Running program t on MIX

We will run programm t of exercise 20 in chapter 1.3.2 TAOCP (p. 161) on MIX. Programm t controls the traffic signal at corner of Del Mare Boulevard and Berkeley Avenue. This project will connect LEDs directly to the X-Register and a push button to the Overflow toggle. This will be done extending the fpga design and routing the appropriate signals to the GPIO connector at the back of MIX.

1 Extending the fpga desing

Make a copy of the folder `rtl` and cd into it.

1.1 mix.pcf

mix.pcf defines the physical constraints. Add the following lines to define new signal wires.

Consult the datasheet `pics/iCE40HX8K–EVB` to understand the meaning of `E4`,`B2`, ...

```
#at the GPIO connector you find
set_io rx   E4      # GPIO 5
set_io tx   B2      # GPIO 7

set_io dmred    F5  # GPIO 9
set_io bred     J2  # GPIO 10
set_io dmamber  B1  # GPIO 11
set_io bamber   H1  # GPIO 12
set_io dmgreen  C1  # GPIO 13
set_io bgreen   G1  # GPIO 14
set_io dmdw     C2  # GPIO 15
set_io bdw      J5  # GPIO 16
set_io dmw      F4  # GPIO 17
set_io bw       H2  # GPIO 18
set_io button   D2  # GPIO 19
```

1.2 mix.v

Add the following lines to the hardware description file `mix.v` to connect the Register rX with the traffic signals.

```
// MIX - 1009
// Don Knuths computer architecture described in "The Art of Computer Programming"

`default_nettype none
module mix(
    input wire clk_in,
    input wire rx,
    output wire tx,
    output wire hlt,
    input wire button,
    output wire dmgreen,
```

```
    output wire dmamber,
    output wire dmred,
    output wire bgreen,
    output wire bamber,
    output wire bred,
    output wire dmw,
    output wire dmdw,
    output wire bw,
    output wire bdw
);

assign dmgreen = RegisterX[19:18] == 2'd1;
assign dmamber = RegisterX[19:18] == 2'd2;
assign dmred = RegisterX[19:18] == 2'd3;
assign bgreen = RegisterX[13:12] == 2'd1;
assign bamber = RegisterX[13:12] == 2'd2;
assign bred = RegisterX[13:12] == 2'd3;
assign dmw = RegisterX[7:6] == 2'd1;
assign dmdw = RegisterX[7:6] == 2'd2;
assign bw = RegisterX[1:0] == 2'd1;
assign bdw = RegisterX[1:0] == 2'd2;
```

Find the code snipped that controls the overflow toggle and add the line commented with "`# traffic signal button`".

```
reg overflow;
reg less;
reg equal;
reg greater;
always @(posedge clk)
    if (reset) overflow <= 0;
    else if (button) overflow <= 1;     #traffic signal button
    else if (add2) overflow <= addof;
    else if (sub2) overflow <= subof;
    else if (ide) overflow <= (rA|rX)? ideout[30] : ideout[12];
```

2 rebuild and flash to iCE40HX8K-EVB

Rebuild the fpga project and upload. ¸`apio clean` is needed, because otherwise the the preloaded memory will not be updated. “‘ apio clean apio build -v apio upload “‘

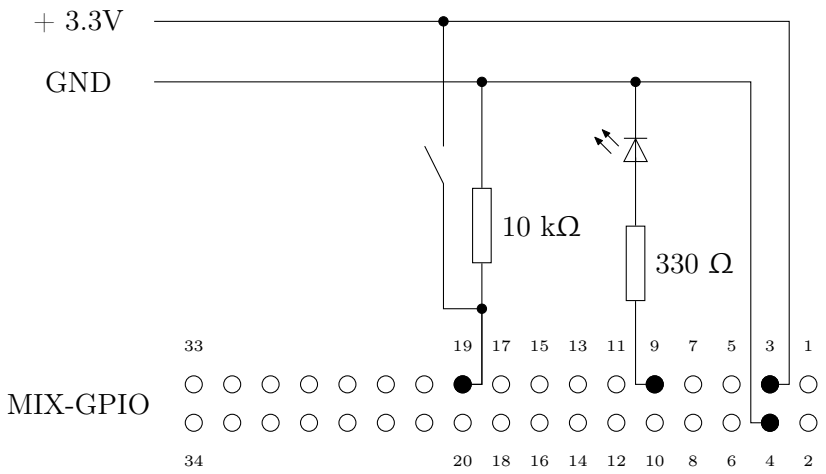
Tipp: change the welcome message to ensure the new rom file has been uploaded.

```
apio clean
apio upload –v
```

3 leds and button

Connect leds and button (don't forget resistors) to the appropriate GPIO connectors as described in the `mix.pcf` file.

Caution: pins 1,3,5 and 7 are "reserved"for the power supply and the USB serial connector.



4 t.mixal

Compile `t.mixal`, upload to MIX and run the traffic signals.