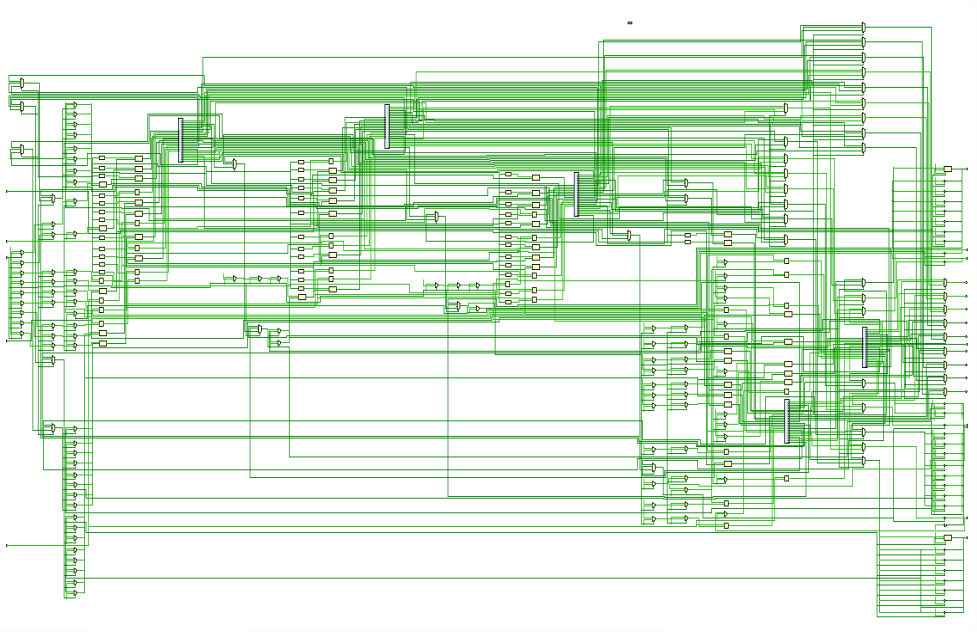


Pynq-Board Demo Design

Benjamin Lagershausen - Keßler

UNIKASSEL
VERSITÄT

1. Motivation und Problemstellung
2. Minimal Design für Bidirektionale I/Os
3. Code-Template für Pojekte im Demo Design
4. Toplevel Design
5. Steuermenü und 7-Segment-Anzeige
6. Boot-Image

1. Motivation und Problemstellung

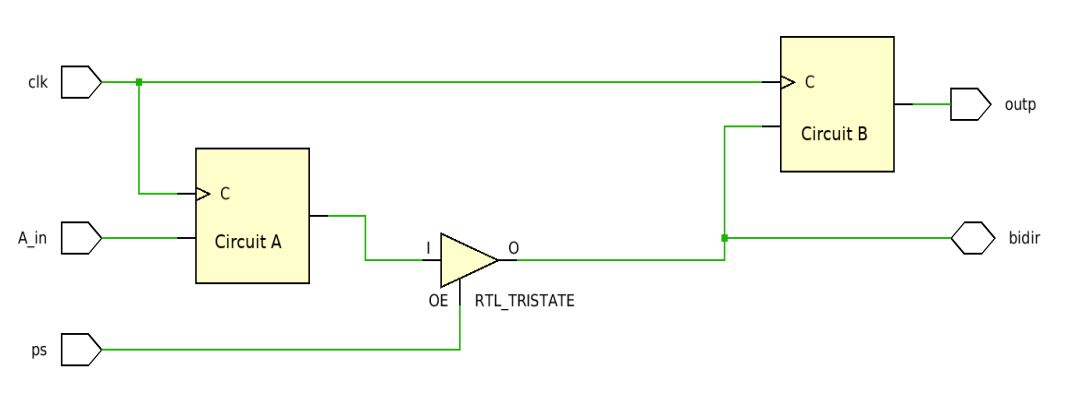
Motivation:

- Umstellung der Hardware in LVs auf Pynq-Boards
- Vielzahl von Designs in VHDL-Kurs und Praktikum
- Einfache und schnelle Demonstration von Schaltungen

Problemstellungen:

- Es werden bidirektionale I/Os benötigt
- Template um neue Designs einzubinden
- Toplevel-Struktur
- Steuerung mit Anzeige
- Erstellen eines Bootimages

2. Minimal Design Bidirektionale I/Os



- Bidir – Bidirektionaler I/O
- PS – Pinmode Select: schaltet einen Tristate-Buffer entweder hochohmig oder leitet das Eingangssignal weiter.
- Problem: Wenn PS = ‘1’, treibt “Circuit A” sowohl “Bidir” als auch “Circuit B”.

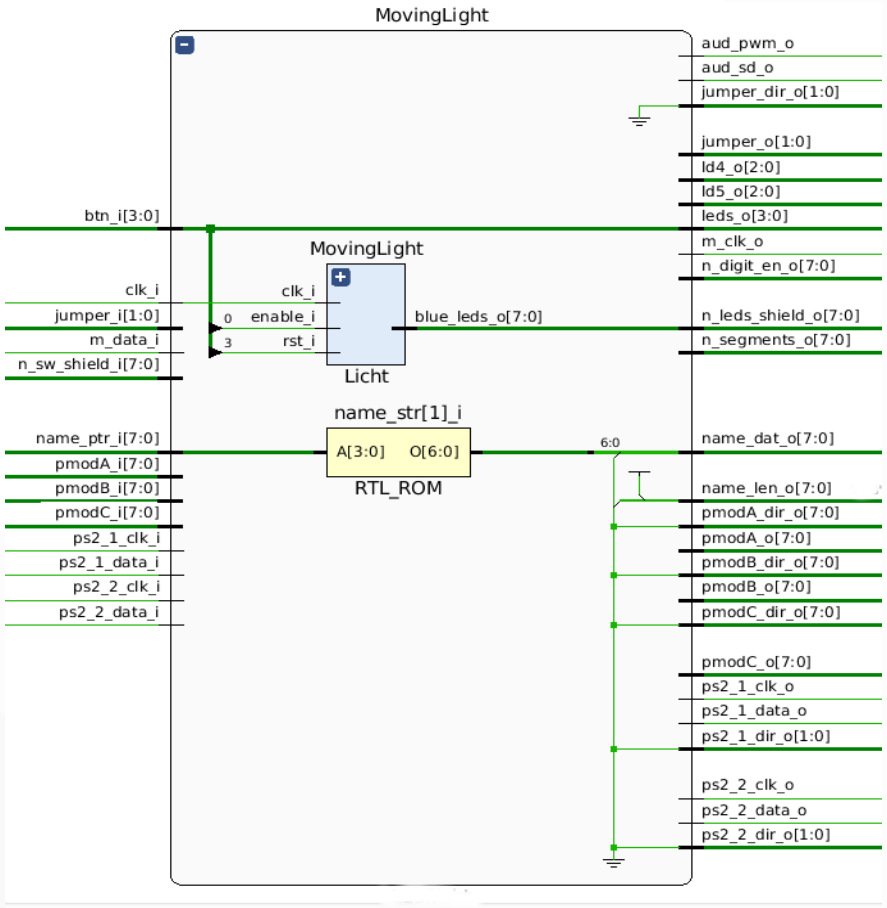
```
4  ENTITY bidir IS
5      PORT(
6          bidir   : INOUT STD_LOGIC;
7          ps, clk, A_in : IN STD_LOGIC;
8          outp    : OUT STD_LOGIC);
9  END bidir;
10
11 ARCHITECTURE test OF bidir IS
12
13 signal a : STD_LOGIC;
14 signal b : STD_LOGIC;
15
16 begin
17     process(clk)
18     begin
19     if rising_edge(clk) then
20         a <= A_in;
21         outp <= b;
22         END IF;
23     end process;
24
25     with ps select
26     bidir <= 'Z' when '0',
27              a when '1';
28     b <= bidir;
29 end test;
```

3.Code-Template für neue Pojekte

```
...
use work.records_p.all;

entity Module is
    Port ( name_ptr_i : in std_logic_vector(CHAR_WIDTH-1 downto 0);
           name_len_o : out std_logic_vector(CHAR_WIDTH-1 downto 0);
           name_dat_o : out std_logic_vector(CHAR_WIDTH-1 downto 0);

           ... Alle I/Os des Pynq-Boards ...
      (Bidirektionale Ports als Input und Output)

           );
end Module;

architecture Behavioral of Module is

-- Name
constant name_str : string := "ICH HEISSE HUbErt";

-- Richtungen
constant pmodA_dir : std_logic_vector (PMOD_WIDTH-1 downto 0) := "00000001";
constant pmodB_dir : std_logic_vector (PMOD_WIDTH-1 downto 0) := (others => '0');
constant pmodC_dir : std_logic_vector (PMOD_WIDTH-1 downto 0) := (others => '0');
constant jumper_dir : std_logic_vector (JUMPER_WIDTH-1 downto 0) := (others => '0');
constant PS2_1_dir : std_logic_vector (1 downto 0) := (others => '0');
constant PS2_2_dir : std_logic_vector (1 downto 0) := (others => '0');

-- Toplevel-Komponente des User-Projektes:
component Hubert IS
    port( clk_i : in std_logic;
          input_i : in std_logic;
          output_o : out std_logic );
END component;
```

3.Code-Template für neue Pojekte

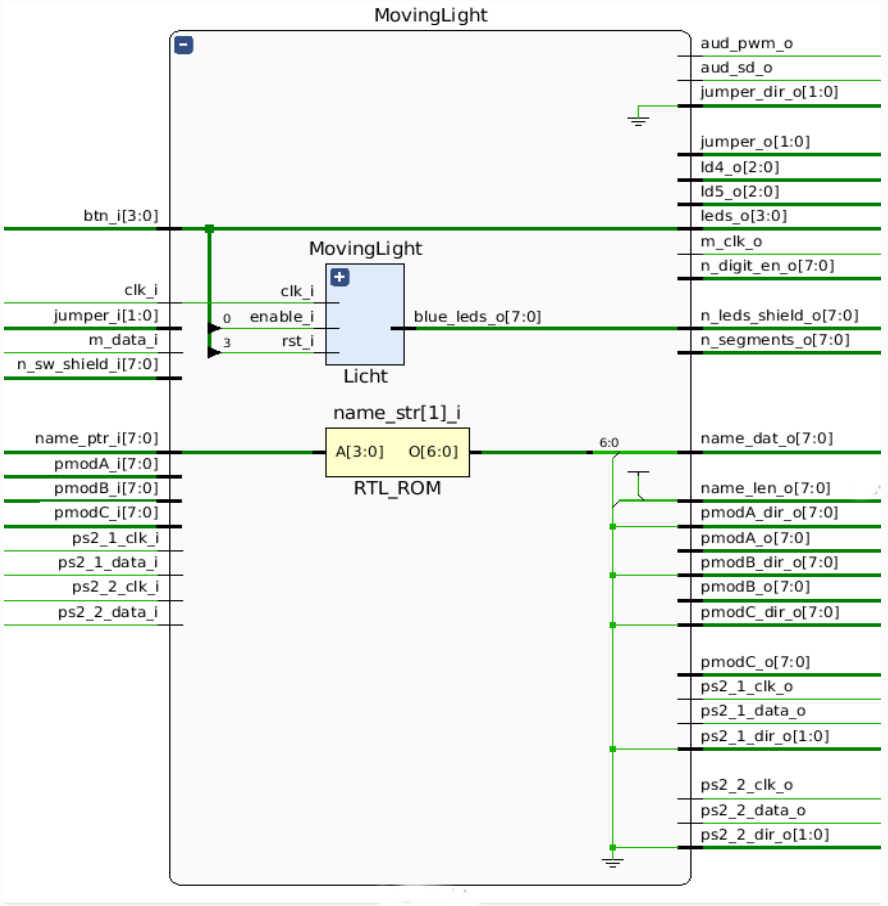
```
begin

-- Namensinterface:
name_len_o <= std_logic_vector(to_unsigned(name_str'length, name_len_o'length));
name_dat_o <= std_logic_vector(to_unsigned(character'pos(name_str(to_integer(unsigned(name_ptr_i)))), 8));

-- I/O Richtungen:
pmodA_dir_o <= pmodA_dir;
pmodB_dir_o <= pmodB_dir;
pmodC_dir_o <= pmodC_dir;
jumper_dir_o <= jumper_dir;
PS2_1_dir_o <= PS2_1_dir;
PS2_2_dir_o <= PS2_2_dir;

-- Instanz der Toplevel-Komponente des User-Projektes:
Hubi : Hubert
port map ( clock => clk_i,
           input_i => button_i(0),
           output_o => pmodA_o(0) );

end Behavioral;
```



3.Code-Template für neue Pojekte

Vorteile

- Einfaches und übersichtliches Design
- Es wird nur ein Constraint-File für alle Designs benötigt
- Das Constraint-File muss nicht mehr bearbeitet werden



4.Toplevel Design

Datentypen

Input Record:

- Alle Signale, die von außen in das FPGA hineinlaufen.

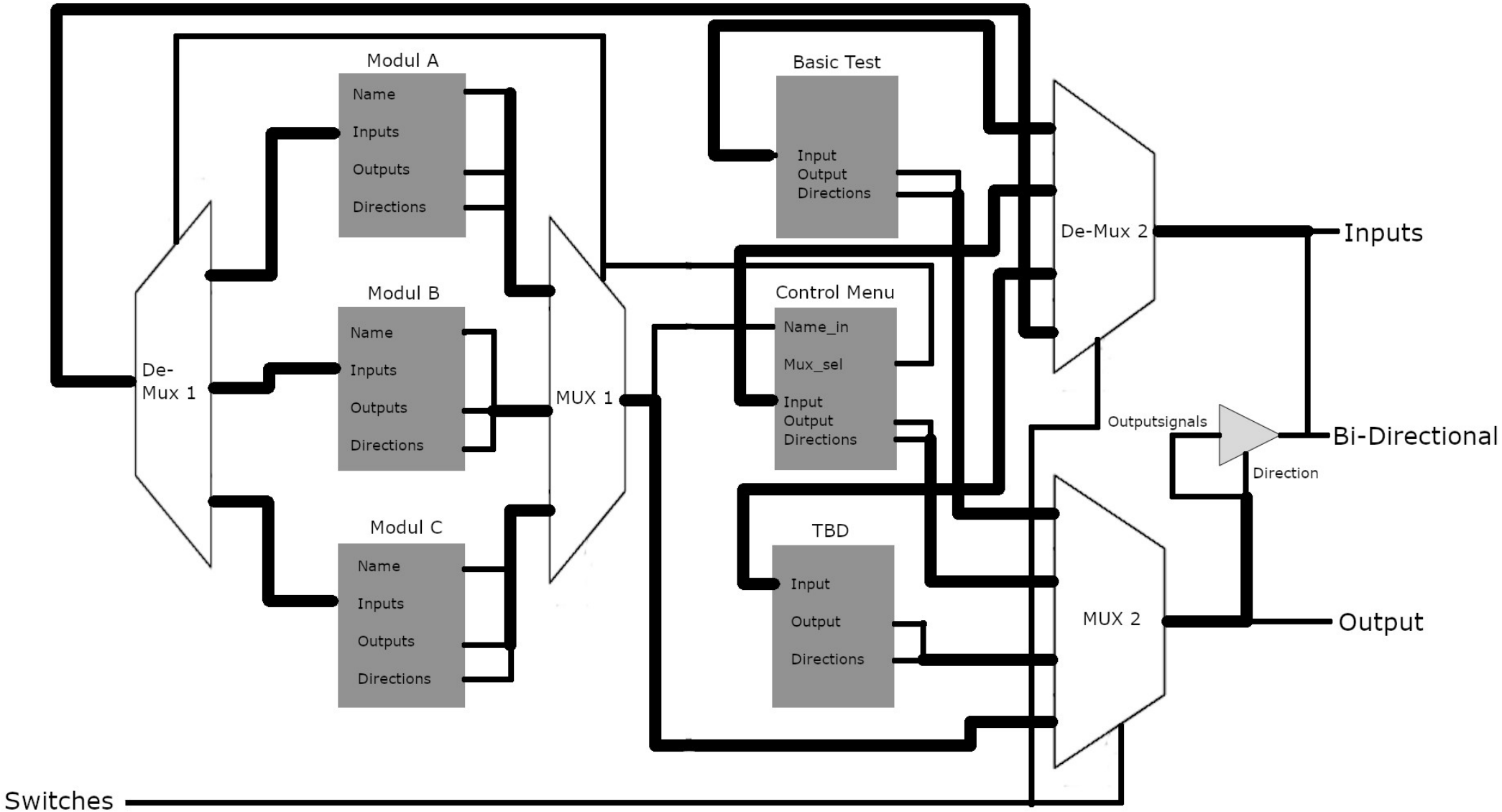
Output Record:

- Alle Signale, die das FPGA verlassen.
- Alle Signale, die die Richtung von Bidirektionalen Ports bestimmen.

Name Record:

- Zuständig für die Übermittlung der Modulnamen an das Steuermenu.

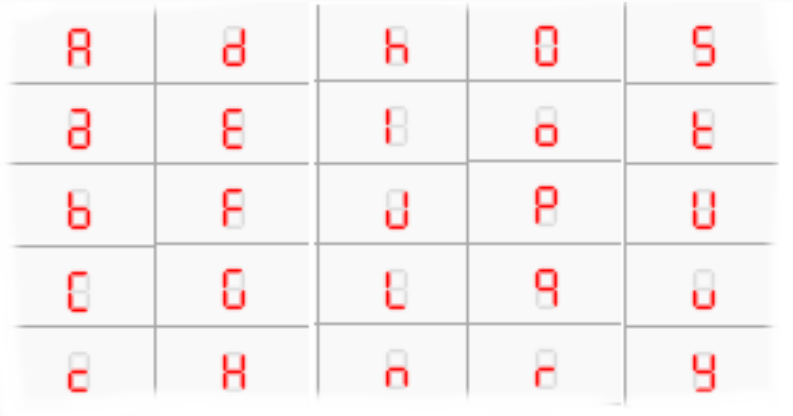
4. Toplevel Design



5. Steuermenu und SSD

<u>Aufgaben :</u>

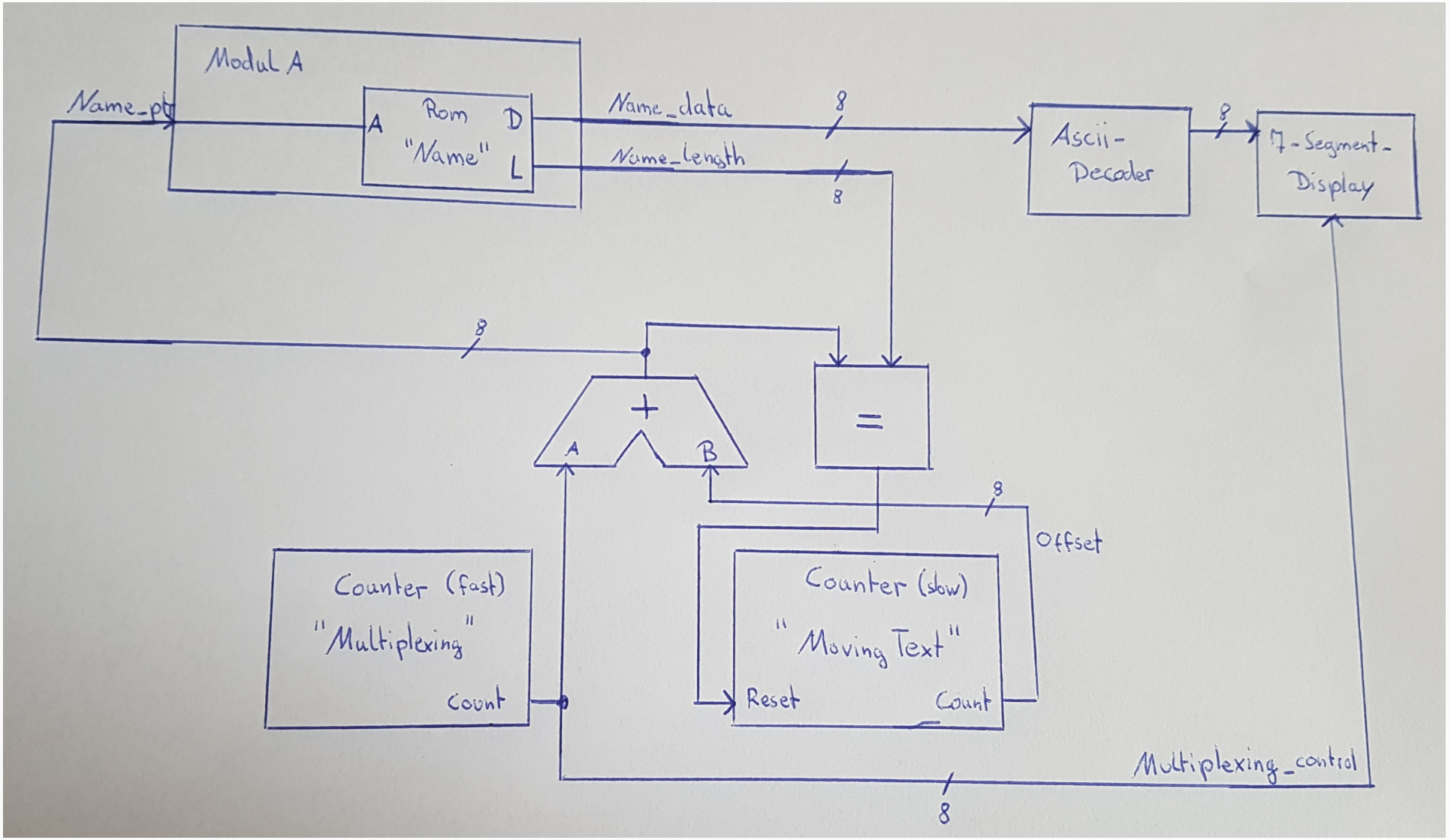
- Abfragen der Taster
  - Entprellung und Flankenerkennung



- Auswahl des Aktiven Moduls

- Anzeige des Modulnamens auf dem SSD des Pynq Shields
  - ASCII → 7-Segment Decoder.
  - Wenn kürzer als 8 Zeichen unbeweglich
  - Wenn länger als 8 Zeichen als Laufschrift

5. Steuermenu und SSD



6. Boot - Image

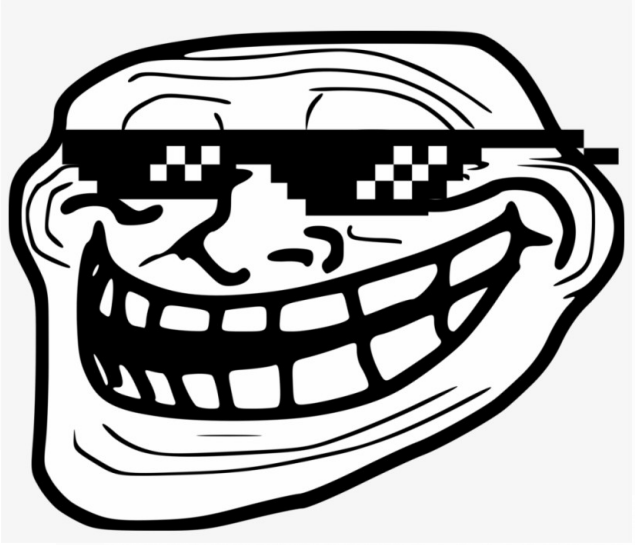
- Der QSPI-Flash des Pynq-Boards ist nur mit dem Zynq-PS verbunden.
- Das Zynq-PS ließt den QSPI-Flash und programmiert den FPGA.

Lösung:

- Block-Design nur mit Zynq-PS erzeugen.
- Block-Design in den VHDL-Code einbinden, jedoch mit NICHTS verbinden.
- Bitstream des Projektes erzeugen.
- Hardware exportieren.
- Im Xilinx-SDK einen FSBL erzeugen.
- Mit dem FSBL und dem Bitstream kann nun ein Boot-Image erzeugt und hochgeladen werden.

6. Boot - Image

- Boot Image erstellen
- Boot Image hochladen
- Läuft!



← Xilinx

Ich →

Denkst du…

Nicht ohne Processing System