

#####PwmOut#####

```
Library ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library ieee_proposed;
use ieee_proposed.fixed_pkg.all;
```

```
ENTITY PwmOut IS
    PORT (PwmUpdate : IN STD_LOGIC;
          Clock      : IN STD_LOGIC;
          reset      : IN STD_LOGIC;
          Direction   : IN STD_LOGIC;
          DutyPulses  : IN unsigned(13 DOWNTO 0);
          PWMOutFWD   : OUT STD_LOGIC;
          PWMOutBACK  : OUT STD_LOGIC);
END PwmOut;
```

ARCHITECTURE Behavior OF PwmOut IS

```
SIGNAL PulsesPWM           : unsigned(13 DOWNTO 0);
SIGNAL CountDuty           : unsigned(13 DOWNTO 0);
SIGNAL DutyTime            : unsigned(13 DOWNTO 0);
SIGNAL PwmDir              : STD_LOGIC;
SIGNAL PwmOutState         : STD_LOGIC;
```

```
BEGIN
PROCESS(Clock,Reset)      --Process for PWM signal out. (process som kjører i hovedprocessen)
BEGIN
IF (rising_edge(Clock) AND reset='0') THEN
CASE PwmOutState IS
    WHEN '0'=>            --State 0 Task: Update the number of DutyPulses.
    IF (PwmUpdate='1') THEN
        DutyTime<=DutyPulses;
        PwmDir<=Direction;
        PwmOutState<='1';
    END IF;
    WHEN '1'=>            --State 1 Task: Loop through the number of Pwm pulses.
    IF (PwmDir='0') THEN
    IF (CountDuty<=DutyPulses) THEN
        PwmOutFwd<='1';
        CountDuty<=resize(CountDuty+1,CountDuty);
    ELSIF (CountDuty<PulsesPwm) THEN
        PwmOutFwd<='0';
        CountDuty<=resize(CountDuty+1,CountDuty);
        PwmOutState<='0';
    END IF;
    ELSIF (PwmDir='1') THEN
```

```

    IF (CountDuty<=DutyPulses) THEN
        PwmOutBack<='1';
        CountDuty<=resize(CountDuty+1,CountDuty);
    ELSIF (CountDuty<PulsesPwm) THEN
        PwmOutBack<='0';
        CountDuty<=resize(CountDuty+1,CountDuty);
        CountDuty<=to_ufixed(0,CountDuty);
        PwmOutState<='0';
    END IF;
END IF;
END CASE;

```

```

ELSIF (rising_edge(Clock) and Reset='1') THEN
    PwmOutState<='0';
    PwmOutFwd<='0';
    PwmOutBack<='0';
    CountDuty<=to_ufixed(0,CountDuty);
END IF;
END PROCESS;

```

```

END Behavior;

```

```

#####PWM Controller code:#####

```

```

Library ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library ieee_proposed;
use ieee_proposed.fixed_pkg.all;

```

```

ENTITY PwmController IS
    PORT (
        ControllerCalcFinish : IN STD_LOGIC;
        dividedClock         : IN STD_LOGIC;
        Clock                 : IN STD_LOGIC;
        reset                 : IN STD_LOGIC;
        motorTorque           : IN sfixed(2 DOWNT0 -16);
        angleVel              : IN sfixed(4 DOWNT0 -12);
        Direction             : OUT STD_LOGIC;
        PwmUpdate             : OUT STD_LOGIC;
        DutyPulses            : OUT ufixed(13 DOWNT0 0)
    );
END PwmController;

```

```

ARCHITECTURE Behavior OF PwmController IS

```

```

    SIGNAL calcState          : STD_LOGIC;

```

```

SIGNAL torque                                : sfixed(2 DOWNT0 -16);
SIGNAL Resistanse                            : sfixed(8 DOWNT0 -3);
SIGNAL K                                     : sfixed(2 DOWNT0 -10);
SIGNAL MaxVoltage                           : sfixed(5 DOWNT0 0);
SIGNAL const10                              : sfixed(2 DOWNT0 0);
SIGNAL const9                               : sfixed(2 DOWNT0 0);
SIGNAL calc1,calc2,calc3,calc4,calc5        : sfixed(5 DOWNT0 -12);
SIGNAL DutyPeriode                          : sfixed(15 DOWNT0 0);
SIGNAL a                                     : sfixed(DutyPeriode'HIGH +1 DOWNT0
DutyPeriode'LOW);
SIGNAL PwmCalcState,divideCalc1             : STD_LOGIC_VECTOR(1 DOWNT0 0);
SIGNAL rotVel                               : sfixed(4 DOWNT0 -12);
SIGNAL PWMpulses                            : sfixed(20 DOWNT0 0);
SIGNAL DutyPeriode1                         : sfixed(2 DOWNT0 -16);

```

```

BEGIN
--Motor constants
K<=to_sfixed(0.4199,K);
Resistanse<=to_sfixed(119.1,Resistanse);
MaxVoltage<=to_sfixed(9,MaxVoltage);
const9<=to_sfixed(0,const9);
PWMpulses<=to_sfixed(1250,PWMpulses);

```

```

--instantiate
Pwm1 : entity work.PwmOut port map (
    PwmUpdate => PwmUpdate,
    Direction => Direction,
    reset    => reset,
    Clock    => Clock,
    PwmUpdate => PwmUpdate
);

```

```

PROCESS(dividedClock,Reset)                --Process for calculate PWM signal out.

```

```

BEGIN

```

```

IF (rising_edge(dividedClock) AND reset='0') THEN --Update the torque value

```

```

CASE PwmCalcState IS
WHEN "00"=>    --State 0 task: read in torque- and velocity value.
    IF (ControllerCalcFinish='1') THEN
        torque<=motorTorque;
        rotVel<=angleVel;
        PwmCalcState<="01";
        PwmUpdate<='0';
    ELSE
        PwmCalcState<="00";
    END IF;

```

```

WHEN "01"=>    --State 1 task:Check direction.
  IF (torque>=const9) THEN
    Direction<='0';
    PwmCalcState<="10";
  Else
    Direction<='1';
    PwmCalcState<="10";
  END IF;
WHEN "10"=>    --State 2 task:Calculate dutypulses.
  CASE calcState IS
    WHEN '0'=>
      CASE divideCalc1 IS
        WHEN "00"=>
          calc3<=resize(torque*resistanse,calc3);
          calc4<=resize(K*MaxVoltage,calc4);
          calc5<=resize(K*K*rotVel,calc5);
          divideCalc1<="01";
        WHEN "01"=>
          calc1<=resize(calc3/calc4,calc1);
          calc2<=resize(calc5/(Resistanse),calc2);
          divideCalc1<="10";
        WHEN "10"=>
          DutyPeriode1<=resize(calc1-calc2,DutyPeriode1);
          divideCalc1<="11";
        WHEN "11"=>
          DutyPeriode<=resize(DutyPeriode1*PWMpulses,DutyPeriode);
          divideCalc1<="00";
          calcState<='1';
        END CASE;
      WHEN '1'=>
        a<=abs(DutyPeriode);
        PwmCalcState<="10";
        calcState<='0';
      END CASE;
    WHEN "11"=>    --State 3 task:Write duty pulses.
      DutyPulses<=ufixed(a);
      PwmCalcState<="00";
      PwmUpdate<='1';

  END CASE;

```

```

ELSIF (rising_edge(dividedClock) and Reset='1') THEN
  PwmCalcState<="00";
  Direction<='0';
  DutyPulses<=to_ufixed(0,16,0);
  calcState<='0';
  divideCalc1<="00";
END IF;
END PROCESS;

```

END Behavior;

ERROR in Quartus:

Info: Command: quartus\_map --read\_settings\_files=on --write\_settings\_files=off PwmController -c PwmController

Info: Found 2 design units, including 1 entities, in source file pwmout/pwmout.vhd

Info: Found 1 design units, including 0 entities, in source file

c:/altera/91/quartus/libraries/vhdl/floatfixlib/fixed\_float\_types\_c.vhdl

Info: Found 2 design units, including 0 entities, in source file

c:/altera/91/quartus/libraries/vhdl/floatfixlib/fixed\_pkg\_c.vhdl

Info: Found 2 design units, including 1 entities, in source file pwmcontroller.vhd

Error (10309): VHDL Interface Declaration error in PwmController.vhd(48): interface object "PwmUpdate" of mode out cannot be read. Change object mode to buffer.

Error (10577): VHDL error at PwmController.vhd(48): actual port "PwmUpdate" of mode "out" cannot be associated with formal port "PwmUpdate" of mode "in"

Error (10600): VHDL error at PwmController.vhd(48): can't read value of interface object "PwmUpdate" of mode OUT

Error (10309): VHDL Interface Declaration error in PwmController.vhd(49): interface object "Direction" of mode out cannot be read. Change object mode to buffer.

Error (10577): VHDL error at PwmController.vhd(49): actual port "Direction" of mode "out" cannot be associated with formal port "Direction" of mode "in"

Error (10600): VHDL error at PwmController.vhd(49): can't read value of interface object "Direction" of mode OUT

Error (10347): VHDL error at PwmController.vhd(52): formal parameter "PwmUpdate" is already associated

Error (10309): VHDL Interface Declaration error in PwmController.vhd(52): interface object "PwmUpdate" of mode out cannot be read. Change object mode to buffer.

Error (10577): VHDL error at PwmController.vhd(52): actual port "PwmUpdate" of mode "out" cannot be associated with formal port "PwmUpdate" of mode "in"

Error (10600): VHDL error at PwmController.vhd(52): can't read value of interface object "PwmUpdate" of mode OUT

Error (10346): VHDL error at PwmController.vhd(47): formal port or parameter "DutyPulses" must have actual or default value

Error (10784): HDL error at PwmOut.vhd(12): see declaration for object "DutyPulses"

Error: Quartus II Analysis & Synthesis was unsuccessful. 12 errors, 0 warnings

Error: Quartus II Full Compilation was unsuccessful. 14 errors, 0 warnings