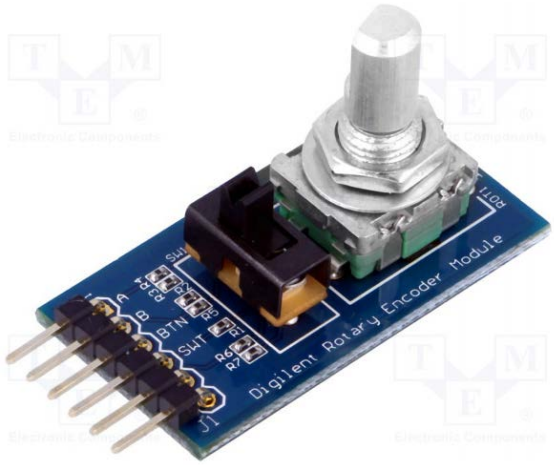
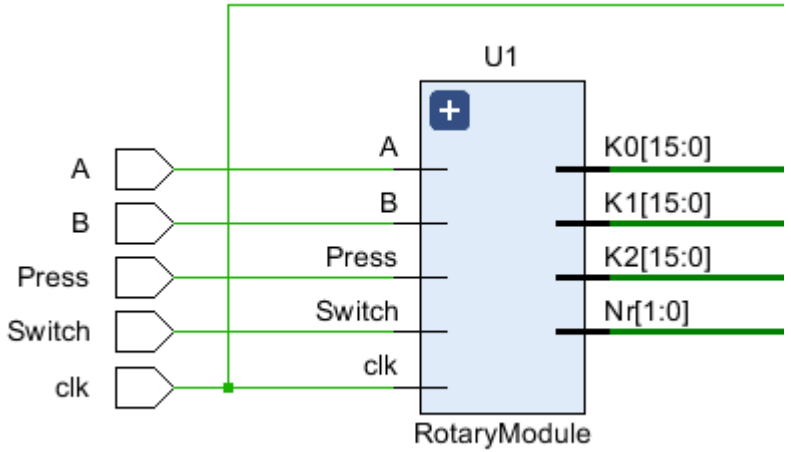
RotaryModule – Til brug i forbindelse med det digitale Scope

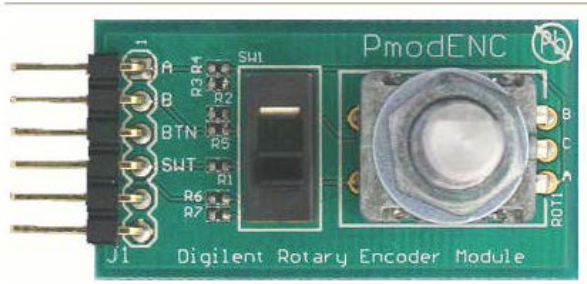


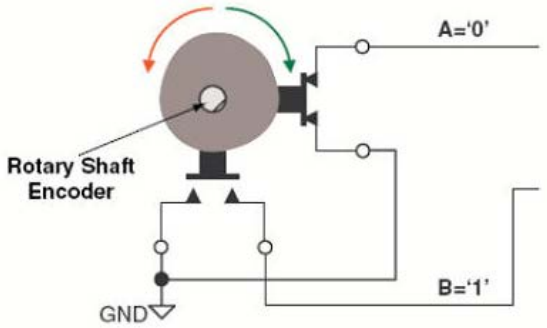


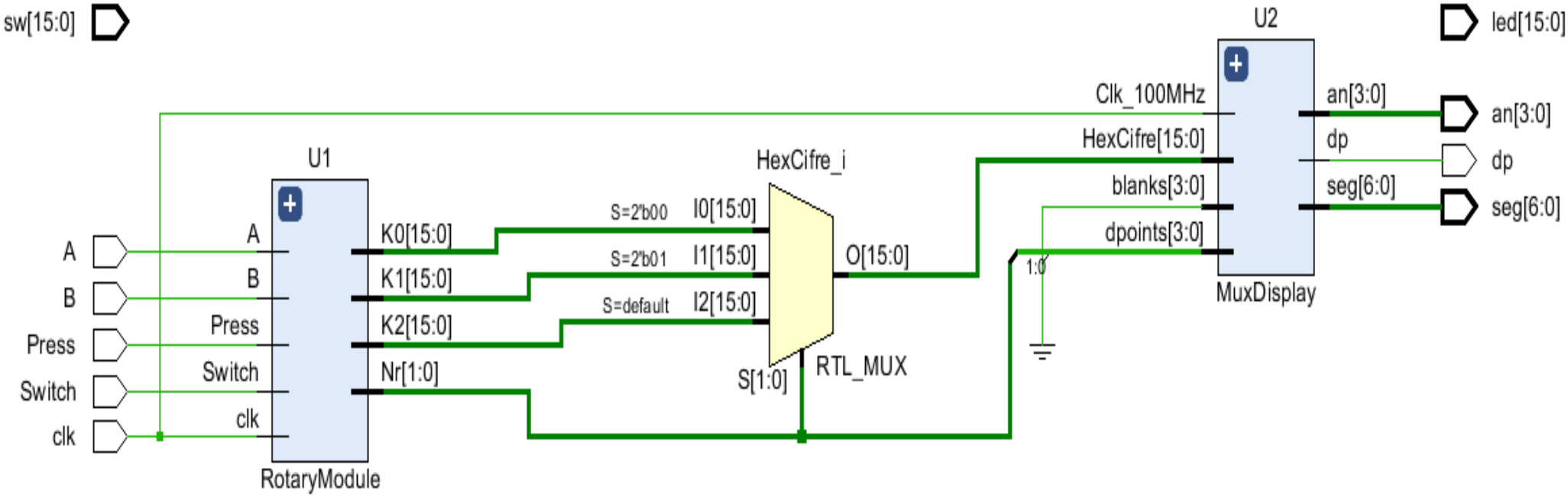
Opgave: Lav en komponent som kan ændre 3 ”konstanter” – K0, K1 og K2

- Nr angiver hvilken konstant som er valgt.
- Et tryk på ”press” ændrer Nr – 0,1,2, 0 osv
- Når der drejes skal konstanten der er valgt tælles op/ned.
- Hvis der drejes langsomt er det med +1/-1
- Hvis der drejes hurtigt er der med +16/-16 (og evt +256/-256)
  Der skal være prelfang og filter på A,B signaler

- Ekstra – Der er også behov for at hver konstant har en Min og Max værdi således at man altid holder sig inden for disse grænser.
- Måske er der også behov for at bruge Switch til en eller anden funktionalitet?
- Opdateringen som afhænger af hvor hurtigt man drejer knappen fungerer vist ikke optimalt – prøv at gøre den bedre.








```
with Nr select
    HexCifre <= K0 when "00",
                K1 when "01",
                K2 when others;

dpoints <= "00"& Nr;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity RotaryModule is
    Port ( clk : in STD_LOGIC;
           A,B : in STD_LOGIC;
           Press : in STD_LOGIC;
           Switch : in STD_LOGIC;
           Nr : out STD_LOGIC_VECTOR (1 downto 0);
           K0 : out STD_LOGIC_VECTOR (15 downto 0);
           K1 : out STD_LOGIC_VECTOR (15 downto 0);
           K2 : out STD_LOGIC_VECTOR (15 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 Kn0:      integer := 0;
    signal Kn1:      integer := 0;
    signal Kn2:      integer := 0;

    procedure Kupdate( signal Knx: inout integer;
```

```
27
28      procedure Kupdate( signal Knx: inout integer;
29                         signal Mess: in integer;
30                                PLUS: boolean) is
31          variable Delta: integer;
32      begin
33          Delta := 1;
34          if Mess< 30 then
35              Delta := 256;
36          elsif Mess< 120 then
37              Delta := 16;
38          end if;
39          if PLUS then
40              Knx <= Knx + Delta;
41          else
42              Knx <= Knx - Delta;
43          end if;
44      end Kupdate;
45
46  begin
47      Nr <= SelNr;
48      K0 <= conv_std_logic_vector( Kn0, 16);
49      K1 <= conv_std_logic_vector( Kn1, 16);
50      K2 <= conv_std_logic_vector( Kn2, 16);
```

En procedure i VHDL kan med fordel bruges til at ”genbruge” den samme kode.

Ved at definere parametre kan kodes arbejde på forskellige signaler.

Knx kan således være:
Kn0, **Kn1**, **Kn2**

Mess er tiden mellem to skift fra Rotary encoder

PLUS er true når der skal adderes til **Knx**

```
52      ------------------------------------------------------------
53      Scaler_1ms: process( clk)
54          variable Scale_100000: integer range 0 to 100000 := 0;
55      begin
56          if rising_edge( clk) then
57              Puls_1ms <= '0';
58              Scale_100000 := Scale_100000+1;
59              if Scale_100000>99999 then
60                  Scale_100000 := 1;
61                  Puls_1ms <= '1';
62              end if;
63          end if;
64      end process;
65
66      Select_nr: process (clk)
67          variable Prelfang: std_logic_vector( 7 downto 0) := "00000000";
68      begin
69          if rising_edge( clk) and Puls_1ms='1' then
70              Prelfang := Prelfang(6 downto 0) & Press;
71              if Prelfang="01111111" then
72                  case SelNr is
73                      when "00" => SelNr <= "01";
74                      when "01" => SelNr <= "10";
75                      when "10" => SelNr <= "00";
76                      when others => SelNr <= "00";
77                  end case;
78              end if;
79          end if;
80      end process;
```

Nedskalering af Clk til 1 msek pulser

Select_nr har til formål at ændre ”focus”, altså hvilken konstant der skal opdateres.
Bemærk at SelNr ændres når Prelfang indeholder ”01111111”

```
82      Rotaty_filter: Process (clk)
83          variable AB: std_logic_vector( 1 downto 0) := "00";
84      begin
85          if rising_edge( clk) then
86              AB := A&B;
87              case AB is
88                  when "00" => Ax <= '0';
89                  when "10" => Bx <= '0';
90                  when "01" => Bx <= '1';
91                  when "11" => Ax <= '1';
92                  when others => null;
93              end case;
94          end if;
95      end process;
96
97
98      Rotary_counter: process( clk)
99          variable Messure_time_plus:  integer := 0;
100         variable Messure_time_minus: integer := 0;
101     begin
102         if rising_edge( clk) and Puls_1ms='1' then
103             Messure_time_plus  := Messure_time_plus +1;
104             Messure_time_minus := Messure_time_minus+1;
105             ABab <= Ax & Bx & ABab(3) & ABab(2);
106             case ABab is
107                 when "1011" =>
```

Filtrering af AB signaler fra Rotary encoder

Denne process holder styr på AB skift og måler tiden imellem disse skift
- Overvej om det kan gøres anderledes

```
Rotary_counter: process( clk)
   variable Messure_time_plus:  integer := 0;
   variable Messure_time_minus: integer := 0;
begin
   if rising_edge( clk) and Puls_1ms='1' then
       Messure_time_plus  := Messure_time_plus +1;
       Messure_time_minus := Messure_time_minus+1;
       ABab <= Ax & Bx & ABab(3) & ABab(2);
       case ABab is
           when "1011" =>
                       Messure <= Messure_time_plus;
                       Messure_time_plus := 0;
                       case SelNr is
                           when "00" => Kupdate( Kn0, Messure, true);
                           when "01" => Kupdate( Kn1, Messure, true);
                           when "10" => Kupdate( Kn2, Messure, true);
                           when others => null;
                       end case;
           when "1110" =>
                       Messure <= Messure_time_minus;
                       Messure_time_minus := 0;
                       case SelNr is
                           when "00" => Kupdate( Kn0, Messure, false);
                           when "01" => Kupdate( Kn1, Messure, false);
                           when "10" => Kupdate( Kn2, Messure, false);
                           when others => null;
                       end case;
           when others => null;
       end case;
   end if;
end process;
```

Det vil være smart at kunne definere Min og Max værdier til de 3 konstanter.
Men det kræver flere parametre til **Kupdate** og ændringer i selve Kupdate koden.

```
procedure Kupdate( signal Knx: inout integer;
                   signal Mess: in integer;
                          PLUS: boolean) is
    variable Delta: integer;
begin
    Delta := 1;
    if Mess< 30 then
        Delta := 256;
    elsif Mess< 120 then
        Delta := 16;
    end if;
    if PLUS then
        Knx <= Knx + Delta;
    else
        Knx <= Knx - Delta;
    end if;
end Kupdate;
```

```
procedure Kupdate( signal Knx: inout integer;
                   signal Mess: in integer;
                          PLUS: boolean) is
    variable Delta: integer;
begin
    Delta := 1;
    if Mess< 30 then
        Delta := 256;
    elsif Mess< 120 then
        Delta := 16;
    end if;
    if PLUS then
        Knx <= Knx + Delta;
    else
        Knx <= Knx - Delta;
    end if;
end Kupdate;
```