FH-OÖ/Hagenberg/HSD
Chip-Design 3 — CHD3
Markus Pfaff © 2025
Übung 9
Strobe-Signale, Strobegenerator, Lauflicht II, asynchrone Eingänge

Unser Lauflicht aus einer der vorigen Übungen ist noch nicht ganz perfekt, denn wir können die Geschwindigkeit derzeit noch nicht bestimmen. Bei einer Taktung mit einer Taktfrequenz von 50 MHz hat das Lauflicht keine erkennbare Funktion mehr, so dass man sich einen niedrigeren Takt wünschen würde. Gibt es diesen nicht, so erzeugt man sich entsprechende *Strobe*-Signale:

***Strobe*-Signale** sind der in der Praxis übliche, einfachste und unproblematischste Weg zur Verlangsamung der Abarbeitung. *Strobe*-Signale sind periodische Signale, die innerhalb eines Zyklus' nur für die Dauer einer einzigen Taktperiode des Systemtaktes aktiv und sonst immer inaktiv sind. Man erzeugt sie beispielsweise mittels eines Zählers, der bei einem bestimmten Zählerwert eine '1' auf dem Ausgang `oStrobe` führt und sonst stets den Wert '0'.

1. Aufgabe *Parameterisierbarer Strobe-Generator*

Ihr Zähler aus der vorletzten Übung soll nun zu dem Strobegenerator `StrobeGen(Rtl)` umfunktioniert werden. Dessen *Entity*:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.Global.all;

entity StrobeGen is

  generic (
    gNrClkCycles : natural := 50E6);

  port (
    -- Sequential logic inside this unit
    iClk         : in  std_ulogic;
    inResetAsync : in  std_ulogic;

    -- Strobe with the above given cycle time
    oStrobe      : out std_ulogic);

end StrobeGen;
```

Der Strobe-Generator hat nur einen Ausgang, der den Namen `oStrobe` trägt. Dieser ist so gut wie immer auf den Wert `'0'` gesetzt. Lediglich nach Ablauf von `gNrClkCycles` Taktzyklen ist dieser Ausgang für die Dauer einer einzigen Taktperiode auf `'1'` zu setzen.
Beachten Sie bitte, dass die *function* `LogDualis` benötigt wird, um die für den Zähler notwendige Bitbreite zu berechnen (*package* `Global`). Wie arbeitet diese *function*?

Diese *function* muss zwar synthesefähig sein (warum?), es ist aber nicht notwendig, dass sie *effizient* in Hardware umgesetzt wird (warum?). Bedeutet es einen Unterschied, ob Sie die *function* `LogDualis` zur Verarbeitung einer *Constant* oder eines *Signals* verwenden?

Erstellen Sie die *architecture* des Strobegenerators und weisen Sie mittels einer*Testbench* per Simulation die Funktionsfähigkeit Ihrer *unit* `StrobeGen` nach.
Synthetisieren Sie Ihr Modell. Wie viele Flipflops erwarten Sie im Ergebnis?

Hat es einen Sinn, diese Beschreibung auf dem Board zu testen, wobei der Ausgang `oStrobe` mittels Leuchtdiode visualisiert wird? Wenn ja, tun Sie dies bitte. Wenn nein, warum nicht?

2. Aufgabe *Das wahre Lauflicht*

Nun sollen Lauflicht und `StrobeGen` miteinander kombiniert werden. Genauer gesagt, der Strobe-Generator soll mittels `oStrobe` die Zeitpunkte für die Zustandsübergänge des Lauflichts steuern.

Hierzu erweitern Sie Ihre Beschreibung des Lauflichts um einen Eingang `iEnable`. Falls dieser Eingang `cActivated` ist, so soll bei der nächsten Taktflanke (50 MHz!) ein Zustandswechsel erfolgen. Ansonsten bleibt der derzeitige Zustand erhalten (mit anderen Worten: er wird nicht verändert).
Sie können nun mittels des `StrobeGen` das Lauflicht in entsprechenden Zeitabständen von Zustand zu Zustand springen lassen, indem Sie den Ausgang `oStrobe` an den Eingang `iEnable` anschliessen. Das Lauflicht wird mit jedem Strobe den Zustand genau einmal wechseln. Warum genau einmal?

Beschreiben Sie dieses System in VHDL und weisen Sie die korrekte Funktion mittels Simulation nach. Beachten Sie, dass die Periodendauer von `oStrobe` mittels `gNrClkCycles` auf einen simulationsfreundlichen Wert eingestellt werden kann.

Nun zum Höhepunkt dieser Übung: Realisieren Sie das automatische Lauflicht auf dem DE1-Board. Vergessen Sie nicht, `gNrClkCycles` einen vernünftigen Wert zuzuweisen, sodass die Zustandswechsel gut zu beobachten sind. Wie viele FlipFlops erwarten Sie sich? Entspricht das Syntheseergebnis Ihren Erwartungen?

3. Aufgabe *Asynchrone Eingänge*

Das *Strobe*-Signal aus der vorigen Aufgabe war ein synchrones Signal (bezogen auf das empfangende System, also das Lauflicht). Es stammt aus einem Systemteil, der mit dem gleichen Takt wie die Lauflicht-FSM betrieben wird.
Soll ein synchrones Design dagegen auf Informationen aus asynchronen Signalen, wie etwa von Tasteneingängen reagieren, so sind besondere Vorkehrungen zu treffen.
Welche Vorkehrungen sind damit gemeint? Wie sieht die zugehörige VHDL-Beschreibung aus, mit der diese Vorkehrung getroffen wird. Erstellen Sie ein entsprechendes VHDL-Design! Synthetisieren Sie Ihr Modell. Wie viele FlipFlops erhalten Sie im Syntheseergebnis?

Eine Ausnahme stellt der Einzelschritt-Taktflanken-Taster dar. Diskutieren Sie, warum dies so ist. Stellt das erzeugte Signal ein asynchrones Signal dar?
Erweitern Sie das Lauflicht so, dass es durch Drücken einer Taste nur noch (zyklisch) durch die ersten vier States läuft. Lässt man die Taste los, verhält es sich wieder normal.
Alternativ: Es wechselt zwischen den States „Alle an“ und „Alle aus“. Weitere Alternative: Es läuft mit der doppelten Geschwindigkeit. Oder: Es läuft mit der halben Geschwindigkeit.