The Design of Class-EF$_2$ Inverters using Multi-Objective Optimization

By © Andrew Peddle

A Thesis submitted

to the School of Graduate Studies in partial fulfillment of the

requirements for the degree of

Master of Electrical Engineering

Memorial University of Newfoundland

May 2022

St. John's Newfoundland and Labrador

ABSTRACT

This thesis explores the use of multi-objective optimization algorithms for the design of high frequency inverters. A state-space model of the ideal Class-EF_2 inverter is derived and its accuracy is validated by MATLAB and LTSpice simulation. The model is then applied to the Multi-Objective Genetic Optimization (MOGO) and Multi-Objective Particle Swarm Optimization (MOPSO) algorithms to design three inverters with varying output power, frequency, and load requirements. The final designs are compared with analytical results to verify the optimization-based design approach. The ideal state-space model is then extended to include the parasitic elements of components, and further extended to consider the internal resistances and capacitances of the switch. These new models are applied to the MOGO and MOPSO algorithms to design the same three inverters as the ideal case. The final designs are simulated in LTSpice to evaluate their performance, and comparisons are presented to demonstrate the effects of the parasitic elements and switching dynamics on the component values and overall circuit operation. A design example is also presented to demonstrate the design of a 6.78 MHz, 100W, 20 Ω Class-EF_2 inverter, and provide designers with insight on how to apply the proposed design approach to their own designs.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. John Quaicoe for his patience, guidance, and financial support which allowed me to perform this research. Without him, completion of this thesis would not have been possible.

I would also like to thank Solace Power Inc. for providing me with a space to work and offering their knowledge and expertise.

Finally, I would like to thank my parents and my partner for their love and support throughout my studies. I couldn't have done it without you.

LIST OF TABLES

Table 2-1	Circuit Parameters for MATLAB and LTSpice Comparison	11
Table 2-2	Optimization Variables for the MOGO Based Design Approach	17
Table 2-3	Optimization Boundary Conditions	18
Table 2-4	Parameters for Tested Design Cases	21
Table 2-5	Optimal Design Factors for the Ideal Class-EF_2 Inverter	22
Table 2-6 (a)	Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case I	23
Table 2-6 (b)	Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case II.	24
Table 2-6 (c)	Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case III.	25
Table 2-7	MOGO Optimized Values for the Ideal Class-EF_2 Inverter: Design Case I	26
Table 2-8	MOGO Circuit Parameters for the Ideal Class-EF_2 Inverter: Design Case I	26
Table 2-9	MOGO Circuit Parameters for the Ideal Class-EF_2 Inverter: Design Case II and Design Case III	27
Table 2-10 (a)	Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case I	31
Table 2-10 (b)	Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case II.	32
Table 2-10 (c)	Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case III.	33
Table 2-11	MOPSO Optimized Circuit Parameters for the Ideal Class-EF_2 Inverter: All Design Cases	34
Table 2-12	Comparison of the Results of the MOGO and MOPSO Design Approaches	35

Table 3-1	Nominal Component Values and Parasitic Elements used for LTSpice and MATLAB Simulation of the Parasitic Class-EF_2 Inverter	46
Table 3-2	Comparison of the Results of the Ideal and Parasitic MOGO Designs	49
Table 3-3	Comparison of the Component Values of the Ideal and Parasitic MOGO Designs	50
Table 3-4	Demonstration of the Calculation of all Parasitic Elements for Design Case I	51
Table 3-5	Values of all Parasitic Elements for Design Case II and Design Case III	51
Table 3-6	Comparison of the Results of the Ideal and Parasitic MOPSO Designs	52
Table 3-7	Comparison of the Component Values of the Ideal and Parasitic MOPSO Designs	53
Table 3-8	Values of all Parasitic Elements for each Design Case: MOPSO	54
Table 3-9	Comparison of the Results of the MOGO and MOPSO Design Approaches	55
Table 4-1	Internal Resistances of the EPC2019 Power MOSFET	65
Table 4-2	Averaged Internal Capacitance Values of the EPC2019 Using Ideal MOGO Design Case II	66
Table 4-3	Internal Resistances and Capacitances for use with the MOGO and MOPSO Design Approaches for the Complete Class-EF_2 Inverter	69
Table 4-4	Comparison of the Results of the Parasitic and Practical MOGO Designs	70
Table 4-5	Comparison of the Component Values of the Parasitic and Practical MOGO Designs	71
Table 4-6	Values of all Parasitic Elements for each MOGO Design Case	72
Table 4-7	Comparison of the Results of the Parasitic and Practical MOPSO Designs	73
Table 4-8	Comparison of the Component Values of the Parasitic and Practical MOPSO Designs	74

Table 4-9	Values of all Parasitic Elements for each MOPSO Design Case	74
Table 4-10	Comparison of the Results of the MOGO and MOPSO Design Approaches	75
Table 5-1	Updated Optimization Boundary Conditions	81
Table 5-2	Optimized Values of Variables for the Ideal Design Example	81
Table 5-3	Component Values for the Ideal Design Example	82
Table 5-4	Internal Capacitances and Resistances for the Practical Design Example	84
Table 5-5	Highest Scoring Variables for the Practical Design Example	84
Table 5-6	Component Values for the Practical Design Example	84
Table 5-7	LTSpice Simulation Results of the Practical Class-EF_2 Inverter Design Example	85

LIST OF FIGURES

Figure 1-1	Class-E ZVS Inverter Circuit Diagram	2
Figure 1-2	Class-EF_n (E/F_n) Inverter Circuit Diagram	3
Figure 2-1	Ideal Class-EF_2 Inverter Circuit Diagram	8
Figure 2-2 (a)	Switch Voltage Comparison of the Ideal State-Space model with LTSpice Simulation	12
Figure 2-2 (b)	Switch Current Comparison of the Ideal State-Space model with LTSpice Simulation	13
Figure 2-2 (c)	Output Voltage Comparison of the Ideal State-Space model with LTSpice Simulation	13
Figure 2-3	Flowchart of the MOGO Algorithm	16
Figure 2-4	Flowchart of the MOPSO Algorithm	30
Figure 2-5 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	36
Figure 2-5 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	36
Figure 2-5 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	36
Figure 2-6 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	37
Figure 2-6 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	37
Figure 2-6 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	37
Figure 2-7 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	38
Figure 2-7 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	38
Figure 2-7 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	38

Figure 3-1	Model of Capacitor (Left) and Inductor (Right) with Parasitic Elements	41
Figure 3-2	Circuit Diagram of the Class-EF_2 Inverter with Parasitic Elements	41
Figure 3-3 (a)	Switch Voltage Waveform from the Parasitic State-Space Model vs. LTSpice Simulation	47
Figure 3-3 (b)	Switch Current Waveform from the Parasitic State-Space Model vs. LTSpice Simulation	47
Figure 3-3 (c)	Output Voltage Waveform from the Parasitic State-Space Model vs. LTSpice Simulation	48
Figure 3-4 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	56
Figure 3-4 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	56
Figure 3-4 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case I	56
Figure 3-5 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	57
Figure 3-5 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	57
Figure 3-5 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case II	57
Figure 3-6 (a)	Switch Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	58
Figure 3-6 (b)	Switch Current Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	58
Figure 3-6 (c)	Output Voltage Waveform of the Parasitic Class-EF_2 Inverter: MOGO & MOPSO Design Case III	58
Figure 4-1	Switch Model for the Practical Class-EF_2 Inverter	61
Figure 4-2	Circuit Model of the Practical Class-EF_2 Inverter	61
Figure 4-3	Plots of C_{ds} and C_{gs} Using Ideal MOGO Design Case II	66

Figure 4-4 (a)	Switch Voltage Waveform of the Practical Class-EF_2 Inverter State-Space Model vs. LTSpice Simulation	67
Figure 4-4 (b)	Switch Current Waveform of the Practical Class-EF_2 Inverter State-Space Model vs. LTSpice Simulation	68
Figure 4-4 (c)	Output Voltage Waveform of the Practical Class-EF_2 Inverter State-Space Model vs. LTSpice Simulation	68
Figure 4-5 (a)	Switch Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case I	76
Figure 4-5 (b)	Switch Current Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case I	76
Figure 4-5 (c)	Output Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case I	76
Figure 4-6 (a)	Switch Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case II	77
Figure 4-6 (b)	Switch Current Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case II	77
Figure 4-6 (c)	Output Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case II	77
Figure 4-7 (a)	Switch Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case III	78
Figure 4-7 (b)	Switch Current Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case III	78
Figure 4-7 (c)	Output Voltage Waveform of the Practical Class-EF_2 Inverter: MOGO & MOPSO Design Case III	78
Figure 5-1	Switch Voltage and Switch Current Waveforms for the Ideal Design Example	83
Figure 5-2 (a)	Switch Voltage Waveform of the Practical Class-EF_2 Design	85
Figure 5-2 (b)	Switch Current Waveform of the Practical Class-EF_2 Design	85
Figure 5-2 (c)	Output Voltage Waveform of the Practical Class-EF_2 Design	86

LIST OF ABBREVIATIONS AND SYMBOLS

AC – Alternating Current

DC – Direct Current

THD – Total Harmonic Distortion

GaN – Gallium Nitride

MOSFET – Metal-Oxide Semiconductor Field Effect Transistor

ZVS – Zero-Voltage Switching

ZDVS – Zero-Derivative Voltage Switching

MOGO – Multi-Objective Genetic Optimization

MOPSO – Multi-Objective Particle Swarm Optimization

MATLAB – Matrix Laboratory

WPT – Wireless Power Transmission

NSGA-II – Non-Dominated Sorting Genetic Algorithm II

ESR – Equivalent Series Resistance

ESL – Equivalent Series Inductance

R – Resistance

L – Inductance

C – Capacitance

τ – Temperature

T_s – Switching Period

t – Time

f_s – Switching Frequency

η – Efficiency

$\mathbb{N}$ - Natural Numbers

$\mathbb{R}$ - Real Numbers

A – Amperes

V – Volts

W – Watts

Ω – Ohms

H – Henrys

F – Farads

Hz – Hertz

TABLE OF CONTENTS

Abstract ii

Acknowledgements iii

List of Tables iv

List of Figures vii

List of Abbreviations and Symbols x

Chapter 1: Introduction and Literature Review 1

1.1 Background Information 1
1.1.1 The Class-EF Inverter 3
1.1.2 Multi-Objective Optimization 5
1.2 Motivation 6
1.3 Outline 6

Chapter 2: The Ideal Class-EF_2 Inverter 8

2.1 State-Space Modelling of the Ideal Class-EF_2 Inverter 8
2.2 Solving the model 11
2.2.1 Simulation of the Ideal Class-EF_2 Inverter 11
2.2.2 Comparison of MATLAB and LTSpice Simulation Results 12
2.3 Multi-Objective Genetic Optimization Algorithm 14
2.3.1 Variables to be Optimized 17
2.3.2 Boundary Conditions 18
2.3.3 Objective and Constraint Functions 19
2.3.4 Design Test Cases 21
2.3.5 Optimal Design Factors Based on Analytical Design Procedure 21
2.4 Design Results Using the MOGO Algorithm 22
2.4.1 Determination of the Inverter Component Values from the MOGO Designs 26
2.5 Multi-Objective Particle Swarm Optimization Algorithm 27
2.5.1 Design Results Using the MOPSO Algorithm 30
2.5.2 Determination of the Inverter Component Values from the MOPSO Designs 34
2.6 Comparison of MOGO and MOPSO Design Approaches 35
2.7 Summary 39

Chapter 3: The Parasitic Class-EF_2 Inverter 40

3.1 State-Space Model of the Class EF_2 Inverter Including Parasitic Elements 40
3.2 Solving the model 44
3.2.1 Determination of the Parasitic Elements 44
3.2.1.1 Parasitic Elements of Inductors 44
3.2.1.2 Parasitic Elements of Capacitors 45

3.2.2 Simulation of the Parasitic Class-EF_2 Inverter 46
3.2.3 Comparison of MATLAB and LTSpice Simulation Results 46
3.3 Design Results Using the MOGO Algorithm 48
3.3.1 Determination of the Inverter Component Values from the MOGO Designs 50
3.3.1.1 Determination of the Parasitic Elements for the MOGO Designs 50
3.4 Design Results Using the MOPSO Algorithm 52
3.4.1 Determination of the Inverter Component Values from the MOPSO Designs 53
3.4.1.1 Determination of the Parasitic Elements for the MOPSO Designs 54
3.5 Comparison of MOGO and MOPSO Design Approaches 54
3.6 Summary 59

Chapter 4: The Practical Class-EF_2 Inverter 60

4.1 State-Space model of the Parasitic Class-EF_2 Inverter Including Switch Dynamics 60
4.2 Solving the Model 64
4.2.1 Determination of the Switching Dynamics 64
4.2.1.1 Internal Resistances of the Switch 64
4.2.1.2 Internal Capacitances of the Switch 65
4.2.2 Simulation of the Practical Class-EF_2 Inverter 67
4.2.3 Internal Resistances and Capacitances for all Design Cases 69
4.3 Design Results Using the MOGO Algorithm 70
4.3.1 Determination of the Inverter Component Values from the MOGO Designs 71
4.4 Results of the MOPSO Designs 72
4.4.1 Determination of the Inverter Component Values from the MOGO Designs 73
4.5 Comparison of MOGO and MOPSO Design Approaches 75
4.6 Summary 79

Chapter 5: Multi-Objective Genetic Optimization Design Example 80

5.1 Design Example Stage 1 – Setting up the Problem 80
5.2 Design Example Stage 2 – Ideal Optimization 81
5.3 Design Example Stage 3 – Determination of Suitable Switch 83
5.4 Summary 86

Chapter 6: Conclusions and Future Work 87

6.1 Contribution of the Thesis 89
6.1.1 List of Publications 89
6.2 Future Work 89

References 91

Appendices 94

Appendix A 94
Appendix B 101
Appendix C 112
Appendix D 124
Appendix E 128
Appendix F 131

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

As wireless power transfer systems become more widely used, the need to quickly produce efficient circuit designs has attracted attention. Class-E and Class-EF inverter topologies are often used in these systems due to their high efficiency, low component count, minimal total harmonic distortion (THD), and fast switching capabilities thanks to fast switching devices based on gallium nitride (GaN) technology. Analysis of these topologies has been studied extensively in literature; however, the design of the Class-EF inverter is often a tedious and iterative process, making it challenging for designers to test their systems quickly and efficiently.

1.1 Background Information

The Class-E inverter has been the most efficient inverter for most megahertz frequency applications ever since it was first proposed in 1975 [1] and has been studied extensively in the literature [2]–[5]. The circuit diagram for the Class-E inverter is shown in Fig. 1-1.

The circuit contains a DC input voltage V_{in}, a choke inductor L_f, a shunt capacitor C_f, a power MOSFET which acts as a switch, and a series resonant filter consisting of L_s, C_s, and a load resistance R_L. The choke inductor L_f is made high enough to ensure a DC input current, and the loaded quality factor of the output filter Q_{Out} is made high enough to ensure the output voltage v_o is sinusoidal.

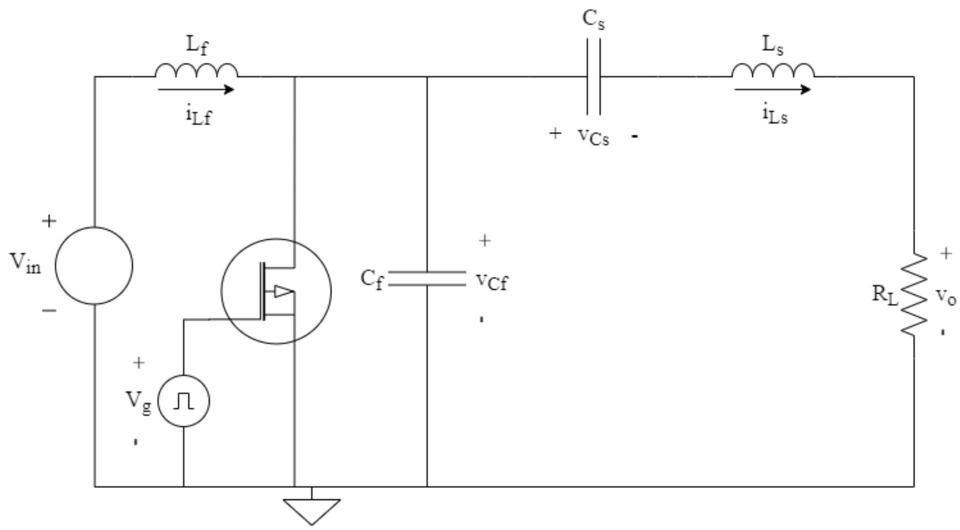

Figure 1-1 – Class-E ZVS Inverter Circuit Diagram

To ensure zero-voltage switching operation, the inverter switching frequency f_s should be greater than the resonant frequency f_o of L_s and C_s [1]. It is useful to think of L_s as being the series combination of two inductors L and L_{res} where L resonates with C_s as described in (1.2) and L_{res} is the residual inductance needed to ensure zero-voltage switching (ZVS) operation. Some helpful equations are shown in (1.1) to (1.3).

$$\omega_o = 2\pi f_o = \frac{1}{\sqrt{L_s C_s}} \tag{1.1}$$

$$\omega_s = 2\pi f_s = \frac{1}{\sqrt{L C_s}} \tag{1.2}$$

$$Q_{Out} = \frac{\omega_s L_s}{R} = \frac{\omega_s (L + L_{res})}{R} = \frac{1}{\omega_s C_s R} + \frac{\omega_s L_{res}}{R} \tag{1.3}$$

Despite its popularity in the field of power electronics, some of the main drawbacks of the Class-E inverter are as follows:

- High voltage stress (>3.5),
- Strong second harmonic producing high switch and output voltage THD, and

- Sub-optimal operation when the load resistance is varied.

To improve the operation of the Class-E inverter, designs which maintain ZVS operation regardless of the load resistance have been developed. The load independent Class-E inverter was first proposed in 1990 [6] and has gathered much attention since then [7]–[9]. The prospect of maintaining ZVS over a large range of load impedances is particularly useful in the field of wireless power transfer, as the load is often variable in these systems.

1.1.1 The Class-EF Inverter

In 2002, a new topology of inverter was proposed which combined aspects of the Class-E amplifier with the Class-F and F^{-1} amplifiers [10]. In some applications, this new family of inverters have been shown to improve the efficiency, reduce the voltage stress, and lower the output voltage THD of the Class-E inverter [10]–[15]. The circuit diagram of the Class-EF_n (or E/F_n) inverter is shown in Fig. 1-2.

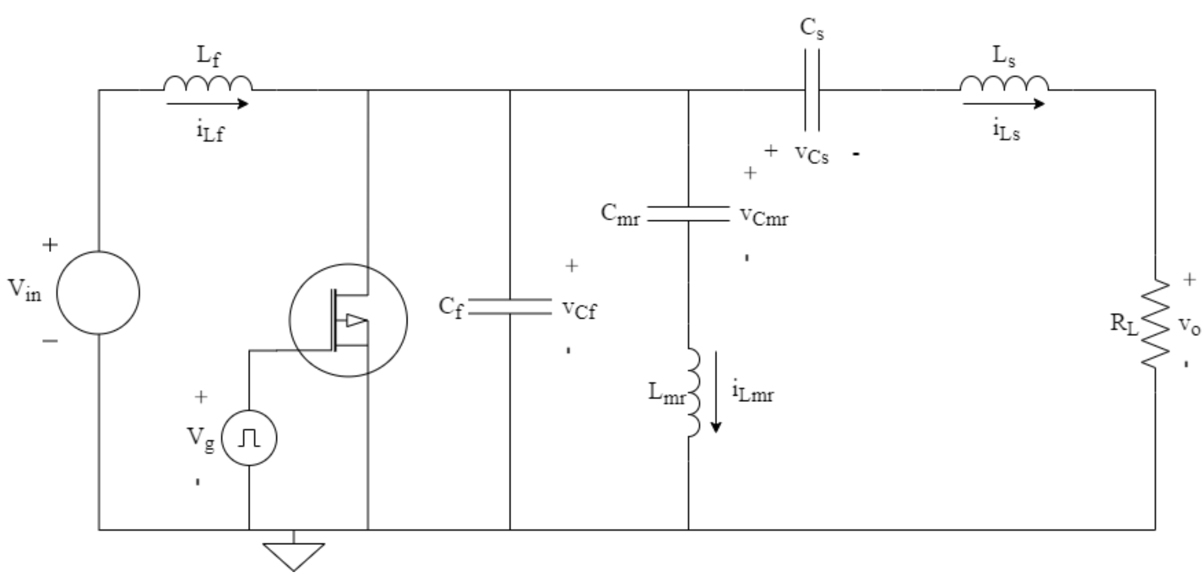

Figure 1-2 – Class-EF_n (E/F_n) Inverter Circuit Diagram

The new addition to the circuit is the series resonant branch containing C_{mr} and L_{mr} which are tuned to resonate at nf_s as described in eq. (1.14). The circuit is known as the Class-EF$_n$ inverter for even values of n, and the Class-E/F$_n$ inverter for odd values of n.

$$\omega_{mr} = n \times 2\pi f_s = \frac{1}{\sqrt{L_{mr}C_{mr}}} \; ; n \in \mathbb{N}, n \neq 1 \tag{1.14}$$

In [11], a piecewise-linear state-space model was applied to determine relationships between component values of the Class-E, Class-EF$_2$ and Class-E/F$_3$ inverters for ZVS and zero derivative voltage switching (ZDVS) operation. This provided useful information about which duty cycles maximized the output power capabilities of the Class-EF and Class-E/F family of inverters and demonstrated the power of state-space modelling for switching converters.

In [12], the Class-Φ_2 inverter was proposed which is a variation of the Class-EF$_2$ inverter that uses a finite choke in place of an infinite choke to increase the maximum frequency of the inverter. The design process presented in the paper involves tuning the magnitude and phase of specific impedances within the circuit to achieve the desired output power characteristics and ensure ZVS operation.

In [14], closed form expressions for the Class-EF and Class-E/F inverters for any duty cycle and any output filter quality factor were derived using circuit analysis. Design equations and performance parameters were also presented for the Class-EF$_2$ inverter which achieve maximum power output capability and maximum frequency.

The Class-EF and Class-E/F family of inverters operate sub-optimally when the load resistance is varied but like in the case of the Class-E inverter, load independent design methods were proposed [7], [8]. However, the load-independent case is outside the scope of this thesis.

It has been shown that a higher throughput and a lower voltage stress can be achieved using the Class-EF_2 inverter over the traditional Class-E inverter [16]. However, the Class-EF_2 inverter requires a higher value of V_{in}, has a lower maximum frequency, and results in a more complex design which is often a tedious, iterative process. All previous works have also considered the circuit to be ideal, making it a challenge to implement in a practical setting.

1.1.2 Multi-Objective Optimization

Multi-objective optimization is often used to tackle complex engineering problems [17]–[20]. The multi-objective genetic optimization (MOGO) algorithm and the multi-objective particle swarm optimization (MOPSO) algorithm are commonly used and are accessible through MATLABs [21] global optimization toolbox [22] or the MOPSO function [23].

In [17], the MOGO algorithm is implemented in combination with finite element analysis to aid in the design of a permanent magnet synchronous motor. The MOGO algorithm successfully improved the torque of the system while maintaining an acceptable level of efficiency thus validating its use as a design method.

In [19], the MOPSO algorithm is implemented to determine the optimal operating frequency and inductor size for a wireless power transfer system for car charging. The proposed design was verified experimentally and provided helpful insight into the design of these systems.

In [20], two variations of the MOPSO algorithm are applied to the design of a water distribution system and the results are compared. This work demonstrates the power of the MOPSO algorithm, but also how widely the solutions can vary when modifications are made to the algorithm.

The MOGO and MOPSO algorithms both face similar challenges in their implementation. The generation and population size will determine how quickly the solution converges but will also have an impact on the optimization runtime. The choice of boundary conditions and constraint functions, as well as genetic operators and particle swarm constants can also drastically change the success of a design [20], [24]. Specific details on the MOGO and MOPSO algorithms will be provided in Chapter 2.

1.2 Motivation

The design process of the ideal Class-EF_2 inverter is often tedious, iterative, and complex. The consideration of non-ideal components and switching elements is also lacking in the literature. This makes it difficult for designers to test their circuit designs quickly and efficiently and requires more tuning to implement the circuit in a practical setting.

In this thesis, a multi-objective optimization-based design approach is proposed for the Class-EF_2 inverter. Three state-space models of varying complexity are derived and tested with both the MOGO and the MOPSO algorithms. It is expected that the results will demonstrate the effectiveness and versatility of the optimization-based design, which allows the parasitic elements and switching dynamics of the circuit to be incorporated in the design of high frequency circuits.

1.3 Outline

Chapter 1 provides background information related to the thesis topic and provides an outline for subsequent chapters.

In chapter 2, the state-space model of the ideal Class-EF_2 inverter is derived and validated. Next, details of the MOGO and MOPSO algorithms for the design of the inverter are provided. Both algorithms are then used to design three circuits of varying frequency, power, and load specifications. Each design is compared with published results to confirm their validity and accuracy.

In chapter 3, the parasitic elements of the components in the Class-EF_2 inverter are considered in the derivation of the state-space model. Methods for estimating the value of each parasitic element are described, and the state-space model is validated by comparison and LTSpice simulation [25]. The MOGO and MOPSO algorithms are then used to design the same three circuits presented in the previous chapter with the new state-space model. The results demonstrate the importance of considering the parasitic elements when designing high-frequency circuits.

In chapter 4, the state-space model is further extended to consider the dynamics of the switching device. Methods for estimating the value of the internal resistances and capacitances are described, and the state-space model is once again validated by comparison and LTSpice simulation. The MOGO and MOPSO algorithms are then used to design the same three circuits as the previous chapters once again demonstrating the importance of considering the switch during the design process.

In chapter 5, a design example is presented which uses the MOGO algorithm to design a 6.78 MHz, 100W, 20Ω Class-EF_2 inverter. This demonstrates the ease-of-use of the optimization-based design approach and shows how the boundary conditions can be restricted to eliminate solutions known to be inferior.

Chapter 6 concludes the thesis and highlights potential future work that can be undertaken to improve and extend the optimization-based design approach to other circuit topologies.

CHAPTER 2

THE IDEAL CLASS-EF$_2$ INVERTER

In this chapter, a state-space model of the ideal Class-EF$_2$ inverter is derived. The validity of this model is confirmed by comparison with LTSpice simulation. Details of the MOGO and MOPSO algorithms for the design of the inverter are provided including the variables and boundary conditions, as well as the objective and constraint functions.

Both algorithms are used to design three circuits with differing frequency, power, and load specifications. Each design is tested and simulated in LTSpice, and comparisons are made with published results to demonstrate their validity and accuracy.

2.1 State-Space Modelling of the Ideal Class-EF$_2$ Inverter

To test and validate the optimization-based design approach, the state-space model of the ideal Class-EF$_2$ inverter was derived, the circuit model of which is shown in Fig. 2-1.

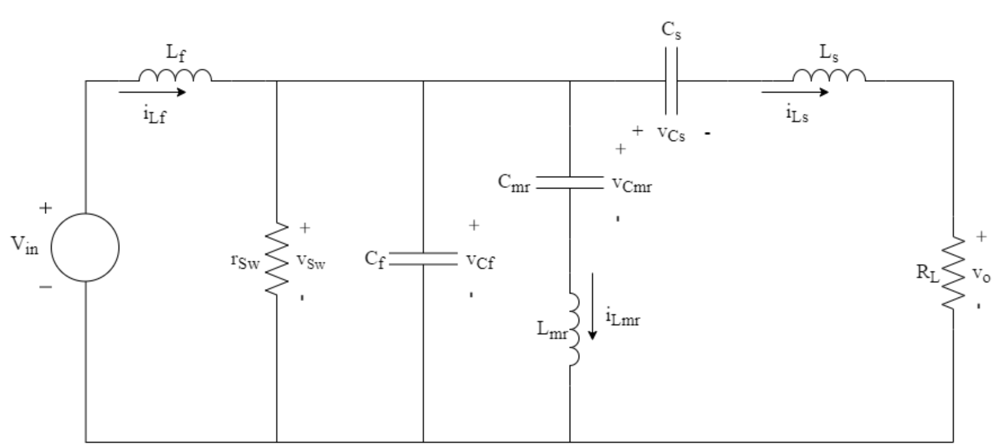

Figure 2-1 – Ideal Class-EF$_2$ Inverter Circuit Diagram

State-space modelling is a powerful tool when analyzing complex systems since the dynamics are represented as a system of linear equations. The Class-EF_2 inverter contains multiple inductors and capacitors, thus making state-space representation a desirable choice. Its solution can also be obtained using trivial matrix operations, making it a good choice for the proposed optimization-based design approach.

To analyze the circuit, the switch is replaced by a variable resistor r_{sw} which acts as an approximate short circuit during the ON-state, and an approximate open circuit during the OFF-state where T_s is the switching period of the inverter.

$$r_{sw}(t) = \begin{cases} r_{on} = 10\ m\Omega\ , nT_s \leq t < (n+D)T_s \\ r_{off} = 1\ M\Omega\ , (n+D)T_s \leq t < (n+1)T_s \end{cases} , t \geq 0\ , t \in \mathbb{R}\ , n \in \mathbb{N} \tag{2.1}$$

Standard state-space representation is used where the state vector **X** is made up of the capacitor voltages and the inductor currents, and the input **U** is the source voltage as shown in (2.2) and (2.3).

$$\mathbf{X} = \begin{bmatrix} v_{C_f} & v_{C_{mr}} & v_{C_s} & i_{L_f} & i_{L_{mr}} & i_{L_s} \end{bmatrix}^T \tag{2.2}$$

$$\mathbf{U} = V_{in} \tag{2.3}$$

To create the state matrices, differential equations for each state variable must first be derived in terms of the other state variables. This process is presented in (2.4) - (2.9).

$$\frac{dv_{C_f}}{dt} = \frac{i_{C_f}}{C_f} = \frac{i_{L_f} - i_{L_{mr}} - i_{L_s} - i_{Sw}}{C_f} \tag{2.4}$$

$$\frac{dv_{C_{mr}}}{dt} = \frac{i_{C_{mr}}}{C_{mr}} = \frac{i_{L_{mr}}}{C_{mr}} \tag{2.5}$$

$$\frac{dv_{C_s}}{dt} = \frac{i_{C_s}}{C_s} = \frac{i_{L_s}}{C_s} \tag{2.6}$$

$$\frac{di_{L_f}}{dt} = \frac{v_{L_f}}{L_f} = \frac{V_{in} - v_{C_f}}{L_f} \tag{2.7}$$

$$\frac{di_{L_{MR}}}{dt} = \frac{v_{L_{mr}}}{L_{mr}} = \frac{v_{C_f} - v_{C_{mr}}}{L_{mr}} \tag{2.8}$$

$$\frac{di_{L_s}}{dt} = \frac{v_{L_s}}{L_s} = \frac{v_{C_f} + i_{L_s}R_L - v_{C_s}}{L_s} \tag{2.9}$$

The final step is to represent (2.4) to (2.9) in the following form, where A, B, C, and D are the state matrices, $\dot{\mathbf{X}}$ is the derivative of (2.2), and **Y** is the output vector.

$$\dot{\mathbf{X}} = A(t)\mathbf{X} + B\mathbf{U} \tag{2.10}$$

$$\mathbf{Y} = C\mathbf{X} + D\mathbf{U} \tag{2.11}$$

This circuit produces a 6th order single input multiple output (SIMO) system. The state vectors and matrices are defined in (2.12) to (2.15).

$$A(t) = \begin{bmatrix} -1/C_f r_{sw}(t) & 0 & 0 & 1/C_f & -1/C_f & -1/C_f \\ 0 & 0 & 0 & 0 & 1/C_{MR} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/C_s \\ -1/L_f & 0 & 0 & 0 & 0 & 0 \\ 1/L_{MR} & -1/L_{MR} & 0 & 0 & 0 & 0 \\ 1/L_s & 0 & -1/L_s & 0 & 0 & R_L/L_s \end{bmatrix} \tag{2.12}$$

$$B = [0 \quad 0 \quad 0 \quad 1/L_f \quad 0 \quad 0]^T \tag{2.13}$$

$$C = I_6 \tag{2.14}$$

$$D = \vec{0} \tag{2.15}$$

Equations (2.12) and (2.13) contain the circuit components of the ideal Class-EF_2 inverter, I_6 in (2.14) represents the 6th order identity matrix, and $\vec{0}$ in (2.15) represents the zero vector. As can be seen, the system can also be classified as continuous time-varying due to the variable resistor model of the switch which changes in time based on the switching period and the inverter duty cycle. It is helpful to think of the system as having two separate state matrices – A_{on} and A_{off} – which represent the system during the switch on and off states as described in (2.16) and (2.17).

$$A_{on} = A(t)|_{r_{sw}(t)=r_{on}} \tag{2.16}$$

$$A_{off} = A(t)|_{r_{sw}(t)=r_{off}} \tag{2.17}$$

2.2 Solving the model

To determine the steady-state values of all state-variables, the system is simulated for roughly 20us using the technique described above with A_{on} and A_{off} to quickly generate solutions. After this, the system is fully solved using a resolution of 300 samples/cycle for analysis. The calculations are performed using equations (2.18) to (2.20) [16].

$$X(t) = X_n(t) + X_f(t) \tag{2.18}$$

$$X_n(t) = e^{At}X(0) \tag{2.19}$$

$$X_{f(t)} = \int_{o}^{\tau} e^{A(t-\tau)}BU(\tau)\, d\tau = A^{-1}(e^{At} - I_6)B \tag{2.20}$$

Where X_n is the natural response matrix and X_f is the forced response matrix.

2.2.1 Simulation of the Ideal Class-EF$_2$ Inverter

The state-space model is simulated using the component values from a previously completed design, given in Table 2-1 [26]. The component values were substituted into the system equations of (2.10) to (2.15) in MATLAB. The circuit was then constructed and simulated in LTSpice.

Table 2-1 – Circuit Parameters for MATLAB and LTSpice Comparison

L_f	54.40 uH	C_f	147.94 pF
L_{MR}	89.04 nH	C_{MR}	96.70 pF
L_s	501.43 nH	C_s	80.30 pF
D	0.3637	V_{in}	32.33 V
f_s	27.12 MHz	R_L	7 Ω

2.2.2 Comparison of MATLAB and LTSpice Simulation Results

A comparison of the resulting switch voltage, switch current, and output voltage waveforms from both simulations is shown in Figs. 2.2 (a-c). The plots demonstrate the accuracy of the ideal state-space model.

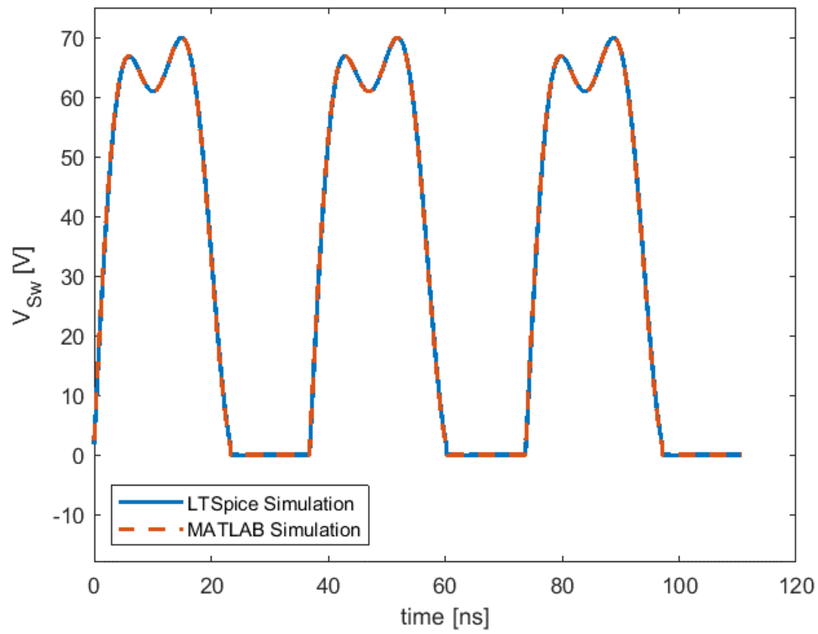

Figure 2-2 (a) – Switch Voltage Comparison of the Ideal State-Space model with LTSpice Simulation

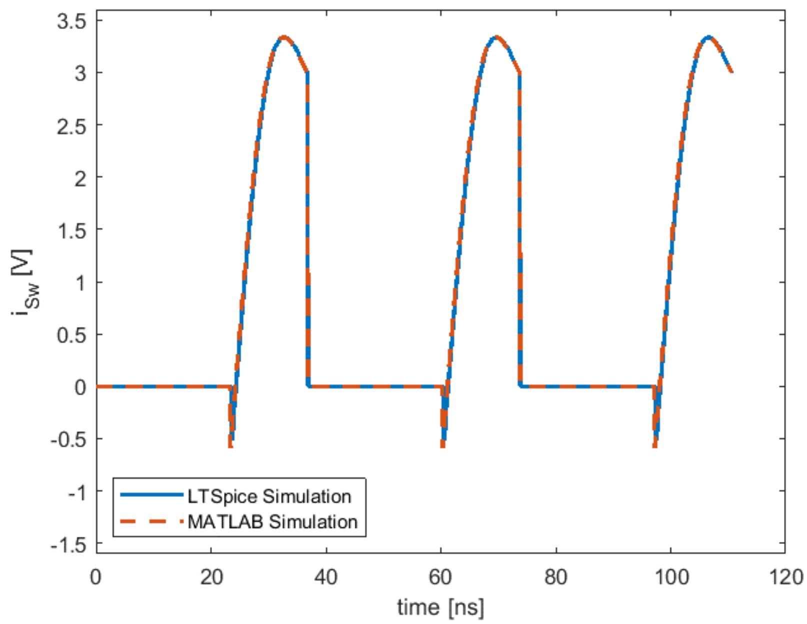

Figure 2-2 (b) – Switch Current Comparison of the Ideal State-Space model with LTSpice Simulation

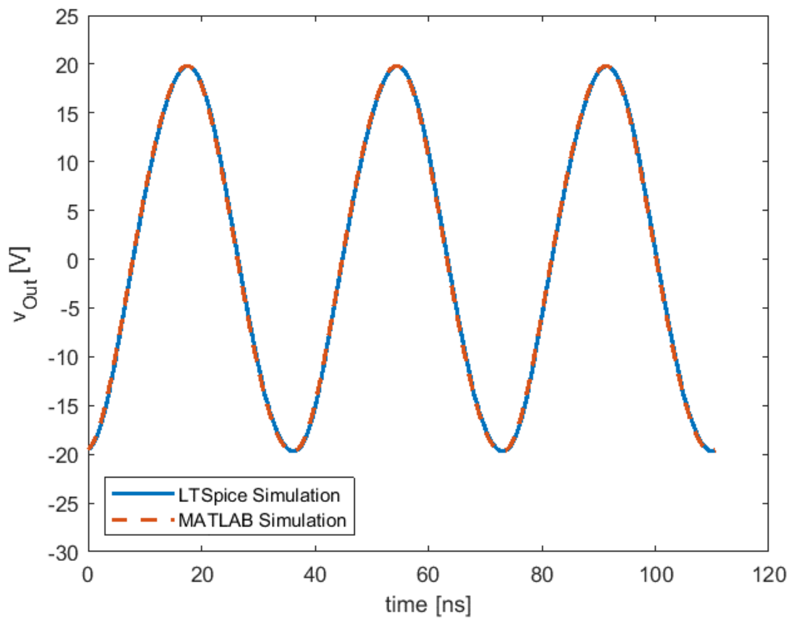

Figure 2-2 (c) – Output Voltage Comparison of the Ideal State-Space model with LTSpice Simulation

The state-space model is a perfect representation of the behaviour of the Class-EF_2 inverter and can therefore be used to develop a design approach for the Class-EF_2 inverter.

2.3 Multi-Objective Genetic Optimization (MOGO) Algorithm

For each design case, the *gamultiobj* function in MATLABs global optimization toolbox was used. This is a controlled, elitist genetic algorithm which is a variant of the NSGA-II [27]. This type of algorithm favors increased diversity in the population, not relying solely on fitness value which helps avoid getting stuck in local minima. The general loop of the *gamultiobj* function for this application is presented below [22], and an accompanying flowchart is shown in Figure 2-3.

Initialization

- <u>Generate an initial population of component values:</u> The population matrix is of size $n \times m$ where n is the defined population size and m is the number of variables which represent the duty cycle D, the input voltage V_{in}, the input inductance L_f, the filter capacitance C_f, and various factors which relate to the other inductors and capacitors in the circuit.
- <u>Pass each member of the population to the fitness function:</u> The member is a vector of size $1 \times m$ and contains all the necessary information to solve the state-space model of the Class-EF_2 inverter. The fitness function will return a vector of size $1 \times l$ where l is the number of objective and constraint functions which are described in detail in Section 2.3.3.
- <u>Evaluate the population based on the objective and constraint functions:</u> The *gamultiobj* function evaluates the population based on its objective and constraint function performance and assigns each member a rank.

All Subsequent Iterations

- <u>Select parents for the next generation:</u> The selection process is done using binary tournament and is based on the rank of each population member. Rank is directly linked to dominance, meaning members that are non-dominated or that are dominated by a small percentage of the population are ranked lower and members that are dominated by a large percentage of the population are ranked higher. Members of lower rank have a better chance of being selected as parents.
- <u>Use the selected parents to create children:</u> This process uses the mutation and crossover genetic operators. Mutation will randomly change the value of a single element in one of the parents (i.e., randomly change the value of a single inductor but leave all other component values the same). Crossover will randomly swap some values of two parent members resulting in two unique children (i.e., child 1 might have the voltage and duty cycle of parent 1 and the inductor and capacitor values of parent 2, and child 2 would have the opposite). Mutation is used to help maintain diversity in the population, while crossover is used to improve searching [24].
- <u>Evaluate the children based on the objective and constraint functions:</u> The *gamultiobj* function scores the children based on their objective and constraint function performance.
- <u>Create the extended population:</u> This is done by combining the current population and the children into a single matrix of size $(n + c) \times m$ where c is the number of children.

- <u>Trim the extended population:</u> The population is trimmed back to size $n \times m$ by removing some members from each rank. This helps maintain diversity in the population as opposed to keeping only the best solutions in each iteration.

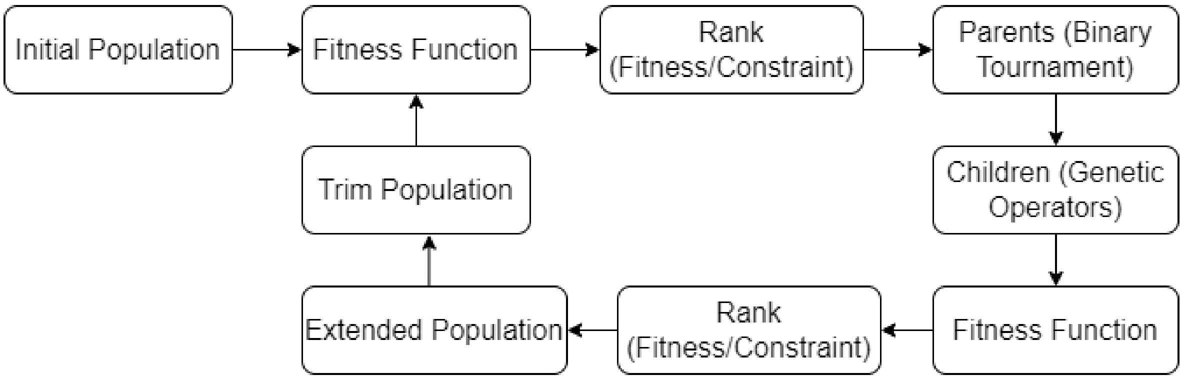

Figure 2-3 – Flowchart of the MOGO Algorithm

To determine an appropriate generation and population size, it is useful to consider the computational cost defined in (2.21).

$$Computational\ Cost = \#\ of\ Generations \times Population\ Size \tag{2.21}$$

A higher computational cost will increase the runtime of the optimization, but a lower computational cost might not arrive at an acceptable solution. This presents the designer with a trade-off between the time invested and the quality of the circuit design.

After experimenting with various combinations of population and generation size, and consulting with experts in the field, a population size of 525 with 100 generations was found to be the most suitable for this application. These values provided acceptable results while keeping runtimes reasonably low across all design cases. Code for the ideal Class-EF_2 fitness function and the MOGO initialization function is provided in Appendix A and Appendix D respectively.

2.3.1 Variables to be Optimized

In the design of the Class-EF_2 inverter, appropriate selection of key variables will lead to the complete design of the inverter by the optimization algorithm. For the Class EF_2 inverter, the variables of interest are shown in Table 2-2.

Table 2-2 – Optimization Variables for the MOGO Based Design Approach

Variable	Definition
D	Duty Cycle
L_f	Input choke inductance
C_f	Switch filter capacitor
k	Scalar multiple relating C_f and C_{MR}
Q_{out}	Output filter quality factor
$x_{L_S C_S}$	Resonant frequency factor for L_s and C_s
V_{in}	Input voltage of the circuit

These variables fully define the Class-EF_2 inverter. By using k, Q_{out} and $x_{L_S C_S}$ rather than inductor and capacitor values the overall search space is reduced and more insight is provided to the designer. C_f, L_{MR}, C_{MR}, L_s and C_s are related by (2.22) to (2.25), where f_s is the switching frequency of the inverter:

$$k = \frac{C_f}{C_{mr}} \tag{2.22}$$

$$2 \times 2\pi f_s = \frac{1}{\sqrt{L_{MR} C_{MR}}} \tag{2.23}$$

$$Q_{out} = \frac{\sqrt{L_s / C_s}}{R} \tag{2.24}$$

$$x_{L_S C_S} \times 2\pi f_s = \frac{1}{\sqrt{L_S C_S}} \tag{2.25}$$

As can be seen from (2.22), the resonant tank containing L_{MR} and C_{MR} is designed to resonate at the 2^{nd} harmonic of f_s, hence the factor 2 in (2.22). This is a property of the Class-EF_2 inverter.

Equation (2.24) defines the resonant frequency factor for the output filter containing L_s and C_s to allow for a residual reactance, X_{res}.

2.3.2 Boundary Conditions

To ensure that the search space for the optimization process is limited to acceptable values, boundary conditions are enforced on each variable. The boundary conditions are presented in Table 2-3.

Table 2-3 – Optimization Boundary Conditions

Variables	D	$L_f\ [\mu H]$	$C_f\ [pF]$	k	Q_{Out}	$x_{L_S C_S}$	$V_{in}\ [V]$
Upper Boundary	0.8	100	5000	5	8	5	72
Lower Boundary	0.2	0.01	0.5	0.2	2	0.2	12

The boundary conditions were chosen to ensure impractical solutions were avoided. Values of V_{in} outside of this range would be impractical for the power requirements of the tested design cases, and values of Q_{Out} outside of this range would be ignored due to their effect on the THD and efficiency of the circuit.

The other boundary conditions were selected with known optimal values in mind. For example, since k is optimally between 0.8 and 1.6 [14] the upper and lower boundaries were selected to ensure the search space wasn't too restricting. This same logic was used for determining the upper and lower boundaries of $x_{L_S C_S}$ and D.

The upper and lower boundaries of L_f and C_f can be tuned to meet the needs of the designer based on the desired switching frequency and availability of components.

2.3.3 Objective and Constraint Functions

To ensure the best overall circuit design, five objective functions to be minimized were implemented:

- The losses in the circuit to maximize efficiency,
- The voltage across the switch at $t = (n_i + D)T_s$,
- The current through the switch and capacitor C_f at $t = \left(n_j + D\right)T_s$,
- The Total Harmonic Distortion (THD) of the load voltage, and
- The error between the desired output power and the calculated output power.

These objective functions are implemented mathematically in (2.26) to (2.30) using the state-variables where *i* represents the cycle which produced the worst-case switch voltage and *j* represents the cycle which produced the worst-case switch current during switching transitions. They are then normalized to values between 0 and 1 as this is a requirement for the *gamultiobj* function.

$$Obj_1 = \frac{\left(1 - \frac{{i_{L_{s}}}_{RMS}{}^{2} R}{V_{in} {i_{L_f}}_{AVG}}\right)}{(1 - \eta_{Min})} \tag{2.26}$$

$$Obj_2 = \frac{v_{C_f}\left((n_i + D)T_s\right)}{{v_{C_f}}_{Max}} \tag{2.27}$$

$$Obj_3 = \frac{i_{L_f}\left(\left(n_j + D\right)T_s\right) - i_{L_{MR}}\left(\left(n_j + D\right)T_s\right) - i_{L_s}\left(\left(n_j + D\right)T_s\right)}{(i_{L_f} - i_{L_{MR}} - i_{L_s})_{Max}} \tag{2.28}$$

$$Obj_4 = \frac{THD(i_{L_s} R_L)}{THD_{Max}} \tag{2.29}$$

$$Obj_5 = \frac{\left| {i_{L_{s_{RMS}}}}^2 R_L - P_{Out_{Desired}} \right|}{P_{Out_{Desired}}} \tag{2.30}$$

To ensure the optimizer avoids undesirable solutions, constraint functions were implemented:

- Voltage across the switch should not drop below 0. Since (2.27) only checks the switch voltage at a specific time, this constraint ensures a non-negative switch voltage for the entire cycle.
- The ripple of the input current should not exceed 10% of its mean value. This constraint ensures that the input current is relatively DC.
- Current through the switch and capacitor C_f should not oscillate at high frequency. For some combinations of component values the switch current looked like a decaying sinusoid and produced a low score for (2.28). This constraint ensures that these types of solutions are unacceptable and hence ignored.
- Efficiency should not drop below the defined minimum η_{Min}. This ensures that (2.26) is always in the range of 0 to 1. If the efficiency is less than η_{Min}, objective function (2.26) will automatically be given a score of 1.
- THD should not exceed the defined maximum THD_{Max}. This ensures that (2.29) is always in the range of 0 to 1. If the THD is greater than THD_{Max}, objective function (2.29) will automatically be given a score of 1.
- Output power should not exceed twice the desired output power. This ensures that (2.30) is always in the range of 0 to 1. If the output power is greater than twice the desire output power, objective function (2.30) will automatically be given a score of 1.

These constraint functions are trivial to check once the model has been solved and an FFT analysis has been completed on the switch current waveform.

2.3.4 Design Test Cases

Three different design cases, each with varied power, frequency, and load requirements were used throughout the thesis to test the optimization algorithms. The specific values of each case are shown in Table 2-4.

Table 2-4 – Parameters for Tested Design Cases

Design	f_s $[MHz]$	R_L $[\Omega]$	P_{Out} $[W]$
I	6.78	5	23
II	13.56	10	40
III	27.12	7	25

2.3.5 Optimal Design Factors Based on Analytical Design Procedure

The published results of an analytical design procedure [14] is used as a basis for validating the proposed optimization design approach presented in this chapter. In the published results, two operating limits, namely maximum power-output capability (*Max* c_p) and maximum switching frequency capability (*Max f*) were determined for eight design factors for the Class-EF_2 inverter. It was shown in [14] that for lower switching frequencies (i.e., $f_s < 8$ MHz), the inverter operated efficiently at *Max* c_p, while the inverter operated efficiently under *Max f* for higher frequencies. These optimal design factors are listed in Table 2-5.

Table 2-5 – Optimal Design Factors for the Ideal Class-EF$_2$ Inverter [14]

Factors	$\boldsymbol{Max\ c_p}$	$\boldsymbol{Max\ f}$
D	0.375	0.372
$k = \frac{C_f}{C_{MR}}$	0.867	1.567
$q_2 = \frac{1}{\omega}\sqrt{\frac{C_f + C_{MR}}{C_f C_{MR} L_{MR}}}$	2.935	2.560
$1/\omega R_L C_f$	7.585	5.686
$1/\omega R_L C_{MR}$	6.576	8.910
$c_p = \frac{P_{Out}}{V_{Stress} \times I_{Stress}}$	0.1323	0.120
V_{Stress}	2.316	2.243
I_{Stress}	3.263	3.719

2.4 Design Results Using the MOGO Algorithm

Using the procedure and information presented in Section 2.2, the results of the MOGO design are obtained and compared with the analytical design approach. Tables 2-6 a-c show the comparative values for the three design cases.

Table 2-6 (a) – Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case I

Factors	Design Case I *Max c_p*		
	Analytical Values	**MOGO Design**	**Percentage Difference**
D	0.375	0.406	7.94%
k	0.867	0.823	5.20%
q_2	2.935	2.977	1.41%
$\frac{1}{\omega R_L C_f}$	7.585	6.980	8.31%
$\frac{1}{\omega R_L C_{MR}}$	6.576	5.745	13.49%
c_p	0.132	0.128	3.58%
V_{Stress}	2.316	2.540	9.22%
I_{Stress}	3.263	3.136	3.96%

Table 2-6 (b) – Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case II.

Factors	Design Case II *Max f*		
	Analytical Values	MOGO Design	Percentage Difference
D	0.372	0.370	0.46%
k	1.567	1.408	10.12%
q_2	2.560	2.615	2.17%
$\frac{1}{\omega R_L C_f}$	5.686	5.365	5.65%
$\frac{1}{\omega R_L C_{MR}}$	8.910	7.556	15.19%
c_p	0.120	0.125	4.43%
V_{Stress}	2.243	2.241	0.10%
I_{Stress}	3.719	3.523	5.27%

Table 2-6 (c) – Comparison of the Results of the Analytical and MOGO Design Approaches: Design Case III.

Factors	Design Case III *Max f*		
	Analytical Values	**MOGO Design**	**Percentage Difference**
D	0.372	0.394	6.00%
k	1.567	1.472	6.08%
q_2	2.560	2.592	1.25%
$\frac{1}{\omega R_L C_f}$	5.686	5.961	4.85%
$\frac{1}{\omega R_L C_{MR}}$	8.910	8.774	1.52%
c_p	0.120	0.122	1.90%
V_{Stress}	2.243	2.337	4.16%
I_{Stress}	3.719	3.484	6.33%

It is observed that for each design case the percentage differences between the analytical values and the MOGO algorithm are at an acceptable level. Each design factor was within 10% of the expected value, the main outlier being the $1/\omega R_L C_{mr}$ design factor for design case I and design case II. The analytical design factors assume that the loaded quality factor of the output filter is high enough to ensure a sinusoidal output waveform. However, since Q_{out} is limited by the optimizer this might not always be the case.

2.4.1 Determination of the Inverter Component Values from the MOGO Designs

To demonstrate how the Class-EF_2 component values are determined, the optimized values obtained for design case I (shown in Table 2.7) are substituted into (2.22) to (2.25). The process for calculating the actual circuit parameters is presented in Table 2-8.

Table 2-7 – MOGO Optimized Values for the Ideal Class-EF_2 Inverter: Design Case I

Variable	D	$L_f\ [\mu H]$	$C_f\ [pF]$	k	Q_{Out}	$x_{L_sC_s}$	$V_{in}\ [V]$
Value	0.406	39.17	672.59	0.823	6.19	0.865	23.93

Table 2-8 – MOGO Circuit Parameters for the Ideal Class-EF_2 Inverter: Design Case I

Parameter	**Calculation**
$L_f\ [\mu H]$	$L_f = 39.17$[1]
$L_{mr}\ [nH]$	$L_{mr} = \dfrac{1}{2^2 \times 4\pi^2 f_s^2 C_{mr}} = 168.57$
$L_s\ [nH]$	$L_s = \dfrac{Q_{Out} R_L}{x_{L_sC_s} \times 2\pi f_s} = 837.44$
$C_f\ [pF]$	$C_f = 672.59$[1]
$C_{mr}\ [pF]$	$C_{mr} = \dfrac{C_f}{k} = 817.20$
$C_s\ [pF]$	$C_s = \dfrac{1}{x_{L_sC_s}^2 \times 4\pi^2 f_s^2 L_s} = 880.34$
$V_{in}\ [V]$	$V_{in} = 23.93$[1]
D	$D = 0.406$[1]

[1] Directly obtained from the optimizer

The component values for Design Case II and Design Case III are presented in Table 2-9.

Table 2-9 – MOGO Circuit Parameters for the Ideal Class-EF$_2$ Inverter: Design Case II and Design Case III

Design Case II		Design Case III	
Parameter	**Value**	**Parameter**	**Value**
L_f $[\mu H]$	55.19	L_f $[\mu H]$	49.36
L_{mr} $[nH]$	221.71	L_{mr} $[nH]$	90.11
L_s $[nH]$	934.94	L_s $[nH]$	245.77
C_f $[pF]$	218.79	C_f $[pF]$	140.63
C_{mr} $[pF]$	155.34	C_{mr} $[pF]$	95.55
C_s $[pF]$	174.40	C_s $[pF]$	166.98
V_{in} $[V]$	35.37	V_{in} $[V]$	21.12
D	0.370	D	0.394

2.5 Multi-Objective Particle Swarm Optimization (MOPSO) Algorithm

The MOPSO design approach uses the MOPSO function written by Victor Martinez-Cagigal [23] which is based on the work done in [28], [29]. However, since this is a basic implementation of the MOPSO algorithm a function for constraint handling as well as an equation for calculating the inertia coefficient as described in [20] were also implemented. This significantly improved the performance of the MOPSO algorithm as it encourages more exploration of the search space and ensures that only feasible solutions are stored in the repository. The general process of the MOPSO algorithm for this application is presented below along with an accompanying flowchart shown in Figure 2-4.

Initialization

- Generate an initial population of particles with random position and zero velocity: The initial population is a matrix of size $n \times m$ where n is the defined population size and m is the number of variables which represent the duty cycle D, the input voltage V_{in}, the input inductance L_f, the filter capacitance C_f, and various factors which relate to the other inductors and capacitors in the circuit. The position of each particle is equivalent to the value of each variable, and the velocity of each particle determines how much the value of each variable will change every iteration.

- Pass the initial population to the fitness function: The initial population is passed to an intermediate function which separates the matrix into a set of $1 \times m$ particle sets, calls the fitness function n times, and stores the fitness values and constraint violations in an $n \times l$ matrix where l is the number of objective and constraint functions. This matrix is then returned to the MOPSO function for evaluation.

- Evaluate the initial population and check for constraint violations: The fitness function values of each particle set are evaluated based on the domination and the number of constraint violations.

- Save non-dominated solutions to the repository: The particle sets which are non-dominated and which contain the lowest number of constraint violations are saved in the repository as described in [20].

All Subsequent Iterations

- Select a leader: A leader is selected from the repository using a roulette wheel selection.

- Update the velocities and positions of all particles in the population: The velocity and position of each particle is updated per the following equation where s is the velocity, w is the inertia coefficient, H_1 and H_2 are confidence factors, z_1 and z_2 are randomly generated numbers between 0 and 1, p_{best} is the particle's personal best position, G_{best} is the best position of the entire swarm, and p is the particles position.

$$s_{i+1} = w_i s_i + H_1 z_1 (p_{best} - p_i) + H_2 z_2 (G_{best} - p_i) \tag{2.31}$$

Using this velocity vector, the position of each particle is updated using the following equation.

$$p_{i+1} = p_i + s_{i+1} \tag{2.32}$$

- Perform mutation on the population: The particle set undergoes mutation to generate the next population.
- Enforce boundary conditions: Boundary conditions are enforced on the new population so that any particle whose new position is outside the boundary will be placed at the edge before being passed to the fitness function.
- Pass the new population to the fitness function: As above.
- Evaluate the new population and check for constraint violations: As above.
- Update the repository: As above.

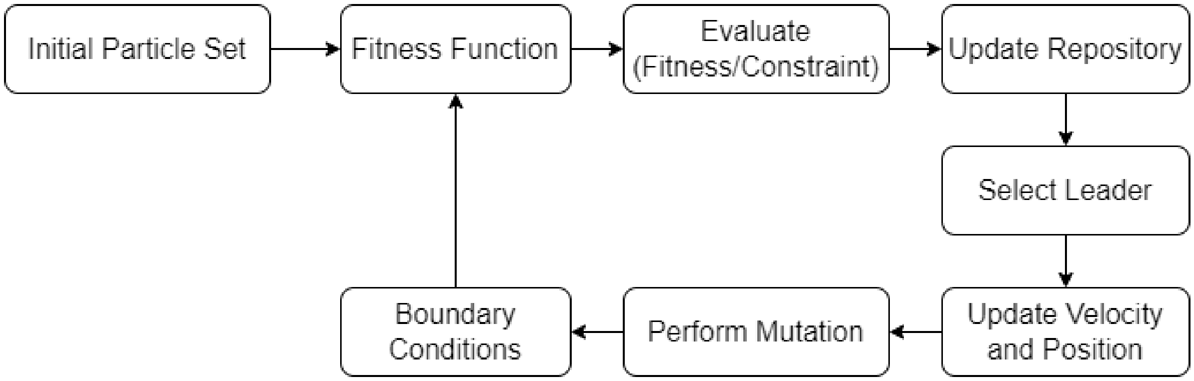

Figure 2-4 – Flowchart of the MOPSO Algorithm

In Section 2.6 a comparison between the MOGO and MOPSO design approaches is presented. To ensure a fair comparison, a constant computational cost is used. The MOGO had a population size of 525 with 100 generations resulting in a computational cost of 52,500.

In the case of the MOPSO algorithm, a population size of 150 with 350 generations was found to yield the same computational cost of 52,500 and was the most suitable values for this application. All other parameters (variables, objective functions, constraint functions, and boundary conditions) remained the same as in the MOGO case presented in sections 2.3.1 to 2.3.5. The MOPSO initialization function and associated functions are provided in Appendix E and Appendix F.

2.5.1 Design Results Using the MOPSO Algorithm

Using the procedure and information presented in Sections 2.3.1 to 2.3.5, the results of the MOPSO design are obtained and compared with the analytical design approach. Tables 2-10 (a-c) show the comparative values for the three design cases.

Table 2-10 (a) – Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case I

Factors	Design Case I *Max f*		
	Analytical Values	**MOPSO Design**	**Percentage Difference**
D	0.372	0.361	2.99%
k	1.567	2.349	49.88%
q_2	2.560	2.388	6.71%
$\frac{1}{\omega R_L C_f}$	5.686	4.937	13.16%
$\frac{1}{\omega R_L C_{MR}}$	8.910	11.596	30.15%
c_p	0.120	0.116	3.21%
V_{Stress}	2.243	2.188	2.46%
I_{Stress}	3.719	3.872	4.11%

Table 2-10 (b) – Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case II.

Factors	Design Case II *Max f*		
	Analytical Values	**MOPSO Design**	**Percentage Difference**
D	0.372	0.392	5.54%
k	1.567	2.468	57.49%
q_2	2.560	2.371	7.38%
$\frac{1}{\omega R_L C_f}$	5.686	5.529	2.76%
$\frac{1}{\omega R_L C_{MR}}$	8.910	13.644	53.14%
c_p	0.120	0.116	3.43%
V_{Stress}	2.243	2.250	0.28%
I_{Stress}	3.719	3.801	2.21%

Table 2-10 (c) – Comparison of the Results of the Analytical and MOPSO Design Approaches: Design Case III.

Factors	Design Case III *Max f*		
	Analytical Values	**MOPSO Design**	**Percentage Difference**
D	0.372	0.3658	6.99%
k	1.567	2.500	59.53%
q_2	2.560	2.366	7.56%
$\frac{1}{\omega R_L C_f}$	5.686	4.777	15.99%
$\frac{1}{\omega R_L C_{MR}}$	8.910	11.941	34.02%
c_p	0.120	0.117	2.38%
V_{Stress}	2.243	2.166	3.45%
I_{Stress}	3.719	3.900	4.87%

It is observed that for each design case the MOPSO method selected high values of k which influenced the associated design factors. Since the THD of the output voltage waveform is considered an optimization objective function, the algorithm favors higher values of k since it was shown in [14] that this reduces the harmonic content of current $i_{L_{mr}}$.

Since k is directly proportional to L_{mr} and thus also directly proportional to the size of its equivalent series resistance, it is believed that the MOPSO algorithm will perform better in future testing where the parasitic elements of components are considered.

2.5.2 Determination of the Inverter Component Values from the MOPSO Designs

The component values for the three design cases are presented in Table 2-11 below. All parameters were calculated using the process presented in Section 2.4.1.

Table 2-11 – MOPSO Optimized Circuit Parameters for the Ideal Class-EF$_2$ Inverter: All Design Cases

Design Case I		Design Case II		Design Case III	
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
L_f $[\mu H]$	34.18	L_f $[\mu H]$	46.30	L_f $[\mu H]$	43.01
L_{mr} $[nH]$	340.26	L_{mr} $[nH]$	400.37	L_{mr} $[nH]$	122.64
L_s $[nH]$	857.93	L_s $[nH]$	997.29	L_s $[nH]$	352.27
C_f $[pF]$	950.88	C_f $[pF]$	212.29	C_f $[pF]$	175.51
C_{mr} $[pF]$	404.87	C_{mr} $[pF]$	86.02	C_{mr} $[pF]$	70.21
C_s $[pF]$	765.04	C_s $[pF]$	155.83	C_s $[pF]$	112.33
V_{in} $[V]$	19.21	V_{in} $[V]$	30.37	V_{in} $[V]$	22.65
D	0.361	D	0.392	D	0.366

2.6 Comparison of MOGO and MOPSO Design Approaches

Each design was simulated in LTSpice using the component values listed in Tables 2-8, 2-9 and 2-11. The results of the parameters of interest from the simulations are recorded in Table 2-12. The output voltage THD is calculated using the first 7 harmonics, and the efficiency, voltage stress, and current stress are defined in (2.31) to (2.33) where all values were taken directly from LTSpice. The waveforms of the switch voltage, switch current, and output voltage from LTSpice simulations of the MOGO (green) and MOPSO (blue) designs are shown in Figs. 2.3 (a-c), 2-4 (a-c), and 2-5 (a-c).

$$\eta = \frac{P_{Out}}{P_{In}} \tag{2.31}$$

$$V_{Stress} = \frac{V_{Sw_{\mathrm{Max}}}}{V_{in}} \tag{2.32}$$

$$I_{Stress} = \frac{I_{Sw_{Max}}}{I_{In}} \tag{2.33}$$

Table 2-12 – Comparison of the Results of the MOGO and MOPSO Design Approaches

Parameter	Design Case I		Design Case II		Design Case III	
$P_{Out_{Desired}}$ $[W]$	23	23	40	40	25	25
$P_{Out_{Actual}}$ $[W]$	23.898	24.132	38.47	37.98	26.53	24.93
η	99.87%	99.77%	99.84%	99.84%	99.70%	99.27%
V_{Stress}	2.540	2.188	2.241	2.250	2.337	2.166
I_{Stress}	3.136	3.872	3.523	3.801	3.484	3.900
Q_{Out}	6.169	6.698	7.322	8.000	5.481	8.000
THD	4.43%	2.36%	2.32%	2.08%	3.23%	1.86%

Despite the variation in component values obtained from the MOGO and MOPSO design algorithms, each circuit scored very well in the measured values of interest. The minimum efficiency was 99.27% and occurred during MOPSO design case III, the THD of the output voltage waveform never exceeded 5% with most of the designs maintaining a THD of less than 2.5%.

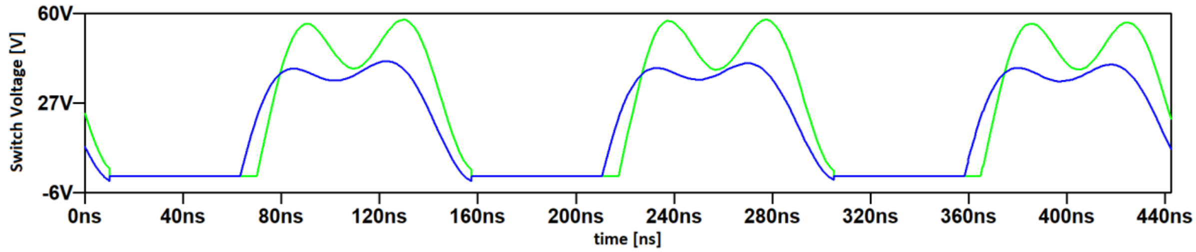

Figure 2-5 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

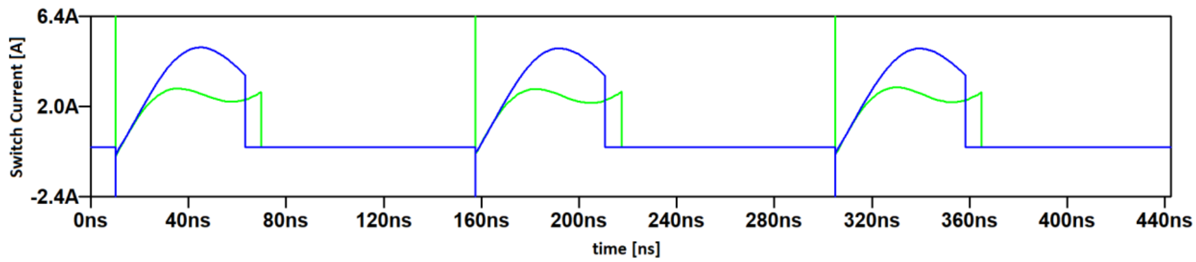

Figure 2-5 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

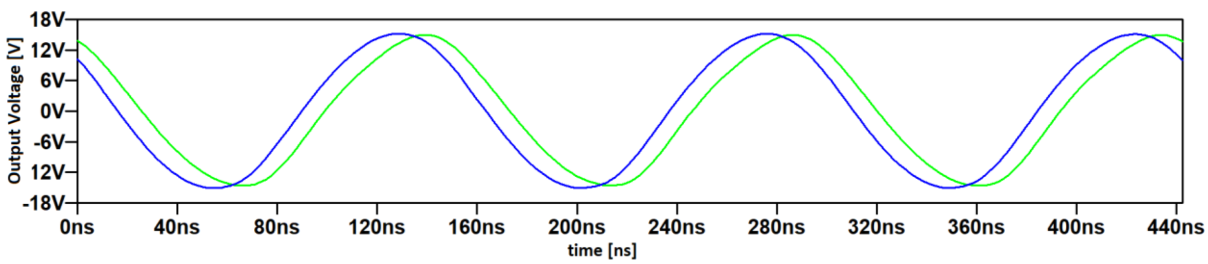

Figure 2-5 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

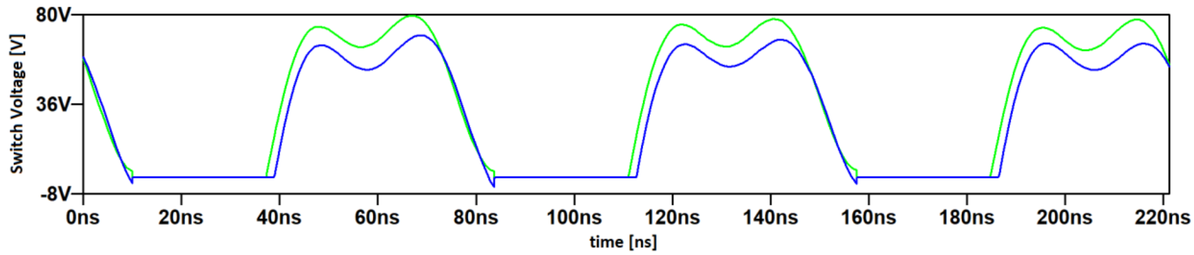

Figure 2-6 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

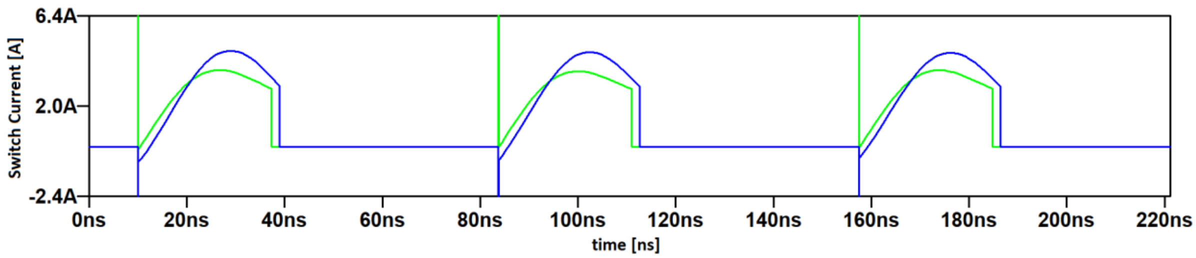

Figure 2-6 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

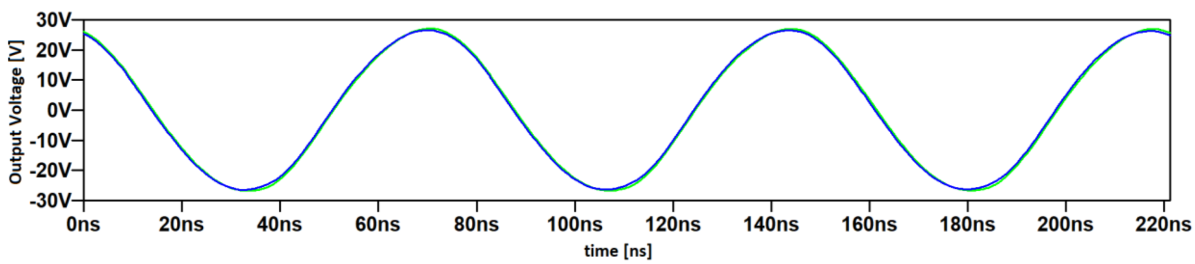

Figure 2-6 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

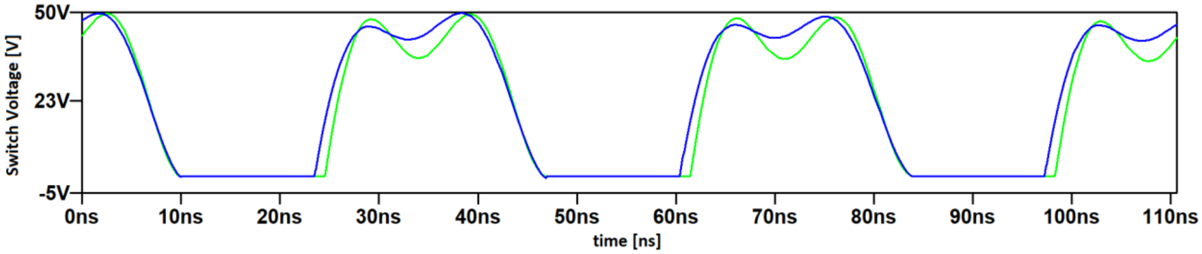

Figure 2-7 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

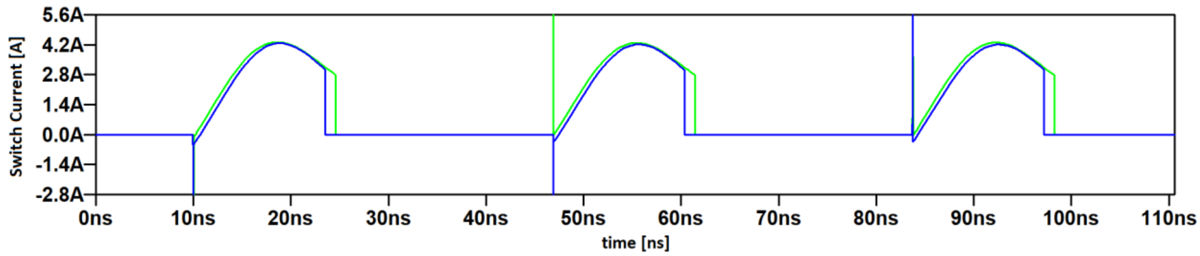

Figure 2-7 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

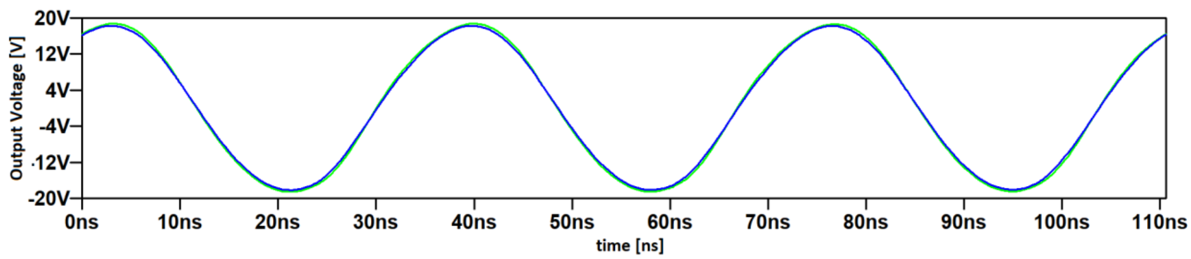

Figure 2-7 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

The above figures showcase the success of the optimization-based design approach for the ideal case. In each design case, the circuit maintained ZVS and ZDVS operation to an acceptable level of error and had relatively sinusoidal output voltage waveforms.

2.7 Summary

In Chapter 2, the design of the ideal Class-EF_2 inverter was investigated. A state-space model was derived and tested using MATLAB, and it was shown to provide an accurate representation of the operation of the circuit by comparison with LTSpice simulation.

The state-space model was then used with both the MOGO and MOPSO algorithms to design three circuits with differing frequency, power, and load specifications. It was shown that the proposed optimization-based design approaches provided comparable results with previously determined analytical values and successfully designed each circuit to the stated specifications.

CHAPTER 3

THE PARASITIC CLASS-EF$_2$ INVERTER

In this chapter, the proposed optimization design approaches introduced in Chapter 2 are extended to the design of the Class-EF$_2$ inverter which considers the parasitic elements of the components. A state-space model that includes the parasitic elements of the Class-EF$_2$ inverter is presented. The validity of this model is confirmed by comparison with LTSpice simulation. First, an approach for determining the parasitic elements is described. This is followed by a comparison of the switch voltage waveforms from the solution of the state-space model in MATLAB and LTSpice simulation.

The developed model with the parasitic elements is tested with the MOGO and MOPSO design approaches for the three design cases presented in the previous chapter. The results demonstrate the importance of considering parasitic elements in high frequency circuit designs and further validate the optimization-based design approach.

3.1 State-Space Model of the Class EF$_2$ Inverter Including Parasitic Elements

In the previous analysis and studies of the Class-EF$_2$ inverter, the components in the circuit were considered ideal. In this section, the parasitic elements of the components are accounted for in developing the state-space model of the circuit. The inductors and capacitors are replaced with their parasitic models as shown in Fig. 3-1. Each capacitor now has an added equivalent series resistance (ESR), R_s and an equivalent series inductance (ESL), L_s. Each inductor now has an added equivalent series resistance (ESR), R_x as well as a parallel winding capacitance, C_x.

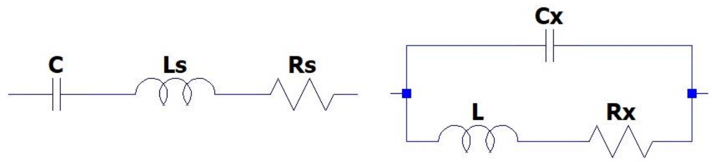

Figure 3-1 – Model of Capacitor (Left) and Inductor (Right) with Parasitic Elements

The circuit model of the Class-EF$_2$ inverter including the parasitic elements is shown in Fig. 3-2. The switch is represented by its ON and OFF resistances as stated in (2.1).

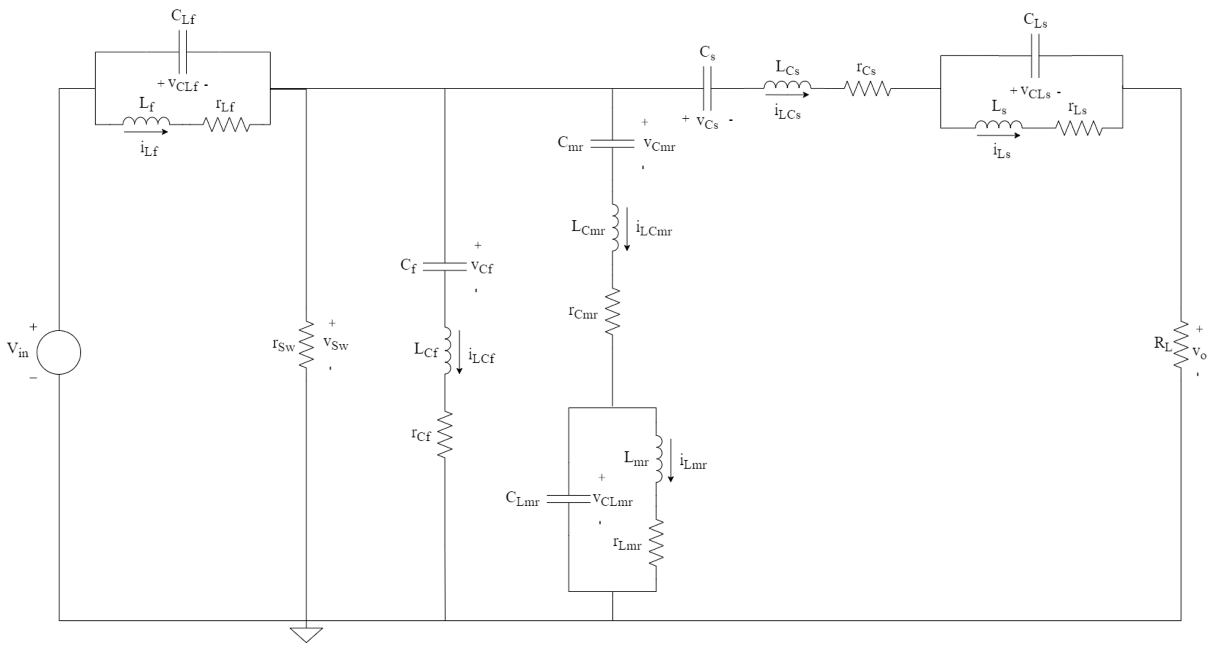

Figure 3-2 – Circuit Diagram of the Class-EF$_2$ Inverter with Parasitic Elements

Using the same procedure as outlined in Section 2.1, the state vector **X** is made up of the capacitor voltages and the inductor currents, and the source voltage V$_{in}$ is represented by the state vector **U**, as shown in (3.1) and (3.2).

$$\mathbf{X} = \begin{bmatrix} v_{C_f} & v_{C_{mr}} & v_{C_s} & v_{C_{L_f}} & v_{C_{L_{mr}}} & v_{C_{L_s}} & i_{L_f} & i_{L_{mr}} & i_{L_s} & i_{L_{C_f}} & i_{L_{C_{mr}}} & i_{L_{C_s}} \end{bmatrix}^T \tag{3.1}$$

$$\mathbf{U} = V_{in} \tag{3.2}$$

The differential equations for each state variable are derived in terms of the other state variables. This process is presented in (3.3) to (3.14).

$$\frac{dv_{C_f}}{dt} = \frac{i_{C_f}}{C_f} = \frac{i_{L_{C_f}}}{C_f} \tag{3.3}$$

$$\frac{dv_{C_{MR}}}{dt} = \frac{i_{C_{mr}}}{C_{mr}} = \frac{i_{L_{C_{mr}}}}{C_{mr}} \tag{3.4}$$

$$\frac{dv_{C_s}}{dt} = \frac{i_{C_s}}{C_s} = \frac{i_{L_{C_s}}}{C_s} \tag{3.5}$$

$$\frac{dv_{C_{L_f}}}{dt} = \frac{i_{C_{L_f}}}{C_{L_f}} = \frac{V_{in} - v_{C_{L_f}} + r_{sw}(i_{L_{C_f}} + i_{L_{C_{mr}}} + i_{L_{C_s}} - i_{L_f})}{r_{sw} C_{L_f}} \tag{3.6}$$

$$\frac{dv_{C_{L_{MR}}}}{dt} = \frac{i_{C_{L_{MR}}}(t)}{C_{L_{MR}}} = \frac{i_{L_{C_{mr}}} - i_{L_{mr}}}{C_{L_{mr}}} \tag{3.7}$$

$$\frac{dv_{C_{L_s}}}{dt} = \frac{i_{C_{L_s}}(t)}{C_{L_s}} = \frac{i_{L_{C_s}} - i_{L_s}}{C_{L_s}} \tag{3.8}$$

$$\frac{di_{L_f}}{dt} = \frac{v_{L_f}(t)}{L_f} = \frac{v_{C_{L_f}} - i_{L_f} r_{L_f}}{L_f} \tag{3.9}$$

$$\frac{di_{L_{MR}}}{dt} = \frac{v_{L_{MR}}(t)}{L_{MR}} = \frac{v_{C_{L_{mr}}} - i_{L_{mr}} r_{L_{mr}}}{L_{mr}} \tag{3.10}$$

$$\frac{di_{L_s}}{dt} = \frac{v_{L_s}(t)}{L_s} = \frac{v_{C_{L_s}} - i_{L_s} r_{L_s}}{L_s} \tag{3.11}$$

$$\frac{di_{L_{C_f}}}{dt} = \frac{v_{L_{C_f}}(t)}{L_{C_f}} = \frac{V_{in} - v_{C_f} - v_{C_{L_f}} - i_{L_{C_f}} r_{C_f}}{L_{C_f}} \tag{3.12}$$

$$\frac{di_{L_{C_{MR}}}}{dt} = \frac{v_{L_{C_{MR}}}(t)}{L_{C_{MR}}} = \frac{V_{in} - v_{C_{mr}} - v_{C_{L_f}} - v_{C_{L_{mr}}} - i_{L_{C_{mr}}} r_{C_{mr}}}{L_{C_{mr}}} \tag{3.13}$$

$$\frac{di_{L_{C_s}}}{dt} = \frac{v_{L_{C_s}}(t)}{L_{C_s}} = \frac{V_{in} - v_{C_s} - v_{C_{L_f}} - v_{C_{L_s}} - i_{L_{C_s}}\left(R_L + r_{C_s}\right)}{L_{C_s}} \tag{3.14}$$

Substituting (3.3) to (3.14) into (2.10) and (2.11) produces a 12^{th} order SIMO system. The state vectors and matrices are defined in (3.15) to (3.18).

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_f} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_{MR}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_s} \\ 0 & 0 & 0 & \frac{-1}{r_{Sw}C_{L_f}} & 0 & 0 & \frac{-1}{C_{L_f}} & 0 & 0 & \frac{1}{C_{L_f}} & \frac{1}{C_{L_f}} & \frac{1}{C_{L_f}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{C_{L_{MR}}} & 0 & 0 & \frac{1}{C_{L_{MR}}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{C_{L_s}} & 0 & 0 & \frac{1}{C_{L_s}} \\ 0 & 0 & 0 & \frac{1}{L_f} & 0 & 0 & \frac{-r_{L_f}}{L_f} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{L_{MR}} & 0 & 0 & \frac{-r_{L_{MR}}}{L_{MR}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{L_s} & 0 & 0 & \frac{-r_{L_s}}{L_s} & 0 & 0 & 0 \\ \frac{-1}{L_{C_f}} & 0 & 0 & \frac{-1}{L_{C_f}} & 0 & 0 & 0 & 0 & 0 & \frac{-r_{C_f}}{L_{C_f}} & 0 & 0 \\ 0 & \frac{-1}{L_{C_{MR}}} & 0 & \frac{-1}{L_{C_{MR}}} & \frac{-1}{L_{C_{MR}}} & 0 & 0 & 0 & 0 & 0 & \frac{-r_{C_{MR}}}{L_{C_{MR}}} & 0 \\ 0 & 0 & \frac{-1}{L_{C_s}} & \frac{-1}{L_{C_s}} & 0 & \frac{-1}{L_{C_s}} & 0 & 0 & 0 & 0 & 0 & \frac{-(r_{C_s}+R)}{L_{C_s}} \end{bmatrix} \tag{3.15}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{r_{Sw}C_{L_f}} & 0 & 0 & 0 & 0 & 0 & \frac{1}{L_{C_f}} & \frac{1}{L_{C_{MR}}} & \frac{1}{L_{C_s}} \end{bmatrix}^T \tag{3.16}$$

$$C = I_{12} \tag{3.17}$$

$$D = \vec{0} \tag{3.18}$$

Equations (3.15) and (3.16) contain the circuit components of the parasitic Class-EF_2 Inverter, I_{12} in (3.17) represents the 12^{th} order identity matrix, and $\vec{0}$ in (3.18) represents the zero vector. The order of the state-space system has doubled from the addition of the parasitic elements.

3.2 Solving the model

3.2.1 Determination of the Parasitic Elements

To determine the parasitic elements of the Class-EF_2 inverter, the following assumptions are made:

- Inductor L_f has a constant ESR of 220 mΩ based on data sheet information of similar sized inductors [30], [31] and contains at least 10 turns,
- Inductors L_{mr} and L_s have a quality factor of 150 and contain at least 10 turns,
- All capacitors have a constant ESR of 50 mΩ based on data sheet information of similar sized capacitors [32].

3.2.1.1 Parasitic Elements of Inductors

The ESR of inductors L_{mr} and L_s were calculated using (3.7) by assuming a constant inductor quality factor of 150 as recommended by experts in the field.

$$r_L = \frac{\omega L}{Q} = \frac{2\pi f_s L}{150} \tag{3.7}$$

The parasitic capacitance (or stray capacitance) C_s was approximated using the process presented in [33] with a combination of AWG 16 and AWG 18 wires, as well as T68-6 and T50-6 RF toroidal cores. The turn-to-turn capacitance can be approximated using (3.8) where parameter θ^* is defined in (3.9). If the inductor contains at least 10 turns, the overall parasitic capacitance converges as shown in (3.10) [33].

$$C_{tt} = C_{ttc} + C_{ttg} = \varepsilon_0 l_t \left[\frac{\varepsilon_r \theta^*}{\ln\left(\frac{D_o}{D_c}\right)} + \cot\left(\frac{\theta^*}{2}\right) - \cot\left(\frac{\pi}{12}\right) \right] \tag{3.8}$$

$$\theta^* = \arccos\left(1 - \frac{ln\left(\frac{D_o}{D_c}\right)}{\varepsilon_r} \right) \tag{3.9}$$

$$C_s = 1.366C_{tt} \tag{3.10}$$

In the above equations, l_t, D_o, D_c, ε_0, ε_r are properties of the core and conductor, and C_{ttc} and C_{ttg} represent the capacitances of the middle and side parts of the proposed basic cell model [3].

Under this assumption, the value of C_x varied between 1.95 pF and 2.26 pF depending on the wire and toroid combination. Due to the small variance of this value, inductors L_{mr} and L_s were assumed to have an average 2.1 pF parasitic capacitance to reduce the number of calculations performed by the optimizer during each iteration. The parasitic capacitance of inductor L_f was calculated using the same method with a T106-2 toroidal core. This resulted in an average parasitic capacitance of 3.35 pF.

3.2.1.2 Parasitic Elements of Capacitors

The ESR for all capacitors was assumed to be 50 mΩ based on observations from RF capacitor data sheets [32].

The parasitic inductance of each capacitor was approximated from manufacturer datasheets using provided self-resonant frequency (SRF) plots. Equation (3.11) is used to calculate any parasitic inductance L_c where L_2 and C_2 are constant points on the SRF plot, m is the estimated slope which is assumed to be constant in all calculations, and C is the nominal capacitance value (C_f, C_{mr}, or C_s) [32].

$$L_C = 10^{\left(\log(L_2) - \left(\frac{\log(C_2) - \log(C)}{m}\right)\right)} \approx 10^{\left(-9.897 - \left(\frac{-9.699 - \log(C)}{-7.213}\right)\right)} \tag{3.11}$$

Equation (3.11) produces parasitic inductance values in the range of 100 pH to 300 pH depending on the size of capacitor C.

3.2.2 Simulation of the Parasitic Class-EF$_2$ Inverter

To validate the state-space model of the parasitic Class-EF$_2$ inverter, the results from ideal MOGO Design Case II are used to solve the state-space model. The values of the parameters including the parasitic elements are shown in Table 3-1. The circuit is then fully defined and simulated in LTSpice for comparison.

Table 3-1 – Nominal Component Values and Parasitic Elements used for LTSpice and MATLAB Simulation of the Parasitic Class-EF$_2$ Inverter

Parameter	Nominal Value	ESR	Parasitic Element
L_f	49.37	220 mΩ	3.35 pF
L_{mr}	249.64	141.80 mΩ	2.1 pF
L_s	723.25	410.80 mΩ	2.1 pF
C_f	214.95	50 mΩ	125.44 pH
C_{mr}	137.96	50 mΩ	133.94 pH
C_s	247.91	50 mΩ	123.00 pH
D	0.3721	-	-
V_{in}	38.22	-	-

3.2.3 Comparison of MATLAB and LTSpice Simulation Results

A comparison of the resulting switch voltage, switch current, and output voltage waveforms from both simulations are shown in Fig. 3-3 (a-c). The plots demonstrate the accuracy of the parasitic state-space model.

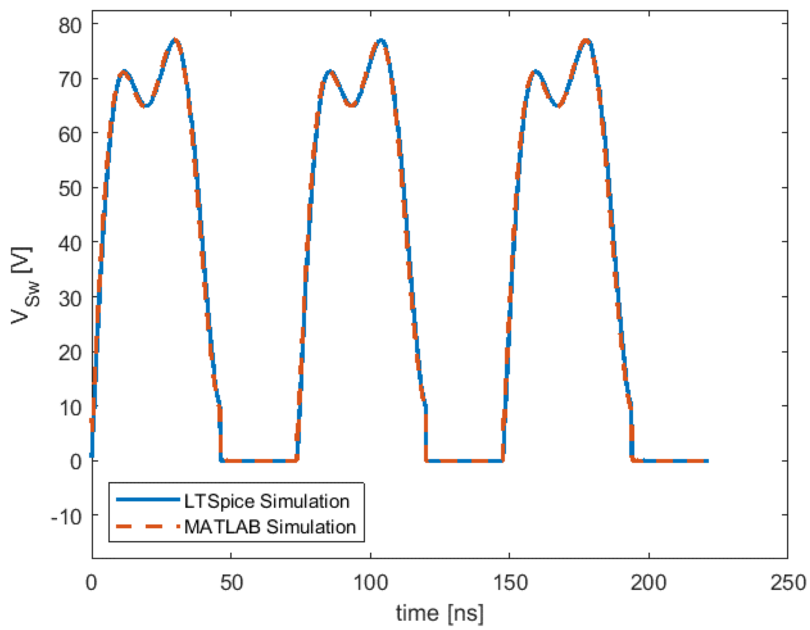

Figure 3-3 (a) – Switch Voltage Waveform from the Parasitic State-Space Model vs. LTSpice Simulation

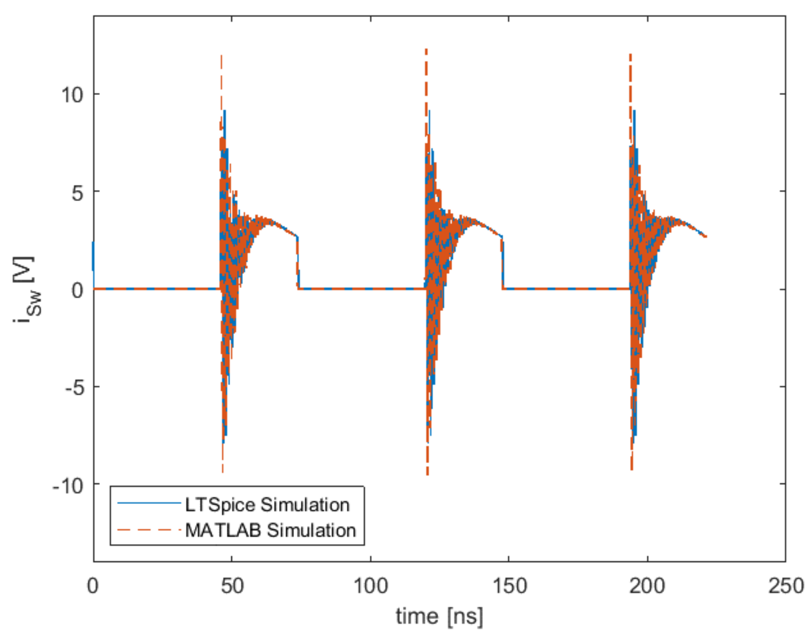

Figure 3-3 (b) – Switch Current Waveform from the Parasitic State-Space Model vs. LTSpice Simulation

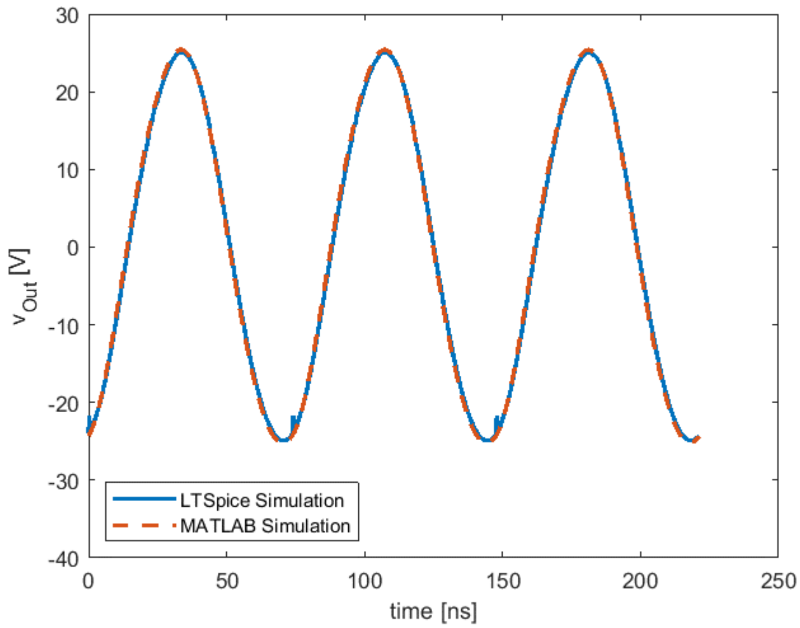

Figure 3-3 (c) – Output Voltage Waveform from the Parasitic State-Space Model vs. LTSpice Simulation

Figures 3-3 (a-c) validate the accuracy of the parasitic Class-EF_2 inverter state-space model. They also demonstrate the impact of the parasitic elements on the operation of the circuit as it no longer exhibits ZVS or ZDVS operation, and there is significant ringing in the switch current waveform.

3.3 Design Results Using the MOGO Algorithm

The MOGO design method was tested using the three design cases presented in Table 2-4. The results of the MOGO designs are obtained and compared with the ideal model. Table 3-2 shows the comparative values for each design case. Code for the Parasitic Class-EF_2 fitness function can be found in Appendix B.

Table 3-2 - Comparison of the Results of the Ideal and Parasitic MOGO Designs

Factors	Design Case I		Design Case II		Design Case III	
	Ideal	Parasitic	Ideal	Parasitic	Ideal	Parasitic
D	0.406	0.394	0.370	0.357	0.394	0.393
k	0.823	1.133	1.408	1.712	1.472	1.128
q_2	2.977	2.744	2.615	2.517	2.592	2.747
$\frac{1}{\omega R_L C_f}$	6.980	6.587	5.365	5.390	5.961	6.503
$\frac{1}{\omega R_L C_{MR}}$	5.745	7.463	7.556	9.230	8.774	7.336
c_p	0.128	0.117	0.125	0.113	0.122	0.111
V_{Stress}	2.540	2.280	2.241	2.210	2.337	2.314
I_{Stress}	3.136	3.385	3.523	3.727	3.484	3.496

The factors for Design Case I and Design Case III are both very similar selecting values of k falling roughly halfway between the max c_p and max f design procedures. Design Case II was a near perfect match with the max f case, however.

3.3.1 Determination of the Inverter Component Values from the MOGO Designs

The nominal values of all components were calculated as shown in 2.4.1, the values of which are shown in Table 3-3.

Table 3-3 - Comparison of the Component Values of the Ideal and Parasitic MOGO Designs

Parameter	Design Case I		Design Case II		Design Case III	
	Ideal	**Parasitic**	**Ideal**	**Parasitic**	**Ideal**	**Parasitic**
$L_f\ [\mu H]$	39.17	38.05	55.19	74.75	49.36	42.57
$L_{mr}\ [nH]$	168.57	218.99	221.71	270.83	90.11	75.34
$L_s\ [nH]$	837.44	890.74	934.94	545.32	245.77	203.08
$C_f\ [pF]$	672.59	712.79	218.79	217.75	140.63	128.91
$C_{mr}\ [pF]$	817.20	629.08	155.34	127.16	95.55	114.28
$C_s\ [pF]$	880.34	766.33	174.40	347.8	166.98	225.20
$V_{in}\ [V]$	23.93	21.88	35.37	37.41	21.12	25.66
D	0.406	0.394	0.370	0.357	0.394	0.393

Notable differences occur in most component values showcasing the effect of the parasitic elements on the design of the Class-EF$_2$ inverter. Design Case II and Design Case III see a significant reduction in the value of L_s likely due to its large impact on the circuit efficiency with the added ESR. In each design case, the changes to C_f and C_{mr} can be attributed to the variance in k.

3.3.1.1 Determination of the Parasitic Elements for the MOGO Designs

A demonstration of the calculation of all parasitic elements for Design Case I is presented in Table 3-4. The parasitic elements for Design Case II and Design Case III are presented in Table 3-5.

Table 3-4 – Demonstration of the Calculation of all Parasitic Elements for Design Case I

Parameter	Calculation
L_{C_f} $[pH]$	$L_{C_f} = 10^{\left(-9.897-\left(\frac{-9.699-\log(C_f)}{-7.213}\right)\right)} = 106.23$
$L_{C_{mr}}$ $[pH]$	$L_{C_{mr}} = 10^{\left(-9.897-\left(\frac{-9.699-\log(C_{mr})}{-7.213}\right)\right)} = 108.09$
L_{C_s} $[pH]$	$L_{C_s} = 10^{\left(-9.897-\left(\frac{-9.699-\log(C_s)}{-7.213}\right)\right)} = 105.17$
$r_{L_{mr}}$ $[m\Omega]$	$r_{L_{mr}} = \frac{2\pi f_s L_{mr}}{150} \times \frac{10^3 m\Omega}{\Omega} = 62.2$
r_{L_s} $[m\Omega]$	$r_{L_s} = \frac{2\pi f_s L_s}{150} \times \frac{10^3 m\Omega}{\Omega} = 253.0$

Table 3-5 – Values of all Parasitic Elements for Design Case II and Design Case III

Parameter	Value	
	Design Case II	Design Case III
L_{C_f} $[pH]$	125.22	134.65
$L_{C_{mr}}$ $[pH]$	134.91	136.92
L_{C_s} $[pH]$	117.34	124.63
$r_{L_{mr}}$ $[m\Omega]$	153.8	85.6
r_{L_s} $[m\Omega]$	309.7	230.7

The parasitic capacitance C_{L_f} is assumed to be 3.35 pF, the parasitic capacitances $C_{L_{mr}}$ and C_{L_s} are assumed to be 2.1 pF, the ESR of all capacitors is assumed to be 50 mΩ, and the ESR of inductor L_f is assumed to be 220 mΩ as stated in 3.1.2.

3.4 Design Results Using the MOPSO Algorithm

The MOPSO design method was tested using the three design cases presented in Table 2-4. The results of the MOPSO designs are obtained and compared with the ideal model. Table 3-6 show the comparative values for each design case.

Table 3-6 - Comparison of the Results of the Ideal and Parasitic MOPSO Designs

Factors	Design Case I		Design Case II		Design Case III	
	Ideal	Parasitic	Ideal	Parasitic	Ideal	Parasitic
D	0.361	0.399	0.392	0.367	0.366	0.358
k	2.349	1.154	2.468	1.438	2.500	1.562
q_2	2.388	2.733	2.371	2.604	2.366	2.561
$\frac{1}{\omega R_L C_f}$	4.937	6.563	5.529	5.673	4.777	5.491
$\frac{1}{\omega R_L C_{MR}}$	11.596	7.571	13.644	8.160	11.941	8.577
c_p	0.116	0.117	0.116	0.114	0.117	0.104
V_{Stress}	2.188	2.333	2.250	2.172	2.166	2.172
I_{Stress}	3.872	3.346	3.801	3.701	3.900	4.009

As was expected, the value of k was significantly decreased in all design cases due to the large loss incurred by the ESR of L_{mr}. The factors proposed for Design Case I were nearly identical (within 2.5%) to the ones from the MOGO design, where the value of k fell roughly halfway between the max c_p and max f design procedures from Table 3-5. However, Design Case II and Design Case III were both very close matches with the max f design procedure.

The discrepancy between the design factors proposed by the MOGO and MOPSO design methods for Design Case II and Design Case III are likely due to the arbitrary stopping condition placed on the algorithms (i.e., computational cost). If the computational cost was increased, or a different stopping condition was used, the MOGO and MOPSO factors would likely be in agreeance.

3.4.1 Determination of the Inverter Component Values from the MOPSO Designs

The component values for the three design cases are presented in Table 3-7 below. All nominal parameters were calculated using the process shown in 2.2.5.

Table 3-7 - Comparison of the Component Values of the Ideal and Parasitic MOPSO Designs

Parameter	Design Case I		Design Case II		Design Case III	
	Ideal	Parasitic	Ideal	Parasitic	Ideal	Parasitic
$L_f\ [\mu H]$	34.18	41.45	46.30	73.86	43.01	77.34
$L_{mr}\ [nH]$	340.26	222.17	400.37	239.45	122.64	88.08
$L_s\ [nH]$	857.93	636.22	997.29	749.49	352.27	146.19
$C_f\ [pF]$	950.88	715.31	212.29	206.89	175.51	152.68
$C_{mr}\ [pF]$	404.87	620.07	86.02	143.83	70.21	97.75
$C_s\ [pF]$	765.04	1129.8	155.83	238.71	112.33	412.16
$V_{in}\ [V]$	19.21	19.34	30.37	42.55	22.65	28.22
D	0.361	0.399	0.392	0.367	0.366	0.358

Like the results in Table 3-5, many of the component values seen drastic changes between the ideal and parasitic designs. The reduction in the value of k caused C_f, C_{mr}, and L_{mr} to change by upwards of 50%. The value of L_s was reduced across the board due to the impact of its ESR on the efficiency of the circuit. This caused the value of C_s to increase in each design case since it is inversely proportional to L_s.

3.4.1.1 Determination of the Parasitic Elements for the MOPSO Designs

All parasitic elements were calculated using the process shown in 3.2.1 and are presented in Table 3-8.

Table 3-8 – Values of all Parasitic Elements for each Design Case: MOPSO

Parameter	Design Case I	Design Case II	Design Case III
	Value	Value	Value
L_{C_f} $[pH]$	106.18	126.11	131.53
$L_{C_{mr}}$ $[pH]$	108.31	132.63	139.92
L_{C_s} $[pH]$	99.66	123.63	114.61
$r_{L_{mr}}$ $[m\Omega]$	63.1	136.0	100.1
r_{L_s} $[m\Omega]$	180.7	425.7	166.1

The parasitic capacitance C_{L_f} is assumed to be 3.35 pF, the parasitic capacitances $C_{L_{mr}}$ and C_{L_s} are assumed to be 2.1 pF, the ESR of all capacitors is assumed to be 50 mΩ, and the ESR of inductor L_f is assumed to be 220 mΩ as stated in 3.1.2.

3.5 Comparison of MOGO and MOPSO Design Approaches

Each design was simulated in LTSpice using the component values listed in Tables 3-3 to 3-5, 3-7, and 3-8. The results from the simulations are recorded in Table 3-9 using the process described in 2.6 and waveforms for the switch voltage, switch current, and output voltage are shown in Figs. 3-4 (a-c), 3-5 (a-c), and 3-6 (a-c).

Table 3-9 – Comparison of the Results of the MOGO and MOPSO Design Approaches

Parameter	Design Case I		Design Case II		Design Case III	
	MOGO	MOPSO	MOGO	MOPSO	MOGO	MOPSO
$P_{Out_{Desired}}$	23	23	40	40	25	25
$P_{Out_{Actual}}$	21.964	21.246	37.246	38.553	26.328	23.949
η	91.72%	92.87%	95.11%	93.94%	93.80%	95.02%
V_{Stress}	2.280	2.333	2.210	2.172	2.314	2.172
I_{Stress}	3.385	3.346	3.727	3.701	3.496	4.009
Q_{Out}	6.819	4.746	3.960	5.603	4.290	2.690
THD	3.23%	4.47%	3.76%	2.73%	4.15%	5.10%

As can be seen, the parasitic design cases for both the MOGO and MOPSO algorithms performed well in all measured values of interest. The inclusion of the parasitic elements resulted in lower efficiencies for all design cases, the minimum being 91.72% during MOGO design case I. The parasitic elements also present a clear trade-off between the THD of the output voltage waveform and the efficiency of the circuit. In all design cases, the circuit with the higher output voltage THD also had the higher efficiency and the lower value of Q_{out}. This is expected since as the value of Q_{out} increases, so to does the value of L_s and its ESR.

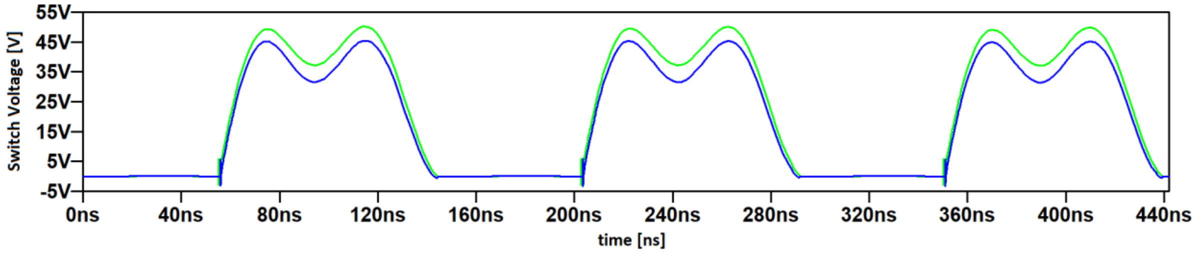

Figure 3-4 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

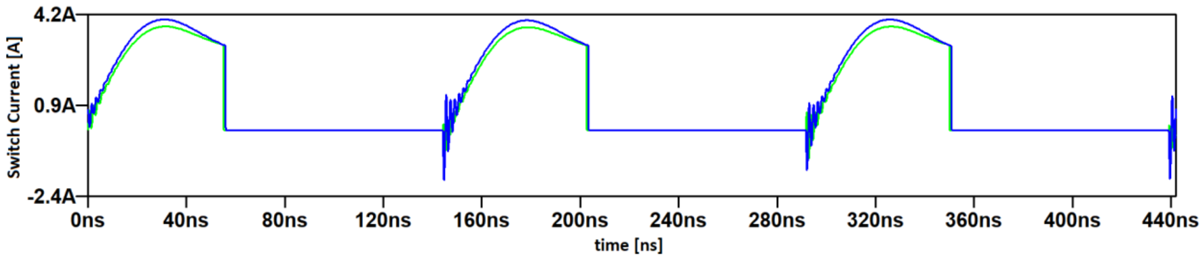

Figure 3-4 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

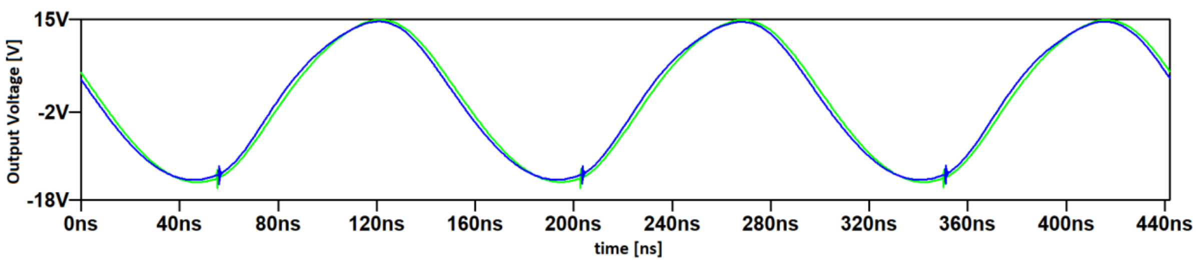

Figure 3-4 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

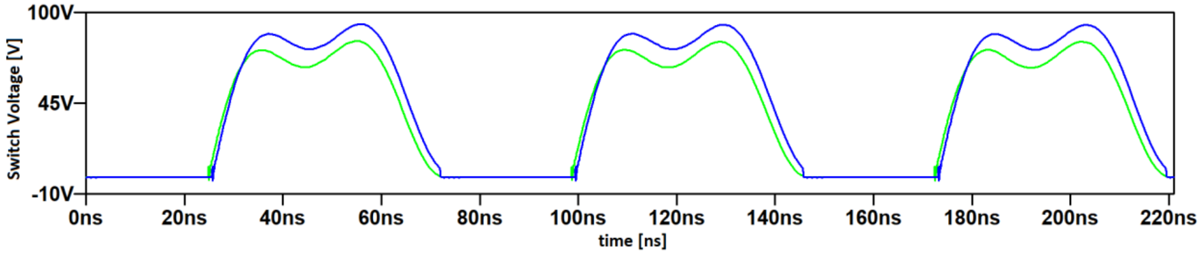

Figure 3-5 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

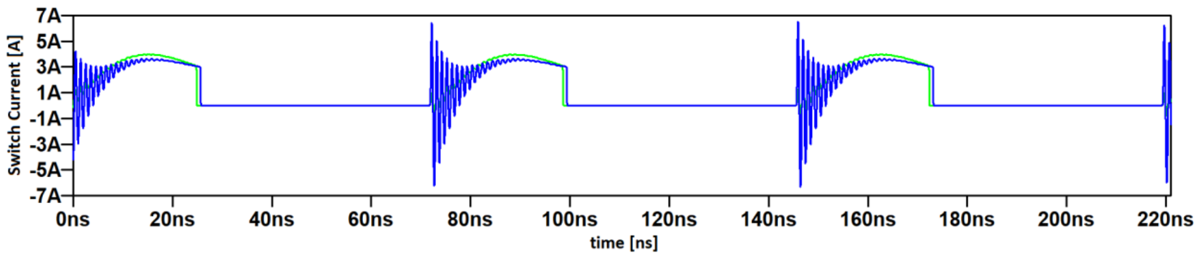

Figure 3-5 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

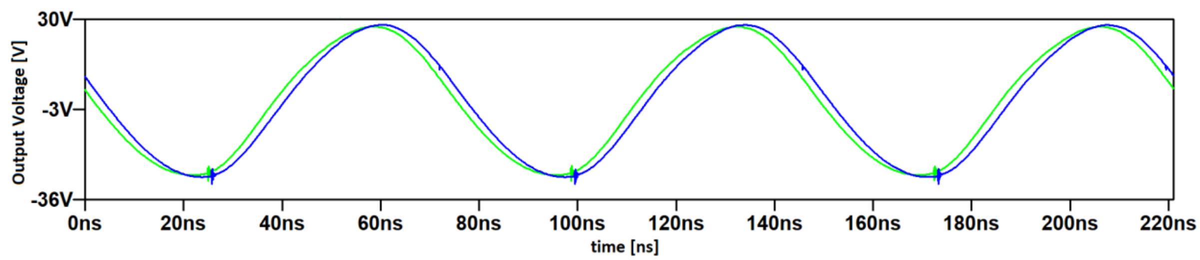

Figure 3-5 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

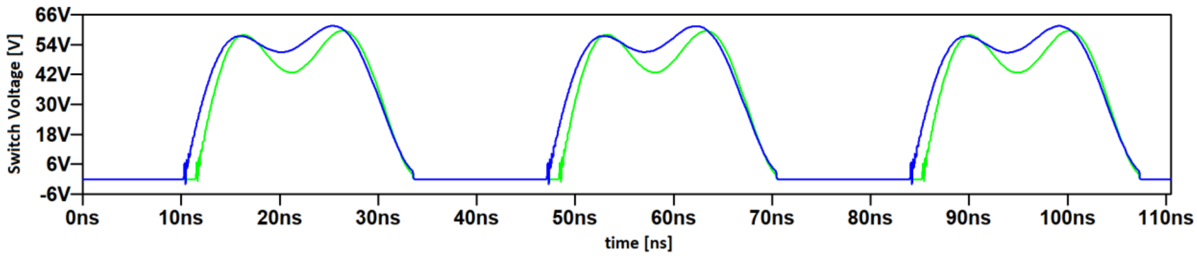

Figure 3-6 (a) – Switch Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

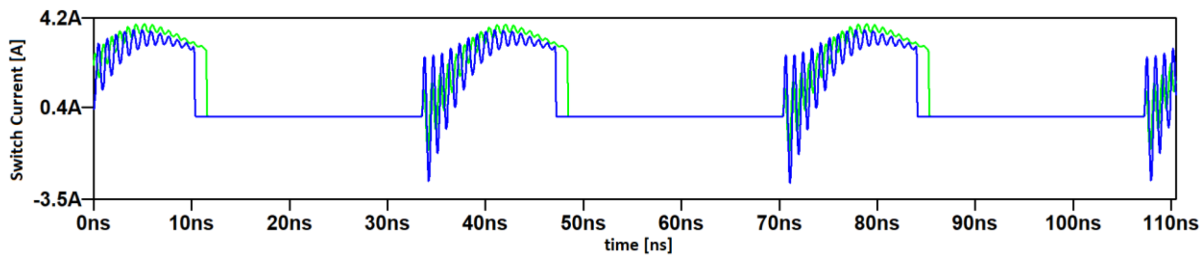

Figure 3-6 (b) – Switch Current Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

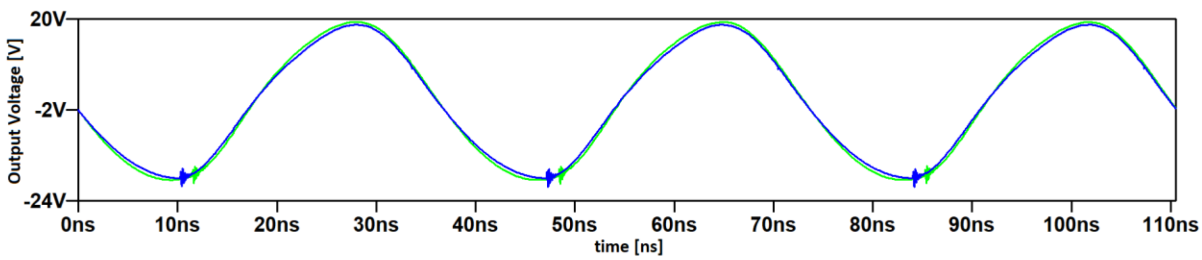

Figure 3-6 (c) – Output Voltage Waveform of the Parasitic Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

The above figures showcase the success of the optimization-based design approach for the parasitic Class-EF$_2$ Inverter. In each design case, the circuits maintained ZVS and ZDVS operation to an acceptable level of error and had relatively sinusoidal output voltage waveforms. However, Fig. 3-5 (b) and Fig. 3-6 (b) show a noticeable amount of ringing in the switch current waveform of the

MOPSO designs which shows how big of an impact even a small amount of hard switching can have when parasitic elements are considered.

3.6 Summary

In Chapter 3, the design of the Parasitic Class-EF_2 inverter was investigated. A 12th order state-space model was proposed which considered the parasitic elements of all components in the circuit and was validated through LTSpice simulation.

The new state-space model was then used with both the MOGO and MOPSO algorithms to design three circuits with differing frequency, power, and load specifications. The proposed optimization-based design approaches provided circuits which performed well in all measured values of interest and successfully adhered to the design specifications.

The addition of the parasitic elements caused large variations in the component values from the ideal designs in the previous chapter. It also demonstrated the negative effects they cause in high-frequency circuits such as the loss of ZVS and ZDVS operation and ringing.

CHAPTER 4

THE PRACTICAL CLASS-EF$_2$ INVERTER

In this chapter, the proposed optimization-based design approaches are further extended to the design of the practical Class-EF$_2$ inverter which considers the parasitic elements of all components as well as the dynamics of the switch. A state-space model which includes the model of the switch is presented. First, a method of estimating the internal capacitances and resistances of the switch is described. Then the model is validated by a comparison of the switch and output waveforms from the solution of the state-space model in MATLAB and by LTSpice simulation.

The developed model with the parasitic elements and switching dynamics is tested with the MOGO and MOPSO design approaches for the three design cases presented in Chapter 2. The results demonstrate the importance of considering the internal capacitances and resistances of the switch when designing high frequency circuits and further validates the optimization-based design approach.

4.1 State-Space model of the Parasitic Class-EF$_2$ Inverter Including Switch Dynamics

In the previous chapters, the switching element of the Class-EF$_2$ inverter was represented as an ON/OFF switch without considering the internal resistances and capacitances. In this section, the parameters of the switch are accounted for in developing the state-space model of the circuit. The switch model now includes the gate-to-source capacitor C_{gs}, the drain-to-source capacitance C_{ds}, the gate resistance r_g, the source resistance r_s and the on and off resistance r_{sw}. The circuit diagram of the new switch model with gate driver signal v_g is shown in Fig. 4-1.

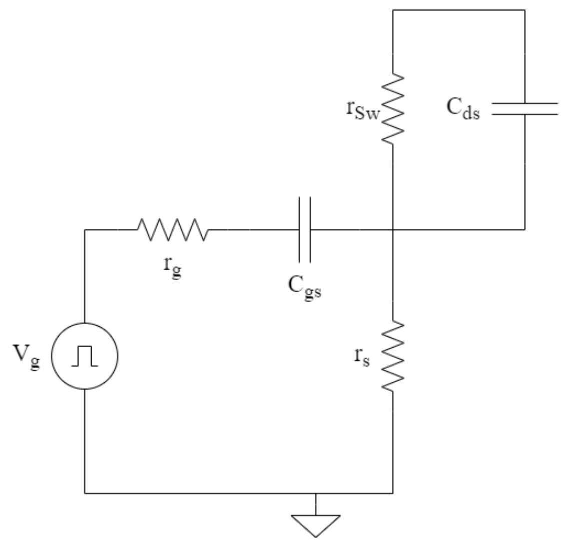

Figure 4-1 – Switch Model for the Practical Class-EF$_2$ Inverter

Inserting the switch model into the parasitic Class-EF$_2$ inverter results in the complete and practical Class-EF$_2$ inverter model shown in Fig. 4-2.

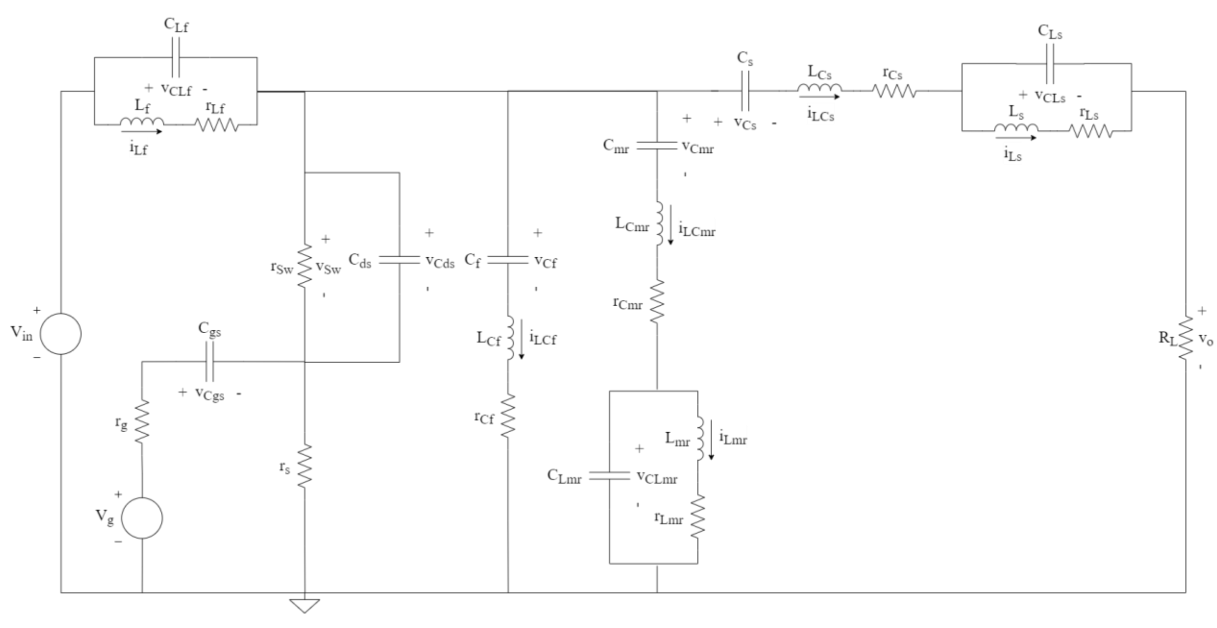

Figure 4-2 – Circuit Model of the Practical Class-EF$_2$ Inverter

Using the same procedure as outlined in Section 2.1, the state vector **X** is made up of the capacitor voltages and the inductor currents. However, **U** now contains the source voltage V_{in} as well as the gate driver voltage v_g as shown in (4.2).

$$\mathbf{X} = \begin{bmatrix} v_{C_f} & v_{C_{mr}} & v_{C_s} & v_{C_{L_f}} & v_{C_{L_{mr}}} & v_{C_{L_s}} & v_{C_{ds}} & v_{C_{ds}} & i_{L_f} & i_{L_{mr}} & i_{L_s} & i_{L_{C_f}} & i_{L_{C_{mr}}} & i_{L_{C_s}} \end{bmatrix}^T \tag{4.1}$$

$$\mathbf{U} = [V_{in} \quad v_g]^T \tag{4.2}$$

The differential equations for each state variable are derived in terms of the other state variables. The resulting differential equations which completely describe the circuit are presented in (4.3) to (4.16).

$$\frac{dv_{C_f}}{dt} = \frac{i_{C_f}}{C_f} = \frac{i_{L_{C_f}}}{C_f} \tag{4.3}$$

$$\frac{dv_{C_{MR}}}{dt} = \frac{i_{C_{mr}}}{C_{mr}} = \frac{i_{L_{C_{mr}}}}{C_{mr}} \tag{4.4}$$

$$\frac{dv_{C_s}}{dt} = \frac{i_{C_s}}{C_s} = \frac{i_{L_{C_s}}}{C_s} \tag{4.5}$$

$$\frac{dv_{C_{L_f}}}{dt} = \frac{i_{C_{L_f}}}{C_{L_f}} = \frac{\left(\frac{r_g + r_s}{r_g r_s}\right)\left(V_{in} - v_{C_{L_f}}\right) - \left(\frac{r_g r_{Sw} + r_s r_{Sw}}{r_g r_s r_{Sw}}\right) v_{C_{ds}} + \frac{v_{C_{gs}} - v_g}{r_g}}{C_{L_f}} \tag{4.6}$$

$$\frac{dv_{C_{L_{MR}}}}{dt} = \frac{i_{C_{L_{MR}}}(t)}{C_{L_{MR}}} = \frac{i_{L_{C_{mr}}} - i_{L_{mr}}}{C_{L_{mr}}} \tag{4.7}$$

$$\frac{dv_{C_{L_s}}}{dt} = \frac{i_{C_{L_s}}(t)}{C_{L_s}} = \frac{i_{L_{C_s}} - i_{L_s}}{C_{L_s}} \tag{4.8}$$

$$\frac{dv_{C_{ds}}}{dt} = \frac{i_{C_{ds}}(t)}{C_{ds}} = \frac{\left(\frac{r_g + r_s}{r_g r_s}\right)\left(V_{in} - v_{C_{L_f}}\right) - \left(\frac{r_g r_s + r_g r_{Sw} + r_s r_{Sw}}{r_g r_s r_{Sw}}\right) v_{C_{ds}} + \frac{v_{C_{gs}} - v_g}{r_g}}{C_{ds}} \tag{4.9}$$

$$\frac{dv_{C_{gs}}}{dt} = \frac{v_{C_{L_f}} + v_{C_{ds}} + v_g - v_{C_{gs}} - V_{in}}{r_g C_{ds}} \tag{4.10}$$

$$\frac{di_{L_f}}{dt} = \frac{v_{L_f}(t)}{L_f} = \frac{v_{C_{L_f}} - i_{L_f} r_{L_f}}{L_f} \tag{4.11}$$

$$\frac{di_{L_{MR}}}{dt} = \frac{v_{L_{MR}}(t)}{L_{MR}} = \frac{v_{C_{L_{mr}}} - i_{L_{mr}} r_{L_{mr}}}{L_{mr}} \tag{4.12}$$

$$\frac{di_{L_s}}{dt} = \frac{v_{L_s}(t)}{L_s} = \frac{v_{C_{L_s}} - i_{L_s} r_{L_s}}{L_s} \tag{4.13}$$

$$\frac{di_{L_{C_f}}}{dt} = \frac{v_{L_{C_f}}(t)}{L_{C_f}} = \frac{V_{in} - v_{C_f} - v_{C_{L_f}} - i_{L_{C_f}} r_{C_f}}{L_{C_f}} \tag{4.14}$$

$$\frac{di_{L_{C_{MR}}}}{dt} = \frac{v_{L_{C_{MR}}}(t)}{L_{C_{MR}}} = \frac{V_{in} - v_{C_{mr}} - v_{C_{L_f}} - v_{C_{L_{mr}}} - i_{L_{C_{mr}}} r_{C_{mr}}}{L_{C_{mr}}} \tag{4.15}$$

$$\frac{di_{L_{C_s}}}{dt} = \frac{v_{L_{C_s}}(t)}{L_{C_s}} = \frac{V_{in} - v_{C_s} - v_{C_{L_f}} - v_{C_{L_s}} - i_{L_{C_s}}\left(R_L + r_{C_s}\right)}{L_{C_s}} \tag{4.16}$$

Substituting (4.3) to (4.16) into (2.10) and (2.11) produces a 14$^{\text{th}}$ order multiple input multiple output (MIMO) system. The state vectors and matrices are defined in (4.17) to (4.20).

$$A = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_f} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_{MR}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{C_s} \\
0 & 0 & 0 & \frac{-(r_g + r_s)}{r_g r_s C_{L_f}} & 0 & 0 & \frac{-(r_s r_{Sw} + r_g r_{Sw})}{r_g r_s r_{Sw} C_{L_f}} & \frac{1}{r_g C_{L_f}} & \frac{-1}{C_{L_f}} & 0 & 0 & \frac{1}{C_{L_f}} & \frac{1}{C_{L_f}} & \frac{1}{C_{L_f}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{C_{L_{MR}}} & 0 & 0 & \frac{1}{C_{L_{MR}}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-1}{C_{L_s}} & 0 & 0 & \frac{1}{C_{L_s}} \\
0 & 0 & 0 & \frac{-(r_g + r_s)}{r_g r_s C_{ds}} & 0 & 0 & \frac{-(r_s r_g + r_s r_{Sw} + r_g r_{Sw})}{r_g r_s r_{Sw} C_{ds}} & \frac{1}{C_{ds} r_g} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{r_g C_{gs}} & 0 & 0 & \frac{1}{r_g C_{gs}} & \frac{-1}{r_g C_{gs}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{L_f} & 0 & 0 & 0 & 0 & \frac{-r_{L_f}}{L_f} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{L_{MR}} & 0 & 0 & 0 & 0 & \frac{-r_{L_{MR}}}{L_{MR}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{L_s} & 0 & 0 & 0 & 0 & \frac{-r_{L_s}}{L_s} & 0 & 0 & 0 \\
\frac{-1}{L_{C_f}} & 0 & 0 & \frac{-1}{L_{C_f}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-r_{C_f}}{L_{C_f}} & 0 & 0 \\
0 & \frac{-1}{L_{C_{MR}}} & 0 & \frac{-1}{L_{C_{MR}}} & \frac{-1}{L_{C_{MR}}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-r_{C_{MR}}}{L_{C_{MR}}} & 0 \\
0 & 0 & \frac{-1}{L_{C_s}} & \frac{-1}{L_{C_s}} & 0 & \frac{-1}{L_{C_s}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-(r_{C_s} + R)}{L_{C_s}}
\end{bmatrix} \tag{4.17}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & \frac{r_g + r_s}{r_g r_s C_{L_f}} & 0 & 0 & \frac{r_g + r_s}{r_g r_s C_{ds}} & \frac{-1}{r_g C_{gs}} & 0 & 0 & 0 & \frac{1}{L_{C_f}} & \frac{1}{L_{C_{MR}}} & \frac{1}{L_{C_s}} \\ 0 & 0 & 0 & \frac{-1}{r_g C_{Lf}} & 0 & 0 & \frac{-1}{r_g C_{ds}} & \frac{1}{r_g C_{gs}} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{4.18}$$

$$C = I_{14} \tag{4.19}$$

$$D = \vec{0} \tag{4.20}$$

Equations (4.17) and (4.18) contain the circuit components of the practical Class-EF_2 Inverter, I_{14} in (4.19) represents the 14^{th} order identity matrix, and $\vec{0}$ in (4.20) represents the zero vector. The complexity of the system has been significantly increased by the addition of the switch model and the second source.

4.2 Solving the Model

4.2.1 Determination of the Switching Dynamics

To simplify the solution of the state-space equations, the LMG1020 gate driver [34] is considered, and the following assumptions are made:

- Gate voltage v_g changes linearly, and has a rise and fall time of 400 ps,
- Resistance r_{sw} changes logarithmically, and has a rise and fall time of 400 ps,
- The switch has a constant temperature of 85˚C, and
- All internal resistance and capacitance values remain constant throughout the optimization.

The 400 ps rise and fall time is based on the characteristics of the LMG1020 gate driver.

4.2.1.1 Internal Resistances of the Switch

Using the EPC2019 switch as an example, typical values for r_g and $r_{Sw(on)}$ can be found in the manufacturer provided datasheet. A typical value for r_s is not provided, however an equation describing the behavior of this resistor can be found within the LTSpice library file and is shown in eq. (4.21) [35].

$$r_s = 0.02528 \times (1 + 0.0065(\tau - 25°C)) \tag{4.21}$$

Substituting the assumed switch temperature of 85°C into eq. (4.21), all internal switch resistances can be determined. Their values are provided in Table 4-1.

Table 4-1 – Internal Resistances of the EPC2019 Power MOSFET

Parameter	Resistance [mΩ]
$r_{Sw(On)}$	36
r_g	400
r_s	6.2

4.2.1.2 Internal Capacitances of the Switch

Continuing to use the EPC2019 as an example, equations describing the behavior of C_{gs} and C_{ds} can be found within the LTSpice library file and are shown in eq. (4.22) to (4.25) where q is the charge of the capacitor [35].

$$C_{ds}(t) = \frac{q_{ds}(t)}{v_{ds}(t)} + 69.15\,pF \tag{4.22}$$

$$q_{ds}(t) = \begin{cases} 824.54 \times 10^{-12} \times ln\left(1 + e^{\left(\frac{20.76 - v_{ds}(t)}{4.72}\right)}\right) + 14.85 \times 10^{-9} \times ln\left(1 + e^{\left(\frac{0.202 - v_{ds}(t)}{67.83}\right)}\right), v_{ds} > 6 \\ 0\,, else \end{cases} \tag{4.23}$$

$$C_{gs}(t) = \frac{q_{gs}(t)}{\overline{v_{gs}}} + 200.42\,pF \tag{4.24}$$

$$q_{gs}(t) = \begin{cases} 8.39 \times 10^{-12} \times ln\left(1 + e^{\left(\frac{\overline{v_{gs}} - 1.845}{0.174}\right)}\right) - 11.52 \times 10^{-15} \times ln\left(1 + e^{\left(\frac{v_{ds}(t) + 5.551}{0.281}\right)}\right), v_{ds} > 6 \\ 0\,, else \end{cases} \tag{4.25}$$

In (4.22) and (4.24) the voltages v_{ds} and v_{gs} are obtained from the MATLAB solution of the ideal case. In (4.24) and (4.25), v_{gs} is averaged to avoid division by 0 and to better approximate the provided capacitance plots in the datasheet. Substituting these voltage vectors into (4.22) to (4.25) provides the designer with a vector of capacitance values which is averaged to determine the values for use with the MOGO and MOPSO design approaches.

Using the ideal MOGO Design Case II solution as an example, plots of C_{ds} and C_{gs} are obtained and shown in Fig. 4-3.

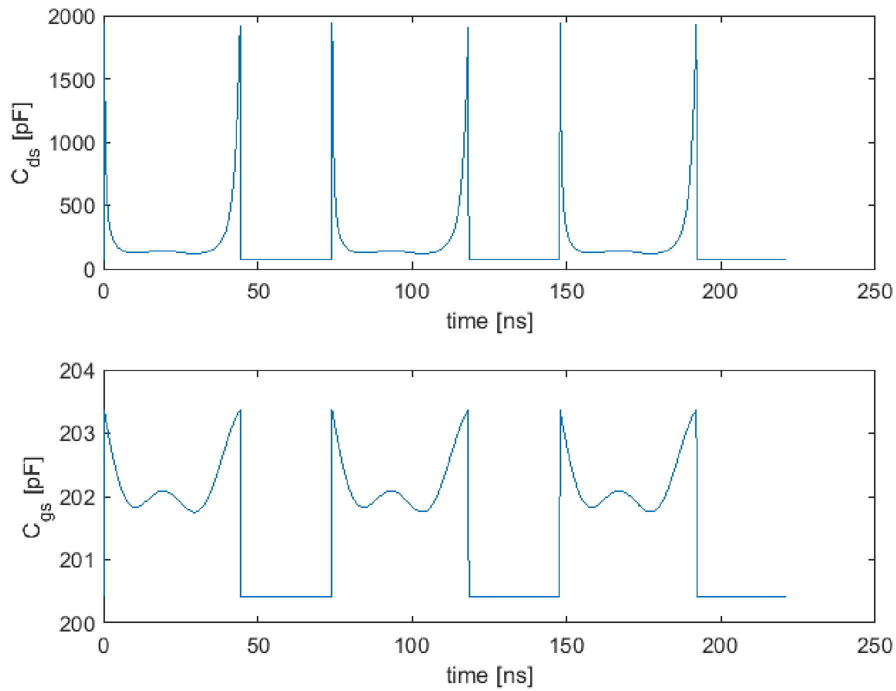

Figure 4-3 – Plots of C_{ds} and C_{gs} Using Ideal MOGO Design Case II

These waveforms are averaged to obtain the final capacitor values shown in Table 5-2.

Table 4-2 – Averaged Internal Capacitance Values of the EPC2019 Using Ideal MOGO Design Case II

Parameter	Capacitance [pF]
C_{ds}	178.20
C_{gs}	201.49

4.2.2 Simulation of the Practical Class-EF$_2$ Inverter

To validate the state-space model of the practical Class-EF$_2$ inverter, the component values provided in Table 3-1, Table 4-1, and Table 4-2 are used to solve the state-space model. The circuit is then fully defined in LTSpice using the manufacturer provided model for the switch. Plots of the switch voltage, switch current, and output voltage waveforms obtained from both the MATLAB solution of the state-space model and LTSpice simulation are shown in Figs. 4-4 (a-c).

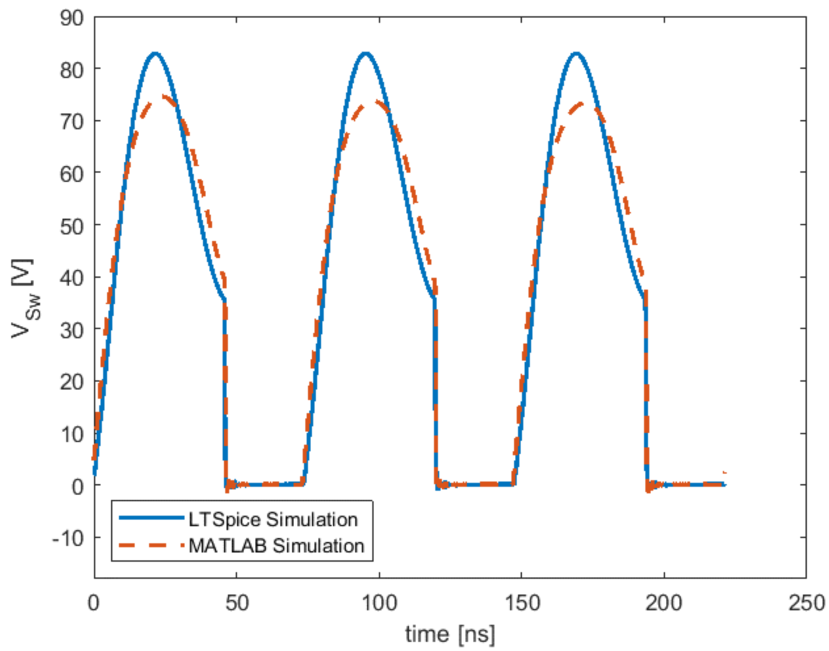

Figure 4-4 (a) – Switch Voltage Waveform of the Practical Class-EF$_2$ Inverter State-Space Model vs. LTSpice Simulation

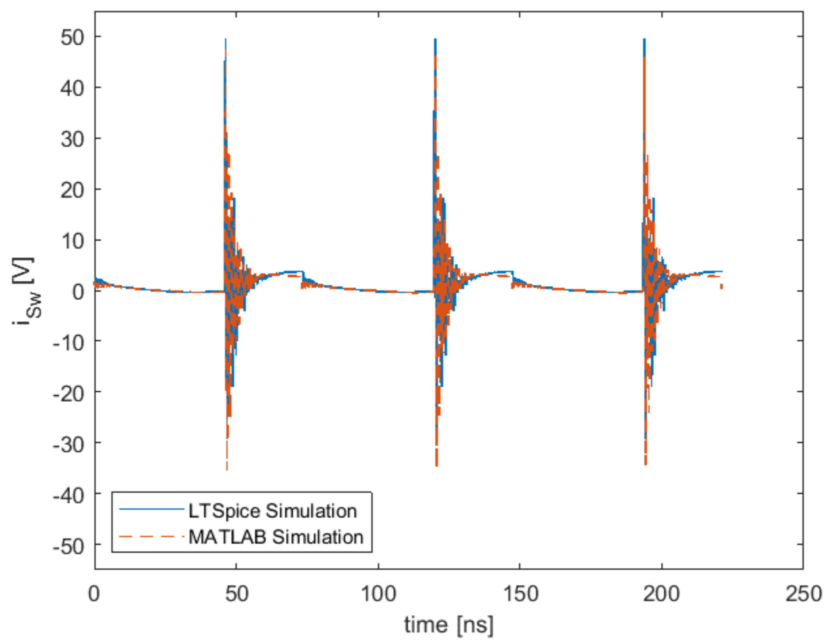

Figure 4-4 (b) – Switch Current Waveform of the Practical Class-EF$_2$ Inverter State-Space Model vs. LTSpice Simulation

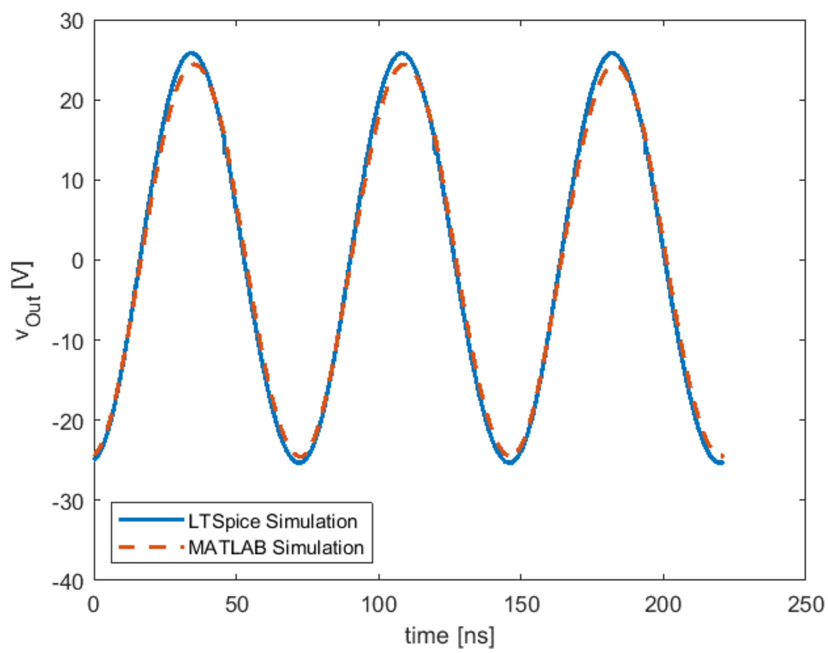

Figure 4-4 (c) – Output Voltage Waveform of the Practical Class-EF$_2$ Inverter State-Space Model vs. LTSpice Simulation

Figures 4-4 (a-c) demonstrate the accuracy of the practical Class-EF_2 inverter state-space model. There are some discrepancies, but they are likely due to the capacitors being averaged rather than recalculated during each sample, and the exclusion of C_{gd} and r_d from the state-space model.

There is also significant hard switching occuring in both the MATLAB and LTSpice simulation demonstrating how large of an impact is caused by the inclusion of a non-ideal switch.

4.2.3 Internal Resistances and Capacitances for all Design Cases

For Design Case I and Design Case II, the EPC2019 switch is selected due to its voltage, current, and output capacitance characteristics. The EPC8010 [36] switch is selected for Design Case III since the output capacitance of the EPC2019 is too high.

Following the process presented in 4.2.1.1 and 4.2.1.2, the internal resistances and capacitances to be used with the MOGO and MOPSO design approaches of the Complete Class-EF_2 inverter are calculated. Their values are presented in Table 4-3.

Table 4-3 – Internal Resistances and Capacitances for use with the MOGO and MOPSO Design Approaches for the Complete Class-EF_2 Inverter

Parameter	MOGO			MOPSO		
	Design Case I	Design Case II	Design Case III	Design Case I	Design Case II	Design Case III
$r_{Sw(on)}$ $[m\Omega]$	36	36	120	36	36	120
r_g $[m\Omega]$	400	400	300	400	400	300
r_s $[m\Omega]$	6.2	6.2	16.9	6.2	6.2	16.9
C_{ds} $[pF]$	216.51	178.20	56.84	244.02	176.06	50.81
C_{gs} $[pF]$	202.99	201.49	41.53	201.47	202.32	41.25

4.3 Design Results Using the MOGO Algorithm

The MOGO design method was tested using three different design cases presented in Table 2-4. The results of the MOGO designs are obtained and compared with the parasitic model. For design factors involving C_f the value is adjusted to include the value of C_{ds} as described in eq. (4.26).

$$C_{f(adjusted)} = C_f + C_{ds} \tag{4.26}$$

Table 4-4 shows the comparative values for each design case. Code for the Practical Class-EF_2 fitness function can be found in Appendix C.

Table 4-4 – Comparison of the Results of the Parasitic and Practical MOGO Designs

Factors	Design Case I		Design Case II		Design Case III	
	Parasitic	Complete	Parasitic	Complete	Parasitic	Complete
D	0.394	0.365	0.357	0.353	0.393	0.403
k	1.133	1.479	1.712	1.600	1.128	1.033
q_2	2.744	2.589	2.517	2.550	2.747	2.806
$\frac{1}{\omega R_L C_f}$	6.587	5.513	5.390	5.090	6.503	6.507
$\frac{1}{\omega R_L C_{MR}}$	7.463	8.156	9.230	8.145	7.336	6.721
c_p	0.117	0.118	0.113	0.118	0.111	0.119
V_{Stress}	2.280	2.190	2.210	2.192	2.314	2.309
I_{Stress}	3.385	3.572	3.727	3.523	3.496	3.336

The factors for Design Case I and Design Case II are both very similar to the max f optimal design factors. The value of k in design case III was reduced however making it a close match with the max c_p optimal design factors.

4.3.1 Determination of the Inverter Component Values from the MOGO Designs

The nominal values of all components were calculated as shown in 2.2.5 and the parasitic elements were calculated as shown in 3.2.1. The parasitic capacitance C_{L_f} is assumed to be 3.35 pF, the parasitic capacitances $C_{L_{mr}}$ and C_{L_s} are assumed to be 2.1 pF, the ESR of all capacitors is assumed to be 50 mΩ, and the ESR of inductor L_f is assumed to be 220 mΩ as stated in 3.1.2.

Values for all components are shown in Table 4-5 and Table 4-6.

Table 4-5 – Comparison of the Component Values of the Parasitic and Practical MOGO Designs

Parameter	Design Case I		Design Case II		Design Case III	
	Parasitic	Practical	Parasitic	Practical	Parasitic	Practical
L_f $[\mu H]$	38.05	49.85	74.75	51.42	42.57	58.19
L_{mr} $[nH]$	218.99	239.32	270.83	238.99	75.34	69.03
L_s $[nH]$	890.74	636.51	545.32	472.35	203.08	293.88
C_f $[pF]$	712.79	635.11	217.75	52.37	128.91	72.01
C_{mr} $[pF]$	629.08	575.63	127.16	144.10	114.28	124.74
C_s $[pF]$	766.33	1204.4	347.8	442.84	225.20	140.07
V_{in} $[V]$	21.88	23.53	37.41	41.69	25.66	24.59
D	0.394	0.365	0.357	0.353	0.393	0.403

Table 4-6 – Values of all Parasitic Elements for each MOGO Design Case

Parameter	Value		
	Design Case I	Design Case II	Design Case III
L_{C_f} $[pH]$	107.95	152.57	145.98
$L_{C_{mr}}$ $[pH]$	109.43	132.59	135.27
L_{C_s} $[pH]$	98.78	113.48	133.11
$r_{L_{mr}}$ $[m\Omega]$	67.97	135.75	78.41
r_{L_s} $[m\Omega]$	180.77	268.29	333.85

The addition of C_{ds} had a significant impact on the value of C_f. This is expected since the output capacitance of the switch is in parallel with C_f so to maintain ZVS its value must be reduced. All other parameters experienced small variations due to changes in k and Q_{out}, but this is likely due to the arbitrary stopping condition placed on the algorithm.

4.4 Results of the MOPSO Designs

The MOPSO design method was tested using three different design cases presented in Table 2-4. The results of the MOPSO designs are obtained and compared with the parasitic model. For design factors involving C_f the value is adjusted to include the value of C_{ds} as described in eq. (4.26). Table 4-7 shows the comparative values for each design case.

Table 4-7 – Comparison of the Results of the Parasitic and Practical MOPSO Designs

Factors	Design Case I		Design Case II		Design Case III	
	Parasitic	Complete	Parasitic	Complete	Parasitic	Complete
D	0.399	0.351	0.367	0.344	0.358	0.391
k	1.154	1.564	1.438	1.880	1.562	1.444
q_2	2.733	2.561	2.604	2.475	2.561	2.602
$\frac{1}{\omega R_L C_f}$	6.563	6.129	5.673	5.029	5.491	5.959
$\frac{1}{\omega R_L C_{MR}}$	7.571	9.587	8.160	9.457	8.577	8.605
c_p	0.117	0.110	0.114	0.111	0.104	0.114
V_{Stress}	2.333	2.151	2.172	2.316	2.172	2.235
I_{Stress}	3.346	3.860	3.701	3.949	4.009	3.576

The factors for all design cases lean towards the max *f* optimal design procedure. Like in the case of the MOGO designs, the algorithm favored a higher value of k when the practical switch was added to the circuit.

4.4.1 Determination of the Inverter Component Values from the MOGO Designs

The nominal values of all components were calculated as shown in 2.2.5 and the parasitic elements were calculated as shown in 3.2.1. The parasitic capacitance C_{L_f} is assumed to be 3.35 pF, the parasitic capacitances $C_{L_{mr}}$ and C_{L_s} are assumed to be 2.1 pF, the ESR of all capacitors is assumed to be 50 mΩ, and the ESR of inductor L_f is assumed to be 220 mΩ as stated in 3.1.2. Values for all components are shown in Table 4-8 and Table 4-9.

Table 4-8 – Comparison of the Component Values of the Parasitic and Practical MOPSO Designs

Parameter	Design Case I		Design Case II		Design Case III	
	Parasitic	Practical	Parasitic	Practical	Parasitic	Practical
$L_f\ [\mu H]$	41.45	45.03	73.86	89.88	77.34	100.00
$L_{mr}\ [nH]$	222.17	281.31	239.45	277.49	88.08	88.37
$L_s\ [nH]$	636.22	812.28	749.49	708.51	146.19	228.80
$C_f\ [pF]$	715.31	522.00	206.89	57.31	152.68	89.88
$C_{mr}\ [pF]$	620.07	489.71	143.83	124.11	97.75	97.43
$C_s\ [pF]$	112.98	947.17	238.71	252.90	412.16	179.81
$V_{in}\ [V]$	19.34	28.02	42.55	40.73	28.22	22.61
D	0.399	0.351	0.367	0.344	0.358	0.391

Table 4-9 – Values of all Parasitic Elements for each MOPSO Design Case

Parameter	Value		
	Design Case I	Design Case II	Design Case III
$L_{C_f}\ [pH]$	110.92	150.67	141.56
$L_{C_{mr}}\ [pH]$	111.91	135.36	139.98
$L_{C_s}\ [pH]$	102.13	122.66	128.58
$r_{L_{mr}}\ [m\Omega]$	79.89	157.62	100.39
$r_{L_s}\ [m\Omega]$	230.69	402.43	259.92

Like in the case of the MOGO designs the addition of C_{ds} had a significant impact on the value of C_f. Once again, this is expected since the output capacitance of the switch is in parallel with C_f so to maintain ZVS its value must be reduced. Like in the case of the MOGO designs, the changes in component values between the parasitic and practical models are much less drastic and are mainly due to variations in the values of k and Q_{out}.

4.5 Comparison of MOGO and MOPSO Design Approaches

Each design was simulated in LTSpice using the component values listed in Table 4-5, Table 4-6, Table 4-8, and Table 4-9. The results from the simulations are recorded in Table 4-10 using the process presented in Section 2.6. Relevant waveforms from LTSpice simulations are presented in Figures 4-5 (a-c), 4-6 (a-c), and 4-7 (a-c).

Table 4-10 - Comparison of the Results of the MOGO and MOPSO Design Approaches

Parameter	Design Case I		Design Case II		Design Case III	
	MOGO	MOPSO	MOGO	MOPSO	MOGO	MOPSO
$P_{Out_{Desired}}$	23	23	40	40	25	25
$P_{Out_{Actual}}$	23.56	23.34	42.19	37.81	23.38	24.79
η	93.10%	93.40%	96.06%	95.34%	92.93%	90.97%
V_{Stress}	2.190	2.151	2.192	2.163	2.309	2.235
I_{Stress}	3.572	3.860	3.523	3.949	3.366	3.576
Q_{Out}	4.598	5.857	3.266	5.293	6.544	4.846
THD	3.60%	3.34%	4.06%	2.63%	2.62%	2.99%

As can be seen, the practical design cases for both the MOGO and MOPSO algorithms performed well in all measured values of interest. The minimum efficiency was 90.97% and occurred during MOPSO design case III, and the THD of the output voltage waveform in most design cases was kept below 4%. The method of estimating the internal resistances and capacitances of the switch based on the ideal case was successful.

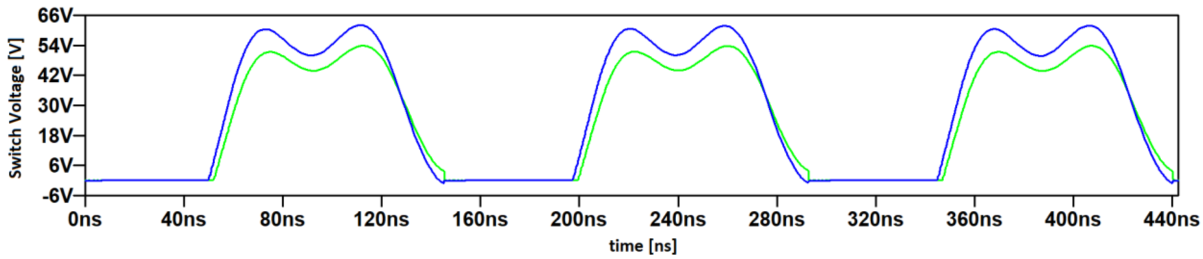

Figure 4-5 (a) – Switch Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

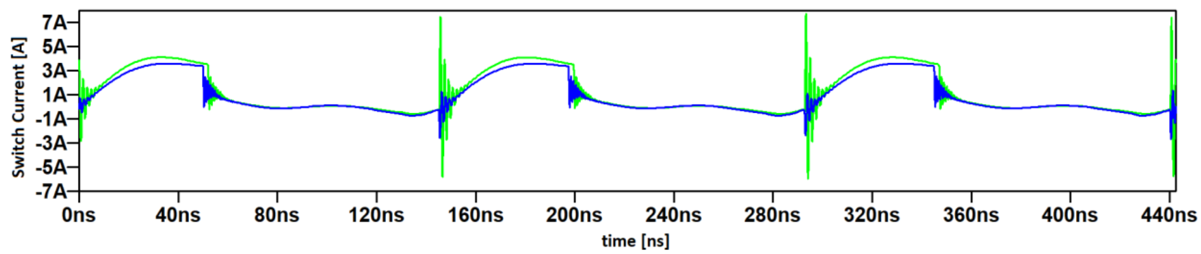

Figure 4-5 (b) – Switch Current Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

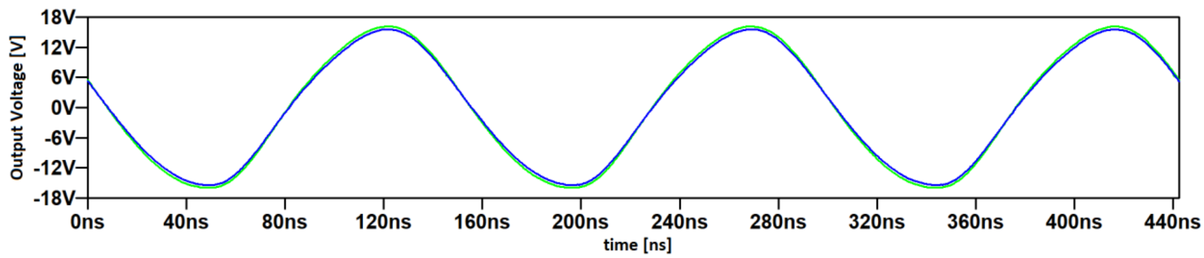

Figure 4-5 (c) – Output Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case I

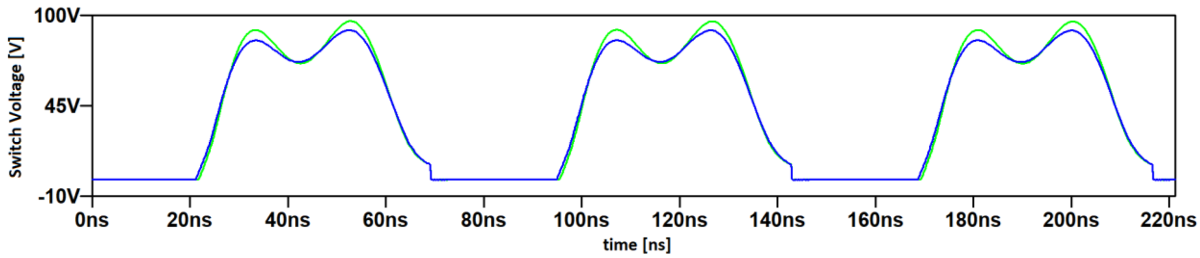

Figure 4-6 (a) – Switch Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

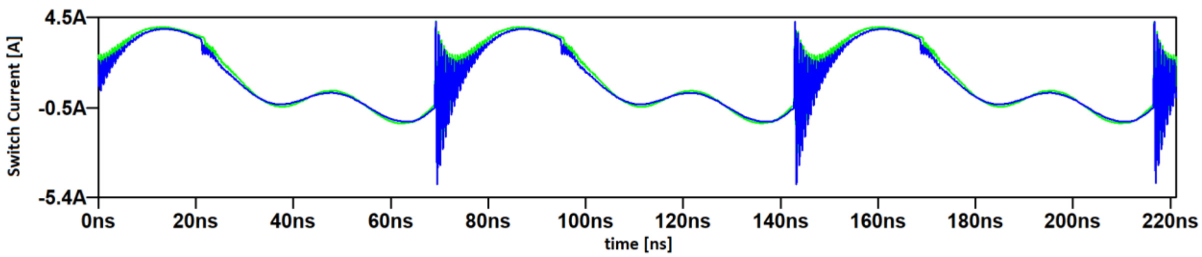

Figure 4-6 (b) – Switch Current Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

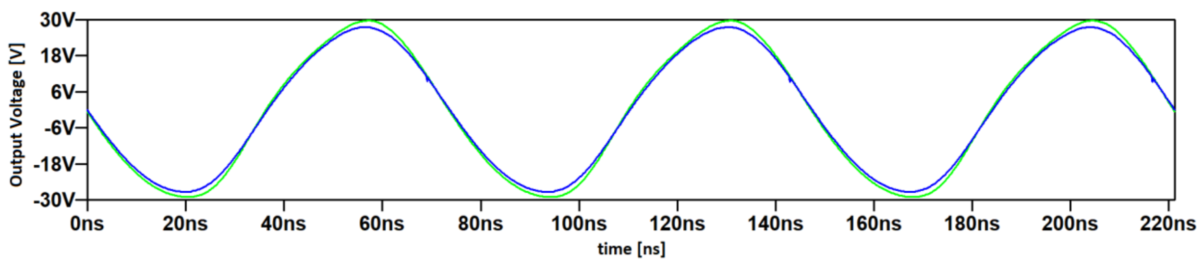

Figure 4-6 (c) – Output Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case II

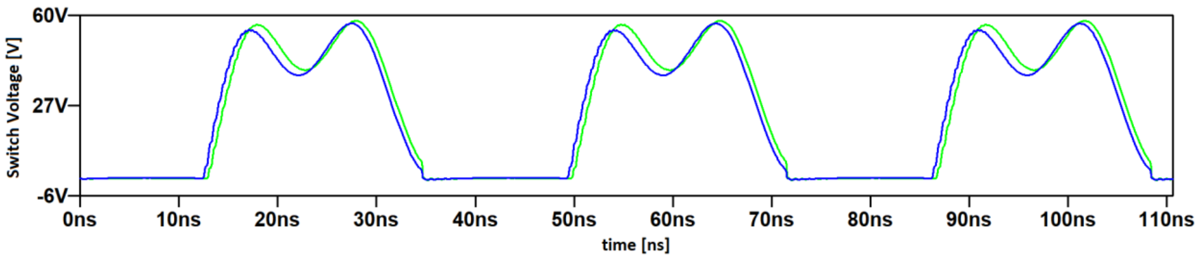

Figure 4-7 (a) – Switch Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

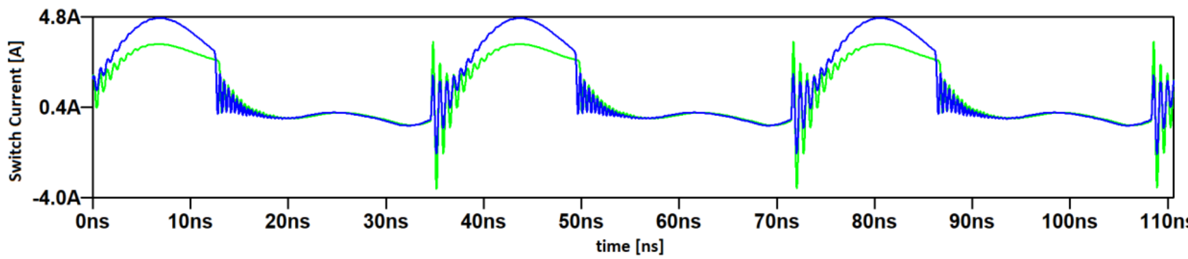

Figure 4-7 (b) – Switch Current Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

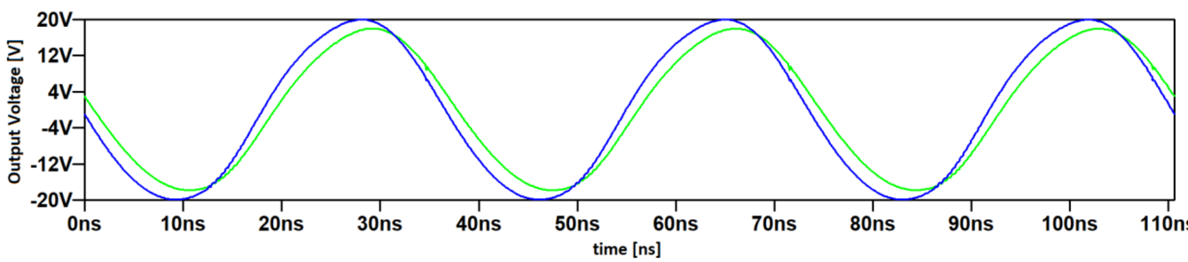

Figure 4-7 (c) – Output Voltage Waveform of the Practical Class-EF$_2$ Inverter: MOGO (Green) & MOPSO (Blue) Design Case III

The above figures demonstrate the success of the optimization-based design approach for the practical Class-EF$_2$. In each design case, the circuits maintained ZVS and ZDVS to an acceptable level of error, had relatively sinusoidal output voltage waveforms, and avoided MOSFET diode conduction. Like in the parasitic case, the switch current waveform exhibits ringing due to hard switching, but it is relatively small outside of MOGO Design Case I.

4.6 Summary

In Chapter 4, the design of the Practical Class-EF_2 inverter was investigated. A 14^{th} order state-space model was proposed which considered the parasitic elements of all components in the circuit as well as the internal resistances and capacitances of the switch. This model was validated through LTSpice simulation by comparison with manufacturer provided models.

The new state-space model was then used with both the MOGO and MOPSO algorithms to design three circuits with differing frequency, power, and load specifications. The proposed optimization-based design approaches provided circuits which performed well in all measured values of interest, successfully adhered to the design specifications, and maintained ZVS and ZDVS to an acceptable level of error when tested in LTSpice.

The addition of the practical switch model caused small variations in all component values, but a large change in the value of C_f. This is necessary to maintain ZVS and ZDVS operation since C_f is in parallel with the output capacitance of the switch.

CHAPTER 5

Multi-Objective Genetic Optimization Design Example

In this chapter, a design example using the MOGO algorithm is presented for a 6.78 MHz, 100W, 20Ω Class-EF_2 Inverter. First, the ideal state-space model from Chapter 2 is used to generate the necessary information to select a switch and estimate the value of C_{ds}. This information is then used with the practical state-space model from Chapter 4 to complete the final design.

The final design was simulated in LTSpice and performed well in all measured values of interest while maintaining ZVS operation to an acceptable level of error.

5.1 Design Example Stage 1 – Setting up the Problem

In the previous chapters, each state-space model was tested with the three design cases presented in Table 2-4 with wide boundary conditions to prove the validity of the MOGO and MOPSO design approaches. In this chapter, a design example using the MOGO algorithm is presented for a 6.78 MHz, 100W, 20Ω Class-EF_2 Inverter. The goal of the design is to use the MOGO algorithm to determine the component values and a suitable switch for the inverter to meet the specified requirements.

For this test, the boundaries for D, k, and $x_{L_S C_S}$ are changed to remove solutions that were found to be inferior in the tests presented in Chapter 2. The upper and lower boundary for V_{in} is also increased since the output power is 2.5 times larger than any of the previous designs. Consequently, the variable boundary conditions used in this design example are shown in Table 5-1.

Table 5-1 – Updated Optimization Boundary Conditions

Variables	D	L_f $[\mu H]$	C_f $[pF]$	k	Q_{Out}	$x_{L_S C_S}$	V_{in} $[V]$
Upper Boundary	0.4	100	5000	2	8	1.2	120
Lower Boundary	0.35	0.01	0.5	0.5	2	0.8	24

5.2 Design Example Stage 2 – Ideal Optimization

Using the boundary conditions presented in Table 5-1, the MOGO algorithm is implemented for the ideal Class-EF_2 inverter state-space model. This will provide the designer with nominal component values which can then be used to generate the switch voltage and current waveforms in MATLAB and LTSpice. These waveforms are necessary for sizing the switch and approximating the value of C_{ds} and C_{gs} as described in Chapter 4.

The optimized values of the variables from the MOGO algorithm are presented in Table 5-2.

Table 5-2 – Optimized Values of Variables for the Ideal Design Example

Variable	D	L_f $[\mu H]$	C_f $[pF]$	k	Q_{Out}	$x_{L_S C_S}$	V_{in} $[V]$
Value	0.3722	71.49	209.94	1.454	7.664	0.926	79.92

Using the process shown in section 2.4.1, the component values for this design are calculated and presented in Table 5-3.

Table 5-3 – Component Values for the Ideal Design Example

Component	Value
L_f $[\mu H]$	71.50
L_{mr} $[nH]$	954.14
L_s $[nH]$	3886.21
C_f $[pF]$	209.94
C_{mr} $[pF]$	144.38
C_s $[pF]$	165.39
V_{in} $[V]$	79.92
D	0.3722

The parameters are substituted into the ideal Class-EF_2 inverter state-space model in MATLAB as well as LTSpice to examine the performance of the circuit and select a switch.

5.3 Design Example Stage 3 – Determination of Suitable Switch

With the ideal design completed, the next step is to select a switch by examining the switch voltage and current waveforms which are presented in Fig. 5-1.

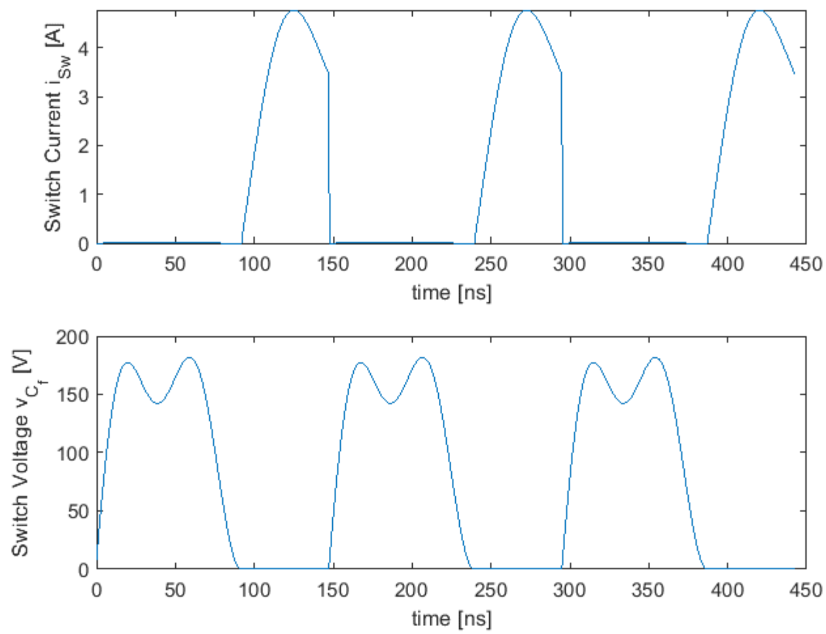

Figure 5-1 – Switch Voltage and Switch Current Waveforms for the Ideal Design Example

The switch voltage waveform peaks at 181.71V and has an average value of 79.92V and the switch current peaks at 4.77A with an average value of 1.33A. For this design example, the EPC2019 switch is selected as it is rated for 200V and 8.5A and has a fairly linear output capacitance for values of v_{ds} above 30 V.

Following the process described in 4.2.1, the MATLAB generated switch voltage vector is substituted into (4.21) to (4.25) giving the following values for the internal resistances and capacitances.

Table 5-4 – Internal Capacitances and Resistances for the Practical Design Example

Internal Capacitances		Internal Resistances	
$C_{ds}\ [pF]$	118.11	$r_g\ [m\Omega]$	400
$C_{gs}\ [pF]$	201.86	$r_s\ [m\Omega]$	6.2

Now that the switch parameters have been determined, they can be substituted into the practical Class-EF$_2$ inverter state-space model in MATLAB and the optimizer can begin its design. The parameters which had the lowest overall objective function scores are presented in Table 5-5.

Table 5-5 - Highest Scoring Variables for the Practical Design Example

Variable	$\boldsymbol{D}$	$\boldsymbol{L_f\ [\mu H]}$	$\boldsymbol{C_f\ [pF]}$	$\boldsymbol{k}$	$\boldsymbol{Q_{Out}}$	$\boldsymbol{x_{L_sC_s}}$	$\boldsymbol{V_{in}\ [V]}$
Value	0.3776	53.87	106.77	1.761	3.335	0.841	79.79

These values can now be used to calculate the nominal components and their parasitic elements as described in sections 2.4.1 and 3.2.1. All component values are shown in Table 5-6.

Table 5-6 - Component Values for the Practical Design Example

Parameter	Nominal Value	ESR [mΩ]	Parasitic Element
L_f	53.87 µH	220	3.35 pF
L_{mr}	1079.1 nH	306.5	2.1 pF
L_s	1860.8 nH	528.5	2.1 pF
C_f	106.77 pF	50	138.22 pH
C_{mr}	127.66 pF	50	134.84 pH
C_s	418.20 pF	50	114.38 pH
D	0.3776	-	-
V_{in}	79.79 V	-	-

The parameters in Table 5-6 were then used to simulate the circuit in LTSpice and evaluate the performance of the circuit using the process presented in Section 2.6. The specific values of interest are presented in Table 5-7 and waveforms for the switch voltage, switch current, and output voltage shown in Fig. 5-2 (a-c).

Table 5-7 – LTSpice Simulation Results of the Practical Class-EF$_2$ Inverter Design Example

Parameter	Value
$P_{Out_{Desired}}$	100
$P_{Out_{Actual}}$	97.862
η	95.64%
V_{Stress}	2.287
I_{Stress}	3.593
Q_{Out}	3.335
THD	4.40%

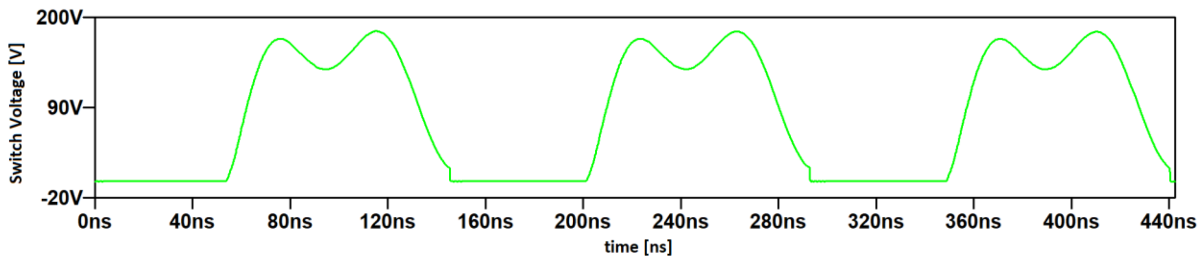

Figure 5-2 (a) – Switch Voltage Waveform of the Practical Class-EF$_2$ Design

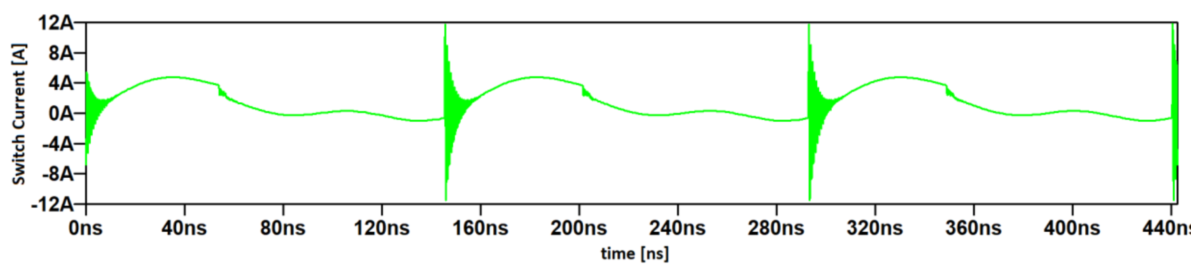

Figure 5-2 (b) – Switch Current Waveform of the Practical Class-EF$_2$ Design

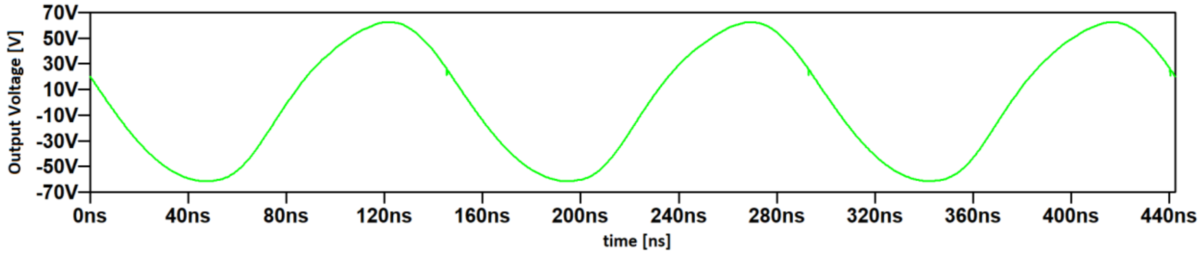

Figure 5-2 (c) – Output Voltage Waveform of the Practical Class-EF$_2$ Design

As can be seen, the practical Class-EF$_2$ inverter achieved an efficiency of 95.64%, a 4.40% output voltage waveform THD, a 2.14% error between the desired output power and the achieved output power and maintained ZVS to an acceptable level of error.

5.4 Summary

In this chapter, a design example was presented for a 6.78 MHz, 100W, 20Ω Class-EF$_2$ Inverter using the MOGO algorithm. The final design had an efficiency of more than 95% and demonstrated the ability to use the proposed optimization-based design approach for high power designs.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis explored the use of multi-objective optimization algorithms for the design of high frequency inverters. State-space models of the ideal, the parasitic, and the practical Class-EF_2 inverter were derived and validated by MATLAB and LTSpice simulation. Each model was then applied to the MOGO and MOPSO algorithms to design three inverters with varying output power, frequency, and load requirements. The validity of the optimization-based design approach was confirmed by comparison with analytical results, and the proposed circuits performed well in all measured values of interest and adhered to the design specifications.

This thesis investigated the design of the ideal, parasitic, and practical Class-EF_2 inverter using the MOGO and MOPSO algorithms. Chapter 2 studied the ideal Class-EF_2 inverter which was used to validate the optimization-based design approach by comparing the results with analytical design factors and equations. The proposed designs successfully maintained ZVS and ZDVS operation to an acceptable level of error and adhered to the stated design specifications.

Chapter 3 introduced the parasitic Class-EF_2 inverter and presented a state-space model which was validated by MATLAB and LTSpice simulation. It was then applied to the MOGO and MOPSO algorithms to design the same three circuits as the ideal case. The addition of the parasitic elements caused large variations in component values and design factors. The added inductors, capacitors and ESRs introduced more dynamics and sources of loss to the system which created a trade-off between the inverter efficiency and the output voltage THD. When not accounted for in the design phase, the parasitic elements may also cause a loss of ZVS and ZDVS operation and introduce unwanted ringing in the switch current waveform.

Chapter 4 introduced the practical Class-EF_2 inverter which included the parasitic elements from Chapter 3 and a practical switch model. A state-space model of the practical Class-EF_2 inverter was derived, and a method for estimating the internal resistances and capacitances was presented. The model was validated by MATLAB and LTSpice simulation using the manufacturer provided switch model. It was then applied to the MOGO and MOPSO algorithms to design the same three circuits as the ideal and parasitic cases. The addition of the practical switch model caused small variations in all component values, but a large change in the value of C_f. This is necessary to maintain ZVS and ZDVS operation since C_f is in parallel with the output capacitance of the switch. The optimization-based design approach successfully designed circuits which performed well in all measured values of interest, maintained ZVS and ZDVS operation to an acceptable level of error, and adhered to the design specifications.

Finally, Chapter 6 presented a design example for a practical 6.78 MHz, 100W, 20Ω Class-EF_2 inverter using the MOGO algorithm. This demonstrated the potential of the optimization-based design approach for higher power inverters and provided insight into the tuning of the optimization settings. The final design adhered to the design specifications and maintained an efficiency of more than 95%.

The work described within this thesis is meant to be a proof of concept for the optimization-based design of high-frequency inverters. However, more work is still required to address some of the issues which arose throughout the work. This includes:

- The exploration of different generation and population sizes, as well as the implementation of different variations of both the MOGO and MOPSO algorithms,

- Exploring the effects of non-constant internal switch resistances and capacitances,
- Confirming the parasitic element estimations using real world measurements, and
- Implementing physical prototypes of the designed circuits to confirm their validity.

6.1 Contribution of the Thesis

This thesis provides useful insight into the design of the Class-EF_2 inverter with parasitic elements and a non-ideal switch by drawing comparisons to the analytical design factors and equations of the ideal case. It also serves as a proof of concept for the use of multi-obective optimization in high-frequency circuit design and the provides the necessary tools for designers and researchers to apply the design approach to other converters.

6.1.1 List of Publications

One publication has been made which included portions of the work presented in chapter 2 and chapter 3.

[1] A. Peddle, B. Ryan, and J. E. Quaicoe, "Design of Class-EF Inverters using Multi-Objective Genetic Optimization," in *the Thirtieth Annual Newfoundland Electrical and Computer Engineering Conference (NECEC)* 2021.

6.2 Future Work

Future work for the optimization-based design approach includes the investigation of the load-independent case. This would be beneficial to designers as the load-independent Class-EF_2 inverter is more appropriate for use in wireless power transfer systems and considering the parasitic elements and the switching dynamics in their design could provide useful insight. State-space models of different inverter and rectifier topologies could also be applied to the optimization-based design approach which would allow the design of resonant DC-DC converters. Finally, the optimization initialization files could be updated such that only standard values of components are proposed.

References

[1] M. K. Kazimierczuk and D. Czarkowski, *Resonant Power Converters*, 2nd ed. Hoboken, New Jersey: John Wiley and Sons, LTD, 2011.

[2] F. H. Raab, "Idealized Operation of the Class E Tuned Power Amplifier," *IEEE Trans. Circuits Syst.*, vol. 24, no. 12, pp. 725–735, 1977, doi: 10.1109/TCS.1977.1084296.

[3] R. E. Zulinski and J. W. Steadman, "Class E Power Amplifiers and Frequency Multipliers with Finite DC-Feed Inductance," *IEEE Trans. Circuits Syst.*, vol. 34, no. 9, pp. 1074–1087, 1987, doi: 10.1109/TCS.1987.1086268.

[4] M. K. Kazimierczuk and K. Puczko, "Power-Output Capability of Class E Amplifier at Any Loaded Q and Switch Duty Cycle," *IEEE Trans. Circuits Syst.*, vol. 36, no. 8, pp. 1142–1143, 1989, doi: 10.1109/31.192430.

[5] H. Koizumi, M. Iwadare, and S. Mori, "Class E2 dc/dc Converter with Second Harmonic Resonant Class E Inverter and Class E Rectifier," *Electr. Eng.*, 1994.

[6] R. E. Zulinski and K. J. Grady, "Load-Independent Class E Power Inverters: Part I - Theoretical Development," *IEEE Trans. Circuits Syst.*, vol. 37, no. 8, pp. 1010–1018, 1990.

[7] L. Roslaniec, A. S. Jurkov, A. Al Bastami, and D. J. Perreault, "Design of single-switch inverters for variable resistance/load modulation operation," *IEEE Trans. Power Electron.*, vol. 30, no. 6, pp. 3200–3214, 2015, doi: 10.1109/TPEL.2014.2331494.

[8] S. Aldhaher, D. C. Yates, and P. D. Mitcheson, "Load-Independent Class E/EF Inverters and Rectifiers for MHz-Switching Applications," *IEEE Trans. Power Electron.*, vol. 33, no. 10, pp. 8270–8287, 2018, doi: 10.1109/TPEL.2018.2813760.

[9] N. Obinata, W. Luo, X. Wei, and H. Sekiya, "Analysis of Load-independent Class-E Inverter at Any Duty Ratio," *IECON Proc. (Industrial Electron. Conf.*, vol. 2019-Octob, pp. 1615–1620, 2019, doi: 10.1109/IECON.2019.8927599.

[10] S. D. Kee, I. Aoki, A. Hajimiri, and D. Rutledge, "The class-E/F family of ZVS switching amplifiers," *IEEE Trans. Microw. Theory Tech.*, vol. 51, no. 6, pp. 1677–1690, 2003, doi: 10.1109/TMTT.2003.812564.

[11] Z. Kaczmarczyk, "High-Efficiency Class E, EF2 , and E/F3 Inverters," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1584–1593, 2006.

[12] J. M. Rivas, Y. Han, O. Leitermann, A. D. Sagneri, and D. J. Perreault, "A high-frequency resonant inverter topology with low-voltage stress," *IEEE Trans. Power Electron.*, vol. 23, no. 4, pp. 1759–1771, 2008, doi: 10.1109/TPEL.2008.924616.

[13] A. Mediano and N. O. Sokal, "A class-E RF power amplifier with a flat-top transistor-voltage waveform," *IEEE Trans. Power Electron.*, vol. 28, no. 11, pp. 5215–5221, 2013, doi: 10.1109/TPEL.2013.2242097.

[14] S. Aldhaher, D. C. Yates, and P. D. Mitcheson, "Modeling and Analysis of Class EF and Class E/F Inverters with Series-Tuned Resonant Networks," *IEEE Trans. Power Electron.*, vol. 31, no. 5, pp. 3415–3430, 2016, doi: 10.1109/TPEL.2015.2460997.

[15] A. Peddle, Y. Salami, M. Rahm, and J. E. Quaicoe, “Design and Comparative Analysis of Variable Load Class-E and Class-EF Inverters at Multiple Megahertz Frequencies and Power Levels,” 2020.

[16] S. Aldhaher, G. Kkelis, D. C. Yates, and P. D. Mitcheson, “Class EF2 inverters for wireless power transfer applications,” *2015 IEEE Wirel. Power Transf. Conf. WPTC 2015*, pp. 2–5, 2015, doi: 10.1109/WPT.2015.7140141.

[17] X. Zhao, Z. Sun, and Y. Xu, “Multi-Objective Optimization Design of Permanent Magnet Synchronous Motor Based on Genetic Algorithm,” *Proc. - 2020 2nd Int. Conf. Mach. Learn. Big Data Bus. Intell. MLBDBI 2020*, pp. 405–409, 2020, doi: 10.1109/MLBDBI51377.2020.00086.

[18] R. Hans, S. C. Kaushik, and S. Manikandan, “Multi-objective optimization of two stage series connected thermo-electric generator using genetic algorithm,” *2015 1st Int. Conf. Futur. Trends Comput. Anal. Knowl. Manag. ABLAZE 2015*, no. Ablaze, pp. 165–169, 2015, doi: 10.1109/ABLAZE.2015.7154977.

[19] N. Hasan, T. Yilmaz, R. Zane, and Z. Pantic, “Multi-objective particle swarm optimization applied to the design of Wireless Power Transfer systems,” *2015 IEEE Wirel. Power Transf. Conf. WPTC 2015*, pp. 1–4, 2015, doi: 10.1109/WPT.2015.7139138.

[20] M. R. Torkomany, H. S. Hassan, A. Shoukry, A. M. Abdelrazek, and M. Elkholy, “An enhanced multi-objective particle swarm optimization in water distribution systems design,” *Water (Switzerland)*, vol. 13, no. 10, pp. 1–21, 2021, doi: 10.3390/w13101334.

[21] MathWorks, “MATLAB,” 2022. https://www.mathworks.com/products/matlab.html.

[22] Mathworks, *Global Optimization Toolbox: User’s Guide*, R2021b ed. 2021.

[23] V. M. Cagigal, “Multi-Objective Particle Swarm Optimization (MOPSO).” MATLAB Central File Exchange, 2021, [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/62074-multi-objective-particle-swarm-optimization-mopso.

[24] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, 1st ed. John Wiley and Sons, LTD, 2001.

[25] Analog Devices, “LTSpice,” 2022. https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html.

[26] A. Peddle, B. Ryan, and J. E. Quaicoe, “Design of Class-EF Inverters using Multi-Objective Genetic Optimization,” 2021.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[28] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, “Handling Multiple Objectives With Particle Swarm Optimization,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, 2004, doi: 10.1002/9780470612163.

[29] M. R. Sierra and C. A. Coello Coello, “Improving PSO-based Multi-Objective optimization using crowding, mutation and ε-dominance,” *Lect. Notes Comput. Sci.*, vol.

3410, pp. 505–519, 2005, doi: 10.1007/978-3-540-31880-4_35.

[30] W. Elektronik, "WE-PD Performance SMT Shielded Power Inductor: 7447783100." 2018.

[31] W. Elektronik, "WE-PD SMT Shielded Power Inductor: 74477020." 2018.

[32] V. Vitramon, "Surface Mount Multilayer Ceramic Chip Capacitors for High Frequency: VJ HIFREQ Series." 2020.

[33] A. Massarini and M. K. Kazimierczuk, "Self-capacitance of inductors," *IEEE Trans. Power Electron.*, vol. 12, no. 4, pp. 671–676, 1997, doi: 10.1109/63.602562.

[34] T. Instruments, "LMG1020 5-V, 7-A, 5-A Low-Side GaN and MOSFET Driver for 1-ns Pulse Width Applications." 2018.

[35] EPC, "EPC2019 - Enhancement Mode Power Transistor." 2021.

[36] EPC, "EPC8010 - Enhancement Mode Power Transistor." 2021.

Appendices

Appendix A

Fitness Function for the Ideal Class-EF_2 Inverter

```
function Outcome =
EF_Ideal_Model(f_s,r_sw_on,r_sw_off,R_L,P_Out,minMaxValues,
inputParam)
%% Sort out parameters from optimizer
Outcome=zeros(1,6);

Cycles=3;                % # Cycles to Fully Calculate
Resolution=300;          % # of Samples/Cycle
P=1/f_s;                 % Switching Period
w=2*pi*f_s;              % Angular Frequency

D=inputParam(1);         % Extract values from optimzer for
calculations
Lf=inputParam(2);
Cf=inputParam(3);
Cmr=Cf/inputParam(4);
Lmr=1/(((2*w)^2)*Cmr);
Ls=(inputParam(5)*R_L)/(w*inputParam(6));
Cs=1/(((inputParam(6)*w)^2)*Ls);
U=inputParam(7);

t=P/Resolution;          % State-Space evaluation time
t_on=D*P;                % Transistor On time
t_off=(1-D)*P;           % Transistor Off time

d_off=floor(Resolution*(1-D));          % # of samples in
Off-State
d_on=Resolution-d_off;                  % # of samples in
On-State

r_off=zeros(1,d_off)+r_sw_off;          % Resistance
Vectors
r_on=zeros(1,d_on)+r_sw_on;
r_sw=[r_off , r_on];

%% Initialize State-Space Model
A_EFon=[-1/(Cf*r_sw_on) , 0 , 0 , 1/Cf , -1/Cf , -1/Cf;
```

```
        0 , 0 , 0 , 0 , 1/Cmr , 0;
        0 , 0 , 0 , 0 , 0 , 1/Cs;
        -1/Lf , 0 , 0 , 0 , 0 , 0;
        1/Lmr , -1/Lmr , 0 , 0 , 0 , 0;
        1/Ls , 0 , -1/Ls , 0 , 0 , -R_L/Ls];

A_EFoff=[-1/(Cf*r_sw_off) , 0 , 0 , 1/Cf , -1/Cf , -1/Cf;
         0 , 0 , 0 , 0 , 1/Cmr , 0;
         0 , 0 , 0 , 0 , 0 , 1/Cs;
         -1/Lf , 0 , 0 , 0 , 0 , 0;
         1/Lmr , -1/Lmr , 0 , 0 , 0 , 0;
         1/Ls , 0 , -1/Ls , 0 , 0 , -R_L/Ls];

B_EF=[0;
      0;
      0;
      1/Lf;
      0;
      0];

C_EF=[1 , 0 , 0 , 0 , 0 , 0;          % V_Cf
      0 , 1 , 0 , 0 , 0 , 0;          % V_Cmr
      0 , 0 , 1 , 0 , 0 , 0;          % V_Cs
      0 , 0 , 0 , 1 , 0 , 0;          % I_Lf
      0 , 0 , 0 , 0 , 1 , 0;          % I_Lmr
      0 , 0 , 0 , 0 , 0 , 1];         % I_Ls

D_EF=zeros(6,1);

%% State vector calculations
SSCycles=550;         % Number of Cycles to reach steady-
state
                      % 300 for 6.78, 550 for 13.56, 1100 for
27.12
X_0=zeros(6,1);       % Zero initial condition

for l=1:1:SSCycles
    if mod(l,2)==0
       X_n=(expm(A_EFon*t_on)*X_0);
       X_f=A_EFon\(expm(A_EFon*t_on)-eye(6))*(B_EF*U);
       X=X_n+X_f;
    else
       X_n=(expm(A_EFoff*t_off)*X_0);
       X_f=A_EFoff\(expm(A_EFoff*t_off)-eye(6))*(B_EF*U);
```

```
        X=X_n+X_f;
    end
        X_0=X;
end

%% Fully Calculate State Vectors for "Cycles" cycles
V_Cf=zeros(1,Resolution*Cycles);   % Pre-allocate all
parameter vectors
V_Cmr=zeros(1,Resolution*Cycles);
V_Cs=zeros(1,Resolution*Cycles);
I_Lf=zeros(1,Resolution*Cycles);
I_Lmr=zeros(1,Resolution*Cycles);
I_Ls=zeros(1,Resolution*Cycles);
I_Cf=zeros(1,Resolution*Cycles);
I_Cmr=zeros(1,Resolution*Cycles);
I_Cs=zeros(1,Resolution*Cycles);
V_Lf=zeros(1,Resolution*Cycles);
V_Lmr=zeros(1,Resolution*Cycles);
V_Ls=zeros(1,Resolution*Cycles);
I_Sw=zeros(1,Resolution*Cycles);

V_ZVS=zeros(1,Cycles);
I_ZDVS=zeros(1,Cycles);

k=1;

for z=1:1:Cycles
   for l=1:1:Resolution
        % Define varying state matricies
        r=r_sw(l);

        A=[-1/(Cf*r) , 0 , 0 , 1/Cf , -1/Cf , -1/Cf;
           0 , 0 , 0 , 0 , 1/Cmr , 0;
           0 , 0 , 0 , 0 , 0 , 1/Cs;
           -1/Lf , 0 , 0 , 0 , 0 , 0;
           1/Lmr , -1/Lmr , 0 , 0 , 0 , 0;
           1/Ls , 0 , -1/Ls , 0 , 0 , -R_L/Ls];

        % Calculate next step using new model
        X_n=(expm(A*t)*X_0);
        X_f=A\(expm(A*t)-eye(6))*(B_EF*U);
        X=X_n+X_f;

        Xdot=(A*X)+(B_EF*U);
```

```
        Y=(C_EF*X)+(D_EF*U);

        % Save state-vectors to individual variables
        V_Cf(k)=Y(1,1);
        V_Cmr(k)=Y(2,1);
        V_Cs(k)=Y(3,1);
        I_Lf(k)=Y(4,1);
        I_Lmr(k)=Y(5,1);
        I_Ls(k)=Y(6,1);
        I_Cf(k)=Cf*Xdot(1,1);
        I_Cmr(k)=Cmr*Xdot(2,1);
        I_Cs(k)=Cs*Xdot(3,1);
        V_Lf(k)=Lf*Xdot(4,1);
        V_Lmr(k)=Lmr*Xdot(5,1);
        V_Ls(k)=Ls*Xdot(6,1);
        I_Sw(k)=I_Lf(k)-I_Cf(k)-I_Lmr(k)-I_Ls(k);

        if l==d_off
            V_ZVS(z)=V_Cf(k);
            I_ZDVS(z)=I_Sw(k)+I_Cf(k);
        end

        % Update initial conditions and counter variable(s)
        X_0=X;
        k=k+1;
    end
end

%% Post Processing
P_inCalc=U*mean(I_Lf);                          % Input Power
P_outCalc=(rms(I_Ls)^2)*R_L;                    % Output Power
Eff=abs(P_outCalc/P_inCalc);                    % Efficiency

I_Rip=max(abs(I_Lf))-min(abs(I_Lf));            % Input Current
Ripple
rf=I_Rip/mean(I_Lf);

V_start=[V_Cf(Resolution+1) , V_Cf(Resolution+2) ,
V_Cf(Resolution+3) , ...
         V_Cf(Resolution+4) , V_Cf(Resolution+5)];
LowV=min(V_start);

ZVS=max(abs(V_ZVS));                            % Take worst case
ZVS and ZDVS
```

```
ZDVS=max(abs(I_ZDVS));
Location=find(abs(V_ZVS)==ZVS);
Value=floor(Resolution*(1-D))+(Resolution*(Location-1));

%% FFT/THD Calculations
Fs=Resolution*f_s;                % Define Sampling Frequency
L=Resolution*Cycles;              % Define Window Length
Harm_out=fft(I_Ls*R_L);           % Calculate Output Voltage
FFT

P2 = abs(Harm_out/L);             % Steps provided in MATLAB
documentation
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

val=zeros(7,1);

for x=1:1:7                           % Find positions of 1st
7 harmonics
    val(x)=find(f==f_s*x);            % for THD calc
end

THD_top=sqrt(P1(val(2))^2+P1(val(3))^2+P1(val(4))^2+P1(val(
5))^2+P1(val(6))^2+P1(val(7))^2);
THD_bot=sqrt(P1(val(1))^2+P1(val(2))^2+P1(val(3))^2+P1(val(
4))^2+P1(val(5))^2+P1(val(6))^2+P1(val(7))^2);
THD=THD_top/THD_bot;

%% I_Cf Calc
I_check=I_Cf(1:d_off);
L1=d_off;                     % Define Window Length
Harm1=fft(I_check);           % Calculate Output Voltage FFT

P2_I = abs(Harm1/L1);         % Steps provided in MATLAB
documentation
P1_I = P2_I(1:floor(L1/2)+1);
P1_I(2:end-1) = 2*P1_I(2:end-1);

I_Num=find(max(P1_I)==P1_I);


%% MOPSO objective function calculations
if(THD>minMaxValues(1))
```

```
    Outcome(:,6)=Outcome(:,6)+1;
end

if(Eff<minMaxValues(2))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(rf>minMaxValues(3))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(I_Num>=minMaxValues(4))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(V_Cf(Value)>V_Cf(Value-1))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(LowV<0)
    Outcome(:,6)=Outcome(:,6)+1;
end

Outcome(:,1)=abs(1-Eff)/(1-minMaxValues(2));
Outcome(:,2)=ZVS/max(V_Cf);
Outcome(:,3)=ZDVS/max(abs(I_Sw+I_Cf));
Outcome(:,4)=THD/minMaxValues(1);
Outcome(:,5)=abs(P_Out-P_outCalc)/P_Out;

%% MOGO Objective function calculations
if(I_Num>=minMaxValues(1))
    Outcome(:,6)=1;
end

if(THD>=minMaxValues(2))
    Outcome(:,6)=1;
end

if(Eff<=minMaxValues(3))
    Outcome(:,6)=1;
end

if(rf>minMaxValues(4))
    Outcome(:,6)=1;
```

```
end

if(V_Cf(Value)>V_Cf(Value-1))
    Outcome(:,6)=1;
end

if(LowV<0)
    Outcome(:,6)=1;
end

Outcome(:,2)=ZVS/max(V_Cf);
Outcome(:,3)=ZDVS/max(abs(I_Cf+I_Sw));
Outcome(:,4)=THD;

if(P_outCalc<(0.667*P_Out) || P_outCalc>(1.333*P_Out))
    Outcome(:,5)=1;
else
    Outcome(:,5)=(3*abs(P_Out-P_outCalc))/P_Out;
end
```

Appendix B

Fitness Function for the Parasitic Class-EF_2 Inverter

```
function Outcome =
EF_Parasitic_Model(f_s,r_sw_on,r_sw_off,R_L,P_Out,minMaxVal
ues,inputParam)
%% Sort out parameters from optimizer
Outcome=zeros(1,6);

Cycles=3;                % # Cycles to Fully Calculate
Resolution=300;          % # of Samples/Cycle
P=1/f_s;                 % Switching Period
w=2*pi*f_s;              % Angular Frequency

% Parasitic Calculation Definitions
C2=200e-12;          % Standard Values used for Parasitic
Inductance Calculation
L2=126.7e-12;
C_para=2.1e-12;      % Average Parasitic Capacitance Value
R_para=50e-3;        % Average Parasitic Resistance Value
QL=150;              % Inductor Quality Factor

D=inputParam(1);         % Extract values from optimzer for
calculations
Lf=inputParam(2);
Cf=inputParam(3);
Cmr=inputParam(4)*Cf;
Lmr=1/(((2*w)^2)*Cmr);
Ls=(inputParam(5)*R_L)/(w*inputParam(6));
Cs=1/(((inputParam(6)*w)^2)*Ls);
U=inputParam(7);


% Parasitic Calculations
C_Lf=3.35e-12;
C_Lmr=C_para;
C_Ls=C_para;
L_Cf=10^(log10(L2)-((log10(C2)-log10(Cf))/-7.213));     % -
7.213 is slope of
L_Cmr=10^(log10(L2)-((log10(C2)-log10(Cmr))/-7.213));   %
Capacitance SRF curve
```

```
L_Cs=10^(log10(L2)-((log10(C2)-log10(Cs))/-7.213));      %
from Data Sheet

r_Lf=220e-3;
r_Lmr=w*Lmr/QL;
r_Ls=w*Ls/QL;
r_Cf=R_para;
r_Cmr=R_para;
r_Cs=R_para;

t=P/Resolution;      % State-Space evaluation time
t_on=D*P;                % Transistor On time
t_off=(1-D)*P;           % Transistor Off time

d_off=floor(Resolution*(1-D));          % # of samples in
Off-State
d_on=Resolution-d_off;                  % # of samples in
On-State

r_off=zeros(1,d_off)+r_sw_off;          % Resistance
Vectors
r_on=zeros(1,d_on)+r_sw_on;
r_sw=[r_off , r_on];

%% Initialize State-Space Model
A_on=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf , 0 , 0;
      0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cmr , 0;
      0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cs;
      0 , 0 , 0 , -1/(r_sw_on*C_Lf) , 0 , 0 , -1/C_Lf , 0 ,
0 , 1/C_Lf , 1/C_Lf , 1/C_Lf;
      0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0 , 0 ,
1/C_Lmr , 0;
      0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls , 0 , 0 ,
1/C_Ls;
      0 , 0 , 0 , 1/Lf , 0 , 0 , -r_Lf/Lf , 0 , 0 , 0 , 0 ,
0;
      0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , -r_Lmr/Lmr , 0 , 0 ,
0 , 0;
      0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , -r_Ls/Ls , 0 , 0 ,
0;
      -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 , -
r_Cf/L_Cf , 0 , 0;
      0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 , 0 ,
0 , 0 , -r_Cmr/L_Cmr , 0;
```

```
      0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0 , 0 ,
0 , 0 , -(r_Cs+R_L)/L_Cs];

A_off=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf , 0 , 0;
       0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cmr , 0;
       0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cs;
       0 , 0 , 0 , -1/(r_sw_off*C_Lf) , 0 , 0 , -1/C_Lf , 0
, 0 , 1/C_Lf , 1/C_Lf , 1/C_Lf;
       0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0 , 0 ,
1/C_Lmr , 0;
       0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls , 0 , 0 ,
1/C_Ls;
       0 , 0 , 0 , 1/Lf , 0 , 0 , -r_Lf/Lf , 0 , 0 , 0 , 0
, 0;
       0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , -r_Lmr/Lmr , 0 , 0 ,
0 , 0;
       0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , -r_Ls/Ls , 0 , 0
, 0;
       -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 , -
r_Cf/L_Cf , 0 , 0;
       0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 , 0 ,
0 , 0 , -r_Cmr/L_Cmr , 0;
       0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0 , 0
, 0 , 0 , -(r_Cs+R_L)/L_Cs];

B_on=[0;
      0;
      0;
      1/(r_sw_on*C_Lf);
      0;
      0;
      0;
      0;
      0;
      1/L_Cf;
      1/L_Cmr;
      1/L_Cs];

B_off=[0;
       0;
       0;
       1/(r_sw_off*C_Lf);
       0;
       0;
```

```
        0;
        0;
        0;
        1/L_Cf;
        1/L_Cmr;
        1/L_Cs];

C=eye(12);
D_EF=zeros(12,1);

%% Steady-State Calculations
SSCycles=600;        % Number of Cycles to reach steady-
state
X_0=zeros(12,1);     % Zero initial condition

for l=1:1:SSCycles
    if mod(l,2)==0
        X_n=(expm(A_on*t_on)*X_0);
        X_f=A_on\(expm(A_on*t_on)-eye(12))*(B_on*U);
        X=X_n+X_f;
    else
        X_n=(expm(A_off*t_off)*X_0);
        X_f=A_off\(expm(A_off*t_off)-eye(12))*(B_off*U);
        X=X_n+X_f;
    end
        X_0=X;
end

%% Fully Calculate State Vectors for "Cycles" cycles
V_Cf=zeros(1,Resolution*Cycles);        % Pre-allocate all
parameter vectors
V_Cmr=zeros(1,Resolution*Cycles);
V_Cs=zeros(1,Resolution*Cycles);
V_C_Lf=zeros(1,Resolution*Cycles);
V_C_Lmr=zeros(1,Resolution*Cycles);
V_C_Ls=zeros(1,Resolution*Cycles);
I_Lf=zeros(1,Resolution*Cycles);
I_Lmr=zeros(1,Resolution*Cycles);
I_Ls=zeros(1,Resolution*Cycles);
I_L_Cf=zeros(1,Resolution*Cycles);
I_L_Cmr=zeros(1,Resolution*Cycles);
I_L_Cs=zeros(1,Resolution*Cycles);

I_Cf=zeros(1,Resolution*Cycles);
```

```
I_Cmr=zeros(1,Resolution*Cycles);
I_Cs=zeros(1,Resolution*Cycles);
I_C_Lf=zeros(1,Resolution*Cycles);
I_C_Lmr=zeros(1,Resolution*Cycles);
I_C_Ls=zeros(1,Resolution*Cycles);
V_Lf=zeros(1,Resolution*Cycles);
V_Lmr=zeros(1,Resolution*Cycles);
V_Ls=zeros(1,Resolution*Cycles);
V_L_Cf=zeros(1,Resolution*Cycles);
V_L_Cmr=zeros(1,Resolution*Cycles);
V_L_Cs=zeros(1,Resolution*Cycles);

I_Sw=zeros(1,Resolution*Cycles);
V_Sw=zeros(1,Resolution*Cycles);
V_ZVS=zeros(1,Cycles);
I_ZDVS=zeros(1,Cycles);

j=1;        % Initialize Counter Variable

for z=1:1:Cycles
   for l=1:1:Resolution
       % Define varying state-space matricies
       r=r_sw(l);

       A=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf , 0 , 0;
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cmr ,
0;
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cs;
          0 , 0 , 0 , -1/(r*C_Lf) , 0 , 0 , -1/C_Lf , 0 , 0
, 1/C_Lf , 1/C_Lf , 1/C_Lf;
          0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0 , 0 ,
1/C_Lmr , 0;
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls , 0 , 0 ,
1/C_Ls;
          0 , 0 , 0 , 1/Lf , 0 , 0 , -r_Lf/Lf , 0 , 0 , 0 ,
0 , 0;
          0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , -r_Lmr/Lmr , 0 ,
0 , 0 , 0;
          0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , -r_Ls/Ls , 0 ,
0 , 0;
          -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 , -
r_Cf/L_Cf , 0 , 0;
          0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 ,
0 , 0 , 0 , -r_Cmr/L_Cmr , 0;
```

```
          0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0 ,
0 , 0 , 0 , -(r_Cs+R_L)/L_Cs];

       B=[0;
          0;
          0;
          1/(r*C_Lf);
          0;
          0;
          0;
          0;
          0;
          1/L_Cf;
          1/L_Cmr;
          1/L_Cs];

       % Calculate next step using new model
       X_n=(expm(A*t)*X_0);
       X_f=A\(expm(A*t)-eye(12))*(B*U);
       X=X_n+X_f;

       Xdot=(A*X)+(B*U);
       Y=(C*X)+(D_EF*U);

       % Save state-vectors to individual variables
        V_Cf(j)=Y(1,1);
        V_Cmr(j)=Y(2,1);
        V_Cs(j)=Y(3,1);
        V_C_Lf(j)=Y(4,1);
        V_C_Lmr(j)=Y(5,1);
        V_C_Ls(j)=Y(6,1);
        I_Lf(j)=Y(7,1);
        I_Lmr(j)=Y(8,1);
        I_Ls(j)=Y(9,1);
        I_L_Cf(j)=Y(10,1);
        I_L_Cmr(j)=Y(11,1);
        I_L_Cs(j)=Y(12,1);

        I_Cf(j)=Cf*Xdot(1,1);
        I_Cmr(j)=Cmr*Xdot(2,1);
        I_Cs(j)=Cs*Xdot(3,1);
        I_C_Lf(j)=C_Lf*Xdot(4,1);
        I_C_Lmr(j)=C_Lmr*Xdot(5,1);
        I_C_Ls(j)=C_Ls*Xdot(6,1);
```

```
        V_Lf(j)=Lf*Xdot(7,1);
        V_Lmr(j)=Lmr*Xdot(8,1);
        V_Ls(j)=Ls*Xdot(9,1);
        V_L_Cf(j)=L_Cf*Xdot(10,1);
        V_L_Cmr(j)=L_Cmr*Xdot(11,1);
        V_L_Cs(j)=L_Cs*Xdot(12,1);

        V_Sw(j)=V_Cf(j)+V_L_Cf(j)+(r_Cf*I_L_Cf(j));
        I_Sw(j)=I_Lf(j)+I_C_Lf(j)-I_L_Cf(j)-I_L_Cmr(j)-
I_L_Cs(j);

       if l==d_off
           V_ZVS(z)=V_Sw(j);
           I_ZDVS(z)=I_L_Cf(j)+I_Sw(j);
       else
       end

       % Update initial conditions and counter variable(s)
       X_0=X;
       j=j+1;
   end
end

%% Post Processing
P_inCalc=U*mean(I_Lf+I_C_Lf);                     % Power Calc
P_outCalc=(rms(I_L_Cs)^2)*R_L;
Eff=abs(P_outCalc/P_inCalc);                       % Efficiency
Calc

i_rip=abs(max(I_Lf)-min(I_Lf));
rf=i_rip/mean(I_Lf);

V_start=[V_Sw(Resolution+1) , V_Sw(Resolution+2) ,
V_Sw(Resolution+3) , ...
         V_Sw(Resolution+4) , V_Sw(Resolution+5)];
LowV=min(V_start);
problem=find(LowV==V_start);

if(problem==1)
    if(V_start(problem+1)>0)
        LowV=1;
    else
    end
elseif(problem==5)
```

```
    if(V_start(problem-1)>0)
        LowV=1;
    else
    end
else
    if(V_start(problem+1)>0 && V_start(problem-1)>0)
        LowV=1;
    else
    end
end

ZVS=max(abs(V_ZVS));                          % Worst Case ZVS &
ZDVS
ZDVS=max(abs(I_ZDVS));
Location=find(abs(V_ZVS)==ZVS);
Value=floor(Resolution*(1-D))+(Resolution*(Location-1));

%% FFT/THD Calculations
Fs=Resolution*f_s;                            % Define Sampling
Frequency
L=Resolution*Cycles;                          % Define Window
Length
Harm_out=fft(I_L_Cs*R_L);                         % Calculate
Output Voltage FFT

P2 = abs(Harm_out/L);                 % Steps provided in MATLAB
documentation
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

val=zeros(7,1);

for x=1:1:7                               % Find positions of 1st
7 harmonics
    val(x)=find(f==f_s*x);                % for THD calc
end

THD_top=sqrt(P1(val(2))^2+P1(val(3))^2+P1(val(4))^2+P1(val(
5))^2+P1(val(6))^2+P1(val(7))^2);
THD_bot=sqrt(P1(val(1))^2+P1(val(2))^2+P1(val(2))^2+P1(val(
4))^2+P1(val(5))^2+P1(val(6))^2+P1(val(7))^2);
THD=THD_top/THD_bot;
```

```
%% I_Cf Calc
I_check=I_Cf(1:d_off);
L1=d_off;                      % Define Window Length
Harm1=fft(I_check);            % Calculate Output Voltage FFT

P2_I = abs(Harm1/L1);          % Steps provided in MATLAB
documentation
P1_I = P2_I(1:floor(L1/2)+1);
P1_I(2:end-1) = 2*P1_I(2:end-1);

I_Num=find(max(P1_I)==P1_I);

%% MOPSO Objective function values
if(THD>minMaxValues(1))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(Eff<minMaxValues(2))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(rf>minMaxValues(3))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(I_Num>=minMaxValues(4))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(V_Sw(Value)>V_Sw(Value-1))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(LowV<0)
    Outcome(:,6)=Outcome(:,6)+1;
end

if(P_outCalc<(0.5*P_Out) || P_outCalc>(1.5*P_Out))
    Outcome(:,6)=Outcome(:,6)+1;
end

Outcome(:,1)=abs(1-Eff);
Outcome(:,2)=ZVS/max(V_Sw);
Outcome(:,3)=ZDVS/max(abs(I_Sw+I_Cf));
```

```
Outcome(:,4)=THD;
Outcome(:,5)=abs(P_Out-P_outCalc)/P_Out;

%% MOGO Objective function values
if(I_Num>=minMaxValues(1))
    Outcome(:,6)=1;
end

if(rf>minMaxValues(4))
    Outcome(:,6)=1;
end

if(V_Sw(Value)>V_Sw(Value-1))
    Outcome(:,6)=1;
end

if(LowV<0)
    Outcome(:,6)=1;
end

if(Eff<=minMaxValues(3))
    Outcome(:,1)=1;
else
    Outcome(:,1)=abs(1-Eff)/(1-minMaxValues(3));
end

if(ZVS>=((1/3)*max(V_Sw)))
    Outcome(:,2)=1;
else
    Outcome(:,2)=(3*ZVS)/max(V_Sw);
end

if(ZDVS>=((1/3)*max(abs(I_Cf+I_Sw))))
    Outcome(:,3)=1;
else
    Outcome(:,3)=(3*ZDVS)/max(abs(I_Cf+I_Sw));
end

if(THD>=minMaxValues(2))
    Outcome(:,4)=1;
else
    Outcome(:,4)=THD/minMaxValues(2);
end
```

```
if(P_outCalc<(0.667*P_Out) || P_outCalc>(1.333*P_Out))
    Outcome(:,5)=1;
else
    Outcome(:,5)=(3*abs(P_Out-P_outCalc))/P_Out;
end
```

Appendix C

Fitness Function for the Practical Class-EF_2 Inverter

```
function Outcome =
EF_Practical_Model(f_s,r_sw_on,r_sw_off,rg,rs,R_L,P_Out,min
MaxValues,inputParam)
%% Sort out Parameters from Optimizer
Outcome=zeros(1,6);

Resolution=300;          % # of samples/cycle
Cycles=6;               % # of cycles
Adjustment=3;            % # of adjustment cycles
P=1/f_s;                 % Switching Period
w=2*pi*f_s;              % Angular Frequency

%% Component and Parasitic Calculations
C2=200e-12;              % Standard Values used for
Parasitic Inductance Calculation
L2=126.7e-12;
C_para=2.1e-12;          % Average Parasitic Capacitance
Value
R_para=50e-3;            % Average Parasitic Resistance
Value
QL=150;                  % Inductor Quality Factor

D=inputParam(1);         % Extract values from optimzer for
calculations
Lf=inputParam(2);
Cf=inputParam(3);
Cmr=(Cf+Cds)*inputParam(4);
Lmr=1/(((2*w)^2)*Cmr);
Ls=(inputParam(5)*R_L)/(w*inputParam(6));
Cs=1/(((inputParam(6)*w)^2)*Ls);
V_in=inputParam(7);

C_Lf=3.35e-12;
C_Lmr=C_para;
C_Ls=C_para;
L_Cf=10^(log10(L2)-((log10(C2)-log10(Cf))/-7.213));      % -
7.213 is slope of
L_Cmr=10^(log10(L2)-((log10(C2)-log10(Cmr))/-7.213));    %
Capacitance SRF curve
```

```
L_Cs=10^(log10(L2)-((log10(C2)-log10(Cs))/-7.213));     %
from Data Sheet

r_Lf=220e-3;
r_Lmr=w*Lmr/QL;
r_Ls=w*Ls/QL;
r_Cf=R_para;
r_Cmr=R_para;
r_Cs=R_para;

t=P/Resolution;          % State-Space evaluation time
t_on=D*P;                % Transistor On time
t_off=(1-D)*P;           % Transistor Off time

%% Define Switch Parameters
d_trans=4;                                    % # of samples
in transition - 3, 4, 6 for design cases
d_off=floor(Resolution*(1-D))-d_trans;        % # of samples
in Off-State
d_on=(Resolution-d_off)-(2*d_trans);          % # of samples
in On-State

Vg_off=zeros(1,d_off);                    % Gate Voltage
Vectors
Vg_trans_on=linspace(0,5,d_trans);
Vg_on=zeros(1,d_on)+5;
Vg_trans_off=linspace(5,0,d_trans);
Vg_total=[Vg_off , Vg_trans_on , Vg_on , Vg_trans_off];

r_off=zeros(1,d_off)+r_sw_off;            % Resistance
Vectors
r_trans_on=logspace(6,-2,d_trans);
r_on=zeros(1,d_on)+r_sw_on;
r_trans_off=logspace(-2,6,d_trans);
r_sw=[r_off , r_trans_on , r_on , r_trans_off];

U1=V_in;                     % Pre-define U1 for initial
calc.
U2_on=Vg_on(1);              % Pre-define U2 for initial
calc.
U2_off=Vg_off(1);
U2=Vg_total(1);
r=r_sw(1);                   % Pre-define r for initial
calc.
```

```
%% State-Space Model Initialization
A_new_on=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf
, 0 , 0;                                            % V_Cf
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
1/Cmr , 0;                                           % V_Cmr
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 1/Cs;                                             % V_Cs
          0 , 0 , 0 , -(rg+rs)/(C_Lf*rg*rs) , 0 , 0 , -
((rs*r_sw_on)+(rg*r_sw_on))/(C_Lf*rg*rs*r_sw_on) ,...
          1/(C_Lf*rg) , -1/C_Lf , 0 , 0 , 1/C_Lf , 1/C_Lf ,
1/C_Lf;                                             % V_C_Lf
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0
, 0 , 1/C_Lmr , 0;                                  % V_C_Lmr
          0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls ,
0 , 0 , 1/C_Ls;                                     % V_C_Ls
          0 , 0 , 0 , -(rg+rs)/(Cds*rg*rs) , 0 , 0 , -
((rs*rg)+(rs*r_sw_on)+(rg*r_sw_on))/(Cds*rs*rg*r_sw_on)
,...
          1/(Cds*rg) , 0 , 0 , 0 , 0 , 0 , 0;
% V_Cds
          0 , 0 , 0 , 1/(Cgs*rg) , 0 , 0 , 1/(Cgs*rg) , -
1/(Cgs*rg) , 0 , 0 , 0 , 0 , 0 , 0;                  % V_Cgs
          0 , 0 , 0 , 1/Lf , 0 , 0 , 0 , 0 , -r_Lf/Lf , 0 ,
0 , 0 , 0 , 0;                                      % I_Lf
          0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , 0 , 0 , -
r_Lmr/Lmr , 0 , 0 , 0 , 0;                               % I_Lmr
          0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , 0 , 0 , -
r_Ls/Ls , 0 , 0 , 0;                                  % I_Ls
          -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 , 0
, 0 , -r_Cf/L_Cf , 0 , 0;                           % I_L_Cf
          0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , -r_Cmr/L_Cmr , 0;               % I_L_Cmr
          0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , -(r_Cs+R_L)/L_Cs];              % I_L_Cs

A_new_off=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf
, 0 , 0;                                            % V_Cf
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
1/Cmr , 0;                                           % V_Cmr
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 1/Cs;                                            % V_Cs
           0 , 0 , 0 , -(rg+rs)/(C_Lf*rg*rs) , 0 , 0 , -
((rs*r_sw_off)+(rg*r_sw_off))/(C_Lf*rg*rs*r_sw_off) ,...
```

```
            1/(C_Lf*rg) , -1/C_Lf , 0 , 0 , 1/C_Lf , 1/C_Lf
, 1/C_Lf;                                            % V_C_Lf
            0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0
, 0 , 1/C_Lmr , 0;                                  % V_C_Lmr
            0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls
, 0 , 0 , 1/C_Ls;                                    % V_C_Ls
            0 , 0 , 0 , -(rg+rs)/(Cds*rg*rs) , 0 , 0 , -
((rs*rg)+(rs*r_sw_off)+(rg*r_sw_off))/(Cds*rs*rg*r_sw_off)
,...
            1/(Cds*rg) , 0 , 0 , 0 , 0 , 0 , 0;
% V_Cds
            0 , 0 , 0 , 1/(Cgs*rg) , 0 , 0 , 1/(Cgs*rg) , -
1/(Cgs*rg) , 0 , 0 , 0 , 0 , 0 , 0;                  % V_Cgs
            0 , 0 , 0 , 1/Lf , 0 , 0 , 0 , 0 , -r_Lf/Lf , 0
, 0 , 0 , 0 , 0;                                     % I_Lf
            0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , 0 , 0 , -
r_Lmr/Lmr , 0 , 0 , 0 , 0;                                   % I_Lmr
            0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , 0 , 0 , -
r_Ls/Ls , 0 , 0 , 0;                                     % I_Ls
            -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , -r_Cf/L_Cf , 0 , 0;                          % I_L_Cf
            0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , -r_Cmr/L_Cmr , 0;               % I_L_Cmr
            0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0
, 0 , 0 , 0 , 0 , 0 , -(r_Cs+R_L)/L_Cs];             % I_L_Cs

B_new=[0 , 0;
       0 , 0;
       0 , 0;
       (rg+rs)/(C_Lf*rg*rs) , -1/(C_Lf*rg);
       0 , 0;
       0 , 0;
       (rg+rs)/(Cds*rg*rs) , -1/(Cds*rg);
       -1/(Cgs*rg) , 1/(Cgs*rg);
       0 , 0;
       0 , 0;
       0 , 0;
       1/L_Cf , 0;
       1/L_Cmr , 0;
       1/L_Cs , 0];

C_new=eye(14);

D_new=zeros(14,2);
```

```
%% Get to Steady-State
SSCycles=550;            % Number of Cycles to reach steady-
state - 300, 550, 1100 for design cases
X_0=zeros(14,1);         % Zero initial condition
U_on=[U1;U2_on];         % On-State Input
U_off=[U1;U2_off];       % Off-State Input

for l=1:1:SSCycles
    if mod(l,2)==0
        X_n=(expm(A_new_on*t_on)*X_0);
        X_f=A_new_on\(expm(A_new_on*t_on)-
eye(14))*(B_new*U_on);
        X=X_n+X_f;
    else
        X_n=(expm(A_new_off*t_off)*X_0);
        X_f=A_new_off\(expm(A_new_off*t_off)-
eye(14))*(B_new*U_off);
        X=X_n+X_f;
    end
        X_0=X;
end

%% Pre-allocate State Variables
V_Cf=zeros(1,Resolution*(Cycles-Adjustment));
V_Cmr=zeros(1,Resolution*(Cycles-Adjustment));
V_Cs=zeros(1,Resolution*(Cycles-Adjustment));
V_C_Lf=zeros(1,Resolution*(Cycles-Adjustment));
V_C_Lmr=zeros(1,Resolution*(Cycles-Adjustment));
V_C_Ls=zeros(1,Resolution*(Cycles-Adjustment));
V_ds=zeros(1,Resolution*(Cycles-Adjustment));
V_gs=zeros(1,Resolution*(Cycles-Adjustment));
I_Lf=zeros(1,Resolution*(Cycles-Adjustment));
I_Lmr=zeros(1,Resolution*(Cycles-Adjustment));
I_Ls=zeros(1,Resolution*(Cycles-Adjustment));
I_L_Cf=zeros(1,Resolution*(Cycles-Adjustment));
I_L_Cmr=zeros(1,Resolution*(Cycles-Adjustment));
I_L_Cs=zeros(1,Resolution*(Cycles-Adjustment));

I_Cf=zeros(1,Resolution*(Cycles-Adjustment));
I_Cmr=zeros(1,Resolution*(Cycles-Adjustment));
I_Cs=zeros(1,Resolution*(Cycles-Adjustment));
I_C_Lf=zeros(1,Resolution*(Cycles-Adjustment));
I_C_Lmr=zeros(1,Resolution*(Cycles-Adjustment));
```

```
I_C_Ls=zeros(1,Resolution*(Cycles-Adjustment));
I_ds=zeros(1,Resolution*(Cycles-Adjustment));
I_gs=zeros(1,Resolution*(Cycles-Adjustment));
V_Lf=zeros(1,Resolution*(Cycles-Adjustment));
V_Lmr=zeros(1,Resolution*(Cycles-Adjustment));
V_Ls=zeros(1,Resolution*(Cycles-Adjustment));
V_L_Cf=zeros(1,Resolution*(Cycles-Adjustment));
V_L_Cmr=zeros(1,Resolution*(Cycles-Adjustment));
V_L_Cs=zeros(1,Resolution*(Cycles-Adjustment));

I_Sw=zeros(1,Resolution*(Cycles-Adjustment));
V_Sw=zeros(1,Resolution*(Cycles-Adjustment));
V_ZVS=zeros(1,(Cycles-Adjustment));
I_ZDVS=zeros(1,(Cycles-Adjustment));

k=1;            % Define counter variable for state-vectors

%% State-Space Test
for z=1:1:Cycles
   for l=1:1:Resolution
       % Define varying state-space matricies
       U=[U1;
          U2];

        A=[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1/Cf
, 0 , 0;                                              % V_Cf
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
1/Cmr , 0;                                            % V_Cmr
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 1/Cs;                                             % V_Cs
           0 , 0 , 0 , -(rg+rs)/(C_Lf*rg*rs) , 0 , 0 , -
((rs*r)+(rg*r))/(C_Lf*rg*rs*r) ,...
           1/(C_Lf*rg) , -1/C_Lf , 0 , 0 , 1/C_Lf , 1/C_Lf
, 1/C_Lf;                                             % V_C_Lf
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Lmr , 0
, 0 , 1/C_Lmr , 0;                                   % V_C_Lmr
           0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , -1/C_Ls
, 0 , 0 , 1/C_Ls;                                     % V_C_Ls
           0 , 0 , 0 , -(rg+rs)/(Cds*rg*rs) , 0 , 0 , -
((rs*rg)+(rs*r)+(rg*r))/(Cds*rs*rg*r) ,...
           1/(Cds*rg) , 0 , 0 , 0 , 0 , 0 , 0;
% V_Cds
           0 , 0 , 0 , 1/(Cgs*rg) , 0 , 0 , 1/(Cgs*rg) , -
1/(Cgs*rg) , 0 , 0 , 0 , 0 , 0 , 0;                   % V_Cgs
```

```
            0 , 0 , 0 , 1/Lf , 0 , 0 , 0 , 0 , -r_Lf/Lf , 0
, 0 , 0 , 0 , 0;                                      % I_Lf
            0 , 0 , 0 , 0 , 1/Lmr , 0 , 0 , 0 , 0 , -
r_Lmr/Lmr , 0 , 0 , 0 , 0;                                   % I_Lmr
            0 , 0 , 0 , 0 , 0 , 1/Ls , 0 , 0 , 0 , 0 , -
r_Ls/Ls , 0 , 0 , 0;                                      % I_Ls
            -1/L_Cf , 0 , 0 , -1/L_Cf , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , -r_Cf/L_Cf , 0 , 0;                          % I_L_Cf
            0 , -1/L_Cmr , 0 , -1/L_Cmr , -1/L_Cmr , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , -r_Cmr/L_Cmr , 0;           % I_L_Cmr
            0 , 0 , -1/L_Cs , -1/L_Cs , 0 , -1/L_Cs , 0 , 0
, 0 , 0 , 0 , 0 , 0 , -(r_Cs+R_L)/L_Cs];        % I_L_Cs

        % Calculate next step using new model
        X_n=(expm(A*t)*X_0);
        X_f=A\(expm(A*t)-eye(14))*(B_new*U);
        X=X_n+X_f;

        Xdot=(A*X)+(B_new*U);
        Y=(C_new*X)+(D_new*U);

        if(z>Adjustment)
            % Save state-vectors to individual variables
            V_Cf(k)=Y(1,1);
            V_Cmr(k)=Y(2,1);
            V_Cs(k)=Y(3,1);
            V_C_Lf(k)=Y(4,1);
            V_C_Lmr(k)=Y(5,1);
            V_C_Ls(k)=Y(6,1);
            V_ds(k)=Y(7,1);
            V_gs(k)=Y(8,1);
            I_Lf(k)=Y(9,1);
            I_Lmr(k)=Y(10,1);
            I_Ls(k)=Y(11,1);
            I_L_Cf(k)=Y(12,1);
            I_L_Cmr(k)=Y(13,1);
            I_L_Cs(k)=Y(14,1);

            I_Cf(k)=Cf*Xdot(1,1);
            I_Cmr(k)=Cmr*Xdot(2,1);
            I_Cs(k)=Cs*Xdot(3,1);
            I_C_Lf(k)=C_Lf*Xdot(4,1);
            I_C_Lmr(k)=C_Lmr*Xdot(5,1);
            I_C_Ls(k)=C_Ls*Xdot(6,1);
```

```
            I_ds(k)=Cds*Xdot(7,1);
            I_gs(k)=Cgs*Xdot(8,1);
            V_Lf(k)=Lf*Xdot(9,1);
            V_Lmr(k)=Lmr*Xdot(10,1);
            V_Ls(k)=Ls*Xdot(11,1);
            V_L_Cf(k)=L_Cf*Xdot(12,1);
            V_L_Cmr(k)=L_Cmr*Xdot(13,1);
            V_L_Cs(k)=L_Cs*Xdot(14,1);

            V_Sw(k)=V_Cf(k)+V_L_Cf(k)+(r_Cf*I_L_Cf(k));
            I_Sw(k)=(U1-V_C_Lf(k)-V_ds(k))/rs;

            if l==(d_off+round(d_trans/2))
                V_ZVS(z-Adjustment)=V_Sw(k);
                I_ZDVS(z-Adjustment)=I_Sw(k)+I_L_Cf(k);
            else
            end
            k=k+1;
        end

        % Update varying parameters
        U2=Vg_total(l);
        r=r_sw(l);

        % Update initial conditions and counter variable(s)
        X_0=X;
    end
end

%% Post Processing
P_inCalc=V_in*mean(I_Lf+I_C_Lf);          % Power
Calculations
P_outCalc=(rms(I_L_Cs)^2)*R_L;
Eff=P_outCalc/P_inCalc;                   % Efficiency

i_rip=abs(max(I_Lf)-min(I_Lf));
rf=i_rip/mean(I_Lf);

V_start=[V_Sw(Resolution+1) , V_Sw(Resolution+2) ,
V_Sw(Resolution+3) , ...
         V_Sw(Resolution+4) , V_Sw(Resolution+5)];

LowV=min(V_start);
problem=find(LowV==V_start);
```

```
if(problem==1)
    if(V_start(problem+1)>0)
        LowV=1;
    else
    end
elseif(problem==5)
    if(V_start(problem-1)>0)
        LowV=1;
    else
    end
else
    if(V_start(problem+1)>0 && V_start(problem-1)>0)
        LowV=1;
    else
    end
end

ZVS=max(abs(V_ZVS));                        % Worst Case ZVS &
ZDVS
ZDVS=max(abs(I_ZDVS));
Location=find(abs(V_ZVS)==ZVS);
Value=floor(Resolution*(1-D))+(Resolution*(Location-1))-
round(d_trans/2);

%% FFT/THD Calculations
Fs=Resolution*f_s;                          % Define Sampling
Frequency
L=Resolution*(Cycles-Adjustment);           % Define Window
Length
Harm_out=fft(I_L_Cs*R_L);                   % Calculate Output
Voltage FFT

P2 = abs(Harm_out/L);                       % Steps provided in
MATLAB documentation
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

val=zeros(7,1);

for x=1:1:7                                 % Find positions of
1st 7 harmonics
    val(x)=find(f==f_s*x);                  % for THD calc
```

```
end

THD_top=sqrt(P1(val(2))^2+P1(val(3))^2+P1(val(4))^2+P1(val(
5))^2+P1(val(6))^2+P1(val(7))^2);
THD_bot=sqrt(P1(val(1))^2+P1(val(2))^2+P1(val(2))^2+P1(val(
4))^2+P1(val(5))^2+P1(val(6))^2+P1(val(7))^2);
THD=THD_top/THD_bot;

%% I_Cf Calc
I_check=I_Cf(1:d_off);
L1=d_off;                    % Define Window Length
Harm1=fft(I_check);          % Calculate Output Voltage FFT

P2_I = abs(Harm1/L1);        % Steps provided in MATLAB
documentation
P1_I = P2_I(1:floor(L1/2)+1);
P1_I(2:end-1) = 2*P1_I(2:end-1);

I_Num=find(max(P1_I)==P1_I);

%% MOPSO Objective function values
if(THD>minMaxValues(1))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(Eff<minMaxValues(2))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(rf>minMaxValues(3))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(I_Num>=minMaxValues(4))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(V_Sw(Value)>V_Sw(Value-1))
    Outcome(:,6)=Outcome(:,6)+1;
end

if(LowV<0)
    Outcome(:,6)=Outcome(:,6)+1;
end
```

```
if(P_outCalc<(0.5*P_Out) || P_outCalc>(1.5*P_Out))
    Outcome(:,6)=Outcome(:,6)+1;
end

Outcome(:,1)=abs(1-Eff);
Outcome(:,2)=ZVS/max(V_Sw);
Outcome(:,3)=ZDVS/max(abs(I_Sw+I_Cf));
Outcome(:,4)=THD;
Outcome(:,5)=abs(P_Out-P_outCalc)/P_Out;

%% MOGO Objective function values
if(I_Num>=minMaxValues(1))
    Outcome(:,6)=1;
end

if(THD>=minMaxValues(2))
    Outcome(:,6)=1;
end

if(Eff<=minMaxValues(3))
    Outcome(:,6)=1;
end

if(rf>minMaxValues(4))
    Outcome(:,6)=1;
end

if(V_Cf(Value)>V_Cf(Value-1))
    Outcome(:,6)=1;
end

if(LowV<0)
    Outcome(:,6)=1;
end

if(ZVS>(max(V_Cf)/5))
    Outcome(:,6)=1;
end

Outcome(:,1)=abs(1-Eff);
Outcome(:,2)=ZVS/max(V_Cf);
Outcome(:,3)=ZDVS/max(abs(I_Cf+I_Sw));
Outcome(:,4)=THD;
```

```
if(P_outCalc<(0.667*P_Out) || P_outCalc>(1.333*P_Out))
    Outcome(:,5)=1;
else
    Outcome(:,5)=(3*abs(P_Out-P_outCalc))/P_Out;
end
```

Appendix D

MOGO Initialization Function

```
%% Reset Button
clear variables
clc

%% Optimization Setup
%converterType="Ideal_EF";         % Select Converter type
%converterType="Parasitic_EF";
converterType="Practical_EF";

f_s=13.56e6;                % Desired Switching Frequency
r_on=10e-3;                 % Transistor On Resistance (Ideal)
%r_on=36e-3;                % Transistor On Resistance
(EPC2019)
%r_on=120e-3;               % Transistor On Resistance
(EPC8010)
r_off=1e6;                  % Transistor Off Resistance (Ideal)
rg=400e-3;                  % Transistor Gate Resistance
(EPC2019)
%rg=300e-3;                 % Transistor Gate Resistance
(EPC8010)
rs=6.2e-3;                  % Transistor Source Resistance
(EPC2019)
%rs=16.9e-3;                % Transistor Source Resistance
(EPC8010)
R_L=10;                     % System Load Resistance
P_out=40;                   % Desired Output Power

switch converterType
    case ('Ideal_EF')
        % Order is [Max harmonic on switch current , Max
THD , ...
        % Min Efficiency , Max ripple factor]
        minMaxValues=[6 , 0.1 , 0.9 , 0.1];
        nvars=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        % LsCs resonant factor , Vin]
        Lb=[0.2 , 10e-9 , 0.5e-12 , 0.5 , 2 , 0.5 , 12];
        Ub=[0.8 , 100e-6 , 5e-9 , 2 , 8 , 2 , 72];
```

```
    case ('Parasitic_EF')
        % Order is [Max harmonic on switch current , Max
THD , ...
        % Min Efficiency , Max ripple factor]
        minMaxValues=[6 , 0.1 , 0.8 , 0.1];
        nvars=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        % LsCs resonant factor , V_in]
        Lb=[0.2 , 10e-9 , 0.5e-12 , 0.5 , 2 , 0.5 , 12];
        Ub=[0.8 , 100e-6 , 5e-9 , 2 , 8 , 2 , 72];
    case ('Practical_EF')
        % Order is [Max harmonic on switch current , Max
THD , ...
        % Min Efficiency , Max ripple factor]
        minMaxValues=[6 , 0.1 , 0.8 , 0.1];
        nvars=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        %LsCs resonant factor , V_in]
        Lb=[0.2 , 10e-9 , 0.5e-12 , 0.5 , 2 , 0.5 , 12];
        Ub=[0.8 , 100e-6 , 5e-9 , 2 , 8 , 2 , 72];
end

%% Optimization Settings
numGen = 100;           % Number of Generations, play with
this value
popSize = 525;          % Population size, play with this
value

output = @(options,state,flag)
myOptOutput(options,state,flag);
options =
optimoptions('gamultiobj','UseVectorized',false,'MaxStallGe
nerations',10,'MaxGenerations',numGen,...
    'FunctionTolerance',1e-4,'PopulationSize',popSize,
'Display', 'iter', 'OutputFcn', output);

switch converterType
    case("EF_Inverter")
        fitnessFunc = @(inputParam) Ideal_EF(f_s, r_on,
r_off, R_L, P_out, minMaxValues, inputParam);
        tic;
```

```
        fitnessFunc([0.4 , 5e-8 , 250e-12 , 1 , 5 , 1 ,
36]);
        funcRuntime = toc();
    case("Parasitic_EF")
        fitnessFunc = @(inputParam) Parasitic_EF(f_s, r_on,
r_off, rg, rs, R_L, P_out, minMaxValues, inputParam);
        tic;
        fitnessFunc([0.4 , 5e-8 , 250e-12 , 1 , 5 , 1 ,
36]);
        funcRuntime = toc();
    case("Practical_EF")
        fitnessFunc = @(inputParam) Practical_EF(f_s, r_on,
r_off, rg, rs, R_L, P_out, minMaxValues, inputParam);
        tic;
        fitnessFunc([0.4 , 5e-8 , 250e-12 , 1 , 5 , 1 ,
36]);
        funcRuntime = toc();
end

seconds = funcRuntime*(popSize)*(numGen+1);
hours = floor(seconds/3600);
minutes = round((seconds - hours*3600)/60);

%% Optimizer
tic;
[x,fval,StopFlag,output,Population,Score] =
gamultiobj(fitnessFunc,nvars,[],[],[],[],Lb,Ub,[],options);
toc;

%% Output Function
function [state,options,optchanged] =
myOptOutput(options,state,flag)

averageScores=mean(state.Score,2);
bestScores = min(averageScores);
positionBest=find(averageScores==bestScores);
lengthBest=length(positionBest);
Check=zeros(1,6)+1;

if lengthBest<1
    winner_Values=state.Population(positionBest,:);
    winner_Scores=state.Score(positionBest,:);
else
    for l=1:1:lengthBest
```

```
        maybeWinner=state.Score(positionBest(1),:);
        if maybeWinner(1)<=Check(1)
            Check=maybeWinner;

winner_Values=state.Population(positionBest(1),:);
            winner_Scores=state.Score(positionBest(1),:);
        else
        end
    end
end

switch flag
    case 'init'
        disp('Starting the algorithm');
        fprintf('Best component values: %d.\n',
winner_Values)
        fprintf('Best scores: %d.\n', winner_Scores)
    case {'iter','interrupt'}
        disp('Iterating...')
        fprintf('Best component values: %d.\n',
winner_Values)
        fprintf('Best scores: %d.\n', winner_Scores)
    case 'done'
        disp('Performing final task');
end

optchanged = false;
end
```

Appendix E

MOPSO Initialization Function

```
%% Reset
clear variables
clc

%% Optimization Setup
%converterType="Ideal_EF";            % Select Converter
type
%converterType="Parasitic_EF";
converterType="Practical_EF";

f_s=13.56e6;                % Desired Switching Frequency
r_on=10e-3;                 % Transistor On Resistance (Ideal)
%r_on=36e-3;                % Transistor On Resistance
(EPC2019)
%r_on=120e-3;               % Transistor On Resistance
(EPC8010)
r_off=1e6;                  % Transistor Off Resistance (Ideal)
rg=400e-3;                  % Transistor Gate Resistance
(EPC2019)
%rg=300e-3;                 % Transistor Gate Resistance
(EPC8010)
rs=6.2e-3;                  % Transistor Source Resistance
(EPC2019)
%rs=16.9e-3;                % Transistor Source Resistance
(EPC8010)
R_L=10;                     % System Load Resistance
P_out=40;                   % Desired Output Power

switch converterType
    case ('Ideal_EF')
        % Order is [Max harmonics on switch current , Max
THD , ...
        % Min Efficiency , Max ripple factor]
        minMaxValues=[6 , 0.1 , 0.9 , 0.1];
        MultiObj.nVar=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        % LsCs resonant factor , V_in]
```

```
        MultiObj.var_min=[0.2 , 10e-9 , 0.25e-12 , 0.5 , 2
, 0.5 , 12];
        MultiObj.var_max=[0.8 , 100e-6 , 2.5e-9 , 2 , 8 , 2
, 72];
    case ('Parasitic_EF')
        % Order is [Max harmonics on switch current , Max
THD , ...
        % Min Efficiency , Max ripple factor]
        minMaxValues=[6 , 0.1 , 0.8 , 0.1];
        MultiObj.nVar=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        % LsCs resonant factor , V_in]
        MultiObj.var_min=[0.2 , 10e-9 , 0.25e-12 , 0.5 , 2
, 0.5 , 12];
        MultiObj.var_max=[0.8 , 100e-6 , 2.5e-9 , 2 , 8 , 2
, 72];
    case ('Practical_EF')
        % Order is [Max harmonics on switch current , Max
THD , ...
        % Min Efficiency , Max rippler factor]
        minMaxValues=[6 , 0.1 , 0.8 , 0.1];
        MultiObj.nVar=7;
        % Order is [D , Lf , Cf , Cf-Cmr factor , Output
quality factor , ...
        % LsCs resonant factor , V_in]
        MultiObj.var_min=[0.2 , 10e-9 , 0.25e-12 , 0.5 , 2
, 0.5 , 12];
        MultiObj.var_max=[0.8 , 100e-6 , 2.5e-9 , 2 , 8 , 2
, 72];
end

% Population size       % play with these values depends on
the problem and
params.Np = 125;        % using 100 for both is a good
general approach
% Repository size
params.Nr = 125;

params.maxgen = 320;    % Maximum number of
generations,increasing this
                        % value increases computational
cost
```

```
params.W = 0.5;          % Initial inertia weight, varies
from 0.5 to 1 over
                         % the course of the optimization

params.C1 = 2;           % Individual confidence factor,
typically 0.2-4 but
                         % use 2 to reduce complexity
params.C2 = 2;           % Swarm confidence factor,
typically 0.2-4 but use 2
                         % to reduce complexity

params.ngrid = 30;       % Number of grids in each
dimension, typically 20-50
params.maxvel = 20;      % Maxmium vel in percentage,
typically 10-100 %
params.u_mut = 0.5;      % Uniform mutation percentage,
typically 0.5

switch converterType
    case("E_Inverter")
        MultiObj.fun = @(x) PSO_Intermediate(f_s, r_on,
r_off, R_L, P_out, minMaxValues, x, params.Np,
converterType);
    case("EF_Inverter")
        MultiObj.fun = @(x) PSO_Intermediate(f_s, r_on,
r_off, rg, rs, R_L, P_out, minMaxValues, x, params.Np,
converterType);
end

%% Optimization
tic;
REP=MOPSO(params,MultiObj);
toc;
```

Appendix F

MOPSO Intermediate Function

```
function Objectives =
PSO_Intermediate(f_s,r_on,r_off,rg,rs,R_L,P_Out,minMaxValue
s,inputParam,numPop,converterType)

Objectives=zeros(numPop,6);

switch converterType
    case('Ideal_EF')
        fitnessfunc = @(inputParam)
EF_Ideal_Model(f_s,r_on,r_off,R_L,P_Out,minMaxValues,inputP
aram);
        for k=1:1:numPop
            Objectives(k,:)=fitnessfunc(inputParam(k,:));
        end
    case('Parasitic_EF')
        fitnessfunc = @(inputParam)
EF_Parasitic_Model(f_s,r_on,r_off,rg,rs,R_L,P_Out,minMaxVal
ues,inputParam);
        for k=1:1:numPop
            Objectives(k,:)=fitnessfunc(inputParam(k,:));
        end
    case('Practical_EF')
        fitnessfunc = @(inputParam)
EF_Practical_Model(f_s,r_on,r_off,rg,rs,R_L,P_Out,minMaxVal
ues,inputParam);
        for k=1:1:numPop
            Objectives(k,:)=fitnessfunc(inputParam(k,:));
        end
end
```