Übung zur Vorlesung Technische Grundlagen der Informatik

Prof. Dr. Andreas Koch
Thorsten Wink

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 09/10

Übungsblatt 9 - Lösungsvorschlag

In dieser Übung werden Sie als Vorbereitung auf das Praktikum im Januar erstmals einen kompletten Hardware-Entwurf durchführen. Am Ende der Übungsstunde werden Sie ihre eigene Hardware auf dem FPGA-Board laufen lassen. Bitte bringen Sie falls vorhanden einen Laptop mit installiertem ISE in die Übung mit. Falls Sie keinen Laptop besitzen, bilden Sie bitte Kleingruppen, so dass jede Gruppe einen Laptop zur Verfügung hat.
Das Übungsblatt enthält mehr Aufgaben, als Sie vorausichtlich in der Übungsstunde bearbeiten könnnen. Je nach Erfahrung kann der erste Teil (Hardwareentwurf) sehr unterschiedlich lange dauern. Die restlichen Aufgaben sind als Hausaufgaben gedacht, eine genaue Trennung ist deshalb nicht vorgesehen.

Aufgabe 9.1 HDL-Code

Wir werden ein einfaches Beispiel verwenden: Einen Zähler, dessen Funktionsweise Ihnen bereits aus den vorherigen Übungen vertraut sein sollte. Da das Evaluationsboard 8 LEDs hat, werden wir einen 8 Bit-Zähler realisieren. Der Takt auf dem Board ist fest vorgegeben, er beträgt 50 MHz. Da dies viel zu schnell ist, um die LEDs beobachten zu können, werden wir einen Taktteiler verwenden, um einen Takt von 1 Hz zu erzeugen.
Beschreiben Sie den Zähler in Verilog. Das Modul muss folgende Ein- und Ausgänge verwenden:

```
module counter(
  input clk,
  input reset,
  output [7:0] led
  );
```

Desweiteren muss eine sog. ucf-Datei zum Projekt hinzufügen. Diese enthält Informationen, wo auf dem Evaluationsboard bestimmte Leitungen vorhanden sind. Die Datei kann auf der Webseite bei den Übungen heruntergeladen werden.

Zuerst implementieren Sie einen Taktteiler: Tipp: verwenden Sie dazu einen Zähler. Danach bauen Sie den 8-Bit-Zähler ein, er soll nur dann inkrementiert werden wenn der 1 Hz-Takt eine steigende Flanke hat.

```
module counter(
  input clk,
  input reset,
  output [7:0] led
  );

reg[25:0] clk_1hz;          //generiert einen 1Hz Takt
reg[7:0] counter;           //8Bit-Counter

assign led = counter;       //Zuweisung an die LED-Ausgänge


always@(posedge clk) begin
  if(reset)begin            //Initialisierung
    clk_1hz <=0;
    counter <=0;
  end
```

```
  else begin
    clk_1hz <= clk_1hz + 1; //Taktteiler
    if(clk_1hz ===26'b10111110101111000010000000)begin   //5.000.000
      counter <= counter + 1;                             //Counter erhöhen
    clk_1hz <=0;                                        //Clk-Counter wieder auf 0 setzen
    end
  end
end


endmodule
```

Aufgabe 9.2 Simulation

Simulieren Sie den Zähler.

Aufgabe 9.3 Synthese

Nachdem Sie die fehlerfreie Funktion Ihres Zählers durch Simulation festgestellt haben, synthetisieren Sie nun das Projekt. Stellen Sie dazu auf *Implementation* um und Klicken Sie auf *Generate Programming File*. Nun werden mehrere Schritte durchgeführt, um den Zähler auf das FPGA abzubilden. Es laufen dabei mehrere komplexe Algorithmen ab. Eine genaue Erklärung des Vorgangs wird in der Vorlesung CMS behandelt und ist hier nicht relevant. Am Ende des Vorgangs wird ein Bitstream erzeugt, welcher im nächsten Schritt auf das FPGA geladen werden kann.

Aufgabe 9.4 Test auf dem Board

Wenn Sie bei der Installation von ISE die Cable-Treiber mitinstalliert haben, können Sie das Evaluationsboard direkt per USB an Ihren Rechner anschließen. Mit *Configure Target Device* wird der Bitstream auf das FPGA geladen und Sie sollten aus den LEDs den Zähler beobachten können. Falls Sie keine Cable-Treiber installiert haben (oder diese nicht funktionieren), wenden Sie sich bitte an Ihren Tutor, um den Bitstream auf einem anderen Laptop herunterzuladen.

Aufgabe 9.5 zweiter Teil: ALU in Verilog

Beschreiben Sie die ALU aus Folie 5/29 in Verilog. Verwenden Sie 32 Bit Ein- und Ausgänge.

```
module alu32(
  input [31:0] A, B,
  input [2:0] F,
  output reg [31:0] Y
  );

  wire [31:0] S, Bout;
  assign Bout = F[2] ? ~B : B;     //B oder K omplement von B
  assign S = A + Bout + F[2];      //Summe
  always @ (*)
    case (F[1:0])
      2'b00: Y <= A & Bout;        //AND
      2'b01: Y <= A | Bout;        //OR
      2'b10: Y <= S;               //Summe
      2'b11: Y <= S[31];           //höchstes Summenbit (Vorzeichen)
    endcase
endmodule
```

Aufgabe 9.6 Zero-Ausgang

Erweitern Sie die ALU um einen Zero-Ausgang, der 1 ist wenn das Ergebnis einer Operation 0 ist.

Es muss ein zusätzlicher Output Z sowie eine zusätzliche Zeile eingefügt werden:

```
assign Z = (Y==32'b0);
```

Aufgabe 9.7 ALU-Simulation

Schreiben Sie eine Testbench, die die ALU aus der vorherigen Aufgabe instanziiert und die Operation Y = A - B berechnet. A ist 13, B ist 26.

```
module tb_alu32;

// Inputs
reg [31:0] A;
reg [31:0] B;
reg [2:0] F;

// Outputs
wire [31:0] Y;
wire Z;

// Instanziieren der ALU
alu32 uut (
  .A(A),
  .B(B),
  .F(F),
  .Y(Y),
  .Z(Z)
);

//Werte zuweisen, nur kombinatorische Logik
initial begin
  A = 13;
  B = 26;
  F = 3'b010;
end
endmodule
```

Hausaufgabe 9.1 Frohe Weihnachten

Das TGDI-Team wünscht Ihnen ein frohes Weihnachtsfest und alles Gute und viel Erfolg im Jahr 2010.

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter www.informatik.tu-darmstadt.de/plagiarism