Digitale Schaltungstechnik, WS 2022/2023

2. Übungsblatt — Abgabe: keine

1 Simulation

Simulationen sind ein wichtiges Werkzeug, um Schaltungen zu entwickeln und Fehler zu beheben, ohne sie z.B. auf einem FPGA ausprobieren zu müssen. Durch eine Simulation können Sie untersuchen, wie sich jedes Signal verhält! Bevor Sie eine Schaltung simulieren können, müssen Sie zunächst eine *Testbench* schreiben.

Eine Testbench wird, wie Ihr Design, in Verilog geschrieben. Im Gegensatz zu Ihrem Design muss Ihre Testbench aber nicht synthetisierbar sein, d.h. es gibt wahrscheinlich kein Hardware-Gegenstück dazu.

Simulieren Sie die in der vorigen Aufgabe implementierte Schaltung:

a) Eine Vorlage für eine Testbench finden Sie auf der Vorlesungshomepage. In der Vorlage wird ein Taktsignal und ein Resetpuls für ein Design erzeugt.

   Fügen Sie die Testbench zum Projekt hinzu, und setzen Sie unter `Source Properties` die `View Association` auf Simulation. Sie finden die Testbench jetzt unter `View: Simulation`.

b) Bearbeiten Sie die Testbench, um Ihr Design hinzuzufügen und benötigte Stimuli (z.B. einen Tastendruck) zu generieren.

c) Sie starten die Simulationsumgebung dann durch Auswahl der Testbench und Doppelklick auf `Simulate Behavioral Model`. Stellen Sie sicher, dass in den Projektoptionen `ISim` als Simulator ausgewählt ist.

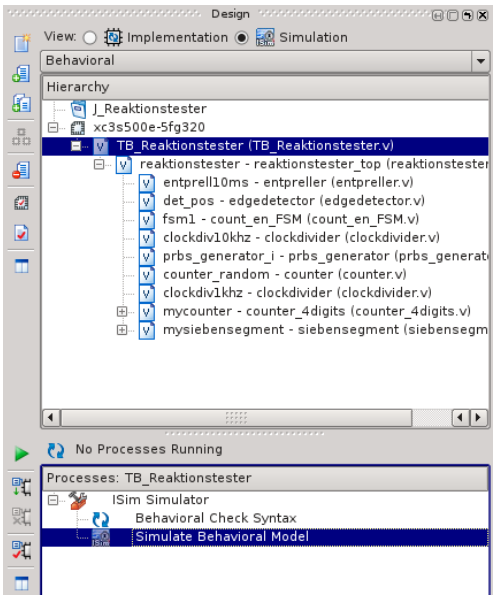

Abbildung 1: Hierarchische Ansicht des Projekts im linken Bereich des ISE-Programmfensters

Eine Einführung in die Oberfläche des Simulators finden Sie unter → `http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug682.pdf`.

Denken Sie an definierte Startwerte der Register oder eine Reset-Schaltung, um undefinierte Signale im Simulator zu verhindern. Eventuell müssen Sie auch Zeitparameter Ihres Designs in der Simulation anpassen, da sie sonst unnötig lange dauert.

2 Takteiler - Verilog

Als erste Übung im Umgang mit Verilog implementieren Sie den Taktteiler, den Sie bereits vom vergangenen Aufgabenblatt kennen.

a) Erstellen Sie ein Modul mit einem Takteingang und einem 32-Bit-Zählerausgang (Schreibweise `[31:0]` in der Port-Deklaration).

Legen Sie eine 32-Bit-Zählervariable vom Typ `reg` an, die in jedem Takt um Eins erhöht wird (Zuweisung innerhalb eines `always`-Blocks mit dem *non blocking* Zuweisungsoperator „<=“).

Um das Modul für die Wiederverwendung geeigneter zu machen, können Sie die Breite der Zählervariable und des Ausgangsports auch mit einem *Parameter* beschreiben, anstatt sie fest vorzugeben:

```
module zaehler
#( parameter breite = 32 ) // mit Angabe des Default-Werts
(
  input clk,
  output ... // hier sollte breite vorkommen
);

...

endmodule
```

Beim Erzeugen einer Instanz kann die Breite dann mit angegeben werden:

```
zaehler #( .breite(5) ) zaehler_instanz
(
  ...
);
```

b) Erzeugen Sie nun ein Top-Level-Modul, in dem Sie eine Instanz Ihres Zählers anlegen und eines seiner Ausgangsbits an eine LED anschließen. Um die Blinkfrequenz auszuwählen, soll es wieder einen 16:1-Multiplexer geben, der durch vier Schiebeschalter gesteuert wird. Einen Multiplexer können Sie durch Verschachtelung des Bedingungsoperators „`(Sel)?A:B`“ beschreiben.