Linköpings tekniska högskola, ISY, Datorteknik

Laborationskompendium
Digitalteknik
Konstruktion av mindre digitala system

Linköping 2019

Laboration 4

Konstruktion av mindre digitala system

Syftet med laborationen är dels att öva på konstruktion av mindre digitala system, och dels att ge en utökad inblick i moderna konstruktionshjälpmedel för digital konstruktion.

Efter genomförd laborations ska ni:

- Genom att rita blockschema, kunna konstruera mindre digitala system med hjälp av programmerbar logik.
- Ha befäst grunderna i det hårdvarubeskrivande språket VHDL.
- Kunna använda ModelSim för enklare simuleringar.
- Ha insikt i modern systemutvecklingsmetodik.

Laborationsutrustningen kommer att kompletteras med en modul som innehåller en CPLD (Complex Programmable Logic Device) av typen XC9572 tillverkad av Xilinx, en tidbasmodul som kan skicka ut pulser med 1, 10, 100 eller 1000 Hz, samt en pulsgivare. Extrakomponenterna beskrivs i de uppgifter där de används.

4.1 Examinationskrav

För godkänd laboration krävs att uppgift 4.1 är godkänd samt en av uppgifterna 4.2 och 4.3. Laborationen är uppdelad på tre tillfällen om två timmar var.

I den här kursen ligger fokus på grundläggande digitalteknik med enkla tillämpningar som exemplifierar kursinnehållet. Den VHDL som ingår i kursen är därför bara ett absolut minimum. För att bli godkänd på en uppgift krävs:

- Ett detaljerat blockschema över konstruktionen där man på ett tydligt sätt kan se kopplingen till er VHDL-kod.
- VHDL-kod som motsvarar blockschemat.
- En fungerande krets.

Varje uppgift ska lösas i ett eget VHDL-projekt i en VHDL-fil. Varje block i blockschemat ska vara någon av de block som är definierade i dokumentet Digitaltekniska byggblock eller grindar/inverterare. För varje block i blockschemat ska motsvarande kod finnas i VHDL-filen. Motsvarande kod definieras i Digitaltekniska byggblock. Det ska alltså vara en ett-till-ett matchning mellan blocken

i blockschemat och motsvarande kod-snuttar i VHDL-filen. På detta sätt blir det en process-sats per block innehållande register. Blocken får modifieras efter behov, t ex kan en räknare anpassas så att den får önskat antal bitar och önskade standardfunktioner för en räknare. Om ett block realiserar en egendesignad sekvenskrets ska även tillståndsdiagram ingå i dokumentationen.

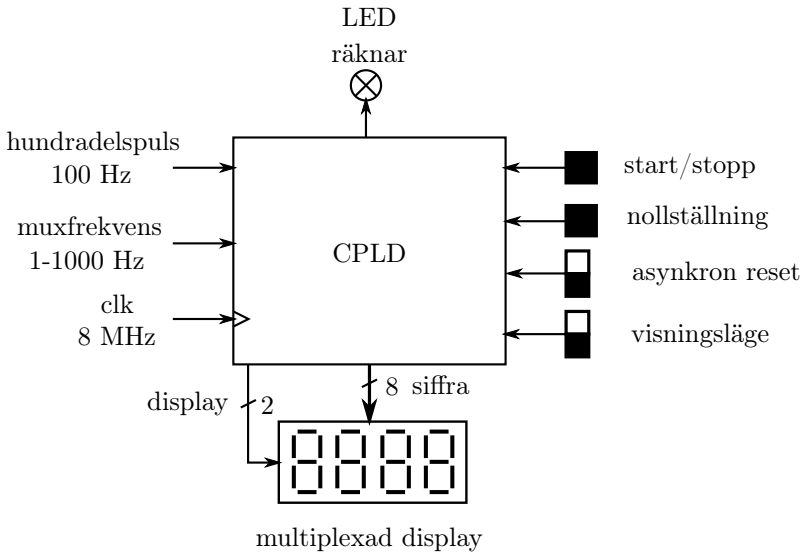
4.2 Förberedelser

Det ska finnas ett ritat blockschema till varje uppgift som ni har för avsikt att göra på laborationstiden. Det är även bra om ni har skrivit den mesta av koden före laborationen. Till uppgift 4.1 och 4.2 finns det även simuleringsövningar som ska redovisas. Inför det avslutande passet ska även lösningarna vara provsyntetiserade i förväg.

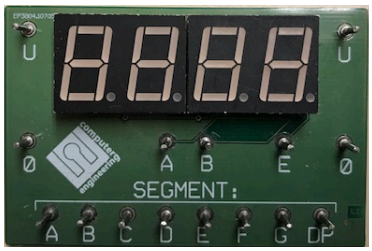
4.3 Uppgifter

Uppgift 4.1. Konstruktion av ett stoppur.

Ett stoppur som styrs av två knappar och två skjutomkopplare ska konstrueras. Uret ska räkna minuter, sekunder och hundradelar. Den ena knappen används som start/stopp och den andra för nollställning. Om tiden är stoppad och startas igen fortsätter räkningen från visad tid. LED-lampan indikerar statusen på start/stopp på så sätt att tänd lysdiod betyder att klockan går. Den ena skjutomkopplaren styr vilka fyra siffror som ska visas på den multiplexade displayen. Om skjutomkopplaren är i nedre läget visas sekunder+hundradelar och om skjutomkopplaren är i sitt övre läge visas minuter+sekunder. När 1 timma har gått börjar tiduret om från 0. Den andra skjutomkopplaren ska i övre läget aktivera asynkront reset.

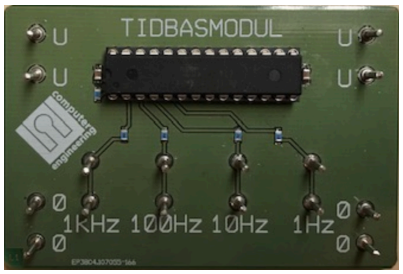


Konstruera det digitala systemet. Använd en CPLD av typen XC9572, den multiplexade displayen



en lysdiod och specificerade kappar och skjutomkopplare. Alla insignaler utom den asynkrona resetsignalen **måste synkroniseras.**

Använd kristalloscillatorn inställd på 8 MHz som systemklocka, clk, som finns på matningsskenan till vänster på labplattan. Muxfrekvensen hämtas från den variabla klockpulsgeneratorn och hundradelspulsen från en tidbasmodul



Systemklockan, clk, är den enda signal som får kopplas till klockingångar på register/vippor varför hundradelspuls och muxfrekvens ska behandlas som ”count enable”-signaler.

Förberedelser:

a) Som laborationsförberedelse ska en simulering i ModelSim över två kaskadkopplade BCD-räknare genomföras, välj sekundsiffrorna. (Jämför gärna med uppgift 2.3c) på laboration 2 där ni kopplade upp tre kaskadkopplade BCD-räknare.) Resultatet ska redovisas med hjälp av kod och en sparad bild från simuleringen alternativt direkt i simuleringsfönstret i Modelsim. (Man ska alltså kunna se en rippel-carry vid omslaget 59→00.)

   Rita blockschema för den simulerade kretsen med namngivna variabler.

b) Rita ett blockschema för hela tidtagaruret och översätt till VHDL-kod.

   Det kan vara bra att utveckla kretsen inkrementellt. Utgå till exempel från den simulerade kretsen i a)-uppgiften. Bygg ut kretsen t ex med hundradelar. Kompilera och simulera för att verifiera att delsystemet fungerar som förväntat. Bygg sedan ut kretsen steg för steg och verifiera genom kompilering och simulering innan ni går vidare. Avsluta med att provsyntetisera kretsen i Xilinx.

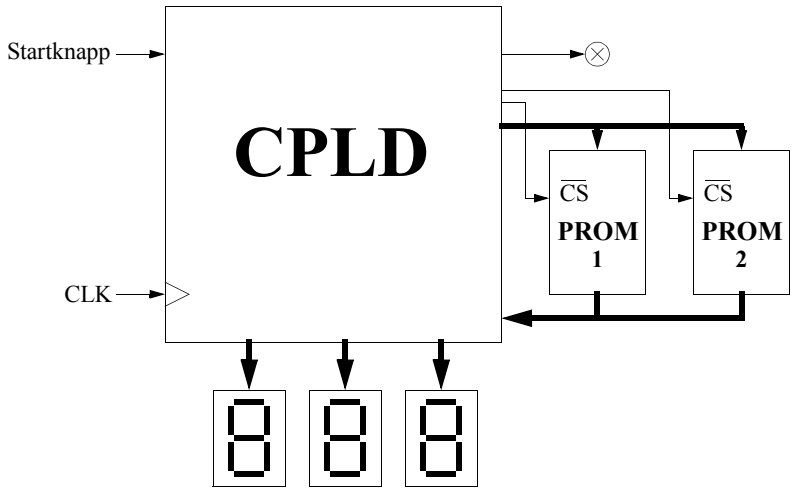
Laborationsuppgifter:

c) Koppla upp och verifiera kretsen.

d) Multiplexning innebär att man visar en siffra i taget och om detta sker tillräckligt fort kommer ögat att uppfatta detta som att alla siffror visas samtidigt. Mät med hjälp av en frekvensräknare upp den displayuppdateringsfrekvens som krävs för att erhålla en flimmerfri visning av siffrorna.

   Resultat: .....

Uppgift 4.2. Räkna ettor i PROM. Minnesinnehållet i labsatsens PROM ligger kvar även vid spänningsbortfall. Konstruera ett digitalt system som räknar det totala antalet ettor i två parallellkopplade PROM, se figur. Resultatet ska presenteras i BCD-form. Varje räkning ska startas med en knappnedtryckning, vilken alltså inkluderar såväl nollställning som start. En lysdiod ska vara tänd under tiden som räkningen pågår.



Använd VHDL och en CPLD samt lämpliga in och utsignaler. Använd de tre avkodade sjusegmentsdisplayerna för att visa antalet ettor i PROMen. Klockfrekvensen bör vara max 100 Hz, ty vi vill kunna se när konstruktionen arbetar. Osynkroniserade insignaler måste synkroniseras. Eftersom adressen till PROM:n är synkron och klockfrekvensen relativt låg, kan utdata från PROM:n räknas som synkrona, dvs PROM räknas som en rent kombinatorisk krets. Parallellkopplade PROM innebär att det är en gemensam adress- och data-buss, och att vi därför behöver utnyttja PROM:ns ”tri-state” funktionalitet på datautgången, vilket styrs med ”chip select”-signalerna. Anslut en asynkron reset till kretsen för nollställning efter spänningspåslag.

TIPS: Mängden hårdvara som behövs kan minimeras om konstruktionen tar minst 128 klockpulser på sig för att lösa uppgiften.

Förberedelser: Innan ni börjar med att skriva koden till uppgiften läs avsnitt 4.4 som beskriver hur ni kan simulera koden. Rita blockschema, översätt till VHDL-kod, simulera i en testbänk enligt beskrivningen i avsnitt 4.4 samt provsyntetisera koden.

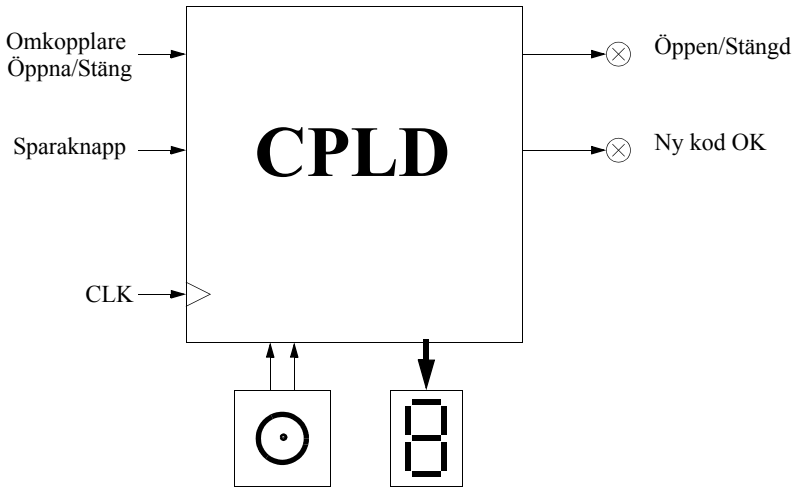
Kretsen ska ge rätt summa oberoende av hur PROMen programmeras. Det finns 2^{128} olika sätt att programmera PROMen, vilket är ett orimligt stort antal tester att utföra. Kan vi med ett rimligt antal tester (olika programmeringar av PROMen) vara säkra på att kretsen fungerar korrekt för samtliga fall? Hur ser dessa tester ut i så fall? Frågorna är viktiga men det finns inget lätt svar på frågorna och svaren beror också på er implementation.

Labuppgifter:

a) Visa resultaten av simuleringarna med testbänken.

b) Visa blockschema, kod och fungerande krets.

Uppgift 4.3. Konstruktion av ett kassaskåpslås. Konstruera styrelektroniken till ett enklare kassaskåpslås, som har en tvåsiffrig kod. Koden skapas bland annat med hjälp av en ratt som sitter på en pulsgivare vars funktion visas på nästa sida.



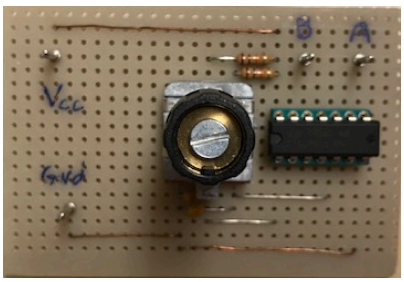
Del a) samt antingen del b) eller del c) ingår i uppgiften.

a) Koda av pulsgivaren (ratten) så att den kan styra den Upp/Ner-räknare som visas på displayen. Siffran är decimal och ska räkna upp vid vridning åt höger samt räkna ner vid vridning åt vänster. Räkningen ska ske med ett steg/puls från pulsgivaren. Siffran ska nollställas varje gång öppnaknappen förs till sitt nedre läge, dvs en övergång från 1 till 0 ska detekteras för nollställningen.

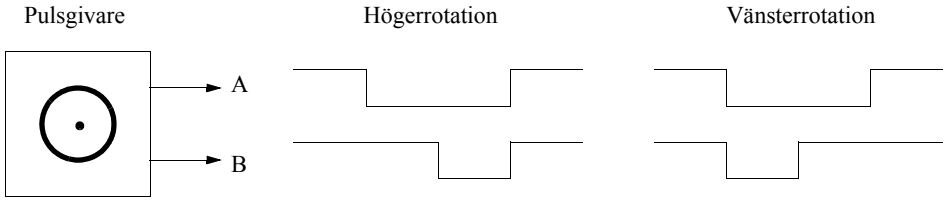
b) För att öppna låset ska först skjutomkopplaren föras till sitt nedre läge varvid en nollställning av inmatad sekvens sker. Därefter används ratten i kombination med en sparaknapp för att mata in sifferkombinationen, dvs när rätt siffra visas på displayen kan den sparas genom ett tryck på sparaknappen. Man kan trycka på sparaknappen hur många gånger som helst i och med att det bara är de två senaste sparade siffrorna som gäller. Om rätt kombination är inmatad när öppnaknappen förs till sitt övre läge ska låset öppnas, vilket indikeras av en tänd lysdiod. När låset är öppet kan man ändra på koden genom att mata in en ny sifferkombination varvid man kan se det som att det skiftas in en ny siffra i koden för varje tryckning på sparaknappen. Låset stängs genom att öppnaknappen förs till sitt nedre läge, vilket medger ett nytt öppningsförsök. Lysdioden för ny kod OK saknar funktion i den här varianten.

c) I ett klassiskt kassaskåp definieras de inmatade siffrorna genom att man vrider ratten åt andra hållet varje gång man vill spara en ny siffra. Sifferkombinationen matas därmed in genom att ratten vrids åt höger tills önskad siffra visas, följt av en vridning åt vänster tills nästa siffra i låskombinationen visas. Slutligen vrids ratten åt höger tills siffran noll visas och när nu skjutomkopplaren förs till sitt övre läge ska låset öppnas. (Under förutsättningen att rätt sifferkombination har matats in.) För att stänga låset förs skjutomkopplaren till sitt nedre läge varvid ett nytt öppningsförsök kan göras. När låset är öppet kan koden ändras genom att man nu matar in en ny kombination följt av att sparaknappen trycks ner. Ett lyckat kodbyte ska indikeras av en tänd lysdiod. När låset är stängt har sparaknappen ingen funktion. Ett öppet lås indikeras av en tänd lysdiod. Om ratten vrids åt fel håll, eller en för lång sekvens matas in ska kombinationen betraktas som felaktig.

Använd VHDL och en CPLD samt föreskrivna in och utsignaler. Systemklockan väljs till 1000 Hz. Ratten är av pulsgivartyp har följande utseende:



och funktion:



Förberedelser: Rita ett blockschema för kretsen och översätt till VHDL-kod. Simulera gärna kretsen och provsyntetisera koden.

Laborationsuppgift: Koppla upp och verifiera kretsens funktion.

4.4 Simulering av uppgift 4.2 med en testbänk

Uppgift 4.2 är svår att simulera i Modelsim, eftersom kretsen ska prata flitigt med utomstående minnen (PROM:en). Det är helt enkelt för jobbigt att manuellt tilldela data-insignalen utifrån adress- och CS-utsignalerna.

Därför har vi gjort en så kallad testbänk som gör detta automatiskt. En testbänk är också skriven i VHDL, men använder funktioner i språket som inte går att stoppa in i en CPLD. Exempel på sådana funktioner är att skapa en klocka i en viss hastighet, eller skriva ut testresultat på skärmen.

Det denna testbänk gör är:

- Den skapar de insignaler som behövs (klocka, reset, start).
- Den emulerar två PROM (så att data-ingången faktiskt beror på adress m.m.).
- Den kontrollerar att resultatet på 7-segmentsdisplayerna är rätt när LED-lampan tänds.

För att detta ska fungera, så är det viktigt att testbänken vet exakt hur din modul ser ut. Alltså vad modulen heter, samt hur alla signaler som kommer in och ut ur den ser ut. D.v.s. allt som står mellan `entity` och `end entity`.

För att detta inte ska bli fel, så får du detta tillsammans med testbänken (förkortat "tb"). Du får alltså en "start-fil" för din kod, där du ska fylla i det som står mellan `architecture` och `end architecture`.

4.4.1 Så här gör du

- Kopiera `Lab4_2.vhd` och `Lab4_2_tb.vhd`.
- Skriv in din design i `Lab4_2.vhd`.
- Starta Modelsim, som vanligt.
- Kompilera både `Lab4_2.vhd` och `Lab4_2_tb.vhd`. Rätta eventuella kompileringsfel, kompilera om etc.
- Simulera modulen `Lab4_2_tb`: `vsim Lab4_2_tb` Eller högerklicka på → "Simulate"
- Lägg till signaler i waveform: `add wave -r *` Eller gör som vanligt.
- Kör simuleringen: `run -a` Eller klicka på ikonen "run all".

Testbänken testar nu din design fyra gånger, med olika innehåll i de två PROM:en. Första gången är PROM:en fyllda med nollor, den andra gången med ettor, och de två senare innehåller PROM:en lite slumpmässig data. Innan första gången aktiveras clear-signalen.

För varje gång så skriver testbänken ut resultatet i transcript-fönstret i Modelsim. OK betyder att testet gick bra, NOK betyder att resultatet blev fel. Det är troligt att det blir ett antal varnings-meddelanden vid tidpunkt 0 ns i transcript-fönstret men dessa kan ignoreras.

När simuleringen fungerar, så använder du din kod till att göra ett CPLD-projekt, som vanligt.