Linköpings tekniska högskola, ISY, Datorteknik

Laborationskompendium Del 2

TSEA22 Digitalteknik

Linköping 2022

Laboration 3

Programmerbara kretsar och VHDL

3.1 Inledning

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

Efter genomförd laboration ska ni:

- kunna konstruera mindre digitala system med hjälp av programmerbar logik, bland annat genom att rita blockschema,
- provat på det hårdvarubeskrivande språket VHDL,
- ha insikt i modern systemutvecklingsmetodik.

3.1.1 Förberedelser

Alla uppgifter måste förberedas. I de flesta fall handlar det om att rita blockschema.

Det är dock en fördel om ni även skrivit så mycket av VHDL-koden som möjligt.

3.1.2 Examination

Utöver laborationsuppgifterna nedan, ska obligatoriska delar av datorlektionen vara redovisade, för att laborationen ska bli godkänd.

Laborationsuppgifterna redovisas i moment med namn som **BV**, vilket betyder att blockschema och VHDL redovisas tillsammans, eller **U** för uppkoppling. Alla moment för en uppgift redovisas med fördel samtidigt.

3.2 Uppgifter

Uppgift 3.3. Konstruera kodlåset från uppgift 2.2 med hjälp av VHDL och en CPLD. Implementera och koppla upp kretsen på ett av sätten föreslagna i Alternativ I, II eller III. Signalerna i det förberedda kodskelettet heter `x1` för vänster och `x0` för höger skjutomkopplare. Utsignalen heter `u`. Klocka och reset har de vanligt förekommande namnen `clk` respektive `reset`. Till er hjälp har ni erfarenheten och exemplen från datorlektionen.

Alternativ I Använd de uträknade booleska uttrycken från uppgift 2.2.
Tips: Skriv nand-grindar på formen `not (... and ... and ...)`.

Alternativ II Använd er av PROM-lösningen från uppgift 2.1.

Alternativ III Skriv VHDL-koden som motsvarar en direkt översättning av tillståndsdiagrammet för att lösa uppgift 2.2.

Ni rekommenderas att verifiera funktionen i simulering med medföljande testbänk. Avsnitt 3.3 beskriver hur testbänken fungerar.

Redovisning:

VU Redovisa **VHDL**-kod och **uppkoppling** för vald implementation.

E Under Fitter Report hittar ni en flik **Equations**, där ni ser vilka ekvationer som syntesverktyget har producerat. Jämför det med det som ni erhöll i uppgift 2.2.

Syntesverktygets ekvationer: .....

.....

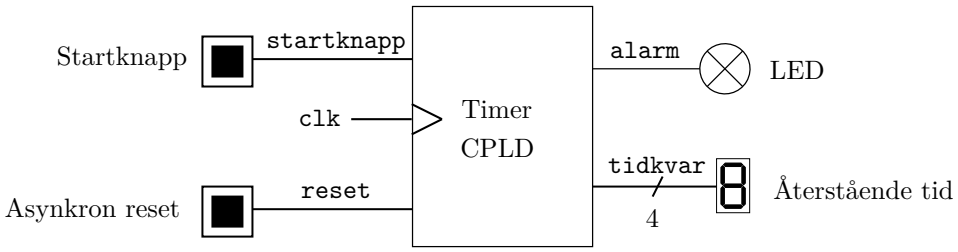
Liknar ekvationerna här SP-formerna beräknade i uppgift 2.2. Om inte vad är skillnaden?

.....

Verifiera att utsignalerna är enligt Moore. Motivera svaret:

.....

Uppgift 3.4. En timer ska konstrueras enligt följande skiss



När startknappen trycks ner ska siffran på displayen hoppa till 8 och därefter räkna ner till 0 och stanna där tills nästa gång timern aktiveras. LED-lampan ska lysa om och endast om displayen visar noll. Insignalen från startknappen måste synkroniseras och enpulsas. Timern ska inte kunna startas om med startknappen under pågående nedräkning. Använd klockmodulen med variabel frekvens och ställ in på ca 1 Hz. Det gör inget om nedräkningen börjar någon klockpuls efter det att knappen har tryckts ner. Det ska även finnas en asynkron reset som nollställer alla register oavsett om timern är aktiv eller ej. Både `reset` och `startknapp` ska vara aktivt höga.

Förberedelse: Rita blockschema och namnge alla signaler.

Laborationsuppgift: Fyll i kodskelettet genom att översätta block för block till kod och använd namnen i blockschemat i koden. Implementera kretsen och verifiera kretsens funktion.

> **Tips:** Låt dig inspireras av enpulsaren och räknarna från datorlektionen. Det är bra att simulera kretsen med tillhörande testbänk för verifiering och felsökning.

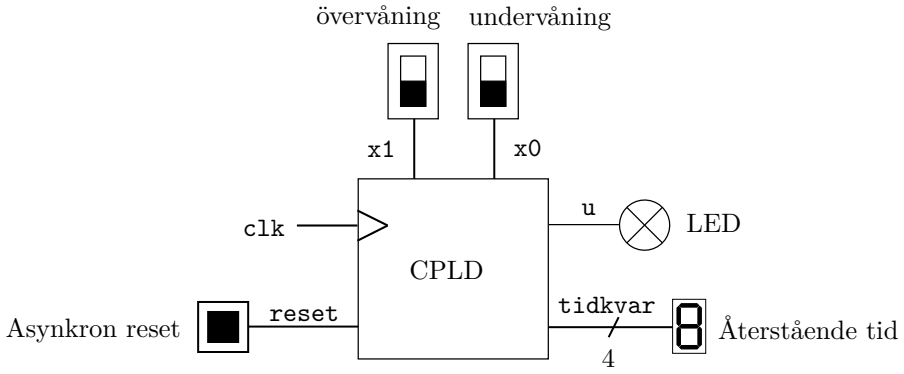
Redovisning:

BV Redovisa **blockschema** och motsvarande **VHDL**-kod.

U Redovisa **uppkopplad** krets.

Redovisad gärna även simuleringen, då det minskar behovet av att verifiera alla funktioner på uppkopplingen.

Uppgift 3.5. Konstruera och realisera en trappbelysningslogik som består av en lampa samt två omkopplare placerade högst upp respektive längst ner i trappan. Manövreras en omkopplare när ljuset är släckt skall ljuset tändas och tvärtom. Om båda brytarna byter läge i samma klockintervall ska lyset byta mod. För att spara energi skall belysningssystemet dessutom förses med en timer som automatiskt släcker ljuset cirka 15 s efter det att det tänts. Även om automatisk släckning skett ska inga extra åtgärder behöva vidtagas nästa gång man ska tända. Det ska även finnas en asynkron reset som nollställer alla register.



Använd en CPLD för att realisera kretsen, skjutomkopplare som ”strömbrytare”, en knapp för reset, en sjusegmentsdisplay för timerfunktionen och en lysdiod som lampa. Det är okej att visa återstående tid hexadecimalt.

Extrauppgift: Använd två sjusegmentsdisplayer och visa timern decimalt. Det finns ingen testbänk för denna uppgift.

Förberedelse: Rita blockschema med namngivna variabler samt fyll i kodskelettet. Kontrollera gärna att koden kan simuleras och syntetiseras korrekt.

Laborationsuppgift: Koppla upp kretsen och verifiera dess funktion.

Redovisning:

BV Redovisa **blockschema** och motsvarande **VHDL**-kod.

U Redovisa **uppkopplad** krets.

3.3 Testbänken för uppgift 3.3

Här följer en beskrivning av testbänken till uppgift 3.3 för att förstå hur en testbänk kan byggas upp och vad felen betyder. Kolla i VHDL-koden i testbänken medan du läser detta. Alla utskrifter görs på engelska, eftersom Å, Ä och Ö inte stöds.
(Not: `assert <test> report ... severity ...;` skriver ut om `<test>` *inte* är uppfyllt).

Först av allt, så skapas en klocka på 5 MHz. Då blir varje klockcykel 0,2 mikrosekunder (µs, eller us), d.v.s. 200 nanosekunder (ns). Testbänken simulerar sedan att skjutomkopplarna manövreras med 1 us intervall, d.v.s. 1000 ns.

I verkligheten manövreras skjutomkopplarna med ca 1 s intervall, vilket motsvarar 1000000000 ns, men det blir så jobbigt att läsa tidutskrifter då, så vi kör med 1000 ns istället.

Ett antal operationer/tester körs efter varandra. Siffrorna nedan indikerar numreringen i utskrifterna.

- Nollställ kretsen några klockcykler.
- Test 0: "Test to unlock normally".
  1. Sätt skjutomkopplarna till `00`. Vänta 1 us. Kolla sedan att `u=0`.
  2. Sätt skjutomkopplarna till `01`. Vänta 1 us. Kolla sedan att `u=0`.
  3. Sätt skjutomkopplarna till `11`. Vänta 1 us. Kolla sedan att `u=1`.
- Test 1: "Test that it stays unlocked".
  4. Sätt skjutomkopplarna till `10`. Vänta 1 us. Kolla sedan att `u=1`.
  5. Sätt skjutomkopplarna till `11`. Vänta 1 us. Kolla sedan att `u=1`.
  6. Sätt skjutomkopplarna till `01`. Vänta 1 us. Kolla sedan att `u=1`.
- Test 2: "Test to enter invalid code".
  7. Sätt skjutomkopplarna till `00`. Vänta 1 us. Kolla sedan att `u=0`.
  8. Sätt skjutomkopplarna till `10`. Vänta 1 us. Kolla sedan att `u=0`.
  9. Sätt skjutomkopplarna till `11`. Vänta 1 us. Kolla sedan att `u=0`.
  10. Sätt skjutomkopplarna till `01`. Vänta 1 us. Kolla sedan att `u=0`.
  11. Sätt skjutomkopplarna till `11`. Vänta 1 us. Kolla sedan att `u=0`.
  12. Sätt skjutomkopplarna till `01`. Vänta 1 us. Kolla sedan att `u=0`.
- Test 3: "Test that x=00 is tested after reset".
  - Sätt skjutomkopplarna till `01`. Ge en kort reset-puls. Vänta 1 us.
  13. Sätt skjutomkopplarna till `11`. Vänta 1 us. Kolla sedan att `u=0`.
- Test 4: "Test with one clock cycle per input".
  Detta testar att inte flera D-vippor råkar ha placerats efter varandra i tillståndsmaskinen. Här väntar testbänken på *fallande* flank på klockan – på så sätt blir det lättare att läsa av vågfönstret.
  - Sätt skjutomkopplarna till `00`. Ge en kort reset-puls, och vänta tills den är klar.
  - Vänta ytterligare två klockpulser, för säkerhets skull.
  - Sätt skjutomkopplarna till `01`. Vänta en klockcykel.
  - Sätt skjutomkopplarna till `11`. Vänta en klockcykel.
  14. Sätt skjutomkopplarna till `10`. Vänta 1 us. Kolla sedan att `u=1`.

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 laboration 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.

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 kopplingen till er VHDL-kod tydligt framgår.
- VHDL-kod som motsvarar blockschemat.
- Tydligt indenterad VHDL-kod (se Appendix B i VHDL-lektionen).
- 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

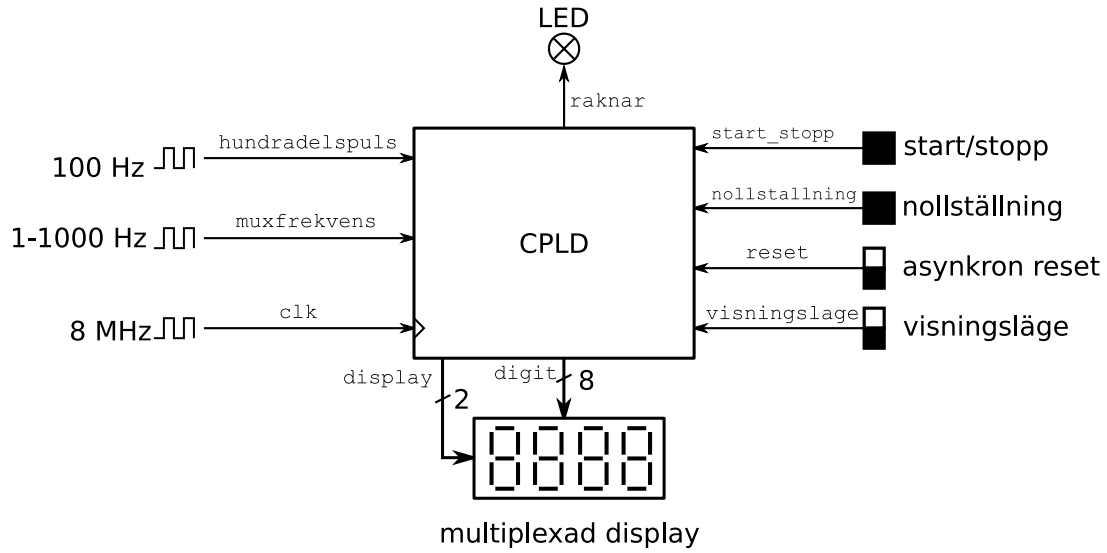
Det är **orimligt** att hinna med uppgifterna bara på laborationstid. Därför måste det mesta göras innan laborationstillfället, så att den handledda tiden kan utnyttjas väl.

Inför varje pass måste det finnas ett **blockschema** till varje planerad uppgift, med signalnamn etc inritat (t.ex. den som presenteras på föreläsningen till 4.1). Schemat ska vara översatt till VHDL, block för block. I möjligaste mån bör koden även ha simulerats och syntetiserats.

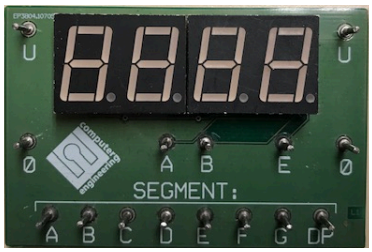
4.3 Uppgifter

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 en timme har gått börjar tiden om från 0. Den andra skjutomkopplaren ska i övre läget aktivera asynkron reset.



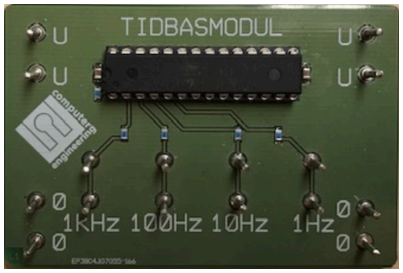
Konstruera det digitala systemet. Använd en CPLD av typen XC9572, den multiplexade displayen



en lysdiod och specificerade kappar och skjutomkopplare. Alla insignaler utom den asynkrona reset-signalen **måste synkroniseras.** För information om hur sjusegmentsdisplayen används, inklusive vilka segment som används till vilken siffra, se uppgift 1.5.

Multiplexning innebär att en siffra i taget visas. Växlas siffrorna långsamt kommer ögat se en siffra i taget. Om frekvensen ökar så kommer ögat uppfatta siffrorna som flimrande, för att sedan övergå till att uppfatta att alla siffror visas samtidigt. Använd `muxfrekvens` för att byta vilken siffra som visas.

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.

Räknaren skall starta, stanna och nollställas när respektive knapp **trycks ned**, inte när den släpps upp. Detta måste ske på ett synkront sätt, så några enstaka klockpulsers fördröjning är OK.

Förberedelser: Gör så mycket som möjligt innan själva labbpasset. Minimikrav: Genomför simuleringen av uppgift S nedan. Rita blockschema och översätt till VHDL-kod enligt BV nedan. *Försök* få kod som både syntetiserar och godkänns av testbänken i simulering.

Laborationsuppgifter och Redovisning:

S Gör en **simulering** i ModelSim över tre kaskadkopplade BCD-räknare. Välj sekundsiffrorna och första minutsiffran. (Jämför gärna med uppgift 2.3(c) på laboration 2 där ni kopplade upp tre kaskadkopplade BCD-räknare, samt uppgift D3 från datorlektionen). Rita **blockschema** för den simulerade kretsen med namngivna variabler.

Resultatet ska **redovisas** med hjälp av kod och en sparad bild från simuleringen, alternativt direkt i simuleringsfönstret i ModelSim. Det ska gå att se de rippel-carries som räknar upp 10s-siffran respektive minut-siffran.

Till den här uppgiften finns det ingen testbänk, utan bara en .do-fil. Skriv en VHDL-fil, och redigera .do-filen för att motsvara er VHDL-fil. Ni behöver visa två minut-uppräkningar.

BV Rita ett **blockschema** för hela tidtagaruret och översätt till **VHDL**-kod. (Alternativt, utgå från blockschemat från föreläsningen.)

Det kan vara bra att utveckla kretsen inkrementellt. Utgå t.ex. från den simulerade kretsen i S-uppgiften. Bygg ut kretsen med exempelvis 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 ISE.

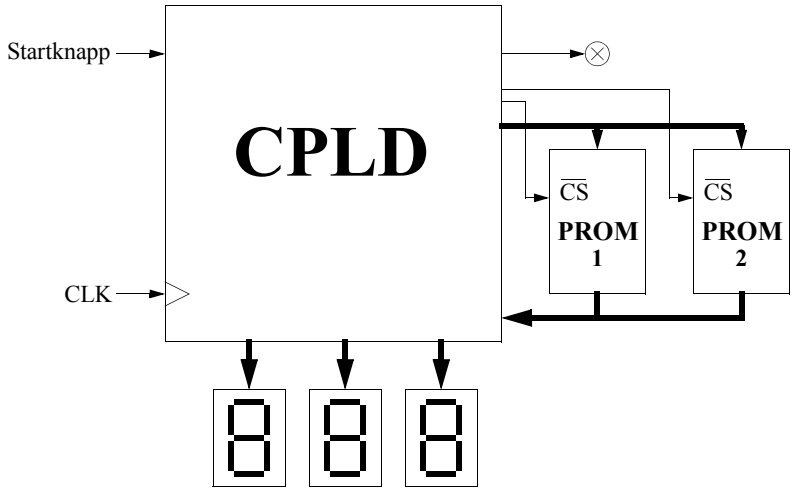
Redovisa blockschema och VHDL-kod

U Redovisa **uppkopplad** krets. Redovisa gärna simulering med testbänk också.

F Testa dig fram till den lägsta muxfrekvens där ni tycker att det inte flimrar. Vid redovisning, mät med en multimeter upp vilken muxfrekvens ni fått. Med vilken **frekvens** uppdateras hela displayen?

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 PROM:en. Klockfrekvensen bör vara max 100 Hz, ty vi vill kunna se när konstruktionen arbetar. Osynkroniserade insignaler måste synkroniseras. Eftersom adressen till PROM:en är synkron och klockfrekvensen relativt låg, kan utdata från PROM:en 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:ens "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 och laborationsuppgifter: Rita blockschema, översätt till VHDL-kod, simulera, 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, och svaren beror också på er implementation. Den existerande testbänken gör sitt bästa att täcka alla rimliga fel.

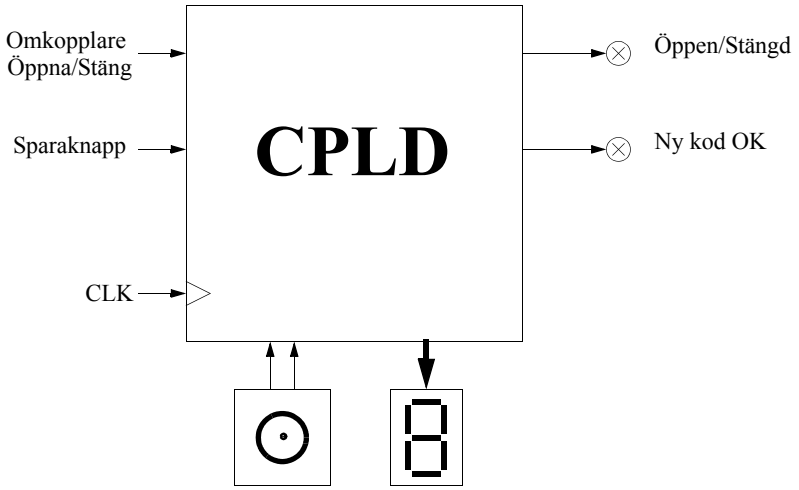
Redovisning:

S Visa resultaten av simuleringarna med testbänken. Detta är obligatoriskt, eftersom laborationsassistenten inte kommer ha tid att programmera PROM:en med så många testfall.

BV Visa **blockdiagram** med matchande **VHDL**-kod.

U Visa **uppkopplad** krets, med något demonstrativt PROM-innehåll.

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.



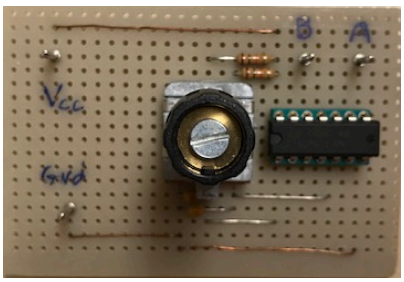
Del Gemensamt samt antingen Alternativ I eller II ingår i uppgiften.

Gemensamt 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.

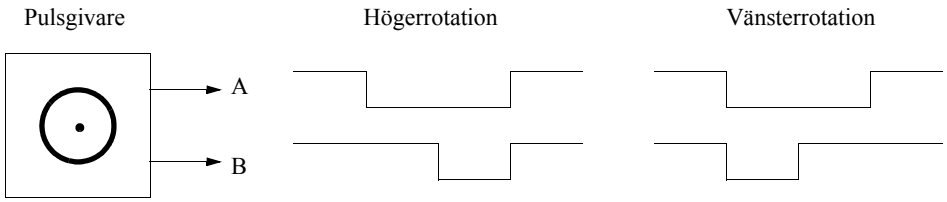
Alternativ I För att öppna låset ska först skjutomkopplaren föras till sitt nedre läge varvid en nollställning av inmatad sekvens sker. Därefter används ratten i kombination med en sparaknapp för att mata in sifferkombinationen, dvs när rätt siffra visas på displayen kan den sparas genom ett tryck på 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.

Alternativ II 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 och har följande utseende:



samt funktion:



Observera att där det i tidsdiagrammet ser ut som om signal A och B ändrar värde samtidigt kan bytet i praktiken ske en klockpuls senare i en av signalerna.

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

Laborationsuppgift: Koppla upp och verifiera kretsens funktion.

Redovisning:

BV Visa **blockschema** med matchande **VHDL**-kod för det alternativ I eller II som valts.

U Demonstrera en fungerande **uppkoppling** av valt alternativ.