Third assignment

PWM via UART

Digital design and verification

Professor: Ivan Bernardo Gómez Mendoza

Estudiante: Matías Amador Gómez

Overflow: We are tasked to design and verificate a digital PWM generator controlled via UART 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 UART protocol communication, pulse-width modulation and overflow scenarios.

Theory background and module design:

PWM: frequency generator and duty

Divide the 50MHz refrence by POW2|5 to obtain PWM base frequency.

System requirements:

Frequency:

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

Lets define the fpwm:

$$fpwm = \frac{Mfpwm}{2^{POW2} \cdot 5^{POW5}}$$; where: $Mfpwm = 50kHz$

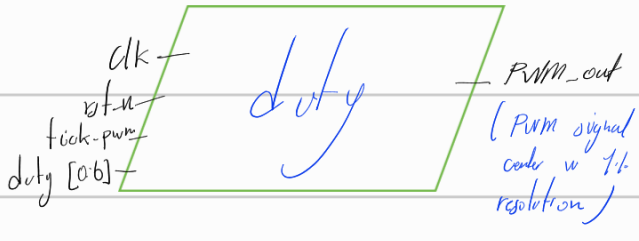
~ $POW_2, POW_5 \in \{0, 1, 2, 3\}$; Minimum freq = 50Hz

Duty: Generate PWM signal center with 1% duty resolution

System requirements:

- Duty steps of 1% from 0 to 99%
- Values ≥ 100 are not valid and can't modify register.
- Centry pulses.

I/O:



UART : RX / TX

Receive serial bytes at 115.200 bauds from extern port.

Send / receive UART commands in order to monitor system status.

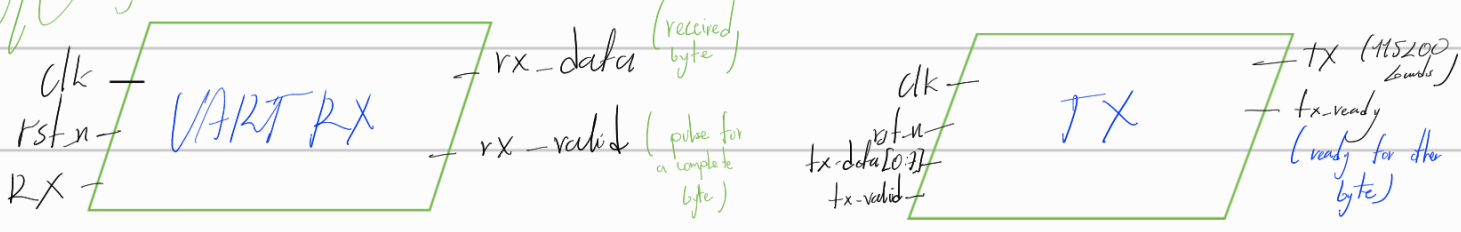
System requirements : - 50 MHz

- 115.200 Bauds/s, 8 data bits, 1 start, non parity.
- At least one rest bit between bytes.

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

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

I/O :



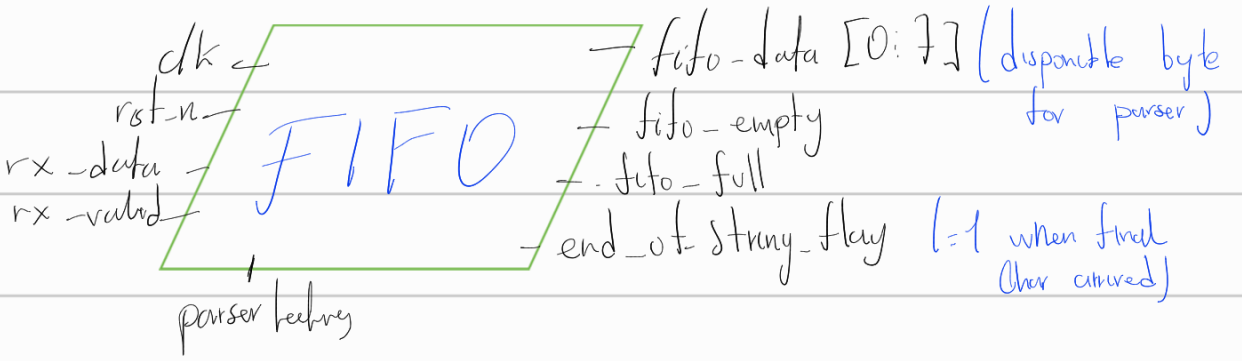
FIFO RX (32 bytes buffer)

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

System requirements :

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

I/O :



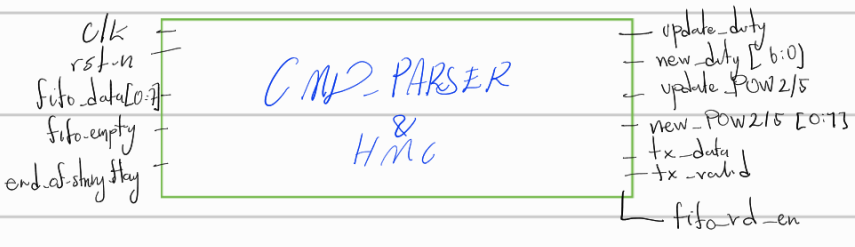
Command parser ~ HMI (FSM)

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

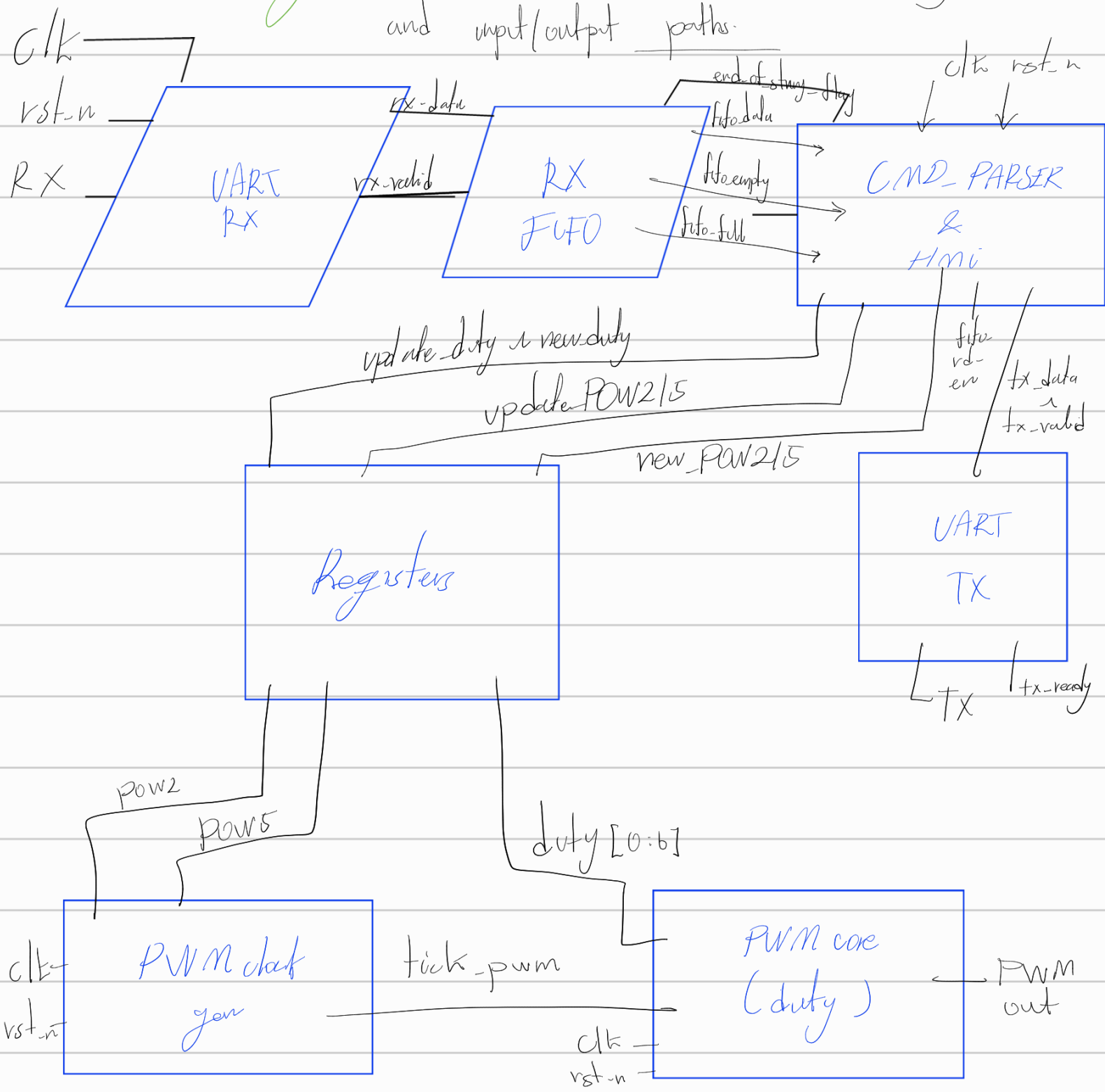
System requirements.

- 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 ; accept values between 0-3, OK ≠ FAIL.
    - ↳ Every unknown command → "FAIL"

I/O



Block diagram in order to sumarize system functionality and input/output paths.



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

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

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

. 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: c) UART:

→ UART RX

on simulation can see one-cycle pulse at each start bit, also, one-cycle pulse once per byte besides. rx_byte shows 0x41 then 0x7A, proving uart rx functionality

=> UART LOOPBACK:

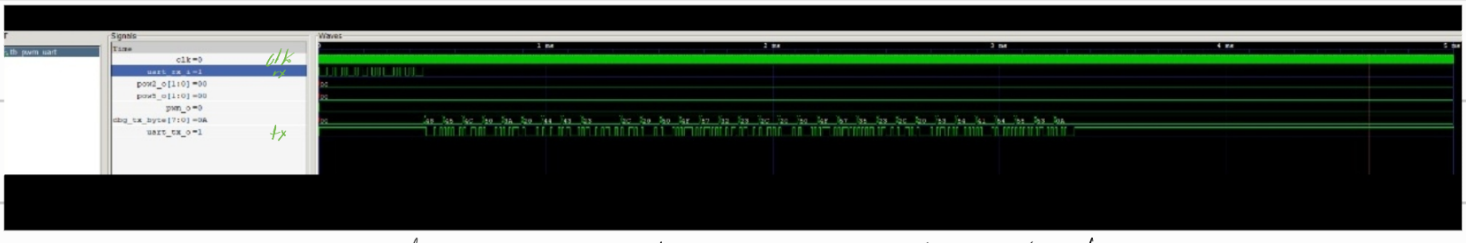
On waves, can see that the transmitter generates correct UART frames (start, 8 data bits, stop) with enable timing and no interruptions.

At the same time, the receiver succesfully decodes both transmitted bytes (0x41 & 0x7A) and asserts rx_valid at the expected moments.

Continue with the top module, which one implements all modules (uart rx, tx, pwm and cmd parser) on one.

Given the rx, as "HELP", we expect on the output uart tx this follow commands:

- DC ## POW2 ## POW5 # STATUS \n



The current top simulation shows the full HELP string on dbg_tx_byte, and well-formed UART frames on uart_tx_0, so the existing system is functioning.

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 freqency via text commands and return clear responses.