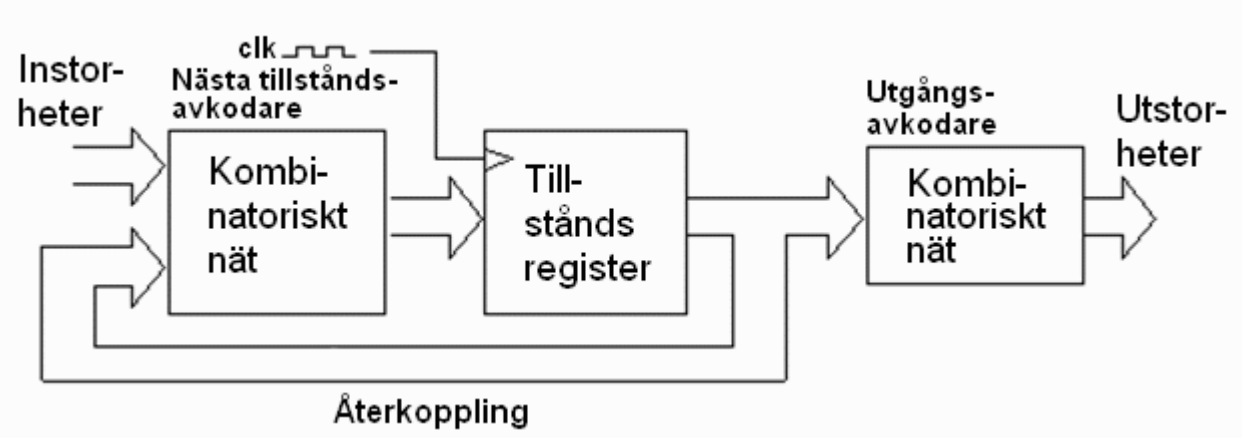
Laboration VHDL introduktion

Digital Design IE1204 (Observera! Ingår *inte* för IE1205)



Observera! För att få laborera måste Du ha:

- bokat en laborationstid i bokningssystemet (Daisy).
- löst ditt personliga web-häfte med förkunskapsuppgifter som hör till laborationen.
- gjort alla förberedelser och förberedelseuppgifter som nämns i labhäftet.

Vid laborationen arbetar ni i grupper om två studenter, men båda studenterna ansvarar var för sig för förberedelserna och för genomförandet.
Ha med er var sitt labhäfte till laborationen. Framsidan används som ditt **kvitto** på att laborationen är genomförd. Spar kvittot tills Du fått hela kursen bokförd i Ladok.

*Eftersom detta är ditt labkvitto **måste** Du fylla i tabellen med bläck.*

Namn:			
Personnummer:			
• **Förkunskapstest (Web-frågehäfte)**			
Häfte nr:		Datum:	
Lab-lärarens kvittens:			
• **Förberedelseuppgifter i labhäftet**			
Lab-lärarens kvittens:			
• **Laborationens genomförande**			
Laborationen utförd den:			
Lab-lärarens kvittens:			

Inledning

Denna laboration handlar om hur man designar digital logik med VHDL-språket och moderna CAD-programvaror. Meningen är att Du ska få en inblick i hur en "Digitaltekniker" arbetar. VHDL-språket är ett mycket komplext programmeringsspråk, och det är inte rimligt att "lära sig" detta till denna första korta Digital Designkurs.
När Du löser laborationsuppgifterna har Du därför fått tillgång till Tutorials och Mallprogram på kurswebben.

Skolan har sedan flera bra VHDL-kurser som kan väljas av den som vill veta mer, och som vill kunna arbeta med Digital Design i framtiden.

Det bästa sättet att lära känna ett program är att installera det på den egna datorn. Då kan man i lugn och ro söka igenom menyer och hjälpsidor, och kan ta sig den tid det tar att reda ut det man eventuellt har missförstått.

Om "datorkrångel" riskerar att förbruka för mycket tid för dig, kan Du också hitta programmen installerade i skolans datorsalar - som en backup-lösning.

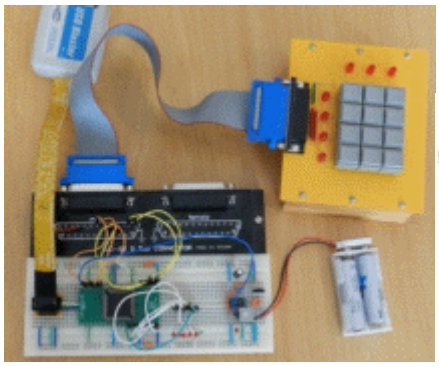
Målet med laborationen

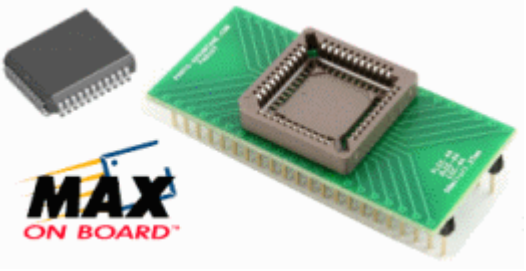
- Bekanta dig med moderna CAD-program, Quartus och ModelSim.
- Visa hur man simulerar en digital konstruktion (en Moore-automat), hur man genererar insignaler sk. stimuli, och hur man observerar utsignaler och beteende (ModelSim-Wave).
- Orientera dig om hur en digitaltekniker kan skriva en VHDL "testbänk" för att försäkra sig om att en konstruktion är helt korrekt.
- Praktisera VHDL-konstruktion från givet tillståndsdiagram (dvs. omarbeta och utöka ett givet mallprogram).
- Praktisera hur man knyter samman konstruktionens "signaler" med målchippets "pinnar".
- Visa hur man programmerar målchippet (MAX3000), och provar funktionen i verkligheten.

Observera! Det kan hända att din laborationstid ligger före det att alla kursmoment som kan behövas för laborationen har förelästs. Du måste i så fall själv läsa på i förväg - det finns länkar till alla föreläsningar och övningar.

Observera! Labutrustningen är färdigbyggd. Inga ledningar ska ändras, inga läggas till eller tas bort.

Ett VHDL-kodlås





Laborationens uppgift är att konstruera ett kodlås som öppnar för en unik fyrsiffrig kod, men vi börjar med att studera ett enklare mall-program, ett kodlås som öppnar för en tangent!

- **Förberedelseuppgift 1 (görs hemma innan lab)**

Installera programmen **Quartus II** och **ModelSim** på din egen dator.

Följ beskrivningen i tutorial på kurswebben - installera programmen på din dator.

- **Labuppgift 1 (görs vid lab i skolan)**

Logga in på labdatorn. På skolans centralt administrerade labdatorer har Du inte rättigheter att installera program. **Quartus II** och **ModelSim** finns redan installerade. Du har inte heller åtkomst till mappar under `C:\`. Vid laborationer ska Du därför använda din "server"-mapp `H:\`.

- Skapa en mapp `H:\MAXWork\` för filerna i denna lab.
- Viktig operativsystem-inställning. Visa filnamnstillägg bör vara inställt vid alla programmeringskurser! Windows 7 show fileextensions

- **Förberedelseuppgift 2 (görs hemma innan lab)**

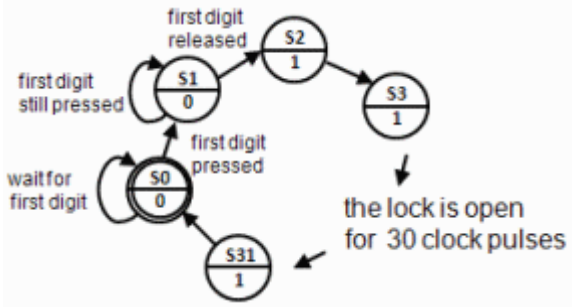
Starta Quartus och skapa ett projekt `codelock`. Ta med innehållet i filen lockmall.vhd som projektets VHDL-fil och kompilera därefter koden.

Följ steg för steg beskrivningen i tutorial på kurswebben - VHDL-program med Quartus.

Läs på om mall-programmets VHDL-kod i beskrivningen på kurswebben
- VHDL för ett kodlås.

Läs på om hur man knyter signalnamn till specifika pinnar för sitt chip i Quartus.
- Pin-planering i Quartus.

Läs på om hur man använder Quartus programmeringsfunktion med en JTAG USB-Blaster.
- Chip-programmering med Quartus.

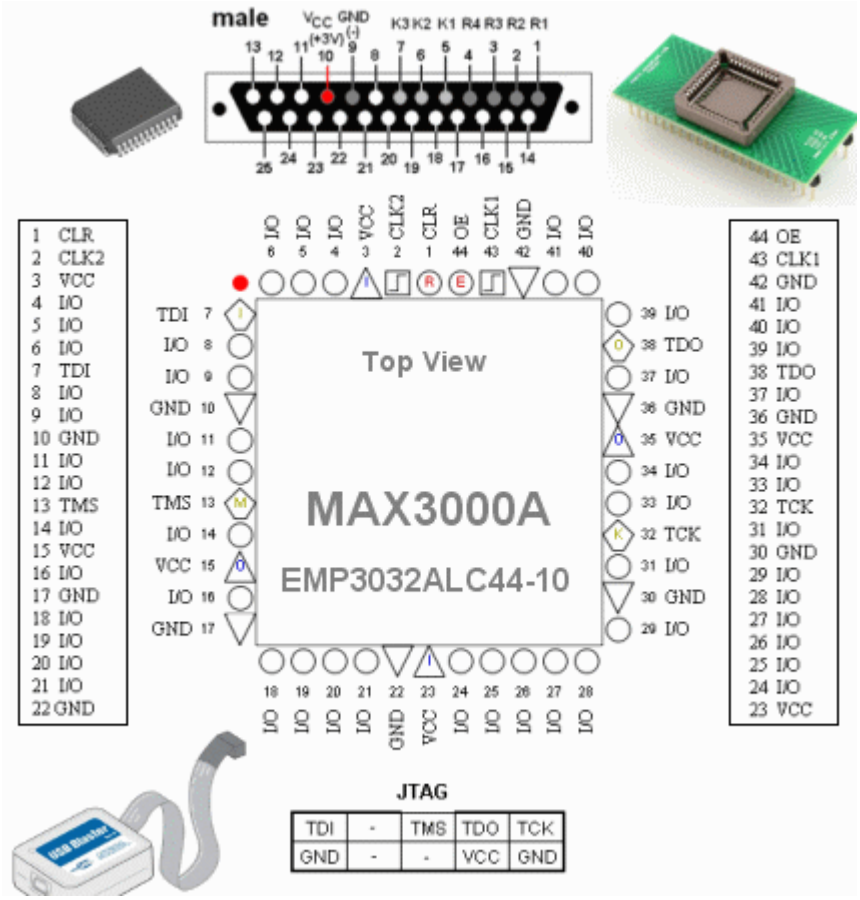


- **Labuppgift 2 (görs vid lab i skolan)**

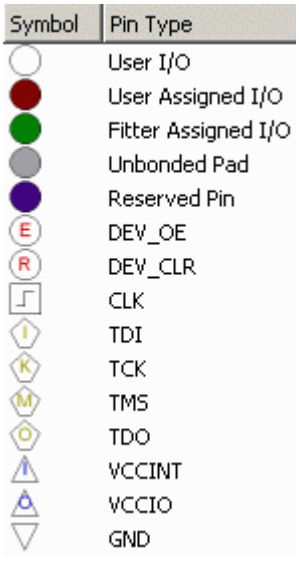
- Starta Quartus och skapa ett projekt `codelock` i din "server"-mapp `H:\MAXWork\`. Använd innehållet i filen lockmall.vhd som projektets VHDL-fil och kompilera därefter koden.
- Labutrustningarna har olika ledningsdragning! Undersök din labutrustning och fyll i Pin-planeringstabellen nedan, och därefter **Pin Planner** i Quartus.

Följ beskrivningen i tutorial på kurswebben - Pin-planering i Quartus

Node Name	Direction	Location
clk	Input	PIN_
K[1..3]	Input	
K[1]	Input	PIN_
K[2]	Input	PIN_
K[3]	Input	PIN_
q[4..0]	Output	
q[4]	Output	PIN_
q[3]	Output	PIN_
q[2]	Output	PIN_
q[1]	Output	PIN_
q[0]	Output	PIN_
R[1..4]	Input	
R[1]	Input	PIN_
R[2]	Input	PIN_
R[3]	Input	PIN_
R[4]	Input	PIN_
UNLOCK	Output	PIN_



Beskrivning av pinn-symbolerna se figuren.



- När Du gjort klart pin-planneringen i Quartus så kompilerar Du om projektet.
- Programmera utrustningen med USB-Blastern.
- Kontrollera att kodlåset öppnar när man trycker ner "1" och sedan släpper tangenten.

- **Förberedelseuppgift 3 (görs hemma innan lab)**

Hemma har Du ingen hårdvara, dvs. ingen labutrustning. I sådana situationer brukar man simulera sin kod för att se om den är riktig.

Det ledande simuleringsprogrammet **ModelSim** finns i en version för Alteras kretsar. Starta ModelSim och simulera VHDL-koden med innehållet i `lockmall.vhd` som VHDL-fil.

Följ beskrivningen i tutorial på kurswebben - Simulera med ModelSim.

 lockmall.vhd  lock.do

- **Labuppgift 3 (görs vid lab i skolan)**

Även när man har tillgång till hårdvara är det vanligt att man blandar simuleringar med hårdvarutest.

Genomför samma simulering i skolan som Du övat på hemma, dvs. visa i wave-fönstret att låset öppnar för "1". Visa labassistenten Ditt "simuleringskunnande".

- **Förberedelseuppgift 4 (görs hemma innan lab)**

Design av Digital hårdvara kan ofta resultera i att man bestämmer sig för att producerar en ASIC - en Applikation Specifik Integrerated Circuit. Det handlar då ofta om flera månaders framtagningstid och tillverkningskostnader av storleksordningen många miljoner kronor.

Då måste man kunna vara säker på att konstruktionen är helt korrekt!

(Vid labben har vi ett bättre utgångsläge än ASIC-konstruktören. Om din konstruktion på en programmerbar CPLD-krets är fel, får Du chansen att programmera om den, - om och om igen.)

Som Du kan förstå är det är testingenjören som är Digitalteknikens hjälte!

VHDL-språket har olika hjälpmedel för att man lättare ska kunna skriva korrekt kod.

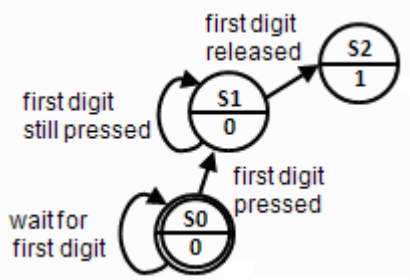
- För att minska risken för misstag när man överför information från datablad och andra förlagor, kan man tex. använda index som löper uppåt eller nedåt, så att det passar framställningsättet som använts på de förlagor man har.
- Man har också möjlighet att skapa egendefinierade datatyper som passar den beskrivning man har av konstruktionen. Man kan därför ofta skriva VHDL-kod som är "uppenbart" korrekt!

- Man kan skriva en VHDL-testbänk. Detta är Simuleringskod som kan användas till att testa många/alla signalkombinationer som kretsen kan komma att utsättas för.

Observera! En testbänk är oftast ett mer komplicerat program än den ursprungliga konstruktionen som den avser att testa!

Dessa rader ur Mall-programmet är exempel på kod som *uppenbart* följer det givna tillståndsdiagrammet.

```
case state is
      when 0 => if (K = "001" and R ="0001")     then nextstate <= 1;
                else nextstate <= 0;
                end if;
      when 1 => if (K = "001" and R ="0001")     then nextstate <= 1;
                elsif (K = "000" and R = "0000") then nextstate <= 2;
                else nextstate <= 0;
                end if;
                . . .
```



Om man däremot tar över kod från någon annan, även om den personen *lovar* att den fungerar, är situationen en annan.

```
case state is
      when 0 => if(((R(2)='0') and (R(3)='0') and (K(2)='0') and (K(3)='1'))
and
                   ( not (( not ((K(1)='0') and (R(1)='0') and (R(4)='1')))
and
                   ( not ((K(1)='1') and (R(1)='1') and (R(4)='0'))))))
                then nextstate <= 1;
                else nextstate <= 0;
                end if;
                . . .
```

Här är vilkoren skrivna på ett sådant sätt att det inte längre uppenbart vad koden gör - och därför vet vi inte om den är korrekt eller om den, alla löften till trots, är felaktig?

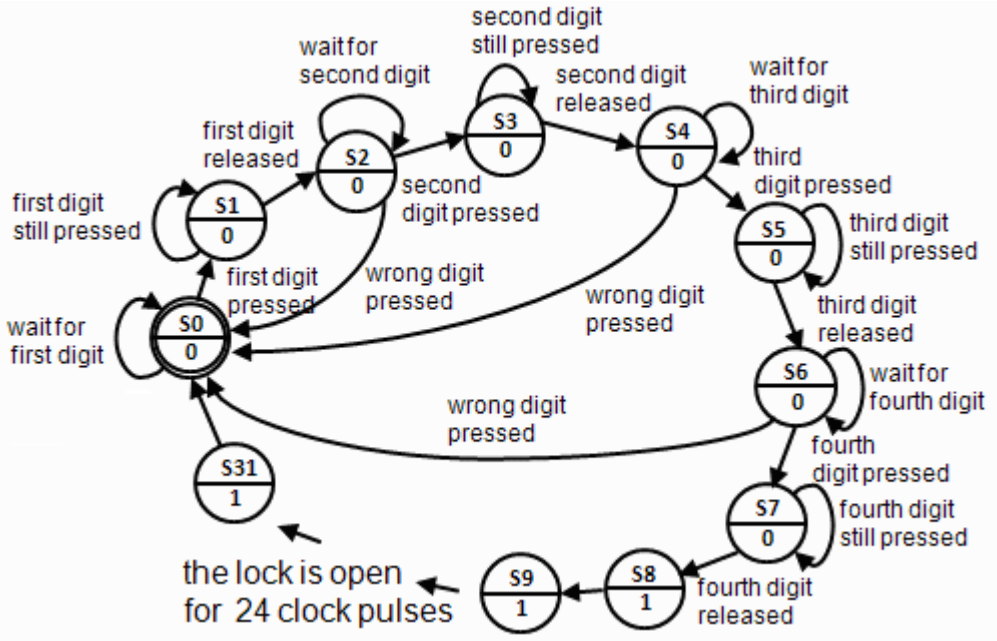
Prova nu kodens korrekthet med en (färdigskriven) testbänk
Följ beskrivningen i tutorial på kurswebben - Testbänk i ModelSim.

lockmall.vhd lockmall_with_error.vhd tb_lockmall.vhd

- **Labuppgift 4 (görs vid lab i skolan)**

- Genomför även simuleringen med testbänken i skolan, och visa labassistenten dina färdigheter med att simulera kod med en testbänk.
- Har Du kunnat avslöja något fel med koden? Visa labassistenten.
- Avsluta **ModelSim** och byt till programmet **Quartus II**. Där byter Du ut innehållet i VHDL-filen från `lockmall.vhd` till `lockmall_with_error.vhd`. Kompilera och ladda sedan ned koden till MAX-chippet.
- Kontrollera om det misstänkta fel-beteendet från simuleringen innebär att det i praktiken går att öppna kodlåset på felaktigt sätt?

- **Förberedelseuppgift 5 (görs hemma innan lab)**



Ett kodlås som öppnar för en enda knapptryckning är ju naturligtvis löjligt enkelt att forcera, normalt brukar man ha en fyrsiffrig kod. Din uppgift blir därför att utveckla mallprogrammet (`lockmall.vhd`) till ett sådant lås med fyra siffror - de fyra sista siffrorna i ditt personnummer.

Förbered ett sådant VHDL-program hemma. Kontrollera att det går att kompilera. Tag med dig koden till skolan på något sätt, tex.:

- Maila texten till dig själv.
- Ta med dig ett USB-minne med koden som en textfil.
- Föra över koden till din servermapp `H:\`.

Har Du möjlighet att simulera koden hemma ökar Du sannolikheten för att Du har en korrekt kod att utgå från i skolan.
Tiden vid laborationen i skolan kommer *inte* att räcka till att skriva programkoden från "scratch"!

- **Labuppgift 5 (görs vid lab i skolan)**

Gör ett kodlås som öppnar för en fyrsiffrig kod. Labassistenten bestämmer sifferkombinationen så att - två av siffrorna tas från ditt förberedelseprogram, och de andra två från din lab-kollegas förberedelseprogram.

- Visa det fungerande kodlåset för labassistenten.

Har Du tid över?

Om Du har förberett Dig väl, och om Du har tidigare erfarenhet av något programmeringsspråk, så har Du förmodligen nu tid över för en "frivillig" uppgift.

- Kan Du förändra programmet så att det ***även öppnar*** för den tidigare *dolda* knapptryckningskombinationen?
  (så att det finns en så kallad "hårdvarutrojan" inuti chippet)

Lycka till!

När Du är klar städa labplatsen.

Material-lista

Om Du någon gång skulle behöva bygga en liknande experimentutrustning, kan Du här se vilka komponenter vi använt.

```
Altera USB-blaster ELFA 73-898-90
ALTERA CPLD EPM3032ALC44-10
Proto Advantage PLCC-44 Socket to DIP-44 Adapter
Kopplingsdäck MB-85 ELFA 48-428-37
Kontaktdonsövergång 25-pol D-sub ELFA 48-426-96
Stiftdon (JTAG) ELFA 43-155-03
Lysdiod med seriemotstånd 5V röd  ELFA 75-012-59
Lysdiod med seriemotstånd 5V gul  ELFA 75-015-11
Elektronikkrets 555 ELFA 73-042-65
Trimpot 500 kO med ratt ELFA 64-635-25
```

Laborationen är sammanställd av William Sandqvist william@kth.se