Technische Informatik
Dr. Henrik Brosenne
M. Sc. Maria Kosche

Wintersemester 2023/24
Georg-August-Universität Göttingen
Institut für Informatik

Übungsblatt 2

Verilog

Hinweise zur Abgabe der Lösungen:

Die Abgabe von Lösungen zu allen Übungsblättern ist grundsätzlich freiwillig.

Die Bearbeitungen der Aufgaben können Sie freiwillig und in geeigneter Form in der Stud.IP-Veranstaltung über das **Vips-Modul** zum entsprechenden Aufgabenblatt hochladen. Sie erhalten dann entsprechend ein kurzes Feedback zu Ihrer Abgabe. Sie können Ihre Bearbeitungen gerne mit LaTeX formatieren, es ist aber auch der direkte Upload von Text oder der Upload von Text- und Bilddateien in gängigen Formaten möglich.

Die Aufgaben können auch in Kleingruppen bearbeitet und abgegeben werden. Wenn Sie nicht alleine abgeben, achten Sie bitte darauf, dass Sie sich selbstständig **vor** der Abgabe im Vips-Modul in einer sogenannten Übungsgruppe zusammenschließen. Hierzu können Sie sich in einer der dort vorhandenen Übungsgruppen gemeinsam eintragen. Nur so erhalten alle Zugang zu Feedback und Kommentaren zu der entsprechenden Abgabe.

Am Ende von Übungsblatt 1 finden Sie Hinweise zu den **Pseudocode-Konventionen** für die Übungen.

Aufgabe 1 – Fibonacci

In der Vorlesung haben Sie Verilog kennengelernt. Zur Simulation empfehlen wir Icarus Verilog, zu finden unter `http://iverilog.icarus.com/`. Hier finden Sie den Sourcecode und Binaries zum Herunterladen. Achten Sie darauf, die Binaries gegebenenfalls der PATH-Variablen Ihres System hinzuzufügen. Anschließend können Sie den Compiler und Virtual Processor wie in der Vorlesung vorgestellt verwenden.

Schauen Sie sich den Code in `add_testbed.v` aus der Vorlesung an. Sie finden die Datei auch auf Stud.IP.

Schreiben Sie ein Verilog-Programm, welches analog zum vorgestellten Programm aus der Vorlesung die n-te Fibonacci-Zahl berechnet.

Zur Erinnerung: Die Fibonacci-Folge ist eine unendliche Folge natürlicher Zahlen, die mit folgender Funktion $\mathrm{fib}\colon \mathbb{N} \to \mathbb{N}$ beschrieben werden kann.

$$\mathrm{fib}(0) = 1$$
$$\mathrm{fib}(1) = 1$$
$$\mathrm{fib}(n) = \mathrm{fib}(n-1) + \mathrm{fib}(n-2) \text{ für } n \geq 2$$

Aufgabe 2 – Modulo-k Zähler

In dieser Aufgabe sollen Sie einen Modulo-k Zähler entwickeln. Gegeben ist der folgende Code, der einen n-bit-Counter beschreibt. Sie finden den Code auch auf Stud.IP.

```
1  module counter (Clock, Reset_n, Q);
2
3    parameter n = 4;
4
5    input Clock, Reset_n;
6    output [n-1:0] Q;
7    reg [n-1:0] Q;
8
9    always @(posedge Clock or negedge Reset_n)
10   begin
11     if (!Reset_n)
12       Q <= 1'd0;
13     else
14       Q <= Q + 1'b1;
15   end
16 endmodule
```

parameter
In Zeile 3 sehen Sie die Verwendung eines Parameters, welcher bei der Instantiierung in einem anderen Modul mit `defparam` gesetzt werden kann. So kann beispielsweise ein 8-bit-Counter wie folgt erzeugt werden:

```
1 counter eight_bit(Clock, Reset_n, Q);
2 defparam eight_bit.n = 8;
```

Der Vorteil eines Parameters ist die Möglichkeit, mehrere verschiedene Counter innerhalb eines Moduls instantiieren zu können.

posedge und negedge
Das Event `posedge` in Zeile 9 bezeichnet eine steigende Flanke des übergebenen Signals und `negedge` entsprechend eine fallende Flanke des Signals.

Ein Modulo-k Zähler zählt von 0 bis $k-1$. Sobald $k-1$ erreicht ist, wird der Zähler auf 0 zurückgesetzt.

Schreiben Sie ein Verilog-Programm, welches einen Modulo-k Zähler für $k = 20$ beschreibt. Verwenden Sie dabei einen n-bit-Counter mit angemessenem n. Ergänzen Sie Ihr Programm um einen Output `threshold`, der genau in den Ticks auf 1 gesetzt wird, in denen der Zählerwert gleich $k-1$ ist.

Fügen Sie entsprechende Systemkonstrukte hinzu und simulieren Sie Ihr Programm mithilfe von gtkwave.