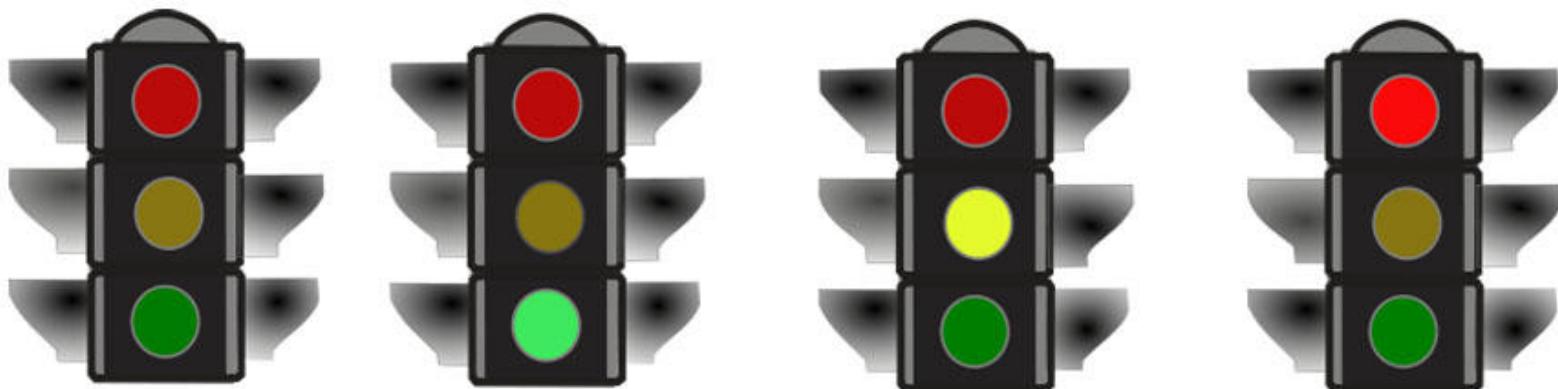


LYSKRYSS



OSLOMET

LAGET AV
Biraveen Nedunchelian
S341909

SAMMENDRAG

Formålet med denne oppgaven er å lage et lyskryss som skal kontrollere biltrafikk. Lyskrysset består av en hovedvei og en sidevei. For å kunne oppnå dette lagde jeg fire forskjellige VHDL-filer som til slutt ble koblet sammen. Jeg lagde en VHDL fil til følgende punkter

- En tilstandsmaskin som skal styre trafikklysene på hovedveien og sideveien.
- En timer som skal kunne styre transisjonene mellom de forskjellige tilstandene
- En teller som skal kunne brukes som en tidsreferanse for timeren
- En prescaler som skal kunne redusere oscillator frekvensen til 1Hz. 1Hz vil tilsvare en periode på et sekund.

Resultatet er en tilstandsmaskin som styrer lysene på hovedvei og sidevei, med en timer som bestemmer når lysene skifter farge. Prescaleren blir brukt til å redusere oscillasjons-frekvensen til kortet som blir butt ned til 1Hz, som vil tilsvare et sekund. Telleren blir brukt til å gi timer-en en tidsreferanse.

INNHOLD

LYSKRYSS	1
SAMMENDRAG	2
INNHOLD	3
TEORI	4
KRAVSPESIFIKASJONER	4
INNLEDNING, HENSIKT OG PROBLEMSTILLING	4
FREM GANGSMETODE	5
TILSTANDSMASKIN	5
TIMER	9
PRESCALER	10
TELLER	11
LYSKRYSS	12
MACBOOK	13
KONKLUSJON	13
KILDEHENVISNING	14

TEORI

For å kunne løse dette prosjektet kreves det basis kunnskap innenfor disse punktene.

- Forståelse av digital logikkdesign.
- Designkunnskap om forskjellige kombinasjons og sekvensløp.
- Metodikk for endelige tilstandsmaskiner er essensielt å ha kunnskap om.
- Oppgaveheftet jeg fikk utlevert av OSLO METROPOLITAN UNIVERSITY-STORBYUNIVERSITET.

KRAVSPESIFIKASJONER

Oppgaven ber oss spesifikt om å lage et blokk-skjerma som fungerer som et trafikklys. Dette skal gjøres ved hjelp av å sammen-flette fire VHDL-filer. Trafikklyset skal simulere en kunstig side og hovedvei som skifter lys. Kravet er at sluttproduktet skal ha en tidmaskin formulert som en VHDL-fil, timer, prescaler og en teller.

INNLEDNING, HENSIKT OG PROBLEMSTILLING

Oppgaven går ut på å lage et lyskryss ved hjelp av Quartus Prime. Lyskrysset som skal lages har to trafikklys, en for hovedvei(m_green/yellow/red) og en annen for sidevei (s_green/yellow/red).

For å lage lyskrysset må jeg først lage 4 forskjellige prosjekter og deretter sette de sammen inn i en ny en. Jeg skal derfor lage/generere VHDL-filer for de fire første delene. Jeg følger rekkefølgen som står i PDF-filen om prosjektet. Det vil si at rekkefølgen til prosjektet skal være slik

1. Lage tilstandsmaskin
2. Lage timer
3. Lage prescaler
4. Lage teller
5. Sammensetting av alle VHDL filene og sette opp lyskrysset.

Kretskortet som skal brukes til å løse denne oppgaven er DE1-SOC ALTERA.

FREM GANGSMÅTE

Løsning for alle er at man måtte lage en VHDL fil for hvert av punktene utenom del 1 og del 5. der lagde jeg en tilstandsmaskin og satt inn tabellen og deretter lot vi Quartus lage VHDL filen for oss. På del 5 satt jeg bare inn alle VHDL filene fra del 1-4 og kjørte en WAVEFORM. For hver av delene startes et nytt prosjekt utenom del 5, der må man sammen-flette de 4 tidligere delene inn i et prosjekt.

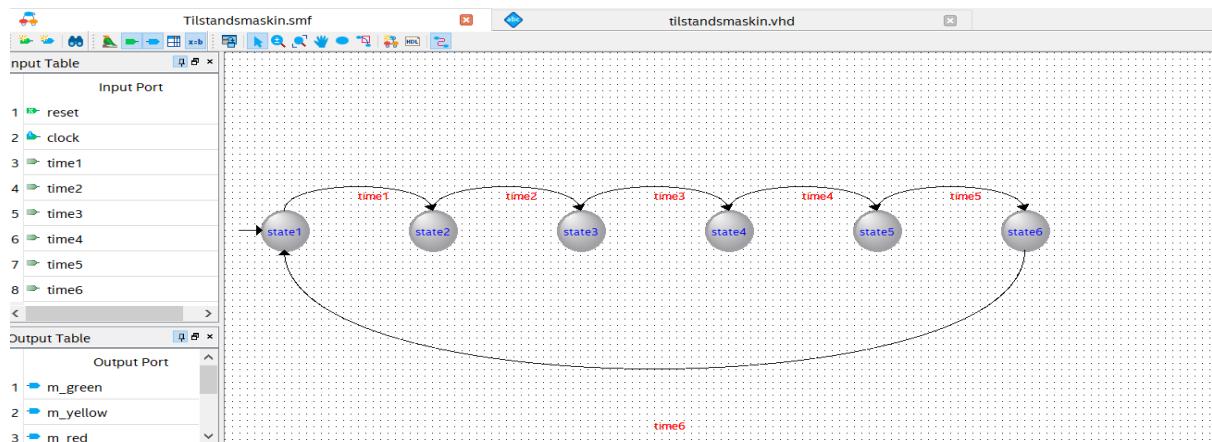
TILSTANDSMASKIN

En tilstandsmaskin er en matematisk modell som blir brukt til å designe dataprogrammer og digitale logiske kretser.

Til del 1 lagde jeg en tilstandsmaskin som jeg vil begrense til 6 forskjellige tilstander. Jeg fikk en tabell som jeg skulle følge etter. Tabellen ser slik ut:

Tilstand	Hovedvei Grønn	Hovedvei Gul	Hovedvei Rød	Sidevei Grønn	Sidevei Gul	Sidevei Rød	Varighet 5 bit	Varighet 6 bit
State1	1	0	0	0	0	1	12 sek	32 sek
State2	0	1	0	0	0	1	2 sek	3 sek
State3	0	0	1	0	1	1	2 Sek	3 Sek
State4	0	0	1	1	0	0	12 sek	20 sek
State5	0	0	1	0	1	0	2 sek	3 sek
State6	0	1	1	0	0	1	2 sek	3 sek

For å lage tilstandene brukes STATE TOOL og deretter lage transisjonspiler ved hjelp av TRANSITION TOOLS. Transisjonspilene måtte navngis, det ble gjort av å venstre-klikke på følgende pil, trykke seg inn på «property» og navngi selve pilen. Etter å ha gjort dette skal man inn til STATE MACHINE WIZARD og sette opp rekkesølgen til tilstandene og også inn verdien(tabellen) til hver av tilstandene. Det eneste som gjenstår nå er å generere VHDL kode. Det gjøres ved å trykke på HDL knappen ved siden av STATE MACHINE WIZARD og Velge VHDL. Nedenunder ser vi VHDL-koden Quartus har generert:



VHDL-KODE DEL 1:

Text Editor - C:/Users/birav/Desktop/LABOPPGAVER/DIGTEK/dig tek rapport del 5/lyskryss - lyskryss - [Prosjektdel1.vhd]

```

1 Copyright (C) 2018, Intel Corporation. All rights reserved.
2 -- Your use of Intel Corporation's design tools, logic functions
3 -- and other software and tools, and its AMPP partner logic
4 -- functions, and any output files from any of the foregoing
5 -- (including device programming or simulation files), and any
6 -- associated documentation or information are expressly subject
7 -- to the terms and conditions of the Intel Prime License
8 -- Subscription Agreement, the Intel Quartus Prime License Agreement,
9 -- the Intel FPGA IP License Agreement, or other applicable license
10 -- agreement, including, without limitation, that your use is for
11 -- the sole purpose of programming logic devices manufactured by
12 -- Intel and sold by Intel or its authorized distributors. Please
13 -- refer to the applicable agreement for further details.
14 --
15 -- Generated by Quartus Prime Version 18.1.0 Build 625 09/12/2018 SJ Lite Edition
16 -- Created on Fri Apr 24 14:14:29 2020
17 --
18 LIBRARY ieee;
19 USE ieee.std_logic_1164.all;
20
21 ENTITY Prosjektdel1 IS
22 PORT (
23   reset : IN STD_LOGIC := '0';
24   clock : IN STD_LOGIC;
25   time1 : IN STD_LOGIC := '0';
26   time2 : IN STD_LOGIC := '0';
27   time3 : IN STD_LOGIC := '0';
28   time4 : IN STD_LOGIC := '0';
29   time5 : IN STD_LOGIC := '0';
30   time6 : IN STD_LOGIC := '0';
31   m_green : OUT STD_LOGIC;
32   m_yellow : OUT STD_LOGIC;
33   m_red : OUT STD_LOGIC;
34   s_green : OUT STD_LOGIC;
35   s_yellow : OUT STD_LOGIC;
36   s_red : OUT STD_LOGIC;
37 );
38 END Prosjektdel1;
39
40 ARCHITECTURE BEHAVIOR OF Prosjektdel1 IS
41 TYPE type_fstate IS (state1,state2,state3,state4,state5,state6);
42 SIGNAL fstate : type_fstate;
43 SIGNAL reg_fstate : type_fstate;
44 BEGIN
45   PROCESS (clock,reset,reg_fstate)
46   BEGIN
47     IF (reset='0') THEN
48       fstate <= state1;
49     ELSEIF (clock='1' AND clock'event) THEN
50       ELSIF (clock='1' AND clock'event) THEN
51         fstate <= reg_fstate;
52       END IF;
53     BEGIN
54       PROCESS (fstate,time1,time2,time3,time4,time5,time6)
55 BEGIN
56   CASE fstate IS
57   WHEN state1 =>
58     IF ((time1 = '1')) THEN
59       reg_fstate <= state2;
60     ELSE
61       reg_fstate <= state1;
62     END IF;
63     m_green <= '0';
64     m_yellow <= '0';
65     m_red <= '0';
66     s_green <= '0';
67     s_yellow <= '0';
68     s_red <= '0';
69     WHEN state2 =>
70       IF ((time2 = '1')) THEN
71         reg_fstate <= state3;
72     ELSE
73       reg_fstate <= state2;
74     END IF;
75     m_green <= '1';
76     s_green <= '0';
77     m_red <= '0';
78     s_yellow <= '0';
79     WHEN state3 =>
80       IF ((time3 = '1')) THEN
81         reg_fstate <= state4;
82     ELSE
83       reg_fstate <= state3;
84     END IF;
85     m_green <= '0';
86     s_green <= '0';
87     m_red <= '0';
88     s_yellow <= '1';
89     WHEN state4 =>
90       IF ((time4 = '1')) THEN
91         reg_fstate <= state5;
92     ELSE
93       reg_fstate <= state4;
94     END IF;
95     m_green <= '0';
96     s_green <= '0';
97     m_red <= '1';
98     s_yellow <= '0';
99     WHEN state5 =>
100       IF ((time5 = '1')) THEN
101         reg_fstate <= state6;
102     ELSE
103       reg_fstate <= state5;
104     END IF;
105     m_green <= '0';
106     s_green <= '0';
107     m_red <= '1';
108     s_yellow <= '0';
109     WHEN state6 =>
110       IF ((time6 = '1')) THEN
111         reg_fstate <= state1;
112     ELSE
113       reg_fstate <= state6;
114     END IF;
115     m_green <= '0';
116     s_green <= '1';
117     m_red <= '0';
118     s_red <= '0';
119     s_yellow <= '1';
120     WHEN OTHERS =>
121       IF ((others = 'X')) THEN
122         reg_fstate <= state1;
123     ELSE
124       reg_fstate <= state6;
125     END IF;
126     m_green <= '0';
127     s_green <= '0';
128     m_red <= '1';
129     s_red <= '0';
130     s_yellow <= '1';
131     WHEN others =>
132       IF ((others = 'X')) THEN
133         reg_fstate <= state1;
134     ELSE
135       reg_fstate <= state6;
136     END IF;
137     m_green <= '0';
138     s_green <= '0';
139     m_red <= '0';
140     s_red <= '0';
141     s_yellow <= '0';
142     WHEN others =>
143       IF ((others = 'X')) THEN
144         reg_fstate <= state1;
145     ELSE
146       reg_fstate <= state6;
147     END IF;
148   END CASE;
149   END PROCESS;
150 END BEHAVIOR;
151 
```

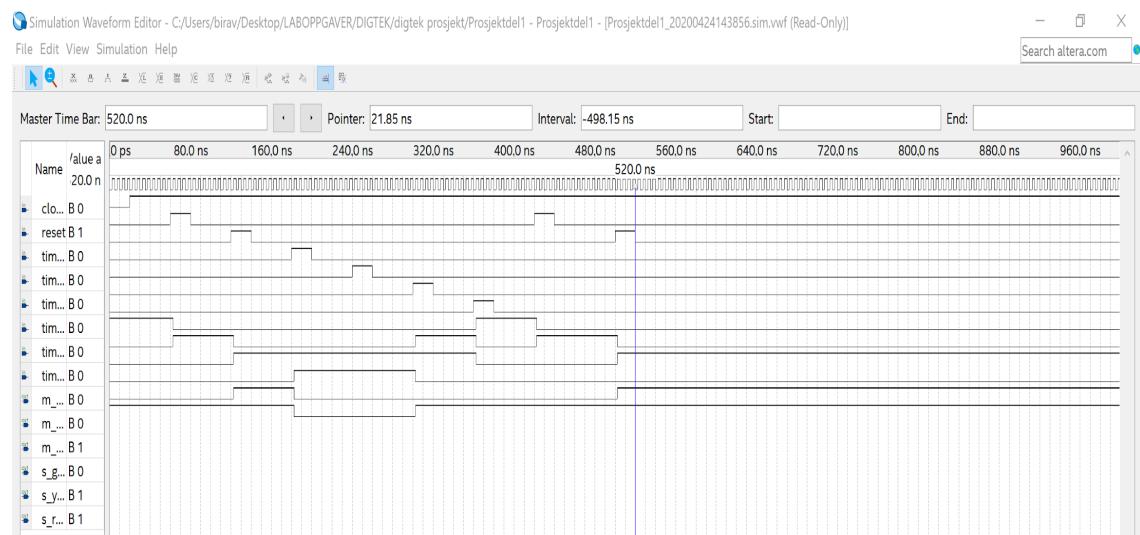
Skriv her for å søke

Text Editor - C:/Users/birav/Desktop/LABOPPGAVER/DIGTEK/dig tek rapport del 5/lyskryss - lyskryss - [Prosjektdel1.vhd]

```

49
50   ELSIF (clock='1' AND clock'event) THEN
51     fstate <= reg_fstate;
52   END IF;
53 END PROCESS;
54
55 BEGIN
56   CASE fstate IS
57   WHEN state1 =>
58     IF ((time1 = '1')) THEN
59       reg_fstate <= state2;
60     ELSE
61       reg_fstate <= state1;
62     END IF;
63     m_green <= '0';
64     m_yellow <= '0';
65     m_red <= '0';
66     s_green <= '0';
67     s_yellow <= '0';
68     s_red <= '0';
69     WHEN state2 =>
70       IF ((time2 = '1')) THEN
71         reg_fstate <= state3;
72     ELSE
73       reg_fstate <= state2;
74     END IF;
75     m_green <= '1';
76     s_green <= '0';
77     m_red <= '0';
78     s_yellow <= '0';
79     WHEN state3 =>
80       IF ((time3 = '1')) THEN
81         reg_fstate <= state4;
82     ELSE
83       reg_fstate <= state3;
84     END IF;
85     m_green <= '0';
86     s_green <= '0';
87     m_red <= '0';
88     s_yellow <= '1';
89     WHEN state4 =>
90       IF ((time4 = '1')) THEN
91         reg_fstate <= state5;
92     ELSE
93       reg_fstate <= state4;
94     END IF;
95     m_green <= '0';
96     s_green <= '0';
97     m_red <= '1';
98     s_yellow <= '0';
99     WHEN state5 =>
100       IF ((time5 = '1')) THEN
101         reg_fstate <= state6;
102     ELSE
103       reg_fstate <= state5;
104     END IF;
105     m_green <= '0';
106     s_green <= '1';
107     m_red <= '0';
108     s_red <= '0';
109     s_yellow <= '1';
110     WHEN state6 =>
111       IF ((time6 = '1')) THEN
112         reg_fstate <= state1;
113     ELSE
114       reg_fstate <= state6;
115     END IF;
116     m_green <= '0';
117     s_green <= '0';
118     m_red <= '1';
119     s_red <= '0';
120     s_yellow <= '0';
121     WHEN OTHERS =>
122       IF ((others = 'X')) THEN
123         reg_fstate <= state1;
124     ELSE
125       reg_fstate <= state6;
126     END IF;
127     m_green <= '0';
128     s_green <= '0';
129     m_red <= '1';
130     s_red <= '0';
131     s_yellow <= '1';
132     WHEN others =>
133       IF ((others = 'X')) THEN
134         reg_fstate <= state1;
135     ELSE
136       reg_fstate <= state6;
137     END IF;
138     m_green <= '0';
139     s_green <= '0';
140     m_red <= '0';
141     s_red <= '0';
142     s_yellow <= '0';
143     WHEN others =>
144       IF ((others = 'X')) THEN
145         reg_fstate <= state1;
146     ELSE
147       reg_fstate <= state6;
148     END IF;
149   END CASE;
150   END PROCESS;
151 END BEHAVIOR;
152 
```

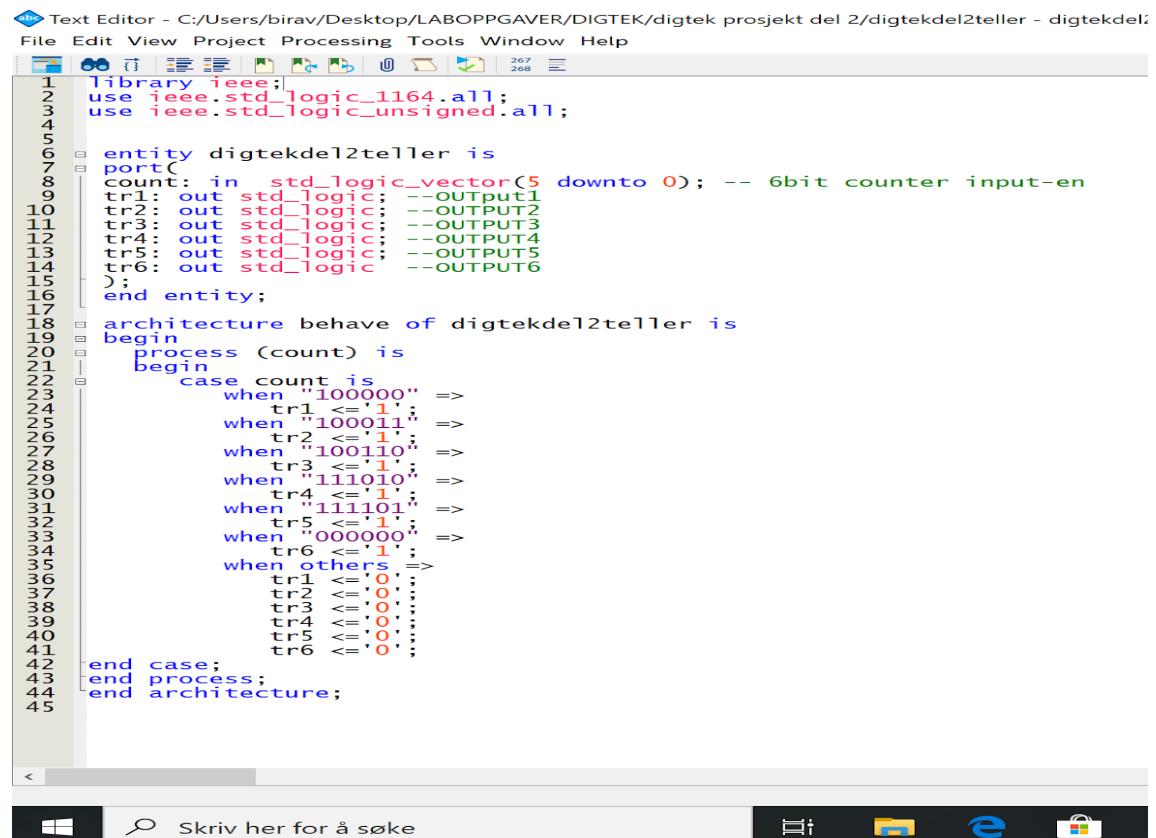
Skriv her for å søke

WAVEFORM DEL1:

TIMER

Del 2 handler om å lage en timer, denne skal sørge for at tilstandsmaskinen skrifter tilstand til riktig tidspunkt. Det startes med å lage en VHDL fil med 6 utganger som jeg kalte for tr1, tr2, tr3, tr4, tr5 og tr6. dette er til for at den skal svare henholdsvis fra time1 til time6. Under har jeg tatt med et skjermbilde av VHDL-koden og WAVEFORMEN til samtlige VHDL kode etter å ha kjørt en simulasjon. Fra og med dette stadiet satt jeg alt VHDL filene som

VHDL-KODE DEL2:

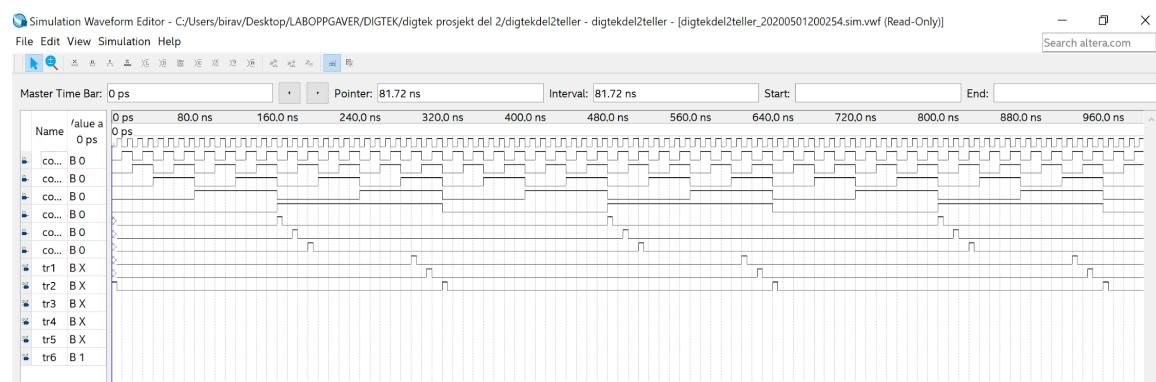


```

1  Text Editor - C:/Users/birav/Desktop/LABOPPGAVER/DIGTEK/digtek prosjekt del 2/digtekdel2teller - digtekdel2teller.vhd
2  File Edit View Project Processing Tools Window Help
3
4
5
6  Library ieee;
7  use ieee.std_logic_1164.all;
8  use ieee.std_logic_unsigned.all;
9
10
11
12
13
14
15
16
17
18  entity digtekdel2teller is
19  port(
20    count: in std_logic_vector(5 downto 0); -- 6bit counter input-en
21    tr1: out std_logic; --OUTPUT1
22    tr2: out std_logic; --OUTPUT2
23    tr3: out std_logic; --OUTPUT3
24    tr4: out std_logic; --OUTPUT4
25    tr5: out std_logic; --OUTPUT5
26    tr6: out std_logic --OUTPUT6
27  );
28  end entity;
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

WAVEFORM DEL2:



PRESCALER

Del 3 handler om å lage en prescaler.

En prescaler er en elektronisk telle krets som brukes til å redusere høye elektriske signaler til lave ved hjelp av heltallsdeling. Formålet med prescaler er å la tidtakeren klokkes i takt på en måte som brukeren ønsker. Prescaleren vil ta den grunnleggende klokkefrekvensen, dele den inn med en viss verdi også «mate» den inn i timeren.

Prescaler som er laget har en tidsstyring med frekvens på 1Hz, dette vil passe godt med bit-telleren.

Dette er fordi man skal bruke en 50MHz klokke på kortet, man derfor trenger en prescaler på 1 Hz.

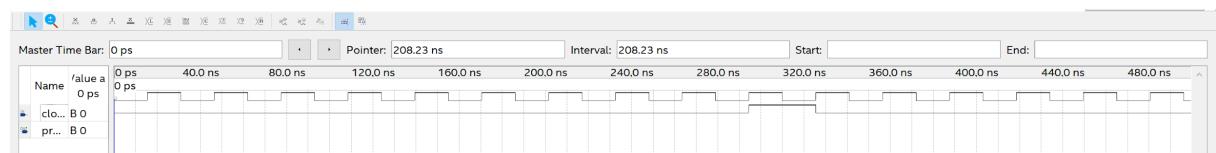
VHDL-KODE DEL3:

```

1  Library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5  entity Prescaler is
6  generic (prescalerValue: Integer :=10);
7  port (
8      clk_50: in std_logic;
9      prescaler_out: out std_logic
10     );
11
12 end Prescaler;
13
14 architecture Behavioral of Prescaler is
15
16  signal prescaler_value : integer :=1;
17  signal temp : std_logic :='0';
18
19 begin
20
21 process(clk_50)
22
23 begin
24
25 if clk_50'event and clk_50 = '1' then
26
27     prescaler_value <= prescaler_value + 1;
28
29 if prescaler_value = 10 then
30
31     temp <= not temp;
32     prescaler_value <= 1;
33
34 else temp<= '0';
35
36 end if;
37
38 end if;
39
40 end process;
41
42 prescaler_out <= temp;
43
44 end Behavioral;

```

WAVEFORM DEL3:

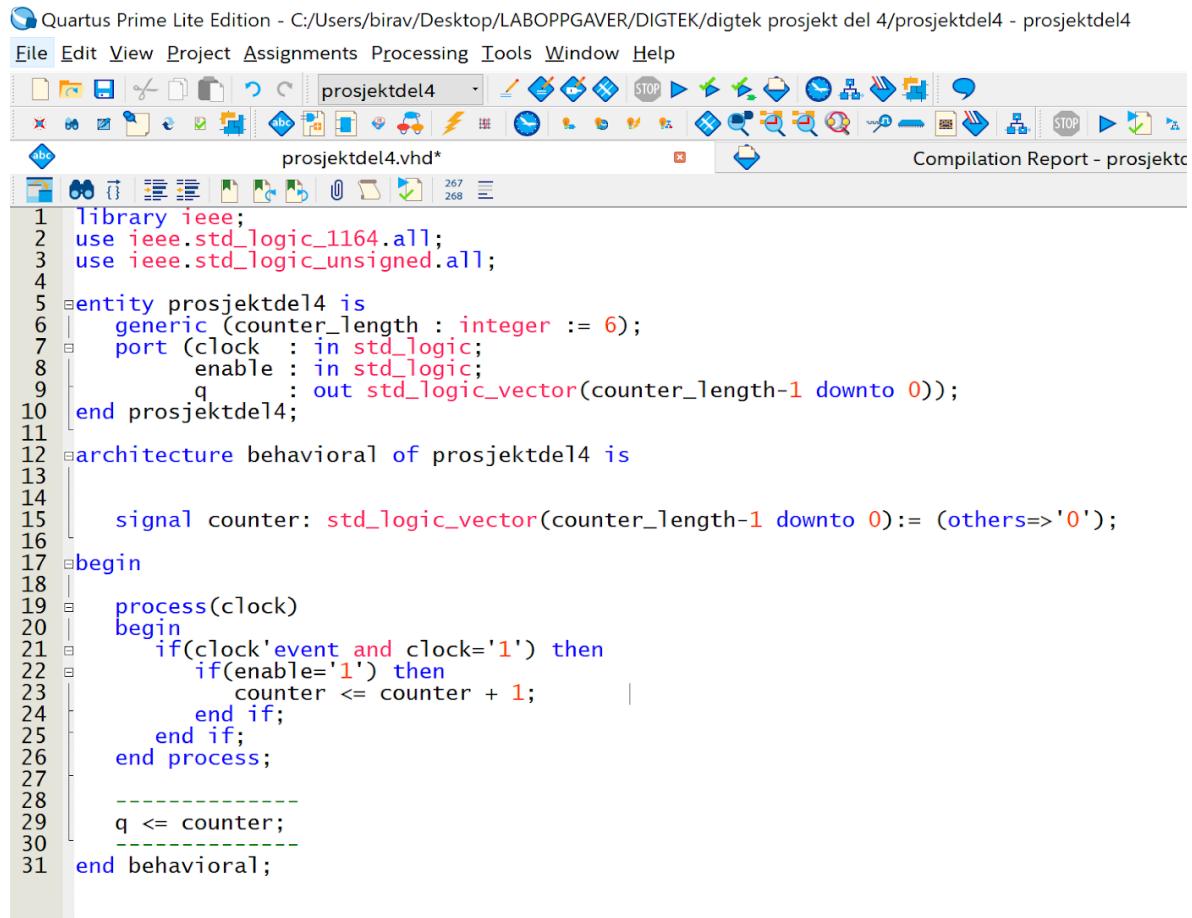


TELLER

DEL 4 handler om å lage en teller.

Telleren bruker 50MHz klokkeinngangen mens Enebler-en (?) vil gjøre at telleren kun kan øke sin verdi hver gang Enable er høy. Dette vil få telleren til å skifte verdi hvert sekund.

VHDL KODE DEL4



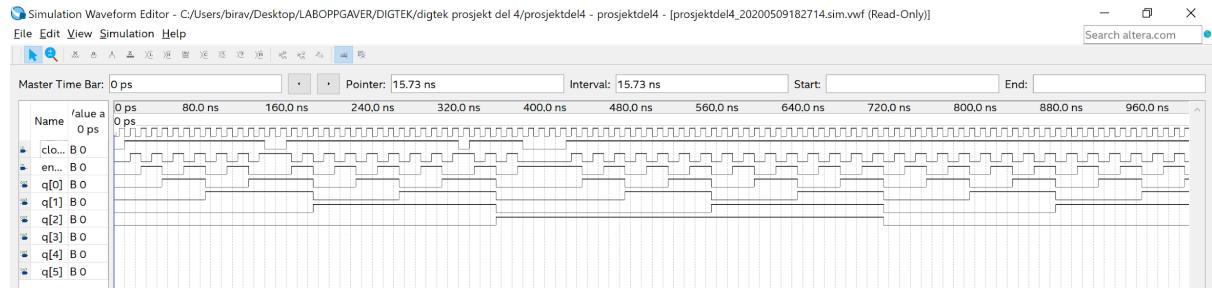
Quartus Prime Lite Edition - C:/Users/birav/Desktop/LABOPPGAVER/DIGTEK/digtek prosjekt del 4/prosjektdel4 - prosjektdel4

```

File Edit View Project Assignments Processing Tools Window Help
projektdel4
projektdel4.vhd*                                         Compilation Report - projektdel4
267 268
1 Library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4
5 entity projektdel4 is
6   generic (counter_length : integer := 6);
7   port (clock : in std_logic;
8         enable : in std_logic;
9         q      : out std_logic_vector(counter_length-1 downto 0));
10 end projektdel4;
11
12 architecture behavioral of projektdel4 is
13
14   signal counter: std_logic_vector(counter_length-1 downto 0):= (others=>'0');
15
16 begin
17
18   process(clock)
19   begin
20     if(clock'event and clock='1') then
21       if(enable='1') then
22         if(counter <= counter + 1);
23         else
24           counter <= counter + 1;
25         end if;
26       end if;
27     end process;
28
29   -----
30   q <= counter;
31 end behavioral;

```

WAVEFORM DEL 4:

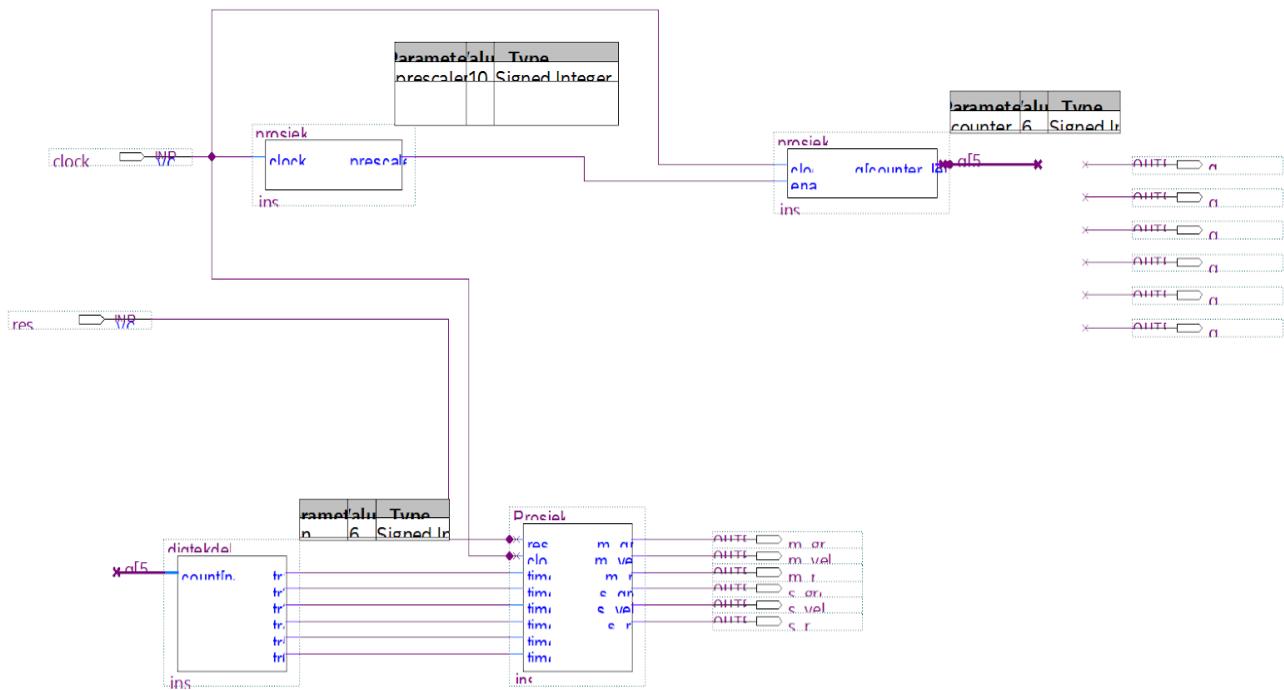


LYSKRYSS

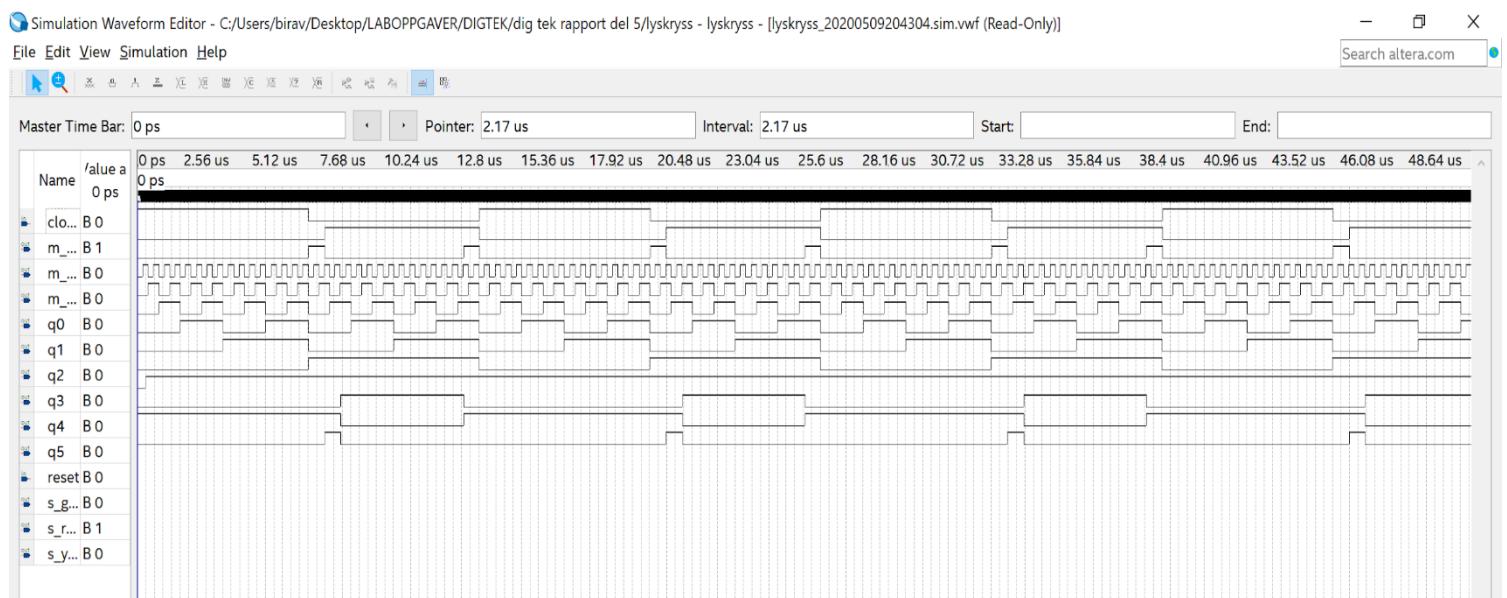
Del 5 handler om å flette sammen av alle VHDL-filene og endelig få lagd det virtuelle lyskrysset.

Nå har vi kommet til prosjektets endepunkt og det er på tide å endelig sette sammen filene. For å gjøre det skal man lage et nytt prosjekt og importere inn alle VHDL filene. Deretter gjøre om VHDL filene om til symboler ved å gå til File→Create/Update→Create Symbol Files For Current File. Nå gjenstår det bare å putte det i et blokkdiagram og kompile for å se om alt er i orden, og deretter kjøre en waveform. I waveformen skal SET END TIME settes på 50us, reset skal settes på høyt (utenom i starten, da skal den være lav) også satt jeg clock_50 på en periode på 20ns.

BLOKKDIAGRAM DEL 5:



WAVEFORM DEL 5:



MACBOOK

Quartus ser ut til å ha et skaleringsproblem når det kommer til datamaskiner laget av Apple. Dette var også et av hindrene jeg støtte på under prosjektet.

KONKLUSJON

Nå som kravene som var stilt av kravspesifikasjonene nå er gjennomført. Det har nå blitt lagd et trafikklys som styrer en hovedvei og en sidevei. Mine hindre under prosjektet var forståelsen av hvordan VHDL fungerer i sin helhet og skaleringsproblemet nevnt tidligere og vist i illustrasjonene mine. For en framtidig referanse ville det vært en fordel for meg å kunne ha mer forståelse om hvordan VHDL fungerer, men nå sitter jeg igjen med en bredere grunnforståelse av hvordan en tidmaskin, timer, prescaler og teller fungerer.

KILDEHENVISNING:

<https://en.wikipedia.org/wiki/Prescaler>

<https://www.geeksforgeeks.org/counters-in-digital-logic/>

<https://de.mathworks.com/help/msblk/ref/dualmodulusprescaler.html>

https://no.wikipedia.org/wiki/Endelig_tilstandsmaskin

Digital teknikk 2. utgave Knut Harald Nygaard

- - Side 1
- - Side 135
- - Side 203