Linköpings tekniska högskola, ISY, Datorteknik

Laborationskompendium

Laboration 3
TSEA51 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: .....

.....

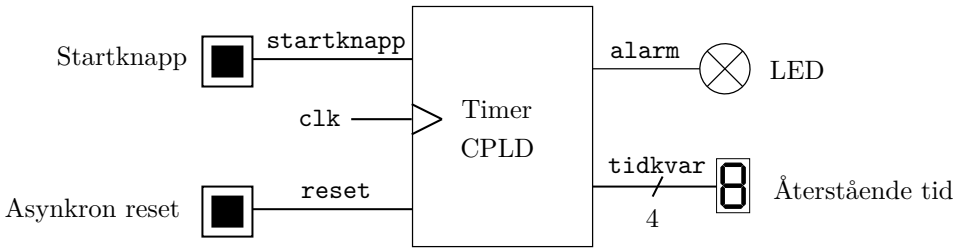
Beräkna SP-formen för er implementation för hand, om ni inte redan har gjort det i uppgift 2.2, och jämför med syntesresultatet här. Är dom lika? Om inte vad är skillnaden?

.....

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.

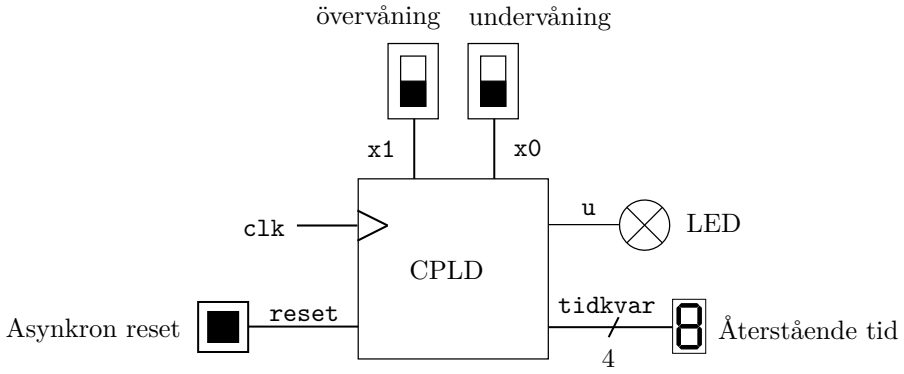
Redovisning:

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`.