Protokoll zum Praktikum des Moduls Technische Informatik an der JLU Gießen

Technische Informatik Versuch 8

Julian Bergmann, Dennis Getzkow

5. September 2013

Versuch 8

1.1 Einführung

In diesem Versuch soll sich mit der Siebensegmentanzeige vertraut gemacht werden und ein Codeschloss mit 3 Versuchen bei 3 8–Bit–Zahlen und neuprogrammierbarer Zahlenkombination erstellt werden.

1.2 Teil 1: Siebensegment

Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

--Versuch 8, Aufgabe 1: Siebensegmentanzeige mit Schaltern verknüpfen, um 8 Schalter
     als 8-Bit-Zahl in Dezimal anzuzeigen.
entity dat31 is
    Port (
    o : in std_logic ; --Oszillator onboard, 50 MHz
    sch: in std_logic_vector(7 downto 0); --8 Schalter
    ANx: OUT std_logic_vector(3 downto 0); --4 Buchstaben/Ziffern anzeigbar
    DPex   : OUT std_logic; --Dezimalpunkt
    SEGex  : OUT std_logic_vector(6 downto 0) --7 Segmente pro Ziffer
     );
end dat31;

architecture Behavioral of dat31 is --Schnittstelle zu 7-seg-Anzeige
component seven_seg_wrapper is
   port(
      CLK  : IN std_logic; --clock
      I    : IN std_logic_vector(31 downto 0); --Was wird angezeigt
      MODE : IN std_logic_vector(1 downto 0);
      --Modus: 10: untere 14 Bit von I sind anzuzgeigende Zahl
      --Modus: 11: I ist 4 Bytes die anzuzgeigende ASCI-Zeichen darstellen
      AN   : OUT std_logic_vector(3 downto 0); --4 Ziffern/Buchstaben
      SEG  : OUT std_logic_vector(6 downto 0); --7 Segmente pro Ziffer
      DP   : OUT std_logic --Dezimalpunkt
   );
end component seven_seg_wrapper;

begin
sevenseg:seven_seg_wrapper
port map(
 CLK => o, --Clock is oszillator
 I(7 downto 0) => sch, --unteren 14 Bits sind anz. Zahl; hier 8-Bit-Zahl sch
 I(31 downto 8) => "00000000000000000000000", --restliche Ziffern 0
 MODE => "10", --Zeige I asl Zahl an
 AN => ANx, --Ausgabeweiterleitung s.o.
 DP => DPex,
 SEG => SEGex
);

end Behavioral;
```

Beschreibung

Neben der Einbindung der gegebenen VHDL–Datei des Betreuers zur Ansteuerung der 7–Segment–Anzeige wurde hier lediglich die Weiterleitung der Schalter an die Anzeige umgesetzt. Da die Anzeige im Modus "'10"' die letzten 14 Bits als Zahl interpretiert, wurde die Schaltereingabe an Position 0-7 und die übrigen Bits auf 0 gesetzt.

1.3 Teil 2: Codeschloss

Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Versuch 8, Teil 2: Codeschloss
entity dat32 is --unbeschr. Siehe Versuch 8, Teil 1
    Port (
    o : in std_logic ;
    sch: in std_logic_vector(7 downto 0);
    tas : in std_logic; --Taster; Enter zur Bestätigung der Zahlen
    ANx: OUT std_logic_vector(3 downto 0);
    DPex   : OUT std_logic;
    SEGex  : OUT std_logic_vector(6 downto 0);
    led: out std_logic_vector(2 downto 0); --3 LED zur Anzeige des Status:
   --001, 010, 100: 1., 2., 3. Zahl einzugeben
   --111: Code wird geprüft, LOCK wie OFFEN alle an, Unterscheidung ist 7-segm.
    reset: in std_logic; --Taster;Lock aufheben, Passwortversuche zurücksetzen
    neupass: in std_logic; --Taster;im offenen Zustand: neues Passwort definieren
    abort: in std_logic --Taster;Aktuelle Eingabe abbrechen, gehe zu 1. Zahl
        eingeben; im öffenen Zustand: schließe ab.
     );
end dat32;

architecture Behavioral of dat32 is
component seven_seg_wrapper is --7-Segment-Schnittstelle
   port(
      CLK  : IN std_logic;
      I    : IN std_logic_vector(31 downto 0);
      MODE : IN std_logic_vector(1 downto 0);
      AN   : OUT std_logic_vector(3 downto 0);
      SEG  : OUT std_logic_vector(6 downto 0);
      DP   : OUT std_logic
   );
end component seven_seg_wrapper;
component debouncer is --verhindert versehentlich mehrfachen Tasterdruck
   port(
      CLK  : in  std_logic;
      D    : in  std_logic;
      EDGE : out std_logic
   );
end component debouncer;
type state_type is(s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,offen); --Zustände, siehe jew.
    Zustand für Beschreibung
signal state :state_type :=s1; --Standard: gebe 1. Zahl ein, Schloss geschlossen
signal zahl1: std_logic_vector(7 downto 0); --eingegebene Zahlen
signal zahl2: std_logic_vector(7 downto 0);
signal zahl3: std_logic_vector(7 downto 0);
signal nzahl1: std_logic_vector(7 downto 0):="00000001"; --korrekte
    Zahlenkombination
signal nzahl2: std_logic_vector(7 downto 0):="00000010";
signal nzahl3: std_logic_vector(7 downto 0):="00000011";
signal zzahl1: std_logic_vector(7 downto 0);--Zahlen bei Neudefinition der korrekten
     Kombination
signal zzahl2: std_logic_vector(7 downto 0);--zzahl3 nicht notwendig: überschreibe
    dann nzahl3. Vorher soll Abbruch möglich sein.
signal schtoi:std_logic_vector(31 downto 0);--Eingabeinformationen für 7-segm.
signal edg: std_logic; --Bestätigungs-Taster nach Debouncer
signal segmod:std_logic_vector(1 downto 0):="10"; --7-Segment-Modus. default:
    letzten 14 Bit als Zahl
signal timer :natural range 1 to 50e6 :=1; --Zähler für 1s Wartezeit bei 50 MHz
signal fehler: std_logic_vector(1 downto 0):="00"; --Fehleingaben: 00, 01 und 10,
    dann Lock
begin
```

```
verschiebe: debouncer --Bilde Enter-Taster auf edge ab, vers. mehrfachen Druck
    verhindern
port map (
    d => tas,
    clk => o,
    edge => edg
);
sevenseg:seven_seg_wrapper --Schnittstelle 7-seg, siehe Teil 1.
port map(
 CLK => o,
 I => schtoi,
 MODE => segmod,
 AN => ANx,
 DP => DPex,
 SEG => SEGex
);
process(o)
begin
    if o'event and o='1' then
        if reset='1' then --Reset-Knopf
            state<=s1; --Zustand: 1. Zahl eingeben
            fehler<="00";--Fehlerzahl auf 0
            segmod<="10";--Zeige letzten 14 Bit als Zahl an
            schtoi <= "00000000000000000000000000000000"; --"000" Anzeigen
        end if;
        case state is
            when s1 => --1. Zahl eingeben für Unlock
                schtoi(31 downto 8) <= "000000000000000000000000";
                schtoi(7 downto 0)<=sch; --8 Schalter für 1. Zahl in 7-seg anzeigen
                led(0)<='1'; --LED: 1. Zahl eingeben
                led(1)<='0';
                led(2)<='0';
                if edg='1' then --Enter
                    zahl1<=sch; --1. Zahl aus Schalterstellung merken (zahl1)
                    state<=s2;--Zustand: gebe 2. Zahl ein
                end if;
            when s2 => --2. Zahl eingeben für Unlock
                led(0)<='0';--LED: 2. Zahl eingeben
                led(1)<='1';
                led(2)<='0';
                schtoi(31 downto 8) <= "000000000000000000000000";
                schtoi(7 downto 0)<=sch;--s.o.
                if abort='1' then --Abbruch: Zustand 1. Zahl eingeben
                  state<=s1;
                end if;
                if edg='1' then --Enter: 2. Zahl merken, Zustand: 3. Zahl eingeben
                    zahl2<=sch;
                    state<=s3;
                end if;
            when s3 =>--2. Zahl eingeben für Unlock
                led(0)<='0';--LED: 3. Zahl eingeben
                led(1)<='0';
                led(2)<='1';
                if abort='1' then --Abbruch
                   state<=s1;
                end if;
                schtoi(31 downto 8) <= "000000000000000000000000";
                schtoi(7 downto 0)<=sch;
                if edg='1' then --Enter
                    zahl3<=sch;--Schalter als Zahl merken => zahl3,
                    state<=s4; --Prüfe Eingabe
                end if;
            when s4 =>--Eingabe wird geprüft
                led(0)<='1'; --Alle LED an
                led(1)<='1';
                led(2)<='1';
                if (zahl1=nzahl1) and (zahl2=nzahl2) and (zahl3=nzahl3) then
```

```
            state<=offen;--Zahlen korrekt: Zustand "offen"
        else --Falsche eingabe
            timer<=1;--Timer zurücksetzen, warte 1s.
            state<=s5;--Zustand: Zeige Fehler an
            if fehler="00" then --Fehleranzahl erhöhen
                fehler<="01";
            elsif fehler="01" then
                fehler<="10";
            elsif fehler="10" then
                fehler<="11";
            end if;
        end if;
    when offen => --Schloss geöffnet
        segmod<="11"; --Zeige ASCI-Zeichen an
        schtoi<="01001111010100000100010101001110";--"OPEN" anzeigen
        fehler<="00";--Fehlerzahl auf 0
        if abort='1' then --Abbruch-Taste 'schließt'
           state<=s1;--Zustand 1. Zahl eingeben
           segmod<="10";--Zeige Zahlen an.
        end if;
        if neupass='1' then --Neues Passwort-Taster
            timer<=1;--1s Warten
            state<=s7;--Zustand: "NEW" anzeigen und 1. neue Zahl eingeben
        end if;
    when s5 => --Fehleingabe, Zeige bisherige Fehler an
        if timer=50e6 then --Nach einer Sekunde:
            segmod<="10";--Zahlen anzeigen
            schtoi(31 downto 2)<="00000000000000000000000000000000";
            schtoi(1 downto 0)<=fehler;--Zeige bisherige Fehleranzahl an
            if fehler ="11" then--3. Versuch falsch
                state<=s6; --Dann LOCK-Zustand
            elsif edg='1' then --Ansonsten warte auf ENTER-Taster für
               state<=s1;-- neuer Versuch, 1. zahl
               segmod<="10";
            end if;
        else --Eine Sekunde lang "ERR" anzeigen
            segmod<="11";
            schtoi<="01000101010100100101001000100000"; --"ERR"
            timer<=timer+1;
        end if;
    when s6 => --LOCK-Zustand
        segmod<="11";
        schtoi<="01001100010011110100001101001011"; --Zeige "LOCK" an
    when s7 => --Neues Passwort
        if timer=50e6 then--Nach 1s: 1. Zahl für neues Passwort eingeben
            segmod<="10";--siehe s1.
            schtoi(31 downto 8) <= "00000000000000000000000000";
            schtoi(7 downto 0)<=sch;
            led(0)<='1';
            led(1)<='0';
            led(2)<='0';
            if edg='1' then
                zzahl1<=sch;--zzahl1 ist Zwischen-Zahl: erst nach 3. Zahl
                    wird übernommen
                state<=s8; --2. neue Zahl eingeben
            end if;
            if abort='1' then --Abbruch: gehe zum Zustand "OFFEN" zurück
                state<=offen;
            end if;
        else --Eine Sekunde lang "NEW" anzeigen
            segmod<="11";
            schtoi<="00000000010011100100010101010111";--"NEW"
            timer<=timer+1;
        end if;
    when s8 =>--2. Zahl für neues Passwort, siehe s2 und s7
        segmod<="10";
        schtoi(31 downto 8) <= "00000000000000000000000000";
```

```
                schtoi(7 downto 0)<=sch;
                led(0)<='0';
                led(1)<='1';
                led(2)<='0';
                if edg='1' then
                    zzahl2<=sch;
                    state<=s9;
                end if;
                if abort='1' then
                    state<=offen;
                end if;
            when s9 =>--3. Zahl für neues Passwort, siehe s3 und s7
                segmod<="10";
                schtoi(31 downto 8) <= "000000000000000000000000";
                schtoi(7 downto 0)<=sch;
                led(0)<='0';
                led(1)<='0';
                led(2)<='1';
                if edg='1' then --Enter: neues Passwort wird übernommen
                    nzahl3<=sch;
                    nzahl2<=zzahl2;
                    nzahl1<=zzahl1;
                    timer<=1;
                    state<=s10; --"ACX" wird 1s angezeigt, danach zu Zustand "OFFEN"
                end if;
                if abort='1' then
                    state<=offen;
                end if;
            when s10 =>
                if timer=50e6 then --nach 1s Zustand offen
                    state<=offen;
                else --1s "ACX" anzeigen
                    segmod<="11";
                    schtoi<="00000000010000010100001101011000";
                    timer<=timer+1;
                end if;
        end case;
    end if;
end process;
end Behavioral;
```

Beschreibung

Hier wurden nun die Schalter, wie auch die 7–Segment–Anzeige, mit jeweils einer Variable verknüpft. In den Zuständen s1, s2 und s3 wurden diese Variablen ebenfalls verknüpft, um die mit den Schaltern eingestellten Zahlen anzuzeigen. Diese Zustände werden benutzt, um die 3 Zahlen einzugeben, mit denen die intern als "'richtig"' eingestellten Zahlen verglichen werden. Die Bestätigung der Eingabe erfolgt hierbei mit dem Enter-Taster, welcher zunächst über "'tas"' eingebunden und nach dem "'debouncer"' als "'edg"' eingebunden wird. Der Debouncer verhindert, dass ein Drücken als mehrfaches Drücken registriert wird.

Nach der Eingabe der 3. Zahl wechselt der Automat in den Zustand s4, wo die Passwort–Gültigkeit geprüft wird. Stimmt das Passwort, wird in den Zustand "'OFFEN"' gewechselt, andernfalls die Fehleranzahl erhöht und bei unter 3 Fehlern zunächst "'ERR"' für 1 s angezeigt, woraufhin die Anzahl der bisherigen Fehler erscheint und nach Drücken der Enter–Taste zur erneuten Eingabe der 1. Zahl im Zustand s1 aufgefordert wird. Bei 3 Fehlern blockiert das Schloss.

In jedem Zustand ist es dabei möglich, die Fehleranzahl wieder auf 0 zu setzen und zu s1 zurück zu kehren, indem der Reset–Taster gedrückt wird. Der entsprechende Code–Teil ist

hier vor der Prüfung, in welchem Zustand sich der Automat gerade befindet.

Auch im offenen Zustand wird die Fehleranzahl auf 0 zurück gesetzt. Hier erscheint nun "'OPEN"' als Text. Der Abbruch–Taster schließt das Schloss erneut und leitet in den Zustand s1. Nun kann auch der "'neue Code"'–Taster verwendet werden. Nach Drücken dieses wird "'NEW"' 1 s angezeigt und eine Zahleneingabe erwartet. Wie in Zustand s1-s3 werden hier in Zustand s7-s9 3 8–Bit–Zahlen erwartet. die ersten beiden zahlen werden hier in zzahl1 und zzahl2 gespeichert, um jederzeit mit dem Abbruch–Taster wieder in den zustand "'OFFEN"' zu gelangen ohne das Passwort zu überschreiben. Nach Eingabe der 3. Zahl wird das Passwort überschrieben, es erscheint "'ACX"' 1 s auf dem Display und man landet wieder im Zustand "'OFFEN"'. s10 wird hier lediglich dazu verwendet, 1 s zu warten, bevor von der Anzeige "'ACX"' wieder in "'OFFEN"' gewechselt wird.