Linköpings universitet | Institutionen för systemteknik
Laborationer i digitalteknik | Datorteknik
2017

Laboration i digitalteknik

– Konstruktion av sekvenskretsar med en CPLD

TSEA22 Digitalteknik D
TSEA51 Digitalteknik Y

Linköpings universitet
SE-581 83 Linköping
013-28 10 00, www.liu.se

1. Inledning

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

Efter genomförd laboration ska ni kunna konstruera sekvenskretsar, dels med enbart kretsar ur 74-serien, och dels med programmerbar logik. Ni ska även förstå grunderna i det hårdvarubeskrivande språket VHDL.

Laborationsutrustningen kommer att kompletteras med en modul som innehåller en CPLD (Complex Programmable Logic Device) tillhörandes serien XC9500 tillverkad av Xilinx. Den minsta varianten i denna serie heter XC9536 och innehåller grindar & vippor motsvarande ungefär 2 vanliga labplattor i den här kursen.

I detta lab-PM hittar ni bland annat följande avsnitt:

- Laborationsuppgifter
- En kort VHDL-introduktion som består av några kodexempel. Detta utgör ett komplement till avsnitt 1.4 i läroboken.
- En handhavandebeskrivning av syntesverktygen från Xilinx

Laborationsuppgifterna 8-10 är obligatoriska medans uppgift 11 är en extrauppgift.

2. Laborationsuppgifter

Uppgift 8: Ett kombinationslås ska konstrueras som en synkron sekvenskrets enligt Moore. Låset har två osynkroniserade insignaler och en utsignal. Insignalerna hämtas från de två studsfria skjutomkopplarna (det övre läget ger logiskt ett och det nedre logiskt noll) och utsignalen avläses på en lysdiod.

Låset öppnas om skjutomkopplarna manövreras i sekvens enligt 1 - 3

1) Båda i nedre läget
2) Vänster i nedre läget, höger i övre läget
3) Vänster i övre läget, höger i övre läget

vilket markeras av att lysdioden tänds. Låset ska förbli öppet ända tills båda omkopplarna förs till nedre läget, varefter nya öppningsförsök ska kunna göras. Vid felaktiga manövreringar ska nya öppningsförsök kunna göras först sedan båda omkopplarna förts till nedre läget.

Konstruera sekvensnätet. Använd D-vippor, NAND-grindar och inverterare. Insignalerna måste synkroniseras.

Kontrollera först funktionen genom manuell klockning från en studsfri tryckomkopplare. Anslut därefter kristalloscillatorn (på spänningsskenan) inställd på 8 MHz. För att kunna se hur nätet beter sig ska tillståndvariablerna anslutas till var sin lysdiod, alt logikprob.

Logiskt kopplingsschema:

Uppgift 9: Konstruera sekvenskretsen i uppgift 8 med hjälp av PROM och D-vippor. Ange minnesinnehållet. Notera att lösningen även nu ska vara enligt Moore.

När uppgiften är redovisad och godkänd ska PROM:en nollställas genom att i PROG-mode samtidigt trycka på de tre röda knapparna.

Logiskt kopplingsschema och minnesinnehåll i PROM:en:

Uppgift 10a: Konstruera sekvensnätet i uppgift 8 med hjälp av VHDL och en CPLD av typen XC9536. Till er hjälp har ni exempel 1 och 2 i VHDL-introduktionen i avsnitt 3. Använd de uträknade booleska uttrycken från uppgift 8. I rapportfilen *Fitter Report*, hittar ni bland annat en *Pin List* som ni behöver för att kunna koppla rätt. Syntesverktyget har valt lämpliga pinnar åt er och detta kan ändras mellan olika syntetiseringar så var uppmärksam på hur ni ska koppla.

Logiskt kopplingsschema:

Uppgift 10b: Under *Fitter Report* hittar ni även en flik *Equations*, där ni ser vilka ekvationer som syntesverktyget har producerat. Jämför det med det som ni erhöll i uppgift 8.

Resultat:……..

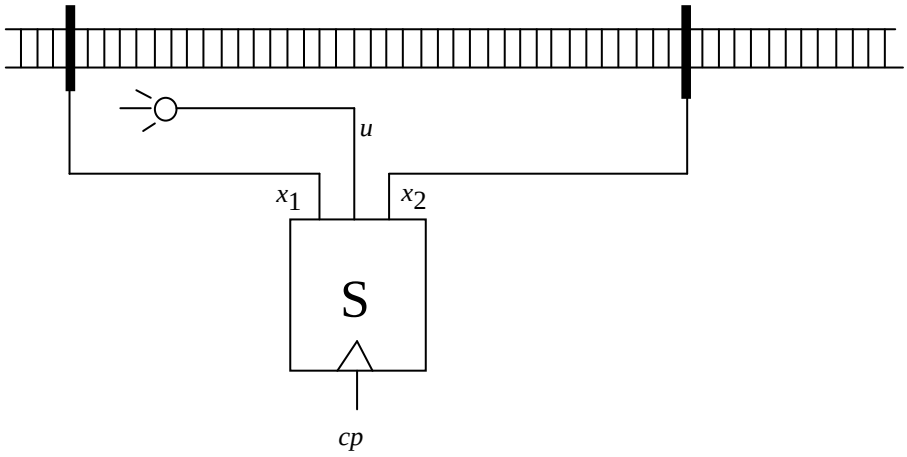
Uppgift 10c: Lös nu samma uppgift som i 10a, men i stället för booleska uttryck ska ni använda er tabell över PROM:et från uppgift 9. Till er hjälp har ni exempel 3 i VHDL-introduktionen i avsnitt 3.

Gå in i syntesverktyget och jämför de ekvationer som syntesverktyget har producerat med det som ni erhöll i uppgift 10b. Verifiera också att utsignalerna är enligt Moore.
(Uppgift 10c behöver inte kopplas upp utan är av teoretisk karaktär.)

Resultat:……..

Slutsats:……..

Uppgift 11:



På en enkelriktad järnvägssträcka (tågen rör sig från vänster till höger) vill man ha ett system, i form av en synkron sekvenskrets S, som varnar om två tåg kör för nära varandra. Man har därför i banan placerat två givare på ett inbördes avstånd som är lika med säkerhetsavståndet. Givarna lämnar signalerna x_1 och x_2. Befinner sig ett tåg över givaren är $x_i = 1$, annars är $x_i = 0$.

Om någon del av ett tåg befinner sig mellan de två givarna och ett nytt tåg når givare x_1 ska en tidigare släckt ($u = 0$) stopplampa tändas ($u = 1$). Det bakre tåget förutsätts då omedelbart tvärnita och stanna. Ett tåg är så långt, kör så sakta och stannar så snabbt att det fortfarande står över givare x_1 när det stannat. Först sedan det främre tåget helt passerat givare x_2 släcks stopplampan ($u = 0$) och det bakre tåget kan fortsätta sin färd. **(Att tända stopplampan så fort någon del av ett tåg befinner sig mellan givarna ger inte korrekt funktion)**.

Använd VHDL och en CPLD. Ersätt givarna med två studsfria skjutomkopplare. Anslut *u* till en lysdiod. För felsökningsändamål är det lämpligt att också koppla tillståndsvariablerna till lysdioder. Klockgeneratorns frekvens ska kunna varieras mellan 1 Hz - 1 kHz. Insignalerna måste synkroniseras.

Observera att ett tåg kan vara såväl längre som kortare än avståndet mellan givarna. Fallet att ett tåg är exakt lika långt som avståndet mellan givarna behöver inte beaktas.

Logiskt kopplingsschema:

3. Kort VHDL-introduktion

Börja med att läsa igenom kapitel 1.4 i läroboken. Exempel-1 nedan utgör en sammanfattning av den VHDL-kod som ni nu bör klara av att förstå, samt skriva själva.

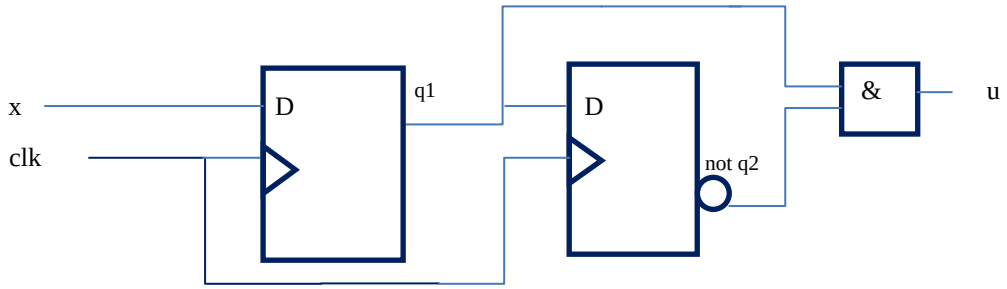
Exempel-1: Några enkla booleska funktioner.

```
library ieee;
use ieee.std_logic_1164.all;


entity K_Net is
  port(x, y, z : in std_logic;
       a, b, c, d : out std_logic);
end K_Net;

architecture equations1 of K_Net is
begin
  a <= not z;                                -- Inverterare
  b <= not (x and y);                        -- NAND-grind
  c <= ((not x) and y) or ((not y) and x); -- XOR-grind
  d <= x xor z;                              -- XOR-grind
end equations1;
```

I exempel-2 introducerar vi två nya saker:

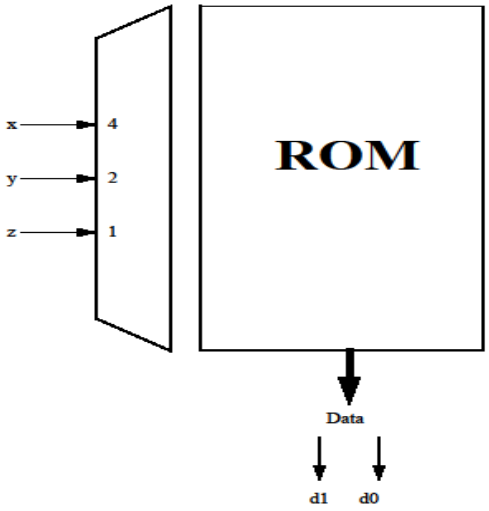
1) Vippor, vilket skapas med hjälp av en process-sats. Med en process-sats kan man göra många olika saker, men vi nöjer oss för enkelhetens skull med att bara skapa vippor. Dessa kommer att bli positivt flanktriggade enligt kodexemplet.
2) Lokala namngivna signaler (q1, q1_plus, q2, q2_plus). Dessa signaler kommer syntesverktyget att ”trolla bort”, men brukar underlätta läsförståelsen, samt gynnar ett strukturellt tänkande.

Exempel-2: En “Enpulsare”.




```
library ieee;
use ieee.std_logic_1164.all;


entity Enpulsare is
  port(clk, x : in std_logic;
       u : out std_logic;
       q1_debug, q2_debug : out std_logic);
end Enpulsare;

architecture equations2 of Enpulsare is

signal q1, q1_plus : std_logic;
signal q2, q2_plus : std_logic;

begin
  process(clk) begin
    if rising_edge(clk) then
      q1 <= q1_plus;            -- Skapar en D-vippa
      q2 <= q2_plus;            -- Skapar en D-vippa
    end if;
  end process;

  q1_plus <= x;
  q2_plus <= q1;
  u <= (not q2) and q1;

  q1_debug <= q1;          -- Vippornas tillstånd blir
  q2_debug <= q2;          -- synliga utanför CPLD:n
end equations2;
```

Exempel-3: Ett PROM med storleken 8x2 bitar




```
library ieee;
use ieee.std_logic_1164.all;

entity prom1 is
  port(x, y, z : in std_logic;
       d0, d1 : out std_logic);
end prom1;

architecture equations3 of prom1 is

signal data : std_logic_vector(1 downto 0);
signal address : std_logic_vector(2 downto 0);

begin
  address(2) <= x;
  address(1) <= y;
  address(0) <= z;
  with address select              -- Skapar ett PROM
    data <= "00"  when "000",      -- 0
            "01"  when "001",      -- 1
            "10"  when "010",      -- 2
            "10"  when "011",      -- 3
            "00"  when "100",      -- 4
            "01"  when "101",      -- 5
            "10"  when "110",      -- 6
            "11"  when "111",      -- 7
            "--"  when others;     -- Dummy

  d0 <= data(0);
  d1 <= data(1);
end equations3;
```

4. Programvara

Mjukvaran för att programera CPLD:erna består av två delar. I avsnitt 4.1 beskrivs handhavandet av programvaran för att mata in en VHDL-beskrivning och att syntetisera en fil som exakt beskriver hur den aktuella CPLD:n ska programmeras. I avsnitt 4.2 beskrivs handhavandet av den mjukvara som programmerar CPLD:n.

En enklare gratisversion finns att ladda ner från Xilinx hemsida, kallad: ISE WebPACK.

4.1. XILINX ISE Project Navigator

Är ett verktyg för att syntetisera VHDL till programerbara kretsar så som CPLD och FPGA.
Här följer en kortfattad handhavande beskrivning:

- Starta ISE Project Navigator (ISE Design Suite 14.5)

- Välj *File - New - Project*. En dialogruta öppnas och du väljer ett filnamn (namn på projektet), en katalog (där du vill spara projektet) samt *HDL Top-level Source Type*. (Exempelvis kan katalogen döpas till vhdl-lab och projektet till uppgift8.)

- Klicka *Next* och dialogrutan *Project Settings* öppnas. Välj Family: XC9500 CPLDs, Device: XC9536 eller XC9572 enligt anvisningar om tillåten komponent, Package: PC44

- Klicka *Next* och därefter *Finish*.

- Välj *Project - New - Source*. En dialogruta öppnas och du väljer ett filnamn samt *VHDL Module*.

- Klicka *Next* och dialogrutan *New Source Wizard* öppnas. Här bestämmer du vilka in- och ut-signaler din konstruktion ska ha. (Detta kan i ett senare skede ändras om så behövs.)

- Klicka *Next* och därefter *Finish*.

- I den källkodsfil som nu öppnas ska du skriva in dina ekvationer på samma sätt som exemplen har visat.

- När källkodsfilen är klar välj *File - Save*

- Välj *Process – Implement Top Module* för att syntetisera koden. En ”jed-fil” skapas, samt en rapportfil dyker upp på skärmen. I rapportfilen kan ni titta på *Pin List*, för att se hur ni ska koppla er konstruktion. Om *Fitter Report* inte dyker upp kan ni öppna den via fönstret *Design Summary*.

- Jed-filen ska flyttas över till en annan katalog för att bli tillgänglig på den dator där kretsen programmeras. För detta ändamål finns det monterat en katalog på din dator som heter ”B:”. Här skapar ni en egen mapp åt er där ni kan placera era jed-filer.

- RTL-viewer, är en funktion där ni kan se hur er konstruktion blev efter syntetiseringen. Välj *Tools – Schematic Viewer – RTL*, samt följ vidare instruktioner. Här får ni reda på hur syntesverktyget har tolkat er konstruktion på blocknivå.
  Välj därefter *Tools – Schematic Viewer – Technology*. Då får ni se hur er konstruktion blir implementerad på CPLD:n. På alla in- och ut-gångar finns en extra komponent som kallas buffer. Den har till uppgift att elektriskt skydda CPLD:n samt säkerställa en hög signalkvalité. Det logiska värdet på bufferns in- och ut-signal är lika.

4.2. XILINX ISE iMPACT

Är ett verktyg för att programera bland annat CPLD:er. Be gärna en handledare om att vara med första gången.

- Kontrollera först att spänningsaggregatet är av.
- Sätt därefter i kretsen åt rätt håll, samt slå på spänningsaggregatet.
- Starta programmet iMPACT samt svara på frågorna med ”OK”, ”No”, ”No” och ”Cancel”.
- Dubbelklicka på *Boundary Scan*.
- Välj *Initialize Chain* så att verktyget hittar kretsen.
- Kilcka på kretsen och välj *Assign New Configuration File*. Här ska du leta upp din ”jed-fil”.
- Kilcka på kretsen igen och välj *Program*, samt därefter *OK*.
- Slå slutligen av spänningsaggregatet och ta ut kretsen genom att trycka på ramen runt kretsen.