

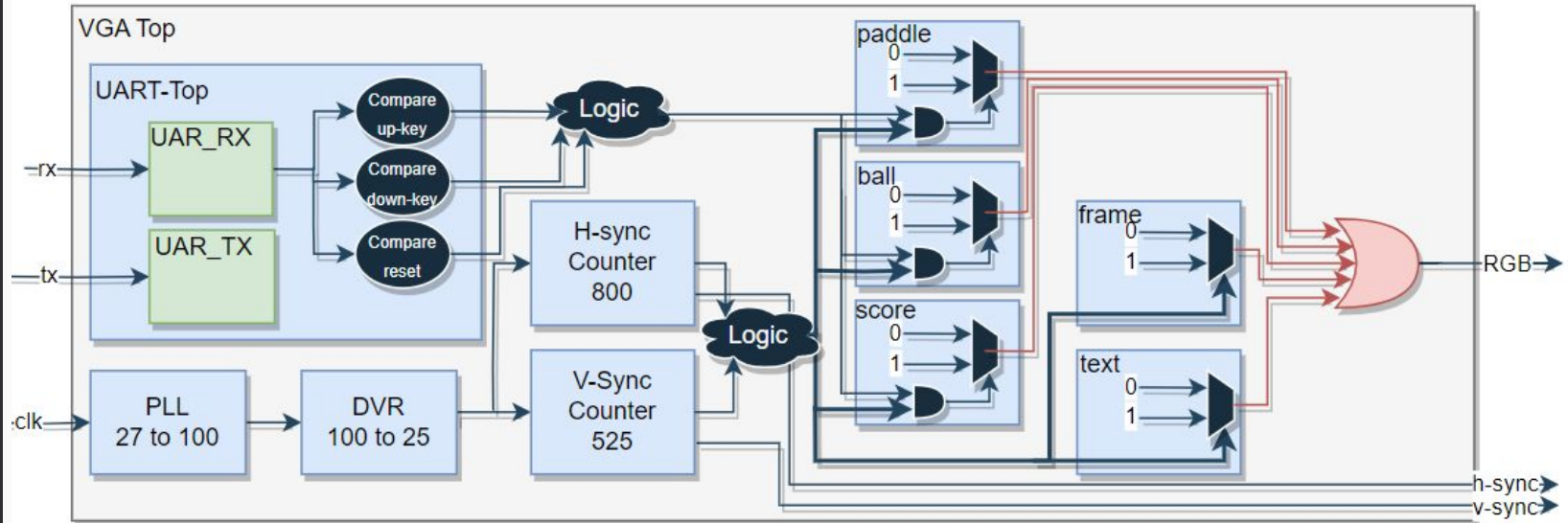
# **Integrate Keyboard With VGA for PONG Game**

## Workshop

Lab 06



# System Context





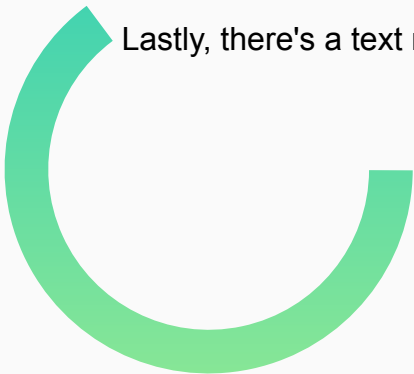
# Module Description

We have a top module where input and output ports are defined, known as the VGA top module. Following that, there's a second module, UART top, which consists of two modules within it. One module is for transmitting data called `uart_tx`, and the other is for receiving data called `uart_rx`.

Apart from these, there's a module that will print based on the paddle's position. Following that, there's a module for the ball's position printing.

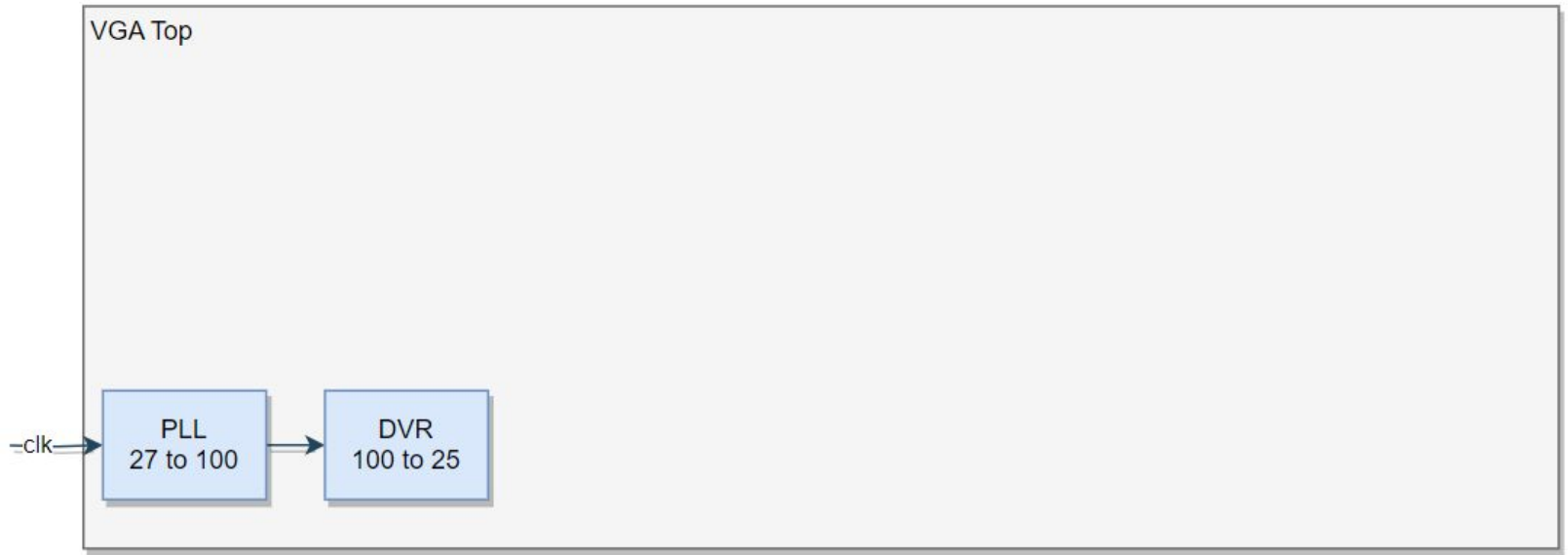
Then, there's a score module that will count scores based on the positions of the paddle and the ball. After that, a frame module will print separate frames for the game and the score.

Lastly, there's a text module that will print 'P1' and 'P2' at the top of the screen.



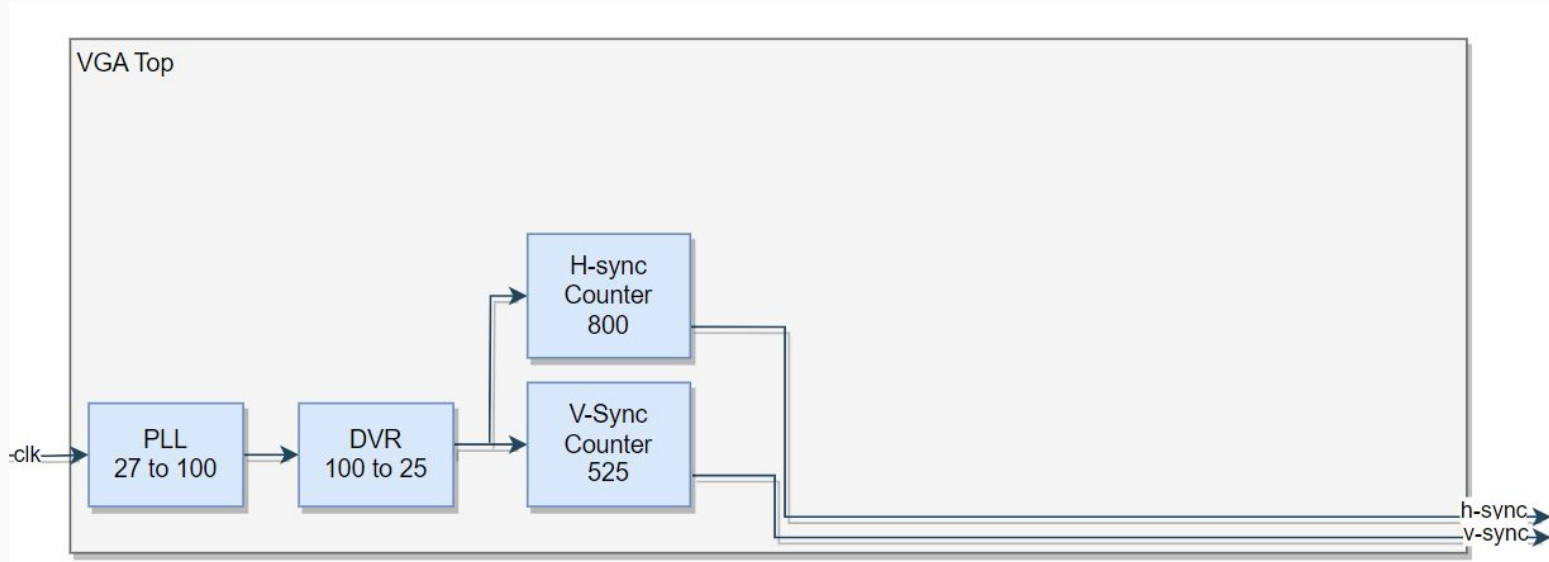
# Modular Hierarchical Description

We start our modular description with the clock. As the Tang Nano 9k operates at 27 MHz, we will first use a PLL to increase it to 100 MHz and then divide it by 4 to achieve a clock frequency of 25 MHz.



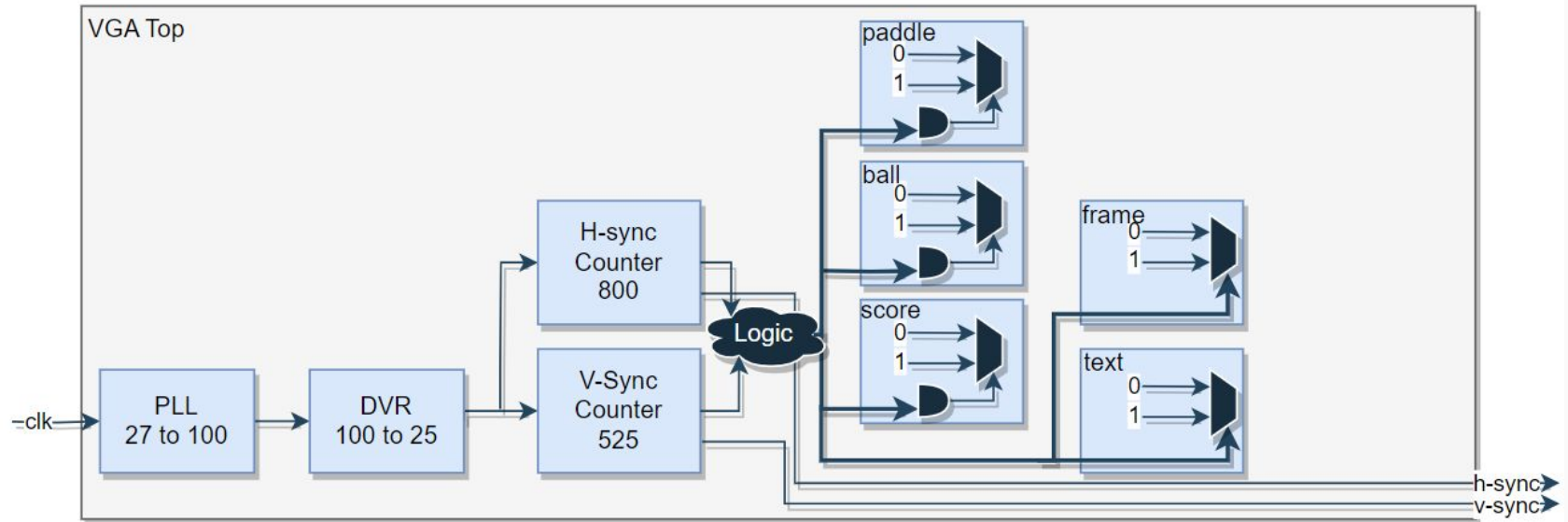
# Modular Hierarchical Description

Now, the 25MHz clock will be fed into the horizontal and vertical counters. The counters will operate at 25MHz, and the logic of the counter will drive the horizontal sync (h-sync) and vertical sync (v-sync).



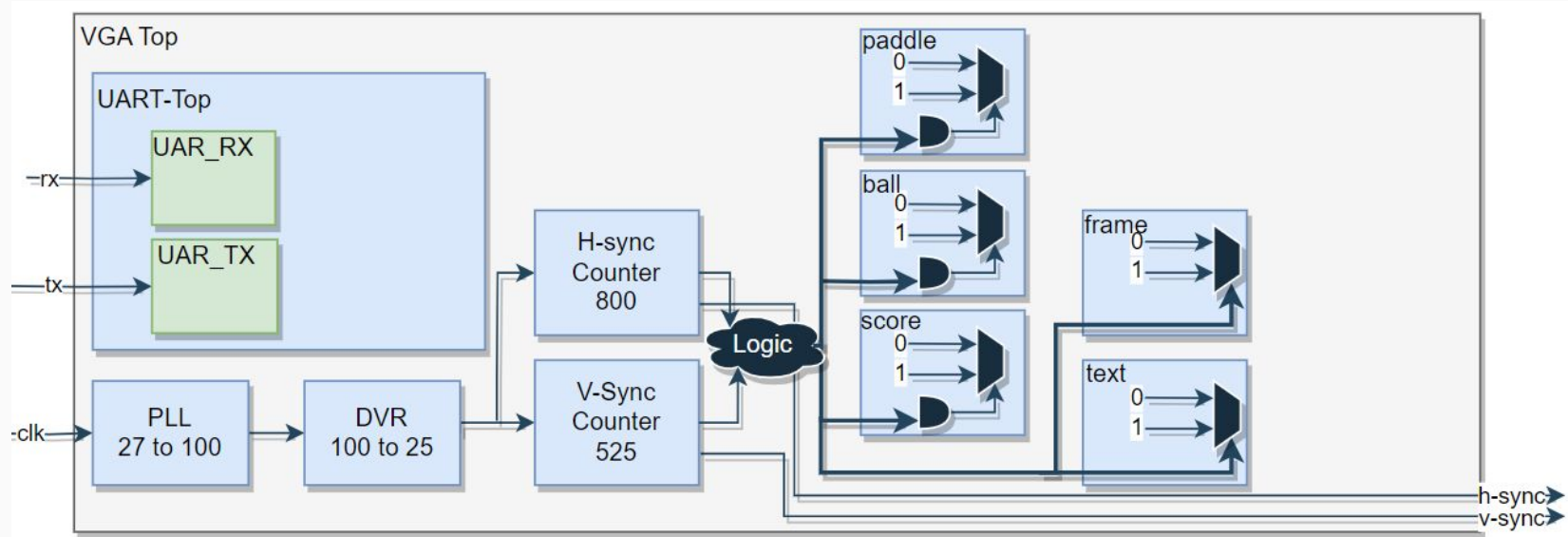
# Modular Hierarchical Description

Now, we can extract the count value from the counter. Let's establish a limit for that counter and perform multiplexing. If the counter falls within that range, print 1; otherwise, print 0. For instance, in the area where we want to print the paddle, if the counter falls within that specific range, print 1; otherwise, print 0. Similarly, for displaying the ball, text, score, or frame, we can utilize this method. The advantage here is that for a particular game, we don't need to allocate memory or store pixels.



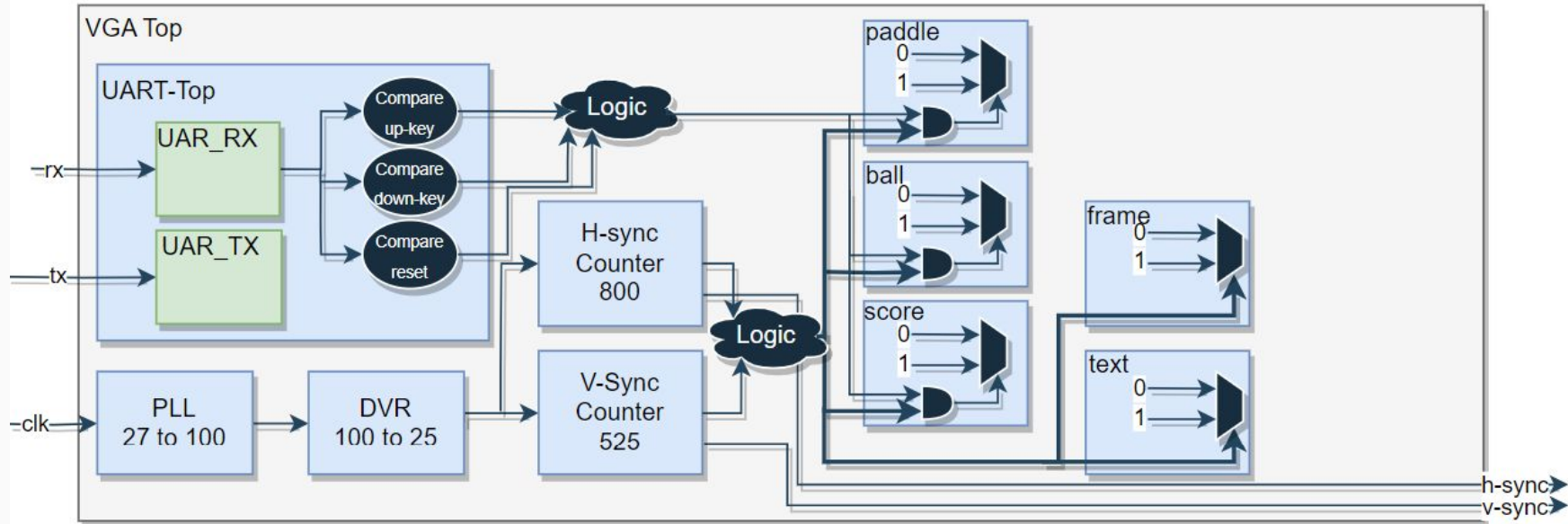
# Modular Hierarchical Description

Now, we also need to control the location of paddle using the keyboard. For instance, we will receive values through `uart_rx` from the keyboard and control the location accordingly. For that reason we introduce UAR-Top. And inside it UAR\_RX to receive data.



# Modular Hierarchical Description

Now, we need to incorporate comparison logic within UART top. When any data is received from UART, we'll compare it to see which ASCII character it matches. For example, if we consider 'p' as an up key for a player like P1 or P2, when 'p' is received via UART, we'll assign it to control the paddle input. This logic will be configured to operate within the counter's range, so when an up-key press occurs, the paddle will print in a different range of the counter, creating a scaled movement.

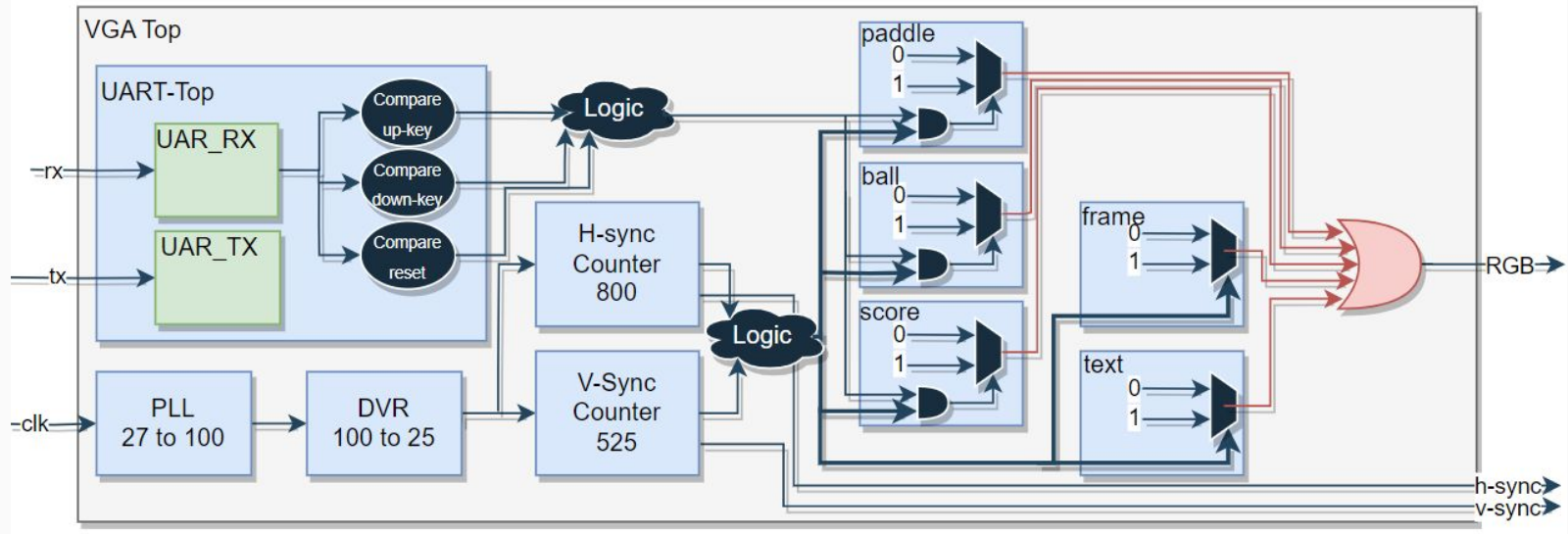




# Modular Hierarchical Description

Now, we'll combine the outputs of all modules where we need to print pixels. We'll map the output of these modules together using an OR operation onto the RGB pins.


Reference Code: [Github Abdul Muheet Ghani](#)



# Task



Now, after completing the game, the task is that when a key is pressed on the keyboard, its ASCII value should be mapped onto the LED of the Tang Nano 9k using the UART\_TX module.



# Testimonial

## Author:

Abdul Muheet Ghani, Research Assistant at MERL-UITU.

## Under The Supervision Of:

Dr.Ali Ahmed (Team Lead MERL).

Sajjad Ahmed (Research Associate)

Sponsored By: Edmund from Symbiotic EDA. for sponsoring FPGA.

Thanks: Lushay Lab (They've provided crucial resources and guidance throughout the project), Also thanks to Slidesgo.

# Future Work

This is version 1.0 of our course. We will continue this training and very soon release further versions. For any questions or to stay connected with us.

Github: [Abdul Muheet Ghani](#).

Gmail: [Dr.Ali Ahmed](#). [Sajjad Ahmed](#). [Abdul Muheet Ghani](#).