Linköpings universitet | Institutionen för systemteknik
Laborationer i digitalteknik | Datorteknik
2018

Simulering med ModelSim

– En kort introduktion

TSEA22 Digitalteknik D
TSEA51 Digitalteknik Y
TSEA52 Digitalteknik I

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

Simulering i ModelSim

Följande dokument beskriver steg för steg hur en VHDL-modell simuleras i Modelsim.
Sist i dokumentet finns några övningsexempel.

1. Skapa katalogen ”H:/TSEAXX/” där TSEAXX byts ut mot aktuell kurskod.
2. Kopiera zip-arkivet från kurshemsidan till ”H:/TSEAXX/” och välj Extract Here.

3. Starta Modelsim
Start → All Programs → ModelsimSE.. →Modelsim
- Close

4. Skapa ett projekt
File → New → Project
- Project name: “XOR_sim”
- Project Location: ”H:/TSEAXX/VHDL_lektion/XOR”
- OK
- Add Existing File: ” H:/TSEAXX/VHDL_lektion/XOR/XOR_gate.vhd”
- OK
- Close

I fliken **Library** finns en lista på alla tillgängliga bibliotek och dess tillhörande objekt. Konstruktion finns under ”work”. I fliken **Project** listas alla VHDL-filer som finns i det nuvarande projektet.

5. Kompilering
Compile→Compile all

6. Simulering
För att simulera kretsen måste insignaler specificeras. Detta kan antingen göras med GUI:t, via transcript-fönstret eller genom att skriva in de kommandon som ska köras i ett makro, en .do-fil, som sedan anropas från transcriptfönstret. Börja med att följa stegen i guiden för GUI:t. När ni kör kommandona från GUI:t kommer motsvarande textkommandon att dyka upp i transcriptfönstret. Skapa sedan en fil sim.do där ni klipper in de kommandon ni vill köra.

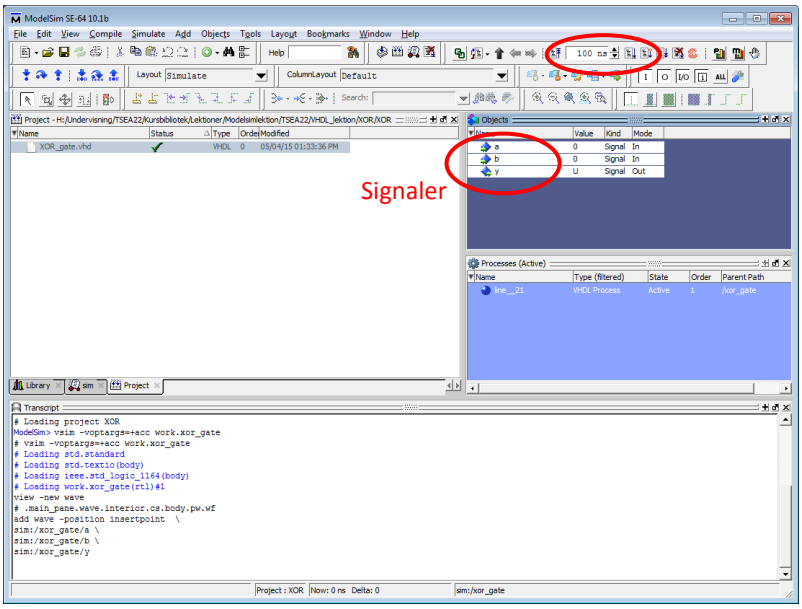
GUI
- Library**:** work/xor_gate
- Höger klick → *Simulate*

- Objects: markera alla signaler
- Höger klick → *Add Wave*

- Wave: högerklicka på “a”, välj ”Clock” : offset=”25ns”, Duty=”50”, Period=”100ns” . **OK**
- Wave: högerklicka på “b”, välj ”Force”: Value=”0”, **OK** ”Force” Value”1”, ”Delay for”=”200ns”, **OK**
- Ändra ”Run Length” till 400 ns

- Klicka på ”run” 

Run Length och run-knapp



Do-fil

Prova nu att göra om simuleringen genom att skriva kommandona i en .do-fil. I XOR-katalogen finns en tom sim.do-fil. Öppna den:

- *File → Open*
- *Välj Do Files (*.do)*
- *Välj sim.do*

Skriv in följande rader kod i sim.do. Dessa svarar mot GUI-inmatningarna gjorda innan.

```
vsim xor_gate
add wave sim:/xor_gate/*
force -freeze sim:/xor_gate/a 1 25, 0 75 -r 100
force -freeze sim:/xor_gate/b 0 0, 1 200
run 400
```

Spara sim.do.
Kör filen genom att i transcript-fönstret skriva

```
> do sim.do
```

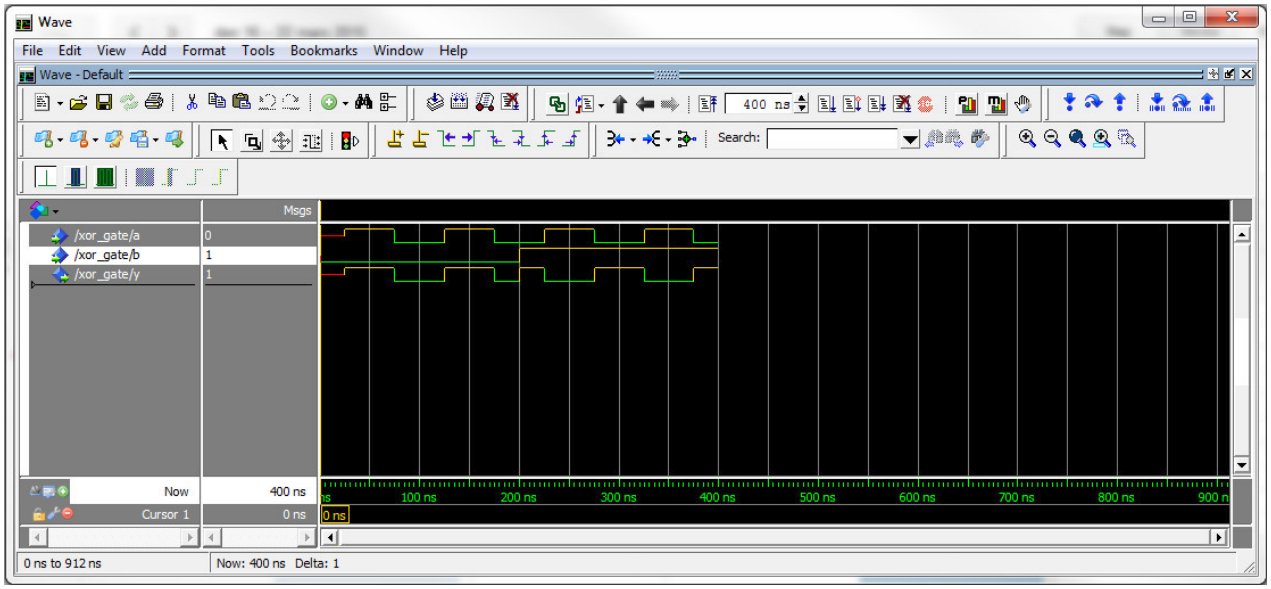
Om man är nöjd med simulerade variabler men vill t ex modifiera insignalerna så behöver inte simuleringen startas om helt. Då räcker det med att använda restart-kommandot enligt följande kod:

```
# vsim xor_gate
# add wave sim:/xor_gate/*
restart –force
force -freeze sim:/xor_gate/a 1 25, 0 75 -r 100
force -freeze sim:/xor_gate/b 0 0, 1 200
run 400
```

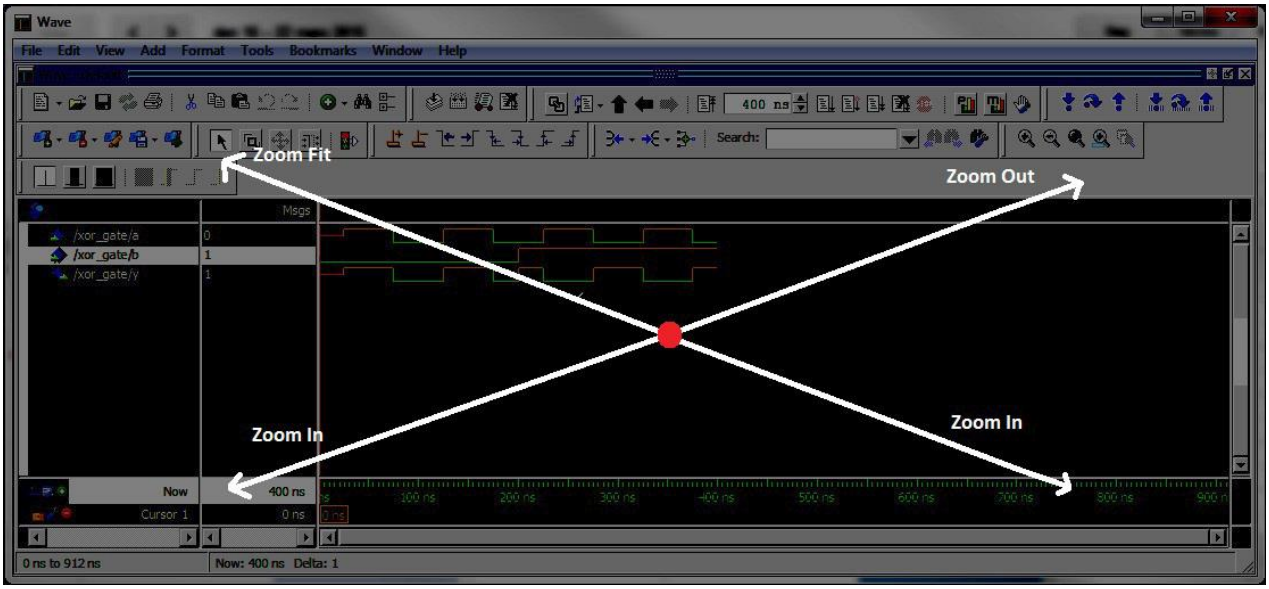
Här har rader i koden kommenterats bort med #. Prova att ändra insignalerna genom att modifiera do-filen och simulera xor-grinden med de nya insignalerna.

6. Visa resultat

Resultatet av en simulering visas i Figur 1. Med vänster musknapp flyttas markören, en gul vertikal linje. Värdena vid markerad tidpunkt visas till höger om signalnamnen. Figur 2 visar vågfönstrets zoom-funktionalitet.



Figur 1. Resultatet av simuleringen.



Figur 2. Wave-fönstrets zoom-funktionalitet. Klicka på musens scrollhjul och rör musen i riktning längs pilarna för att få motsvarande funktion.

Mäta tid

För att mäta tid mellan två tidpunkter behöver man lägga till en markör. Detta görs på följande sätt.

- Lägga till markör: *Add → Cursor*

Dividers

I variabellistan kan det vara trevligt att dela in variabler i t ex in- och ut-gångar. Det finns möjlighet att lägga till så kallade dividers, en slags rubrikrader i signallistan, på följande sätt.

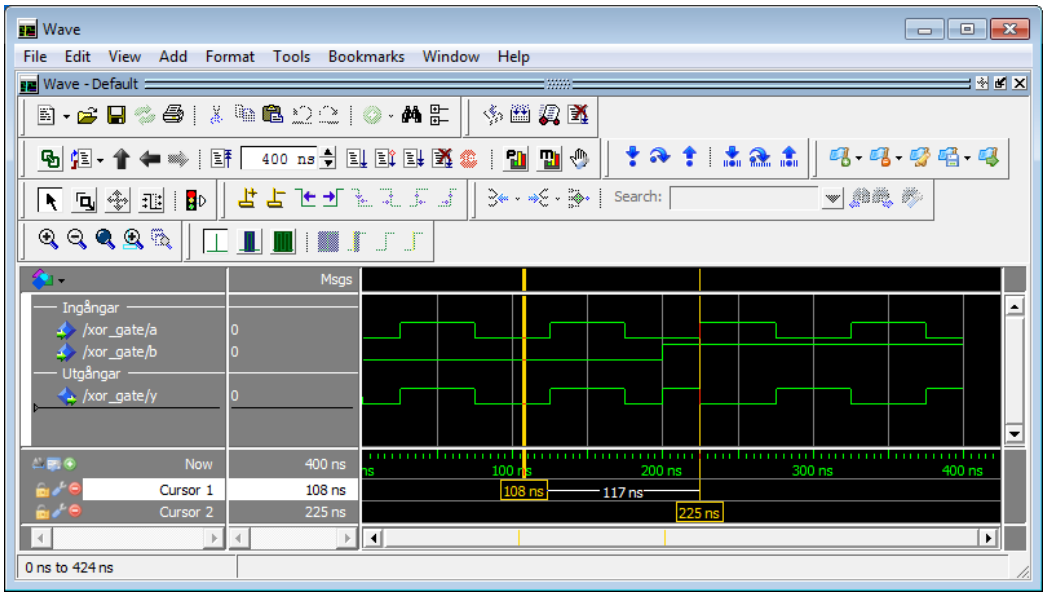
- Högerklicka på signalen ”a” → *Add →New Divider*
  Divider Name=”Ingångar”
- Högerklicka på signalen ”y” → *Add →New Divider*

Divider Name=”Utgångar”

Byta färg för logiska nivåer

I Figur 1är hög signalnivå indikerad med guldfärg. Detta erhålls genom att göra följande inställning:

- ModelSim fönstret: *Tools→Edit Preferences.. → Wave Windows → LOGIC_1 → Palette → Gold*



7. Modifiera VHDL-filen

Antag att ni vill ändra utsignal z så att den inverterar utsignal från XOR-grinden, dvs z = not y. Börja med att beskriva den modifierade utsignalen i vhdl-filen XOR_gate.vhd. Detta kan göras i modelsim genom följande steg

- Gå till Modelsims huvudfönster och fliken **Project**
- Höger klicka på XOR_gate.vhd → *Edit*
- Ändra så att signalen ”z <= not y;”

Filen kompileras genom att klicka på

- *Project → Compile→Compile Selected*

Vid kompilering kan det hända att det dyker upp röda felmeddelanden i transcript-fönstret. Dessa går att dubbelklicka på så ges mer information om vad som gått fel. Om koden kompilerat felfritt står det med grön text i transcript-fönstret att ”Compile of XOR_gate.vhd was successful”. Då är det dags att simulera kretsen

- do sim.do.

Det går att inkludera kompileringssteget i do-filen genom att lägga till raden vcom XOR_gate.vhd först. Exempel på kod:

```
vcom XOR_gate.vhd
restart -force
force -freeze sim:/xor_gate/a 1 25, 0 75 -r 100
force -freeze sim:/xor_gate/b 0 0, 1 200
run 400
```

8. Kombinera signaler i vågfönstret.

Antag att räknetillståndet på en räknare simulerats fram med 4 tillståndsvariabler. I detta fall vore det trevligare om det i vågfönstret istället för de fyra binära signalerna visade motsvarande hexadecimala siffra. Detta är möjligt. Låt oss göra stegen som krävs för att kombinera utsignalerna för XOR-grinden.

- Markera de signaler i vågfönstret som skall slås ihop till en vektor i detta fall signal y och z.
- *Höger klick på signal → Combine Signals …*
  Fyll i vektornamn, Result Name=”u”, **OK**

Nu ska den nya vektorsignalen dyka upp i vågfönstret och värdena indikeras med binära tvåbitsord. För att visa värdet på vektorn som positiva heltal gör:

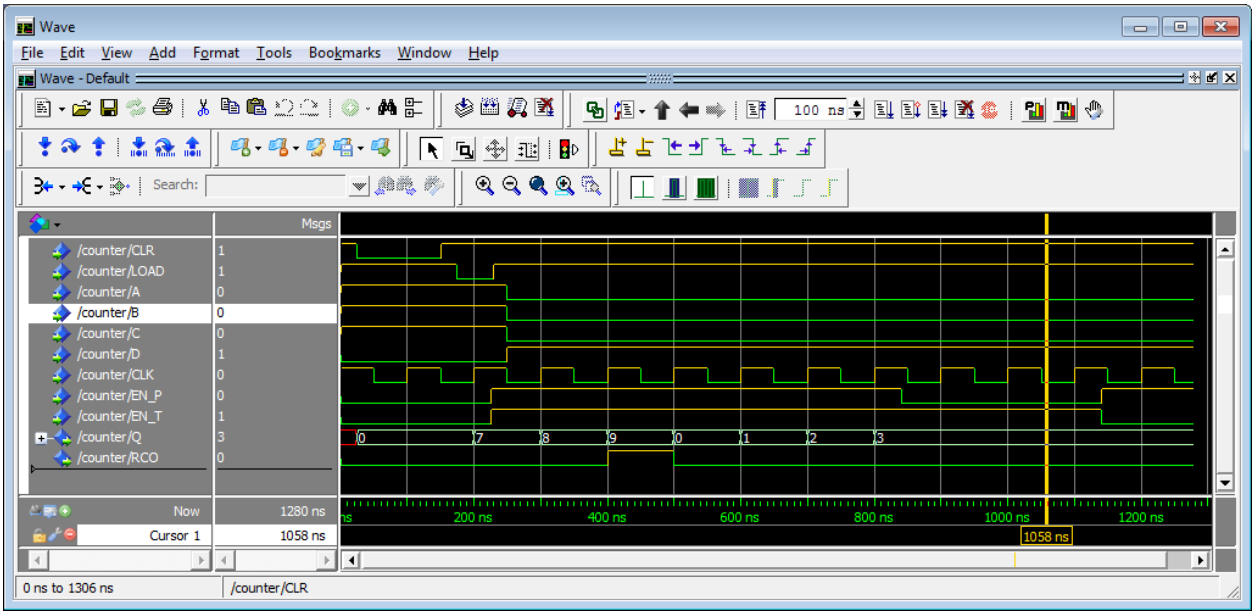
- Wave fönstret: *Högerklicka på signalnamnet → Radix →Unsigned*

9. Övriga funktionalitet som kan var bra att känna till

Byta färg på en signal
Wave fönstret: *Höger klick på signal → Properties → Wave Color*
Spara vågfönster från en simulering
Wave fönstret: *File→Export→Image..*

Uppgift 1. Konstruera med hjälp av VHDL en BCD-räknare som har samma funktion som 74LS160 och simulera konstruktionen i Modelsim. I den kopierade katalogen finns en underkatalog med namn Counter. I den katalogen finns Counter.vhd och sim.do som kan användas för att skapa räknaren och simuleringen. Använd funktioner som finns i VHDL, tex ”a <= a + 1” (minimering mha Karnaughdiagram behöver INTE göras). Simulera det beteende som finns uppritat i databladet för räknaren och som exemplifieras nedan.

Insignaler: CLR, LOAD, A, B, C, D, CLK, EN_P, EN_T
Utsignaler: Q3, Q2, Q1, Q0, RCO



Uppgift 2. Börja med att koda och simulera uppgifterna i lab 3. Skapa ett nytt projekt för varje uppgift.

Lösningsförslag på uppgift 1.

```
-------------------------------------
-- Counter.vhd
-------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-------------------------------------

entity counter is
port(
      CLK, CLR, LOAD, A, B, C, D, EN_P, EN_T: in std_logic;
      RCO : out std_logic;
      Q: out std_logic_vector(3 downto 0));
end entity;

-------------------------------------

architecture rtl of counter is

signal Q_int:unsigned(3 downto 0);

begin
process (CLR,CLK)
  begin
    if (CLR='0') then
      Q_int <= "0000";
    elsif rising_edge(CLK) then
      if LOAD = '0' then
        Q_int <= D & C & B & A;
      elsif EN_P = '1' and EN_T = '1' then
          if Q_int = 9 then
            Q_int <= "0000";
          else
            Q_int <= Q_int + 1;
          end if;
        end if;
    end if;
end process;
RCO <= '1' when EN_T = '1' and Q_int = 9 else '0';
Q <= std_logic_vector(Q_int);

end rtl;

# ----------------------------------------
# sim.do
# ----------------------------------------

 vsim Counter
 add wave sim:/Counter/CLR
 add wave sim:/Counter/LOAD
 add wave sim:/Counter/A
 add wave sim:/Counter/B
 add wave sim:/Counter/C
 add wave sim:/Counter/D
 add wave sim:/Counter/CLK
 add wave sim:/Counter/EN_P
 add wave sim:/Counter/EN_T
 add wave sim:/Counter/Q
```

```
add wave sim:/Counter/RCO

# restart -force

force -freeze sim:/Counter/CLR 1 0, 0 50, 1 150
force -freeze sim:/Counter/LOAD 1 0, 0 170, 1 220
force -freeze sim:/Counter/A 1 0, 0 250
force -freeze sim:/Counter/B 1 0, 0 250
force -freeze sim:/Counter/C 1 0, 0 250
force -freeze sim:/Counter/D 0 0, 1 250
force -freeze sim:/Counter/CLK 1 0, 0 50 -r 100
force -freeze sim:/Counter/EN_P 0 0, 1 210, 0 850, 1 1150
force -freeze sim:/Counter/EN_T 0 0, 1 210, 0 1150

run 1300
```