ADS Lab 5 & project Report

Eva Maria Deltor Cortés and Pau Jordan Oliveras Cejas

# VHDL Code

## Daq top entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: UPC  -- Engineer: Pau Jordan Oliveras Cejas  --  -- Create Date: 12/05/2021 09:43:04 PM  -- Module Name: daq\_top - daq\_top\_b\_arc  -- Description: This module should contain the entire hardware component of the lab5 assignement.  --  -- Dependencies:  --  -- Revision:  -- Revision 0.01 - File Created  --  ----------------------------------------------------------------------------------  -- This component does not contain any design logic, just interconnects the smaller components and presents  -- an interface to wrap within an axi peripheral.  **library** IEEE**;**  **use** IEEE**.**std\_logic\_1164**.ALL;**  **use** IEEE**.**numeric\_std**.all;**  **entity** daq\_top **is**  **Port** **(** -- Clock and negated reset  CLK **:** **in** std\_logic**;**  RSTN **:** **in** std\_logic**;**  -- UI  trigger\_n\_p**,** trigger\_down**,** trigger\_up**:** **in** std\_logic**;**  mode\_indicator **:** **out** std\_logic\_vector **(**3 **downto** 0**);**  y\_scale\_select **:** **out** std\_logic\_vector **(**2 **downto** 0**);**  trigger\_mode **:** **in** std\_logic**;**    -- VGA signals  red**,** green**,** blue **:** **out** std\_logic\_vector **(**3 **downto** 0**);**  hsync**,** vsync **:** **out** std\_logic**;**  -- ADC ports  ncs**,** sclk **:** **out** std\_logic**;**  sdata1**,** sdata2 **:** **in** std\_logic**;**    -- Temperature management  alarm **:** **in** std\_logic**;**  temperature **:** **in** std\_logic\_vector **(**11 **downto** 0**);**  t\_temperature **:** **in** std\_logic\_vector **(**11 **downto** 0**)**  );  end daq\_top;  architecture daq\_top\_b\_arc of daq\_top is  -- Component declarations  component daq\_vga\_controller  Generic (  addr\_width : natural := 12;  data\_width : natural := 12  );  Port (  clk, rst : in std\_logic;  -- VGA output  RGB : out std\_logic\_VECTOR (11 downto 0);  hsync, vsync : out std\_logic;  -- Memory read port  addr : out std\_logic\_vector (addr\_width - 1 downto 0);  data : in std\_logic\_vector (data\_width - 1 downto 0);  -- Trigger level  trigger\_level : in std\_logic\_vector (8 downto 0);      -- Temperature management  alarm : in std\_logic;  temperature : in std\_logic\_vector (11 downto 0);  t\_temperature : in std\_logic\_vector (11 downto 0);  -- Scaling and UI  y\_scale\_select, x\_scale\_select : in std\_logic\_vector (2 downto 0);  polarity : in std\_logic\_vector(1 downto 0);  mode\_indicator : in std\_logic\_vector( 3 downto 0);  frequency\_x100\_bcd : in std\_logic\_vector(27 downto 0)  );  end component;    component sync\_ram\_dualport  Generic (  addr\_width : natural := 12;  data\_width : natural := 12  );  Port(  clk\_in, clk\_out : in std\_logic;  -- Memory write signals  we : in std\_logic;  addr\_in : in std\_logic\_vector (addr\_width - 1 downto 0);  data\_in : in std\_logic\_vector (data\_width - 1 downto 0);  -- Memory read signals  addr\_out : in std\_logic\_vector (addr\_width - 1 downto 0);  data\_out : out std\_logic\_vector (data\_width - 1 downto 0)    );  end component;  component daq\_trigger\_controller  Generic (  addr\_width : natural := 12;  data\_width : natural := 12  );  Port (  clk, rst : in std\_logic;  -- Memory write signals  we : out std\_logic;  addr : out std\_logic\_vector (addr\_width - 1 downto 0);  data : out std\_logic\_vector (data\_width - 1 downto 0);  -- Trigger control buttons inputs and level output  trigger\_up : in std\_logic;  trigger\_down : in std\_logic;  trigger\_n\_p : in std\_logic;  trigger\_level : out std\_logic\_vector (8 downto 0);  mode\_indicator : out std\_logic\_vector (3 downto 0);  trigger\_mode : in std\_logic;  -- Data input port  adc\_data1 : in std\_logic\_vector (data\_width - 1 downto 0);    -- VGA sync port  vsync : in std\_logic;    -- Scaling and UI  y\_scale\_select, x\_scale\_select : out std\_logic\_vector (2 downto 0);  polarity : out std\_logic\_vector(1 downto 0);  -- Frequency measurement.  trigger\_crossing : out std\_logic  );  end component;  component daq\_adc\_controller  Port (  clk, rst : in std\_logic;  sdata1, sdata2 : in std\_logic;  ncs, sclk : out std\_logic;  data1, data2 : out std\_logic\_vector (11 downto 0)  );  end component;  component frequency\_meter  Port ( trigger\_edge : in STD\_LOGIC;  clk : in STD\_LOGIC;  rst : in STD\_LOGIC;  frequency\_x100\_bcd : out std\_logic\_vector(27 downto 0)  );  end component;      -- Signal Declarations  signal rst : std\_logic;    -- interconnects:  -- VGA <-> Memory Unit  signal addr\_out : std\_logic\_vector (11 downto 0);  signal data\_out : std\_logic\_vector (11 downto 0);  -- VGA <-> Trigger Controller  signal y\_scale\_s, x\_scale\_s : std\_logic\_vector (2 downto 0);  signal trigger\_level : std\_logic\_vector (8 downto 0);  signal vsync\_s : std\_logic;  signal polarity : std\_logic\_vector( 1 downto 0);  signal select\_mode\_s : std\_logic\_vector(3 downto 0);  -- Memory Unit <-> Trigger Controller  signal we : std\_logic;  signal addr\_in : std\_logic\_vector (11 downto 0);  signal data\_in : std\_logic\_vector (11 downto 0);  -- ADC controller <-> Trigger Controller  signal data1 : std\_logic\_vector (11 downto 0);  -- Trigger contrller <-> Frequency Meter  signal trigger : std\_logic;  -- Frequency Meter <-> VGA  signal frequency\_x100\_bcd : std\_logic\_vector(27 downto 0);  begin  rst <= NOT RSTN;  vsync <= vsync\_s;  y\_scale\_select(1 downto 0) <= polarity;  y\_scale\_select(2) <= trigger\_mode;  mode\_indicator <= select\_mode\_s;  daq\_vga\_controller\_1 : daq\_vga\_controller  generic map (  addr\_width => 12,  data\_width => 12  )  port Map(  clk => CLK,  rst => rst,  RGB (11 downto 8) => red,  RGB (7 downto 4) => green,  RGB (3 downto 0) => blue,  hsync => hsync,  vsync => vsync\_s,  addr => addr\_out,  data => data\_out,  trigger\_level => trigger\_level,  alarm => alarm,  temperature => temperature,  t\_temperature => t\_temperature,  y\_scale\_select => y\_scale\_s,  x\_scale\_select => x\_scale\_s,  mode\_indicator => select\_mode\_s,  polarity => polarity,  frequency\_x100\_bcd => frequency\_x100\_bcd    );  daq\_memory\_unit\_1 : sync\_ram\_dualport  generic map (  addr\_width => 12,  data\_width => 12  )  port map (  clk\_in => CLK,  clk\_out => CLK,  data\_out => data\_out,  addr\_out => addr\_out,  we => we,  addr\_in => addr\_in,  data\_in => data\_in  );  daq\_trigger\_controller\_1 : daq\_trigger\_controller  generic map (  addr\_width => 12,  data\_width => 12  )  port map (  clk => CLK,  rst => rst,  we => we,  addr => addr\_in,  data => data\_in,  trigger\_level => trigger\_level,  trigger\_up => trigger\_up,  trigger\_down => trigger\_down,  trigger\_n\_p => trigger\_n\_p,  adc\_data1 => data1,  vsync => vsync\_s,  mode\_indicator => select\_mode\_s,  y\_scale\_select => y\_scale\_s,  x\_scale\_select => x\_scale\_s,  polarity => polarity,  trigger\_mode => trigger\_mode,  trigger\_crossing => trigger  );  daq\_adc\_controller\_1 : daq\_adc\_controller  port map (  clk => CLK,  rst => rst,  ncs => ncs,  sclk => sclk,  sdata1 => sdata1,  sdata2 => sdata2,  data1 => data1,  data2 => open  );    frequency\_meter\_1 : frequency\_meter  port map ( trigger\_edge => trigger,  clk => clk,  rst => rst,  frequency\_x100\_bcd => frequency\_x100\_bcd  );    end daq\_top\_b\_arc; |

## Trigger controller entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Engineer: Pau Oliveras, Eva Deltor  -- Create Date: 12/06/2021 03:05:30 PM  -- Module Name: daq\_trigger\_controller - Behavioral  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** daq\_trigger\_controller **is**  **Generic** **(**  addr\_width **:** natural **:=** 11**;** -- Parametrize data dimensions.  data\_width **:** natural **:=** 12  **);**  **Port** **(**  clk**,** rst **:** **in** std\_logic**;**  -- Memory write signals  we **:** **out** std\_logic**;**  addr **:** **out** std\_logic\_vector **(**addr\_width **-** 1 **downto** 0**);**  data **:** **out** std\_logic\_vector **(**data\_width **-** 1 **downto** 0**);**  -- Trigger control buttons inputs and level output  trigger\_up **:** **in** std\_logic**;**  trigger\_down **:** **in** std\_logic**;**  trigger\_n\_p **:** **in** std\_logic**;**  trigger\_level **:** **out** std\_logic\_vector **(**8 **downto** 0**);**  mode\_indicator **:** **out** std\_logic\_vector **(**3 **downto** 0**);**  trigger\_mode **:** **in** std\_logic**;** -- 0 as per lab5 specifications, 1 >= trigger  -- Data input port  adc\_data1 **:** **in** std\_logic\_vector **(**data\_width **-** 1 **downto** 0**);**  -- VGA sync port  vsync **:** **in** std\_logic**;**  -- UI  y\_scale\_select**,** x\_scale\_select **:** **out** std\_logic\_vector **(**2 **downto** 0**);**  polarity **:** **out** std\_logic\_vector**(**1 **downto** 0**);**    -- Frequency measurement.  trigger\_crossing **:** **out** std\_logic  **);**  **end** daq\_trigger\_controller**;**  **architecture** Behavioral **of** daq\_trigger\_controller **is**  **constant** max\_sample\_period\_ticks **:** integer **:=** 843751**;**  -- types  **type** x\_scale\_t **is** **array** **(**7 **downto** 0**)** **of** integer **range** max\_sample\_period\_ticks **-** 1 **downto** 0**;**    -- config  **constant** x\_scales **:** x\_scale\_t **:=** **(**843750**,** 3375**,** 844**,** 423**,** 169**,** 108**,** 81**,** 54**);**  **constant** rst\_val **:** std\_logic **:=** '1'**;**  **constant** max\_signal\_level **:** integer **:=** 2**\*\***9 **-** 1**;**  **constant** initial\_trigger\_level **:** integer **:=** 256**;** --Initial value defined by the lab assigment  **constant** trigger\_button\_chg\_amount **:** integer **:=** 16**;** -- Increase defined by the lab assigment.  **constant** initial\_sample\_period\_ticks **:** integer **:=** 108**;**  **constant** max\_samples **:** integer **:=** 1280**;**  **constant** debounce\_counter\_max **:** integer **:=** 2**\*\***23**-**1 **;** --minimum number of clocks that we need to wait until the next press of the button is dected  **constant** initial\_y\_scale **:** unsigned **:=** "011"**;** -- Initial selected vertical scale.  **constant** initial\_x\_scale **:** unsigned **:=** "011"**;** -- Initial selected horizontal scale.  **constant** hold\_off\_ticks **:** natural **:=** 100**;** -- Hold off period after a trigger condition.  -- components  **component** button\_frontend -- Used for debouncing button inputs.  **Generic** **(**  debounce\_period **:** integer **:=** 2**\*\***24**-**1**;**  continous\_press\_period **:** integer **:=** 2**\***23**-**1  **);**  **Port** **(** btn\_in **:** **in** std\_logic**;**  btn\_out **:** **out** std\_logic**;**  clk**,** rst **:** **in** std\_logic**);**  **end** **component;**  -- button\_sync\_p signals  **signal** t\_up\_pressed**,** t\_down\_pressed**,** t\_np\_pressed **:** std\_logic**;** -- Button press signals.  -- button ui. Different available modes to change parameters.  **type** select\_mode\_t **is** **(**  edge\_select**,**  trigger\_level\_modify**,**  y\_scale\_modify**,**  x\_scale\_modify  **);**  -- trigger\_control\_p signals  **type** trigger\_mode\_t **is** **(**  trig\_rising\_edge**,**  trig\_falling\_edge**,**  trig\_free\_run  **);**    **signal** trigger\_level\_s **:** integer **range** 0 **to** max\_signal\_level**;**  **signal** trigger\_np\_s **:** trigger\_mode\_t**;** -- zero positive edge, 1 negative edge.  **signal** select\_mode **:** integer **range** 0 **to** 3**;**  **signal** y\_scale\_s**,** x\_scale\_s **:** unsigned **(** 2 **downto** 0**);**  -- trigger\_p signals  **signal** vsync\_edge **:** std\_logic**;**  **signal** trigger**,** trigger\_exact **:** std\_logic**;** -- Indicates that the trigger condition has been met.  **signal** **last\_value** **:** integer **range** 0 **to** max\_signal\_level**;**  **signal** last\_vsync **:** std\_logic**;**  **signal** signal\_level **:** integer **range** 0 **to** max\_signal\_level**;**  **signal** hold\_off\_counter **:** integer **range** 0 **to** 255**;**    -- memwrite\_p signals  **signal** memwrite\_flag **:** std\_logic**;**  **signal** sample\_period **:** integer **range** 0 **to** max\_sample\_period\_ticks**;**  **signal** period\_counter **:** integer **range** 0 **to** max\_sample\_period\_ticks**;**  **signal** sample\_index **:** integer **range** 0 **to** max\_samples**;** -- its all the positions that the memory array will have to store the values from the adc      **begin**  -- Signal assignements  -- Triger level  trigger\_level **<=** std\_logic\_vector**(to\_unsigned(**trigger\_level\_s**,** trigger\_level'**length));**    -- Signal level  signal\_level **<=** **to\_integer(**unsigned**(**adc\_data1**(**data\_width **-** 1 **downto** data\_width **-** 9**)));**  -- Scales and UI  y\_scale\_select **<=** std\_logic\_vector**(**y\_scale\_s**);**  x\_scale\_select **<=** std\_logic\_vector**(**x\_scale\_s**);**  polarity **<=** "00" **when** trigger\_np\_s **=** trig\_rising\_edge **else**  "01" **when** trigger\_np\_s **=** trig\_falling\_edge **else**  "10" **when** trigger\_np\_s **=** trig\_free\_run **else**  "00"**;**  -- Sample period  sample\_period **<=** x\_scales**(to\_integer(**x\_scale\_s**));**    -- Select mode one-hot indicator  **with** select\_mode **select** mode\_indicator **<=**  "0001" **when** 0**,**  "0010" **when** 1**,**  "0100" **when** 2**,**  "1000" **when** 3**,**  "0000" **when** **others;**  -- Trigger flags  trigger\_crossing **<=** trigger**;**    -- Processes  -- Trigger control process  trigger\_control\_p **:** **process(**clk**,** rst**)**  **begin**  **if** **rising\_edge(**clk**)** **then**  **if** rst **=** rst\_val **then** -- Trigger control parameters set to default values.  trigger\_level\_s **<=** initial\_trigger\_level**;**  trigger\_np\_s **<=** trig\_rising\_edge**;**  x\_scale\_s **<=** initial\_x\_scale**;**  y\_scale\_s **<=** initial\_y\_scale**;**  select\_mode **<=** 0**;**  **else**  **case** select\_mode **is** -- Selection mode.  **when** 0 **=>** -- Trigger polarity control. (positive edge, negative edge)  **if** t\_up\_pressed **=** '1' **then**  **if** trigger\_np\_s **=** trig\_rising\_edge **then**  trigger\_np\_s **<=** trig\_falling\_edge**;**  **else**  trigger\_np\_s **<=** trig\_rising\_edge**;**  **end** **if;**  **elsif** t\_down\_pressed **=** '1' **then**  trigger\_np\_s **<=** trig\_free\_run**;**  **end** **if;**  **when** 1 **=>** -- Trigger level position control. (up, down)  **if** t\_up\_pressed **=** '1' **and** trigger\_level\_s **<=** max\_signal\_level **-** trigger\_button\_chg\_amount **then**  --if we want to increase the trigger level, we  trigger\_level\_s **<=** trigger\_level\_s **+** trigger\_button\_chg\_amount**;**    **elsif** t\_down\_pressed **=** '1' **and** trigger\_level\_s **>=** 0 **+** trigger\_button\_chg\_amount **then**  --if we want to decrease the trigger level  trigger\_level\_s **<=** trigger\_level\_s **-** trigger\_button\_chg\_amount**;**  **end** **if;** -- if no button is pressed we mantain the previous trigger level  **when** 2 **=>** -- Vertical scale adjust.  **if** t\_up\_pressed **=** '1' **and** y\_scale\_s **>** 0 **then**  y\_scale\_s **<=** y\_scale\_s **-** 1**;**  **elsif** t\_down\_pressed **=** '1' **and** y\_scale\_s **<** 7 **then**  y\_scale\_s **<=** y\_scale\_s **+** 1**;**  **end** **if;**    **when** 3 **=>** -- Horiztontal scale adjust.  **if** t\_up\_pressed **=** '1' **and** x\_scale\_s **>** 0 **then**  x\_scale\_s **<=** x\_scale\_s **-** 1**;**  **elsif** t\_down\_pressed **=** '1' **and** x\_scale\_s **<** 7 **then**  x\_scale\_s **<=** x\_scale\_s **+** 1**;**  **end** **if;**  **end** **case;**  -- Select mode change  **if** t\_np\_pressed **=** '1' **then**  **if(**select\_mode **<** 3**)** **then**  select\_mode **<=** select\_mode **+** 1**;**  **else**  select\_mode **<=** 0**;**  **end** **if;**  **end** **if;**  **end** **if;**  **end** **if;**  **end** **process** trigger\_control\_p **;**    -- trigger process  trigger\_p**:** **process(**clk**)**  **begin**  **if** **rising\_edge(**clk**)** **then**  **if** rst **=** rst\_val **then** -- Set trigger flags to initial values.  hold\_off\_counter **<=** 0**;**  trigger\_exact **<=** '0'**;**  trigger **<=** '0'**;**  **else**  -- There are two trigger flags. trigger\_exact and trigger.  -- The former fires only when the significant bits of the signal equal the threshold exactly,  -- as per lab 5 specifications.  -- The latter is more permisive, as it fires when the signal crosses the treshold. This one is used for frequency measurement and  -- if trigger\_mode set to 1 also for signal display. When a trigger flag fires, a hold off period is enforced to avoid repeated trigger conditions for high  -- frequency signals. trigger\_exact flag does not need this, as the trigger flag is only used for display, so only the first time it fires causes an effect.  **if** hold\_off\_counter **=** 0 **then** -- Hold off period has ellapsed.  **if** **(** **(**trigger\_np\_s **=** trig\_rising\_edge**)** **xor** **(**signal\_level **<=** trigger\_level\_s**))** **and** **(** **(**trigger\_np\_s **=** trig\_rising\_edge**)** **xor** **(last\_value** **>** trigger\_level\_s**))** **then**  trigger **<=** '1'**;** -- If trigger condition according to falling or rising edge is true, set trigger flag.  hold\_off\_counter **<=** hold\_off\_ticks**;** -- Start hold off countdown.  **end** **if;**  **else**  trigger **<=** '0'**;**  hold\_off\_counter **<=** hold\_off\_counter **-** 1**;**  **end** **if;**  **if** **(** **(**signal\_level **=** trigger\_level\_s**)** **and** **(** **(**trigger\_np\_s **=** trig\_rising\_edge **and** **last\_value** **<** trigger\_level\_s**)** **or** **(**trigger\_np\_s **=** trig\_falling\_edge **and** **last\_value** **>** trigger\_level\_s**)))** **then**  trigger\_exact **<=** '1'**;** -- If lab 5 trigger condition occurs, set flag.  **else**  trigger\_exact **<=** '0'**;** -- Else keep the flag unset.  **end** **if;**  **last\_value** **<=** signal\_level**;** -- Remember last value to detect edge polarity.  **end** **if;**  **end** **if;**  **end** **process** trigger\_p**;**  --  sync\_p**:** **process(**clk**)**  **begin**  **if** **rising\_edge(**clk**)** **then**  **if** rst **=** rst\_val **then**  last\_vsync **<=** '1'**;**  **else**  **if** **(**vsync **=** '0' **and** last\_vsync **=** '1'**)** **then** -- Detect falling edges on vsync.  -- the data acquisition must start when the vsync signal is at level 0 (assigment)  vsync\_edge **<=** '1'**;**  -- we have defined our internal signal because it is not only that the vsync is a zero, but that there has been a falling edge  **end** **if;**  last\_vsync **<=** vsync**;** -- Record last vsync value for detecting edges.  **if** vsync\_edge **=** '1' **and** **((**trigger **=** '1' **and** trigger\_mode **=** '1'**)** **or** **(**trigger\_exact **=** '1' **and** trigger\_mode **=** '0'**)** **or** trigger\_np\_s **=** trig\_free\_run**)** **then** -- When a falling edge has happened and the trigger condition is met, start the memwrite process.  memwrite\_flag **<=** '1'**;** -- Set the memwriteflag.  vsync\_edge **<=** '0'**;** -- Clear the vsync edge flag.  **else**  memwrite\_flag **<=** '0'**;** -- Clear the memwrite flag.  **end** **if;**  **end** **if;**  **end** **if;**  **end** **process** sync\_p**;**  -- memory write process  memwrite**:** **process(**clk**)**  **begin**  **if** **rising\_edge(**clk**)** **then**  **if** rst **=** rst\_val **or** **(**memwrite\_flag **=** '1' **and** sample\_index **=** max\_samples**)** **then** -- Check that memory write is not taking place before servicing flag.  period\_counter **<=** 0**;**  sample\_index **<=** 0**;**  **elsif** sample\_index **<** max\_samples **then** -- Samples are still to be acquired.  **if** **(**period\_counter **>=** sample\_period **-** 1**)** **then** -- Sampling period has ellapsed.  period\_counter **<=** 0**;** -- Reset sampling period counter.  -- Memory write  data **<=** adc\_data1**;** --we send to the memory data\_in the value that we have in the adc at that moment  addr **<=** std\_logic\_vector**(to\_unsigned(**sample\_index**,** addr'**length));** --we save the value in the position that we want based on the number of the current sample that we are storing  we **<=** '1'**;** --write enable signal of the memory; we write in it the values that we have just defined: data and addr  --Index update  sample\_index **<=** sample\_index **+** 1**;** --we increase the position in the memory for the next sample to be saved    **else** -- Idle while the sampling period ellapses.  we **<=** '0'**;** -- Memory write disabled.  period\_counter **<=** period\_counter **+** 1**;** -- Increment the sampling period counter.  **end** **if;**  **else** -- Memory write finished. Idle until next memwrite flag.  we **<=** '0'**;**  **end** **if;**  **end** **if;**  **end** **process** memwrite**;**  -- component mapping. Button debounce.  bf\_1 **:** button\_frontend  **Generic** **Map** **(**  debounce\_period **=>** 2**\*\***24 **-** 1**,**  continous\_press\_period **=>** 2**\*\***24 **-** 1  **)**  **Port** **Map** **(**  clk **=>** clk**,**  rst **=>** rst**,**  btn\_in **=>** trigger\_up**,**  btn\_out **=>** t\_up\_pressed  **);**  bf\_2 **:** button\_frontend  **Generic** **Map** **(**  debounce\_period **=>** 2**\*\***24 **-** 1**,**  continous\_press\_period **=>** 2**\*\***24 **-**1  **)**  **Port** **Map** **(**  clk **=>** clk**,**  rst **=>** rst**,**  btn\_in **=>** trigger\_down**,**  btn\_out **=>** t\_down\_pressed  **);**  bf\_3 **:** button\_frontend  **Generic** **Map** **(**  debounce\_period **=>** 2**\*\***24 **-** 1**,**  continous\_press\_period **=>** 2**\*\***24 **-**1  **)**  **Port** **Map** **(**  clk **=>** clk**,**  rst **=>** rst**,**  btn\_in **=>** trigger\_n\_p**,**  btn\_out **=>** t\_np\_pressed  **);**  **end** Behavioral**;** |

## Button frontend entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: UPC  -- Engineer: Pau Oliveras  --  -- Create Date: 01/13/2022 04:04:27 PM  -- Design Name:  -- Module Name: button\_frontend - arc  -- Project Name:  -- Target Devices:  -- Tool Versions:  -- Description:  --  -- Dependencies:  --  -- Revision:  -- Revision 0.01 - File Created  -- Additional Comments:  --  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** button\_frontend **is**  **Generic** **(**  debounce\_period **:** integer **:=** 2**\*\***24**-**1**;**  continous\_press\_period **:** integer **:=** 2**\***23**-**1  **);**  **Port** **(** btn\_in **:** **in** std\_logic**;**  btn\_out **:** **out** std\_logic**;**  clk**,** rst **:** **in** std\_logic**);**  **end** button\_frontend**;**  **architecture** arc **of** button\_frontend **is**  **signal** btn\_in\_stable**,** btn**,** btn\_edge**,** btn\_pressed**,** last\_btn **:** std\_logic**;**  **type** button\_state\_t **is** **(** ready**,** pressed **);**  **signal** button\_state **:** button\_state\_t**;**  **constant** rst\_val **:** std\_logic **:=** '1'**;**  **begin**    -- Button input signals syncronization process.  -- The process allows us to detect whether we have a push in the button, and assign the value to our internal signal to work with it in another process  -- We have added a required minimum time so as to detect that a button push has been done in reality and not confuse it with a MISSING\_PARAULA signal.  -- The required minimum time is counted by the debounce\_counter  button\_sync\_p : process(clk)  variable debounce\_counter : integer range 0 to debounce\_period;  begin  if rising\_edge(clk) then -- Read inputs  if rst = rst\_val then  button\_state <= pressed;  debounce\_counter := debounce\_period;  else -- if we do not have a reset we set the inputs to our internal signals  -- Tu avoid metastability:  btn\_in\_stable <= btn\_in;  btn <= btn\_in\_stable;    last\_btn <= btn; -- To detect edges:  --Edge detect  -- it will be an edge if the actual state (high/low) its not the same as the previous one  btn\_edge <= btn and not last\_btn;  if(button\_state = ready and btn\_edge = '1') then  --if we are ready and we have a rising edge on any of the buttons we inidcate with our internal signal  --that the button has been pressed  btn\_out <= '1';  button\_state <= pressed;  debounce\_counter := debounce\_period;  elsif (button\_state = pressed) then --if we have the inidcative that the button has been pressed  --if it hasn't, we act as the button has not been pushed:  if(debounce\_counter > 0) then  btn\_out <= '0';  button\_state <= pressed;  debounce\_counter := debounce\_counter - 1;  -- if it has (we check if there is a button pressed or not):  elsif(btn = '1') then -- we recheck the state of the buttons and assign it to our signals  btn\_out <= '1';  button\_state <= pressed;  debounce\_counter := continous\_press\_period; -- we reset the time counter  else  button\_state <= ready; --otherwise we set that we are ready (meaning waiting) for a push of any button    end if;  else --we would get here when we are in the ready state, therefore the values of the buttons must be zero and we should be waiting for the next button push, then ready state  btn\_out <= '0';  button\_state <= ready;  end if;  end if;  end if;  end process button\_sync\_p;  end arc; |

## ADC Driver entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: ADS//UPC  -- Engineer: Eva Maria Deltor, Pau Jordan Oliveras  --  -- Create Date: 22.11.2021 19:00:52  -- Module Name: AD\_converter\_source - Behavioral  -- Revision:  -- Revision 0.01 - File Created  -- Revision 1.0 - Design working and tested.  -- Revision 1.1 - Sample rate changed from 1.x to 1.0 MS/s with a 108 MHz clock.  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **use** ieee**.**std\_logic\_unsigned**.ALL;**  **entity** daq\_adc\_controller **is**  **Port** **(**  clk**,** rst**:** **in** std\_logic**;** -- Clock @ 108 MHz and reset signal.  sdata1**,** sdata2 **:** **in** std\_logic**;** -- inputs for each channel.  ncs**:** **out** std\_logic**;** -- Negated chip select.  sclk**:** **out** std\_logic**;** -- Serial clock.  data1**,** data2 **:** **out** std\_logic\_vector **(**11 **downto** 0**));** -- Parallel latched data output.  **end** daq\_adc\_controller**;**  **architecture** Behavioral **of** daq\_adc\_controller **is**  -- ADC controller signals.  **signal** clk\_counter**:** integer**;** -- Holds the quantity of clock cycles ellapsed since the start of the conversion.  **signal** presc\_counter**:** integer**;** -- Used as a prescaler for generating the Serial Clock for the ADC.  **signal** converted\_value**:** std\_logic\_vector **(**14 **downto** 0**);** -- Stores the recieved data.  **signal** ncs\_s**,** sclk\_s **:** std\_logic**;** --Signals for Chip select active low and serial clock.  -- signals for controlling shift register  **signal** oe **:** std\_logic**;** -- Latches current shift register parallel output on rising edge.  **signal** ce **:** std\_logic**;** -- Enables shift register clock, shifting the register each clock cycle.    **component** o\_generic\_sr -- Shift register used for storing the incoming serial data bits.  **generic** **(**  Q\_width **:** natural **:=** 15**);**  **port** **(**D **:** **in** std\_logic**;**  CLK**,** CE**,** OE **:** **in** std\_logic**;**  Q **:** **out** std\_logic\_vector **(**q\_width **-** 1 **downto** 0**));**  **end** **component;**    **begin**  process(clk) -- All the time diagram logic is implemented in a single syncronous process.  begin  if clk'event and clk='1' then -- Rising edge.  if rst = '1' then -- Syncronus reset performs the following:  clk\_counter <= 0; -- Conversion counter to 0.  ncs\_s <= '1'; -- Chip select disable.  sclk\_s <= '1'; -- Serial clock to high.  presc\_counter <= 0; -- Prescaler reset.  oe <= '0'; -- Shift register output disable.  ce <= '0';    else -- Free running conversion. Using clk\_counter as the time reference, several actions are performed.  if clk\_counter = 0 then -- Start of conversion. Then:  ncs\_s <= '0'; -- Enable chip select.  elsif (clk\_counter >= 2) and (clk\_counter <= 95) then -- Conversion in progress.  -- Serial clock generation using the prescaler. 13,5 MHz.  if presc\_counter = 0 then -- Period start. Then:  sclk\_s <= '0'; -- Serial clock to low.  ce <= '0'; -- Disable shift register clock after shifting one bit.  elsif presc\_counter = 3 then -- Serial clock mid period. Then:  sclk\_s <= '1'; -- Serial clock to high.  end if;    if presc\_counter = 5 then -- Period end. Reset prescaler counter.  presc\_counter <= 0;  if clk\_counter >= 7 and clk\_counter <= 92 then -- If the period has ended and the actual time is inside the conversionw window:  ce <= '1'; -- Shift in the incoming data bit by enabling the shift register's clock.  end if;  else  presc\_counter <= presc\_counter + 1; -- If period end has not been reached, increment the prescaler counter.  end if;  end if;  if clk\_counter = 92 then -- End of conversion. Then:  oe <= '1'; -- Rising edge. Latch the existing data in the shift register and enable parallel output, as it now holds the converted value.  elsif clk\_counter = 95 then -- Reset signals after hold period.  sclk\_s <= '1'; -- Disable ADC.  ncs\_s <= '1'; -- Disable ADC.  oe <= '0'; -- Return shift register output enable to low.  end if;    if clk\_counter = 107 then -- Idle time end. Restart conversion.  clk\_counter <= 0;  presc\_counter <= 0;  else  clk\_counter <= clk\_counter + 1; -- If the end has not been reached, increment clock cycle counter.  end if;  end if;  end if;  end process;    -- Combinational logic: Component map and signals to output ports.    shift\_register\_1 : o\_generic\_sr -- Shift register instanciation and mapping to signals.  generic map ( Q\_width => 15 ) -- 12 Bits of data and 3 bits of padding.  port map ( D => sdata1,  CLK => clk,  OE => oe,  Q => converted\_value,  CE => ce);    data1 <= converted\_value(11 downto 0); -- Data output. Always holds last converted value. Undefined until at least one conversion is made.  data2 <= (others => '0'); -- Single channel operation. Second output port kept for future use.  ncs <= ncs\_s; -- ADC Control signals to outptut.  sclk <= sclk\_s;  end Behavioral; |

## Shift register entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: UPC  -- Engineer: Eva Deltor, Pau Oliveras  --  -- Create Date: 11/25/2021 01:22:30 PM  -- Design Name:  -- Module Name: o\_generic\_sr - o\_generic\_sr\_arc  -- Revision:  -- Revision 0.01 - File Created  -- Additional Comments:  --  ----------------------------------------------------------------------------------  -- This component shifts incoming serial data into an arbitray size shift register.  -- A rising edge in the output enable signal latches the current shift register contents to the output.  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **entity** o\_generic\_sr **is**  **Generic** **(**  Q\_width **:** natural **:=** 12**);**    **Port** **(** clk**,** CE**,** OE **:** std\_logic**;**  D **:** **in** STD\_LOGIC**;**  Q **:** **out** STD\_LOGIC\_VECTOR **(**Q\_width **-** 1 **downto** 0**));**  **end** o\_generic\_sr**;**  **architecture** o\_generic\_sr\_arc **of** o\_generic\_sr **is**  **signal** OE\_s **:** std\_logic**;**  **begin**  **process** **(**clk**)**  **variable** sr **:** std\_logic\_vector**(**Q\_width **-** 1 **downto** 0**);**  **begin**  **if** **rising\_edge(**clk**)** **then** -- Syncronous process, rising edge.  **if** CE **=** '1' **then** -- If the clock is enabled:  sr **:=** sr**(**Q\_width **-** 2 **downto** 0**)** **&** D**;** -- Shift register to the left.  **end** **if;**  **if** OE **=** '1' **and** OE\_s **=** '0' **then** -- Output enable rising edge. Then:  Q **<=** sr**;** -- Latch shift register contents to the output.  **end** **if;**  OE\_s **<=** OE**;**  **end** **if;**  **end** **process;**  end o\_generic\_sr\_arc; |

## Vga controller entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: UPC  -- Engineer: Eva Deltor, Pau Oliveras  -- Revision:  -- Revision 0.01 - File Created  --  ----------------------------------------------------------------------------------  -- This component does not contain any design logic, just interconnects the smaller components and presents  -- an interface for the vga controller.  **library** ieee**;**  **use** ieee**.**STD\_LOGIC\_1164**.ALL;**  **use** ieee**.**NUMERIC\_STD**.ALL;**  **entity** daq\_vga\_controller **is**  **Generic** **(**  addr\_width **:** natural **:=** 12**;**  data\_width **:** natural **:=** 12  **);**  **Port** **(**  clk**,** rst **:** **in** std\_logic**;**  -- VGA output  RGB **:** **out** std\_logic\_VECTOR **(**11 **downto** 0**);**  hsync**,** vsync **:** **out** std\_logic**;**  -- Memory read port  addr **:** **out** std\_logic\_vector **(**addr\_width **-** 1 **downto** 0**);**  data **:** **in** std\_logic\_vector **(**data\_width **-** 1 **downto** 0**);**  -- Trigger level  trigger\_level **:** **in** std\_logic\_vector **(**8 **downto** 0**);**    -- Temperature managment  alarm **:** **in** STD\_LOGIC**;**  temperature **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  t\_temperature **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  -- UI  y\_scale\_select**,** x\_scale\_select **:** **in** std\_logic\_vector **(**2 **downto** 0**);**  mode\_indicator **:** **in** std\_logic\_vector**(**3 **downto** 0**);**  polarity **:** **in** std\_logic\_vector**(**1 **downto** 0**);**    -- Frequency Measurement  frequency\_x100\_bcd **:** **in** std\_logic\_vector**(**27 **downto** 0**)**  **);**  end daq\_vga\_controller;  architecture beh of daq\_vga\_controller is  component vga\_sync\_gen  generic (  h\_pixels : integer := 800;  h\_sync : integer := 96;  h\_start\_pixel : integer := 144;  h\_end\_pixel : integer := 784;  v\_lines : integer := 521;  v\_sync : integer := 2;  v\_start\_line : integer := 31;  v\_end\_line : integer := 511;  h\_bits : integer := 12;  v\_bits : integer := 12);  Port ( CLK, RST : in STD\_LOGIC;  E : in STD\_LOGIC;  HS : out STD\_LOGIC;  VS : out STD\_LOGIC;  PIXEL\_X : out unsigned(h\_bits - 1 downto 0);  PIXEL\_Y : out unsigned(v\_bits - 1 downto 0);  DISPLAY\_E : out STD\_LOGIC);  end component;  component threshold\_plotter  Port ( PX : in unsigned(11 downto 0);  PY : in unsigned(11 downto 0);  RGB\_in : in STD\_LOGIC\_VECTOR (11 downto 0);  RGB\_out : out STD\_LOGIC\_VECTOR (11 downto 0);  -- Trigger level  trigger\_level : in std\_logic\_vector (8 downto 0);  vertical\_scale : in unsigned (2 downto 0)  );  end component;    component signal\_plotter  Port ( PX : in unsigned(11 downto 0);  PY : in unsigned(11 downto 0);  RGB\_in : in STD\_LOGIC\_VECTOR (11 downto 0);  RGB\_out : out STD\_LOGIC\_VECTOR (11 downto 0);  alarm : in std\_logic;  -- Trigger level  signal\_data : in std\_logic\_vector (11 downto 0);  vertical\_scale : in unsigned (2 downto 0)  );  end component;    component temperature\_plotter  Port ( RGB\_in : in STD\_LOGIC\_VECTOR (11 downto 0);  PX : in unsigned (11 downto 0);  PY : in unsigned (11 downto 0);  alarm : in STD\_LOGIC;  temperature : in STD\_LOGIC\_VECTOR (11 downto 0);  t\_temperature : in STD\_LOGIC\_VECTOR (11 downto 0);  RGB\_out : out STD\_LOGIC\_VECTOR (11 downto 0));  end component;    component frequency\_plotter  Generic (  frequency\_width : natural := 32  );  Port (  clk : std\_logic;  PX : in unsigned(11 downto 0);  PY : in unsigned(11 downto 0);  RGB\_in : in STD\_LOGIC\_VECTOR (11 downto 0);  RGB\_out : out STD\_LOGIC\_VECTOR (11 downto 0);  frequency\_x100\_bcd : in std\_logic\_vector(27 downto 0);  y\_scale\_select, x\_scale\_select : in std\_logic\_vector (2 downto 0);  mode\_indicator : in std\_logic\_vector(3 downto 0);  polarity : std\_logic\_vector(1 downto 0);  alarm : in STD\_LOGIC    );  end component;  -- Signal Declarations  --vga test signals  signal pixel\_presc\_s, disp\_s, VGA\_VS\_s, VGA\_HR\_s : std\_logic;  signal pixel\_x\_s, pixel\_y\_s : unsigned(11 downto 0);  signal RGB\_s : std\_logic\_vector(11 downto 0);  signal line\_counter : integer range 0 to 1023;    -- vga signal  signal signal\_value : unsigned(8 downto 0);    -- constants  constant rst\_val : std\_logic := '1';  constant black : std\_logic\_vector (11 downto 0) := (others => '0');    -- interconnects:  -- plotters RGB interconnect  signal i\_rgb\_1, i\_rgb\_2, i\_rgb\_3 : std\_logic\_vector(11 downto 0);    signal fp\_x : unsigned (11 downto 0);  signal fp\_y : unsigned (11 downto 0);    begin  hsync <= VGA\_HR\_s;  vsync <= VGA\_VS\_s;    addr <= std\_logic\_vector(pixel\_x\_s);    fp\_x <= pixel\_x\_s - 900 when pixel\_x\_s >= 900 else (others=>'1');  fp\_y <= pixel\_y\_s - 850 when pixel\_y\_s >= 850 else (others=>'1');      RGB <= RGB\_s when disp\_s = '1' else (others => '0');    vga\_timing: vga\_sync\_gen  generic map (  h\_pixels => 1688,  h\_sync => 112,  h\_start\_pixel => 360,  h\_end\_pixel => 1640,  v\_lines => 1066,  v\_sync => 3,  v\_start\_line => 41,  v\_end\_line => 1065,  h\_bits => 12,  v\_bits => 12)  port map (  CLK => clk,  RST => rst,  E => '1',  HS => VGA\_HR\_s,  VS => VGA\_VS\_s,  PIXEL\_X => pixel\_x\_s,  PIXEL\_Y => pixel\_y\_s,  DISPLAY\_E => disp\_s);    signal\_plotter\_1 : signal\_plotter  port map (  PX => pixel\_x\_s,  PY => pixel\_y\_s,  RGB\_in => black,  RGB\_out => i\_RGB\_1,  signal\_data => data,  vertical\_scale => unsigned(y\_scale\_select),  alarm => alarm  );    threshold\_plotter\_1 : threshold\_plotter  port map (  PX => pixel\_x\_s,  PY => pixel\_y\_s,  RGB\_in => i\_RGB\_1,  RGB\_out => i\_RGB\_2,  trigger\_level => trigger\_level,  vertical\_scale => unsigned(y\_scale\_select)  );  temperature\_plotter\_1 : temperature\_plotter  port map (  PX => pixel\_x\_s,  PY => pixel\_y\_s,  RGB\_in => i\_RGB\_2,  RGB\_out => i\_RGB\_3,  alarm => alarm,  temperature => temperature,  t\_temperature => t\_temperature  );    frequency\_plotter\_1 : frequency\_plotter  port map (  clk => clk,  PX => fp\_x,  PY => fp\_y,  RGB\_in => i\_RGB\_3,  RGB\_out => RGB\_s,  alarm => alarm,  y\_scale\_select => y\_scale\_select,  x\_scale\_select => x\_scale\_select,  polarity => polarity,  mode\_indicator => mode\_indicator,  frequency\_x100\_bcd => frequency\_x100\_bcd  );  end beh; |

## Vga syncronization signals generator entity

|  |
| --- |
| **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** vga\_sync\_gen **is**  **generic** **(**  h\_pixels **:** integer **:=** 800**;** -- Numero total cicles horitzontals  h\_sync **:** integer **:=** 96**;** -- Amplada en pixels pols horitzontal  h\_start\_pixel **:** integer **:=** 144**;** --a on comenca temps de display  h\_end\_pixel **:** integer **:=** 784**;** -- el primer que no pertany a temps de display  v\_lines **:** integer **:=** 521**;** --Numero de linies total  v\_sync **:** integer **:=** 2**;** --Amplada en linies pols vertical  v\_start\_line **:** integer **:=** 31**;** -- Primera linia que pertany a display  v\_end\_line **:** integer **:=** 511**;** -- Primera linia que no pertany a display  h\_bits **:** integer **:=** 10**;** -- N bits contador horitzontal  v\_bits **:** integer **:=** 10**);** -- N bits contador vertical    **Port** **(** CLK**,** RST **:** **in** STD\_LOGIC**;**  E **:** **in** STD\_LOGIC**;** -- Activar contador pixels  HS **:** **out** STD\_LOGIC**;** -- Sortida pols HS  VS **:** **out** STD\_LOGIC**;** -- Sortida pols VS  PIXEL\_X **:** **out** unsigned**(**h\_bits **-** 1 **downto** 0**);** -- Numero de Pixel X  PIXEL\_Y **:** **out** unsigned**(**v\_bits **-** 1 **downto** 0**);** -- Numero de Pixel Y  DISPLAY\_E **:** **out** STD\_LOGIC**);** -- Actiu si Pixel actual es de display  **end** vga\_sync\_gen**;**  **architecture** vga\_sync\_gen\_arc **of** vga\_sync\_gen **is**  **signal** c\_x **:** integer **range** **(**2**\*\*(**h\_bits**)-**1**)** **downto** 0**;**  **signal** c\_y **:** integer **range** **(**2**\*\*(**v\_bits**)-**1**)** **downto** 0**;**  **signal** HS\_s**,** VS\_s**,** DISPLAY\_E\_s**:** std\_logic**;**  **begin**  HS **<=** HS\_s**;**  VS **<=** VS\_s**;**  PIXEL\_X **<=** **to\_unsigned(**c\_x **-** h\_start\_pixel**,** PIXEL\_X'**length);** -- n pixel X es comptador horitzontal - primer pixel  PIXEL\_Y **<=** **to\_unsigned(**c\_y **-** v\_start\_line**,** PIXEL\_Y'**length);** -- n linia Y es comptador vertical - primera linia  DISPLAY\_E **<=** DISPLAY\_E\_s**;**  **process(**CLK**)**  **begin**  **if(**CLK **=** '1' **and** CLK'**event)** **then** -- CLK  **if(**RST **=** '1'**)** **then** --Comptadors a zero  c\_x **<=** 0**;**  c\_y **<=** 0**;**  HS\_s **<=** '0'**;**  VS\_s **<=** '0'**;**  **elsif** **(**E **=** '1'**)** **then** -- Si es un clk on toca canviar de pixel  **if(**c\_x **>=** **(**h\_pixels**-**1**))** **then** -- Si la linia ha acabat  c\_x **<=** 0**;** -- Es reinicia comptador horitzontal i comprovem el comptador vertical  HS\_s **<=** '0'**;** -- Pols sync horitzontal  **if(**c\_y **>=** **(**v\_lines**-**1**))** **then** -- Si el frame ha acabat  c\_y **<=** 0**;** -- Es reinicia comptador vertical  VS\_s **<=** '0'**;** -- Pols sync vertical  **else**  **if** **(**c\_y **>=** **(**v\_sync **-** 1**))** **then** -- Si ja no es linia de pols vertical, acaba pols.  VS\_s **<=** '1'**;**  **end** **if;**  c\_y **<=** c\_y **+** 1**;** -- Incrementa comptador vertical  **end** **if;**  **else**  **if(**c\_x **>=** **(**h\_sync **-** 1**))** **then** -- Si ja no es linia de pols horitzontal, acaba pols.  HS\_s **<=** '1'**;**  **end** **if;**  **if(**c\_x **>=** h\_start\_pixel**-**1 **and** c\_x **<** h\_end\_pixel**-**1 **and** c\_y **>=** v\_start\_line **and** c\_y **<** v\_end\_line**)** **then**  DISPLAY\_E\_s **<=** '1'**;** -- Si es pixel de display  **else**  DISPLAY\_E\_s **<=** '0'**;** -- Si no ho es  **end** **if;**  c\_x **<=** c\_x **+** 1**;** --Incrementa comptador horitzontal  **end** **if;**  **end** **if;**  **end** **if;**  **end** **process;**        **end** vga\_sync\_gen\_arc**;** |

## Temperature plotter entity

|  |
| --- |
| **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** temperature\_plotter **is**  **generic(**  --threshold display -> it does have a width of 35 lines  threshold\_y\_start **:** integer **:=** 768 **+** 30**;**  threshold\_y\_finish**:** integer **:=** 768 **+** 30 **+** 35**;**    --temperature bar display -> it does have a width of 30 lines  temp\_bar\_y\_start **:** integer **:=** 768 **+** 30**;**  temp\_bar\_y\_finish **:** integer **:=** 768 **+** 30 **+** 30**;**  temp\_bar\_x\_start **:** integer **:=** 0  **);**  **Port** **(** RGB\_in **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  PX **:** **in** unsigned **(**11 **downto** 0**);**  PY **:** **in** unsigned **(**11 **downto** 0**);**  alarm **:** **in** STD\_LOGIC**;**  temperature **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  t\_temperature **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RGB\_out **:** **out** STD\_LOGIC\_VECTOR **(**11 **downto** 0**));**  **end** temperature\_plotter**;**  **architecture** temperature\_plotter\_arc **of** temperature\_plotter **is**  **signal** RGB\_internal\_out **:** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  -- The idea here is that this component get the rgb that must be plotted from the signal plotter, and if we are not in the range  -- that we are interested in for plotting the monotoring application we will take the input and set it to the ouput,  -- but if we are, we are going to modify the RBG that we have at the input properly depending on the conditions  **constant** red **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"F00"**;**  **constant** blue **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"00F"**;**  **constant** green **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"0F0"**;**  **begin**  prova **:** **process(**PY**,** PX**)**  **begin**  RGB\_out **<=** RGB\_internal\_out**;**  **if** PY **<** threshold\_y\_start **or** PY **>** threshold\_y\_finish **then** -- we are not in the range of plotting the temperature monitoring  RGB\_internal\_out **<=** RGB\_in**;**  **else**  -- as the screen is "painted" from left to righ and up to down, the first element that  -- we will need to 'paint' is the threshold of the temperature, which is defined by t\_temperature  -- we know that in the y\_axis it will be from 540 to 575 (as it does have a width of 35 lines, statement defined)  **if** PX **=** **shift\_left(**unsigned**(**t\_temperature**),** 4**)** **then** --if we are in where the threshold should be placed, print it blue  RGB\_internal\_out **<=** blue**;**  **else** -- now we should take into account that the temperature bar, does have an smaller number of lines that the threshold, it goes from 542 to 572 (30 lines)  **if** temp\_bar\_y\_start **<=** PY **and** PY **<=** temp\_bar\_y\_finish **then**-- an smaller range  --now the x\_axis range will be from 0 to 80 (in the worst case), but in fact depends on the temperature  **if** temp\_bar\_x\_start **<=** PX **and** PX **<=** **shift\_left(**unsigned**(**temperature**),** 4**)** **then**  --HOWEVER we must take into account whether the temperature is higher than the threshold  --and therefore we need to check it with alarm  **if** alarm **=** '0' **then**  RGB\_internal\_out **<=** green**;**  **else**  RGB\_internal\_out **<=** red**;**  **end** **if;**  **else**  RGB\_internal\_out **<=** RGB\_in**;** -- we do not need to modify anything  **end** **if;**  **else**  RGB\_internal\_out **<=** RGB\_in**;** -- we do not need to modify anything  **end** **if;**  **end** **if;**  **end** **if;**  **end** **process;**  **end** temperature\_plotter\_arc**;** |

## Signal plotter entity

|  |
| --- |
| **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** signal\_plotter **is**  **Generic** **(**  pixels\_width **:** integer **:=** 9**;**  y\_divisions\_bits **:** integer **:=** 5**;**  x\_divisions\_bits **:** integer **:=** 5  **);**  **Port** **(** PX **:** **in** unsigned**(**11 **downto** 0**);**  PY **:** **in** unsigned**(**11 **downto** 0**);**  RGB\_in **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RGB\_out **:** **out** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  signal\_data **:** **in** std\_logic\_vector**(**11 **downto** 0**);**  alarm **:** **in** STD\_LOGIC**;**  vertical\_scale **:** **in** unsigned **(**2 **downto** 0**)**  **);**  **end** signal\_plotter**;**  **architecture** signal\_plotter\_arc **of** signal\_plotter **is**  **constant** zeros **:** std\_logic\_vector **(**15 **downto** 0**)** **:=** x"0000"**;**  **constant** signal\_color **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"FF0"**;**  **constant** divisions\_color **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"222"**;**  **constant** offset\_up **:** natural **:=** 256**;**  **constant** offset **:** natural **:=** 768**;**  **signal** signal\_padded **:** std\_logic\_vector **(**19 **downto** 0**);**  **signal** signal\_scaled **:** unsigned **(**19 **downto** 0**);**  **signal** signal\_pixels **:** unsigned **(**19 **downto** 0**);**  **signal** rgb\_i **:** std\_logic\_vector **(**11 **downto** 0**);**  **begin**    rgb\_i **<=** divisions\_color **when** **(**PY **<=** offset **and** PY **>=** offset\_up **)** **and** **((**unsigned**(**PY**(**y\_divisions\_bits **downto** 0**))** **=** 0**)** **or** **(**unsigned**(**PX**(**x\_divisions\_bits **downto** 0**))** **=** 0**))** **else** RGB\_in**;**    signal\_padded **<=** x"00" **&** signal\_data**;**    signal\_scaled **<=** **shift\_right(**unsigned**(**signal\_padded**),** **to\_integer(**vertical\_scale**));**      signal\_pixels **<=** **(**offset **-** signal\_scaled**);**    RGB\_out **<=** signal\_color **when** **(**PY **<=** offset **and** PY **>=** offset\_up **)** **and** **(**PY **=** signal\_pixels **and** signal\_scaled **<** offset **and** alarm **=** '0' **)** **else** rgb\_i**;**  **end** signal\_plotter\_arc**;** |

## Threshold plotter entity

|  |
| --- |
| **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** threshold\_plotter **is**  **Port** **(** PX **:** **in** unsigned**(**11 **downto** 0**);**  PY **:** **in** unsigned**(**11 **downto** 0**);**  RGB\_in **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RGB\_out **:** **out** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  -- Trigger level  trigger\_level **:** **in** std\_logic\_vector **(**8 **downto** 0**);**  vertical\_scale **:** **in** unsigned **(**2 **downto** 0**)**  **);**  **end** threshold\_plotter**;**  **architecture** signal\_plotter\_arc **of** threshold\_plotter **is**  **constant** offset **:** integer **range** 0 **to** 1023 **:=** 768**;**  **signal** trigger\_padded**,** trigger\_scaled**,** trigger\_level\_s **:** unsigned **(**trigger\_level'**length** **+** 8 **downto** 0**);**  **constant** trigger\_color **:** std\_logic\_vector **(**11 **downto** 0**)** **:=** x"22F"**;**  **begin**      trigger\_padded **<=** "000000" **&** unsigned**(**trigger\_level**)** **&** "000"**;**  trigger\_scaled **<=** **shift\_right(**unsigned**(**trigger\_padded**),** **to\_integer(**vertical\_scale**));**    trigger\_level\_s **<=** offset **-** trigger\_scaled**;**  RGB\_out **<=** **(**trigger\_color**)** **when** PY **=** trigger\_level\_s **and** trigger\_scaled **<=** offset **and** PX **<=** 20 **else** **(**RGB\_in**)** **;**  **end** signal\_plotter\_arc**;** |

## Memoria char entity

|  |
| --- |
| **library** ieee**;**  **use** ieee**.**std\_logic\_1164**.all;**  **use** ieee**.**numeric\_std**.all;**  **entity** memoria\_char **is**  **port(**  addr **:** **in** std\_logic\_vector**(**10 **downto** 0**);**  do **:** **out** std\_logic\_vector**(**7 **downto** 0**)**  **);**  **end** memoria\_char**;**  **architecture** syn **of** memoria\_char **is**  **type** char\_mem **is** **array** **(**0 **to** 1791**)** **of** std\_logic\_vector**(**7 **downto** 0**);**  **constant** char\_ROM **:** char\_mem **:=** **(**  x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** -- U+00 (nul)  x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** -- U+01  x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** x"00"**,** -- U+02    -- FILE SECTION OMMITED IN THE REPORT. JUST PIXEL MAP for all ASCII CHARACTERS    x"00"**,** x"38"**,** x"00"**,** x"1E"**,** x"33"**,** x"33"**,** x"1E"**,** x"00"**,** -- U+F3 (o aigu)  x"1E"**,** x"33"**,** x"00"**,** x"1E"**,** x"33"**,** x"33"**,** x"1E"**,** x"00"**,** -- U+F4 (o circumflex)  x"6E"**,** x"3B"**,** x"00"**,** x"1E"**,** x"33"**,** x"33"**,** x"1E"**,** x"00"**,** -- U+F5 (o ~)  x"00"**,** x"33"**,** x"00"**,** x"1E"**,** x"33"**,** x"33"**,** x"1E"**,** x"00"**,** -- U+F6 (o umlaut)  x"18"**,** x"18"**,** x"00"**,** x"7E"**,** x"00"**,** x"18"**,** x"18"**,** x"00"**,** -- U+F7 (division)  x"00"**,** x"60"**,** x"3C"**,** x"76"**,** x"7E"**,** x"6E"**,** x"3C"**,** x"06"**,** -- U+F8 (o stroke)  x"00"**,** x"07"**,** x"00"**,** x"33"**,** x"33"**,** x"33"**,** x"7E"**,** x"00"**,** -- U+F9 (u grave)  x"00"**,** x"38"**,** x"00"**,** x"33"**,** x"33"**,** x"33"**,** x"7E"**,** x"00"**,** -- U+FA (u aigu)  x"1E"**,** x"33"**,** x"00"**,** x"33"**,** x"33"**,** x"33"**,** x"7E"**,** x"00"**,** -- U+FB (u circumflex)  x"00"**,** x"33"**,** x"00"**,** x"33"**,** x"33"**,** x"33"**,** x"7E"**,** x"00"**,** -- U+FC (u umlaut)  x"00"**,** x"38"**,** x"00"**,** x"33"**,** x"33"**,** x"3E"**,** x"30"**,** x"1F"**,** -- U+FD (y aigu)  x"00"**,** x"00"**,** x"06"**,** x"3E"**,** x"66"**,** x"3E"**,** x"06"**,** x"00"**,** -- U+FE (thorn)  x"00"**,** x"33"**,** x"00"**,** x"33"**,** x"33"**,** x"3E"**,** x"30"**,** x"1F"**);** -- U+FF (y umlaut)  **begin**  do **<=** char\_ROM**(to\_integer(**unsigned**(**addr**)));**  **end** syn**;** |

## Frequency Meter entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company:  -- Engineer: Eva Deltor  --  -- Create Date: 01/18/2022 02:20:43 PM  -- Design Name:  -- Module Name: frequency\_meter - arc  -- Project Name:  -- Target Devices:  -- Tool Versions:  -- Description:  --  -- Dependencies:  --  -- Revision:  -- Revision 0.01 - File Created  -- Additional Comments:  --  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **use** IEEE**.**std\_logic\_unsigned**.all;**  **entity** frequency\_meter **is**  **Port** **(** trigger\_edge **:** **in** STD\_LOGIC**;**  clk **:** **in** STD\_LOGIC**;**  rst **:** **in** STD\_LOGIC**;**  frequency\_x100\_bcd **:** **out** std\_logic\_vector**(**27 **downto** 0**));**  **end** frequency\_meter**;**  **architecture** arc **of** frequency\_meter **is**  **signal** counter\_time**:** natural **range** 0 **to** 108e6**;**    **begin**  **process(**CLK**)**  **variable** u**,**d**,**c**,**umillar**,**dmillar**,**cmillar**,**umillon **:** natural **range** 0 **to** 10**;**  **begin**  **if(**CLK **=** '1' **and** CLK'**event)** **then**  **if(**RST **=** '1'**)** **then**  counter\_time **<=** 0**;**  u **:=** 0**;**  d **:=** 0**;**  c **:=** 0**;**  umillar **:=** 0**;**  cmillar **:=** 0**;**  dmillar **:=** 0**;**  umillon **:=** 0**;**  **else**  **if** **(**trigger\_edge **=** '1'**)** **then**  -- in the extrem case we will count a trigger for each clock therefore we will need to count up to  u **:=** u **+** 1**;** --unitats  **if(**u **>=** 10**)** **then**  d **:=** d **+** 1**;** --desenes  u **:=** 0**;**  **end** **if;**    **if(**d **>=** 10**)** **then**  c **:=** c **+** 1**;** --centenes  d **:=** 0**;**  **end** **if;**    **if** **(**c **>=** 10**)** **then**  umillar **:=** umillar **+** 1**;**  c **:=** 0**;**  **end** **if;**    **if** **(**umillar **>=** 10**)** **then**  dmillar **:=** umillar **+** 1**;**  umillar **:=** 0**;**  **end** **if;**    **if** **(**dmillar **>=** 10**)** **then**  cmillar **:=** umillar **+** 1**;**  dmillar **:=** 0**;**  **end** **if;**    **if** **(**cmillar **>=** 10**)** **then** --we do not need more than 9 million, in fact the limit is 1080000  umillon **:=** umillon **+** 1**;**  cmillar **:=** 0**;**  **end** **if;**  **end** **if;**  -- I need a counter so as to count up to 1080000 this will be the one that controls the time  **if** **(** counter\_time **>=** 1079999**)** **then**  counter\_time **<=** 0**;**  -- define the external output  frequency\_x100\_bcd **<=** std\_logic\_vector**(to\_unsigned(**umillon**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**cmillar**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**dmillar**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**umillar**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**c**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**d**,** 4**))** **&** std\_logic\_vector**(to\_unsigned(**u**,** 4**));** -- reset the value of the decimal counter  u **:=** 0**;**  d **:=** 0**;**  c **:=** 0**;**  umillar **:=** 0**;**  cmillar **:=** 0**;**  dmillar **:=** 0**;**  umillon **:=** 0**;**  **else**  counter\_time **<=** counter\_time **+** 1**;**  **end** **if;**    **end** **if;**  **end** **if;**  **end** **process;**  **end** arc**;** |

## Frequency plotter entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company:  -- Engineer:  --  -- Create Date: 01/18/2022 02:22:46 PM  -- Design Name:  -- Module Name: frequency\_plotter - combinational  -- Project Name:  -- Target Devices:  -- Tool Versions:  -- Description:  --  -- Dependencies:  --  -- Revision:  -- Revision 0.01 - File Created  -- Additional Comments:  --  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** frequency\_plotter **is**  **Generic** **(**  frequency\_width **:** natural **:=** 32  **);**  **Port** **(**  clk **:** std\_logic**;**  PX **:** **in** unsigned**(**11 **downto** 0**);**  PY **:** **in** unsigned**(**11 **downto** 0**);**  RGB\_in **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RGB\_out **:** **out** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  frequency\_x100\_bcd **:** **in** std\_logic\_vector**(**27 **downto** 0**);**  y\_scale\_select**,** x\_scale\_select **:** std\_logic\_vector**(**2 **downto** 0**);**  mode\_indicator **:** std\_logic\_vector**(**3 **downto** 0**);**  polarity **:** std\_logic\_vector**(**1 **downto** 0**);**  alarm **:** **in** STD\_LOGIC  **);**  **end** frequency\_plotter**;**  **architecture** combinational **of** frequency\_plotter **is**  -- Types  type text\_t is array (7 downto 0) of string (0 to 39);  type scale\_label\_t is array (7 downto 0) of string (0 to 4);  type polarity\_text\_t is array ( 0 to 2) of string ( 0 to 7);  -- config  constant text\_color : std\_logic\_vector := x"FFF";  -- String constants  constant scale\_1 : scale\_label\_t := (" 8192", " 4096", " 2048", " 1024", " 512", " 256", " 128", " 64");  constant scale\_2 : scale\_label\_t := ("500 m", " 2 m", "500 u", "250 u", "100 u", " 64 u", " 48 u", " 32 u");  constant copyr : character := character'val(137);  constant polarity\_text : polarity\_text\_t := ("Rising ", "Falling ", "Free run" );    -- UI  signal y\_scale\_index, x\_scale\_index : natural range 0 to 7;  signal polarity\_index : integer range 2 downto 0;  signal frequency\_text : string(0 to 8);  signal arrow : string(0 to 3);  -- Text  signal lines : text\_t;  signal space\_line : std\_logic;  -- RGB Plotting  signal char\_pixel : natural range 0 to 7;  signal px\_s, py\_s : unsigned(11 downto 0);  -- Char memory interface  signal char\_addr : std\_logic\_vector(10 downto 0);  signal char\_line : std\_logic\_vector(7 downto 0);  component memoria\_char  port(  addr : in std\_logic\_vector(10 downto 0);  do : out std\_logic\_vector(7 downto 0)  );  end component;  begin  frequency\_text(0) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(27 downto 24))));  frequency\_text(1) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(23 downto 20))));  frequency\_text(2) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(19 downto 16))));  frequency\_text(3) <= '.';  frequency\_text(4) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(15 downto 12))));  frequency\_text(5) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(11 downto 8))));  frequency\_text(6) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(7 downto 4))));  frequency\_text(7) <= ',';  frequency\_text(8) <= character'val(48 + to\_integer(unsigned(frequency\_x100\_bcd(3 downto 0))));    y\_scale\_index <= to\_integer(unsigned(y\_scale\_select));  x\_scale\_index <= to\_integer(unsigned(x\_scale\_select));  polarity\_index <= to\_integer(unsigned(polarity));    arrow(0) <= '>' when mode\_indicator(0) = '1' else ' ';  arrow(1) <= '>' when mode\_indicator(1) = '1' else ' ';  arrow(2) <= '>' when mode\_indicator(2) = '1' else ' ';  arrow(3) <= '>' when mode\_indicator(3) = '1' else ' ';    lines(0) <= " "& "Frequency = " & frequency\_text & " kHz "; --40  lines(1) <= (others => ' ');    lines(2) <= arrow(0) & "Trigger polarity: " & polarity\_text(polarity\_index) & " "; --40  lines(3) <= arrow(1) & "Trigger level " ;  lines(4) <= arrow(2) & "Vertical scale: " & scale\_1(y\_scale\_index) & " counts/div "; --40  lines(5) <= arrow(3) & "Horizontal scale: " & scale\_2(x\_scale\_index) & "s/div "; --40      lines(6) <= (others => ' ');  lines(7) <= copyr & " 2021-2022 Eva Deltor & Pau Oliveras ";    process(CLK)  -- Char select  variable line\_index : natural range 0 to 7;  variable char\_index : natural range 0 to 39;  variable char\_line\_index : std\_logic\_vector(2 downto 0);  variable char\_ascii : std\_logic\_vector(7 downto 0);  begin  if rising\_edge(CLK) then  px\_s <= PX;  py\_s <= PY;  space\_line <= PY(3);    line\_index := to\_integer(py\_s(11 downto 4)); -- Selects line using LSB + 3.    char\_index := to\_integer(px\_s(11 downto 3)); -- Selects character using LSB + 3.    char\_ascii := std\_logic\_vector(to\_unsigned(character'pos(lines(line\_index)(char\_index)), char\_ascii'length)); -- Gets ascii vector from text in current line, character.    char\_line\_index := std\_logic\_vector(py\_s(2 downto 0)); -- Gets current character line from PY's least significant 3 bits.    char\_addr <= char\_ascii & char\_line\_index; -- Construct address for character memory.    char\_pixel <= to\_integer(px\_s(2 downto 0)); -- Pixel number within line from PX least significant 3 bits.  end if;  end process;    RGB\_out <= (others => '1') when (px\_s < 8\*40 and py\_s < 16\*8 and char\_line(char\_pixel) = '1' and space\_line = '0') else RGB\_in;      memoria\_char\_1 : memoria\_char  port map(  addr => char\_addr,  do => char\_line  );    end combinational; |

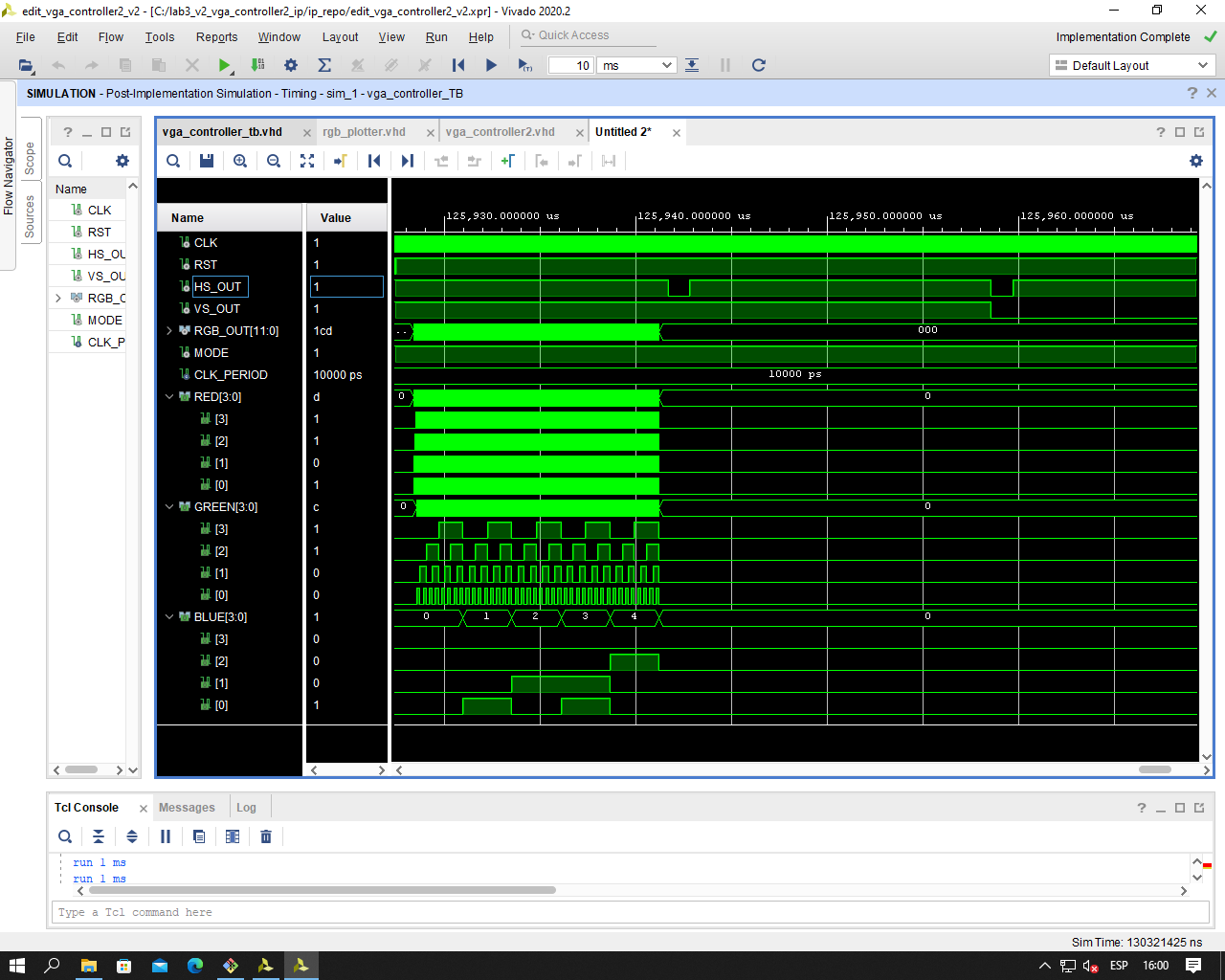
## Vga\_controller entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: UPC-ADS  -- Engineer: Pau Jordan Oliveras, Eva Maria Deltor  --  -- Create Date: 10/14/2021 05:48:35 PM  -- Design Name:  -- Module Name: vga\_controller - vga\_controller\_arc  -- Revision:  -- Revision 0.01 - File Created  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** vga\_controller **is**  **generic** **(** -- We define the constants of the component  h\_pixels **:** integer **:=** 1688**;** -- Total number of hortizontal pixels (visible and non visible) - total number of pixel clock cycles  h\_sync **:** integer **:=** 112**;** -- Sync pulse width in clock cycles  h\_start\_pixel **:** integer **:=** 360**;** --pixel in which the visible part starts  h\_end\_pixel **:** integer **:=** 1640**;** -- first pixel out of the visible region  v\_lines **:** integer **:=** 1066**;** --Total number of lines in a frame (visible and non visible)  v\_sync **:** integer **:=** 3**;** --Sync vertical pulse width in lines  v\_start\_line **:** integer **:=** 41**;** -- line in which the visible part starts  v\_end\_line **:** integer **:=** 1065**;** -- first line out of the visible region  h\_bits **:** integer **:=** 11**;** -- N bits horizontal counter (max util 1688)  v\_bits **:** integer **:=** 11**);** -- N bits vertical counter (max util 1066)    **Port** **(**  mode **:** **in** std\_logic**;**  heartbeat **:** **out** STD\_LOGIC**;** -- this signal is not in the assigment but has been added to know if the clock is running.  clk **:** **in** STD\_LOGIC**;**  resetn **:** **in** STD\_LOGIC**;**  vsync **:** **out** STD\_LOGIC**;**  hsync **:** **out** STD\_LOGIC**;**  PIXEL\_X **:** **out** unsigned**(**h\_bits **downto** 0**);** -- Numero de Pixel X  PIXEL\_Y **:** **out** unsigned**(**v\_bits **downto** 0**);** -- Numero de Pixel Y  red **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  green **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  blue **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**  **end** vga\_controller**;**  **architecture** vga\_controller\_arc **of** vga\_controller **is**  **signal** c\_x **:** integer **range** **(**2**\*\*(**h\_bits**)-**1**)** **downto** 0**;** -- signal that counts the hortizontal pixels  **signal** c\_y **:** integer **range** **(**2**\*\*(**v\_bits**)-**1**)** **downto** 0**;** -- signal that counts the lines  **signal** HS\_s**,** VS\_s**,** DISPLAY\_E\_s**:** std\_logic**;** -- signal for the vga sync signals, and signal to indicate current pixel is active.  **signal** heartbeat\_counter **:** integer **range** **(**2**\*\***26**)** **downto** 0**;** --counter that overflows at 2 Hz (as was used by the led in the board, the frequency should be seen by the human eye)  **signal** pixel\_x\_s **:** unsigned**(**h\_bits **downto** 0**);** --signal that coutns the visible x axis pixels  **signal** pixel\_y\_s **:** unsigned**(**v\_bits **downto** 0**);** -- signal that counts the visible y axis lines  **component** rgb\_plotter -- component defined in the rgb\_plotter.vhd file, generates the color patterns given the pixel counters and the enable signal (generated in this current vhd)  **generic** **(**  --colors definitions  --horizontal: if we know that we have a horitzontal display area of 1024 - it will be divided by 3 in 342  -- Here we define all the constants of the component RGB plotter  horizontal\_red\_fin**:** integer **:=** 342**;** --ends red line  horizontal\_green\_fin**:** integer **:=** 682**;**-- ends the green line  horizontal\_blue\_fin**:** integer **:=** 1023**);**-- ends the blue line    **Port** **(** ENABLE **:** **in** STD\_LOGIC**;**  MODE **:** **in** STD\_LOGIC**;**  PIXEL\_X **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  PIXEL\_Y **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RED **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  GREEN **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  BLUE **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**  **end** **component;**    **begin**  rgb\_plotter1 **:** rgb\_plotter --we include an instance of the rgb plotter component.  **Port** **map(**  ENABLE **=>** DISPLAY\_E\_s**,**  MODE **=>** MODE**,**  PIXEL\_X **=>** std\_logic\_vector**(**pixel\_x\_s**),**  PIXEL\_Y **=>** std\_logic\_vector**(**pixel\_y\_s**),**  RED **=>** red**,**  GREEN **=>** green**,**  BLUE **=>** blue  **);**  **process(**CLK**)**  **begin**  **if(**CLK **=** '1' **and** CLK'**event)** **then** -- CLK  **if(**resetn **=** '0'**)** **then** --we define all the counters to 0 once the reset is activated  heartbeat\_counter **<=** 0**;**  c\_x **<=** 0**;**  c\_y **<=** 0**;**  HS\_s **<=** '0'**;**  VS\_s **<=** '0'**;**  **else** -- the counter is incremented each clock cycle, when it reaches 33554432 the led is tuned off, else it is turned on.  -- When the counter overflows it is set to zero.  -- We have defined this procedure as we needed to have a visual clue that the system is working, without the screen.  **if(**heartbeat\_counter **>** 2**\*\***25**)** **then**  heartbeat **<=** '0'**;**  **else**  heartbeat **<=** '1'**;**  **end** **if;**    **if(**heartbeat\_counter **>=** **(**2**\*\***26 **-** 1**))** **then**  heartbeat\_counter **<=** 0**;**  **else**  heartbeat\_counter **<=** heartbeat\_counter **+** 1**;**  **end** **if;**      **if(**c\_x **>=** **(**h\_pixels**-**1**))** **then** -- If we have finished the line  c\_x **<=** 0**;** -- Reset the horiztonal counter and check the vertical one  HS\_s **<=** '0'**;** -- Hortizonal sync pulse  **if(**c\_y **>=** **(**v\_lines**-**1**))** **then** -- If the frame has ended  c\_y **<=** 0**;** -- Reset the vertical counter  VS\_s **<=** '0'**;** -- Vertical sync pulse  **else**  **if** **(**c\_y **>=** **(**v\_sync **-** 1**))** **then** -- If it is no longer a vertical pulse line, pulse ends.  VS\_s **<=** '1'**;**  **end** **if;**  c\_y **<=** c\_y **+** 1**;** -- Increase vertical counter  **end** **if;**  **else**  **if(**c\_x **>=** **(**h\_sync **-** 1**))** **then** -- If it is no longer a hortizontal pulse line, pulse ends.  HS\_s **<=** '1'**;**  **end** **if;**  **if(**c\_x **>=** h\_start\_pixel**-**1 **and** c\_x **<** h\_end\_pixel**-**1 **and** c\_y **>=** v\_start\_line **and** c\_y **<** v\_end\_line**)** **then**  DISPLAY\_E\_s **<=** '1'**;** -- If the selected pixel is inside the display area  **else**  DISPLAY\_E\_s **<=** '0'**;** -- If the selected pixel is not inside the display area  **end** **if;**  c\_x **<=** c\_x **+** 1**;** --Increment the horizontal counter  **end** **if;**  **end** **if;**  **end** **if;**  **end** **process;**      -- We assign the internal signals to the ports  hsync **<=** HS\_s**;**  vsync **<=** VS\_s**;**  PIXEL\_X **<=** pixel\_x\_s**;**  PIXEL\_Y **<=** pixel\_y\_s**;**  pixel\_x\_s **<=** **to\_unsigned(**c\_x **-** h\_start\_pixel**,** pixel\_x\_s'**length);** -- n pixel X es comptador horitzontal - primer pixel  pixel\_y\_s **<=** **to\_unsigned(**c\_y **-** v\_start\_line**,** pixel\_y\_s'**length);** -- n linia Y es comptador vertical - primera linia    **end** vga\_controller\_arc**;** |

## Rgb\_plotter entity

|  |
| --- |
| ----------------------------------------------------------------------------------  -- Company: ADS  -- Engineer: Pau Jordan Oliveres, Eva María Deltor  --  -- Create Date: 11/03/2021 06:05:29 PM  -- Design Name:  -- Module Name: rgb\_plotter - rgb\_plotter\_arc  --This component generates two color patterns based on the mode. The enable signal enables or disables the output and the pixel\_x and pixel\_y inputs are used to generate the pattern.  --It is a fully combinational component.  -- Revision:  -- Revision 0.01 - File Created  -- Additional Comments:  --  ----------------------------------------------------------------------------------  **library** IEEE**;**  **use** IEEE**.**STD\_LOGIC\_1164**.ALL;**  **use** IEEE**.**NUMERIC\_STD**.ALL;**  **entity** rgb\_plotter **is**  **generic** **(**  -- Here we define all the constants of the component RGB plotter  horizontal\_red\_fin**:** integer **:=** 342**;** --ends red line  horizontal\_green\_fin**:** integer **:=** 683**;**-- ends the green line  horizontal\_blue\_fin**:** integer **:=** 1023**);**-- ends the blue line      **Port** **(** ENABLE **:** **in** STD\_LOGIC**;**  MODE **:** **in** STD\_LOGIC**;**  PIXEL\_X **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  PIXEL\_Y **:** **in** STD\_LOGIC\_VECTOR **(**11 **downto** 0**);**  RED **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  GREEN **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);**  BLUE **:** **out** STD\_LOGIC\_VECTOR **(**3 **downto** 0**));**  **end** rgb\_plotter**;**  **architecture** rgb\_plotter\_arc **of** rgb\_plotter **is**  **signal** pixel\_x\_s **:** integer**;** -- this signal counts which pixel in the x axis is currently in we only take into account the visible ones  **signal** pixel\_y\_s **:** integer**;** -- this signal counts which pixel in the y axis is currently in we only take into account the visible ones  **begin**  pixel\_x\_s **<=** **TO\_INTEGER(**unsigned**(**PIXEL\_X**));** -- the input port is an unsigned type, our signal is an integrer type  pixel\_y\_s **<=** **TO\_INTEGER(**unsigned**(**PIXEL\_Y**));** -- the input port is an unsigned type, our signal is an integrer type    **process(**ENABLE**,** MODE**,** PIXEL\_X**,** PIXEL\_Y**)** -- if any of this signals changes the process needs to be run  **begin**  **if** **(**ENABLE **=** '0'**)** **then** -- when the enable is zero; the output (color of the screen) is black.  RED **<=** x"0"**;**  GREEN **<=** x"0"**;**  BLUE **<=** x"0"**;**  **else**  **if(** MODE **=** '0'**)then** -- hortizontal stripes when mode = 0  -- we compare pixel y against the constants defined at the beining of the file, which define the bounds of each color stripe  **if** **(**pixel\_y\_s **<=** horizontal\_red\_fin**)** **then**  RED **<=** x"F"**;** -- red all in 1  GREEN **<=** x"0"**;** -- green all in 0  BLUE **<=** x"0"**;** -- blue all in 0  -- so in this case will be red.  --Same procedure in the following situations.  **elsif** **(**pixel\_y\_s **<=** horizontal\_green\_fin**)** **then**  RED **<=** x"0"**;**  GREEN **<=** x"F"**;**  BLUE **<=** x"0"**;**  **elsif** **(**pixel\_y\_s **<=** horizontal\_blue\_fin**)** **then**  RED **<=** x"0"**;**  GREEN **<=** x"0"**;**  BLUE **<=** x"F"**;**  **else** -- In case of not being in any of the previous situations (error situation) the screen will be in white.  RED **<=** x"F"**;**  GREEN **<=** x"F"**;**  BLUE **<=** x"F"**;**  **end** **if;**  **else** -- if we are in mode 1; we simply assign the different color values some bits of the PIXEL\_X counter, resulting in a color pattern.  RED **<=** PIXEL\_X**(**3 **downto** 0**);** --changes fast as represents the LSB  GREEN **<=** PIXEL\_X**(**7 **downto** 4**);**--changes slower than red, as we can have more than one color cycle  BLUE **<=** '0' **&** PIXEL\_X**(**10 **downto** 8**);** -- the slowest change; MSB    **end** **if;**  **end** **if;**  **end** **process;**  **end** rgb\_plotter\_arc**;** |

# Post-implementation simulation



# Timing report

|  |
| --- |
| Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.  -----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  | Tool Version : Vivado v.2020.2 (win64) Build 3064766 Wed Nov 18 09:12:45 MST 2020  | Date : Mon Dec 13 16:23:09 2021  | Host : c5b1 running 64-bit major release (build 9200)  | Command : report\_timing\_summary -max\_paths 10 -file vga\_controller2\_v2\_timing\_summary\_routed.rpt -pb vga\_controller2\_v2\_timing\_summary\_routed.pb -rpx vga\_controller2\_v2\_timing\_summary\_routed.rpx -warn\_on\_violation  | Design : vga\_controller2\_v2  | Device : 7z020-clg484  | Speed File : -1 PRODUCTION 1.12 2019-11-22  -----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  Timing Summary Report  ------------------------------------------------------------------------------------------------  | Timer Settings  | --------------  ------------------------------------------------------------------------------------------------  Enable Multi Corner Analysis : Yes  Enable Pessimism Removal : Yes  Pessimism Removal Resolution : Nearest Common Node  Enable Input Delay Default Clock : No  Enable Preset / Clear Arcs : No  Disable Flight Delays : No  Ignore I/O Paths : No  Timing Early Launch at Borrowing Latches : No  Borrow Time for Max Delay Exceptions : Yes  Merge Timing Exceptions : Yes  Corner Analyze Analyze  Name Max Paths Min Paths  ------ --------- ---------  Slow Yes Yes  Fast Yes Yes  check\_timing report  Table of Contents  -----------------  1. checking no\_clock (0)  2. checking constant\_clock (0)  3. checking pulse\_width\_clock (0)  4. checking unconstrained\_internal\_endpoints (0)  5. checking no\_input\_delay (46)  6. checking no\_output\_delay (52)  7. checking multiple\_clock (0)  8. checking generated\_clocks (0)  9. checking loops (0)  10. checking partial\_input\_delay (0)  11. checking partial\_output\_delay (0)  12. checking latch\_loops (0)  1. checking no\_clock (0)  ------------------------  There are 0 register/latch pins with no clock.  2. checking constant\_clock (0)  ------------------------------  There are 0 register/latch pins with constant\_clock.  3. checking pulse\_width\_clock (0)  ---------------------------------  There are 0 register/latch pins which need pulse\_width check  4. checking unconstrained\_internal\_endpoints (0)  ------------------------------------------------  There are 0 pins that are not constrained for maximum delay.  There are 0 pins that are not constrained for maximum delay due to constant clock.  5. checking no\_input\_delay (46)  -------------------------------  There are 46 input ports with no input delay specified. (HIGH)  There are 0 input ports with no input delay but user has a false path constraint.  6. checking no\_output\_delay (52)  --------------------------------  There are 52 ports with no output delay specified. (HIGH)  There are 0 ports with no output delay but user has a false path constraint  There are 0 ports with no output delay but with a timing clock defined on it or propagating through it  7. checking multiple\_clock (0)  ------------------------------  There are 0 register/latch pins with multiple clocks.  8. checking generated\_clocks (0)  --------------------------------  There are 0 generated clocks that are not connected to a clock source.  9. checking loops (0)  ---------------------  There are 0 combinational loops in the design.  10. checking partial\_input\_delay (0)  ------------------------------------  There are 0 input ports with partial input delay specified.  11. checking partial\_output\_delay (0)  -------------------------------------  There are 0 ports with partial output delay specified.  12. checking latch\_loops (0)  ----------------------------  There are 0 combinational latch loops in the design through latch input  ------------------------------------------------------------------------------------------------  | Design Timing Summary  | ---------------------  ------------------------------------------------------------------------------------------------  WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints WPWS(ns) TPWS(ns) TPWS Failing Endpoints TPWS Total Endpoints  ------- ------- --------------------- ------------------- ------- ------- --------------------- ------------------- -------- -------- ---------------------- --------------------  4.437 0.000 0 319 0.106 0.000 0 319 4.129 0.000 0 224  All user specified timing constraints are met.  ------------------------------------------------------------------------------------------------  | Clock Summary  | -------------  ------------------------------------------------------------------------------------------------  Clock Waveform(ns) Period(ns) Frequency(MHz)  ----- ------------ ---------- --------------  axi\_clock {0.000 4.630} 9.259 108.003  ------------------------------------------------------------------------------------------------  | Intra Clock Table  | -----------------  ------------------------------------------------------------------------------------------------  Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints WPWS(ns) TPWS(ns) TPWS Failing Endpoints TPWS Total Endpoints  ----- ------- ------- --------------------- ------------------- ------- ------- --------------------- ------------------- -------- -------- ---------------------- --------------------  axi\_clock 4.437 0.000 0 319 0.106 0.000 0 319 4.129 0.000 0 224  ------------------------------------------------------------------------------------------------  | Inter Clock Table  | -----------------  ------------------------------------------------------------------------------------------------  From Clock To Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints  ---------- -------- ------- ------- --------------------- ------------------- ------- ------- --------------------- -------------------  ------------------------------------------------------------------------------------------------  | Other Path Groups Table  | -----------------------  ------------------------------------------------------------------------------------------------  Path Group From Clock To Clock WNS(ns) TNS(ns) TNS Failing Endpoints TNS Total Endpoints WHS(ns) THS(ns) THS Failing Endpoints THS Total Endpoints  ---------- ---------- -------- ------- ------- --------------------- ------------------- ------- ------- --------------------- ------------------- |

# Area report

|  |
| --- |
| Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.  -----------------------------------------------------------------------------------------------------------  | Tool Version : Vivado v.2020.2 (win64) Build 3064766 Wed Nov 18 09:12:45 MST 2020  | Date : Mon Dec 13 16:34:02 2021  | Host : c5b1 running 64-bit major release (build 9200)  | Command : report\_utilization -file C:/ADS\_do/lab3/report/utilization\_report.txt -name utilization\_1  | Design : vga\_controller2\_v2  | Device : 7z020clg484-1  | Design State : Routed  -----------------------------------------------------------------------------------------------------------  Utilization Design Information  Table of Contents  -----------------  1. Slice Logic  1.1 Summary of Registers by Type  2. Slice Logic Distribution  3. Memory  4. DSP  5. IO and GT Specific  6. Clocking  7. Specific Feature  8. Primitives  9. Black Boxes  10. Instantiated Netlists  1. Slice Logic  --------------  +-------------------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +-------------------------+------+-------+-----------+-------+  | Slice LUTs | 129 | 0 | 53200 | 0.24 |  | LUT as Logic | 129 | 0 | 53200 | 0.24 |  | LUT as Memory | 0 | 0 | 17400 | 0.00 |  | Slice Registers | 223 | 0 | 106400 | 0.21 |  | Register as Flip Flop | 223 | 0 | 106400 | 0.21 |  | Register as Latch | 0 | 0 | 106400 | 0.00 |  | F7 Muxes | 0 | 0 | 26600 | 0.00 |  | F8 Muxes | 0 | 0 | 13300 | 0.00 |  +-------------------------+------+-------+-----------+-------+  1.1 Summary of Registers by Type  --------------------------------  +-------+--------------+-------------+--------------+  | Total | Clock Enable | Synchronous | Asynchronous |  +-------+--------------+-------------+--------------+  | 0 | \_ | - | - |  | 0 | \_ | - | Set |  | 0 | \_ | - | Reset |  | 0 | \_ | Set | - |  | 0 | \_ | Reset | - |  | 0 | Yes | - | - |  | 0 | Yes | - | Set |  | 0 | Yes | - | Reset |  | 3 | Yes | Set | - |  | 220 | Yes | Reset | - |  +-------+--------------+-------------+--------------+  2. Slice Logic Distribution  ---------------------------  +--------------------------------------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +--------------------------------------------+------+-------+-----------+-------+  | Slice | 72 | 0 | 13300 | 0.54 |  | SLICEL | 53 | 0 | | |  | SLICEM | 19 | 0 | | |  | LUT as Logic | 129 | 0 | 53200 | 0.24 |  | using O5 output only | 0 | | | |  | using O6 output only | 92 | | | |  | using O5 and O6 | 37 | | | |  | LUT as Memory | 0 | 0 | 17400 | 0.00 |  | LUT as Distributed RAM | 0 | 0 | | |  | LUT as Shift Register | 0 | 0 | | |  | Slice Registers | 223 | 0 | 106400 | 0.21 |  | Register driven from within the Slice | 86 | | | |  | Register driven from outside the Slice | 137 | | | |  | LUT in front of the register is unused | 125 | | | |  | LUT in front of the register is used | 12 | | | |  | Unique Control Sets | 27 | | 13300 | 0.20 |  +--------------------------------------------+------+-------+-----------+-------+  \* \* Note: Available Control Sets calculated as Slice \* 1, Review the Control Sets Report for more information regarding control sets.  3. Memory  ---------  +----------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +----------------+------+-------+-----------+-------+  | Block RAM Tile | 0 | 0 | 140 | 0.00 |  | RAMB36/FIFO\* | 0 | 0 | 140 | 0.00 |  | RAMB18 | 0 | 0 | 280 | 0.00 |  +----------------+------+-------+-----------+-------+  \* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1  4. DSP  ------  +-----------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +-----------+------+-------+-----------+-------+  | DSPs | 0 | 0 | 220 | 0.00 |  +-----------+------+-------+-----------+-------+  5. IO and GT Specific  ---------------------  +-----------------------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +-----------------------------+------+-------+-----------+-------+  | Bonded IOB | 104 | 0 | 200 | 52.00 |  | IOB Master Pads | 49 | | | |  | IOB Slave Pads | 52 | | | |  | Bonded IPADs | 0 | 0 | 2 | 0.00 |  | Bonded IOPADs | 0 | 0 | 130 | 0.00 |  | PHY\_CONTROL | 0 | 0 | 4 | 0.00 |  | PHASER\_REF | 0 | 0 | 4 | 0.00 |  | OUT\_FIFO | 0 | 0 | 16 | 0.00 |  | IN\_FIFO | 0 | 0 | 16 | 0.00 |  | IDELAYCTRL | 0 | 0 | 4 | 0.00 |  | IBUFDS | 0 | 0 | 192 | 0.00 |  | PHASER\_OUT/PHASER\_OUT\_PHY | 0 | 0 | 16 | 0.00 |  | PHASER\_IN/PHASER\_IN\_PHY | 0 | 0 | 16 | 0.00 |  | IDELAYE2/IDELAYE2\_FINEDELAY | 0 | 0 | 200 | 0.00 |  | ILOGIC | 0 | 0 | 200 | 0.00 |  | OLOGIC | 0 | 0 | 200 | 0.00 |  +-----------------------------+------+-------+-----------+-------+  6. Clocking  -----------  +------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +------------+------+-------+-----------+-------+  | BUFGCTRL | 1 | 0 | 32 | 3.13 |  | BUFIO | 0 | 0 | 16 | 0.00 |  | MMCME2\_ADV | 0 | 0 | 4 | 0.00 |  | PLLE2\_ADV | 0 | 0 | 4 | 0.00 |  | BUFMRCE | 0 | 0 | 8 | 0.00 |  | BUFHCE | 0 | 0 | 72 | 0.00 |  | BUFR | 0 | 0 | 16 | 0.00 |  +------------+------+-------+-----------+-------+  7. Specific Feature  -------------------  +-------------+------+-------+-----------+-------+  | Site Type | Used | Fixed | Available | Util% |  +-------------+------+-------+-----------+-------+  | BSCANE2 | 0 | 0 | 4 | 0.00 |  | CAPTUREE2 | 0 | 0 | 1 | 0.00 |  | DNA\_PORT | 0 | 0 | 1 | 0.00 |  | EFUSE\_USR | 0 | 0 | 1 | 0.00 |  | FRAME\_ECCE2 | 0 | 0 | 1 | 0.00 |  | ICAPE2 | 0 | 0 | 2 | 0.00 |  | STARTUPE2 | 0 | 0 | 1 | 0.00 |  | XADC | 0 | 0 | 1 | 0.00 |  +-------------+------+-------+-----------+-------+  8. Primitives  -------------  +----------+------+---------------------+  | Ref Name | Used | Functional Category |  +----------+------+---------------------+  | FDRE | 220 | Flop & Latch |  | LUT6 | 76 | LUT |  | OBUF | 56 | IO |  | IBUF | 48 | IO |  | LUT4 | 43 | LUT |  | LUT5 | 19 | LUT |  | LUT2 | 13 | LUT |  | CARRY4 | 13 | CarryLogic |  | LUT3 | 11 | LUT |  | LUT1 | 4 | LUT |  | FDSE | 3 | Flop & Latch |  | BUFG | 1 | Clock |  +----------+------+---------------------+  9. Black Boxes  --------------  +----------+------+  | Ref Name | Used |  +----------+------+  10. Instantiated Netlists  -------------------------  +----------+------+  | Ref Name | Used |  +----------+------+ |