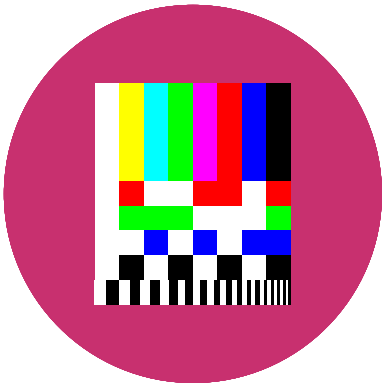
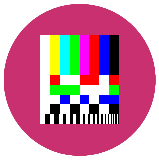


Display

Lecture Digital Design (DiD)

Orientation: Information and Communication Technology (ISC)
Specialisation: Data Engineering (DE)
Course: Digital Design (DiD)
Authors: Silvan Zahno, Axel Amand

Date: 26.11.2024
Version: v2.2



Contents

1 Introduction

The aim of the project is to apply the knowledge acquired at the end of the semester directly using a practical example. The goal is to create a controller which can via a VGA interface display a predefined image. This display system is represented in the illustration Figure 1.

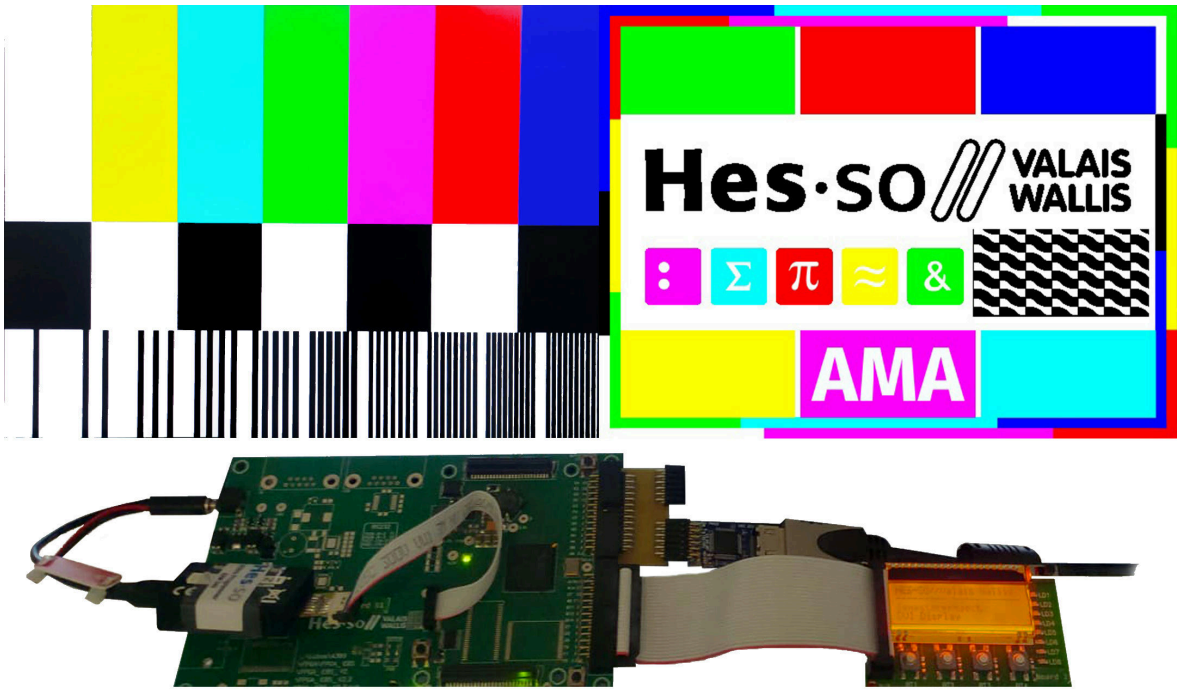
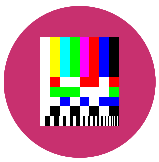


Figure 1: Display setup (EBS3)

The task is to complete the minimal specification 2, which can be optionally extended with additional functions, e.g. switching between a calculated and a pre-recorded image, animating the screen ...

> Additional functions can earn some extra points.



2 Specification

2.1 Functions

The basic functions are defined as follows:

- If the **`start`** button is pressed, a test image is displayed on the monitor.
- If the **`stop`** button is pressed, the test image is removed and the monitor turns black.
- The test image is generated by the circuit *(not pre-recorded)* and consists of all possible color combinations that can be combined with the 3bpp DVI module. The illustration Figure 2 shows a possible test image that can be displayed.
- The resolution must be 640px x 480px @ 60 Hz.

Figure 2: Possible test image showing all color combinations

2.2 Circuit

The FPGA development board forms the core of the system. A LED - chapter 3.2 board and a DVI module - chapter 3.3 are connected to it. The screen is connected to the module output via HDMI. The entire system is schematically shown in the illustration Figure 3.

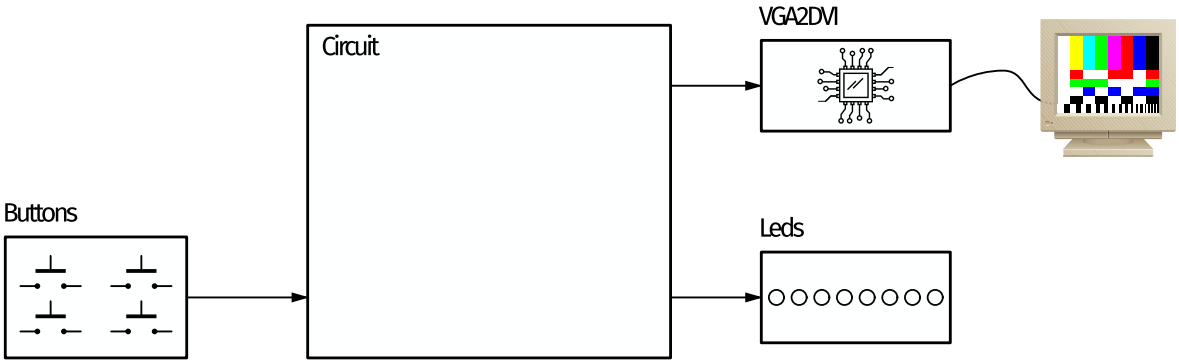
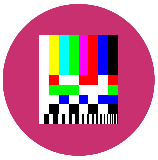


Figure 3: Display circuit



The complete circuit works as follows:

- Four buttons are used to control the system: **`start`**, **`stop`** as well as two freely available buttons **`button_3`** and **`button_4`**. These can be used for optional functions.
- By pressing the **`start`** button, an image, calculated according to the pixel position (e.g. color bars), is continuously transmitted to the PMod module via the VGA interface.
- By pressing the **`stop`** button, a black image is transmitted.
- The PMod module has configuration pins as well as the previously mentioned VGA video interface to receive the image data. The following signals are used for this purpose: **`vga_dataEnable`**, **`vga_pixelClock`**, **`vga_hsync`**, **`vga_vsync`** as well as **`vga_rgb[2:0]`**.
- The module converts VGA video signals to TMDS signals and sends them to the monitor via the HDMI interface.
- The **`testOut`** pins can be used to provide additional information about the system, for example for debugging purposes or to control LED.

The empty TopLevel circuit shows all signals connected to the FPGA board, see Figure 4:

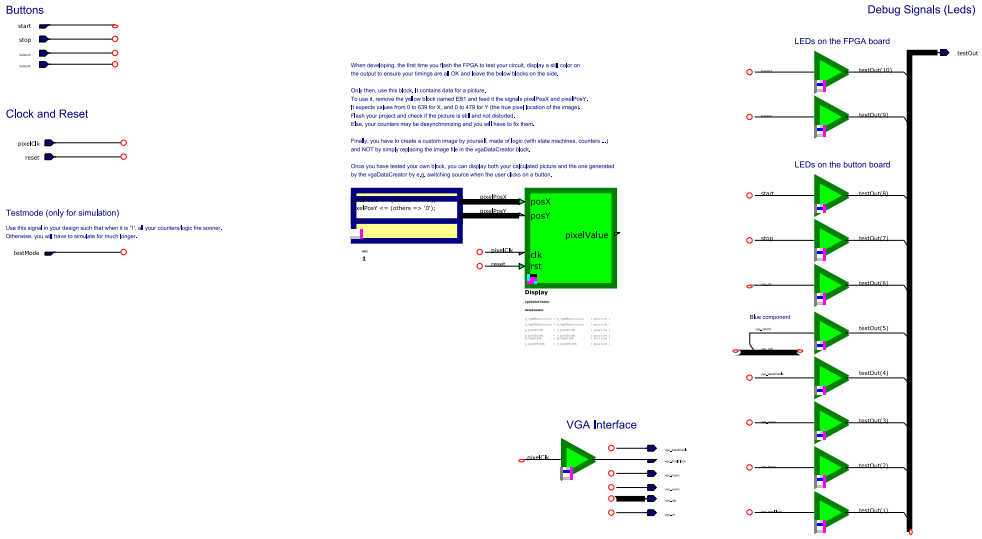
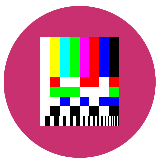


Figure 4: Empty Toplevel circuit

The **testOut** signal allows to turn on and off the LEDs present on the Section 3.2.

The already existing **vgaDataCreator** block provides a pixel color according to the given X and Y positions (from 0 to 639 for X, 0 to 479 for Y) to display an image. It allows to roughly detect a synchronization error (i.e. the image looks distorted, wavy, cut off ...). **This block is provided for testing purposes and does not replace the primary request to display a custom image.**



2.3 VGA Timing (example)

Figure 5 and Figure 6 show the VGA timing for the resolution 640px x 480px @ 60Hz.

The image is built line by line starting from the upper left corner. The **`vga_rgb`** signal contains the data of a pixel. A new pixel can be transmitted at each cycle of **`vga_pixelClock`**. **`vga_int`** can be activated with RGB signals, allowing to display brighter colors. **`vga_dataEnable`** indicates if the data signal is active and can be read. The remaining two signals **`vga_hsync`** and **`vga_vsync`** indicate if a new line (**`hsync`**) or a new page (**`vsync`**) starts.

The timing conditions for the signals **`vga_hsync`** and **`vga_vsync`** must be perfectly understood and respected. More information can be found in the VESA specification [1] as well as on the TinyVGA website [2].

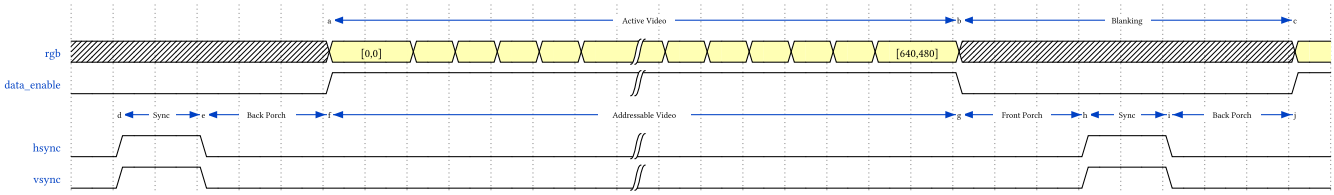


Figure 5: Temporal sequence of the VGA signals (**`hsync`** and **`vsync`** mixed)

Outside the **Active Pixels** area, the RGB value must always be 0.

For an image of 640px x 480px, 800px x 525px are theoretically transmitted, the additional pixels being necessary for the vertical and horizontal **`front porch`** and **`back porch`**. These are intended to give an old CRT monitor enough time for the electron beam to reposition itself between lines and pages.

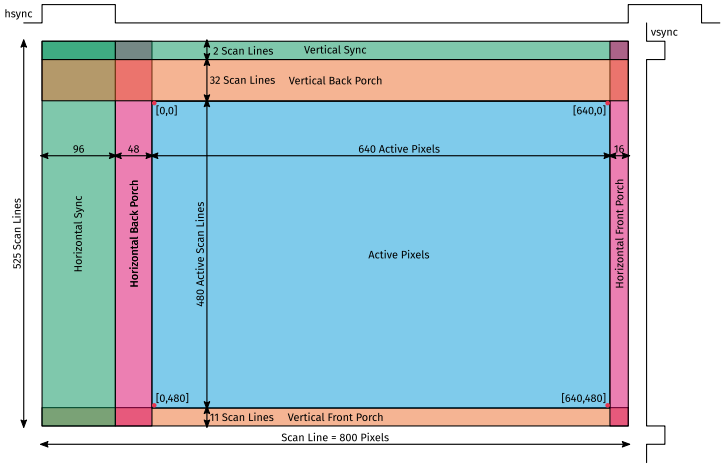
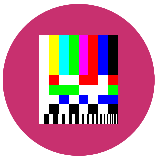


Figure 6: VGA Video Decoding

The timings of the signals **`vga_hsync`** and **`vga_vsync`** are precisely predefined.



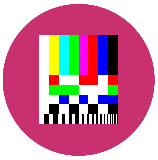
An example for the resolution 1920px x 1440px @ 60Hz is given in the listing Table 1:

```
Name                 1920x1440 @ 60Hz
Aspect Ratio         4:3

Pixel Clock          234     MHz
Pixel Time             4.27  ns
Horizontal freq.      90     kHz
Line Time             11.11  µs
Vertical freq.        60     Hz
Frame Time            16.66  ms


Horizontal Timings

Visible Area         1920
Front Porch           128
Sync Width            208
Back Porch            344
Total (blanks)        672
Total (all)          2600
Sync Polarity         neg


Vertical Timings

Visible Area         1440
Front Porch             1
Sync Width              3
Back Porch             56
Total (blanks)         60
Total (all)          1500
Sync Polarity         pos


Active Pixels
Visible Area   2,764,800
```

Table 1: VGA configuration for 1920px x 1440px @ 60Hz

The above figures are examples. The student must understand and adapt them to the required resolution.

.



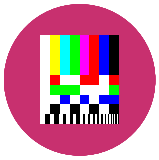
2.4 HDL-Designer Project

A predefined HDL-Designer project can be downloaded from Cyberlearn or Git. The file structure of the project is as follows:

```
did_display
+--Board/           # Project and files for programming the FPGA
|   +--concat/      # Complete VHDL file including PIN-UCF file
|   +--ise/         # Xilinx ISE project
+--Display/         # Library for the components of the student solution
+--Display_test/    # Library for the simulation testbenches
+--doc/             # Folder with additional documents relevant to the project
|   +--Board/       # All schematics of the hardware boards
|   +--Components/  # All data sheets of hardware components
+--img/             # Pictures
+--Libs/            # External libraries which can be used e.g. gates, io, sequential
+--Prefs/           # HDL-Designer settings
+--Scripts/         # HDL-Designer scripts
+--Simulation/      # Modelsim simulation files
+--Tools/           # Specific tools, like a picture to BRAM translator
```

The project folder path must not contain spaces.

The project folder **`doc/`** contains many valuable information. Datasheets, project evaluation as well as help documents for HDL-Designer to name just a few.



3 Components

The system consists of three different hardware boards, visible in the figure Figure 1.

- A FPGA development board, see figure Figure 7.
- A control board with 4 buttons and 8 LED, see figure Figure 8.
- A PMod to DVI module for displaying the image on a screen via HDMI, see figure Figure 9.

3.1 FPGA -Board

On the EBS3 board, the oscillator produces a clock signal (**`clock`**) with a frequency of $f_{\text{clk}} = 100$ MHz. This clock is named **`lcdClock`** and is **only** intended for the dedicated LCD block of the Buttons - LCD, Chapter 3.2 board *(currently unused)*.

Also, this clock is reduced by PLL to $f_{\text{clk}} = 25$ MHz, named **`pixelClk`**. This is the one that should be used for the development of your circuit.

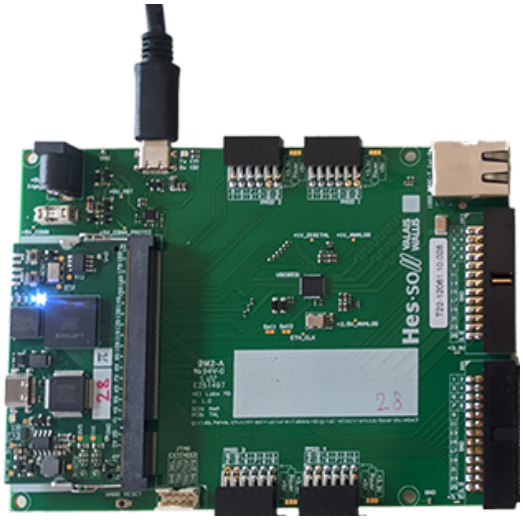


Figure 7: EBS3 FPGA Board [3]

3.2 Buttons and LED

The button and LED board [4] is connected to the FPGA board. It has 4 buttons and 8 LED that can be used in the design, as well as an LCD display [5], [6].

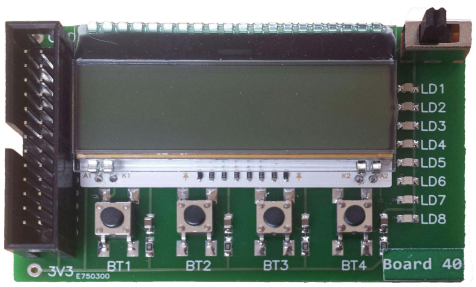


Figure 8: Buttons-LED - LCD board [4]

3.3 PMod - DVI module

The PMod VGA to DVI module converts the VGA signals to TMDS signals. This allows to connect a HDMI monitor to the system.

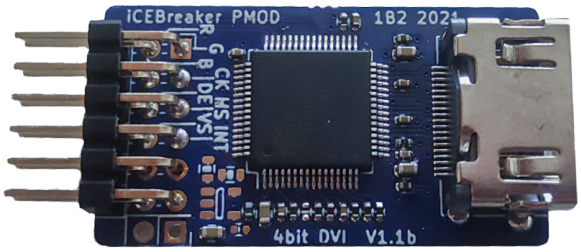


Figure 9: PMod - DVI module

The block diagram of the Texas Instrument TFP410 chip [7] can be found in the diagram Figure 10.

Study the datasheet [7] as well as the schema of the PMod module [8].

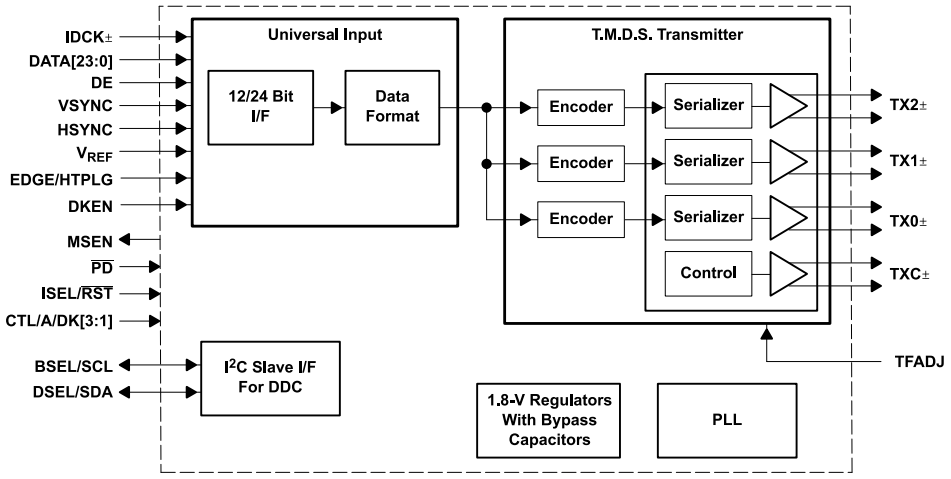
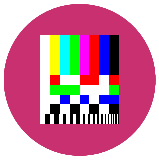


Figure 10: Functional diagram of the PMod TI TFP410 chip [7]



4 Evaluation

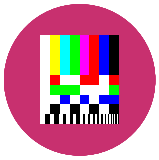
In the **doc/** folder, the file **evaluation-bewertung-display.pdf** shows the detailed evaluation scheme from Table 2.

The final grade includes the report, the code as well as a presentation of the system.

Evaluated aspects	Points
Report	**55**
Introduction	3
Specification	5
Project	20
Verification and validation	10
Integration	9
Conclusion	3
Formal aspects of the report	5
Functionality of the circuit	**30**
Quality of the solution	**10**
Presentation	**10**
Total	**105**

Table 2: Evaluation grid

The evaluation grid already gives indications about the structure of the report. For a good report, consult the document **How to write a project report?** [9].



5 First steps

To start the project on the right foot:

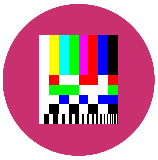
- Read the specifications and information presented carefully.
- Browse the documents in the **`doc/`** folder of your project.
- Analyze in detail the blocks that already exist.
- Develop a detailed functional diagram. You must be able to explain the signals and their functions.
- Implement and simulate the different blocks.
- Confirm the operation of the given hardware with the pre-installed program.
- Test the solution on the FPGA and find any errors

5.1 Procedure

In order to minimize the number of bugs occurring at the same time, it is recommended to proceed as follows:

1. Develop a block diagram taking into account all your inputs, outputs and features you want to implement, without implementing said blocks (leave the signals at **`'0'`**). Read the following points before starting.
2. Implement the strict necessary to display a single color on the screen, without taking into account the buttons and/or the position of the displayed pixel. Remember, however, that the color should only be active in the **Active Pixels** area, otherwise left black (**`"000"`**). *Leave the **vgaDataCreator** and **eb1** blocks on the side.*
   - This allows to check the timings of the signals **vga_pixelClock**, **vga_hsync**, **vga_vsync** and **vga_dataEnable**.
3. Now use the **vgaDataCreator** block to display a pre-recorded image on the screen. To do this, remove the yellow **eb1** block *(the latter holds the **pixelPosX** and **pixelPosY** signals at 0)*. Feed it with the pixel position on the image at the right time, knowing that it needs a clock pulse to output the color data of the pointed pixel.
   - **pixelPosX** is an **unsigned** of **10 bits** accepting input values from 0 to 639.
   - **pixelPosY** is an **unsigned** of **9 bits** accepting input values from 0 to 479.
   - Displaying an image will help to detect desynchronization errors, drift, accuracy of position counters … depending on whether the latter is distorted, wavy, cut, mobile …
4. Then implement a block that calculates an image on the fly according to the pixel position on the screen. The image should contain all possible color combinations that can be displayed, see Figure 2.
5. Finally, add the buttons and associated features. At a minimum, it should be possible to stop the display (black image) and restart it (test image).

At any time, it is possible to display certain signals on the LED to facilitate debugging. To do this, use the **`testOut`** signal linked to the LED module. It is possible to measure these with oscilloscopes or logic analyzers to help with error detection.

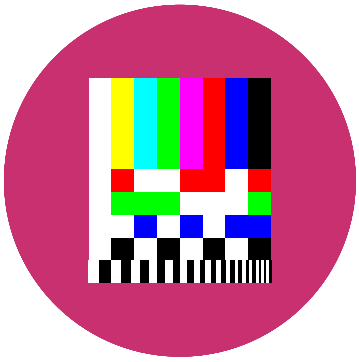


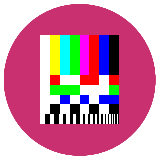
5.2 Tips

Here are some additional tips to avoid problems and time loss:

- Divide the problem into different blocks: use the empty Toplevel document. It is recommended to have a balanced mix between the number of components and the size/complexity of the components.
- Analyze the different input and output signals, their types, their sizes ... It is advisable to use the data sheets.
- Respect the DiD chapter “Methodology for the development of digital circuits (MET)” when creating the system. [10].
- Respect the proposed incremental procedure. Abuse tests as soon as possible.
- Save and document your intermediate steps. Architectures that have not worked, basic codes to test the architecture ... are all material that can be added to the report.

Don’t forget to have fun.





Bibliography

[1] VESA, "VESA and Industry Standards and Guidelines for Computer Display Monitor Timing (DMT)." [Online]. Available: https://glenwing.github.io/docs/VESA-DMT-1.13.pdf

[2] TinyVGA, "VGA Signal Timing." Accessed: Jul. 07, 2022. [Online]. Available: http://www.tinyvga.com/vga-timing

[3] A. Amand and S. Zahno, "FPGA-EBS3 Electronic Technical Documentation." 2022.

[4] Silvan Zahno, "Schematic: Parallelport HEB LCD V2." 2014.

[5] Sitronix, "Datasheet Sitronix ST7565R 65x1232 Dot Matrix LCD Controller/Driver." 2006.

[6] Electronic Assembly, "Datasheet: DOGM Graphics Series 132x32 Dots." 2005.

[7] T. Instrument, "Datasheet Digital Transmitter Texas Insturment TFP410." 2014.

[8] E. Piotr, "Schematic: iCEBreaker PMOD 4bit DVI." 2018.

[9] Christophe Bianchi, François Corthay, and Silvan Zahno, "Comment Rédiger Un Rapport de Projet?." 2021.

[10] François Corthay, Silvan Zahno, and Christophe Bianchi, "Méthodologie de Conception de Cicuits Numériques." 2021.