LABORATION

DATORKONSTRUKTION TSEA83

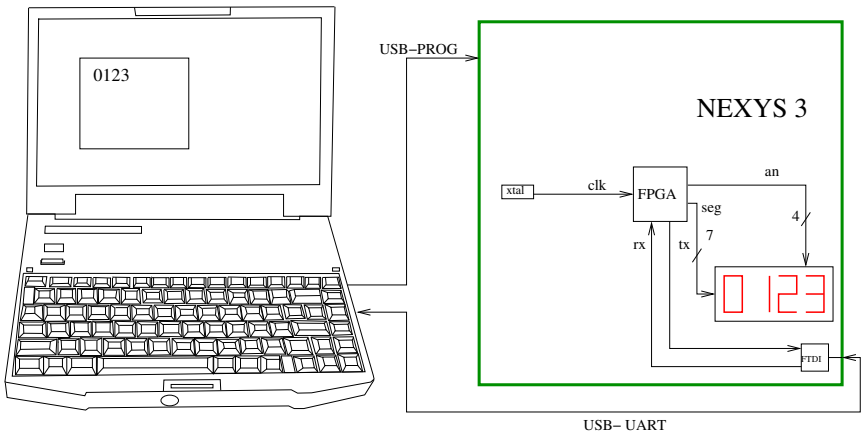
UART

Version: 1.2
2013 (OS)
2020 (AN)

blank sida

Innehåll

blank sida



Figur 1: Laborationsuppställning. Siffror skickas på seriell form från datorns tangentbord till FPGA-kortets display.

1 Inledning

Siffror inmatade från en dators tangentbord ska sändas seriellt över en RS232-förbindelse och presenteras på 7-segmentsdisplayen på FPGA-kortet Nexys-3, se figur 1.

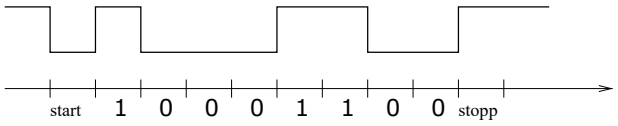
1.1 Syfte

I denna laboration får du lära dig att:

- använda FPGA-kortet Nexys 3, som du sedan ska använda i ditt projekt.
- skriva VHDL-kod, som knyter an till både blockschema och signalschema.
- simulera din konstruktion med ModelSim.
- syntetisera och testa din konstruktion.

1.2 Förberedelser

Innan du kommer till laborationen måste du vara väl förberedd. Alla uppgifter som ska redovisas är utmärkta med ett pekfinger. En del av dem kan med fördel utföras hemma som förberedelseuppgifter.



Figur 2: Överföring av siffran 1. ASCII-koden 0x31 skickas med LSB först.

2 Teori

2.1 RS232

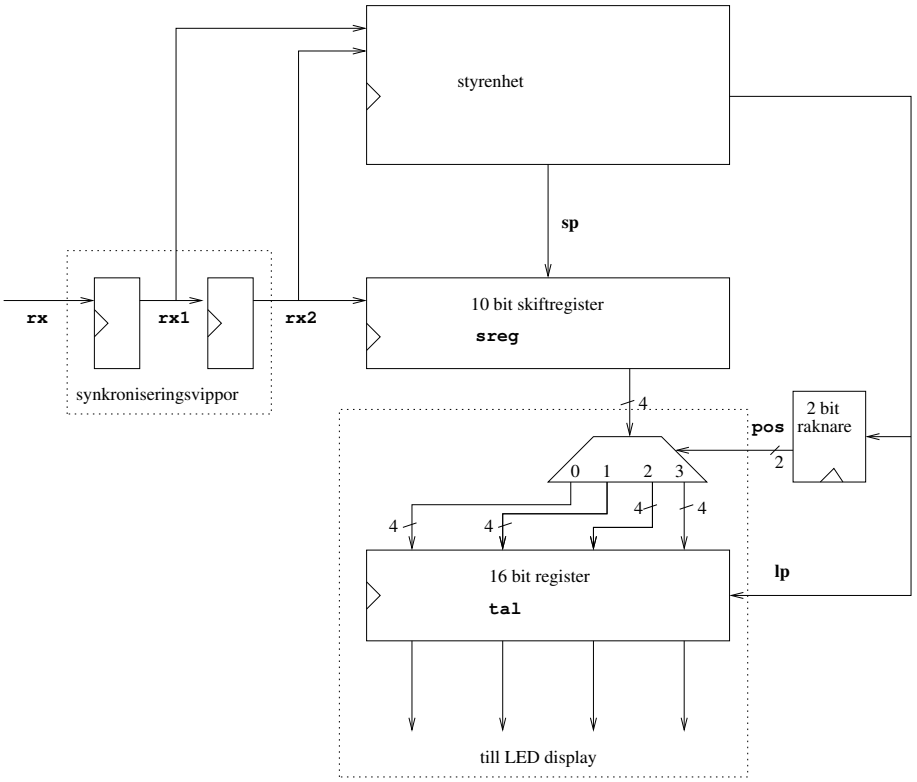
RS232 är en (gammal) standard för seriell kommunikation mellan en sändare och en mottagare. Vanligast är att kommunikationen pågår åt båda hållen samtidigt, så kallad full duplex. En krets som klarar detta brukar kallas UART (Universal Asynchronous Receiver Transmitter). I vårt fall ska endast sändning från datorn till FPGA-kortet implementeras. Dessutom nöjer vi oss med att skicka siffror, som kan visas på displayen. FPGA-kortet är utrustat med en FTDI-krets (Googla på detta ord), vilket innebär att en USB-kabel kan användas. De seriella signalerna `rx` (receive) och `tx` (transmit) är endast tillgängliga inuti FPGA-kretsen, se figur 1.
RS232 är en asynkron och långsam överföringsmetod. Tecken skickas ASCII-kodade med minst signifikant bit först. Tecknet föregås av en startbit (0:a) och avslutas med en stoppbit (1:a), se figur 2. Varje tecken är således 10 bitar långt. I denna laboration används hastigheten 115200 bit/s. Detta innebär att varje bit är $8,68\mu$s lång, eller med 100 MHz klocka 868 klockpulser.

2.2 Partitionering

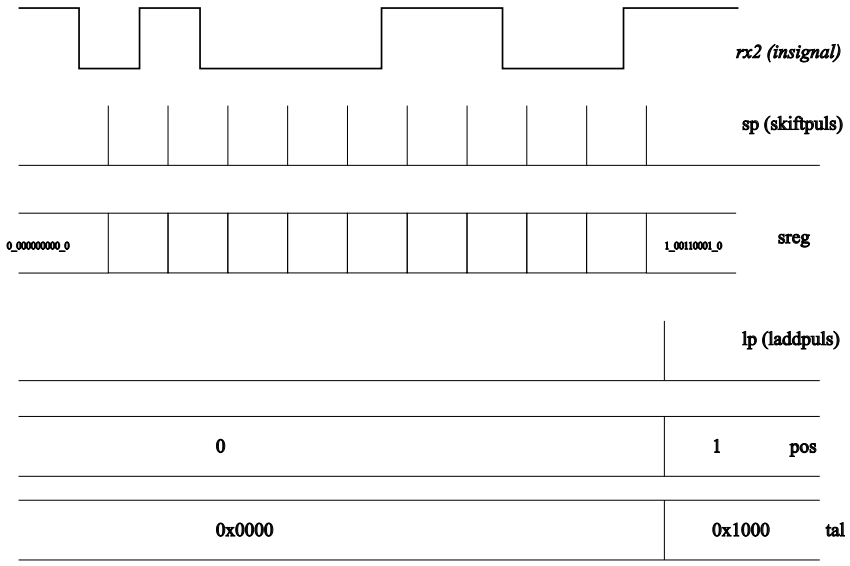
Ett blockschema med tillhörande signalschema för vår UART visas i figur 3. Klockfrekvensen är 100 MHz. Konstruktionen är indelad i följande block:

- *Synkroniseringsvippor*. Förändringarna på insignalen ska komma i takt med vår klocka.
- *Styrenhet*. Denna producerar två styrsignaler, båda enpulsade:
  - Skiftpulsen `sp`, som kommer mitt i (ungefär) varje bit.
  - Laddpulsen `lp`, som kommer efter den sista skiftpulsen.
- *Skiftregister*. De 10 bitarna i varje siffra skiftas in i skiftregistret.
- *Register* för 4 siffror. Laddas av laddpulsen, samtidigt räknas räknaren upp.

a)



b)



Figur 3: a) Blockschema. b) Signalschema. Siffran 1 tas emot och skrivs in i position 1 i displayen.

2.3 Goda råd om VHDL

2.3.1 Klockade processer

Nästan precis så här måste alla klockade processer se ut. Vi rekommenderar att du helt enkelt använder denna kodsnutt som mall. Rad 1 och rad 2 ska se ut som nedan. Rad 3 bör vara med. Signalen `clk` har frekvensen 100 MHz. Du får inte klocka på något annat än `clk`.

```
process(clk) begin
  if rising_edge(clk) then
    if rst='1' then
      <init>
    elsif <villkor> then
      <gör något>
    else
      <gör något annat>
    end if;
  end if;
end process;
```

3 Uppgifter

3.1 Förberedelse - Kodning

Börja med att ladda ner lab-skelettet `lab_uart.tgz` från hemsidan. Packa upp med `tar xzvf lab_uart.tgz`.

Översätt blockschemat i figur 3 till VHDL-kod. Använd samma signalnamn som i figuren 3. Tänk på att skriva lättläst och strukturerad kod. Kod som inte är lättläst riskerar att bli underkänd, då det kan vara svårt att avgöra om det fungerar.

3.1.1 Lättläst kod

För att blockschemat i figur 3 ska vara ett naturligt steg i konstruktionen, ska blockschemat och signalschemat översättas till VHDL-kod med samma struktur. Översätt alltså enligt följande:

- *Synkroniseringsvippor*. 1 process.
- *Styrenhet*. 1 eller 2 processer.
- *Skiftregister*. 1 process.
- *16 bit register + demultiplexer*. 1 process.
- *2 bit räknare*. 1 process.

3.2 Uppgift - Simulering

Simulera konstruktionen genom att köra kommandot:

```
make lab.sim
```

Glöm inte heller bort `module add courses/TSEA83`. Högerklicka på `lab_tb` uppe till vänster i ModelSim-fönstret och välj `Add->To Wave->All Items in design`. Ett wave-fönster hoppar nu upp. Skriv sedan `run 500 us` i ModelSim-fönstret.
Testbänken, som körs är `lab_tb.vhd`. Ta reda på vad testbänken gör! Du ska nu kunna se samma signaler som i figuren 3.
Laborationsassistenten kommer att vilja se simuleringen.

3.3 Uppgift - Demonstration

Gör en bit-fil för konstruktionen genom att köra kommandot:

```
make lab.bitgen
```

Nu kan du programmera FPGA-n med

```
make lab.prog
```

Starta terminalprogrammet `gtkterm` (t ex genom att skriva gtkterm i ett terminalfönster) och konfigurera enligt följande (välj Configuration/Port):

parameter	värde
Port	/dev/ttyUSB0
Baud Rate	115200
Parity	None
Bits	8
Stopbits	1
Flow control	none

Det ska nu gå att skriva siffror på PC:ns tangentbord som då skickas via PC:ns UART till din UART i Nexys3-kortet, och siffrorna ska visas efter varandra på Nexys3-kortets 7-segments-display.

4 Versionshistorik

2013	OS	Ursprunglig version
2016	AN	Omritade figurer
2019	AN	Korrektur av figur 2 och 3
2020	AN	Förtydliganden av hur gtkterm startas

Referenser

[1] Ragnemalm,Seger: *Digital konstruktion TSEA43*, http://www.da.isy.liu.se/courses/tsea83

A User Constraint file

Innehåll i filen `lab.ucf`:

```
##############################################
# Define Device, Package, And Speed Grade
##############################################
#
CONFIG PART = xc6slx16-3-csg324;

##############################################
# clk, rst
##############################################
##Clock signal
Net "clk" LOC=V10 | IOSTANDARD=LVCMOS33;
Net "clk" TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;

Net "rst" LOC = B8 | IOSTANDARD = LVCMOS33;

#######################################
# Multiplexed display                 #
#######################################
## 7 segment display
Net "seg<7>" LOC = M13 | IOSTANDARD = LVCMOS33;
Net "seg<6>" LOC = T17 | IOSTANDARD = LVCMOS33;
Net "seg<5>" LOC = T18 | IOSTANDARD = LVCMOS33;
Net "seg<4>" LOC = U17 | IOSTANDARD = LVCMOS33;
Net "seg<3>" LOC = U18 | IOSTANDARD = LVCMOS33;
Net "seg<2>" LOC = M14 | IOSTANDARD = LVCMOS33;
Net "seg<1>" LOC = N14 | IOSTANDARD = LVCMOS33;
Net "seg<0>" LOC = L14 | IOSTANDARD = LVCMOS33;


Net "an<0>" LOC = N16 | IOSTANDARD = LVCMOS33;
Net "an<1>" LOC = N15 | IOSTANDARD = LVCMOS33;
Net "an<2>" LOC = P18 | IOSTANDARD = LVCMOS33;
Net "an<3>" LOC = P17 | IOSTANDARD = LVCMOS33;

## Usb-RS232 interface
Net "rx" LOC = N17 | IOSTANDARD=LVCMOS33;
```