

Prosjektoppgåve i ELE111 2025

Innleiing

Denne oppgåva er ein del av vurderingsgrunnlaget i ELE111. Det blir gitt karakter på oppgåva, A-F. Opgåva tel 40% av karakteren i emnet.

Det er planen at du skal arbeida med oppgåva gjennom heile semesteret.

Oppgåveformulering

Det skal lagast eit design for eit kommunikasjonssystem, der to D2-155-kort skal kunne senda meldingar til kvarandre. Eit kort skal vera sendar, eit anna mottakar. Kommunikasjonen skal følga RS-232-protokollen. Det skal skrivast VHDL-kode for sendar og mottakar. Det bør lagast eit felles Quartus-prosjekt som inneholdt både sendar og mottakar.

Det er ein mål at koden frå dette prosjektet skal kunne brukast vidare i ELE113. Det skal derfor lagast ein entity for sendar og ein for mottakar som har ein veldefinert grensesnitt (portar).

Mal for enteties for sendar og mottakar er gitt i Figur 4 og Figur 5.

Innlevering

Oppgåva skal leverast på Wiseflow. Fristen for innlevering blir kunngjort på Wiseflow og på Canvas.

Det skal leverast:

- Komplett(e) Quartusprosjekt som zip-fil.
- Eit pdf-dokument med
 - Forklaring og beskrivelse av det leverte systemet
 - Test-rapport med utklipt frå Modelsim og SignalTap, og evaluering av testresultat.
 - Test-rapport frå testing av system på kortet.

Ved karaktersetjinga vil både rapport og VHDL–kode telja med.

RS-232-protokollen

RS-232 er ein standard for kommunikasjon mellom to punkt, over relativt kort avstand og låg til middels datarate. Den blir bruk til f. eks

- PC <-> skriver
- PC <-> modem
- Server <-> terminal
- PC <-> mikrokontroller.

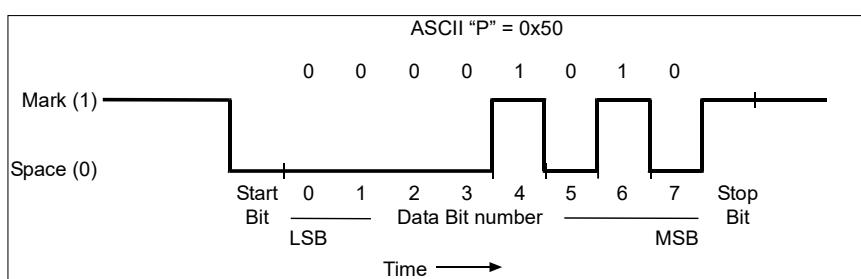
RS-232 bruker ein par-kabel (to leiarar), gjerne tvinna parkabel.

Datarater frå 75 bit/s til 256000 bit/s

RS-232 er erstatta av USB i mange samanhengar.

RS-232 blir brukt til punkt-til punkt-kommunikasjon, slik at det er ein sendar og ein mottakar. Sendaren set opp spenning mellom kablane i trådparet, mottakaren måler spenninga.

- › Ved bruk av RS-232 må sendar og mottakar vera einige om en del reglar for kommunikasjonen. Vi må bestemma:
 - › Tal på databit i kvar melding – 5-8 bit
 - › Synkroniseringsbit – 1 eller 2 stoppbit
 - › Paritetsbit, odde, like eller ingen paritet.
 - › Baud-rate



Figur 1 RS-232C Seriell overføring av et 8-bits signal (kilde: Rapid)

Figur 1 viser tidsforløp for spenninga på ei RS232-linje under sending av ei melding. Spenninga ligg høg når linja ikkje sender melding. Ei melding startar med eit startbit, det er alltid '0'. Så kjem databit, så mange som det er bestemt på førehand. I Figur 1 er det 8 bit data, det er ikkje brukt paritet. Dersom det blir brukt paritet, kjem paritetsbitet etter siste databit. Melding blir avslutta av eit eller to stoppbit. Stoppbit er alltid '1'.

I RS-232 kan vi ha frå 5 til 8 bit pr melding. Linja ligg med høg spenning når han er ledig. (Høg spenning = Vcc)



Før databit blir sendt, sender vi startbitet, det er alltid '0' (Låg spenning, OV). Etter startbitet kjem databit, så mange vi har bestemt,

'1' = høg spenning, '0' = låg spenning

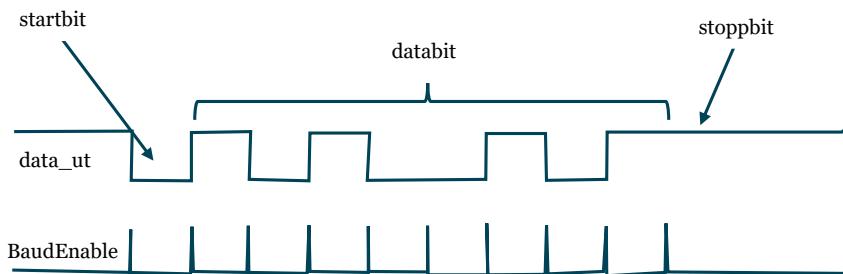
Til slutt kjem eit eller to stop-bit, alltid '1'. Dersom vi har med paritet, kjem paritetsbitet mellom siste data-bit og stop-bitet.

Baud-rate

Baudrate er antall symbol overført pr sekund. I RS-232 bruker vi eit symbol pr databit, men i tillegg må vi bruka nokre symbol til å senda startbit, stopp-bit og eventuell paritet. Dersom baud-raten er $R_B \frac{\text{symbol}}{\text{sekund}}$ er symbol-perioden $T_B = \frac{1}{R_B}$

Både sendar og mottakar må ha ein tilstandsmaskin for å generera baud-raten. Tilstandsmaskinen skal senda ut ein puls, BaudEnable, ein gong pr baud-periode, for den baudraten som er vald.

I sendaren bestemmer BaudEnable at neste bit skal sendast til utgangen, dvs startbitet startar på den første BaudEnable-pulsen, første databit på den neste BaudEnable-pulsen, og så eit nytt bit for kvar BaudEnable-puls til heile meldinga er sendt. Eksempel på BaudEnable og data-ut er vist i Figur 2

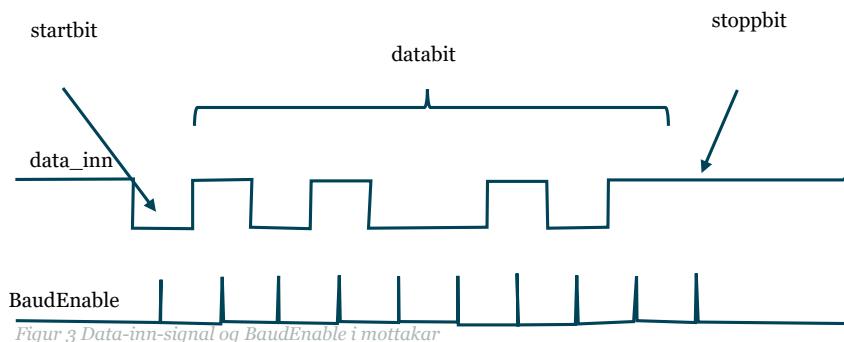


Figur 2 Data-ut-signal og BaudEnable i sender



For mottakar blir det litt ulikt. Mottakar må lesa av data-inn kontinuerleg. Data-inn ligg høg så lenge ingenting blir sendt. Ei melding startar alltid med eit startbit som er '0', så når mottakaren oppdagar ein fallande flanke på inn-data, veit vi at vi har mottatt startbitet, og ei melding er underveis. Denne fallande flanken i starten av start-bitet er signalet til å setja i gang baud-generatoren på mottakaren. Baud-generatoren skal gi ein BaudEnable-puls i kvar baud-periode, men i mottakaren skal denne koma midt i baud-perioden, ikkje i starten. Dette er fordi BaudEnable-pulsen fortel mottakaren når det er best å lesa av verdien på det bitet vi har mottatt, og det er mest stabilt midt i symbolperioden. Dvs vi kan ha ein BaudEnable-puls ein halv baud-periode etter at vi ha detektert starten på startbitet, og så neste puls ein baud-periode etterpå.

Figur 3 viser tidsdiagram med data-inn-signal og BaudEnable.



Figur 3 Data-inn-signal og BaudEnable i mottakar

Sendar og mottakar skal kunne konfigurerast til eit utval standard datarater for RS-232. i tillegg bør vi kunne ha datarate 1 Mbaud/sek. Det blir enklare å bruka Modelsim og SignalTap med denne høge dataraten.

Koden for sendar og mottakar skal vera mest mogleg generell, og kunne gjenbrukast i ELE113. Entity for sendar og mottakar skal derfor og innhalda kode eller komponent for å generera baudrate.

Baudrate for sendar og mottakar skal velgast ved ein 16-bit input, `baud_rate_divider`. Denne skal innhalda eit binærtal som gir baud-perioden (T_B) i talet på klokkeperiodar. Baud-perioden T_B i klokkeperioder for ein gitt datarate R_B finn vi ved:

$$T_B = \frac{50\,000\,000}{R_B} \text{ klokkeperiodar.}$$

Vi har ikkje nok svitsjar på DE2-kortet til å setja `baud_rate_divider` direkte. Du må derfor legga kode eller komponent for å setja rett `baud_rate_divider` for eit utval baud-rater ved hjelp av `SW(16 downto 14)`.

Baudrater og forslag til SW-setting er vist i Tabell 1

Tabell 1 Baud-rater som skal brukast i systemet, og forslag til SW-innstilling

	SW(16 downto 14)	Baudrate
0	000	4800
1	001	9600
2	010	19200
3	011	34800
4	100	57600
5	101	74880
6	110	115200
7	111	1 MHz

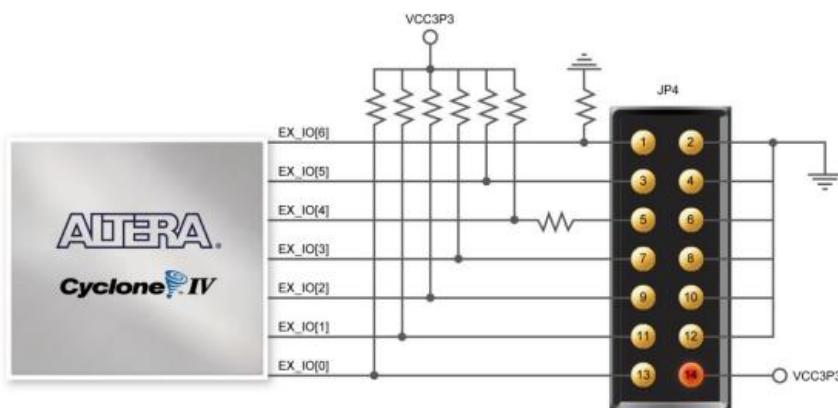


Figure 4-20 Connections between FPGA and 14-pin general purpose I/O

Vi bruker den generelle kontakten EX_IO til å kopla saman to DE2-115-kort.

EX_IO-kontakten har 7 pinnar vi kan bruka til ulike ting. Dei er nummerert frå 1 til 7 på kortet, men når vi definerer denne som ein port i VHDL, bruker vi nummerering frå 0 til 35

`EX_IO : inout std_logic_vector(6 downto 0);`

EX_IO-porten kan deklarerast som `inout`. EX_IO-pinne 12 er fast jord- og vi bør kopla saman EX_IO-pinne 12 på kort for at vi skal ha felles jord-potensiale, og unngå spenningsforskjell mellom sender og mottakar.

Bruk f. eks EX_IO(0) (EX_IO-pinne 1 på kortet) som data-ut på sender, og EX_IO(6) som data-opp i mottakar. Når inngangsdelen av EX_IO(6) ~~denne~~ blir brukt til å ta i mot data i mottakaren, må utgangsdelen av EX_IO(6) bli sett i tri-state ('Z')

```
EX_IO(6) <='Z';
data_inn <= EX_IO(6);
```

Når vi bruker ulike EX_IO-pinner til sender og mottakar, kan vi testa systemet på eit kort ved å kopla sende- og mottakarpinnane saman.



Entity for sendar

```
entity sender is
  port(
    clk : in std_logic;
    rst_n : in std_logic;
    startPuls : in std_logic;
    baud_rate_divider : in std_logic_vector(15 downto 0);
    dataIn : in std_logic_vector(7 downto 0);
    txReady : out std_logic;
    dataOut : out std_logic
  );
end entity sender;
```

Figur 4 Entity for sendar.

Koden for sendar skal innhalla ein entity som vist i Figur 4. Avhengig av korleis du vel å realisera sendaren, vil denne entiteten bli ein komponent i topp-nivå-entitetten for sendaren.

Forklaring til portane:

```
clk : in std_logic; -- systemklokke
rst_n : in std_logic; -- reset, aktiv låg
startPuls : in std_logic; --puls som startar
  sending av 8 bit med data
baud_rate_divider : in std_logic_vector(15 downto 0);
  -- Baud-periode i antall (50 MHz) klokkeperiodar
dataIn : in std_logic_vector(7 downto 0);
  -- 8 bit med data som skal overførast i ei
  melding
txReady : out std_logic; -- Når txReady = '1' ventar
  sender på ny sendEnable for å starta sending av
  nye data.
dataOut : out std_logic -- serielle data , skal koplast
  mot EX_IO-port
```

```
entity mottaker is
  port(
    clk          : in  std_logic;
    rst_n       : in  std_logic;
    dataInn     : in  std_logic;
    baudRateDivider : in std_logic_vector(15 downto 0);
    error        : out std_logic;
    dataValidUt : out std_logic;
    dataUt       : out std_logic_vector(7 downto 0));
end entity mottaker;
```

Figur 5 Entity for mottakar

Koden for mottakar skal innhalda ein entity som vist i Figur 5. Avhengig av korleis du vel å realisera mottakaren, vil denne entityen bli ein komponent i topp-nivå-entiteten for mottakaren.

Forklaring til portane:

```
clk          : in  std_logic; -- systemklokke
rst_n       : in  std_logic; -- reset aktiv låg
dataInn     : in  std_logic; -- Serielle data inn
baudRateDivider : in std_logic_vector(15 downto 0);
                  -- Baud-periode i antall (50 MHz) klokkeperiodar
error        : out std_logic; -- flag som er høg om
                  mottatt data ikkje er gyldig RS-232-data.
dataValidUt : out std_logic; -- høg når feilfrie data
                  er mottatt og tilgjengeleg i dataUt
dataUt       : out std_logic_vector(7 downto 0) -- 8 bit
                  mottatt data
```

Dersom du ønskjer å gjera koden meir fleksibel, f. eks er det tillatt å kunne konfigurera RSR-2-parameter som talet på bit og paritet gjennom bruk av generics eller kontrollsignal i entity for sender og mottakar.



I dette kapittelet er det sett opp kva som skal med i systemet som skal utviklast. For å få topp-karakter må alle element med i leveringa. For å få bestått må minimum punkt **1a, 1f, 2, 3, 4a, 5a, 6a og 7a** leverast. Minimumskrava er merka med **blå farge**. Det er lov å levera kode som ikkje fungerer.

1. Prosjektrapport som beskriver systemet

- a. System-beskrivelse på blokk-nivå
Rapporten skal beskriva systemet som er laga, eller som var tenkt lagd.
Dersom ikkje alt fungerer, skal rapporten dokumentera kva som ikkje fungerer, og gjerne med forslag til forbedringar
- b. Test-rapport frå testing med eit kort i loop
- c. Test-rapport frå testing kort mot kort
- d. Test-rapport frå Modelsim-simulering
- e. Test-rapport frå SignalTap-testing.
- f. Referanseliste. Referansar kan skrivast i ein godkjend standard, til dømes standarden IEEE Ein god hugseregel er at kvar gong ein kjem med ein påstand skal ein ha ein referanse.
Dersom du hentar kode frå andre, eller bruker kode du har skrive sjølv i andre oppgåver, skal det med ein referanse.

2. VHDL-kode for sendar

Sendar skal kunne senda data over EX_IO(0) som samsvarar med RS232-protokollen.

Koden skal innehalda entity for sendar som vist i Figur 4. Koden skal som minimum handtera meldingar med 8 bit, ingen paritet.

Det er tillat å laga kode som kan konfigurerast med 5-8 bit pr melding og med eller utan paritet, ved hjelp av generics eller kontrollsignal

3. VHDL-kode for mottakar

Mottakar skal kunne ta imot data over EX_IO(6) som samsvarar med RS232-protokollen.

Koden skal innehalda entity for mottaker som vist i Figur 5. Koden skal som minimum handtera meldingar med 8 bit, ingen paritet.

Det er tillat å laga kode som kan konfigurerast med 5-8 bit pr melding og med eller utan paritet, ved hjelp av generics eller kontrollsignal

4. Baud-rate-generator for sendar

- a. Generator for ein fast baud-rate
- b. Konfigurerbar baud-rate, som i Tabell 1

5. Baud-rate-generator for mottakar

- a. Generator for ein fast baud-rate
- b. Konfigurerbar baud-rate, som i Tabell 1

Kode for sendar og mottakar kan godt kombinerast i eit

Quartus-prosjekt, der kortet blir konfigurert som sender eller mottakar med SW-brytar.

6. Data-kjelder i sendar

- a. [Melding, 8 bit, hardkoda i VHDL-koden for sendar.](#)
- b. Melding, 8 bit gitt med SWo-7
- c. Lag digital klokke på sendar, send 3 byte klokke-data til mottakar.
- d. 3 byte med klokkedata, med synkronisering på byte-nivå.

7. Visining av data i mottakar

- a. [LED](#)
- b. 7-segment-display
- c. LCD-skjerm

8. Verifikasjon

Sendar og mottakar bør verifiserast. Detta kan gjerast med Modelsim eller SignalTap, helst begge to.

- a. Modelsim
- b. SignalTap

9. Test

Når systemet er ferdig må det testast på kortet.

- a. Mot seg sjølv, med loop mellom sendar og mottakar på EX_IO-kontakten
- b. Mot anna DE2-115-kort

10. Synkronisering

- a. Klokken (CLOCK_50) på sendar og mottakar er ikkje synkroniserte. Det betyr at vi har klokke-domene-kryssing mellom sendar og mottakar, og data inn på mottakar må synkroniserast.
- b. Reset (frå KEY(3)) må synkroniserast til klokka.
- c. Ved sending av fleire data-byte må mottakaren kunne fastslå kva byte som er først i ein sekvens. Det bør lagast ein mekanisme som mottakaren kan bruka for å kunne presentera data i rett rekkefølge.