Visualization of a True Random Number Generator

Project 7: Xiphera

Matti Jämsen, Virpi Sumu

20.8.2021

Aalto-yliopisto
Sähkötekniikan korkeakoulu

Team members

Matti Jämsen

2nd year Electrical Engineering

Virpi Sumu

1st year Automation with a background in Computer Science

Xiphera Oy

- **Matti Tommiska, CEO**
- **Valtteri Marttila, developer**

Project topic

Xiphera needs a way of demonstrating the advantages their products offer in an easily understandable, visual way to a wider audience, for example at trade fairs and exhibitions.

Xiphera has an IP block for generating true, robust random numbers. The project aims to visualize the advantages of their IP block when compared to generic pseudo random number generators.

Project components

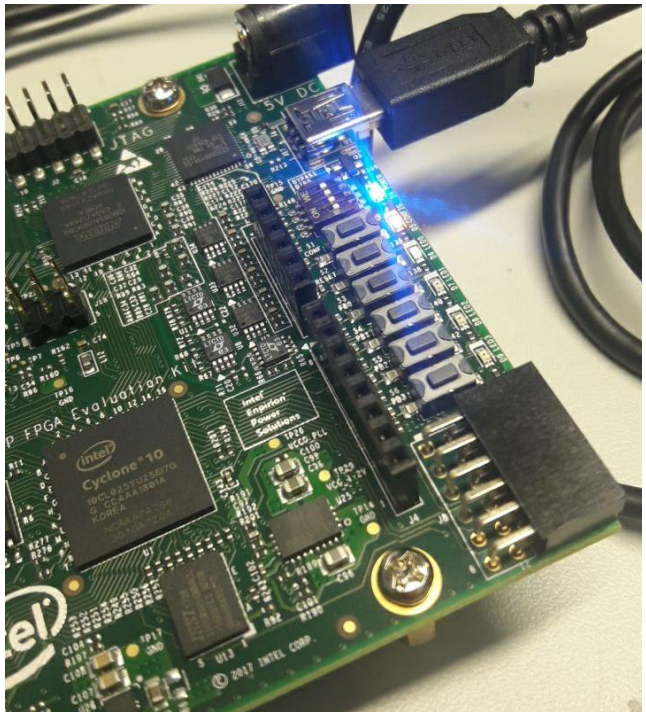
The main components of the project are

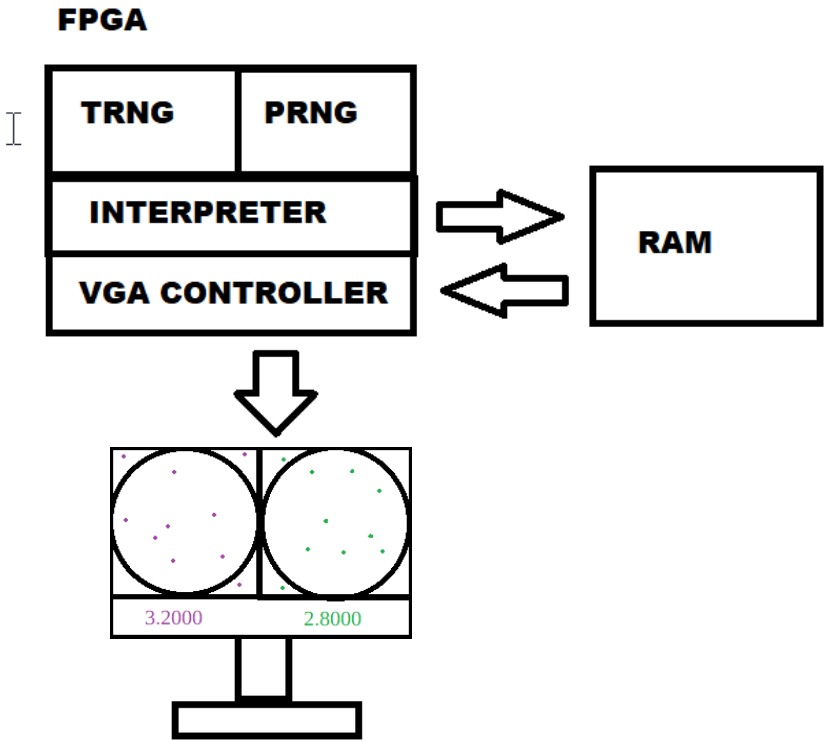
- Handling the Xiphera XIP8001B IP block in Intel 10 LP FPGA Ev. kit
- Pseudo random number generator, implemented with a linear feedback shift register to compare deterministic properties with TRNG
- Converting random bit vectors to integer coordinates within a unit circle
- Storage of visualization parameters in RAM
- Output to screen using the VGA standard

Additionally, the project includes the physical components

- VGA-PMOD connector
- Handheld screen

Project components





Process

- Learning the basics
  - Hardware description languages (VHDL, Verilog)
  - FPGA (Field Programmable Gate Array)
  - Cryptography (random number properties and generation)
  - Hardware design mindset – not consecutive, but concurrent
  - VGA standard & connector, testing with Arduino
- All final designs in VHDL
- Appropriately testbenched
- ...modelling the physical components still to be completed

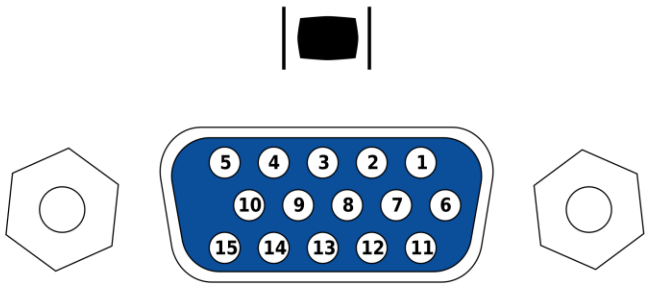
VHDL learning curve

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ex01_structure is
    port(r0, r1, clk : in bit;
         q0, q1, : out bit);
end entity;

architecture struct of ex01_structure is
    constant limit : integer := 255;
    signal int_clk : bit;
begin
    storage : process is
        variable st_r0, st_r1 : bit;
    begin
        wait until clk;

    end process;
end architecture;
```

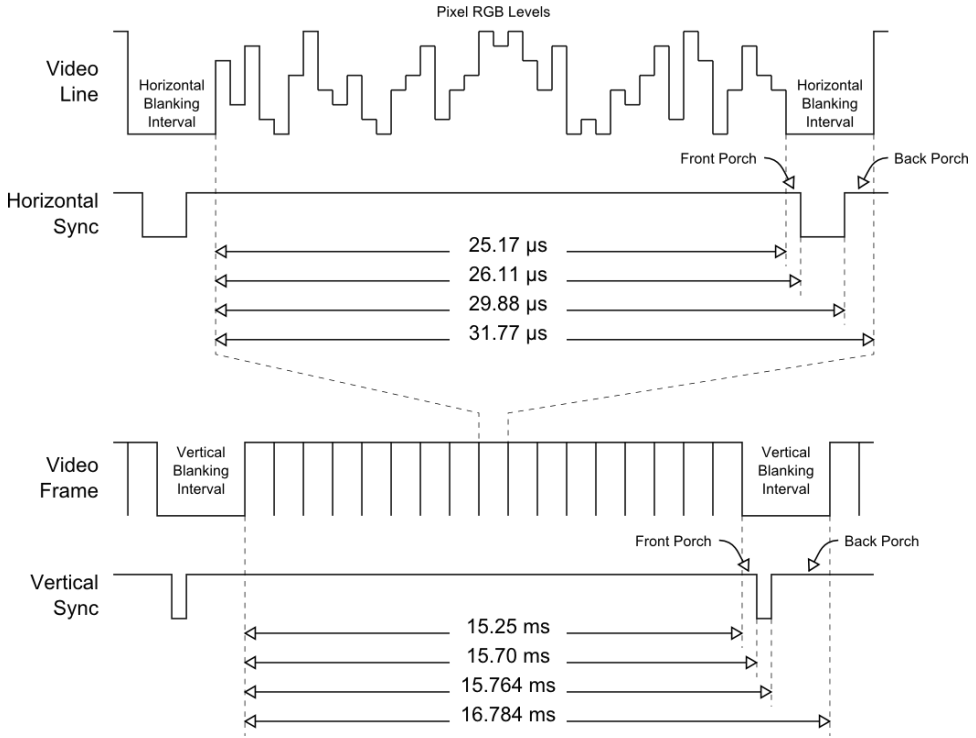
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity lfsr_16_tb is
end entity;

architecture arch of lfsr_16_tb is
    component lfsr is
        port (
            clk, rst  : in std_logic;
            starter   : in std_logic_vector;
            res       : out std_logic_vector(31 downto 0);
            rd        : in std_logic;
            empty     : out std_logic
        );
    end component;

    signal clk_s, rst_s, rd_s, empty_s  : std_logic;
    signal starter_s    : std_logic_vector(15 downto 0) := x"1234";
    signal res_s        : std_logic_vector(31 downto 0) := x"00000000";

begin
    dut: lfsr port map (clk_s, rst_s, starter_s, res_s, rd_s, empty_s);

    clk_proc: process
    begin
        clk_s <= '1';
        wait for 10 ns;
        clk_s <= '0';
        wait for 10 ns;
    end process;

    update: process
    variable result : std_logic_vector(31 downto 0) := x"00000000";
```

VGA standard & connector

Connector:

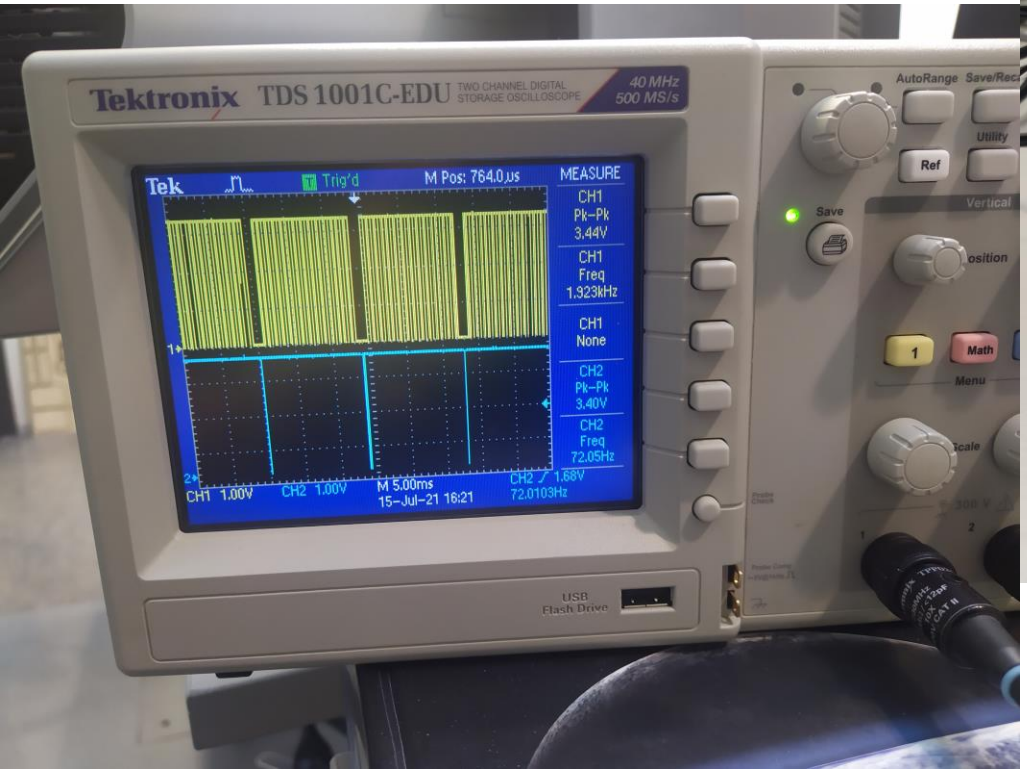


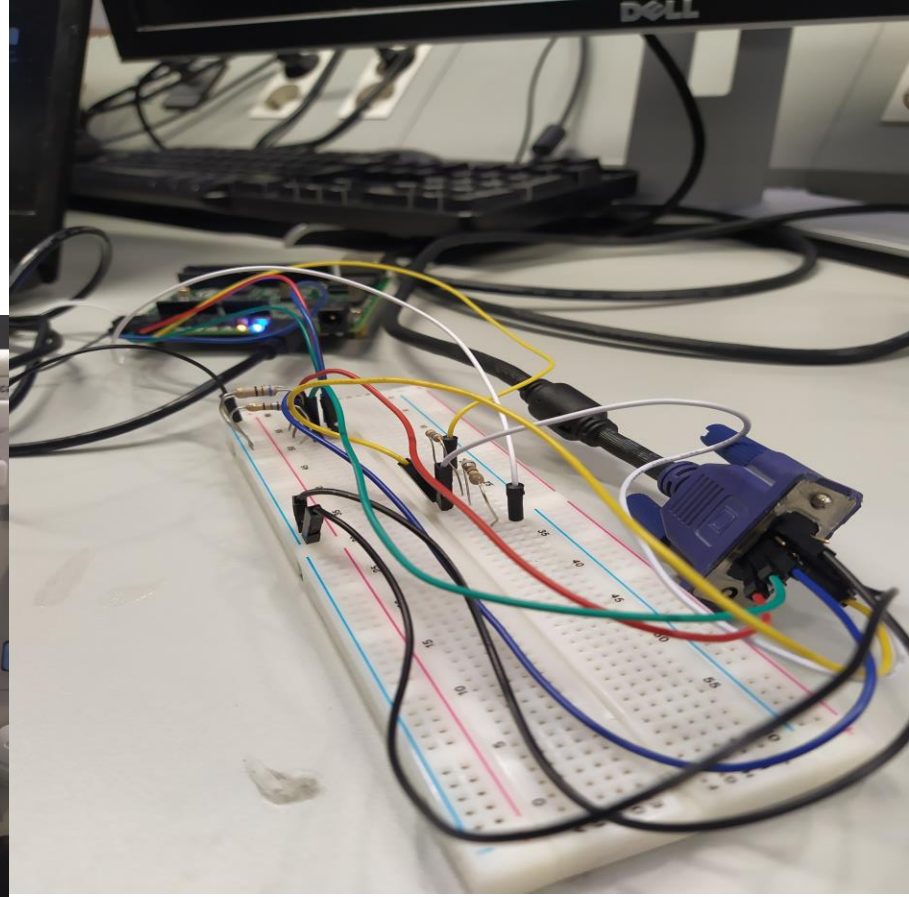
https://en.wikipedia.org/wiki/VGA_connector#/media/File:VGA_Port.svg



https://i.stack.imgur.com/vvq87.png

VGA standard & connector

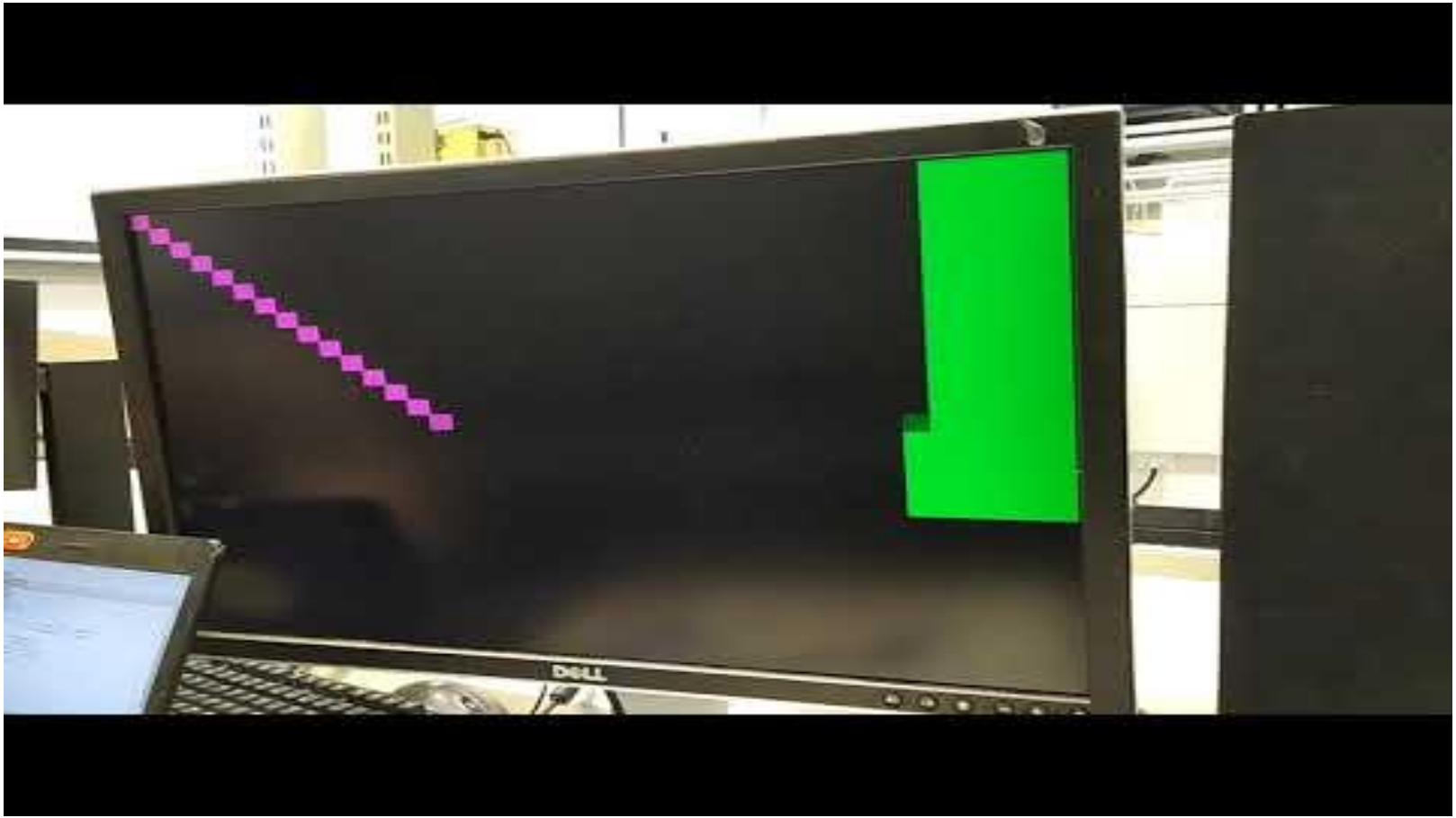




Working connector prototype

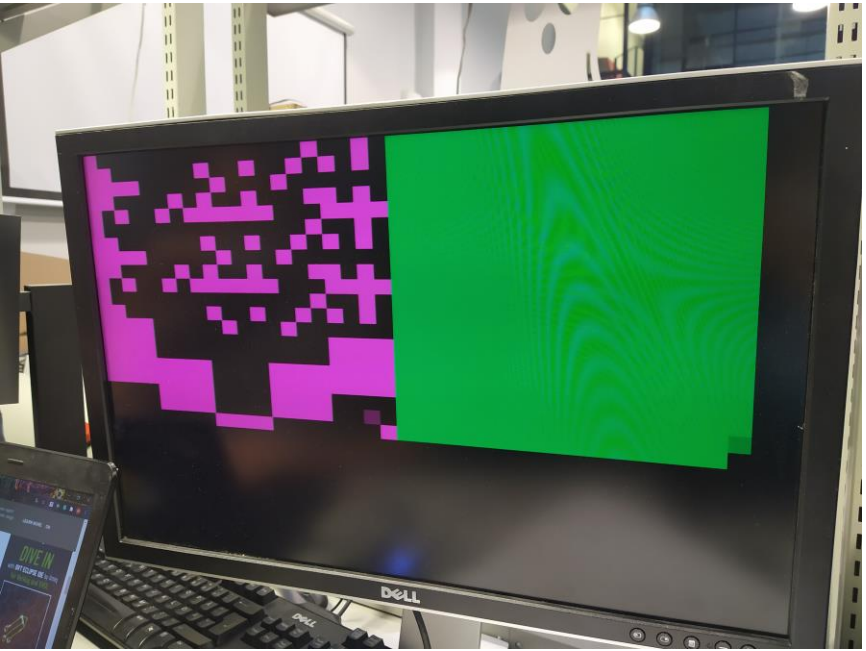
Verifying the VGA signal

Visualization



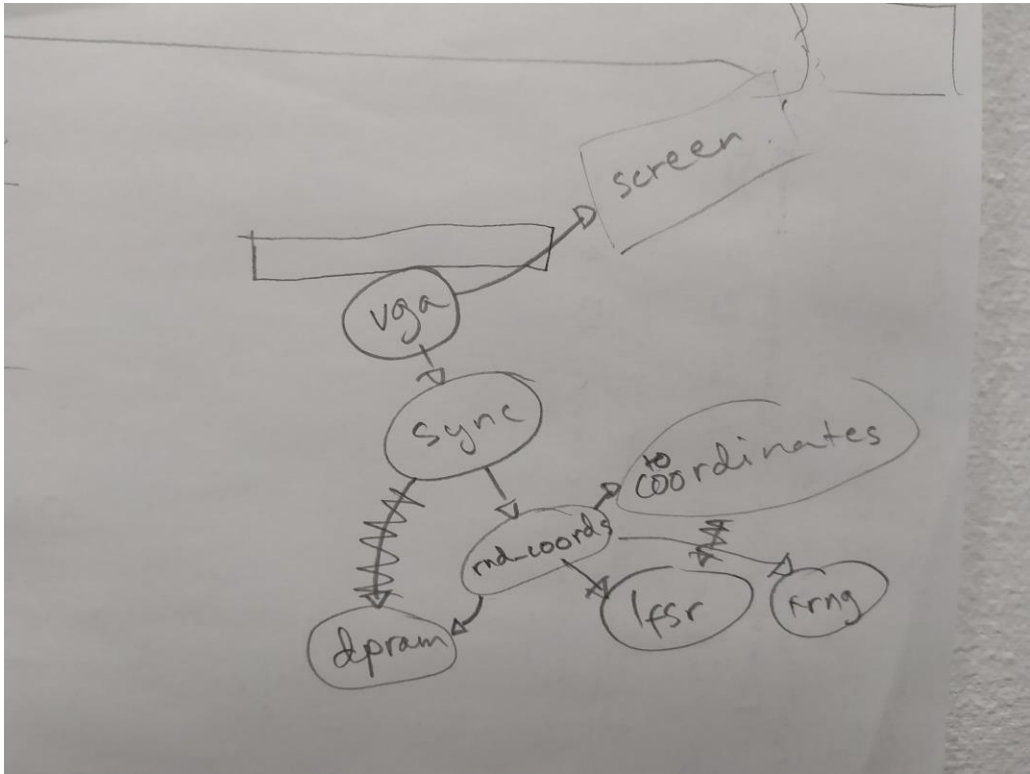
A?

Visualization

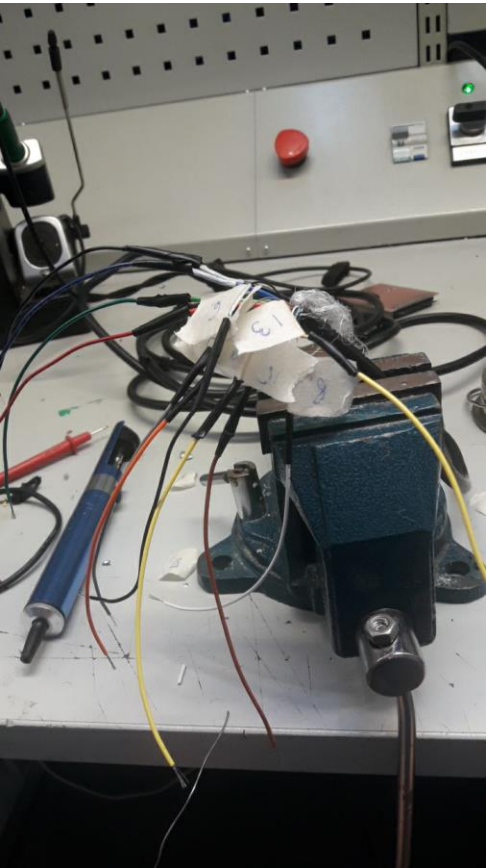


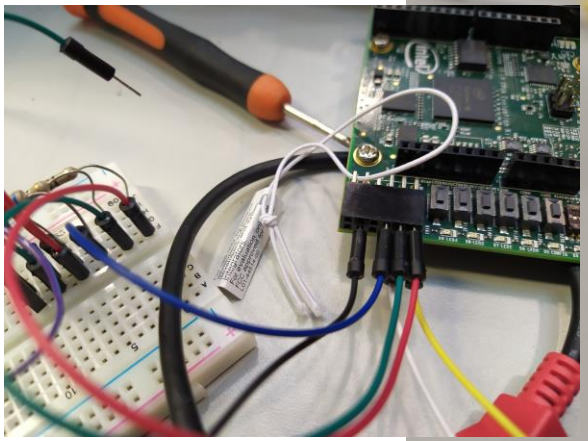
Trying out different ways of accessing pixels

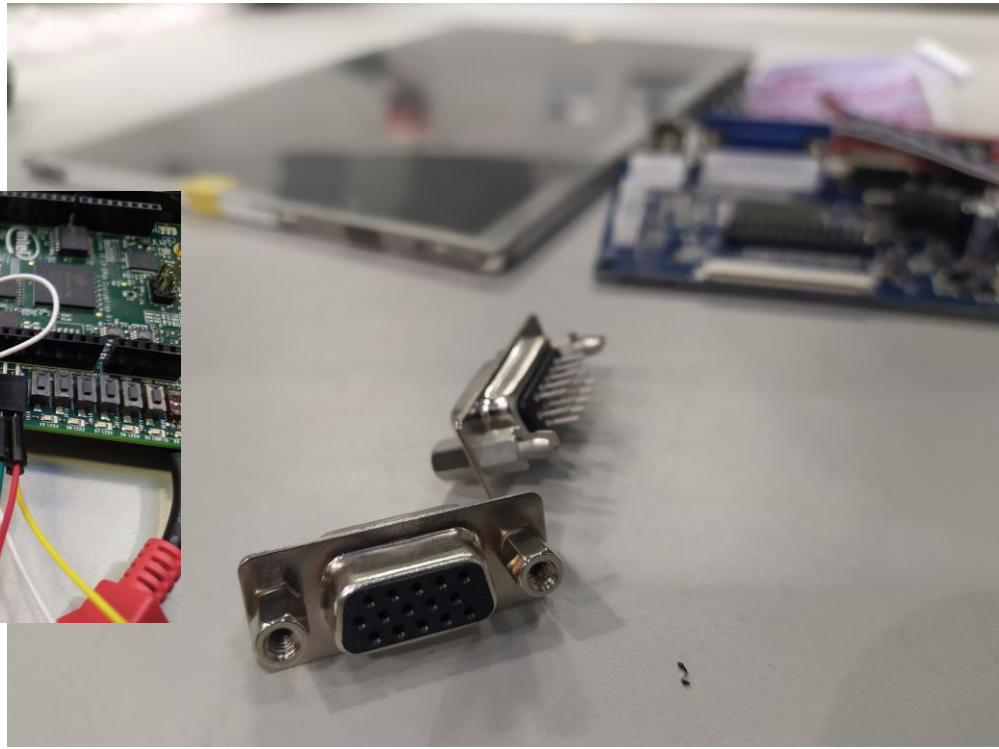
Towards a final design: draft of project hierarchy



Physical components







Work in progress

Current situation and final result

Done:

VGA controller

RAM controller

TRNG wrapper (not tested yet)

PRNG with wrapper

PMOD => VGA prototype

Sketch of what the finished visualization should look like

Under work:

Finish Interpreter with pi approximator and few pixel maps

Pixel size down

Nice casing

VHDL polishing

User manual

Comments & questions?

Thank you for your attention!