

Tutorium zu Computer-Engineering im SS19

Termin 2

Jakob Otto

HAW Hamburg

2. April 2019

- Praktikum

- ▶ wdh. UCF-Files
- ▶ tick-generator
- ▶ shiftregister
- ▶ lookup table



Aufgabenzettel

UCF-Datei

kleine Wiederholung

- UCF-Datei mappt ein- und Ausgänge auf pins
- Format:

```
NET <port-name> LOC = <pin> | IOSTANDARD=LVCMOS33;
```

- Als Beispiel:

```
NET "clk" LOC = "V10" | IOSTANDARD=LVCMOS33;
```

Woher die ganzen pins?!

- Normalerweise aus irgendwelchen Datenblättern..
- Hier: Aus der **CE_Board-Doku**
- Weitere Doku zum Nexys2-board gibts **HIER**

Ideen zur Praktikumsaufgabe

Tick-Generator

- erzeugt ticks mit einer gewünschten Frequenz auf Basis eines gegebenen Taktes
- kann Frequenz nur absenken!
- im Grunde zählt der Tickgenerator nur Takte
- Ticks werden wie ein druck eines buttons gewertet.
 - ▶ Enable-Signal

Tickgenerator-idee I

```
entity TickGen is
  port(
    tick : out std_logic;
    clk : in std_logic
  );
end entity TickGen;
```

Tickgenerator-idee II

```
tickGen: process (clk) is
    constant maxValue : integer := XXXXX;
    variable count : integer range 0 to maxValue := 0; -- nicht gut
    variable tick_v : std_logic;
begin
    if (rising_edge(clk)) then -- geht wohl noch
        count := count + 1;
        if (count = maxValue) then
            count := 0;
            tick_v := '1';
        else
            tick_v := '0';
        end if;
    end if;
    tick <= tick_v;
end process tickGen;
```

Auswahl der Anoden

- Die Anoden müssen zyklisch mit durchgewechselt werden
- Dazu nutzt ein zyklisches shiftregister!
- shiftregister soll '1' mit frequenz des generierten ticks shiften.
- Dadurch werden nacheinander die einzelnen Anoden durchgeschaltet.

```
sequlo:  
process (clk) is  
begin  
    -- kennt ihr..  
end process sequlo;
```

shift:

```
process(shiftRegister_cs) is  
begin  
    shiftRegister_ns <= shiftRegister_cs(shiftRegister_cs'left-1 d  
end process shift;
```

shiftregister sollte mit "1110" initialisiert werden!

→ Anode ist low-aktiv

Lookup table

- Die nibble werden mit 4 bit codiert
- zum Anzeigen müssen die nibble auf die Kathodenausgänge umgesetzt werden.
- Die Ausgangswerte werden dazu in einem lookup-table hinterlegt werden
- → case-when!

Lookup table

```
case nibVal_v is
  when "0000" =>
    segments_v := "10000001";
  when "0001" =>
    segments_v := "11001111";
  when "0010" =>
    segments_v := "10010010";
  -- usw
end case;
```