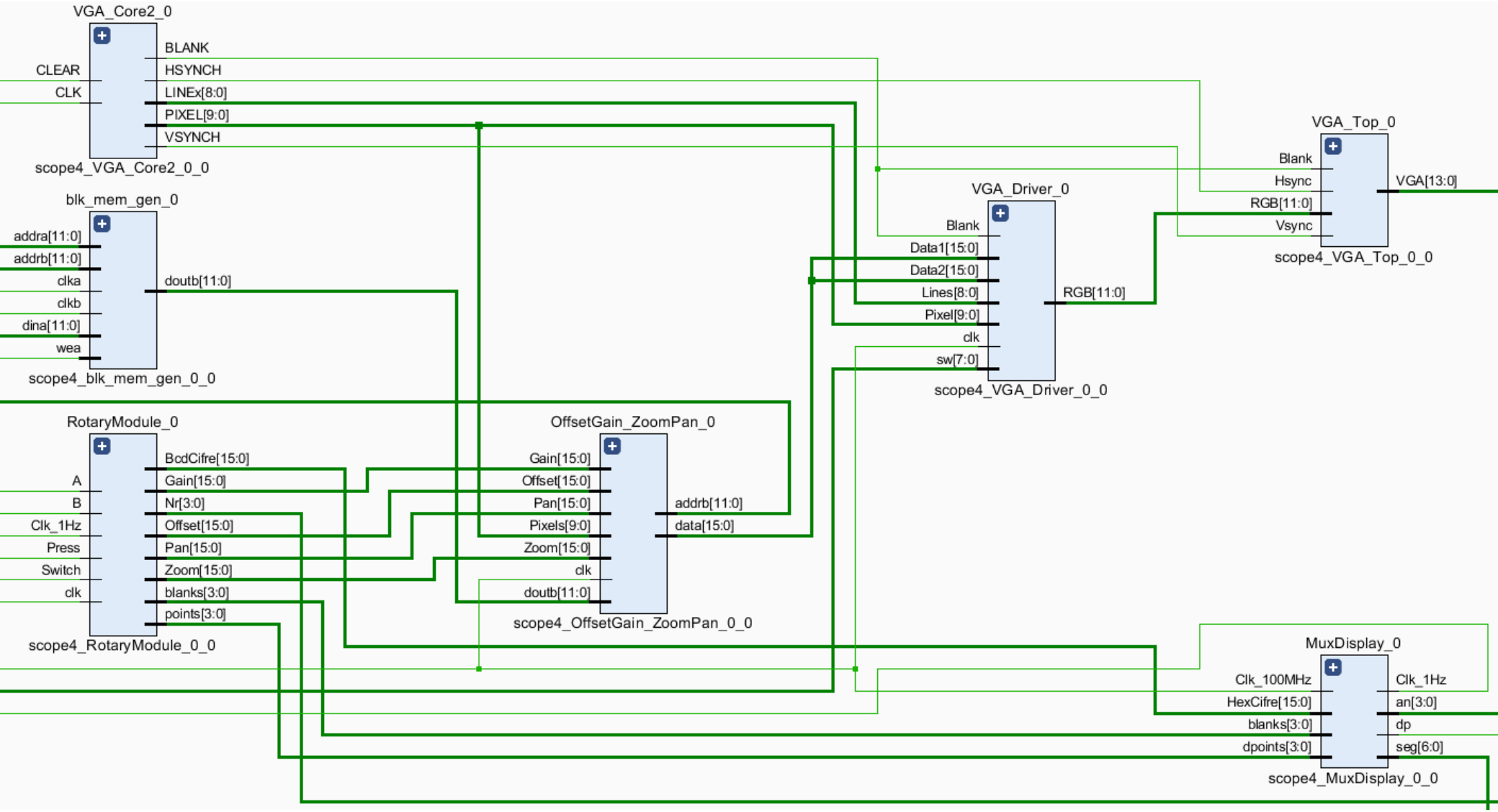
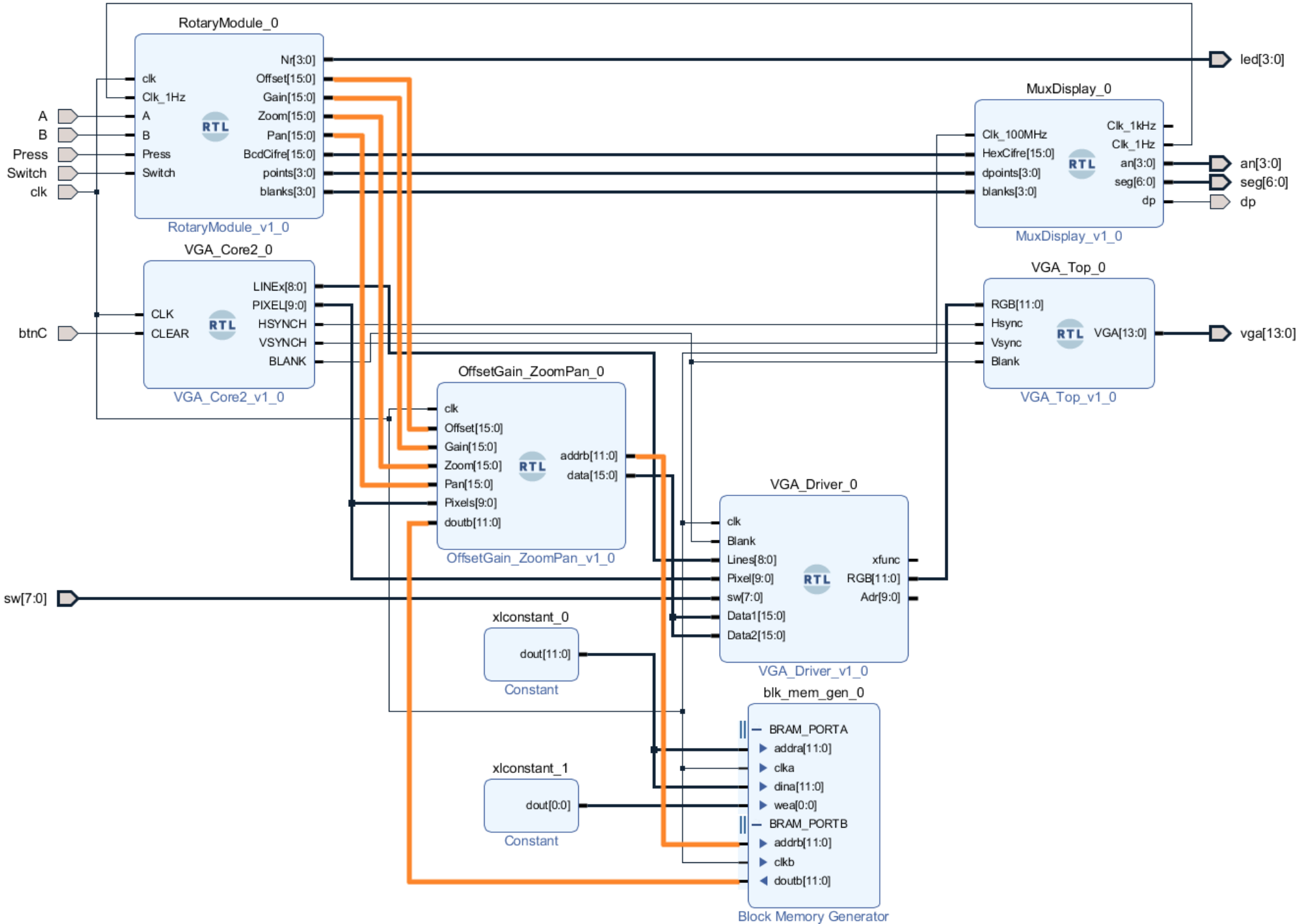
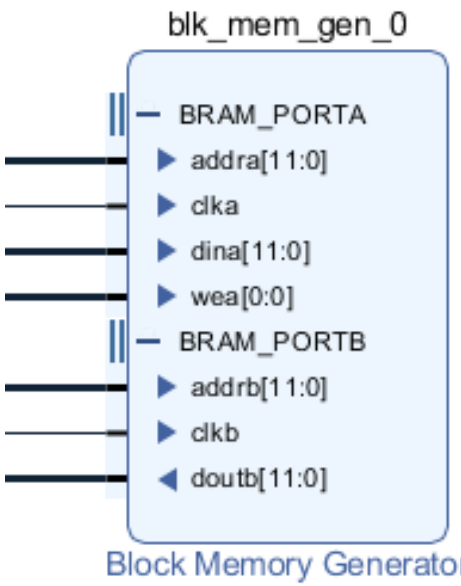
Scope4_2019


VGA_Core2_0
CLEAR
CLK
BLANK
HSYNCH
LINEx[8:0]
PIXEL[9:0]
VSYNCH
scope4_VGA_Core2_0_0
blk_mem_gen_0
addra[11:0]
addrb[11:0]
clka
clkb
dina[11:0]
wea
doutb[11:0]
scope4_blk_mem_gen_0_0
RotaryModule_0
A
B
Clk_1Hz
Press
Switch
clk
BcdCifre[15:0]
Gain[15:0]
Nr[3:0]
Offset[15:0]
Pan[15:0]
Zoom[15:0]
blanks[3:0]
points[3:0]
scope4_RotaryModule_0_0
OffsetGain_ZoomPan_0
Gain[15:0]
Offset[15:0]
Pan[15:0]
Pixels[9:0]
Zoom[15:0]
clk
doutb[11:0]
addrb[11:0]
data[15:0]
scope4_OffsetGain_ZoomPan_0_0
VGA_Driver_0
Blank
Data1[15:0]
Data2[15:0]
Lines[8:0]
Pixel[9:0]
clk
sw[7:0]
RGB[11:0]
scope4_VGA_Driver_0_0
VGA_Top_0
Blank
Hsync
RGB[11:0]
Vsync
VGA[13:0]
scope4_VGA_Top_0_0
MuxDisplay_0
Clk_100MHz
HexCifre[15:0]
blanks[3:0]
dpoints[3:0]
Clk_1Hz
an[3:0]
dp
seg[6:0]
scope4_MuxDisplay_0_0


RotaryModule_0
clk
Clk_1Hz
A
B
Press
Switch
Nr[3:0]
Offset[15:0]
Gain[15:0]
Zoom[15:0]
Pan[15:0]
BcdCifre[15:0]
points[3:0]
blanks[3:0]
RotaryModule_v1_0
A
B
Press
Switch
clk
led[3:0]
MuxDisplay_0
Clk_100MHz
HexCifre[15:0]
dpoints[3:0]
blanks[3:0]
Clk_1kHz
Clk_1Hz
an[3:0]
seg[6:0]
dp
MuxDisplay_v1_0
an[3:0]
seg[6:0]
dp
VGA_Core2_0
CLK
CLEAR
LINEx[8:0]
PIXEL[9:0]
HSYNCH
VSYNCH
BLANK
VGA_Core2_v1_0
btnC
VGA_Top_0
RGB[11:0]
Hsync
Vsync
Blank
VGA[13:0]
VGA_Top_v1_0
vga[13:0]
OffsetGain_ZoomPan_0
clk
Offset[15:0]
Gain[15:0]
Zoom[15:0]
Pan[15:0]
Pixels[9:0]
doutb[11:0]
addrb[11:0]
data[15:0]
OffsetGain_ZoomPan_v1_0
VGA_Driver_0
clk
Blank
Lines[8:0]
Pixel[9:0]
sw[7:0]
Data1[15:0]
Data2[15:0]
xfunc
RGB[11:0]
Adr[9:0]
VGA_Driver_v1_0
sw[7:0]
xlconstant_0
dout[11:0]
Constant
blk_mem_gen_0
BRAM_PORTA
addra[11:0]
clka
dina[11:0]
wea[0:0]
BRAM_PORTB
addrb[11:0]
clkb
doutb[11:0]
xlconstant_1
dout[0:0]
Constant
Block Memory Generator



Basic | Port A Options | Port B Options

Mode: Stand Alone

Memory Type: Simple Dual Port RAM

Basic | Port A Options | Port B Options | Other Options | Summary

Memory Size

Port A Width: 12 — Range: 1 to 4608 (bits)

Port A Depth: 4096 — Range: 2 to 1048576

The Width and Depth values are used for Write Operations in Port A

Operating Mode: No Change — Enable Port Type: Always Enabled

```
memory_initialization_radix=10;
memory_initialization_vector=
2048,
2063,
2079,
2094,
2109,
2125,
2140,
2155,
2170,
2186,
2201,
2216,
2231,
2246,
2262,
...
2026,
2032,
2037;
```

```
clear all; clc;
s = fopen('Sinus_v2019.coe','wb');  %opens the output file

fprintf(s,'%s\n','memory_initialization_radix=10;');
fprintf(s,'%s\n','memory_initialization_vector=');

t = (0: +10*pi/4096: 10*pi); % tiden
a = sin(t);                  % sinuskurve
x = exp(-t/30);              % exponentialkurve
a = 2000*a .*x + 2048;       % kurve med offset
a = round(a);                % rundet af

for n=1:length(a)-2
    fprintf(s, '%d', a(n));
    if (n == length(a)-2)
        fprintf(s,';\n'); % sidste linje slutter med ;
    else
        fprintf(s,',\n'); % ellers slut linje med ,
    end
end

fclose(s);
disp('Done');
```

Other Options

Memory Initialization

☑ Load Init File

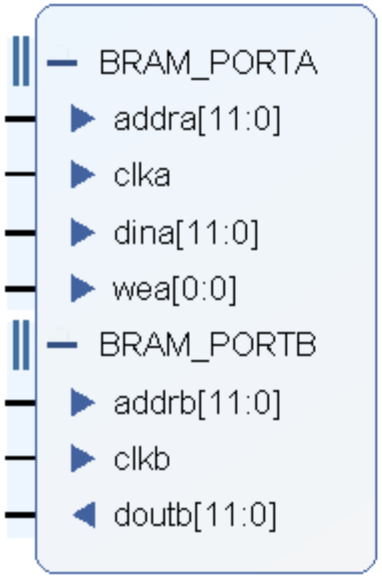
Coe File: ./../../../../XUP_E19/Lek_09/Scope4_2019/Sinus_v2019.coe | Browse | Edit

Block Memory Generator (8.4)

Documentation | IP Location

IP Symbol | Power Estimation

Show disabled ports



Component Name: blk_mem_gen_0

Basic | Port A Options | **Port B Options** | Other Options | Summary

Memory Size

Port B Width: 12

Port B Depth : 4096

The Width and Depth values are used for Read Operation in Port B

Operating Mode: Write First | Enable Port Type: Always Enabled

Port B Optional Output Registers

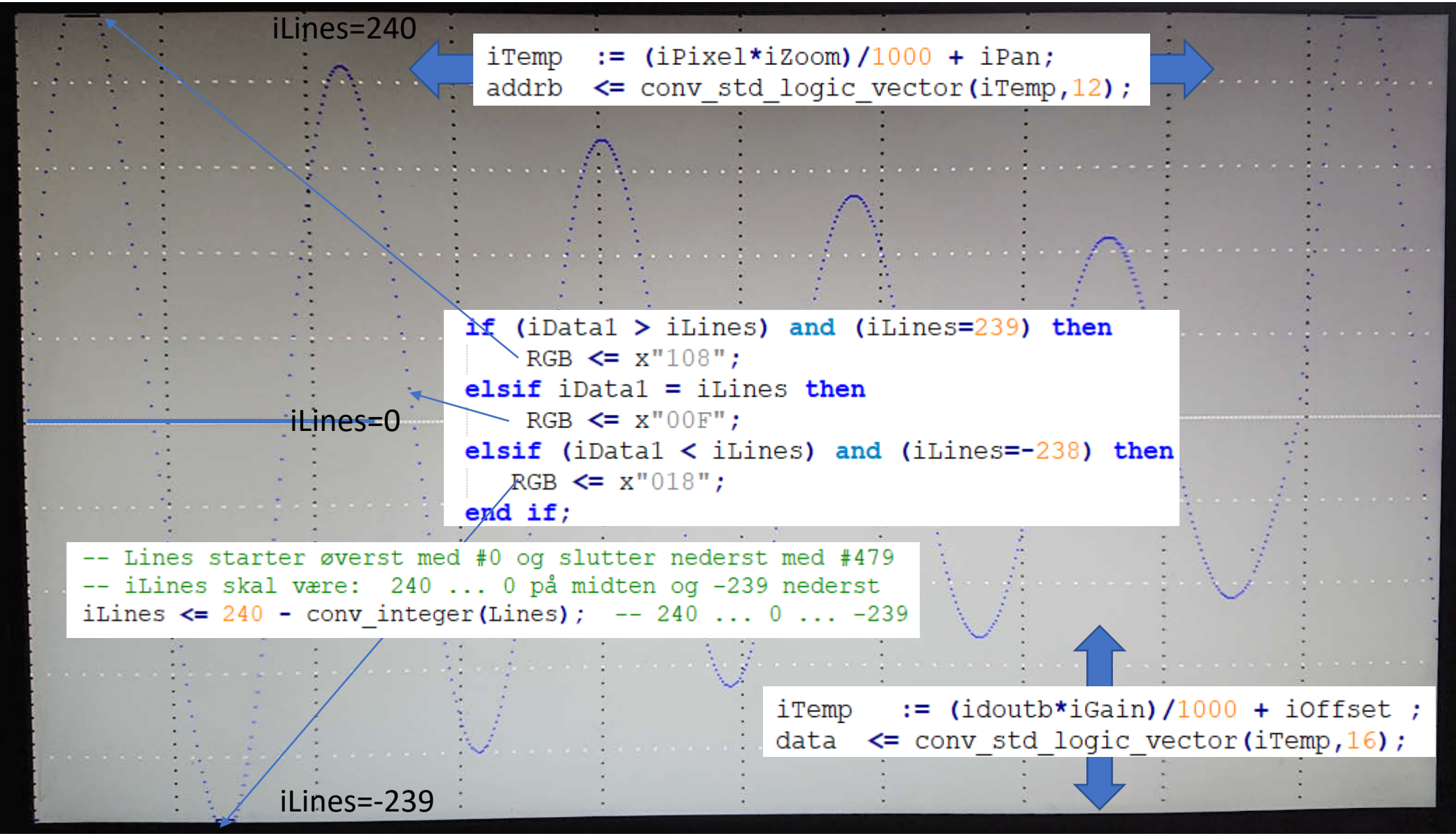
- [ ] Primitives Output Register
- [x] Core Output Register
- [ ] SoftECC Output Register
- [ ] REGCEB Pin

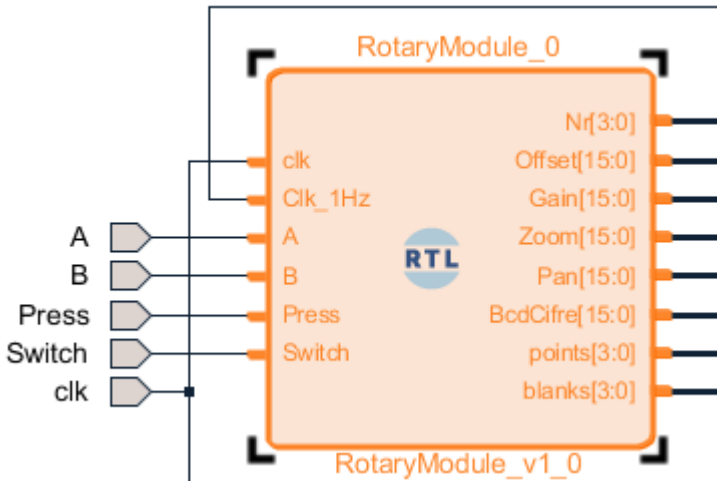
Port B Output Reset Options

- [ ] RSTB Pin (set/reset pin)
- Output Reset Value (Hex): 0
- [ ] Reset Memory Latch
- Reset Priority: CE (Latch or Register Enable)

READ Address Change B

- [ ] Read Address Change B


iLines=240
iTemp := (iPixel*iZoom)/1000 + iPan;
addrb <= conv_std_logic_vector(iTemp,12);
if (iData1 > iLines) and (iLines=239) then
RGB <= x"108";
elsif iData1 = iLines then
RGB <= x"00F";
elsif (iData1 < iLines) and (iLines=-238) then
RGB <= x"018";
end if;
iLines=0
-- Lines starter øverst med #0 og slutter nederst med #479
-- iLines skal være: 240 ... 0 på midten og -239 nederst
iLines <= 240 - conv_integer(Lines); -- 240 ... 0 ... -239
iTemp := (idoutb*iGain)/1000 + iOffset ;
data <= conv_std_logic_vector(iTemp,16);
iLines=-239

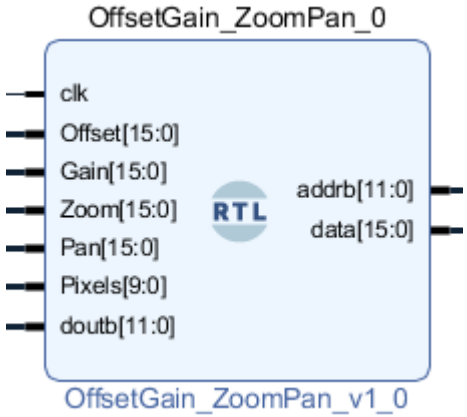



```
entity RotaryModule is
    Port ( clk :      in STD_LOGIC;
           Clk_1Hz:   in STD_LOGIC;
           A,B :      in STD_LOGIC;
           Press :    in STD_LOGIC;
           Switch :   in STD_LOGIC;
           Nr :      out STD_LOGIC_VECTOR (3 downto 0);
           Offset:   out STD_LOGIC_VECTOR (15 downto 0);
           Gain :    out STD_LOGIC_VECTOR (15 downto 0);
           Zoom :    out STD_LOGIC_VECTOR (15 downto 0);
           Pan :     out STD_LOGIC_VECTOR (15 downto 0);
           BcdCifre: out STD_LOGIC_VECTOR (15 downto 0);
           points:   out STD_LOGIC_VECTOR (3 downto 0);
           blanks:   out STD_LOGIC_VECTOR (3 downto 0));
end RotaryModule;
```

```
architecture Behavioral of RotaryModule is
    Signal Ax,Bx:     std_logic := '0';
    signal ABab:      std_logic_vector( 3 downto 0) := "0000";
    signal Messure:   integer := 0;

    signal SelNr:     STD_LOGIC_VECTOR (1 downto 0) := "00";
    signal Puls_1ms: std_logic := '0';

    signal HexCifre: STD_LOGIC_VECTOR (15 downto 0);

    --#########################################################################
    procedure Kupdate( signal Knx: inout integer;  -- Konstant som skal ændres
    end Kupdate;
    --#########################################################################
    signal Kn0:       integer :=     0;  -- Offset
    signal Kn1:       integer :=   100;  -- Gain/1000
    signal Kn2:       integer :=  2000;  -- Zoom/1000
    signal Kn3:       integer :=     0;  -- Pan
begin
```

```
--#############################################################################
procedure Kupdate( signal Knx: inout integer;  -- Konstant som skal ændres
                          Mess: integer;       -- Tiden siden sidste ændring
                          PLUS: boolean;       -- TRUE=> Delta skal lægges til
                          D100: integer;       -- Stor Deltaværdi
                          D10:  integer;       -- Mellem Deltaværdi (1 er default)
                          MIN:  integer;       -- Min værdi for Konstant
                          MAX:  integer) is    -- Max værdi for Konstant
    variable Delta: integer;
    variable vKnx:  integer;   -- Variable version af Knx
begin
    Delta := 1;              -- Default Delta
    vKnx  := Knx;            -- Signal bliver til variable
    if Mess< 30 then         -- Hvis der drejes hurtigt på knappen
        Delta := D100;
    elsif Mess< 120 then     -- Hvis der drejes lidt hurtigt
        Delta := D10;
    end if;
    if PLUS then
        vKnx := vKnx + Delta;  -- Læg Delta til
    else
        vKnx := vKnx - Delta;  -- Træk Delta fra
    end if;
    if vKnx>MAX then
        vKnx := MAX;           -- Sæt til Max værdi
    end if;
    if vKnx<MIN then
        vKnx := MIN;           -- Sæt til Min værdi
    end if;
    Knx <= vKnx;             -- Variable blive til Signal
end Kupdate;
--#############################################################################
```

```
Rotary_counter: process( clk)
    variable Messure_time: integer := 0;
    variable Mess:         integer;
    variable PM:           boolean; --Plus Minus
begin
    if rising_edge( clk) and Puls_1ms='1' then
        Messure_time  := Messure_time +1;
        ABab <= Ax & Bx & ABab(3) & ABab(2);
        case ABab is
            when "1011"| "1110" =>
                PM       := (ABab = "1011"); -- True når AAbb == "1011" eller False
                Mess     := Messure_time;
                Messure_time := 0;
                case SelNr is
                    when "00" => Kupdate( Kn0, Mess, PM, 200, 16, -2000, 2000); -- Offset
                    when "01" => Kupdate( Kn1, Mess, PM,  50, 16,    1, 2000); -- Gain
                    when "10" => Kupdate( Kn2, Mess, PM,  50, 10,    1, 8000); -- Zoom
                    when "11" => Kupdate( Kn3, Mess, PM, 100, 10,    0, 4096); -- Pan
                    when others => null;
                end case;
            when others => null;
        end case;
    end if;
end process;


Offset <= conv_std_logic_vector( Kn0, 16);
Gain   <= conv_std_logic_vector( Kn1, 16);
Zoom   <= conv_std_logic_vector( Kn2, 16);
Pan    <= conv_std_logic_vector( Kn3, 16);
```




```
entity OffsetGain_ZoomPan is
    Port ( clk :    in STD_LOGIC;
           Offset : in STD_LOGIC_VECTOR (15 downto 0);
           Gain :   in STD_LOGIC_VECTOR (15 downto 0);
           Zoom :   in STD_LOGIC_VECTOR (15 downto 0);
           Pan :    in STD_LOGIC_VECTOR (15 downto 0);
           Pixels : in STD_LOGIC_VECTOR (9 downto 0);
           addrb : out STD_LOGIC_VECTOR (11 downto 0);
           doutb :  in STD_LOGIC_VECTOR (11 downto 0);
           data :  out STD_LOGIC_VECTOR (15 downto 0));
end OffsetGain_ZoomPan;
```

```
architecture Behavioral of OffsetGain_ZoomPan is
begin
    ------------------------------------------------------------------------
    Lodret_Skalering: process( clk, Offset,Gain,doutb)
        variable idoutb:  integer;
        variable iOffset:   integer;
        variable iGain:    integer;
        variable iTemp:   integer;
    begin
        if rising_edge(clk) then
            idoutb  := conv_integer(doutb)-2048;
            iGain   := conv_integer(Gain);
            if offset(15)='0' then
                iOffset := conv_integer(Offset); -- Offset er positivt
            else
                iOffset := - conv_integer(not Offset+1); -- Skift fortegn
            end if;
            ------------------------------------------------
            iTemp   := (idoutb*iGain)/1000 + iOffset ;
            data  <= conv_std_logic_vector(iTemp,16);
        end if;
    end process;
```

```
    ------------------------------------------------------
    Vandret_skalering: process( clk, Zoom,Pan,Pixels)
        variable iPixel:  integer;
        variable iZoom:   integer;
        variable iPan:    integer;
        variable iTemp:   integer;
    begin
        if rising_edge(clk) then
            iPixel := conv_integer(Pixels);
            iZoom  := conv_integer(Zoom);
            iPan   := conv_integer(Pan);
            ------------------------------------------------
            iTemp  := (iPixel*iZoom)/1000 + iPan;
            addrb  <= conv_std_logic_vector(iTemp,12);
        end if;
    end process;
end Behavioral;
```

Opgave 1 – lav et kredsløb som kan fylde BRAM med data fra den analoge verden
Opgave 2 - Udvid scopet med en ekstra kanal.