Federal University of Paraíba
Department of Electrical Engineering
Master in Electrical Engineering

Digitally Driven Control for Inductive Buck Converters in Portable Battery-Powered Applications

Rodrigo Pedroso Mendes

João Pessoa-PB, 29 de julho de 2024

Rodrigo Pedroso Mendes

Digitally Driven Control for Inductive Buck Converters in Portable Battery-Powered Applications

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEE, da Universidade Federal da Paraíba - UFPB, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Antônio Augusto Lisboa de Souza

Coorientador: Cícero da Rocha Souto

João Pessoa-PB

2024

Catalogação na publicação
Seção de Catalogação e Classificação

```
M538d Mendes, Rodrigo Pedroso.
         Digitally driven control for inductive buck
      converters in portable battery-powered applications /
      Rodrigo Pedroso Mendes. - João Pessoa, 2024.
         165 f. : il.

         Orientação: Antônio Augusto Lisboa de Souza.
         Coorientação: Cícero da Rocha Souto.
         Dissertação (Mestrado)  - UFPB/CEAR.

         1. Engenharia elétrica. 2. Conversor CC-CC. 3.
      Controle digital. 4. Indutivo. 5. Abaixador de tensão.
      6. Conversor DPWM. 7. Conversor ADC com janela. I.
      Souza, Antônio Augusto Lisboa de. II. Souto, Cícero da
      Rocha. III. Título.

UFPB/BC                                         CDU 621.3(043)
```

Elaborado por Larissa Silva Oliveira de Mesquita - CRB-15/746

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação

Digitally Driven Control for Inductive Buck Converters in Portable Battery-Powered Applications

Elaborada por

Rodrigo Pedroso Mendes

como requisito parcial para obtenção do grau de

Mestre em Engenharia Elétrica.

Comissão Examinadora:

Antonio Augusto Lisboa de Souza

Antônio Augusto Lisboa de Souza

[illegible]

Cícero da Rocha Souto

[illegible]

Darlan Alexandria Fernandes

GOV.BR
Documento assinado digitalmente
FERNANDO RANGEL DE SOUSA
Data: 02/08/2024 08:36:05-0300
CPF: ***.649.114-**
Verifique as assinaturas em https://v.ufsc.br

Fernando Rangel de Sousa

João Pessoa-PB, 29 de julho de 2024

Contents

List of Figures

List of Tables

Abstract

In this work, a digitally driven control system for an integrated fixed-frequency inductive buck converter is presented. A comprehensive review of small signal modeling and compensation is included, with a digital control based on a standard Proportional Integral Derivative (PID) controller. To enable this control, focus was given on optimizing the Analog to Digital Converter (ADC) and the Digital Pulse Width Modulator (DPWM) for the intended application.

The novel ADC architecture achieves a typical step size of 2.136 mV, a quiescent current of 15.5 µA and an estimated area of 0.0088 mm^2. The proposed DPWM incorporates a feed-forward of the input voltage to boost line regulation and includes a thermometer encoded capacitor array to improve Differential Nonlinearity (DNL). The typical quiescent current of the DPWM is 35.1 µA with an estimated area of 0.0439 mm^2.

Special care was taken to ensure the robustness of the ADC and DPWM against mismatch, process, temperature, and supply voltage variations. The proposed control architecture can be applied to implement DC-DC converters for portable applications, powered either by a Lithium-Ion battery (2.7 V to 4.2 V) or, during the charging of the battery, via a USB port (4.7 V to 5.5 V). The load current may vary from 0 to 200 mA, the switching frequency is 2 MHz, and the output voltage aligns with the range of the native transistors available in the CMOS 180 nm technology adopted (1.62 V to 1.98 V).

Keywords: DC-DC Converter, digital control, inductive, step-down, digital pulse width modulation, window analog to digital converter, buck.

Resumo

Neste trabalho, é apresentado um sistema de controle acionado digitalmente para um conversor indutivo abaixador integrado de frequência fixa. Uma revisão abrangente da modelagem de pequenos sinais e da compensação é incluída, utilizando um controlador Proporcional Integral Derivativo (PID) digital padrão. Para viabilizar esse controle, focou-se na otimização do Conversor Analógico para Digital (ADC) e do Modulador de Largura de Pulso Digital (DPWM) para a aplicação pretendida.

A nova arquitetura do ADC alcança um tamanho de passo típico de 2,136 mV, uma corrente quiescente de 15,5 µA e uma área estimada de 0,0088 mm^2. O DPWM proposto incorpora um mecanismo de avanço da tensão de entrada para melhorar a regulação de linha e inclui uma matriz de capacitores com código termométrico para melhorar a Não Linearidade Diferencial (DNL). A corrente de quiescente típica do DPWM é de 35,1 µA, com uma área estimada de 0,0439 mm^2.

Foi dedicada atenção especial para garantir a robustez do ADC e do DPWM contra variações de descasamento, processo, temperatura e tensão de alimentação. A arquitetura de controle proposta pode ser aplicada na implementação de conversores CC-CC para aplicações portáteis alimentadas por uma bateria de íon de lítio (2,7 V a 4,2 V) ou, durante o carregamento da bateria, via uma porta USB (4,7 V a 5,5 V). A corrente de carga pode variar de 0 a 200 mA, a frequência de comutação é de 2 MHz, e a tensão de saída está alinhada com os transistores nativos disponíveis na tecnologia CMOS de 180 nm que foi adotada (1,62 V a 1,98 V).

Keywords: Conversor CC-CC, controle digital, indutivo, abaixador de tensão, conversor digital para modulação em largura de pulso, conversor analógico digial com janela.

List of abbreviations and acronyms

AC	Alternating Current
ACM	Average Current Mode
ADC	Analog-to-Digital Converter
ALU	Arithmetic Logic Unit
AMS	Analog Mixed Signal
CCM	Continuous Conduction Mode
CMOS	Complementary Metal-Oxide-Semiconductor
DAC	Digital-to-Analog Converter
DC	Direct Current
DCM	Discontinuous Conduction Mode
DPWM	Digital Pulse Width Modulation
DNL	Differential Nonlinearity
DRC	Design Rule Check
DVFS	Dynamic Voltage and Frequency Scaling
ESR	Equivalent Series Resistance
GM	Gain Margin
HDL	Hardware Description Language
IC	Integrated Circuit
INL	Integral Nonlinearity
IQ	Quiescent Current
LDO	Low Dropout Regulator
LHS	Latin Hypercube Sampling
LPM	Low Power Mode

MOS	Metal-Oxide-Semiconductor
MSPS	Mega Samples Per Second
MUX	Multiplexer
NMOS	N-type Metal-Oxide-Semiconductor
PSAOT	Pulse-Skipping Adaptive On-Time
PSAM	Pulse-Skipping Asynchronous Mode
PDK	Process Design Kit
PFM	Pulse Frequency Modulation
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PLL	Phase-Locked Loop
PM	Phase Margin
PMIC	Power Management Integrated Circuit
PMOS	P-type Metal-Oxide-Semiconductor
PWM	Pulse Width Modulation
RC	Resistor-Capacitor
RMS	Root Mean Square
SAR	Successive Approximation Register
SoC	System on Chip
SPL	Skip Prevention Logic
SPS	Samples Per Second
SR	Set-Reset
STG	Signal Transition Graph
USB	Universal Serial Bus
VCO	Voltage-Controlled Oscillator
ZOH	Zero-Order Hold

Introduction

Battery-powered portable devices have become essential in our daily tasks, serving as tools for managing finances, appointments, communication, and other tasks. Inside these devices, there is a crucial component known as the Power Management Integrated Circuit (PMIC). The PMIC not only manages the charging of the battery but also converts the voltage levels to meet the different requirements of the system.

To accomplish the tasks, PMICs typically incorporate one or more DC-DC converters (a device capable of converting a voltage level into another). A step-down DC-DC converter is employed to reduce the input voltage, providing a lower output voltage. Conversely, a step-up DC-DC converter elevates the input voltage as needed.

The most basic DC-DC converter achievable with semiconductor components is a voltage regulator and it works by converting the extra power from the difference between the input and output voltage into heat. Regulators can only work as a step-down DC-DC by construction and the maximum theoretical efficiency is limited by the ratio between the output and input voltages (V_{out}/V_{in}).

For the construction of step-up DC-DC converters or highly efficient counterparts, switched architectures are required. These architectures cyclically store and reuse the energy available in capacitors (switched-capacitor converters) or inductors (switched-inductor converters).

The switching frequency can be constant or variable depending on the choice of the controller topology. Variable frequency topologies are optimized for low power consumption or designed to spread the spectrum of the switching noise. On the other hand, fixed frequency topologies provide good load regulation, low ripple, and maintain the switching noise at a specific frequency with a higher current consumption.

The focus of this work was a digital control for fixed frequency converters. Therefore, instead of defining methods to determine which application requires a fixed frequency DC-DC or a variable frequency one, our concern was the implementation of the control loop in the digital domain, providing that a fixed frequency converter is better suitable for the intended application.

As a matter of fact, most applications need a DC-DC converter that operates both in fixed and variable switching frequency modes. A communication device, for example, need a low ripple supply when transmitting, but it tolerates a higher ripple supply during idle in order to save power. Therefore, at least one mode of operation that involves fixed frequency control loop will be present in most battery powered applications.

Motivation

Competition in the portable device market is tight: users demand higher processing power and longer battery life. Dynamic Voltage and Frequency Scaling (DVFS), which consists in adjusting the microprocessor input voltage and frequency according to the work load, is one of the solutions created to address these issues. This technique requires devices capable of efficiently converting the 2.7 V - 4.2 V voltage from a Lithium-ion battery in a dynamically configurable lower voltage. The control loop of converters for these applications are traditionally implemented in the analog domain due to the difficulty of building low power ADC and DPWM, but the implementation of the control loop in the digital domain can provide more flexibility and re-usability. Our main motivation was thus to look for (and adapt) ADC and DPWM low-power topologies suitable to the indented application, as a mean for enabling a control loop in the digital domain.

Tools

Since the cost of manufacturing an integrated circuit is high, foundries provide Process Design Kits (PDKs) with simulation models that have a fairly good accuracy, which for this work is a PDK of the Global Foundries 180 nm process. The work was carried out by means of computer modeling of the proposed circuits with the Virtuoso software from Cadence®, which is a well known and established software provider for the microelectronics field. From Virtuoso, Schematic-XL, Maestro, and Layout-XL were mainly used. Spectre with Xcelium integration was also installed in order to perform Verilog-AMS simulations. For high level modeling, the MATLAB® was used.

In addition, artificial intelligence tools, specifically ChatGPT 3.7, were used to verify the clarity of the text and suggest rephrasing to improve readability. It is important to note that no content presented in this work was generated by artificial intelligence.

Goals

The goal of this work was to design a digitally driven control for a fixed frequency buck converter in the context of PMIC for portable battery-powered applications. The control loop consists of an ADC, a digital state machine with a standard Proportional Integral Derivative (PID) logic, and a DPWM converter. The desirable performance is described below:

- Input voltage between 2.7 V and 5.5 V. Typical 3.7 V.
- Output voltage between 1.62 V and 1.98 V. Typical 1.8 V.

- Load current between 0 to 200 mA.
- Switching frequency of 2 MHz.
- The sum of the ADC accuracy, line transients variations, and load transients variations must be within 2% of the voltage reference once the voltage reference is settled. The system can violate the ±2% specification when the voltage reference is being updated.
- Minimize power consumption.
- Reduce complexity and circuit area whenever possible.

Text structure

The remaining document is structured as follows:

- Chapter 1: Review of the theory and the state of the art.
- Chapter 2: High level design and establishment of high level guidelines for the design of the ADC, DPWM, and digital PID.
- Chapter 3: Development of the implementation details of the ADC, DPWM, and digital PID.
- Chapter 4: Result of the ADC, DPWM, and top level simulations.
- Chapter 5: Discussion of the results.

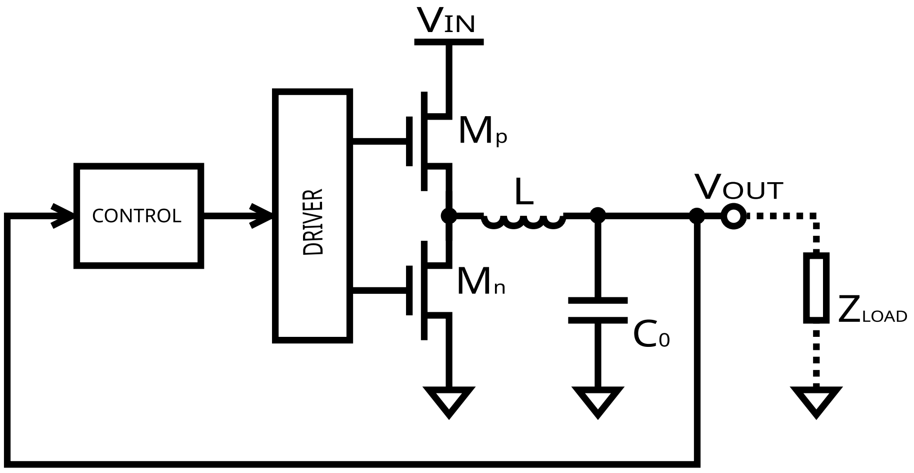
1 Basic concepts and State of the art

This chapter summarizes basic concepts of buck converters and some previous research on the subject of integrated digitally controlled buck converters.

1.1 Basic concepts

The main components of a buck converter are inductor, capacitor, power switches, driver, and control as shown in Figure 1. This particular architecture is prevalent in most battery portable devices in order to efficiently convert the battery voltage into a constant DC supply for the application. The circuit can operate in different conduction modes and control schemes. A summary of these aspects is given in the following subsections.

Figure 1 – Simplified block diagram of a step-down switched-inductor DC-DC converter.

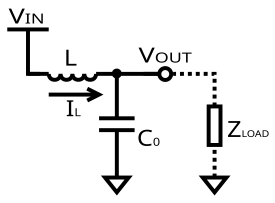


Source: The author.

1.1.1 Conduction modes

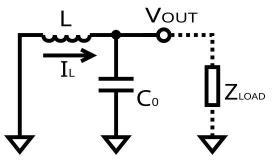
In Continuous Conduction Mode (CCM), the circuit operates in two stages: inductor charge stage and inductor discharge stage. During the inductor charge stage (M_P is on and M_N is off as depicted in Figure 2), the surplus energy is stored in the inductor instead of being converted into heat. During the inductor discharge stage (the transistor M_P is off and M_N is on as depicted on Figure 3), the energy stored in the inductor is transferred to the load. The converter switches between the two stages cyclically with a period that is called switching period. The inverse of the switching period is the switching frequency of the converter.

Figure 2 – Inductor charge stage.



Source: The author.

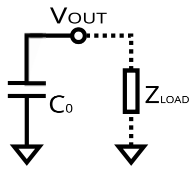
Figure 3 – Inductor discharge stage.



Source: The author.

In Discontinuous Conduction Mode (DCM), the operation is similar, but the converter enters a third stage (Figure 4) with both transistor $\mathrm{M_P}$ and $\mathrm{M_N}$ off when the inductor current reaches 0 A.

Figure 4 – Equivalent circuit for null inductor current in DCM.



Source: The author.

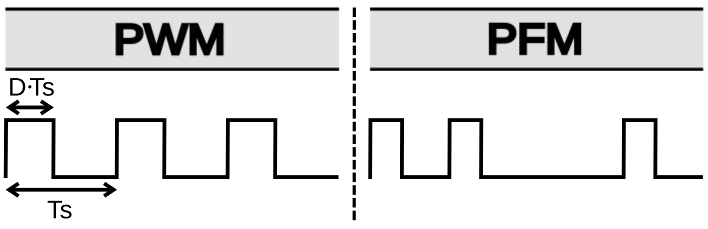
1.1.2 Buck control schemes

Commonly, a buck can be controlled using Pulse Width Modulation (PWM) or Pulse Frequency Modulation (PFM).

PWM modulation involves the representation of an analog signal with two levels (0 or 1) oscillating at a fixed switching period (T_S). The amplitude information of the analog signal is encoded in the duty cycle (D), which represents the ratio between the time the signal remains at one and the switching period. PFM modulation is similar, but the pulse width is usually fixed with a variable switching frequency. Both modulation methods are illustrated in Figure 5.

When employing the PWM modulation method, the control can operate in voltage mode or current mode. In voltage mode, the control relies on the output voltage information to determine the duty cycle. In current mode, the control measures both the output voltage and the inductor current to determine the duty cycle.

Figure 5 – PWM and PFM modes.



Source: The author.

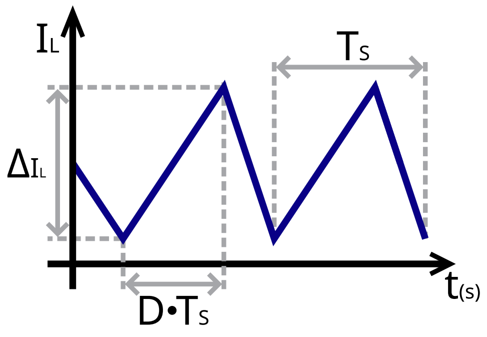
1.1.3 Average model with voltage mode control in CCM

In order to control the output voltage with a constant switching frequency, the converter must change the duty cycle, therefore mathematical relations between the duty cycle and the output voltage are important for the correct design of the control loop.

This section provides mathematical models of the DC-DC converter correlating the duty cycle with the output voltage in CCM. Besides, approximate equations for the capacitor voltage ripple and the inductor current ripple are also shown. Most of the models shown in this chapter are based on or inspired by the material available in the book Pulse-Width Modulated DC-DC Power Converters [1].

The inductor current in CCM is shown in Figure 6, where I_L represents the inductor current, T_S is the switching period, and D is the duty cycle (a real number between 0 and 1).

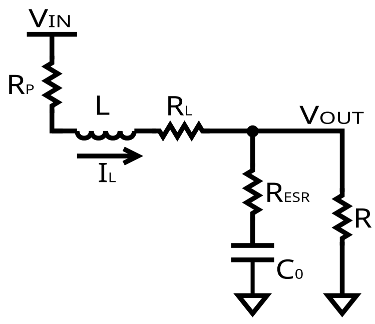
Figure 6 – Inductor current in continuous conduction mode.



Source: The author.

The equivalent circuit of the converter during the inductor charge stage is shown in Figure 7, where R_P is the resistance of the transistor M_P, L is the inductance, C_0 is the capacitance, R_L is the equivalent series resistance of the inductor, R_{ESR} is the equivalent series resistance of the capacitor, V_{IN} is the input voltage, and V_{OUT} is the output voltage.

Figure 7 – Inductor charge stage including parasitic resistances.



Source: The author.

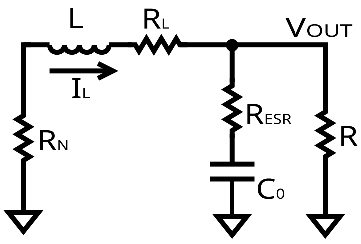
Derivatives of the inductor current and the capacitor voltage for the inductor charge stage are shown below (V_{C_0} represents the voltage across C_0):

$$\left.\frac{dI_L}{dt}\right._{CHG} = -\frac{I_L}{L}\cdot\left(\frac{R_{ESR}\cdot R}{R_{ESR}+R}+R_P+R_L\right)-\frac{V_{C_O}}{L}\cdot\frac{R}{R_{ESR}+R}+\frac{V_{IN}}{L} \tag{1.1}$$

$$\left.\frac{dV_{C_O}}{dt}\right._{CHG} = \frac{I_L}{C_O}\cdot\frac{R}{R_{ESR}+R}-\frac{V_{C_O}}{C_O}\cdot\frac{1}{R_{ESR}+R} \tag{1.2}$$

The model of the converter during the inductor discharge stage is shown in Figure 8. All the parameters are identical to the inductor charge stage except for R_N, which represents the resistance of transistor M_N.

Figure 8 – Inductor discharge stage including parasitic resistances.



Source: The author.

Derivatives of the inductor current and the capacitor voltage are now equal to:

$$\left.\frac{dI_L}{dt}\right._{DIS} = -\frac{I_L}{L}\cdot\left(\frac{R_{ESR}\cdot R}{R_{ESR}+R}+R_N+R_L\right)-\frac{V_{C_O}}{L}\cdot\frac{R}{R_{ESR}+R} \tag{1.3}$$

$$\left.\frac{dV_{C_O}}{dt}\right._{DIS} = \frac{I_L}{C_O}\cdot\frac{R}{R_{ESR}+R}-\frac{V_{C_O}}{C_O}\cdot\frac{1}{R_{ESR}+R} \tag{1.4}$$

The average of the derivative of the inductor current and capacitor voltage within one switching cycle will be equal to:

$$\overline{\frac{dI_L}{dt}} = \frac{1}{T_S}\cdot\left(\int_0^{D\cdot T_S}\left.\frac{dI_L}{dt}\right._{CHG}dt+\int_{D\cdot T_S}^{T_S}\left.\frac{dI_L}{dt}\right._{DIS}dt\right) \tag{1.5}$$

$$\overline{\frac{dV_{C_O}}{dt}} = \frac{1}{T_S}\cdot\left(\int_0^{D\cdot T_S}\left.\frac{dV_{C_O}}{dt}\right._{CHG}dt+\int_{D\cdot T_S}^{T_S}\left.\frac{dV_{C_O}}{dt}\right._{DIS}dt\right) \tag{1.6}$$

$$\begin{aligned}\overline{\frac{dI_L}{dt}} = &-\frac{\overline{I_L}}{L}\cdot\left[\frac{R_{ESR}\cdot R}{R_{ESR}+R}+R_P\cdot D+R_N\cdot(1-D)+R_L\right]\\ &-\frac{\overline{V_{C_O}}}{L}\cdot\frac{R}{R_{ESR}+R}+D\cdot\frac{V_{IN}}{L}\end{aligned} \tag{1.7}$$

$$\overline{\frac{dV_{C_O}}{dt}} = \frac{\overline{I_L}}{C_O}\cdot\frac{R}{R_{ESR}+R}-\frac{\overline{V_{C_O}}}{C_O}\cdot\frac{1}{R_{ESR}+R} \tag{1.8}$$

In steady state, the average inductor current and the average capacitor voltage will not change, therefore:

$$\overline{\frac{dV_{C_O}}{dt}} = 0 \implies \overline{V_{C_O}} = R \cdot \overline{I_L} \tag{1.9}$$

$$\overline{\frac{dI_L}{dt}} = 0 \implies \overline{V_{C_O}} = \frac{D \cdot V_{IN}}{1 + \frac{R_P \cdot D + R_N \cdot (1-D) + R_L}{R}} \tag{1.10}$$

Since the average capacitor current must also be 0 in stead state, the average output voltage will be:

$$\overline{V_{OUT}} = \overline{V_{C_O}} \tag{1.11}$$

1.1.3.1 Approximate capacitor voltage ripple and inductor current ripple

Assuming that the parasitic resistances $\mathrm{R_P}$, $\mathrm{R_N}$, and $\mathrm{R_L}$, are negligible, the inductor current ripple (Δ_{I_L}) can be calculated as follows:

$$\Delta_{I_L} = \frac{D \cdot T_S \cdot \left(V_{IN} - \overline{V_{OUT}}\right)}{L} \tag{1.12}$$

Assuming that $\mathrm{R_{ESR}}$ is also negligible, the output voltage ripple ($\Delta_{V_{OUT}}$) can be calculated as follows:

$$\Delta_{V_{OUT}} = \frac{\frac{\Delta_{I_L}}{2} \cdot \frac{T_S}{2}}{2 \cdot C_0} = \frac{\Delta_{I_L} \cdot T_S}{8 \cdot C_0} \tag{1.13}$$

When $\mathrm{R_{ESR}}$ is large and it dominates the ripple contribution, the output voltage ripple is:

$$\Delta_{V_{OUT}} = \Delta_{I_L} \cdot R_{ESR} \tag{1.14}$$

An upper bound for the output voltage ripple can be established as:

$$max\left(\Delta_{V_{OUT}}\right) < \frac{\Delta_{I_L} \cdot T_S}{8 \cdot C_0} + \Delta_{I_L} \cdot R_{ESR} \tag{1.15}$$

1.1.3.2 Continuous time small signal model

The small signal equations can be obtained by analyzing the sensitivity of Equations 1.7 and 1.8 to small variations in the duty cycle and the input voltage.

In what follows, small signal variations were changed to lowercase and dependencies on large signals were kept in uppercase.

$$\begin{aligned}\overline{\frac{di_L}{dt}} = &-\frac{\overline{i_L}}{L}\cdot\left[\frac{R_{ESR}\cdot R}{R_{ESR}+R}+R_P\cdot D+R_N\cdot(1-D)+R_L\right]\\ &-\frac{\overline{v_{C_O}}}{L}\cdot\frac{R}{R_{ESR}+R}\\ &-\frac{\overline{I_L}}{L}\cdot d\cdot(R_P-R_N)+d\cdot\frac{V_{IN}}{L}\\ &+\frac{v_{in}}{L}\cdot D\end{aligned} \tag{1.16}$$

$$\overline{\frac{dv_{C_O}}{dt}} = \frac{\overline{i_L}}{C_O}\cdot\frac{R}{R_{ESR}+R}-\frac{\overline{v_{C_O}}}{C_O}\cdot\frac{1}{R_{ESR}+R} \tag{1.17}$$

Using the space state representation:

$$\frac{dx}{dt} = A\cdot x+B\cdot u \tag{1.18}$$

$$\overline{v_{out}} = C\cdot x+D\cdot u \tag{1.19}$$

In this case, x, i_L, A, B, C, D and u are equal to:

$$x = \begin{bmatrix}\overline{i_L}\\ \overline{v_{C_O}}\end{bmatrix} \tag{1.20}$$

$$u = \begin{bmatrix}d\\ v_{in}\end{bmatrix} \tag{1.21}$$

$$A = \begin{bmatrix}-\frac{\frac{R_{ESR}\cdot R}{R_{ESR}+R}+R_P\cdot D+R_N\cdot(1-D)+R_L}{L} & -\frac{R}{L\cdot(R_{ESR}+R)}\\ \frac{R}{C_O\cdot(R_{ESR}+R)} & -\frac{1}{C_O\cdot(R_{ESR}+R)}\end{bmatrix} \tag{1.22}$$

$$B = \begin{bmatrix}\frac{V_{IN}\cdot(R+R_N+R_L)}{L\cdot[R+R_P\cdot D+R_N\cdot(1-D)+R_L]} & \frac{D}{L}\\ 0 & 0\end{bmatrix} \tag{1.23}$$

$$C = \begin{bmatrix}\frac{R_{ESR}\cdot R}{R_{ESR}+R} & \frac{R}{R_{ESR}+R}\end{bmatrix} \tag{1.24}$$

$$D = \begin{bmatrix}0 & 0\end{bmatrix} \tag{1.25}$$

The transfer functions $H(s) = \frac{\overline{v_{out}(s)}}{d(s)}$ and $F(s) = \frac{\overline{v_{out}(s)}}{v_{in}(s)}$ can be calculated as follows (I is the identity matrix):

$$\begin{bmatrix} H(s) & F(s) \end{bmatrix} = C\left(sI - A\right)^{-1} B + D \tag{1.26}$$

Therefore:

$$H(s) = G_H \cdot \frac{1 - \frac{s}{z_1}}{\left(1 - \frac{s}{p_1}\right) \cdot \left(1 - \frac{s}{p_2}\right)} \tag{1.27}$$

$$F(s) = G_F \cdot \frac{1 - \frac{s}{z_1}}{\left(1 - \frac{s}{p_1}\right) \cdot \left(1 - \frac{s}{p_2}\right)} \tag{1.28}$$

G_H, G_F, z_1, p_1 and p_2 are equal to:

$$G_H = \frac{V_{IN} \cdot R \cdot (R_L + R + R_N)}{(R + R_S)^2} \tag{1.29}$$

$$G_F = \frac{D \cdot R}{(R + R_S)} \tag{1.30}$$

$$z_1 = -\frac{1}{C_O \cdot R_{ESR}} \tag{1.31}$$

$$p_1 = -\frac{b + j \cdot \sqrt{4 \cdot a - b^2}}{2 \cdot a} \tag{1.32}$$

$$p_2 = -\frac{b - j \cdot \sqrt{4 \cdot a - b^2}}{2 \cdot a} \tag{1.33}$$

, in which:

$$a = \frac{C_O \cdot (R_{ESR} + R) \cdot L}{(R + R_S)} \tag{1.34}$$

$$b = \frac{C_O \cdot (R_{ESR} + R) \cdot \left(\frac{R_{ESR} \cdot R}{R_{ESR} + R} + R_S\right) + L}{(R + R_S)} \tag{1.35}$$

$$R_S = R_P \cdot D + R_N \cdot (1 - D) + R_L \tag{1.36}$$

1.1.3.3 Discrete time small signal model

The derivation of the discrete-time small-signal model is complex. Various works addressing this topic have been documented, with a precise model for the entire frequency range presented in [2]. This model assumes that the sampling occurs in fixed intervals and includes the delay between the sampling event and the trailing edge of the PWM signal.

When the ADC operates by averaging within an interval, as proposed in [3], obtaining a model becomes even more challenging. However, if the control loop's bandwidth is significantly lower than the sampling frequency, the discretization of the average model offers a first approximation. This involves multiplying Equation 1.27 by the transfer function of the Zero-Order Hold (ZOH), followed by decomposition into partial fractions. Each term is then converted to discrete time as shown in [4]. The result of this process is presented below:

$$\frac{H(z)}{G_H} = 1 + \frac{p2}{p1 - p2} \cdot \left(1 - \frac{p1}{z1}\right) \cdot \frac{z-1}{z - e^{p1 \cdot Ts}} + \frac{p1}{p2 - p1} \cdot \left(1 - \frac{p2}{z1}\right) \cdot \frac{z-1}{z - e^{p2 \cdot Ts}} \tag{1.37}$$

The analytical analysis presented in this chapter will be used in Chapter 2.

1.1.4 Control loop performance measurements

Since the buck works in closed loop, it is important to list the basic control loop performance parameters. Some of them are:

- **Line over/undershoot**: This parameter is measured by rapidly increasing or decreasing the input voltage, followed by recording the maximum deviation between the steady state and transient output voltage levels.
- **Load over/undershoot**: This parameter is measured by rapidly increasing or decreasing the load current, followed by recording the maximum deviation between the steady state and transient output voltage levels.
- **Voltage reference step response**: A step in the voltage reference is induced, and the output voltage is characterized in terms of rise time t_r, settling time t_s, and overshoot M[5].
- **Phase Margin (PM)**: In a negative feedback loop, the amount by which the phase of open loop frequency response exceeds -180° when the gain is equal to 0 dB [5].
- **Gain Margin (GM)**: In a negative feedback loop, the gain margin is the amount by which the open loop gain can to be increased so the phase is greater than -180° [5].

1.1.5 Basic ADC performance measurements

A digital buck will probably require an ADC in order to digitize the output voltage, therefore is is important to establish some basic performance parameters for the ADC. According to [6] the main basic parameters are:

- **Resolution**: The number of bits in the digital code
- **Accuracy**: The closeness of the digital code to the actual analog input value, often expressed as a percentage of the full-scale range. In the case of the differential window ADC proposed in this work, the control loop will try to keep the difference between the output voltage and the voltage reference as low as possible. Therefore, the accuracy measurement is more relevant for the code that represents 0 V.
- **Step Size**: The analog voltage difference between two consecutive digital codes.
- **Linearity**: The degree of linearity in the relationship between analog input and digital output is crucial for the performance of an ADC. Non-linearity is commonly assessed through two metrics: Differential Non-Linearity (DNL) and Integral Non-Linearity (INL). These metrics are defined by Equations 1.38 and 1.39, respectively [7]. In these equations, V_i represents the input voltage corresponding to the code i, N denotes the resolution of the ADC, and Q_{ADC} denotes the ideal step size.

$$DNL(i) = \frac{V_{i+1} - V_i}{Q_{ADC}} - 1,\ where \quad 0 < i < 2^N - 2 \tag{1.38}$$

$$INL(i) = \frac{V_i - V_0}{Q_{ADC}} - i,\ where \quad 0 < i < 2^N - 1 \tag{1.39}$$

- **Sampling Rate**: The rate at which the ADC samples the analog input signal, typically measured in Samples Per Second (SPS).
- **DC current**: The current trough the ADC supply during operation.

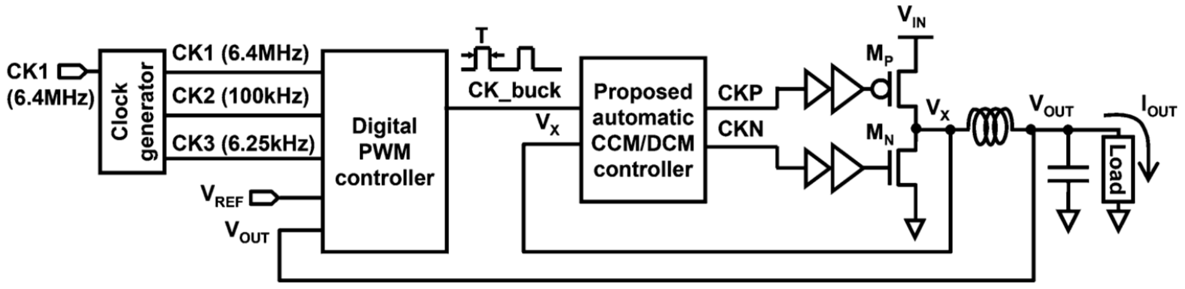
1.2 Summary of previous research on integrated digitally controlled DC-DC converters

There are several works available on the subject of integrated digitally controlled DC-DC converters. The following subsections aim to provide the review of a few articles which represent some relevant techniques available at the time of this writing.

1.2.1 A 0.6 V Input CCM/DCM Operating Digital Buck Converter in 40 nm CMOS

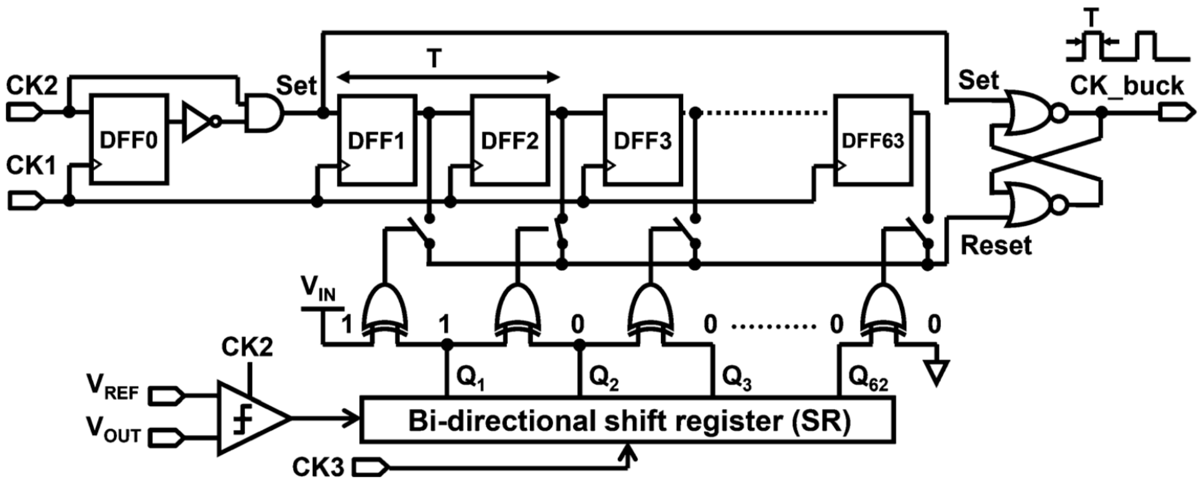
ZHANG et al. [8] designed a 0.6 V Input DC-DC converter for photovoltaic applications. The converter has a digital control loop and automatic switching between continuous and discontinuous operating mode to ensure low power consumption. The top level block diagram and the control loop block diagram are shown in Figures 9 and 10, respectively.

Figure 9 – Block diagram of the converter [8].



Source: ZHANG et al. [8].

Figure 10 – Block diagram of the PWM controller [8].

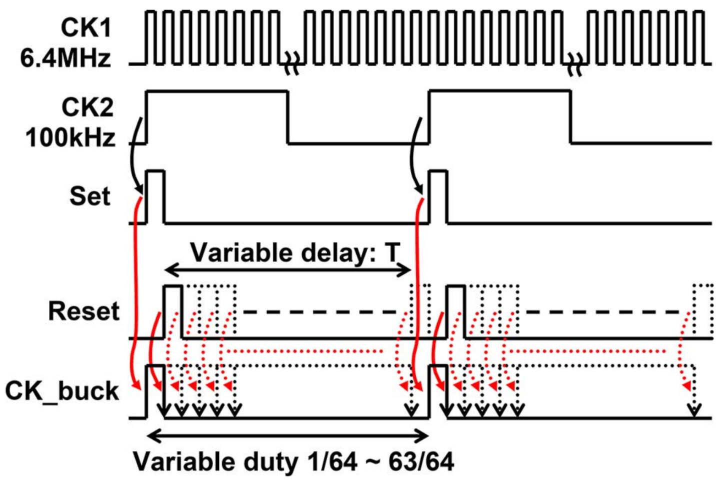


Source: ZHANG et al. [8].

A clock signal of 6.4 MHz (CK1) controls a chain of 64 shift registers in which a reset pulse moves from the beginning to the end of the chain once every 10 µs. The division of CK1 by 64 generates another clock pulse (CK2) which generates a set pulse also once every 10 µs. A third division of CK2 by 16 (CK3) controls a bidirectional shift register. If the output voltage is lower than the reference, the bi-directional register will shift right. If the output voltage is higher than the average, the bidirectional shift register will shift left.

The set signal together with the reset signal selected by the bidirectional shift register are connected to a Set-Reset (SR) flip-flop, which generates a 100 kHz PWM signal with a duty cycle resolution of 1,5625% (64 steps). The duty cycle is updated with a frequency of 6.25 kHz. Check the timing diagram (Figure 11) for more information.

Figure 11 – Timing diagram [8].



Source: ZHANG et al. [8].

A summary of the main parameters is available in Table 1. Although the specification of a DC-DC converter designed for photovoltaic applications cannot be compared with the specification of a DC-DC converter designed for generic battery powered applications, the PWM control technique with only one comparator between the input voltage and the reference voltage worth some consideration.

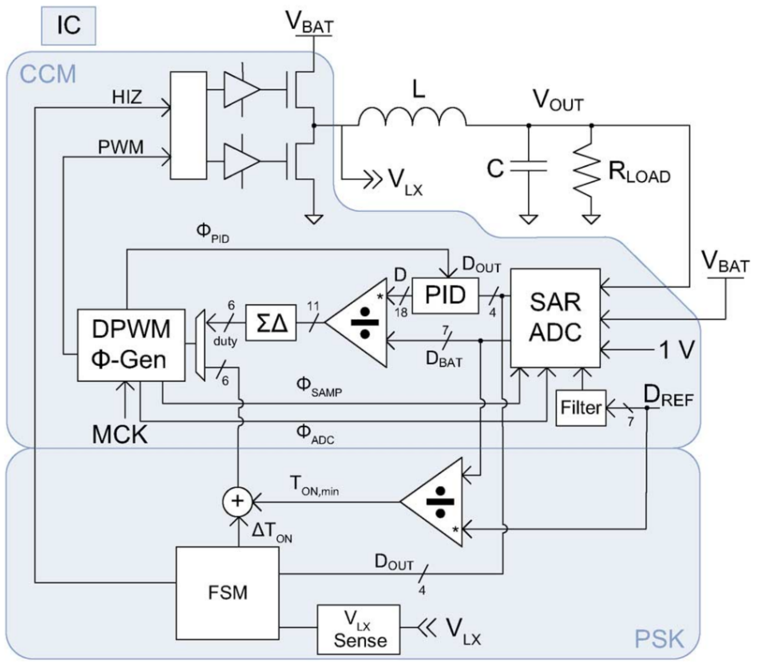
Table 1 – Summary of the main parameters in [8].

Parameter	**Minimum**	**Typical**	**Maximum**
Process	-	CMOS 40 nm	-
Input voltage	0.6 V	-	1.1 V
Output voltage	0.3 V	-	0.55 V
Output current	50 µA	-	10 mA
Inductor	-	220 µH	-
Capacitor	-	167 nF	-
Efficiency	-	-	94%
Switching frequency	-	100 kHz	-
System clock	-	6.4 MHz	-
DPWM step	-	1.5625%	-

1.2.2 A 65 nm, 1 A Buck Converter With Multi-Function SAR-ADC-Based CCM/PSK Digital Control Loop

A 1 A DC-DC converter with embedded digital control implemented in 65 nm CMOS technology was developed by Sébastien Cliquennois et al. [9]. The proposed DC-DC converter has a multi-function Successive Approximation Register (SAR) ADC and a non-linear PID controller. It can also switch automatically between continuous conduction mode and pulse-skipping mode to achieve better efficiency over the load current range. The block diagram is shown in Figure 12.

Figure 12 – Simplified block diagram of the DC-DC converter proposed in [9].

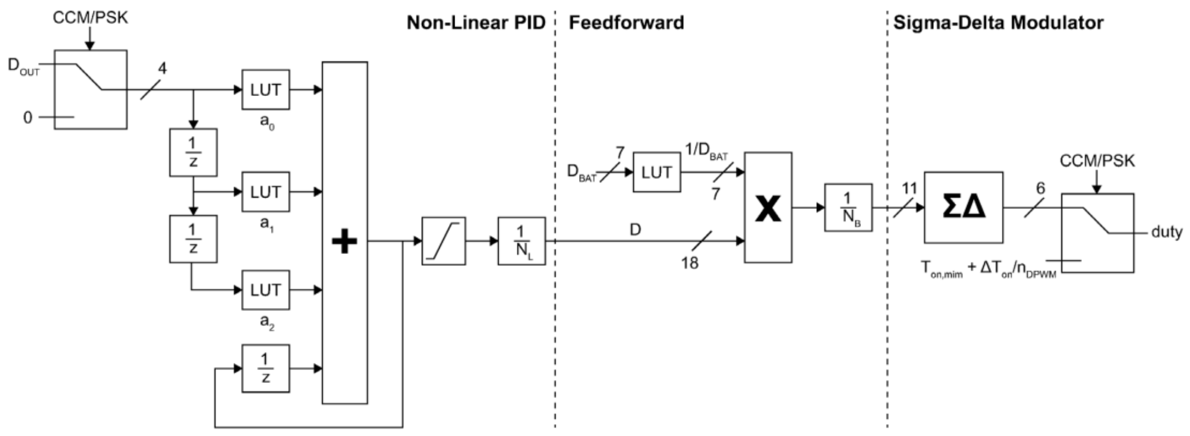


Source: Sébastien Cliquennois et al. [9].

A 7-bit SAR ADC is used in time-sharing to digitize both V_{OUT} with a sampling frequency of 6.4 MHz and V_{BAT} with a sampling frequency of 0.8 MHz. In continuous mode, the digitized V_{OUT} and V_{BAT} are fed to a digital controller, whose block diagram is shown in Figure 13.

The DPWM works based on a ripple counter operated at 307.2 MHz to provide a 48 level PWM. The transition between continuous conduction mode and the pulse skipping mode as well as the operation of the pulse skipping mode are not relevant in the context of this work, because of the focus on the control for fixed frequency operation.

Figure 13 – Block diagram of the digital controller [9].



Source: Sébastien Cliquennois et al. [9].

A summary of the main parameters can be found in Table 2.

Table 2 – Summary of the main parameters in [9].

Parameter	**Minimum**	**Typical**	**Maximum**
Process	-	CMOS 65 nm	-
Input voltage	2.3 V	-	4.8 V
Output voltage	0.6 V	-	1.35 V
Output current	0 A	-	1 A
Inductor	-	470 nH	-
Load Capacitance	-	10 µF	-
Line Transient (600mV in 10 µs)	-	10 mV	-
Load Transient (200mA in 500 ns)	-	20 mV	-
Efficiency	-	-	85%
Switching frequency	-	6.4 MHz	-
SAR clock frequency	-	153.6 MHz	-
DPWM clock frequency	-	307.2 MHz	-
Output ripple	-	-	10 mV
ADC step	-	10 mV	-
DPWM step	-	2.083%	-
Active area without power transistors	-	0.038 mm^2	-
Current consumption without power transistors	-	115.5 µA	-
Current consumption (ADC)	-	18 µA	-
Current consumption (DPWM)	-	82.5 µA	-
Current consumption (PID)	-	15 µA	-

1.2.3 A 4 µA Quiescent-Current Dual-Mode Digitally Controlled Buck Converter IC for Cellular Phone Applications

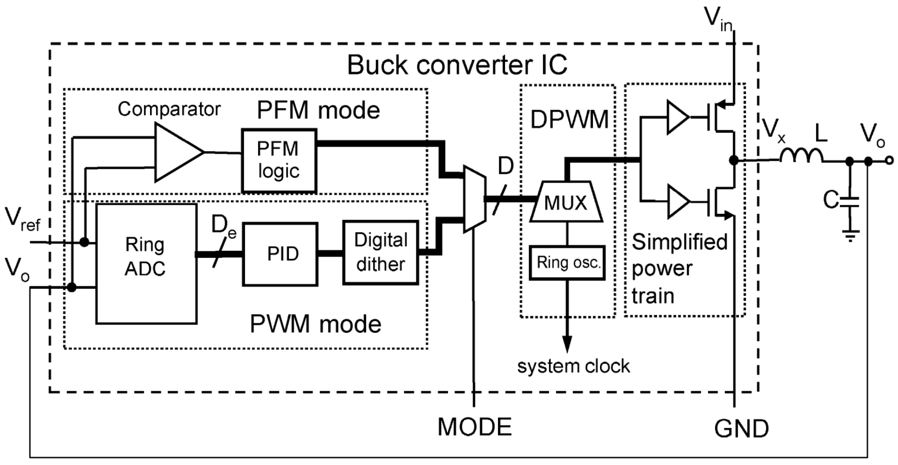
The DC-DC converter developed by Jinwen Xiao et al. [3] minimizes quiescent current by switching automatically between PFM and PWM. The block diagram can be visualized in Figure 14. In PWM mode, a ring ADC topology (Figure 15) digitizes the difference between the output voltage and the reference voltage which is then applied to a PID controller. The output of the PID controller is then connected to a DPWM converter (Figure 16).

The ADC has two ring oscillators operating in sub-threshold voltage biased by a differential pair. Every tap of the ring oscillator is connected to counters re-started at the beginning of the switching cycle, this structure averages and digitizes the difference between the output voltage and the reference voltage (Figure 15).

The implementation of the DPWM consists in a ring-oscillator-Multiplexer (MUX) topology (Figure 16) and latches. The set signal of the PWM will be generated when the square wave that propagates through the ring reaches the tap selected by the MUX.

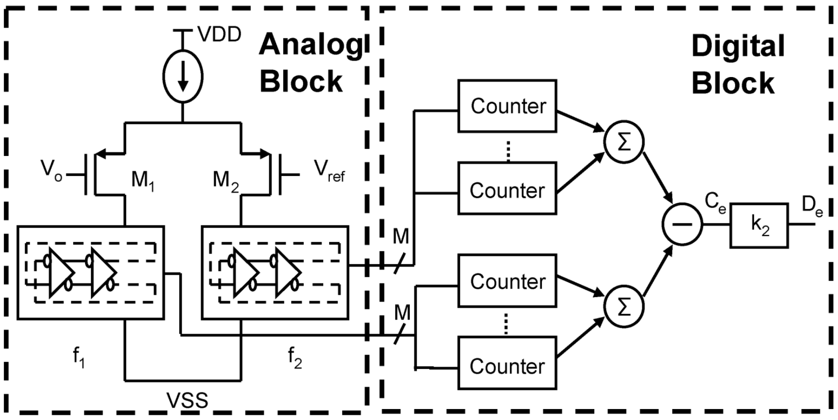
The reset signal is generated when the square wave reaches a predefined tap of the ring oscillator.

Figure 14 – Architecture proposed by Jinwen Xiao et al. [3].



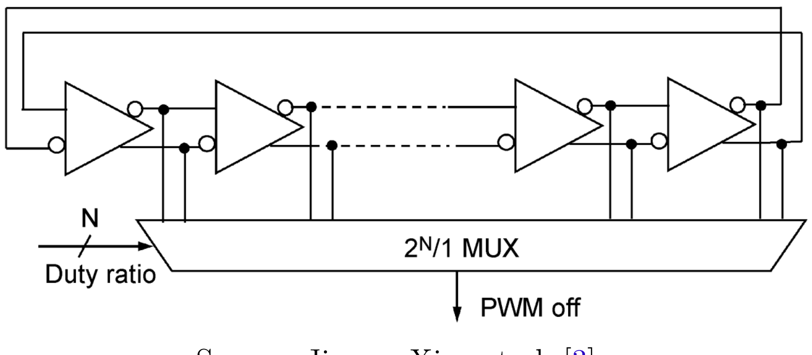
Source: Jinwen Xiao et al. [3].

Figure 15 – Ring ADC proposed by Jinwen Xiao et al. [3].



Source: Jinwen Xiao et al. [3].

Figure 16 – DPWM proposed by Jinwen Xiao et al. [3].



Source: Jinwen Xiao et al. [3].

A summary of the main parameters can be found in Table 3.

Table 3 – Summary of the main parameters in [3].

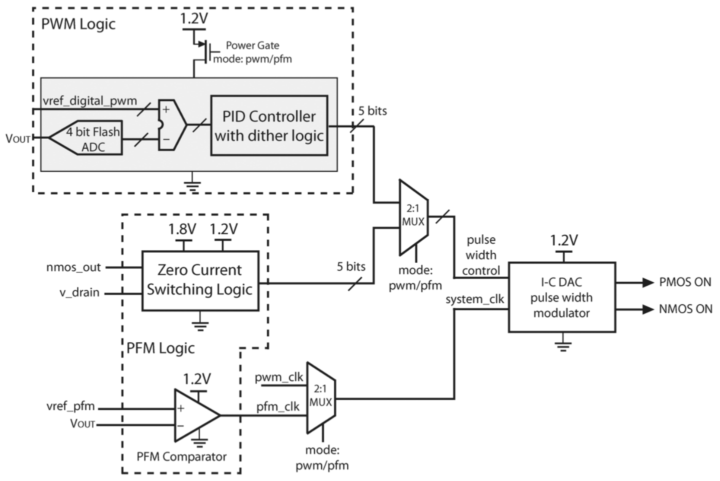
Parameter	Minimum	Typical	Maximum
Process	-	CMOS 0.25 µm	-
Input voltage	2.8 V	-	5.5 V
Output voltage	1.0 V	-	1.8 V
Output current	100 µA	-	400 mA
Inductor	-	10 µH	-
Load Capacitance	-	47 µF	-
Load Transient (50-150 mA)		25 mV	-
Switching frequency (PWM)	0.5 MHz	-	1.5 MHz
Output ripple (PWM)	-	2 mV	-
Efficiency	-	-	above 90%
ADC step	-	16 mV	-
DPWM step	-	3.2%	-
Active area	-	2 mm^2	-
Active area (ADC+DPWM+PID)	-	0.23 mm^2	-
Current consumption (ADC)	-	37 µA	-

1.2.4 20 µA to 100 mA DC-DC Converter With 2.8-4.2 V Battery Supply for Portable Applications in 45 nm CMOS

Saurav et al. [10] designed a DC-DC converter for portable applications capable of delivering currents from 20 µA to 100 mA. The converter operates in two modes (PWM and PFM) and it can be supplied with a Li-ion battery (2.8 V - 4.2 V).

The general architecture is depicted in Figure 17. In PWM mode, the error between the reference voltage and the output voltage is converted to digital by a 4-bit flash ADC.

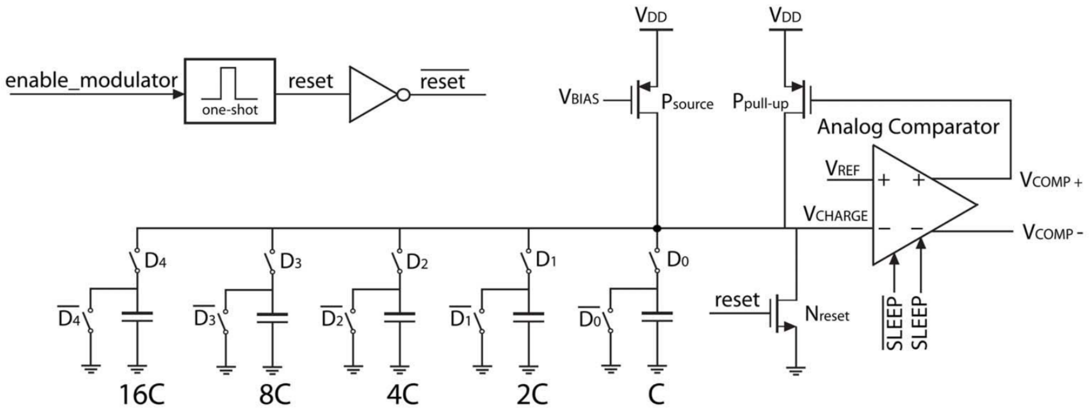
Figure 17 – Architecture proposed by Saurav et al. [10].



Source: Saurav et al. [10].

The digital representation of the error is then fed to the PID controller which in turn generates the input of the DPWM (Figure 18) which is based on a network of capacitors with values in powers of 2. By changing the total capacitance connected to V_{CHARGE}, the PID controller can vary the duty cycle of the PWM signal.

Figure 18 – Digital to Pulse Width Modulation proposed by Saurav et al. [10].



Source: Saurav et al. [10].

A summary of the main parameters can be found in Table 4.

Table 4 – Summary of the main parameters in [10].

Parameter	**Minimum**	**Typical**	**Maximum**
Process	-	CMOS 45 nm	-
Input voltage	2.8 V	-	4.2 V
Output voltage	0.4 V	-	1.2 V
Output current	20 µA	-	100 mA
Inductor	-	10 µH	-
Load Capacitance	-	2 µF	-
Load Transient (10–50 mA in 25 µs)	-	10 mV	-
Switching frequency (PWM)	-	2 MHz	-
Efficiency	-	-	87.2%
ADC step	-	50 mV	-
DPWM step	-	1%	-
Active area (ADC+DPWM+PID)	-	0.07 mm^2	-
Current consumption (DPWM)	-	approximately 6 µA	-

1.2.5 A 180 nA Quiescent Current Digital Control Dual-Mode Buck Converter With a Pulse-Skipping Load Detector for Long-Range Applications

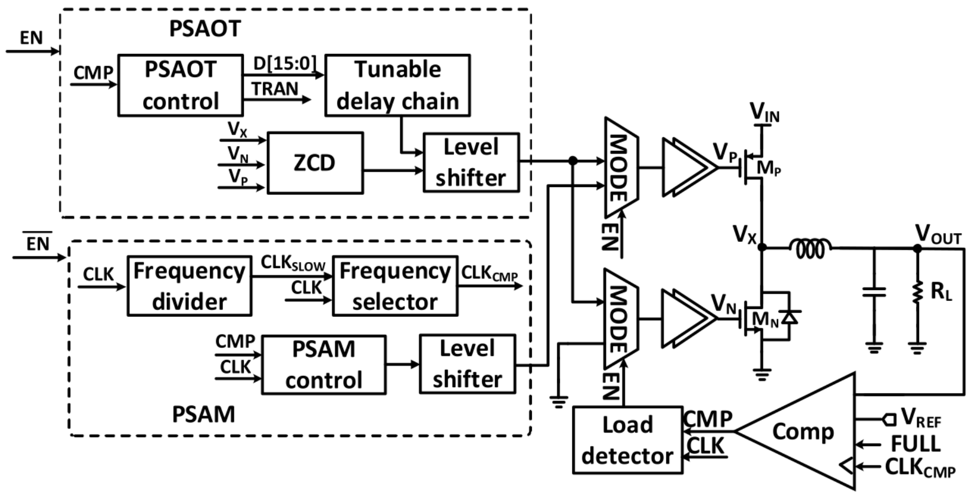
TSAI, T.-H. et al. [11] suggest a DC-DC converter that operates in DCM with automatic switching between Pulse-Skipping Adaptive On-Time (PSAOT), and Pulse-Skipping Asynchronous Mode (PSAM). The block diagram is shown in Figure 19.

The PSAM mode has a low Quiescent Current (IQ) and is suitable for low currents. In this mode, the PSAM circuitry will turn on the PMOS switch (with a fixed on-time) if the output voltage is lower than the voltage reference. The clock frequency of the PSAM can be configured in order to further decrease IQ at a cost of a slower transient response.

On the other hand, the PSAOT mode has a higher IQ and is suitable for high currents. In this mode, the PSAOT circuitry also turns on the PMOS switch if the output voltage is lower than the voltage reference, but, differently from the PSAM mode, the PSAOT circuitry adjusts the PMOS on-time according to the load.

The comparison between the voltage reference and the output voltage is carried out by a clocked comparator and the load estimation is done by a digital circuit that counts the number of clock cycles between two high logic levels in the output of the comparator.

Figure 19 – Architecture proposed by TSAI, T.-H. et al. [11].



Source: TSAI, T.-H. et al. [11].

A summary of the main parameters can be found in Table 5.

Table 5 – Summary of the main parameters in [11].

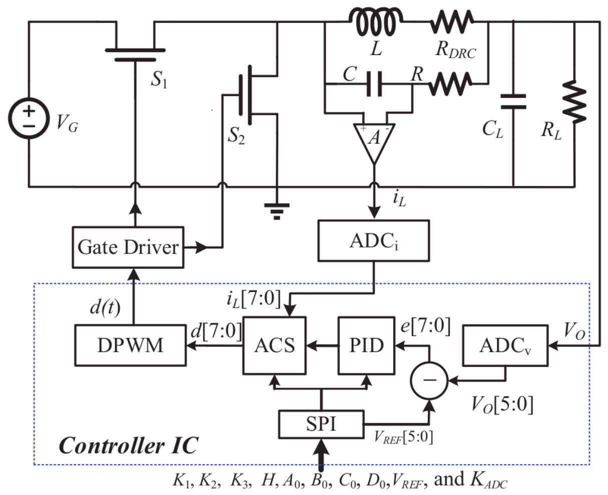
Parameter	Minimum	Typical	Maximum
Process	-	CMOS 350 nm	-
Input voltage	4.2 V	-	5.0 V
Output voltage	-	3.3 V	-
Output current	0 A	-	100 mA
Inductor	-	4.7 µH	-
Load Capacitance	-	10 µF	-
Output Ripple at 10 mA	-	16 mV	-
Efficiency	-	-	92.9%
Switching Frequency	-	-	500 kHz
Active area	-	3.78 mm^2	-

1.2.6 A Digital Controller IC for High-Frequency DC-DC Switching Converters

The architecture of the digital controller proposed by CHEN, N. et al. (Figure 20) consists of a delay line ADC (Figure 21), a current mode digital control (Figure 22), and a Phase-Locked Loop (PLL) based DPWM (Figure 23).

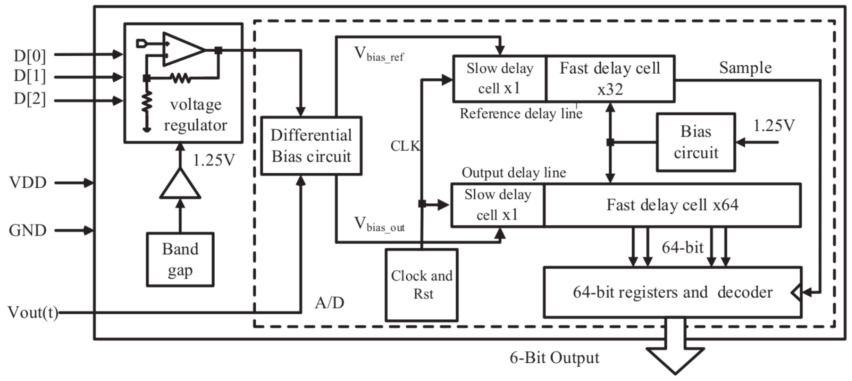
There are two ADCs. The first one is based on two delay lines with bias currents proportional to the difference between the output voltage and the voltage reference. One of the delay lines is connected to the input of a 64-bit register and the last inverter of the second delay line works like a clock for the same register. Consequently, this configuration generates a thermometer code with 64 levels that represents the difference between the voltage reference and the output voltage (Figure 21). The second ADC is external to the controller Integrated Circuit (IC) and its architecture is not mentioned, but it measures the inductor current in the end of the cycle so the digital logic can calculate the duty cycle for the next cycle.

Figure 20 – Architecture proposed by CHEN, N. et al. [12].



Source: CHEN, N. et al. [12].

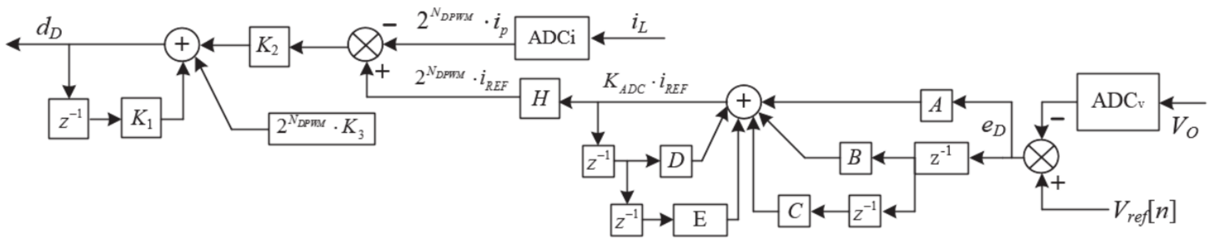
Figure 21 – ADC proposed by CHEN, N. et al. [12].



Source: CHEN, N. et al. [12].

The digital control has one loop to control the inductor current and another loop to control the output voltage. This configuration has the advantage of eliminating the imaginary poles of the converter, but it requires an extra ADC to measure the inductor current.

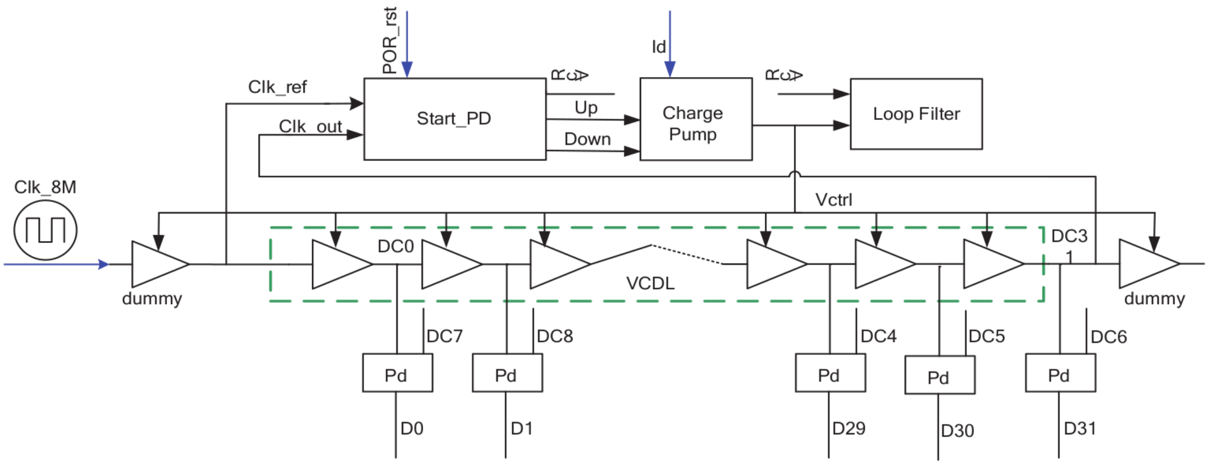
Figure 22 – Control proposed by CHEN, N. et al. [12].



Source: CHEN, N. et al. [12].

The DPWM is implemented through a combination of a 8 MHz clock, a 3-bit counter, and a voltage controlled delay line with 32 stages. If the frequency of the delay line is not equal to the 8 MHz clock, a phase discriminator and charge pump will vary the supply of the delay line until synchronization is achieved. Hence, a 1 MHz DPWM with 7-bit (3-bit from the counter and 5-bit from the delay line) is built from this combination.

Figure 23 – DPWM proposed by CHEN, N. et al. [12].



Source: CHEN, N. et al. [12].

A summary of the main parameters can be found in Table 6.

Table 6 – Summary of the main parameters in [12].

Parameter	Minimum	Typical	Maximum
Process	-	CMOS 180 nm	-
Input voltage	-	5.0 V	-
Output voltage	-	1.8 V	-
Output current	-	1.5 A	-
Inductor	-	2.2 µH	-
Load Capacitance	-	2.2 µF	-
ADC step	-	5 mV	-
DPWM step	-	0.4%	-
Switching Frequency	-	1 MHz	-
Output undershoot 1 A to 1.5 A	-	600 mV	-

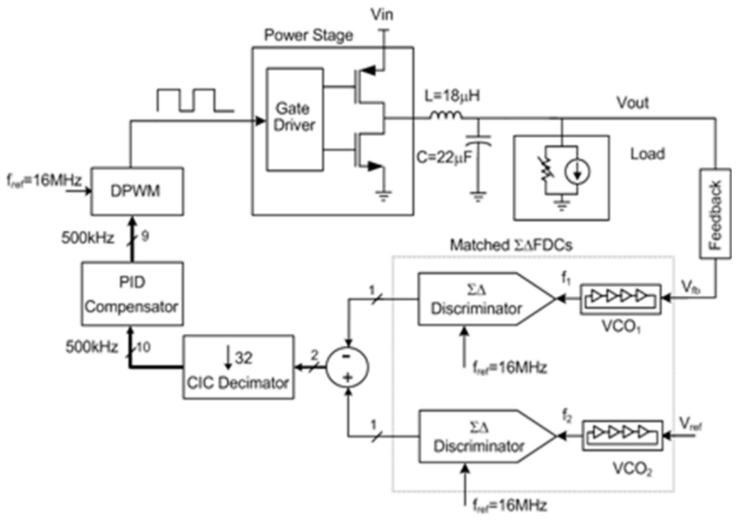
1.2.7 A 300 mA 14 mV-ripple digitally controlled buck converter using frequency domain ΔΣ ADC and hybrid PWM generator

Figure 24 depicts the block diagram of the DC-DC converter proposed by AHMAD, H. H. and BAKKALOGLU, B. [13].

The two ADCs (one for the voltage reference and one for the output voltage) are built from a Voltage Controlled Oscillator (VCO) followed by a ΔΣ frequency discriminator.

The DPWM has a 5-bit counter that is re-started at the beginning of each switching cycle. The low significant bits of the DPWM input selects one of the 16 taps of a current starved delay line to work as the counter clock signal. Once the most significant bits of the DPWM input matches the count, the inductor charge cycle ends.

Figure 24 – Architecture proposed by AHMAD, H. H. and BAKKALOGLU, B. [13].



Source: AHMAD, H. H. and BAKKALOGLU, B. [13].

A summary of the main parameters can be found in Table 7.

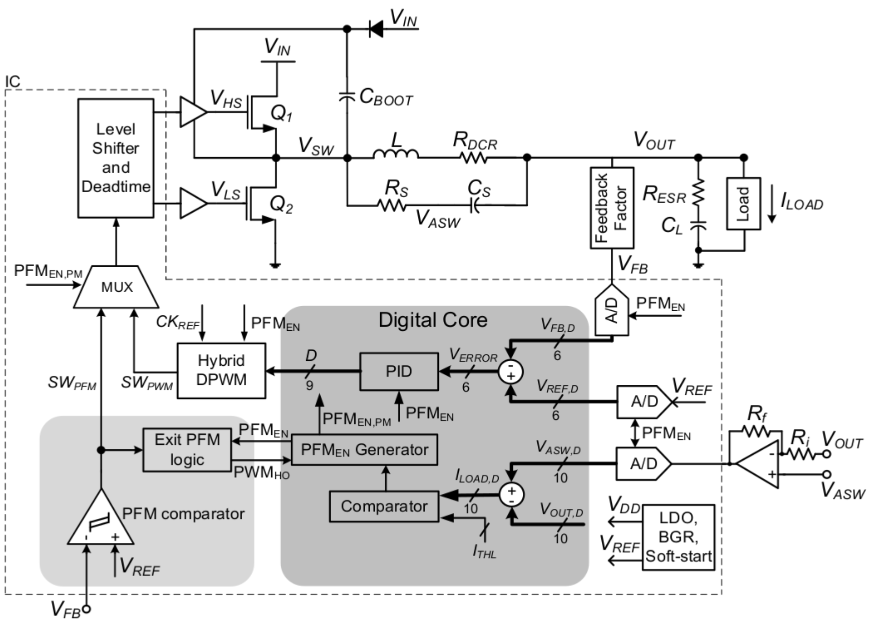
Table 7 – Summary of the main parameters in [13].

Parameter	Minimum	Typical	Maximum
Process	-	180 nm	-
Input voltage	2.5 V	3.3 V	5.5 V
Output voltage	-	1.8 V	-
Output current	0 A	-	300 mA
Inductor	-	18.8 µH	-
Load Capacitance	-	22.0 µF	-
Load Transient (0–300 mA)	-	130 mV	-
Switching Frequency	-	500 kHz	-
System clock	-	16 MHz	-
DPWM step	-	0.2%	
Current consumption (ADC)	-	220 µA	-
Current consumption (DPWM)	-	370 µA	-
Active area	-	2.2 mm^2	-

1.2.8 A Digitally Controlled DC-DC Buck Converter with Automatic Digital PFM to PWM Transition Scheme

The Architecture proposed by BEOHAR, N. et al. [14] is shown in Figure 25. Both the ADC and DPWM are based on [13]. According to BEOHAR, N. et al., the time based ADC structure has an inherent meta-stability problem due to the high frequency of the VCO and the possibility of timing violation.

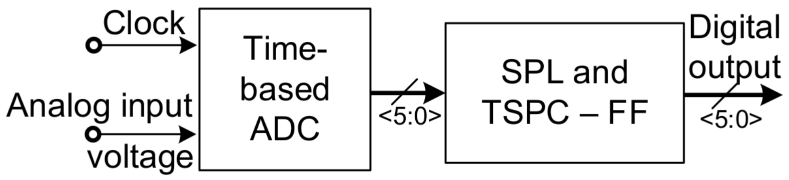
Figure 25 – Architecture proposed by BEOHAR, N. et al. [14].



Source: BEOHAR, N. et al. [14].

In order to address this issue, BEOHAR, N. et al. propose a Skip Prevention Logic (SPL) capable of detecting and blocking spurious signals caused by meta-stability (Figure 26).

Figure 26 – ADC proposed by BEOHAR, N. et al. [14].



Source: BEOHAR, N. et al. [14].

A summary of the main parameters can be found in Table 8.

Table 8 – Summary of the main parameters in [14].

Parameter	**Minimum**	**Typical**	**Maximum**
Process	-	IBM 180 nm	-
Input voltage	3.3 V	-	5.0 V
Output voltage	0.9 V	-	3.3 V
Output current	-	7.5 A	-
Inductor	-	800 nH	-
Load Capacitance	-	15 µF	-
Switching Frequency	-	1 MHz	-
Over/Undershoot	-	88.3 mV	-
Active area	-	1.61 mm^2	

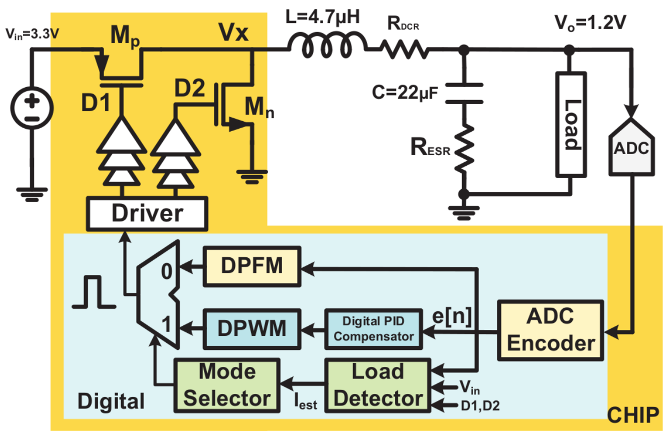
1.2.9 All-Digital Current-Sensorless Multi-Mode DC-DC Converter for Battery Powered Applications

The block diagram of the DC-DC converter proposed by YAO, G.-S. et al. [15] is shown in Figure 27. YAO, G.-S. et al. do not mention the ADC and DPWM architectures in the article. However, information in the block diagram and in the references suggest that an external ADC and a delay-locked loop based DPWM might have been used. The prototype was manufactured with a particular emphasis placed on the transition between PFM and PWM. A summary of the main parameters can be found in Table 9.

1.2.10 Digital Low Power Mode Control Technique for High Frequency Synchronous DC-DC Buck Converters

Figure 28 shows the DC-DC converter proposed by BOTTAMEDI, D. et al. [16]. The ADC and PWM control architecture are not mentioned in the article because the focus was the addition of a Low Power Mode (LPM) to the system. However, a prototype was manufactured and some parameters can be found in Table 10.

Figure 27 – Architecture proposed by YAO, G.-S. et al. [15].

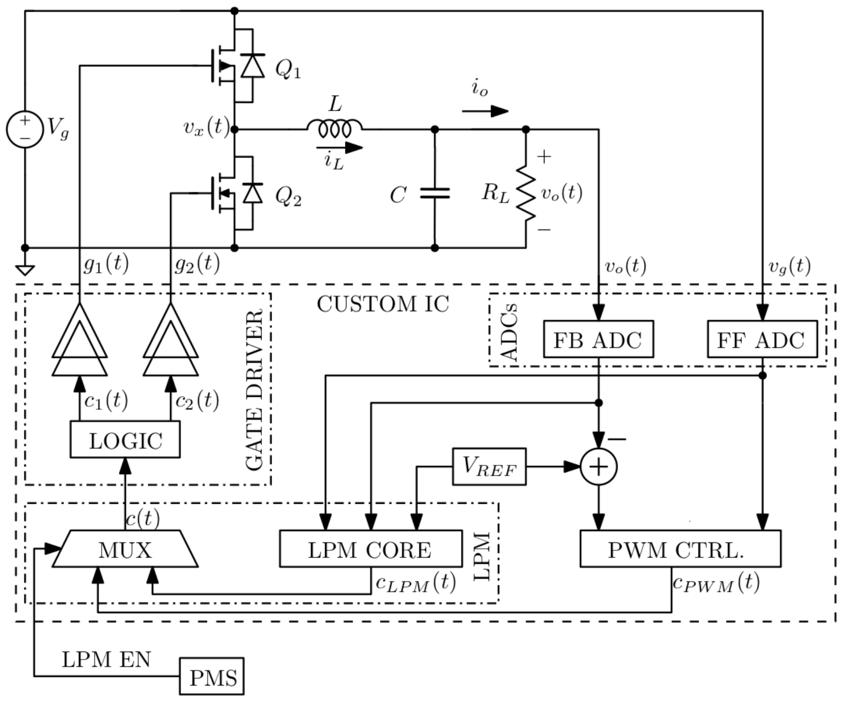


Source: YAO, G.-S. et al. [14].

Table 9 – Summary of the main parameters in [15].

Parameter	Minimum	Typical	Maximum
Process	-	TSMC 90 nm	-
Input voltage	-	3.3 V	-
Output voltage	-	1.2 V	-
Output current	10 mA	-	1 A
Inductor	-	4.7 µH	-
Load Capacitance	-	22 µF	-

Figure 28 – Architecture proposed by BOTTAMEDI, D. et al. [16].



Source: BOTTAMEDI, D. et al. [16].

Table 10 – Some parameters in [16].

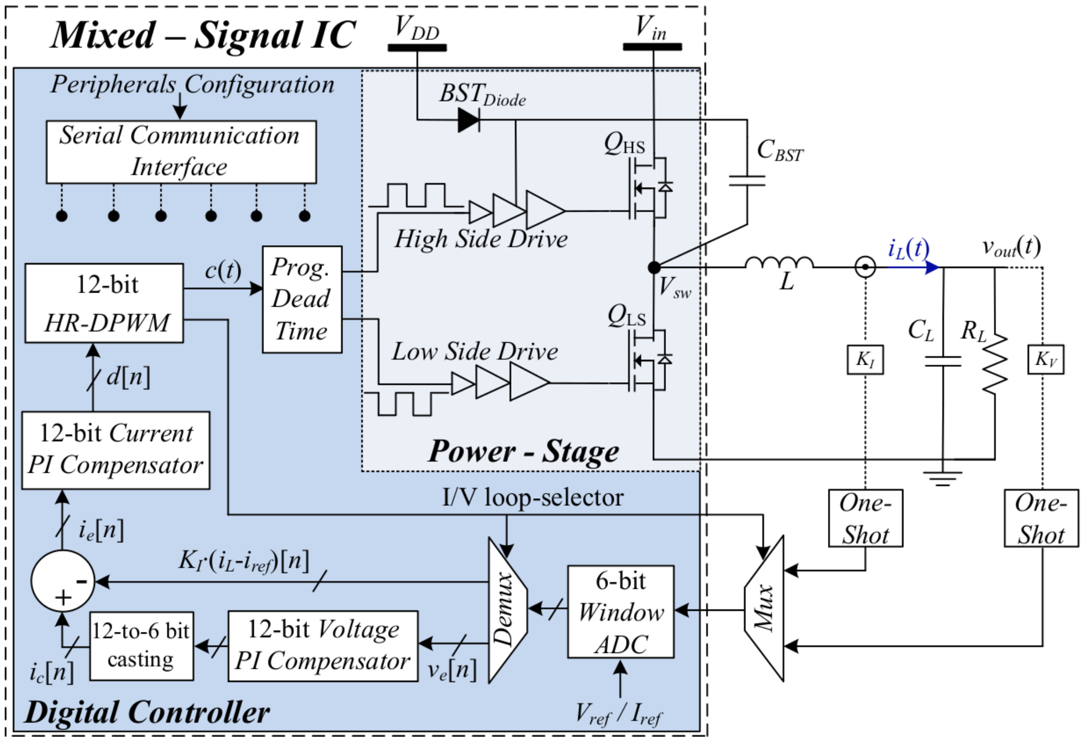
Parameter	Minimum	Typical	Maximum
Process	-	CMOS 28 nm	-
Input voltage	-	3.3 V	-
Output current	-	-	1.5 A
Switching Frequency	-	1.8 MHz	-

1.2.11 Fully-Integrated Digital Average Current-Mode Control 12 V-to-1.x V Voltage Regulator Module IC

The DC-DC converter proposed by VEKSLENDER, T. et al. [17] is illustrated in Figure 29. This converter operates in average current mode (ACM) and employs a single 6-bit delay line ADC to digitize both the inductor current and the output voltage (Figure 30). To convert the voltage and current information into a time-based signal, an external hardware is required.

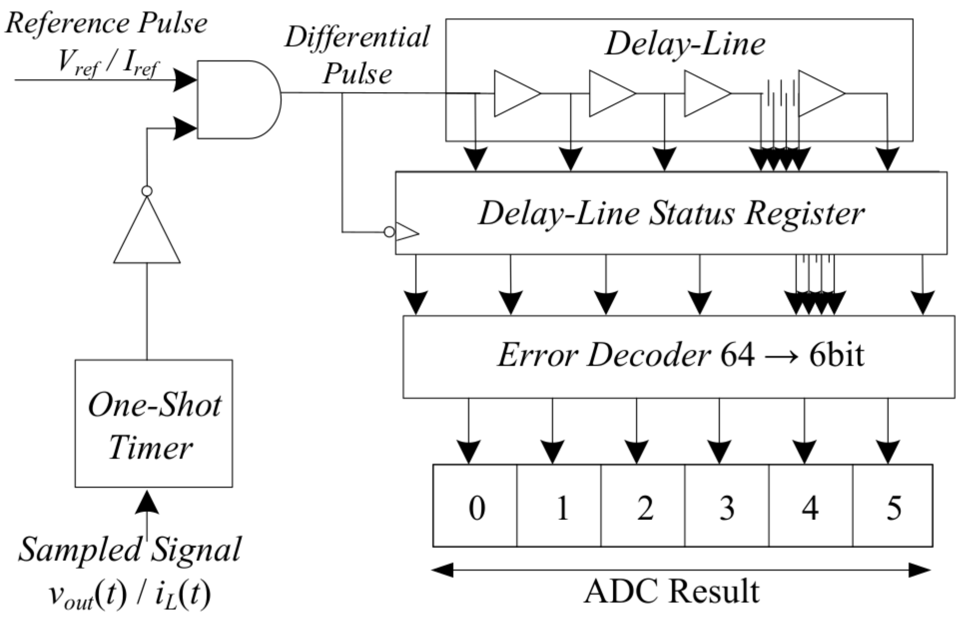
The DPWM (Figure 31) utilizes a mixed structure incorporating a counter to divide the main clock and a delay line for fine resolution.

Figure 29 – Architecture proposed by VEKSLENDER, T. et al. [17].



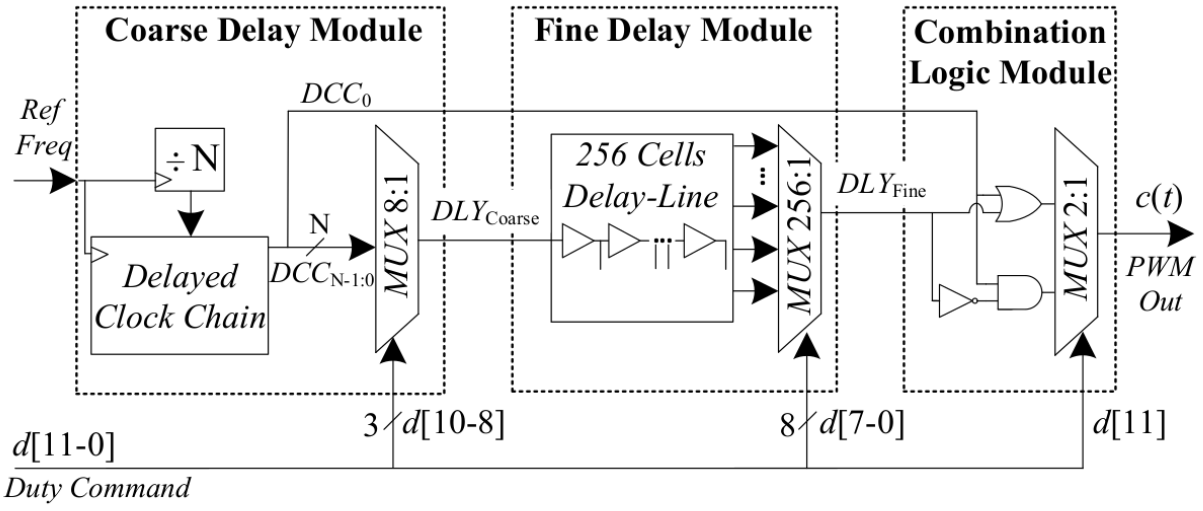
Source: VEKSLENDER, T. et al. [17].

Figure 30 – ADC proposed by VEKSLENDER, T. et al. [17].



Source: VEKSLENDER, T. et al. [17].

Figure 31 – DPWM proposed by VEKSLENDER, T. et al. [17].



Source: VEKSLENDER, T. et al. [17].

A main contribution of VEKSLENDER, T. et al. is the implementation of the ADC and DPWM using only standard cells from the technology and Hardware Description Language (HDL). A summary of the main parameters can be found in Table 11.

Table 11 – Summary of the main parameters in [17].

Parameter	Minimum	Typical	Maximum
Process	-	180 nm 5 V CMOS process	-
Input voltage	-	12 V	-
Output voltage	-	1.5 V	-
Output current	-	1.5 A	6.6 A
Inductor	-	2.2 µH	-
Load Capacitance	-	50 µF	-
Load Transient (1.5–3.0 A)	-	40 mV	-
Switching Frequency	-	1.25 MHz	-
Total active area	-	4.4 mm^2	-
Active area (DPWM)	-	0.03 mm^2	-
Active area (ADC)	-	0.022 mm^2	-
Iq (Digital Core)	-	58 µA/MHz	-

1.2.12 Time-Based PWM Controller for Fully Integrated High Speed Switching DC-DC Converters – An Alternative to Conventional Analog and Digital Controllers

KHAN, Q. A. et al. [18] present a review of a non-conventional PID digital control technique with time-based processing, as illustrated in Figure 32.

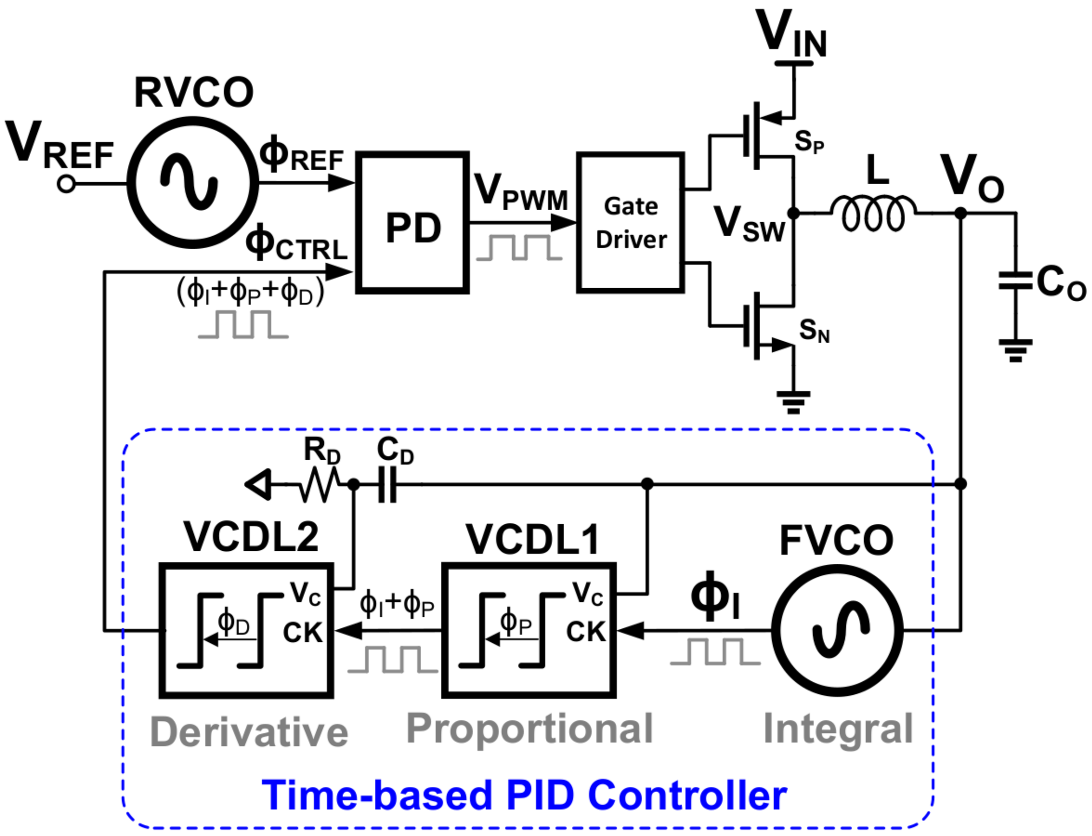
One VCO connected to the output voltage and the other to the voltage reference codify the voltage signals in frequency. A phase detector will then generate the integral of the difference in frequencies as represented by the known relationship given by Equation 1.40. Proportional and derivative elements are implemented with voltage-controlled delay lines. The main advantages of this architecture are the absence of quantization noise and the elimination of the need for a DPWM, given that the output inherently takes the form of a PWM signal.

$$w = \frac{d\phi}{dt} \implies \phi = \int w \cdot t \tag{1.40}$$

KHAN, Q. A. et al. highlight the potential impact of mismatch between the two VCOs on the output voltage. To mitigate this, the authors propose a strategy involving the simultaneous operation of both oscillators with the voltage reference, followed by tuning one oscillator until they achieve frequency lock.

Another challenge discussed is cycle slip arising from the phase detector's limited detection range between 0 and 2π. This issue, according to KHAN, Q. A. et al., can be mitigated by the usage of a slip detector.

Figure 32 – Architecture proposed by KHAN, Q. A. et al. [18].



Source: KHAN, Q. A. et al. [18].

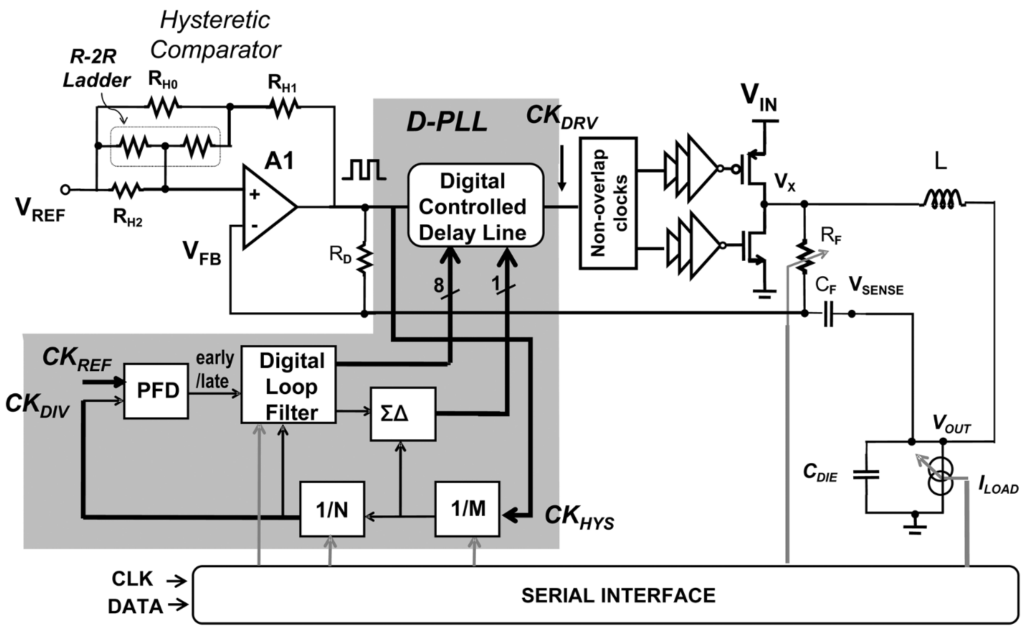
1.2.13 A 90–240 MHz hysteretic controlled DC-DC buck converter with digital PLL frequency locking

The system introduced by Li et al. [19] is shown in Figure 33. This system consists in a hysteretic buck that is synchronized with a reference clock through the utilization of a digital PLL.

In the typical operation of a hysteretic buck, the process of charging the output voltage is initiated only when it descends below a predefined threshold. Additionally, the PMOS switch remains active either for a predetermined interval or until the inductor current reaches a specific threshold. This operation leads to an inherent variability in the frequency of hysteretic bucks, directly influenced by the load current.

The synchronization mechanism involves the comparison between the frequency of the hysteretic buck and the reference clock. Subsequently, a voltage-controlled delay line is finely adjusted to achieve synchronization. With this combination, the architecture proposed by LI. et al. can lock in frequency with the reference clock for a wide range of load currents.

Figure 33 – Architecture proposed by LI, P. et al. [19].



Source: Li et al. [19].

A summary of the main parameters can be found in Table 12.

Table 12 – Summary of the main parameters in [19].

Parameter	Minimum	Typical	Maximum
Process	-	CMOS 130 nm	-
Input voltage	-	1.3 V	-
Output voltage	0.4 V	-	0.96 V
Output current	-	-	240 mA
Inductor	-	8.2 nH	-
Load Capacitance	-	20 nF	-
Switching Frequency	90 MHz	-	240 MHz
Undershoot @ 0.12A step	-	100 mV	-
Efficiency	-	-	80 %
Active area	-	0.42 mm ^{2}	-

1.2.14 Analysis of the reviewed literature

As elucidated in the previous chapters, the structure of most digital control loops for DC-DC converters is composed of ADC, digital logic, and DPWM. Nonetheless, certain architectures, as proposed by [18] and [19], deviate from this conventional approach.

The system proposed in [18] cannot be synchronized with a reference clock, while the findings from [19], although insightful, highlight a tendency toward high switching frequencies and increased ripple. Focusing exclusively on architectures adhering to the ADC, digital logic, and DPWM structure, some observations emerge:

1.2.14.1 Analog-to-Digital Converter (ADC)

Various ADC architectures, including a 1-bit comparator ADC [8, 11], charge redistribution ADC [9], ring oscillator ADC [3], flash ADC [10], delay line ADC [12, 14, 17], and VCO-based ADC [13], have been documented.

The 1-bit comparator ADC introduces inherent high quantization noise, necessitating a slow control loop to filter out some of the noise.

On the other hand, the charge redistribution ADC can be constructed with higher resolution. However, the current consumption of the voltage reference buffers typically reaches hundreds of micro-amperes for sample rates in the range of Mega Samples Per Second (MSPS). An exception is noted in [9], where the current consumption is reported as low as 18 µA.

Delay line ADC, VCO-based ADC, and ring oscillator ADC operate on the common principle of converting voltage information into time delay or frequency. The method employed to extract digital values from time information may utilize $\Delta\Sigma$ discriminators, counters, or latches. Time-based ADCs are prevalent in the reviewed literature because they can be built to operate within a narrow window of interest and have low current consumption.

1.2.14.2 Digital Pulse Width Modulation (DPWM)

DPWM architectures encompass digital counters at elevated frequencies [8] [9], ring oscillators [3], analog capacitor-based designs [10], voltage-controlled delay lines [12], and hybrid configurations [13].

Most of the challenge regarding the DPWM revolves around limiting cycle oscillations. If the voltage step size caused by the quantization of the DPWM is higher than the scaled version of the ADC step size (the ADC step size divided by the ratio of a possible resistor divider between the output voltage and the ADC), the control will not be able to find a steady state value and sub-harmonic oscillations will occur [20]. When reducing DPWM step size proves impractical, authors suggest incorporation of a technique called dithering [3, 20]. Dithering involves a selective addition of one to the DPWM input to introduce artificial resolution and reduce the step size. This modulation can occur randomly, with the probability of adding one at a given clock cycle being equal to the value represented by the extra bits necessary or, alternatively, the addition can be performed cyclically, as if another PWM (with a frequency lower than the switching frequency) is employed control the addition of one to the DPWM input.

None of the proposed DPWM architectures seems to incorporate mechanisms for dynamic duty cycle adjustment according to the input voltage fluctuations. This could potentially lead to low line transient performance for converters operating in voltage mode.

To mitigate this problem and enhance line regulation, digitizing both the input voltage and output voltage is suggested in [9, 16].

1.2.14.3 Digital logic

Most designs employ some form of digital PID control. However, there are a couple of exceptions as the adaptive on-time method suggested in [11] and the slow bi-directional shift register proposed by [8]. It is worth mentioning that these alternative approaches face a drawback of slower transient response. On the other hand, the challenges associated with digital PID revolve around efficiently implementing the necessary multiplication hardware to mitigate current consumption and minimize area footprint.

1.2.14.4 Summary of the main parameters

Not all authors provide details about the structure of the ADC and the DPWM; most focus on the overall system performance of the buck. For the authors who reported parameters for the ADC and DPWM, Table 13 presents a summary of these parameters.

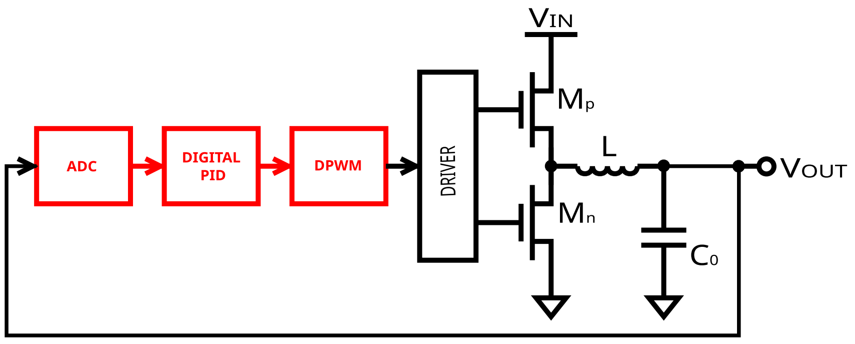
Table 13 – Comparison between the main parameters.

Parameter	[3]	[13]	[9]	[10]	[17]	[12]	[14]
Year	2004	2010	2011	2011	2017	2019	2021
Input	2.8 – 5.5 V	2.5 – 5.5 V	2.3 – 4.8 V	2.8 – 4.2 V	12 V	5.0 V	3.3 – 5 V
Output	1.0 – 1.8 V	1.8 V	0.6 – 1.35 V	0.4 – 1.2 V	1.5 V	1.8 V	0.9 – 3.3 V
Technology	0.25 µm	0.18 µm	65 nm	45 nm	0.18 µm	0.18 µm	0.18 µm
Load	0.1 – 400 mA	0 – 300 mA	0 – 1 A	0.02 – 100 mA	6.6 A	1.5 A	7.5 A
Inductor	10 µH	18.8 µH	470 nH	10 µH	2.2 µH	2.2 µH	800 nH
Capacitor	47 µF	22 µF	10 µF	2 µF	50 µF	2.2 µF	15 µF
ADC topology	Ring-Osc	VCO	SAR	Flash	Delay Line	Delay Line	Time based
ADC step	16 mV	-	10 mV	50 mV	-	5 mV	-
ADC area	0.15 mm^2	-	0.024 mm^2	-	0.022 mm^2	-	-
ADC current	37 µA	220 µA	18 µA	-	-	-	-
ADC resolution	-	5-bit (window)	4-bit (window)	4-bit (window)	6-bit (window)	6-bit (window)	6-bit (window)
DPWM topology	Ring-MUX	Counter + DLL	Counter	I-C DAC	Counter + Delay line	Counter + DLL	Counter + DLL
DPWM step	3.125%	0.195%	2.083%	1%	0.0244%	0.39%	0.195%
DPWM area	-	-	0.0015 mm^2	-	0.03 mm^2	-	-
DPWM current	-	370 µA	82.5 µA	6 µA	-	-	-
DPWM resolution	5-bit + Dither	9-bit	48 levels + Dither	5-bit + Dither	12-bit	8-bit	9-bit
Total area (control)	-	-	0.038 mm^2	0.07 mm^2	0.16 mm^2	-	-
Line transient	-	-	10 mV (600 mV in 10 µs)	-	-	-	-
Load transient	25 mV (50–150 mA)	130 mV (0–300 mA)	20 mV (0–200 mA in 500 ns)	10 mV (10–50 mA in 25 µs)	40 mV (1.5–3.0 A)	600 mV (1–1.5 A)	88.3 mV (500 mA in 600 ns)
Reference clock	0.5 – 1.5 MHz	16 MHz	307.2 MHz	2 MHz	20 MHz	8 MHz	-
Switching frequency	0.5 – 1.5 MHz	500 kHz	6.4 MHz	2 MHz	1.25 MHz	1 MHz	1 MHz

2 High Level Design

A buck converter is composed of power switches, an inductor, a capacitor, a driver, and a control block, as depicted in Figure 1. The idea behind a digitally controlled buck is to replace the conventional analog control with a digital control. In order to accomplish this goal, analog-to-digital conversion and digital-to-PWM conversion must also be carried out. Consequently, the block diagram of a digitally controlled buck takes the form depicted in Figure 34. The architecture operates in CCM with PWM control in voltage mode, requiring only one ADC. The choice of CCM with PWM control will become more apparent during the development of the DPWM implementation.

Figure 34 – Block diagram of a digitally controlled DC-DC.



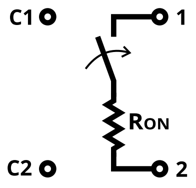
Source: The author.

The goal of this work was to develop the control, which consist in three blocks highlighted in red: ADC, digital PID, and DPWM. However, before designing these blocks, it is necessary to establish their performance requirements from the overall system specifications. This chapter will explain this development process, starting with the modeling of the power switches, followed by the selection of inductor and a Small Signal Analysis to stablish the value of the capacitor and the remaining requirements.

2.1 Model of the power switches and drivers

Assuming full integration of the converter, the power switches will be implemented with MOS transistors. From a high-level perspective, a MOS transistor functioning as a switch can be modeled as depicted in Figure 35. Here, R_{on} denotes the resistance between drain and source. Nodes C1 and C2 serve as control, while 1 and 2 represent output nodes. The parasitic capacitance between drain and source C_{ds} and between gate and source C_{gs} were omitted. C_{gs} will influence the design of the drivers and the drivers may add a substantial delay that affects the control loop (as will be shown in Figure 42). However, since the focus of this work is the control, C_{gs} can be neglected replacing it by the effect of the delay in the control loop.

Figure 35 – Power switches high level model.



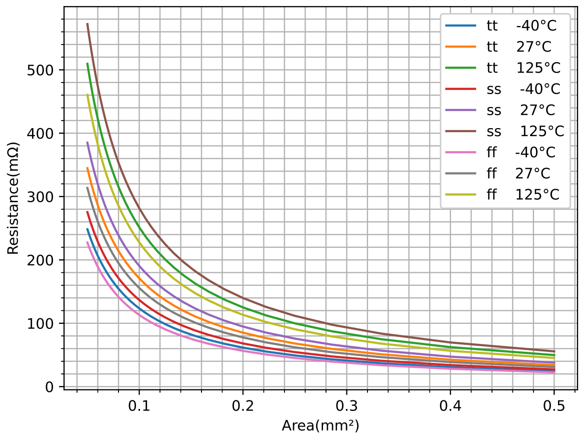
Source: The author.

The technology selected for this study has 5 V transistors (nfet50x and pfet50x) suitable for implementing power switches. Figures 36 and 37 depict simulations of the resistance of these transistors as a function of the total area for typical, slow, and fast corners (including temperature variation). The length was set to the lowest possible value, the gate-to-source voltage equal was set to 2.7 V (the minimum input voltage) and the DC simulation involved applying a 200 mA DC current across the drain and source terminals, followed by probing the voltage difference between them.

The work described in [13] also utilizes a CMOS 180 nm technology with a maximum load of 300 mA. The author reports a total area of 0.9 mm^2 for the analog part, including the power transistors. Assuming a similar area, resistances lower than 200 mΩ are feasible for the nfet50x and pfet50x transistors.

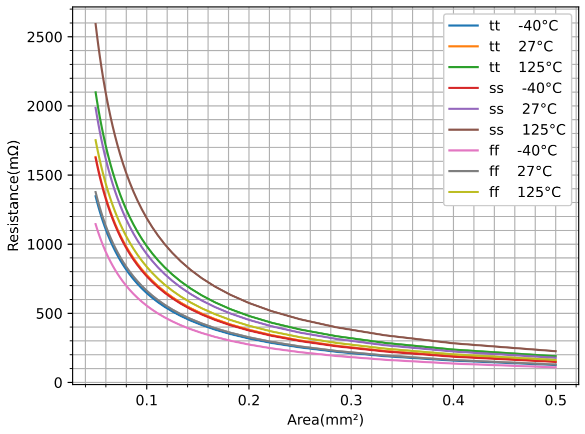
Given the focus of this work, a Verilog-A model of the power switches and drivers has been developed (see Annex A). The purpose of this model is to aid the verification of the control integrated with the entire system. While the resistance of the NMOS and PMOS switches were set to 300 mΩ by default, these values can be adjusted via the available parameters in the model. However, 300 mΩ seems to be a conservative choice allowing some headroom for parasitic resistance of bonding wires and routing.

Figure 36 – nfet50x resistance.



Source: The author.

Figure 37 – pfet50x resistance.



Source: The author.

2.2 Inductor selection

The inductance calculation can be done by means of the ratio between the inductor current ripple and the DC inductor current at maximum load, as expressed in Equation 2.1.

$$r = \frac{\Delta_{I_L}}{max\left(\overline{I_L}\right)} \tag{2.1}$$

Typically, the ratio r is selected to be in the range of 0.25 to 0.5 at the maximum rated load current [21]. Reducing r below 0.25 may result in an impractically large inductor size, while increasing it above 0.5 can lead to higher losses due to the elevated RMS value of the inductor current. The specific choice depends on the design, but in the scope of this work, the typical range for r was selected. Substituting Equation 1.12 into Equation 2.1 and using typical input voltage (3.7 V), output voltage (1.8 V) and switching frequency (2 MHz) results in:

$$r = \frac{D \cdot T_S \cdot \left(V_{IN} - \overline{V_{OUT}}\right)}{max\left(\overline{I_L}\right) \cdot L} = \frac{\frac{1.8}{3.7} \cdot 500 \cdot 10^{-9} \cdot (3.7 - 1.8)}{200 \cdot 10^{-3} \cdot L} \approx \frac{2.3 \cdot 10^{-6}}{L} \tag{2.2}$$

Which implies,

$$4.6\mu H \leq L \leq 9.2\mu H \tag{2.3}$$

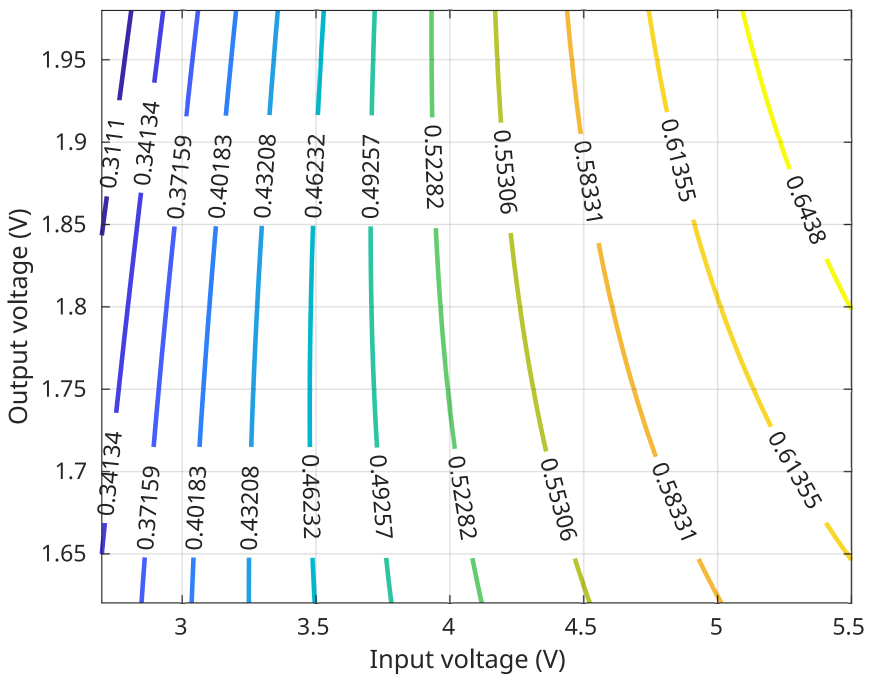
Table 14 displays the characteristics of some commercially available inductors within this range. Those with smaller physical sizes are listed in the top rows, and all have a 20% inductance tolerance. The Table only includes inductors from Murata as the manufacturer offers the SimSurfing® tool, greatly facilitating the extraction of simplified Spice models under various DC bias conditions. Furthermore, Murata is a manufacturer renowned for being the recommended choice for several reference designs.

Table 14 – Summary of the commercial inductor parameters from Murata.

Manufacture Reference	**Inductance**	**Maximum DC Resistance**	**Maximum DC Current**	**Package**
LQM21PN4R7MGDS	4.7 µH	290 mΩ	750 mA	805
DFE201610E-4R7M	4.7 µH	288 mΩ	1.1 A	806
DFE252012F-4R7M	4.7 µH	190 mΩ	1.5 A	1008
DFE252012F-6R8M	6.8 µH	330 mΩ	1.2 A	1008
DFE252012F-8R2M	8.2 µH	410 mΩ	1.1 A	1008

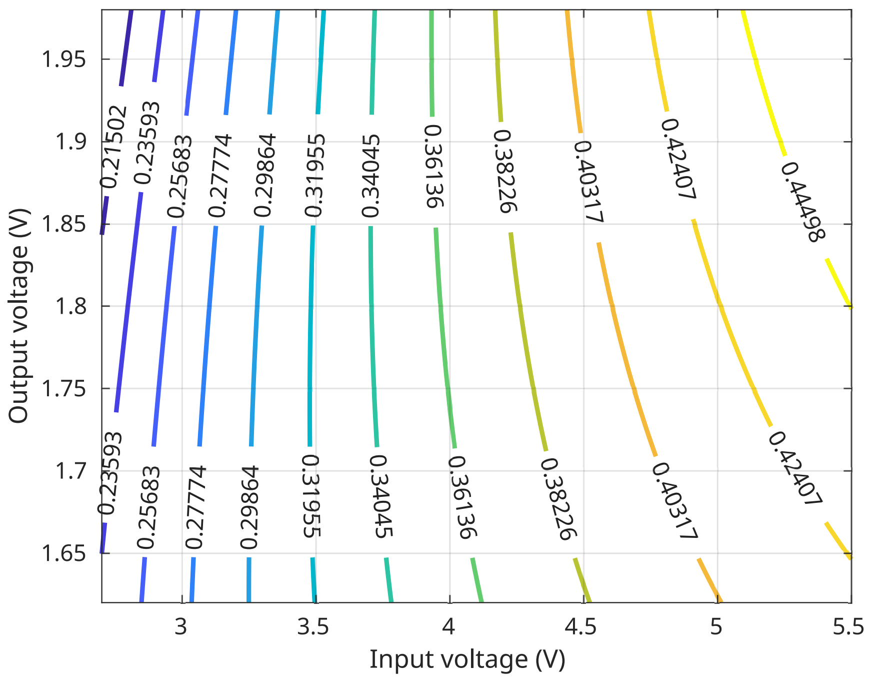
Since the ratio r was calculated with typical values of input and output voltages, it is expected that this ratio will fall outside of the 0.25-0.5 range for some combination of valid input and output voltages. Figures 38, 39, and 40 depicts this dependency for the different inductance values.

Figure 38 – Contour plot of ratio r for L = 4.7 µH and different combinations of input and output voltages.



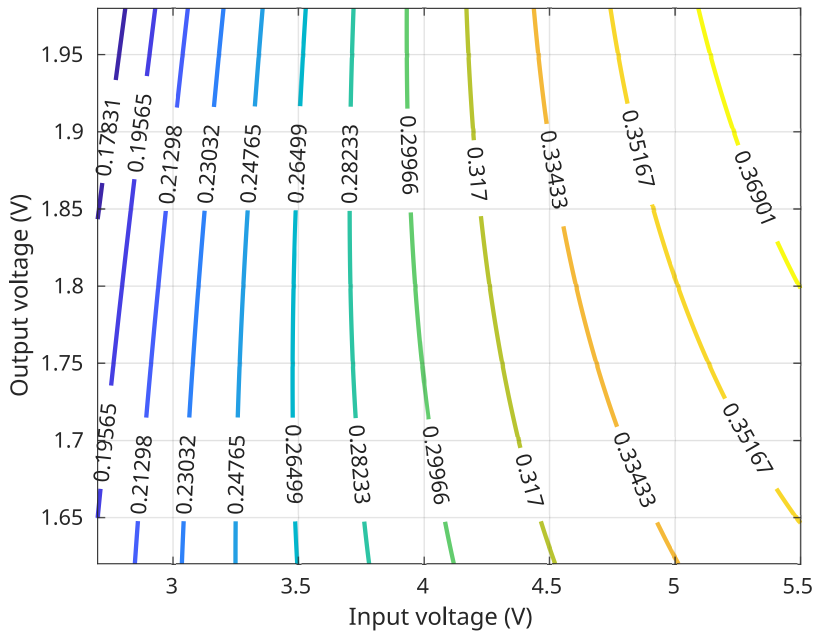
Source: The author.

Figure 39 – Contour plot of ratio r for L = 6.8 µH and different combinations of input and output voltages.



Source: The author.

Figure 40 – Contour plot of ratio r for L = 8.2 µH and different combinations of input and output voltages.



Source: The author.

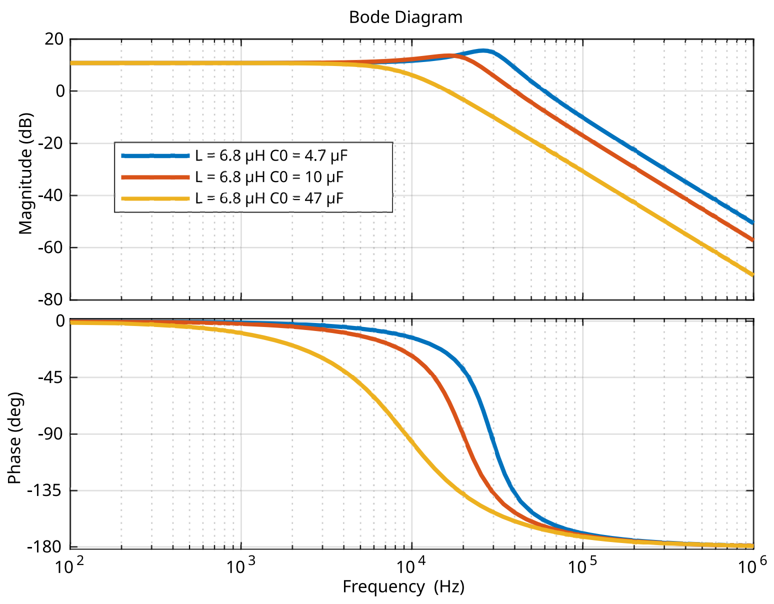
An inductor of 4.7 µH ensures operation within the desirable range when powered by the battery, exceeding the range during charging. Opting for a 6.8 µH inductor maintains operation within the range for the majority of cases, except when the battery is nearly depleted. On the other hand, selecting an 8.2 µH inductor keeps the operation within the range except when the battery is significantly discharged. Various considerations can be weighed in favor of each option. For instance, if minimizing Printed Circuit Board (PCB) footprint is a priority, the 4.7 µH inductor would be the preferable choice, since it is the only one available in a 805 package. Alternatively, if maintaining operation within the 0.25-0.5 range for most scenarios is desirable, the 6.8 µH inductor will be the more suitable option. Therefore, both the 4.7 µH and 6.8 µH inductors are feasible choices. However, taking the manufacturing tolerances also into account, the 6.8 µH might be a better choice because it falls approximately in the middle of the desirable range. In this work, a 6.8 µH inductor is assumed.

2.3 Small signal analysis, compensation, and transient response

This section begins with the presentation of the small signal model of the system and the analysis of the controller. Subsequently, the methodology for approximating the transient behavior and capacitor selection are discussed.

The small signal analysis is initiated by generating a Bode plot of Equation 1.27 for typical input and output voltage, as depicted in Figure 41. Throughout the analysis presented in this section, unless otherwise specified, the load resistance is assumed to be 9 Ω, power switch resistances are set to 300 mΩ, inductor DC resistance is 330 mΩ, and the capacitor is considered to have negligible Equivalent Series Resistance (R_{ESR}).

Figure 41 – Bode plot of $H(s)$.



Source: The author.

Recall that Equation 1.27 represents $H(s) = \frac{\overline{v_{out}(s)}}{d(s)}$, the ratio of small signal variation in the average output voltage to small variations in the duty cycle. The controller will then maintain the output voltage within specified limits by sensing variations in the output voltage and adjusting the duty cycle accordingly.

The Bode plot of $H(s)$ highlights some challenges in the controller design. Firstly, $H(s)$ exhibits a low DC gain, resulting in poor steady-state error in closed-loop. Secondly, $H(s)$ has two imaginary poles, leading to a sharp decrease in phase. Although it is possible to increase the gain using a Proportional and Integral (PI) controller, this would lead to a slow response due to the necessity of limiting the bandwidth below the resonant frequency of $H(s)$. If both low steady-state error and higher bandwidth are required, it becomes necessary to add two zeros to the system besides the pole of the PI controller [22], resulting in a PID controller.

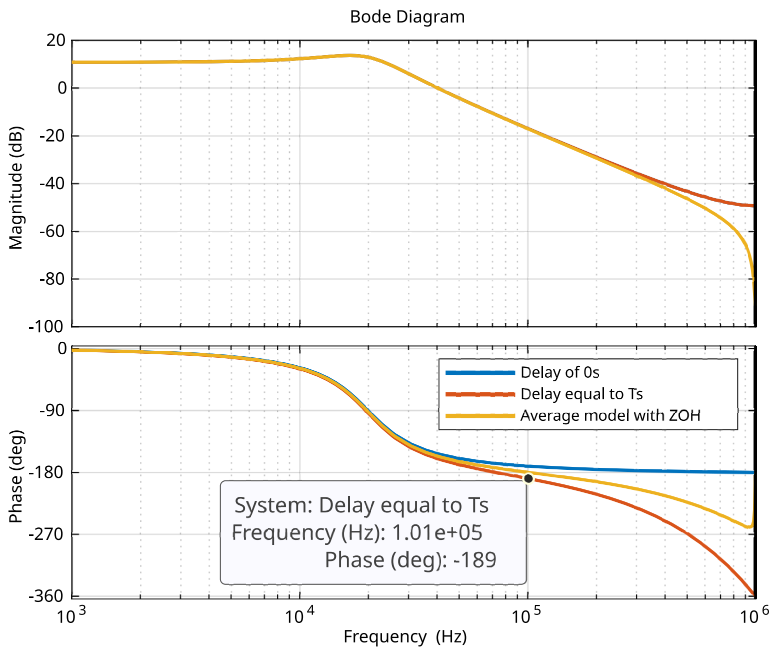
Since the error signal will be sampled and the controller implemented digitally, the transfer function of the PID controller can be expressed in discrete-time or z-domain form, as shown in Equation 2.4 [22]. Here, e represents the difference between the reference voltage and the a scaled version of the output voltage, while K_0, K_1, and K_2 are constants.

$$Hc(z) = \frac{d(z)}{e(z)} = \frac{K_0 + K_1 \cdot z^{-1} + K_2 \cdot z^{-2}}{1 - z^{-1}} \tag{2.4}$$

Before calculating the value of the PID constants, it is essential to derive $H(z)$, which is the discrete-time equivalent of $H(s)$. As discussed in Section 1.1.3.3, a precise discrete-time model valid across the entire frequency range is presented in [2]. This model incorporates the delay between the sampling moment and the observation of the control effort at the output and, given the dependence of this delay on the implementation, $H(z)$ will vary accordingly. However, an approximation can be achieved using the discrete version of the average model (Equation 1.37).

The difference between $H(z)$ obtained with the average model and the method described in [2] is illustrated in Figure 42. The conditions remain the same as those for Figure 41, except for the output capacitance (C_O), which was fixed at 10 µF. There is an approximate ±9° phase difference at 100 kHz, increasing to approximately ±18° at 200 kHz. Therefore, a good approximation can be achieved by limiting the bandwidth to less than 10% of the switching frequency.

Figure 42 – Comparison between different methods of obtaining $H(z)$.

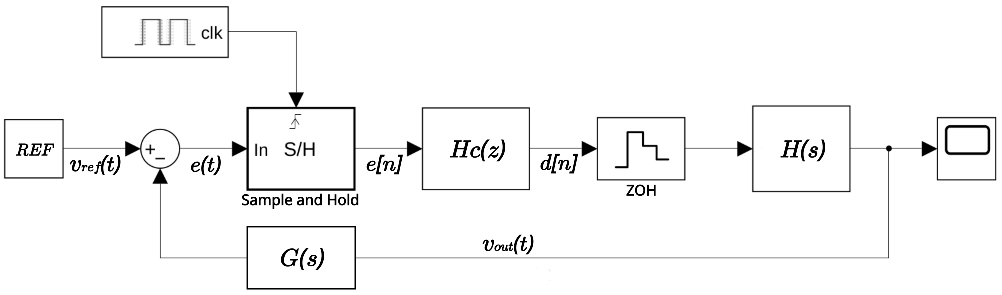


Source: The author.

Using the Simulink® software, small signal block diagram of the average model can be drawn as shown in Figure 43. It is important to note that while the filter blocks are represented by their transfer functions in the s and z domains, their interfaces are in the time (t) and discrete time (n) domains, respectively.

Additionally, $G(s)$ represents the scaling factor between the output voltage and the voltage reference. For instance, if a voltage divider is used to reduce the output voltage to a level comparable to the voltage reference, $G(s)$ will represent the transfer function of this voltage divider. $G(s)$ equal to 1 is used in rest of the analysis. This should not be a problem as long as $G(s)$ does not introduce significant phase lag at the crossover frequency and it can be corrected later by dividing the PID constants by $G(0)$ for values different than one.

Figure 43 – Average model small signal block diagram using Simulink®.



Source: The author.

2.3.1 PID tuning

The PID tuning involves adjusting the constants K_0, K_1, and K_2 to achieve the desired performance. Although parameters like rise time, settling time, and overshoot are not explicitly defined in this work, minimizing overshoot is desirable to achieve the specified $\pm 2\%$ accuracy. To mitigate delays and implementation dependencies, the target bandwidth is set to 100 kHz. Additionally, a minimum phase margin of 54° (45° minimum plus 9° for delay inaccuracies, see Figure 42) is chosen as a compromise between settling time and overshoot.

This study presents two tuning methods. The first method is inspired by the procedure commonly employed for tuning type-III compensator in analog control [23]. It involves placing the first pole at a low frequency and positioning the zeros at the resonant frequency (f_0) of $H(s)$.

Observe that Equation 2.4 has only one pole and it is already set at 0 Hz. Therefore, it is only necessary to place the zeros at the resonant frequency by making:

$$\frac{K_1}{K_0} = -2 \cdot e^{-2 \cdot \pi \cdot f_0} \tag{2.5}$$

and,

$$\frac{K_2}{K_0} = e^{-4 \cdot \pi \cdot f_0} \tag{2.6}$$

K_0 can be adjusted to obtain the required bandwidth and phase margin. Table 15 shows the values of the PID constants and the performance of system $Hc(z) \cdot H(z)$ when K_0 is adjusted for a bandwidth of 100 kHz.

Table 15 – Summary of the PID tuning using the first method.

Capacitor	K_0	K_1	K_2	**Settling time**	**Overshoot**	**Phase Margin**
4.7 µF	10.1924	-18.6061	8.4913	24 µs	18.48%	50.7°
10 µF	23.0065	-43.2213	20.2995	28 µs	12.94%	58.9°
47 µF	111.2008	-216.0725	104.9618	8.5 µs	2.5%	69.8°

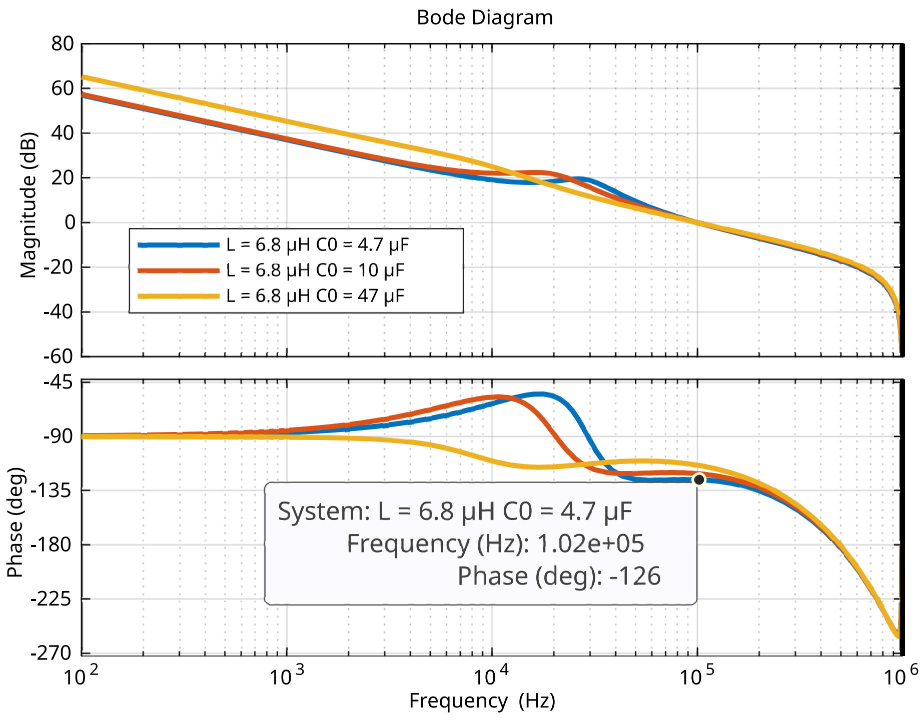
The second method involves using an algorithm to tune the PID controller. Table 16 displays the values of the PID constants and the performance of system $Hc(z) \cdot H(z)$ obtained using the pidtune function available in MATLAB® software.

Table 16 – Summary of the PID tuning using the MATLAB® pidtune function.

Capacitor	K_0	K_1	K_2	**Settling time**	**Overshoot**	**Phase Margin**
4.7 µF	10.3655	-19.1072	8.8053	28 µs	14.4%	54°
10 µF	23.0129	-43.251	20.3046	34 µs	12.1%	59.0°
47 µF	112.2467	-215.8838	103.8022	15.5 µs	7.4%	66°

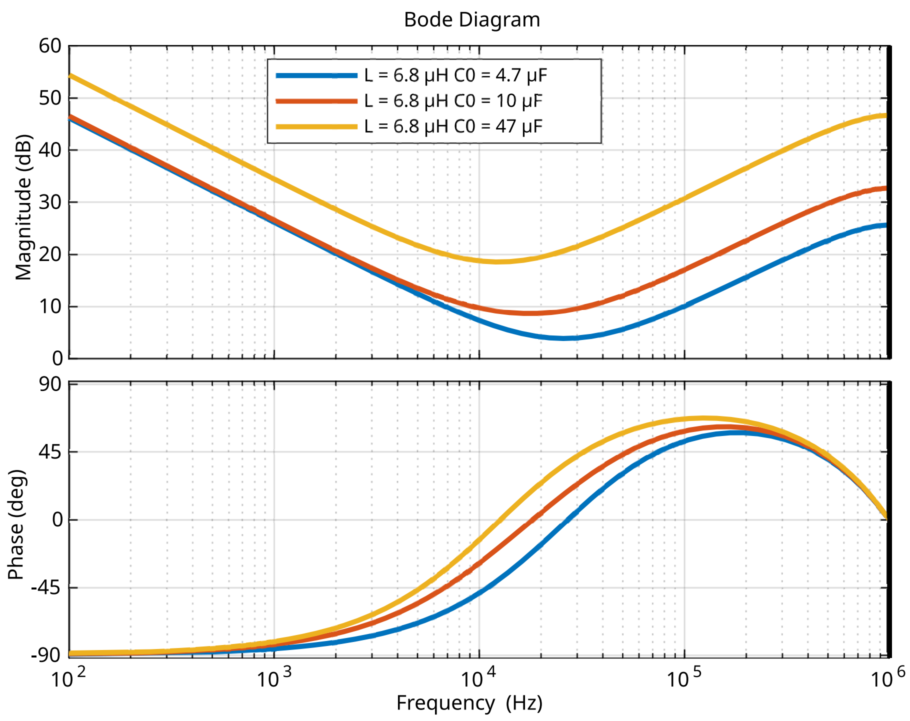
Although the two methods yield similar results, the pidtune function appears to do a better job at optimizing both the overshoot and the phase margin. As a result, the values obtained through this function will serve as a reference for this work. Additionally, Figures 44 and 45 depict the Bode plots of the system and controller, respectively.

Figure 44 – Bode plot of $Hc(z) \cdot H(z)$ for the PID tuned using the MATLAB® pidtune function.



Source: The author.

Figure 45 – Bode plot of $Hc(z)$ for the PID tuned using the MATLAB® pidtune function.



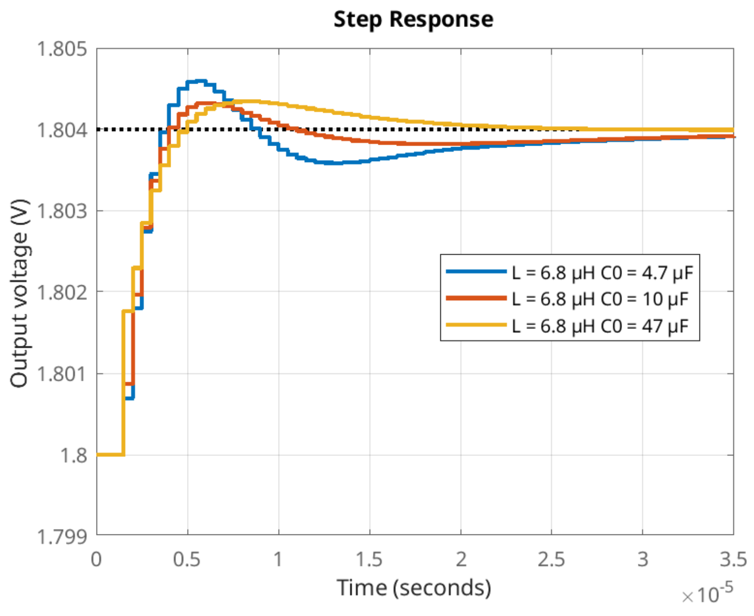
Source: The author.

2.3.2 Voltage reference step response

Up to this point, the transfer function of $H(s)$, and consequently $H(z)$, has been computed for typical output voltage. However, Equations 1.10, 1.27, 1.29, 1.31, 1.32, 1.33, 1.34, 1.35, and 1.36 show that the small signal model remains relatively consistent across the entire output voltage range when R_P and R_N have the same value or their difference are negligible when compared to $R_L + R$. Additionally, it is worth noting that $H(s = 0)$ or $H(z = 1)$ also provides the DC relationship between the average output voltage and the duty cycle under the same conditions, which indicates that the model can also be used for large signals.

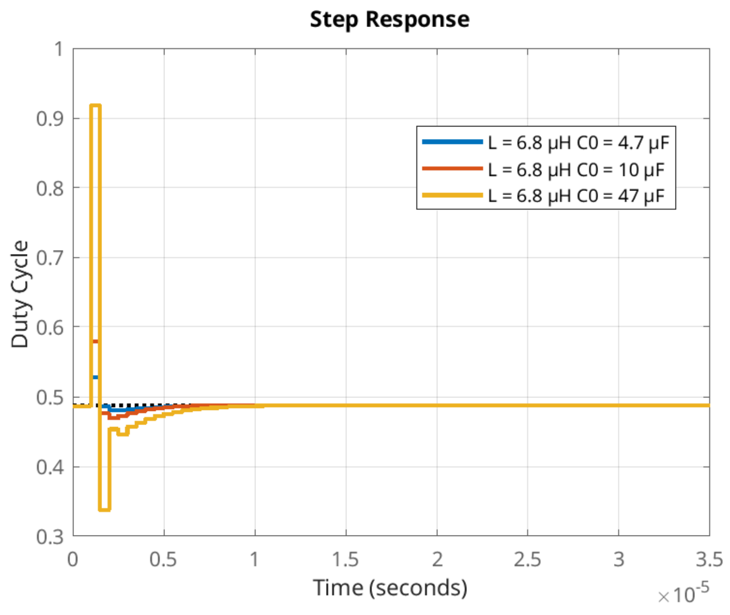
Figure 46 illustrates the closed-loop step response of $Hc(z) \cdot H(z)$, while Figure 47 presents the corresponding duty cycle. There is nothing particularly significant about the 4 mV step depicted in Figure 47. Because the model is linear, a step with a different value will scale the response proportionally. This value was selected for the plot because, with a 47 µF capacitor, it results in a duty cycle variation of 45% approximately, which covers nearly the entire practical range of duty cycle variation. More details will be given in Section 2.3.4.

Figure 46 – Closed loop step response of $Hc(z) \cdot H(z)$ for the PID tuned using the MATLAB® pidtune function.



Source: The author.

Figure 47 – Duty cycle of the closed loop step response of $Hc(z) \cdot H(z)$ for the PID tuned using the MATLAB® pidtune function.



Source: The author.

2.3.3 Load step response

Assuming that R_{ESR} is negligible and employing the substitutions from Equations 1.34 and 1.35 into Equations 1.31, 1.32, and 1.33, which are then further substituted into Equation 1.27, the Equation 2.7 will be obtained.

$$H(s) \approx \frac{V_{IN} \cdot \frac{R}{1+s \cdot C_O \cdot R}}{\frac{R}{1+s \cdot C_O \cdot R} + s \cdot L + R_S} = \frac{V_{IN} \cdot Z_C(s)}{Z_L(s) + Z_C(s)} \tag{2.7}$$

in which

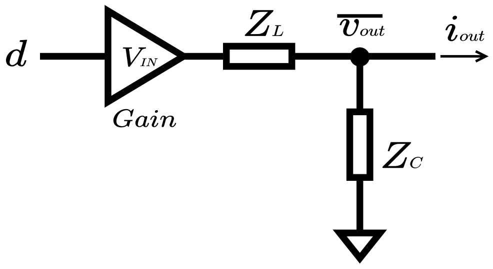
$$Z_C(s) = \frac{R}{1 + s \cdot C_O \cdot R} \tag{2.8}$$

$$Z_L(s) = s \cdot L + R_S \tag{2.9}$$

and R_S is calculated based on the inductor DC resistance and power switch resistance given by Equation 1.36.

Considering the structure of the buck converter and the form of $H(s)$ provided in Equation 2.7, Figure 48 illustrates the equivalent circuit representing the relationship between small signal variations in the average output voltage, load current, and duty cycle.

Figure 48 – Equivalent circuit including output current.



Source: The author.

Applying Kirchhoff's first law yields the following expression:

$$\frac{d(s) \cdot V_{IN} - \overline{v_{out}(s)}}{Z_L(s)} - i_{out}(s) = \frac{\overline{v_{out}(s)}}{Z_C(s)} \tag{2.10}$$

In closed-loop with no variation of the reference voltage, the relationship between the variation in the average output voltage and the duty cycle is expressed as (refer to Figure 43 to visualize the relationship):

$$d(s) = -\overline{v_{out}(s)} \cdot H_c(s) \tag{2.11}$$

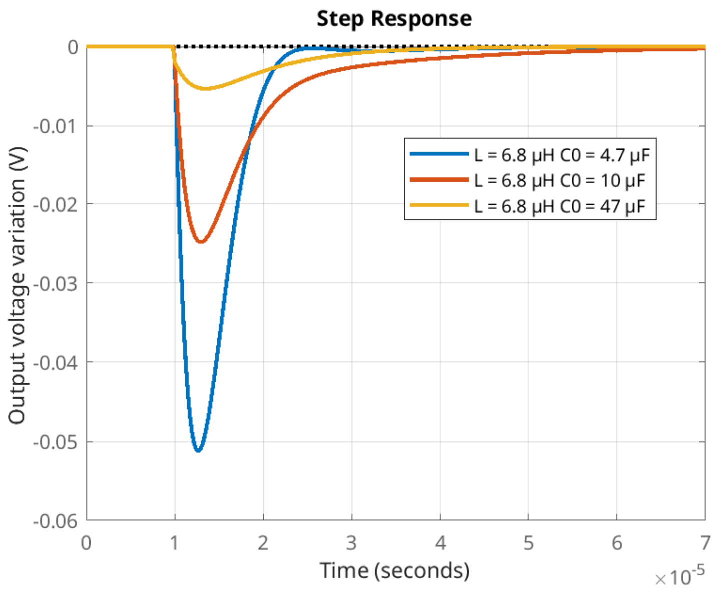
and an approximation of $H_c(s)$ can be derived from $H_c(z)$ using the Tustin's method [5].

Substituting Equations 2.11 and 2.7 into Equation 2.10 yields:

$$\frac{\overline{v_{out}(s)}}{i_{out}} = -\frac{Z_L(s)}{H_c(s) \cdot V_{IN} + \frac{V_{IN}}{H(s)}} \tag{2.12}$$

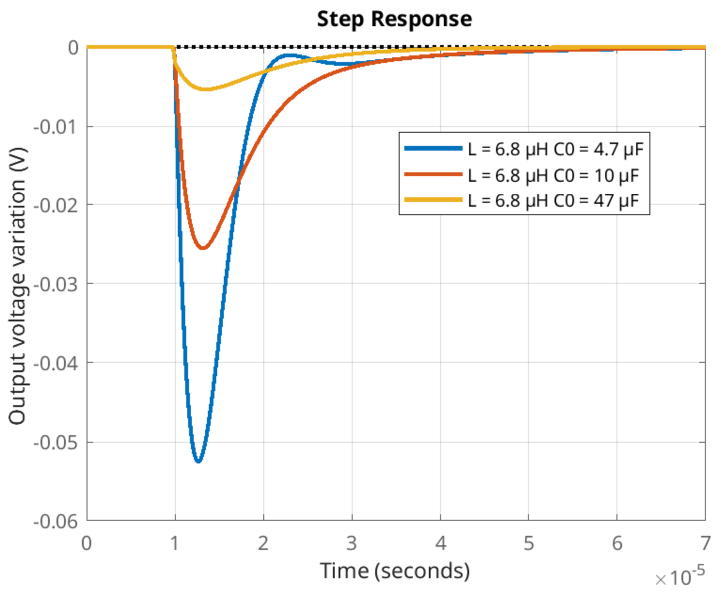
A plot of the average output voltage variation for a 200 mA current step is shown in Figure 49. It must be taken into account that a 200 mA step may result in a large variation of the output resistance R, which may render the model developed in this section invalid. However, a plot of the same step for a $R = 1$ MΩ demonstrates little variation in the average output voltage undershoot (see Figure 50), indicating that the calculations remains relatively valid for large load current variations.

Figure 49 – Load transient for the PID tuned using the MATLAB® pidtune function for $R = 9\ \Omega$.



Source: The author.

Figure 50 – Load transient for the PID tuned using the MATLAB® pidtune function for $R = 1\text{M}\ \Omega$.



Source: The author.

2.3.4 Bandwidth, output capacitor, and the maximum duty cycle change for one-bit variation of the ADC

Equation 2.4 can be represented in discrete time by Equation 2.13. Notice that after a step in the voltage reference, the first correction applied by the control will be $K_0 \cdot e[n]$. In fact, by analyzing the maximum duty cycle in Figure 47, it is possible to notice that it corresponds to the step in the voltage reference multiplied by the K_0 constants in Table 16, which corresponds to the initial combined effect of the derivative, proportional, and integral response.

$$d[n] = d[n-1] + K_0 \cdot e[n] + K_1 \cdot e[n-1] + K_2 \cdot e[n-2] \tag{2.13}$$

From a high-level perspective, the ADC will replace the sample-and-hold block depicted in Figure 43, incorporating additional quantization of the error signal. Due to the quantized nature of the error signal outputted by the ADC, a one-bit variation in the ADC will effectively act as a step for the control. In this case, linearity around zero error (when the control is making only small adjustments to stabilize the output voltage at the target) can be maintained if the control output remains within the practical duty cycle limits of 0% to 100%. Since the minimum $e[n]$ scales up with the ADC step size, it is natural to assume that the ADC step size will limit the maximum allowed value of K_0, thus affecting the system's bandwidth and load transients.

To analyze the dependency between the load transient and the ADC step size, the 'pidTune' function from MATLAB® is used to tune the controller for different values of output capacitors and target system bandwidths, maintaining a minimum phase margin of 54°. As a result, Table 17 shows the output voltage undershoot under load steps of 200 mA, while Table 18 shows the maximum duty cycle variation (the controller's output) for a 1 mV step. To obtain the values for a different step value, is must be scaled proportionally.

Notice that for a fixed capacitance, the control reacts more rapidly and reduces the undershoot as the bandwidth increases. Additionally, as the bandwidth increases, the overall control gain increases, leading to a larger correction in the duty cycle. For a fixed bandwidth, the undershoot is lower with higher capacitance because larger capacitors can store more charge and oppose rapid variations in the output voltage until the control reacts. Also, as the capacitor increases, the control needs to have a higher high-frequency gain to compensate for the lower frequency imaginary poles in $H(s)$.

These two effects combine to result in a tendency for lower load transients and higher maximum duty cycle corrections as the bandwidth and output capacitance increase. This means that achieving lower undershoot will generally require lower ADC step sizes.

Table 17 – Output voltage undershoot for 200 mA current step.

	80 kHz	100 kHz	120 kHz
4.7 µF	62.4 mV	51.2 mV	43.2 mV
10 µF	30 mV	24.8 mV	21.2 mV
47 µF	6.5 mV	5.3 mV	4.8 mV

Table 18 – Maximum duty cycle variation for a step of 1 mV in the input of the controller.

	80 kHz	100 kHz	120 kHz
4.7 µF	0.7%	1%	2.2%
10 µF	1.7%	2.3%	2.8%
47 µF	8.6 %	11.2%	12.8%

2.3.5 Capacitor selection

As discussed in Section 2.3.4, achieving ±2% accuracy under load transients may not be feasible with small capacitors. Additionally, large capacitors necessitate an ADC with smaller step sizes. A suitable compromise is to select a capacitor in the range of approximately 10 µF for a system bandwidth of 100 kHz.

Since there are several commercial capacitors available, comparing them is beyond the scope of this work. However, Table 19 presents the characteristics of two generic Class II multilayer ceramic capacitors, with manufacture references GRM188C81A106MA73 (10 µF) and GRM187R61A226ME15D (22 µF), both with 20% tolerance. It is important to note that the effective capacitance is significantly reduced under the bias condition for this study. Considering the temperature coefficient and tolerances, the effective capacitance ranges from 3.5 µF to 8.2 µF for the 10 µF capacitor and it ranges from 8 µF to 17 µF for the 22 µF capacitor. Therefore, the 22 µF capacitor is a better choice.

Table 19 – Summary of some commercial capacitor parameters from Murata.

Value	**ESR**	**Effective capacitance**			**Temperature variation -55°C to 85°C**
		DC=1.62 V AC=10 mV	**DC=1.80 V AC=10 mV**	**DC=1.98 V AC=10 mV**	
10 µF	< 10 mΩ	6.7 µF	6.6 µF	6.4 µF	-31% to +1.3%
22 µF	< 10 mΩ	13.9 µF	13.2 µF	12.5 µF	-22% to +1.7%

2.3.6 Input voltage step response

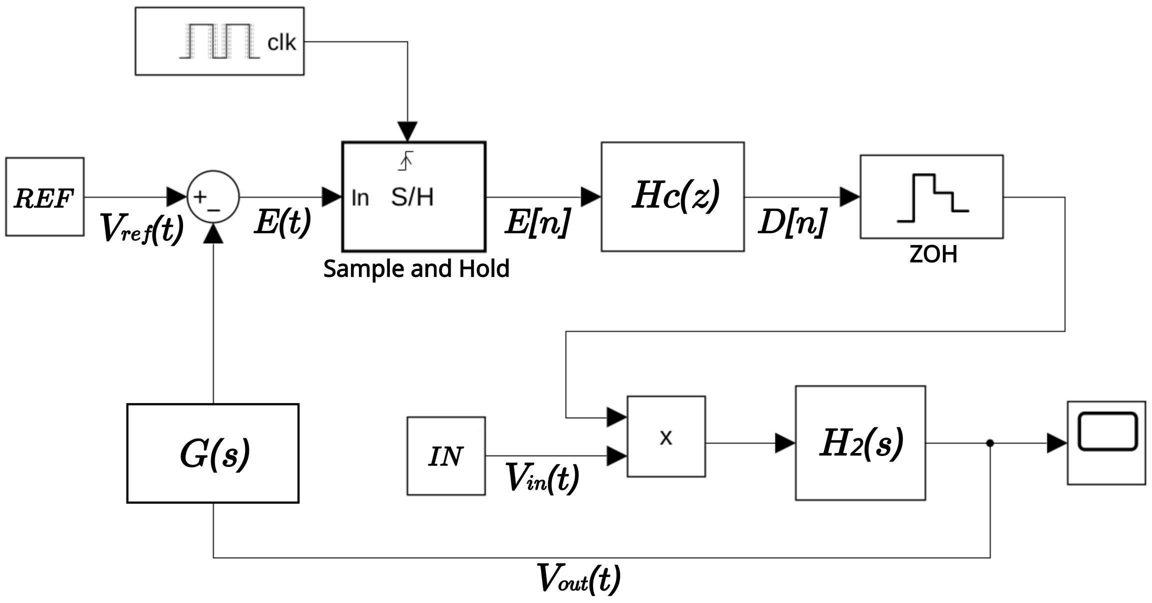
In contrast to the insensitivity to output voltage, the gain of the transfer function $H(s)$, and consequently $H(z)$, is directly proportional to V_{IN}, as shown in Equation 1.29. However, if an alternative transfer function $H_2(z)$ is defined as:

$$H_2(z) = \frac{H(z)}{V_{IN}} \tag{2.14}$$

the gain of $H_2(z)$, consequently $H_2(s)$, will be independent of the input voltage.

Considering the independence of $H_2(z)$ on the input voltage and output voltage, coupled with the capability of $H(z)$ (and hence $H_2(z)$) to yield an approximate DC value (as detailed in Section 2.3.2), the small signal model can be expanded to incorporate large signal variations, as depicted in Figure 51. This model is valid as long as $D[n]$ remains between 0 and 1. Also, note that the selected bandwidth of 100 kHz for the closed loop system will only be achieved with a 3.7 V input voltage (the value assumed during the PID tuning), for different input voltages the gain and bandwidth of the system will vary, because the substitution suggested in Equation 2.14 will only make $H_2(s)$ independent of the input voltage, not the entire system.

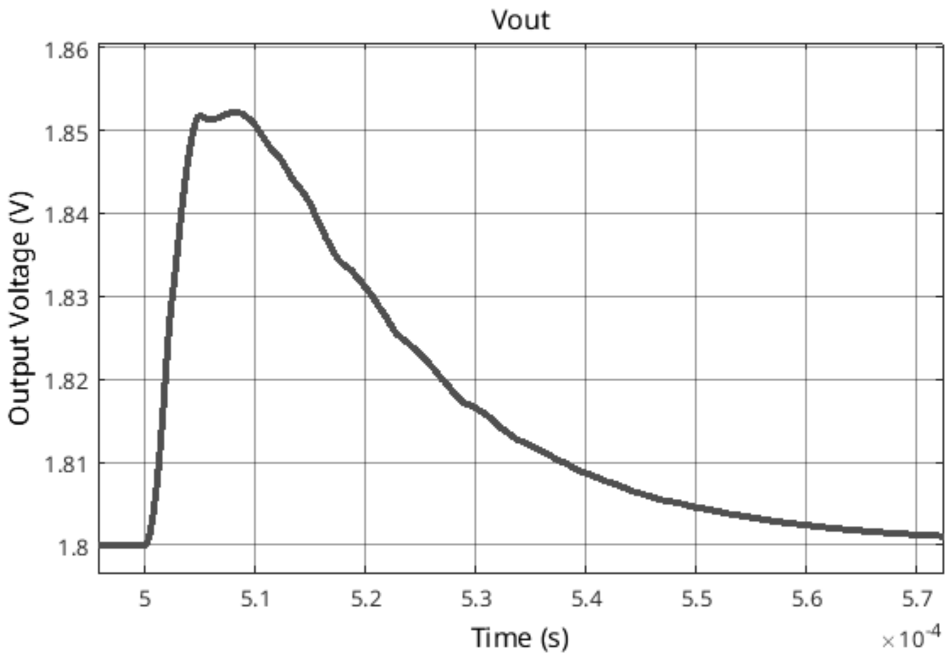
Figure 51 – Large signal block diagram using Simulink® that includes input voltage dependency.



Source: The author.

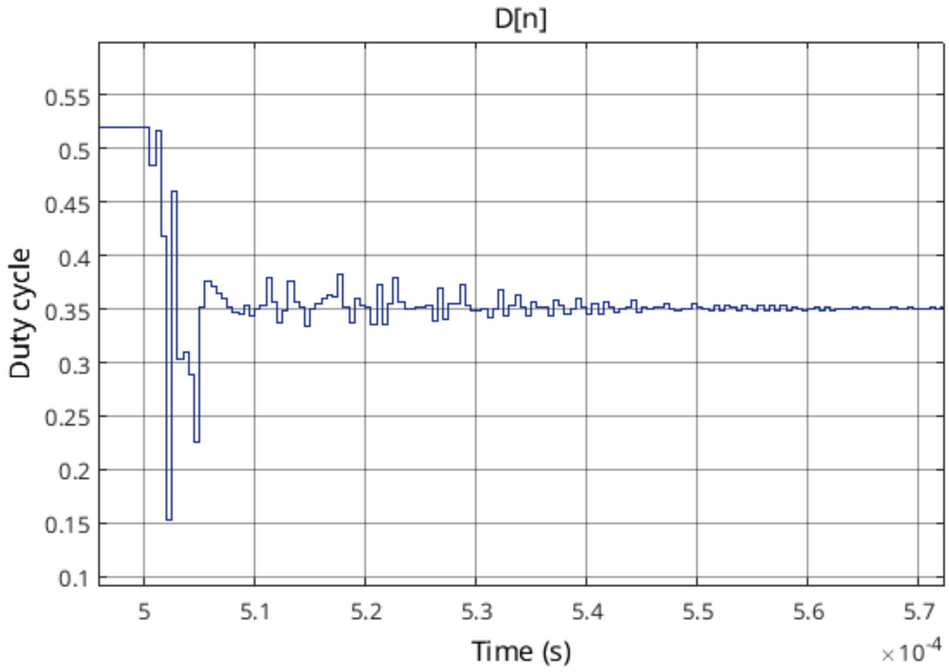
When running the model depicted in Figure 51 using the Simulink® software, with an input voltage step from 3.7 V to 5.5 V and an output capacitance of 10 µF, the resulting output voltage variation and duty cycle variation are illustrated in Figures 52 and 53, respectively. It is possible to notice that the ±2% accuracy specification can not be achieved unless some solution for the input voltage dependency can be found.

Figure 52 – Output voltage under a step of 3.7 to 5.5 V in the input voltage for the PID tuned using the MATLAB® pidtune function.



Source: The author.

Figure 53 – Duty cycle for the input voltage transient of 3.7 to 5.5 V.

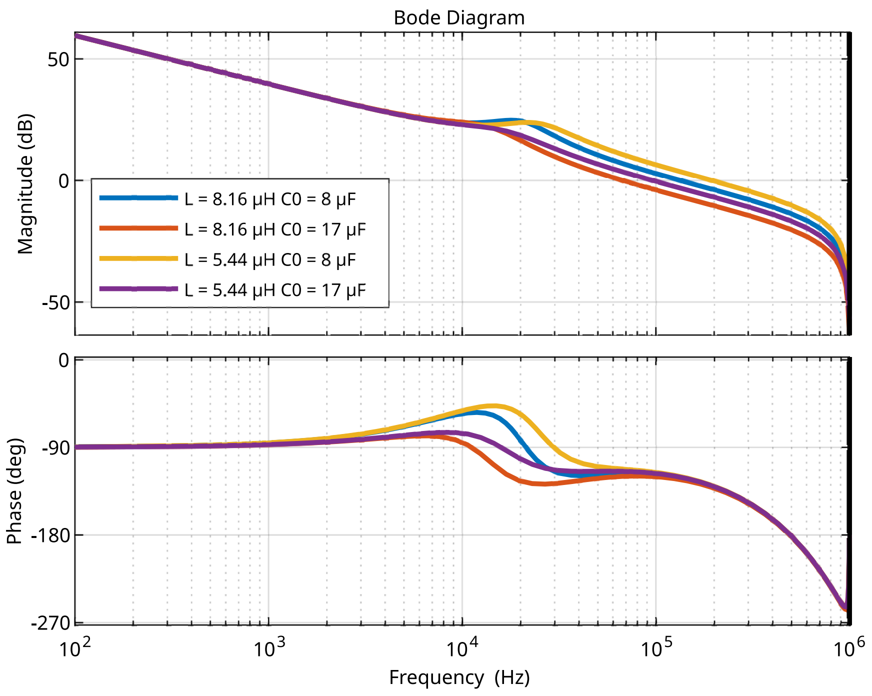


Source: The author.

2.3.7 Dependency on inductor and capacitor tolerances

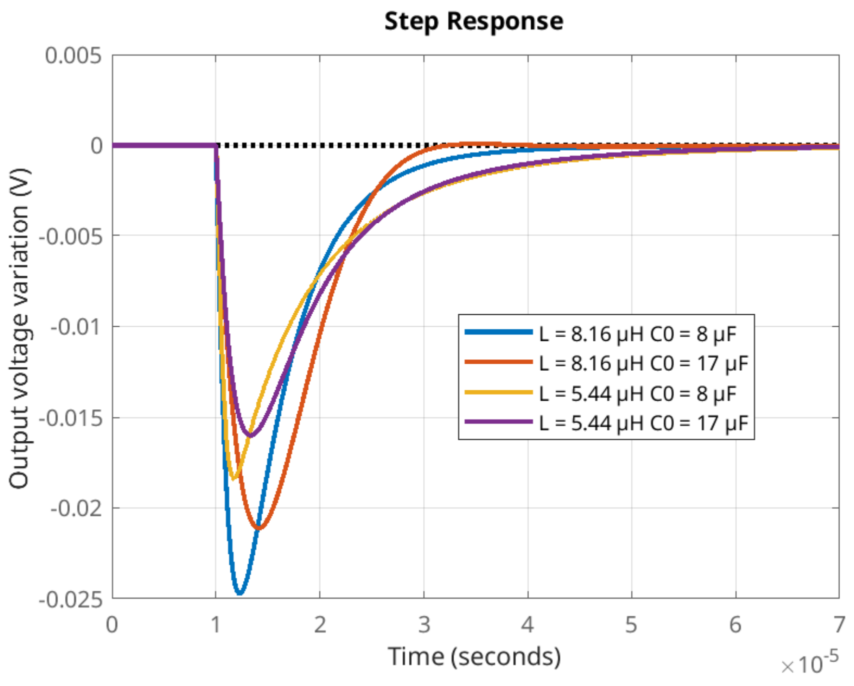
This sections shows how the tolerance of the inductor and capacitor can change the system's performance.

Figure 54 – Bode plot of $Hc(z) \cdot H(z)$ for the PID tuned using the MATLAB® pidtune function for L = 6.8 µH and C_O = 13.2 µF including tolerances.



Source: The author.

Figure 55 – Load transient for the PID tuned using the MATLAB® pidtune function for for L = 6.8 µH and C_O = 13.2 µF including tolerances.



Source: The author.

In order to achieve this goal, the PID constants were tuned for an inductor of 6.8 µH and an output capacitor of 13.2 µF using the MATLAB® pidtune function. Subsequently, without altering these constants, both a Bode plot of $Hc(z) \cdot H(z)$ and a load transient plot were generated including variations in the value of the inductor and output capacitor, as illustrated in Figures 54 and 55. It is noticeable that load transients deteriorate with increased inductance and decreased capacitance. Additionally, there is a notable variation in bandwidth. Although it is feasible to fine-tune the PID for a specific combination of inductor and capacitor values, compensating for variations due to temperature poses greater challenges. Therefore, this analysis provides a conservative estimate for the impact of variations in inductor and capacitor values.

2.4 Output voltage ripple

With an input voltage (V_{IN}) of 5.5 V, an average output voltage ($\overline{V_{OUT}}$) of 1.98 V, and a minimum inductance (L) of 5.44 µH, the maximum inductor current ripple can be calculated using Equation 1.12:

$$\Delta_{I_L} = \frac{\frac{1.98}{5.5} \cdot 500 \cdot 10^{-9} \cdot (5.5 - 1.98)}{5.44 \cdot 10^{-6}} = 116.5\ mA \tag{2.15}$$

Assuming a maximum R_{ESR} of 10 mΩ and a minimum output capacitance (C_O) of 8 µF, an upper bound for the output voltage ripple can be calculate using Equation 1.15:

$$max\left(\Delta_{V_{OUT}}\right) < \frac{0.1165 \cdot 500 \cdot 10^{-9}}{8 \cdot 8 \cdot 10^{-6}} + 0.1165 \cdot 0.01 \implies max\left(\Delta_{V_{OUT}}\right) < 2.1\ mV \tag{2.16}$$

2.5 Quantization

The ADC and DPWM quantization introduces noise and also alters the gain of the loop. From a stability perspective, the gain of the loop will be divided by Q_{ADC} and multiplied by Q_{DUTY}, where Q_{DUTY} represents the duty cycle step size of the DPWM and Q_{ADC} represents the step size of the ADC in volts. Consequently, the PID constants calculated using the methods outlined in Section 2.3.1 must be divided by $\frac{Q_{DUTY}}{Q_{ADC}}$ in the final implementation.

From a high-level perspective, the DPWM will replace the ZOH in Figure 51, incorporating additional quantization of the duty cycle (Q_{DUTY}). If the multiplication by the input voltage is included as part of the DPWM, the voltage step size of the DPWM (Q_{DPWM}) can also be defined, representing the granularity of the correction that the DPWM can apply to the output voltage. This parameter can be defined as:

$$Q_{DPWM} = Q_{DUTY} \cdot V_{IN} \tag{2.17}$$

2.6 Requirements for the ADC

This section aims to summarize the information obtained so far in order to offer guidelines for the establishment of the ADC requirements. A list of the parameters along with some explanations is included below:

- **Architecture**: In Chapter 1, various ADC architectures for integrated buck converters were discussed. Among these, time-based architectures employing ring oscillators [3], VCOs [13], or two matched delay lines [12, 14, 17] stand out for their low power consumption, small chip area, and their ability to digitize differential voltages within a narrow range, typically a few dozen millivolts. These characteristics make them particularly suitable for this work, given the required accuracy of $\pm 2\%$ for the buck.
- **ADC Accuracy**: Given a minimum output voltage of 1.62 V, the target output voltage must not be set lower than 1.653 V to accommodate a $\pm 2\%$ variation, allowing a maximum difference of 33 mV (1.653 V - 1.62 V) between the target and the actual output voltage. Considering a maximum deviation of ± 25 mV (Figure 55) for the average output voltage under load transients and a maximum ripple of 2 $\mathrm{mV_{PP}}$ (Equation 2.16), there is a spare margin of ± 7 mV (33 mV - 25 mV - 1 mV) for inaccuracies in the ADC. Assuming that a differential ADC will be employed to digitize the difference between the scaled version of the output voltage and the voltage reference, accuracy becomes particularly important for the output code that represents 0 V, since the goal of the control is to keep this difference as low as possible. In this case, the ADC must not deviate more than ± 7 mV multiplied by the scaling factor $G(0)$.
- **Resolution**: The number of bits ranges between 5 to 6 bits in [13, 12, 14, 17] and 4 bits in [10]. It is worth noting that implementing the transfer function represented in Equation 2.4 requires multiplication in the digital domain, leading to an increase in hardware complexity as the ADC resolution rises.
- **Step Size**: A window of ± 30 mV (in order to cover the load transients with some margin) corresponds to a differential ADC with a 60 mV window when $G(0)$ equals 1. This results in a step size of 3.75 mV for 4-bits, 1.875 mV for 5-bits, and 0.9375 mV for 6-bits. All of these step sizes must be multiplied by $G(0)$ in order to obtain the actual ADC step size.
- **Sampling Rate**: The sample rate of the ADC is determined by the switching frequency of the converter. In our case, 2 MSPS was chosen.
- **Linearity**: As discussed in Section 2.5, the step size (Q_{ADC}) affects the gain and the bandwidth of the system. Therefore deviations of Q_{ADC} with relation to the ideal

step size is an important parameter, which can be measured by means of Differential Non Linearity (DNL), specially around 0 V. There is no specification for DNL, but it must be minimized since variations in the step size affects the gain of the loop.

- **Current Consumption**: Not all authors report the current consumption for the ADC alone. However, among those who do, a range of current spanning from 18 µA to 220 µA is reported for delay-based ADCs.

2.7 Requirements for the DPWM

This section aims to summarize the information obtained so far in order to offer guidelines for the establishment of the DPWM requirements. A list of the parameters along with some explanations is included below:

- **Architecture**: As discussed in Section 2.3.6, the accuracy of ±2% is violated under input voltage transients. Some authors [9, 16] suggest digitizing both the input voltage and output voltage to overcome this problem. However, if the DPWM is able to keep Q_{PWM} constant and update the DC value of the duty cycle according to the input voltage, ideally there would be no effect of the input voltage on the output voltage. If the goal is to use only one ADC, a DPWM architecture with this feature shall be exploited.

- **Step Size**: Q_{PWM} must be smaller than $\frac{Q_{ADC}}{G(0)}$ in order to avoid limiting cycle oscillations [20]. When it is not possible to meet this condition, the digital PID can utilize the dithering technique.

- **Accuracy**: In principle, the accuracy of the DPWM does not need to be specified since variations can be corrected by the controller. However, the accuracy will affect the resolution.

- **Resolution**: Ideally the DPWM resolution should cover the maximum allowed range for fast transient response, which is 0 to V_{IN}. However, taking into account that Q_{PWM} must be lower than 3.75 mV for a 4-bit ADC, this would result (for a minimum input voltage of 2.7V) in a DPWM with resolution of:

$$N > log_2\left(\frac{2.7}{0.00375}\right) \approx 9.5\ bits \tag{2.18}$$

  which may be unfeasible to implement if the maximum chip area is constrained. In addition to this, the resolution must also cover the accuracy of the DPWM, because as the accuracy varies, the range of 0 to V_{IN} should still be covered.

- **Linearity**: Deviations of Q_{PWM} with relation to the ideal step size is an important parameter, which can be measured by means of Differential Non Linearity (DNL). There is no specification for DNL, but it must be as low as possible.
- **Sampling Rate**: The rate at which the DPWM is updated is determined by the switching frequency of the converter. In this case, 2 MSPS.
- **Current Consumption**: Not all authors report the current consumption for the DPWM alone. However, among those who do, a range of current spanning from 6 µA to 370 µA is reported.

3 Implementation

This chapter aims to develop the implementation details of the ADC, DPWM, and digital PID.

3.1 Analog to digital converter (ADC)

As discussed in Section 2.6, the need for an ADC with rail-to-rail input is unnecessary because the scaled version of the output voltage must fall within a predefined range near the voltage reference. It was also discussed that ring oscillators, VCOs, and matched delay lines have been employed in order to minimize power consumption and chip area. However, delay line architecture is also prone to meta-stability, as highlighted in [14].

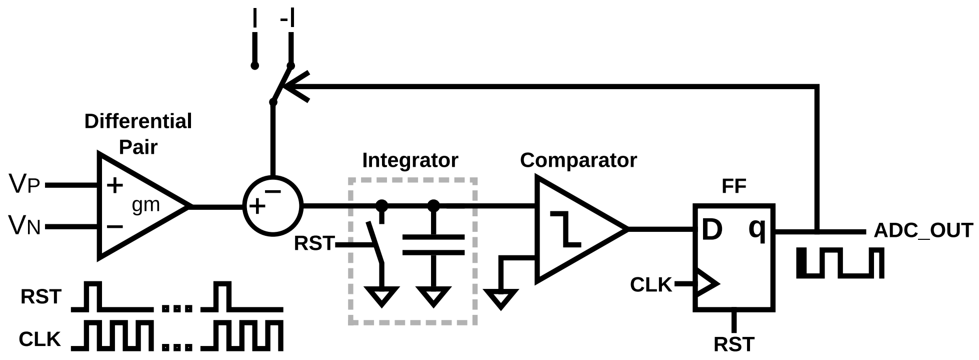
The architecture presented in this chapter was designed to mitigate the risk of meta-stability by working with a clocked comparator in the same frequency domain of the reference clock. However, it is important to note that the proposed architecture is dependent on the temperature and relies on the availability of a stable clock with a frequency at least one order of magnitude higher than the switching frequency. This latest assumption is typically reasonable for battery-powered applications incorporating a microprocessor.

Before detailing the proposed architecture, recall that the work developed in [3, 12] involve the use of differential biasing for two VCOs and delay lines, respectively. In [3], the differential bias is accomplished by a single differential pair that senses the scaled version of the output voltage and the voltage reference, thus providing a differential bias for the two VCOs. Specifically, the differential current in this configuration is equal to the difference between the output voltage and the voltage reference (V_e) multiplied by the transconductance of the differential pair (g_m).

Moving forward, consider the block diagram of the proposed ADC, depicted in Figure 56. Inspired by a $\Delta\Sigma$ modulator structure [24], this system multiplies the differential input by the transconductance of the differential pair (g_m), subtracts the results from either I or $-I$ (the output of a 1-bit current Digital-to-Analog Converter (DAC)), and integrates this difference over time. The comparator checks the sign of the integrated error (which is proportional to a voltage since the integrator is a capacitor) at the positive edge of the clock and updates the DAC accordingly. A reset signal restarts both the integrator and the D Flip Flop at fixed intervals of clock pulses. Accumulating the output over the same interval yields a digital value representing the product $(V_P - V_N) \cdot g_m$ over the range $[-I, I]$.

Furthermore, if the current I is proportional to the current flowing through the drain of the transistors in the differential pair, the step size will depend only on the interval between two resets and the ratio between the transconductance to the drain current of the transistor ($\frac{g_m}{I_d}$), which, in weak inversion, is approximate constant for a given temperature [25].

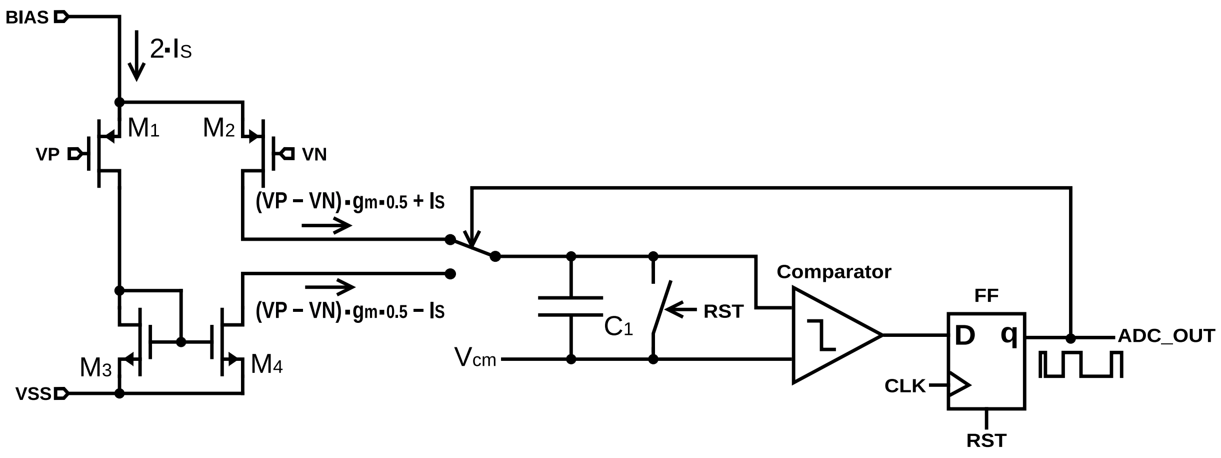
Figure 56 – Block diagram of the proposed ADC.



Source: The author.

Figure 57 illustrates a preliminary implementation of the proposed ADC architecture. In this setup, capacitor C_1 serves as the integrator and charge-to-voltage conversion, while the selection between summing $(V_P - V_N)\cdot g_m\cdot 0.5 + I_S$ or $(V_P - V_N)\cdot g_m\cdot 0.5 - I_S$ encompass the transconductance amplifier, adder, and DAC sub-blocks shown in Figure 56. Please note that the integrated error is now compared against a non-zero voltage (V_{CM}).

Figure 57 – Simple implementation of the ADC.



Source: The author.

While this implementation is simple, it does have some limitations:

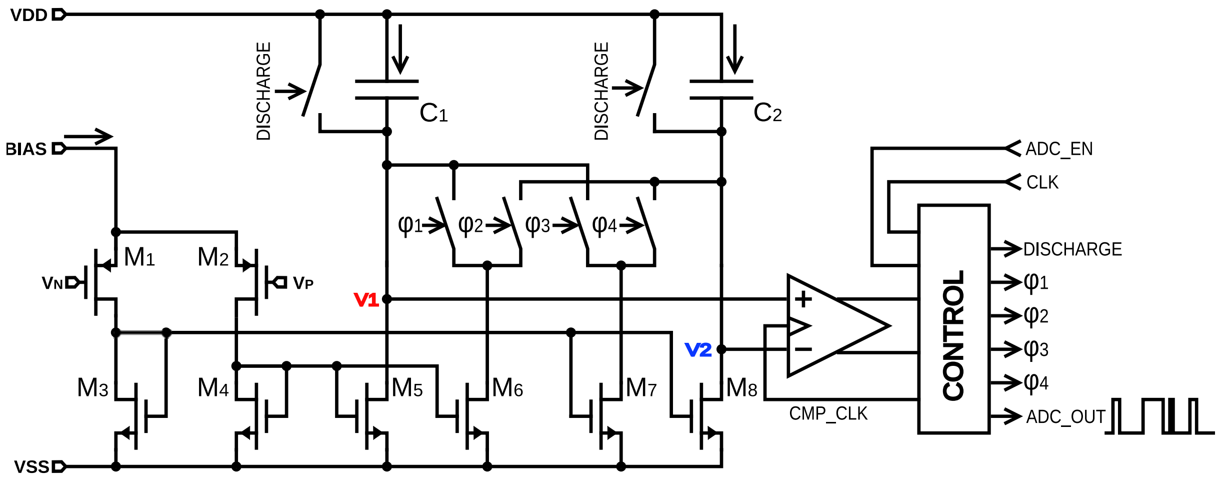
- The inputs of the comparator are connected to nodes with different impedances, potentially causing offset depending on the comparator architecture, especially if a simple clocked comparator replaces the comparator and D Flip Flop.
- Charge injection from the switches may result in errors over several clock pulses.

- Transistors M_2 and M_4 will shift between saturation and triode depending on the switch position, potentially injecting unwanted charge into capacitor C_1 during transitions. This issue can be mitigated by rerouting unused current to ground or VDD.
- Either $(V_P - V_N) \cdot g_m \cdot 0.5 - I_S$ or $(V_P - V_N) \cdot g_m \cdot 0.5 + I_S$ is summed to capacitor C_1, fixing the ADC's range without the ability to scale it according to design requirements.

Given the challenges encountered during our initial development, the proposed implementation adopts the architecture depicted in Figure 58. This symmetric design ensures uniformity in both nodes V_1 and V_2, minimizing disparities in charge injection and impedance. It also provides a way to scale the voltage step trough the addition of a current mirror with a scale of K to 1.

The comparator operates by checking the difference between V_1 and V_2 at every rising edge of CMP_CLK. If V_1 is greater than V_2, the output Z will rise and remain high until the failing edge of the CMP_CLK. If V_2 is greater than V_1, the output $\mathrm{Z_N}$ will rise and remain high until the failing edge of the CMP_CLK. The architecture of the clocked comparator used in this study is also detailed in Annex B. It is worth noting that while this specific comparator architecture was employed, the system probably works with different comparator architectures as well as long as it functions in the same way.

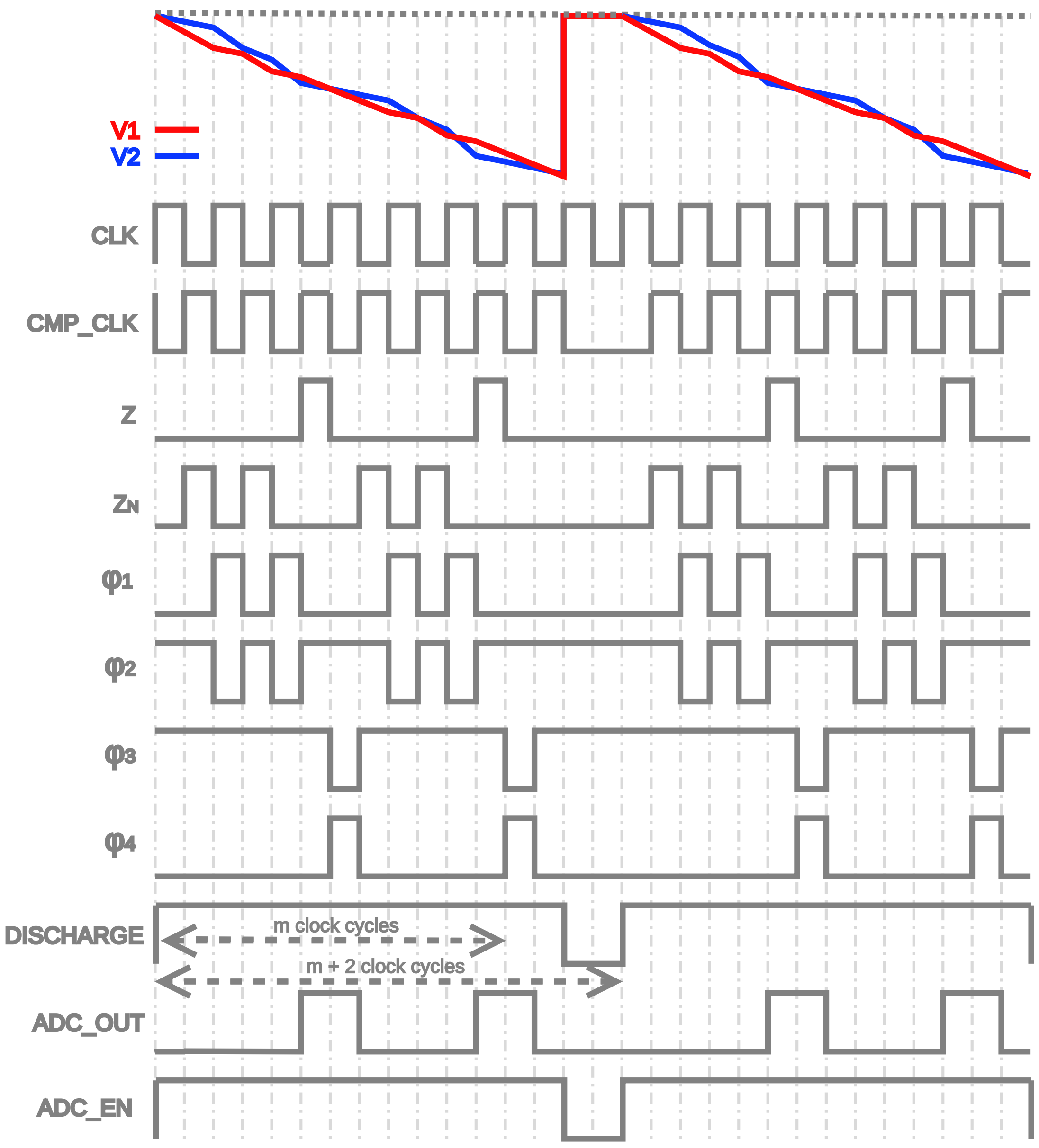
Figure 58 – Schematic of the proposed ADC.



Source: The author.

The timing diagram in Figure 59 highlight the circuit's operation with the expected time scale, while the time diagrams in Figures 60 and 61 highlight the order between the transitions. It is worth emphasizing that Figures 60 and 61 do not represent the transitions with the correct time scale. The time scale was intentionally distorted to represent the correct order between the transitions and the causal relationship between them.

Figure 59 – Timing diagram of the proposed ADC.



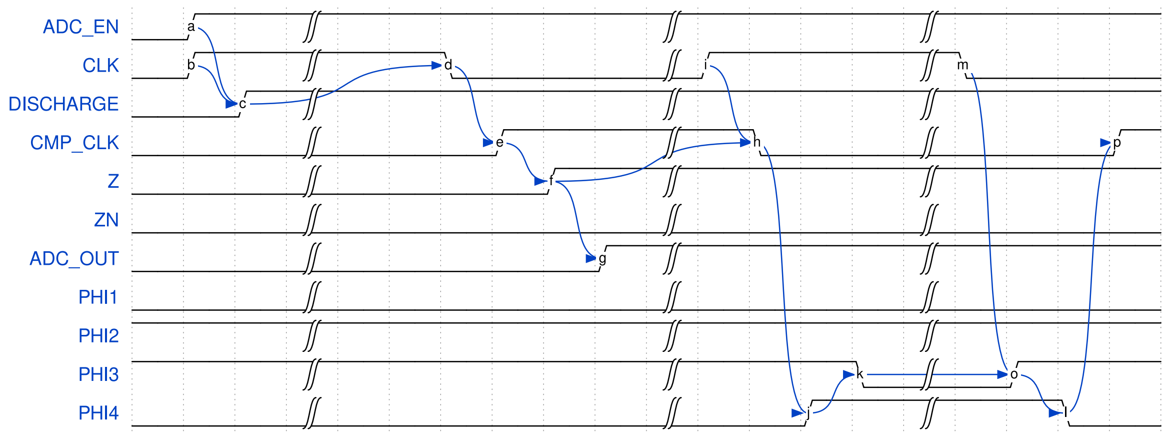
Source: The author.

Initially, switches controlled by signals $\varphi 1$ and $\varphi 4$ are closed, while switches controlled by signals $\varphi 2$ and $\varphi 3$ are open. When ADC_EN is high, capacitors C_1 and C_2 begin charging. At each falling edge of the clock (CLK), the comparator evaluates the voltage difference between V_1 and V_2. If $V_1 < V_2$, the switch controlled by $\varphi 1$ will open while the one controlled by $\varphi 2$ will close on the subsequent clock (CLK) cycle high level. Conversely, if $V_1 > V_2$, the switch controlled by $\varphi 4$ will open while the one controlled by $\varphi 3$ will close on the subsequent clock cycle high level. When the ADC_EN triggers low, the conversion halts and the internal logic and capacitors reset.

The control block was designed to ensure the sequence illustrated in Figures 59, 60, and 61. As highlighted in Figures 60 and 61, it is necessary to introduce a non-overlapping period between $\varphi 1$ and $\varphi 2$ (as well as between $\varphi 3$ and $\varphi 4$) to avoid simultaneous connection of both capacitors via the switches. Although this is not explicitly depicted in Figure 59 because of the time scale used, it is a crucial requirement for the correct operation of the design. It is important to note that signals such as DISCHARGE, $\varphi1$, $\varphi2$, $\varphi3$, and $\varphi4$ are inverted because PMOS transistors are the preferred choice for implementing the switches controlled by these signals.

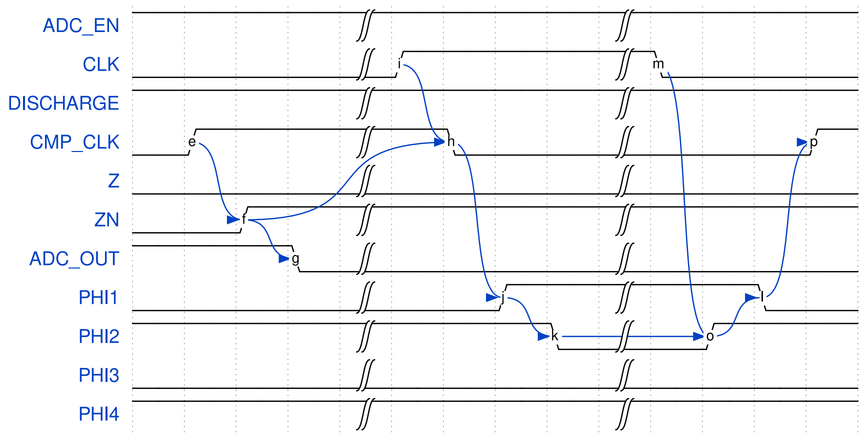
Additionally, the comparator should only evaluate the difference between V_1 and V_2 after the switches controlled by $\varphi1$, $\varphi2$, $\varphi3$, and $\varphi4$ have returned to their starting position. This compensates for the effects of charge injection.

Figure 60 – Detailed timing diagram after ADC_EN rising edge and also when $V_1 > V_2$.



Source: The author.

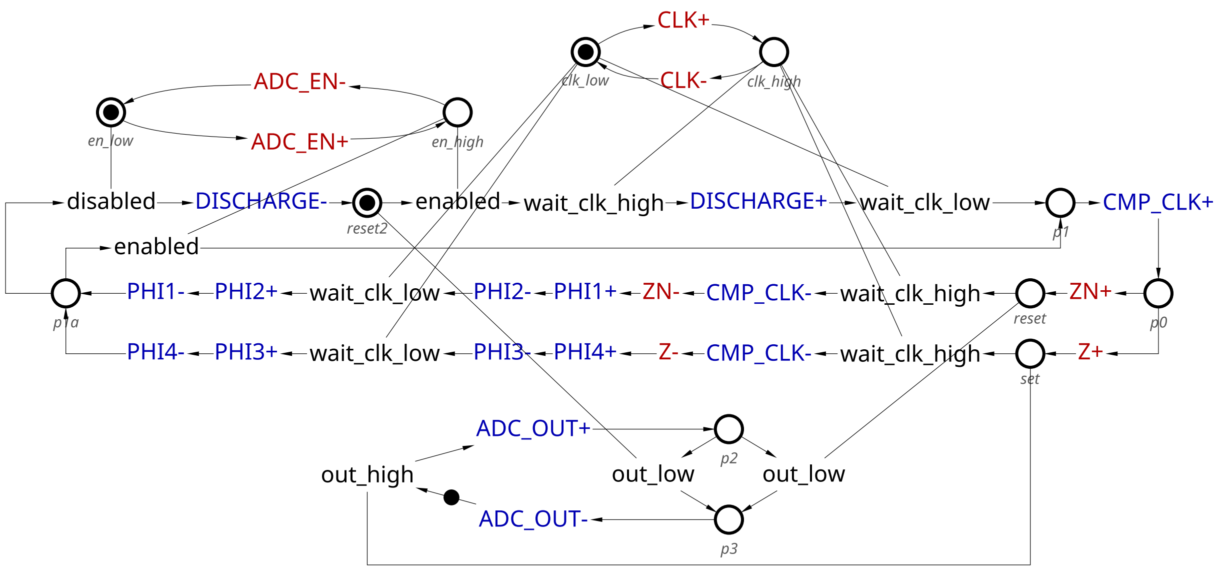
Figure 61 – Detailed timing diagram when $V_1 < V_2$.



Source: The author.

Although all signals are synchronized with the clock, their transitions follow an asynchronous sequence of events. As a result, they can be modeled using Signal Transition Graph (STG) and synthesized using tools like Workcraft [26]. Figure 62 depicts a conceptual non-synthesizable STG, which can be made synthesizable with the addition of synchronization elements [27]. Additionally, a Verilog-A model of this non-synthesizable STG can be compiled using a Python program developed by the author [28]. In this work, a manual implementation of this logic was used for simulations, as described in Annex B.

Figure 62 – Signal Transition Graph of the ADC control.



Source: The author.

3.1.1 Analysis of the relationship between the differential input and the accumulated ADC output

In order determine the relationship between the differential input and the accumulated output, consider first the differential current ($\Delta I = (I_2 - I_1)$) that charges capacitors C_1 and C_2 in the system illustrated in Figure 58. This differential current can have three different values based on the configuration of the switches controlled by the signals $\varphi 1$, $\varphi 2$, $\varphi 3$, and $\varphi 4$:

- When the switches controlled by $\varphi 1$ and $\varphi 4$ are closed, the differential current (ΔI) is determined by Equation 3.1. Here, g_m represents the transconductance of transistors M_1 and M_2, V_P and V_N denote the voltages at inputs V_P and V_N respectively, and K is the mirror ratio.

$$\Delta I_a = (K+1) \cdot g_m \cdot (V_P - V_N) \tag{3.1}$$

- If the switches controlled by $\varphi 1$ and $\varphi 3$ are closed, the differential current is given by Equation 3.2, where I_{bias} is the bias current.

$$\Delta I_b = K \cdot g_m \cdot (V_P - V_N) - I_{bias} \tag{3.2}$$

- When the switches controlled by $\varphi 2$ and $\varphi 4$ are closed, the differential current is determined by Equation 3.3.

$$\Delta I_c = K \cdot g_m \cdot (V_P - V_N) + I_{bias} \tag{3.3}$$

Assuming that both capacitors C_1 and C_2 have the same value C, and T represents the period of the clock, the voltage difference between V_1 and V_2 will vary by a certain amount between two falling edges of the clock. Specifically, the variation will be either equal to $\frac{(\Delta I_a+\Delta I_b)\cdot T}{2\cdot C}$ or $\frac{(\Delta I_a+\Delta I_c)\cdot T}{2\cdot C}$. Also, note that, during the first clock cycle, the switches remain in their initial position because the comparator has not made its first decision yet as depicted in Figure 59.

If the ADC_EN signal remains high for m falling edges of the clock and the ADC_OUT signal is high during n rising edges of the clock (accumulated output), the difference between V_1 and V_2 by the end of m falling edges can be determined by Equation 3.4.

$$V_1 - V_2 = \frac{(m+1)\cdot \Delta I_a \cdot T}{2\cdot C} + (m-n)\cdot \frac{\Delta I_c \cdot T}{2\cdot C} + n\cdot \frac{\Delta I_b \cdot T}{2\cdot C} \tag{3.4}$$

It is reasonable to assume that $\Delta I_a + \Delta I_b$ and $\Delta I_a + \Delta I_c$ have opposite signs because the comparator output would be always high or low otherwise. Therefore, by the end of m clock cycles, the difference $V_1 - V_2$ will be bounded by:

$$\frac{(\Delta I_a + \Delta I_b)\cdot T}{2\cdot C} < V_1 - V_2 < \frac{(\Delta I_a + \Delta I_c)\cdot T}{2\cdot C} \tag{3.5}$$

By substituting Equations 3.1, 3.2, and 3.3 into Equation 3.4, and then substituting the result into Equation 3.5, gives:

$$-\frac{Q_{ADC}}{2} < (V_P - V_N) - \left(n\cdot Q_{ADC} - m\cdot \frac{Q_{ADC}}{2}\right) < \frac{Q_{ADC}}{2} \tag{3.6}$$

, in which

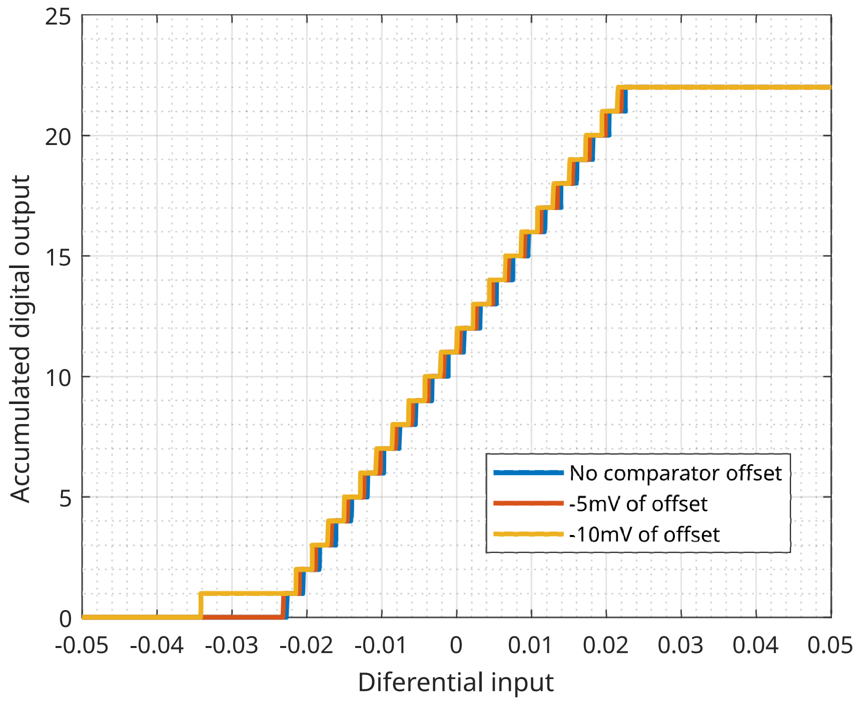
$$Q_{ADC} = \frac{2\cdot I_{bias}}{g_m\cdot(2\cdot K\cdot m + m - K)} \tag{3.7}$$

A high Level Matlab model (Annex D) was developed in order to validate the accuracy of Equations 3.6 and 3.7. The impact of Equation 3.7 on the design choices will be discussed in Section 3.1.3.2.

3.1.2 Influence of the comparator offset

At the start of the conversion process, the capacitors are discharged, making the architecture highly dependent on the offset of the clocked comparator. By running the high-level model described in Annex D, it becomes evident that an increase in offset compromises the linearity of the steps near the minimum (and maximum if the sign of the offset is inverted) as depicted in Figure 63. This occurs because the output of the ADC remains at 1 (or 0) for a few clock periods immediately after the capacitors are allowed to charge. Nonetheless, since only the middle range of the ADC is relevant (close to 0 V), it is possible to utilize a larger value for m (see Figure 59) and then exclude the non-linear extremities.

Figure 63 – ADC transfer curve including effects of offset in the clocked comparator.



Source: The author.

3.1.3 Sizing of the devices, bias, supply and clock

This section discusses factors to consider during the sizing of the devices and selection of the supply, clock, and bias. While the sizing process itself will not be detailed, as it was manually performed based on intuition acquired during simulations, the observations driving this intuition will be discussed. These insights can be valuable for reproducing the results presented in Chapter 4 using a different technology or for tuning the parameters according to a different specification.

3.1.3.1 Supply

Recall that this study focuses on digitally driven control for DC-DC converters for battery powered applications, which are designed to convert the voltage from a Lithium-ion battery to a level compatible with the native transistors of the chosen technology (1.8 V for the CMOS 180 nm technology in this work). It is assumed that the technology has transistors capable of withstanding 5.0 V between drain and source for implementation of the power switches, although the specific design of these switches is not the focus here.

Specifically, the technology selected for this work has transistors that can also handle 5.0 V between source and gate.

Consequently, the ADC (Analog-to-Digital Converter) can be powered by either the battery voltage or the output of the DC-DC converter. Opting for the DC-DC output can reduce the power consumption of the ADC. However, in this scenario, the DC-DC can not autonomously start and requires startup circuitry, a feature that can be found in commercial analog DC-DC converters, for example, the LM3687 from Texas Instruments utilizes a startup LDO (Low Dropout Regulator). Additionally, a voltage divider is necessary in this condition because the differential pair of the ADC can not operate if the input voltage is close to the supply voltage.

In this study, it is assumed that both the ADC and the digital PID will be powered by the output of the DC-DC converter. This decision does not impact the fundamental operation of the control once the output voltage has reached the minimum level required for the correct operation of the circuit, which in this study is assumed to be 1.62 V.

3.1.3.2 Sizing of transistors in the differential pair and clock frequency selection

The bias point of the differential pair greatly influences the step size, as indicated in Equation 3.7. This equation also highlights the dependency on the NMOS mirror ratio (K) and the number of clock cycles during which the ADC is enabled (m). There are trade-offs to consider.

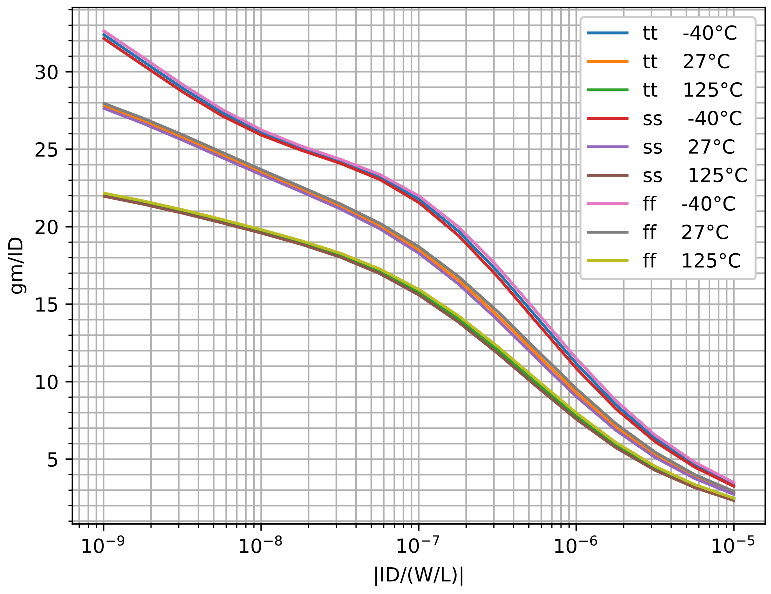
As K increases, a common-mode current of $K \cdot I_{BIAS}$ charges both C_1 and C_2 at the same rate. Consequently, since the maximum voltage in both capacitors cannot exceed the supply voltage minus the saturation voltage of the NMOS mirrors, the maximum difference between V_1 and V_2 must decrease as K increases, making the circuit more susceptible to noise. In addition, with an increase in m, the minimum clock frequency for the system also rises, since it must be greater than $(m + 2) \cdot F_S$ (see Figure 59), where F_S denotes the switching frequency of the DC-DC converter. Moreover, it is essential to note that for a differential pair to operate in the linear region, the modulus of the differential input ($|V_P - V_N|$) must be significantly smaller than $\frac{2 \cdot I_d}{g_m}$ [25].

In Section 2.6, it is discussed the need of digitizing a ± 60 mV window. If a voltage divider with a ratio of 0.5 is employed (resulting in the ADC input being half of the DC-DC output voltage), this range shrinks to ± 30 mV, requiring a $\frac{g_m}{I_d} << 66.67$.

Figure 64 demonstrates that an $\frac{I_d}{\frac{W}{L}}$ of 25 nA yields a maximum $\frac{g_m}{I_d}$ of 24.5, which falls below the limit required for the ADC range. It is not significantly smaller than 66.67, however it is not deep into non-linear region either. The specific choice depends on a trade-off between the area of the differential pair and the step size.

Assuming $\frac{I_d}{\frac{W}{L}} = 25$ nA, $K = 1.5$ to maintain a low value for K, and $m = 22$ to achieve a 4-bit resolution ADC while allowing for some headroom for clocked comparator mismatch effects (as mentioned in Section 3.1.2), the step size would be 1.89 mV for -40 °C, 2.10 mV for 27 °C, and 2.47 mV for 125 °C. Assuming a voltage divider with a ratio of 0.5, this results in a minimum step size of 3.78 mV, which is sufficient to cover the required window. The ADC frequency must be greater than 48 MHz, which can be round-up to 50 MHz.

Figure 64 – PMOS transistor transconductance divided by the drain current as a function of the drain current divided by the width to length ratio. Drain to source voltage was set to 0.9 V, width was set to 1 µm, and length also to 1 µm.



Source: The author.

3.1.3.3 Sizing of the NMOS mirrors

To ensure proper operation, the bias point of NMOS transistors must be carefully chosen to prevent the source to gate voltage of M_3 and M_4 from driving M_1 and M_2 out of saturation. This consideration becomes especially crucial if the simple NMOS mirrors are replaced by a cascaded architecture.

3.1.3.4 Bias and sizing of the switches and capacitors

The switches, controlled by signals $\varphi 1$, $\varphi 2$, $\varphi 3$, and $\varphi 4$, should be sized to minimize charge injection. However, their width must be sufficient to ensure that the voltage drop across any of them does not drive transistors M_6 and M_7 out of saturation.

Additionally, capacitors C_1 and C_2 must have a much higher capacitance than any parasitic capacitance of the NMOS mirrors. This helps to prevent errors from accumulating over multiple clock periods since they may be connected to nodes V_1 and V_2 asymmetrically depending on the comparator output. In this study, the capacitance of C_1 and C_2 is approximately 200 times greater than the parasitic capacitance of the ADC switches.

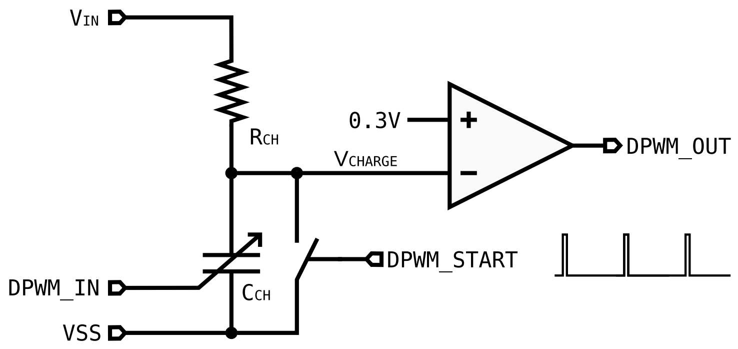
The bias current must ensure that capacitors C_1 and C_2 charge close to the limit of driving M_5, M_6, M_7, M_8 out of saturation for the worst process corner. This guarantees the utilization of the maximum available range for the difference between V_1 and V_2, making the circuit less sensitive to noise.

3.2 DPWM

In Section 2.7, the necessity of maintaining Q_{PWM} constant to avoid the requirement for a second ADC to digitize the input voltage is discussed. This involves varying Q_{DUTY} according to the input voltage, as expressed in Equation 2.17.

An approach commonly employed in analog buck converters is the feed-forward of the input voltage into the ramp generator [29]. In order to achieve similar functionality in a digital buck converter, the architecture illustrated in Figure 18 can be modified, making the current that charges the configurable capacitors dependent on the input voltage. Figure 65 presents a high-level block diagram of this proposed architecture.

Figure 65 – Block diagram of the proposed DPWM.



Source: The author.

The process of calculating the duty cycle starts with Equation 3.8, which indicates the time taken to charge the node $\mathrm{V_{CHARGE}}$ from 0 to 300 mV. Here, R_{CH} represents the resistor value, while C_{CH} denotes the total capacitance, dependent on the state of the digital input $DPWM\_IN$.

$$T_{CH} = -R_{CH} \cdot C_{CH} \cdot \ln\left(1 - \frac{0.3}{V_{IN}}\right) \tag{3.8}$$

The comparison with 300 mV was chosen because it is 9 times less than the minimum input voltage V_{IN}. Therefore, expanding Equation 3.8 using a Taylor series approximation and ignoring higher-order terms due to V_{IN} being significantly greater than 300 mV, Equation 3.9 is obtained.

$$T_{CH} \approx \frac{0.3 \cdot R_{CH} \cdot C_{CH}}{V_{IN}} \tag{3.9}$$

Assuming that C_{CH} consists of several unit capacitors (C_{UNIT}) connected in parallel based on the digital value written to the DPWM input ($DPWM\_IN$), Equation 3.9 can be expressed as Equation 3.10.

$$T_{CH} \approx DPWM\_IN \cdot \frac{0.3 \cdot R_{CH} \cdot C_{UNIT}}{V_{IN}} \tag{3.10}$$

The duty cycle can be computed as the ratio between T_{CH} and the switching period T_S, as demonstrated in Equation 3.11.

$$D = \frac{T_{CH}}{T_S} = DPWM\_IN \cdot \frac{0.3 \cdot R_{CH} \cdot C_{UNIT}}{V_{IN} \cdot T_S} \tag{3.11}$$

Additionally, by the definition of the duty cycle step size, Equation 3.12 can be derived.

$$D = DPWM\_IN \cdot Q_{DUTY} \tag{3.12}$$

Combining Equations 3.11 and 3.12, Equation 3.13 is obtained.

$$Q_{DUTY} = \frac{0.3 \cdot R_{CH} \cdot C_{UNIT}}{V_{IN} \cdot T_S} \tag{3.13}$$

Substituting Equation 3.13 into Equation 2.17 yields Equation 3.14, demonstrating that Q_{DPWM} remains independent of the input voltage in the proposed architecture.

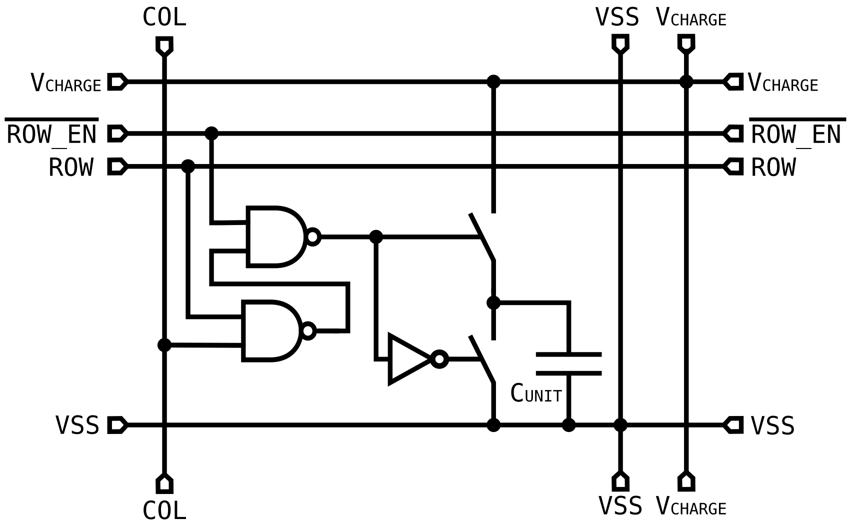
$$Q_{DPWM} = \frac{0.3 \cdot R_{CH} \cdot C_{UNIT}}{T_S} \tag{3.14}$$

The DNL can also be improved by changing the way the capacitors are connected. In Figure 18, you will notice that the capacitors are arranged in powers of 2, allowing direct connection of the switches to the binary input of the DPWM. However, consider the transition between input states 01111_{b} and 10000_{b}. This transition disconnects 15 unit capacitors from node $\mathrm{V_{CHARGE}}$ and connects 16 different unit capacitors to it. If there is significant difference between the parasitic capacitances in these two configurations, the capacitance may decrease rather than increase, resulting in negative DNL, which could impact the loop.

In order to mitigate this issue, a thermometer-encoded matrix-like connection inspired by the approach used in [30] can be employed.

This configuration involves structuring the unit capacitors as depicted in Figure 66. Each unit capacitor has three control signals: $\overline{\text{ROWN\_EN}}$ (active low), ROW (active high), and COL (active high). The $\overline{\text{ROWN\_EN}}$ signal activates the unit capacitor regardless of the state of other signals, whereas ROW and COL signals activate the unit capacitor only when both are active.

Figure 66 – Proposed schematic of the unit capacitor.



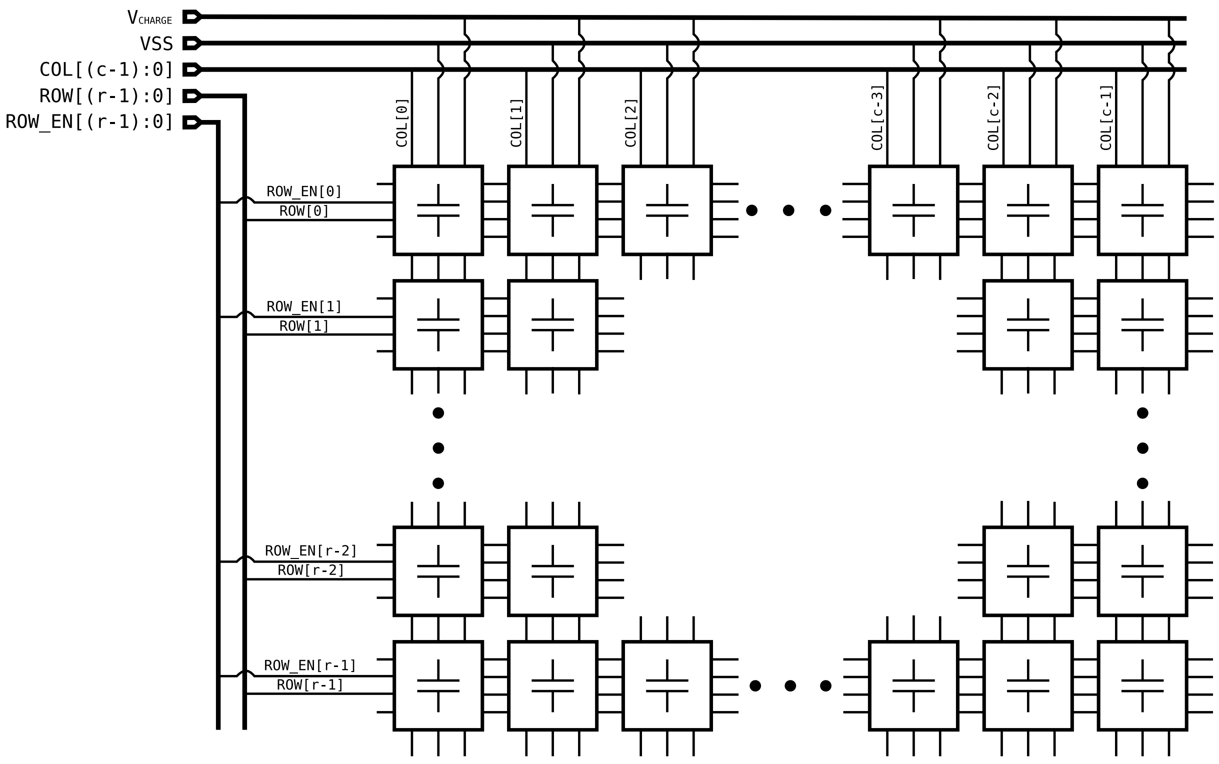
Source: The author.

Arranging all these unit cells in a matrix configuration, as shown in Figure 67, enables the activation of any number of capacitors using thermometer encoding [31]. For example, consider a matrix with r rows and c columns and suppose a total of $DPWM\_IN$ unit capacitors need to be activated, where $DPWM\_IN = c \cdot n + p$ and $p \leq c$, with n and p being natural numbers. In order to achieve this, the $\overline{\text{ROWN\_EN}}$ signals of the n first rows can be activated, the ROW signal of the subsequent row can be activated, and the COL signals can be used to activate the remaining p unit capacitors.

In this study, a matrix containing 12 rows by 16 columns of unit capacitors (192 unit capacitors) with minimum size was selected. This yields an estimated area (including switches and logic with an assumption of 60% area utilization) comparable to the total area reported for the DPWM developed in [17], which also utilizes a CMOS 180 nm technology. Additionally, the size of this matrix corresponds to 40% of the total area occupied by the ADC, DPWM, and digital PID reported in [10], although in this last case a CMOS 45 nm technology was employed.

In principle, the connection of the 192 unit capacitors to the node V_{CHARGE} must correspond to a duty cycle near 100% for the minimum input voltage in order to ensure a fast transient response. Consequently, a DPWM voltage step size (Q_{PWM}) can be obtained by dividing the minimum input voltage (2.7 V) by 192, which results in pproximately 14 mV. However, due to the parasitics, the capacitance between the nodes V_{CHARGE} and VSS will not be 0 when all 192 unit capacitors are disconnected. In addition to that, the

Figure 67 – Proposed block diagram of the capacitor matrix.



Source: The author.

comparator also has a propagation delay T_d. Therefore, Equation 3.11 can be rewritten as Equation 3.15, in which C_{PAR} represents the total value of the parasitic capacitance.

$$D = DPWM\_IN \cdot \frac{0.3 \cdot R_{CH} \cdot C_{UNIT}}{V_{IN} \cdot T_S} + \frac{0.3 \cdot R_{CH} \cdot C_{PAR}}{V_{IN} \cdot T_S} + \frac{T_d}{T_S} \tag{3.15}$$

Expressing the parasitic capacitance as a function of the number of unit capacitors $(OFFSET)$ leads to:

$$C_{PAR} = OFFSET \cdot C_{UNIT} \tag{3.16}$$

Substituting Equations 3.14 and 3.16 in Equation 3.15 gives:

$$D = (DPWM\_IN + OFFSET) \cdot \frac{Q_{DPWM}}{V_{IN}} + \frac{T_d}{T_S} \tag{3.17}$$

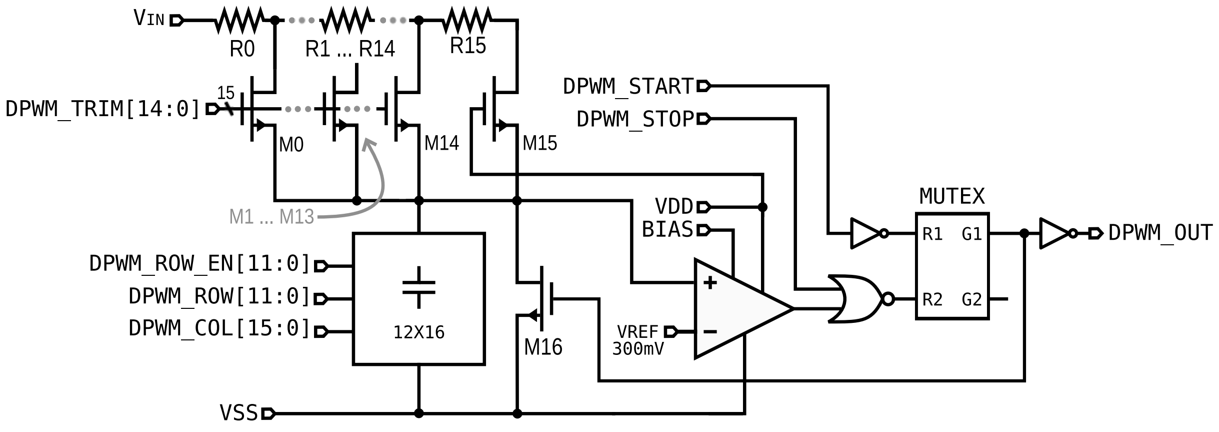
Note that the term $\frac{T_d}{T_S}$, when multiplied by the input voltage V_{IN}, produces a voltage offset dependent on V_{IN}, therefore minimization of the comparator propagation delay is necessary to reduce this dependency.

From Equation 3.17, it is evident that a voltage step size lower than 14 mV can be achieved for the worst corner, since the smallest step size across process corners must still ensure coverage of the maximum allowed duty cycle for the minimum input voltage.

Given the significant process variation, trimming of the RC constant becomes necessary to address these challenges (refer to Section 4.2.4 for details).

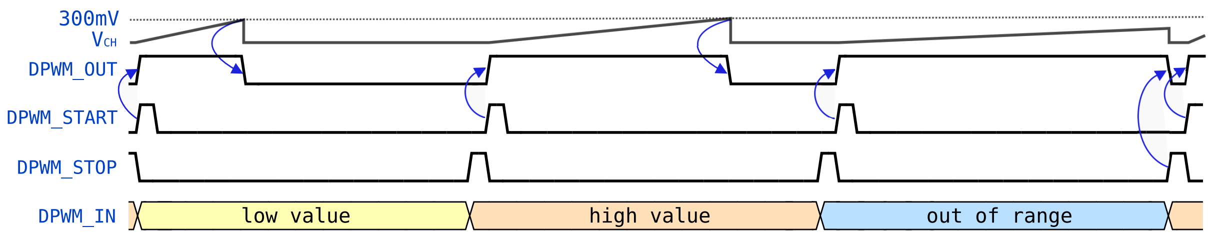
The schematic of the proposed DPWM is depicted in Figure 68. To enforce a maximum pulse width for the DPWM output, avoiding glitches, a signal to limit the maximum allowed duty cycle (DPWM_STOP) was incorporated. Given that the ADC requires a clock of 50 MHz, both DPWM_START and DPWM_STOP signals can be generated with one cycle of the 50 MHz clock. This configuration results in a minimum and maximum duty cycle of 4% and 96%, respectively, for a PWM of 2 MHz. The timing diagram illustrating these signals is presented in Figure 69, while Annex C shows the architecture of the comparator used during simulations. It is worth noting that the DPWM may function with different comparator architectures, as long as the propagation delay is substantially lower than the switching period T_S.

Figure 68 – Proposed DPWM circuit.



Source: The author.

Figure 69 – DPWM timing diagram.



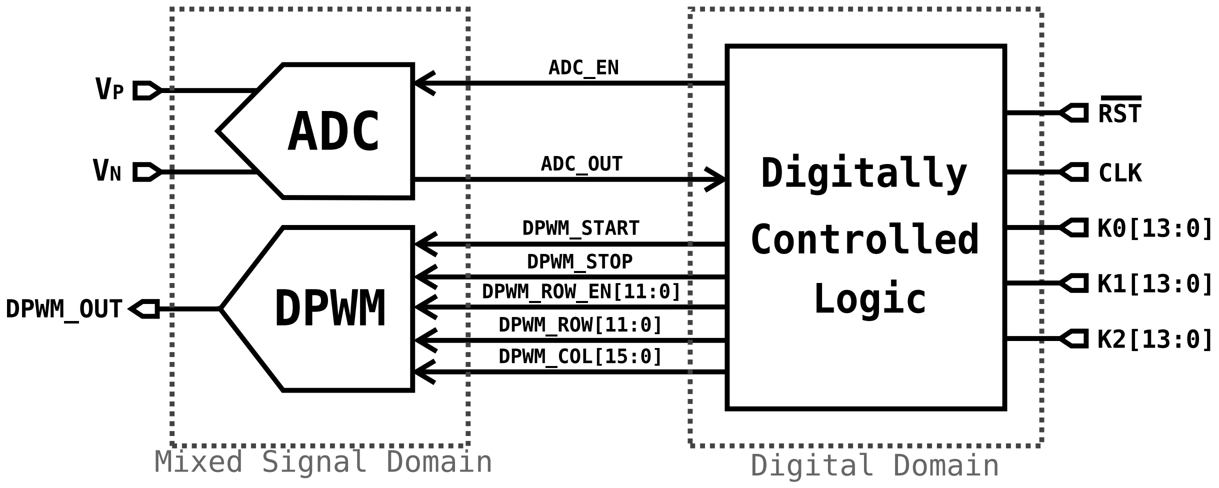
Source: The author.

3.3 Digitally Controlled Logic

In this section, the implementation of the digitally controlled logic is detailed. The logic is responsible for several tasks: controlling the ADC_EN signal, accumulating the

ADC output, handling the computation of the digital PID, and generating the DPWM control signals (DPWM_START and DPWM_STOP). Refer to Figure 70 for a block diagram illustrating how the digitally controlled logic interfaces with the ADC and the DPWM.

Figure 70 – Detailed block diagram of the digitally driven control.



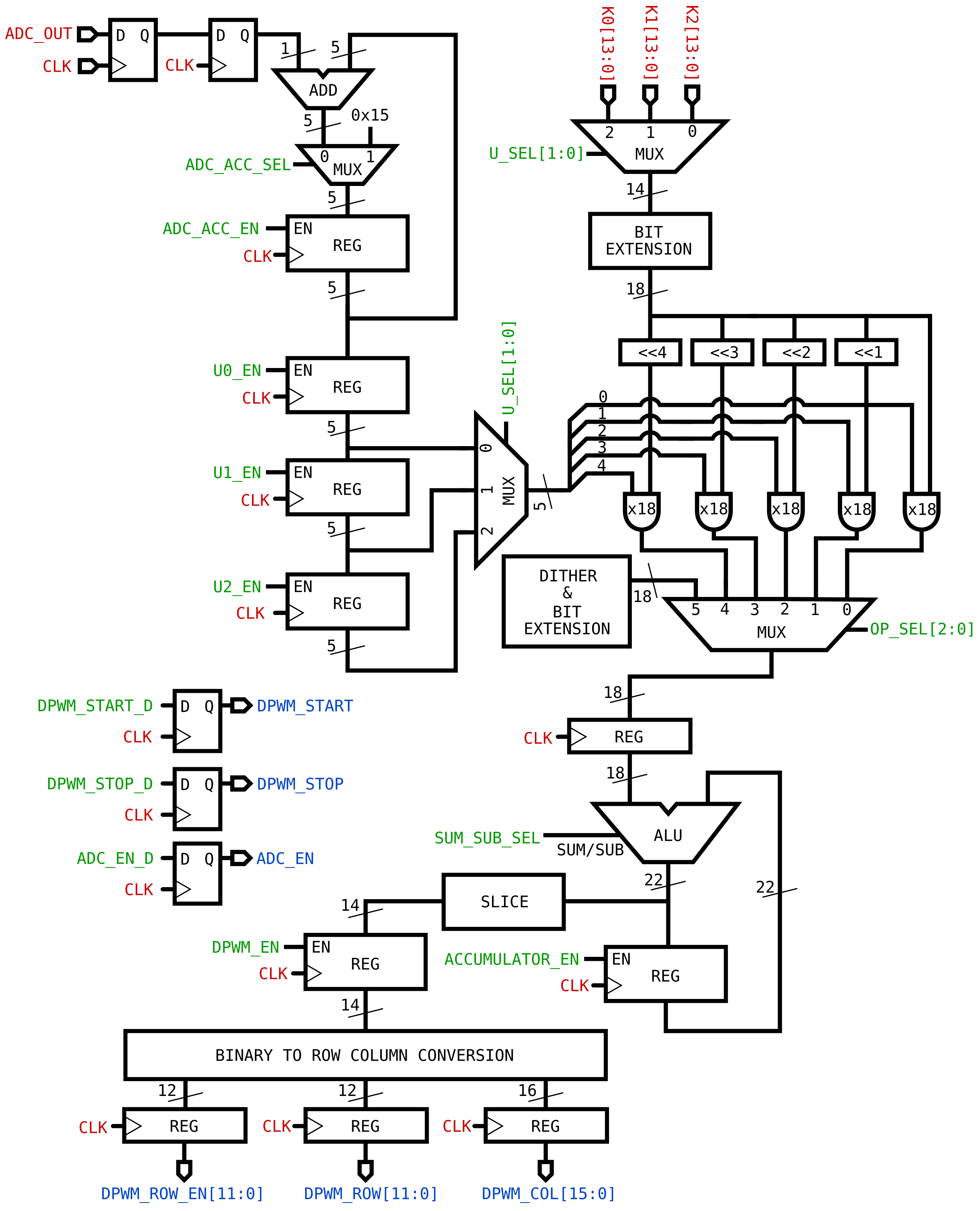
Source: The author.

Recall that Equation 2.4 can be represented in discrete time by Equation 2.13. This representation can be implemented using registers to store the last output and the last two inputs. The multiplication and addition operations can be achieved with combinational logic and lookup tables [9]. Alternatively, considering that the ADC requires a clock of at least 50 MHz, the same clock can be used to perform mathematical operations using registers, multiplexers, and shift logic with a single Arithmetic Logic Unit (ALU). A state machine can control the flow of data through this minimalist data path, executing the necessary operations every 25 clock cycles.

This approach results in the sampling of one input and the production of one output every 500 ns. Refer to Figure 71 for the proposed data path of the digitally controlled logic. In the figure, the blue signals represent outputs, the red signals are inputs, and the green signals are internal signals that the state machine can use to control the flow of data. The reset state of all registers, except for DPWM_STOP, is 0. The ALU utilize 21 bits with 8 bits to represent binary fractions, resulting in a granularity of $\frac{1}{256}$ or approximately 0.0039. Dithering, a technique used to artificially reduce the voltage step size of the DPWM, is employed using 4 of these 8 bits. This is necessary as the voltage step size of DPWM is higher than $\frac{Q_{ADC}}{G(0)}$ (refer to Section 2.7).

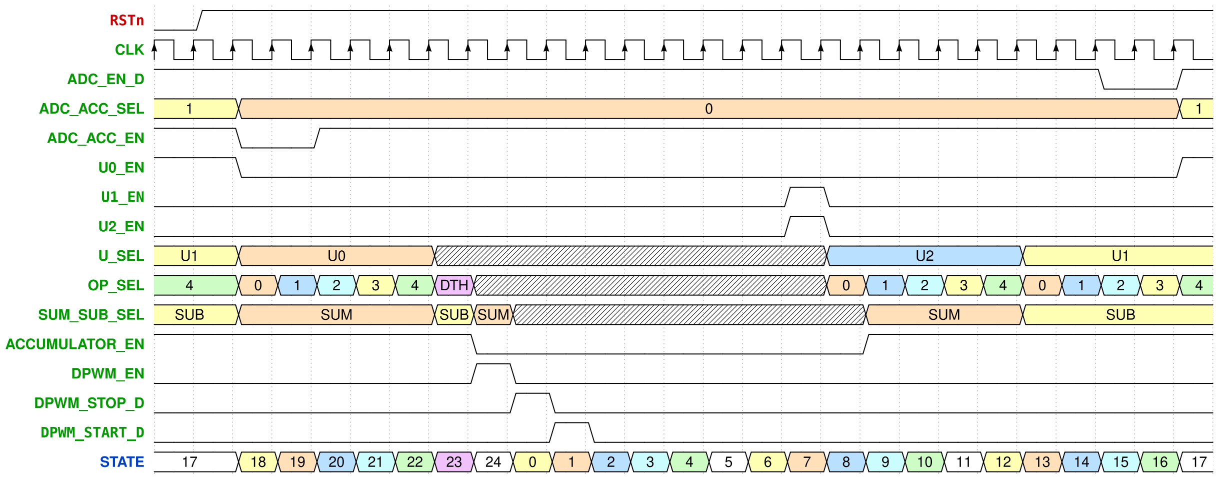
Figure 72 depicts the timing diagram of the state machine and illustrates the value of internal signals at each cycle. Refer to Annex E for more implementation details.

Figure 71 – Datapath of the digitally controlled logic.



Source: The author.

Figure 72 – Timing diagram of the digitally controlled logic state machine.



Source: The author.

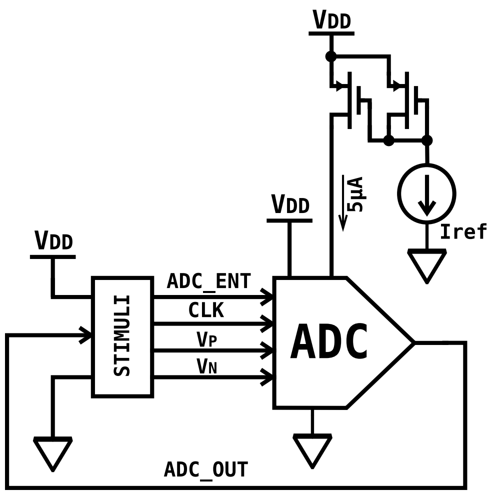
4 Results

This chapter presents the simulation results for the ADC, DPWM, and the entire system. The performances of the ADC and DPWM are given in the form of metrics as the step size, current, accuracy, DNL, and INL. The sensitivity of these metrics to supply, temperature, buck input voltage, process, and mismatch were simulated and the results were documented. Due to the long simulation time, the system level simulations include line, load, and voltage reference transients over temperature variation only, since effects of the process and mismatch in the control loop can be compensated by tuning the digital PID. A detailed analysis and discussions are provided in Chapter 5.

4.1 ADC

The ADC was simulated with a bias current of 5 µA, including a variation of ±25% over process corners. This current was generated by an ideal current source connected to the input of a PMOS current mirror. The output of the PMOS mirror was then connected to the ADC. This approach was employed to address mismatch effects and the limited range of a realistic current reference (see Figure 73).

Figure 73 – Diagram of the ADC test bench.



Source: The author.

The remaining stimuli was generated using Verilog-A code compiled from a Python script (refer to Annex F), utilizing the 'vagen' module developed by the author [32]. This code performs, in a transient simulation, the following sequence of actions:

- 1) The supply voltage is raised to the specified value for the corner.

- 2) A common mode voltage of half of the supply voltage is applied to the ADC inputs.
- 3) A 50 MHz clock is generated, and the ADC_EN signal is enabled for 22 clock cycles. The Verilog-A code will accumulate the number of occurrences of ADC_OUT being high during these cycles
- 4) To account for comparator mismatch effects (discussed in Section 3.1.2), the test bench deducts 3 from the accumulated result to obtain the digital output. This also avoids validating the ADC beyond the intended range, thus saving simulation time.
- 5) Steps 3 to 4 are repeated within a binary search algorithm in order to find the differential voltages that correspond to the digital codes 0 and 15. The step size, calculated as $\frac{V_{15}-V_0}{15}$, is determined, where V_i represents the differential voltage for code i.
- 6) Steps 3 to 4 are repeated, sweeping the differential voltages from V_0 to V_{15} in increments of one twentieth of the step size.

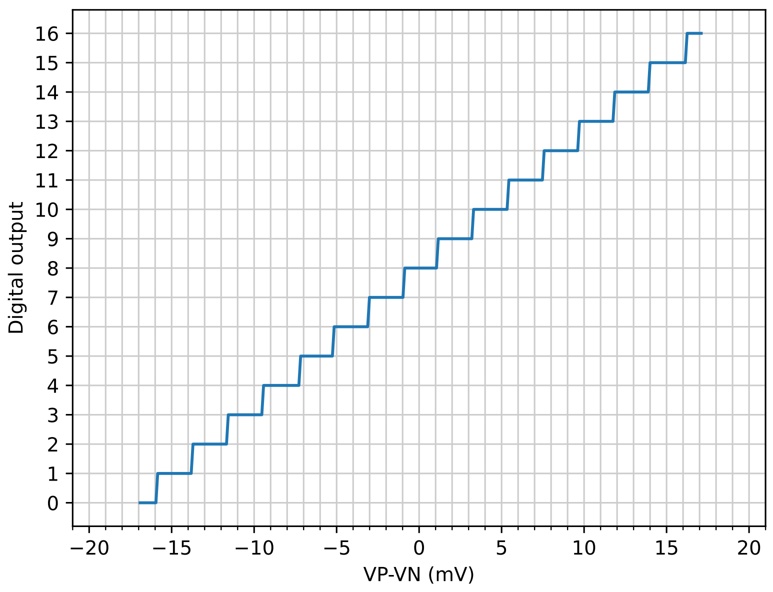
From this simulation, the ADC transfer curve and the following parameters were extracted:

- Q_{ADC}: The step size calculated by the Verilog-A code.
- I_Q: The average current flowing trough VSS during the 22 clock cycles when the ADC is enabled.
- $Accuracy$: The difference between the measured V_i and $(i-8) \cdot Q_{ADC}$, with Q_{ADC} representing the step size calculated by the verilog-A code.
- DNL: Based on Equation 1.38 and the step size calculated by the Verilog-A code. Note that, with the voltage increments used in the simulation, the DNL can only be calculated within an accuracy of ±0.05 LSB.
- INL: Based on Equation 1.39 and the step size calculated by the Verilog-A code.

4.1.1 Typical process, supply, and temperature

The ADC transfer curve for a typical supply voltage (VDD of 1.8 V) and a temperature of 25 °C is illustrated in Figure 74. Additionally, Table 20 presents a summary of the extracted parameters under the same conditions.

Figure 74 – Typical ADC transfer curve.



Source: The author.

Table 20 – Typical ADC Parameters.

Parameter	**Description**	**Value**	**Unit**
Accuracy	Difference between input voltage and the ideal curve	±100	μV
I_Q	Average current flowing trough VSS	15.490	μA
Q_{ADC}	Step size	2.136	mV
DNL	Differential non-linearity	±0.025	LSB
INL	Integral non-linearity	±0.05	LSB

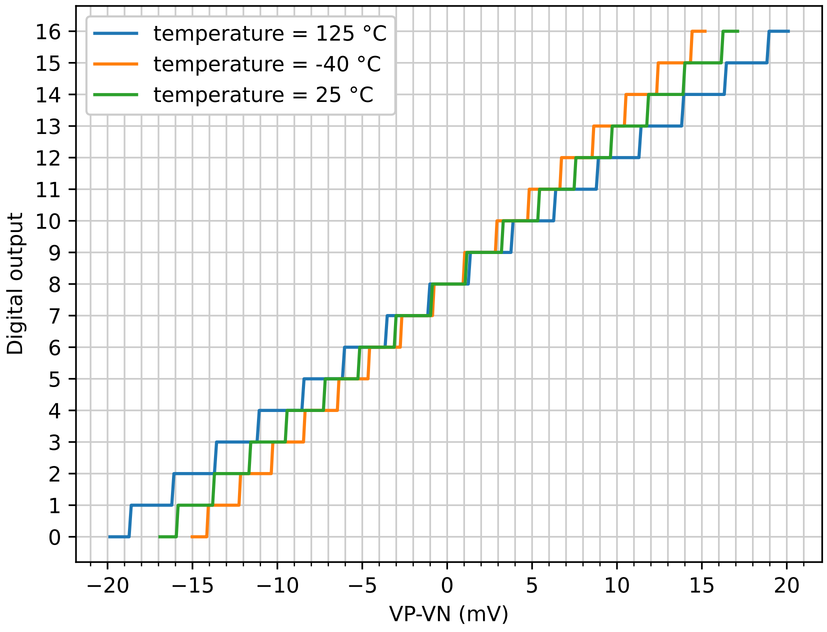
4.1.2 Dependency on temperature

The ADC transfer curves for a typical supply voltage (VDD of 1.8 V) and temperatures of -40 °C, 25 °C, and 125 °C is illustrated in Figure 75. Additionally, Table 21 presents a summary of the extracted parameters under the same conditions.

Table 21 – ADC Parameters with temperature variation.

Parameter	**Description**	**-40 °C**	**25 °C**	**125 °C**	**Unit**
Accuracy	Difference between input voltage and the ideal curve	±100	±100	±100	μV
I_Q	Average current flowing trough VSS	14.220	15.490	17.170	μA
Q_{ADC}	Step size	1.897	2.136	2.509	mV
DNL	Differential non-linearity	±0.025	±0.025	±0.025	LSB
INL	Integral non-linearity	±0.05	±0.05	±0.05	LSB

Figure 75 – ADC transfer curve with temperature variation.

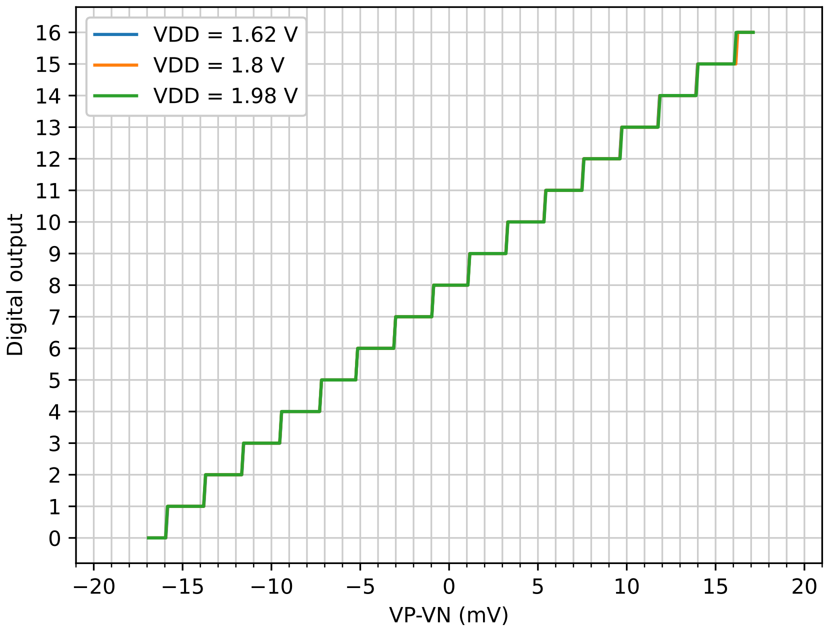


Source: The author.

4.1.3 Dependency on supply

The dependency on the supply voltage is highlighted by the ADC transfer curves illustrated in Figure 76, which were obtained for a temperature of 25 °C and supply voltages (VDD) of 1.62 V, 1.8 V, and 1.92 V. Additionally, Table 22 provides a summary of the extracted parameters under these conditions.

Figure 76 – ADC transfer curve with supply variation.



Source: The author.

Table 22 – ADC Parameters with supply variation.

Parameter	**Description**	**1.62 V**	**1.8 V**	**1.92 V**	**Unit**
$Accuracy$	Difference between input voltage and the ideal curve	±100	±100	±100	μV
I_Q	Average current flowing trough VSS	14.920	15.490	16.08	μA
Q_{ADC}	Step size	2.135	2.136	2.138	mV
DNL	Differential non-linearity	±0.025	±0.025	±0.025	LSB
INL	Integral non-linearity	±0.05	±0.05	±0.05	LSB

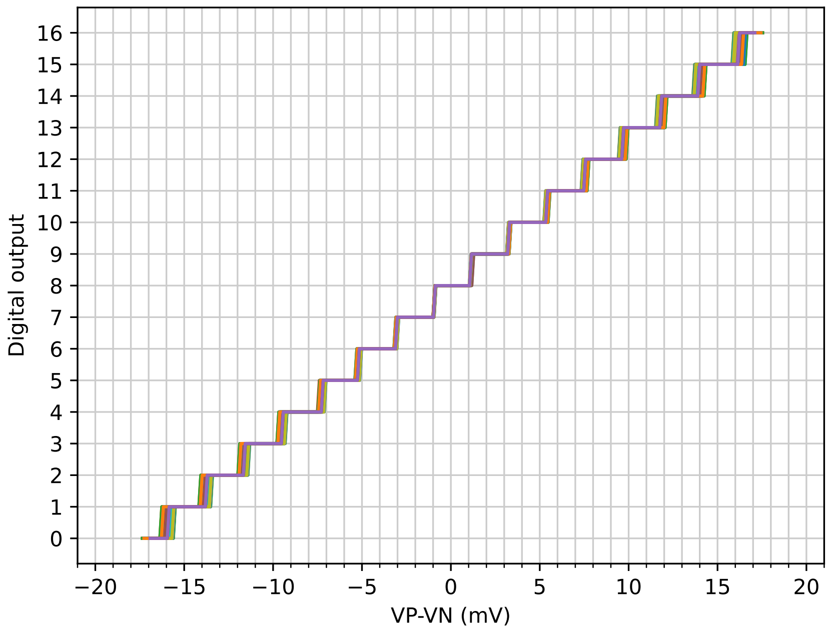
4.1.4 Dependency on process

The ADC transfer curves for a typical supply voltage (VDD of 1.8 V) and a temperature of 25 °C across process corners are depicted in Figure 77. Additionally, Table 23 provides a summary of the extracted parameters under these conditions.

Table 23 – ADC Parameters with process variation.

Parameter	**Description**	**Min.**	**Max.**	**Unit**
$Accuracy$	Difference between input voltage and the ideal curve	-100	100	μV
I_Q	Average current flowing trough VSS	13.4	18.5	μA
Q_{ADC}	Step size	2.093	2.191	mV
DNL	Differential non-linearity	-0.025	0.025	LSB
INL	Integral non-linearity	-0.05	0.05	LSB

Figure 77 – ADC transfer curve with process variation.

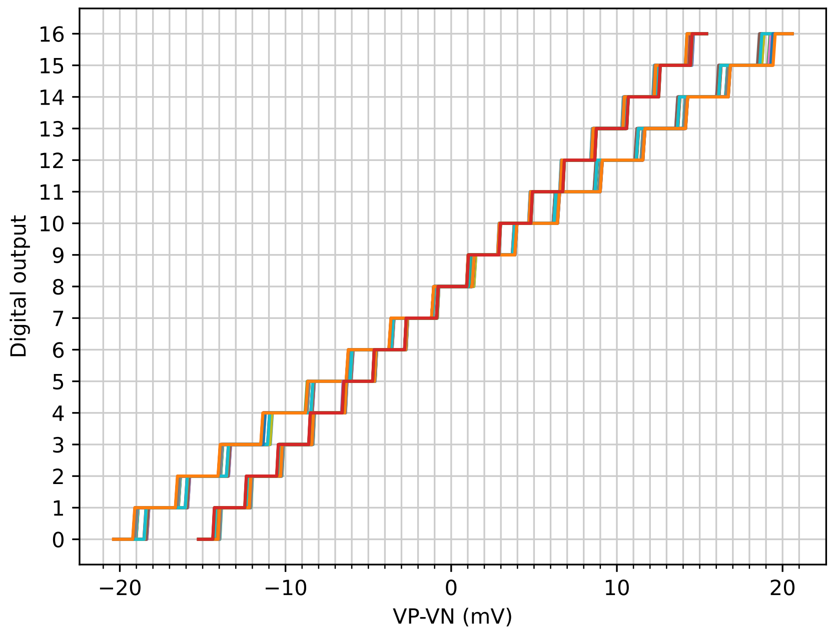


Source: The author.

4.1.5 Dependency on process, supply, and temperature

Varying the supply (VDD of 1.62 V and 1.98 V), the temperature (-40 °C and 125 °C), and the process corners yields the transfer curve illustrated in Figure 78 and the summary of the extracted parameters shown in Table 24.

Figure 78 – ADC transfer curve with process, supply, and temperature variation.



Source: The author.

Table 24 – ADC Parameters with process, supply, and temperature variation.

Parameter	**Description**	**Min.**	**Max.**	**Unit**
$Accuracy$	Difference between input voltage and the ideal curve	-100	100	μV
I_Q	Average current flowing trough VSS	11.810	21.360	μA
Q_{ADC}	Step size	1.872	2.573	mV
DNL	Maximum absolute value of the differential non-linearity	-0.025	0.025	LSB
INL	Maximum absolute value of the Integral non-linearity	-0.05	0.05	LSB

4.1.6 Sensitivity to mismatch

Sensitivity to mismatch was assessed via a Monte Carlo simulation, involving 100 runs utilizing the Latin Hypercube Sampling (LHS) method, under typical supply (VDD of 1.8 V) and temperature of 25 °C. The resulting standard deviation is summarized in Table 25.

Table 25 – Standard deviation of the ADC parameters taking into account the mismatch.

Parameter	**Description**	**Standard Deviation**	**Unit**
$Accuracy_{MAX}$	Maximum absolute difference between input voltage and the ideal curve	2.53	mV
I_Q	Average current flowing trough VSS	272	nA
Q_{ADC}	Step size	48.9	μV
DNL_{MAX}	Maximum absolute value of the differential non-linearity	0.0133	LSB
INL_{MAX}	Maximum absolute value of the integral non-linearity	0.0343	LSB

4.1.7 Estimated Area

The active area of the ADC was estimated using the auto-placement feature of the Layout-XL® tool, with 60% area utilization, leaving 40% of the total area available for routing. No violations of the minimum distance between components were detected by the Design Rule Check (DRC). Under these conditions, the estimated area measures 94.2 µm by 94.2 µm.

4.2 DPWM

The DPWM was simulated with a bias current of 1 µA for the comparator, with a variation of ±25% considered over process corners. As in the case of the ADC, this current was produced by an ideal current source connected to the input of a PMOS current mirror and the output of the PMOS mirror was linked to the DPWM. This approach was employed to include mismatch effects and the limited range of a realistic current reference.

A voltage reference of 300 mV was established using an ideal voltage source in series with a resistor. The resistance was varied within 1 kΩ to 100 kΩ to assess the sensitivity of the proposed circuit to the impedance of the voltage reference (see Figure 79).

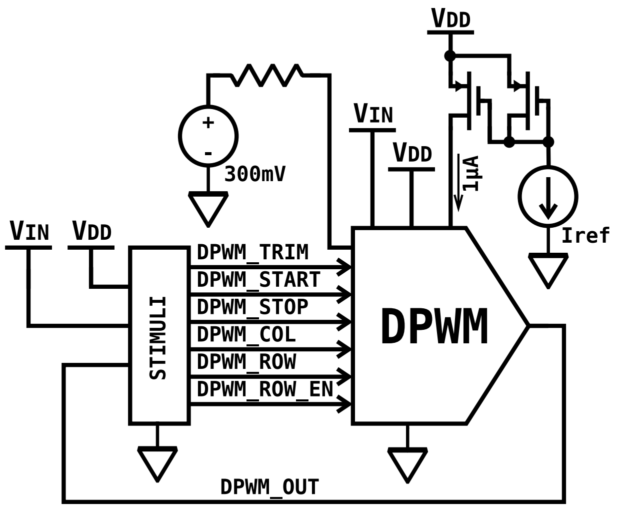
Similar to the ADC setup, the additional stimuli for the DPWM were generated through Verilog-A code, compiled from a Python script (see Annex G), employing the 'vagen' module. This code executes the following sequence of actions in a transient simulation:

- 1) Load 0 into $DPWM\_IN$ (converting $DPWM\_IN$ to the required ROW, ROW_EN, and COL signals).
- 2) Trigger a pulse on the DPWM_START pin.
- 3) Wait for 460 ns (which is equal to the switching period minus two periods of the 50 MHz clock in order discount the DPWM_START and DPWM_STOP cycle

duration).

- 4) Trigger a pulse on the DPWM_STOP pin.
- 5) If the DPWM_OUT signal transitions low immediately after DPWM_START transitions low, or just before DPWM_STOP transitions high, the Verilog-A code will not consider the pulse as valid, allowing it to be ignored during plotting.
- 6) Increment the $DPWM\_IN$ by one and repeat steps 2-6 until all 192 unit capacitors are connected.

Figure 79 – Diagram of the DPWM test bench.



Source: The author.

From this simulation, the DPWM transfer curve and the following parameters were extracted:

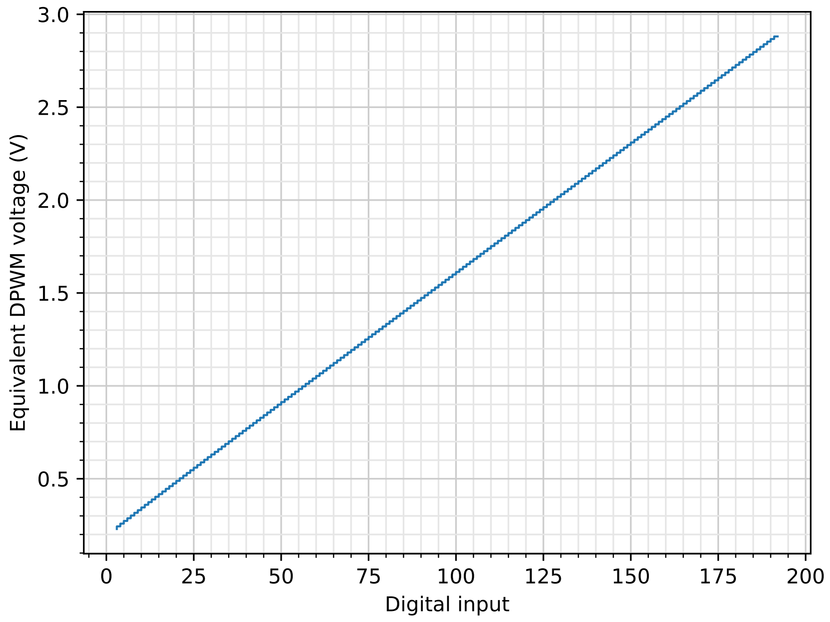
- Q_{DPWM}: This parameter represents the DPWM voltage step size and it is calculated as $Q_{DPWM} = Q_{DUTY} \cdot V_{IN}$, where Q_{DUTY} is the difference between the maximum and minimum valid duty cycle obtained during the sweep, divided by the difference between the corresponding $DPWM\_IN$ values. V_{IN} is the input voltage of the buck.
- I_Q: This parameter denotes the average current flowing through VSS during operation.
- *Accuracy*: It quantifies the difference between the equivalent output voltage (which is the product of the measured duty cycle D and the input voltage V_{IN}) and $(DPWM\_IN + 12.5) \cdot Q_{DPWM}$, where Q_{DPWM} represents the voltage step size. Here, 12.5 represents the average offset resulting from parasitic capacitance of the switches and the comparator delay, which was obtained trough linear regression of the typical curve (Figure 80).

- DNL: This parameter is computed utilizing the voltage step size and extending the usage of Equation 1.38 also for the DPWM.
- INL: Similar to DNL, this parameter is computed utilizing the voltage step size and extending the usage of Equation 1.39 also for the DPWM.

4.2.1 Typical process, supply, buck input voltage, and temperature

The DPWM transfer curve for a temperature of 25 °C, with typical comparator supply (VDD of 1.8 V) and buck input voltage (V_{IN} of 3.7 V), is depicted in Figure 80. Additionally, Table 26 provides a summary of the extracted parameters under these conditions.

Figure 80 – Typical DPWM transfer curve ($D \cdot V_{IN}$).



Source: The author.

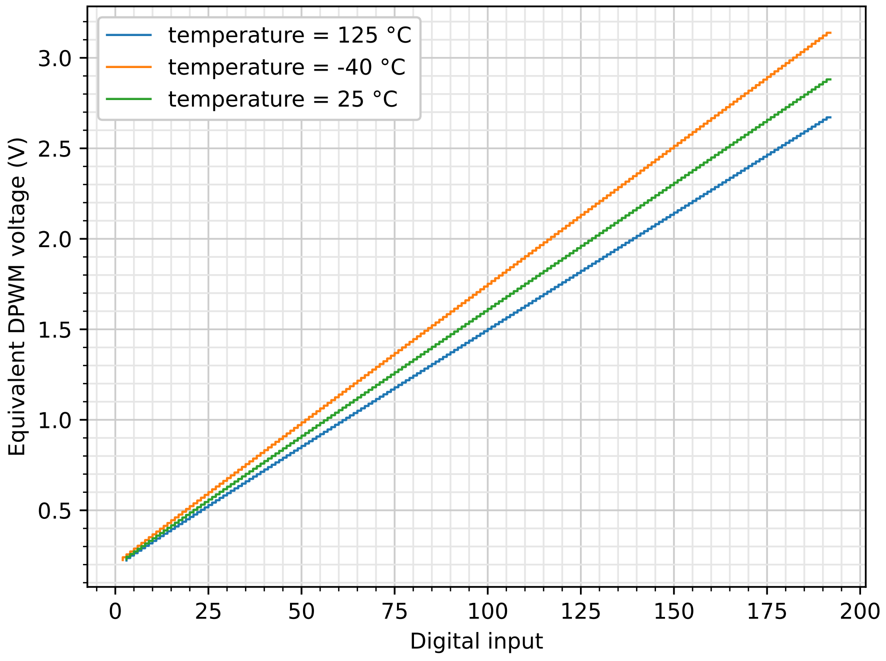
Table 26 – Typical DPWM Parameters.

Parameter	**Description**	**Value**	**Unit**
$Accuracy$	Difference between the output voltage given by the DPWM and the ideal curve	±5.4	mV
I_Q	Average current flowing trough VSS	35.1	μA
Q_{DPWM}	Voltage step size	14.0	mV
DNL	Differential non-linearity	0.01±0.04	LSB
INL	Integral non-linearity	0.4±0.4	LSB

4.2.2 Dependency on temperature

The DPWM transfer curves for a temperatures of -40 °C, 25 °C, and 125 °C, with typical comparator supply (VDD of 1.8 V) and buck input voltage (V_{IN} of 3.7 V), are depicted in Figure 81. Additionally, Table 27 provides a summary of the extracted parameters under these conditions.

Figure 81 – DPWM transfer curve with temperature variation $(D \cdot V_{IN})$.



Source: The author.

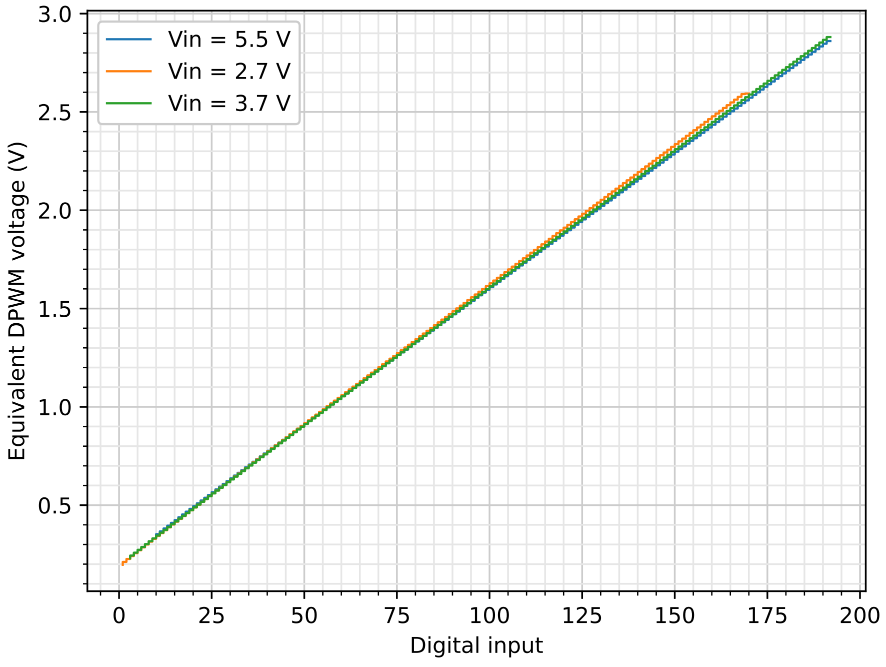
Table 27 – DPWM Parameters with temperature variation.

Parameter	**Description**	**-40 °C**	**25 °C**	**125 °C**	**Unit**
$Accuracy$	Difference between the output voltage given by the DPWM and the ideal curve	-10.0±5.4	±5.4	10.0±5.4	mV
I_Q	Average current flowing trough VSS	32.4	35.1	39.3	μA
Q_{DPWM}	Voltage step size	15.3	14.0	12.9	mV
DNL	Differential non-linearity	-0.01±0.03	0.01±0.04	0.01±0.04	LSB
INL	Integral non-linearity	0.3±0.3	0.4±0.4	0.45±0.45	LSB

4.2.3 Dependency on the buck input voltage

The dependency on the supply voltage is demonstrated by the DPWM transfer curves shown in Figure 82, obtained at a temperature of 25 °C and buck input voltages (V_{IN}) of 2.7 V, 3.7 V, and 5.5 V. The comparator supply voltages (VDD) for these cases were 1.62 V, 1.8 V, and 1.92 V, respectively. Additionally, Table 28 presents a summary of the extracted parameters under these specified conditions

Figure 82 – DPWM transfer curve with buck input voltage variation ($D \cdot V_{IN}$).



Source: The author.

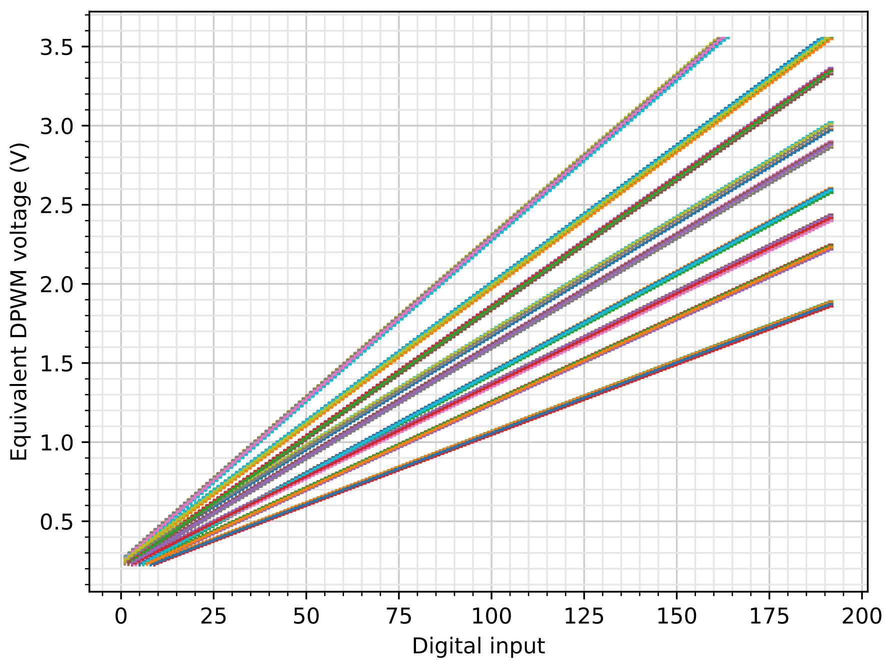
Table 28 – DPWM Parameters with buck input voltage variation.

Parameter	**Description**	**2.7V**	**3.7V**	**5.5V**	**Unit**
Accuracy	Difference between the output voltage given by the DPWM and the ideal curve	-8.0±4.0	±5.4	15.0±5.4	mV
I_Q	Average current flowing trough VSS	32.2	35.1	39.6	μA
Q_{DPWM}	Voltage step size	14.2	14.0	13.8	mV
DNL	Differential non-linearity	0.01±0.03	0.01±0.04	0.01±0.04	LSB
INL	Integral non-linearity	0.3±0.3	0.4±0.4	0.4±0.4	LSB

4.2.4 Dependency on process

The DPWM transfer curves for a temperature of 25 °C, typical comparator supply (VDD of 1.8 V), and typical buck input voltage (V_{IN} of 3.7 V) across process corners are illustrated in Figure 83. Additionally, Table 29 provides a summary of the extracted parameters under these conditions.

Figure 83 – DPWM transfer curve including process variation before trimming ($D \cdot V_{IN}$).



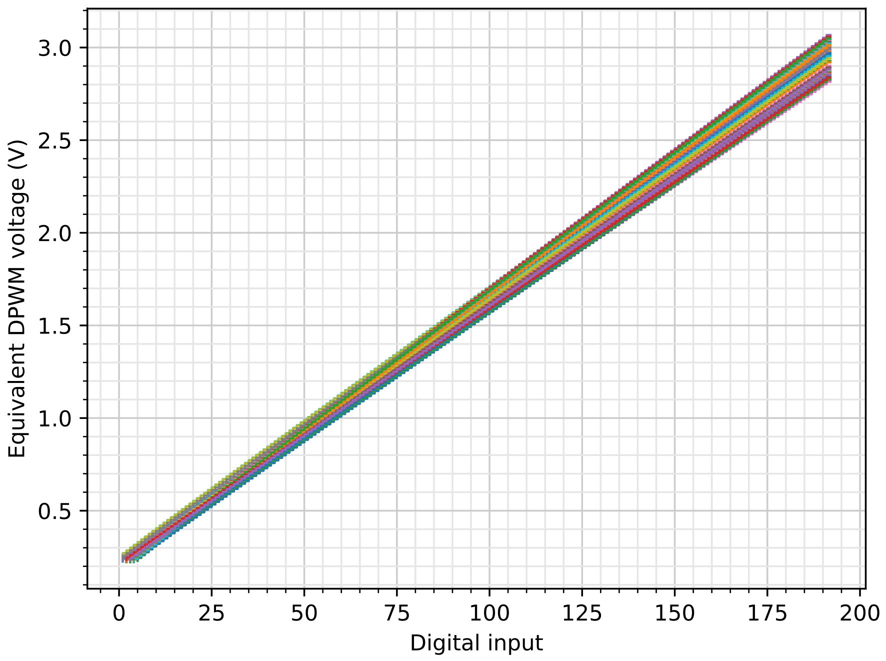
Source: The author.

Table 29 – DPWM Parameters with process variation before trimming.

Parameter	**Description**	**Min.**	**Max.**	**Unit**
Accuracy	Difference between the output voltage given by the DPWM and the ideal curve	-61.8	54.3	mV
I_Q	Average current flowing trough VSS	28.5	45.0	μA
Q_{DPWM}	Voltage step size	8.9	20.5	mV
DNL	Differential non-linearity	-0.03	0.06	LSB
INL	Integral non-linearity	0.0	0.9	LSB

It is worth noting that the process variation of the RC constant significantly influences the step size. To mitigate this issue, multiple resistors were added in parallel and they can be configured to compensate for the process variation. In order to validate the variation after trimming process, a trimming test sequence was incorporated into the stimuli (see Annex G). Utilizing the Cadence® Maestro software, the trimming sequence can be executed in one transient simulation, and the trimming value can be transferred to the transient simulation that performs the sweep. This transfer is made possible by the 'calcVal' function. The results from this procedure are depicted in Figure 84 and summarized in Table 30.

Figure 84 – DPWM transfer curve including process variation after trimming ($D \cdot V_{IN}$).



Source: The author.

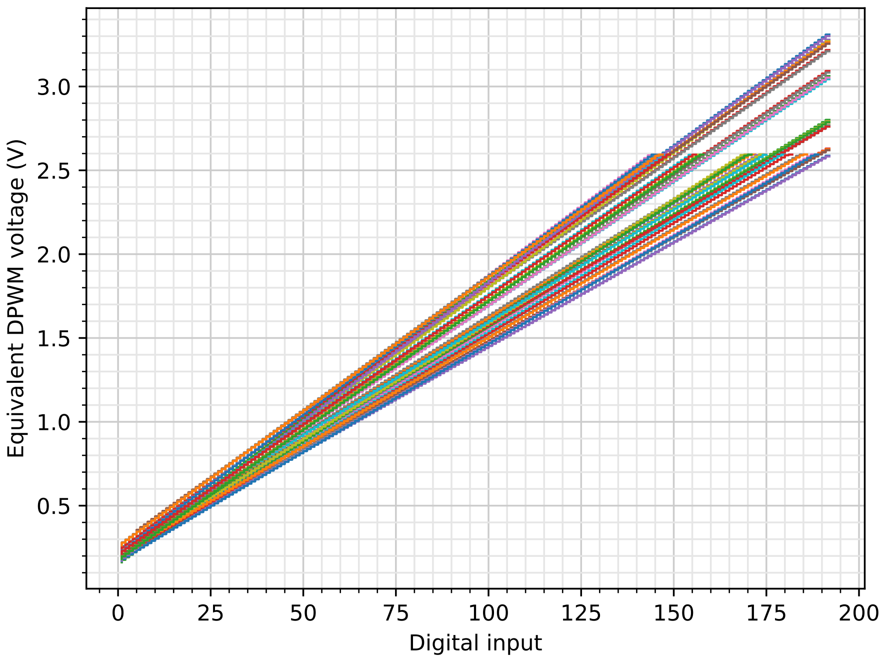
Table 30 – DPWM Parameters with process variation after trimming.

Parameter	Description	Min.	Max.	Unit
$Accuracy$	Difference between the output voltage given by the DPWM and the ideal curve	-51.1	54.3	mV
I_Q	Average current flowing trough VSS	28.5	43.6	μA
Q_{DPWM}	Voltage step size	13.7	15.0	mV
DNL	Differential non-linearity	-0.03	0.06	LSB
INL	Integral non-linearity	0.0	0.9	LSB

4.2.5 Dependency on process, buck input voltage, supply, and temperature

Variations in the comparator supply voltage (VDD of 1.62 V and 1.98 V), the buck input voltage (V_{IN} of 2.7 V and 5.5 V), the temperature (-40 °C and 125 °C), and the process corners result in the transfer curve depicted in Figure 85, along with a summary of the extracted parameters presented in Table 31. The process corner was trimmed by executing the trimming sequence (see Section 4.2.4) at a temperature of 25 °C, with a typical comparator supply (VDD of 1.8 V), and a typical buck input voltage (V_{IN} of 3.7 V). These conditions are assumed to be the default for testing and calibrating the chip before it is dispatched to the customer.

Figure 85 – DPWM transfer curve with process (after trimming), supply, buck input voltage, and temperature variation ($D \cdot V_{IN}$).



Source: The author.

Table 31 – DPWM Parameters with process (after trimming), supply, and temperature variation.

Parameter	**Description**	**Min.**	**Max.**	**Unit**
$Accuracy$	Difference between the output voltage given by the DPWM and the ideal curve	-72.6	84.1	mV
I_Q	Average current flowing trough VSS	24.0	55.2	μA
Q_{DPWM}	Voltage step size	12.5	16.7	mV
DNL	Differential non-linearity	-0.06	0.13	LSB
INL	Integral non-linearity	0.0	1.1	LSB

4.2.6 Sensitivity to mismatch

Sensitivity to mismatch was assessed via a Monte Carlo simulation, involving 200 runs utilizing the Latin Hypercube Sampling (LHS) method, under a temperature of 25 °C, with a typical comparator supply (VDD of 1.8 V), and a typical buck input voltage (V_{IN} of 3.7 V). The resulting standard deviation is summarized in Table 32.

4.2.7 Estimated Area

The active area of the DPWM was estimated using the auto-placement feature of the Layout-XL® tool, with 60% area utilization, leaving 40% of the total area available for routing. No violations of the minimum distance between components were detected by the

Design Rule Check (DRC). Under these conditions, the estimated area measures 209.5 µm by 209.5 µm.

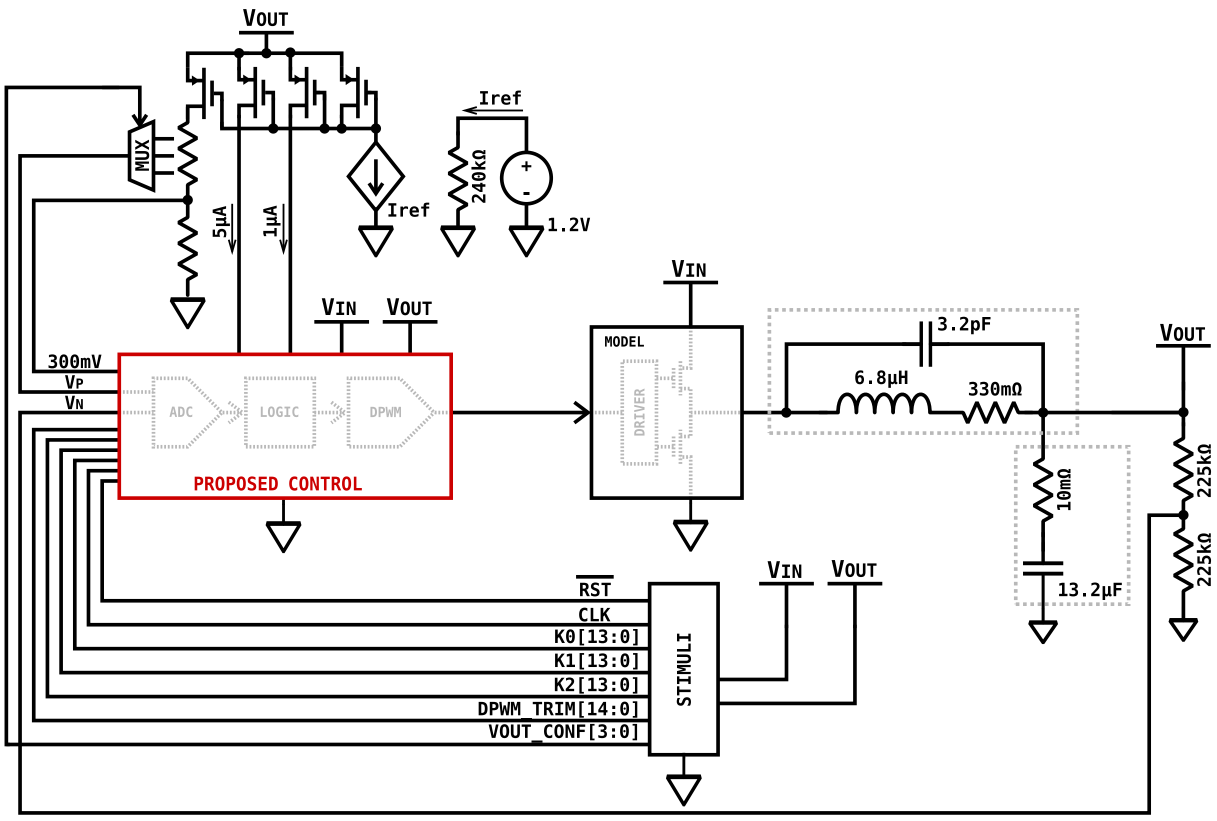
Table 32 – Standard deviation of the DPWM Parameters taking into account the mismatch.

Parameter	**Description**	**Standard Deviation**	**Unit**
$Accuracy_{MAX}$	Maximum absolute difference between the output voltage given by the DPWM and the ideal curve	2.5	mV
I_Q	Average current flowing trough VSS	1.1	μA
Q_{DPWM}	Voltage step size	100	μV
DNL_{MAX}	Maximum absolute value of the differential non-linearity	0.006	LSB
INL_{MAX}	Maximum absolute value of the integral non-linearity	0.068	LSB

4.3 System level simulations

The block diagram of the test bench for system-level simulations is depicted in Figure 86.

Figure 86 – Diagram of the system level test bench.



Source: The author.

The Verilog-A model of the power switch (see Annex A), the schematic of the ADC, the schematic of the DPWM, and the synthesizable Verilog model of the digitally controlled logic (see Annex E) were used. The simplified models for the inductor and capacitor were extrapolated using the SimSurfing tool from Murata, since the complete AC model is not suitable for transient simulations.

Although the design of the voltage reference was not the focus of this work, it is assumed that it can be configured. Several studies address this topic. For example, the work in [33] outputs a variable current that, when connected to the same type of resistor used in the bandgap circuit, generates a temperature-compensated configurable voltage reference. The bias current that this circuit outputs depends on the temperature coefficient and the characteristics of the resistor used in the bandgap.

A similar bias was emulated by utilizing a current-controlled current source, a PMOS mirror, an ideal voltage source, and resistors. The purpose of this emulation was to obtain a qualitative method for detecting secondary issues in the proposed circuit. For example, since the ADC is clocked, kickback due to parasitic capacitance could affect the reference, jeopardizing the correct operation of the circuit. Although no problems were detected, this test bench provided a better alternative than directly using ideal voltage and current sources, allowing the early detection of potential issues.

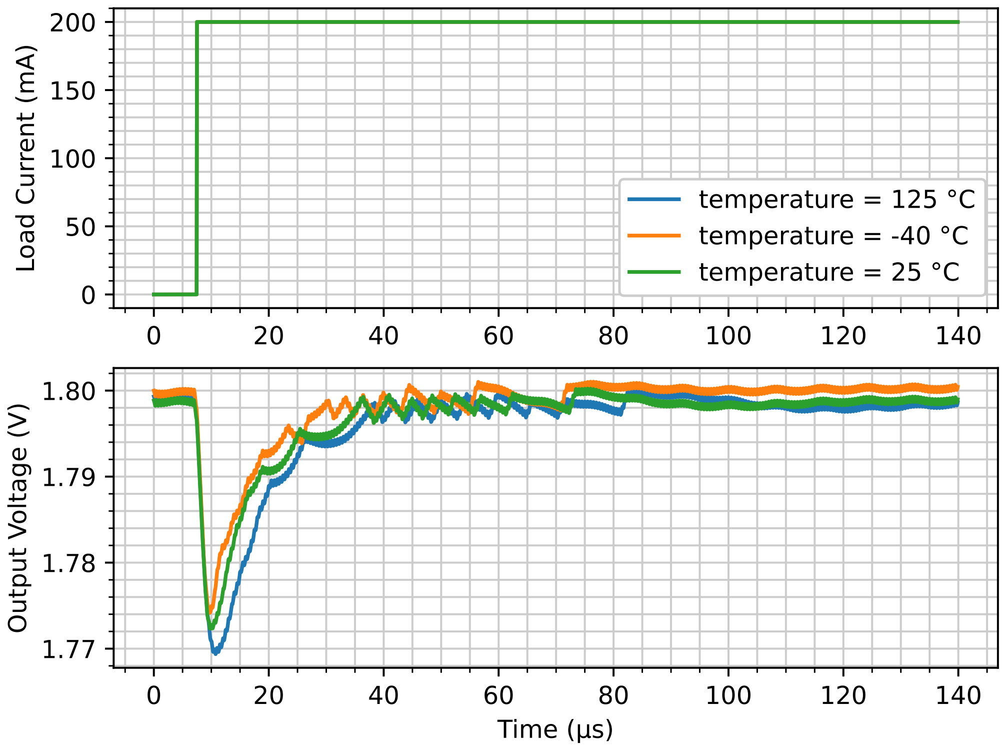
Similar to the ADC setup, additional stimuli for the system were generated through Verilog-A code, compiled from a Python script (see Annex H) employing the 'vagen' module. From a high-level perspective, this code executes the following sequence of actions in a transient Verilog-AMS simulation:

- Raise the input of the buck converter (V_{IN}) to 3.7 V and set the output (V_{OUT}) to 1.8 V.
- Release the output of the buck converter and allow it to be maintained by the capacitor.
- Start the clock and release the reset of the digital logic.
- Wait for 300 µs.
- Apply a step change in the load current, input voltage, or voltage reference, depending on the specific test intended.

4.3.1 Load Transient

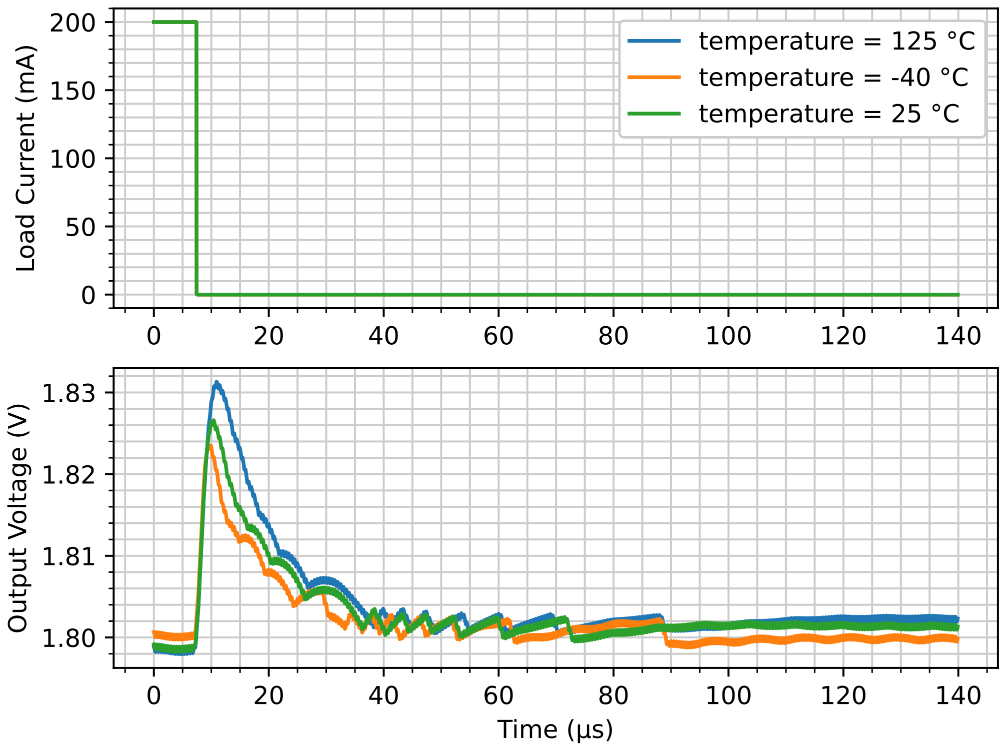
Sensitivity to fast load transients are depicted in Figures 87 and 88 while the sensitivity to load transients slopes compatible with the bandwidth of the system is shown in Figures 89 and 90.

Figure 87 – Output voltage undershoot when the load current varies from 0 A to 200 mA in 100 ns.



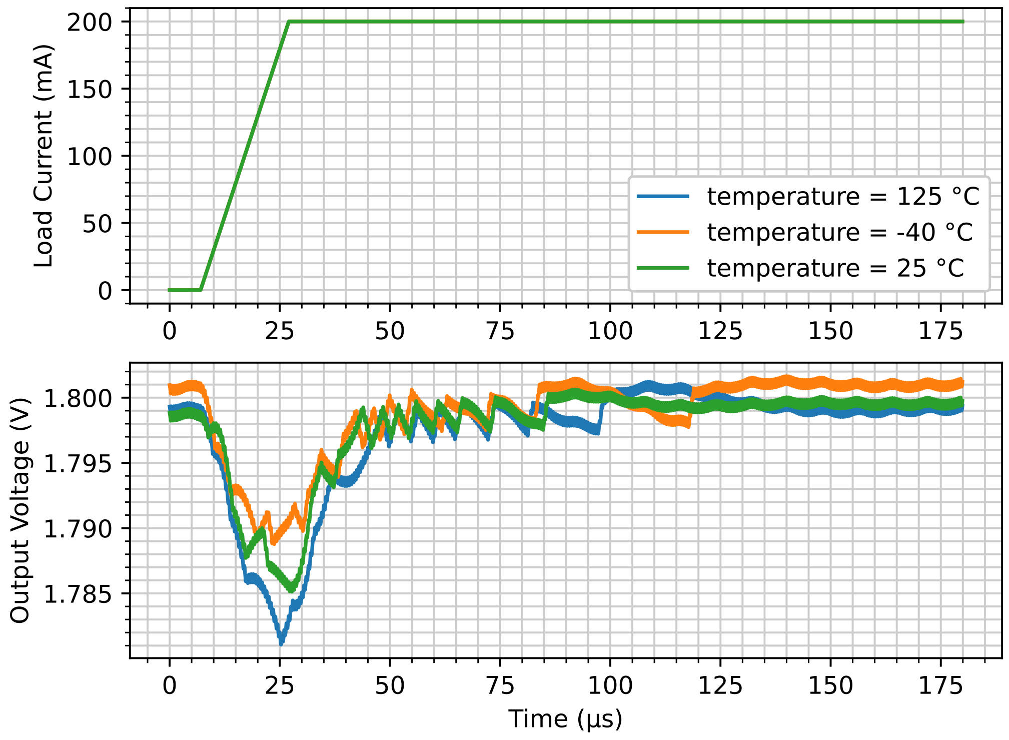
Source: The author.

Figure 88 – Output voltage overshoot when the load current varies from 200 mA to 0 A in 100 ns.



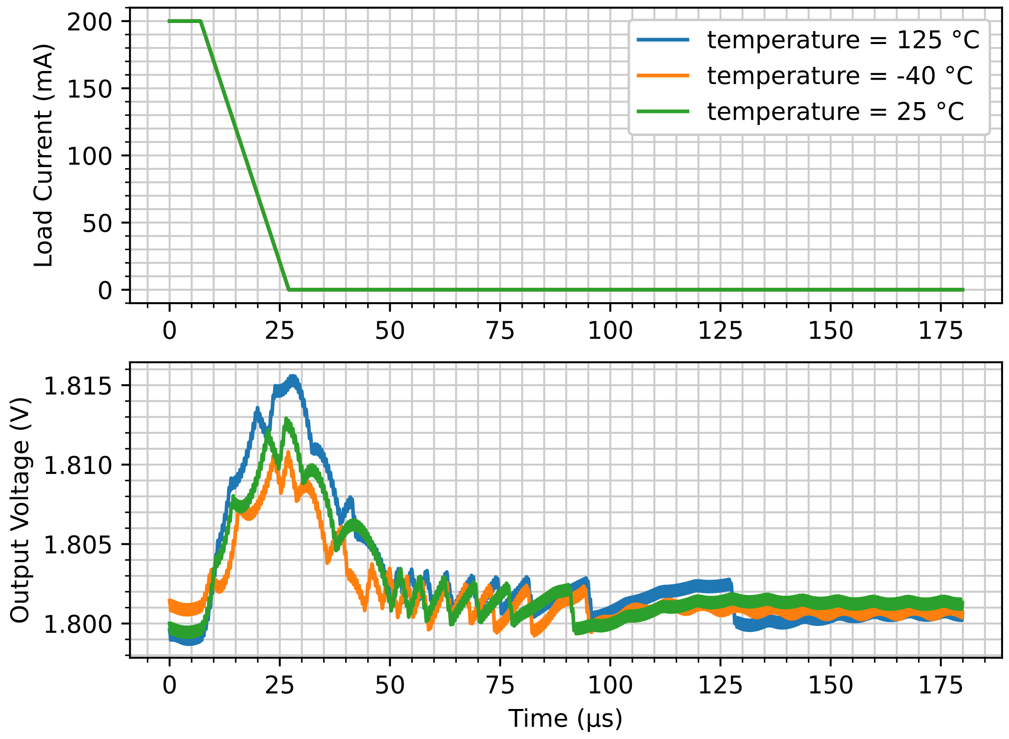
Source: The author.

Figure 89 – Output voltage undershoot when the load current varies from 0 A to 200 mA in 20 µs.



Source: The author.

Figure 90 – Output voltage overshoot when the load current varies from 200 mA to 0 A in 20 µs.

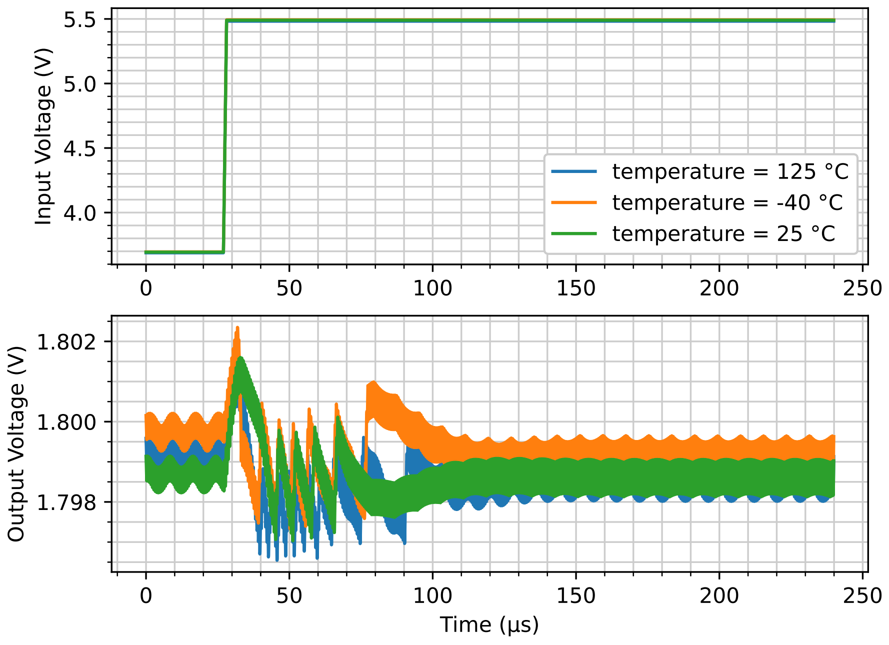


Source: The author.

4.3.2 Supply Transient

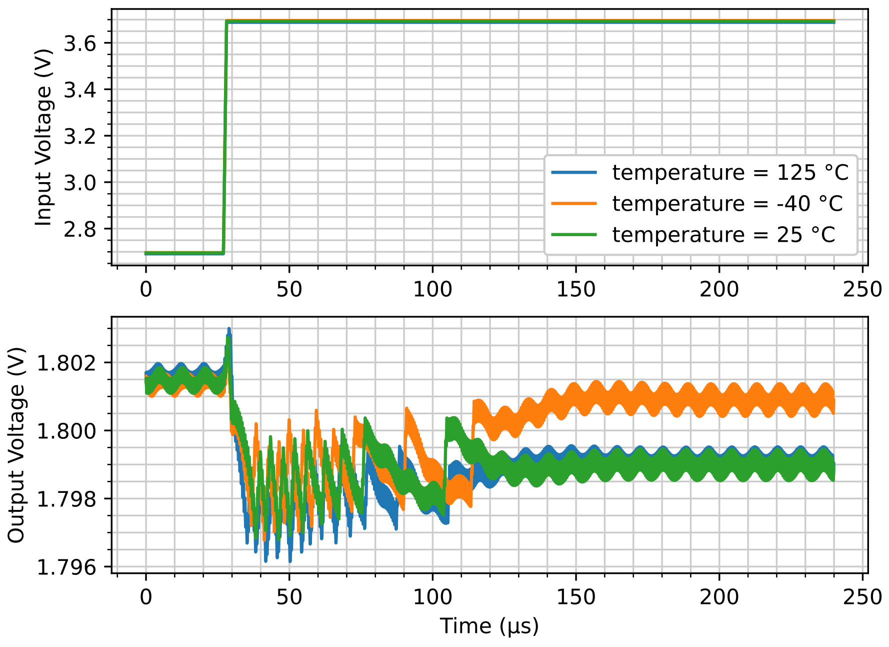
Sensitivity to fast input voltage transients are depicted in Figures 91, 92, 93, and 94.

Figure 91 – Output voltage transient when the input voltage varies from 3.7 V to 5.5 V in 1 µs.



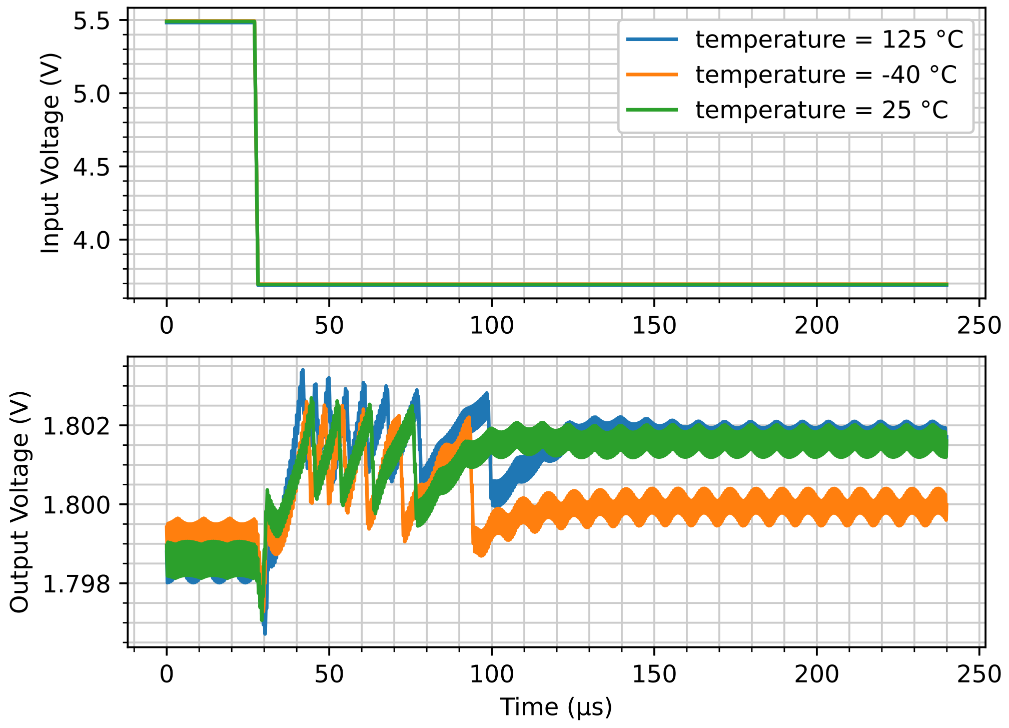
Source: The author.

Figure 92 – Output voltage transient when the input voltage varies from 2.5 V to 3.7 V in 1 µs.



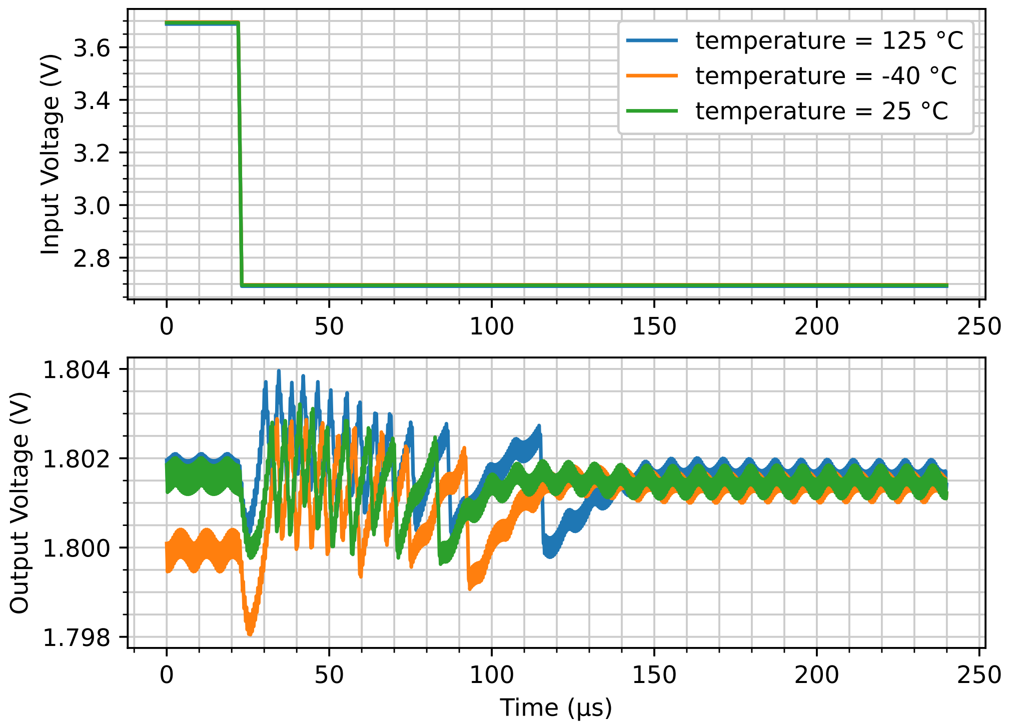
Source: The author.

Figure 93 – Output voltage transient when the input voltage varies from 5.5 V to 3.7 V in 1 µs.



Source: The author.

Figure 94 – Output voltage transient when the input voltage varies from 3.7 V to 2.7 V in 1 µs.

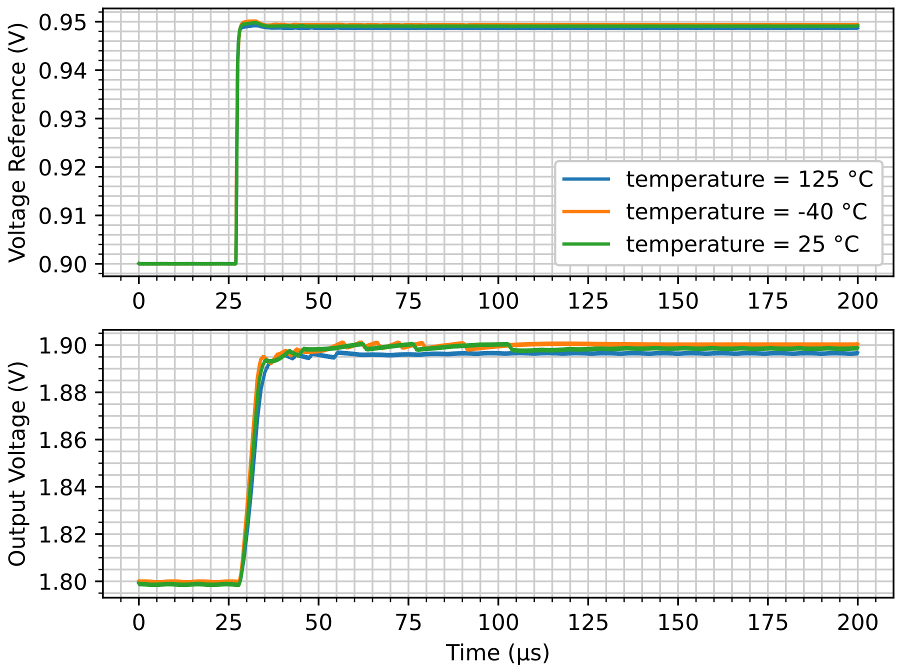


Source: The author.

4.3.3 Step in the voltage reference

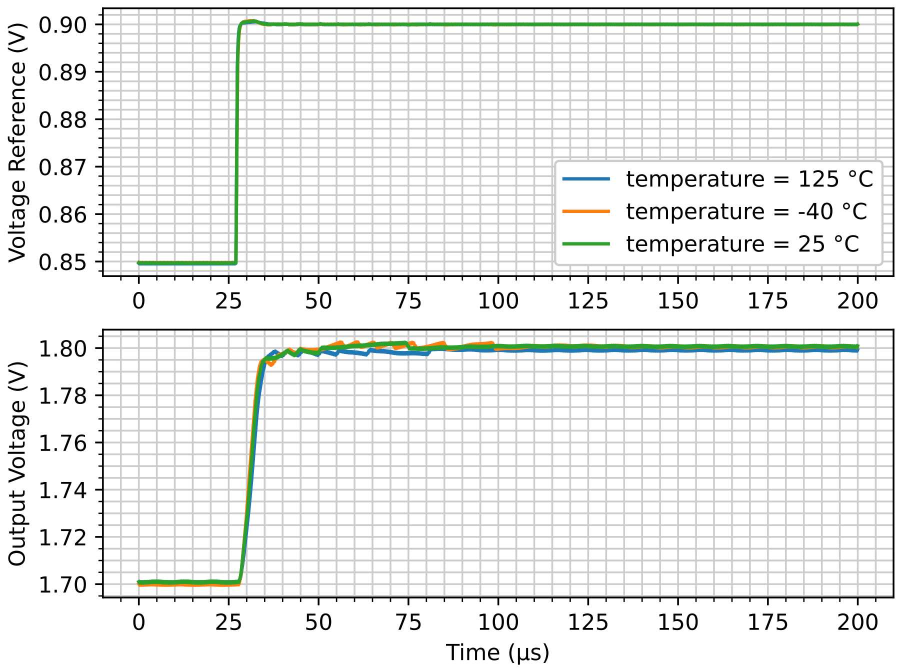
Undershoot and overshoots when changing the configuration of the voltage reference are depicted in Figures 95, 96, 97 and 98.

Figure 95 – Output voltage overshoot when the voltage reference varies from 0.9 V to 0.95 V in 1 µs.



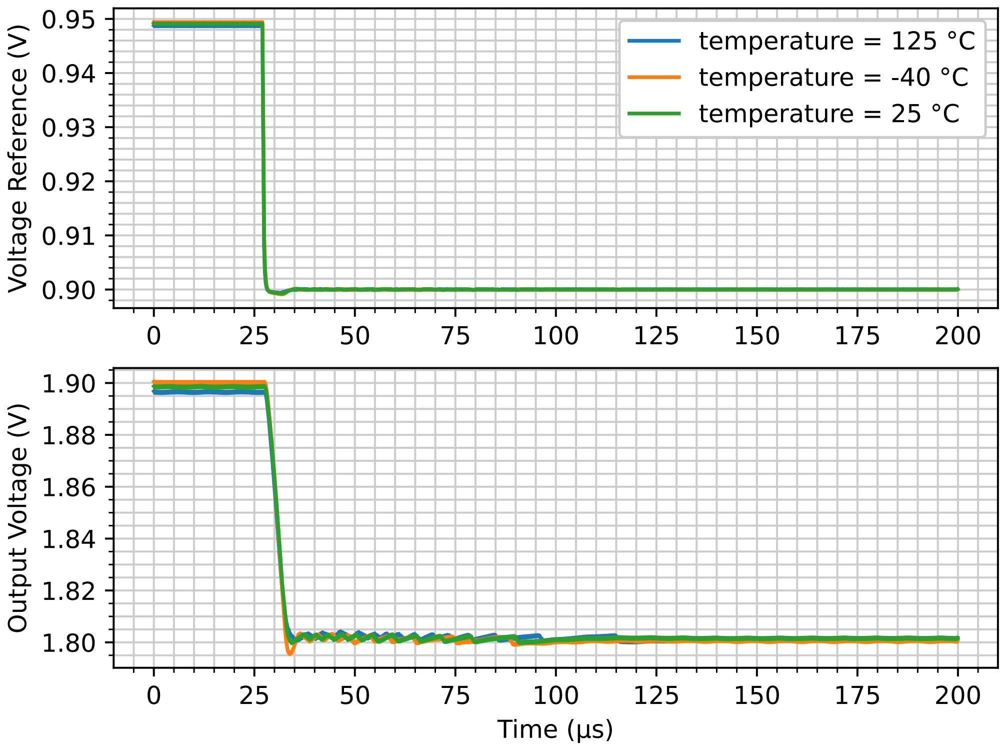
Source: The author.

Figure 96 – Output voltage overshoot when the voltage reference varies from 0.85 V to 0.9 V in 1 µs.



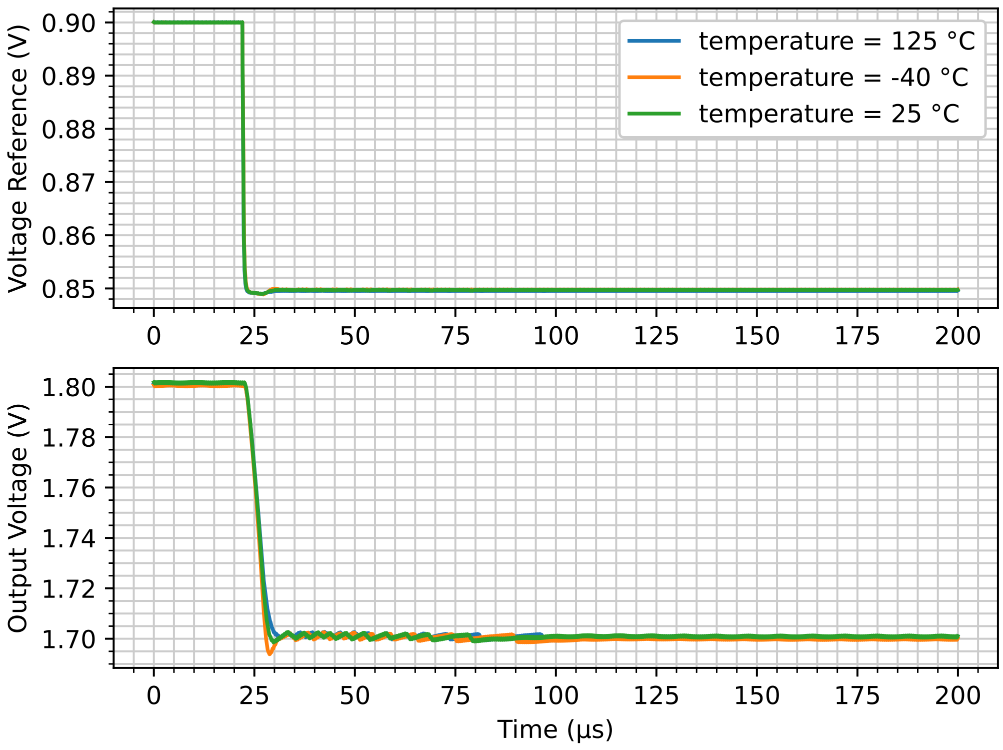
Source: The author.

Figure 97 – Output voltage Undershoot when the voltage reference varies from 0.95 V to 0.9 V in 1 µs.



Source: The author.

Figure 98 – Output voltage Undershoot when the voltage reference varies from 0.9 V to 0.85 V in 1 µs.



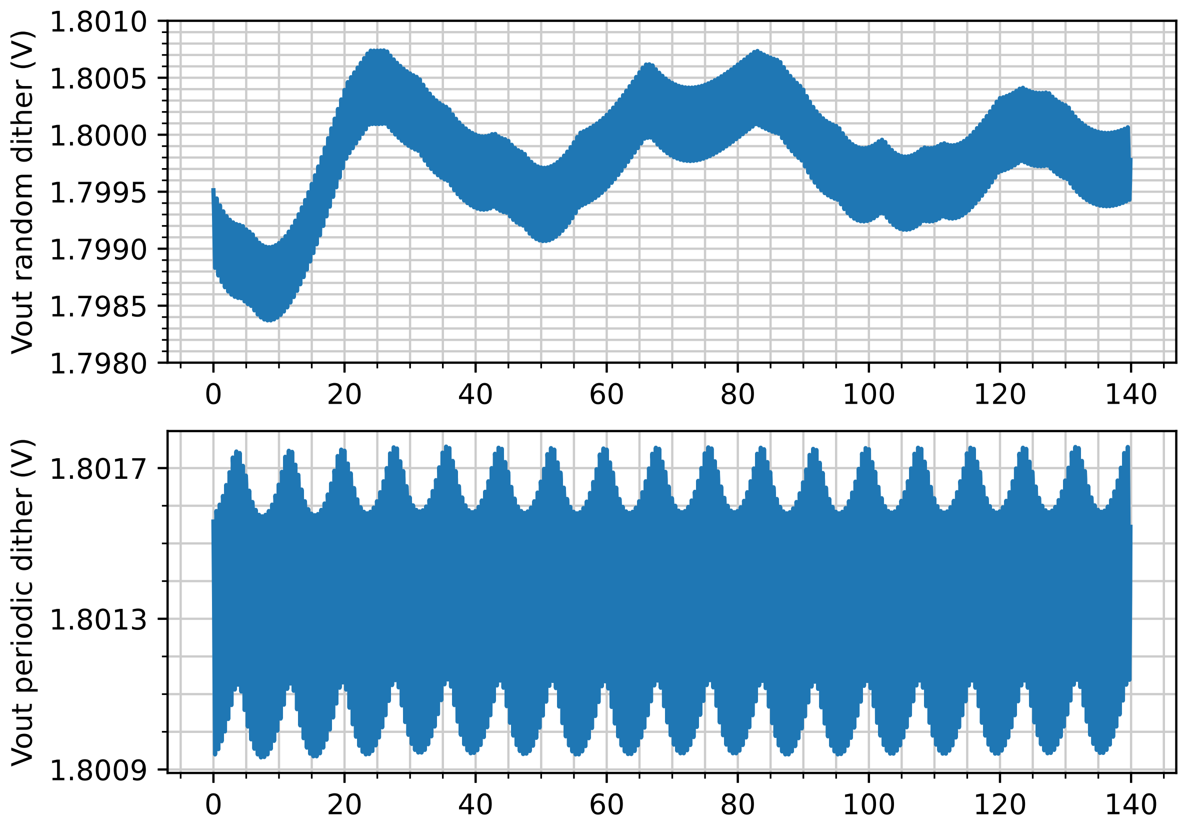
Source: The author.

4.3.4 Dithering

All system-level simulation results presented previously in Chapter 4 were carried out assuming that the dithering occurs periodically, using a 4-bit counter that divides the switching frequency (2 MHz) by 16. This creates a variation in the duty cycle that repeats at a frequency of 125 KHz. The Verilog model of the digitally controlled logic (see Annex E) can also dither the duty cycle using a pseudo-random generator. This can be controlled by defining (or not defining) the macro 'RAND_DITHER'.

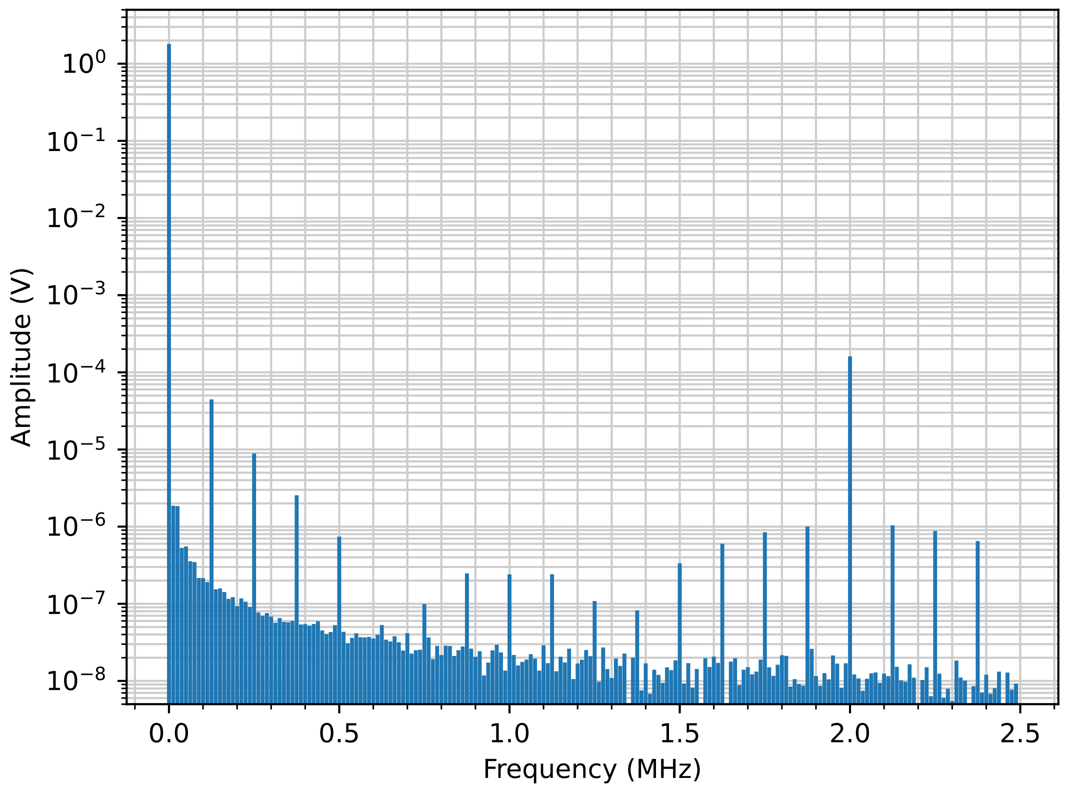
The difference in the output voltage with these two types of dithering is represented in Figure 99. Down-sampling the output voltage from the simulation step size (less than 10 ps) to a sample period of 50 ns and then calculating the Fast Fourier Transform of the resulting samples yields Figures 100 and 101.

Figure 99 – Output voltage ripple for random and periodic dithering.



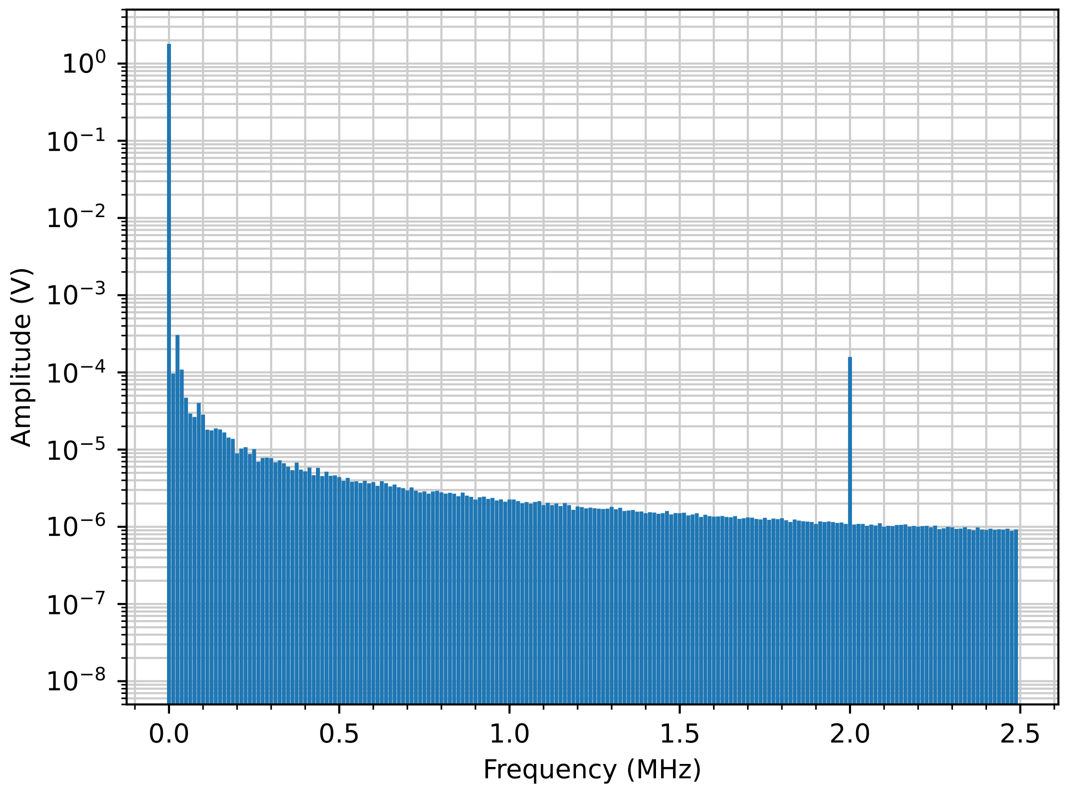
Source: The author.

Figure 100 – FFT of the output voltage for periodic dithering.



Source: The author.

Figure 101 – FFT of the output voltage for random dithering.



Source: The author.

5 Discussions

In this chapter, the results presented in Chapter 4 are discussed and compared with the literature. Refer to Table 33 (an extension of Table 13) for a comparison between the simulated parameters for this work and the reviewed literature. It is first discussed the results obtained for the ADC, followed by the DPWM, and finally for the entire system.

Table 33 – Comparison between the main parameters of this work and the reviewed literature.

Parameter	[3]	[13]	[9]	[10]	[17]	[12]	[14]	this work
Year	2004	2010	2011	2011	2017	2019	2021	2024
Input	2.8 – 5.5 V	2.5 – 5.5 V	2.3 – 4.8 V	2.8 – 4.2 V	12 V	5.0 V	3.3 – 5 V	2.7 - 5.5 V
Output	1.0 – 1.8 V	1.8 V	0.6 – 1.35 V	0.4 – 1.2 V	1.5 V	1.8 V	0.9 – 3.3 V	1.62 - 1.98 V
Technology	0.25 µm	0.18 µm	65 nm	45 nm	0.18 µm	0.18 µm	0.18 µm	0.18 µm
Load	0.1 – 400 mA	0 – 300 mA	0 – 1 A	0.02 – 100 mA	6.6 A	1.5 A	7.5 A	0 - 200 mA
Inductor	10 µH	18.8 µH	470 nH	10 µH	2.2 µH	2.2 µH	800 nH	6.8 µH
Capacitor	47 µF	22 µF	10 µF	2 µF	50 µF	2.2 µF	15 µF	22 µF
ADC topology	Ring-Osc	VCO	SAR	Flash	Delay Line	Delay Line	Time based	Time based
ADC step	16 mV	-	10 mV	50 mV	-	5 mV	-	1.73 - 2.72 mV
ADC area	0.15 mm^2	-	0.024 mm^2	-	0.022 mm^2	-	-	0.0088 mm^2 (estimated)
ADC current	37 µA	220 µA	18 µA	-	-	-	-	11 - 22.2 µA
ADC resolution	-	5-bit (window)	4-bit (window)	4-bit (window)	6-bit (window)	6-bit (window)	6-bit (window)	17 levels (window)
DPWM topology	Ring-MUX	Counter + DLL	Counter	I-C DAC	Counter + Delay line	Counter + DLL	Counter + DLL	R-C DAC
DPWM step	3.125%	0.195%	2.083%	1%	0.0244%	0.39%	0.195%	12.2 - 17.0 mV
DPWM area	-	-	0.0015 mm^2	-	0.03 mm^2	-	-	0.0439 mm^2 (estimated)
DPWM current	-	370 µA	82.5 µA	6 µA	-	-	-	20.7 - 58.5 µA
DPWM resolution	5-bit + Dither	9-bit	48 levels + Dither	5-bit + Dither	12-bit	8-bit	9-bit	192 levels + Dither
Total area (control)	-	-	0.038 mm^2	0.07 mm^2	0.16 mm^2	-	-	-
Line transient	-	-	10 mV (600 mV in 10 µs)	-	-	-	-	< 6 mV (1.8 V in 1 µs)
Load transient	25 mV (50–150 mA)	130 mV (0–300 mA)	20 mV (0–200 mA in 500 ns)	10 mV (10–50 mA in 25 µs)	40 mV (1.5–3.0 A)	600 mV (1–1.5 A)	88.3 mV (500 mA in 600 ns)	31 mV (0 - 200 mA in 100 ns)
Reference clock	0.5 – 1.5 MHz	16 MHz	307.2 MHz	2 MHz	20 MHz	8 MHz	-	50 MHz
Switching frequency	0.5 – 1.5 MHz	500 kHz	6.4 MHz	2 MHz	1.25 MHz	1 MHz	1 MHz	2 MHz

5.1 ADC

The proposed ADC has a typical step size of 2.136 mV, which varies between -8.9% and +9.4% over process corners and 3σ of mismatch effects. Additionally, the step size varies between -11.2% and +17.5% at temperatures of -40°C and 125°C, respectively. Sensitivity to supply variations is negligible. Considering variations due to process corners, temperature, and mismatch, the step size will vary between -19.2% and +27.3% of the typical value. Despite these variations, the ADC appears to be linear within a predefined range. In this design, the DNL and INL did not exceed the granularity of the ADC transfer curve used during simulation. In order to compensate for the comparator mismatch effects discussed in Section 3.1.2 and to save simulation time, the accumulation process occurred for 22 cycles. However, the linearity was measured only for the range in which the ADC output remained high, between 3 and 19 clock cycles. The equivalent digital output was adjusted by subtracting 3 in order to plot the transfer curves shown in Figures 74, 75, 76, 77, and 78. It is highly probable that the DNL and INL will be compromised in the range of 0 to 2 and 19 to 22 cycles. This is acceptable because, as mentioned in Section 3.1.2, the ADC is designed so that only the central range is actually used.

The average current flowing trough VSS varies between 11 µA and 22.2 µA across process corners, mismatch, and temperature with a typical value of 15.5 µA. The differential ADC input voltage in the middle of the range will be within ±7.5 mV.

Considering the reviewed literature, not all authors report the ADC step size, as the focus is often on the overall buck converter performance. Among those who do report it, it is evident that the step size used in this work is one of the smallest (noting that the 2.136 mV step size will effectively be doubled due to the resistor divider). Smaller step sizes generally improve accuracy and enable the control to react more quickly (refer to Section 2.3.4).

INL and DNL for ADCs implemented in buck converters do not seem to be commonly reported in the literature. However, there is no reason to believe that delay line, ring oscillator, or VCO architectures would be more or less linear than the proposed architecture. It is suggested in [3] that the differential window ADC with ring oscillators deviates only ±3 mV, which means that the ±7.5 mV achieved in this work may be in the higher end of the spectrum. Regarding quiescent current, the typical value of 15.5 µA for the proposed architecture appears to be on the lower end of the spectrum, 14% smaller than the value reported by [9] for the SAR ADC. The estimated area of the proposed ADC is 0.0088 mm^2 (assuming 60% area utilization for the components and leaving 40% free for routing), which is 60% smaller than the area reported by [17]. These last two comparisons should be taken cautiously, as the proposed architecture has only been designed up to the schematic phase, while most of the work reported in the literature has been manufactured and measured in the lab.

Despite these considerations, the proposed architecture shows potential to achieve a low step size with low current consumption and a small active area. A disadvantage of the proposed architecture is that, although the design uses a reduced number of transistors, it requires manual layout using the analog flow. In contrast, architectures that utilize delay lines, or ring oscillators can be fully or partially synthesized using the digital design flow.

5.2 DPWM

The proposed DPWM features a variable duty cycle step size, maintaining a constant voltage step size regardless of the buck input voltage. After trimming, the typical voltage step size is 14.0 mV, with variations ranging from -4.3% to +9.3% due to process corners and 3σ of mismatch effects. Temperature variations also affect the step size, ranging from +9.3% at -40°C to -7.9% at 125°C. Sensitivity to supply variations is negligible. Overall, considering process corners, temperature, supply, and mismatch, the step size variation spans from -12.9% to +21.4% of the typical value.

With a typical voltage step size and a a buck input voltage range of 2.7 V to 5.5 V, the duty cycle step size varies between 0.25% and 0.52%, which is neither the smallest nor the largest reported in the reviewed literature. However, the DPWM voltage step size is not lower than the ADC step size, necessitating dithering.

Average current flowing through VSS varies between 20.7 µA and 58.5 µA across process corners, mismatch, and temperature, with a typical value of 35.1 µA, which is within the limits found in the reviewed literature.

The product of the duty cycle and the buck input voltage deviates between -65.1 mV and 91.6 mV from the ideal curve defined as $(DPWM_IN + 12.5) \cdot Q_{DPWM}$. The DNL ranges between -0.018 LSB and 0.148 LSB, while the INL varies from -0.20 LSB to 1.2 LSB, also across process corners, mismatch, and temperature.

There are a few drawbacks to the proposed architecture: It requires trimming to reduce the variation of the voltage step size over process corners, and the estimated area is 46% larger than the largest DPWM reported in the literature, despite the resolution of the proposed architecture not being as high.

Despite these disadvantages, the independence of the voltage step size from the buck input voltage has the potential to improve line regulation, as discussed in Section 5.3.

5.3 System

Both the ADC and DPWM voltage step sizes vary with temperature, process, and mismatch. As discussed in Section 2.5, variations in the step size change the loop gain, thus altering the bandwidth and affecting the transient response. Without considering the changes in the quantization noise when the step size varies, the PID constants can be adjusted to compensate for process and mismatch variations. However, temperature variations cannot be compensated unless the system includes a temperature sensor to dynamically adjust the PID constants. Keeping these limitations in mind, the system was simulated for different temperatures in a transient Verilog-AMS simulation for fixed values of PID constants.

The transient simulation results show an output voltage variation of less than 31 mV for fast full load transients and less than 19 mV for full load transients with rise and fall times of 20 µs. For line transients with input voltage variation of less than 1.8 V in 1 µs, the output voltage variation is less than 6 mV. Summing the output voltage variation for fast load transients (31 mV) with the accuracy of the ADC (7.5 mV) divided by the G(0) (the voltage divider ratio), a maximum output deviation of 46 mV from the voltage reference is obtained. This deviation is approximately 2.6% for an output voltage of 1.8 V. For load transients with rise and fall times of 20 µs, the maximum deviation will be 34 mV or 1.9%. Therefore, the initially specified goal of 2% can only be achieved under certain conditions.

Comparing load transients with the reviewed literature is challenging. As discussed in Section 2.3.4, the load transient depends on the output capacitance and the system's bandwidth, which also depends on the control and switching frequency. Additionally, Figure 55 suggests that the output voltage variation under load transients decreases with smaller inductors. These dependencies can be intuitively explained by noting that, in steady state, the DC inductor current matches the load current. A larger inductor takes more time or requires a higher voltage difference to change its current, which affects the load transients. Conversely, a faster control can adjust the duty cycle more rapidly, forcing the inductor current to change more quickly. Additionally, a larger capacitor can hold more charge, which helps stabilize the output voltage while the inductor current is adjusting.

Taking all these considerations into account, perhaps the best framework for comparison is [3], which utilizes an inductor 47% larger, a capacitor 113% larger, and a switching frequency 25% lower. The work proposed in [10], although it also utilizes a 2 MHz switching frequency, appears to be tested under a 40 mA step with a 25 µs rise/fall time to achieve an output voltage variation that depends solely on the LC filter without control action, as the output variation seems to be smaller than the ADC step size. Given these considerations, the load transient for the proposed architecture seems to be within the range reported in the literature.

From the voltage mode architectures [3, 13, 9, 10, 14], the work developed by [10] digitizes the buck input voltage to implement a DPWM feed-forward scheme in the digital domain. Perhaps that is the reason why the information about line transients was readily available in [10], but it does not seem to be present in the remaining reviewed literature.

In this work, the suggested input voltage feed-forward scheme was implemented directly in the analog domain, embedded in the architecture of the DPWM. The result was an output voltage variation due to line transients that was 40% smaller than reported by [10]. However, it is important to note that the architecture still needs refinement, and the results presented in [10] were measured in the lab, while the results reported in this work correspond to the design at the schematic level for the ADC and DPWM along with the verilog level for the logic. Nevertheless, implementing the feed-forward scheme directly in the DPWM shows potential to achieve good line transient performance with low complexity, leaving the digitization process for the higher complex task of compensating the loop with a digital PID control.

Figure 99 also depict the output voltage ripple for the periodic and random dither implemented in the digital control, along with the corresponding FFT of the output voltage in Figures 100 and 101. It is important to note that these results correspond to the steady state, where the control has already stabilized the output voltage within the middle range of the ADC, and the output of the ADC is no longer varying. However, during transients, the FFT will change due to the quantization noise of the ADC. No study was made to evaluate the effect that the two types of dithering may have in a potential application for the propose digital buck.

Conclusions and Future works

This work focused on developing a digitally driven control system for buck converters in battery power applications. The system operates with an input voltage ranging from 2.7 V to 5.5 V, delivers an output voltage between 1.62 V and 1.98 V, operates at a switching frequency of 2 MHz, and supports load currents up to 200 mA. The target accuracy for the output voltage was set at $\pm 2\%$ of the configured voltage reference.

Simulation results at the schematic level suggest that achieving the $\pm 2\%$ accuracy is feasible only if the rise and fall times of full load transients are greater than 20 µs. To address potential issues, the configured voltage reference can be trimmed during production to compensate for the ± 7.5 mV inaccuracy of the ADC, which translates to 15 mV when divided by the voltage divider gain. This adjustment can enable a $\pm 2\%$ even during fast load transients.

Line transients were found to be less than 6 mV, which is well within the specified limits. In order to achieve the required load and line transient responses, a window ADC with a step size of 2.136 mV and a DPWM incorporating a input voltage feed-forward scheme were proposed. Significant effort was dedicated to characterizing these two sub-blocks across variations in process, mismatch, input voltage, and temperature. Both components were also designed to minimize quiescent current, number of transistors, and active area.

Although the proposed system was validated only at the schematic phase, the results are promising. The ADC can potentially be designed within an active area of 0.0088 mm^2, with a typical quiescent current of 15.5µA. Additionally, the line transient response highlights the potential of the analog input voltage feed-forward implemented in the DPWM, despite its large active area.

5.3.1 Future work

During the development of this work, the Tiny Tapeout [34] initiative started to accept submission of analog designs. This means that the ADC proposed in this work can be manufactured at a total cost lower than \$500 US dollars. Unfortunately, the work was already at a higher stage of development at the time, but in the future, the ADC can be ported to the open-source SkyWater 130 nm process and a test chip can be manufactured. Future improvements in the ADC and DPWM architectures are also possible:

- This work focuses on optimizing the ADC for low area and power consumption, though this optimization resulted in a limited range with high temperature dependency.

Since most of the current consumption and area of a buck converter operating in PWM are concentrated on the power switches and drivers, implementing a Successive Approximation Register (SAR) ADC as proposed by [9] may improve accuracy and range with only a small increase in the overall system's area and power consumption. The 10 mV step size proposed by [9] is too high for achieving the accuracy required in this work and will need improvement. Special attention must also be given to optimizing the quiescent current of the voltage reference buffer. A study can be conducted to assess the impact of the current consumption and area of such ADC on the overall system.

- The area of the proposed DPWM is among the largest found in the reviewed literature. By designing a VCO that adjusts its frequency based on the buck input voltage, a counter-based DPWM (as proposed by [9]) with variable frequency can be implemented. This new approach can reduce the area of the proposed DPWM while maintaining the input voltage feedforward scheme to improve line transients.

Bibliography

[1] KAZIMIERCZUK, M. K. *Pulse-Widh Modulated DC–DC Power Converters*. Ohio, USA: Wiley, 2016.

[2] MAKSIMOVIC, D.; ZANE, R. Small-Signal Discrete-Time Modeling of Digitally Controlled PWM Converters. *IEEE Transactions on Power Electronics*, v. 22, n. 6, p. 2552–2556, 2007.

[3] XIAO, J. et al. A 4-µA Quiescent-Current Dual-Mode Digitally Controlled Buck Converter IC for Cellular Phone Applications. *IEEE Journal of Solid-State Circuits*, v. 39, n. 12, p. 2342–2348, 2004.

[4] CASTRUCCI, P.; SALES, R. M. *Controle Digital*. São Paulo, Brasil: Edgard Blucher Ltda., 1990.

[5] FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. *Digital Control of Dynamic Systems*. California, USA: Ellis-Kagle Press, 1998.

[6] BEHZARD, R. *Principles of Data Conversion System Design*. [S.l.]: University of Michigan, 1995.

[7] DEVICES, A. *INL/DNL Measurements for High-Speed Analog-to-Digital Converters (ADCs)*. 2024. `https://www.analog.com/en/resources/technical-articles/inldnl-measurements-for-types-of-highspeed-adcs.html`. [Online; accessed 08-Mar-2024].

[8] ZHANG, X. et al. A 0.6 V Input CCM/DCM Operating Digital Buck Converter in 40 nm CMOS. *IEEE Journal of Solid-State Circuits*, v. 49, n. 11, p. 2377–2386, 2014.

[9] CLIQUENNOIS, S. et al. A 65-nm, 1-A Buck Converter With Multi-Function SAR-ADC-Based CCM/PSK Digital Control Loop. In: *2011 Proceedings of the ESSCIRC (ESSCIRC)*. [S.l.: s.n.], 2011. p. 427–430.

[10] BANDYOPADHYAY, S. et al. 20 µA to 100 mA DC–DC Converter With 2.8-4.2 V Battery Supply for Portable Applications in 45 nm CMOS. *IEEE Journal of Solid-State Circuits*, v. 46, n. 12, p. 2807–2820, 2011.

[11] TSAI, T.-H. et al. A 180 nA Quiescent Current Digital Control Dual-Mode Buck Converter With a Pulse-Skipping Load Detector for Long-Range Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, v. 70, n. 7, p. 3040–3048, 2023.

[12] CHEN, N. et al. A Digital Controller IC for High-Frequency DC-DC Switching Converters. In: *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. [S.l.: s.n.], 2019. p. 1645–1648.

[13] AHMAD, H. H.; BAKKALOGLU, B. A 300mA 14mV-ripple digitally controlled buck converter using frequency domain $\Delta\Sigma$ ADC and hybrid PWM generator. In: *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*. [S.l.: s.n.], 2010. p. 202–203.

[14] BEOHAR, N. et al. A Digitally Controlled DC-DC Buck Converter with Automatic Digital PFM to PWM Transition Scheme. In: *2021 IEEE Applied Power Electronics Conference and Exposition (APEC)*. [S.l.: s.n.], 2021. p. 517–522.

[15] YAO, G.-S. et al. All-Digital Current-Sensorless Multi-Mode DC-DC Converter for Battery Powered Applications. In: *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. [S.l.: s.n.], 2019. p. 1144–1145.

[16] BOTTAMEDI, D. et al. Digital Low Power Mode Control Technique for High-Frequency Synchronous DC-DC Buck Converters. In: *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. [S.l.: s.n.], 2019. p. 326–329.

[17] VEKSLENDER, T. et al. Fully-integrated digital average current-mode control 12V-to-1.xV voltage regulator module IC. In: *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*. [S.l.: s.n.], 2017. p. 2043–2050.

[18] KHAN, Q. A.; KIM, S.-J.; HANUMOLU, P. K. Time-Based PWM Controller for Fully Integrated High Speed Switching DC-DC Converters — An Alternative to Conventional Analog and Digital Controllers. In: *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*. [S.l.: s.n.], 2018. p. 226–231.

[19] LI, P. et al. A 90–240MHz hysteretic controlled DC-DC buck converter with digital PLL frequency locking. In: *2008 IEEE Custom Integrated Circuits Conference*. [S.l.: s.n.], 2008. p. 21–24.

[20] PETERCHEV, A.; SANDERS, S. Quantization resolution and limit cycling in digitally controlled PWM converters. *IEEE Transactions on Power Electronics*, v. 18, n. 1, p. 301–308, 2003.

[21] Texas Instrument. Aplication report: Selecting Inductors for Buck Converters. *Texas Instrument*, 2013.

[22] HAGEN, M.; YOUSEFZADEH, V. Applying Digital Technology to PWM Control-Loop Designs. *Texas Instrument*.

[23] SHEEHAN, R.; DIANA, L. Switch-mode power converter compensation made easy. *Texas Instrument*, 2016.

[24] BAKER, B. How delta-sigma ADCs work, part 1. *Texas Instrument*, 2016.

[25] GRAY, P. R. et al. *ANALYSIS AND DESIGN OF ANALOG INTEGRATED CIRCUITS*. USA: John Wiley & Sons, 2009.

[26] Newcastle University. *Workcraft*. 2024. `https://workcraft.org/`. [Online; accessed 12-April-2024].

[27] Newcastle University. *Analog-to-asynchronous elements*. 2024. `https://workcraft.org/a2a/start`. [Online; accessed 12-April-2024].

[28] MENDES, R. P. *STG to verilogA converter*. 2023. `https://github.com/rpm2003rpm/stg2veriloga/`. [Online; accessed 19-July-2023].

[29] LI, S. et al. Design of Fast Transient Response Voltage-Mode Buck Converter With Hybrid Feedforward and Feedback Technique. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, v. 9, n. 1, p. 780–790, 2021.

[30] FATH, P. et al. An open-sourced 1.44-Ms/s 703-µW 12-bit non-binary SAR-ADC using 448-aF capacitors in 130-nm CMOS. *e+i Elektrotechnik und Informationstechnik*, 2024.

[31] UNIVERSE, V. *Thermometer Code*. 2019. `https://vlsiuniverse.blogspot.com/2016/06/binary-to-thermometer-encoder.html`. [Online; accessed 24-Nov-2023].

[32] MENDES, R. P. *VerilogA generator*. 2023. `https://github.com/rpm2003rpm/vagen/`. [Online; accessed 19-July-2023].

[33] ZHANG, K. et al. A programmable CMOS voltage reference based on a proportional summing circuit. In: *2007 7th International Conference on ASIC*. [S.l.: s.n.], 2007. p. 534–537.

[34] Tiny Tapeout. *Tiny Tapeout*. 2024. `https://tinytapeout.com/`. [Online; accessed 05-Jun-2024].

ANNEX A – Power switch model

```
/*****************************************************************
 * Module: DIG_BUCK_PSW_TOP                                      *
 * Power module top level for the digital buck                   *
 * Date: 2023-08-09                                              *
 *****************************************************************/
`include "constants.vams"
`include "disciplines.vams"


/*****************************************************************
 *                         Module declaration                    *
 *****************************************************************/
module DIG_BUCK_PSW_TOP(VSS,
                        VDD_5V0,
                        VDD_1V8,
                        SW_5V0,
                        PSW_EN_1V8,
                        PWM_1V8);

/*****************************************************************
 *                                 Ports                         *
 *****************************************************************/
inout  VSS;
inout  VDD_5V0;
inout  VDD_1V8;
inout  SW_5V0;
input  PSW_EN_1V8;
input  PWM_1V8;


/*****************************************************************
 *                                  Disciplines                  *
 *****************************************************************/
electrical VSS;
electrical VDD_5V0;
electrical VDD_1V8;
electrical SW_5V0;
electrical PSW_EN_1V8;
electrical PWM_1V8;


/****************************************************************
```

```
 *                                Parameters                          *
 *******************************************************************/
parameter NMOS_RES_PAR         = 300m;
parameter PMOS_RES_PAR         = 300m;
parameter NMOS_DLY_PAR         = 0n;
parameter PMOS_DLY_PAR         = 0n;
parameter NMOS_RISE_FALL_PAR   = 1n;
parameter PMOS_RISE_FALL_PAR   = 1n;

parameter OCP_LIM_PAR          = 300m;

parameter PMOS_ACK_DLY_PAR       = 1n;
parameter PMOS_ACK_RISE_FALL_PAR = 100p;

parameter NMOS_ACK_DLY_PAR       = 1n;
parameter NMOS_ACK_RISE_FALL_PAR = 100p;

parameter PAR_CAP_PAR          = 5p;

/*******************************************************************
 *                               Variables                            *
 *******************************************************************/
integer PSW_EN_1V8_VAR;
integer PMOS_REQ_VAR;
integer NMOS_REQ_VAR;
integer NMOS_ACK_VAR;
integer PMOS_ACK_VAR;
integer PWM_VAR;
integer ZCR_VAR;
integer OCP_VAR;
integer state;

real    PWRUP_TRAN;
real    PWRUP_SMOOTH;

real    PMOS_TRAN;
real    PMOS_SMOOTH;

real    NMOS_TRAN;
real    NMOS_SMOOTH;


/*******************************************************************
 *                               Analog Block                         *
 *******************************************************************/
analog begin
```

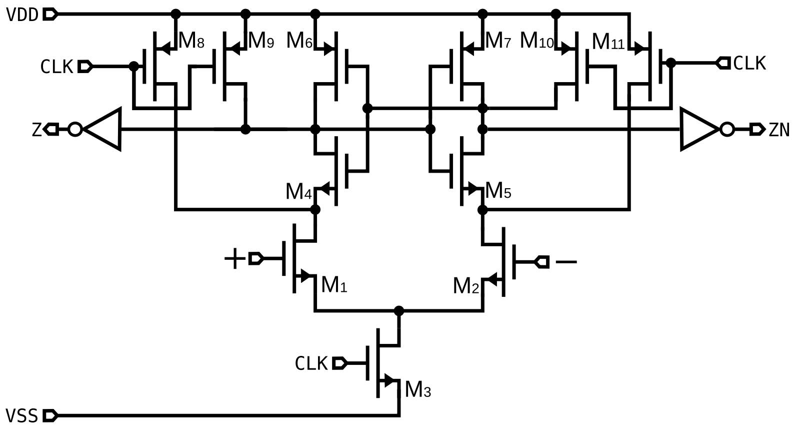
```

    @(cross(V(PWM_1V8, VSS)-(V(VDD_1V8, VSS)/2+0.05),-1,100p,10m))
        $discontinuity(0);
    @(cross(V(PSW_EN_1V8, VSS)-(V(VDD_1V8, VSS)/2+0.05),-1,100p,10m))
        $discontinuity(0);
    @( cross(PMOS_SMOOTH - 0.5, 0, 100p, 10m))
        $discontinuity(0);
    @( cross(NMOS_SMOOTH - 0.5, 0, 100p, 10m))
        $discontinuity(0);
    @( cross(OCP_LIM_PAR + I(SW_5V0, VDD_5V0), 0, 100p, 1p))
        $discontinuity(0);
    @( cross(I(VSS, SW_5V0) - 100u,   0, 100p, 1p))
        $discontinuity(0);

    PWM_VAR = (V(PWM_1V8, VSS) > (V(VDD_1V8, VSS)/2 + 0.05));
    PSW_EN_1V8_VAR = (V(PSW_EN_1V8, VSS) > (V(VDD_1V8, VSS)/2 + 0.05));
    OCP_VAR = OCP_LIM_PAR + I(SW_5V0, VDD_5V0) < 0;
    ZCR_VAR = I(VSS, SW_5V0) - 100u < 0;
    NMOS_ACK_VAR = NMOS_SMOOTH - 0.5 > 0;
    PMOS_ACK_VAR = PMOS_SMOOTH - 0.5 > 0;

    case(state)

        0: begin
            if ( PWM_VAR && PSW_EN_1V8_VAR) begin
               state = 1;
            end
        end

        1: begin
            if ( PWM_VAR ) begin
               NMOS_REQ_VAR = 0;
               state = 2;
            end
        end

        2: begin
            if ( !NMOS_ACK_VAR ) begin
               PMOS_REQ_VAR = 1;
               state = 3;
            end
        end

        3: begin
            if ( PMOS_ACK_VAR ) begin
                state = 4;
            end
        end

        4: begin
            if ( !PWM_VAR ) begin
                PMOS_REQ_VAR = 0;
                state = 5;
            end else if ( OCP_VAR ) begin
                PMOS_REQ_VAR = 0;
                state = 7;
            end
        end

        7: begin
            if ( !PMOS_ACK_VAR ) begin
               NMOS_REQ_VAR = 1;
               state = 8;
            end else if ( !PWM_VAR ) begin
                state = 5;
            end
        end

        8: begin
            if ( !PSW_EN_1V8_VAR && NMOS_ACK_VAR) begin
                state = 9;
            end else if ( !PWM_VAR ) begin
                state = 6;
            end
        end

        5: begin
            if ( !PMOS_ACK_VAR ) begin
                NMOS_REQ_VAR = 1;
                state = 6;
            end
        end

        6: begin
            if ( !PSW_EN_1V8_VAR && NMOS_ACK_VAR) begin
                state = 9;
            end else if ( NMOS_ACK_VAR ) begin
                state = 1;
            end
        end

        9: begin
```

```
                if ( ZCR_VAR ) begin
                    NMOS_REQ_VAR = 0;
                    state = 0;
                end
            end


        endcase

        // Switches
        PMOS_TRAN = transition(PMOS_REQ_VAR,
                               PMOS_DLY_PAR,
                               PMOS_RISE_FALL_PAR*9.1024,
                               PMOS_RISE_FALL_PAR*9.1024);
        PMOS_SMOOTH = tanh(20*PMOS_TRAN - 10)/2 + 0.5;

        NMOS_TRAN = transition(NMOS_REQ_VAR,
                               NMOS_DLY_PAR,
                               NMOS_RISE_FALL_PAR*9.1024,
                               NMOS_RISE_FALL_PAR*9.1024);
        NMOS_SMOOTH = tanh(20*NMOS_TRAN - 10)/2 + 0.5;

        I(SW_5V0, VDD_5V0) <+ V(SW_5V0, VDD_5V0)/PMOS_RES_PAR*PMOS_SMOOTH;
        I(VSS, SW_5V0)     <+ V(VSS, SW_5V0)/NMOS_RES_PAR*NMOS_SMOOTH;

        // Bulk diodes
        I(SW_5V0, VDD_5V0) <+ 7e-9*limexp(V(SW_5V0, VDD_5V0)/$vt);
        I(VSS, SW_5V0)     <+ 7e-9*limexp(V(VSS, SW_5V0)/$vt);

        // Parasitic capacitance
        I(SW_5V0, VDD_5V0) <+ PAR_CAP_PAR*ddt(V(SW_5V0, VDD_5V0));
        I(VSS, SW_5V0)     <+ PAR_CAP_PAR*ddt(V(VSS, SW_5V0));

    end


endmodule


```

ANNEX B – ADC comparator and control

B.1 Clocked comparator

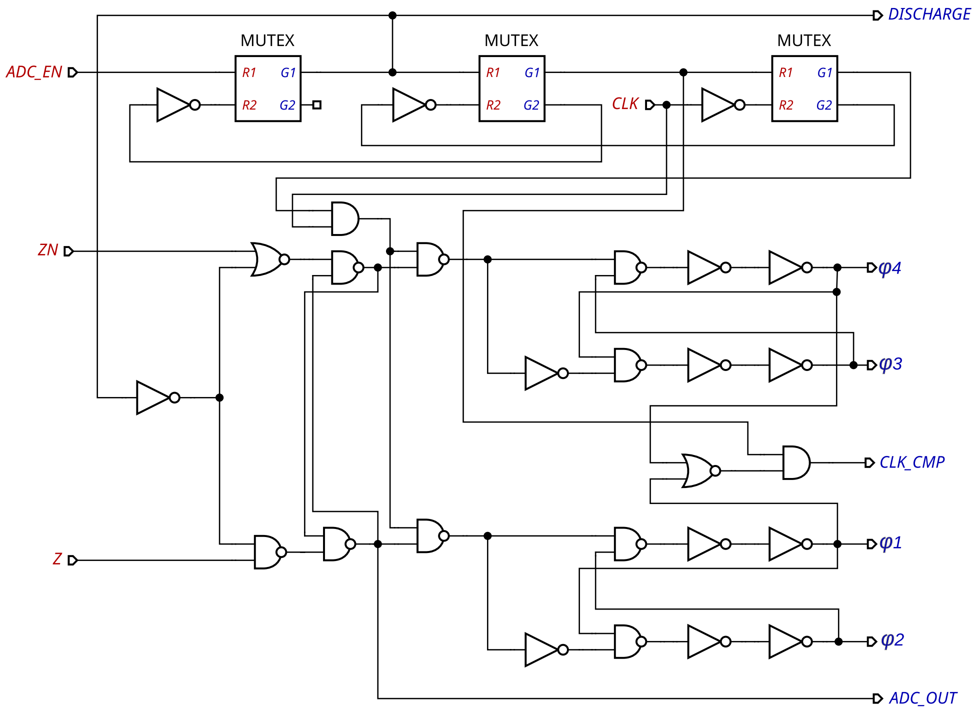
Figure 102 – Proposed clocked comparator for the ADC



Source: The author.

B.2 ADC control

Figure 103 – Gate level implementation of the ADC control

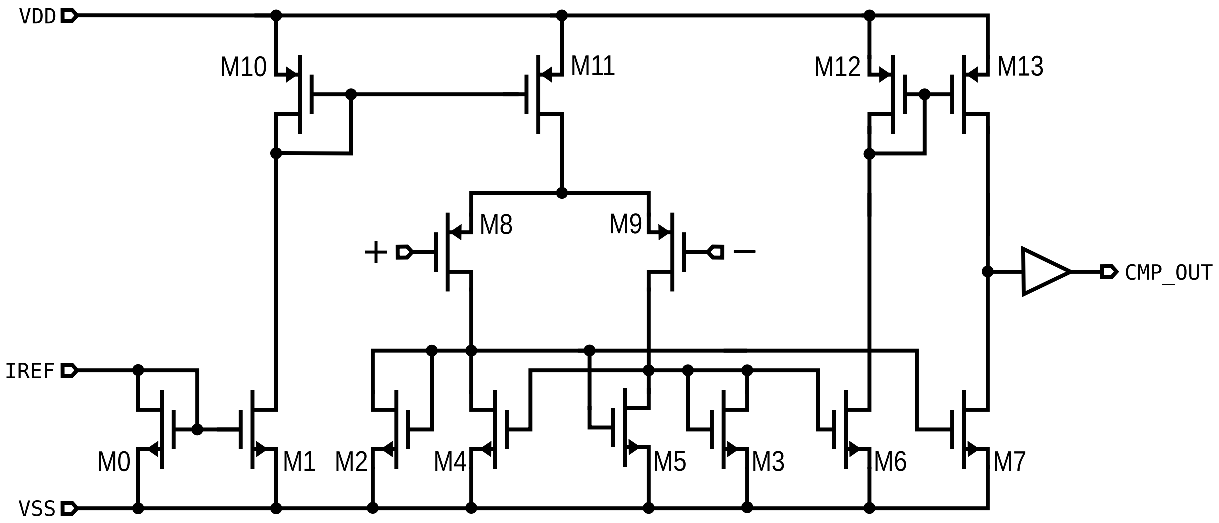


Source: The author.

ANNEX C – Comparator utilized in the DPWM

The architecture is the classic textbook analog comparator shown on Figure 104, which has been chosen due to its design simplicity.

Figure 104 – Proposed comparator for the DPWM



Source: The author.

ANNEX D – Matlab model for analysis of the ADC

```matlab
ans = [];
err = linspace(-0.03,0.03,100000);
k = 1.5;
alfa = 0.5; %Ratio between the mirror (k+1) and the mirror connected to the pair
offset = 0;
C = 143.3725e-15*16;
T = 20e-9;
VDD = 1.62;
gm_id = 20.5;
IBIAS = 5e-6;
IS = IBIAS/2;
m = 22;
for vd = err
    dout = 0;
    V1 = VDD -IS*(1 - gm_id*vd/2)*(k+1)*0.5*alfa*T/C;
    V2 = VDD -IS*(1 + gm_id*vd/2)*(k+1)*0.5*alfa*T/C;
    for i = 0:(m-1)
        if V1 - V2 > offset
            V1 = V1 -IS*(1 - gm_id*vd/2)*(k+0.5)*alfa*T/C -IS*alfa*T/C;
            V2 = V2 -IS*(1 + gm_id*vd/2)*(k+0.5)*alfa*T/C;
            dout = dout + 1;
        else
            V1 = V1 -IS*(1 - gm_id*vd/2)*(k+0.5)*alfa*T/C;
            V2 = V2 -IS*(1 + gm_id*vd/2)*(k+0.5)*alfa*T/C -IS*alfa*T/C;
        end
    end
    %plot(ans2)
    %waitforbuttonpress;
    ans = [ans dout];
end
figure(1);
plot(err, ans);
ylabel('Digital output after decimation')
xlabel('Diferential input')
disp(["Step = " string(4/gm_id*1/(2*m*k + m - k))])
grid on
grid minor
```

ANNEX E – Synthesizable verilog model of the digitally controlled logic

```
/***************************************************************************
 *                              Global definitions                         *
 ***************************************************************************/
`define K_WIDTH        14
`define DPWM_COL_WIDTH 16
`define DPWM_ROW_WIDTH 12
//`define RAND_DITHER


/***************************************************************************
 *                              Module declaration                         *
 ***************************************************************************/
module DIG_BUCK_SYNC_CTRL ( VDD_1V8,
                            VSS,
                            K2_1V8,
                            K1_1V8,
                            K0_1V8,
                            ADC_DOUT_1V8,
                            RST_N_1V8,
                            PSW_EN_1V8,
                            DPWM_STOP_1V8,
                            DPWM_START_1V8,
                            DPWM_ROW_EN_N_1V8,
                            DPWM_ROW_1V8,
                            DPWM_COL_1V8,
                            ADC_EN_1V8,
                            CLK_50M_1V8 );

/***************************************************************************
 *                                    Ports                                *
 ***************************************************************************/
inout VDD_1V8;
inout VSS;
input   [(`K_WIDTH-1):0] K2_1V8;
input   [(`K_WIDTH-1):0] K0_1V8;
input   [(`K_WIDTH-1):0] K1_1V8;
input   ADC_DOUT_1V8;
input   RST_N_1V8;
output  PSW_EN_1V8;
```

```verilog
output  DPWM_STOP_1V8;
output  DPWM_START_1V8;
output  [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_EN_N_1V8;
output  [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_1V8;
output  [(`DPWM_COL_WIDTH-1):0] DPWM_COL_1V8;
output  ADC_EN_1V8;
input   CLK_50M_1V8;


/*****************************************************************************
 *                              Local parameters                             *
 *****************************************************************************/
localparam S0  = 5'b00000;
localparam S1  = 5'b00001;
localparam S2  = 5'b00010;
localparam S3  = 5'b00011;
localparam S4  = 5'b00100;
localparam S5  = 5'b00101;
localparam S6  = 5'b00110;
localparam S7  = 5'b00111;
localparam S8  = 5'b01000;
localparam S9  = 5'b01001;
localparam S10 = 5'b01010;
localparam S11 = 5'b01011;
localparam S12 = 5'b01100;
localparam S13 = 5'b01101;
localparam S14 = 5'b01110;
localparam S15 = 5'b01111;
localparam S16 = 5'b10000;
localparam S17 = 5'b10001;
localparam S18 = 5'b10010;
localparam S19 = 5'b10011;
localparam S20 = 5'b10100;
localparam S21 = 5'b10101;
localparam S22 = 5'b10110;
localparam S23 = 5'b10111;
localparam S24 = 5'b11000;
localparam STATE_WIDTH = 5;


localparam U_WIDTH = 5;
localparam U0 = 2'b00;
localparam U1 = 2'b01;
localparam U2 = 2'b10;
localparam UX = 2'bxx;
localparam ACCUMULATION_INITIAL_VALUE = 5'h15;
```

```verilog

`ifdef RAND_DITHER
localparam DTH_SEED = 16'h2456;
localparam DTH_WIDTH = 16;
localparam DTH_OUT_WIDTH = 4;
localparam DTH_BIT0 = 6;
localparam DTH_BIT1 = 2;
localparam DTH_BIT2 = 15;
`else
localparam DTH_WIDTH = 4;
`endif

localparam MATH_WIDTH = 22;
localparam MATH_DECIMAL_WIDTH = 8;
localparam SUM = 1'b0;
localparam SUB = 1'b1;

localparam K0 = 2'b00;
localparam K1 = 2'b01;
localparam K2 = 2'b10;
localparam KX = 2'bxx;
localparam RAND = 3'b000;
localparam SHIFT0 = 3'b001;
localparam SHIFT1 = 3'b010;
localparam SHIFT2 = 3'b011;
localparam SHIFT3 = 3'b100;
localparam SHIFT4 = 3'b101;
localparam OPX = 3'bxxx;

localparam DPWM_COL_BIT_WIDTH = 4;
localparam DPWM_ROW_BIT_WIDTH = 4;


localparam TRUE = 1'b1;
localparam FALSE = 1'b0;

localparam DONTCARE = 1'bx;


/*****************************************************************************
 *                        Registers  and wires                               *
 *****************************************************************************/
//STATE
reg  [(STATE_WIDTH-1):0] STATE_Q;
reg  [(STATE_WIDTH-1):0] STATE_D;

```

```verilog
//ADC SYNCHRONIZATION AND ENABLE
reg  ADC_EN_D;
reg  ADC_EN_Q;
wire ADC_DOUT;
wire ADC_OUT_SYNC1_D;
wire ADC_OUT_SYNC2_D;
reg  ADC_OUT_SYNC1_Q;
reg  ADC_OUT_SYNC2_Q;

//DECIMATION AND INPUTS
reg  [(U_WIDTH-1):0] U0_Q;
reg  [(U_WIDTH-1):0] U1_Q;
reg  [(U_WIDTH-1):0] U2_Q;
wire [(U_WIDTH-1):0] U0_D;
wire [(U_WIDTH-1):0] U1_D;
wire [(U_WIDTH-1):0] U2_D;
wire [(U_WIDTH-1):0] U;
reg  [(U_WIDTH-1):0] DEC_Q;
wire [(U_WIDTH-1):0] DEC_D;
wire [(U_WIDTH-1):0] DEC_N;
reg  [1:0] U_SEL;
reg  U0_EN;
reg  U1_EN;
reg  U2_EN;
reg  ADC_ACC_SEL;
reg  ADC_ACC_EN;

//DITHERING
reg  [(DTH_WIDTH-1):0] DTH_Q;
wire [(`K_WIDTH-1):0] DTH;

//OPERAND SELECTION
wire [(`K_WIDTH+U_WIDTH-2):0] OP_D;
reg  [(`K_WIDTH+U_WIDTH-2):0] OP_Q;
reg  [(`K_WIDTH-1):0] K0_Q;
reg  [(`K_WIDTH-1):0] K1_Q;
reg  [(`K_WIDTH-1):0] K2_Q;
wire [(`K_WIDTH-1):0] K0_D;
wire [(`K_WIDTH-1):0] K1_D;
wire [(`K_WIDTH-1):0] K2_D;
wire [(`K_WIDTH-1):0] K;
reg  [2:0] K_SEL;
reg  [3:0] OP_SEL;

//ADDER
reg  SUM_SUB_SEL;
```

```verilog
wire [(MATH_WIDTH-1):0] ADDER_IN2_POS_NEG;
wire [(MATH_WIDTH-1):0] ADDER_IN2;
wire [(MATH_WIDTH-1):0] ADDER_IN1;
wire [(MATH_WIDTH-1):0] ADDER_OUT;
wire [MATH_WIDTH:0] ADDER_OUT_W_CARRY;

//ACCUMULATOR
reg  [(MATH_WIDTH-1):0] ACCUMULATOR_Q;
wire [(MATH_WIDTH-1):0] ACCUMULATOR_D;
reg  ACCUMULATOR_EN;

//DPWM
reg  [(MATH_WIDTH-MATH_DECIMAL_WIDTH-1):0] DPWM_Q;
wire [(MATH_WIDTH-MATH_DECIMAL_WIDTH-1):0] DPWM_D;
wire [(DPWM_COL_BIT_WIDTH+DPWM_ROW_BIT_WIDTH-1):0] DPWM;
reg  [(`DPWM_COL_WIDTH-1):0] DPWM_COL_Q;
wire [(`DPWM_COL_WIDTH-1):0] DPWM_COL_D;
reg  [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_EN_Q;
wire [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_EN_D;
reg  [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_Q;
wire [(`DPWM_ROW_WIDTH-1):0] DPWM_ROW_D;
reg  DPWM_EN;

//CYCLE CONTROL
reg  PSW_EN_Q;
reg  DPWM_START_Q;
reg  DPWM_STOP_Q;
wire PSW_EN_D;
wire DPWM_START_D;
wire DPWM_STOP_D;
reg  DPWM_START;
reg  DPWM_STOP;
reg  PSW_EN;


/*****************************************************************************
 *                              State machine                                *
 *****************************************************************************/

always @* begin
    case (STATE_Q)
        S0: begin
            STATE_D = S1;
            ADC_EN_D = TRUE;
            U_SEL = UX;
            U0_EN = FALSE;
```

```
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = TRUE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S1: begin
                STATE_D = S2;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = TRUE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S2: begin
                STATE_D = S3;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S3: begin
                STATE_D = S4;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S4: begin
                STATE_D = S5;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S5: begin
                STATE_D = S6;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S6: begin
                STATE_D = S7;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            S7: begin
                STATE_D = S8;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = TRUE;
                U2_EN = TRUE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = DONTCARE;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
```

```
        end
        S8: begin
            STATE_D = S9;
            ADC_EN_D = TRUE;
            U_SEL = U2;
            U0_EN = FALSE;
            U1_EN = FALSE;
            U2_EN = FALSE;
            ADC_ACC_SEL = FALSE;
            ADC_ACC_EN = TRUE;
            SUM_SUB_SEL = DONTCARE;
            ACCUMULATOR_EN = FALSE;
            DPWM_EN = FALSE;
            PSW_EN = FALSE;
            DPWM_START = FALSE;
            DPWM_STOP = FALSE;
            K_SEL = K2;
            OP_SEL = SHIFT0;
        end
        S9: begin
            STATE_D = S10;
            ADC_EN_D = TRUE;
            U_SEL = U2;
            U0_EN = FALSE;
            U1_EN = FALSE;
            U2_EN = FALSE;
            ADC_ACC_SEL = FALSE;
            ADC_ACC_EN = TRUE;
            SUM_SUB_SEL = SUM;
            ACCUMULATOR_EN = TRUE;
            DPWM_EN = FALSE;
            PSW_EN = FALSE;
            DPWM_START = FALSE;
            DPWM_STOP = FALSE;
            K_SEL = K2;
            OP_SEL = SHIFT1;
        end
        S10: begin
            STATE_D = S11;
            ADC_EN_D = TRUE;
            U_SEL = U2;
            U0_EN = FALSE;
            U1_EN = FALSE;
            U2_EN = FALSE;
            ADC_ACC_SEL = FALSE;
            ADC_ACC_EN = TRUE;
```

```
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K2;
                OP_SEL = SHIFT2;
            end
            S11: begin
                STATE_D = S12;
                ADC_EN_D = TRUE;
                U_SEL = U2;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K2;
                OP_SEL = SHIFT3;
            end
            S12: begin
                STATE_D = S13;
                ADC_EN_D = TRUE;
                U_SEL = U2;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K2;
                OP_SEL = SHIFT4;
            end
            S13: begin
                STATE_D = S14;
                ADC_EN_D = TRUE;
                U_SEL = U1;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT0;
            end
            S14: begin
                STATE_D = S15;
                ADC_EN_D = TRUE;
                U_SEL = U1;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT1;
            end
            S15: begin
                STATE_D = S16;
                ADC_EN_D = FALSE;
                U_SEL = U1;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
```

```verilog
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT2;
            end
            S16: begin
                STATE_D = S17;
                ADC_EN_D = FALSE;
                U_SEL = U1;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = TRUE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT3;
            end
            S17: begin
                STATE_D = S18;
                ADC_EN_D = TRUE;
                U_SEL = U1;
                U0_EN = TRUE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = TRUE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT4;
            end
            S18: begin
                STATE_D = S19;
                ADC_EN_D = TRUE;
                U_SEL = U0;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = FALSE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K0;
                OP_SEL = SHIFT0;
            end
            S19: begin
                STATE_D = S20;
                ADC_EN_D = TRUE;
                U_SEL = U0;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = FALSE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K0;
                OP_SEL = SHIFT1;
            end
            S20: begin
                STATE_D = S21;
                ADC_EN_D = TRUE;
                U_SEL = U0;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K0;
                OP_SEL = SHIFT2;
            end
            S21: begin
                STATE_D = S22;
                ADC_EN_D = TRUE;
                U_SEL = U0;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K0;
                OP_SEL = SHIFT3;
            end
            S22: begin
                STATE_D = S23;
                ADC_EN_D = TRUE;
                U_SEL = U0;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K0;
                OP_SEL = SHIFT4;
            end
            S23: begin
                STATE_D = S24;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = RAND;
            end
            S24: begin
                STATE_D = S0;
                ADC_EN_D = TRUE;
                U_SEL = UX;
                U0_EN = FALSE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = FALSE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUM;
                ACCUMULATOR_EN = FALSE;
                DPWM_EN = TRUE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = KX;
                OP_SEL = OPX;
            end
            default:  begin
                STATE_D = S18;
                ADC_EN_D = TRUE;
                U_SEL = U1;
                U0_EN = TRUE;
                U1_EN = FALSE;
                U2_EN = FALSE;
                ADC_ACC_SEL = TRUE;
                ADC_ACC_EN = TRUE;
                SUM_SUB_SEL = SUB;
                ACCUMULATOR_EN = TRUE;
                DPWM_EN = FALSE;
                PSW_EN = FALSE;
                DPWM_START = FALSE;
                DPWM_STOP = FALSE;
                K_SEL = K1;
                OP_SEL = SHIFT3;
            end
        endcase
    end

    always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
        if (RST_N_1V8 == 1'b0) begin
            STATE_Q  <= S17;
        end else begin
            STATE_Q  <= STATE_D;
        end
    end


    /*****************************************************************************
     *                         ADC output synchronization                        *
     *****************************************************************************/
    always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
        if (RST_N_1V8 == 1'b0) begin
            ADC_OUT_SYNC1_Q <= 1'b0;
            ADC_OUT_SYNC2_Q <= 1'b0;
            ADC_EN_Q <= 1'b0;
        end else begin
            ADC_OUT_SYNC1_Q <= ADC_OUT_SYNC1_D;
            ADC_OUT_SYNC2_Q <= ADC_OUT_SYNC2_D;
            ADC_EN_Q <= ADC_EN_D;
        end
    end

    assign ADC_OUT_SYNC1_D = ADC_DOUT;
    assign ADC_OUT_SYNC2_D = ADC_OUT_SYNC1_Q;


    /*****************************************************************************
     *                         Decimation and last inputs                        *
     *****************************************************************************/
    always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
        if (RST_N_1V8 == 1'b0) begin
            U0_Q  <= {(U_WIDTH){1'b0}};
            U1_Q  <= {(U_WIDTH){1'b0}};
            U2_Q  <= {(U_WIDTH){1'b0}};
            DEC_Q <= {(U_WIDTH){1'b0}};
        end else begin
            U0_Q  <= U0_D;
            U1_Q  <= U1_D;
            U2_Q  <= U2_D;
            DEC_Q <= DEC_D;
        end
    end

    assign U0_D = U0_EN ? DEC_Q : U0_Q;
    assign U1_D = U1_EN ? U0_Q : U1_Q;
    assign U2_D = U2_EN ? U1_Q : U2_Q;

    assign DEC_N = ADC_ACC_SEL ? ACCUMULATION_INITIAL_VALUE :
                   DEC_Q + {{(U_WIDTH-1){1'b0}}, ADC_OUT_SYNC2_Q};

    assign DEC_D = ADC_ACC_EN ? DEC_N : DEC_Q;

    assign U = U_SEL == U0 ? U0_Q :
               U_SEL == U1 ? U1_Q :
               U_SEL == U2 ? U2_Q : {U_WIDTH{1'bx}};


    /*******************************************************************************
     *                              Dithering                                      *
     *******************************************************************************/
    `ifdef RAND_DITHER
    always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
        if (RST_N_1V8 == 1'b0) begin
            DTH_Q[(DTH_WIDTH-1):0] <= DTH_SEED;
        end else if (DPWM_START == 1'b1) begin
            DTH_Q[0]  <= DTH_Q[15] ^ DTH_Q[12] ^ DTH_Q[10] ^
                         DTH_Q[8] ^ DTH_Q[6] ^ DTH_Q[2] ^ 1'B1;
            DTH_Q[1]  <= DTH_Q[0];
            DTH_Q[2]  <= DTH_Q[1];
            DTH_Q[3]  <= DTH_Q[2] ^ DTH_Q[15];
            DTH_Q[4]  <= DTH_Q[3];
            DTH_Q[5]  <= DTH_Q[4];
            DTH_Q[6]  <= DTH_Q[5] ^ DTH_Q[15];
            DTH_Q[7]  <= DTH_Q[6];
            DTH_Q[8]  <= DTH_Q[7];
            DTH_Q[9]  <= DTH_Q[8] ^ DTH_Q[15];
            DTH_Q[10] <= DTH_Q[9];
            DTH_Q[11] <= DTH_Q[10];
            DTH_Q[12] <= DTH_Q[11];
            DTH_Q[13] <= DTH_Q[12] ^ DTH_Q[15];
            DTH_Q[14] <= DTH_Q[13];
            DTH_Q[15] <= DTH_Q[14];
        end
    end
```

```verilog

assign DTH = {{(`K_WIDTH - MATH_DECIMAL_WIDTH){1'b0}},
              {DTH_Q[DTH_BIT2], DTH_Q[DTH_BIT1], DTH_Q[DTH_BIT0]},
              {(MATH_DECIMAL_WIDTH - DTH_OUT_WIDTH){1'b0}}};

`else
always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
    if (RST_N_1V8 == 1'b0) begin
        DTH_Q[(DTH_WIDTH-1):0] <= 0;
    end else  if (DPWM_START == 1'b1) begin
        DTH_Q = DTH_Q + {{(DTH_WIDTH-1){1'b0}}, 1'b1};
    end
end

assign DTH = {{(`K_WIDTH - MATH_DECIMAL_WIDTH){1'b0}},
              DTH_Q,
              {(MATH_DECIMAL_WIDTH - DTH_WIDTH){1'b0}}};
`endif


/*******************************************************************************
 *                              Operator selection                             *
 *******************************************************************************/
always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
    if (RST_N_1V8 == 1'b0) begin
        K0_Q <= {`K_WIDTH{1'b0}};
        K1_Q <= {`K_WIDTH{1'b0}};
        K2_Q <= {`K_WIDTH{1'b0}};
        OP_Q <= {(`K_WIDTH+U_WIDTH-1){1'b0}};
    end else begin
        K0_Q <= K0_D;
        K1_Q <= K1_D;
        K2_Q <= K2_D;
        OP_Q <= OP_D;
    end
end


assign K = K_SEL == K0 ? K0_Q :
           K_SEL == K1 ? K1_Q :
           K_SEL == K2 ? K2_Q : {`K_WIDTH{1'bx}};


assign OP_D = OP_SEL == RAND ?
                {{(U_WIDTH-1){1'b0}}, DTH} :
              OP_SEL == SHIFT0 ?
```

```
                 {{(U_WIDTH-1){1'b0}}, {`K_WIDTH{U[0]}} & K}:
               OP_SEL == SHIFT1 ?
                 {{(U_WIDTH-2){1'b0}}, {`K_WIDTH{U[1]}} & K, 1'b0}:
               OP_SEL == SHIFT2 ?
                 {{(U_WIDTH-3){1'b0}}, {`K_WIDTH{U[2]}} & K, 2'b00}:
               OP_SEL == SHIFT3 ?
                 {{(U_WIDTH-4){1'b0}}, {`K_WIDTH{U[3]}} & K, 3'b000}:
               OP_SEL == SHIFT4 ?
                 {{`K_WIDTH{U[4]}} & K, 4'b0000}:
               {(`K_WIDTH+U_WIDTH-1){1'bx}};


/******************************************************************************
 *                              Accumulator                                   *
 ******************************************************************************/
always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
    if (RST_N_1V8 == 1'b0) begin
        ACCUMULATOR_Q <= {(MATH_WIDTH){1'b0}};
    end else begin
        ACCUMULATOR_Q <= ACCUMULATOR_D;
    end
end

assign ACCUMULATOR_D = ACCUMULATOR_EN ? ADDER_OUT : ACCUMULATOR_Q;


/******************************************************************************
 *                          Adder with overflow                               *
 ******************************************************************************/
assign ADDER_IN2_POS_NEG = SUM_SUB_SEL ? ~ADDER_IN2 : ADDER_IN2;
assign ADDER_OUT_W_CARRY = {ADDER_IN1[MATH_WIDTH-1], ADDER_IN1} +
                           {ADDER_IN2_POS_NEG[MATH_WIDTH-1], ADDER_IN2_POS_NEG} +
                           {{MATH_WIDTH{1'b0}}, SUM_SUB_SEL};

assign ADDER_OUT = (ADDER_OUT_W_CARRY[MATH_WIDTH] == 1'b1) &&
                   (ADDER_OUT_W_CARRY[MATH_WIDTH-1] == 1'b0) ?
                   {1'b1, {(MATH_WIDTH-1){1'b0}}} :
                   (ADDER_OUT_W_CARRY[MATH_WIDTH] == 1'b0) &&
                   (ADDER_OUT_W_CARRY[MATH_WIDTH-1] == 1'b1) ?
                   {1'b0, {(MATH_WIDTH-1){1'b1}}} :
                   ADDER_OUT_W_CARRY[(MATH_WIDTH-1):0];

assign ADDER_IN1 = ACCUMULATOR_Q;
assign ADDER_IN2 = {{(MATH_WIDTH-`K_WIDTH-U_WIDTH+1){1'b0}}, OP_Q};


```

```
/*****************************************************************************
 *                                DPWM output                                *
 *****************************************************************************/
always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
    if (RST_N_1V8 == 1'b0) begin
        DPWM_COL_Q    <= {(`DPWM_COL_WIDTH){1'b0}};
        DPWM_ROW_EN_Q <= {(`DPWM_ROW_WIDTH){1'b0}};
        DPWM_ROW_Q    <= {(`DPWM_ROW_WIDTH){1'b0}};
        DPWM_Q        <= {(MATH_WIDTH-MATH_DECIMAL_WIDTH){1'b0}};
    end else begin
        DPWM_COL_Q    <= DPWM_COL_D;
        DPWM_ROW_EN_Q <= DPWM_ROW_EN_D;
        DPWM_ROW_Q    <= DPWM_ROW_D;
        DPWM_Q        <= DPWM_D;
    end
end

assign DPWM_D = DPWM_EN ? ADDER_OUT[(MATH_WIDTH-1):MATH_DECIMAL_WIDTH]:DPWM_Q;

localparam DPWM_LIM_SIGN    = MATH_WIDTH - MATH_DECIMAL_WIDTH - 1;
localparam DPWM_LIM_WIDTH   = DPWM_COL_BIT_WIDTH + DPWM_ROW_BIT_WIDTH;
localparam SIGN_CHECK_WIDTH = DPWM_LIM_SIGN - DPWM_LIM_WIDTH + 1;

assign DPWM = DPWM_Q[DPWM_LIM_SIGN] == 1'b1 &&
              DPWM_Q[(DPWM_LIM_SIGN-1):(DPWM_LIM_WIDTH-1)] !=
              ↪  {(SIGN_CHECK_WIDTH){1'b1}} ?
                  {1'b1, {(DPWM_LIM_WIDTH-1){1'b0}}} :
              DPWM_Q[DPWM_LIM_SIGN] == 1'b0 &&
              DPWM_Q[(DPWM_LIM_SIGN-1):(DPWM_LIM_WIDTH-1)] !=
              ↪  {(SIGN_CHECK_WIDTH){1'b0}} ?
                  {1'b0, {(DPWM_LIM_WIDTH-1){1'b1}}} :
              DPWM_Q[(DPWM_LIM_WIDTH-1):0];

assign DPWM_COL_D =
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h8 ? 16'h0001 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h7 ? 16'hFFFF :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h6 ? 16'hFFFF :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h5 ? 16'hFFFF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h0 ? 16'h0001 :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h1 ? 16'h0003 :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h2 ? 16'h0007 :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h3 ? 16'h000F :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h4 ? 16'h001F:
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h5 ? 16'h003F :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h6 ? 16'h007F :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h7 ? 16'h00FF :
```

```verilog
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h8 ? 16'h01FF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'h9 ? 16'h03FF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'hA ? 16'h07FF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'hB ? 16'h0FFF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'hC ? 16'h1FFF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'hD ? 16'h3FFF :
       DPWM[(DPWM_COL_BIT_WIDTH-1):0] == 4'hE ? 16'h7FFF :
       16'hFFFF;

assign DPWM_ROW_EN_D =
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h8 ? 12'hFFF :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h9 ? 12'hFFF :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hA ? 12'hFFE :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hB ? 12'hFFC :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hC ? 12'hFF8 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hD ? 12'hFF0 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hE ? 12'hFE0 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'hF ? 12'hFC0 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h0 ? 12'hF80 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h1 ? 12'hF00 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h2 ? 12'hE00 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h3 ? 12'hC00 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h4 ? 12'h800 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h5 ? 12'h000 :
       DPWM[(DPWM_LIM_WIDTH-1):DPWM_COL_BIT_WIDTH] == 4'h6 ? 12'h000 :
       12'h000;

assign DPWM_ROW_D = {~DPWM_ROW_EN_D[(`DPWM_ROW_WIDTH-2):0], 1'b1};


/*******************************************************************************
 *                              Cycle control                                  *
 *******************************************************************************/
always @(posedge CLK_50M_1V8 or negedge RST_N_1V8) begin
    if (RST_N_1V8 == 1'b0) begin
        PSW_EN_Q <= 1'b0;
        DPWM_START_Q <= 1'b0;
        DPWM_STOP_Q <= 1'b1;
    end else begin
        PSW_EN_Q <= PSW_EN_D;
        DPWM_START_Q <= DPWM_START_D;
        DPWM_STOP_Q <= DPWM_STOP_D;
    end
end

assign PSW_EN_D = PSW_EN ? 1'b1 : PSW_EN_Q;
assign DPWM_START_D = PSW_EN_Q & DPWM_START;
assign DPWM_STOP_D = PSW_EN_Q & DPWM_STOP;

/******************************************************************************
 *                      Inputs and outputs mapping                          *
 ******************************************************************************/
assign ADC_DOUT = ADC_DOUT_1V8;
assign ADC_EN_1V8 = ADC_EN_Q;
assign DPWM_ROW_EN_N_1V8 = DPWM_ROW_EN_Q;
assign DPWM_ROW_1V8 = DPWM_ROW_Q;
assign DPWM_COL_1V8 = DPWM_COL_Q;
assign PSW_EN_1V8 = PSW_EN_Q;
assign DPWM_STOP_1V8 = DPWM_STOP_Q;
assign DPWM_START_1V8 = DPWM_START_Q;
assign K0_D = K0_1V8;
assign K1_D = K1_1V8;
assign K2_D = K2_1V8;

endmodule
```

ANNEX F – ADC Test Bench Stimulus

```python
from vagen import *

#Create a module
mod = HiLevelMod("DIG_BUCK_ADC_STML")

vdd1v8 = mod.par(name = "VDD_1V8_PAR", value = 1.8)
cmm = mod.par(name = "CMM_PAR", value = 0.9)
seq = mod.par(name = "SEQUENCE_PAR", value = 1)

#Pins
VDD_1V8    = mod.vdc(name="VDD_1V8",   width  = 1, direction="inout", rise =
↪  500e-9, fall = 500e-9)
digDomain = VDD_1V8
VP_1V8     = mod.vdc(name="VP_1V8",    width  = 1, direction="output", rise =
↪  100e-9, fall = 100e-9)
VN_1V8     = mod.vdc(name="VN_1V8",    width  = 1, direction="output", rise =
↪  100e-9, fall = 100e-9)
ADC_EN_1V8= mod.dig(name="ADC_EN_1V8",    domain = digDomain, width=1,
↪  direction="output")
CLK_1V8    = mod.dig(name="CLK_50M_1V8", domain = digDomain, width=1,
↪  direction="output", serRes = 1e3)
OUT_1V8    = mod.dig(name="ADC_DOUT_1V8", domain = digDomain, width=1,
↪  direction="input")
VDOUT      = mod.vdc(name="VDOUT",     width  = 1, direction="output", rise =
↪  10e-9,  fall = 10e-9)
STEPPIN    = mod.vdc(name="STEPPIN",   width  = 1, direction="output", rise =
↪  10e-9,  fall = 10e-9)
OFFSETPIN = mod.vdc(name="OFFSETPIN", width  = 17,  direction="output", rise =
↪  10e-9,  fall = 10e-9)

#Test sequence
MARKER = mod.marker("ADC_TRAN")
DOUT = mod.var(value = 0)
LASTDOUT = mod.var(value = 0)
VMAX = mod.var(value = 0.0)
VMIN = mod.var(value = 0.0)
VAVG = mod.var(value = 0.0)
x = mod.var(value = 0.0)
ntrsh = [mod.var(value = 0) for i in range(0,15)]
xtrsh = [mod.var(value = 0.0) for i in range(0,17)]
j = mod.var(value = 0)
i = mod.var(value = 0)
```

```
T = (1/50)
LASTVAL = mod.var(value = 0.0)
STEP = mod.var(value = 6.25e-6)
RESOL = 22
NMIN = 3
NMAX = 18


#Convert
def convert():
    return CmdList(
        CLK_1V8.write(True),
        ADC_EN_1V8.write(True),
        DOUT.eq(0),
        For(j.eq(0), j < RESOL, j.eq(j + 1))(
            WaitUs(T/2),
            CLK_1V8.write(False),
            WaitUs(T/2),
            CLK_1V8.write(True),
            DOUT.eq(DOUT + Integer(OUT_1V8.read())),
        ),
        ADC_EN_1V8.write(False),
        WaitUs(T/2),
        CLK_1V8.write(False),
        WaitUs(T/2),
    )


def search(target, resol, step, minval, variable):
    return CmdList(
        variable.eq(minval),
        For(i.eq(2**(resol-1)), i > 0, i.eq(i >> 1))(
            variable.eq(variable + Real(i)*step),
            VP_1V8.applyV(variable),
            WaitUs(1),
            convert(),
            If(DOUT > target)(
                variable.eq(variable - Real(i)*step)
            )
        )
    )


#Sequence
mod.seq(True)(

    #SUPPLY UP
    VDD_1V8.applyV(vdd1v8),
    WaitUs(2),
    VP_1V8.applyV(cmm),
    VN_1V8.applyV(cmm),
    WaitUs(2),
    MARKER.mark("SUPPLY_OK"),

    #Find Vmax
    search(NMAX, 13, STEP, cmm, VMAX),

    #Find Vmin
    search(NMIN, 13, STEP, cmm - 50e-3, VMIN),

    #Update ideal step
    STEPPIN.applyV((VMAX-VMIN)/(NMAX-NMIN)),
    STEP.eq((VMAX-VMIN)/(20*(NMAX-NMIN))),
    WaitUs(1),
    MARKER.mark("IDEAL_STEP_MEAS"),
    WaitUs(1),
    LASTVAL.eq(VMIN-19.5*STEP),
    LASTDOUT.eq(NMIN),
    xtrsh[0].eq(VMIN-10*STEP),
    xtrsh[16].eq(VMAX+10*STEP),
    VMIN.eq(VMIN-9.5*STEP),
    VMAX.eq(VMAX+10.5*STEP),



    #CONVERT
    MARKER.mark("SWEEP_START"),
    For(x.eq(VMIN), x <= VMAX, x.eq(x + STEP))(
        VP_1V8.applyV(x),
        WaitUs(1),
        convert(),
        CmdList(*[If(DOUT == (i+NMIN))(ntrsh[i-1].eq(ntrsh[i-1] + 1),
        ↪  xtrsh[i].eq(xtrsh[i] + x)) for i in range(1,16)]),
        If((DOUT != LASTDOUT))(
            STEPPIN.applyV((x - LASTVAL)/Real(DOUT - LASTDOUT)),
            LASTVAL.eq(x),
            LASTDOUT.eq(DOUT),
        ),
        VDOUT.applyV(10e-3*Real(DOUT)),
    ),
    CmdList(*[xtrsh[i].eq(xtrsh[i]/Real(ntrsh[i-1])) for i in range(1,16)]),
    CmdList(*[OFFSETPIN[i].applyV(xtrsh[i] - cmm) for i in range(0,17)]),
    WaitUs(1),
    MARKER.mark("SWEEP_STOP"),

    #PWRU-DOWN
    WaitUs(0.1),
    VDD_1V8.applyV(0),
    WaitUs(4)
)

print(mod.getVA())

```

ANNEX G – DPWM Test Bench Stimulus

```python
#Create a module
mod = HiLevelMod("DIG_BUCK_DPWM_STML")

trim = mod.par(name = "TRIM_PAR", value = 255)
vdd5v0 = mod.par(name = "VDD_5V0_PAR", value = 5.0)
vdd1v8 = mod.par(name = "VDD_1V8_PAR", value = 1.8)
SEQ = mod.par(name = "SEQ_PAR", value = 0)

#Pins
VDD_1V8 = mod.vdc(name="VDD_1V8", width=1, direction="inout", rise = 1e-6, fall = 1e-6)
VDD_5V0 = mod.vdc(name="VDD_5V0", width=1, direction="inout", rise = 1e-6, fall = 1e-6)
digDomain = VDD_1V8
PSW_EN_1V8 = mod.dig(name="PSW_EN_1V8", domain = digDomain, width=1, value = True, direction="output")
DPWM_OUT_1V8 = mod.dig(name="DPWM_OUT_1V8", domain = digDomain, width=1, direction="input")
DPWM_TRIM_1V8 = mod.dig(name="DPWM_TRIM_1V8", domain = digDomain, width=4, value = trim, direction="output")
DPWM_STOP_1V8 = mod.dig(name="DPWM_STOP_1V8", domain = digDomain, width=1,value = True, direction="output")
DPWM_START_1V8 = mod.dig(name="DPWM_START_1V8", domain = digDomain, width=1, direction="output")
DPWM_ROW_EN_N_1V8 = mod.dig(name="DPWM_ROW_EN_N_1V8", domain = digDomain, width=12, direction="output")
DPWM_ROW_1V8 = mod.dig(name="DPWM_ROW_1V8", domain = digDomain, width=12, direction="output")
DPWM_COL_1V8 = mod.dig(name="DPWM_COL_1V8", domain = digDomain, width=16, direction="output")
TRIMPIN = mod.vdc(name = "TRIM_VAL", width=1, direction="inout", rise = 100e-12, fall = 100e-12)
VALID_SWEEP = mod.vdc(name = "VALID_SWEEP", width=1, direction="inout", rise = 10e-12, fall = 10e-12)
IN_VAL = mod.vdc(name = "IN_VAL", width=1, direction="inout", rise = 10e-12, fall = 10e-12)

cycleEnd = Cross(DPWM_OUT_1V8.diffHalfDomain, "falling", 10e-12, 1e-9)
curTime = mod.var(0.0)
trimval = mod.var(0)
first = mod.var(False)
last = mod.var(False)
```

```
vari = mod.var(0)

def writeOutput(val):
    return CmdList(
        DPWM_COL_1V8.write((1 << (( val & 15 ) + 1)) - 1),
        DPWM_ROW_1V8.write(1 << ( val >> 4 )),
        DPWM_ROW_EN_N_1V8.write(~((1 << ( val >> 4 )) - 1))
    )

def oneCycle(val, firstVar, lastVar):
    return CmdList(
        writeOutput(val),
        WaitUs(1/50),
        DPWM_STOP_1V8.write(False),
        DPWM_START_1V8.write(True),
        WaitUs(1/50),
        DPWM_START_1V8.write(False),
        WaitUs(0.5/50),
        If(DPWM_OUT_1V8.read())(
            firstVar.eq(True)
        ),
        WaitUs(22.5/50),
        If(DPWM_OUT_1V8.read())(
            lastVar.eq(True)
        ),
        DPWM_STOP_1V8.write(True)
    )

#Test sequence
def sweepSeq():
    return CmdList(
        first.eq(False),
        last.eq(False),
        vari.eq(0),
        While((vari < 192) & (~first))(
            oneCycle(vari, first, last),
            vari.inc()
        ),
        vari.dec(),
        VALID_SWEEP.applyV(1),
        IN_VAL.applyV(Real(vari)*0.01),
        While((vari < 192) & (~last))(
            oneCycle(vari, first, last),
            vari.inc()
        ),
        WaitUs(1/50),
        VALID_SWEEP.applyV(0)
    )

#Trimming
def trimSeq():
    cmds = CmdList(writeOutput(120))
    i = 8
    while (i > 0):
        cmds.append(
            trimval.eq(trimval + i),
            DPWM_TRIM_1V8.write(trimval),
            WaitUs(1/50),
            DPWM_STOP_1V8.write(False),
            DPWM_START_1V8.write(True),
            curTime.eq(abstime),
            WaitUs(1/50),
            DPWM_START_1V8.write(False),
            WaitSignal(cycleEnd),
            If((abstime-curTime) < (1.844/3.7*500e-9))(
                trimval.eq(trimval - i)
            ),
            DPWM_STOP_1V8.write(True)
        )
        i = i >> 1
    cmds.append(TRIMPIN.applyV(Real(trimval)*0.01))
    return cmds


#SEQ 1. Trimming.
mod.seq(SEQ  == 1)(
    VDD_1V8.applyV(1.8),
    VDD_5V0.applyV(3.7),
    WaitUs(3),

    trimSeq(),
    WaitUs(3),

    VDD_1V8.applyV(0),
    VDD_5V0.applyV(0),
    WaitUs(10),
)

#SEQ 2. Sweep.
MARKER = mod.marker("DPWM_TRAN")
mod.seq(SEQ == 2)(
    VDD_1V8.applyV(vdd1v8),
    VDD_5V0.applyV(vdd5v0),
    WaitUs(3),
    MARKER.mark("SUPPLY_OK"),
    WaitUs(1),

    MARKER.mark("SWEEP_START"),
    sweepSeq(),
    MARKER.mark("SWEEP_STOP"),

    #Finish
    WaitUs(0.5),
    VDD_1V8.applyV(0),
    VDD_5V0.applyV(0),
    WaitUs(3),
)
print(mod.getVA())

```

ANNEX H – System Level Test Bench Stimulus

```python
from vagen import *

#Create a module
mod = HiLevelMod("DIG_BUCK_TOP_STML")
K0_PAR = mod.par(name="K0_PAR", value = 0)
K1_PAR = mod.par(name="K1_PAR", value = 0)
K2_PAR = mod.par(name="K2_PAR", value = 0)
seq = mod.par(name="SEQ_PAR", value = 0)

#Pins
VDD_5V0 = mod.vdc(name="VDD_5V0", width=1, direction="inout", rise = 1e-6, fall =
↪  1e-6)
VDD_1V8 = mod.smu(name="VDD_1V8", width=1, direction="inout")
VDD_1V8_INT = mod.idc(name="VDD_1V8_INT", width=1, rise = 1e-6, fall = 1e-6)
digDomain = VDD_1V8
CLK_50M_1V8 = mod.dig(name="CLK_50M_1V8", domain = digDomain, width=1,
↪  direction="output")
RST_N_1V8 = mod.dig(name="RST_N_1V8", domain = digDomain, width=1,
↪  direction="output")
ADC_VREF_CONF_1V8 = mod.dig(name="DIG_BUCK_VOUT_CONF_1V8", value = 8, domain =
↪  digDomain, width=4, direction="output")
DPWM_TRIM_1V8 = mod.dig(name="DIG_BUCK_DPWM_TRIM_1V8", value = 9, domain =
↪  digDomain, width=4, direction="output")
DIG_BUCK_K1_1V8 = mod.dig(name="DIG_BUCK_K1_1V8", domain = digDomain, value =
↪  K1_PAR, width=14, direction="output")
DIG_BUCK_K0_1V8 = mod.dig(name="DIG_BUCK_K0_1V8", domain = digDomain, value =
↪  K0_PAR, width=14, direction="output")
DIG_BUCK_K2_1V8 = mod.dig(name="DIG_BUCK_K2_1V8", domain = digDomain, value =
↪  K2_PAR, width=14, direction="output")

#Test sequence
MARKER = mod.marker("TRAN")
clk = mod.clock(CLK_50M_1V8)
sw = mod.sw(VDD_1V8, VDD_1V8_INT, cond = 1)

mod.seq(seq == 1)(
    #SUPPLY UP
    VDD_5V0.applyV(3.7),
    WaitUs(2),
    VDD_1V8.applyV(1.8, 2.4),

    WaitUs(20),
    RST_N_1V8.write(True),
    VDD_1V8.applyR(1e9),
    WaitUs(1),
    MARKER.mark("START"),

    #PWRU-UP
    WaitUs(4),
    RST_N_1V8.write(True),
    WaitUs(1/50),
    clk.on(50e6),
    MARKER.mark("END_PWR_UP"),

    WaitUs(300),
    VDD_1V8_INT.applyI(200e-3),
    WaitUs(300),
    VDD_1V8_INT.applyI(0),
    WaitUs(300),

    #PWRUP-DOWN
    RST_N_1V8.write(False),
    WaitUs(1),
    clk.off(),
    WaitUs(4),
    MARKER.mark("END_PWR_DOWN"),

    #Finish
    WaitUs(4)
)


mod.seq(seq == 2)(
    #SUPPLY UP
    VDD_5V0.applyV(3.7),
    WaitUs(2),
    VDD_1V8.applyV(1.8, 2.4),

    WaitUs(20),
    RST_N_1V8.write(True),
    VDD_1V8.applyR(1e9),
    WaitUs(1),
    MARKER.mark("START"),

    #PWRU-UP
    WaitUs(4),
    RST_N_1V8.write(True),
    WaitUs(1/50),
    clk.on(50e6),
    MARKER.mark("END_PWR_UP"),

    WaitUs(300),
    VDD_5V0.applyV(5.5),
    WaitUs(300),
    VDD_5V0.applyV(3.7),
    WaitUs(300),

    VDD_5V0.applyV(2.7),
    WaitUs(300),
    VDD_5V0.applyV(3.7),
    WaitUs(300),

    #PWRUP-DOWN
    RST_N_1V8.write(False),
    WaitUs(1),
    clk.off(),
    WaitUs(4),
    MARKER.mark("END_PWR_DOWN"),

    #Finish
    WaitUs(4)
)


mod.seq(seq == 3)(
    #SUPPLY UP
    VDD_5V0.applyV(3.7),
    WaitUs(2),
    VDD_1V8.applyV(1.8, 2.4),

    WaitUs(20),
    RST_N_1V8.write(True),
    VDD_1V8.applyR(1e9),
    WaitUs(1),
    MARKER.mark("START"),

    #PWRU-UP
    WaitUs(4),
    RST_N_1V8.write(True),
    WaitUs(1/50),
    clk.on(50e6),
    MARKER.mark("END_PWR_UP"),

    WaitUs(300),
    ADC_VREF_CONF_1V8.write(13),
    WaitUs(300),
    ADC_VREF_CONF_1V8.write(8),
    WaitUs(300),

    ADC_VREF_CONF_1V8.write(3),
    WaitUs(300),
    ADC_VREF_CONF_1V8.write(8),
    WaitUs(300),

    #PWRUP-DOWN
    RST_N_1V8.write(False),
    WaitUs(1),
    clk.off(),
    WaitUs(4),
    MARKER.mark("END_PWR_DOWN"),

    #Finish
    WaitUs(4)
)


print(mod.getVA())

```