Rechnerarchitektur 1
DI Alwin Zulehner

12. Mai 2016
Johannes Kepler Universität

Übung 6

Abgabe bis **Donnerstag, 19. Mai 10:00** via EPIIC: `http://ep.iic.jku.at`.

SystemVerilog Dateien müssen vor der Abgabe in EPIIC ausgeführt werden. Dateien, die nicht kompilieren können nicht abgegeben werden.

1. Multiplizierer (10 Punkte)

Implementiere einen 16-Bit Multiplizierer als SystemVerilog Modul (strukturelle Beschreibung). Die 16 Partialprodukte sollen mit Hilfe von Carry-Save Addierern baumartig zusammengefasst werden. Erstelle zusätzlich ein Modul *testbench*, das den Multiplizierer für mindestens 5 nicht triviale Fälle testet.

Hinweis: Bitweise Operatoren (∼, &, —, ˆ) können auf Signale mit beliebiger Breite angewendet werden. Der Replikationsoperator `{n{a}}` repliziert das Signal `a` n-mal.

2. Serieller Dividierer (14 Punkte)

Ziel dieser Aufgabe ist die Implementierung eines seriellen Dividierer mit 8-Bit Dividend, 4-Bit Divisor und 4-Bit Quotient in SystemVerilog. Dazu sind folgende Schritte nötig:

(a) Implementiere ein Register mit als parametrisiertes Modul mit der Signatur `module register(d,q,en,clk)`. Das Register wird nur geschrieben wenn `en=1`. Der Parameter des Moduls gibt die Bitbreite des Registers an. Eine verhaltensbasierte Beschreibung ist ausreichend.

(b) Implementiere ein ladbares Schieberegister als parametrisiertes Modul mit der Signatur `module shiftregister(d,q,reset,clk)`. Bei `reset=1` soll geladen und bei `reset=0` geschoben werden. Der Parameter gibt die Bitbreite des Registers an. Eine verhaltensbasierte Beschreibung ist ausreichend.

(c) Implementiere den seriellen Dividierer als SystemVerilog Modul mit der Signatur `module divider(dividend, divisor, quotient, reset, clk)`. Wenn der 1-Bit Input `reset` auf 1 gesetzt wird sollen Dividend und Divisor in die entsprechenden Register geladen und der Quotient auf 0 gesetzt werden. Wenn `reset` anschließend auf 0 gesetzt wird startet die Berechnung. Das Modul soll mit Ausnahme des enthaltenen Subtrahierer und der Register aus (a) und (b) rein strukturell beschrieben werden.

(d) Impementiere ein Modul *testbench* das den seriellen Dividierer für mindestens 5 nicht triviale Fälle testet.

Alle Register sollen Daten nur bei steigender Taktflanke übernehmen. Es ist ausreichend wenn nur eine Datei mit der kompletten Implementierung abgegeben wird.