

# Third assignment

## PWM via USART

### Digital design and verification

Professor: Iván Leonardo Gómez Mendoza

Festivalo: Matías Andrés Gómez

Overflow: We are tasked to design and verify a digital PWM generator controlled via USART interface (only simulating on this case)

Objectives: The resulting system allows the duty cycle and frequency of a PWM signal to be adjust over a serial link, providing immediate feedback to the user through status messages.

in addition to design details, the report introduces the theory behind USART protocol communication, pulse-width modulation and overflow scenarios.

# Theory background and modular design:

PWM : frequency generator and duty

Divide the 50MHz reference by  $\text{POW2/5}$  to obtain PWM base frequency.

System requirements:

Frequency:

- clock frequency = 50 MHz
- Maximum PWM frequency = 50 kHz.

Let's define the  $f_{\text{pwm}}$ :

$$f_{\text{pwm}} = \frac{M f_{\text{pwm}}}{2^{\text{POW2}} \cdot 5^{\text{POW5}}} \quad \text{j where: } M f_{\text{pwm}} = 50 \text{ kHz}$$

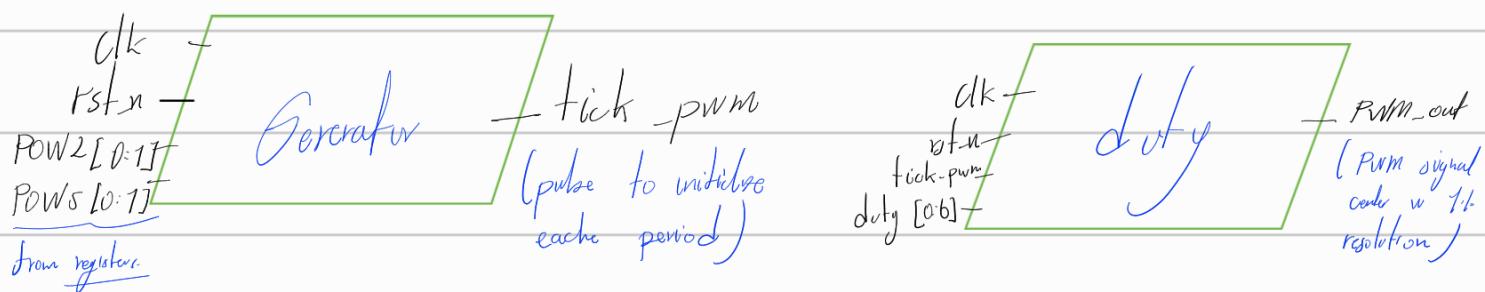
$\sim \text{POW2}, \text{POW5} \in \{0, 1, 2, 3\}$  j minimum freq = 50Hz

Duty : Generate PWM signal center with 1.1. duty resolution

System requirements:

- Duty steps of 1.1. from 0 to 99.1.
- Values  $\geq 100$  are not valid and can't modify register.
- Centry pulses.

i/o:



# VART : RX/TX

Send / receive serial bytes over 115.200 bauds from extern port.

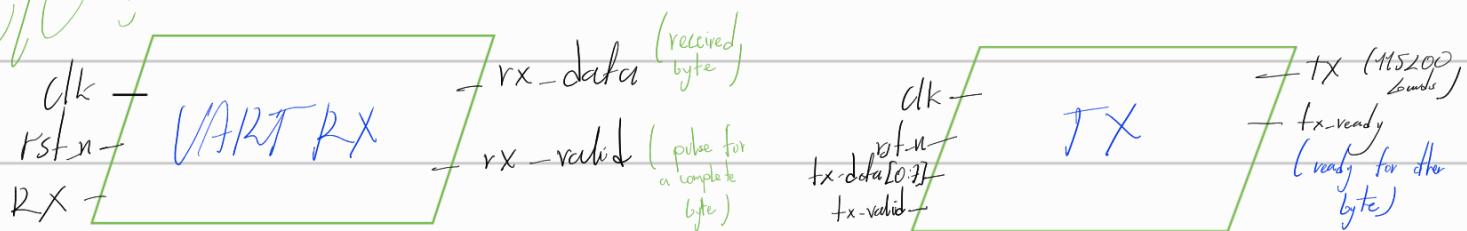
System requirements : - 50 MHz

- 115.200 Bauds, 8 data bits, 1 start, no parity.
- At least one rest bit between bytes.

$$\text{Time constant: } \text{BaudDiv} = \frac{50.000.000}{115.200} \approx 434$$

- Also send output from user requests (OK, FAIL)

I/O :



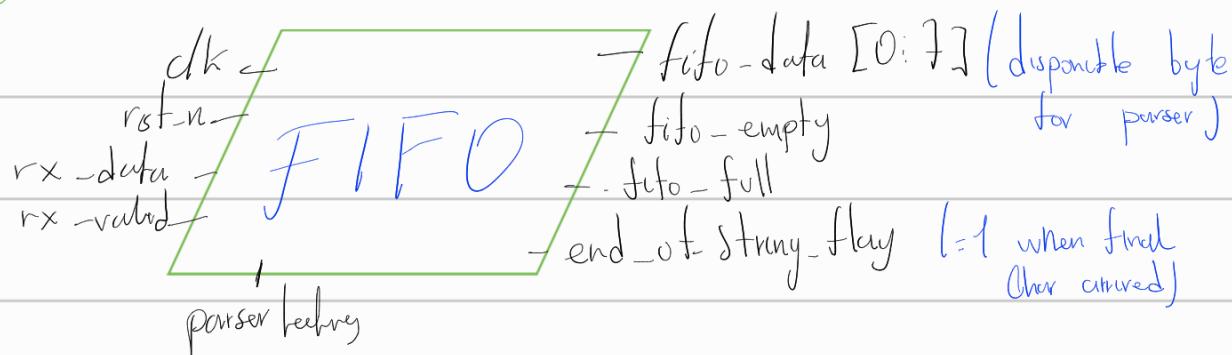
## FIFO RX (32 bytes buffer)

- Stores up to 32 bytes and signals when string end arrives.

System requirements :

- 32 bytes large
- Print "start" when is full.
- Reset pointer when a finish. character arrives.
- Flags management.

I/O :



# Command parser & HTML (FSM)

Read FIFO commands, decode partitions, update registers and generate outputs.

## System requirements.

- Manage status via UART

- Commands :

- "HELP" → deploy a help interface.

- "STATUS" → Send actual duty and PWM frequency.

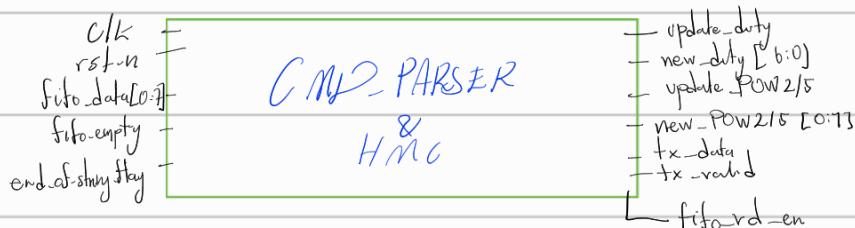
- "DC##" → duty change: Send "OK" or "FAIL" if duty > 98.

- "POW2"/"POW5" → change downscaler

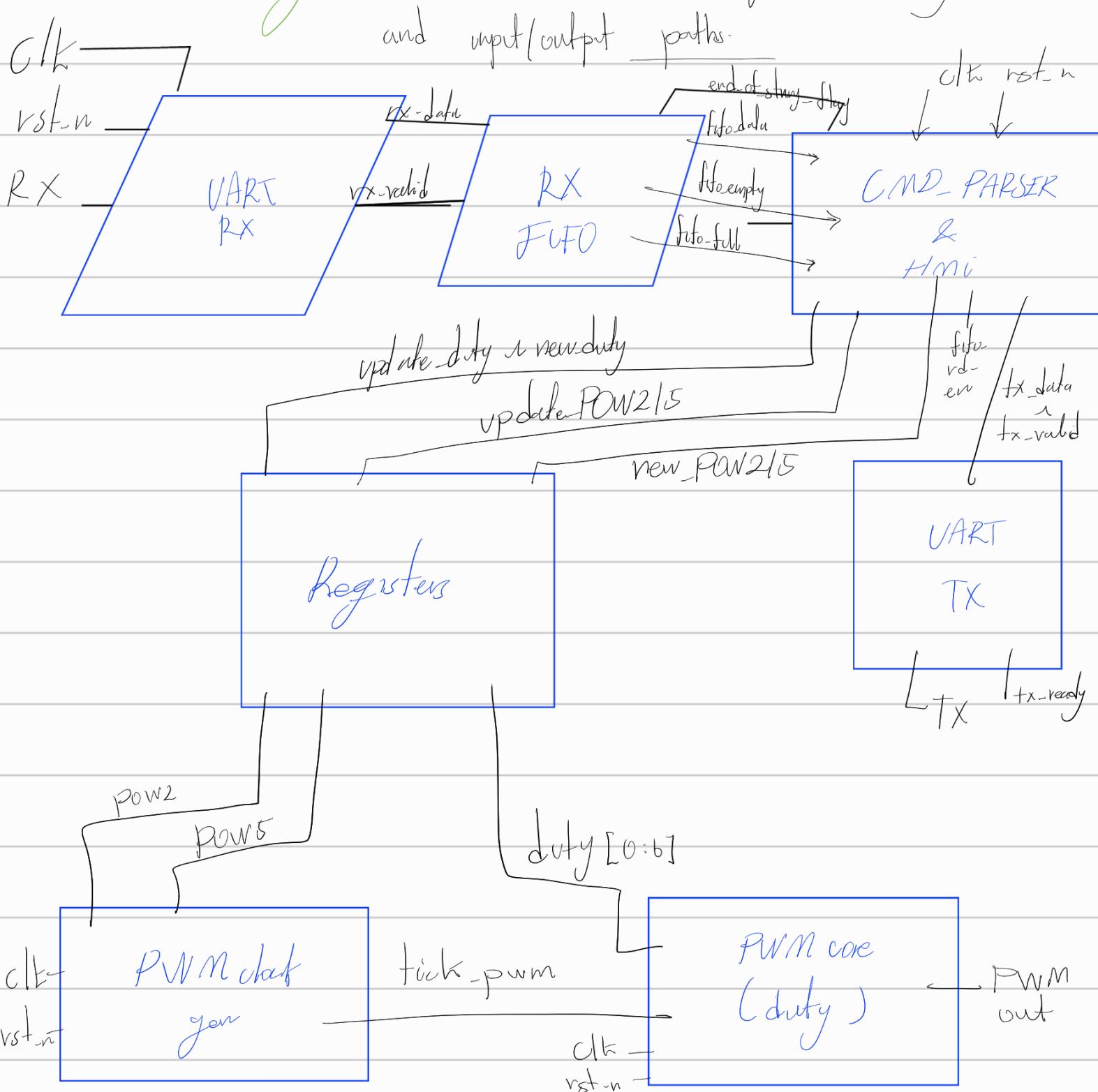
j accept values between 0-3, OK ≠ FAIL.

Every unknown command → "FAIL"

C/I/O



# Block Diagram in order to summarize system functionality



- This port description completes the functional details of the diagram shown, thus links each signal to the respective module that uses or produces it

Verification : To verify functionality w/ a physical FPGA, a systemVerilog Testbench is built around the DUT (device under test)

↳ The tb provide stimulus, capture responses and check correctness.

The simulation uses bus functional models (BFMs) for UART transmission and reception.

- ; UART TX : stimulus via send uart byte.
- ; UART RX : As monitor from the DUT

Test cases:  
i) Nominal commands

ii) Boundary cases

iii) Invalid inputs

iv) Overflow

v) Randomized sequences.

## Conclusions:

- Although the primary goal is simulation, the design can and should be synthesised onto a FPGA. It should respect synchronous design good practices as:
  - All flip flops triggered by the same clock.
  - Asynchronous inputs synchronised.
  - Resets handled cleanly
- This report has detailed the design of a PWM generator controlled through a UART interface, covering functional requirements, module architecture, theoretical foundations and verification methodology
- Via simulation (GTKwave software) using UART protocol we demonstrate that the system provides flexible control of duty cycle and frequency via text commands and return clean responses.