; --Chip disabled
140                     WE<='1';
141                     OE_inout<='0';
142                     OE_inout_proc<='0';
143                     clk_ram_enable<='0';
144                     preescaler<=0;
145                     clk_ram_counter<=0;
146
147                     if (WT='1') then
148                        ESTAT<=WRITE_RAM;
149                        ADRESS_RAM<=ADRESS_IN_CAM;     --latch
150                        DATA_RAM_OUT<=DATA_IN_CAM;     --latch
151                        OE_inout<='1';
152                        --
153                        CE<='0';   --WRITE
154                        ADV<='0';
155                        WE<='0';
156                        --
157                     elsif (RD='1') then
158                        ESTAT<=READ_RAM_BURST;
159                        ADRESS_RAM<=ADRESS_IN_VGA;
160                     elsif (READ_PROC='1') then
161                        ESTAT<=READ_RAM_ASYNC;
162                        ADRESS_RAM<=ADRESS_PROC;
163                        OE_inout<='0';
164                        OE_inout_proc<='1';
165                        --
166                        CE<='0';   --READ
167                        ADV<='0';
168                        WE<='1';
```

```
169                        --
170                     elsif (WRITE_PROC='1') then
171                        ESTAT<=WRITE_RAM;
172                        ADRESS_RAM<=ADRESS_PROC;     --latch
173                        DATA_RAM_OUT<=DATA_PROC_IN;  --latch
174                        OE_inout<='1';
175                        --
176                        CE<='0';   --WRITE
177                        ADV<='0';
178                        WE<='0';
179                        --
180                     else
181                        ESTAT<=STANDBY;
182                     end if;
183                  ------------------------- WRITE RAM (ASYNC)
184                  when WRITE_RAM =>
185                     if (preescaler<2) then --Espera de 60ns
186                        preescaler<=preescaler+1;
187                     else
188                        preescaler<=0;
189                        ESTAT<=STANDBY;
190                        -- al cap de 60 ns es posa en standby
191                        ADV<='1';
192                        CE<='1';
193                        WE<='1';
194                     end if;
195                  ------------------------- READ RAM (BURST SYNC)
196                  when READ_RAM_BURST =>
197                     OE_inout<='0';
198                     CE<='0';
199                     WE<='1';
200                     if (clk_ram_counter<1) then
201                        ADV<='0';
202                     else
203                        ADV<='1';
204                     end if;
205
206                     if (preescaler<1) then --Espera d'un període per activar clk_ram i
       iniciar el compte de bytes
207                        preescaler<=preescaler+1;
208                     else
209                        clk_ram_enable<='1';
210                        clk_ram_counter<=clk_ram_counter+1;
211                        if (clk_ram_counter=2) then
212                           ESTAT<=SEND_DATA_VGA;
213                        end if;
214                     end if;
215                  ------------------------- READ RAM (ASYNC)
216                  when READ_RAM_ASYNC =>
217                     if (preescaler<3) then --Espera de 80ns (min 70 ns)
218                        preescaler<=preescaler+1;
219                     else
220                        preescaler<=0;
221                        ESTAT<=STANDBY;
222                        DATA_PROC_OUT<=DATA_RAM_IN;
223                     end if;
224                  ------------------------- SEND DATA TO VGA PORT (a partir de
```

```
      clk_ram_counter=3)
225                when SEND_DATA_VGA =>
226                   DATA_OUT_VGA<=DATA_RAM_IN;
227                   if (clk_ram_counter<=10) then --Es llegeixen i s'envien els 8 bytes
228                      clk_ram_counter<=clk_ram_counter+1;
229                   else --Un cop s'han llegit els 8 bytes
230                      CE<='1'; --Chip disabled
231                      clk_ram_counter<=0;
232                      clk_ram_enable<='0';
233                      ESTAT<=STANDBY;
234                   end if;
235                ------------------------
236             end case;
237          end if;
238       end if;
239    end process;
240 end Behavioral;
241
242
```

```
1  ----------------------------------------------------------------------------------
2  -- Engineer:  Xavier Picas Pagerols
3  --
4  -- Create Date:    10:34:22 01/26/2016
5  -- Module Name:    VGA_DRIVER - Structural
6  -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7  --
8  -- Per a Nexys2 board
9  ----------------------------------------------------------------------------------
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use ieee.std_logic_unsigned.all;
13
14 entity VGA_DRIVER is
15    port(CLK25: in std_logic; --25MHz
16         CLK50: in std_logic; --50MHz (pel bloc de PIXEL_COLOR)
17         RST:   in std_logic;
18         HS:          out std_logic;
19         VS:          out std_logic;
20         RGB:         out std_logic_vector(7 downto 0); -- 8 bit color (256 colors
   diferents) 3-bit R, 3-bit G, 2-bit B
21         IMATGE: in std_logic_vector(2 downto 0);
22         RD:                out std_logic;-- Per avisar la RAM
23         READ_EN:        in std_logic;-- Avís que ja es pot començar a llegir de la ram
24         DATA_IN_RAM:       in std_logic_vector (15 downto 0); --Dades de la RAM
25         ADRESS_OUT_RAM:  out std_logic_vector( 22 downto 0)); --Adreça on llegir el
   color
26 end VGA_DRIVER;
27
28
29 architecture STRUCTURAL of VGA_DRIVER is
30 ----------------------------------------------------
31 --COMPONENTS
32    component HC
33       port (CLK25,RST: in std_logic;
34             EN: out std_logic;
35             Q: out std_logic_vector (9 downto 0));
36    end component;
37
38    component VC
39       port (CLK25,EN,RST: in std_logic;
40             Q: out std_logic_vector(9 downto 0));
41    end component;
42
43    component COMP_D
44       port (H_CNT,V_CNT: in std_logic_vector(9 downto 0);
45             AV: out std_logic);
46    end component;
47
48    component COMP_S
49       port (IN_CNT: in std_logic_vector(9 downto 0);
50             MAX: in std_logic_vector (9 downto 0);
51             OUT_COMP: out std_logic);
52    end component;
53
54    component PIXEL_COLOR is
55       port(H_CNT,V_CNT: in std_logic_vector(9 downto 0);
```

```
56              CLK50, CLK25, RST, AV: in std_logic;
57              RGB: out std_logic_vector(7 downto 0);
58              IMATGE: in std_logic_vector(2 downto 0);
59
60              READ_EN: in std_logic;
61              RD: out std_logic;
62              DATA_IN_RAM: in std_logic_vector(15 downto 0);
63              ADRESS_OUT_RAM: out std_logic_vector (22 downto 0));
64      end component;
65
66   ------------------------------------------
67   -- Signals
68      signal E: std_logic;
69      signal AV_comD: std_logic;
70      signal hcnt, vcnt: std_logic_vector(9 downto 0);
71
72   -- Constants
73      constant MAX_H: std_logic_vector(9 downto 0):="0001100000"; --96   Valors màxims
     pels compadadors
74      constant MAX_V: std_logic_vector(9 downto 0):="0000000010"; --2
75
76      ------------------------------------------------------
77      begin
78
79         HC_1: HC port map (CLK25=>CLK25, RST=>RST, EN=>E, Q=>hcnt);
80         VC_1: VC port map (CLK25=>CLK25, RST=>RST, EN=>E, Q=>vcnt);
81         COMP_D_1: COMP_D port map (H_CNT=>hcnt, V_CNT=>vcnt, AV=>AV_comD);
82         COMP_S_1: COMP_S port map (IN_CNT=>hcnt, MAX=>MAX_H, OUT_COMP=>HS);
83         COMP_S_2: COMP_S port map (IN_CNT=>vcnt, MAX=>MAX_V, OUT_COMP=>VS);
84         PIXEL_COLOR_1: PIXEL_COLOR port map(H_CNT=>hcnt, V_CNT=>vcnt, CLK50=>CLK50,
     CLK25=>CLK25, RST=>RST, AV=>AV_comD, RGB=>RGB, IMATGE=>IMATGE, READ_EN=>READ_EN, RD=>
     RD, DATA_IN_RAM=>DATA_IN_RAM, ADRESS_OUT_RAM=>ADRESS_OUT_RAM);
85
86   end STRUCTURAL;
87
88
```

```
1  ----------------------------------------------------------------------------------
2  -- Engineer:  Xavier Picas Pagerols
3  --
4  -- Create Date:    10:40:22 01/26/2016
5  -- Module Name:    HC - Behavioral
6  -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7  --
8  -- Per a Nexys2 board
9  ----------------------------------------------------------------------------------
10
11 library IEEE;
12 use IEEE.STD_LOGIC_1164.ALL;
13 use ieee.std_logic_unsigned.all;
14
15 entity HC is
16    port (CLK25,RST: in std_logic;
17          EN: out std_logic;
18          Q: out std_logic_vector (9 downto 0));
19 end HC;
20
21 architecture Behavioral of HC is
22 signal count: std_logic_vector (9 downto 0);
23 begin
24       process (CLK25)
25       begin
26          if (CLK25'event and CLK25='1') then
27             if RST='1' then
28                count<="1100011111";--799
29                EN<='1';
30             elsif (count="1100011111") then --799 reinicia la horitzontal
31                count<="0000000000";
32                EN<='0';
33             elsif count="1100011110" then -- a 798 posa TC a '1'
34                count <= count+1;
35                EN<='1';
36             else
37                count <= count+1;
38                EN<='0';
39             end if;
40          end if;
41       end process;
42
43       Q<=count;
44
45 end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    10:56:44 01/26/2016
5   -- Module Name:    VC - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use ieee.std_logic_unsigned.all;
13
14  entity VC is
15     port (CLK25,EN,RST: in std_logic;
16           Q: out std_logic_vector(9 downto 0));
17  end VC;
18
19  architecture Behavioral of VC is
20  signal count: std_logic_vector (9 downto 0);
21  begin
22        process (CLK25)
23        begin
24           if (CLK25'event and CLK25='1') then
25              if RST='1' then
26                 count<="1000001100"; -- 524
27              elsif EN='1' then
28                 if count="1000001100" then --524
29                    count<="0000000000";
30                 else
31                    count <= count+1;
32                 end if;
33              end if;
34           end if;
35        end process;
36
37        Q<=count;
38
39  end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    10:44:21 01/26/2016
5   -- Module Name:    COMP_D - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10
11  library IEEE;
12  use IEEE.STD_LOGIC_1164.ALL;
13  use ieee.std_logic_unsigned.all;
14
15  entity COMP_D is
16     port (H_CNT,V_CNT: in std_logic_vector(9 downto 0);
17           AV: out std_logic);
18  end COMP_D;
19
20  architecture Behavioral of COMP_D is
21     constant hmin:integer:=144;
22     constant hmax:integer:=784;
23     constant vmin:integer:=35;
24     constant vmax:integer:=515;
25
26  begin
27     process (H_CNT,V_CNT)
28     begin
29        if (H_CNT>=hmin and H_CNT<hmax and V_CNT>=vmin and V_CNT<vmax) then --ELS DOS
    CONTADORS HAN D'ESTAR A DISPLAY AREA
30           AV<='1';
31        else
32           AV<='0';
33        end if;
34     end process;
35  end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    10:45:10 01/26/2016
5   -- Module Name:    COMP_S - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10
11  library IEEE;
12  use IEEE.STD_LOGIC_1164.ALL;
13  use ieee.std_logic_unsigned.all;
14
15  entity COMP_S is
16  port (IN_CNT: in std_logic_vector(9 downto 0);
17        MAX: in std_logic_vector (9 downto 0);
18        OUT_COMP: out std_logic);
19  end COMP_S;
20
21  architecture Behavioral of COMP_S is
22
23  begin
24     process (IN_CNT,MAX)
25     begin
26        if (IN_CNT>=0 and IN_CNT<MAX) then -- 96 px per Hcount o 2 linies per Vcount
27           OUT_COMP<='1';
28        else
29           OUT_COMP<='0';
30        end if;
31     end process;
32  end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    12:15:20 02/15/2016
5   -- Module Name:    PIXEL_COLOR - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity PIXEL_COLOR is
15        port(H_CNT: in std_logic_vector(9 downto 0);
16             V_CNT: in std_logic_vector(9 downto 0);
17             CLK50: in std_logic;
18             CLK25: in std_logic;
19             RST:   in std_logic;
20             AV:    in std_logic;
21             RGB:   out std_logic_vector(7 downto 0);
22             IMATGE: in std_logic_vector(2 downto 0);
23
24             READ_EN: in std_logic;
25             RD: out std_logic;
26             DATA_IN_RAM: in std_logic_vector(15 downto 0);
27             ADRESS_OUT_RAM: out std_logic_vector (22 downto 0));
28  end PIXEL_COLOR;
29
30  architecture Behavioral of PIXEL_COLOR is
31  ------------------------------------------------
32  -- COMPONENTS
33     component RAM_DATA_IMPORTER is
34        port (READ_EN, CLK50, RST: in std_logic;
35              RD: out std_logic;
36              IMATGE: in std_logic_vector(2 downto 0);
37              H_CNT, V_CNT: in std_logic_vector(9 downto 0);
38              DATA_IN_RAM: in std_logic_vector(15 downto 0);
39              ADRESS_OUT_RAM: out std_logic_vector (22 downto 0);
40              CODED_DATA: out std_logic_vector(127 downto 0));
41     end component;
42
43     component DATA_DECODER is
44        port(CLK25,RST,AV: in std_logic;
45             CODED_DATA: in std_logic_vector(127 downto 0);
46             RGB: out std_logic_vector(7 downto 0));
47     end component;
48  ------------------------------------------------
49     signal data: std_logic_vector(127 downto 0);
50
51  begin
52
53     RAM_DATA_IMPORTER_1: RAM_DATA_IMPORTER port map(READ_EN=>READ_EN, CLK50=>CLK50, RST
    =>RST, RD=>RD, IMATGE=>IMATGE, H_CNT=>H_CNT, V_CNT=>V_CNT, DATA_IN_RAM=>DATA_IN_RAM,
    ADRESS_OUT_RAM=>ADRESS_OUT_RAM, CODED_DATA=>data);
54     DATA_DECODER_1: DATA_DECODER port map(CLK25=>CLK25, RST=>RST, AV=>AV, CODED_DATA=>
    data, RGB=>RGB);
```

```
55
56    end Behavioral;
57
58
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    13:08:07 02/15/2016
5   -- Module Name:    RAM_DATA_IMPORTER - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity RAM_DATA_IMPORTER is
15     port (READ_EN: in std_logic;
16           CLK50: in std_logic;
17           RST: in std_logic;
18           RD: out std_logic;
19           IMATGE: in std_logic_vector(2 downto 0);
20           H_CNT: in std_logic_vector(9 downto 0);
21           V_CNT: in std_logic_vector(9 downto 0);
22           DATA_IN_RAM: in std_logic_vector(15 downto 0);
23           ADRESS_OUT_RAM: out std_logic_vector (22 downto 0);
24           CODED_DATA: out std_logic_vector(127 downto 0));
25  end RAM_DATA_IMPORTER;
26
27   architecture Behavioral of RAM_DATA_IMPORTER is
28     type TIPUS is (RESET, INICI, IMPORTACIO_DADES, ESPERA_NOVA_LECTURA, ESPERA_HCNT);
29     signal ESTAT:TIPUS;
30
31     signal data1, data2: std_logic_vector(127 downto 0);
32     signal counter: integer range 0 to 16;
33     signal data_enabled: std_logic; -- '0' per data1 // '1' per data 2
34     signal adress_out: std_logic_vector(22 downto 0);
35     signal preescaler: std_logic;
36
37  begin
38
39     ADRESS_OUT_RAM<=adress_out;
40
41     process(CLK50)
42     begin
43        if (CLK50='0' and CLK50'event) then
44           if (RST='1' OR READ_EN='0') then --
45              data1<=(others=>'1');
46              data2<=(others=>'1');
47              CODED_DATA<=data1; -- Si READ_EN='0' s'envia el color blanc
48              ESTAT<=RESET;
49           else
50              case ESTAT is
51                 ---------------------------------- RESET
52                 when RESET =>
53                    data1<=(others=>'1');
54                    data2<=(others=>'1');
55                    counter<=15;
56                    data_enabled<='0';
57                    preescaler<='0';
```

```
58                   adress_out<=(others=>'0');
59                   ESTAT<=INICI;
60                ---------------------------------- ESPERA DEL PRIMER PIXEL per
    començar a llegir de la RAM
61                when INICI =>
62                   case IMATGE is  --Decidir adreça inicial
63                      when "000" => adress_out<="00000000000000000000000";--0
64                      when "001" => adress_out<="00001001011000000000000";--307200
65                      when "010" => adress_out<="00010010110000000000000";--614400
66                      when "011" => adress_out<="00011100001000000000000";--921600
67                      when "100" => adress_out<="00100101100000000000000";--1228800
68                      when "101" => adress_out<="00101110111000000000000";--1536000
69                      when "110" => adress_out<="00111000010000000000000";--1843200
70                      when others => adress_out<=(others=>'0');
71                   end case;
72                   counter<=15;
73                   data_enabled<='0';
74                   preescaler<='0';
75                   if (H_CNT="0010000111" and V_CNT="00100011") then --HCNT=135 i
    VCNT=35 -> començar a llegir la ram
76                      RD<='1'; --S'envia senyal de lectura de RAM
77                      if (preescaler='0') then --Espera d'un clock per passar a llegir
    les dades
78                         preescaler<='1';
79                      else
80                         preescaler<='0';
81                         ESTAT<=IMPORTACIO_DADES;
82                      end if;
83                   else
84                      ESTAT<=INICI;
85                   end if;
86                ----------------------------------  IMPORTACIÓ DE LES DADES
87                when IMPORTACIO_DADES =>
88                   RD<='0';
89                   case counter is
90                      when 3 =>
91                         if (data_enabled='0') then
92                            data1(127 downto 112)<=DATA_IN_RAM;
93                         else
94                            data2(127 downto 112)<=DATA_IN_RAM;
95                         end if;
96                      when 4 =>
97                         if (data_enabled='0') then
98                            data1(111 downto 96)<=DATA_IN_RAM;
99                         else
100                           data2(111 downto 96)<=DATA_IN_RAM;
101                        end if;
102                     when 5 =>
103                        if (data_enabled='0') then
104                           data1(95 downto 80)<=DATA_IN_RAM;
105                        else
106                           data2(95 downto 80)<=DATA_IN_RAM;
107                        end if;
108                     when 6 =>
109                        if (data_enabled='0') then
110                           data1(79 downto 64)<=DATA_IN_RAM;
111                        else
```

```
112                              data2(79 downto 64)<=DATA_IN_RAM;
113                           end if;
114                        when 7 =>
115                           if (data_enabled='0') then
116                              data1(63 downto 48)<=DATA_IN_RAM;
117                           else
118                              data2(63 downto 48)<=DATA_IN_RAM;
119                           end if;
120                        when 8 =>
121                           if (data_enabled='0') then
122                              data1(47 downto 32)<=DATA_IN_RAM;
123                           else
124                              data2(47 downto 32)<=DATA_IN_RAM;
125                           end if;
126                        when 9 =>
127                           if (data_enabled='0') then
128                              data1(31 downto 16)<=DATA_IN_RAM;
129                           else
130                              data2(31 downto 16)<=DATA_IN_RAM;
131                           end if;
132                        when 10 =>
133                           ESTAT<=ESPERA_NOVA_LECTURA;
134                           if (data_enabled='0') then
135                              data1(15 downto 0)<=DATA_IN_RAM;
136                           else
137                              data2(15 downto 0)<=DATA_IN_RAM;
138                           end if;
139                        when 15 =>
140                           counter<=0;
141                        when others =>
142                           ESTAT <= IMPORTACIO_DADES;
143                     end case;
144                     if (counter<15) then
145                        counter<=counter+1;
146                     end if;
147                  ----------------------------------- ESPERA PER A UNA NOVA LECTURA
    (s'entra amb counter=11)
148                  when ESPERA_NOVA_LECTURA =>
149                     case counter is
150                        when 12 =>
151                           if (H_CNT/="1100001110") then --augmentar adreça si no és
    l'últim pixel --782
152                              adress_out<=adress_out+8;
153                           end if;
154                        when 13 =>
155                           if (H_CNT<="1011111111") then -- Enviar senyal mentre no
    s'arribi al final de línia --767
156                              RD<='1';
157                           else
158                              RD<='0';
159                           end if;
160                        when 14 =>
161                           if (H_CNT<="1100000111") then --Enviar les dades al
    descodificaor mentre no sigui final de linia -- 775
162                              ESTAT<=IMPORTACIO_DADES;
163                              if (data_enabled='0') then
164                                 CODED_DATA<=data1;
```

```
165                                 data_enabled<='1';
166                              else
167                                 CODED_DATA<=data2;
168                                 data_enabled<='0';
169                              end if;
170                           else -- Últim pixel
171                              CODED_DATA<=(others =>'0');
172                              if(V_CNT="1000000010") then --És la última línia --514
173                                 ESTAT<=INICI;
174                              else -- Si no és la última línia, s'espera a que comenci
    la següent
175                                 ESTAT<=ESPERA_HCNT;
176                              end if;
177                           end if;
178                        when others => --counter=11
179                           ESTAT<=ESPERA_NOVA_LECTURA;
180                     end case;
181                     counter<=counter+1;
182                  ---------------------------------- ESPERA COMENÇAMENT D'UNA NOVA LÍNIA
183                  when ESPERA_HCNT =>
184                     counter<=15;
185                     data_enabled<='0';
186                     preescaler<='0';
187                     if (H_CNT="0010000111") then --HCNT=135 -> començar a llegir la ram
188                        RD<='1'; --S'envia senyal de lectura de RAM
189                        if (preescaler='0') then --Espera d'un clock per passar a llegir
    les dades
190                           preescaler<='1';
191                        else
192                           preescaler<='0';
193                           ESTAT<=IMPORTACIO_DADES;
194                        end if;
195                     else
196                        ESTAT<=ESPERA_HCNT;
197                     end if;
198                  ------------------------------------
199               end case;
200            end if;
201         end if;
202      end process;
203   end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    13:08:55 02/15/2016
5   -- Module Name:    DATA_DECODER - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity DATA_DECODER is
15     port(CLK25: in std_logic;
16          RST: in std_logic;
17          AV: in std_logic;
18          CODED_DATA: in std_logic_vector(127 downto 0);
19          RGB: out std_logic_vector(7 downto 0));
20  end DATA_DECODER;
21
22  architecture Behavioral of DATA_DECODER is
23     signal R4: std_logic_vector (3 downto 0);
24     signal RED: std_logic_vector (2 downto 0);
25     signal G4: std_logic_vector (3 downto 0);
26     signal GREEN: std_logic_vector (2 downto 0);
27     signal B4: std_logic_vector (3 downto 0);
28     signal BLUE: std_logic_vector (1 downto 0);
29     signal pixel_counter: std_logic_vector(2 downto 0);
30     signal I1: std_logic_vector(7 downto 0);
31     signal RGB444: std_logic_vector(15 downto 0);
32     signal DATA_TO_DECODE: std_logic_vector (127 downto 0);
33
34     constant I0: std_logic_vector(7 downto 0):="00000000";
35
36  begin
37
38     process(CLK25) --PROCÉS PRINCIPAL -- COMPTADOR
39     begin
40        if (CLK25='1' and CLK25'event) then
41           if (RST='1') then
42              pixel_counter<="111";
43           else
44              if (pixel_counter<"111") then
45                 pixel_counter<=pixel_counter+1;
46              else
47                 pixel_counter<="000";
48                 DATA_TO_DECODE<=CODED_DATA; --latch
49              end if;
50           end if;
51        end if;
52     end process;
53
54
55     ------------------------------------------------
56     --MULTIPLEXOR 2x1
57     with AV select
```

```
58          RGB <= I0 when '0',
59                 I1 when others;
60
61      ------------------------------------------------
62      -- Selecció dels bytes corresponent
63      with pixel_counter select
64          RGB444 <= DATA_TO_DECODE(127 downto 112) when "000",
65                    DATA_TO_DECODE(111 downto 96) when "001",
66                    DATA_TO_DECODE(95 downto 80) when "010",
67                    DATA_TO_DECODE(79 downto 64) when "011",
68                    DATA_TO_DECODE(63 downto 48) when "100",
69                    DATA_TO_DECODE(47 downto 32) when "101",
70                    DATA_TO_DECODE(31 downto 16) when "110",
71                    DATA_TO_DECODE(15 downto 0) when others;
72
73      B4<=RGB444(11 downto 8);
74      G4<=RGB444(7 downto 4);
75      R4<=RGB444(3 downto 0);
76
77      ------------------------------------------------
78      --Decodificador de RGB444 a RGB-8bits
79      I1 <=RED & GREEN & BLUE;
80
81      with R4 select
82          RED <= "000" when "0000",
83                 "000" when "0001",
84                 "001" when "0010",
85                 "001" when "0011",
86                 "010" when "0100",
87                 "010" when "0101",
88                 "011" when "0110",
89                 "011" when "0111",
90                 "100" when "1000",
91                 "100" when "1001",
92                 "101" when "1010",
93                 "101" when "1011",
94                 "110" when "1100",
95                 "110" when "1101",
96                 "111" when "1110",
97                 "111" when others;
98
99      with G4 select
100         GREEN <= "000" when "0000",
101                  "000" when "0001",
102                  "001" when "0010",
103                  "001" when "0011",
104                  "010" when "0100",
105                  "010" when "0101",
106                  "011" when "0110",
107                  "011" when "0111",
108                  "100" when "1000",
109                  "100" when "1001",
110                  "101" when "1010",
111                  "101" when "1011",
112                  "110" when "1100",
113                  "110" when "1101",
114                  "111" when "1110",
```

```
115                  "111" when others;
116
117      with B4 select
118         BLUE <= "00" when "0000",
119                 "00" when "0001",
120                 "00" when "0010",
121                 "00" when "0011",
122                 "01" when "0100",
123                 "01" when "0101",
124                 "01" when "0110",
125                 "01" when "0111",
126                 "10" when "1000",
127                 "10" when "1001",
128                 "10" when "1010",
129                 "10" when "1011",
130                 "11" when "1100",
131                 "11" when "1101",
132                 "11" when "1110",
133                 "11" when others;
134
135   end Behavioral;
```

```
1   ----------------------------------------------------------------------------------
2   -- Engineer:  Xavier Picas Pagerols
3   --
4   -- Create Date:    16:44:53 03/14/2016
5   -- Module Name:    PROCESSING_IMAGE - Behavioral
6   -- Project Name:   TFG - PLATAFORMA PER A UNA CÀMERA INTEL·LIGENT
7   --
8   -- Per a Nexys2 board
9   ----------------------------------------------------------------------------------
10  library IEEE;
11  use IEEE.STD_LOGIC_1164.ALL;
12  use IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14  entity PROCESSING_IMAGE is
15     port(ENABLE: in std_logic;
16          CLK50: in std_logic;
17          RST: in std_logic;
18          INTERRUPTORS: in std_logic_vector(7 downto 0);
19          IMATGE: out std_logic_vector(2 downto 0);
20          READ_EN: out std_logic;
21          READ_PROC: out std_logic;
22          WRITE_PROC: out std_logic;
23          DATA_RAM: inout std_logic_vector(15 downto 0);
24          ADRESS_RAM: out std_logic_vector(22 downto 0));
25  end PROCESSING_IMAGE;
26
27  architecture Behavioral of PROCESSING_IMAGE is
28     type TIPUS is (RESET, ESPERA_ENABLE, PRINCIPAL, ESCOMBRAT_MASCARA, EROSIO_DILATACIO
    , ESPERA_WRITE_PROC, READ_MASK, ESPERA_READ_PROC, EXTRACCIO_CONTORN,
    ESPERA_READ_CONTORN, ESPERA_WRITE_CONTORN, BN_GRISOS, ESPERA_READ_BNGRISOS,
    ESPERA_WRITE_BNGRISOS);
29     type TIPUS_ENABLE is (S0, S1, S2);
30     type TIPUS_FILTRE is (EROSIO, DILATACIO, OBERTURA, TANCAMENT, OBE_TAN,
    CONTORN_1EROSIO, CONTORN_OBE_TAN, B_N, GRISOS, COLOR);
31     type TIPUS_ALGORISME is (EROSIO, DILATACIO);
32     type TIPUS_COPY is (RD, WT);
33
34     signal ESTAT:TIPUS;
35     signal ESTAT_ENABLE:TIPUS_ENABLE;
36     signal FILTRE: TIPUS_FILTRE;
37     signal ALGORISME: TIPUS_ALGORISME;
38     signal FASE_COPY:TIPUS_COPY;
39
40     signal DATA_IN: std_logic_vector (15 downto 0);
41     signal DATA_OUT, DATA_OUT_AUX, DATA_OUT_BN, DATA_OUT_BNAUX: std_logic_vector (15
    downto 0);
42     signal OE_inout: std_logic; --OUTPUT ENABLE (inout)
43     signal BN: std_logic; --per seleccionar DATA_OUT
44     signal mask1, mask2, mask3, mask4, mask6, mask7, mask8, mask9, centre:
    std_logic_vector(22 downto 0);--Adreces per a dues imatges
45     signal DATA_mask1, DATA_mask2, DATA_mask3, DATA_mask4, DATA_mask6, DATA_mask7,
    DATA_mask8, DATA_mask9, DATA_centre: std_logic_vector(15 downto 0);
46     signal preescaler: integer range 0 to 5;
47     signal row_counter: integer range 0 to 480;
48     signal column_counter: integer range 0 to 640;
49     signal MASK: integer range 1 to 9; --nº mask
50     signal en: std_logic; --Per la màquina d'estats de ENABLE
```

```
51      signal FASE_CONTORN: integer range 1 to 3; -- 1: LECTURA IMATGE ORIGINAL // 2:
    LECTURA IMATGE EROSIONADA // 3: ESCRIPTURA CONTORN
52      signal pixel_original, pixel_erosio, COPY_DATA: std_logic_vector(15 downto 0);
53      signal pas: integer range 1 to 8;   -- passos a realitzar en els diferents processos
54      signal R4,G4,B4,RED,GREEN,BLUE: std_logic_vector (3 downto 0);
55      signal counter_contorn: integer range 0 to 307200; --nº pixels a comparar
56
57   begin
58      process(CLK50)
59      begin
60         if (CLK50='1' and CLK50'event) then
61            if (RST='1') then
62               ESTAT<=RESET;
63               en<='0';
64            else
65            ------------------------------------------------MÀQUINA D'ESTATS ENABLE
66               case ESTAT_ENABLE is
67                  when S0 =>
68                     en<='0';
69                     if (ENABLE='1') then
70                        ESTAT_ENABLE<=S1;
71                     else
72                        ESTAT_ENABLE<=S0;
73                     end if;
74                  when S1 =>
75                     en<='1';
76                     ESTAT_ENABLE<=S2;
77                  when S2 =>
78                     en<='0';
79                     if (ENABLE='0') then
80                        ESTAT_ENABLE<=S0;
81                     else
82                        ESTAT_ENABLE<=S2;
83                     end if;
84               end case;
85            ------------------------------------------------------------------------
    MÀQUINA D'ESTATS PRINCIPAL
86               case ESTAT is
87               ---------------------------- RESET
88                  when RESET =>
89                     mask1<=(others=>'0');
90                     mask2<=(others=>'0');
91                     mask3<=(others=>'0');
92                     mask4<=(others=>'0');
93                     mask6<=(others=>'0');
94                     mask7<=(others=>'0');
95                     mask8<=(others=>'0');
96                     mask9<=(others=>'0');
97                     centre<=(others=>'0');
98                     IMATGE<="000";
99                     OE_inout<='0';
100                    preescaler<=0;
101                    row_counter<=0;
102                    column_counter<=0;
103                    READ_EN<='0';
104                    FASE_CONTORN<=1;
105                    ESTAT<=ESPERA_ENABLE;
```

```
106                     pas<=1;
107                     FASE_COPY<=RD;
108                     BN<='0';
109                     counter_contorn<=0;
110                ---------------------------- ESPERA SENYAL D'ENABLE PER COMENÇAR A FER EL
    PROCESSAMENT
111                  when ESPERA_ENABLE =>
112                     if (en='1') then
113                        READ_EN<='0';
114                        case INTERRUPTORS is
115                           when "00000000" => -- COLOR
116                              FILTRE<=COLOR;
117                              ESTAT<=PRINCIPAL;
118                           when "00000001" => -- BN
119                              FILTRE<=B_N;
120                              ESTAT<=PRINCIPAL;
121                           when "00000011" => -- GRISOS
122                              FILTRE<=GRISOS;
123                              ESTAT<=PRINCIPAL;
124                           when "00000010" => -- EROSIO
125                              FILTRE<=EROSIO;
126                              ESTAT<=PRINCIPAL;
127                           when "00000100" => -- DILATACIO
128                              FILTRE<=DILATACIO;
129                              ESTAT<=PRINCIPAL;
130                           when "00001000" => -- OPEN
131                              FILTRE<=OBERTURA;
132                              ESTAT<=PRINCIPAL;
133                           when "00010000" => -- CLOSE
134                              FILTRE<=TANCAMENT;
135                              ESTAT<=PRINCIPAL;
136                           when "00100000" => -- OPENCLOSE
137                              FILTRE<=OBE_TAN;
138                              ESTAT<=PRINCIPAL;
139                           when "01000000" => -- CONTORN_1EROSIO
140                              FILTRE<=CONTORN_1EROSIO;
141                              ESTAT<=PRINCIPAL;
142                           when "10000000" => -- CONTORN_OPENCLOSE
143                              FILTRE<=CONTORN_OBE_TAN;
144                              ESTAT<=PRINCIPAL;
145                           when others =>
146                              ESTAT<=ESPERA_ENABLE;
147                        end case;
148                     else
149                        ESTAT<=ESPERA_ENABLE;
150                     end if;
151                ---------------------------- ESTAT PRINCIPAL
152                  when PRINCIPAL =>
153                     case FILTRE is
154                        when COLOR => --No cal processar la imatge
155                           IMATGE <= "000"; --Llegir de la primera imatge
156                           ESTAT<=ESPERA_ENABLE;
157                           READ_EN<='1';
158                        when B_N =>
159                           case pas is
160                              when 1 => -- BN a la 2
161                                 centre<=(others=>'0');
```

```
162                                 ESTAT<=BN_GRISOS;
163                                 BN<='1';
164                              when others => --fi
165                                 pas<=1;
166                                 BN<='0';
167                                 ESTAT<=ESPERA_ENABLE;
168                                 READ_EN<='1';
169                                 IMATGE <= "001";
170                           end case;
171                        when GRISOS =>
172                           case pas is
173                              when 1 => --passar a GRISOS a la 2
174                                 centre<=(others=>'0');
175                                 ESTAT<=BN_GRISOS;
176                                 BN<='1';
177                              when others => --fi
178                                 pas<=1;
179                                 BN<='0';
180                                 ESTAT<=ESPERA_ENABLE;
181                                 READ_EN<='1';
182                                 IMATGE <= "001";
183                           end case;
184                        when EROSIO =>
185                           case pas is
186                              when 1 => --passar a BN a la 1
187                                 centre<=(others=>'0');
188                                 ESTAT<=BN_GRISOS;
189                                 BN<='1';
190                              when 2 => -- erosionar la imatge 1
191                                 BN<='0';
192                                 centre<="00001001011000000000000"; --307200
193                                 ESTAT<=ESCOMBRAT_MASCARA;
194                                 ALGORISME<=EROSIO;
195                              when others => --fi
196                                 pas<=1;
197                                 ESTAT<=ESPERA_ENABLE;
198                                 READ_EN<='1';
199                                 IMATGE<="010";
200                           end case;
201                        when DILATACIO =>
202                           case pas is
203                              when 1 => --passar a BN a la 1
204                                 centre<=(others=>'0');
205                                 ESTAT<=BN_GRISOS;
206                                 BN<='1';
207                              when 2 => -- dilatar la imatge 1
208                                 BN<='0';
209                                 centre<="00001001011000000000000"; --307200
210                                 ESTAT<=ESCOMBRAT_MASCARA;
211                                 ALGORISME<=DILATACIO;
212                              when others => --fi
213                                 pas<=1;
214                                 ESTAT<=ESPERA_ENABLE;
215                                 READ_EN<='1';
216                                 IMATGE<="010";
217                           end case;
218                        when OBERTURA =>
```

```
219                         case pas is
220                            when 1 => --passar a BN a la 1
221                               centre<=(others=>'0');
222                               ESTAT<=BN_GRISOS;
223                               BN<='1';
224                            when 2 => -- erosionar la imatge 1
225                               BN<='0';
226                               centre<="00001001011000000000000"; --307200
227                               ESTAT<=ESCOMBRAT_MASCARA;
228                               ALGORISME<=EROSIO;
229                            when 3 => -- dilatar la imatge 2
230                               centre<="00010010110000000000000"; --614400
231                               ESTAT<=ESCOMBRAT_MASCARA;
232                               ALGORISME<=DILATACIO;
233                            when others => --fi
234                               pas<=1;
235                               ESTAT<=ESPERA_ENABLE;
236                               READ_EN<='1';
237                               IMATGE<="011";
238                         end case;
239                      when TANCAMENT =>
240                         case pas is
241                            when 1 => --passar a BN a la 1
242                               centre<=(others=>'0');
243                               ESTAT<=BN_GRISOS;
244                               BN<='1';
245                            when 2 => -- dilatar la imatge 1
246                               BN<='0';
247                               centre<="00001001011000000000000"; --307200
248                               ESTAT<=ESCOMBRAT_MASCARA;
249                               ALGORISME<=DILATACIO;
250                            when 3 => -- erosionar la imatge 2
251                               centre<="00010010110000000000000"; --614400
252                               ESTAT<=ESCOMBRAT_MASCARA;
253                               ALGORISME<=EROSIO;
254                            when others => --fi
255                               pas<=1;
256                               ESTAT<=ESPERA_ENABLE;
257                               READ_EN<='1';
258                               IMATGE<="011";
259                         end case;
260                      when OBE_TAN=>
261                         case pas is
262                            when 1 => --passar a BN a la 1
263                               centre<=(others=>'0');
264                               ESTAT<=BN_GRISOS;
265                               BN<='1';
266                            when 2 => -- erosionar la imatge 1
267                               BN<='0';
268                               centre<="00001001011000000000000"; --307200
269                               ESTAT<=ESCOMBRAT_MASCARA;
270                               ALGORISME<=EROSIO;
271                            when 3 => -- dilatar la imatge 2
272                               centre<="00010010110000000000000"; --614400
273                               ESTAT<=ESCOMBRAT_MASCARA;
274                               ALGORISME<=DILATACIO;
275                            when 4 => -- dilatar la imatge 3
```

```
276                                 centre<="000111000010000000000000";--921600
277                                 ESTAT<=ESCOMBRAT_MASCARA;
278                                 ALGORISME<=DILATACIO;
279                              when 5 => --erosionar la imatge 4
280                                 centre<="001001011000000000000000";--1228800
281                                 ESTAT<=ESCOMBRAT_MASCARA;
282                                 ALGORISME<=EROSIO;
283                              when others => --fi
284                                 pas<=1;
285                                 ESTAT<=ESPERA_ENABLE;
286                                 READ_EN<='1';
287                                 IMATGE<="101";
288                           end case;
289                        when CONTORN_1EROSIO=>
290                           case pas is
291                              when 1 => --passar a BN a la 1
292                                 centre<=(others=>'0');
293                                 ESTAT<=BN_GRISOS;
294                                 BN<='1';
295                              when 2 => -- erosionar la imatge 1
296                                 BN<='0';
297                                 centre<="000010010110000000000000"; --307200
298                                 ESTAT<=ESCOMBRAT_MASCARA;
299                                 ALGORISME<=EROSIO;
300                              when 3 => --extraccio contorn
301                                 counter_contorn<=0;
302                                 centre<="000010010110000000000000"; --307200 inici
    imatge original
303                                 ESTAT<=EXTRACCIO_CONTORN;
304                              when others => --fi
305                                 pas<=1;
306                                 ESTAT<=ESPERA_ENABLE;
307                                 READ_EN<='1';
308                                 IMATGE<="010";
309                           end case;
310                        when CONTORN_OBE_TAN=>
311                           case pas is
312                              when 1 => --passar a BN a la 1
313                                 centre<=(others=>'0');
314                                 ESTAT<=BN_GRISOS;
315                                 BN<='1';
316                              when 2 => -- erosionar la imatge 1
317                                 BN<='0';
318                                 centre<="000010010110000000000000"; --307200
319                                 ESTAT<=ESCOMBRAT_MASCARA;
320                                 ALGORISME<=EROSIO;
321                              when 3 => -- dilatar la imatge 2
322                                 centre<="000100101100000000000000"; --614400
323                                 ESTAT<=ESCOMBRAT_MASCARA;
324                                 ALGORISME<=DILATACIO;
325                              when 4 => -- dilatar la imatge 3
326                                 centre<="000111000010000000000000";--921600
327                                 ESTAT<=ESCOMBRAT_MASCARA;
328                                 ALGORISME<=DILATACIO;
329                              when 5 => --erosionar la imatge 4
330                                 centre<="001001011000000000000000";--1228800
331                                 ESTAT<=ESCOMBRAT_MASCARA;
```

```
332                                 ALGORISME<=EROSIO;
333                              when 6 => -- erosionar la imatge 5
334                                 BN<='0';
335                                 centre<="00101110111000000000000";--1536000
336                                 ESTAT<=ESCOMBRAT_MASCARA;
337                                 ALGORISME<=EROSIO;
338                              when 7 => --extraccio contorn
339                                 counter_contorn<=0;
340                                 centre<="00101110111000000000000";--1536000 inici
    imatge original
341                                 ESTAT<=EXTRACCIO_CONTORN;
342                              when others => --fi
343                                 pas<=1;
344                                 ESTAT<=ESPERA_ENABLE;
345                                 READ_EN<='1';
346                                 IMATGE<="110";
347                           end case;
348                     end case;
349                ---------------------------- ESCOMBRAT PER TOTA LA IMATGE PER FER LA
    EROSIÓ/DILATACIÓ
350                   when ESCOMBRAT_MASCARA =>
351                      if (column_counter=0 OR column_counter=639 OR row_counter=0 OR
    row_counter=479) then --marges de la imatge
352                         DATA_OUT_AUX<=X"FFFF";
353                         WRITE_PROC<='1';
354                         ADRESS_RAM<=centre+307200; -- S'escriu a la seguent imatge
355                         OE_inout<='1';
356                         ESTAT<=ESPERA_WRITE_PROC;
357                      else
358                         ESTAT<=READ_MASK;
359                         mask1<=centre-641;
360                         mask2<=centre-640;
361                         mask3<=centre-639;
362                         mask4<=centre-1;
363                         mask6<=centre+1;
364                         mask7<=centre+639;
365                         mask8<=centre+640;
366                         mask9<=centre+641;
367                      end if;
368
369                      if (column_counter<639) then
370                         column_counter<=column_counter+1;
371                      else -- ultim pixel
372                         column_counter<=0;
373                         if (row_counter<479) then
374                            row_counter<=row_counter+1;
375                         else --ultima linia
376                            ESTAT<=PRINCIPAL;
377                            pas<=pas+1;
378                            WRITE_PROC<='0';
379                            OE_inout<='0';
380                            preescaler<=0;
381                            row_counter<=0;
382                            column_counter<=0;
383                         end if;
384                      end if;
385
```

```
386                ---------------------------- LECTURA DELS NOU PIXELS DE LA MÀSCARA
387                   when READ_MASK =>
388                      case MASK is
389                         when 1 =>
390                            ADRESS_RAM<=mask1;
391                         when 2 =>
392                            ADRESS_RAM<=mask2;
393                         when 3 =>
394                            ADRESS_RAM<=mask3;
395                         when 4 =>
396                            ADRESS_RAM<=mask4;
397                         when 5 =>
398                            ADRESS_RAM<=centre;
399                         when 6 =>
400                            ADRESS_RAM<=mask6;
401                         when 7 =>
402                            ADRESS_RAM<=mask7;
403                         when 8 =>
404                            ADRESS_RAM<=mask8;
405                         when 9 =>
406                            ADRESS_RAM<=mask9;
407                      end case;
408                      READ_PROC<='1';
409                      ESTAT<=ESPERA_READ_PROC;
410                ---------------------------- ESPERA QUE LA RAM ENVII LES DADES
411                   when ESPERA_READ_PROC =>
412                      READ_PROC<='0';
413                      if (preescaler<5) then --Espera de 80ns (min 70 ns)
414                         preescaler<=preescaler+1;
415                      else
416                         preescaler<=0;
417                         case MASK is
418                            when 1 =>
419                               DATA_mask1<=DATA_IN;
420                               ESTAT<=READ_MASK;
421                               MASK<=MASK+1;
422                            when 2 =>
423                               DATA_mask2<=DATA_IN;
424                               ESTAT<=READ_MASK;
425                               MASK<=MASK+1;
426                            when 3 =>
427                               DATA_mask3<=DATA_IN;
428                               ESTAT<=READ_MASK;
429                               MASK<=MASK+1;
430                            when 4 =>
431                               DATA_mask4<=DATA_IN;
432                               ESTAT<=READ_MASK;
433                               MASK<=MASK+1;
434                            when 5 =>
435                               DATA_centre<=DATA_IN;
436                               ESTAT<=READ_MASK;
437                               MASK<=MASK+1;
438                            when 6 =>
439                               DATA_mask6<=DATA_IN;
440                               ESTAT<=READ_MASK;
441                               MASK<=MASK+1;
442                            when 7 =>
```

```
443                              DATA_mask7<=DATA_IN;
444                              ESTAT<=READ_MASK;
445                              MASK<=MASK+1;
446                           when 8 =>
447                              DATA_mask8<=DATA_IN;
448                              ESTAT<=READ_MASK;
449                              MASK<=MASK+1;
450                           when 9 =>
451                              DATA_mask9<=DATA_IN;
452                              ESTAT<=EROSIO_DILATACIO;
453                              MASK<=1;
454                        end case;
455                     end if;
456                  ---------------------------- FILTRE D'EROSIÓ/DILATACIÓ
457                     when EROSIO_DILATACIO =>
458                        if (ALGORISME=EROSIO) then --EROSIO
459                           if (DATA_mask1=X"0000" AND DATA_mask2=X"0000" AND DATA_mask3=X
       "0000" AND DATA_mask4=X"0000" AND DATA_centre=X"0000" AND DATA_mask6=X"0000" AND
       DATA_mask7=X"0000" AND DATA_mask8=X"0000" AND DATA_mask9=X"0000") then -- tots els
       pixels de la màscara negres
460                              DATA_OUT_AUX<=X"0000";
461                           else
462                              DATA_OUT_AUX<=X"FFFF";
463                           end if;
464                        else --DILATACIO
465                           if (DATA_mask1=X"0000" OR DATA_mask2=X"0000" OR DATA_mask3=X
       "0000" OR DATA_mask4=X"0000" OR DATA_centre=X"0000" OR DATA_mask6=X"0000" OR
       DATA_mask7=X"0000" OR DATA_mask8=X"0000" OR DATA_mask9=X"0000") then --  algun pixel
       de la màscara negre
466                              DATA_OUT_AUX<=X"0000";
467                           else
468                              DATA_OUT_AUX<=X"FFFF";
469                           end if;
470                        end if;
471                        WRITE_PROC<='1';
472                        ADRESS_RAM<=centre+307200; --es guarda a la següent imatge
473                        OE_inout<='1';
474                        ESTAT<=ESPERA_WRITE_PROC;
475                  ---------------------------- ESPERA QUE LA RAM ESCRIGUI LES DADES
476                     when ESPERA_WRITE_PROC =>
477                        WRITE_PROC<='0';
478                        if (preescaler<3) then --Espera de 80ns (min 70 ns)
479                           preescaler<=preescaler+1;
480                        else
481                           preescaler<=0;
482                           OE_inout<='0';
483                           ESTAT<=ESCOMBRAT_MASCARA;
484                           centre<=centre+1;
485                        end if;
486                  ---------------------------- TRANSFORMA A BINARI/GRISOS
487                     when BN_GRISOS =>
488                        if (FASE_COPY=RD) then
489                           ADRESS_RAM<=centre;
490                           READ_PROC<='1';
491                           ESTAT<=ESPERA_READ_BNGRISOS;
492                        else
493                           WRITE_PROC<='1';
```

```
494                     DATA_OUT_BNAUX<=COPY_DATA;
495                     OE_inout<='1';
496                     ESTAT<=ESPERA_WRITE_BNGRISOS;
497                     ADRESS_RAM<=centre+307200; --es copia a la seguent
498                  end if;
499               ----------------------------
500                  when ESPERA_READ_BNGRISOS =>
501                     READ_PROC<='0';
502                     if (preescaler<5) then --Espera de 80ns (min 70 ns)
503                        preescaler<=preescaler+1;
504                     else
505                        preescaler<=0;
506                        COPY_DATA<=DATA_IN;
507                        ESTAT<=BN_GRISOS;
508                        FASE_COPY<=WT;
509                     end if;
510               ----------------------------
511                  when ESPERA_WRITE_BNGRISOS =>
512                     WRITE_PROC<='0';
513                     if (preescaler<3) then --Espera de 80ns (min 70 ns)
514                        preescaler<=preescaler+1;
515                     else
516                        preescaler<=0;
517                        OE_inout<='0';
518                        FASE_COPY<=RD;
519                        if (centre<614399) then -- final imatge 1
520                           centre<=centre+1;
521                           ESTAT<=BN_GRISOS;
522                        else
523                           ESTAT<=PRINCIPAL;
524                           pas<=pas+1;
525                        end if;
526                     end if;
527               ---------------------------- EXTRACCIO DEL CONTORN DE L'OBJECTE
528                  when EXTRACCIO_CONTORN =>
529                     case FASE_CONTORN is
530                        when 1 =>  -- informaico imatge original
531                           ADRESS_RAM<=centre;
532                           READ_PROC<='1';
533                           ESTAT<=ESPERA_READ_CONTORN;
534                        when 2 => -- informacio imatge erosionada
535                           ADRESS_RAM<=centre+307200;
536                           READ_PROC<='1';
537                           ESTAT<=ESPERA_READ_CONTORN;
538                        when 3 =>
539                           if ((pixel_original=X"FFFF" AND pixel_erosio=X"FFFF") OR (
    pixel_original=X"0000" AND pixel_erosio=X"0000")) then
540                              DATA_OUT_AUX<=X"FFFF";
541                           else
542                              DATA_OUT_AUX<=X"0000";
543                           end if;
544                           ADRESS_RAM<=centre+307200; --es guarda sobre la imatge
    erosionada
545                           OE_inout<='1';
546                           WRITE_PROC<='1';
547                           ESTAT<=ESPERA_WRITE_CONTORN;
548                     end case;
```

```
549               ---------------------------- LECTURA DELS PIXELS PER EXTREURE ELS CONTORNS
550                  when ESPERA_READ_CONTORN =>
551                     READ_PROC<='0';
552                     if (preescaler<5) then --Espera de 80ns (min 70 ns)
553                        preescaler<=preescaler+1;
554                     else
555                        preescaler<=0;
556                        case FASE_CONTORN is
557                           when 1 =>
558                              pixel_original<=DATA_IN;
559                              ESTAT<=EXTRACCIO_CONTORN;
560                              FASE_CONTORN<=FASE_CONTORN+1;
561                           when others =>
562                              pixel_erosio<=DATA_IN;
563                              ESTAT<=EXTRACCIO_CONTORN;
564                              FASE_CONTORN<=FASE_CONTORN+1;
565                        end case;
566                     end if;
567               ---------------------------- ESCRIPTURA DEL CONTORN
568                  when ESPERA_WRITE_CONTORN =>
569                     WRITE_PROC<='0';
570                     FASE_CONTORN<=1;
571                     if (preescaler<3) then --Espera de 80ns (min 70 ns)
572                        preescaler<=preescaler+1;
573                     else
574                        preescaler<=0;
575                        OE_inout<='0';
576                        if (counter_contorn<307199) then
577                           centre<=centre+1;
578                           counter_contorn<=counter_contorn+1;
579                           ESTAT<=EXTRACCIO_CONTORN;
580                        else
581                           ESTAT<=PRINCIPAL;
582                           pas<=pas+1;
583                        end if;
584                     end if;
585               ----------------------------
586               end case;
587            end if;
588         end if;
589      end process;
590
591
     ------------------------------------------------------------------------------------
     -------
592      -- CONTROL DE DATA_RAM inout
593
594      DATA_IN<=DATA_RAM;--Sempre es llegeix DATA_RAM
595
596      process(OE_inout, DATA_OUT)
597      begin
598         if (OE_inout='1') then
599            DATA_RAM<=DATA_OUT;
600         else
601            DATA_RAM<=(others=>'Z');
602         end if;
603      end process;
```

```
604
605      ----------------------------------------------------------------------
606      --MULTIPLEXOR PER ESCOLLIR DATA_OUT
607
608      with BN select
609         DATA_OUT <= DATA_OUT_BN when '1',
610                     DATA_OUT_AUX when others;
611
612      ----------------------------------------------------------------------
613      -- PROCES DE BINARITZACIÓ DE LA IMATGE BN / ESCALA DE GRISOS
614
615      B4<=DATA_OUT_BNAUX(11 downto 8);
616      G4<=DATA_OUT_BNAUX(7 downto 4);
617      R4<=DATA_OUT_BNAUX(3 downto 0);
618
619      with R4 select
620         RED <= "0000" when "0000",
621                "0000" when "0001",
622                "0000" when "0010",
623                "0000" when "0011",
624                "0000" when "0100",
625                "0000" when "0101",
626                "1111" when "0110",
627                "1111" when "0111",
628                "1111" when "1000",
629                "1111" when "1001",
630                "1111" when "1010",
631                "1111" when "1011",
632                "1111" when "1100",
633                "1111" when "1101",
634                "1111" when "1110",
635                "1111" when others;
636
637      with G4 select
638         GREEN <= "0000" when "0000",
639                  "0000" when "0001",
640                  "0000" when "0010",
641                  "0000" when "0011",
642                  "0000" when "0100",
643                  "0000" when "0101",
644                  "1111" when "0110",
645                  "1111" when "0111",
646                  "1111" when "1000",
647                  "1111" when "1001",
648                  "1111" when "1010",
649                  "1111" when "1011",
650                  "1111" when "1100",
651                  "1111" when "1101",
652                  "1111" when "1110",
653                  "1111" when others;
654
655      with B4 select
656         BLUE <= "0000" when "0000",
657                 "0000" when "0001",
658                 "0000" when "0010",
659                 "0000" when "0011",
660                 "0000" when "0100",
```

```
661                "0000" when "0101",
662                "1111" when "0110",
663                "1111" when "0111",
664                "1111" when "1000",
665                "1111" when "1001",
666                "1111" when "1010",
667                "1111" when "1011",
668                "1111" when "1100",
669                "1111" when "1101",
670                "1111" when "1110",
671                "1111" when others;
672
673      process(RED,GREEN,BLUE)
674      begin
675         if (FILTRE=GRISOS) then --ESCALA DE GRISOS
676            if (RED="0000" AND GREEN="0000" AND BLUE="0000") then
677               DATA_OUT_BN<="0000000000000000"; --negre
678            else
679               if ((RED="0000" AND GREEN="0000") OR (RED="0000" AND BLUE="0000") OR (
    GREEN="0000" AND BLUE="0000")) then
680                  DATA_OUT_BN<="0000010001000100"; --gris fosc
681               else
682                  if (RED="1111" AND GREEN="1111" AND BLUE="1111") then
683                     DATA_OUT_BN<="1111111111111111"; --blanc
684                  else
685                     DATA_OUT_BN<="0000100010001000"; --gris clar
686                  end if;
687               end if;
688            end if;
689         else --BN
690            if ((RED="0000" AND GREEN="0000") OR (RED="0000" AND BLUE="0000") OR (GREEN=
    "0000" AND BLUE="0000")) then
691               DATA_OUT_BN<="0000000000000000"; --negre
692            else
693               DATA_OUT_BN<="1111111111111111"; --blanc
694            end if;
695         end if;
696      end process;
697
698   end Behavioral;
```

```
1   #UCF FILE
2   #Per a Nexys2 board
3   ##############################################
4   #GENERAL
5   net CLK50   loc = B8;
6   net RST     loc = B18;#BTN0
7   net TRIGGER loc = D18;#BTN1
8
9   net INTERRUPTORS(7) loc = R17; #SW7
10  net INTERRUPTORS(6) loc = N17; #SW6
11  net INTERRUPTORS(5) loc = L13; #SW5
12  net INTERRUPTORS(4) loc = L14; #SW4
13  net INTERRUPTORS(3) loc = K17; #SW3
14  net INTERRUPTORS(2) loc = K18; #SW2
15  net INTERRUPTORS(1) loc = H18; #SW1
16  net INTERRUPTORS(0) loc = G18; #SW0
17
18
19
20  ####################################
21  #VGA DRIVER
22  net HS_VGA loc = T4;
23  net VS_VGA loc = U3;
24
25  net RGB(7) loc = R8; #RED2
26  net RGB(6) loc = T8; #RED1
27  net RGB(5) loc = R9; #RED0
28  net RGB(4) loc = P6; #GRN2
29  net RGB(3) loc = P8; #GRN1
30  net RGB(2) loc = N8; #GRN0
31  net RGB(1) loc = U4; #BLUE1
32  net RGB(0) loc = U5; #BLUE0
33
34  ####################################
35  #CAM DRIVER
36  NET PCLK CLOCK_DEDICATED_ROUTE = TRUE;
37
38  net XCLK    loc = M13;   ## PMOD JB
39  net PDN     loc = R18;
40  net HS_CAM  loc = R15;
41  net VS_CAM  loc = T17;
42  net PCLK    loc = P17;
43  net SDA     loc = R16;
44  net SCL     loc = T18;
45
46  net CAM_DATA(7) loc = G15; ## PMOD JC
47  net CAM_DATA(6) loc = J16;
48  net CAM_DATA(5) loc = G13;
49  net CAM_DATA(4) loc = H16;
50  net CAM_DATA(3) loc = H15;
51  net CAM_DATA(2) loc = F14;
52  net CAM_DATA(1) loc = G16;
53  net CAM_DATA(0) loc = J12;
54
55  ####################################
56  #RAM DRIVER
57  net OE       loc = T2;
```

```
  58   net WE        loc = N7;
  59   net ADV       loc = J4;
  60   net CLK_RAM   loc = H5;
  61   net UB        loc = K4;
  62   net LB        loc = K5;
  63   net CE        loc = R6;
  64   net CRE       loc = P7;
  65
  66   net DATA_RAM(15) loc = T1;
  67   net DATA_RAM(14) loc = R3;
  68   net DATA_RAM(13) loc = N4;
  69   net DATA_RAM(12) loc = L2;
  70   net DATA_RAM(11) loc = M6;
  71   net DATA_RAM(10) loc = M3;
  72   net DATA_RAM(9)  loc = L5;
  73   net DATA_RAM(8)  loc = L3;
  74   net DATA_RAM(7)  loc = R2;
  75   net DATA_RAM(6)  loc = P2;
  76   net DATA_RAM(5)  loc = P1;
  77   net DATA_RAM(4)  loc = N5;
  78   net DATA_RAM(3)  loc = M4;
  79   net DATA_RAM(2)  loc = L6;
  80   net DATA_RAM(1)  loc = L4;
  81   net DATA_RAM(0)  loc = L1;
  82
  83   net ADRESS_RAM(22)  loc = K6;
  84   net ADRESS_RAM(21)  loc = D1;
  85   net ADRESS_RAM(20)  loc = K3;
  86   net ADRESS_RAM(19)  loc = D2;
  87   net ADRESS_RAM(18)  loc = C1;
  88   net ADRESS_RAM(17)  loc = C2;
  89   net ADRESS_RAM(16)  loc = E2;
  90   net ADRESS_RAM(15)  loc = M5;
  91   net ADRESS_RAM(14)  loc = E1;
  92   net ADRESS_RAM(13)  loc = F2;
  93   net ADRESS_RAM(12)  loc = G4;
  94   net ADRESS_RAM(11)  loc = G5;
  95   net ADRESS_RAM(10)  loc = G6;
  96   net ADRESS_RAM(9)   loc = G3;
  97   net ADRESS_RAM(8)   loc = F1;
  98   net ADRESS_RAM(7)   loc = H6;
  99   net ADRESS_RAM(6)   loc = H3;
 100   net ADRESS_RAM(5)   loc = J5;
 101   net ADRESS_RAM(4)   loc = H2;
 102   net ADRESS_RAM(3)   loc = H1;
 103   net ADRESS_RAM(2)   loc = H4;
 104   net ADRESS_RAM(1)   loc = J2;
 105   net ADRESS_RAM(0)   loc = J1;
```

ANNEX II. Datasheets

Els datasheets de la càmera OV7675, de la memòria RAM externa i de la placa Nexys2 es troben en el CD degut a la magnitud d'aquests documents.