Einführung in

Computer Microsystems

Sommersemester 2010

Wolfgang Heenes

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Aufgabenblatt

28.04.2010

Aufgabe 1: Installation Xilinx ISE

Als erstes muss die Entwicklungsumgebung ISE installiert werden. Die Software ist für Windows und Linux verfügbar.[1] Als Programmpaket wird das **ISE WebPACK** verwendet, um es herunterzuladen muss man bei Xilinx registriert sein. Die URL zum Download lautet:

- `http://www.xilinx.com/ise/logic_design_prod/webpack.htm`

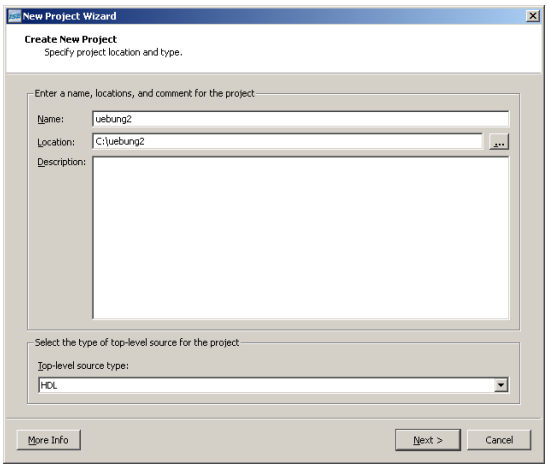
Fertig installiert nimmt das Tool etwa 4 GB Speicherplatz ein!

Weitere Informationen und Tutorials unter:

1. `http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/irn.pdf`
2. `http://www.xilinx.com/support/documentation/dt_ise11-1.htm`

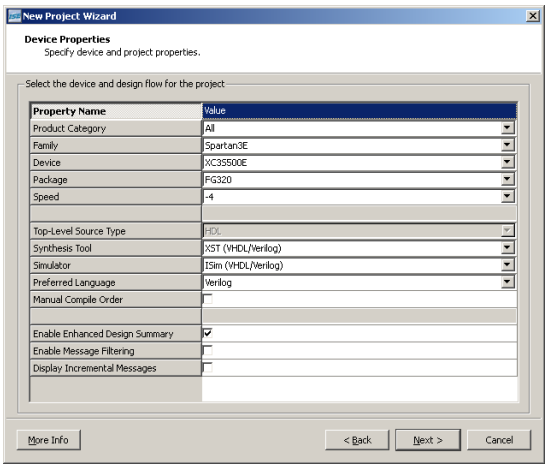
Aufgabe 2: Erstellung und Simulation eines Projektes

Legen Sie mit Xilinx ISE ein neues Projekt an. Nutzen Sie dazu den New Project Wizard (*File → New Project...*). Als Projektname geben Sie z. B. **uebung2** ein. Außerdem können Sie ein Arbeitsverzeichnis wählen.

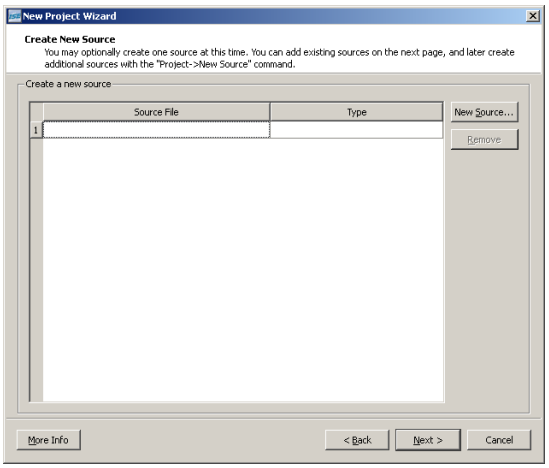


Geben Sie als Family **Spartan3E**, als Device **XC3S500E**, als Package **FG320** und für Speed **-4** an. Wichtig ist, dass Sie als Simulator **ISIM (VHDL/Verilog)** auswählen. Die folgende Abbildung zeigt das Auswahlfenster.

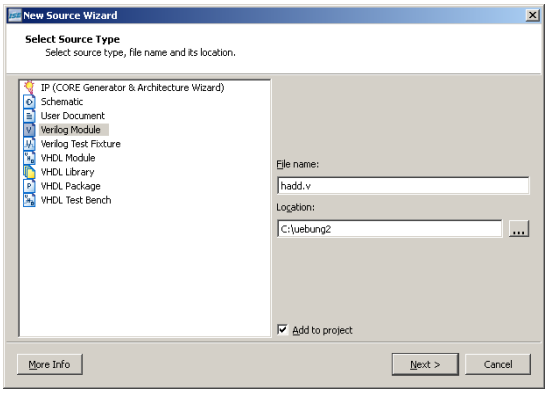
[1] Studierende mit anderen Betriebssystemen können die Übungen im Pool durchführen. Auf den Rechnern der RBG ist ebenfalls eine Installation vorhanden. Mit **ise** wird das Programm gestartet.



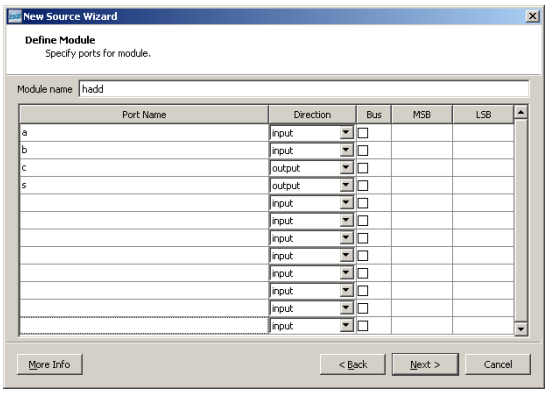
Das nächste Fenster erlaubt es neue Quellcodedateien zu erstellen. Klicken Sie auf *New Source...*.



Geben Sie als Dateinamen **hadd.v** an und wählen Sie als Typ **Verilog Module**.



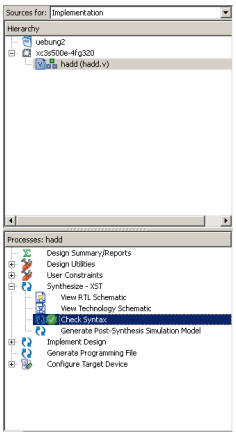
Der New Source Wizard erlaubt die Eingabe der Eingänge und Ausgänge. Erstellen sie zwei Eingänge (**a** und **b**) sowie zwei Ausgänge (**c** und **s**).



In den folgenden Fenstern klicken Sie *Next* bzw. *Finish*. Danach geht ein Fenster zur Eingabe des Quellcodes auf. Sie können nun folgendes Beispiel eines Halbaddierers vervollständigen.

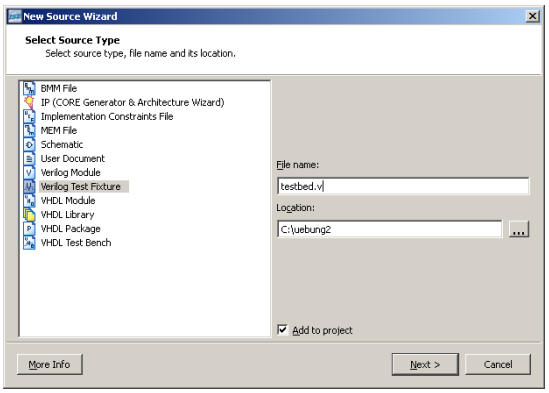
```
module hadd(
    input a,
    input b,
    output c,
    output s
    );
assign s = a ^ b; // ^ exklusiv-oder
assign c = a & b; // & und
endmodule
```

Nach der Eingabe soll die Syntax des Programms überprüft werden. Oben links bei *Sources for* muss *Implementation* ausgewählt und das Programm *hadd.v* markiert sein. Unter dem Punkt *Synthesize-XST* kann die Syntaxüberprüfung durch *Check Syntax* gestartet werden.



Erzeugen Sie nun eine *Testbench*[2]. Dazu starten sie den New Source Wizard (*Project* → *New Source*). Wählen Sie diesmal *Verilog Test Fixture* und als Dateinamen **testbed.v**.

[2] Simulationsfile, Stimuli



Die nächsten Fenster werden mit *Next* und *Finish* bestätigt. Es wird wieder ein Verilog HDL File vorgegeben, welches entsprechend folgendem Beispiel zu ergänzen ist. Natürlich können auch andere Zeiten angeben werden.

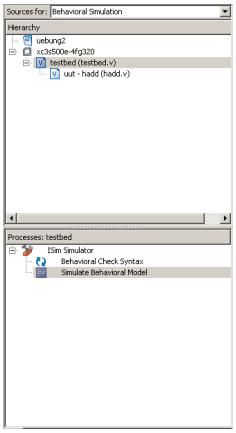
```
module testbed;
    // Inputs
    reg a;
    reg b;
    // Outputs
    wire c;
    wire s;

    // Instantiate the Unit Under Test (UUT)
    hadd uut (
        .a(a),
        .b(b),
        .c(c),
        .s(s)
    );

    initial begin
        // Initialize Inputs
        a = 0; b = 0;

        // Simulate
        #10 a = 1; b = 0;
        #20 a = 1; b = 1;
    end
endmodule
```

Für die Simulation muss im Fenster oben links bei *Sources for* jetzt *Behavioral Simulation* ausgewählt werden. Nach Markieren von *testbed* kann unter *ISim Simulator* mit *Simulate Behavioral Model* die Simulation gestartet werden.



Aufgabe 3: Simulation eines 4-Bit Zählers

Gegeben ist folgendes Verilog HDL Programm:

```
module zaehler(
    input clock,
    input up_down,
    output reg [3:0] qa
    );
initial qa = 4'b0;
integer direction;
always @(posedge clock) begin
    if (up_down) begin
        direction = 1;
    end else begin
        direction = -1;
    end
    qa <= qa + direction;
end // end always
endmodule
```

Simulieren Sie den Zähler und weisen Sie über die Simulation die korrekte Funktionsweise nach.

Aufgabe 4: Addierer/Subtrahierer

Gegeben sind folgende Verilog HDL Module:

```
module HalfAdder(A, B, Sum, Carry);
  input A, B;
  output Sum, Carry;

  assign Sum = A ^ B;
  assign Carry = A & B;
endmodule

module FullAdder(A, B, CarryIn, Sum, CarryOut);
  input A, B, CarryIn;
  output Sum, CarryOut;

  wire sum1, carry1, carry2;

  HalfAdder ha1(A, B, sum1, carry1);
  HalfAdder ha2(CarryIn, sum1, Sum, carry2);

  assign CarryOut = carry1 | carry2;
endmodule
```

a) Konstruieren Sie mit Hilfe der beiden obigen Module einen 4-Bit Ripple-Carry Addierer mit den Eingängen `A` und `B` und dem Ausgang `Sum`. Schreiben Sie eine Testbench, welche die Additionsfunktion testet und führen Sie eine Verhaltenssimulation durch.

b) Erweitern Sie den 4-Bit Addierer aus a) um eine Subtraktionsfunktion. Ein zusätzliches Eingangssignal `Sub` soll von Addition auf Subtraktion umschalten. Der „-"-Operator darf dazu *nicht* verwendet werden. Erweitern Sie Ihre Testbench aus a) um den zusätzlichen Test der Subtraktion. Testen Sie auch negative Differenzen.

Aufgabe 5: Paralleler Multiplizierer

Konstruieren Sie aus den Modulen von Aufgabe 4 einen voll parallelen 4-Bit Multiplizierer für *positive* Zahlen. Der „*"-Operator darf dazu *nicht* verwendet werden. Wieviele Bits werden für das Ergebnis benötigt? Testen Sie die Funktion des Multiplizierers durch Simulation mit einer Testbench.