Laboration i digital konstruktion I: Lab 1

Bengt Oelmann, Mitthögskolan, ITE

1.0 *Tutorial* : En genomgång av verktyg och prototypkort

1.1 Syfte

Den här laborationen är av typen “demonstration” och har till syfte att visa utvecklingsverktygen för FPGA prototypframställning som kommer att användas i de kommande laborationerna i kursen. Utvecklingsmiljön består av VHDL-kompilator, VHDL-simulator, FPGA syntesverktyg och FPGA prototypkort.

1.2 Förberedelser

```
mkdir ~/VHDL
cd VHDL
mkdir lab0
cd lab0
```

1.3 Utförande

Laborationen görs genom att följa en detaljerad instruktion som beskriver steg för steg vad som ska göras.

1.3.1 VHDL simulator

Följ länken “Quick HDL Tutorial” som nås via web-sidan http://www.ite.mh.se/~bengt/digitalk.htm.

1.3.2 FPGA syntes

Följ instruktionen “FPGA synthesis tutorial” som nås via web-sidan http://www.ite.mh.se/~bengt/digitalk.htm.

1.4 Redovisning

Ingen laborationsrapport.

2.0 Första konstruktionen : Adderare

2.1 Syfte

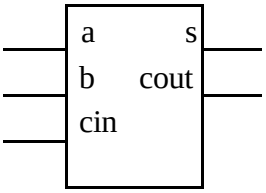
Den här laborationen visar hur man i VHDL kan välja olika sätt att beskriva samma funktion och hur de olika beskrivningarna kan verifieras mot varandra i en och samma testbänk.

2.2 Förberedelser

```
cd ~/VHDL
mkdir lab1
cd lab1
qhlib work
cp ~kurser/digkon1/lab/VHDL_kod/add8_tb.vhdl .
```

2.3 Utförande

2.3.1 Skriv en syntetiserbar modell av en 1-bits full-adderare (fa) i text-editorn *emacs*. Kalla filen för *fa.vhdl*



Modul-gränssnittet ska vara enligt följande:

```
entity fa is
port ( a, b, cin : in std_logic;
       s, cout : out std_logic);
end fa;
```

Datatypen `std_logic` finns definierad i `IEEE.STD_LOGIC_1164`.

2.3.2 Simulera full adderaren

Kompilera källkoden (fa.vhdl)

```
qvhcom fa.vhdl
```

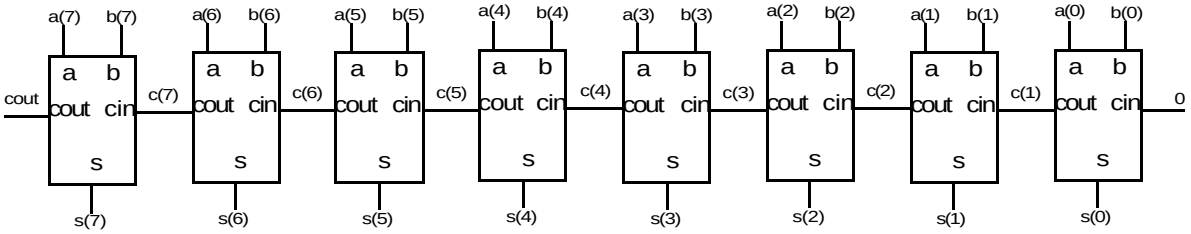
starta simulatorn

```
qhsim
```

Testa samtliga in-data kombinationer.

2.3.3 Strukturell beskrivning av en 8-bitars adderare

Bygg en strukturell beskrivning av en 8-bitars adderare (*add8*) m.h.a full-adderaren. Genom att kaskadkoppla 8 stycken 1-bits adderare kan vi bygga upp en adderare som kan addera två stycken 8-bitars tal.



2.3.4 Modul-gränssnittet ska vara enligt följande:

```
entity add8 is
port ( a, b : in std_logic_vector(7 downto 0);
       cin : in std_logic;
       s : out std_logic_vector(7 downto 0);
       cout : out std_logic);
end add8;
```

Eftersom vi ska skriva två stycken olika beskrivningar av *add8* där modul-gränssnittet är samma (d.v.s `entity` deklarationen) så lägger vi `entity` deklarationen i en separat fil. Kalla filen för *add8_entity.vhdl*

Skriv en strukturell beskrivning av *add8*, kalla arkitekturen för *struct* och spara den i filen *add8_struct.vhdl*

2.3.5 Verifiera den strukturella beskrivningen

Kompilera *add8_struct.vhdl* och läs in den i simulatorn för att testa ett par olika kombinationer av indata för att se att den fungerar.

2.3.6 Skriv in en syntetiserbar beteende-beskrivning av *add8*

Kalla arkitekturen för *rtl* och spara den i filen *add8_rtl.vhdl*. Addtitionen görs med + *tecken* av de två bitvektorerna.

2.3.7 Verifiera beteende-beskrivningen

Kompilera *add8_rtl.vhdl* och läs in den i simulatorn för att testa ett par olika kombinationer av indata för att se att den fungerar.

2.3.8 Verifiera att den strukturella beskrivningen och beteende beskrivningen är ekvivalent.

Ta in den färdiga testbänken *add8_tb.vhdl* i *emacs*. Förklara vad den gör.

Kompilera *add8_tb.vhdl* och ladda in den i simulatorn. Simulera. Vad händer ?

2.4 Redovisning

VHDL-koden och simuleringsresultat för den strukturella beskrivningen och beteende-beskrivningen visas för lab. handledaren vid lab. tillfället.