Versjon 1.0/07.09.2015

INF3430/4431 Høsten 2015

Laboppgave 2

VHDL-programmering
Funksjoner og prosedyrer/Bibliotek
Styring av sjusegmenter

Innledning.

Målene med denne laboppgaven er

- å lære om subprogrammer og biblioteker i VHDL
- å lære å lage *hardware*-beskrivelser i VHDL, samt å lage testbenker for å simulere disse.

Laboppgaven er delt opp i ett antall deloppgaver. Den første deloppgaven er en øvelse i å lage subprogrammer og samle disse i et bibliotek.
De neste deloppgavene går ut på å designe kode som er rettet mot testkortet. De er organisert med stigende vanskelighetsgrad. For hver deloppgave skal du følge samme designflyt som i laboppgave 1.

Oppgave 1

I denne deloppgaven skal vi se på subprogrammer (function og procedure).

a).

Simuler den vedlagte koden i pargen.vhd og tb_pargen.vhd hvor det leses inn to 16-bit vectorer og genereres et paritetssignal. I koden brukes både metoden som er brukt i læreboka (se Zwo kap. 4) og den mer vanlige xor baserte metoden. Sammenlign disse to metodene ved å synthetisere pargen.vhd og se på resultatet med RTL og Technology view'erene i ISE. Er det noen forskjell mellom forbruk av LUT'er?. Hvilken metode har kortest forsinkelse?

b).

Koden i pargen.vhd skal nå endres til å bruke to funksjoner med samme navn (bruk overloading av funksjoner i VHDL) til å generere pariteten for hver av de to inndata signalene.

c).

Koden i pargen.vhd fra deloppgave a. skal nå endres til å bruke en prosedyre med samme navn som navnet valgt i deloppgave b. (bruk overloading av funksjoner i VHDL) til å generere pariteten i en procedyre.

d).

Funksjonene i deloppgave b. skal nå flyttes over i en subprog_pck package. Pargen.vhd fra deloppgave b.skal nå endres til å bruke funksjonene fra subprog_pck package.

e).

Prosedyren i deloppgave c. skal nå også flyttes over i subprog_pck package. Pargen.vhd fra deloppgave c.skal nå endres til å bruke prosedyren fra subprog_pck package.

Godkjenning:

VHDL kildefil for de enkelte delspørsmålene

Framgangsmåte ved VHDL-koding.

Arbeidet med å lage en *hardware*-beskrivelse i VHDL kan deles inn i fire hovedmomenter:

- Prøv å danne seg et korrekt bilde av problemstillingen.
- Les oppgaveteksten nøye
- Studer dokumentasjonen til testkortet.
- Lag et blokkskjema over det som skal lages med innganger og utganger til hver modul.

Strukturering av en løsning.
Den konstruksjonen du skal implementere i FPGA-en har et ytre grensesnitt mot de andre komponentene på testkortet, og en indre struktur.
Det ytre grensesnittet, i form av innsignaler og utsignaler, tilsvarer *entity*-deklarasjonen i en VHDL-beskrivelse. Den indre strukturen tilsvarer VHDL-beskrivelsens *architecture*-del.
Det er ofte hensiktsmessig å dele opp den indre strukturen i en *dataveis*truktur og en *kontrolls*truktur.

Dataveistrukturen inneholder elementer som registre, addere, multipleksere som er sammenkoblet med databusser. Denne strukturen er velegnet til å beskrives i blokkskjema. Hvis strukturen er kompleks, lønner det seg å dele den opp i mindre blokker

Kontrollogikkens innsignaler og utsignaler kan beskrives i blokkskjema sammen med dataveistrukturen. Den indre oppbyggingen beskrives best i form av: sannhetsverditabeller, boolske ligninger og tilstandsdiagrammer. Deler av arbeidet med å strukturere konstruksjonen kan med fordel utføres ved å bruke muligheten til å skrive kommentarer i VHDL-kildefilen.

Kode den strukturerte løsningen i VHDL.
Med utgangspunkt i en veldokumentert struktur skal det normalt være en grei jobb å lage en fungerende kode.

Navneregler for VHDL design-filer

For å identifisere VHDL-kildefiler er det lurt å ha visse navnekonvensjoner og regler for hva de forskjellige filene skal inneholde.
I alle design dere gjør fra og med deloppgave 2 skal vi lagre entitet og arkitektur i forskjellige filer og følge navnereglene gitt av følgende tabell:

Table 1. Navneregler for VHDL-kildefiler

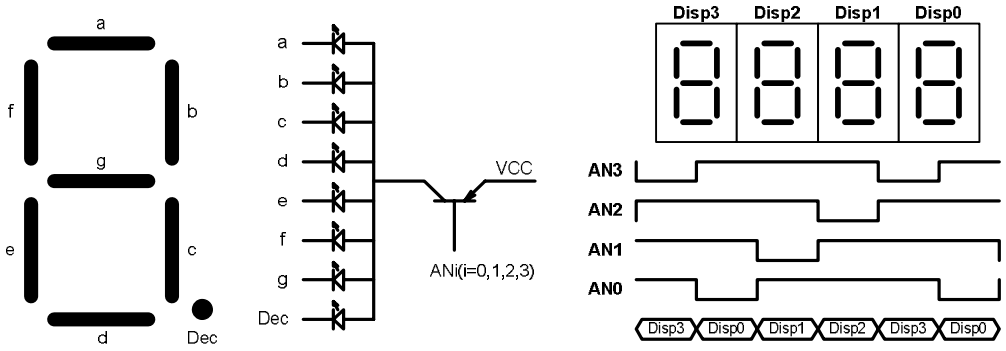
Fil-innhold	**Filnavn**
Entity	<design_unit>_ent.vhd
RTL architecture	<design_unit>_rtl.vhd
STR architecture	<design unit>_str.vhd
Behavorial architecture (simuleringsmodell ikke ment for syntese)	<design_unit>_beh.vhd
Configuration	<design_unit>_cfg.vhd
package def	<library_unit>_pkg.vhd
package body	<library_unit>_bdy.vhd
Testbench	tb_<design_unit>.vhd
testbench config	tb_<design_unit>_cfg.vhd

<design_unit> og <library_unit> bør være navn som identifiserer funksjonen/innholdet i filen. (eksempel: seg7ctrl_ent.vhd, seg7ctrl_rtl.vhd, tb_seg7ctrl.vhd.).

Det er vanlig med et mindre antall design pakker, ofte bare en, som er syntetiserbar (eksempel: mydesign_pck.vhd og mydesign_bdy.vhd). I tillegg kommer en eller flere testbenk pakker (eksempel: tb_mydesign_pck.vhd og tb_mydesign_bdy.vhd)

Testbenk for VHDL-simulering

Det sentrale spørsmålet ved utformingen av testbenken er hvilke testvektorer man må generere for å få en fullstendig simulering av kretsens oppførsel. Avhengig av kretsens kompleksitet kan dette være en meget enkel eller meget vanskelig oppgave. I mange tilfeller er det lurt å ta utgangspunkt i en tabell med alle relevante innsignalkombinasjoner og forventede utsignalverdier.



Figur 1. Multiplexing av sjusegmenter

Tabell 1. Sannhetstabell for et 7-segment

Di(3:0) i=3,2,1,0	Dec(i) i=3,2,1,0	abcdefgdec	Tegn
0000	0	00000011	0
0001	0	10011111	1
0010	0	00100101	2
0011	0	00001101	3
0100	0	10011001	4
0101	0	01001001	5
0110	0	01000001	6
0111	0	00011111	7
1000	0	00000001	8
1001	0	00011001	9
1010	0	00010001	A
1011	0	11000001	b
1100	0	01100011	C
1101	0	10000101	D
1110	0	01100001	E
1111	0	01110001	F
0000	1	00000010	0.
0001	1	10011110	1.
0010	1	00100100	2.
0011	1	00001100	3.
0100	1	10011000	4.
0101	1	01001000	5.
0110	1	01000000	6.
0111	1	00011110	7.
1000	1	00000000	8.
1001	1	00011000	9.
1010	1	00010000	A.
1011	1	11000000	b.
1100	1	01100010	C.
1101	1	10000100	D.
1110	1	01100000	E.
1111	1	01110000	F.

Rapport.

En kortfattet rapport som oppsummerer hva som er gjort, med problemer/utfordringer. Tegn gjerne figurer med f.eks. blokkdiagrammer.

Dersom ikke annet er spesifisert skal dere for hver deloppgave levere:

- VHDL-kildefil(er)
- VHDL-testbenk
- *Do*-fil(er) for simulering i Modelsim.
- Place & Route report/Static Timing report der man har gjennomført Place og route.

Oppgave 4 og 5 skal demonstreres for labveilder.

Alle innleverte VHDL-filer skal følge navnreglene for vhdl-filer og retningslinjene for indentering som beskrevet i kokeboken.

Oppgave 2.

Lag en *function* i VHDL som implementerer sannhetstabellen Tabell 1 for D0/Dec(0).
Det skal ikke lages testbenk i denne oppgaven.

Oppgave 3.

I denne oppgaven skal dere implementere kontroll av 7-segmentene slik at alle er aktive *samtidig*. Dette er mulig få til ved å lage en konstruksjon der de 4 displayene aktiveres i sekvens, og der frekvensen er tilstrekkelig høy til at de 4 displayene lyser samtidig. En slik konstruksjon kan implementeres ved hjelp av en teller. NB! ANi (i=3,2,1,0) må være aktive et antall 50MHz perioder for at tegn skal vises riktig. Blir AN-pulsene for korte vil tegn fra nabosegmenter flyte sammen.

Denne modulen skal ha følgende entitet:

```
entity seg7ctrl is
  port
  (
    mclk         : in std_logic; --50MHz, positiv flanke
    reset        : in std_logic; --Asynkron reset, aktiv høy
    d0           : in std_logic_vector(3 downto 0);
    d1           : in std_logic_vector(3 downto 0);
    d2           : in std_logic_vector(3 downto 0);
    d3           : in std_logic_vector(3 downto 0);
    dec          : in std_logic_vector(3 downto 0);
    abcdefgdec_n : out std_logic_vector(7 downto 0);
    a_n          : out std_logic_vector(3 downto 0)
  );
end entity seg7ctrl;
```

Figur 1. Entiteten seg7ctrl

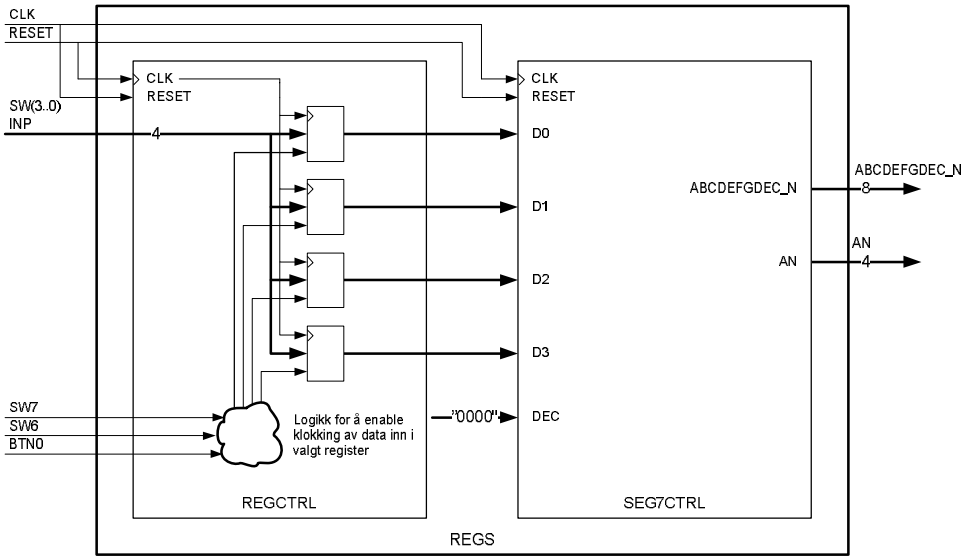
Denne modulen skal syntetiseres, men ikke testes på testkortet i denne deloppgaven.
Det skal lages testbenk der simuleringsmodellen (`seg7model_beh.vhd`) for 7-segmentene inngår.
Ved å velge Radix Hexadecimal på DISPi-utgangene (i=3,2,1,0) av modellen skal dere få fram i waveformvieweren tall som vil vises på sjusegmentene. Det vil også gå tydelig fram når et display ikke er aktivt.
Både RTL-kode og PostSynth-kode (etter syntese) skal simuleres. PostSynth-VHDL-koden vil bli lagt under <ise_project>/netgen/synthesis. Ved aritmetikk operasjoner som for eksempel +, -, * og /, skal kun pakken "ieee.numeric_std.all" brukes.

Veiledning:
Lag et blokkskjema som viser dataveiene fra d0, d1, d2 og d3 til 7-segment displayene.
Hvordan skal telleren brukes? Telleren og dekoderen fra laboppgave 1 kan komme til nytte her.

Oppgave 4.

Benytt entiteten i deloppgave 3 sammen med switchene SWi (i=7,6,4...0) og trykknappene BTN1 og BTN0 til å implementere en funksjon som virker på følgende måte:
Verdien fra switchene SW(3 downto 0) skal lagres i fire 4-bits registre Di(3 downto 0), i = 3,2,1,0. Utgangen fra Di skal vises på DISPi, i=3,2,1,0. Switchene SW(7 downto 6) sammen med et knappetrykk på BTN0 skriver inn verdien av SW(3 downto 0), kalt INP i Tabell 2, i Di, der *i* tilsvarer binærtallet dannet av SW7 og 6. Innlesning av nye verdier til registrene skal synkroniseres med positiv flanke av 50MHz klokken på kortet. Ved å aktivisere RESET nullstilles alle registrene. Hvor mange bits må telleren ha i deloppgave 3, og hvilke bit benyttes for å få tydelige tegn på hver av 7-segmentene?

Lag en ”regs” komponent som vist i figuren under og koble sammen ”seg7ctrl” og ”regctrl” componentene i en strukturell (”str”) topp nivå arkitektur. Alle registere skal være i reset når reset signalet er aktivt ’1’.



Figur 2.

Tabell 2.

CLK	BTN1 (RESET)	BTN0 (LOAD)	SW7	SW6 (SEL)	D3	D2	D1	D0
-	1	-	-	-	00	00	00	00
-	0	0	-	-	D3	D2	D1	D0
↑	0	1	0	0	D3	D2	D1	INP
↑	0	1	0	1	D3	D2	INP	D0
↑	0	1	1	0	D3	INP	D1	D0
↑	0	1	1	1	INP	D2	D1	D0

Oppgave 5

Endre designet fra deloppgave 4 slik at det blir en digital stoppeklokke. Nå blir load og sel signalene unødvendige. De erstattes med et start signal i BTN0 og et stop signal i BTN2. Reset i BTN1 beholdes. Etter at stop er trykket vil det å trykke på start knappen igjen fortsette tellingen. Reset setter alt til null og venter på start signalet. Etter at displayet viser 9999 går tellingen til 0000 og begynner på nytt igjen.

Hint1: Erstatt regs komponenten med en clock komponent som har mclk, reset, start og stop som innganger.

Hint2: En klokkefrekvens på 50 MHz gir 20 ns for hver klokke periode. Det bør lages en teller i clock modulen som teller opp til 1 sekund og da øker display verdien med 1 for hvert sekund.

LYKKE TIL!