Hochschule für Angewandte Wissenschaften Hamburg
University of Applied Sciences Hamburg
Fakultät Technik und Informatik
Department Informatik

HAW HAMBURG

Aufgabe A0 Email

Jedes Team sendet eine Email an
CE.S12S@Hamburg-UAS.eu
und als CC an jedes einzelne Teammitglied.[1]
Dies ist gemäß den in der Vorlesung durchgesprochenen Richtlinien zu tun.

Also mit einem Subject, das Ihren Teamnamen (der Ihnen während des ersten Labortermins mitgeteilt wird) gefolgt von „_CEP_TI4_A0_V#“ (nur Großbuchstaben) enthält.
"#" ist durch eine Dezimalziffer zu ersetzen, die die Versionsnummer der Abgabe identifiziert. ← Eigentlich hätte ich gehofft, dass dieser Satz nicht nötig wäre ☺

In der Email schreiben Sie die vollständigen Namen aller Teammitglieder, deren Matrikelnummer und deren (Haupt-)Email-Adresse.

Bsp:

Ihr Team sei
S2T3

Teammitglieder seien
Beate Beispielfrau, Matr.Nr.:1234567, Beate.Beispielfrau@haw-hamburg.de
Max Mustermann, Matr.Nr.:1234568, Max.Mustermann@haw-hamburg.de

Die Email wird das erste Mal geschickt.

Die Email hat also folgenden Aufbau

Subject: S2T3_CEP_TI4_A0_V1

Mail: Name: Beispielfrau, Beate,
Matr.Nr.: 1234567,
email: Beate.Beisplielfrau@haw-hamburg.de

Name: Mustermann, Max,
Matr.Nr.: 1234568,
email: Max.Mustermann@haw-hamburg.de

Bemerkungen:

- Subject ist das englische Wort für Betreff. Es ist also die Betreffzeile gemeint!
- Ihr Subject muss nicht farbig sein.
- Ein Team besteht aus 2 Mitgliedern.

[1] Sie können im CC-Feld auch mehr als eine Email-Adresse für jedes einzelne Teammitglied angeben.
Beispielsweise: beispl_b@informatik.haw-hamburg.de, beate.beispielfrau@yahoo.de, muster_m@informatik.haw-hamburg.de, mäxchen@maxi.org
Wenn ich Sie erreichen will, mache ich ein Reply-To-All auf die A0-Email. Sie haben es in der Hand, wo diese Email dann hingeht.

Vorbemerkung:
Zur Lösung der ersten 3 Aufgaben müssen sie Schaltungen in ein Spartan FPGA XC3S1200E implementieren. Dazu sind Kenntnisse der SW(-Packete) ModelSim und ISE nötig, die Sie in DT erlernt haben.
Es sei noch einmal auf die Dokumente

- Anleitung Xilinx / ModelSim
- Digilent Nexys2 Board Reference Manual (www.digilentinc.com/Data/Products/NEXYS2/Nexys2_rm.pdf)
- XST (Xilinx® Synthesis Technology) User Guide
- Spartan-3E FPGA Family Data Sheet (www.xilinx.com/support/documentation/data_sheets/ds312.pdf)

verwiesen.
Diese können auch unter Moodle gefunden werden (www.elearning.ls.haw-hamburg.de).

Aufgabe A1 Schieberegister als Vorbereitung für A2

In dieser Aufgabe sollen Sie als Vorbereitung für A2 ein spezielles 24 Bit Shiftregister-"Objekt" implementieren.

Idee im Hintergund:
Im 24 Bit Shiftregister-"Objekt" (im Folgenden als **SR** abgekürzt) soll ein zusammenhängendes 8 Bit-Feld (**Cabin**) auf und ab wandern. Das SR modelliert (später in A2) einen Fahrstuhlschacht (**Elevator Shaft**) und das 8 Bit-Feld beschreibt die Position des Fahrstuhls bzw. der Fahrstuhlkabine (**Cabin**) in diesem Fahrstuhlschacht. Die Cabin bewegt sich im Fahrstuhlschacht und kann also eine von 17 Positionen einnehmen.

Die Position der Cabin im Fahrstuhlschacht ist auf 2 Arten zu visualisieren.

- Die Position der Cabin (Zahlenwert - LSB-Position der Cabin) im Fahrstuhlschacht soll auf der 7-Segment-Anzeige in Hex dargestellt werden. Dazu kann der im SR enthaltene 24 Bit-Wert an die vorgegeben Komponente „transcoder“ gesandt werden, deren Aufgabe es ist, die Position der Cabin anzuzeigen.
- Die Position der Cabin soll mit dem "Fahrstuhl-Modell"-Board angezeigt werden.

Details:

„Intelligentes“ 24 Bit Shiftregister-"Objekt": **SR**
Das SR modelliert den Fahrstuhlschacht. Das SR unterteilt sich in eine Ansteuerungslogik und das konkrete "nur" 24 Bit Shiftregister SSR (simple shift register).
Das im SSR „wandernde“ 8 Bit-Feld beschreibt später die Cabin bzw. deren Position im Fahrstuhlschacht.
Das MSB des SSR ist das obere Ende des Fahrstuhlschachts und das LSB das untere Ende.
Beim Reset ist das SSR vollständig zu löschen
Bei einem up-Signal wird der SSR-Inhalt (und damit auch die Cabin) nach oben/aufwärts und bei einem down-Signal nach unten/abwärts geschoben (bewegt).

SR muss u.a. die folgenden Signale unterstützen:

Ausgänge:

ess	**E**levator **S**haft **S**tate – der 24 Bit-Wert, der im SSR gespeichert ist.
Ausgänge zum "Fahrstuhl-Modell"-Board hin	
cab_dc	**Cab**in **D**own **C**ommand – die Cabin soll einen Schritt abwärts fahren (1 Bit down shift)
cab_uc	**Cab**in **U**p **C**ommand – die Cabin soll einen Schritt aufwärts fahren (1 Bit up shift)
csr_msb_in	Bit das MSB-seitig eingeschoben wird.
csr_lsb_in	Bit das LSB-seitig eingeschoben wird.
em_clk	„Takt“ für das "Fahrstuhl-Modell"-Board - hierbei handelt es sich um einen „gated clk“ - dieser darf nicht im FPGA mit einem clk-Eingang verbunden werden.
em_nreset	Reset für das "Fahrstuhl-Modell"-Board.

Eingänge:

up	Die Cabin soll einen Schritt aufwärts fahren (1 Bit up shift für das SSR)
down	Die Cabin soll einen Schritt abwärts fahren (1 Bit down shift für das SSR)
clk	50Mhz Takt
reset	Reset

Ansteuerungslogik für das SSR:

Die von Ihnen zu entwerfende Ansteuerungslogik wertet die an das SR übergebenden up- und down-Kommandos aus und steuert das SSR an.

Beim Reset ist das SSR vollständig zu löschen (s.o.). Für den Betrieb bedarf es jedoch des zusammenhängenden Bitfelds "Cabin". Nach dem Reset soll in einer Initialisierungsphase die Cabin durch "Einschieben" im SSR aufgebaut werden soll. (Abweichend von der Realität ist der Fahrstuhlschacht also zunächst leer und der Fahrstuhl muss in der Initialisierungsphase erst noch „erscheinen“. Nach Abschluss der Initialisierungsphase soll sich die Cabin ganz unten bzw. auf der Position 0 (gemessen am LSB der Cabin) befinden. Jetzt befindet sich das Modell in einem „gültigen“ Zustand. Von nun an darf die Cabin das SSR nicht mehr verlassen – es sind also immer exakt 8 zusammenhängende Bits 1 und der Rest 0.

Wie bereits eingefordert soll sich bei einem **up**-Signal die Cabin aufwärts und bei einem **down**-Signal abwärts bewegen. Die Cabin darf jedoch niemals das SSR "verlassen". D.h. wenn die Cabin sich ganz oben (im SSR) befindet sollen up-Kommandos ignoriert werden und analog sollen down-Kommandos ignoriert werden, wenn sich die Cabin ganz unten (im SSR) befindet.

Bemerkungen zum "Fahrstuhl-Modell"-Board:

Beachten Sie dass die Kommunikation mit dem "Fahrstuhl-Modell"-Board **nicht** mit 50MHz erfolgen kann. Später (A2) soll die Schrittgeschwindigkeit der Cabin zwischen 0.5 und 1 Sekunde betragen.

Die up- und down-Kommandos an das SR.

Diese können entweder über die Taster BTN3 und BTN2 oder über die Taster "lev_dr(1)" / ButtonDown1 und "lev_ur(1)" / ButtonUp1 des "Fahrstuhl-Modell"-Board gegeben werden

- Sofern die up-Kommandos über den Taster / Button 3 (BTN3) und die down-Kommandos über den Taster / Button 2 (BTN2) gegeben werden, ist zu beachten, dass diese Taster nicht entprellt sind. Der vorgegebene Code gLowPass (Generic Low Pass) stellt sicher, dass nur High-Pulse einer vorgegebenen Länge durchgelassen werden. (Der Code ist unschön, aber er löst diese Aufgabe und ist generisch – für die Simulation können Sie kleine Werte und später für die HW größere Werte einstellen). Sie können den Code in geeigneter Weise nutzen um die up- und down-Signale zu entprellen. Die entprellten Signale leiten Sie weiter an die Ansteuerungslogik für das Shiftregister sr.
- Sofern Sie die up-/down-Kommandos über die Taster "lev_dr(1)" / ButtonDown1 und "lev_ur(1)" / ButtonUp1 des "Fahrstuhl-Modell"-Board geben werden, ist zu beachten, dass diese in A2 einen veränderte Bedeutung haben. Ferner ist zu beachten, dass das "Fahrstuhl-Modell"-Board nicht mit 50Mhz arbeiten kann.

Der gLowPass ist vorgegeben

```
entity gLowPass is
    generic (
        nob : integer := 16
    );--}generic
    port (
        so   : out std_logic;
        si   : in  std_logic;
        clk  : in  std_logic;
        sres : in  std_logic;
    );--}port
end entity gLowPass;
```

Ein Wert n für nob bedeutet, das nur High Pulse akzeptiert werden, die min. eine Länge von (etwa) 2^n 50Mhz-Takten aufweisen.

"si" ist das zu entprellende Signale und "so" ist das entprellte Signal.

Der Transcoder ist vorgegeben

```
entity transcoder is
    port (
        seg7c_cp : out std_logic_vector(  7 downto 0 );  --
        seg7a_cp : out std_logic_vector(  3 downto 0 );  --
        ess      : in  std_logic_vector( 23 downto 0 );  --
        seg7_shf : in  std_logic;                        --
        clk      : in  std_logic;                        --
        sres     : in  std_logic                         --
    );--}port
end entity transcoder;
```

Signale für der Transcoder:

Ausgänge:

seg7c_cp	8 Bit zur Ansteuerung der LED Kathoden 7-Segment-Anzeige.
seg7a_cp	4 Bit zur Ansteuerung der LED Anoden der 7-Segment-Anzeige.

Eingänge

ess	(**E**levator **S**haft **S**tate), die 24 Bit des von Ihnen zu implementierenden SSR
seg7_shf	Pulse, dessen High-Pegel genau einen Takt lang ist und der die Auflagen für den Refresh-Zyklus einer Ziffer der 7-Segment-Anzeige erfüllt. Vorschlag: Zyklisch exakt alle 5,24288 ms bzw. $2^{18}*(^{1}/_{50MHz})$ kommt.
clk50m	50Mhz Takt
sreset	synchroner Reset, High active

Randbegingungen:

Die Toplevel-Entity der zu synthetisierende Schaltung(en) soll als "**`dut`**" (**D**esign **U**nder **T**est) im VHDL-Design-Toplevel instanziiert werden.

Testmuster sollen in einer Entity "**`sg`**" (**S**timuli **G**enerator) erzeugt werden. Sofern Sie mögen können Sie auch einen Response Checker (Entity "**`rc`**") einsetzen, der Sie bei der Testauswertung unterstützt. Das **`dut`** und die Entities **`sg`** und optional **`rc`** sollen in einer Entity **`toplevel`** instanziiert werden. Die Entities **`toplevel`**, **`sg`** und otional **`rc`** sind behavioral. Nur das **`dut`** muss HW repräsentieren und muss im synthetisierbaren RTL-Style modelliert werden.

Für A1 ist SR das DUT. Die Entity "**`sr`**" wird also als **`dut`** im Toplevel "**`toplevel`**" instanziiert. Innerhalb von SR sollen die "Blöcke" SSR und zugehörige Ansteuerungslogik klar erkennbar sein. Z.B. entweder als eigenständige Entities/Komponenten oder als eigenständige Prozesse innerhalb SR.

Eine Minimierung der Resource-Kosten hat Priorität.

Tipps:

- Teilen Sie **nicht** immer wieder aufs Neue den 50MHz Takt herunter, sondern generieren Sie einmalig geeignete Ticks.
- Trennen Sie im Block SR das "strunzdoofe" 24 Bit Shiftregister von seiner intelligenten Ansteuerlogik

Vorbereitungsunterlagen:

- ASM-Chart für Ansteuerung des SSR.
- VHDL-Code (Entwurf) aller von Ihnen zu entwickelnden Komponenten (incl. Testbench)

Im Rahmen der Aufgabe sind durchzuführen/gewährleisten:

- Testmustererzeugung in VHDL-Stimuligenerator
- VHDL RTL-Simulation
- VHDL Timing-Simulation
- VHDL Code-Qualität
- Korrekte Funktion der HW / des FPGA
- Auswertung der Reports
  - HW-Kosten (Wie viele FFs sind in welchem Block?)
  - Wo ist der kritische Pfad?

Freiwillige optionale Zusatzaufgaben:

Nutzen Sie nicht den vorgegebenen Code, sondern entwickeln Sie selber einen eigenen Transcoder.