µSystems Research Group

School of Engineering

Newcastle University

Power-compute co-design for robust pervasive IoT applications

Sergey Mileiko

Technical Report Series

NCL-EEE-MICRO-TR-2020-218

November 2020

Contact: `s.mileiko2@ncl.ac.uk`

NCL-EEE-MICRO-TR-2020-218

µSystems Research Group
School of Engineering
Merz Court
Newcastle University
Newcastle upon Tyne, NE1 7RU, UK

`http://async.org.uk/`

Abstract

The modern development of internet of things (IoT) requires the IoT devices to be more compact and energy autonomous. Many of them require to be able to operate with unstable and low power supplies that come from various energy sources such as energy harvesters. This creates a challenge for building IoT devices that need to be robust to energy variations.

In this research we propose methods for improving energy characteristics of IoT devices from the perspective of two main challenges: (i) improving the efficiency and stability of power regulators, and (ii) enhancing the energy robustness of the IoT devices. The existing design methods do not consider these two aspects holistically. One important feature of our approach is holistic use of event-based, temporal representation of data, which involves using asynchronous techniques and duty-cycle-based encoding.

For power regulation we use switched-capacitor converters (SCC) because they offer compactness and ease of on-chip implementation. In this research we adapt the existing methods and develop new techniques for SCC design based on asynchronous circuits. This allows us to improve their performance and stability. We also investigate the methods of parasitic charge redistribution, and apply them to self-oscillating SCC, improving their performance. The key contribution within (i) is development of the methods of SCC design with improved characteristics.

The majority of novel IoT systems are shifting towards the “AI at the edge” vision, for example, involving neural networks (NN). We consider a perceptron-based neural network as a typical IoT computing device. In our research we propose a novel NN design approach using the principle of pulse-width modulation (PWM). PWM-encoded signals represent information with their duty cycle values which may be made independent of the voltages and frequencies of the carrier signals. As a result, the device is more robust to voltage variations, and, thus, the power regulation can be simplified. This is the second major contribution addressing challenge (ii).

The advantages of the proposed methods are validated with simulations in the Cadence environment. The simulations demonstrate the operation of the designed power regulators, and the improvements of their efficiency. The simulations also demonstrate the principle of operation of the PWM-based perceptron and prove its power and frequency elasticity.

The thesis gives future research directions into a deeper study of the holistic co-design of a variation-robust power-compute paradigm and its impact on developing future IoT applications.

Contents

List of Figures

List of Tables

List of Acronyms

AF - Activation Function

AI - Artificial Intelligence

BP - Back-Propagation

CSC - Complete State Coding

EMI - Electromagnetic Interference

FP - Floating Point

IoT - Internet of Things

KVL - Kirchhoff's Voltage Law

MIM - Metal-Insulator-Metal

MOM - Metal-Oxide-Metal

MOS - Metal-Oxide-Semiconductor

NN - Neural Network

PWM - Pulse Width Modulation

SCC - Switched-Capacitor Converter

STG - Signal Transition Graph

VAC - Voltage Accumulator

Acknowledgments

I would like to express my gratitude to Alex Yakovlev, my supervisor, for his wisdom and guidance throughout my postgraduate studies. I would also like to thank my second supervisor, Danil Sokolov, for introducing me to the world of asynchronous circuits and providing invaluable guidance during my research. I am grateful to Alexander Drozd, who supervised me during my master studies; he gave me a basic knowledge of hardware design and introduced me to the process of science.

I also extend my thanks to Alexander Kushnerov, who acted as my unofficial supervisor, sharing his huge experience in the field of switched-capacitor converters. The methods proposed in chapter 3, are developed in close collaboration with him. A huge impact on my research was made by Rishad Shafik, Fei Xia, and Thanasin Bunnam, who helped me with an understanding of neural network design. Chapter 4 was born as a result of our cooperative work. My colleagues, Alex Bystrov and Andrey Mokhov have helped me a lot, by having very informative discussions about my research.

I am thankful to my girlfriend, Maryna, for all of her love, support and patience throughout my PhD. She was there to help me at difficult times, and to share in the good times. I am grateful to my family for the concern and support during this research. My thanks to all my friends, especially to Georgy Lukyanov, who made my life even more fun by involving me into various sport and social activities.

This research was supported by EPSRC, grants SAVVIE (EP/K012908/1) and A4A (EP/L025507/1).

Chapter 1

Introduction

1.1 Challenges of IoT applications

Advances in sensing devices are causing a shift towards the fourth industrial revolution [1]. The number of Internet of Things (IoT) devices is expected to grow significantly in the next few years. The modern IoT devices are facing two grand challenges: *energy efficiency* and *energy autonomy*.

The progress of IoT devices is moving towards the full independence of these devices in terms of energy consumption. This progress relies on the development of energy harvesting technologies. Modern energy harvesters can receive energy from everywhere. However, as power sources they cannot provide a stable supply voltage for the devices.

Typical power flow in an IoT device is shown in Fig. 1.1. The device receives power from the environment using an energy harvester. The power regulator converts the conditioned power coming from the harvester to stable voltage levels that supply the computational unit.

Thus, IoT devices require quite complex techniques of power regulation. On the one hand, the power regulators that can produce a stable voltage to the devices can be rather complicated and consume a lot of power by themselves. On the other hand, the simplification of the power regulators leads to instability of the supply voltage and results in errors in the device's operation.

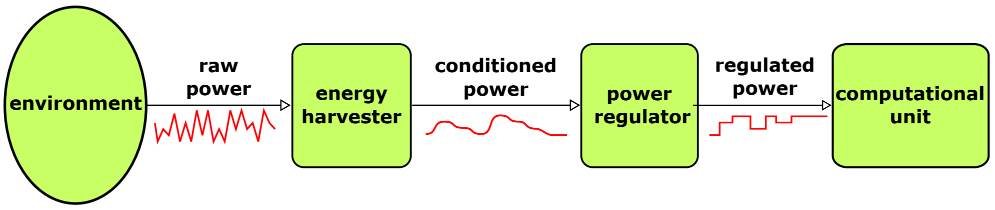

Figure 1.1: Typical power flow in an IoT device.

In this research we approach the problem of IoT power supply from two perspectives:

- We propose methods for improving the efficiency and stability of the power regulators;
- We improve the power robustness of the targeted IoT devices, thereby allowing the power regulation part to be simplified.

1.2 Requirements to the power regulators for IoT devices

There are two common types of DC-DC converters: linear regulators and switching converters.

In linear regulators, input and output are connected directly and their efficiency depends on the relation between input and output voltages. If difference between them is big, then the efficiency of the converter can be quite low [2]. Linear regulators are used in applications, where power losses are not critically important, and demanded voltage levels have to be obtained at any cost.

Switching converters use energy-storing components, which are periodically connected to the power supply. Energy is delivered by portions from the input to the output. This type of converters has higher efficiency than linear regulators. The two main groups of switching DC-DC converters are *inductive* and *capacitive* converters.

Inductive DC-DC converters use one or several inductors for energy transfer. They are very widespread because of their universality. They can be used in different applications with different voltage and current levels. However, inductive converters have several disadvantages [3]:

- It is expensive to realise an inductance on-chip, thus, it is always implemented externally. It has a relatively big size, thus the converter occupies a lot of space;
- The output of the converter produces voltage spikes that can be harmful to the components under the voltage supply. These voltage spikes must be damped by filters that also require extra space;
- The pulsating input current of the inductive converter can produce an electromagnetic interference (EMI) that can affect other components.

Capacitive DC-DC converters, also known as switched-capacitor converters (SCCs), are often used in low power applications. They do not have those disadvantages that inductive converters have [2]. They do not produce high EMI. Capacitive converters, in contrast to inductive ones, can be easily implemented on-chip. That gives them a huge advantage in terms of size. Thus, they can be used in small devices without increasing their dimensions. But in comparison to linear regulators, SCCs keep higher efficiency, especially at target voltages.

Considering the advantages of SCCs, it has been chosen as the best option for power regulation of the IoT applications. The main problem of SCCs, however, is their low efficiency caused by the following reasons [3]:

- Efficiency reduction when the expected output voltage is different from the target voltage that can be produced by the SCC;
- The shoot-through currents, which appear when the transistors that are supposed to operate in counter-phase, are turned on at the same time.
- The losses associated with the parasitic capacitances;

A solution to the first problem is straightforward: increasing the number of converter's ratios. There are several solutions that successfully resolve this issue, such as in [4–6]. However, these solutions lead to more complicated control circuitry for such SCCs.

In order to resolve the problem with the shoot-through currents, the SCCs require more complex control: it must prevent the short-through currents by separating the switching phases in time.

The importance of SCC controllers calls for advanced methods of their design. The power control could significantly benefit from the use of asynchronous logic [7, 8] that does not rely on the global clock signal and performs at the pace determined by the operating conditions [9]. Thus such circuits are adaptable to the rate of changes in the controlled system and can react to the asynchronous signals from the sensors without the need for synchronizers. These benefits are already realised by the analogue engineers who are keen to use asynchronous circuits. At present they typically perform an ad hoc design of clock-less power control circuits and rely on exhaustive simulation to validate their correctness. Some solutions with the successful use of asynchronous logic for power regulation have been reported in [8–10].

In this research, we propose the methods of asynchronous controller design for SCCs. These methods lead to improving the SCC efficiency, and the efficiency of the IoT device in general.

1.3 Improvement of power robustness of IoT devices

Modern IoT devices are often used for the tasks of decision making. The most common way of performing these tasks is using Neural Networks (NNs) [11]. We choose NNs as our targeted IoT device because of the following reasons:

- NNs are quite common in IoT applications, and the modern trends demonstrate the increase of the popularity of NNs in IoT;
- Computations in NNs do not require a high degree of accuracy. Some inaccuracy in NNs can be compensated by an algorithm of updating weights.

We propose an approach of migrating the computations involved in NNs to the Pulse-Width Modulation (PWM) domain. Our key motivation to use the duty-cycle time-domain representation of data is base on its potential fundamental resilience to dynamic variations in the amplitude and frequency of the information-carrying signal. We assume that such variations are inevitable for energy-autonomous systems that draw energy from the environment. The other motivating factor is the natural ability of CMOS logic to perform multiplication and addition operation on the duty-cycled inputs. This is enabled by the inherent effects of proportionally ratioed current switching in CMOS networks between P and N subnets during the operational cycle of each gate. This gives way to implementing the PWM-based compute functions directly in CMOS logic gates.

1.4 Summary

The future development of IoT devices will be directed towards improving their size and energy autonomy. In this work, we approach the IoT improvements from two sides: improving the efficiency and stability of the power regulator, as well as increasing the robustness of the IoT device by itself.

The switched-capacitor converter is chosen as a power regulator for an IoT device. We propose a method of designing an efficient control for a general type of SCCs, as well as a method of reducing the parasitic losses for self-oscillating SCCs.

From another side, we help to reduce the complexity of the design of the converter part by improving the IoT device's robustness to the voltage variations by switching the computations into a PWM domain.

The main contributions of this thesis are as follows:

- We develop methods for the design of asynchronous controllers for two-phase and multi-phase SCCs.
- We apply the method of parasitic charge redistribution to self-oscillating SCCs. This results in more efficient converters compared to state-of-the-art solutions.
- We propose a novel design approach for a perceptron-based NN using the principle of PWM. This approach leads to the improvement of power elasticity of the NN-based device.

1.5 Main publications on the thesis

- Mileiko S, Kushnerov A, Sokolov D, Yakovlev A. *Self-timed control of two-phase switched capacitor converters*. In: 2016 IEEE International Conference on the Science of Electrical Engineering, ICSEE 2016. 2017, Eilat, Israel: Institute of Electrical and Electronics Engineers Inc. (materials of this paper are used in chapter 3).

- Mileiko S, Kushnerov A, Sokolov D, Yakovlev A. *Self-timed control of multiphase switched capacitor converters*. In: 2017 European Conference on Circuit Theory and Design (ECCTD). 2017, Catania, Italy: IEEE. (materials of this paper are used in chapter 3).

- Mileiko S, Kushnerov A, Sokolov D, Yakovlev A. *Self-oscillating switched capacitor converter with parasitic charge redistribution*. In: Annual Research Conference (ARC). 2018, Newcastle, UK. (materials of this paper are used in chapter 3).

- Mokhov A, De Gennaro A, Tarawneh G, Wray J, Lukyanov G, Mileiko S, Scott J, Yakovlev A, Brown A. *Language and hardware acceleration backend for graph processing*. In: Languages, Design Methods, and Tools for Electronic System Design. 2018, Verona, Italy: Springer Verlag.

- Mileiko S, Shafik R, Yakovlev A, Edwards J. *A pulse width modulation based power-elastic and robust mixed-signal perceptron design*. In: Design, Automation and Testing in Europe (DATE). 2019, Florence, Italy. (materials of this paper are used in chapters 4 and 5).

- Mileiko S, Bunnam T, Xia F, Shafik R, Yakovlev A, Das S. *Neural Network Design for Energy-Autonomous AI Applications using Temporal Encoding*. In: Philosophical Transactions of the Royal Society. Volume 378. Issue 2164. 2019.(materials of this paper are used in chapters 4 and 5).

- Mileiko S, Bunnam T, Xia F, Shafik R, Yakovlev A. *Dynamics of Time-domain Power-elastic Circuits for Pervasive Machine Learning*. In: International Symposium on Circuits & Systems (ISCAS). 2020, Seville, Spain.(materials of this paper are used in chapter 5).

1.6 Thesis layout

This thesis is organised as follows:

Chapter 1 - Introduction. In this chapter, we briefly discuss the motivations for the thesis and summarise the contributions.

Chapter 2 - Background. We discuss the major challenges of SCC design and the existing methods that address these challenges. We overview the methods of asynchronous circuits design based on signal transition graphs, which will be used for SCC control design. We investigate the modern trends in AI hardware design.

Chapter 3 - Design of efficient SCCs. In this chapter, we develop methods of asynchronous controller design for two-phase and multi-phase SCCs. We also combine the method of parasitic charge redistribution with the method of design of self-oscillating SCCs in order to improve the efficiency of this type of converters.

Chapter 4 - Power-elastic PWM-based perceptron. This chapter proposes a novel approach to design a perceptron-based neural network in PWM domain. The chapter includes the theoretical background supported by the design of the perceptron prototype. It ends up in the design of NN based on PWM perceptrons.

Chapter 5 - Simulation results. To validate the methods proposed in chapters 3 and 4 we simulate the proposed designs in CADENCE and MATLAB environment. The simulation results are analysed and compared with state-of-the-art solutions.

Chapter 6 - Conclusions. This is a summary of the contributions as discussed in this thesis, and future research areas for the development of power and computational parts of IoT devices.

Chapter 2

Background

In this chapter, we review the current achievements in the investigated field. Section 2.1 describes the principle of operation of the capacitive power regulators. We start from the simplest two-phase SCC that demonstrates the basic principles of energy transferring in SCCs. After that, we move to more complex types of SCCs, such as multi-phase and self-oscillating SCCs. In this section, we also describe the main challenges of the on-chip SCC design. We investigate the losses associated with the shoot-through currents and the parasitic bottom plate capacitance. We also review the state-of-the-art solutions that handle these challenges.

In section 2.2, we review the methods of asynchronous circuits design that will be applied to SCC control design methods.

Section 2.3 describes the modern achievements in the perceptron-based neural network design and the challenges associated with it.

2.1 Principle of operation and challenges of SCCs

Capacitive DC-DC converters, also known as switched-capacitor converters (SCCs), are often used in low power applications. They do not have those disadvantages that inductive converters have [2]. They do not produce high Electromagnetic Interference (EMI). Capacitive converters, in contrast to inductive ones, can be easily implemented on-chip. That gives them a huge advantage in terms of size. Thus, they can be used in small devices without increasing their dimensions. Moreover, in comparison to linear regulators, SCCs keep higher efficiency, especially at target voltages. Considering the advantages of SCCs, they have been chosen as the best option for power regulation of the IoT applications.

2.1.1 Two-phase SCC

A typical SCC cell is shown in Fig. 2.1a. It consists of four switches and one capacitor. In principle, each switch in this cell may be turned on/off independently. However, in the so-called two-phase SCCs [2] all the switches are divided into two groups, each of which is turned on/off in counter-phase. Depending on the control scheme, four different SCCs can be built on a single cell: voltage follower, inverter, voltage doubler and divider by two (the latter is shown in Fig. 2.1b). Two clocks ϕ_1 and ϕ_2 control the pairs of switches S_1, S_3 and S_2, S_4 respectively. To avoid the shoot-through currents, ϕ_1 and ϕ_2 must have dead time, as shown in Fig. 2.2a. However, large changes in the SCC operating conditions, e.g. temperature fluctuations or voltage drop, can lead to large skews of the clock pulses, such that ϕ_1 and ϕ_2 overlap, as shown in Fig. 2.2b. On the other hand, the switches can be more sensitive than the controller and, under some conditions, may turn on/off slower. If the dead time has not increased, the SCC still will suffer from shoot-through currents.

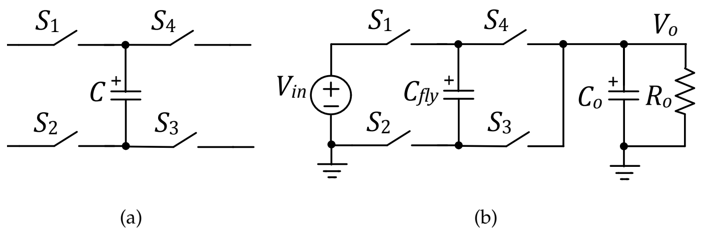

Figure 2.1: Typical SCC cell (a) and its use in the divider by two SCC (b).

Denoting the on-resistances of the switches by $R_1, ..., R_4$, we can represent the overlapping case of Fig. 2.2b by the topologies shown in Fig. 2.3. When these topologies cycle, the SCC reaches a steady state. If the load is disconnected, the steady state means

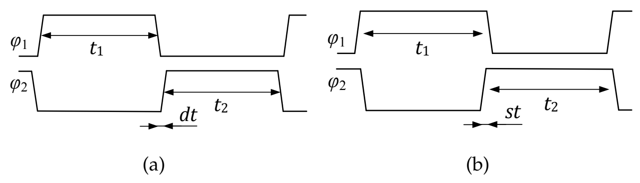

Figure 2.2: Two clocks ϕ_1 and ϕ_2 with dead time dt (a) and overlap st (b).

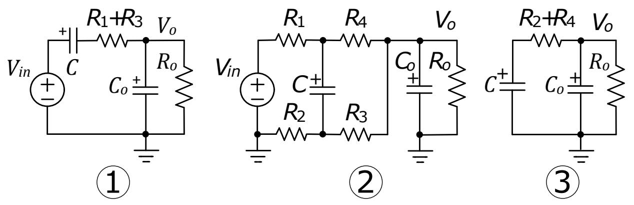

Figure 2.3: Topologies of the dividing by two SCC including the parasitic one.

that the capacitors are charged to constant voltages of V_C and V_o. To find these voltages, we apply the Kirchhoff's Voltage Law (KVL) to each topology. This leads to the following system of linear equations:

$$\begin{array}{r} 1 \\ 2 \\ 3 \end{array}\begin{cases} V_{in} - V_C = V_o \\ V_o = V_{R2} + V_{R3} \\ V_C = V_o \end{cases} \tag{2.1}$$

Note that if $R_1 + R_4 = R_2 + R_3$, the second topology gives $V_o = V_{in}/2$, but with high power loss. Thus, having dead time is necessary for correct SCC operation. Since real switches cannot turn on/off immediately, the current through the flying capacitor C in the steady state will look as shown in Fig. 2.4, where t_r is the rise time, t_{on} is the on-time and t_f is the fall time.

This current causes heating of the corresponding switches and other resistances where it flows. The total power dissipated by all the resistive elements is called conduction losses. These losses are modelled by an equivalent resistance, R_{eq}, as shown in Fig. 2.5a. The target voltage V_{TRG} is the no-load output voltage that is defined as $V_{TRG} = M \cdot V_{in}$, where M is the conversion ratio. Assuming that in Fig. 2.2a all the switches are identical, and $t_1 = t_2$, one can use the general expression from [12] to obtain equivalent resistance

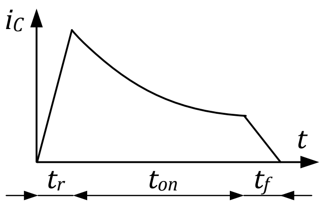

Figure 2.4: Current through the flying capacitor C (not in scale).

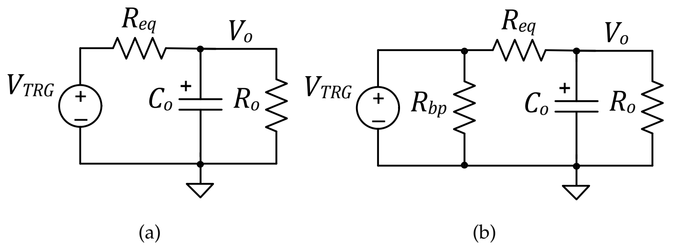

Figure 2.5: The equivalent circuits of a SCC representing only conduction losses (a) and both conduction and bottom-plate capacitance losses (b).

for our case:

$$R_{eq} = (\frac{R}{f})\frac{t_r + (1 - e^{-2\beta})RC + t_f e^{-2\beta}}{(t_r + 2(1 - e^{-\beta})RC + t_f e^{-2\beta})^2} \tag{2.2}$$

where $\beta = t_{on}/(RC)$, and $R = 2R_{sw}$. For ultra-fast switches we can assume that $t_r = t_f = 0$, then (2.2) is reduced to:

$$R_{eq} = \frac{1}{4fC}\frac{1 - e^{-2\beta}}{(1 - e^{-\beta})^2} = \frac{1}{4fC}coth(\frac{\beta}{2}) \tag{2.3}$$

where $\beta = (T - 2dt)/(RC)$. If dead time $dt = 0$ then, according to [13], the asymptotic limits of (2.3) are:

$$\lim_{\beta\to\infty} R_{eq} = \frac{1}{4fC}; \quad \lim_{\beta\to 0} R_{eq} = 2R_{sw} \tag{2.4}$$

In simpler words, the physical meaning of (2.4) is that when the capacitor has enough time to charge/discharge, the R_{eq} is defined by the conductance losses in the switches; whereas, when the phases are small, the capacitor charging losses start dominating in defining R_{eq}.

Although the conduction losses in case of integrated SCC constitute a significant part of all the losses, the model of Fig. 2.5a needs refining. For example, losses caused by recharge of bottom plate capacitances can be modelled by the parallel resistance R_{bp}, as shown in Fig. 2.5b.

The detailed methods of design of two-phase SCC are described in section 3.1.

2.1.2 Multi-phase SCC

One of the biggest disadvantages of the SCCs is their efficient operation only in a certain number of target voltages. These voltages are defined by the ratios of the SCC. The voltages beyond the target can also be generated, but only at the expense of the efficiency reduction. To avoid an essential efficiency reduction, the converters with a large number of ratios are used. The state of the art solution is the multi-phase SCC with binary resolution [14]. This type of converters can provide $2^N - 1$ ratios, where N is the number of flying capacitors. Fig. 2.6 shows the schematics of a simple step-down multi-phase converter with $N = 2$ flying capacitors.

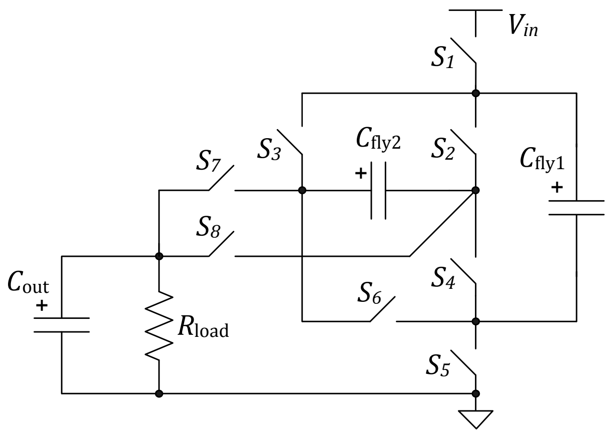

Figure 2.6: Schematic of the multi-phase SCC with two flying capacitors.

The converter has $4 \times N$ switches that allow setting special topologies of the flying capacitors to provide a certain conversion ratio. The task of controlling these switches is not trivial and can become quite complicated with higher N.

Requirements for the switches' controller are as follows:

- Oscillate among several predefined topologies for each specific ratio.
- Control the delay between phases (switching frequency).
- Provide a dead-time between phases to prevent the shoot-through currents.
- Prevent the errors during the ratio change (when the ratio is changed at the same time with an internal signal).

To support these requirements, the method of designing the controller must be more complicated than for the two-phase case. The method allows systematizing the process of the design for SCC control. A simple algorithm based on this method can be generated. The inputs to the algorithm will be the states of the switches in each topology, and its output – a self-timed circuit of the SCC controller. A self-timed controller simplifies the design of SCC because it does not need a clock signal, which sometimes can be complicated to deliver and adjust. Moreover, such a controller does not consume dynamic power in a standby mode, as there is no clock signal to oscillate. We propose the method of generating such controllers in section 3.2.

2.1.3 Self-oscillating SCC

Self-oscillating SCCs differ from the regular ones by the fact that the power switches are a part of the control circuit. The main advantage of the self-oscillating SCCs is their ability to operate with relatively high switching efficiency at very low output power. This advantage is provided by the simplification of the control circuit that helps reduce the leakage and the dynamic power consumption of the converter [15–17]. Self-oscillating SCCs can also be used to operate with larger loads. However, the advantage of this type of SCC becomes less perceptible with the increase of output power.

Let us consider the self-oscillating SCC shown in Fig. 2.7. It consists of two ring oscillators whose power supply buses are connected in series. Each pair of inverters in the top and bottom oscillators is connected to its flying capacitor C_{flyi}. The function of these capacitors is twofold: to transfer the charge, and to synchronize the top and bottom oscillators.

The switching frequency of SCC is determined by the leakage-based delay elements [18]. A leakage-based delay element is shown in Fig. 2.8. The principle of its operation is as follows: When the output of the power inverter (*inv*) changes from '0' to '1' or from '1' to '0', the gates of the transistors become disconnected from the voltage sources. The voltage values in these points are slowly changing because of the leakage in these transistors. We can control the duration of the delay by changing the resistance of the pass transistor (T_p) using the control voltage (V_{ctrl}), which can come from the feedback circuit.

From the environment perspective, SCC is a three-port circuit (two input ports and one output port). Their values are defined by the equation (2.5), where for the ideal case $V_{high} > V_{med} > V_{low}$.

$$V_{high} + V_{low} - 2V_{med} = 0 \tag{2.5}$$

If our inputs to the SCC are the voltage source (V_{in}) and ground (GND), the SCC can act as a voltage divider, voltage multiplier, or negative voltage generator, depending on which terminal is specified as an output (Fig. 2.9). Different SCC operation modes can be utilised for achieving different conversion ratios when composing several SCC cells [4], [15].

The detailed methods of design of self-oscillating SCC are described in section 3.3.

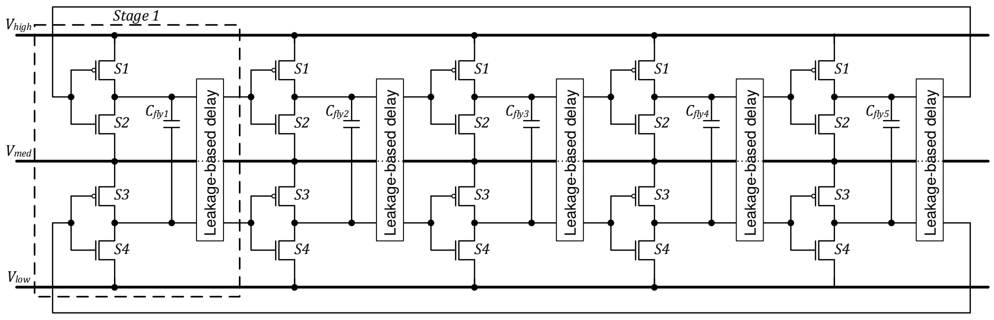

Figure 2.7: Structure of the self-oscillating SCC.

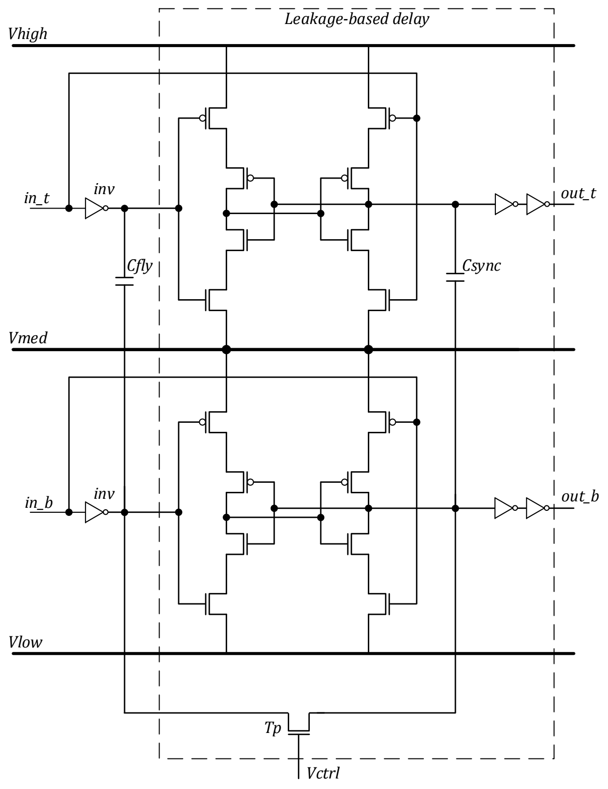

Figure 2.8: Leakage-based delay element in self-oscillating SCC.

2.1.4 Bottom plate capacitance and parasitic charge redistribution

In an on-chip implementation, the bottom plate of each C_{flyi} has a parasitic capacitance C_{bpi} to the substrate. This capacitance is charged by V_{med} in one switching state, and discharged to V_{low} in the other, as shown in Fig. 2.10. Therefore, the charge on the bottom plate capacitors is effectively wasted in the process of recharging.

The losses, associated with the parasitic bottom plate capacitance, bring the largest contribution to the entire losses of any SCC implemented on-chip.

According to [19], the limitation of the maximum efficiency of the SCC can be

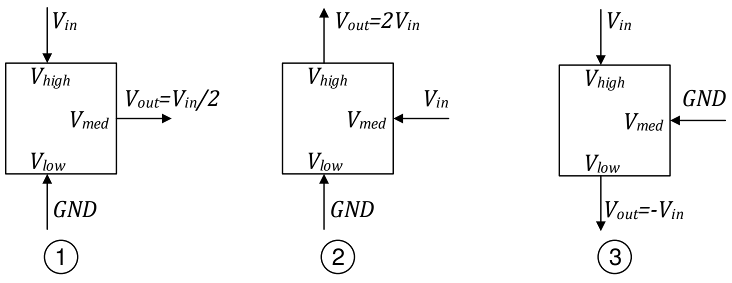

Figure 2.9: Operation of the SCC cell as a voltage divider (1), voltage multiplier (2), and negative voltage generator (3).

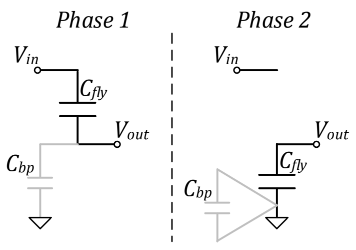

Figure 2.10: Charging and discharging phase of the bottom plate parasitic capacitor.

obtained using the following equation:

$$\eta_{max} = \frac{1}{1 + \sqrt{\alpha_{BP} K_{BP} K_c}} \tag{2.6}$$

where K_{BP} and K_c are topological constants defined by converters ratio; and α_{BP} is relative size of the parasitic bottom plate capacitance to the size of the flying capacitor itself.

The value of α_{BP} is defined by the technology used, as shown in Table 2.1. The most popular capacitor technologies in CMOS, namely metal-insulator-metal (MIM) and metal-oxide-metal (MOM) capacitors, have the same values of α_{BP} around 1.5%. Another widespread technology, metal-oxide-semiconductor (MOS) capacitors, is not recommended for designing the flying capacitors for SCCs because of the large α_{BP} value

Table 2.1: Bottom plate capacitance for different capacitor technologies.

Capacitor technology	**Bottom plate capacitance** $(\mathbf{ff_{BP}})$
Metal-insulator-metal (MIM)	1.5%
Metal-oxide-metal (MOM)	1.5%
Metal-oxide-semiconductor (MOS)	7%
Deep trench	1.5%
Ferroelectric	$< 0.1\%$

of around 7%. The technology of deep trench capacitors looks promising for SCC design mostly because of the possibility to implement larger capacitance values on the same area comparing to other technologies. However, deep trench capacitors can be implemented only using FD-SOI technology that is relatively new, and not very popular. The lowest values of α_{BP} can be observed in ferroelectric capacitors. This technology, however, is considered "exotic" and too expensive for the SCC design.

The charge loss at the bottom plate parasitic capacitance causes significant efficiency reduction. There are certain methods of reducing charge losses, e.g. via parasitic charge redistribution [20]. According to this method, two parallel SCCs are operating in a counter phase. During the switching between the phases the bottom plates of both SCCs are connected through the pass transistor (Fig. 2.11). Thereby, half of the charge of the charged capacitor is passed to the discharged one and is not wasted to the ground. Therefore, the losses associated with the parasitic capacitance are halved. The application of this method is particularly described in section 3.3.

Another approach to reducing charge losses is by scalable parasitic charge redistribution [19]. Here the number of SCCs and the number of operating phases are increased and the process of charge redistribution is conducted in several stages. This method allows reducing the parasitic losses by several times at the expense of more complicated control circuitry.

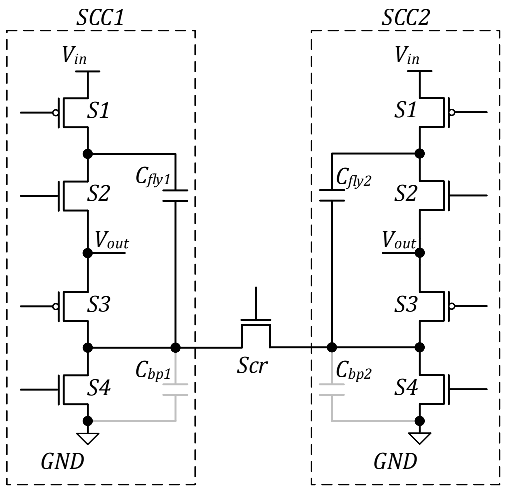

Figure 2.11: The method of the parasitic charge redistribution, applied to the regular SCC structure.

2.2 Principles of asynchronous circuits design

Asynchronous circuits are event-driven, i.e. they react to changes in a system at the rate they occur [21,22]. This makes them particularly useful for on-chip power management, where the ability to quickly respond to dynamically changing loads across the chip is essential for reliable operation and efficiency [23].

Asynchronous design is a highly developed field with a variety of different design methods and techniques [24, 25]. One of the most popular ways of specifying the asynchronous control circuits is using the Signal Transition Graphs (STGs). The STGs are compatible with multiple synthesis tools, such as PETRIFY [26], MPSAT [27], ATACS [28]. These tools take an STG specification of a complete controller and produce a speed-independent circuit implementation [29].

The STGs are a kind of Petri nets [30] in which transitions are labelled with the rising and falling edges of circuit signals [31, 32]. Graphically, the places are represented as circles, transitions as text labels, consuming and producing arcs are shown by arrows, and tokens are depicted by dots. For simplicity, the places with one incoming and one outgoing arc are often hidden, allowing arcs (with implicit places) between pairs of transitions.

STGs can be used to model the environment that a circuit reacts to, the input signals, the intermediate signal changes within the circuit, internal signals, and the output signals which are the reaction of the circuit to its environment. Conventionally, input, output and internal signals are identified by their colour – red, blue and green, respectively. Each signal can transition either high, indicated by the $'+'$ suffix, or low, indicated by the $'-'$ suffix [33].

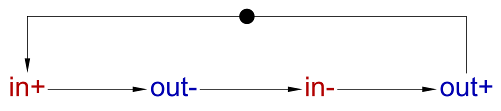

Figure 2.12: A Signal Transition Graph describing the behaviour of inverter.

Figure 2.12 shows an example of STG describing the behaviour of a simple inverter. The arcs between the signals represent the sequence of events that happen in the circuit. The dynamic behaviour of STG is defined as a token game, changing marking according to the enabling and firing rules described in [30].

The STG specifications can be verified and validated in WORKCRAFT toolsuite [34–36]. It provides a convenient framework for capturing STG specifications, their formal verification [37], automatically resolve Complete State Coding (CSC) conflicts [38], and logic synthesis of speed-independent circuits [26,39].

2.2.1 Self-timed buck controller design

In this section, we consider an example of asynchronous circuits design flow using the STGs. This example is the design of the self-timed controller for the buck DC-DC converter proposed in [9]. This design is used to demonstrate the successful application of the methods of asynchronous design for power electronics. The methods presented in this section will be applied to another type of power regulators - SCC.

The structure of the converter is shown in Fig. 2.13. The controller switches the power regulating PMOS and NMOS transistors ON and OFF as a reaction to under-

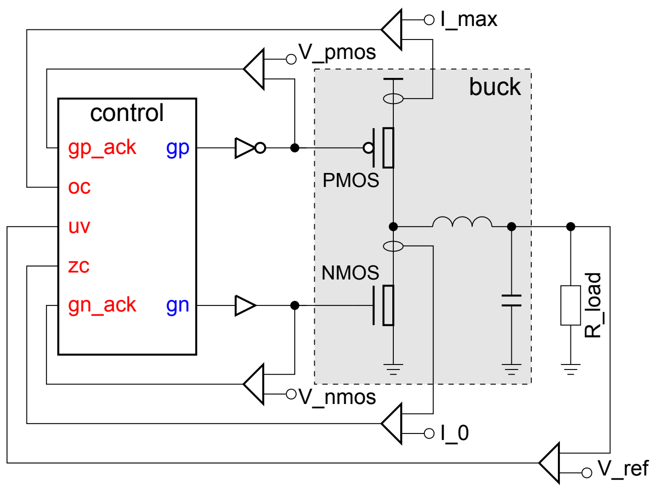

Figure 2.13: Structure of the buck converter.

voltage (UV), over-current (OC) and zero-crossing (ZC) conditions. These conditions are detected and signalled by a set of specialised sensors implemented as comparators of the measured current and voltage levels against some reference values (V_ref, I_max and I_0 respectively). The gp and gn signals are buffered to drive the very large power regulating transistors (occupy more than 50% of the buck area) and their effect on the buck can be significantly delayed. Therefore, the controller is explicitly notified (by the gp_ack and gn_ack signals) when the power transistor threshold levels (Th_pmos and Th_nmos) are crossed.

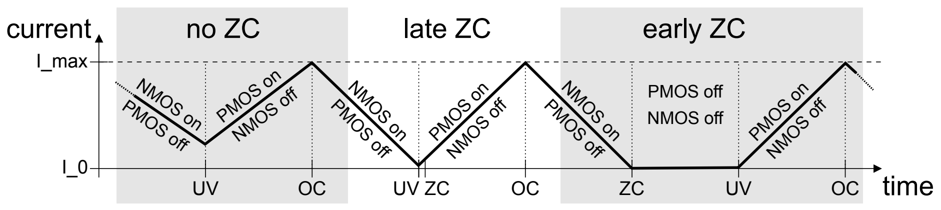

Figure 2.14: Scenarios of operation of the buck converter.

This specification in Fig. 2.14 shows the alternation of the UV and OC conditions which are handled by switching the power regulating PMOS and NMOS transistors of the buck ON and OFF. Detection of the ZC condition after UV does not change this behaviour, however, if ZC is detected before UV then both the PMOS and NMOS transistors remain OFF until the UV event.

According to the phase diagram there are three distinctive scenarios:

- **no ZC** – UV happens without ZC;
- **late ZC** – UV is followed by ZC;
- **early ZC** – UV happens after ZC.

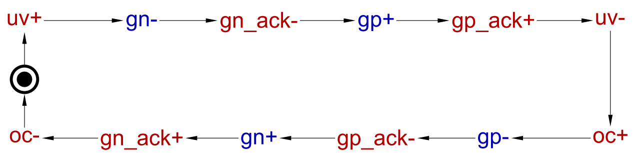

Figure 2.15: STG for the first scenario with no ZC.

Initially, the NMOS transistor is ON and the PMOS transistor is OFF which should lead to the UV condition. When UV is detected the NMOS transistor needs to be switched OFF. When the OFF state of NMOS is confirmed the PMOS transistor can be switched ON to charge the buck. Eventually, the buck will saturate leading to reset of UV and OC conditions. At this stage, the PMOS transistor needs to be switched OFF. After the OFF state of the PMOS transistor is confirmed the NMOS transistor is switched ON. This leads to the release of OC and brings the controller to the initial state. The behaviour of the controller for this scenario is described with an STG in Fig. 2.15.

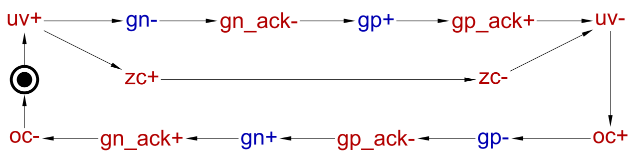

Figure 2.16: STG for the second scenario with late ZC.

The scenario for late ZC (Fig. 2.16) is formalised in a very similar way. Both phases of ZC just happen concurrently with switching NMOS transistor OFF and PMOS transistor ON.

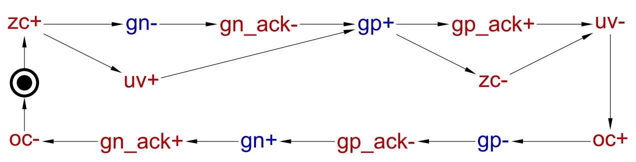

Figure 2.17: STG for the third scenario with early ZC.

The scenario for early arrival of ZC (Fig. 2.17) is a bit different. Here the NMOS transistor needs to be switched OFF as soon as ZC is detected, without waiting for UV. However, switching the PMOS transistor ON is still delayed till UV condition.

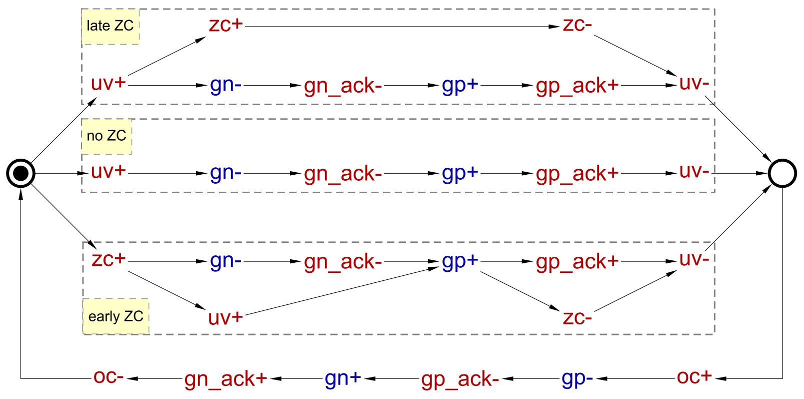

Figure 2.18: Combined STG of the buck controller.

The STG in Fig. 2.18 combines all the three scenarios mentioned above. Furthermore, all three scenarios had the same parts in the STG. Thus, these parts were simplified to avoid duplication.

This STG has been transformed into a circuit using the Petrify tool. The resulted circuit is shown in Fig. 2.19. The circuit satisfies the requirements to the buck controller, and its properties have been validated using the Workcraft environment [36].

The example of the buck controller design demonstrates the simplicity of the method of asynchronous design. The STG description is intuitively clear, and the tools embedded in Workcraft environment allow us not only to synthesize the circuit but also to verify and validate it.

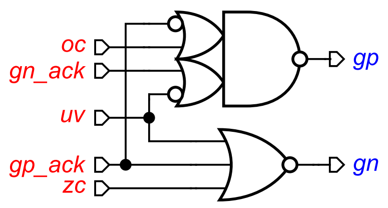

Figure 2.19: Self-timed circuit of the buck controller.

2.3 AI hardware design for energy efficiency

Advances in sensing devices are causing a shift towards the fourth industrial revolution [1]. The large volumes of the data produced by these devices are enabling a new generation of Artificial Intelligence (AI) systems at the micro-edge that are designed to infer important decisions in the real world [40]. A promising direction of these AI Systems is the leap towards perpetual computability, allowing always available local AI service. To enable this, designers of pervasive AI system are facing two grand challenges: *energy efficiency* and *energy autonomy* [41–44].

Energy efficiency refers to economising the energy consumption of elementary compute operations. The aim is to prolong operating lifetime with a given energy budget, typically defined by the batteries. Reducing energy requires careful design considerations at device-, circuit- and system levels. Examples include reducing device geometry [45], scaling operating voltage [46] and designing circuits with reduced or approximate logic [47].

New generations of pervasive AI-based systems require maintenance-free long-life. As such traditional energy-efficient design principles applied in battery-operated systems are not feasible, as they need periodic re-charging and replacements. Portable energy harvesters, which produce electrical energy to supply to computation loads by scavenging energy from the environment, are gradually making inroads. Such a scheme of energy harvesting can remove the need for maintenance in favour of energy autonomy. However, mitigating their energy variations needs computational capability over a dynamic power envelope, otherwise known as power elasticity [48, 49].

Despite advances in low-power design methodologies, the energy footprint of existing AI systems, such as NNs, has generally remained high [50]. Our persistence in using arithmetic-heavy circuits with growing algorithmic complexities is a major contributor to this. For instance, object detection using deep NNs may require a hundred to over ten thousand times the energy needed by the traditional histogram of oriented gradient techniques [51]. Due to such poor efficiency, the widespread adoption of energy-autonomous AI hardware at the micro-edge has proven challenging [52].

To appreciate the importance of efficient AI hardware design, we show the example of a perceptron, whose idea originates from Rosenblatt's work of 1958 [53]. It is a basic building block of NNs used in AI applications [54–56]. It consists of an input vector, a set of weights and a bias to produce binary classification outcomes, as follows:

$$f(x) = \begin{cases} 1, & \text{if } \mathbf{w}.\mathbf{x} + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

where w is a vector of real-valued weights, $\mathbf{w}.\mathbf{x}$ is the dot product $\sum_{i=1}^{m} w_i x_i$ with m number of inputs, and b is the bias. The process of deciding the appropriate weights ($\mathbf{w}$), often also known as training, serves as the basic principle of supervised learning. When m becomes large, it approximates the behaviour of a biological neuron.

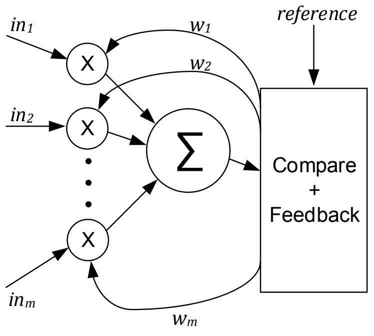

Figure 2.20: Structural organisation of a perceptron, which is the basic building block of NNs.

Figure 2.20 shows the typical structure of a perceptron [57, 58]. At its core is an adder that sums m weighted inputs. The result of the addition is compared with a reference during the training phase, during which the weights are updated to ensure the reference is matched. For hardware implementation, multiplication and addition are crucial arithmetic circuits in a perceptron [59]. Such arithmetic operations require significant area and power costs, which depend on the number of input-weight pairs, the precision of the multipliers/adders, their underlying technology nodes and algorithmic complexities.

Over the years, substantial research has been dedicated to improving the energy efficiency of AI hardware [60]. A vast body of this research has predominantly remained within the remits of Landauer's logic boundaries for energy or power reduction [61]. Reducing threshold voltage that defines the logic boundaries and designing new low-complexity architectures are key to achieving this. Andri *et al.* [52] proposed a NN architecture that showed how high-performance NN operations can be achieved by parallel logic blocks. These blocks are designed using low-threshold technology nodes that are faster and ultra-low power. Prado *et al.* [62] showed a logic approximation method applied in parallel NNs. Due to low-complexity architecture the individual components are faster and more energy-efficient. Among others, Qiqieh *et al.* [47] proposed logic compression approaches for reducing power consumption, area and critical path delay of NNs. By combining the circuit-level approaches with online system-wide techniques, significant energy reduction was reported.

However, reducing power or energy alone using the above principles, does not solve the problem of energy-autonomous pervasive AI systems [48]. These systems will need to be able to not only work with limited power supplies but also survive extreme variations as power regulation and energy storage options are limited and expensive in low-end micro-edge devices [48, 63]. Indeed, these systems will need to be built with natural power elasticity to operate over a large power domain [48, 63, 64].

Existing perceptron designs are predominantly digital, although a number of analogue implementations have been reported [65, 66]. The digital designs can operate over a range of powers defined by paired voltages (V_{dd}) and frequencies (f). These designs are however vulnerable to dynamic power supply variations, for example, conditions where the power source voltage changes in time and continuous V_{dd} and f pairing can prove expensive under limited energy budgets. As such, existing designs have poor power elasticity that prevents them from providing useful computation under unreliable or unstable power supply conditions.

In chapter 4, we propose a novel perceptron design, which is supposed to have a higher degree of power elasticity.

Chapter 3

Design of efficient SCCs

In this chapter, we improve the power management part of the IoT devices by proposing novel methods for the SCC design.

The section 3.1 describes a method for the control design for two-phase SCCs. In this section we formally specify the intended behaviour of SCC controller using STGs and then synthesize a speed-independent controller implementation.

In the section 3.2, we expand this method for the case of multi-phase SCC. In this case, the controller operates differently when choosing different conversion ratios. The design method for such a controller becomes more sophisticated.

In the section 3.3 we improve the design of self-oscillating SCC demonstrated in section 2.1.3. Our improvement resolves the issue of bottom plate parasitic capacitance discussed in section 2.1.4.

In section 3.4, we simulate the power regulation part designed in this chapter. The controller and converter circuits are designed using the $ams350nm$ technology with nominal supply voltage $3.3V$. The converter part requires large power transistors in order to support a large load and to reduce the conduction losses. The simulations start with a demonstration of the switch used in SCC design. After that, we simulate the controllers for two-phase and multi-phase SCCs and show their operation together with the converter part. The section ends with the simulation of the self-oscillating SCC, which was compared to the state-of-the-art solution. The self-oscillating SCC is designed for applications with lower power and does not require large power transistors. By this reason. it was designed using the *umc65nm* technology.

3.1 Two-phase SCC control

In order to specify the behaviour of an SCC controller formally we use STGs [67]. Subsequently, this will enable us to synthesize its speed-independent circuit implementation automatically using the WORKCRAFT environment [35].

The designed controller has the following interface: it provides two output signals – *ph*1 and *ph*2 that control two pairs of switches S_1, S_3 and S_2, S_4 respectively; it also requires two delay elements, for dead time and for the switching phase. Both these delays are external for the controller. The dead time delay is inserted between *dt_delay_req/ack*, and the phase delay – between *ph_delay_req/ack*.

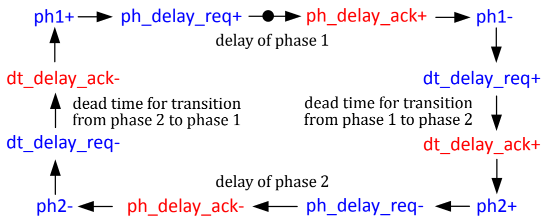

Figure 3.1: Signal transition graph of the two-phase SCC controller.

An STG in Fig. 3.1 specifies the behavior of the two-phase SCC controller. Initially, phase 1 is active and waits when the phase delay is elapsed. Upon receiving *ph_delay_ack*, the output *ph*1 goes down (to turn S_1 and S_3 off) and runs the dead time delay by rising *dt_delay_req*. As soon as *dt_delay_ack* goes up, phase 2 is activated: *ph*2 signal is set high (to turn S_2 and S_4 on) and the phase delay is triggered. The phase 2 completes after the phase delay finishes and the signal *ph_delay_ack* goes down. The end of the phase 2 (triggered by the signal *ph*2 going low) starts the dead time delay by switching the *dt_delay_req* to $'0'$. As soon as *dt_delay_ack* goes down, we start the phase 1 (rising the signal *ph*1) and return to the initial state.

The STG of Fig. 3.1 is built and translated into the asynchronous circuit using

WORKCRAFT toolsuite [36]. The obtained circuit is shown in Fig. 3.2 with additional elements of reset. WORKCRAFT also allows us to verify that the circuit is speed-independent, i.e. operates correctly regardless of the values of delays of any circuit component. The only assumption is about two inverters with dotted lines: the delays of their output wires must be negligible. This is usually achieved by placing the inverters close to the main gates during the place-and-route stage.

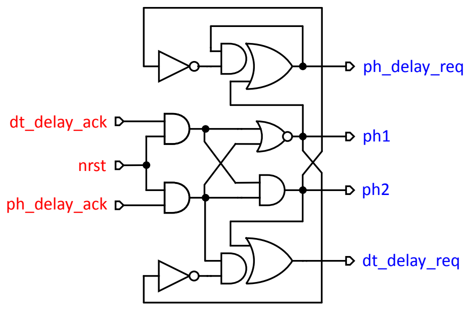

Figure 3.2: Synthesized circuit of the two-phase SCC controller.

3.2 Multi-phase SCC control

Table 3.1: Specification of the topologies.

Ratio	Phase	S1	S2	S3	S4	S5	S6	S7	S8
$\frac{1}{4}$	1	1	0	0	0	0	1	0	1
	2	0	0	1	0	1	0	0	1
	3	0	0	0	1	1	0	1	0
$\frac{2}{4}$	1	1	0	1	1	0	0	0	1
	2	0	0	1	1	1	0	1	0
$\frac{3}{4}$	1	0	1	0	0	1	0	1	0
	2	1	0	0	1	0	0	1	0
	3	1	0	1	0	0	0	0	1

Table 3.1 shows all the topologies for different ratios of the multi-phase SCC in Fig. 2.6. The table, and the methods of obtaining it are described in [13]. The topologies are alternating in 3 phases for the ratios $1/4$ and $3/4$, and in 2 phases for the ratio $2/4$. The value $'1'$ in the table indicates that the corresponding switch is turned on, and the value $'0'$ – turned off.

The first step of the controller design is defining STG specification for each ratio, obeying the following rules:

- In the initial state, all the switches are turned off (switch control signals are $'0'$) and all the signals between the controller and the delay elements (delay requests and acknowledgements) are equal to $'1'$.
- The switching sequence of an i delay element must be following: $dir-$, $dia-$, $dir+$, $dia+$.
- The switching starts with resetting the first delay request (signal $d1r$), and ends with setting the last delay acknowledgement (signal $d3a$ in the example case). For the ratios with the smaller number of phases the delays before the last are skipped. In the example, for the ratio $1/2$ the delay 1 is followed by the delay 3, and the delay 2 is skipped.
- The switches of a certain topology must be turned on after the reset of the delay requests and turned off after the set of the delay request.
- The switches, which remain on in several topologies in a row, do not turn off between these topologies.

The STG for the ratio $1/4$ is shown in Fig. 3.3. The arcs from delay requests to delay acknowledgements in this STG can be removed because of the transitivity property, however, they do not affect the STG, and are left for better visual representation. The timing diagrams depicted in Fig. 3.4 demonstrate the causality links between the signals in the STG.

The next step is to design an STG of the part of the controller that chooses the ratio and communicates with an environment. This STG for the example SCC is shown in Fig. 3.5. The initial state is marked by places with tokens. The first transition is sending the ratio

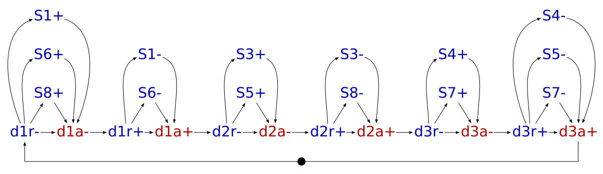

Figure 3.3: Formal STG specification for 1/4 ratio control.

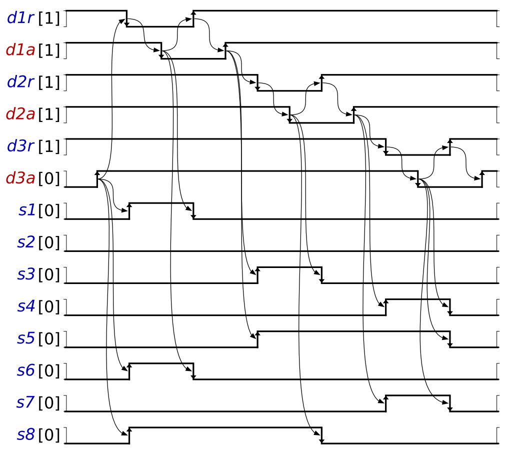

Figure 3.4: Informal timing diagram for 1/4 ratio switching.

request (switching the signal *ratio_r* to ′1′). After that, the environment is allowed to send the ratio to the controller. When the ratio arrives, the controller starts the first phase of a chosen ratio by resetting the first delay request, as it was described above. At the same time, it allows resetting the ratio signal from the environment. When the acknowledge signal of the last phase arrives, the system turns all the switches off and sends the new ratio request.

The STG contains the arcs that lead to the input signals. The controller cannot delay or prevent its inputs because they are produced by the external environment. However, we can assume the following behaviour of the environment:

- The environment sends only one ratio signal, and only when it is requested. This ratio signal resets right after the request is reset.
- The ratio signal resets before the delay acknowledgment arrives. This assumption

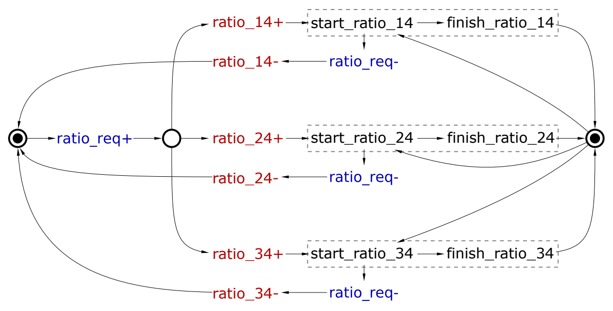

Figure 3.5: STG specification of multi-phase SCC controller.

means that delay provided by the delay element takes more time than the communication between the controller and the environment. This assumption is not necessary, however, the circuit can become more complicated without it.

- The delay acknowledgement signals arrive later than the switch control signals change their value.

The delay element provides two delays: the phase delay for the transition from $'1'$ to $'0'$, and the dead time delay for the transition from $'0'$ to $'1'$. One of the possible implementations of such delay could be connecting the outputs of both delay elements to an *OR* gate. Because the phase delay is always greater than the dead time delay, the duration of transition from $'1'$ to $'0'$ would be equal to the phase delay, and from $'0'$ to $'1'$ - to the dead time delay.

Since the dead time delay is short, it can be implemented as a chain of inverters. In step-down SCCs, where the power stage is built using transmission gates, Vdd of the controller can be used as a bias for the substrates of power p-MOS transistors. In this case, change in Vdd will cause a change in the delay of switches and in the delay of inverters. These changes will occur in the same direction that should contribute to avoiding the shoot-through currents. The phase delay is much longer than the dead time delay and requires more sophisticated solutions, e.g. using a leakage-based delay element [15]. Other techniques to realize long delays can be found in [68, 69]. However, this element must specify the following requirement: its delay of $'1'$ must be not larger than the delay of $'0'$. Otherwise, it may not set to $'1'$, when the new $'0'$ arrives.

3.3 Self-oscillating SCCs with parasitic charge redistribution

Following the idea of the parasitic charge redistribution proposed in [20], one could design two self-oscillating SCCs that share their parasitic charge. However, such design would require a complicated control, that compromises our goal of designing low complexity self-oscillating SCC.

In our approach, instead, we add an extra capacitor C_{store}, which stores the parasitic charge. The bottom plates of each C_{flyi} is connected to this C_{store} through its switch SW_{ch}. When a stage is changing its phase, the corresponding SW_{ch} is conducting. Thereby, when C_{flyi} is switching from the charging phase to the discharging phase, a part of the parasitic charge goes to the store capacitor and thus is not wasted. Similarly, when C_{flyi} switches back to the charging phase, the C_{bpi} is first partially charged from C_{store}, and therefore requires less energy for being fully charged.

The structure of the designed converter is shown in Fig. 3.6. In contrast to the regular self-oscillating SCC, the inverters are replaced by *pmos* and *nmos* separately controlled by the logic blocks since in the process of charge redistribution both of the transistors must be turned off. The operation of the logic blocks is described by the following steps:

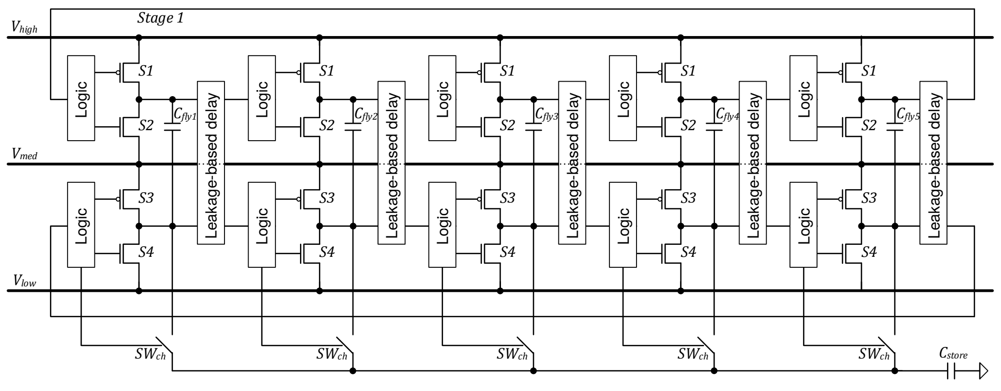

Figure 3.6: Structure of the self-oscillating SCC with charge redistribution.

The operation of the logic block can be described by the following steps:

- Both top (*Logic_t*) and bottom (*Logic_b*) logic blocks have the same structure. The difference is that they have different voltage levels, and the bottom block of the i stage controls the S_{ci}.

- Initially the output of the leakage-based delay element (input of the logic block) is $'0'$. The outputs of the logic block that control the transistors are also $'0'$. It means that the *pmos* of the inverter is on and the *nmos* and SW_{ch} are off, and C_{flyi} is in a charging phase.
- When the input of the logic block changes to $'1'$, the *pmos* output also changes to $'1'$. The *pmos* now switches off, and the output of the inverter is $'Z'$. At this moment SW_{ch} is turned on and C_{bpi} discharges to C_{store}.
- After the C_{bpi} finishes discharging, the SW_{ch} is turned off, and the *nmos* output is switched to $'1'$. The capacitor switches to the discharging phase.
- After the signal propagates through all the inverters of the chain, the input of the logic block changes back to $'0'$. When it happens, the *nmos* output switches to $'0'$, and the inverter switches to the $'Z'$ state again. At this state the SW_{ch} is turned on, and the parasitic capacitor charges from the store.
- After the C_{bpi} finishes charging, the SW_{ch} is turned off, and the *pmos* output is switched to $'0'$. The flying capacitor switches to the charging phase.

The logic blocks are implemented as shown in Fig. 3.7. The power losses associated with this circuit are much lower than the power saved by the charge redistribution process. The delay element must be specified according to the size of the bottom plate capacitor, and the resistance of the switch SW_{ch}. This delay must be as low as possible because, during this delay, the capacitor is not operating which causes additional losses. On the other hand, this delay should be large enough to allow each C_{bp} to finish its charge/discharge process. Normally, this delay is in the range of $5 - 10ns$.

The inverters at the outputs of the logic block are used as drivers for the power transistors. Their strength should be chosen accordingly to the size of the transistors they drive.

Although the method in [20] uses a simple *nmos* as a charge recycling switch, in this design it should be implemented as a transmission gate to allow the switch having low

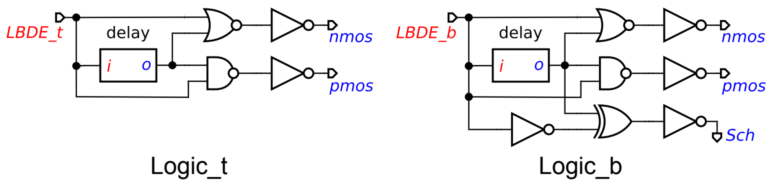

Figure 3.7: Implementation of the top and bottom logic blocks.

resistance in both directions. When we start transmitting the charge from C_{bp} to C_{store}, the V_{gs} of the *nmos* equals to 0, and it has very high resistance.

3.4 SCC simulation

3.4.1 Switch design and simulation

Since the currents through the switches in the SCC do not change direction, we can use unidirectional switches such as diodes and transistors. However, in general case, the so-called four-quadrant switches are used. This is bidirectional switches that can be realized in step-down SCCs as transmission gates. The resistance of transmission gate can be made very small, while to realize large capacitance on-chip we need large area or/and non-standard technology process. Thus, to have low R_{eq} the SCC should operate with high switching frequency.

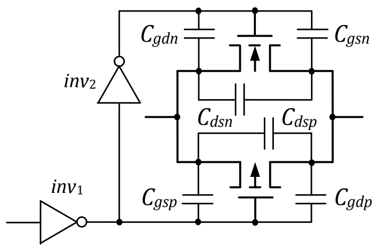

Figure 3.8: Circuit used as a bidirectional switch.

However, here we face the problem of parasitic capacitances of the MOS transistors, known as the charge injection (clock feedthrough) effect. It is expressed in that the current through the transistor in the moments of turning on/off has spikes. Parasitic bottom-plate capacitances of the flying capacitors will increase these spikes even more. The transmission gate should have constant resistance in a wide range of the input voltages. For this, the PMOS transistor is made wider than the NMOS transistor. Since the mobility of electrons is approximately 3 times greater than that of holes, minimal width ratio is 3. There are several methods to reduce the charge injection [70, 71]. However, these methods over-complicates the transmission gate circuit, providing a not very high advantage in avoiding the charge injection. We use the method, where PMOS transistor is turned on/off first by a power inverter [72]. The corresponding circuit of the bidirectional switch is shown in Fig. 3.8. The width of the transistors in this circuit, including the inverters, is given in table 3.2.

Table 3.2: Width of the transistors relatively to the power NMOS.

Element	Transistor width	
	NMOS	PMOS
inv_1	$0.1 \cdot nWidth$	$0.3 \cdot nWidth$
inv_2	$0.04 \cdot nWidth$	$0.12 \cdot nWidth$
switch	$nWidth$	$3 \cdot nWidth$

Such transistor size has been chosen to support the fast switching time. In the case of the small power switches, the transistors can be smaller than the technology limits. In this case, the smallest technology transistors should be used. In the case of the large power switch, the inverter inv_1 may require an extra buffer to drive. The transistors of this buffer should be also 8-10 times smaller.

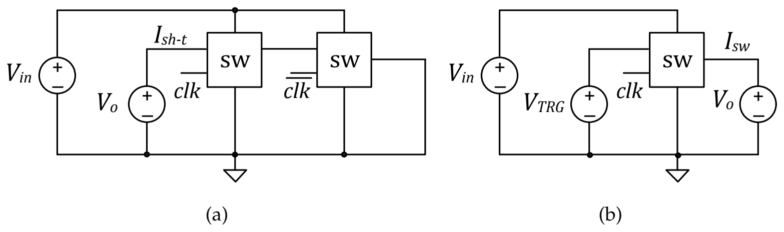

Figure 3.9: Examination circuit for the switch to measure the switching losses (a) and the R_{on} (b).

The parameters of the switch are examined in the two circuits presented in Fig. 3.9.

The circuit in Fig. 3.9a measures the losses caused by the shoot-through currents. Two switches are connected serially between $V_o = V_{in}/2$ and ground. The switches are operating in a counter-phase. The energy losses in these switches are representing the losses of the switches S_2 and S_3 of the two-phase SCC in Fig. 2.1. The losses for the different values of $nWidth$ are demonstrated in Fig. 3.10.

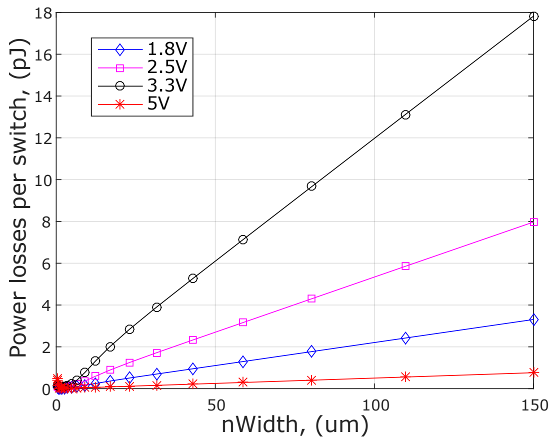

Figure 3.10: Energy losses per one switch caused by the shoot-through currents.

The circuit in Fig. 3.9b measures the ON-resistance R_{on} of the switch. Fig. 3.11 shows the currents through the transistors and the total current, I_{sw}, for the switch with $nWidth = 150\mu m$, and the following parameters: $V_{in} = V_{dd} = 3.3V$, $V_{TRG} = V_{in}/2 = 1.65V$, and $V_o = 1.6V$. For the level of $I_{sw} = 2.7mA$ the switch resistance $R_{sw} = ((V_{TRG} - V_o))/I_{sw} = 18\Omega$.

3.4.2 Two-phase SCC simulation

Based on Fig. 3.11, we set the dead time $dt = 0.5ns$. The output *dt_delay_req* of the controller is connected to *dt_delay_ack*, such that the dead time is provided only by delays of the corresponding logic gates. Fig. 3.12 shows the output pulses of the controller ph1 and ph2. To realize the phase delay, we used the element ”delay” of the simulator.

Since the obtained $R_{sw} = 18\Omega$ is low, the SCC can provide a relatively high output current. This means that its efficiency will be high even with heavy enough load. If the

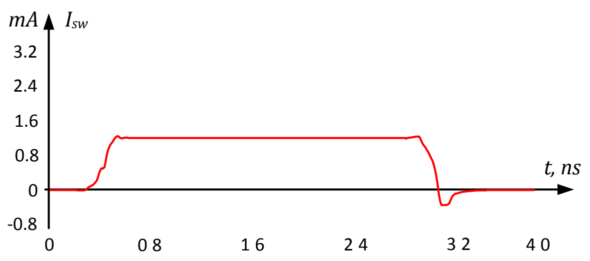

Figure 3.11: Current through the switch transistors for f_{clk}=200MHz. Solid line: NMOS, dashed line: PMOS, dotted line: total current.

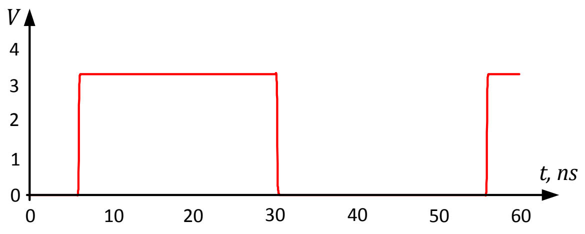

Figure 3.12: Output pulses of the SCC controller for f=20MHz and dt=0.5ns. Solid line: ph1, dashed line: ph2.

losses due to parasitic capacitances are small, the efficiency can found as:

$$\eta = \frac{V_o}{V_{TRG}} = \frac{R_L}{R_L + R_{eq}} \tag{3.1}$$

The best efficiency is obtained in the case when β in 2.3 tends to 0. This means that the current through the flying capacitor C is a rectangle. Such a form of the current is impossible in practice due to parasitic elements. For optimal $f = 20MHz$ the current through the capacitor $C = 1.5nF$ is shown in Fig. 3.13.

Since the switches S_1 and S_2 are connected to V_{in} and gnd and have not been optimized to work at these conditions, they demonstrate the highest spikes as shown in Fig. 3.14. The input and output currents also have spikes and are shown in Fig. 3.15.

The equation 3.1 represents the efficiency only in the case of the small parasitic capacitances. In reality, the efficiency of the on-chip devices is significantly affected by these parasitics. For this paper, we use the metal-insulator-metal (MIM) capacitors that have the bottom plate capacitance $\alpha_{bp} = 1.5\%$ and the top plate capacitance

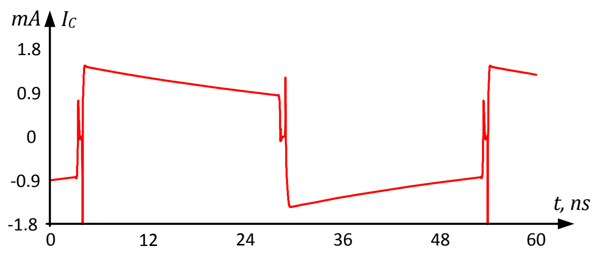

Figure 3.13: Steady-state current through C for f=20MHz and dt=0.5ns.

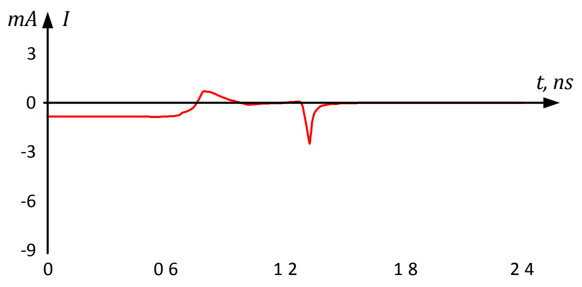

Figure 3.14: Worst case of the current spikes during the transitions. Solid line: S_1, dashed line: S_2.

$\alpha_{tp} = 1.5\%$ [19,73]. During every switching phase, these parasitic capacitors are charged and discharged with $\delta V = V_{in}/2$. Thus, the losses associated with them increase with the higher switching frequencies. Although, the losses, associated with the resistance of the switches and the flying capacitor, are resulting in the output voltage drop; the losses, caused by the parasitic capacitances, increase the input current. To estimate the SCC efficiency we divide the output power by the input. It results in the following equation:

$$\eta = \frac{V_o^2}{R_L \cdot V_{in} \cdot I_{in}} \tag{3.2}$$

The converters with the large switches and capacitors have smaller R_{eq} and are able to support the larger loads. At the same time, they will have larger parasitic capacitances, that will lead to the extra unnecessary losses when operating with smaller loads. To demonstrate this, the following SCCs have been simulated:

- Large SCC with $C_{fly} = 1.5nF$ and switch $nWidth = 150\mu m$
- Medium SCC with $C_{fly} = 300pF$ and switch $nWidth = 20\mu m$

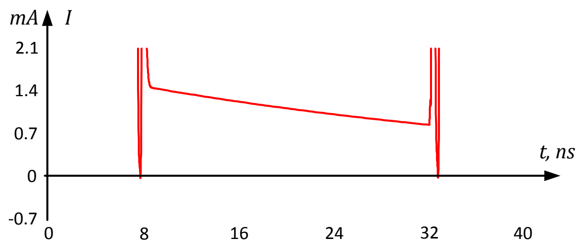

Figure 3.15: Steady-state currents of the SCC for f=20MHz and dt=0.5ns. Solid line: input current, dashed line: output current.

- Small SCC with $C_{fly} = 60pF$ and switch $nWidth = 1.2\mu m$

The examination circuit (Fig. 3.9b) was used to measure the R_{on}. For the large switch it is 18Ω, for the medium - 136Ω, for the small - 2371Ω. Each SCC was supporting three loads: large $R_L = 1K\Omega$, medium $R_L = 20K\Omega$, and small $R_L = 400K\Omega$. The resulting efficiency of the converters for $V_{in} = 3.3V$ is shown in Fig. 3.16

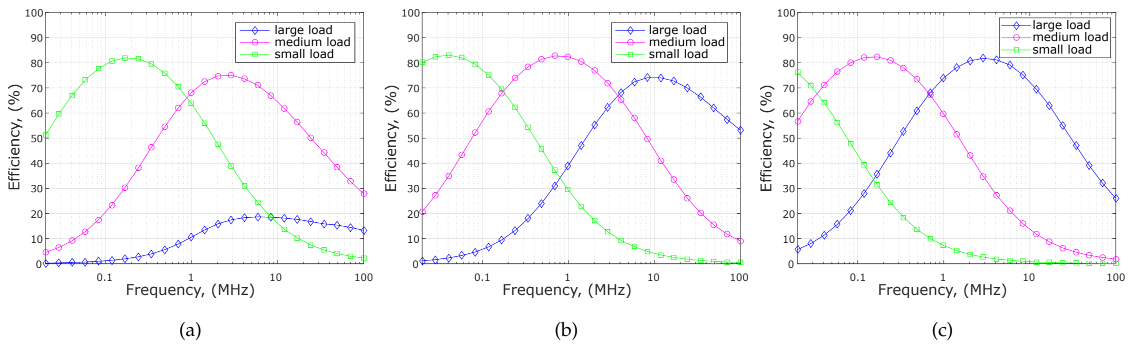

Figure 3.16: Efficiency of small (a), medium (b), and large (c) two-phase SCC with different loads.

The plots show that every SCC has its peak efficiency, which depends on the following parameters: switching frequency, SCC size, and load size. The small SCC shows better efficiency with the small load, however, with the large load, its efficiency drops significantly.

Another interesting observation is that the larger SCC requires lower switching frequency to perform with the same efficiency.

3.4.3 Multi-phase SCC simulation

To demonstrate the operation of the multi-phase SCC, it has been set in series to the ratios $1/4, 2/4, 3/4$ and $1/4$. The expected values of V_{out} are $0.825V$, $1.65V$, $2.475V$ and $0.825V$ respectively.

The results of the simulation are shown in Fig. 3.17. When the requested ratio was set to $1/4$, the output voltage became $0.82V$. Then, after changing the ratio to $2/4$, the voltage changed to $1.64V$. When the ratio was set to $3/4$, the voltage became $2.46V$. Afterwards, when the ratio was set back to $1/4$, the voltage returned to $0.82V$. A high degree of inertia of the output voltage is due to the relatively high output capacitance and large load resistance.

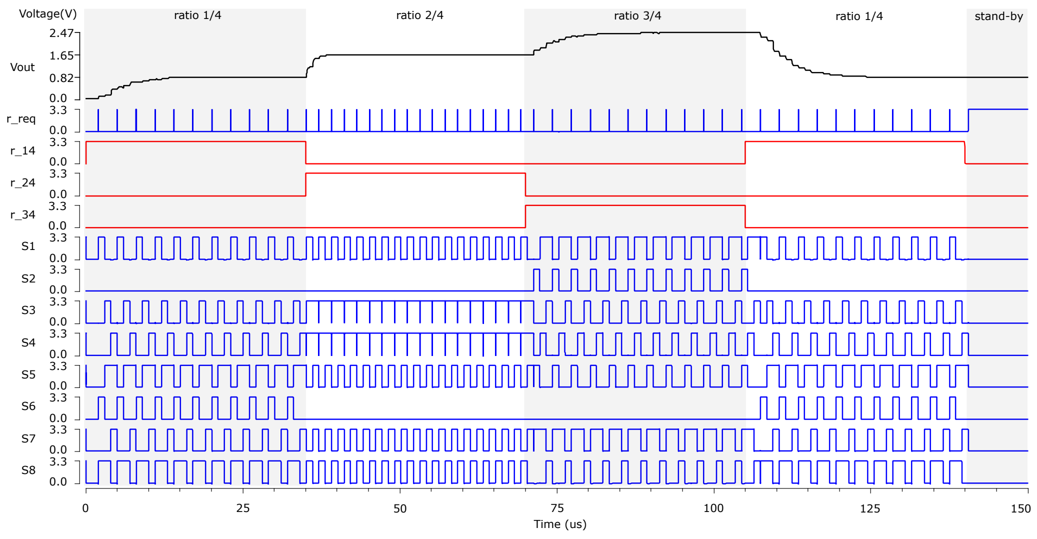

Figure 3.17: Inputs/outputs of the multi-phase SCC controller.

The signal *ratio_req* was periodically sent to the mode control block, but most of the time it was $'0'$. At the end of the simulation, when there were no ratio signals, it has remained in $'1'$, waiting for the next ratio signal arrives.

The control signals to the switches $(S1 - S8)$ were sent according to the specification of the topologies from Table 3.1. The phase period was $1\mu s$ as it was set to the phase delay element. The dead time was around $1.5ns$ - the delay of the logic gates has been added to the delay.

Fig. 3.18 shows the efficiency of a large SCC ($nWidth = 150\mu m$, $C_{fly1} = 1nF$, $C_{fly2} = 0.5nF$) with a large load ($R_L = 1K\Omega$), operating with different ratios. The efficiency of the ratio 2/4 is higher than other ratios. However, this efficiency is lower than one, achieved using the two-phase converter. Fig. 3.19 shows the power of the controllers in different operation modes. Although, for all the ratios the power consumption grows linearly with frequency increase, in a stand-by mode the controller does not consume any dynamic power.

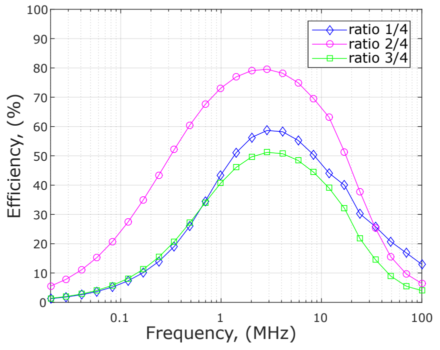

Figure 3.18: Efficiency of the large multi-phase SCC with the large load for different ratios.

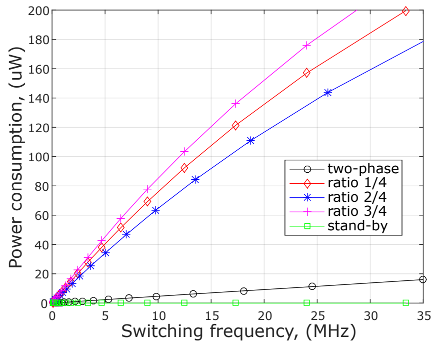

Figure 3.19: Power consumption of the generated SCC controllers.

3.4.4 Simulations of the self-oscillating SCC

The designed circuit of the self-oscillating SCC has been simulated with the following parameters of the simulation:

- Type of the SCC - step-down with the ratio 1/2.
- Number of stages - 5.
- Technology - umc65nm.
- Input voltage - $V_{in} = 2.5V$.
- Flying capacitor - $C_{fly}i = 50pF$.
- Width of *pmos* - $P_{width} = 10\mu m$.
- Width of *nmos* - $N_{width} = 5\mu m$.
- Load resistance - $R_{load} = 5K\Omega$.

For the flying capacitors, we used MIM capacitors, which have $\lambda_{bp} = 1.5\%$. That means that the parasitic capacitors have the capacitance equal to 1.5% of the flying capacitors. In this case $Cbp = 0.75pF$.

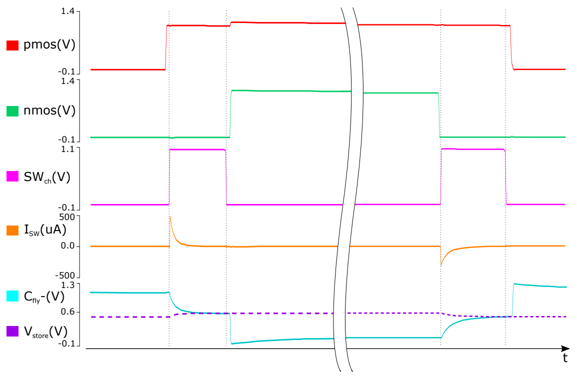

Figure 3.20: Charge redistribution simulation results.

The results of the simulation in fig. 3.20 show the process of the parasitic charge redistribution, and all the signals associated with it. The figure shows the signals of the lower part of one stage of the SCC. The flying capacitor of this stage is switching from the charging state to the discharging (the input of the inverter changes from $'0'$ to $'1'$). The *pmos* switches to $'1'$ first, and the *nmos* - only after a certain delay. During this delay, the inverter is in a third state, and the charge redistribution switch (SW_{ch}) turns on. The current through this switch (I_{sw}) discharges the bottom plate of the flying capacitor ($C_{fly}-$) to the store capacitor. When the delay finishes, the voltages of these capacitors are almost equal.

After some time the flying capacitor is switching back to the charging state (the input of the inverter changes from $'1'$ to $'0'$). The switching process occurs in the opposite way: the *nmos* switches to $'0'$ first, and the bottom plate of the flying capacitor is charging from the store.

The average voltage of the C_{store} is around $550mV$. It is a bit lower than $V_{out}/2$, because of the losses associated with the charge transfer.

In Fig. 3.21 we compare the efficiency of the designed SCC (Fig. **??**) with the initial self-oscillating SCC (Fig. 2.7). The peak efficiency of the SCC with the charge redistribution is higher by $1.5 - 2\%$, and the efficiency gain is higher with the higher frequencies. However, this converter requires extra space for the store capacitor, which size is equal to C_{fly}. In another simulation, we used the smaller C_{store} that equals to $0.1C_{fly}$, and its efficiency was almost the same as the model with the large C_{store}.

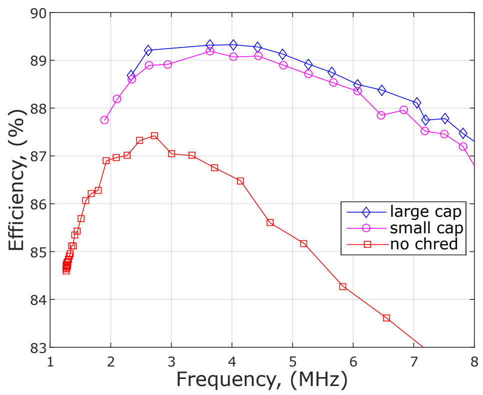

Figure 3.21: Efficiency vs frequency with $C_{fly} = 50pF$ for the SCC without charge redistribution and with it, with the large and the small store capacitor.

However, the advantage of this method is lower with the lower size of C_{fly}. Fig. 3.22 shows the efficiency of the converter with smaller flying capacitors: $C_{fly} = 20pF$. The difference in the peak efficiencies is smaller here (around 1%).

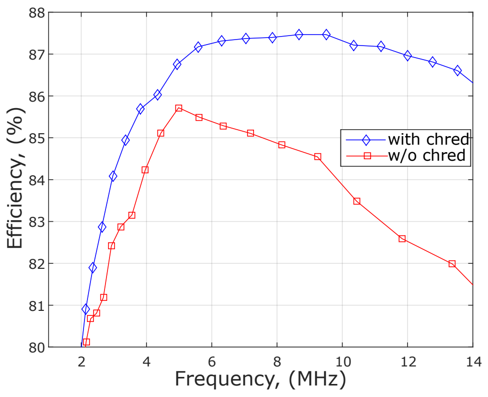

Figure 3.22: Efficiency vs frequency with $C_{fly} = 20pF$ for the SCC with and without charge redistribution.

It may happen that, if we use too small flying capacitors, the method can give even worse efficiency, because the losses, associated with the additional logic gates can be larger than the energy we save by the charge redistribution.

3.5 Summary

In this chapter, we look at the shoot-through currents in SCCs through the prism of hazards in digital circuits. Namely, we define a sequence of safe transitions in accordance with the theory of asynchronous circuits. Common properties of asynchronous circuits allow us to state that the asynchronous control of SCCs will have advantages over the synchronous ones when the operating conditions are intermittent.

We believe that the steps described in Sections 3.1 and 3.2 can be automated. Ideally, the user would only input the required ratios and SCC topologies, and the tool would automatically produce a completed implementation of the corresponding controller.

While the multi-phase SCC controller is quite large, its size can be reduced. One of the possible ways of doing this is by using the David cells [74] in the controller part that interfaces the delay elements. The circuit, which is based on the David cells connected into a ring with delays, can generate the phase and dead time delays for the controller. In this case, the controller would not bother of handling the delay request and acknowledgement signals, and its circuit can be significantly simplified.

Another optimization is to simplify the operation of the signal *ratio_req* in such a way that it does not interrupt the generation of the switch control signals. These optimization possibilities are a subject for future research.

The dead time delay is usually rather small: it is comparable to the delay of a logic gate and is three orders of magnitude smaller than a phase delay. Therefore inaccuracy in this delay caused by the process variation would not have a critical effect on the efficiency of the entire system.

The problem of the method of parasitic charge redistribution (proposed in section 3.3) is that it works only when the parasitic capacitance of the added logic gates is, at least, twice smaller than the bottom plate parasitic capacitance. Therefore the method will have less advantage in the large scale technologies with larger logic gates.

In order to further optimize the method of parasitic charge redistribution, the delays should be calculated more accurately. From the one hand, we should have as a small delay as possible, because during the delay the flying capacitor is not operating, that adds some extra losses to the SCC. From the other hand, this delay should be large enough to give the parasitic capacitor enough time to charge or discharge, otherwise, the method becomes meaningless. The charge can be transferred faster if we use larger charge redistribution switches (SW_{ch}). However, larger switches consume more dynamic power. The optimal values of the switch size and the delay can be calculated using the initial parameters of the SCC.

The simulations of the self-timed SCC controller demonstrated in sections 3.4.2 and 3.4.3 validate the correctness of the controller's operation. The power consumption of the controllers may vary depending on the chosen topology. The two-phase SCC controller consumes less power compared to the multi-phase one with the ratio $1/2$. Although, they both perform the same operation (divide the input voltage by two), the two-phase SCC is more efficient. But the multi-phase SCC is able to provide multiple conversion ratios, whereas the two-phase SCC operates only with a single ratio.

The method designed in section 3.3 applies the parasitic charge redistribution to the self-oscillating SCC. The simulations in section 3.4.4 show that the proposed method improves the efficiency of the SCC by $1 - 2\%$. Although it is quite a small number, it can be significantly higher in the SCC systems that consist of several SCC units like in [15].

Chapter 4

Power-elastic PWM-based perceptron

In this chapter, we improve the computational part of the AI-related IoT devices. This improvement aims to increase the robustness of the device to supply voltage variations.

The chapter focuses on the design of the PWM-based perceptron, including the fundamental theories, the circuits of its constituent parts, methods of PWM-based arithmetic, leading to the construction of NNs. The design methods form the basis of extensive analysis supporting the validation of the perceptrons integrated into a NN.

A perceptron capable of voltage and frequency elasticity may be constructed by exploiting the fact that relative temporal properties, such as duty cycle, are resilient to voltage and frequency variations. As the supply voltage reduces, any oscillatory activity, such as a clock signal, may show the reduced amplitude and reduced frequency. However, the ratio between the time within a period when the clock signal is high and the time within a period when the clock signal is low stays the same as both would increase at the same rate.

Our method, therefore, is dedicated to finding ways of exploiting this fact by transferring computation from the digital domain, which is affected by voltage and frequency variations, to the relative temporal domain, which is not. This means making use of PWM-based techniques.

Section 4.1 describes the main idea of PWM to voltage conversion. Based on this idea, we develop the PWM arithmetic described in section 4.2. We use this theoretical background to design a PWM arithmetic circuit (voltage accumulator). In section 4.3 we propose our approach to design the second part of the perceptron, which converts the data from voltage to PWM domain. Section 4.4 combines the designed circuits into complete perceptron design. In section 4.5 we validate our approach by designing a neural network based on the PWM perceptrons and discussing the main issues associated with it.

4.1 Principles of duty cycle to voltage conversion

Figure 4.1 shows an inverter-based PWM to voltage converter, which produces an output voltage whose value represents the value carried by the input PWM signal, i.e. its duty cycle. Here we exploit the principle that if the input of an inverter is a periodic signal, such as a clock, the average voltage on its output is inversely proportional to the duty cycle of the input signal. In other words, the analogue average value of the inverter′s output voltage encodes the value of the duty cycle of the input signal. Since an inverter is a digital component, whose output equals to logic ′0′ or ′1′ at any moment in time, it needs to be ″analogised″ (i.e. transcoded) to convert the input duty cycle into the output voltage that is a corresponding proportion of the supply voltage. This may be achieved in the following ways:

- increasing the input switching frequency,
- increasing the output capacitance,
- limiting the output current.

For the inverter-based PWM to voltage converter shown in Figure 4.1, with the input clock duty cycle at 50%, the average output voltage is around $V_{dd}/2$ (Figure 4.2). This is due to the fact that during the interval of time when the input is low the output capacitance is charged with current from the power source via the PMOS transistor, and during the interval of input being high, the capacitance is discharged via the NMOS transistor. With a 50% duty cycle these two periods of time are the same length and, assuming the transistors are balanced, their voltages average out to half the supply voltage. When the duty cycle deviates from 50% the average value of the output voltage deviates from $V_{dd}/2$ proportionally in the same direction.

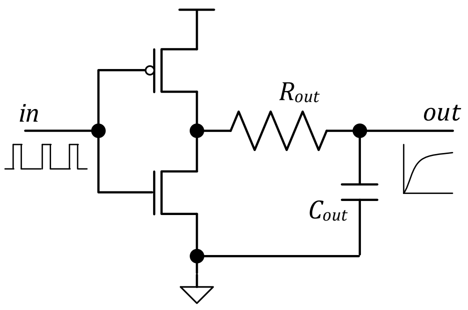

Figure 4.1: A CMOS-based inverter circuit.

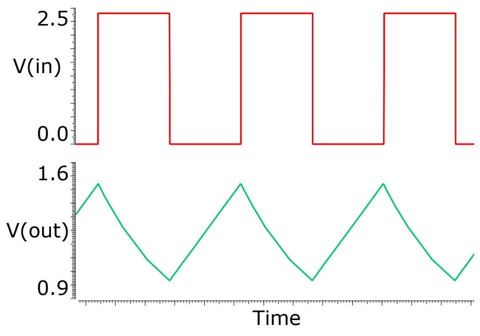

Figure 4.2: Inverter output with PWM-coded input.

If the frequency is high enough that the output capacitor is never fully charged or discharged, the inverter may be equivalently represented as a resistive voltage divider (see Figure 4.3). The output voltage of such a divider can be calculated using the following equation.

$$V_{out} = (V_{dd} - GND) \cdot \frac{R_n^* + R_{out}^*}{(R_n^* + R_{out}^*) + (R_p^* + R_{out}^*)}. \tag{4.1}$$

where R_n and R_p are parasitic resistances of NMOS and PMOS transistors. During the charging phase (t_{low}) the input of the inverter is low and current passes through the PMOS and the output resistor. During the discharging phase (t_{high}), the input of the inverter is high and current goes through the output resistor and NMOS. As a result, the resistance values can be calculated from the lengths of time of each phase:

$$R_n^* + R_{out}^* = (R_n + R_{out}) \cdot \frac{t_{low} + t_{high}}{t_{high}}; \tag{4.2}$$

$$R_p^* + R_{out}^* = (R_p + R_{out}) \cdot \frac{t_{low} + t_{high}}{t_{low}}. \tag{4.3}$$

Assuming that $R_n \approx R_p$ (this transistor balancing can be achieved by the appropriate relative sizing of the PMOS and NMOS transistors, for instance, by setting the PMOS width to 2.7 times the NMOS width for the UMC65nm technology) and $GND = 0$, the equation 4.1 is simplified to:

$$V_{out} = V_{dd} \cdot \frac{t_{low}}{t_{low} + t_{high}} = V_{dd} \cdot (1 - DC), \tag{4.4}$$

where DC is the input duty cycle - the ratio between the length of time when the input clock is high during a clock period and the length of the clock period.

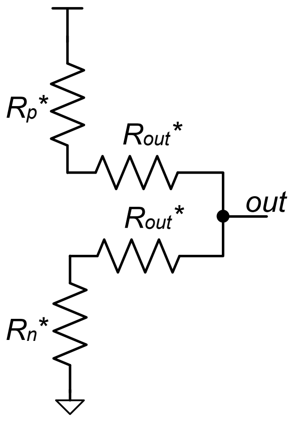

Figure 4.3: PWM inverter equivalent circuit, approximated as a voltage divider.

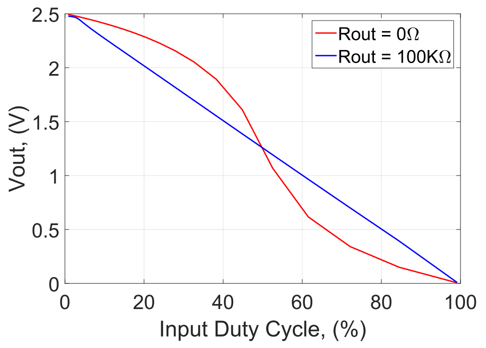

Figure 4.4: Output voltage of the PWM inverter vs input duty cycle.

Figure 4.4 shows the relationship between the input duty cycle and the output voltage of the PWM inverter. In the case when there is no output resistor, the dependency of the output voltage on the input is not linear. The reason for this non-linearity is that the PMOS and NMOS resistances change with the change of their drain voltages. Thus, $R_p \neq R_n$ when the value of V_{out} is different from $V_{dd}/2$. This non-linearity, given the arithmetic functional requirements of a perceptron, is undesirable and needs to be either removed or compensated for. Compensation means very high per-inverter overheads which need to be precise in the analogue domain. However, by adding an output resistor $R_{out} \gg (R_p, R_n)$, the difference between PMOS and NMOS resistances no longer affects the output, and the input duty cycle to output voltage relationship becomes completely linear. This requires no high-precision tuning in the analogue domain.

4.2 PWM arithmetic

A perceptron needs to perform arithmetic operations. Converting from PWM to voltage is not the only function of the PWM inverters. They can also be used to construct arithmetic units, such as adders and weighted accumulators. Below we discuss these two operations and their circuits relevant to NNs.

The circuit of a PWM adder is shown in Figure 4.5. To add n PWM-coded numbers we use n inverters connected in parallel. Each inverter has an output resistor. The result is stored in the output capacitor in the form of its average voltage.

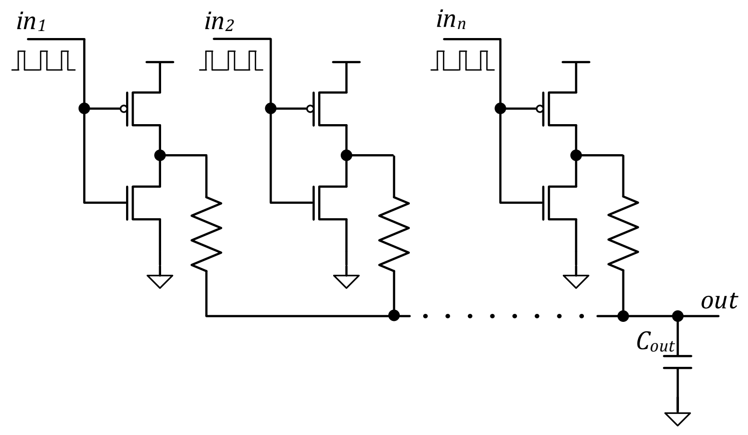

Figure 4.5: PWM adder circuit performed by parallel inverters, with outputs connected via a capacitor.

This kind of adder works on the principle of current summation and charge (i.e. voltage) accumulation. In other words, the values encoded in the input PWM signals are accumulated in the voltage on the output capacitor, and such circuits can be called voltage accumulators (VACs). To calculate the VAC output voltage, we use the principle of current summation and rewrite 4.1 using conductances instead of resistances. The following equation is for a single PWM inverter:

$$V_{out} = V_{dd} \cdot \frac{G_p^*}{G_p^* + G_n^*}, \tag{4.5}$$

where $G_p^* = \frac{1}{R_p^* + R_{out}^*}$ and $G_n^* = \frac{1}{R_n^* + R_{out}^*}$.

Likewise, equation 4.4 can be expressed as follows:

$$G_p^* = G \cdot \frac{t_{low}}{t_{low} + t_{high}} = G \cdot (1 - DC), \tag{4.6}$$

$$G_n^* = G \cdot \frac{t_{high}}{t_{low} + t_{high}} = G \cdot DC, \tag{4.7}$$

where $G = \frac{1}{R_p + R_{out}} = \frac{1}{R_{out} + R_n}$.

Since the inverters in Figure 4.5 are connected in parallel, the output voltage of a multi-inverter VAC can be given by:

$$V_{out} = V_{dd} \cdot \frac{\sum_{i=1}^{n} G_{pi}^*}{\sum_{i=1}^{n} (G_{pi}^* + G_{ni}^*)}. \tag{4.8}$$

Using equations 4.6 and 4.7, equation 4.8 can be simplified as:

$$V_{out} = V_{dd} \cdot (1 - \frac{\sum_{i=1}^{n} DC_i}{n}). \tag{4.9}$$

In simple terms, the output voltage of a multi-inverter VAC is inversely proportional to the average value of the duty cycles of its inputs, which is exactly what is required.

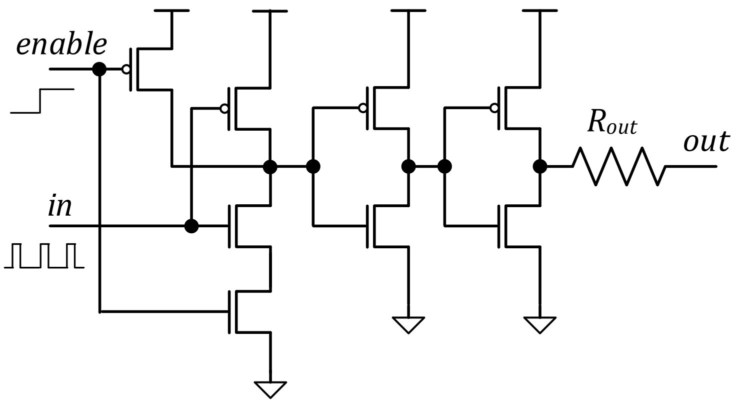

Figure 4.6: A single cell of the PWM weighted adder, based on a NAND gate.

In order to design a perceptron, the ability to integrate weighted additions is another crucial design requirement. The VACs must be capable of programming the input weights when required. This is performed by replacing the inverters by two-input

NAND gates (Figure 4.6). One input of this gate is the PWM-coded signal, and the other is a digital switch signal for enabling or disabling this cell. The output of a disabled cell is always connected to V_{dd} having the same effect as an enabled cell with zero input duty cycle. In this way, the perceptron can be programmed to determine which NAND gates participate in the accumulation. This programming may be carried out in the digital domain without affecting the voltage and frequency elasticity of the computation.

Figure 4.7 shows a perceptron arithmetic VAC architecture for 3×3 weighted addition based on these types of gates. As can be seen, the circuit adds 3 PWM-coded inputs multiplied by 3-bit weights. Every weight bit is implemented on a separate cell. The least significant bit goes to the cells with the smallest transistor sizes and the largest output resistors (cells '$\times$1'). The second bit is computed at the cells with doubled transistor widths and halved output resistances (cells '$\times$2'). And the most significant bit is coded with 4 times the transistor widths, and 1/4 times the output resistances (cells '$\times$4').

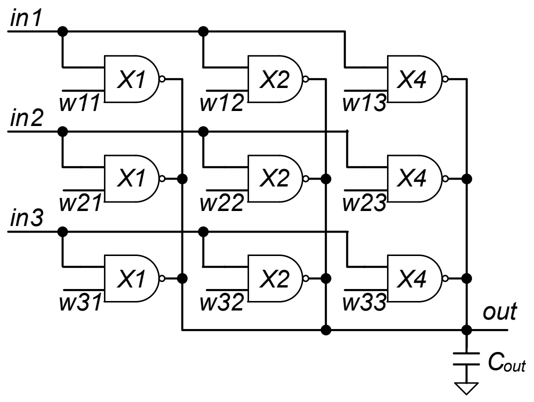

Figure 4.7: PWM weighted addition VAC with 3 inputs and 3-bit weights.

The output voltage of the 3×3 weighted addition VAC can be calculated using 4.9, considering the $\times 2$ and $\times 4$ cells as 2 and 4 single cells respectively.

$$V_{out} = V_{dd} \cdot (1 - \frac{\sum_{i=1}^{n} DC_i \cdot W_i}{n \cdot (2^k - 1)}). \tag{4.10}$$

where n is the number of inputs, k is the number of bits of the weight, DC_i is the duty cycle of the input i, and W_i is the weight of the input i.

In the case of the 3×3 weighted addition VAC, where $n = 3$ and $k = 3$, the output voltage is:

$$V_{out} = V_{dd} \cdot \left(1 - \frac{\sum_{i=1}^{3} DC_i \cdot W_i}{21}\right). \tag{4.11}$$

The arithmetic part of this equation is the weighted sum of duty cycles DC_{sum}:

$$DC_{sum} = \frac{\sum_{i=1}^{3} DC_i \cdot W_i}{21}. \tag{4.12}$$

Thus, the definition of the 3×3 weighted addition VAC is that its output voltage is proportional to the weighted sum of its input duty cycles, which is exactly as required:

$$V_{out} = V_{dd} \cdot (1 - DC_{sum}). \tag{4.13}$$

4.3 Voltage to PWM conversion

In order to design a VAC based perceptron, we need to provide an output interface for it. The output of the perceptron must be used as an input for the perceptrons of any subsequent layer in an NN. Therefore the output voltage of the PWM arithmetic unit (its VAC) should be converted back to the PWM format.

The schematics of the voltage to PWM converter is shown in Figure 4.8. The converter circuit was proposed originally by [75]. The converter is a ring oscillator with different power supplies: the odd-numbered inverters are supplied with a voltage of $V_{dd}/2$, and the even-numbered inverters are supplied with the input voltage, which is the output voltage of the VAC. The difference between the supply voltages of the odd- and even-numbered inverters determines the output duty cycle. If the input voltage equals $V_{dd}/2$, the inverters have an equal delay and the output duty cycle is 50%. If the input voltage increases, the period of switching from 0 to 1 increases, and the output duty cycle goes down. If the input voltage is lower than $V_{dd}/2$, the switching from 1 to 0 takes more time, and the output duty cycle goes up.

Given that the VAC theoretically achieves a linear relationship between its input duty cycle and its output analogue average voltage, the voltage to PWM converter should also ideally achieve a linear conversion relationship. In that case, the overall relationship

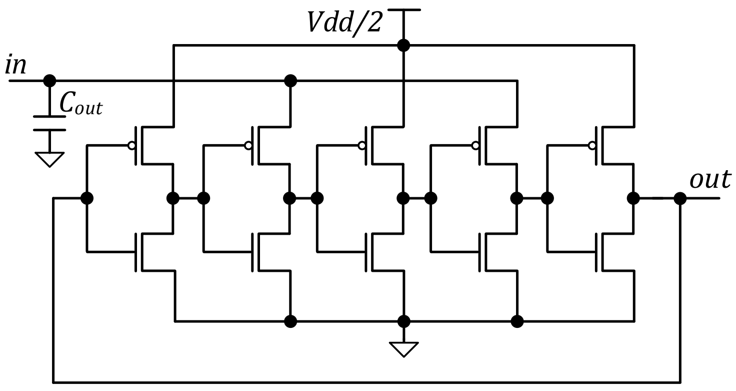

Figure 4.8: The ring oscillator based voltage to PWM converter.

between the input duty cycle signal and the output duty cycle signal would also be linear, for the simple case where the perceptron is programmed to do no arithmetic processing. In theory, the inverter chain-based voltage to PWM converter should be able to achieve this if the inverters are set to work in the linear regions of their transistors.

4.4 PWM-coded perceptron design

The PWM-based perceptron consists of two main parts. The first part is the PWM arithmetic unit in the form of a VAC. This converts the PWM-coded inputs to a voltage which encodes the result of the computation as programmed by the enable signals. The second part then converts this voltage result to PWM format for use as inputs by subsequent perceptrons as their inputs.

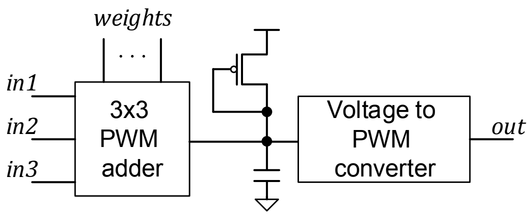

Figure 4.9: Structure of the perceptron: PWM adder, voltage to PWM converter, and compensation transistor.

This structure is shown in Figure 4.9, with the 3×3 weighted addition VAC as an example PWM-based arithmetic unit. Any desired VAC arithmetic unit can be put in this place to satisfy specific perceptron functionality requirements. The simple glue logic consisting of a PMOS transistor between the two blocks will be discussed in detail in Section 5.2.

The size of such a perceptron is such that its design may be entirely analysed and validated through simulations within the VLSI CAD environment in which it is implemented. At least some of this analysis must be conducted in the analogue signal domain as the voltage signal between the two parts of the perceptron holds the computation results in its analogue value. As a result, simulations in a VLSI CAD tool environment that support mixed-signal or analogue studies are the best way of analysing and validating such designs. In this work, we implement our perceptron and analyse it using the Cadence Analogue Design Environment. Detailed results will be shown in Section 5.1.

4.5 PWM-coded neural network design

The proposed PWM perceptrons can be used in constructing traditional NNs such as the example shown in Figure 4.10. In this NN, the input vector (*in*) is fed to the input layer, and the activity propagates through a number of hidden layers to reach the output layer, where the output vector (*out*) is generated. Then, the output vector is compared to the target vector and the error is back propagated to update the weights of each layer using gradient descent. This procedure is iterated concerning the specified approach.

In this work, the *in* and *out* signals are of the PWM-type. The value of such a signal, which is between 0 and 1, is represented by its duty cycle value between 0% and 100%. The VAC arithmetic units then compute on such *in* values. This is illustrated by the example described by equation 4.12, where each *in* is multiplied by its weight and all results are accumulated by the VAC in the DC_{sum} voltages. In other words, the weight and sum blocks in Figure 4.10 are implemented by the proposed perceptron's VAC. Then, every DC_{sum}, which is an analogue voltage across a capacitor, is fed to the activation function (AF) whose output is in PWM format to be used as the input of the next layer. To

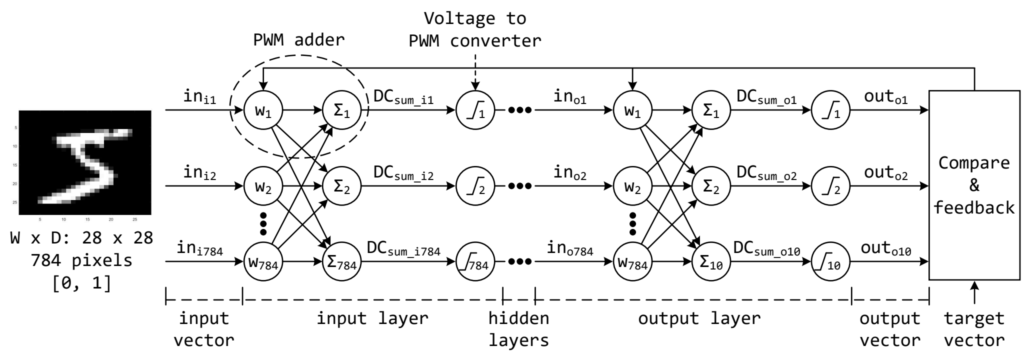

Figure 4.10: Neural network for MNIST. The DC_{sum} signals are voltages. *in* and *out* signals are duty cycles.

include the AF, equation 4.12 can be modified as expressed in equation 4.14. This requires that the voltage to PWM conversion also implements the AF. Potential modifications from the basic ring oscillator may be necessary, although the basic ring oscillator already approximates a popular AF. This will be discussed in detail in Section 5.6.

$$out = f(DC_{sum}) = f(\frac{\sum_{i=1}^{n} in_i \cdot W_i}{n \cdot (2^k - 1)}) \tag{4.14}$$

Finally, *out* is obtained, the error is calculated and every weight is adjusted by the back-propagation (BP) algorithm. The comparison to the target vector is not necessarily implemented with a perceptron-like device and may be implemented by some external controller, which is outside the scope of this paper.

For the PWM-coded NNs, a number of design choices must be made: weight encoding, maximum weight, AF and number of layers, among others. This section establishes a method of making the best use of the proposed PWM-based perceptron to construct NNs to perform specific computational tasks. We will explore aspects of NN design, including weight types, AF, maximum weight and number of layers. We will use the well-known handwriting digit (MNIST) classification problem [76], which is widely used for machine learning algorithm testing [77], as the benchmark application and case study for this investigation. The goal is to suitably determine the best NN configurations for the proposed PWM-coded NN.

Regarding the circuit design, the weight is discretised to an integer value. This is different from most related work where floating-point (FP) numbers are used for weights. As the circuit size depends on the bit-width of the weight, it is crucial to find the smallest bit-width that still provides the specified error rate tolerance.

The integer weight training can be designed as illustrated in Figure 4.11. The MNIST input vector (*in*) is multiplied by the integer weight (*W*) and the results are accumulated as *out*. Then, *out* is divided by $n \cdot (2^k - 1)$ (i.e. normalising), which yields the final value of *out* between 0 and 1.

Consequently, *out* is scaled to the same range and comparable to the target vector. Then, *out* passes the AF, and the output vector is obtained and compared to the target vector. Next, the FP update is computed from the gradient descent, the learning rate, and the error. To adjust the integer weight, the update is scaled back to the integer number by multiplying by $n \cdot (2^k - 1)$ and rounding. Next, the integer weight is updated and capped if it exceeds the specified bit-width (e.g. the example 3×3 weighted addition VAC in Figure 4.7 has 3-bit weights). Finally, the training process iterates until the number of specified epochs is reached. Note that the weight capping can be disabled to allow unlimited weight adjustments to mimic FP training.

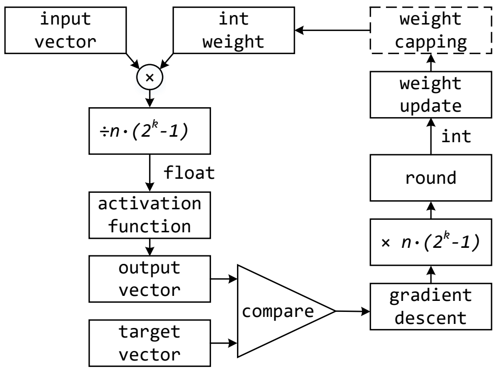

Figure 4.11: Integer weight training.

The activation function is necessary for an NN-based learning process because it provides non-linearity to the computation so that the learning is not limited to linear problems. It also helps map the resulting values in a certain range, depending on the function.

In this work, the input and output ranges of the AF are the main concern because they need to match the output format of the problem and the circuit behaviour. In other words, depending on the purpose of the NN, it may expect its input and output variables to take values within certain ranges. These ranges then need to be mapped onto the working signal range of our perceptron, which is restricted by the duty-cycle representation between 0% and 100%. Here we take a popular MNIST benchmark [78] as an exemplar to explore this aspect of NN design using our perceptron as the basic building block.

In the context of MNIST, the AFs are needed to provide a fully positive output to comply with the target vector [76]. Also, our perceptron design stores the VAC result as the voltage across C_{out} between its two blocks, which means that the DC_{sum} signals are entirely positive voltages. And such a voltage gets converted to a PWM duty cycle, which is also entirely positive. For these reasons, the well-known AF ReLU [54, 79], which has an entirely positive output range, is best suited.

Certain other popular AFs are less suitable for this initial investigation. For instance, the sigmoid function is clearly non-linear across an input range between -5 and 5 [54], which requires the representation of negative values. The non-linearity also means that major modifications to the voltage to PWM part need to be investigated for implementing such AFs. hence, we decided to concentrate on trying to mimic the ReLU AF using our perceptron's voltage to PWM converter.

$$f(x) = \begin{cases} 0 & , x < 0 \\ x & , x > 0 \end{cases} \tag{4.15}$$

$$f(x) = \begin{cases} 0 & , x < 0 \\ x & , 0 < x < 1 \\ 1 & , x > 1 \end{cases} \tag{4.16}$$

The ReLU function in equation 4.15 [79] is depicted in Figure 4.12. One of its attractions is that it is easily differentiable, facilitating gradient descent. To mimic the output of the VAC, it is better than the sigmoid function because the charge in the output capacitor (C_{out}) is emptied when the VAC result is negative. Otherwise, the capacitor is charged and the positive result is obtained. However, the output of this function must be capped at 1 to represent the limit of the PWM range as shown in equation 4.16 and Figure 4.13. This work will attempt to construct an AF that approximates the capped ReLU function.

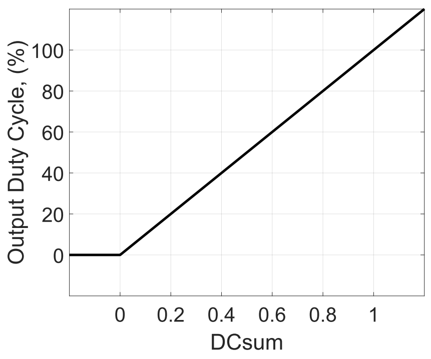

Figure 4.12: ReLU function.

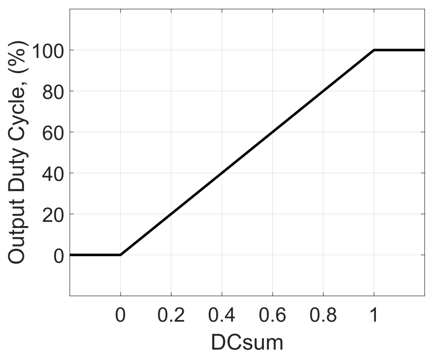

Figure 4.13: Capped ReLU function.

The size of an entire NN designed for the MNIST problem is such that it is not possible to analyse it entirely within a VLSI CAD environment. For instance, to analyse an image of 784 pixels (cf. the example in Fig. 4.10) there need to be 784 perceptrons in the first layer of the NN alone and this is clearly beyond analogue simulations at the VLSI level. Effort must be expended in building models in a higher-level language to investigate the design properly.

4.6 Summary

We propose the first mixed-signal (analogue/digital/relative temporal) perceptron design using the principles of PWM. Central to our design are a number of parallel inverters that suitably transcode the input-weight pairs from the spatial domain to the relative temporal domain. This approach aims to deliver high resilience to amplitude and frequency variations in the supply voltage, exploiting the fact that PWM-based solutions are typically agnostic to such variations.

Another advantage of the proposed design is its simplicity. Whilst conventional implementations of the perceptron require complex logic to perform multiplication and addition, the proposed approach uses only one gate (either an inverter or a two-input NAND) per bit for every input. Thus, for the 3×3 weighted addition VAC, we used only 54 transistors. This significantly reduces the logic requirement and, therefore, the power consumption of the entire device.

One more interesting feature of this design is the possibility to perform training including the network itself, in order to compensate for variability-induced errors. Any design variability will be compensated by the weight adjustment during the training stage.

Chapter 5

Simulation results

In this chapter, we designed the prototypes of the computational parts of the IoT device and simulated them in the Cadence Analog Design Environment tool [80]. This chapter validates the operation of the PWM-based perceptron designed in chapter 4.

A prototype circuit of the PWM perceptron is designed using $umc65nm$ technology. We used the high voltage transistors (with $2.5V$ nominal voltage) in the purpose of better observation.

The simulations start with validation of the perceptron, including its parts (voltage accumulator and voltage to PWM converter). After that, we analyzed the resilience of the perceptron to the static and dynamic variations of the key parameters. Based on this analysis, we specify the trade-offs between different perceptron parameters, such as size, frequency, power consumption, and performance.

In the last part, we analyze the PWM-based neural network described in section 4.5. This simulation cannot be performed in Cadence environment, because the NN circuit is too large. Thus, we used Cadence to generate a perceptron model and used this model in Matlab [81] simulation.

5.1 Simulation flow

In this section, we analyse the behaviour of the PWM-based perceptron designed in Chapter 4. We start the analysis with Cadence simulations of two parts of the perceptron separately: VAC and voltage to PWM converter. After that, we simulate the whole perceptron circuit and generate its model describing perceptron's input/output function. The operation of the perceptron is simulated with static and dynamic variations of the operating conditions. The results of the simulations are analysed from the viewpoint of power robustness.

We cannot simulate the whole neural network based on the PWM perceptrons using Cadence environment, because even a single perceptron simulation takes a lot of time and computational resources. To resolve this issue, we use Cadence to generate the input/output model of the perceptron. And use this model in Matlab to simulate the designed PWM-based NN. To validate this method, we design the circuit of 3 perceptrons connected in series and simulate this circuit in Cadence and Matlab, comparing the simulation results.

The flow of the PWM perceptron simulations is shown in in Figure 5.1.

5.2 Analysis and validation of PWM-coded perceptron

Below we analyse the behaviour of the perceptron circuit under different parametric variations, generated by the design tool.

The first constituent part of the perceptron is the VAC. Figure 5.2 shows the charging of the capacitor in the VAC based on three inverters connected in parallel as shown in Figure 4.5. The frequencies and duty cycles of the inputs are: $f_1 = 140MHz$, $DC_1 = 70\%$, $f_2 = 120MHz$, $DC_2 = 30\%$, $f_3 = 100MHz$, $DC_3 = 50\%$. The capacitor has been charged to the voltage value, proportional to the average duty cycle of the inputs. The charging time of the capacitor depends on the RC value, and the input frequency does not affect it. However, if the frequency is too low, it may result in a too high ripple of the output voltage, and, thereafter, reduction of accuracy.

To support our VAC design based on inverters/NANDs and voltage summation on

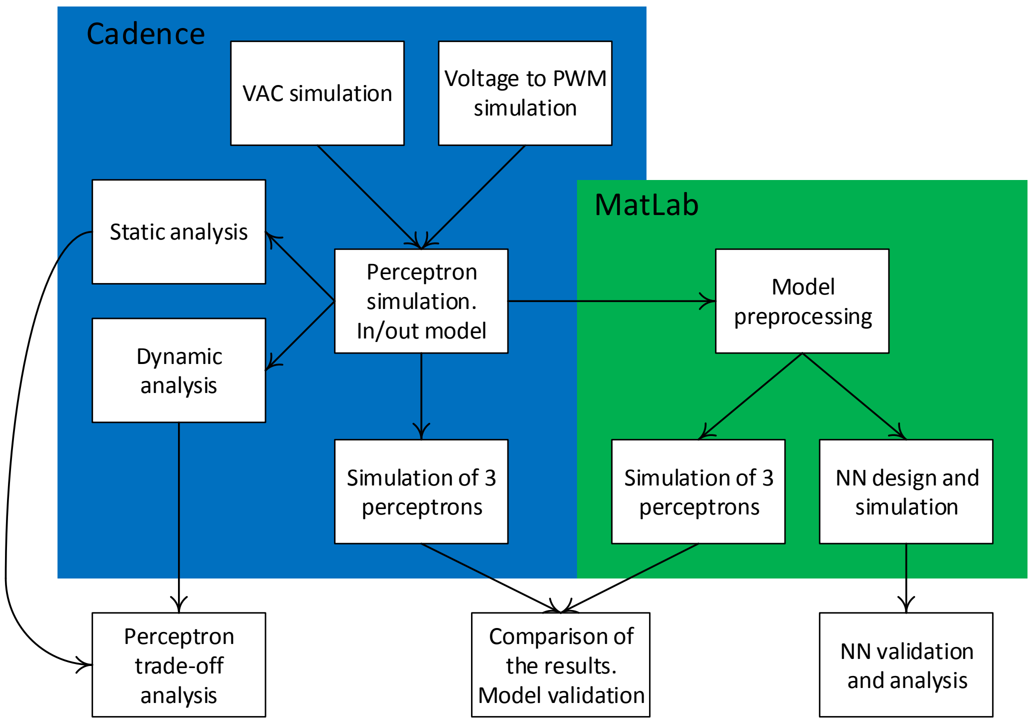
Cadence
VAC simulation
Voltage to PWM simulation
Static analysis
Perceptron simulation. In/out model
Dynamic analysis
Simulation of 3 perceptrons
MatLab
Model preprocessing
Simulation of 3 perceptrons
NN design and simulation
Perceptron trade-off analysis
Comparison of the results. Model validation
NN validation and analysis

Figure 5.1: Perceptron simulation flow.

a capacitor, we implemented the 3×3 weighted addition VAC shown in Figure 4.7 in Cadence and ran simulation experiments on it. The results of these simulations are compared to theoretical results obtained from 4.11 and compared in Table 5.1. The differences between the theoretical and simulation results do not exceed 10%. These results validate the correctness of the PWM-based weighted addition VAC design.

The second constituent part of the proposed perceptron is the voltage to PWM converter. This converts the result of VAC arithmetic computation stored as an analogue voltage (a DC_{sum} signal) back to the PWM format for output to subsequent perceptrons, as presented in Section 4.3. The Cadence Analog Environment simulation results of the voltage to PWM converter are shown in Figure 5.3. Ideally, the voltage to PWM conversion should be linear. The real relationship between the output and the input is almost linear for input voltages between $0.7V$ and $2.3V$. However, outside this range, the ring oscillator stops oscillating.

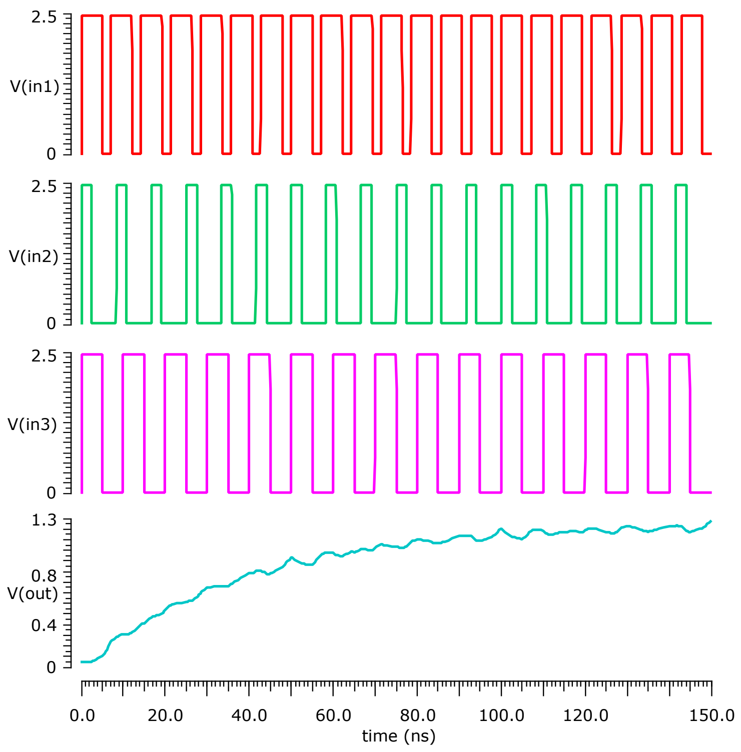

Figure 5.2: Capacitor charging in the 3 inverters VAC.

The reason for this is that the input voltage is used as a power supply for the inverters. When the voltage is below the threshold, it is not enough to trigger the NMOS transistors of the inverters, and the output of the ring oscillator becomes a constant voltage.

Another interesting effect is that the linearity of the output increases with increasing the number of inverters, but the difference between 9 and 13 inverters is small. Thus, a chain of 9 inverters should be considered a reasonable voltage to PWM converter and we use this design in our subsequent studies.

Figure 5.4 shows the combined operation of both parts of the perceptron: the 3×3 weighted PWM addition VAC (Figure 4.7) connected to the voltage to PWM converter (Figure 4.8). The three inputs of the perceptron are connected together, and all the weights are 7 (all the cells are enabled). The line labelled 'ideal' is for the desired case when the output duty cycle equals to the input duty cycle. Analysing these results we

Table 5.1: Experimental and theoretical results of the 3 × 3 weighted adder.

DC1	W1	DC2	W2	DC3	W3	V_{out} theoretical	V_{out} simulation
70%	7	80%	7	90%	7	0.50V	0.51V
50%	1	50%	2	50%	4	2.08V	2.11V
20%	5	60%	6	80%	7	1.29V	1.33V
95%	7	90%	6	80%	6	0.50V	0.45V
30%	1	40%	4	50%	2	2.16V	2.21V
80%	7	20%	3	50%	4	1.54V	1.61V

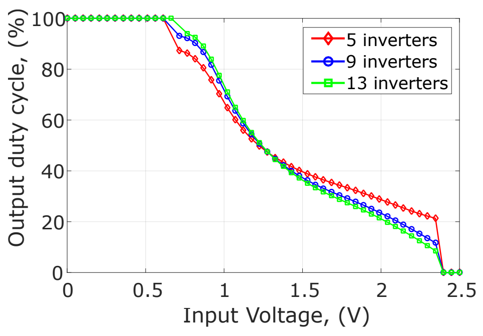

Figure 5.3: Output duty cycle of the voltage to PWM converter.

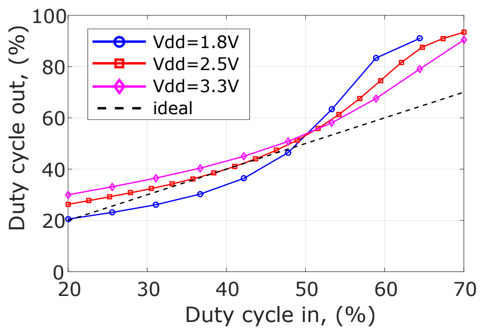

Figure 5.4: Output vs input duty cycle of the perceptron.

can say that:

- In this simulation for the ideal case, we expect the output duty cycle to be equal to the input duty cycle. However, the real output is slightly different from the ideal; and this difference increases with the input duty cycle above 50%.
- The output duty cycles for different supply voltages are similar. The difference does not exceed 10%. This indicates voltage variation resilience in the perceptron design.
- The input duty cycle has a limited range - from 20% to 70%. Beyond this range, the output stops oscillating and becomes a constant signal.

The observed reduction of operational range and loss of linearity in the voltage to

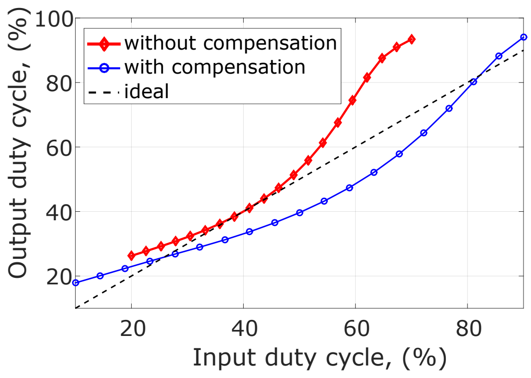

Figure 5.5: Output vs input duty cycle of the perceptron with and without compensation.

PWM converter are caused by the fact that the voltage DC_{sum} powers the voltage to PWM converter. When the input duty cycle is above 70%, DC_{sum} is below 30% of V_{dd}. For $Vdd = 2.5V$ this is below the threshold voltage. And in this case, the NMOS transistors of the ring oscillator are always off, and the output stops oscillating. In other words, there is a mismatch between the voltage ranges of the two parts of the perceptron. The output voltage range of the PWM weighted addition VAC is from $0V$ to $2.5V$ (Figure 4.4); the input voltage range of the voltage to PWM converter is from $0.7V$ to $2.3V$ (Figure 5.3).

We may limit the range of the output voltage of the PWM weighted addition VAC. This can be done by adding a small glue logic between the two blocks of the perceptron. This may consist of no more than a compensation PMOS transistor, whose gate and drain are connected to the capacitor as shown in Figure 4.9. In this case, when the voltage on the capacitor goes below the threshold, the PMOS starts charging this capacitor, and when the voltage is above the threshold, the PMOS is off.

The input and output duty cycles of the perceptron with compensation are depicted in Figure 5.5. The output is closer to the ideal, and its range is much wider: from 10% to 90%.

5.3 Power elasticity and resilience in static operation

To demonstrate the perceptron's resilience to power variations we simulated the 3×3 PWM-based weighted addition VAC circuit (Figure 4.7) with different values of supply voltage and input signal amplitude. The results are shown in Figure 5.6. As can be seen, the output voltage grows almost linearly with increased V_{dd}. As expected, higher duty cycle shows lower output voltages and vice versa. In the case of the unstable supply voltage, the absolute value of the output voltage does not bear any reliable information. In this case, we should consider the relative relationship between the output voltage and the supply voltage. This relationship should be proportional to the input duty cycle independently from V_{dd}. This is demonstrated by Figure 5.7 where the y axis represents not the absolute value of V_{out}, but the ratio between V_{out} and V_{dd} that is more relevant for unstable power conditions.

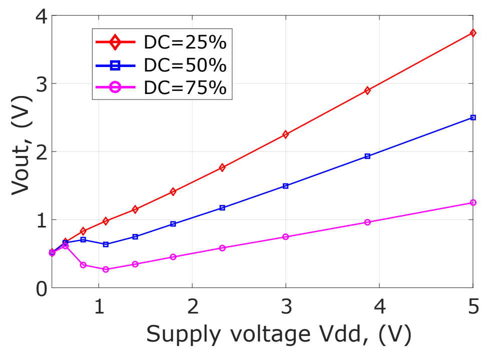

Figure 5.6: Output voltage (absolute values) vs static variation of power supply.

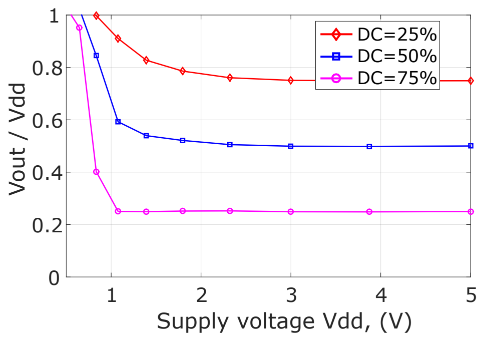

Figure 5.7: Output voltage (relative values) vs static variation of power supply.

The circuit shows high resilience to static supply voltage variations. Starting from 1 - 1.5V the ratio V_{out} and V_{dd} remains the same for each duty cycle value of the input signal.

Further simulation experiments are carried out to investigate the VAC's resilience to static frequency variations. Two sizes of the 3×3 VAC are investigated: the small - with the output capacitor $C_{out} = 10pF$ and the output resistors of each cell $R_{out} = 100K\Omega$; and the large - with $C_{out} = 100pF$ and $R_{out} = 1M\Omega$. The duty cycle of all the inputs is 50%, and all the weights equal to 7 (all the cells are enabled). Figure 5.8 shows that both VACs produce the output $1.25V$, that equals to $V_{dd}/2$. The average output voltage remains the same on the simulated range of frequencies: from $1kHz$ to $1GHz$.

On the other hand, the value of C_{out} does affect other aspects of perceptron performance. C_{out} contributes to the RC time constant of the VAC circuit, providing a low-

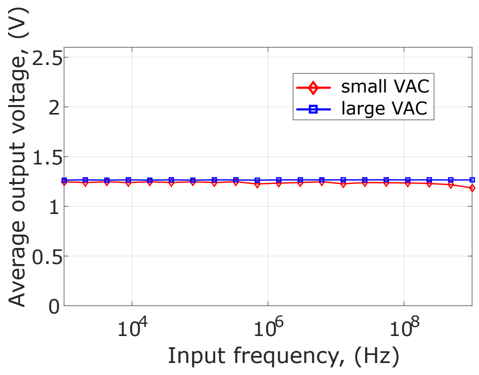

Figure 5.8: Output voltage vs static variation of input frequency.

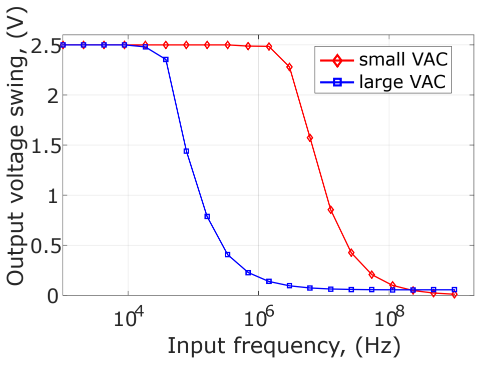

Figure 5.9: Output voltage swing vs frequency.

pass filter effect on the voltage DC_{sum}. As a result, a larger C_{out} is less suitable than a smaller C_{out} for the fast response, but it would provide better robustness in the presence of frequency variations. In addition, as the voltage to PWM converter depends on the charge on C_{out} for energy, a smaller C_{out} may encounter difficulties in keeping DC_{sum} constant enough to complete the conversion.

Figure 5.9 shows the DC_{sum} voltage swing in the presence of static frequency variations. As can be seen, with reduced input frequency the voltage swing increases, and at some point, the VAC operates as a simple inverter with the output voltage DC_{sum} oscillating between V_{dd} and GND. Ideally, we would like the voltage swing to be not larger than $0.2V$. It means that the frequency of the input PWM signals should not be lower than $1MHz$ for the large VAC and $100MHz$ for the small VAC.

In addition, Figure 5.10 shows that VAC size and frequency also affect power consumption. The small VAC has higher power consumption. This is due to the output resistor limiting the charging current. The resistor is $10\times$ larger in the large VAC, and the current and the power consumption are smaller.

In the large VAC, we increase the size of C_{out} and reduce the charging current. This increases the charging time of the capacitor. To investigate this we simulated the time when the voltage on C_{out} reaches the average output value (which is $V_{dd}/2 = 1.25V$ for the 50% input duty cycle). The capacitor is initially charged to $V_{dd} = 2.5V$. The charging

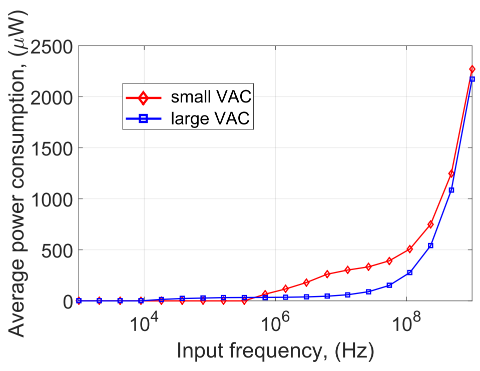

Figure 5.10: Power vs frequency of the 3x3 VAC.

time of the capacitor is around $0.14\mu s$ for the small VAC and $14.5\mu s$ for the large VAC, which is true for the entire range of frequencies. This $\sim 100\times$ ratio is because the RC product is $100\times$ as large for the large VAC as for the small VAC.

5.4 Analysis of perceptron operation in dynamics

Fig. 5.11 shows the process of discharging the output capacitor. Initially, the voltage on the capacitor is equal to V_{dd}. As can be seen, the response time of the discharging process is related to the output resistance R and the output capacitance C (the charging process is similar to a direction change). The time constant RC is proportional to the charging/discharging delay for the V_{out}'s average value $\overline{V}_{out}$. In addition, the ripples in V_{out}'s instantaneous value resulting from PWM carrier oscillations have higher amplitudes for smaller RC values, because of the low-pass filtering effect of the RC circuit. In these experiments, different resistance and capacitance values are selected to have different RC as well as different capacitance and resistance values composing to the same RC value. In these experiments, the PWM frequency is set to $f_{pwm} = 30MHz$.

In the static state, when the operating conditions, input frequency and duty cycle are constant, V_{out} has a saw-tooth shape. Fig. 5.12 presents the shape of the output capacitor voltage of a single PWM inverter. Only $\overline{V}_{out}$ is used for perceptron computation, with the VAC's arithmetic result encoded by $\overline{V}_{out}/V_{dd}$ (See equation (4.10)). However, the voltage swing of the ripples as a result of PWM carrier oscillations may affect the accuracy and should also be analysed.

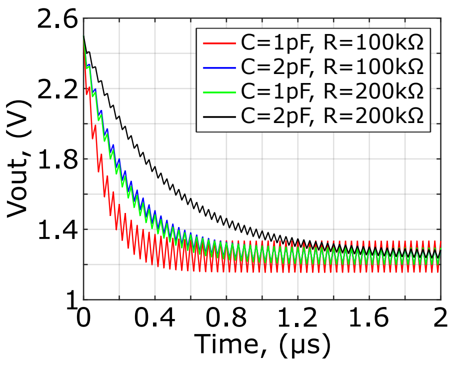

Figure 5.11: Output capacitor discharge over time.

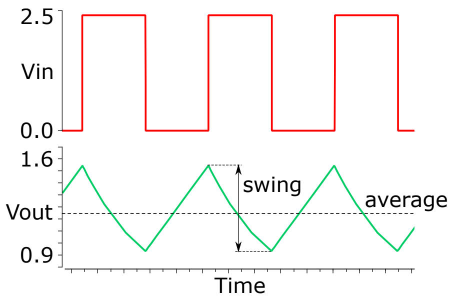

Figure 5.12: Capacitor voltage DC_{sum} corresponding to PWM-coded input.

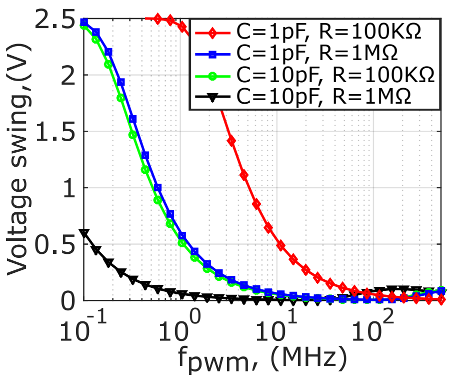

Figure 5.13: Output voltage swing vs frequency.

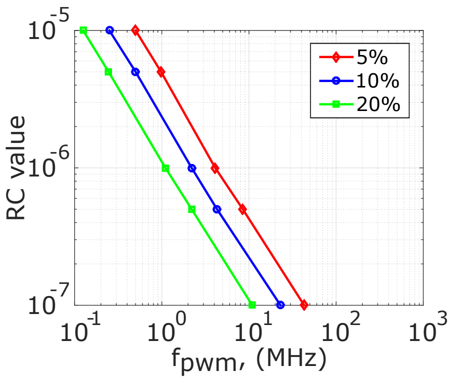

Figure 5.14: Frequency limits for different RC values.

The ripple voltage swing depends on three parameters: PWM carrier frequency (f_{pwm}), output resistance (R), and output capacitance (C). Fig. 5.13 shows the relation between the voltage swing amplitude and input frequency for different output resistors and capacitors. The ripple voltage swing increases at lower f_{pwm} values. As the carrier frequency continues to reduce, the voltage swing eventually becomes equal to V_{dd}, and the VAC starts operating as a digital device without any PWM arithmetic.

In Fig. 5.13, the voltage swing is almost the same for the cases with $C = 1pF, R =$

$1M\Omega$ and $C = 10pF, R = 100K\Omega$. The small difference between these cases is due to parasitic capacitance and resistance of the transistors that are not included in the analysis. This means that the voltage swing can be broadly defined by the perceptron's RC value.

Perceptron accuracy is affected by the ripple voltage swing amplitude. We may set an acceptable ripple swing as a percentage of V_{dd}. This acceptable voltage swing then (Fig. 5.13) results in the lowest tolerable f_{pwm} limit. If we define the acceptable voltage swing as 5%, 10% or 20% of V_{dd}, we can find the tolerable frequency ranges for different RC values (Fig. 5.14). The acceptable ranges are to the right and above the relevant boundary lines in Fig. 5.14.

Notice that the boundaries in Fig. 5.14 are straight lines, which is expected as $RC = \tau$ is time and f_{pwm} is frequency, related to each other via a hyperbola. When set in logarithmic scales the relationship between them traces a straight line:

$$\tau = R \cdot C = \frac{A}{f_{pwm}} + \tau_0, \tag{5.1}$$

where A is a constant factor and τ_0 is a constant offset. From Fig. 5.14 it is evident that for this particular system $A \approx 1$ and τ_0 depends on the acceptable voltage swing. This means that a single, or at most two, experiment points would be needed to determine the boundary condition for any specific acceptable voltage swing value.

We can use this method to determine the design parameter space for the R and C values as well as the lowest value of the PWM carrier frequency f_{pwm}, for any given tolerable V_{out} ripple swing.

Next, we study the behaviour of the 3×3 perceptron VAC circuit in Fig. 4.7 with dynamically changing power supply and input data. This is important because the PWM-based perceptron design targets unpredictable energy supply and variable V_{dd} situations. In the simulation using Cadence Analog Design tools the supply voltage changes from $3V$ (120% of nominal V_{dd}) to $2V$ (80% of nominal V_{dd} during a period of $2\mu s$. This substantial drop of V_{dd} may be caused by energy supply uncertainties. During this period, the VAC is subjected to four different sets of parametric values on its input PWM signals as shown in Table 5.2:

Fig. 5.15 depicts the simulation results from the parametric changes (Table 5.2). In

Table 5.2: Voltage, duty cycle and expected voltage ratios.

Phase (Time)	$\mathbf{DC_{in}}$	**W**	$\mathbf{V_{out}/V_{dd}}$
1 (0-2.5μs)	all $DC = 75\%$	$W_{1\|2} = 7$, $W_3 = 0$	50%
2 (2.5-5μs)	all $DC = 75\%$	$W_{1\|3} = 0$, $W_2 = 7$	25%
3 (5-7.5μs)	all $DC = 75\%$	$W_{1\|2\|3} = 7$	75%
4 (7.5-10μs)	all $DC = 75\%$	$W_{1\|3} = 7$, $W_2 = 0$	50%

this experiment, the input signals all have different f_{pwm} values, $100MHz$, $111MHz$, and $125MHz$ for $in1$, $in2$ and $in3$ respectively, and $C = 10pF$ and $R = 4.76k\Omega$.

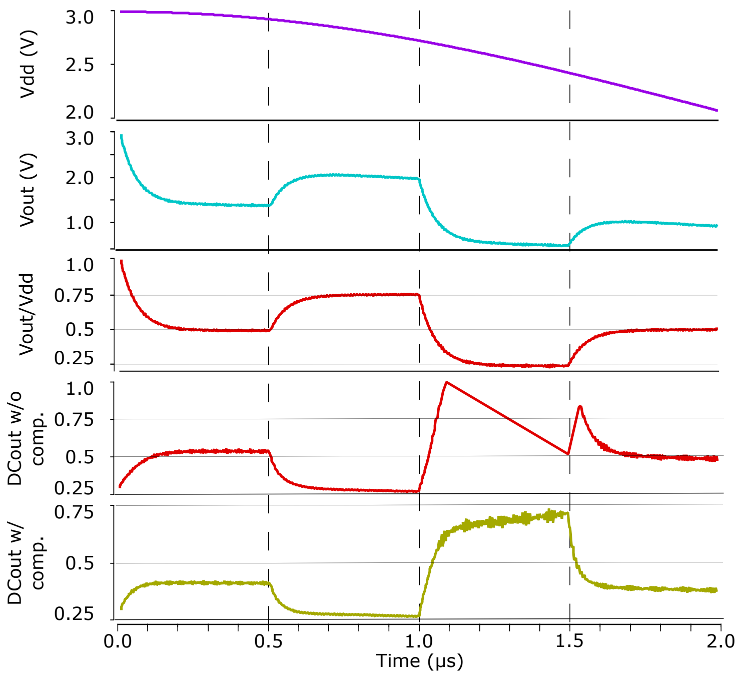

Figure 5.15: VAC operation under dynamic supply voltage and input data variations.

The figure shows that the VAC fulfils its design objective, i.e. providing functional correctness under variable energy supply conditions. The $\overline{V}_{out}/V_{dd}$ output values stabilize at the correct expected percentages after the charging/discharging processes complete, even though there exist inconsistencies in the different f_{pwm} signals and rather large V_{dd} changes.

However, the voltage-to-PWM converters proposed in section 4.3 cannot take advantage of this and fail to provide a correct PWM output from the correct $\overline{V}_{out}/V_{dd}$ output of the VAC. One version of the voltage-to-PWM converter does not have compensation to take the threshold voltage of transistors into account. This works correctly when V_{out} stays above the threshold voltage, but fails when V_{out} reduces below the threshold voltage. The second version of the voltage-to-PWM converter attempts to compensate for this by artificially keeping V_{out} above the threshold voltage, but the method of doing so is computationally incorrect, resulting in potentially significant errors in the PWM output duty cycle (DC_{out}). This shows that further research is needed for a computationally correct compensation for the voltage-to-PWM conversion stage.

It is also evident that the allowed rate of change for the input signals is determined by the RC value of design. For the VAC to function correctly, $\overline{V}_{out}/V_{dd}$ must be given enough time to settle before an input change can be allowed. In this instance, $\tau = R \cdot C = 47.6ns$ and the input change period is $500ns$ or $\approx 10\tau$, giving plenty time for $\overline{V}_{out}/V_{dd}$ to settle.

5.5 Perceptron design trade-off analysis

Another design-space exploration made possible by this dynamic analysis is the investigation of various trade-offs. For instance, Fig. 5.11 indicates that the computational speed is faster for smaller perceptron RC values. This is because the establishment of the perceptron arithmetic result $\overline{V}_{out}/V_{dd}$ comes earlier if V_{out} charging/discharging finishes earlier. However, Fig. 5.14 indicates that smaller RC values require higher PWM carrier frequencies, which may lead to implementation limitations on the design of the PWM frequency generating subsystem and power dissipation. In addition, the perceptron arithmetic result $\overline{V}_{out}/V_{dd}$ is held in the charge of the output capacitor C, for which a larger C will provide a more stable $\overline{V}_{out}/V_{dd}$ for longer and better support the reading of this value by subsequent perceptrons or output circuits. There are also limitations on the upper and lower values for both R and C for on-chip implementations.

In addition to computation speed, tolerable PWM frequency range, power dissipation and $\overline{V}_{out}/V_{dd}$ resilience, another design metric is energy consumption, which is usually measured in energy per operation, or the energy required for completing a single operation. As an example, we define a single operation as having continuously charged or discharged the output capacitor for 5 time constants ($5\tau = 5RC$). This arbitrary definition achieves 98% of the target value of $\overline{V}_{out}/V_{dd}$ for a 2% charging/discharging error. If this level of precision is not needed, the per operation time definition can be shortened. However, the choice of the operation completion time does not affect the generality of this method.

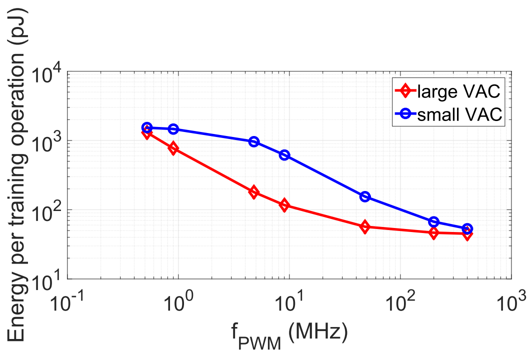

Figure 5.16: Energy per operation at $f_{pwm} = 2 \times \min(f_{pwm})$.

From Fig. 5.16 it can be seen that the energy per operation is lower for larger RC values and higher for smaller RC values at the same PWM carrier frequencies. According to Fig. 5.11, larger RC values cause longer computation time (each operation takes longer time) which increases energy per operation. However, as evidenced from Fig. 5.13 they also result in smaller ripple voltage swings reducing dynamic power dissipation. Evidently, for the systems observed in these experiments, the latter factor outweighs the former. Being able to create quantitative comparisons using dynamic analysis makes it possible to study design trade-offs in detail. This is important because there are no general rules in some of these trade-offs. For instance, the power dissipation difference caused by the same ripple voltage swing difference may depend on the perceptron arithmetic specification.

Table 5.3 summarizes the impact of choosing larger or smaller RC values in terms of speed, power, area and the range of permitted f_{pwm} choices.

Table 5.3: Speed, power, area and frequency range trade-offs.

RC value	**Speed**	**Power**	**Area**	f_{pwm} **choice range**
Larg RC	lower	lower	higher	wider
Small RC	higher	higher	lower	narrower

5.6 Validation and analysis of PWM-coded neural network

This section explores an NN system built using the proposed PWM-based perceptron. This NN is designed for solving the MNIST problem and has the structure shown in Figure 4.10. Firstly a high-level model of the perceptron is constructed so that analysis can be carried out in MATLAB, at a higher level than analogue VLSI simulations, which is impractical for systems of this size. Then this model is used in MATLAB investigations on system properties to validate our NN-design approach.

In this section, the duty cycle output of the perceptron is modelled in the form of a mathematical equation with parameters. Then, the model and the voltage to PWM converter itself are studied to verify that the device approximately incorporates the capped ReLU AF.

The equations in Section 4.5 pertain to ideal cases. These can be used for comparing with how the implemented perceptron actually delivers. In order to make this comparison at the whole system level, we need to generate a high-level mathematical model based on observations made whilst experimenting with the perceptron circuit at a low level.

We experimented in the Cadence Analog Design Environment with a single perceptron, two perceptrons connected in series, mimicking the simplest two-layer NN, and three perceptrons connected in series, emulating the simplest NN with a depth of 3. This is as far as analogue VLSI simulations could practically go, as the three-layer study took many hours on a competitively specified server machine.

The outputs of these perceptron connection topologies are shown in Figure 5.17. In the ideal case, the input and output duty cycles should be equal when every weight is at the maximum value (dashed line). However, there is a non-linear relationship between the input and output of the single perceptron (red line) and the degree of non-linearity increases when the depth of the NN is increased (blue and green lines). In addition, the output begins to saturate in the last (third) stage when the input (DC_{sum}) reaches 0.85.

To model this relationship, a third-order polynomial equation, which is easy to differentiate, is curve-fitted to the response of the single perceptron using basic regression in MATLAB. The result is shown in equation 5.2. Note that the saturation point of the model is set at the maximum output duty cycle, which is 98%. Then, the model is connected in the same two- and three-stage series topologies as in the Cadence experiments and their outputs are plotted in Figure 5.18 - 5.20, together with the relevant Cadence results for comparison. All figures show that this model accurately estimates the input-output relationship of the perceptron. The accuracy can be obtained as the R-squared values of stages one, two and three, which are 99.88%, 99.33% and 97.66% respectively.

$$DC_{out} = 107.27V_{Cout}^3 - 53.25V_{Cout}^2 + 52.92V_{Cout} + 13.44 \tag{5.2}$$

$$f(x) = \begin{cases} 0 & , x < 0 \\ DC_{out} & , 0 < x < 1 \\ 0.98 & , x > 0.98 \end{cases} \tag{5.3}$$

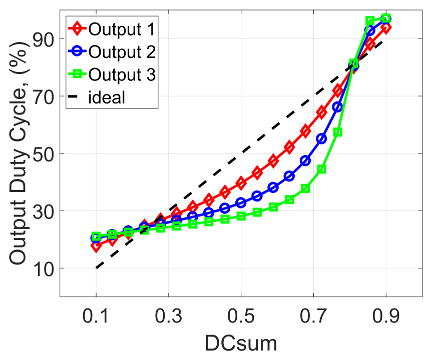

Figure 5.17: PWM output.

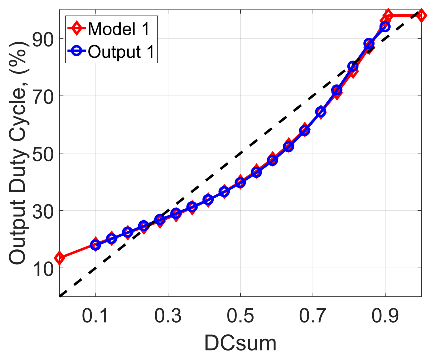

Figure 5.18: Output vs model stage 1.

We use the perceptron model in equation 5.2 to assemble models of large-size MNIST NNs then simulate these systems in full in MATLAB. The model plot is shown in

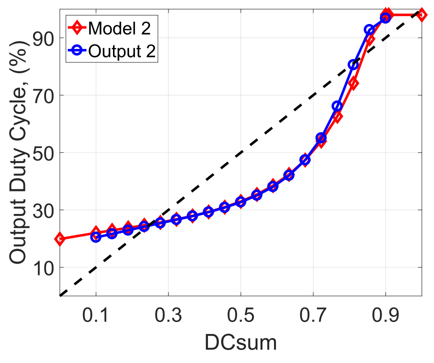

Figure 5.19: Output vs model stage 2.

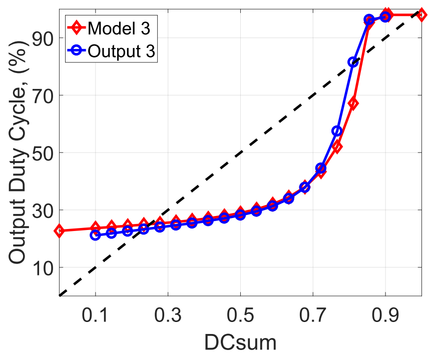

Figure 5.20: Output vs model stage 3.

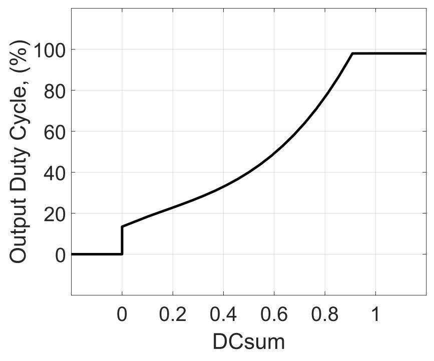

Figure 5.21: PWM perceptron output model function.

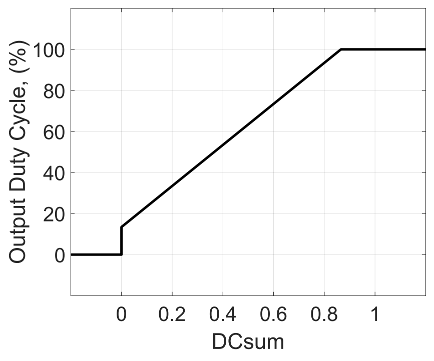

Figure 5.22: Capped ReLU function with PWM-like offset.

Fig. 5.21. We also create a capped ReLU function with offset (Oft.ReLU), expressed in equation 5.4. As can be seen in the equation, this offset ReLU function takes the constant 13.44 from the perceptron model in equation 5.2 to have the same offset nonlinearity as the perceptron model in Fig. 5.21, but otherwise has a similar straight-line behaviour to the non-offset capped ReLU in Fig. 4.13. The plot of this offset ReLU can be found in Fig. 5.22.

This function is used to investigate whether the step nonlinearity or the curvature nonlinearity higher in the curve of equation 5.2 is more important when it comes to NN performance, through comparisons with both the perceptron model in equation 5.2 and the capped ReLU function without offset in Fig. 5.22.

$$f(x) = \begin{cases} 0 & ,x < 0 \\ DC_{out} + 13.44 & ,0 < x < 1 \\ 1 & ,x > 0.86.56 \end{cases} \tag{5.4}$$

There are two groups of simulations using MATLAB: without/with limiting the maximum weight. All implement the training procedure described in Figure 4.11 for the MNIST problem, which is selected as our benchmark. Without defining the maximum weight, the weight is adjusted freely like the basic FP training while the proposed NN is demonstrated by the limited weight simulation. Four AFs: ReLU, capped ReLU (Cap.ReLU), capped ReLU with offset (Oft.ReLU) and PWM perceptron (PWM percept.) are applied in three network configurations: two (784/10), three (784/300/10) and four (784/300/100/10) layers.

The PWM perceptron AF is implemented by the PWM perceptron on its own unmodified - the justification being that it may be considered as an approximation of the capped ReLU (cf. Figure 5.21 and Figure 4.13).

As can be seen from this data, these systems being simulated include hundreds of perceptrons and are well beyond analysing in the VLSI design domain.

For the unlimited weight simulation, the learning rate is swept from 0.001 to 0.1 for every AF. The configurations with the smallest error are listed in Table 5.4. The limited weight simulation is carried out in the same way except that the initial weight is swept from ±1 to ±255. This is because a small weight causes a small update which can keep the final weight within the specified range. Then, the configurations with higher than 90% accuracy are selected to sweep their maximum weights down until the accuracy is nearly equal to 90%. This is to save the circuit area by using the smallest bit-width. The simulation results are listed in Table 5.5.

In the single perceptron, two-perceptron and three-perceptron experiments, both the resultant model and the Cadence simulations indicate that the voltage to PWM converter, without modifications, may serve as an approximate capped ReLU AF, qualitatively. In addition, the three-perceptron, three-stage full analogue simulation analysis shows

Table 5.4: Simulation result of the floating-point weight neural network.

No.	No. Perceptron	Activation Function	Learning Rate	Error
1	784/10	ReLU	0.010	1.40
2	784/10	Cap.ReLU	0.008	1.75
3	784/10	Oft.ReLU	0.009	5.09
4	784/10	PWM percept.	0.004	8.54
5	784/300/10	ReLU	0.040	1.63
6	784/300/10	Cap.ReLU	0.009	1.91
7	784/300/10	Oft.ReLU	0.002	79.54
8	784/300/10	PWM percept.	0.004	27.01
9	784/300/100/10	ReLU	0.040	2.07
10	784/300/100/10	Cap.ReLU	0.010	3.60
11	784/300/100/10	Oft.ReLU	0.010	90.20
12	784/300/100/10	PWM percept.	0.090	79.07

that the single-perceptron MATLAB model can be used in multi-stage system analysis without worrying about the fidelity of high-level MATLAB models when multiple layers of perceptrons exist in a system.

Quantitatively, however, the use of a nonlinear perceptron to approximate linear behaviour becomes increasingly problematic when the depth of the network increases, as the non-linearity accumulates. This is shown to be true by the subsequent whole-NN experiments.

The unlimited weight simulation result in Table 5.4 gives us traditional NN examples which contain FP weights. It shows that both ReLU and capped ReLU functions give less than 4% errors at every depth.

The results for the PWM perceptron AF are similar to those from the capped ReLU with offset. They obtain small error rates at the shallowest NN depth, while the capped ReLU without offset outperforms all others at every depth. This confirms that it is mainly

Table 5.5: Simulation result of the integer weight neural network.

No.	**No. Perceptron**	**Activation Function**	**Learning Rate**	**Initial Weight**	**Max Weight**	**Error**
1	784/10	ReLU	0.030	±3	±31	9.28
2	784/10	Cap.ReLU	0.040	±3	±63	6.12
3	784/10	Oft.ReLU	0.004	±7	±255	7.10
4	784/10	PWM percept.	0.030	±1	±255	9.98
5	784/300/10	ReLU	0.020	±255	±255	79.49
6	784/300/10	Cap.ReLU	0.020	±255	±255	79.49
7	784/300/10	Oft.ReLU	0.020	±3	±255	18.35
8	784/300/10	PWM percept.	0.010	±15	±255	25.17
9	784/300/100/10	ReLU	0.010	±31	±255	88.50
10	784/300/100/10	Cap.ReLU	0.010	±31	±255	88.50
11	784/300/100/10	Oft.ReLU	0.020	±127	±255	64.09
12	784/300/100/10	PWM percept.	0.010	±63	±255	53.25

the step transition at $DC_{sum} = 0$ that causes the convergence problem in our NN, more than the curvature nonlinearity higher in the curve of Fig. 5.21. Therefore, compensating the circuit to shift the output duty cycle back to 0% appears to be a promising route of investigation. This will be a subject in our future work.

Table 5.5 shows the results with weight limitations. All results at two-layer NN are worse than the ones in Table 5.4 due to the weight capping and rounding, except for the PWM perceptron, which does better. The PWM perceptron continues to perform better at higher layer depths than in the unlimited weight case, confirming an advantage for it when weight is limited and represented by an integer. However, it again fails to approximate the ReLU function quantitatively at higher layer depths, by returning obviously better performances than the latter.

Our proposed PWM-based perceptron's non-linearity, as well as range limits of the PWM duty cycle (it starts from $\sim 13\%$ rather than 0% as shown in Figs. 5.18 and 5.21), makes it less suitable for deeper NNs. Its lack of support for negative values also limits its wider usability as the fundamental element of NNs, without further modification to better incorporate established AFs. The approximation of ReLU, although qualitatively promising, proves to be quantitatively unsatisfactory at higher NN depths, although in some cases this results in the perceptron's AF being better than the ReLU AF. The high error rate also comes from the resolution loss in basic weight rounding which may be solved by implementing a rounding technique and PF inference quantization presented in [82] and [83] respectively.

In other words, even if the negative value representation problem is solved, computing AFs in the analogue and relative temporal domains remains a challenge that must be solved. As a result, a future work direction is the incorporation of more general arithmetic operations in these domains, which is needed to improve the accuracy of AF implementations.

A related and interesting unsolved problem is the 'comparing with target vector' function in Figure 4.10, which is currently relegated to external controllers. It is a duty cycle in and digital out block and can potentially be designed by extending the methods in this work.

Table 5.6 summarises our design compared to related work. The work in [84] quantizes the entire NN and yields the lowest error. However, it is designed for a digital-based processing unit which contains the CPU-memory bottleneck issue implying extra power budget and latency. Furthermore, real power measurement is missing as it estimates the power consumption from the literature. Weight rounding methods are proposed in [82]. Although they achieve the second lowest error, they require binarized input data which is a challenge for analogue applications. Moreover, it does not include an investigation of hardware implementation. The memristor crossbar NN in [85] acquires the lowest accuracy with the highest power consumption even its input image size is reduced. Charge trap transistor-based NN which performs MNIST classification from the original data is presented in [77]. It mainly aims to save power and requires a specific CMOS technology to fabricate the special transistors.

Even though the error of our design is higher, it is still within the same order of magnitude, and we have yet to investigate more sophisticated techniques for compensating

Table 5.6: Performance comparison.

Work	**Weight type**	**MNIST**	**NN conf.**	**Error**	**Power (μW)**	**Power elastic**	**Hardware**
[84]	integer	quantized	WAGE	0.4	n/a	N	MCU
[82]	integer	binarized	MLP	$< 3\%$	n/a	N	n/a
[85]	n/a	reduced & quantized	MLP	13.5%	53,000 (NN)	N	memristor crossbar
[77]	fixed-point	original	MLP	$\sim$5%	14,800 (NN)	N	Spec. CMOS
This	integer	original	MLP	$< 10\%$	14-1,080 (VAC)	Y	Std. CMOS

the voltage to PWM part to improve the duty cycle coverage and eliminate the step nonlinearity at $DC_{sum} = 0$, which promises to reduce error. We also do not investigate beyond standard CMOS technology as that is out of scope for this investigation. Our focus is on tolerating unstable and unpredictable power supply voltages, a necessity for energy autonomous AI devices. In this regard, this solution is unique as existing research in the literature invariably requires stable and known voltages and operating frequencies. Note that the power figures in Table 5.6 are not directly comparable. The figure for [82] is obtained from measurements carried out on a fabricated chip at the 28nm technology node with special non-CMOS transistor techniques aiming to showcase the advantage of that technology in low-power operations, whilst that for this work is obtained from simulating one VAC at the 65nm technology node with a deliberately high (2.5V) nominal supply voltage to facilitate studying voltage instability scenarios. This is similar to [85] where the circuit is implemented with non-CMOS technology. A fair power consumption comparison with [82] and [85] is not yet possible without fabricating and testing real chips, preferably at the same VLSI technology node, as whole-NN simulations at the VLSI level is not practical. In addition, the power figures for [82] and [85] are themselves not comparable with each other as [85] covers the entire system including peripherals and [82] covers the NN engine only.

5.7 Summary

In this chapter, we simulated the PWM-based perceptron. The perceptron's arithmetic unit design is shown to fully accomplish its design aim of power and frequency resilience. It is also shown to be working within reasonable boundaries in pragmatic applications, especially in NNs of limited depth which are nevertheless of significant size. Future improvements should be concentrated on improving its linearity, its threshold voltage independence and its representation of negative values to increase its usability of NNs of greater depth and more sophisticated AFs.

Our design methods, supported by extensive analysis and validations, have proven the original hypothesis, and demonstrated the following features:

- *power elasticity* and *resilience* across a dynamic range of V_{dd} (statistically varying by $5\times$) and f (statistically varying by up to 6 orders of magnitude). Such elasticity is achieved for the VAC without requiring any voltage regulator circuit and clock pairing between V_{dd} and f. Dynamic power supply variations also show good resilience. However, further compensation will be required at lower voltages to avoid large errors at the whole-perceptron level;
- minimal use of additional analogue (ideally, only passive components) electronics, coupled with low-complexity PWM-coded arithmetic using primarily digital components, making our approach highly power efficient and suitable for low-cost fabrication;
- extensive validation and analysis using multi-layer PWM-coded NNs show good scalability of the proposed approach; however, deeper NNs may need circuit-level compensation after each layer or high-precision representation techniques to improve the overall accuracy and efficiency.

Chapter 6

Conclusions

6.1 Contributions

In this research, we approached the problem of IoT energy supply from two main directions: improving the efficiency and stability of power regulators, and increasing the power elasticity of the computation unit.

For IoT applications the most suitable type of power regulators is SCC. We discuss the major challenges of on-chip SCC and demonstrate the methods of solving these challenges. In sections 3.1 and 3.2 we discuss the methods of designing the self-timed SCC controllers, which resolve the problem of the shoot-through currents. Another issue with bottom plate parasitic capacitance is resolved in section 3.3, where we apply the method of parasitic charge redistribution to the self-oscillating SCC. The correctness of the proposed methods is validated by Cadence simulations. The simulations also show the increasing of efficiency of these methods.

We use a Neural Network as a typical IoT computation unit. We aim to improve its resilience to supply voltage variations in order to reduce the complexity and cost of the power regulator circuit.

We propose the first mixed-signal (analogue/digital/relative temporal) perceptron design using the principles of PWM. Central to our design are a number of parallel inverters that suitably transcode the input-weight pairs from the spatial domain to the relative temporal domain. This approach aims to deliver high resilience to amplitude and frequency variations in the supply voltage, exploiting the fact that PWM-based solutions are typically agnostic to such variations.

Another advantage of the proposed design is its simplicity. Whilst conventional implementations of the perceptron require complex logic to perform “lication and addition, the proposed approach uses only one gate (either an inverter or a two-input NAND) per bit for every input. Thus, for the 3×3 weighted addition VAC, we used only 54 transistors. This significantly reduces the logic requirement and, therefore, the power consumption of the entire device.

Extensive experimentation on the perceptron and its use in neural networks of relatively significant sizes helps to explore the perceptron design’s advantages, usability and limitations. Also through experimental studies, design improvements are found which further strengthen the perceptron’s case. These experimental explorations also lead to further insights into the design and provide guidance on potential future work.

6.2 Future work

This thesis gives the possibilities for future research in both power regulation and computation parts.

The methods proposed in sections 3.1 and 3.2 allow designing self-timed controllers for SCC. The steps described in the methods can be automated. Ideally, a user would have to input only the required ratios and topologies of a designed SCC, and an algorithm based on this method would produce a completed implementation of the corresponding controller.

While the controller generated in section 3.2 is quite large, its size can be reduced. One of the possible ways of doing this is by using the David cells [74] in the controller part that interfaces the delay elements. Another optimization is to simplify the operation of the signal *ratio_req* in such a way that it does not interrupt the generation of the switch control signals. These optimization possibilities are a subject for future research.

The method of design of self-oscillating SCC with parasitic charge redistribution proposed in section 3.3 can be expanded to the more complex charge redistribution algorithms such as in [19]. However, these algorithms would require additional logic, which would have its own parasitic capacitance.

There are some other ways of applying the parasitic charge redistribution method to the self-oscillating SCC. For example, we can have two parallel self-oscillating SCC that operate in a counter-phase and share their parasitic charge between each other. This method is more similar to [20], and it does not require the additional store capacitor. However, the logic in this method would be more complicated and would consume more power in comparison to the method described in this paper.

The method of PWM-based perceptron design demonstrated the high degree of resilience to voltage and frequency variations in the VAC part. However, the voltage to PWM converter based on the ring oscillator designed in section 4.3 is less resilient, especially, to the dynamic variations. The main issue of the proposed solution is that the transistor's threshold voltage remains constant with the dynamically changing supply voltage. The question of finding a cost-effective circuit solution for voltage-to-PWM conversion for a wide dynamic range of voltages (covering near and sub-threshold voltage level) remains an open challenge.

The VAC design proposed in section 4.2 can also be improved. If not only the input but also the weight was PWM-coded, the number of cells would be significantly reduced. In that case, the adder would require only one cell per input. However, this design will require more complicated feedback circuitry.

Bibliography

[1] K. Schwab, *The fourth industrial revolution*. London : Portfolio Penguin, 2017.

[2] T. V. Breussegem and M. Steyaert, “Cmos integrated capacitive dc-dc converters.” Springer, 2013.

[3] G. Villar-Piqué, H. J. Bergveld, and E. Alarcón, “Survey and benchmark of fully integrated switching power converters: Switched-capacitor versus inductive approach,” *IEEE Transactions on Power Electronics*, vol. 28, no. 9, pp. 4156–4167, 2013.

[4] L. Salem and P. Mercier, “A 45-ratio recursively sliced series-parallel switched-capacitor dc-dc converter achieving 86% efficiency,” 09 2014, pp. 1–4.

[5] A. Kushnerov and S. Ben-yaakov, “Unified algebraic synthesis of generalized fibonacci switched capacitor converters,” 09 2012.

[6] W. Jung, D. Sylvester, and D. Blaauw, “12.1 a rational-conversion-ratio switched-capacitor dc-dc converter using negative-output feedback,” vol. 2016, 01 2016, pp. 218–219.

[7] S. Unger, “Asynchronous sequential switching circuit,” 1969.

[8] D. Sokolov, V. Dubikhin, V. Khomenko, D. Lloyd, A. Mokhov, and A. Yakovlev, “Benefits of asynchronous control for analog electronics: Multiphase buck case study,” 03 2017, pp. 1751–1756.

[9] D. Sokolov, V. Khomenko, A. Mokhov, A. Yakovlev, and D. Lloyd, “Design and verification of speed-independent multiphase buck controller,” 05 2015, pp. 29–36.

[10] O. Benafa, A. Ogweno, D. Shang, and A. Yakovlev, "Design of a dco based on worst-case delay of a self-timed counter and a digitally controllable delay path," 06 2016, pp. 1–4.

[11] H. Ning and Z. Wang, "Future internet of things architecture: Like mankind neural system or social organization framework?" *IEEE Communications Letters*, vol. 15, no. 4, pp. 461–463, 2011.

[12] M. Evzelman and S. Ben-yaakov, "The effect of switching transitions on switched capacitor converters losses," 11 2012.

[13] A. Kushnerov, "High-efficiency self-adjusting switched capacitor dc-dc converter with binary resolution," Ph.D. dissertation, 08 2009.

[14] A. Kushnerov and S. Ben-Yaakov, "Unified algebraic synthesis of generalised Fibonacci switched capacitor converters," *IET Power Electronics*, vol. 7, no. 3, pp. 540–544, 2014.

[15] W. Jung, S. Oh, S. Band, Y. Lee, Z. Foo, G. Kim, Y. Zhang, D. Sylvester, and D. Blaauw, "An ultra-low power fully integrated energy harvester based on self-oscillating switched-capacitor voltage doubler," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 12, 2014.

[16] M. J. Turnquist, M. Hiienkari, J. Makipaa, and L. Koskinen, "A fully integrated self-oscillating switched-capacitor dc-dc converter for near-threshold loads," in *2015 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov 2015, pp. 1–4.

[17] M. Turnquist, M. Hiienkari, J. Mäkipää, and L. Koskinen, "A fully integrated 2:1 self-oscillating switched-capacitor dc-dc converter in 28 nm utbb fd-soi," *Journal of Low Power Electronics and Applications*, vol. 6, p. 17, 09 2016.

[18] M. Fojtik, D. Kim, G. Chen, Y.-S. Lin, D. Fick, J. Park, M. Seok, M.-T. Chen, Z. Foo, D. Blaauw, , and D. Sylvester, "A millimeter-scale energy-autonomous sensor system with stacked battery and solar cells," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 801–813, March 2013.

[19] N. Butzen and M. Steyaert, "Scalable parasitic charge redistribution: Design of high-efficiency fully integrated switched-capacitor dc-dc converters," *IEEE Journal of Solid-State Circuits*, vol. PP, pp. 1–11, 10 2016.

[20] T. Andersen, F. Krismer, J. Kolar, T. Toifl, C. Menolfi, L. Kull, T. Morf, M. Kossel, M. Brandli, P. Buchmann, and P. Francese, "A 4.6w/mm2 power density 86% efficiency on-chip switched capacitor dc-dc converter in 32 nm soi cmos," pp. 692–699, 01 2013.

[21] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*, 01 2001.

[22] J. Beaumont, A. Mokhov, D. Sokolov, and A. Yakovlev, "Compositional design of asynchronous circuits from behavioural concepts," 09 2015, pp. 118–127.

[23] J. Audy, "Navigating the path to a successful ic switching regulator design," in *Tutorial at IEEE International Solid-State Circuits Conference (ISSCC)*, 2008.

[24] L. Lavagno, *Algorithms for Synthesis and Testing of Asynchronous Circuits*, 01 1993.

[25] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli, "Algorithms for synthesis of hazard-free asynchronous circuits," 02 1995.

[26] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic Synthesis of Asynchronous Controllers and Interfaces*, 01 2002, vol. 8.

[27] V. Khomenko, M. Koutny, and A. Yakovlev, "Detecting state encoding conflicts in stg unfoldings using sat." *Fundam. Inform.*, vol. 62, pp. 221–241, 01 2004.

[28] W. Belluomini, C. Myers, and H. Hofstee, "Verification of delayed-reset domino circuits using atacs," *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 04 1999.

[29] W. Bartky and D. Muller, "A theory of asynchronous circuits," in *International Symposium of the Theory of Switching*, 1959.

[30] C. Petri, "Kommunikation mit automaten (communicating with automata)," vol. 3, 01 1962.

[31] E. Chung and L. Kleeman, "Avoiding hazards in self-timed digital circuits derived from signal transition graphs," *Australian Telecommunications Research*, vol. 29, pp. 25–38, 01 1995.

[32] L. Rosenblum and A. Yakovlev, "Signal graphs: From self-timed to timed ones." 01 1985, pp. 199–206.

[33] J. Beaumont, "Compositional circuit design with asynchronous concepts," 2018.

[34] I. Poliakov, V. Khomenko, and A. Yakovlev, "Workcraft – a framework for interpreted graph models," vol. 5606, 06 2009, pp. 333–342.

[35] D. Sokolov, V. Khomenko, and A. Mokhov, "Workcraft: Ten years later," in *This asynchronous world. Essays dedicated to Alex Yakovlev on the occasion of his 60th birthday*, A. Mokhov, Ed. Newcastle University, 2016, available online http://async.org.uk/ay-festschrift/paper25-Alex-Festschrift.pdf.

[36] "WORKCRAFT homepage," http://workcraft.org/.

[37] V. Khomenko, "Model checking based on prefixes of petri net unfoldings," Ph.D. dissertation, 04 2003.

[38] M. Schaefer, W. Vogler, D. Wist, and R. Wollowski, "Avoiding irreducible csc conflicts by internal communication," vol. 95, 07 2008, pp. 3 – 12.

[39] V. Khomenko, M. Koutny, and A. Yakovlev, "Logic synthesis for asynchronous circuits based on stg unfoldings and incremental sat." *Fundam. Inform.*, vol. 70, pp. 49–73, 03 2006.

[40] N. Javaid, A. Sher, H. Nasir, and N. Guizani, "Intelligence in iot-based 5g networks: Opportunities and challenges," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 94–100, October 2018.

[41] A. Biswas and A. P. Chandrakasan, "Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 488–490.

[42] M. Chen, Y. Miao, X. Jian, X. Wang, and I. Humar, "Cognitive-LPWAN: Towards Intelligent Wireless Services in Hybrid Low Power Wide Area Networks," *arXiv e-prints*, p. arXiv:1810.00300, Sep 2018.

[43] M. T. Sharbati, Y. Du, J. Torres, N. D. Ardolino, M. Yun, and F. Xiong, "Low-power, electrochemically tunable graphene synapses for neuromorphic computing," *Advanced Materials*, vol. 30, no. 36, p. 1802353, 2018.

[44] E. O. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, vol. 5, pp. 52 – 68, 2018.

[45] R. Shafik and A. Yakovlev, *Chapter: From Power-Efficient to Power-Driven Computing, in Many-Core Computing: Hardware and Software.* Ed: G. Merrett and B. M. Al-Hashimi, IET, 2019.

[46] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez, G. V. Merrett, and B. M. Al-Hashimi, "Learning transfer-based adaptive energy minimization in embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 6, pp. 877–890, June 2016.

[47] I. Qiqieh, R. Shafik, G. Tarawneh, D. Sokolov, and A. Yakovlev, "Energy-efficient approximate multiplier design using bit significance-driven logic compression," in *Proceedings of the Conference on Design, Automation & Test in Europe*, ser. DATE '17. European Design and Automation Association, 2017, pp. 7–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3130379.3130382

[48] R. Shafik, A. Yakovlev, and S. Das, "Real-power computing," *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1445–1461, Oct 2018.

[49] S. Beeby and N. M. White, *Energy harvesting for autonomous systems*. Artech House, 2010.

[50] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695.

[51] A. Suleiman, Y.-H. Chen, J. Emer, and V. Sze, "Towards closing the energy gap between hog and cnn features for embedded vision," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[52] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, Jan 2018.

[53] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[54] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA, USA: PWS Publishing Co., 1996.

[55] E. Wilson and D. W. Tufts, "Multilayer perceptron design algorithm," in *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, Sep. 1994, pp. 61–68.

[56] H. Adeli and C. Yeh, "Perceptron learning in engineering design," *Computer-Aided Civil and Infrastructure Engineering*, vol. 4, no. 4, pp. 247–256, 1989. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8667.1989.tb00026.x

[57] S. Hung and H. Adeli, "A model of perceptron learning with a hidden layer for engineering design," *Neurocomputing*, vol. 3, no. 1, pp. 3 – 14, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0925231291900165

[58] Byeongjang Jeong and Yong Hoon Lee, "Design of weighted order statistic filters using the perceptron algorithm," *IEEE Transactions on Signal Processing*, vol. 42, no. 11, pp. 3264–3269, Nov 1994.

[59] Wang Qinruo, Yi Bo, Xie Yun, and Liu Bingru, "The hardware structure design of perceptron with fpga implementation," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, vol. 1, 2003, pp. 762–767 vol.1.

[60] Y. hsin Chen, T.-J. Yang, and J. S. Emer, "Understanding the limitations of existing energy-efficient design approaches for deep neural networks," in *Energy*, vol. 2, no. L1, 2018, p. L3.

[61] R. W. Keyes and R. Landauer, "Minimal energy dissipation in logic," *IBM Journal of Research and Development*, vol. 14, no. 2, pp. 152–157, 1970.

[62] M. de Prado, M. Denna, L. Benini, and N. Pazos, "Quenn: Quantization engine for low-power neural networks," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*. ACM, 2018, pp. 36–44.

[63] A. Yakovlev, "Energy-modulated computing," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[64] D. Shang, X. Zhang, F. Xia, and A. Yakovlev, "Asynchronous design for new on-chip wide dynamic range power electronics," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014, pp. 1–6.

[65] R. LiKamWa, Y. Hou, Y. Gao, M. Polansky, and L. Zhong, "Redeye: Analog convnet image sensor architecture for continuous mobile vision," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 255–266.

[66] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, 2016, pp. 262–263.

[67] L. Rosenblum and A. Yakovlev, "Signal graphs: from self-timed to timed ones," in *IEEE int. Workshop on Timed Petri Nets*, 1985.

[68] G. Kim, M.-K. Kim, B.-S. Chang, and W. Kim, "A low-voltage, low-power cmos delay element," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 7, pp. 966–971, 1996.

[69] C.-Y. Yu, C.-C. Chung, C.-J. Yu, and C.-Y. Lee, "A low-power dco using interlaced hysteresis delay cells," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 673–677, 2012.

[70] J. W. Peterson, "Solid state transmission gate," Patent US4 473 761 (A), 09 25, 1984.

[71] B. C. Grugett, "Biasing circuit for reducing body effect in a bi-directional field effect transistor," Patent US5 767 733 (A), 06 16, 1998.

[72] F. Maloberti, "Charge injection compensation," in *Analog Design for CMOS VLSI Systems*. Springer, 2001.

[73] T. Andersen, F. Krismer, J. Kolar, T. Toifl, C. Menolfi, L. Kull, T. Morf, M. Kossel, M. Brandli, P. Buchmann, and P. Francese, "A deep trench capacitor based 2:1 and 3:2 reconfigurable on-chip switched capacitor dc-dc converter in 32 nm soi cmos," 03 2014, pp. 1448–1455.

[74] R. David, "Modular design of asynchronous circuits dened by graphs," *IEEE Transactions on Computers*, vol. 26, no. 8, pp. 727–737, 1977.

[75] I. Vaisband, M. Azhar, E. G. Friedman, and S. Köse, "Digitally controlled pulse width modulator for on-chip power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2527–2534, 2014.

[76] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[77] Y. Du, L. Du, X. Gu, J. Du, X. S. Wang, B. Hu, M. Jiang, X. Chen, S. S. Iyer, and M. F. Chang, "An analog neural network computing engine using cmos-compatible charge-trap-transistor (ctt)," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[78] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[79] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 3517–3521.

[80] "CADENCE homepage," http://www.cadence.com/.

[81] "MATLAB homepage," http://www.mathworks.com/.

[82] L. K. Muller and G. Indiveri, "Rounding methods for neural networks with low resolution synaptic weights," *eprint arXiv:1504.05767*, p. arXiv:1504.05767, 2015. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2015arXiv150405767M

[83] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*, ser. arXiv e-prints, 2017. [Online]. Available: https://ui.adsabs.harvard.edu/#abs/2017arXiv171205877J

[84] S. Wu, G. Li, F. Chen, and L. Shi, *Training and Inference with Integers in Deep Neural Networks*, ser. arXiv e-prints, 2018. [Online]. Available: https://ui.adsabs.harvard.edu/#abs/2018arXiv180204680W

[85] H. Jiang, K. Yamada, Z. Ren, T. Kwok, F. Luo, Q. Yang, X. Zhang, J. J. Yang, Q. Xia, Y. Chen, H. Li, Q. Wu, and M. Barnell, "Pulse-width modulation based dot-product engine for neuromorphic computing system using memristor crossbar array," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Conference Proceedings, pp. 1–4.