**SIMULATION OF PROCESSING ELEMENT DOCUMENT OF TAPEOUT 4**

**Prepared by**

**KLOK SYSTEMS**

**Prepared for**

**NATIONAL UNIVERSITY OF SINGAPORE**

@ 2022 Klok Systems India Private Limited or its affiliates. All rights reserved. This document, in part or in full, cannot be shared without written permission from Klok Systems India Private Limited.

Table of Contents

# CONTENTS

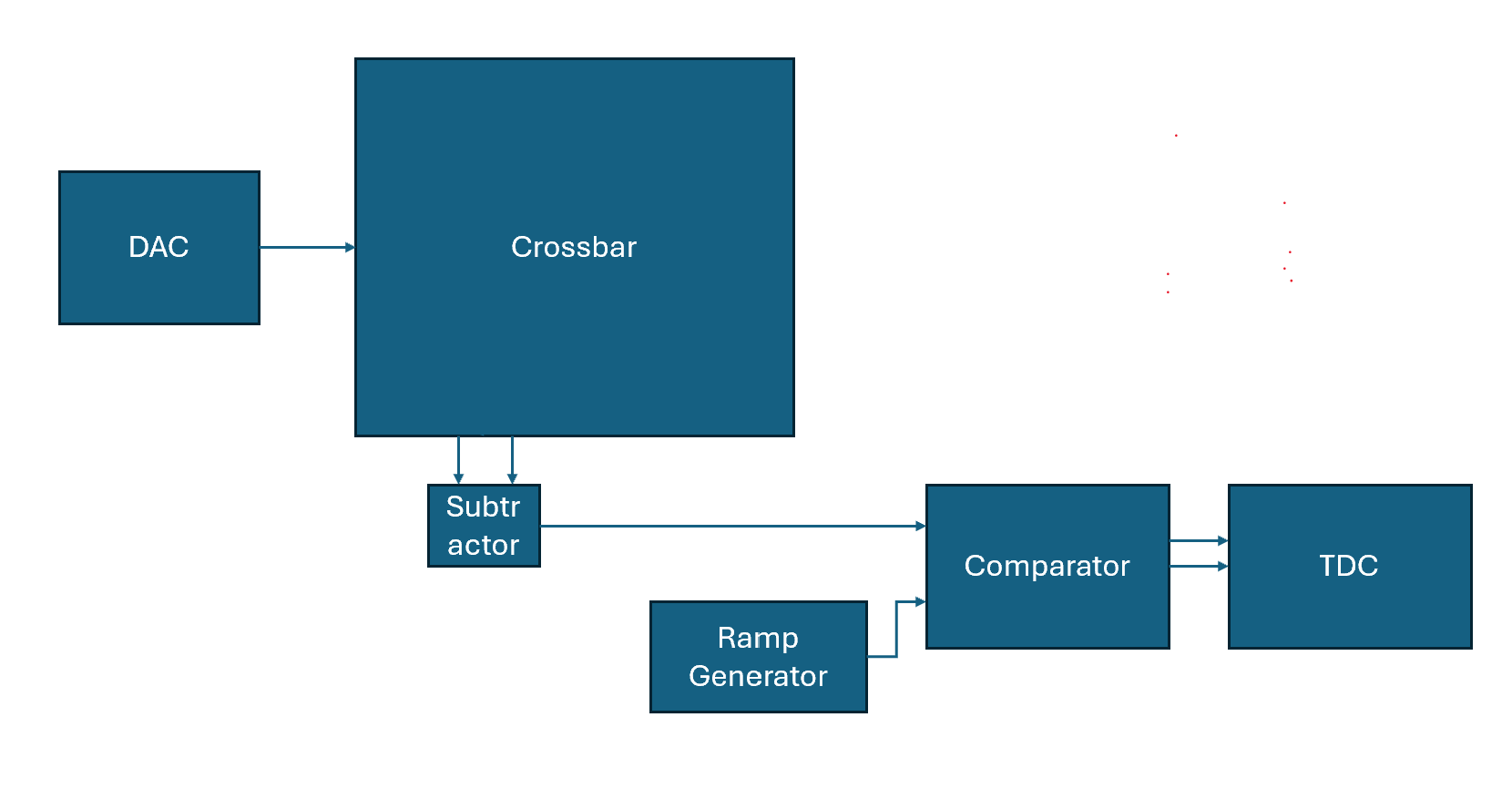
1. PWM
   1. PWM BLOCK DIAGRAM………………………………………………………4
   2. PWM ARCHITECTURE…………………………………………………………4

List of Figures

[Figure 1: Block diagram of pwm……………………………………………………………4](file:///D:\NUS\NUS_DOC\nus\design_document_tdc.docx#_Toc183715998)

[Figure 2: Pwm architecture………………………………………………………………….6](file:///D:\NUS\NUS_DOC\nus\design_document_tdc.docx#_Toc183715999)

## PWM



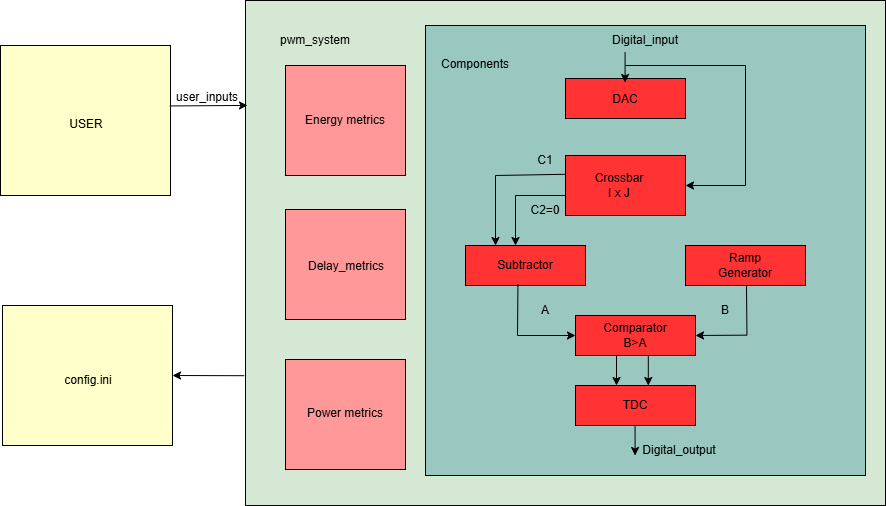
***Figure 1: Block diagram of pwm***

The block diagram of pwm consists of

* **DAC** which takes digital input passes the analog output to the crossbar
* **Crossbar (processing element)** which is built in the form of a matrix with rows and columns and the adjacent column results are provided to the subtractor.
* **Subtractor** takes the difference from the adjacent columns and passes the subtracted value to the comparator.
* **Ramp generator generates** the ramp values when enabled and start signal goes high
* **Comparator** Compare both input values such that when the ramp value exceeds the subtracted value the stop signal goes high.
* **TDC** takes two inputs basically start and stop time and measures time difference between them and provides the digital output.
* Python was used to design the pwm\_system

## IMPLEMENTED ARCHITECTURE

The implemented architecture which is completely structured in python which consists of user, config, energy metrics, delay metrics, components of pwm which is depicted in the below figure shown:



***Figure 2: Pwm Architecture***

1. The **user** file allows the user to send the digital inputs and select the number of rows in order to pass the inputs to the selected rows.
2. The **config.ini** file is the file which is consists of the parameters that is invoked by the components package.
3. The **Energy metrics** which is initialized in the pwm system calculates overall energy consumed by the components of the pwm.
4. The **Delay metrics** which is initialized in the pwm system calculates overall delay caused by the components of the pwm.
5. The **Power metrics** which is initialized in the pwm system calculates overall delay caused by the components of the pwm.
6. The **Components** which are initialized in the pwm system involves the core functionality of the pwm.

### PWM ARCHITECTURE

1. **Config.ini:** The file consists of the parameters for the components built in the pwm where each component attribute is declared in this configuration file and all these are accessed with help of a standard library in python **configparser** which helps the user to read, write, and modify the configuration file easily.

* **[**SYSTEM]

1. Vdd = 1.8.
2. Time\_step = 100e-12.

* [DAC]

1. Num\_bits = 3.
2. Pulse\_period = 1e-9.

* [CROSSBAR]

1. Vin = 0.3.
2. Roff = 50000.
3. Ron = 5000.
4. On\_off\_ratio = 10.
5. Capacitance = 1e-12.
6. Array\_columns =32.
7. Array\_rows = 32.
8. Weight\_bits = 3.
9. A = 10.

* [RAMPGENERATOR]

1. Slope = 6e5.
2. Enable = True.

* [TDC]

1. Num\_tdc = 10.
2. Time\_precision = 1e-9.
3. **USER:** The user file is a script which is designed to initialize and run the pwmsystem and which uses pathlib library to handle the file paths, NumPy for handling arrays which is used to declare the weights and also prints the results of the pwm system.

* Key features

1. Inputs are user configurable such that allows the user to send specific inputs to the selected rows where the selection of rows are also user configurable.
2. Debug statements are present in case if the user provides inputs, selected rows, weights out of range.
3. Since the array size is 32 \* 32 user can set the weights manually which are hardcoded as a NumPy array pass it to the pwm\_system.
4. Since the pwm\_system is able to access the attributes from the configuration file the user file helps to identify that path using pathlib library.
5. **Pwm\_system:** The pwm is like the heart of the architecture which initializes the components of the pwm, initialize the metrics, process inputs to each component and has the core functionality and the components are able to access the attributes from the configuration file with the help of configparser library.

* Key features

1. it consists of a constructor to load the configurations and initialize the components and metrics.
2. Load config method which initializes the configparser and read the config path.
3. Initialize components method initializes all the components with their attributes.
4. Initialize metrics method initializes all the components with their respective metrics that is energy, delay, power etc.
5. set weights methods is able to access the weights that is set in the user file and pass it to the crossbar.
6. Process inputs method is the core logic that passes input/output to each component where the dac takes the digital input and provides the analog output and currently the crossbar is designed to take digital input provides the output as 32 columns where every odd numbered column is computed to zero and even numbered column set to work based on the formula and 16 subtractors will take the difference between the odd sum and even sum after that ramp generator is enabled to generate the ramp value initially the start time, stop time and current time will be 0 so that for each subtracted value when ramp value exceeds the subtracted value crossing point is found and at that point stop time is recorded hence the tdc is designed to take the difference between start time and stop time and calculate the digital output.
7. The calculate metrics method calculates the energy, delay and power.
8. **DAC:** The dac is designed to take attributes like no of bits, pulse period, Vdd which converts the digital input to analog by multiplying the digital input with pulse period.
9. **Crossbar:** The crossbar is designed to take the attributes like on resistance, off resistance, on\_off\_ratio, Capacitance, Vdd, pulse period, rows, columns, weights, scaling factor A, Vin.

* Key features

1. The compute method based on the input, selected rows corresponding to that column weighted sum is calculated.
2. The column output for even numbered columns is calculated using the formula

Y = (Vin\*pulse period) \* (weighted sum)

(Ron\*Capacitance\*A) (on off ratio)

The attributes values are taken as per the configuration file where even numbered columns take this formula which uses a separate method to sum up the even numbered column and the odd numbered columns doesn’t take this formula currently the outputs are taken as zero.

1. **Subtractor:** The subtractor module is designed to take the difference between the odd sum and even sum generated by the crossbar.
2. **Ramp generator:** The ramp generator is designed to generate the ramp values when enabled which takes the attributes like slope, time step and the product of slope with time step generates the ramp value.
3. **Comparator:** The comparator is used to compare the ramp values and subtracted values and returns ramp value that is greater than the subtractor values.
4. **TDC:** The tdc module is designed to take the attributes like num of tdc bits, time precison and consist of a method to measure time that is difference between start and stop time and return digital value within the range of tdc bits where the digital output will be represented with list of 16 values as there are 16 subtractors.