ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COST OPTIMIZATION OF FLYBACK CONVERTER USING GENETIC ALGORITHM

M.Sc. THESIS

Giray BALCI

Department of Electrical Engineering

Electrical Engineering Programme

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

COST OPTIMIZATION OF FLYBACK CONVERTER USING GENETIC ALGORITHM

M.Sc. THESIS

Giray BALCI
(504161023)

Department of Electrical Engineering

Electrical Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Murat YILMAZ

JUNE 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FLYBACK ÇEVİRİCİ TASARIMINDA GENETİK ALGORİTMA KULLANARAK MALİYET OPTİMİZASYONU

YÜKSEK LİSANS TEZİ

Giray BALCI
(504161023)

Elektrik Mühendisliği Anabilim Dalı

Elektrik Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Murat YILMAZ

HAZİRAN 2019

Giray BALCI, a M.Sc. student of ITU Graduate School of Science Engineering and Technology 504161023 successfully defended the thesis entitled "COST OPTIMIZATION OF FLYBACK CONVERTER USING GENETIC ALGORITHM", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Murat YILMAZ** Istanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Özgür ÜSTÜN** Istanbul Technical University	.....
	Asst. Prof. Dr. Salih Barış ÖZTÜRK Istanbul Okan University	.....

Date of Submission : **03 May 2019**
Date of Defense : **17 June 2019**

To my family,

FOREWORD

"A person who never made a mistake, never tried anything new."

-Albert Einstein

I would like to thank to;

My master thesis advisor Asst. Prof. Dr. Murat Yılmaz for guiding me through this thesis, pushing my limits and motivating me with his energy,

Arçelik Company for helping me to find a challenging problem as a master thesis,

My friends Mustafa Said Uçar, Berkay Genç and Alp Atakan İşbakan for their close friendship and their support for the entire time,

and last but not least; MY FAMILY, for their support, not only for this thesis but throughout my life. Without them, I could have never become the person who I am now.

JUNE 2019

Giray BALCI
(Electrical Engineer)

TABLE OF CONTENTS

Page

FOREWORD ..... **ix**
TABLE OF CONTENTS ..... **xi**
ABBREVIATIONS ..... **xv**
SYMBOLS ..... **xvii**
LIST OF TABLES ..... **xxi**
LIST OF FIGURES ..... **xxiii**
LIST OF CODE SNIPPETS ..... **xxv**
SUMMARY ..... **xxvii**
ÖZET ..... **xxix**
1. INTRODUCTION **1**
1.1 Motivation and Objective 1
1.2 Literature Review Of Optimization Algorithms 2
1.3 Optimization Algorithms 5
1.3.1 Gradient based algorithms 6
1.3.1.1 Gradient descent method 6
1.3.1.2 Newton method 7
1.3.1.3 Non-linear conjugate gradient method 7
1.3.2 Gradient free (nature inspired) algorithms 7
1.3.2.1 Particle swarm intelligence 7
1.3.2.2 Ant colony optimization 7
1.3.2.3 Nelder mead method 8
1.3.2.4 Exhaustive search method 8
1.3.2.5 Simulated annealing method 8
1.3.2.6 Genetic algorithm 8
2. GENETIC ALGORITHMS **9**
2.1 Operators 11
2.1.1 Crossover operator 11
2.1.2 Mutation operator 13
2.1.2.1 Flipping 13
2.1.2.2 Interchanging 13
2.1.3 Genotype-phenotype mapping 14
2.1.4 Fitness score calculations 14
2.1.5 Selection operators 14
2.1.5.1 Elitist selection 15
2.1.5.2 Comma selection 15
2.1.5.3 Plus selection 15
2.1.5.4 Roulette wheel selection 15

2.1.5.5 Tournament selection 15
2.1.5.6 Rank selection 16
2.1.5.7 Truncation selection 16
2.1.6 Termination 16
2.1.6.1 Maximum generations 16
2.1.6.2 Elapsed time 17
2.1.6.3 Stall generations 17
2.1.6.4 Stall time limit 17
3. SINGLE OUTPUT FLYBACK CONVERTER DESIGN 19
3.1 CCM and DCM Operation 23
3.2 Equations for Flyback Converter 24
3.3 RCD Snubber Design 29
3.4 Magnetic Component Design 29
3.5 Power Loss Calculations 33
3.5.1 MOSFET loss 33
3.5.2 Secondary rectifier diode power loss 37
3.5.3 Transformer power loss 38
3.5.4 Snubber power loss 41
3.5.5 Rectifier diode power loss 41
3.5.6 Electrolytic capacitor power loss 42
4. OPTIMIZATION OF FLYBACK CONVERTER USING GA 45
4.1 MATLAB GA Toolbox 45
4.2 Optimization Problem 46
4.3 Component Database 50
4.4 Initialization 54
4.4.1 Library parameter initialization 55
4.4.2 User specific value initialization 56
4.4.3 Component library initialization 56
4.4.4 Genetic algorithm initialization 56
4.5 Fitness (Cost) Function 58
4.6 Penalty (Constraints) Function 58
4.6.1 Voltage constraints 61
4.6.2 Current constraints 62
4.6.3 Power loss constraints 62
4.6.4 Junction temperature constraints 63
4.6.5 Other constraints 63
4.7 Output Function 63
4.8 Plot Function 64
5. REALIZATION OF CONVERTER 71
5.1 Selected Components by Genetic Algorithm 71
5.2 Simulation With the Selected Components 73
5.3 Schematic and PCB Design 77
5.4 Test Equipment 78
5.5 Test Results 78
5.5.1 Efficiency measurements 79

5.5.2 Oscilloscope waveforms 81
6. CONCLUSIONS AND RECOMMENDATIONS 89
6.1 Practical Application of This Study and Future Work 90
REFERENCES 93
APPENDICES 97
APPENDIX A MATLAB CODES 99
APPENDIX A.1 Component Class Definition Codes 99
APPENDIX A.2 Component Library Importing Codes 104

ABBREVIATIONS

AWG	**:** American Wire Gauge
EMC	**:** Electromagnetic Compatibility
ESR	**:** Equivalent series resistance
GA	**:** Genetic Algorithm
IC	**:** Integrated Circuit
LEB	**:** Leading edge blanking
MLT	**:** Mean length per turn
MMF	**:** Magnetomotive force
MOSFET	**:** Metal-oxide semiconductor field effect transistor
PCB	**:** Printed circuit board
RF	**:** Resistance factor
PWM	**:** Pulse width modulation
SMPS	**:** Switched mode power supply
SUS	**:** Stochastic universal sampling
THD	**:** Total harmonic distortion

SYMBOLS

A_e : Core cross sectional area in m^2
a_L : Inductance in nH per $turns^2$
b_w : Coil former's height
B_{sat} : Saturation flux density of core
C_{DCBus} : DC bus capacitor's capacitance value
C_{out} : Output capacitor's capacitance value
C_{sn} : Snubber capacitor's capacitance value
C_{ds} : Drain-source capacitance of MOSFET
C_{gd} : Gate-drain capacitance of MOSFET
C_{gs} : Gate-source capacitance of MOSFET
$C_{oss_{application}}$: Output capacitance at application voltage
$C_{oss_{datasheet}}$: Output capacitance at datasheet measurement voltage
C_{iss} : Input capacitance of MOSFET
C_{oss} : Output capacitance of MOSFET
C_{rss} : Reverse transfer capacitance of MOSFET
δ : Skin depth of conductor
Δ : Penetration ratio
Δ' : Modified penetration ratio
D : Duty cycle
D_{ch} : DC bus capacitor charge duty
D_{max} : Maximum duty cycle
d : Diameter of the round wire
d_w : Equivalent thickness to foil conductor
η_w : Porosity factor
$f(i)$: Fitness function
F : Fringing flux factor
f_{line} : Line frequency
G : E core's winding area
h_i : Insulation thickness between primary secondary layer
h_{ip} : Insulation thickness between primary conductors
h_{is} : Insulation thickness between primary conductors
$I_d(t)$: Diode average forward current
$I_{d_{avg}}$: Diode average forward current
$I_{d_{RMS}}$: Diode RMS forward current
$I_{ds_{RMS}}$: MOSFET drain source RMS current
$I_{ds_{start}}$: MOSFET drain source current at the start of period
$I_{ds_{end}}$: MOSFET drain source current at the end of period
I_{f1} : RMS current at f1 frequency
$I_{(n)_{AC}}$: (n) winding AC current. n: primary or secondary
$I_{(n)_{RMS}}$: (n) winding RMS current. n:primary or secondary

$I_{sec_{RMS}}$: Secondary side lumped RMS current
I_{EDC} : Equivalent DC current level
ΔI : Ripple current
$I_{ds_{peak}}$: MOSFET drain source peak current
I_{out} : Output DC current
$I_{sec_{cap}}$: Output capacitor ripple current
K_{RF} : Ripple factor of the primary current
k_u : Utilization factor of winding
l_{gap} : Airgap length
L_{leak} : Leakage inductance of transformer
L_m : Magnetizing inductance
m : Number of layers in winding
n : Turns ratio of transformer
η : Efficiency
MLT_{prim} : Primary winding mean length per turn
n_{lp} : Number of primary winding layer
n_{ls} : Number of secondary winding layer
N' : Number of turns in one layer of winding
N_{prim} : Primary turns
$N_{prim_{min}}$: Minimum primary turns
N_{sec} : Secondary turns
Ω : Ohm
$p(i)$: Probability function
p : Population size
P_{core} : Power loss of magnetic core
P_{w/m^3} : Power loss of magnetic core for unit volume
P_d : Total power loss of diode
P_{d_c} : Conduction power loss of diode
P_{d_c} : Switching power loss of diode
P_{elcap} : Power loss of electrolytic capacitor
P_{in}, P_{out} : Input, output power
P_{mos_b} : Blocking power loss of MOSFET
P_{mos_c} : Conduction power loss of MOSFET
P_{mos_g} : Gating power loss of MOSFET
$P_{mos_{sw}}$: Switching power loss of MOSFET
$P_{mos_{sw(Coss)}}$: Switching power loss of MOSFET due to C_{oss}
$P_{mos_{sw(overlap)}}$: Switching power loss of MOSFET due to overlapping waveform
$P_{mos_{swON}}$: Switching power loss of MOSFET while turn on
$P_{mos_{swOFF}}$: Switching power loss of MOSFET while turn off
P_{sn} : Power loss of snubber resistor
$P_{w(n)_{DC}}$: (n) winding DC power loss. n: primary or secondary
$P_{w(n)_{AC}}$: (n) winding AC power loss. n: primary or secondary
Q_g : Total gate charge of MOSFET
ρ : Specific resistivity of conductor
R_{AC} : AC resistance of winding
R_{DC} : DC resistance of winding
R_d : Diode resistance
$R_{ds_{on}}$: MOSFET conduction resistance

R_{ESR} : Output capacitor's ESR value
$R_{(n)_{DC}}$: (n) winding DC resistance. n: primary or secondary
$R_{th_{JA}}$: Junction to ambient thermal resistance
R_{sn} : Snubber resistor's resistance value
μ_0 : Permeability of free space
μ_r : Relative permeability of free material
t_{on} : MOSFET turn on time
t_{off} : MOSFET turn off time
T_{sw} : Switching period
$tan\delta$: Dissipation factor of electrolytic capacitor
$V_{DC_{CCM}}$: Minimum DC voltage for CCM operation
$V_{DC_{max}}$: Maximum DC bus voltage
$V_{DC_{min}}$: Minimum DC bus voltage
V_{ds} : MOSFET drain source voltage
$V_{ds_{nom}}$: Nominal MOSFET drain source voltage
$V_{ds_{max}}$: Maximum MOSFET drain source voltage
$V_{d_{fwd}}$: Diode forward voltage drop
$V_{d_{rvs}}$: Diode reverse voltage
$V_{line_{max}}$: RMS value of maximum line voltage
$V_{line_{min}}$: Minimum line voltage RMS value
V_{out} : Output voltage
ΔV_{out} : Output ripple voltage
V_{RO} : Reflected output voltage
V_{sn} : Snubber capacitor voltage
ΔV_{sn} : Snubber capacitor ripple voltage
$W_{A_{prim}}$: Primary winding area
$W_{A_{sec}}$: Secondary winding area
$W_{A_{total}}$: Total required winding area
$V_{ds_{application}}$: Drain source voltage at application
$V_{ds_{datasheet}}$: Drain source voltage at datasheet measurement
V_{core} : Volume of transformer magnetic core
X_c : Capacitive reactance
ζ_1' : Skin effect factor
ζ_2' : Proximity effect factor

LIST OF TABLES

Page

Table 3.1 : Comparison between CCM and DCM operation of flyback converter. 24
Table 3.2 : 24 to 32 AWG table 32
Table 4.1 : Digikey price breakdown example. 48
Table 4.2 : Database example for diode. 51
Table 4.3 : Database example for electrolytic capacitor. 51
Table 4.4 : Database example for ceramic capacitor. 51
Table 4.5 : Database example for MOSFET. 52
Table 4.6 : Database example for resistor. 52
Table 4.7 : Database example for transformer core. 52
Table 4.8 : Database example for wire. 52
Table 4.9 : Database example for controller. 53
Table 4.10 : Database example for shunt resistor. 53
Table 4.11 : Database example for TL431. 53
Table 4.12 : Database example for optocoupler. 53
Table 4.13 : Number of each component type in database. 54
Table 4.14 : Example of parameter breakdown of an individual. 59
Table 5.1 : Specifications of the designed flyback converter. 71
Table 5.2 : Selected components by genetic algorithm. 72
Table 5.3 : Laboratory equipment used for measurements. 78
Table 5.4 : Transformer calculations and measurements. 80

LIST OF FIGURES

Page

Figure 1.1 : Number of papers using aptimization algorithms in power converters [3]. 3
Figure 1.2 : Optimization topics in power converters [3]. 3
Figure 1.3 : Classification of optimization algorithms. 5
Figure 2.1 : Genetic algorithm terminology. 9
Figure 2.2 : GA algorithm flowchart. 10
Figure 2.3 : Crossover stage. 11
Figure 2.4 : Single point crossover. 12
Figure 2.5 : Multiple point crossover. 12
Figure 2.6 : Uniform crossover. 12
Figure 2.7 : Arithmetic crossover. 12
Figure 2.8 : Mutation: Flipping. 13
Figure 2.9 : Mutation: Interchanging. 13
Figure 3.1 : Flyback converter 19
Figure 3.2 : Flyback converter with important parasitics. 20
Figure 3.3 : Flyback converter MOSFET on-time with parasitics. 21
Figure 3.4 : Flyback converter MOSFET off-time - with parasitics. 22
Figure 3.5 : DC bus capacitor charge duty [23]. 25
Figure 3.6 : Voltage levels on the Q1 switch [23]. 26
Figure 3.7 : Ripple factor of the primary current [23]. 26
Figure 3.8 : Flyback transformer winding schema. 30
Figure 3.9 : Usable winding area 31
Figure 3.10 : MMF distrubution on the flyback transformer [27] 32
Figure 3.11 : Fringing flux core parameters 33
Figure 3.12 : Fringing flux parameters [28] 33
Figure 3.13 : MOSFET (A)Turn-on transition (B)Turn-off transition. 35
Figure 3.14 : Parasitic MOSFET capacitances (Infineon - IPN80R2K0P7). 36
Figure 3.15 : Current distribution in a wire due to skin effect. 39
Figure 3.16 : Current distribution in a transformer due to proximity effect [33]. 40
Figure 3.17 : Electrolytic capacitor model with parasitics. 42
Figure 3.18 : Electrolytic capacitor ESR change with temperature and frequency [35]. 43
Figure 4.1 : MATLAB optimization toolbox. 46
Figure 4.2 : Plot Function - evolution of best score. 64
Figure 4.3 : Plot Function - average distance between individuals. 65
Figure 4.4 : Plot Function - best, worst and mean scores in population. 65
Figure 4.5 : Plot Function - tracking of the stopping criteria. 66

Figure 4.6 : Plot Function - score of each individual in population. 66
Figure 4.7 : Prices of the components in GA. 67
Figure 4.8 : 2D plot of converter and component costs. 68
Figure 4.9 : 3D plot of converter and component costs. 69
Figure 5.1 : LTspice model of the flyback converter power section. 73
Figure 5.2 : LTspice model of the flyback converter control section. 73
Figure 5.3 : DC bus ripple voltage waveform. 74
Figure 5.4 : DC bus capacitor ripple current waveform. 74
Figure 5.5 : DC bus capacitor ripple current high frequency component waveform. 74
Figure 5.6 : MOSFET drain-source voltage waveform. 75
Figure 5.7 : Snubber capacitor voltage waveform. 75
Figure 5.8 : Shunt resistor current waveform. 76
Figure 5.9 : Secondary diode voltage waveform. 76
Figure 5.10 : Output voltage waveform. 76
Figure 5.11 : Output capacitor ripple current waveform. 77
Figure 5.12 : Altium Designer power section schematic design. 78
Figure 5.13 : Altium Designer control section schematic design. 78
Figure 5.14 : Altium Designer PCB design. 79
Figure 5.15 : Realized converter BOTTOM and TOP side. 79
Figure 5.16 : Working environment. 80
Figure 5.17 : Output power vs efficiency plots with different input voltages. 80
Figure 5.18 : Input voltage vs efficiency plots. 81
Figure 5.19 : 230 V RMS - DC bus voltage. 82
Figure 5.20 : 230 V RMS - MOSFET drain source voltage. 82
Figure 5.21 : 230 V RMS - Snubber voltage. 83
Figure 5.22 : 230 V RMS - Secondary side rectifier diode voltage. 83
Figure 5.23 : Shunt resistor voltage. 84
Figure 5.24 : Output voltage waveform. 84
Figure 5.25 : 265 V RMS - MOSFET drain source voltage. 85
Figure 5.26 : 85 V RMS - DC bus voltage. 85
Figure 5.27 : 265 V RMS - Secondary side rectifier diode voltage. 86
Figure 5.28 : Start-up sequence of the converter. 86
Figure 5.29 : 230 V RMS - No load, MOSFET drain source voltage. 86
Figure 5.30 : Transformer saturation current. 87
Figure 6.1 : Histogram of prices for 200 converter. 90

LIST OF CODE SNIPPETS

Page

Code Snippet 4.1 : Example of class definition for diode. 55
Code Snippet 4.2 : Commercial component library parameter initialization. 55
Code Snippet 4.3 : User ranged library parameter initialization. 56
Code Snippet 4.4 : Flyback converter design specification variables. . . . 56
Code Snippet 4.5 : Genetic algorithm options. 57
Code Snippet 4.6 : Genetic algorithm initialization. 58
Code Snippet 4.7 : Fitness (cost) function. 58
Code Snippet 4.8 : Price mapping function. 59
Code Snippet 4.9 : Penalty (constraint) function. 60
Code Snippet 4.10: Voltage constraints. 62
Code Snippet 4.11: Current constraints. 62
Code Snippet 4.12: Power loss constraints. 63
Code Snippet 4.13: Junction temperature constraints. 63
Code Snippet 4.14: Other constraints. 63
Code Snippet 4.15: Penalty (constraint) function. 64
Code Snippet 5.1 : Designed converter specifications. 71

COST OPTIMIZATION OF FLYBACK CONVERTER USING GENETIC ALGORITHM

SUMMARY

In this master thesis, a genetic algorithm implementation for flyback converter cost optimization is given. Using proposed genetic algorithm in MATLAB, a converter is designed. All of the power stage components are selected by genetic algorithm. Using these selected components simulation is done using LTspice simulation program. After that, using calculated components schematic and PCB drawing is made in Altium Designer program. PCB is manufactured and realized converter is tested. Efficiency measurements and oscilloscope waveforms are within design specifications. Calculated, simulated and realized values are in good accordance hence design is validated.

By this master thesis, a component database is created which contains 317 discrete components ranges from electrolytic capacitors to MOSFETs to resistors to controller ICs. A systematic way of importing these components with their important parameters into MATLAB is implemented. Flyback converter equations are implemented in MATLAB for using in genetic algorithm runs.

By using genetic algorithm, designed converter cost is 1.414 $. Designed converter is compared with the formerly designed comparable flyback converter. Former converter is proposed by Power Integrations company for a project. Proposed converter cost is 2.212 $. To be able to compare two converters, both converter prices are calculated using components prices taken from Digikey website. A cost reduction of 36% is obtained.

Layout of the thesis is given as following:

In first chapter of the thesis literature review of optimization algorithm and widely used optimization algorithms are given with their mathematical equations and explanations.

In second chapter, one of the widely used optimization algorithms, genetic algorithm is explained. General terminology and genetic algorithm operators are explained thoroughly. Crossover operators, mutation operators, mapping, fitness score calculations, selection operators and termination conditions are given.

In third chapter, a single output flyback converter calculations is given with necessary equations and explanations. When designing, significant parasitics are also considered. Design of a RCD snubber circuitry is also given. Since magnetic design is one of the most important aspects of SMPS design transformer calculations are given extensively. Finally, power loss calculations are given for each type of component.

In fourth chapter, chapter 2 and chapter 3 is combined. Genetic algorithm implementation for flyback converter design is given in this chapter. Component database formation, implemented MATLAB codes are given. Fitness, penalty and plotting functions are explained.

In the fifth chapter a flyback converter is realized using components that are selected by genetic algorithm. LTspice simulation results, schematic, PCB design, oscilloscope waveform and efficiency measurements are given.

In the sixth chapter, conclusions and further recommendations are given for future work.

FLYBACK ÇEVİRİCİ TASARIMINDA GENETİK ALGORİTMA KULLANARAK MALİYET OPTİMİZASYONU

ÖZET

Bu yüksek lisans tezinde bir flyback çevirici için genetik algoritma kullanarak maliyet optimizasyonu yapan bir algoritma geliştirilmiştir. Geliştirilen algoritma MATLAB ortamında kodlanmış ve fonksiyonel bir flyback çevirici tasarımı gerçekleştirilmiştir. Bir flyback çeviricinin bütün güç katı komponentleri genetik algoritma kullanılarak seçilmiştir. MATLAB'deki matematiksel hesaplamaların ardından LTspice benzetim programı ve komponent modelleri kullanılarak devrenin zaman domeninde benzetimi yapılmıştır. Sonrasında genetik algoritma tarafından seçilmiş komponentler kullanılarak şematik ve PCB çizimi yapılmıştır. PCB üretimi ve dizgisi yapılıp gerçeklenen devre test edilmiştir. Devrenin farklı yük ve gerilimlerde verim ölçümleri alınmıştır. Osiloskop yardımı ile dalga şekilleri incelenmiş ve hesaplamalara ve benzetim sonuçlara yakın sonuçlar alınmıştır. Hesaplanan, benzetim sonucu elde edilen ve gerçeklenen devreden alınan ölçüm sonuçları birbiri ile uyum içerisinde olduğu görülmüş ve tasarım doğrulanmıştır.

Bu tez çalışmasının önemli çıkış noktalarından birisi elektronik komponentlerin lineer olmayan fiyatlandırma politikasıdır. Hem üretici hem de tüketici tarafında kullanımı artan komponentlerin birim fiyatları düşmektedir. Bir komponentin nominal teknik değerleri daha yüksek olsa bile fiyatlandırmada daha düşük nominal değerlere sahip bir komponentten daha uygun bit fiyat alabilmektedir. Örneğin 2 A maksimum akım değerine sahip bir MOSFET yüksek adetli tüketimi nedeniyle 1 A'lik bir komponentten daha ucuz olabilmektedir.

Güç elektroniği devreleri doğası gereği yalnızca teknik açıdan dahi bakılsa optimizasyon gerektiren, tasarımı iterasyonlar ile yapılan devrelerdir. Göz önünde bulundurulması gereken birçok teknik parametre içerisine lineer olmayan fiyatlandırma da girdiğinde el ile hesaplamalar çok karmaşık ve zaman alıcı bir hal almaktadır. Ve birçok nokta gözden kaçıp hatalı tasarımlar ile sonuçlanabilmektedir. Bu çok boyutlu optimizasyon problemini klasik optimizasyon yöntemleri ile çözmek ise neredeyse imkansıza yakındır. Zorlamalı arama yöntemleri ise değişken sayısı arttıkça çözüm süreleri karesel bir oranda arttığı için çıkmaza girmektedir. Örneğin bu tez için oluşturulan veritabanındaki komponentleri kullanarak 91×10^{24} farklı flyback çevirici devresi tasarlanabilmektedir. Bu devrelerin büyük bir oranı istenen çalışma bölgesi için uygun değildir; ancak bu devrelerin tek tek ele alınması, kombinasyon çokluğundan dolayı mümkün değildir. Bu çok boyutlu optimizasyon probleminin çözümü için genetik algoritma kullanımı önerilmiştir.

Genetik algoritma bir rastgele arama algoritmasıdır. Yani arama sürecinde belli bir sistematik içerisinde rastgele hareketler yaparak en uygun çözümü bulmaya çalışır. Bu yüzden genetik algoritmanın her çalıştırılması sonucunda farklı bir sonuç çıkarması doğaldır. Sonuçların tekrarlanabilir olması için rastgele sayı üreteçlerine tohum

sayılar verilerek üretilen rastgele sayıların aynı olması sağlanmakta ve sonuçların tekrarlanabilir olası sağlanmaktadır. Bu tip algoritmaların diğer algoritmalar ile değerlendirmesi ise istatistik metodları kullnılarak yapılabilir. Problem boyutuna göre genetik algoritma yüzlerce ya da binlerce kez rastgele sayılar ile tekrar edilerek ulaşılan sonuçların bir dağılımı çıkarılabilir. Genetik algoritma sonuçları olasılık dağılımı şeklinde verilebilir.

Bu tez çalışması ile genetik algoritma hesaplamalarında kullanmak üzere piyasada bulunan 317 ayrık komponentin verileri bir veritabanı şeklinde oluşturulmuştur. Oluşturulan komponent tipleri şu şekildedir:

- Diyot
- Elektrolitik kapasitör
- Yüksek gerilim seramik kapasitör
- Direnç
- Ferit nüve ve bobin
- MOSFET
- Emaye kaplı kablo
- Şönt direnç
- Kontrol entegresi
- TL431 entegresi
- Optokuplör

Bu veritabanını oluştururken her komponentin önemli parazitik bileşenleri ve maksimum değerleri veritabanına eklenmiştir. Ayrıca transformatör sarım sayıları, hava aralığı gibi fiyatı olmayan; ancak dolaylı olarak tasarıma etki eden önemli değişkenlerin de belirli aralıklarda tabloları oluşturulmuş ve optimizasyon probleminde kullanılmıştır. Genetik algoritma ile kullanmak üzere flyback çevirici tasarımı için temel olan 20 adet değişken belirlenmiş, bunların ayrı ayrı tabloları oluşturulmuştur. Bu veritabanı ile MATLAB arasında veri aktarımı için ilgili kodlar yazılmıştır. Ayrıca MATLAB içerisinde bu komponentleri sistematik bir şekilde kontrol edebilmek için her komponent türüne özel *class* tanımlamaları yapılmıştır. Bu sayede ileriye dönük yeni parametreler eklenmesi kolaylaşmıştır.

Genetik algoritma kullanılarak tasarlanan devrenin maliyeti 1.414$ olarak hesaplanmıştır. Bu devre daha önceden Power Integrations tarafından bir proje için tasarlanan flyback çevirici ile kıyaslanmıştır. Daha önceden tasarlanan çeviricinin maliyeti 2.212$ olarak hesaplanmıştır. Her iki devrenin fiyatı da Digikey internet sitesinden alınan fiyatlar yardımıyla hesaplanmıştır. Genetik algoritma ile tasarlanan devre % 36 oranında daha az maliyetli olarak bulunmuştur.

Tez bölümlerinin konu dağılımı ise şu şekildedir: İlk bölümde optimizasyon algoritmaları ile ilgili genel bir literatür araştırması yapılmış ve literatürde çokça kullanılan optimizasyon algoritmalarının genel bir tanıtımı yapılmıştır.

İkinci bölümde genetik algoritma ile ilgili daha ayrıntılı bilgilere yer verilmiştir. Genetik algoritmanın temel çalışma mantığı ve operatörleri anlatılmıştır. Genetik değişim operatörü, mutasyon operatörü, eşleştirme, skor hesaplamaları, seçim operatörleri ve sonlandırma koşulları ile ilgili bilgiler verilmiştir.

Üçüncü bölümde tek çıkışlı bir flyback çevirici tasarımı için gerekli hesaplamalar verilmiştir. Devrenin tasarımında önemli olan parazitik devre elemanlarının

hesaplamaları yapılmıştır. Bir flyback çevirici için önemli bir devre olan RCD baskılayıcı devre tasarımı aktarılmıştır. Ayrıca SMPS devre tasarımı için önemli bir yere sahip olan manyetik tasarım konusu da incelenmiştir. Flyback transformatörü hesaplamaları, kaçak endüktans, saçaklanma akısı, AC direnç vb. bileşenler de göz önünde bulundurularak yapılmıştır. Devrenin verim hesabı ise bütün komponentlerin ayrı ayrı güç kaybı denklemleri verilerek yapılmıştır.

Dördüncü bölüm ikinci ve üçüncü bölümlerin birleştirildiği bölüm olmuştur. MATLAB içerisinde genetik algoritma araç kutusu kullanılarak ikinci ve üçüncü bölümde verilen bilgiler kodlamaya dökülmüştür. Oluşturulan komponent veritabanının içeriğinin açıklamaları da bu bölümdedir. Genetik algoritmayı çalıştırmak için gerekli kod parçaları ayrı ayrı verilmiştir. Skor ve kısıt fonksiyonlarının önemli kısımları yine bu bölümdedir.

Beşinci bölümde ise genetik algoritma tarafından seçilen komponentler ile tasarlanan devrenin gerçekleme aşamasıdır. Öncelikle devre elemanlarının benzetim çalışması LTspice kullanılarak yapılmıştır. Sonrasında şematik ve PCB çalışmaları yapılmış ve devre baskısı yapılmıştır. Gerçeklenen devre üzerinde alınan ölçüm sonuçları bu bölümdedir.

Altıncı bölüm sonuçlar ve gelecek çalışmalara öneriler ve yapılabilecek örnek çalışmaları içermektedir.

1. INTRODUCTION

Power electronics converters include several elements which requires parameter optimization throughout the design process. These elements are not only electrical parameters but also magnetic, thermal, mechanical and economical parameters. Initial design is generally made by simplifications and assumptions to fasten the first calculations. Simplifications and assumptions are solely depending on designer's experience and knowledge on the topic. Further iterations throughout the design process is required for fine tuning the parameters. Number of iterations is related to requirements of the final converter. As requirements are increased, design time and engineering cost is increased.

1.1 Motivation and Objective

The emergence point of this thesis is the non linear pricing for electronic components. There are two factors affecting the price of the components. First one is the usage quantity of the particular component in a company that is buying the component. Second one is the manufacturing quantity of the particular component at the manufacturer. In both cases, with increasing number of quantity, unit price of the component decreases. Sales department of manufacturers and purchasing department of customers desire to maximize trading quantity of a particular component to lower unit price as much as possible. To be able to increase usage quantities, customer company's purchasing department forces design teams to use same components as much as possible. Aside from economical point of view there is a technical point of view for electrical circuit design. In the design of power electronics converter, there are optimization and iteration processes. With the addition of non linear pricing parameter for each component, this multidimensional optimization problem increases its complexity.

For example, an SMPS design may require a 680 μF electrolytic output capacitor. But usage quantity of 1000 μF electrolytic capacitor may be higher than 470 μF

electrolytic capacitor. Hence, unit price of 1000 μF electrolytic capacitor may be lower than a 680 μF electrolytic capacitor. Same scenario can be applied for using 2 parallel 470 μF electrolytic capacitor with lower total price than 680 μF electrolytic capacitor that is required for circuit in hand. But for an electrolytic capacitor there are lots of other parameters other than capacitance value, such as rated voltage and ripple current value. Designer should double check for each parameter of electrolytic capacitor before considering to use a cheaper solution.

Electrolytic capacitor is only a single one aspect of a power electronics converter design. Considering all of the components that are used in the power section of the converter, possibilities for feasible solution and parameters to be double checked by designer increases exponentially. To solve this multidimensional complex optimization problem, genetic algorithm is used in this master thesis. Genetic algorithm is a very useful optimization algorithm for problems where there are numerous possible solution for problem and some of which are more favourable than others.

The final objective of this thesis is to use genetic algorithm to minimize a flyback converter's total cost using a predefined list of components. Also using this approach, is to find a flyback converter that is designed previously and compare it to the new converter that is designed by genetic algorithm.

1.2 Literature Review Of Optimization Algorithms

Optimization techniques for power electronics converters first appeared in 1970's with computer aided heuristic techniques for boost and buck-boost converters [1] [2]. Up to year 2000, there are few works in power electronics converter optimization because of low computational power of computers. With the improvements in computers and processing power, after 2000s, an exponential growth in this research area can be seen in 1.2 [3].

In [4] Balda et al, using optimization tools, analysed a heat sink performance for an electric motor drive, concluding that by increasing fan speed, decreasing ambient temperature or changing mechanical design can increase cooling effect.

In [5] Chung et al, investigated optimization of a buck converter using genetic algorithm by decoupling power components and feedback stage. By using decoupling,

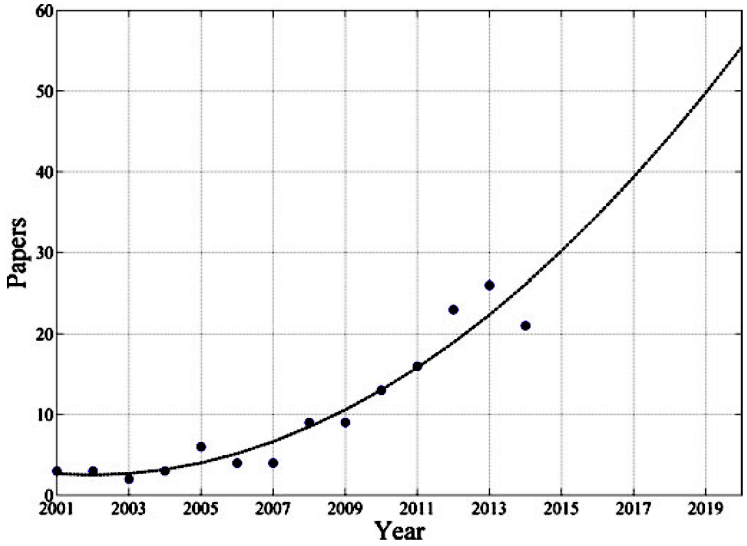

Figure 1.1 : Number of papers using aptimization algorithms in power converters [3].

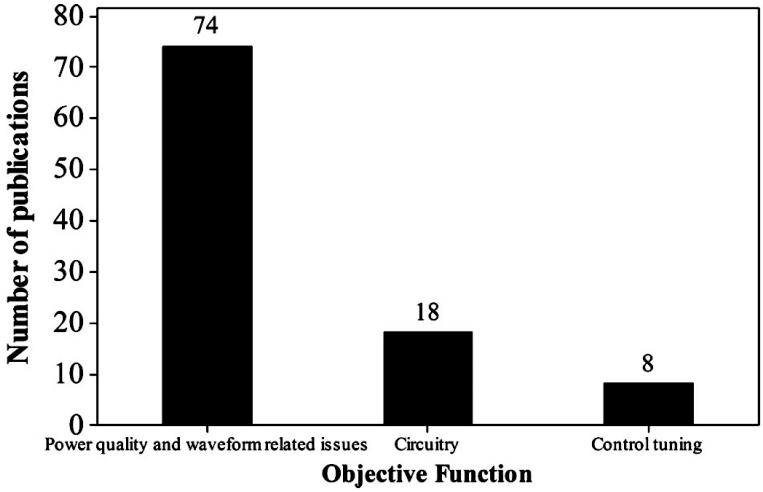

Figure 1.2 : Optimization topics in power converters [3].

loosely bounded power and feedback components are separated from each other, lowering optimization effort.

A work in [6] Ramsden and Muttik, obtained an active filter rating reduction for rectifier loads, but work also shows that is it ineffective for cycloconverter loads.

In [7], Suresh and Kirubakaran used genetic algorithm to select boost converter L and C components and obtain a power loss optimization for boost converter.

In [8] Bergogne et al, using discrete and continuous cost functions in genetic algorithm optimization for buck converter considering electrical and thermal properties is compared. It is shown that using discrete cost functions for discrete components such as MOSFETs and diodes results in better performance compared to continuous

cost function. In another work with the former work, Busquets-Monge et al [9], simulated and realized optimization of a boost PFC converter using GA with discrete cost function considering thermal and EMC related issues. Also 15 % of cost reduction is obtained with GA.

There are also works on optimizing PWM switching for inverter applications to reduce THD and limit harmonics. These works resulted in finding optimum switching points in a period [10] [11].

Sarvi and Salimian compared genetic algorithm (GA) and particle swarm optimization in multilevel converters for harmonic optimization and resulted in genetic algorithm have better THD levels [12].

Piette, Clavel and Marechal's work on minimizing cabling inductance through optimization in power electronics converter and achieved a systematic way of optimization of minimizing cable inductance [13].

There are also works on controller using genetic algorithms. In [14], Kostov and Kyyra applied genetic algorithm to a peak current mode controlled Buck converter. By using genetic algorithm, analog controller resistor and capacitor values are determined. Resulting converter has a better transient response. In [15], a Luo converter controller stage is investigated.

In [16], Yousefi et al, investigated the pole placement of a Cuk converter using genetic algorithm and particle swarm optimization. In both optimization method, dynamic response of the converter have better results. Also particle swarm optimization have better results than genetic algorithm.

Deblecker and Lobry used GA optimization for medium frequency transformer design for SMPS circuits [17].

Usage of genetic algorithm requires decisions about initial population size, fitness function, selection operator, crossover probability, mutation rate and stopping criteria all of which affects the performance of the algorithm. There are also works in these areas [18] - [19].

1.3 Optimization Algorithms

By using optimization tools, a methodological way can be implemented for a given design optimization problem. There are two main optimization algorithm types, namely, gradient(calculus) based and gradient free. Both types of optimization algorithms have advantages and disadvantages for different kinds of problems. Gradient based algorithms require a differentiable function of the problem to be effective and widely used for solving continuous problems. In most cases finite difference gradient calculations are used for obtaining gradient information.

On the other hand, gradient free optimization algorithms, sometimes referred as nature inspired algorithms, do not require a clearly defined function. Gradient free algorithms are stochastic. Gradient based algorithms are better suited for problems that have large number of design variables and require less parameter tuning. Nature inspired algorithms such as genetic algorithm and swarm intelligence show better performance for reaching global optimum. These algorithms are widely used because they can be applied to complex problems and works with parallel processes [3]. In Fig. 1.3, some of the optimization techniques are given in a hierarchical representation.

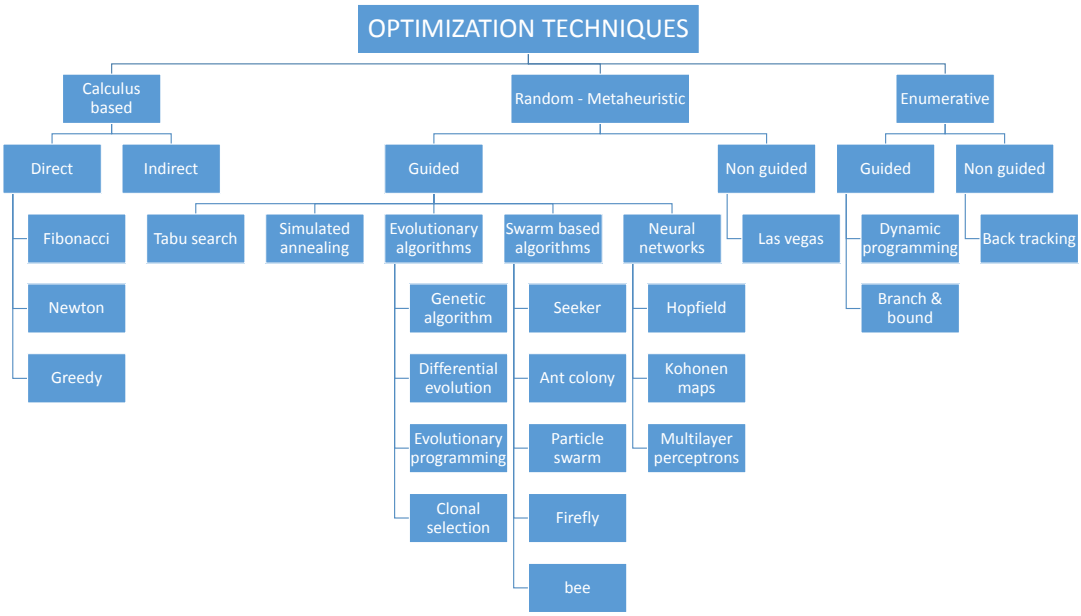

Figure 1.3 : Classification of optimization algorithms.

1.3.1 Gradient based algorithms

Gradient based algorithms use derivatives of the functions to find the optimum value for a given problem. These algorithms are also called deterministic algorithms since they result in same solution in each run. If function is one dimensional algorithm uses slope of the function; if function is multidimensional, algorithm uses gradient of the function. Mainly there are three steps for gradient based algorithms, namely,

1. Search direction
2. Step size
3. Convergence check

Search direction is decided by the derivative of the function. Step size is proportional to the distance to the minimum point of the function and generally determined by the steepness of the slope or gradient. Convergence check is the control mechanism for algorithm to check whether it is reached to a minimum point or not.

Gradient based algorithms are widely used for optimization purposes. Gradient based algorithms are fast in performance and can be scaled to large problems. But there are drawbacks using these algorithms. Major drawback is the requirement of a smooth gradient function or differentiability. Because of this reason, non-linear or partial functions are difficult to optimize with gradient based algorithms. Also these algorithms may stuck in the local minima if the function has many local minimum points. There are numerous gradient based algorithms in the literature but three of the algorithms will be mentioned in this thesis.

1.3.1.1 Gradient descent method

Gradient descent uses first order derivative to find the global minimum of a function. Differentiability is the key point for this algorithm to work properly. If step directions are inverted, one can reach the function's maximum and this is called gradient ascent method.

1.3.1.2 Newton method

Newton method resembles to gradient descent method; but uses second order derivative to find minimum of a function. Function is converted to the second order Taylor series and then tries to find a minimum or maximum point where the derivative of the function is equal to zero.

1.3.1.3 Non-linear conjugate gradient method

Non-linear conjugate gradient method is used for quadratic functions such as $f(x) = |Ax - b|^2$. When the gradient is equal to zero, minimum of the function is obtained. If the function is roughly quadratic near the minimum, this method works well.

1.3.2 Gradient free (nature inspired) algorithms

Gradient free algorithms are the optimization algorithms that do not require a differentiable function to work. Generally used when the function of the problem is difficult to obtain or can not be obtained. Discrete, discontinuous or noisy functions can be optimized with these algorithms. Gradient free algorithms are called nature inspired algorithms because many of the techniques are adapted from nature. These algorithms are also referred as stochastic or heuristic because, each run of algorithm may result in different solution.

1.3.2.1 Particle swarm intelligence

Swarm intelligence algorithm mimics the bird and bee flock movements in the space. Each individual orient itself to the neighbouring individuals. In each iteration algorithm creates a swarm. Each particle has a direction and speed. Direction is determined by particle's current direction, particle's all-time best and swarm's all-time best. Randomness is also key in this algorithm. Swarm intelligence is useful for exploratory problems.

1.3.2.2 Ant colony optimization

In nature, ants search food by moving between nest and food source that are at random locations. Each ant leaves pheromones in their trails. Increasing density of pheromones means that higher probability of food source. Ant colony optimization algorithm

mimics the food search behaviour of the ants. Ant colony algorithm is useful for combinatorial optimization problems.

1.3.2.3 Nelder mead method

Nelder mead method is another technique for finding minimum or maximum of a problem. Method consists of $n+1$ vertices, called simplex, for n dimensional space. For example, for a two-dimensional space, three vertices form a triangular shape. In each iteration, algorithm deletes the worst valued vertex and replaces with another one.

1.3.2.4 Exhaustive search method

Exhaustive search algorithm is sometimes referred as brute force method which tries every possible solution in the search space. For an exhaustive search algorithm there are two basic requirements. First one is that it should search every possible solution, second is it should guarantee that repetitive searches should not occur. As name suggest, this method is computationally exhaustive for big solution spaces; but it is not stochastic as other gradient free algorithms.

1.3.2.5 Simulated annealing method

Simulated annealing algorithm imitates the annealing metal process. In metallurgy, annealing is used for decreasing ductility of metals and give robustness to the metal. First, metal's temperature is increased above critical temperature (transformation temperature) and then slowly cooled down. To simulate this behaviour, a temperature variable is included in the algorithm. When temperature is high, algorithm accepts solutions that are worse than current solutions. As temperature is lowered acceptance rate is decreased. At first, high acceptance provides variety, but decreasing temperature provides convergence to a feasible solution.

1.3.2.6 Genetic algorithm

Genetic algorithm is originally come up with Charles Darwin an English biologist. Later, in the computer science, it is used for solving complex problems that are otherwise nearly impossible to solve. Algorithm basically depends on the principle of the selection of the fittest individuals and transferring its genes to the next generation. In the next chapter, details of the genetic algorithm and its operators will be covered.

2. GENETIC ALGORITHMS

Genetic algorithms (GA) are heuristic searching methods to the various kinds of optimization problems. To better understand GA, terminology used in GA is explained as following:

Each individual solution to the problem is called chromosome or genotype. Chromosomes consists of several genes. Population is the batch of individual solutions that are present in the current generation. Population size is the number of individuals in a population. In each generation a new set of individuals, namely a new population, is generated. With the increasing generation numbers, population supposedly reach a global maximum or global minimum, depending on the problem type. Fitness score is the each individual's performance to the given problem. Higher fitness score means better solution for the problem. Solution space is the all combinational possible solutions that can be applicable to the problem. The main reason for the selection of GA to the optimization problems is the immensity of the problem's solution space. When there are tens of thousands of solutions to the problem and some of which are the optimum results, GA is a useful tool. By using GA and its operators, only a small portion of the solution space is evaluated and yields in best or near-best solutions.

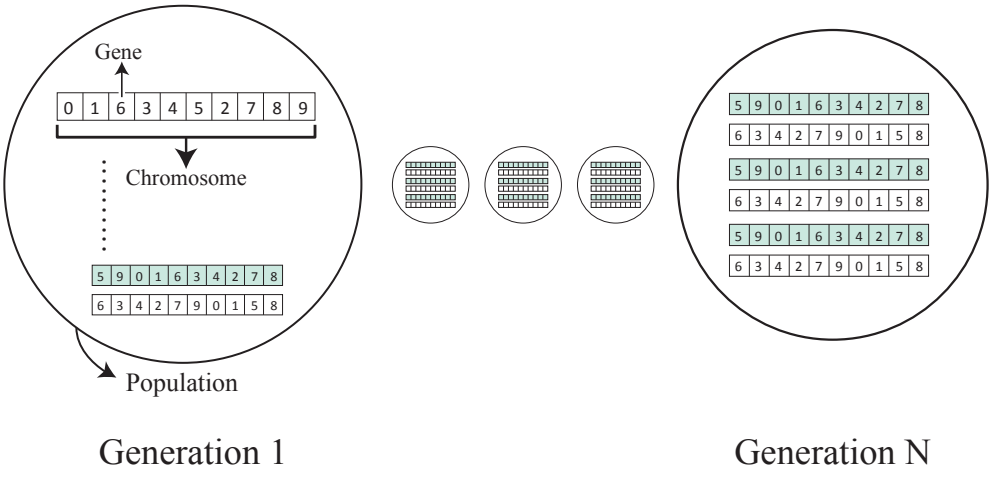

Figure 2.1 : Genetic algorithm terminology.

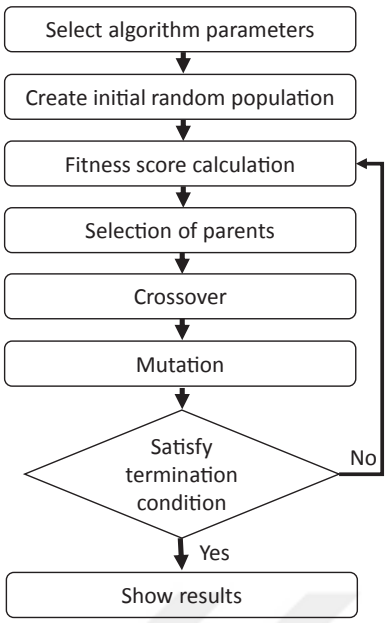

Figure 2.2 : GA algorithm flowchart.

In the application of genetic algorithm, there are specific steps which must be followed. GA algorithm steps are given as following:

First step in the GA application is the selection of exogenous parameters. Exogenous parameters are the global parameters that are related with algorithm such as population size and mutation rate.

Second step is the random generation of first population according to the exogenous parameters. First and second steps are executed only once.

Third step is the fitness computation. In this step success of each individual for the given problem is calculated using fitness function.

Then, phenotype mapping is done if required. Phenotype mapping is used for encoding problem variables into chromosomes. Encoding list of available design parameters into binary coding is a phenotype mapping example.

Forth step is the selection of parents that will form the next generation. There are numerous ways to select parents but in its basic form, individuals that are having best fitness score have the highest probability for the selection. After fifth step, third step and forth step is repeated in a loop until a termination condition is satisfied.

Fifth step is the crossover between selected parents. Crossover is one of the key steps for variety in the population.

Sixth step is mutation. Mutation is also critical for generation of genetic variety. Genes that are not present in the first population can be obtained by this operator.

Final step is the termination. Third, forth, fifth and sixth steps are repeated in a loop for each generation, for every individual. After some time a predetermined termination condition is reached such as time limit or loop limit. In its simplest form, genetic algorithm flowchart is given in Fig. 2.2.

2.1 Operators

Genetic algorithm operators are the tools that are used when constructing and evaluating the genetic algorithm problem. General functions of the operators are briefly explained above. In this section operators will be covered thoroughly.

There are 6 operators that are used in genetic algorithm:

1. Crossover operator
2. Mutation operator
3. Genotype - Phenotype mapping
4. Fitness score calculations
5. Selection operators
6. Termination

2.1.1 Crossover operator

Crossover operator, along with mutation operator, provides genetic diversity in the population by swapping random chunk of genes between parents during crossover stage. Generally, there are two parents in the crossover stage; but there may be three or more parents. Fig. 2.3 shows the crossover stage with two parent chromosomes. In this stage, at random location, random sized chunk of genes are selected from first parent and then swapped with the genes of second parent at the same location.

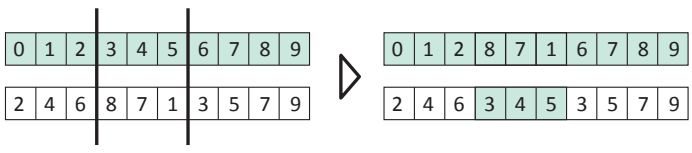

Figure 2.3 : Crossover stage.

Crossover is called single point crossover if two parent breaks the chromosome and swap genes at only one point. Fig. 2.4 shows the single point crossover stage.

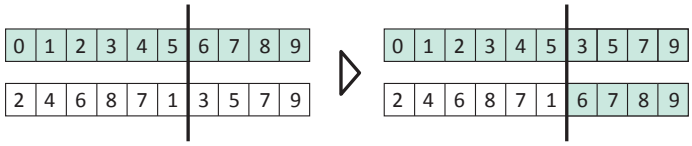

Figure 2.4 : Single point crossover.

Crossover is called multi point crossover if two parent breaks the chromosome and swap genes at two or more points. Fig. 2.5 shows the multiple point crossover stage.

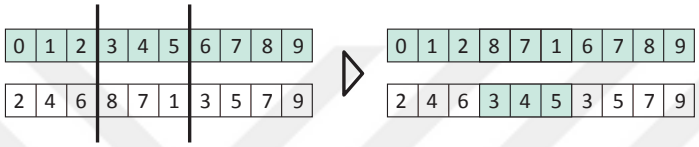

Figure 2.5 : Multiple point crossover.

Crossover is called uniform crossover if swapping action for each gene has probability of 0.5. Fig. 2.6 shows the uniform crossover stage.

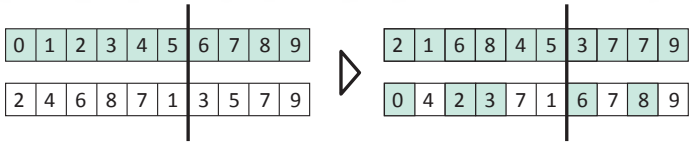

Figure 2.6 : Uniform crossover.

Crossover is called arithmetic crossover if swapping action for each gene is calculated with weighted average from each parents. Fig. 2.7 shows the arithmetic crossover stage.

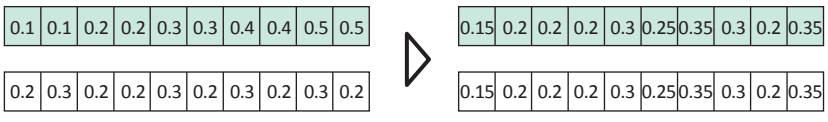

Figure 2.7 : Arithmetic crossover.

2.1.2 Mutation operator

Mutation operator creates random changes in the chromosomes and provides genetic diversity in the population. Initial population is generally do not contain every possible gene in its chromosomes. This operator provides chance to reach every part of the solution space throughout the generations. Mutation operator also prevents the algorithm to stuck in a local minima. It is important to use mutation operator without any biasing. Biasing creates an unwanted orientation for algorithm.

2.1.2.1 Flipping

Flipping is randomly changing the genes of a chromosome based on a mutation chromosome. If a chromosome coding is binary, parent chromosome includes only zeros and ones as genes. Flipping mechanism changes ones to zeros and vice versa. If chromosome coding is not binary, as shown in Fig. 2.8, mutation changes gene into another gene in available solution space. In Fig. 2.8, mutation chromosome is generated randomly, only first, third and eighth genes are selected for mutation.

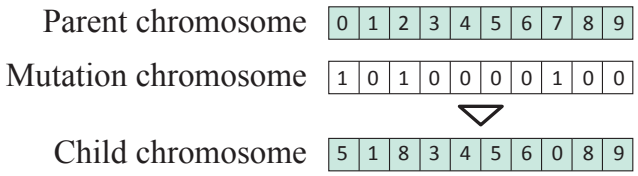

Figure 2.8 : Mutation: Flipping.

2.1.2.2 Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged. Changing action and selected bits are shown in bold in Fig. 2.9

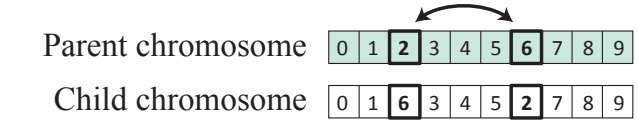

Figure 2.9 : Mutation: Interchanging.

2.1.3 Genotype-phenotype mapping

Genotype-phenotype mapping is the change of the representation of the chromosome. In some of the optimization problems, binary or discrete coding of chromosomes is required for algorithm to perform easily. For example, for this thesis, each gene is represented by a number and this number corresponds to a specific circuit element, such as semiconductor or passive component. In the algorithm back side, calculations are made using numbers, but output of the algorithm is shown to the user as component specifications, such as "100 μF capacitor". When mapping chromosomes, a biasing should be avoided since biasing forces algorithm to end up in incorrect results. Mapping step is not always required since in continuous optimization problems genotypes and phenotypes are same.

2.1.4 Fitness score calculations

Fitness score calculations are evaluated by fitness function. Fitness value determines the quality of a solution. In some applications fitness value can be penalized using penalty functions. Penalty functions calculates the feasibility of the solution and deteriorate the fitness value if required. For this thesis, fitness function calculates the total converter cost.

Also most approaches are aimed for minimization of the fitness function call in the algorithm run. Some of the fitness functions calls are very expensive such as constructing objects or running a simulation model for each function call. By minimizing the number of calls, algorithm can perform faster [20].

2.1.5 Selection operators

Selection operators are used for selection of the parents that will be used for the breeding of the next generation. Selection operators that are used in literature frequently is given below with brief descriptions.

2.1.5.1 Elitist selection

This selection type selects parents which has the highest fitness scores between offspring. Elitist selection is used for keeping the best fitness scored individual in the population.

2.1.5.2 Comma selection

Selects the α best solutions from γ offspring. In this selection type, all the selected solutions are from offspring population, not including any of the individuals from parent population.

2.1.5.3 Plus selection

Selects the α best solutions from γ offspring and α solutions that are parent of best solutions.

2.1.5.4 Roulette wheel selection

This selection type selects randomly distributed solutions with a probability depending on their fitness score. RWS may suffer from premature convergence and may stuck in a local optimum. One of the mostly used algorithm in the literature. There is also variant of this selection which is stochastic universal sampling (SUS), simply aiming reduction of premature convergence.

$$p(i) = \frac{f(i)}{\sum_{j=1}^{p} f(j)} \tag{2.1}$$

$p(i)$ is the probability function and $f(i)$ is the fitness function. p is the population size.

2.1.5.5 Tournament selection

This selection type selects k random individuals from population. Then matches with each other. Higher fitness valued individual wins and participates to the crossover. Whole process repeats r times. Tournament selection gives equal chance to all individuals; hence diversity is increased. But convergence speed may be lower. Tournament selection is one of the mostly used algorithm in the literature.

$$p(i) = \begin{cases} \frac{C_{r-1}^{k-1}}{C_r^k} & if \quad i \in [1, r-k-1]] \\ 0 & if \quad i \in [r-k, r]] \end{cases} \tag{2.2}$$

2.1.5.6 Rank selection

This selection ranks the population by their fitness score, giving the lowest one 1 point, and the highest one H points. Important point in this selection is the rank, not the exact value of fitness score. Linear probability function is given as:

$$p(i) = \frac{rank(i)}{H \times (H-1)} \tag{2.3}$$

Rank selection also has exponential probability function:

$$p(i) = 1 \times exp\left(-\frac{rank(i)}{c}\right) \quad c = \frac{(H \times 2 \times (H-1))}{6 \times (H-1) + H} \tag{2.4}$$

2.1.5.7 Truncation selection

This selection orders the individuals by their fitness score. Then only fittest p portion is selected for crossover and repeated $1/p$ times. This selection method is less used in literature. Generally used for very large populations.

2.1.6 Termination

Termination of the genetic algorithm is the condition for algorithm to stop. There are several approaches to termination and they are given below.

Genetic algorithm may stuck in local optima and tries to find a better solution infinitely if there is no termination condition. Termination conditions guarantees to stop algorithm after some time. Some of the termination conditions are given below. There are also restart strategies to find global optimum point by running the algorithm for number of times. Even some of the runs face stagnation problems, in total, algorithm finds an optimum point.

2.1.6.1 Maximum generations

Genetic algorithm is looped through predefined number of generations. When the maximum generation number is exceed, algorithm terminates. Best fitness level

individual is considered as solution. Maximum generation is generally kept between 100 and 300 generations.

2.1.6.2 Elapsed time

Genetic algorithm can be terminated by predetermined time limit. Depending on the problem type, time limit can be a few minutes to a few tens of minutes.

2.1.6.3 Stall generations

Genetic algorithm is terminated if there is no improvement in the fitness levels for a predetermined number of generations.

2.1.6.4 Stall time limit

Genetic algorithm is terminated if there is no improvement in the fitness levels for a predetermined time limit.

For the best result user can define a condition for each type of termination.

3. SINGLE OUTPUT FLYBACK CONVERTER DESIGN

Although design of a flyback converter seems easy, it requires a broad knowledge of power electronics from passive components to semiconductor switches and also to magnetic design. Dependence between each component requires an iterative design process. One of the main advantages to use a flyback converter is the isolation between primary and secondary sides. Also lower part count makes it easier to design and more cost friendly.

Textbook circuit schematic for a utility connected flyback converter is given in Fig. 3.1. A simple explanation of the operation of the circuit components is as following.

Diode bridge, DB1, and C1 rectifies the utility voltage and creates a low ripple, high voltage DC bus for flyback converter. Controlled switch, Q1, along with the control circuitry, controls the power transfer from primary side to secondary side. LM1 is the momentary energy storage component for switch on time. Transformer, TR1 is the energy transfer device from primary to secondary side. Secondary rectifier diode, D1, blocks the output capacitor's DC voltage being discharged through secondary winding. Output bulk capacitor, C2, holds the output voltage relatively constant for desired DC output voltage.

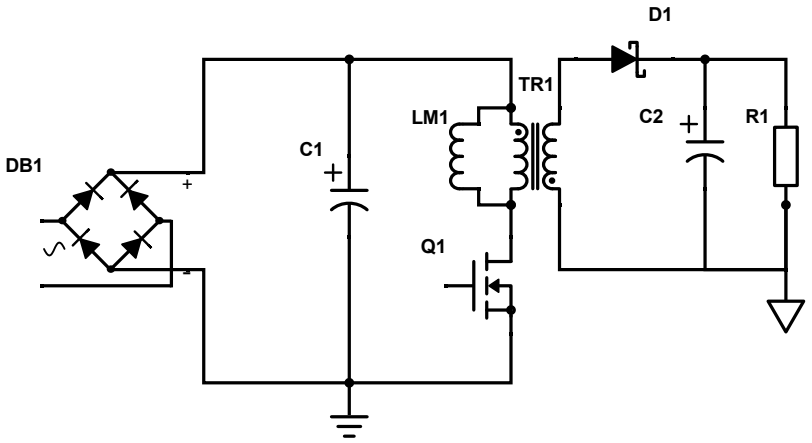

Figure 3.1 : Flyback converter

Schematic references are given below:

- DB1: Bridge rectifier diode
- C1: DC bus electrolytic capacitor
- LM1: Transformer magnetizing inductance
- TR1: Ideal transformer
- Q1: MOSFET
- D1: Secondary rectifier diode
- C2: Output electrolytic capacitor
- R1: Resistive load

Although Fig. 3.1 simplifies the flyback converter representation, it is far from a realistic flyback converter. Fig. 3.2 is a more realistic circuit representation with the important parasitic components.

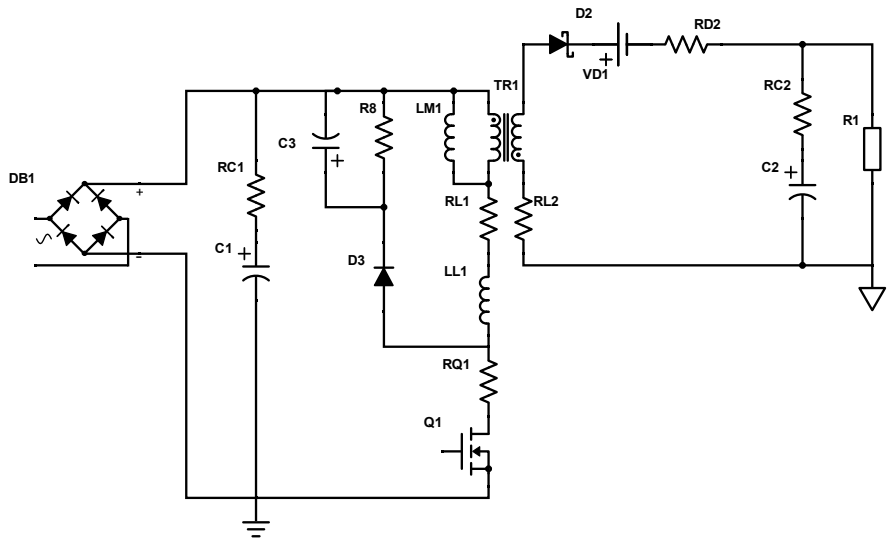

Figure 3.2 : Flyback converter with important parasitics.

Main component references are given above. Rest of the references used in the Fig. 3.2 are given below:

- RC1: DC bus electrolytic capacitor ESR
- C3: Snubber capacitor
- R8: Snubber resistor
- D3: Snubber diode
- RL1: Transformer primary winding DC resistance
- LL1: Transformer leakage inductance

- RQ1: MOSFET on resistance
- RL2: Transformer secondary winding DC resistance
- VD1: Secondary rectifier diode forward voltage
- RD2: Secondary rectifier diode on resistance
- RC2: Output capacitor's ESR

Now much of the parasitics in mind, a detailed circuit operation can be explained as following:

Firstly, charging of the C1, DC bus capacitor will be explained. To be able to have a relatively constant voltage at the input of the converter, C1 capacitor should have high enough capacitance to hold voltage level when the bridge diodes are reverse biased. At the peaks of the line voltage C1 capacitor charges through two bridge diodes and RC1, ESR of the C1. Capacitor is charged twice the line frequency for full wave bridge rectifier. This charging path is shown blue color in left hand side of the Fig. 3.3. Having a high voltage DC voltage at the input of the flyback converter, now actual converter operation can be explained.

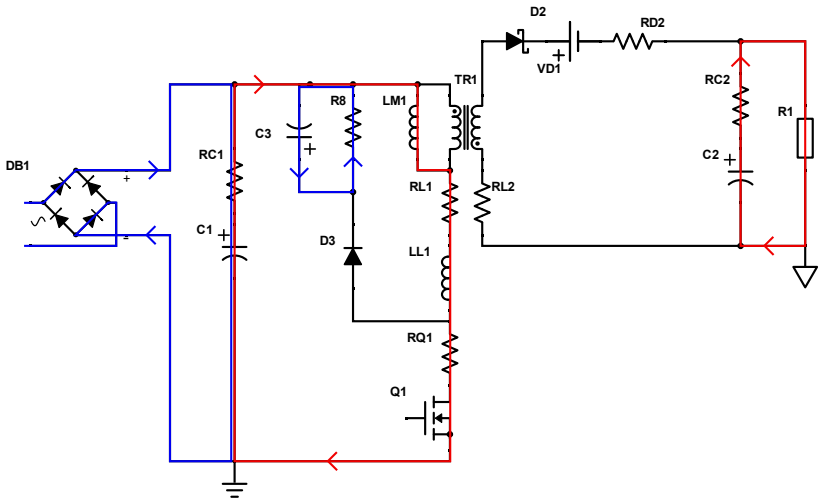

Figure 3.3 : Flyback converter MOSFET on-time with parasitics.

In Fig. 3.3, flyback converter's switch on-time is shown. Red loops are the main current loops, while small blue loop between C3 and R8, is auxiliary loop. When Q1 switch turns on, C1 acts as a voltage source and drives current through ESR of C1, magnetizing inductance of the transformer, resistance of the primary winding, leakage inductance of the transformer and on resistance of the Q1, respectively and closing

the loop on C1. At this instance, there is no current flow in the secondary side of the transformer. Voltage between dotted terminal and non-dotted terminal of the primary side of the transformer and secondary side dotted and non-dotted terminals is positive. C2 capacitor voltage and transformer secondary side voltage adds up and reverse biases the D2 diode. Because of this, primary side of the transformer acts as an inductor.

For the load side current loop, C2 capacitor discharges through ESR of C2 to the load, R1 and closes current loop onto itself.

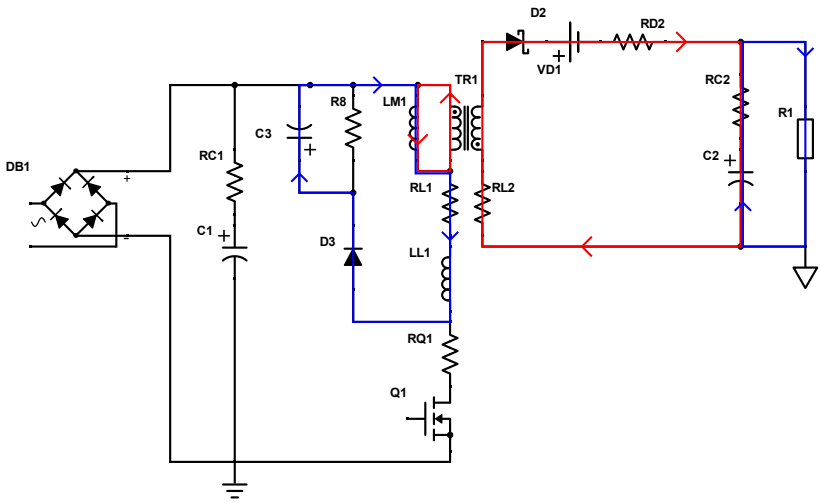

Figure 3.4 : Flyback converter MOSFET off-time - with parasitics.

In Fig. 3.4, flyback converter's switch off-time is shown. Again, red loops are the main current loops, while blue loops are auxiliary loops. Since the current in an inductor can not change instantaneously, when the Q1 switch is turned off, LM1 current tries to complete its loop through Q1. But having closed its gate, Q1 does not permit current flow. So parasitic capacitance of the Q1 (which is not shown in above figure) charges up to a certain voltage. When the transformer primary voltage reaches to a certain voltage level, which is called reflected voltage from secondary, LM1 current circulates through TR1 primary winding. Since TR1 is an ideal transformer, primary current is transferred to the secondary side by the turns ratio. So by this mechanism energy is transferred to the secondary winding. Current flow is from non-dotted winding of primary side to the non-dotted winding of the secondary winding. Current flow in secondary winding forces D2 to be forward biased and secondary side current circulates through VD1, RD2, RC2, C2 and RL2, respectively.

At the primary side, leakage inductance of the winding is also important. On-time current also flows in the leakage inductor and sudden changes is not possible as in magnetizing inductance, LM1. When Q1 switch is turned off, current flow in LL1, forces Q1 parasitic capacitance to charge higher voltages than DC bus voltage. When a certain voltage level is reached, D3, snubber diode forward biased. This voltage is called snubber voltage and it is higher than the sum of DC bus voltage and reflected voltage of the secondary side.

It can be seen that there is no direct transformer effect of the flyback transformer. First, energy is stored in the primary side inductor. And then, this energy is released in the secondary side. Because of this operational principle flyback transformers are commonly referred as coupled inductors.

It can be seen that resistive power loss in,

- RL1 and RQ1 is only during Q1 on-time,
- RL2 and RD2 is only during D2 on-time,
- RC2 is always,
- and RC1 is also during Q1 on-time and bridge diode conduction time. RC1, ESR of C1 is a special type of resistance and it will be explained in the power loss calculations section.

Flyback converter fundamental waveforms are trivial to give at this point and can be found in any text book. Especially Hart's Power Electronics textbook is a good reference for this subject [21]. Also simulation results of fundamental waveforms of flyback converter can be found in Chapter 5.

3.1 CCM and DCM Operation

CCM and DCM operation of the flyback converter transformer is a little different from conventional inductors. While there is no measurable continuous conduction current on any of the windings, total conduction time in primary and secondary windings is considered as conduction mode of the transformer. Hence it should be thought as continuous magnetic field on transformer for CCM conduction mode. If the current on the primary winding is non-zero while turn-on event of the Q1 and turn-off event of the D2 diode, it is called continuous conduction mode. Otherwise, it is called discontinuous conduction mode.

Table 3.1 : Comparison between CCM and DCM operation of flyback converter.

Parameter	CCM	DCM
Ripple and RMS current	Low	High
MOSFET conduction loss	Low	High
MOSFET turn-off loss	Low	High
Core loss	Low	High
Cross regulation	Better	Worse
EMI filter requirement	Smaller	Bigger
Output filter	Smaller	Bigger
Diode recovery loss	Yes	No
Right hand plane zero problem	Yes	No
Voltage stress for secondary diode	Higher	Lower
Light load efficiency	Lower	Higher
Transformer size	Bigger	Smaller

Differences between DCM and CCM operation is given in Table 3.1 [22]

3.2 Equations for Flyback Converter

In this section, the equations that are used for flyback converter design is given. The basis of the calculations are AN-4137 by Fairchild Semiconductor [23]. These equations are also the basis for the genetic algorithm calculations.

Required input power is calculated by the following formula, using efficiency estimation. Efficiency is generally between 60% to 80%. In the genetic algorithm, an iterative approach is followed for calculations. First, efficiency is assumed and then using power loss calculations, actual efficiency is obtained. Then this new value is used as estimation. In a few iterations the error between estimation and calculated efficiencies becomes negligible. By using this approach, calculated values converges to actual values.

$$P_{in} = \frac{P_{out}}{\eta} \tag{3.1}$$

P_{in}: Input voltage. P_{out}: Output voltage. η: Efficiency.

DC bus capacitor is 2-3 μF per watt input power for universal input converter. Universal input voltage range is considered as 85-265 V RMS. Minimum DC bus voltage is calculated considering DC bus capacitor value is selected.

$$V_{DC_{min}} = \sqrt{2 \times (V_{line_{min}})^2 - \frac{P_{in} \times (1 - D_{ch})}{C_{DCBus} \times f_{line}}} \tag{3.2}$$

$V_{line_{min}}$: Minimum line voltage RMS value. D_{ch}: DC bus capacitor, charge duty. Generally 2-3 ms durations. Charge duty value is generally 0.2-0.3 for full bridge rectifier. f_{line}: Line frequency. C_{DCBus}: DC bus capacitor's capacitance value. DC bus charge duty is shown in Fig. 3.5.

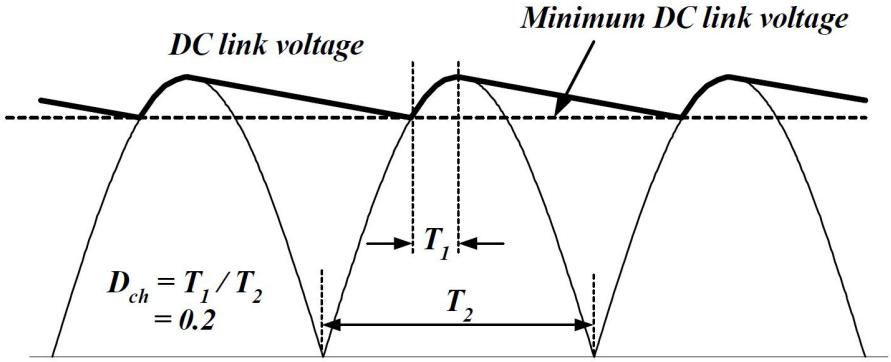

Figure 3.5 : DC bus capacitor charge duty [23].

Maximum DC bus voltage is calculated by the maximum line voltage:

$$V_{DC_{max}} = \sqrt{2} V_{line_{max}} \tag{3.3}$$

$V_{line_{max}}$: Maximum line voltage RMS value. V_{DCmax}: Maximum DC bus voltage.

Reflected output voltage is calculated by:

$$V_{RO} = \frac{D_{max}}{1 - D_{max}} V_{DC_{min}} \tag{3.4}$$

V_{RO}: Reflected output voltage. D_{max}: Maximum duty cycle of the converter.

Ideal condition drain-source voltage on the semiconductor switch is calculated below. This voltage is not considering snubber voltage. When leakage inductance is considered, below voltage is not valid and snubber voltage should be considered.

$$V_{ds_{nom}} = V_{DC_{max}} + V_{RO} \tag{3.5}$$

In Fig. 3.6 different voltage levels on the Q1 switch is shown. Short duration spike on top of the V_{DC} and V_{RO} is due to leakage inductance and should be clamped to a

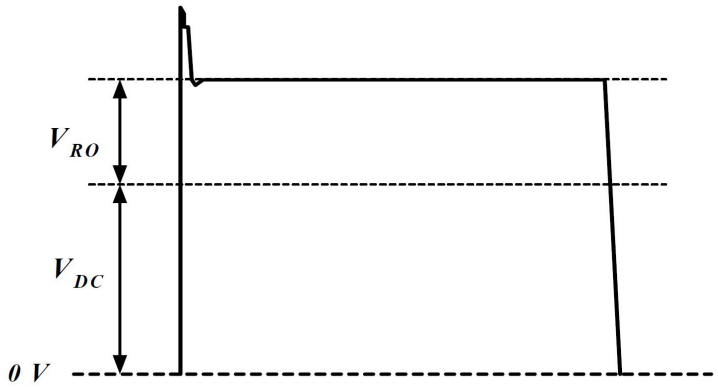

Figure 3.6 : Voltage levels on the Q1 switch [23].

snubber voltage. If it is not considered in the design, Q1 switch is likely to fail due to voltage breakdown.

$V_{ds_{nom}}$: Nominal maximum voltage on the switch. Ignoring leakage inductance.

K_{RF} is the ripple factor in the primary current. For DCM operation $K_{RF} = 1$ and for CCM operation $K_{RF} < 1$. In Fig. 3.7 K_{RF} is visualised.

$$K_{RF} = \frac{\Delta I}{2I_{EDC}} \tag{3.6}$$

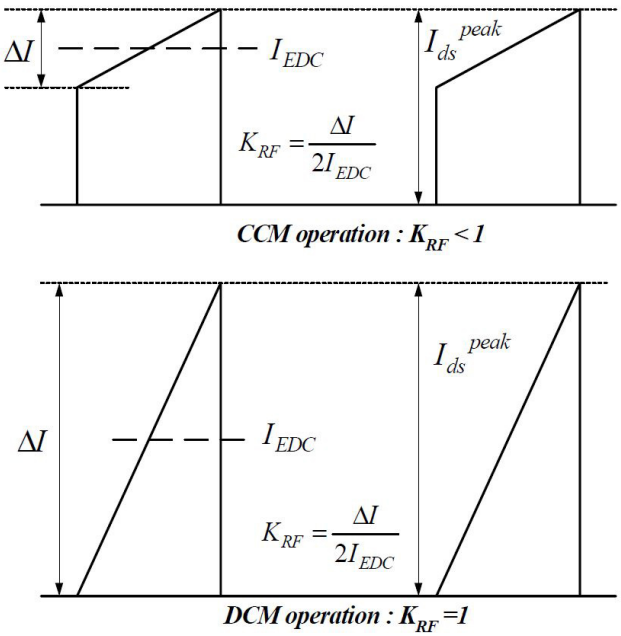

Figure 3.7 : Ripple factor of the primary current [23].

Magnetizing inductance of the transformer L_m is calculated below. Magnetizing inductance depends on the applied volt-seconds, power level, switching frequency and desired ripple current factor.

$$L_m = \frac{(V_{DC_{min}} \times D_{max})^2}{2\,P_{in}\,f_s\,K_{RF}} \tag{3.7}$$

Primary side current values are calculated below:

$$I_{ds_{RMS}} = \sqrt{\left[3(I_{EDC})^2 + (\frac{\Delta I}{2})^2\right]\frac{D_{max}}{3}} \tag{3.8}$$

$$where \qquad I_{EDC} = \frac{P_{in}}{V_{DC_{min}} \times D_{max}} \tag{3.9}$$

$$and \qquad \Delta I = \frac{V_{DC_{min}} \times D_{max}}{L_m\,f_s} \tag{3.10}$$

I_{EDC} is the equivalent DC current level. ΔI is the ripple current. Primary side peak current value is calculated as:

$$I_{ds_{peak}} = I_{EDC} + \frac{\Delta I}{2} \tag{3.11}$$

Required turns ratio is calculated by following equation:

$$n = \frac{V_{RO}}{V_{out} + V_{d_{fwd}}} \tag{3.12}$$

n: Turns ratio. V_{out}: Output voltage. $V_{d_{fwd}}$: Forward voltage drop of secondary side rectifier diode.

Required minimum DC voltage for CCM operation is calculated by:

$$V_{DC_{CCM}} = \left(\frac{1}{\sqrt{2L_m f_s P_{in}}} - \frac{1}{V_{RO}}\right)^{-1} \tag{3.13}$$

Minimum primary winding turns is calculated by:

$$N_{prim_{min}} = \frac{L_m I_{ds_{peak}}}{B_{sat} A_e} \times 10^6 \tag{3.14}$$

By rounding up minimum primary turns, actual primary turns can be found.

$$N_{prim} = ceil(N_{prim_{min}}) \tag{3.15}$$

L_m: Magnetizing inductance in Henry. B_{sat}: Saturation flux density in Tesla. A_e: Core cross sectional area in m^2.

Secondary winding turns. Found by rounding up primary turns.

$$N_{sec} = ceil(\frac{N_{prim}}{n}) \tag{3.16}$$

Air gap of core in millimetres is found by:

$$l_{gap} = \mu_0 \frac{A_e}{2} 10^3 \left(\frac{N_{prim}^2}{L_m} - \frac{10^9}{a_L} \right) \tag{3.17}$$

μ_0: Permeability of free space, $\mu_0 = 4 \times \pi \times 10^{-7}$. a_L: Inductance factor of the core, in $nH/turns^2$.

RMS current of the secondary winding is:

$$I_{sec_{RMS}} = I_{ds_{rms}} \sqrt{\frac{1 - D_{max}}{D_{max}}} \frac{V_{RO}}{V_{out} + V_{d_{fwd}}} \tag{3.18}$$

For single output flyback converter diode current is equal to secondary winding current.

$$I_{d_{RMS}} = I_{sec_{RMS}} \tag{3.19}$$

$I_{d_{RMS}}$ is the secondary side rectifier diode RMS current.

Reverse voltage on secondary fast rectifier diode:

$$V_{d_{rvs}} = V_{out} + \frac{V_{DC_{max}}(V_{out} + V_{d_{fwd}})}{V_{RO}} \tag{3.20}$$

Ripple current of the output capacitor:

$$I_{sec_{cap}} = \sqrt{(I_{d_{RMS}})^2 - (I_{out})^2} \tag{3.21}$$

Ripple voltage of the output capacitor is calculated by:

$$\Delta V_{out} = \frac{I_{out} D_{max}}{C_{out} f_s} + \frac{I_{ds_{peak}} V_{RO} R_{ESR}}{V_{out} + V_{d_{fwd}}} \tag{3.22}$$

R_{ESR}: Equivalent series resistance of the secondary capacitor.

3.3 RCD Snubber Design

A simple RCD snubber can be calculated by following formulas. Power loss in the snubber network is directly related to the leakage inductance of the flyback transformer.

Power loss equation is:

$$P_{sn} = \frac{1}{2} f_s L_{leak} (I_{ds_{peak}})^2 \frac{V_{sn}}{V_{sn} - V_{RO}} \tag{3.23}$$

Snubber voltage is usually selected 2-2.5 times of the V_{RO}. Assuming that snubber voltage is selected, snubber resistor is calculated as:

$$R_{sn} = \frac{(V_{sn})^2}{P_{sn}} \tag{3.24}$$

By selecting the ripple voltage of the snubber capacitor a small percentage of the snubber voltage, snubber capacitor can be calculated:

$$C_{sn} = \frac{V_{sn}}{\Delta V_{sn} R_{sn} f_s} \tag{3.25}$$

Maximum voltage on the semiconductor switch can be calculated as:

$$V_{ds_{max}} = V_{DC_{max}} + V_{sn} \tag{3.26}$$

3.4 Magnetic Component Design

Magnetics is one of the most important and exhaustive design stage in a power electronics converter design. From the electrical circuit point of view, inductance is the required parameter; but from the magnetical point of view, electromotive force and flux density is involved in the design stage. This additional magnetic point of view requires knowledge of material, B-H curves, frequency characteristics of material, temperature dependency of material and DC bias characteristics of the material. Power losses related magnetic core and windings are even more complicated. There is no complete analytical calculation on core losses of a magnetic material from its chemical and structural composition. All calculations are relied on empirical loss data [24].

Being said that, in Equation 3.7, required inductance value for the desired operating point is given. Variables in Equation 3.7 are all electrical parameters. While, when magnetic material is considered in the design stage, saturation flux density, core size and a_L value also should be considered. Equation 3.17 and 3.14 considers a_L and flux density variables which are related to magnetic material. For a final step, all the windings should fit into selected core.

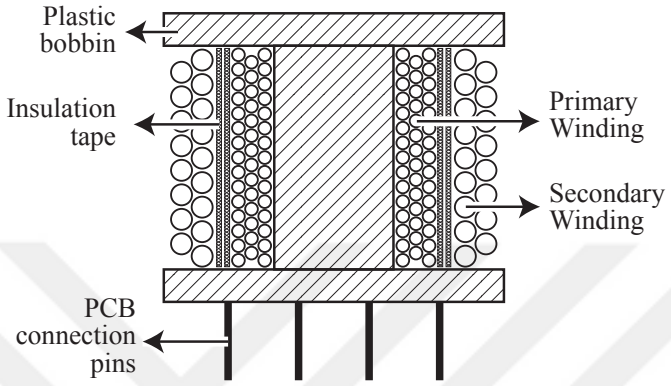

Figure 3.8 : Flyback transformer winding schema

In Fig. 3.8, flyback transformer winding schema is shown. For a general approach, primary winding is wound on the inner side of the coil former. Between primary and secondary windings, for the insulation requirement, insulation tape is applied. On the outer side of the coil former secondary winding is wound with required number of turns. This arrangement has several advantages, and one major disadvantage. Advantages are, ease of manufacturing and ease of insurance of required insulation level. One major disadvantage is the low coupling between primary and secondary windings.

For physical placement of the windings, below calculations are followed:

Bare copper area of the windings and turns numbers gives total copper area. But there are factors that are affecting the winding area usage that yields in non-ideal utilization of the winding area. These factors are enamel layer around copper, round wires and imperfections of the winding process. Enamel layer is used for basic insulation of the wire and creates 0.02 to 0.04 mm of additional thickness in the diameter for 0.2 to 0.5 mm diameter wire [25]. Placement of circular shaped wire adjacent into each other creates unusable areas. All of these factors are combined in the term called utilization

factor (or copper filling factor), k_u. For different applications and design approaches k_u can be taken as 0.05 to 0.65 [26]. For this thesis, $k_u = 0.5$ will be used.

Fig. 3.9 shows six wire strands' cross sectional area and unusable area.

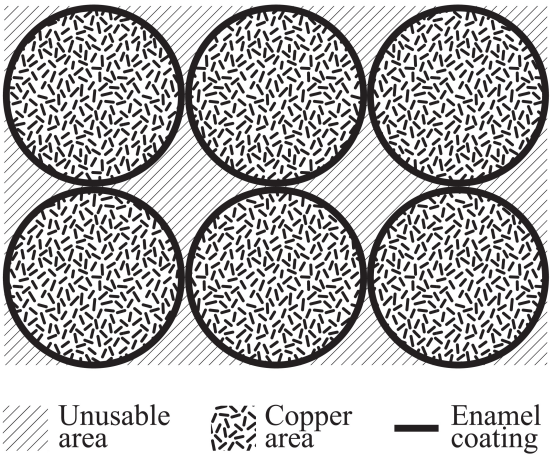

Figure 3.9 : Usable winding area

Required winding area for primary and secondary windings are calculated by:

$$W_{A_{prim}} = \frac{N_{prim} \times A_w}{k_u} \tag{3.27}$$

$$W_{A_{sec}} = \frac{N_{sec} \times A_w}{k_u} \tag{3.28}$$

$W_{A_{prim}}, W_{A_{sec}}$ is winding area of primary and secondary windings, respectively. A_w is cross-sectional area of wire. k_u is utilization factor of winding.

And total winding area is:

$$W_{A_{total}} = W_{A_{prim}} + W_{A_{sec}} \tag{3.29}$$

$W_{A_{total}}$ is the total required winding area.

Leakage calculations are taken from Thummala et al's work which is focused on transformer parameter estimation and loss analysis [27]. Calculations are based on the non-interleaved winding schema and uses MMF distribution in the transformer.

Table 3.2 : 24 to 32 AWG table

AWG	Diameter[mm]	Crosssectional Area[mm^2]	Ohm/m	Max Frequency[kHz]
24	0.500	0.19635	0.0870	68
25	0.450	0.15904	0.1075	85
26	0.400	0.12566	0.1360	107
27	0.355	0.09898	0.1727	130
28	0.315	0.07793	0.2193	170
29	0.280	0.06157	0.2776	210
30	0.250	0.04908	0.3482	270
31	0.224	0.03940	0.4338	340
32	0.200	0.03141	0.5441	430

$$L_{leak} = \mu \frac{MLT_{prim}}{b_w} N_{prim}^2 \left[\frac{n_{lp}h_p + n_{ls}h_s}{3} + \frac{(2n_{lp}-1)(n_{lp}-1)}{6}\left(\frac{h_{ip}}{n_{lp}}\right) + \frac{(2n_{ls}-1)(n_{ls}-1)}{6}\left(\frac{h_{is}}{n_{ls}}\right)\right] \tag{3.30}$$

μ_0 is the permeability of free space. l_w is the mean length turn (MLT). b_w is the coil former height. n_{lp} and n_{ls} is the number of primary and secondary layers respectively. h_{ip} and h_{is} is the insulation thickness between conductors for each of the primary and secondary layers respectively (for enamelled wire enamel thickness). h_i is the insulation thickness between primary secondary layer (insulation paper etc.).

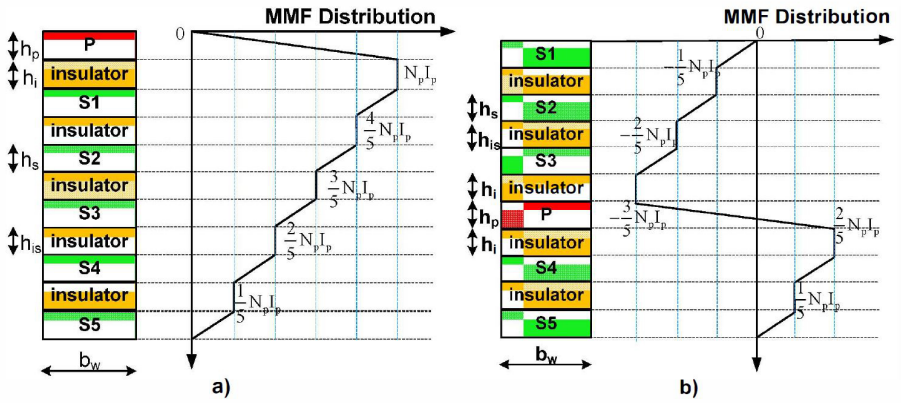

Figure 3.10 : MMF distrubution on the flyback transformer [27]

Fringing flux factor:

$$F = 1 + \frac{l_{gap}}{\sqrt{A_e}}\, ln(\frac{2G}{l_{gap}}) \tag{3.31}$$

G is the E core's winding area length. For a EE core transformer, winding area length should be taken as 2G length.

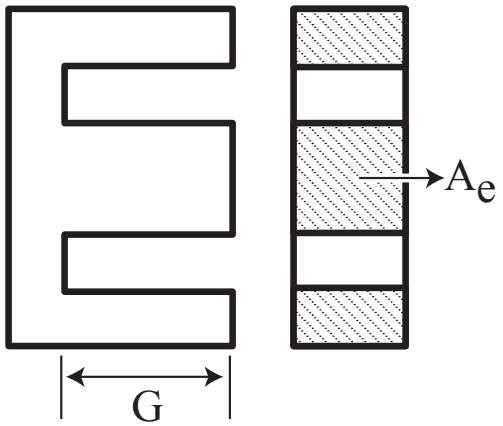

Figure 3.11 : Fringing flux core parameters

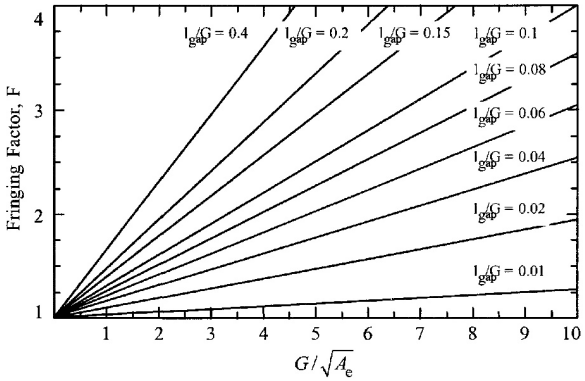

Figure 3.12 : Fringing flux parameters [28]

3.5 Power Loss Calculations

In this section power loss calculations will be given for each component.

For a typical flyback converter design overall efficiency is around 60-80%. For a rough estimation, 33% of the total power loss is due to MOSFET, 57% is due to secondary rectifier diode, 5% is due to magnetics and 5% is others [29]. It is seen that great portion of the power loss comes from secondary rectifier diode for the flyback converter.

3.5.1 MOSFET loss

MOSFET is the controlled switch of the flyback converter. There are four power loss mechanism which are related to the MOSFET, namely:

1. Conduction losses (P_{mos_c})
2. Switching losses ($P_{mos_{sw}}$)
3. Blocking losses (P_{mos_b})
4. Gating losses (P_{mos_g})

Blocking losses are normally neglected due to their insignificant contribution to the total MOSFET power loss [30].

Total loss can be approximated by the following formula

$$P_{mos_{total}} \approx P_{mos_c} + P_{mos_{sw}} + P_{mos_g} \tag{3.32}$$

Conduction loss is due to MOSFET's internal on resistance, R_{DSon}. Calculation of conduction loss is similar to resistive power loss; but MOSFET's R_{DSon} changes with drain source current and gate-source voltage. To simplify calculations, generally, worst case R_{DSon} is taken into account, especially for low power applications. For high power applications, or more precise calculations, after determining operating gate-source voltage and drain source current datasheet plots can be used for power loss calculations. For this thesis, simplified worst case scenario calculations will be used.

$$P_{mos_c} = R_{ds_{on}} \times I^2_{ds_{RMS}} \tag{3.33}$$

Switching losses occur because of non-ideal behaviour of semiconductor devices. Non-zero voltage and current waveforms during transitions creates overlapping areas which create power loss. In Fig. 3.13, MOSFET switching transitions are given for turn-on and turn-off events, respectively. Turn-on event can be explained as following,

- In 1, drive voltage is applied to the gate of the MOSFET. Gate voltage is increased from zero to V_{th}, threshold voltage, mainly charging the C_{gs} capacitor. This period is named as turn-on delay because neither V_{ds}, nor I_{ds} changes their values.
- In 2, gate voltage is increased from V_{th} to V_{miller}. In this period MOSFET is in linear mode. Ids current is proportional to the gate voltage.

- In 3, gate voltage stays at V_{miller} voltage. This is because of gate current is discharging the C_{gd} capacitor. Discharging C_{gd} capacitor rapidly decreases drain source voltage of the MOSFET.

- In 4, V_{ds} and I_{ds} are reached almost to their final values. In this period C_{gs} and C_{gd} capacitors are charged further to lower $R_{ds_{on}}$ of the MOSFET. So increasing gate voltage from miller voltage to the its final value is important for lowering conduction losses.

Turn-off procedures are the inverted version of the turn-on procedures.

One important point for Fig. 3.13, is that it is given for a CCM inductor current waveform. For turn-on event, in 2, current reaches its final value starting from zero current, namely its continuous conduction current level. For a DCM flyback converter, current would start from zero and increases linearly until the end of the period, which is different from Fig. 3.13.

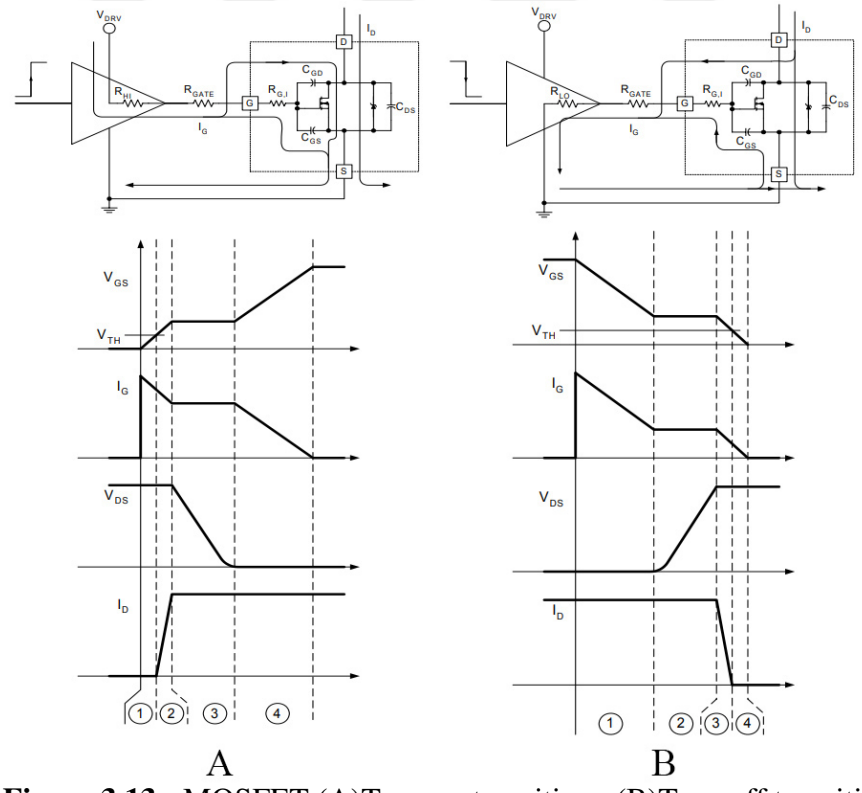

Figure 3.13 : MOSFET (A)Turn-on transition (B)Turn-off transition.

Switching loss calculations are complicated for MOSFETs because of highly non-linear behaviour of parasitic capacitances. Typical parasitic capacitances for a high voltage MOSFET is given in Fig. 3.14. It can be seen that by increasing voltage,

capacitances decreases non-linearly, for C_{rss}, even increases after a particular voltage level. So for analytical calculations it is very hard to find exact capacitance values. In datasheets C_{oss} value is generally given where the knee point occurs. For example in Fig. 3.14 around 25-50 V is where the knee point occurs. After that point a first order approximation can be made using Equation 3.34.

$$C_{oss_{application}} = C_{oss_{datasheet}} \times \sqrt{\frac{V_{ds_{datasheet}}}{V_{ds_{application}}}} \tag{3.34}$$

Also in datasheets, parasitic capacitances are gives as C_{iss}, C_{oss} and C_{rss}. Input capacitance, output capacitance and reverse transfer capacitance, respectively.

$$C_{rss} = C_{gd} \tag{3.35}$$

$$C_{oss} = C_{gd} + C_{ds} \tag{3.36}$$

$$C_{iss} = C_{gd} + C_{gs} \tag{3.37}$$

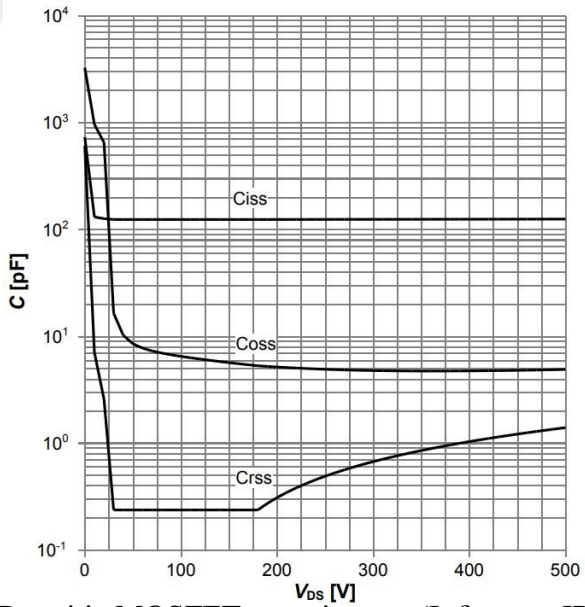

Figure 3.14 : Parasitic MOSFET capacitances (Infineon - IPN80R2K0P7).

Having C_{oss} value at application drain source voltage, switching loss due to MOSFET output capacitance can be calculated using Equation 3.38 and 3.39. Extensive explanations and waveforms can be found in Halder's work about power loss model of flyback converter's main switch [31].

$$P_{mos_{sw(Coss)}} = \frac{1}{2} C_{oss} V_{ds}^2 f_s \tag{3.38}$$

Also there is another term for switching loss in MOSFET. This second power loss is due to overlapping current waveforms. In Equation 3.39 power loss due to overlapping waveforms is given. Notice that switching on and off transients are given separately. The reason is point out the DCM and CCM operation. For DCM operation, current level of switch on transient is zero. So $I_{ds_{start}} = 0$, left hand side of the equation is zero. There is no power loss at turn on transient. Also in CCM operation, if there is a high current ripple $I_{ds_{start}}$ and $I_{ds_{end}}$ values are different, otherwise they can be taken as same value.

C_{oss} loss irrespective of the load current. Every switching action creates a power loss, but power loss due to overlapping waveforms is dependent on the load current.

$$P_{mos_{sw(overlap)}} = P_{mos_{swON}} + P_{mos_{swOFF}} = f_s \left[\frac{V_{ds} I_{d_{start}} t_{on}}{6} \right] + f_s \left[\frac{V_{ds} I_{d_{end}} t_{off}}{6} \right] \tag{3.39}$$

Final equation for switching loss of MOSFET is the sum of each component.

$$P_{mos_{sw}} = P_{mos_{sw(Coss)}} + P_{mos_{sw(overlap)}} \tag{3.40}$$

Last part of the MOSFET power loss is the gating loss. In Equation 3.41 gating loss is given. V_{cc} is the gate drive voltage and Q_g is the total gate charge.

$$P_{mos_g} = Q_g f_s V_{cc} \tag{3.41}$$

3.5.2 Secondary rectifier diode power loss

It is said that up to 60% of the power loss comes from secondary rectifier diode. So the selection and calculation of the power loss in secondary rectifier diode is critical.

There are two power loss mechanism which are related to the power diode, namely:

1. Conduction loss (P_{d_c})
2. Switching (reverse recovery) loss ($P_{d_{sw}}$)

$$P_d = P_{d_c} + P_{d_{sw}} \tag{3.42}$$

Conduction loss in a diode consists of two mechanisms. First one is due to forward voltage of the diode, V_f. Forward voltage is relatively constant, but decreases with temperature for Shottky and bipolar diodes. For GaN and SiC diodes, forward voltage may be decrease or increase depending on the current level [32]. Second loss mechanism is due to resistance of the diode. Resistance is also relatively constant, but increases with temperature.

$$P_{d_c} = \frac{1}{T_{sw}} \int_0^{T_{sw}} V_{fwd} I_d(t) dt \tag{3.43}$$

$$P_{d_c} = V_{fwd} \times I_{d_{avg}} + R_d \times I^2_{d_{RMS}} \tag{3.44}$$

$I_d(t)$ is the instantaneous value of diode current. $I_{d_{avg}}$ is the average current through diode and $I_{d_{RMS}}$ is the RMS current through diode.

Diode datasheets include instantaneous current vs voltage curves. One can obtain forward voltage and resistance values investigating these curves. Some of the manufacturers gives V_{fwd} and R_d values in the form of Equation 3.44 to ease power loss calculations for designers.

3.5.3 Transformer power loss

Power loss computations for transformers can be tedious. There are three main component of the transformer power loss.

1. DC loss for each winding ($P_{w(n)_{DC}}$)
2. AC loss for each winding ($P_{w(n)_{AC}}$)
3. Core Loss (P_{core})

DC winding loss is the conductor loss due to DC resistance of the winding. Parameters that effects the DC loss are winding length, cross sectional area and conductor material.

$$P_{w(n)_{DC}} = R_{(n)_{DC}} \times I^2_{(n)_{RMS}} \tag{3.45}$$

AC winding loss is the conductor loss due to changing magnetic field. Changing magnetic field creates two types of loss in the windings, namely skin effect losses and proximity losses. Skin effect is the tendency of electrons to flow near the surface of the conductor due to changing magnetic field. Skin depth, δ, is defined as the depth where current density is reduced to $1/e$ of the surface level current density. Skin depth is calculated by Equation 3.46.

$$\delta = \sqrt{\frac{2\rho}{2\pi f \mu_0 \mu_r}} \tag{3.46}$$

Where δ is skin depth of conductor in meters, ρ is specific resistivity in Ωm, f is frequency in hertz, μ_0 is permeability of free space and μ_r is permeability of material

Resistivity of copper at room temperature is $\rho_{cu} = 1.68 \times 10^{-8}\,\Omega m$. Permeability of copper is $\mu_{cu} = 1.256 \times 10^{-6}$. For ease of calculations, below equation can be used for copper at 100 Celsius degree.

$$\delta = \frac{0.076}{\sqrt{f_{sw}}} \tag{3.47}$$

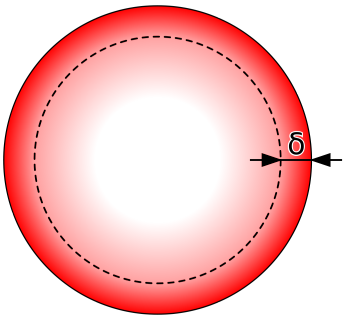

Figure 3.15 : Current distribution in a wire due to skin effect.

Skin effect considers only a single wire strand. When multiple turns of wire are wound adjacent to each other, like in a transformer, proximity effect should be considered. Proximity effect is the tendency of electrons in a conductor to flow in a manner to create smaller current loops in a changing magnetic field. Electrons flow in each layer near to the center of the transformer. Also when layers are compared, inner layers have higher current density. This effect is shown in Fig. 3.16. Because of skin effect and proximity effect, high current applications require a special type of wire called litz wire. Litz wire minimizes the losses related to these two effect.

Skin effect and proximity effect increases windings AC resistance and calculation of the increasing resistance is complex. Dowell's equation is widely used method for calculation of increasing AC resistance of the transformer. Parameters that effects the AC loss are wire diameter, winding schema, waveform and switching frequency. Equations for the calculation of AC resistance are given below:

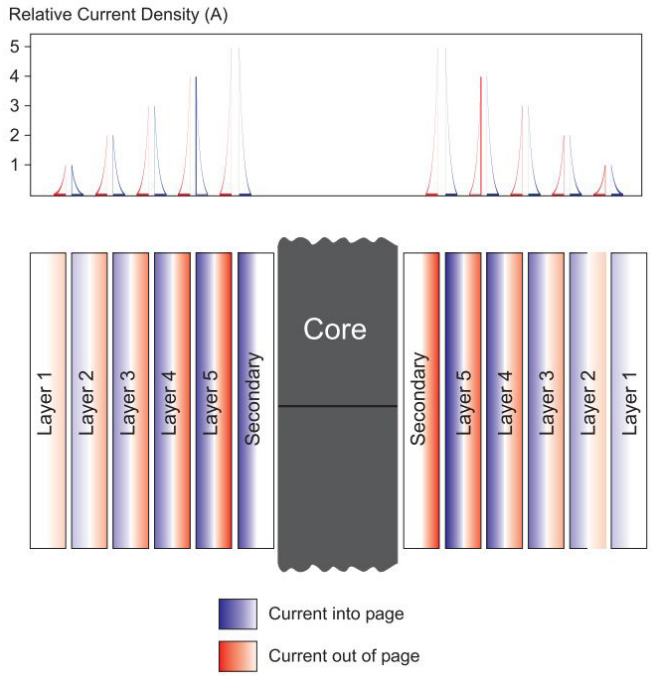

Figure 3.16 : Current distribution in a transformer due to proximity effect [33].

$$P_{w(n)_{AC}} = R_{(n)_{DC}} \times RF \times I^2_{(n)_{AC}} \tag{3.48}$$

Resistance factor, RF, is calculated below:

$$RF = \frac{R_{AC}}{R_{DC}} = \Delta' \left[\zeta_1' + \eta_w^2 \frac{2}{3}(m^2 - 1)\zeta_2' \right] \tag{3.49}$$

where m is the number of layers in winding, ζ_1' is skin effect factor:

$$\zeta_1' = \frac{sinh(2\Delta') + sin(2\Delta')}{cosh(2\Delta') - cos(2\Delta')} \tag{3.50}$$

ζ_2' is proximity effect factor:

$$\zeta_2' = \frac{sinh(\Delta') - sin(\Delta')}{cosh(\Delta') + cos(\Delta')} \tag{3.51}$$

Δ' is the modified penetration ratio:

$$\Delta' = \sqrt{\eta_w}\Delta \tag{3.52}$$

Δ is the penetration ratio:

$$\Delta = \frac{d_w}{\delta} \tag{3.53}$$

η_w is the porosity factor:

$$\eta_w = \frac{d_w N'}{b_w} \tag{3.54}$$

N' is the number of turns in one layer.

d_w is the equivalent thickness to foil conductor. For round wire it is calculated as:

$$d_w = \sqrt{\frac{\pi}{4}}d \tag{3.55}$$

d is the diameter of the round wire.

Core loss is the loss of core material. Parameters that effects the DC loss are magnetic DC bias on material, changing magnetic field, switching frequency, duty cycle and applied magnetic field's shape. Core loss data is given in the datasheet as kW/m^3 or W/m^3 for different switching frequency and temperature. By multiplying unit core loss by core volume, power loss due to core material can be obtained.

$$P_{core} = P_{w/m^3} \times V_{core} \tag{3.56}$$

3.5.4 Snubber power loss

Snubber power loss is given in Section 3.3. For subject integrity, loss equation will be given again:

$$P_{sn} = \frac{1}{2} f_s L_{leak} (I_{ds_{peak}})^2 \frac{V_{sn}}{V_{sn} - V_{RO}} = \frac{V_{sn}^2}{R_{sn}} \tag{3.57}$$

3.5.5 Rectifier diode power loss

Primary rectifier diodes rectifies low frequency AC voltage into DC voltage. There are three loss mechanism related to rectifier diodes, namely, conduction losses, switching losses and blocking losses. Switching and blocking losses can be ignored since they are small compared to conduction losses. Conduction loss of a rectifier diode can be calculated as,

$$P_d = V_{fwd} \times I_{d_{avg}} + R_d \times I^2_{d_{RMS}} \tag{3.58}$$

3.5.6 Electrolytic capacitor power loss

Electrolytic capacitors are the bulk energy storage components of the circuit. There are two loss mechanism related to electrolytic capacitors, namely, leakage loss and equivalent series resistance (ESR) loss. Leakage losses are related with the parallel resistor, R_p, shown in Fig. 3.17. R_p value is generally very high, in tens of mega ohms range. This resistor changes with applied voltage and temperature. But unless capacitor is used near to its rated voltage and temperature, R_p value is very high and do not create a significant power loss. ESR losses are the main loss mechanism for an electrolytic capacitor. ESR of the capacitor is shown in Fig. 3.17 as R_{ESR}. R_{ESR} is depicted as a constant resistor, but in reality it changes with temperature, load life and applied frequency. First two effect generally ignored, but frequency effect should be considered. Electrolytic capacitor's ESR is calculated by Equation 3.59.

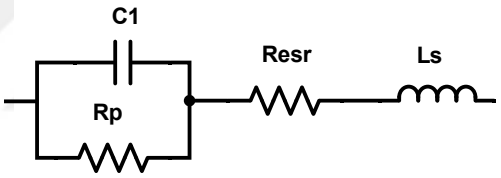

Figure 3.17 : Electrolytic capacitor model with parasitics.

$$R_{ESR} = X_c \times tan\delta \quad where \quad X_c = \frac{1}{2\pi f C} \tag{3.59}$$

X_c is the capacitive impedance. $tan\delta$ is dissipation factor. f and C is the applied frequency and capacitance, respectively.

In Fig. 3.18 it can be seen that with increasing frequency, ESR value decreases up to a certain frequency (10 kHz for this example). Above that frequency ESR is limited to a minimum level. Power loss of DC bus capacitor is another complex topic since capacitor response to the different ripple current frequencies changes. In a work by Hava, Ayhan and Aban, ESR related power loss in different inverter applications is investigated [34]. For each current harmonic value, different capacitor ESR value creates power loss. Power loss equation is given below.

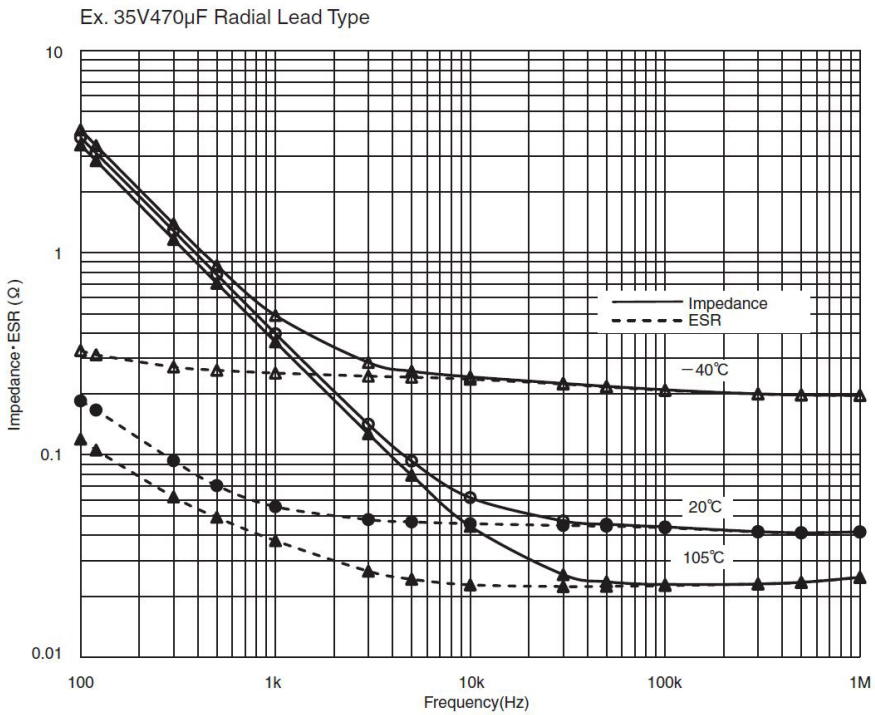

Figure 3.18 : Electrolytic capacitor ESR change with temperature and frequency [35].

$$P_{elcap} = I_{f1}^2 \times R_{ESR_{f1}} + I_{f2}^2 \times R_{ESR_{f2}} + ... \tag{3.60}$$

This approach requires decomposition of current waveform into harmonics for each capacitor since most of the time capacitor current waveform is not a sinusoidal. For a more practical approach, fundamental frequency components can be used. For flyback converter application, DC bus capacitor is subject to two different current component. First one is the line frequency charging current (for full wave rectification 100 or 120 hertz), and the second one is the switching frequency discharging current (in kHz region). For output capacitor, only one frequency component should be considered, which is the high frequency switching component.

4. OPTIMIZATION OF FLYBACK CONVERTER USING GA

Theoretical background of genetic algorithm and flyback converter design is given in Chapter 2 and 3, respectively. In this chapter, this two seemingly separated topics are combined and optimization of flyback converter is implemented from the cost point of view.

For implementation of the genetic algorithm there are numerous programming languages. C/C++ and FORTRAN is used for low level programming such as embedded systems. Genetic algorithm implementation of low level programming languages are hard but execution speed is fast compared to high level programming languages. Python is another widely used programming language for genetic algorithm implementation. Python is a high level programming languages and provides useful libraries and frameworks for developers. MATLAB is one of the mostly used high level programming language both in academy and industry for genetic algorithm implementation. For this thesis, MATLAB will be used for algorithm implementation. MATLAB has a toolbox for this purpose called Optimization Toolbox. In Optimization toolbox, there are number of other optimization methods. But genetic algorithm will be used for this thesis purpose.

4.1 MATLAB GA Toolbox

In this section MATLAB genetic algorithm toolbox will be explained. Genetic algorithm toolbox is a sub category of optimization toolbox in the MATLAB. Optimization toolbox can be opened using graphical interface of MATLAB under APPS tab or using script below in the command window.

```
optimtool
```

Optimization toolbox window is shown in Fig. 4.1. When optimization toolbox window is opened there are three main sections, namely, *"Problem Setup and Results"*, *"Options"* and *"Quick Reference"*. In *"Problem Setup and Results"* section, solver type, the function that is going to be optimized, number of variables and constraints can be

selected. In *"Solver"* drop-down menu, there are 8 solver selection and each of the solver has slightly different options. For this thesis *"ga - Genetic Algorithm"* solver will be used. In *"Options"* section, all of the genetic algorithm related parameters can be selected. General parameter description of genetic algorithm is mentioned in Chapter 2 and will be omitted here. Third section is the *"Quick Reference"* and it guides the user for each of the drop-down menus in the toolbox.

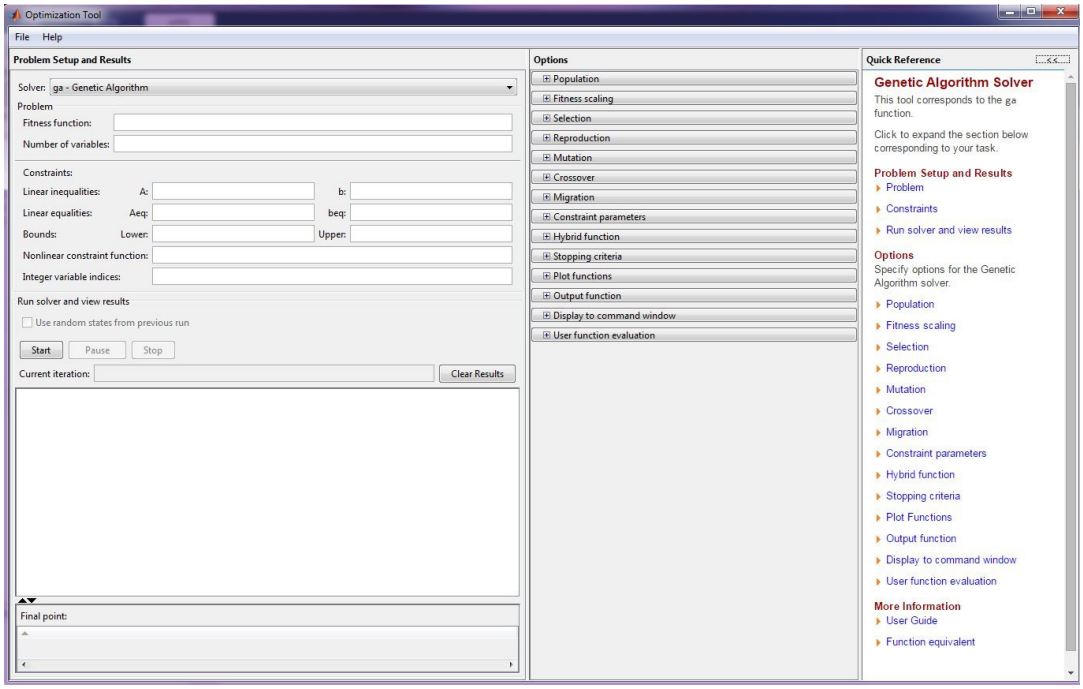

Figure 4.1 : MATLAB optimization toolbox.

MATLAB graphical user interface has a genetic algorithm engine running in the background using internal m-files. All of the functionality of the graphical user interface can also be obtained by using user coded scripting and m-files. In this thesis m-files are used because it is more flexible and easy to use with complex problems.

4.2 Optimization Problem

Classical approach to a flyback converter design is as following: there are some design requirements, such as maximum input voltage or output voltage ripple. By using these requirements and some assumptions designer calculates required components. Resulted values may be 615 μF of electrolytic output capacitor or 0.123 mm of air gap in transformer. After this point designer goes to a online component sales store and looks for close enough component values. Available values for this example may be 680 μF electrolytic output capacitor and 0.15 mm air gapped and lower inductance

valued transformer. Then recalculates the converter operating point with the selected components. If everything goes right, results are in required specifications, but if not there must be more iterations.

This design approach can be referred as start to finish design. But with the genetic algorithm used in this thesis, design approach is completely reversed and can be referred as finish to start design.

In genetic algorithm, manufactured or manufacturable components are selected before calculations. Manufactured term is used for discrete, commercially available components while manufacturable term is used especially for transformer. Because manufacturable air gap distances are discrete and also required turns should be fit in the core. By the selected components, converter's operating point is calculated. Then this operating point is checked with the desired requirements. For example, with the selected 470 μF output capacitor, obtained output ripple voltage is more than ten percent. If design requirement is five percent, this converter is marked as non feasible. And other circuits are evaluated in the same manner. Since all the calculations and constrain checks are performed in computer, very fast results are obtained.

For this thesis there are 20 variables for genetic algorithm to optimize hence problem of merit is a 20 dimensional optimization problem. Each of the variables have an effect on the total converter cost. 15 of the 20 variables for the optimization problem are commercially available discrete components. Because of this, component parameters are taken from datasheets and price values are taken from Digikey website. Digikey is one of the largest electronic component supplier around the world. Website has also multiple price offering almost for every component, decreasing price value for wholesale. For the component price database, lowest possible price value is taken from Digikey. For example for STN1NK60Z MOSFET, single unit price is 0.81$. For same MOSFET, when 28,000 units are purchased, unit price decreases down to 0.27507$ as seen in Table 4.1. For every component, maximum bulk purchasing number changes. While some of the components have maximum 20,000 units in bulk purchasing, others have 2000 units.

20 variables that used in optimization problem are given below along with their explanations:

Table 4.1 : Digikey price breakdown example.

Price Break	Unit Price[$]	Extended Price[$]
1	0.81	0.81
10	0.72	7.2
10	0.56310	56.31
500	0.42588	212.94
1000	0.34747	347.37
4000	0.30881	1,235.22
8000	0.28978	2,318.20
12000	0.28026	3,363.12
28000	0.27507	7,701.96

1. **Bridge Diode** is the AC line rectifier diodes. In algorithm, for full bridge rectifier to be symmetrical, four diodes are selected as same diodes. So for four of the diodes, single optimization variable used. When price is calculated in fitness function, single unit price is multiplied by four.

2. **DC Bus Capacitor** is the bulk storage capacitor. Rectified line voltage is stored in the high voltage electrolytic capacitor. For this thesis, DC Bus capacitor is assumed to be a single capacitor. Parallel usage is omitted. So single unit price is considered.

3. **Snubber Capacitor** is selected as high voltage ceramic capacitor. Single unit price is considered.

4. **Snubber Resistor** is selected between a range of values. Starting from 100 ohms to 22 mega ohms. Single unit price is considered.

5. **Snubber Diode** is selected as a high voltage breakdown diode. Single unit price is considered.

6. **Transformer Core and Coil Former**. Transformer core is selected to be a EE core from TDK and Cosmo Ferrites companies. The size range is starting from E13 core and ends with E25 core. Different core materials are also available. Materials are selected for 50 kHz to 250 kHz optimum frequency range because algorithm gives result in this frequency range. Coil formers are generally specific to core size. For example for E16 core there are only two coil formers, namely, vertical and horizontal orientation. Calculations are based on vertical coil formers. So each core in database is only applicable with a single coil former. Because of one to one

relation between core and coil former its is given in a single entry in the database. Price value is given as the sum of both.

7. **Air gap of Core** is the transformer air gap. Although in theory, it can be any value, from the production point of view, it is discrete valued because of insulation material thickness. Air gap is selected in a range between 0.05mm to 0.3mm to keep fringing flux effect at the minimum. There is no direct price value related to air gap. But its effect is seen in the transformer total price through core size, turns numbers etc.

8. **Transformer Primary Wire** is the selected wire for primary winding of the transformer. Wire database is created from 24 AWG (0.5 mm diameter) to 32 AWG (0.2 mm diameter). All of the wires are enamelled copper magnet wire for transformer winding. Price value is given as price per meter value.

9. **Transformer Primary Turns** is the number of turns for the primary winding. Primary turns range is selected between 70 and 130 turns. There is no direct price value related to turns number. Turns number affects total price through total primary wire length.

10. **Transformer Secondary Wire** is the selected wire for secondary winding of the transformer. Wire is selected from same database as primary winding. Same explanations are valid.

11. **Transformer Secondary Turns** is the number of turns for the secondary winding. Secondary turns range is selected between 2 to 20 turns. There is no direct price value related to turns number. Turns number affects total price through total secondary wire length.

12. **Transformer Secondary Parallel Wire Number** is the number of parallel secondary winding. Paralleling is used for obtaining low AC power loss and reduce current density in the secondary winding. There is no direct price value related to parallel wire number. Its effect is seen as multiplication of total wire price

13. **MOSFET** is the main switch of the converter. It is selected between high voltage, 500V to 900V MOSFETs. Single unit price is considered.

14. **Secondary Diode** is the uncontrolled switch of the converter. It is selected between schottky diodes. Single unit price is considered.

15. **Secondary Capacitor** is the secondary side bulk storage capacitor. Secondary capacitor is assumed to be an electrolytic capacitor. Single unit price is considered but it is multiplied by the parallel capacitor number.

16. **Secondary Parallel Capacitor Number** is the number of parallel secondary capacitors. This methodology is used for providing ripple current, ripple voltage constraints while minimizing cost. Selected capacitor by former variable is multiplied by this variable.

17. **Shunt Resistor** is the primary current sense resistor for current mode control. Single unit price is considered.

18. **Controller** contains the necessary Flyback control circuitry along with MOSFET driver. Selection of controller decides the switching frequency and shunt resistor sensing voltage. Single unit price is considered.

19. **TL431** is the shunt regulator on the secondary side of the circuit. TL431 provides a reference voltage and error amplifier for control circuitry. Single unit price is considered.

20. **Optocoupler** is the isolation barrier in the feedback circuitry. Optocoupler transfers the error signal from secondary to the primary side. Single unit price is considered.

Parameters that are taken from datasheets for each component is described in the next section.

4.3 Component Database

One of the most important points for genetic algorithm to perform is the creation of an extensive component database. Component database determines the search space. For each type of component, such as MOSFET, resistor or diode, spreadsheet of commercially available components is created. For each component, important parameters and price information is added to the spreadsheet. For each type of component, spreadsheet format and two example components are given in this section.

Table 4.2 : Database example for diode.

Info	ID	V_{fwd}	R_d	V_{rvs}	T_j	I_p	I_{avg}	I_{RMS}	R_{JA}	is Fast	Price
Vishay S1K	1	0.65	0.25	800	150	30	1	-	85	0	0.04
STPS2L60	2	0.43	0.06	60	150	75	2	10	25	1	0.0616

In Table 4.2, spreadsheet format for diodes is given. Spreadsheet contains high voltage line rectifier diodes and secondary side low voltage, fast rectifier schottky diodes. *Info* is used for human readable component information. *ID* is used for MATLAB algorithm to more easily identify components. V_{fwd} is the forward voltage drop of diode. R_d is the diode resistance. V_{rvs} is maximum reverse breakdown voltage. T_j is the maximum junction temperature. I_p is the non-repetitive maximum current. I_{avg} is the maximum average current. I_{RMS} is the maximum RMS current. R_{JA} is the junction to ambient thermal resistance. *is Fast* is the parameter to check whether a component is Schottky diode or not. *Price* is the price of component.

Table 4.3 : Database example for electrolytic capacitor.

Info	ID	C	V_{max}	I_{ripple} @100kHz	k_{RC} @120Hz	tan$f$	Price
CKE450V4u7	1	4.7μF	450	0.108	0.4	0.2	0.27668
KXM10V1000u	2	1000μF	10	1.25	0.6	0.19	0.1318

In Table 4.3, spreadsheet format for electrolytic capacitors is given. Spreadsheet contains high voltage DC bus electrolytic capacitors and secondary side low voltage electrolytic capacitors. C is the capacitance value in μF. V_{max} is the rated voltage. I_{ripple} *@100kHz* is the permissible ripple current value at 100 kHz. k_{RC} *@120Hz* is the ripple current constant to find permissible ripple current at 120 Hz. $tanf$ is the dissipation factor.

Table 4.4 : Database example for ceramic capacitor.

Info	ID	C	V_{max}	tan$f$	Price
AVX	1	10 nF	1000	0.025	0.08211
YageoCC	2	1 nF	2000	0.025	0.04761

In Table 4.4, spreadsheet format for high voltage ceramic capacitors is given. Parameter explanations are same as the Table 4.3 explanations.

In Table 4.5, spreadsheet format for MOSFETs is given. $R_{ds_{on}}$ is the maximum drain source resistance in conduction. C_{oss}@25V is the output capacitance of the MOSFET at 25 V drain source voltage in picofarads. Qgate is the gate charge in nano coulombs.

Table 4.5 : Database example for MOSFET.

Info	ID	$R_{ds_{on}}$	C_{oss}@25V	Qgate	V_{max}	T_j	I_{DC}@100C	I_p	R_{JA}	Price
AOD1N60	1	9	14.5	8	600	150	0.8	4	55	0.1944
FQD1N80	2	20	20	7.2	800	150	0.63	4	110	0.1944

V_{max} is the drain source breakdown voltage. I_{DC} *@100C* is the maximum permissible DC current at 100 Celsius degree. I_p is the maximum peak current.

Table 4.6 : Database example for resistor.

Info	ID	R	V_{max}	P_{max}	Price
100R	1	100	200	0.5	0.0057
390k	1	390000	200	0.5	0.0049

In Table 4.6, spreadsheet format for resistors is given. R is the resistance value. V_{max} is the maximum allowed voltage between leads of the resistor. P_{max} is the maximum permissible power loss for resistor.

Table 4.7 : Database example for transformer core.

Info	ID	A_L	A_e	B_{sat}	Volume	$P_{C_{loss}}$	CFW_{in}	CFW_{out}	CFH	Price
E16B66307	1	1000	20.1	0.25	7.56E-7	600	6.3E-3	1.09E-2	1E-2	0.49418
E25B66317	2	1950	52.5	0.27	3.02E-6	600	7.9E-3	1.7E-2	1.7E-2	0.59432

In Table 4.7, spreadsheet format for transformer cores and coil formers are given. A_L is the inductance factor in nH per turn. A_e cross sectional area of the core in mm^2. B_{sat} saturation flux density of core in T. *Volume* is the total core volume in m^3. It is used for core loss calculations. $P_{C_{loss}}$ is the core power loss data in watts per m^3. CFW_{in} is the coil former inner width value in meter. CFW_{out} is the coil former outer width value in meter. *CFH* is the coil former height value in meter.

Table 4.8 : Database example for wire.

Info	ID	Area	Diameter	Resistance [Ω/m]	Price
24AWG	1	1.96E-7	0.5	0.08706	0.12622
30AWG	2	4.91E-8	0.25	0.3482	0.38706

In Table 4.8, spreadsheet format for magnet wires is given. *Area* is the cross sectional area of the conductor in m^2. *Diameter* is the wire diameter in mm. *Resistance [Ω/m]* is the resistance value of the wire in ohm per meter. *Price* value for wire is given as per meter value.

In Table 4.9, spreadsheet format for controller is given. f_{sw} is the nominal switching frequency of the controller IC. V_{Isense} is the voltage value for current shunt resistor

Table 4.9 : Database example for controller.

Info	ID	f_{sw}	V_{Isense}	D_{max}	$V_{cc_{min}}$	Price
NCP1200	1	100000	0.9	0.8	10.3	0.3565
NCP1252	2	60000	1	0.45	13.1	0.4944

sense pin. D_{max} is the maximum duty cycle value allowed by IC as percent. $V_{cc_{min}}$ is the minimum supply voltage for IC to start operating. It is used for auxiliary winding calculations.

Table 4.10 : Database example for shunt resistor.

Info	ID	R	P_{max}	Price
1R	1	1	0.25	0.01904
4R7	2	4.7	1	0.102

In Table 4.10, spreadsheet format for shunt resistor is given. Parameter explanations are same as the Table 4.6 explanations.

Table 4.11 : Database example for TL431.

Info	ID	V_{ref}	Price
Onsemi	1	2.495	0.05005
ST	2	1.24	0.1275

In Table 4.11, spreadsheet format for TL431 IC is given. V_{ref} is the reference voltage of the internal regulator of the TL431.

Table 4.12 : Database example for optocoupler.

Info	ID	CTR_{min}	CTR_{max}	Price
TCP817b	1	130	260	0.06503
FOD817d	2	300	600	0.10696

In Table 4.12, spreadsheet format for optocoupler is given. CTR_{min} is the minimum current transfer ratio, CTR_{max} is the maximum current transfer ratio.

Once the component database spreadsheet format is completed, expanding the database is a trivial work. For this thesis, numbers of components in each component type is given in Table 4.13. Using the components in database there are 91×10^{24} possible flyback converter design most of which are not feasible converters. But genetic algorithm is able to find feasible converters and eventually a cost optimized solution.

Table 4.13 : Number of each component type in database.

Component Type	Number of Components
Diode	52
Electrolytic capacitor	52
Ceramic capacitor	42
MOSFET	27
Resistor	48
Ferrite Core	11
Wire	9
Controller	39
Shunt resistor	19
TL431	8
Optocoupler	10

After creating an adequate component database, an interface between MATLAB and spreadsheet is written. For each type of component in the spreadsheet, a MATLAB class is defined. An example of a MATLAB component class is Code Snippet 4.1.

In the code snippet, all of the parameters in the spreadsheet are defined as properties. In the methods section a constructor function is written to initialize component. In the *calcPowerLoss* function, power loss calculation is implemented for a diode using input parameters of average and RMS current values.

Using a similar approach, systematical component parameter storage and necessary component calculations can be made easily. In Appendix A, all of the class definitions are given.

4.4 Initialization

In MATLAB, genetic algorithm is implemented with two main code line. First one is the genetic algorithm options and second one is the algorithm itself. There are lots of coding embedded in the MATLAB for genetic algorithm toolbox. With the help of toolbox, development time can be decreased. But before jumping into genetic algorithm explanations itself, there needs to be some initialization parameter explanations.

Code Snippet 4.1: Example of class definition for diode.

```
classdef Diode  %diode class
    properties
        ID
        Vfwd
        Vrvs
        R
        Tj
        Ipeak
        Iavg
        Irms
        Rja
        isSMD
        isFast
        Price
    end

    methods  %constructor
        %get all the parameters in the class properties
        function obj = Diode(ID, Vfwd, R, Vrvs, Tj, Ipeak, Iavg, Irms, Rja, ...
                isSMD, isFast, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Vfwd = Vfwd;
                obj.R = R;
                obj.Vrvs = Vrvs;
                obj.Tj = Tj;
                obj.Ipeak = Ipeak;
                obj.Iavg = Iavg;
                obj.Irms = Irms;
                obj.Rja = Rja;
                obj.isSMD = isSMD;
                obj.isFast = isFast;
                obj.Price = Price;
            end
        end

        function powerLoss = calcPowerLoss(obj, Iavg, Irms)
            powerLoss = obj.Vfwd * Iavg + obj.R*Irms*Irms;
        end
    end
end
```

4.4.1 Library parameter initialization

Firstly, libraries that will be used for choosing genetic algorithm variables are initialized. In Code Snippet 4.2 commercially available discrete component library variables are initialized. In Code Snippet 4.3, user ranged library variables are initialized.

Code Snippet 4.2: Commercial component library parameter initialization.

```
%------------------------- GA design variables ---------
%component library variables
global Diodes;
global CapacitorELs;
global CapacitorCERs;
global Resistors;
global Cores;
global MOSFETs;
global Wires;
global Shunts;
global Controllers;
global TL431s;
global Optos;
```

Code Snippet 4.3: User ranged library parameter initialization.

```
%discrete numbered variables
global Nprimarys;  %primary transformer turns range
global Nsecondarys;  %secondary transformer turns range
global Lgaps;  %transformer gap distances in mm
global SecondaryCapNumb;  %secondary parallel capacitor number
global NsecondaryParallel;  %transformer secondary parallel windings

Nprimarys = 70:1:130;
Nsecondarys = 2:1:20;
Lgaps = 0.05:0.05:0.3;
SecondaryCapNumb = 1:3;
NsecondaryParallel = 1:4;
```

4.4.2 User specific value initialization

Secondly, flyback converter design parameters that are supplied by user are initialized. All of the variables are design inputs that affects the genetic algorithm's final result. For example *vOut* and *iOut* parameters directly affects the output power of the converter. In Code Snippet 4.4, these variables are given.

Code Snippet 4.4: Flyback converter design specification variables.

```
global vOut;  %output voltage
global iOut;  %output current
global outputRipple;  %output voltage ripple
global inputRipple;  %input voltage ripple
global snubberRipple;  %snubber capacitor
global Kl;  %number of outputs
global vLineMin;  %minimum AC input voltage
global vLineMax;  %maximum AC input voltage
global fLine;  %line frequency
global primaryWireCurrentDensity;
global secondaryWireCurrentDensity;
global nEstimated;  %estimated efficieny
global deratingCoef;  %derating coefficient
global tAmbient;  %maximum ambient temperature
```

4.4.3 Component library initialization

Thirdly, component database is imported from excel file into MATLAB. Coding of this stage is trivial and requires file and matrix handling. Because of this, component library initialization codes are given in Appendix A

4.4.4 Genetic algorithm initialization

Lastly, in the Code Snippet 4.5, genetic algorithm options are given. *opts_populationSize* defines the population size for each generation. *opts_maximumGenerations*, is the maximum allowed generations unless genetic algorithm is terminated by another condition. *opts_maxStallGenerations* is the number of generations when the algorithm is terminated if there is no predetermined value of fitness increase. *opts_eliteCount* is the number of elite individuals.

opts_crossoverFraction is the crossover ratio of the population. *PlotFcn* is used for different plotting options. Different plot functions and their outputs are given in Fig. 4.2 through Fig. 4.6.

Code Snippet 4.5: Genetic algorithm options.

```
opts_populationSize = 50;
opts_maxGenerations = 280;
opts_maxStallGenerations = round(opts_maxGenerations / 2);
opts_eliteCount = round(opts_populationSize*0.1);
opts_crossoverFraction = 0.8;

opts = optimoptions(@ga, ...
    'PopulationSize', opts_populationSize, ...
    'MaxGenerations', opts_maxGenerations, ...
    'MaxStallGenerations', opts_maxStallGenerations, ...
    'EliteCount', opts_eliteCount, ...
    'CrossoverFraction', opts_crossoverFraction, ...
    'FunctionTolerance', 1e-10, ...
    'PlotFcn', {@gaplotbestf, @gaplotdistance, @gaplotrange, @gaplotstopping, ...
    @gaplotscores}, ...
    'OutputFcn', @myoutputfcn);
```

To better explain the genetic algorithm option parameters in MATLAB, an example can be given. For example, if population size is 80, elite ratio is 0.1 and crossover ratio is 0.8;

- there are $80 \times 0.1 = 8$ elite children
- there are $80 - 8 = 72$ children other than elite children. $72 \times 0.8 = 57.6 \approx 58$ crossover children
- and there are $72 - 58 = 14$ mutation children

After the options are determined, genetic algorithm can be initialized. This is done by code line in Code Snippet 4.6. First line in Code Snippet is the random number generator feed. This is used for reproducibility of the results. third and forth line of the Code Snippet is the upper and lower boundary of the selectable component space. For all of the components lower bound is 1. For upper bound, each component type has different upper bound. Upper bound is the number of components for that type of component.

ga(...) function is the main function that runs the genetic algorithm. Arguments of the function is as following:

- *@priceOfConverter* is fitness function
- *length(lb)* is number of variables in the problem, which is 20 in this thesis
- four blank matrices are the linear equality and liner inequality constants

- lb and ub are the lower and upper boundaries of the variables
- *@constraints* is the constraints function for GA
- *1:length(lb)* is the range of integer variables in the GA problem
- *opts* is the options for GA given in Code Snippet 4.5

Code Snippet 4.6: Genetic algorithm initialization.

```
rng(1, 'twister'); %for reproducibility

lb = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
ub = [componentCounts(1) componentCounts(2) componentCounts(3) ...
    componentCounts(4) componentCounts(1) componentCounts(5)...
    componentCounts(12) componentCounts(7) componentCounts(13) ...
    componentCounts(7) componentCounts(14) componentCounts(16) ...
    componentCounts(6) componentCounts(1) componentCounts(2) ...
    componentCounts(15) componentCounts(8) componentCounts(9) ...
    componentCounts(10) componentCounts(11)];

    [xbestDisc, fbestDisc, exitflagDisc, output, population, score] = ...
    ga(@priceOfConverter, length(lb), [], [], [], [], lb, ub, ...
    @constraints, 1:length(lb), opts);
```

4.5 Fitness (Cost) Function

Fitness or cost function is the function that is subject to minimization problem. In this thesis, the minimization problem is the minimization of the converter total cost. In Code Snippet 4.7, cost of converter is calculated. *x* variable is the matrix of selected component IDs. Since *x* matrix contains integer IDs, it it should be converted into a matrix that contains selected component prices. *mapPrices(...)* function does this conversion. Price mapping function is given in Code Snippet 4.8.

Code Snippet 4.7: Fitness (cost) function.

```
function cost = priceOfConverter(x)
global flyback;

% Map the discrete variables
x = mapPrices(x);

%cost of the converter
cost = 4*x(1) + x(2) + x(3) + x(4) + x(5) + x(6) + x(8)*flyback.MLTPrim*x(9)+ ...
       x(10)*flyback.MLTSec*x(11)*x(12) + flyback.transNaux*flyback.MLTAux*...
       flyback.transWireAux.Price + x(13) + x(14) + x(15)*x(16) + x(17) + ...
       x(18) + x(19) + x(20);
```

Content of *x* matrix and price calculations are given in Section 4.2.

4.6 Penalty (Constraints) Function

Penalty function is the function that calculates the feasibility of the randomly generated individual. In MATLAB, it is called constraints function. For every

Code Snippet 4.8: Price mapping function.

```
function x = mapPrices(x)

x(1) = Diodes(x(1)).Price; %Bridge rectifier diode
x(2) = CapacitorELs(x(2)).Price;  %DC bus electrolytic capacitor
x(3) = CapacitorCERs(x(3)).Price;  %Snubber ceramic capacitor
x(4) = Resistors(x(4)).Price;  %Snubber resistor
x(5) = Diodes(x(5)).Price;  %Snubber diode
x(6) = Cores(x(6)).Price;  %Transformer core
x(8) = Wires(x(8)).Price;  %Primary winding
x(9) = Nprimarys(x(9));  %Primary turns
x(10) = Wires(x(10)).Price;  %Secondary winding
x(11) = Nsecondarys(x(11));  %Secondary turns
x(13) = MOSFETs(x(13)).Price;  %MOSFET
x(14) = Diodes(x(14)).Price;  %Secondary diode
x(15) = CapacitorELs(x(15)).Price;  %Secondary output capacitor
x(17) = Shunts(x(17)).Price;  %Shunt resistor
x(18) = Controllers(x(18)).Price;  %Controller
x(19) = TL431s(x(19)).Price;  %TL431
x(20) = Optos(x(20)).Price;  %Optocoupler
```

generated individual throughout the genetic algorithm operation, constraints function is calculated to check feasibility of the solution.

In Code Snippet 4.9, input argument, x, contains necessary information of IDs of an individual. Genotype-phenotype matching is done in line 4 to line 23 for each component. Also an example of genotype-phenotype mapping is shown in Table 4.14. IDs are the genotype in this example. Using phenotype matching, every ID is turned into an electronics component such as given in column three and four.

Table 4.14 : Example of parameter breakdown of an individual.

GA Variable Number	ID	Explanation	Selected Component
1	42	Bridge diode	SMC S2J
2	30	DC bus capacitor	Illinois CKE 450V 22μF
3	25	Snubber capacitor	AVX 1kV 3.3nF
4	23	Snubber resistor	1210 205 kOhm
5	37	Snubber diode	SMC S5M
6	7	Transformer core and coil former	TDK E20 66311N87
7	4	Airgap of core	0.2
8	9	Transformer primary wire	32 AWG
9	5	Transformer primary turns	74
10	4	Transformer secondary wire	27 AWG
11	5	Transformer secondary turns	6
12	2	Secondary parallel wire number	2
13	7	MOSFET	ST N1NK60Z
14	20	Secondary diode	Vishay SS3P6
15	20	Secondary capacitor	Illinois KXM 6V3 560μF
16	2	Secondary parallel capacitor number	2
17	4	Shunt resistor	2R
18	6	Controller	ONSemi NC1253 100kHz
19	3	TL431	Diodes TL431
20	2	Optocoupler	TaiwanSemi TCP817D

Code Snippet 4.9: Penalty (constraint) function.

```
function [c, ceq] = constraints(x)

%Get all data from GA calculated array
diodeBridge = Diodes(x(1));  %Bridge diode
capDCBus= CapacitorELs(x(2));  %DC bus electrolytic capacitor
capSnubber = CapacitorCERs(x(3));  %Snubber ceramic capacitor
resSnubber = Resistors(x(4));  %Snubber resistor
diodeSnubber = Diodes(x(5));  %Snubber diode
trCore = Cores(x(6));  %Transformer core
trLgap = Lgaps(x(7));  %Airgap
trWirePrim = Wires(x(8));  %Primary wire
trNprim = Nprimarys(x(9));  %Primary turns
trWireSec = Wires(x(10));  %Secondary wire
trNsec = Nsecondarys(x(11));  %Secondary turns
nSecParal = NsecondaryParallel(x(12));  %Transformer secondary parallel wire
mosfet = MOSFETs(x(13));  %MOSFET
diodeSec = Diodes(x(14));  %Secondary diode
capSec = CapacitorELs(x(15));  %Secondary output capacitor
secCapNum = SecondaryCapNumb(x(16));  %Secondary Capacitor Number
resShunt = Shunts(x(17));  %Shunt resistor
controller = Controllers(x(18));  %Switching frequency
tl431 = TL431s(x(19));  %Shunt regulators
trWireAux = Wires(9);  %Auxiliary wire
%find operating point of the converter with the GA selected components &
%design parameters
opts = [
    vOut %V
    iOut %A
    outputRipple; %percent
    Kl %single output
    vLineMin %V
    vLineMax %V
    fLine %hertz
    nEstimated %initial estimated efficiency
    deratingCoef %component derating coef
    tAmbient; %ambient temperature
    ];
```

After genotype-phenotype matching, operating point with the selected components of a flyback converter is found. For this calculations, an m-file is written which calculates operating point of flyback converter with the equations given in Chapter 3. These calculations are made for each individual created by genetic algorithm. For example a 50 population and 100 generation genetic algorithm problem, 5000 flyback converter is calculated. From the microprocessor usage point of view, this calculations are the most expensive calculations for algorithm.

It can be thought as realizing a converter with the genetic algorithm selected components and testing it under prerequisite conditions. For example a DC bus capacitor may be selected as 35V rated component by the genetic algorithm. But desired maximum input voltage may be 265V RMS. In this condition, maximum DC bus voltage is $265 \times \sqrt{2} = 374\,V$. So, selected component can not operate in this condition and fails. Hence even if the other components are within their maximum ratings, converter (individual) is not a feasible solution.

Another example can be given for transformer. There are 7 parameters in genetic algorithm related to the transformer, namely:

1. Core size & material
2. Air gap length
3. Number of primary turn
4. Primary wire diameter
5. Number of secondary turns
6. Secondary wire diameter
7. Number of secondary parallel wires

Selection of any number or component for each of the variables results in a transformer. But this transformer may be, and most likely, will not feasible for the circuit in hand.

To check feasibility of every selected component, constraints are divided into 5 categories, namely:

- Voltage constraints
- Current constraints
- Power loss constraints
- Junction temperature constraints
- Other constraints

4.6.1 Voltage constraints

In Code Snippet 4.10, voltage constraints of all of the components is given. For each type of converter there is a breakdown voltage level. In component database, breakdown voltage levels are given. MATLAB code compares circuit operating point voltage and the selected component's maximum permissible voltage. If the *voltages[...]* matrix elements are all negative valued, then voltage constraints are satisfied.

To provide a safety margin, a derating term is used for each component. Generally, 10 percent of safety margin is used. For example, a 700V rated MOSFET is considered as $700 \times 0.9 = 630V$ MOSFET to give safety margin.

Code Snippet 4.10: Voltage constraints.

```
%circuit operating point - selected component max voltage
voltages = [
    flyback.vDiodeRecMax - diodeBridge.Vrvs*deratingCoef; ...
    flyback.vCapDCBusMax - capDCBus.Vmax*deratingCoef; ...
    flyback.vCapDCBusRipple - vLineMin*sqrt(2)*inputRipple; ...
    flyback.vCapSnubberMax - capSnubber.Vmax*deratingCoef; ...
    flyback.vCapSnubberRipple - flyback.vCapSnubberMax * snubberRipple; ...
    flyback.vDiodeSnubberMax - diodeSnubber.Vrvs*deratingCoef; ...
    flyback.vResSnubberMax - resSnubber.Vmax*deratingCoef; ...
    flyback.vMosfetDSMax - mosfet.Vrvs*deratingCoef; ...
    flyback.vDiodeSecMax - diodeSec.Vrvs*deratingCoef; ...
    flyback.vCapSecMax - capSec.Vmax * deratingCoef; ...
    flyback.vCapSecRipple - vOut * outputRipple; ...
    controller.VcurrentSense - flyback.vResShunt; ...
    flyback.vResShunt - controller.VcurrentSense*1.2; ...
    ];
```

4.6.2 Current constraints

Same approach is applied in current constraints as in voltage constraints. Apart from component rated value checks, two additional current constraints are added to check if the ripple current is no more than two times of the average current. Since this violation yields in negative current in primary side. Other additional current constraint is the current density constraint for primary and secondary side of the transformer wires. For high number of turns as in primary wire, it is recommended to use 5 A/mm^2 and for secondary wire it is 6-10 A/mm^2 to minimize temperature increase [23].

Code Snippet 4.11: Current constraints.

```
currents = [
    flyback.iDiodeRecPeak - diodeBridge.Ipeak*deratingCoef; ...
    flyback.iDiodeRecAvg - diodeBridge.Iavg*deratingCoef; ...
    flyback.iCapDCBusHFRMS - capDCBus.Iripp; ...
    flyback.iCapDCBusLFRMS - capDCBus.Iripp*capDCBus.kRC120; ...
    flyback.iMosfetDSRMS - mosfet.Idc*deratingCoef; ...
    flyback.iMosfetDSPeak - mosfet.Ipeak*deratingCoef; ...
    flyback.iLmDelta / 2 - flyback.iEDC; ...
    flyback.iTransPrimCurrentDensity - primaryWireCurrentDensity; ...
    flyback.iTransSecCurrentDensity - secondaryWireCurrentDensity; ...
    flyback.iDiodeSecPeak - diodeSec.Ipeak*deratingCoef; ...
    flyback.iDiodeSecAvg - diodeSec.Iavg*deratingCoef; ...
    flyback.iCapSecRMS - capSec.Iripp; ... %no derating because of temp. coef
    ];
```

4.6.3 Power loss constraints

Power loss constraint are given for snubber resistor and shunt resistor. Also transformer loss ratio in total converter loss is added as to be no more than 30% of total loss. Semiconductor losses are calculated as maximum junction temperature in next section.

Code Snippet 4.12: Power loss constraints.

```
losses = [
    flyback.pLossResSnubber - resSnubber.Pmax*deratingCoef^2; ...
    flyback.pLossTransformer - (flyback.pIn - flyback.pOut) * 0.3; ...
    flyback.pLossResShunt - resShunt.Pmax*deratingCoef^2; ...
];
```

4.6.4 Junction temperature constraints

Semiconductor power losses are converted into junction temperature for diodes and MOSFET. Junction temperature is compared with the component's maximum junction temperature.

Code Snippet 4.13: Junction temperature constraints.

```
temperatures = [
    flyback.tDiodeBridgeJunc - diodeBridge.Tj*deratingCoef; ...
    flyback.tMosfetJunc - mosfet.Tj*deratingCoef; ...
    flyback.tDiodeSecJunc - diodeSec.Tj*deratingCoef;
];
```

4.6.5 Other constraints

Other constraints include operating saturation flux, control mechanism for secondary diode to be a fast diode, maximum duty cycle and to check if the windings fill in the coil former winding area.

Code Snippet 4.14: Other constraints.

```
otherConstraints = [
    flyback.bOperating - trCore.Bsat; ... %core saturation
    0.5 - diodeSec.isFast; ... %to check fast diodes
    flyback.dutyMax - controller.DutyMax; ... %controller dutymax check
    flyback.windingAreaFillFactor - 1;  %to check whether windings fit in core
    ];
```

In Code Snippet 4.15, all of the constraints are combined in a *c* matrix for MATLAB to handle more easily. For a flyback converter to be a feasible individual, all of the elements in the *c* matrix should be negative valued.

For this optimization problem, there is no equality constraints, so *ceq* matrix is returned as blank.

4.7 Output Function

Output function is the function that can be programmed to obtain information while genetic algorithm is running in MATLAB. This function is not a compulsory step of genetic algorithm but can be used for obtaining some insight of the algorithm's

Code Snippet 4.15: Penalty (constraint) function.

```
c = [voltages; currents; losses; temperatures; otherConstraints];

% No equality constraints
ceq = [];
```

progress. Output function is called each time when the generation number is increasing one step forward. For example, all of the population with their genotypes throughout the genetic algorithm can be stored in a spreadsheet using output function.

4.8 Plot Function

Plot functions are useful tools that shows the evolution of the genetic algorithm. There lots of predefined plotting functions in MATLAB but five of them are used in this thesis because of their usefulness. Full documentary of the plotting options can be found in Global Optimization Toolbox User's Guide [36]. Plotting functions that are used are these:

```
gaplotbestf
gaplotdistance
gaplotrange
gaplotstopping
gaplotscores
```

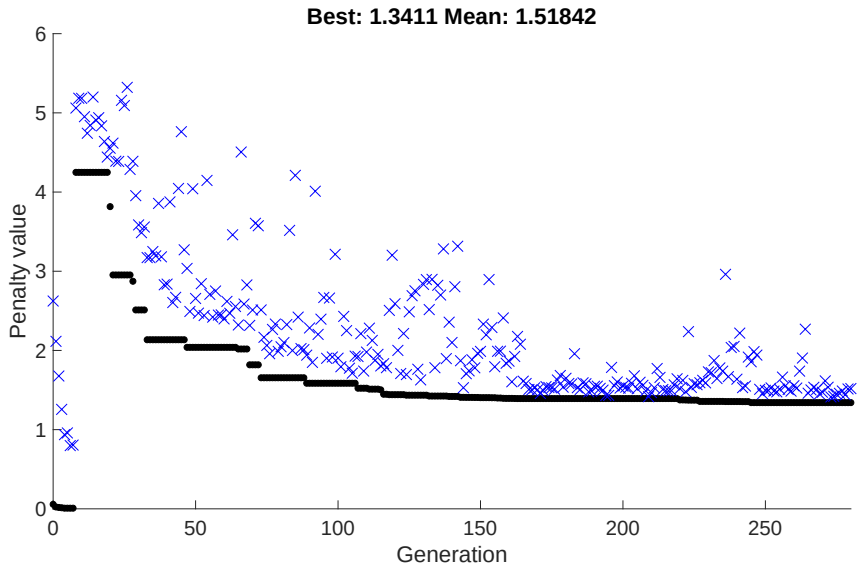

Figure 4.2 : Plot Function - evolution of best score.

In Fig. 4.2, evolution of the best score in the population is shown. Dots(line) shows the best individual while crosses shows the mean of population. At first, for a few generations of population, best and mean scores are very low and steps up after a

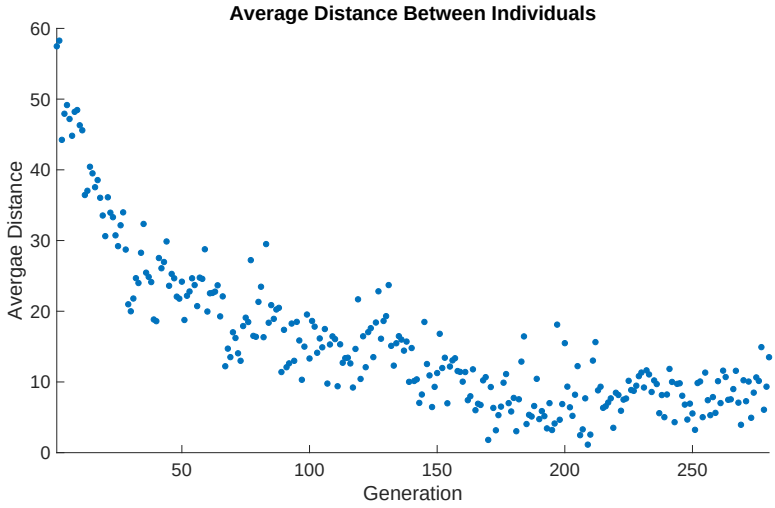

Figure 4.3 : Plot Function - average distance between individuals.

particular point. This is because of the feasibility check in the algorithm. Up to a point all of the individuals in the population are infeasible solutions. When an individual is not a feasible solution, MATLAB calculates its penalty score as maximum fitness value in the population and adds the constraint violations to it. Since there are lots of constraints for this optimization problem, penalty value seems to converge to zero for initial populations. But when a feasible individual is obtained, there is a step change in the penalty value. After that point population converges to a feasible minimum cost value.

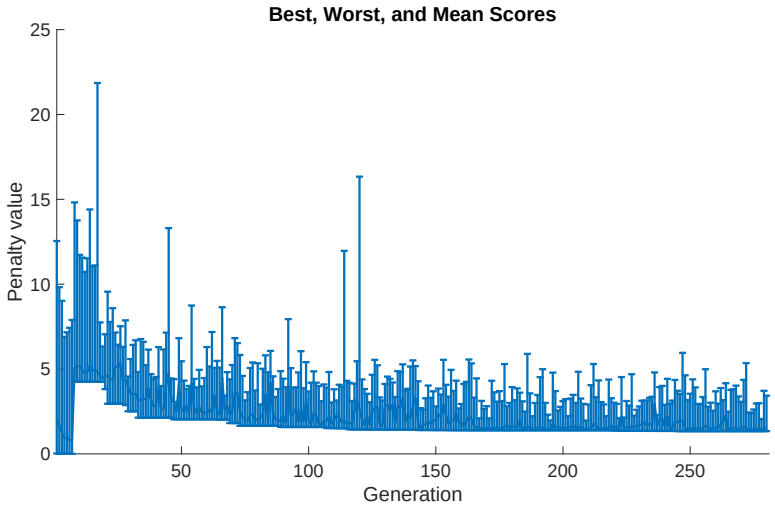

Figure 4.4 : Plot Function - best, worst and mean scores in population.

Fig. 4.3 shows the average distance between individuals. Diversity in initial populations is good since it increases the search space. But eventually diversity

decreases to converge a solution. To prevent stucking in local minimums, one of the best ways is to restart the algorithm for a few times.

Fig. 4.4 shows the evolution of best, worst and mean scores in the population. It can be seen that there is a slight decrease in the scores due to convergence. Also in some generations there are individuals that are mutant and have very high penalty values.

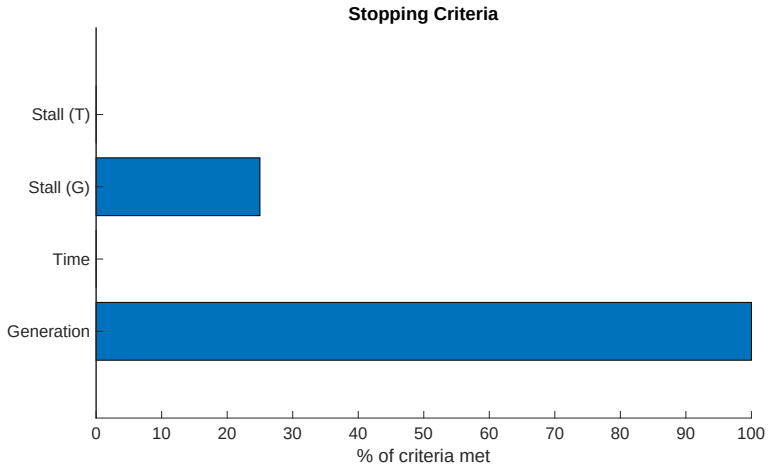

Figure 4.5 : Plot Function - tracking of the stopping criteria.

Fig. 4.5 shows the stopping criteria percentage. There can be four stopping conditions namely, stall time, stall generations, time and number of generations. In the plot, how much of the stopping criteria is met can be seen. For this example, algorithm is stopped because maximum number of generations is reached.

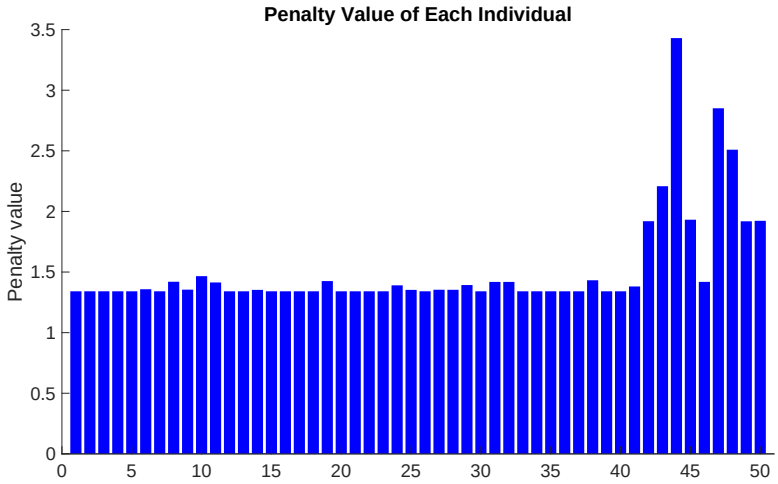

Figure 4.6 : Plot Function - score of each individual in population.

Fig. 4.6 show the scores of each individual in the population. For each generation this plot is updated throughout the genetic algorithm run.

Fig. 4.7 shows the price distribution in a converter. Particular figure is given for realized converter in Chapter 5. For each run of the genetic algorithm slightly different price bar can be obtained. But when it is investigated pricey components can be given as: DC bus capacitor, transformer core and coil former, MOSFET and controller.

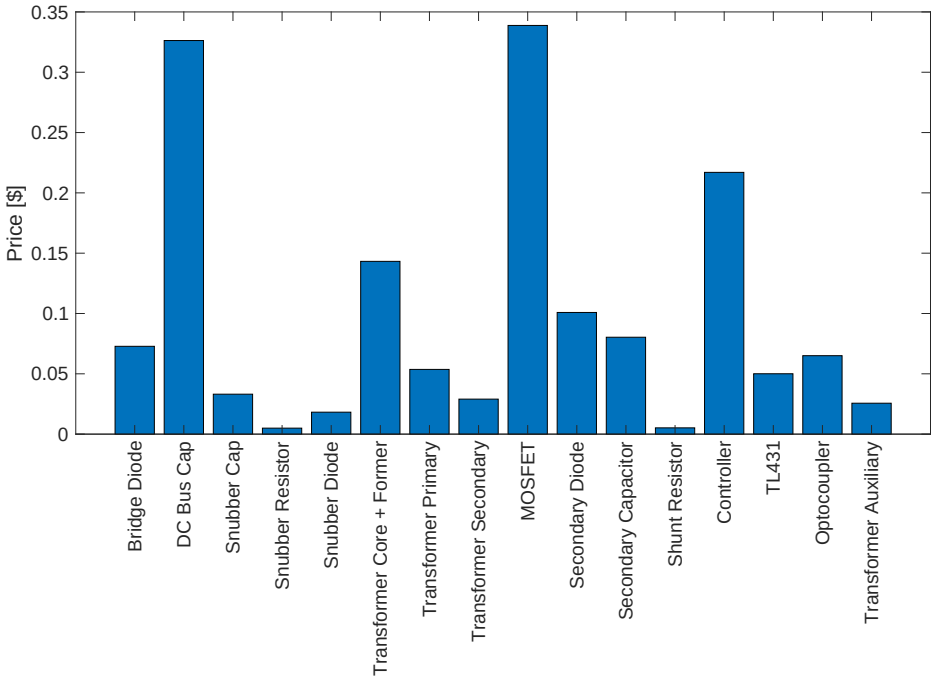

Figure 4.7 : Prices of the components in GA.

In Fig. 4.9, 3D representation of 200 GA generated flyback converter is shown. This 3D plot is generated by putting Fig. 4.7 one after another in the x-axis. Fig. 4.8 and Fig. 4.9 are particularly useful for explaining the purpose of the thesis. In Fig. 4.9, x-axis is the converter cost, y-axis is the name of the components and z-axis is the component cost. y-axis components are the components of the converter with a x-axis value converter. Peaks shows the pricey components.

Fig. 4.8 is the 2D representation of the same plot from upside down view. Yellow colors shows the higher priced converter while blue are cheaper. For example there is a yellow priced component as secondary capacitor with converter cost of around 3$. There are lots of blue components with higher total converter cost behind this converter. By selecting a higher priced component, there is a compromise for that component cost but when total converter cost is considered it has benefits.

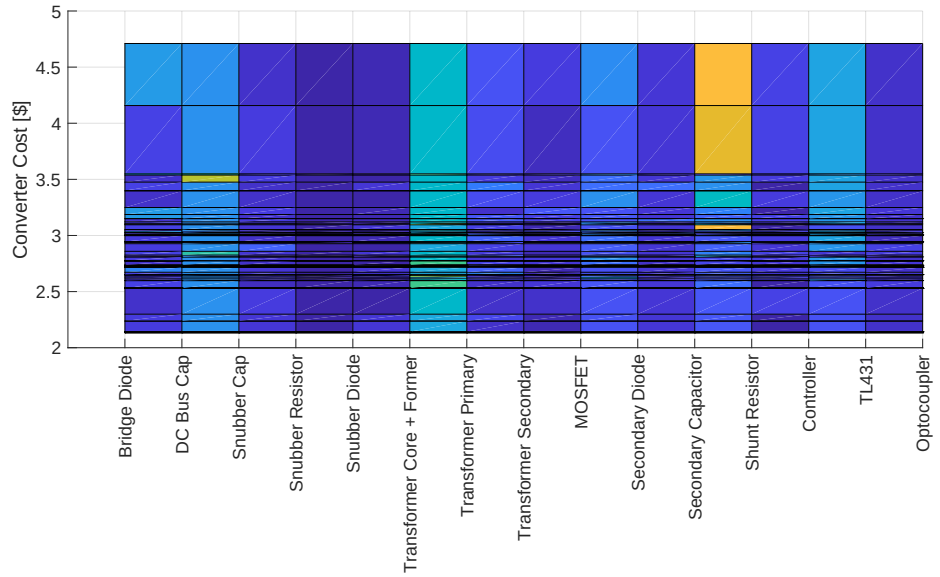

Figure 4.8 : 2D plot of converter and component costs.

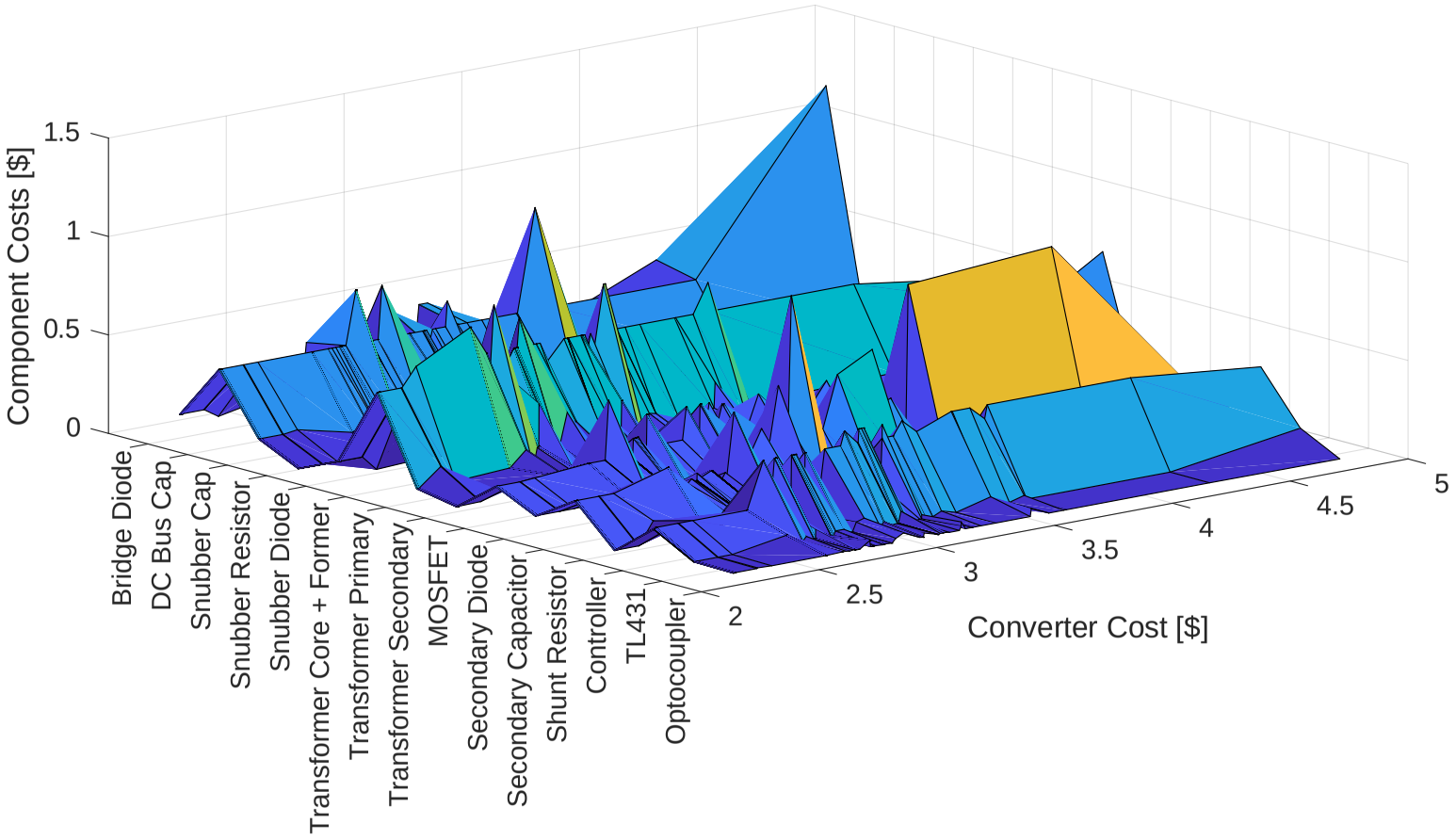

Figure 4.9 : 3D plot of converter and component costs.

5. REALIZATION OF CONVERTER

In this section realization of converter steps are described. Genetic algorithm selected components are used for the realization.

Specifications of the converter is given in Table 5.1. This specifications are selected to compare new design with a previously designed flyback converter by Power Integrations company.

Table 5.1 : Specifications of the designed flyback converter.

Parameter	Value	Unit
Output voltage	5	V
Output current	0.73	A
Output voltage ripple	4	%
DC Bus voltage ripple	30	%
Snubber voltage ripple	10	%
Number of outputs	1	-
Minimum input voltage RMS	85	V
Maximum input voltage RMS	265	V
Line frequency	50	Hz
Operating temperature	60	C^o

Full list of input specifications are given in Code Snippet 5.1.

Code Snippet 5.1: Designed converter specifications.

```
vOut = 5;  %V
iOut = 0.73;  %A
outputRipple = 0.04;  %ratio
inputRipple = 0.3;  %ratio
snubberRipple = 0.1;  %ratio
Kl = 1;
vLineMin = 85;  %Vac
vLineMax = 265;  %Vac
fLine = 50;  %hertz
primaryWireCurrentDensity = 5;  %A/mm^2
secondaryWireCurrentDensity = 10;  %A/mm^2
nEstimated = 0.7;  %ratio
deratingCoef = 0.9;  %ratio
tAmbient = 60;  %celcius
```

5.1 Selected Components by Genetic Algorithm

Selected components for Flyback converter by genetic algorithm is given in Table 5.2.

Although the main components are selected by genetic algorithm, before schematic drawing, controller IC datasheet should be investigated. Selected controller IC is

Table 5.2 : Selected components by genetic algorithm.

GA Variable Number	ID	Explanation	Selected Component	Price [$]
1	45	Bridge diode	SMC RS1J	0.0182
2	38	DC bus capacitor	Illinois CKS 450V 22μF	0.3263
3	40	Snubber capacitor	Samsung 630V 2.2nF	0.0331
4	13	Snubber resistor	1210 169kOhm	0.0049
5	44	Snubber diode	SMC RS1K	0.0182
6	10	Transformer core and coil former	Cosmo E16-1605-vert	0.1432
7	2	Airgap of core	0.1	-
8	9	Transformer primary wire	32 AWG	0.02/meter
9	31	Transformer primary turns	100	-
10	9	Transformer secondary wire	32 AWG	0.02/meter
11	10	Transformer secondary turns	11	-
12	4	Secondary parallel wire number	4	-
13	10	MOSFET	TSM1NB60	0.18926
14	33	Secondary diode	ONSemi MBRD330	0.1008
15	22	Secondary capacitor	Illinois KXM 16V 470μF	0.0803
16	1	Secondary parallel capacitor number	1	-
17	6	Shunt resistor	3R	0.0052
18	38	Controller	ONSemi NC12510 100kHz	0.217
19	1	TL431	ONSemi TL431	0.05
20	2	Optocoupler	TaiwanSemi TCP817D	0.065

NCP12510 from On Semiconductor. It has two subversions, namely, 100 kHz and 65 kHz switching frequency. 100 kHz version is selected by genetic algorithm. Although the controller's maximum switching frequency is 100 kHz, with the decreasing load current switching frequency also decreases down to 26 kHz. This action maintains the high efficiency while keeping voltage regulation in the desired level. Further decrease in load current puts IC into skipping mode. In this mode IC has no fixed switching frequency and tracks the feedback voltage. Also in this mode peak current level of current sense pin is also decreased. Start-up voltage is provided by high voltage DC bus side with high value resistors for IC. But within a few cycles, auxiliary winding provides required supply voltage for IC. NCP12510 has also over voltage and over power protection if desired. But in this thesis, these two feature is not used for simplicity.

In the realization part, selected MOSFET by the genetic algorithm was not available to purchase at that time. So another MOSFET is used for realization of the converter. The MOSFET that is used is 7NM70 by UTC. Simulations and realization is made using this component.

5.2 Simulation With the Selected Components

Flyback components that are selected by genetic algorithm are simulated using LTspice simulation tool. LTspice is a spice simulator designed by Linear Technology. It is easy to use and flexible in many ways. It has a component library for Linear Technology semiconductor devices and other manufacturers. Also user can manually add third party spice models and simulate with ease.

To simulate circuit, firstly, spice models of the semiconductor devices should be found or generated. These models are found and added to LTspice as third party models. Passives and transformer is modelled with the calculated parameters. In Fig. 5.1, power section of the circuit in LTspice model is shown. In Fig. 5.2, control section of the converter is shown.

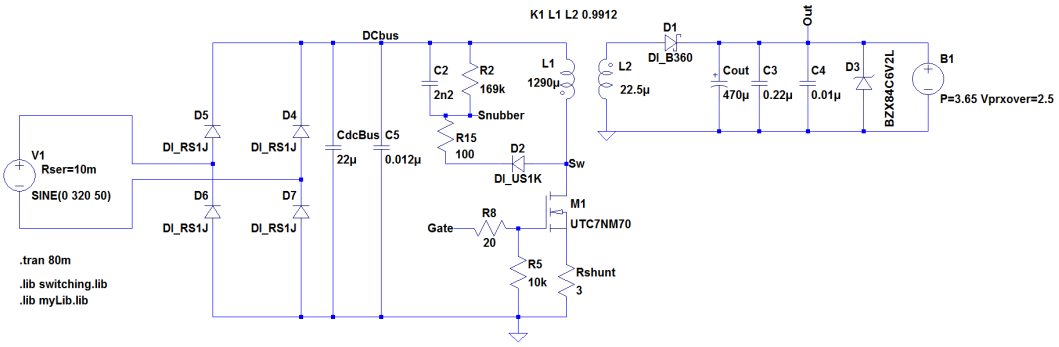

Figure 5.1 : LTspice model of the flyback converter power section.

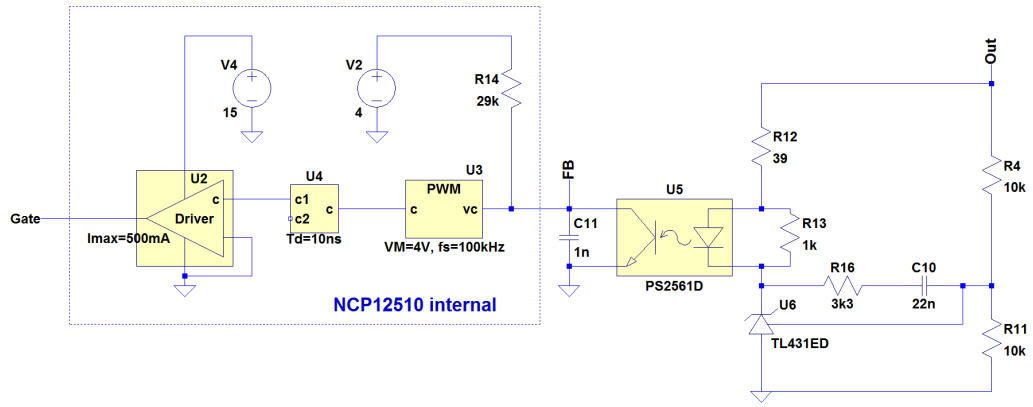

Figure 5.2 : LTspice model of the flyback converter control section.

Steady state simulation waveforms are given in Fig. 5.3 to 5.11.

Simulation results shows that DC bus ripple voltage is around 8 V. Which is similar with the calculations.

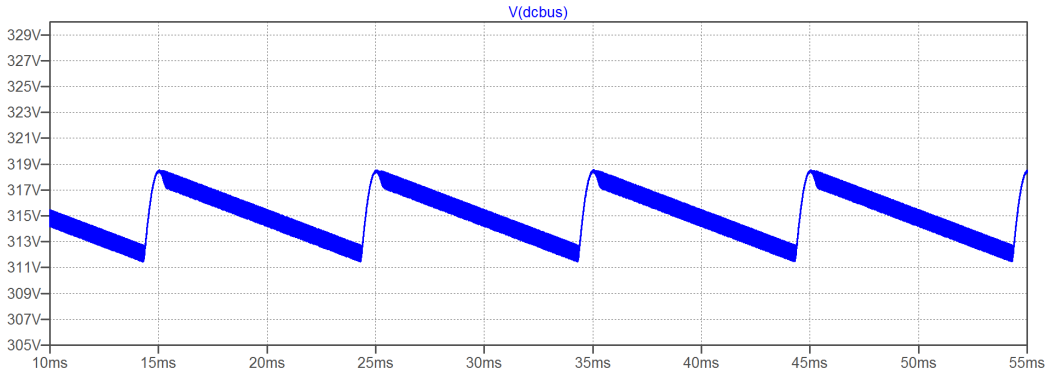

Figure 5.3 : DC bus ripple voltage waveform.

In Fig. 5.4, DC bus ripple current waveform is shown. Ripple current has two components, namely, high frequency current and low frequency current. Negative portion of the current in the Fig. 5.4 is the high frequency component. High frequency component of the current is due to SMPS operation. Low frequency component of the current is due to charging operation from mains line. In Fig. 5.5, high frequency component of the current can be seen more clearly.

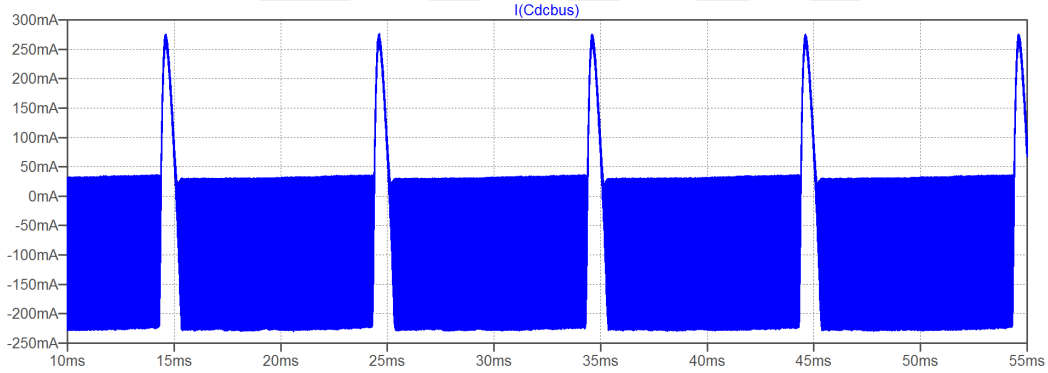

Figure 5.4 : DC bus capacitor ripple current waveform.

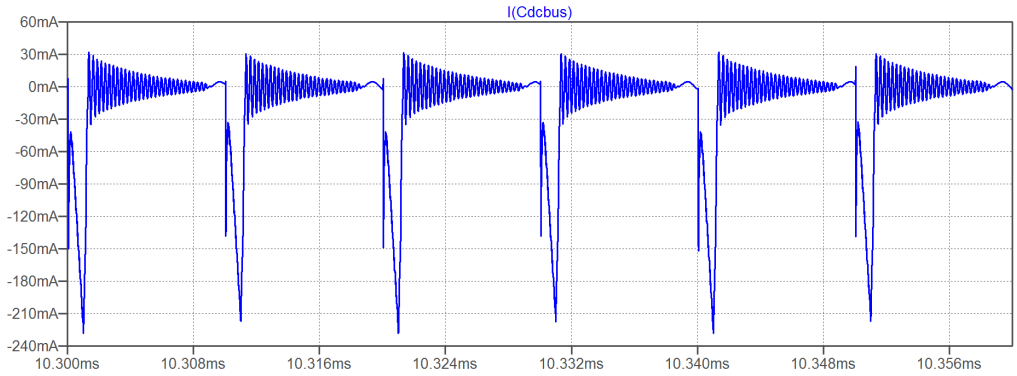

Figure 5.5 : DC bus capacitor ripple current high frequency component waveform.

In Fig. 5.6, MOSFET drain - source voltage waveform is shown. High frequency decaying sinusoidal waveform is due to parasitic components that are connected to the

switch node. High frequency resonance is the interaction between MOSFET's output capacitance, C_{oss}, and transformer leakage inductance.

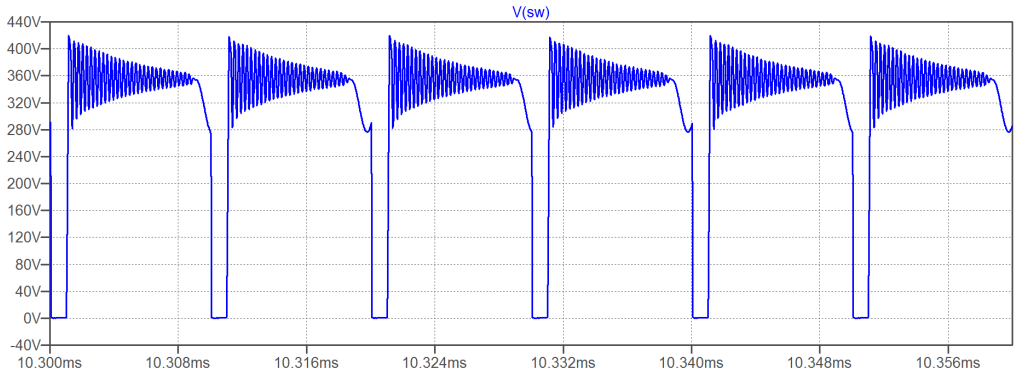

Figure 5.6 : MOSFET drain-source voltage waveform.

In Fig. 5.7, snubber voltage waveform is shown. In each turn-off event of MOSFET, the energy stored in the leakage inductance is released through snubber diode to the snubber capacitor. With this energy, snubber capacitor voltage increases. During the switching cycle, capacitor voltage is discharged through snubber resistor, decreasing capacitor's voltage.

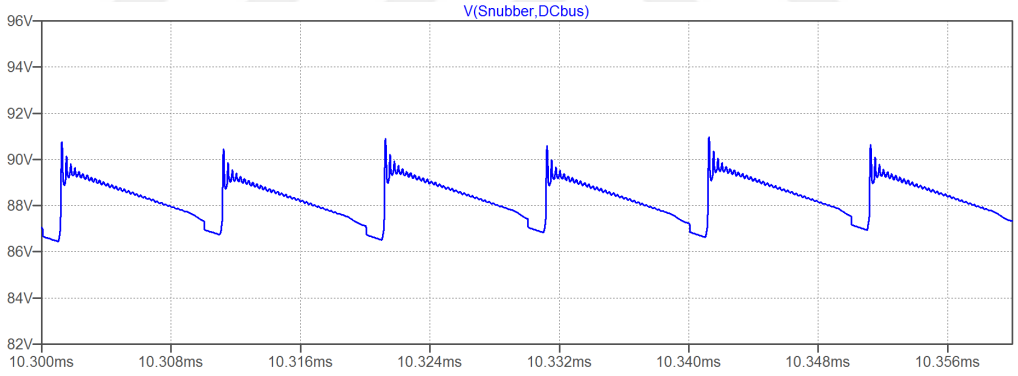

Figure 5.7 : Snubber capacitor voltage waveform.

In Fig. 5.8, shunt resistor current (also MOSFET drain-source current) is shown. When the MOSFET is turned on, current increases linearly because of transformers primary inductance and DC bus voltage. The excessive current spike on the turn-on event is due to transformer's parasitic capacitance. Despite being in the range of several tens of picofarads, which is selected as 30 pF for this simulation, very fast voltage change on the switch node creates this excessive current spike. To prevent premature turn-off sequence in the controller, leading edge blanking (LEB), circuitry is included in the controllers. LEB circuitry disables the current sense pin for a few hundreds of nanoseconds.

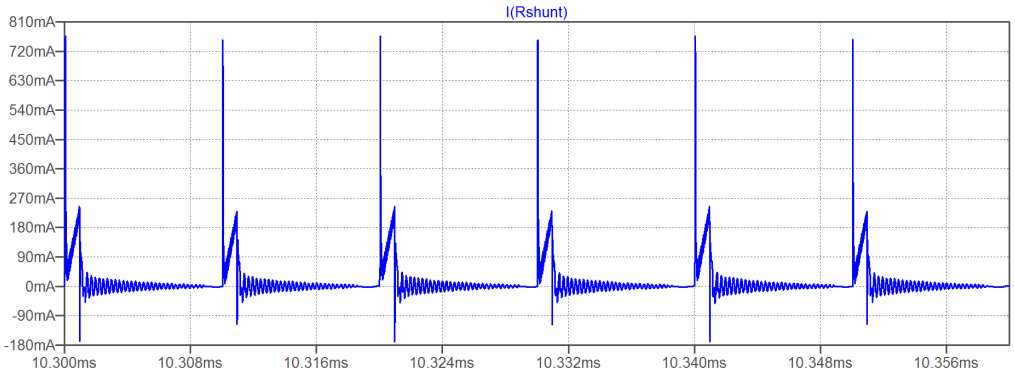

Figure 5.8 : Shunt resistor current waveform.

In 5.9, secondary rectifier diode waveform is shown. For the conduction time of the diode, forward voltage is around 0.7 V. When MOSFET turns on and transformer effect reverse biases the secondary rectifier diode, reverse voltage on the diode becomes the DC bus voltage divided by turns ratio.

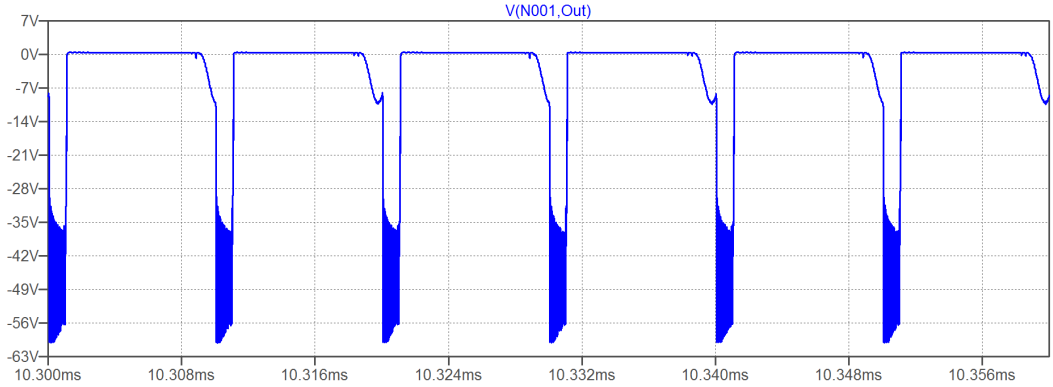

Figure 5.9 : Secondary diode voltage waveform.

In Fig. 5.10, output voltage waveform is shown. It can be seen that output voltage peak to peak ripple is less than 150 mV, which is selected as maximum 200 mV in the design requirements.

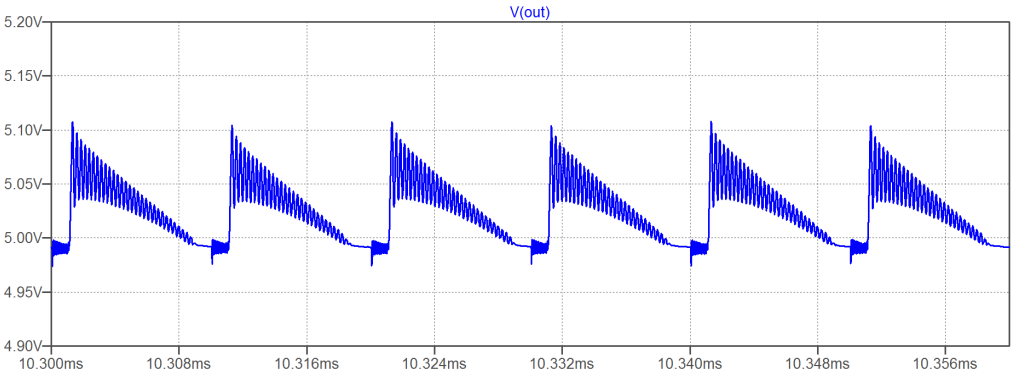

Figure 5.10 : Output voltage waveform.

Capacitor ripple current is also another constraint for capacitors. Exceeding maximum ripple current of the capacitor reduces capacitor life time. In Fig. 5.11, output capacitor's ripple current is shown. Peak value is equal to secondary diode current and linearly reduces down.

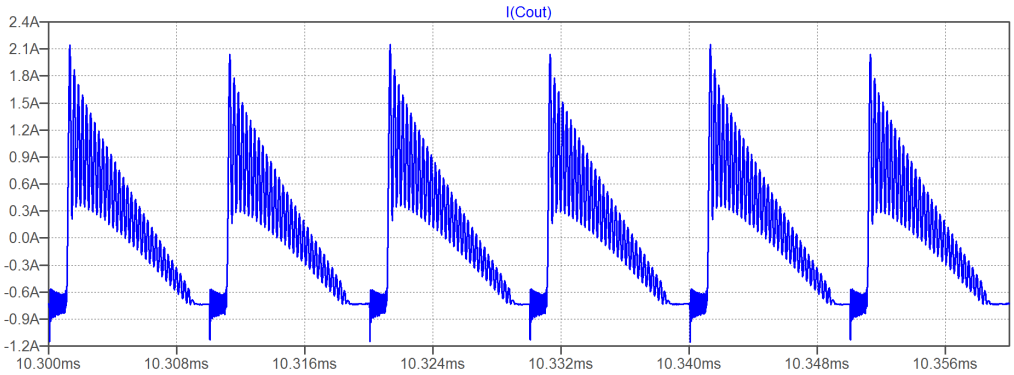

Figure 5.11 : Output capacitor ripple current waveform.

5.3 Schematic and PCB Design

Schematic and PCB design of the Flyback converter is done in Altium Designer program. Altium Designer is used widely in the industry. In Fig. 5.12 power section, in Fig. 5.13 control section of the Flyback converter is shown. Schematic and PCB is drawn in a manner that most of the genetic algorithm resulted flyback converters can be realized using this design. For example, there are two different DC Bus capacitor references in the schematic so that it is possible to use a 5 mm pitch and 8 mm pitch electrolytic capacitors can be used. Because of this, for genetic algorithm resulted converter, some of the components are not used and a cross symbol is placed on top of them in the schematic representation.

In Fig. 5.13, there are three circuit blocks. From left to right respectively, control IC block, high voltage start-up resistors and optocoupler-TL431 feedback section is given.

PCB of the converter is shown in Fig. 5.14. For ease of prototyping, single layer board is designed.

In Fig. 5.15, finished converter is shown. With the real converter in hand, waveform investigation and efficiency measurements are made in the next section.

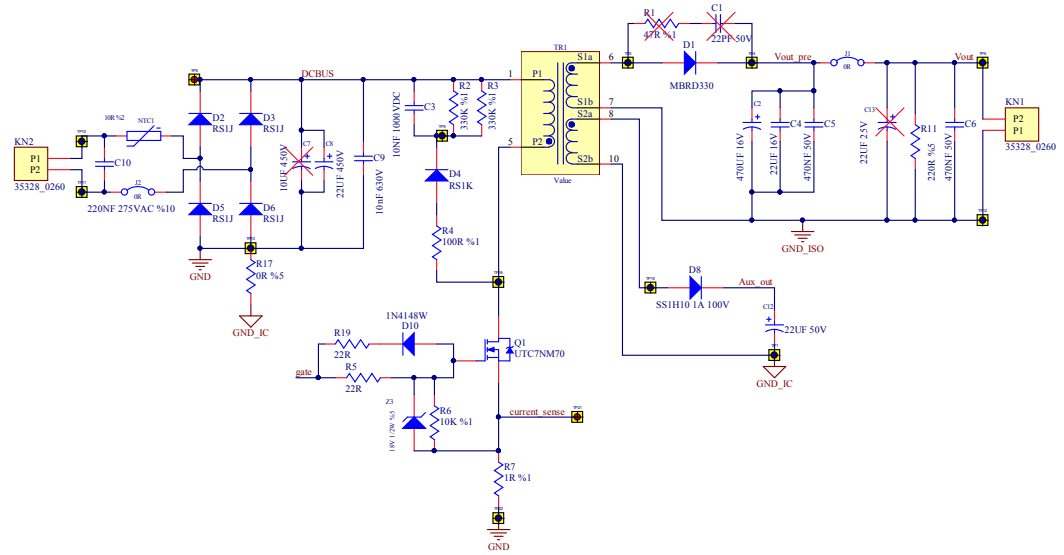

Figure 5.12 : Altium Designer power section schematic design.

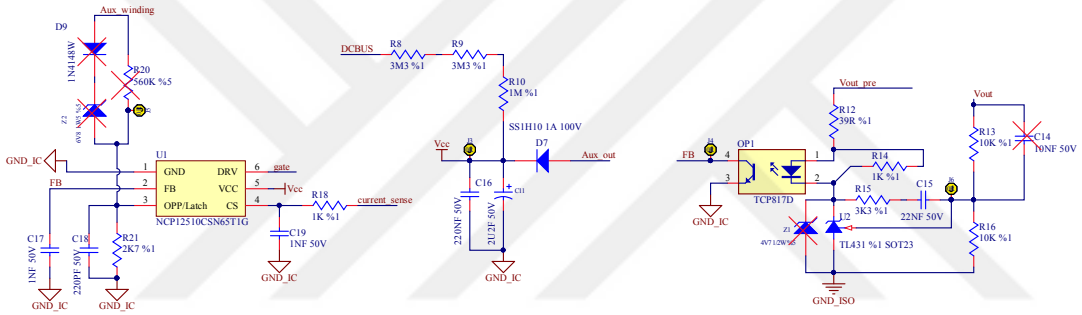

Figure 5.13 : Altium Designer control section schematic design.

5.4 Test Equipment

Efficiency and waveform examinations are made using the laboratory equipment given in Table 5.3.

Table 5.3 : Laboratory equipment used for measurements.

Equipment	Model
Oscilloscope	Lecroy WaveRunner 604Zi
AC Power Source	Chroma Programmable AC Source 6560
Power Analyzer	Yokogawa WT500
Programmable Electronic Load	GwInstek PEL-300
Handheld Multimeter	Fluke 177

5.5 Test Results

In this section, measurements and tests that are applied to realized converter is given. In Fig. 5.16, general view of laboratory table and equipment setup is given.

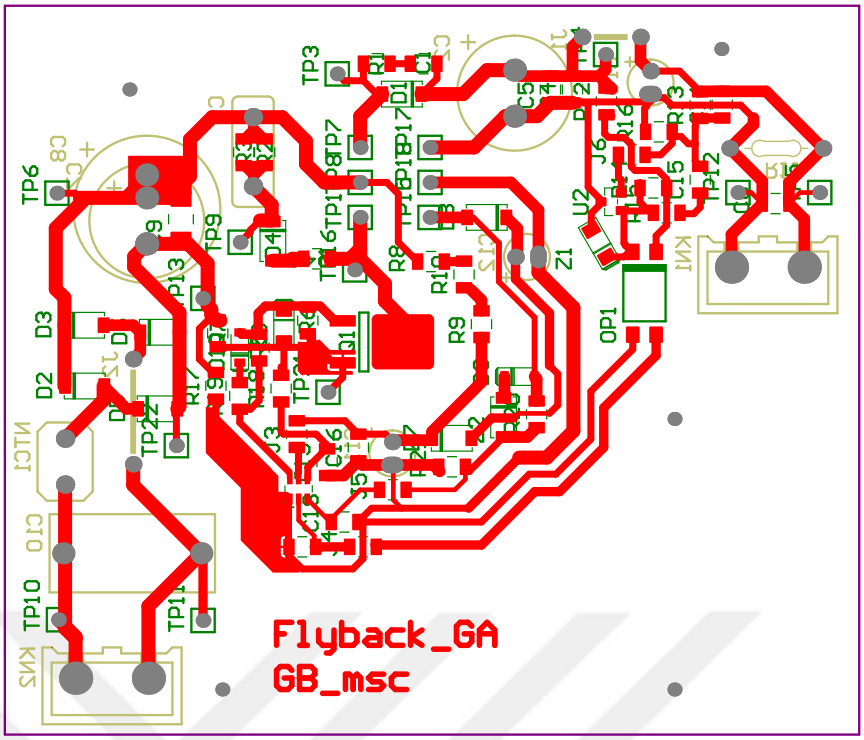

Figure 5.14 : Altium Designer PCB design.

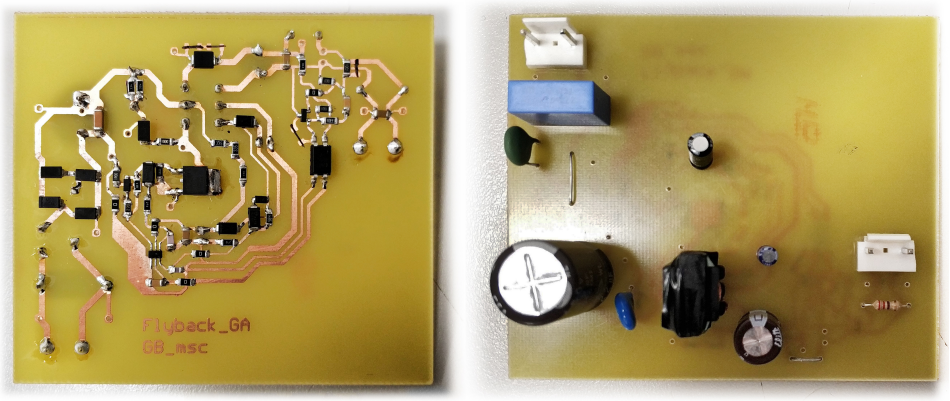

Figure 5.15 : Realized converter BOTTOM and TOP side.

In Table 5.4, measured and calculated values of transformer parameters are given. Measured parameters are close enough and calculation are validated. Biggest error is in leakage inductance value. The reason behind this issue is the hand wound process has imperfections and differs from calculation methodology.

5.5.1 Efficiency measurements

Efficiency measurements are taken under 4 different conditions, namely:

- 85 V RMS input voltage with different loading
- 230 V RMS input voltage with different loading

Figure 5.16 : Working environment.

Table 5.4 : Transformer calculations and measurements.

Parameter	Calculated Value	Measured Value	Unit
Primary inductance	1290	1369	mH
Leakage inductance	22	31	μH
Primary DC resistance	1.445	1388	Ω
Secondary DC resistance	49	64	mΩ
Saturation current	576	630	mA

- 265 V RMS input voltage with different loading
- Full load with different input voltages

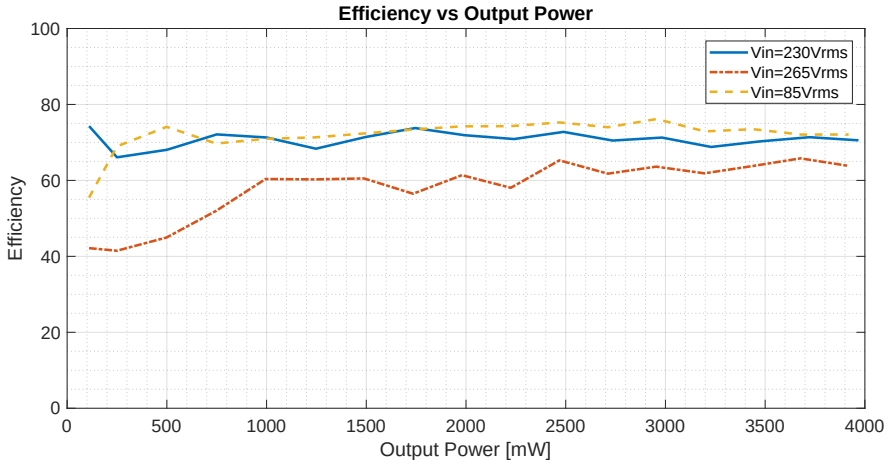

Figure 5.17 : Output power vs efficiency plots with different input voltages.

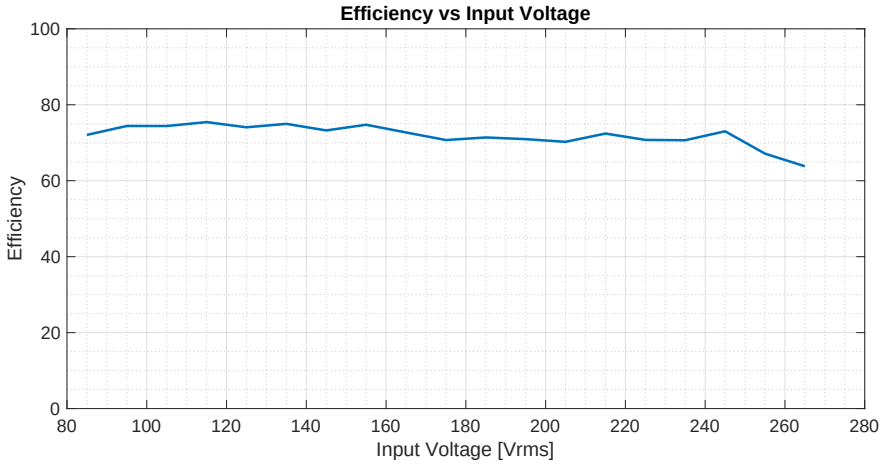

Figure 5.18 : Input voltage vs efficiency plots.

In Fig 5.17 and 5.18 efficiency results are given. Average efficiency is around 70%. It can be seen that for 85 V RMS and 230 V RMS input voltage, efficiency curve is relatively flat for entire load range. For 265 V RMS input voltage, low load efficiency drops down but with increasing loading, efficiency is increased. It can be seen in Fig 5.18, for full load condition entire voltage range efficiency is relatively flat.

5.5.2 Oscilloscope waveforms

In this section, fundamental waveforms of the circuit is given. Unless otherwise is stated, graphics are taken under full load condition and 230 V RMS input voltage.

In Fig. 5.19, DC bus voltage is shown. It can be seen that peak to peak ripple voltage is around 12 V at full load condition.

In Fig. 5.20, MOSFET drain-source voltage waveform is shown. Peak MOSFET voltage is measured as 444 V. Also it can be seen that converter operates in DCM mode.

In Fig. 5.21, snubber voltage waveform is shown with respect to the ground. It can be seen that ripple voltage is very low, around 5 V with the peak voltage of 419 V. Considering the DC bus voltage of $230 \times \sqrt{2} = 320$, snubber voltage is 100 V.

In Fig. 5.22, secondary diode voltage waveform is shown. Peak reverse voltage on the diode is measured as 42 V.

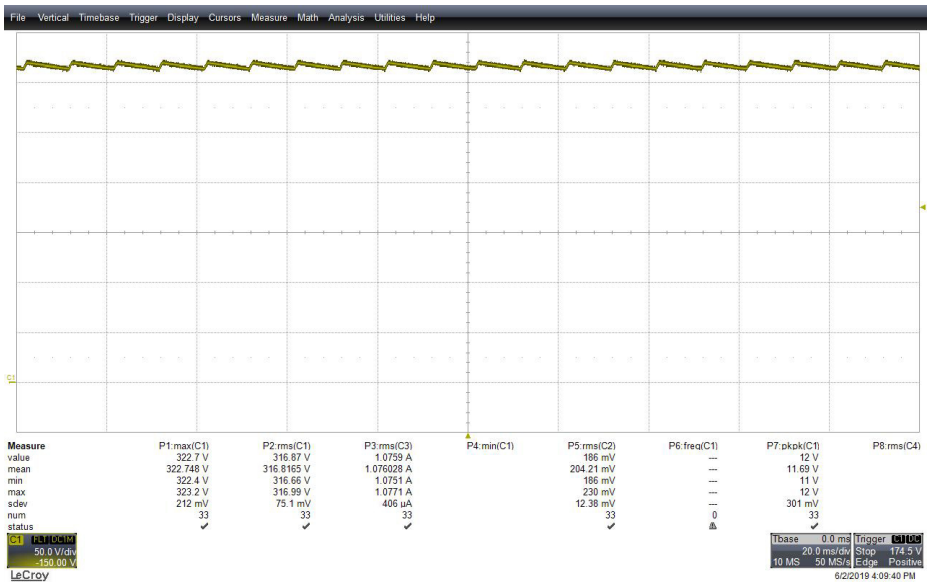

Figure 5.19 : 230 V RMS - DC bus voltage.

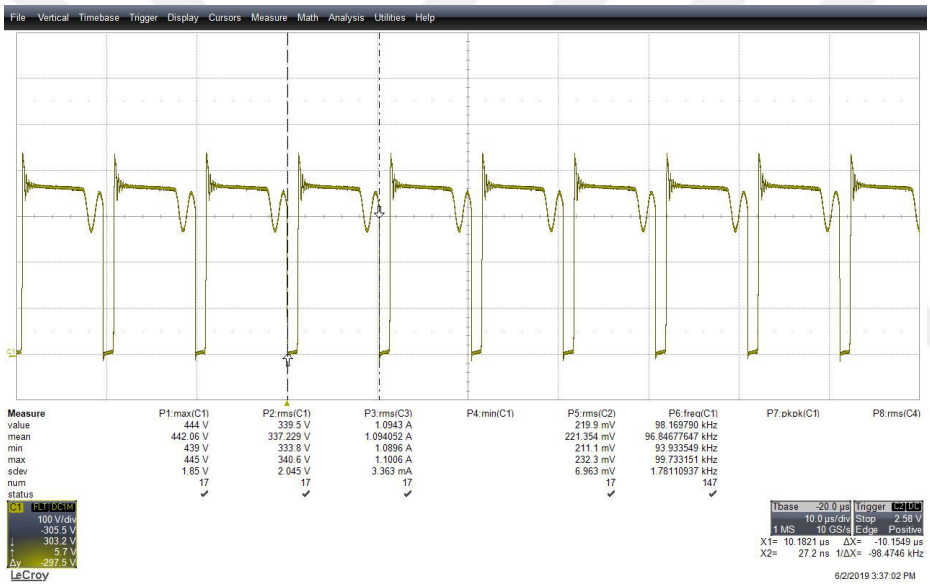

Figure 5.20 : 230 V RMS - MOSFET drain source voltage.

Fig. 5.23 shows the voltage on the shunt resistor. Measured voltage waveform is noisy, but general waveform can be seen. Electrical noise is due to parasitic inductance and capacitors on the PCB and components. For shunt resistor, calculated value by genetic algorithm was 3 ohm. But closest value was 3.6 ohm in hand. So considering 3.6 ohm in circuit, current can be calculated. Spike current at the starting of the period is $4.2/3.6 = 1.16$ A. After a small resonance, current increases linearly up to $0.82/3.6 = 0.227$ A. Measured value of 0.82 is also the peak current sense pin voltage of the NCP12510 controller IC.

In Fig. 5.24, output voltage waveform is given. There is a high spike voltage at the output, but this spike is not a result of main circuit design. When it is investigated, it is

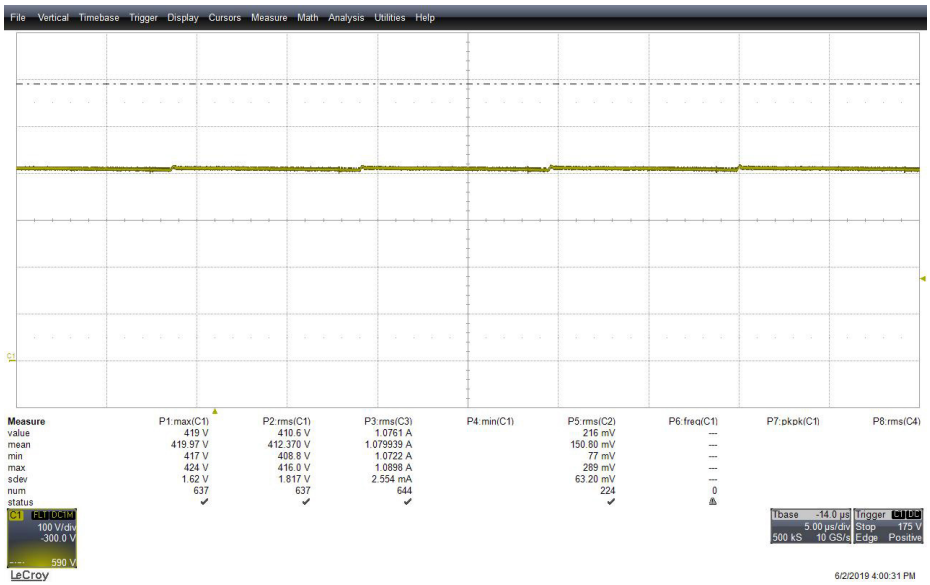

Figure 5.21 : 230 V RMS - Snubber voltage.

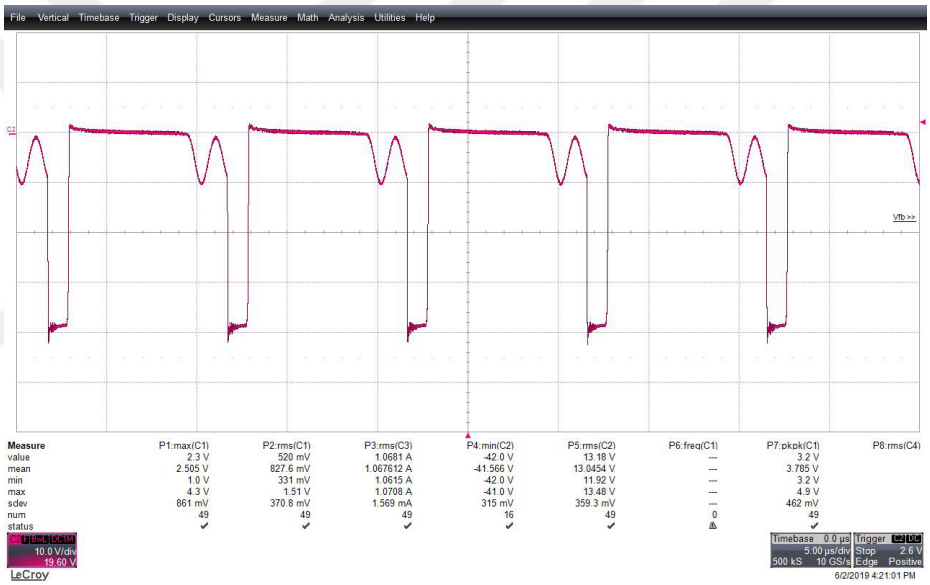

Figure 5.22 : 230 V RMS - Secondary side rectifier diode voltage.

found that this noise voltage is due to turn on event of the MOSFET and resonance frequency is around 65 MHz. Solution to this problem is to use ferrite bead at the output and have a better PCB layout. So ignoring the high frequency the high frequency noise, output voltage ripple is around 200 mV.

Each components have worst case scenario at different input voltage. For MOSFET from the maximum voltage point of view, worst case is the maximum load and maximum input voltage. In Fig. 5.25 this condition is given. Maximum drain source voltage is measured as 491 V.

Figure 5.23 : Shunt resistor voltage.

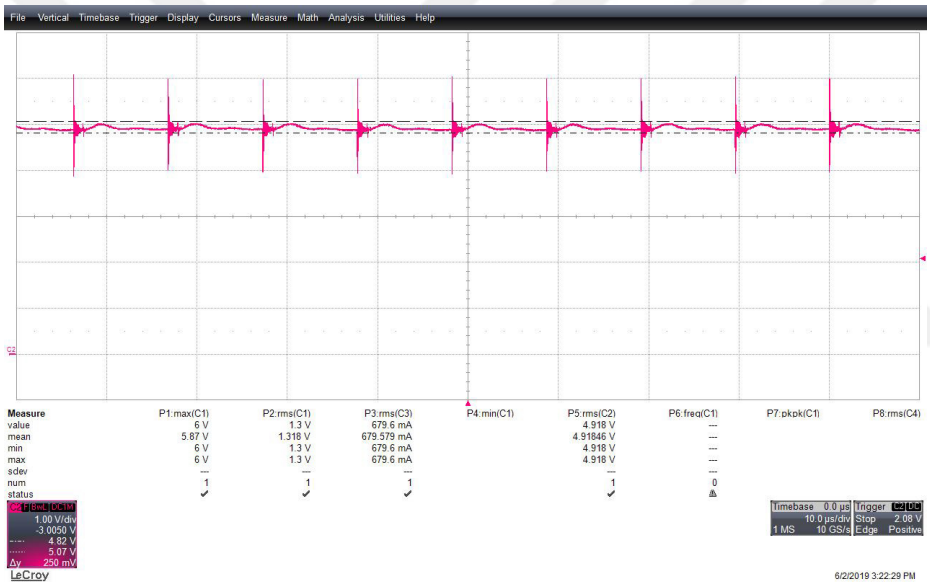

Figure 5.24 : Output voltage waveform.

For DC bus ripple voltage, worst case scenario is the minimum input voltage and maximum load. In Fig. 5.26 DC bus voltage at 85 V RMS input voltage is given. Maximum ripple voltage is measured as 19 V.

For secondary rectifier diode, from the maximum reverse voltage point of view worst case scenario is the maximum input voltage and maximum load. In Fig. 5.27, maximum reverse voltage on the diode is measured as 46 V.

In Fig. 5.28, start up sequence of the converter is shown. At start up, maximum overshoot voltage is 6 V. Output voltage reaches to its final value in 5 miliseconds.

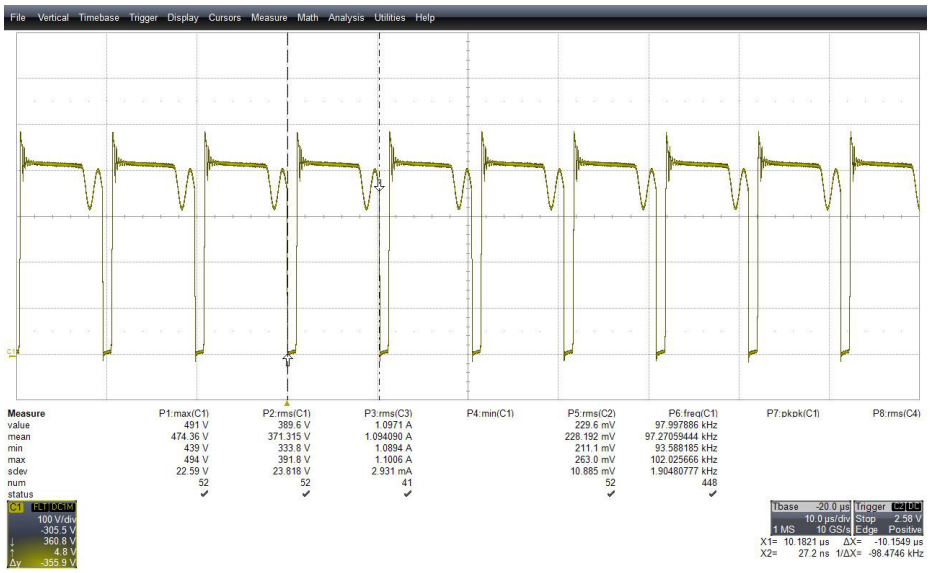

Figure 5.25 : 265 V RMS - MOSFET drain source voltage.

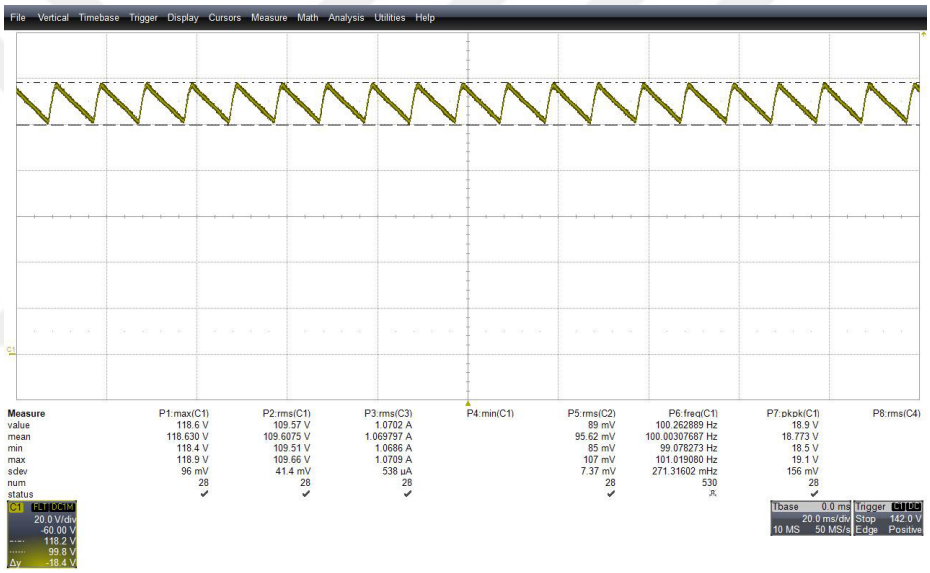

Figure 5.26 : 85 V RMS - DC bus voltage.

In Fig. 5.29, no load condition MOSFET drain source and feedback pin voltage is given. Switching has no fixed frequency. When feedback voltage level reaches to a certain level MOSFET is turned on.

In Fig. 5.30, satusaturation current level of the transformer is given. It can be seen that around 700 mA transformer saturates which was calculated as 600 mA. Magnetizing inductance can be calculated using this figure.

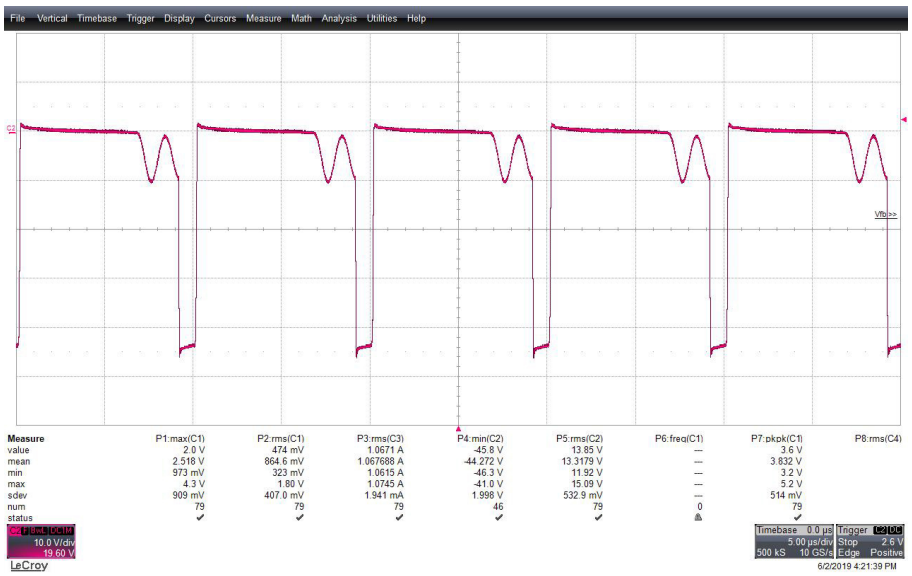

Figure 5.27 : 265 V RMS - Secondary side rectifier diode voltage.

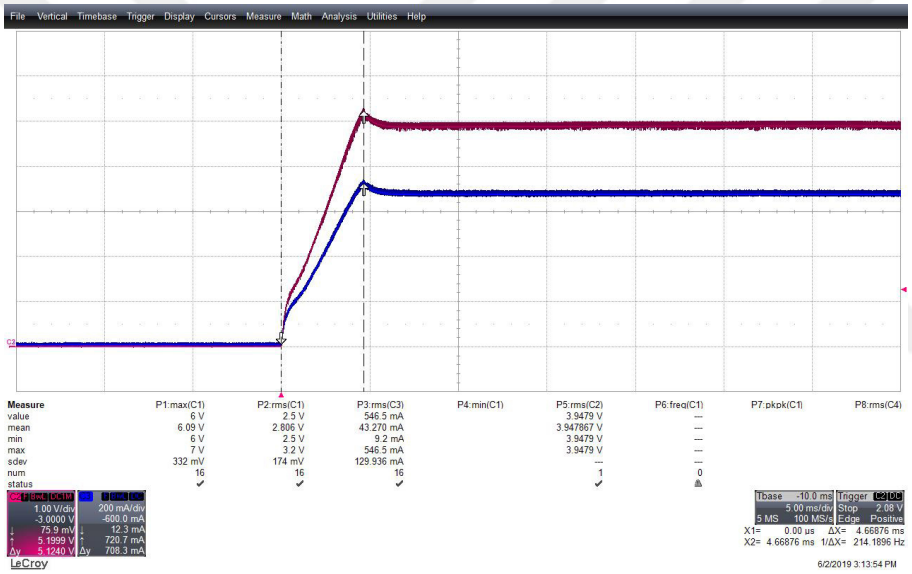

Figure 5.28 : Start-up sequence of the converter.

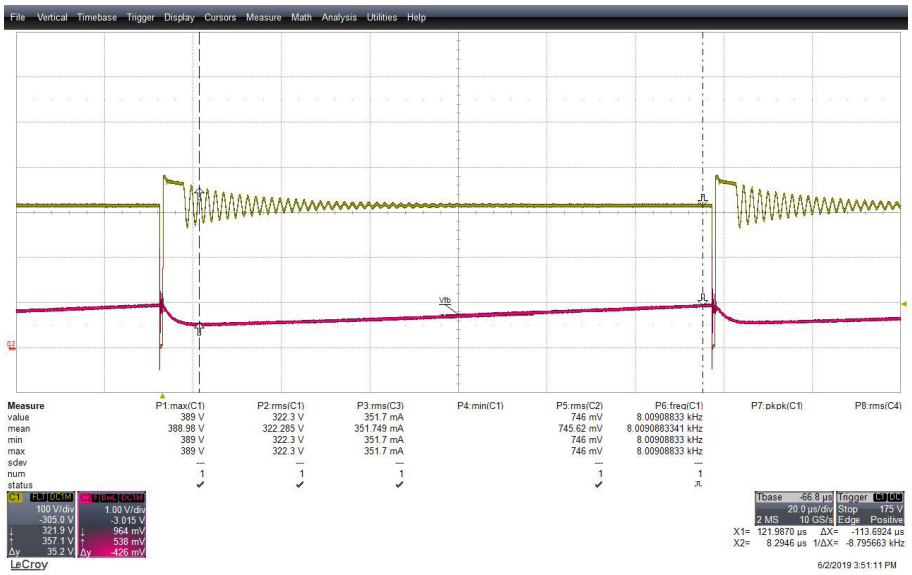

Figure 5.29 : 230 V RMS - No load, MOSFET drain source voltage.

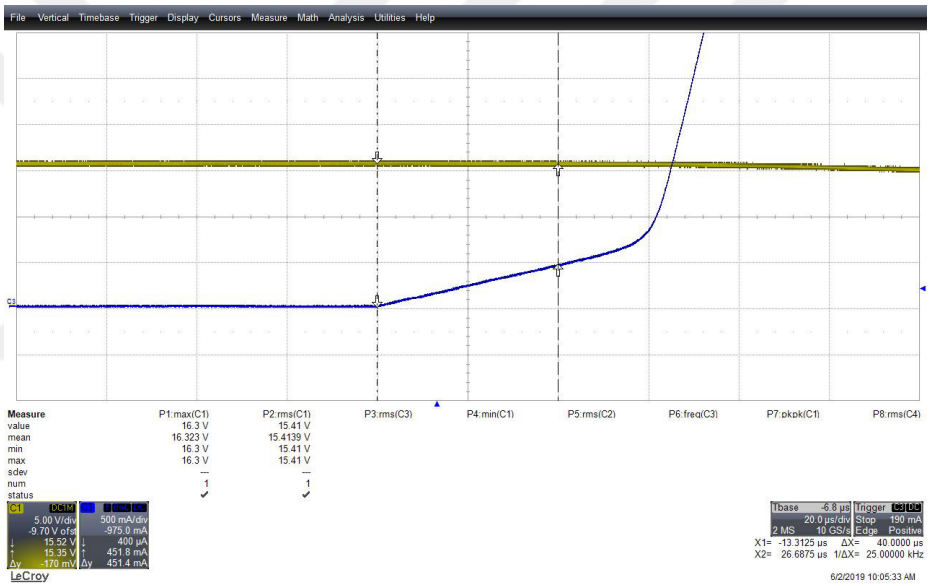

Figure 5.30 : Transformer saturation current.

6. CONCLUSIONS AND RECOMMENDATIONS

With this master thesis a genetic algorithm implementation of flyback converter design is done. Using this implementation cost optimization of the converter is obtained.

In the thesis, optimization problem is defined as a problem that has 20 variables. Each variable is explained and its effect on the design is given. General approach to the genetic algorithm optimization problem is given. Designing a converter using genetic algorithm requires a different approach than a classical design approach.

A component library is created with 317 commercially available discrete components. For other variables such as transformer turns ratio and air gap distance, matrix of technically feasible range of numbers are created. Using these components there are 91×10^{24} possible flyback converter combinations. There is no possible way to calculate and optimize the converter cost with derivative based or brute force methods. Random search algorithms are very effective for this type of immense problems. Using genetic algorithm and a few tens of thousand of converter calculations very effective results are obtained.

But due to random nature, reproducibility of the random search algorithms are not possible. To overcome this problem random number seeds are used. For the same reason, evaluation of the random search algorithms are a little different than classical optimization algorithms. Because in each run, due to randomized nature of the algorithms, obtained results are different. To evaluate random seed algorithms, there is a simple method and it is widely used. The methods to overcome this problem is statistical methods. Proposed algorithm is run through numerous time to obtain lots of results. Depending on the problem, algorithm run number may be hundreds or thousands. Histogram of the obtained results is plotted to see distribution of the results. Same approach is used for this thesis. To evaluate genetic algorithm performance, algorithm is run two hundreds time with same parameters. Results are given in Fig. 6.1. Obtained histogram have a bell curved shape. When datas are investigated with statistical methods, average value is 1.7899$ and standard deviation is 0.2339$. The

converter that has lowest cost found by genetic algorithm is 1.414$. If bell curve is assumed to have normal distribution, there is a 5.4% chance of finding a lower priced converter than the one found as best solution. So this confidence level is enough for most situations.

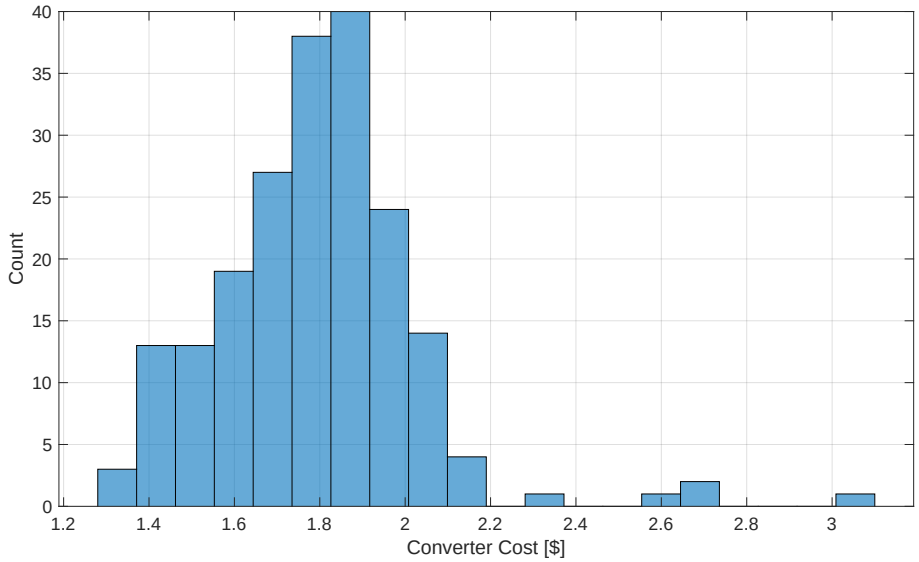

Figure 6.1 : Histogram of prices for 200 converter.

Considering previously designed flyback converter with 2.212$ cost by Power Integrations, mean value of converters that are generated by GA is 20% less costs. When compared with the best solution, cost down is 36%.

6.1 Practical Application of This Study and Future Work

Emergence point of this thesis is due to a practical problem that is faced in the design optimization. Almost every designer in the electrical engineering field is under the pressure of cost reduction and optimization. With this study, a multi dimensional optimization problem with considering technical and economical aspects of the design stage is combined using genetic algorithm.

Current component library has components for only up to 15 W of flyback converter design. As a future work, component library can be expanded to cover more broad range of converters.

Also as a future work, instead of optimizing cost, efficiency can be optimized by changing cost function. Since all of the power loss calculations are already implemented, this chance is very easy to do.

At this point paralleling output capacitor and secondary side winding is possible with genetic algorithm. As a future work, paralleling DC bus capacitor can be implemented with the addition of another design variable. Also output post LC filter can be implemented for high power applications. Addition of post LC filter requires at least another two design variable.

When implementing the component database and MATLAB libraries, generalized approach is followed as much as possible. Component database can be used for any type of converter design. Most of the fundamental parameters for each component is added to the libraries. Also in MATLAB environment, each component is implemented as its own class. By this approach, addition of another parameter can be made with ease.

For this thesis purposes flyback calculations are focused on single output flyback converters. But flyback converters are widely used as multiple output converters in the industry. So by modifying some of the equations, current calculations can be easily expanded into multiple output flyback converters.

Constraint function are also very general. Comparison of circuit operating point with the component's maximum datasheet values are valid for every type of converter. So without any change or with a few small modifications, constraint functions can be expanded to other type of converters, such as buck, boost or forward converter.

By changing converter calculation equations implementation of other converters with genetic algorithm is also possible. In these calculations some of the equations will be still valid such as DC bus and line rectification calculations.

Each of the above mentioned topics can be a research topic and should be investigated further.

REFERENCES

[1] **Sussman, G. and Stallman, R.** (1975). Heuristic techniques in computer-aided circuit analysis, *IEEE Transactions on Circuits and Systems*, *22*(11), 857–865.

[2] **Rahman, S. and Lee, F.C.** (1981). Nonlinear program based optimization of boost and buck-boost converter designs, *1981 IEEE Power Electronics Specialists Conference*, pp.180–191.

[3] **León-Aldaco, S.E.D.**, **Calleja, H. and Alquicira, J.A.** (2015). Metaheuristic Optimization Methods Applied to Power Converters: A Review, *IEEE Transactions on Power Electronics*, *30*(12), 6791–6803.

[4] **Gordon, M.H.**, **Stewart, M.B.**, **King, B.**, **Balda, J.C. and Olejniczak, K.J.** (1995). Numerical optimization of a heat sink used for electric motor drives, *IAS '95. Conference Record of the 1995 IEEE Industry Applications Conference Thirtieth IAS Annual Meeting*, volume 2, pp.967–970 vol.2.

[5] **Zhang, J.**, **Chung, H.**, **Lo, W.L.**, **Hui, S.Y.R. and Wu, A.** (2000). Decoupled optimization technique for design of switching regulators using genetic algorithms, *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, volume 3, pp.495–498 vol.3.

[6] **Duro, B.**, **Ramsden, V.S. and Muttik, P.** (1999). Minimization of active filter rating in high power hybrid filter systems, *Proceedings of the IEEE 1999 International Conference on Power Electronics and Drive Systems. PEDS'99 (Cat. No.99TH8475)*, volume 2, pp.1043–1048 vol.2.

[7] **Suresh, P. and Kirubakaran, D.** (2014). Power Loss Optimization of Bosst Convertor Using Genetic Algorithm, *International Journal of Scienctific and Engineering Research*, *5*(4), 112–117.

[8] **Helali, H.**, **Bergogne, D.**, **Slama, J.B.H.**, **Morel, H.**, **Bevilacqua, P.**, **Allard, B. and Brevet, O.** (2005). Power converter's optimisation and design. Discrete cost function with genetic based algorithms, *2005 European Conference on Power Electronics and Applications*, pp.7 pp.–P.7.

[9] **Busquets-Monge, S.**, **Soremekun, G.**, **Hertz, E.**, **Crebier, C.**, **Ragon, S.**, **Zhang, J.**, **Boroyevich, D.**, **Gurdal, Z.**, **Lindner, D.K. and Arpilliere, M.** (2002). Design optimization of a boost power factor correction converter using genetic algorithms, *APEC. Seventeenth Annual IEEE Applied Power Electronics Conference and Exposition (Cat. No.02CH37335)*, volume 2, pp.1177–1182 vol.2.

[10] **Shi, K.L. and Li, H.** (2005). Optimized PWM strategy based on genetic algorithms, *IEEE Transactions on Industrial Electronics*, *52*(5), 1458–1461.

[11] **Hasanzadeh, A.**, **Zolghadri, M.R.**, **Kaboli, S.H. and Homaifar, A.** (2003). A genetic algorithm based programmed PWM optimum switching pattern calculation, *The Fifth International Conference on Power Electronics and Drive Systems, 2003. PEDS 2003.*, volume 2, pp.1081–1085 Vol.2.

[12] **Sarvi, M. and Salimian, M.R.** (2010). Optimization of specific harmonics in multilevel converters by GA amp;amp; PSO, *45th International Universities Power Engineering Conference UPEC2010*, pp.1–4.

[13] **Piette, N.**, **Clavel, E. and Marechal, Y.** (1998). Optimization of cabling in power electronics structure using inductance criterion, *Conference Record of 1998 IEEE Industry Applications Conference. Thirty-Third IAS Annual Meeting (Cat. No.98CH36242)*, volume 2, pp.925–928 vol.2.

[14] **Kostov, K.S. and Kyyra, J.J.** (2005). Genetic algorithm optimization of peak current mode controlled buck converter, *Proceedings of the 2005 IEEE Midnight-Summer Workshop on Soft Computing in Industrial Applications, 2005. SMCia/05.*, pp.111–116.

[15] **Achiammal, B. and Kayalvizhi, R.** (2014). Genetic Algorithm based PI controller for Negative Output Elementary LUO converter, *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp.1099–1103.

[16] **Yousefi, M.R.**, **Emami, S.A.**, **Eshtehardiha, S. and Bayati Poudeh, M.** (2008). Particle swarm optimization and genetic algorithm to optimizing the pole placement controller on Cuk converter, *2008 IEEE 2nd International Power and Energy Conference*, pp.1461–1465.

[17] **Versèle, C.**, **Deblecker, O. and Lobry, J.** (2010). Multiobjective optimal design of transformers for isolated switch mode power supplies, *SPEEDAM 2010*, pp.1687–1692.

[18] **Diaz-Gomez, P.A. and Hougen, D.F.** (2007). Initial Population for Genetic Algorithms: A Metric Approach., *GEM*, pp.43–49.

[19] **Haupt, R.L.** (2000). Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors, *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (C*, volume 2, pp.1034–1037 vol.2.

[20] **Kramer, O.** (2017). *Genetic algorithm essentials*, volume679, Springer.

[21] **Daniel W Hart, P.** (2010). *Power Electronics*, McGraw-Hill Education, `https://books.google.com.hk/books?id=mX6NxAEACAAJ`.

[22] **Picard, J.** (2010). Under the Hood of Flyback SMPS Designs, **Technical Report**, Texas Instruments.

[23] **Choi, H.S.** (2003). Design Guidelines for Off-line Flyback Converters Using Fairchild Power Switch (FPS), **Technical Report**, Fairchild Semiconductor.

[24] **Ray Ridley, A.N.** (2006). Modeling Ferrite Core Losses, **Technical Report**, Switching Power Magazine.

[25] **Elektroisola**, (2015), Technical data for enamlled copper wire, based on IEC60317, website.

[26] **Erickson, R.W. and Maksimovic, D.** (2001). *Fundamentals of Power Electronics*, Springer, 2ed edition.

[27] **Thummala, P.**, **Schneider, H.**, **Ouyang, Z.**, **Zhang, Z. and Andersen, M.A.** (2013). Estimation of transformer parameters and loss analysis for high voltage capacitor charging application, *2013 IEEE ECCE Asia Downunder*, IEEE, pp.704–710.

[28] **McLyman, C.** (2016). *Transformer and Inductor Design Handbook*, CRC Press, `https://books.google.com.tr/books?id=JR7OBQAAQBAJ`.

[29] **Brown, M.** (2001). *Power Supply Cookbook*, Newnes.

[30] **Dusan Graovac, Marco Pürschel, A.K.** (2006). MOSFET Power Losses Calculation Using the Datasheet Parameters, **Technical Report**, Infineon Technologies.

[31] **Halder, T.** (2013). Comprehensive power loss model of the main switch of the Flyback converter, *2013 International Conference on Power, Energy and Control (ICPEC)*, pp.792–797.

[32] (2011). Calculation of conduction losses in a power rectifier, **Technical Report**, ST Microelectronics.

[33] **Ridley, R.** (2005). Proximity Loss in Magnetics Winding, **Technical Report**, Switching Power Magazine.

[34] **Hava, A.M.**, **Ayhan, U. and Aban, V.V.** (2012). A DC bus capacitor design method for various inverter applications, *2012 IEEE Energy Conversion Congress and Exposition (ECCE)*, IEEE, pp.4592–4599.

[35] (2011). Judicious Use of Aluminum Electrolytic Capacitors, **Technical Report**, Nippon Chemicon.

[36] **Mathworks**, (2019). Global Optimization Tool Box User's Guide, MATLAB, 1 Apple Hill Drive Natick, MA, r2019a edition.

APPENDICES

APPENDIX A.1 : Component Class Definition Codes
APPENDIX A.2 : Component Library Importing Codes

APPENDIX A MATLAB CODES

MATLAB codes are given in this appendix.

APPENDIX A.1 Component Class Definition Codes

```
classdef CapacitorCER
    %ceramic capacitor class
    properties
        ID;
        C;
        Vmax;
        tolerance;
        Tmax;
        Type;
        tanf;
        Price;
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = CapacitorCER(ID, C, Vmax, tolerance, Tmax, Type, ...
                tanf, Price)
            if nargin > 0
                obj.ID = ID;
                obj.C = C;
                obj.Vmax = Vmax;
                obj.tolerance = tolerance;
                obj.Tmax = Tmax;
                obj.Type = Type;
                obj.tanf = tanf;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef CapacitorEL
    %electrolytic capacitor class
    properties
        ID;
        C; %capacitance
        Vmax; %rated voltage
        tolerance; %capacitance tolerance
        Tmax; %max operating temperature
        Iripp; %max ripple current @100kHz
        kRC120; %ripple cureent constant for 120 hertz
        Ileak; %leakge current @rated voltage and temperature
        tanf; %max dissipation factor
        Life; %rated load life time
        Dia; %diameter
        Len; %height
        Price; %price of the component
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = CapacitorEL(ID, C, Vmax, tolerance, Tmax, Iripp, ...
                kRC120, Ileak, tanf, Life, Dia, Len, Price)
            if nargin > 0
                obj.ID = ID;
                obj.C = C;
                obj.Vmax = Vmax;
                obj.tolerance = tolerance;
```

```
                obj.Tmax = Tmax;
                obj.Iripp = Iripp;
                obj.kRC120 = kRC120;
                obj.Ileak = Ileak;
                obj.tanf = tanf;
                obj.Life = Life;
                obj.Dia = Dia;
                obj.Len = Len;
                obj.Price = Price;
            end
        end

        function powerLoss = calcPowerLoss(obj, iLowFreq, iHighFreq, fLow, fHigh)
            XcLowFreq =  1/(2*pi*fLow*obj.C);
            XcHighFreq = 1/(2*pi*fHigh*obj.C);

            ESRLowFreq = XcLowFreq * obj.tanf;
            ESRHighFreq = XcHighFreq*obj.tanf + 20e-3; %ensure some lead res.

            powerLoss = ESRLowFreq*iLowFreq^2 + ESRHighFreq*iHighFreq^2;
        end
    end
end
```

```
classdef Controller
    %Controller IC class
    properties
        ID
        Fswitching
        VcurrentSense
        DutyMax
        VccMin
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Controller(ID, Fswitching, VcurrentSense, DutyMax, ...
                VccMin, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Fswitching = Fswitching;
                obj.VcurrentSense = VcurrentSense;
                obj.DutyMax = DutyMax;
                obj.VccMin = VccMin;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef Core
    %Core class
    properties
        ID
        Al
        Perm
        Ae
        Bsat
        SelfGap
        Volume
        CoreLoss
        CoilFormerInnerWidth
        CoilFormerOuterWidth
        CoilFormerHeight
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Core(ID, Al, Perm, Ae, Bsat, SelfGap, Volume, CoreLoss, ...
```

```
                CoilFormerinWidth, CoilFormeroutWidth, CoilFormerHeight, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Al = Al;
                obj.Perm = Perm;
                obj.Ae = Ae;
                obj.Bsat = Bsat;
                obj.SelfGap = SelfGap;
                obj.Volume = Volume;
                obj.CoreLoss = CoreLoss;
                obj.CoilFormerInnerWidth = CoilFormerinWidth;
                obj.CoilFormerOuterWidth = CoilFormeroutWidth;
                obj.CoilFormerHeight = CoilFormerHeight;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef Diode  %diode class
    properties
        ID
        Vfwd
        Vrvs
        R
        Tj
        Ipeak
        Iavg
        Irms
        Rja
        isSMD
        isFast
        Price
    end

    methods  %constructor
        %get all the parameters in the class properties
        function obj = Diode(ID, Vfwd, R, Vrvs, Tj, Ipeak, Iavg, Irms, Rja, ...
                isSMD, isFast, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Vfwd = Vfwd;
                obj.R = R;
                obj.Vrvs = Vrvs;
                obj.Tj = Tj;
                obj.Ipeak = Ipeak;
                obj.Iavg = Iavg;
                obj.Irms = Irms;
                obj.Rja = Rja;
                obj.isSMD = isSMD;
                obj.isFast = isFast;
                obj.Price = Price;
            end
        end

        function powerLoss = calcPowerLoss(obj, Iavg, Irms)
            powerLoss = obj.Vfwd * Iavg + obj.R*Irms*Irms;
        end
    end
end
```

```
classdef MOSFET
    %MOSFET class
    properties
        %can be found n datasheets
        ID;
        RdsON;
        Coss;
        Qg;
        Vrvs;
        Tj;
        Idc;
        Ipeak;
```

```
        Rja;
        isSMD;
        Price;
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = MOSFET(ID, RdsON, Coss, Qg, Vrvs, Tj, Idc, Ipeak, Rja, ...
                isSMD, Price)
            if nargin > 0
                obj.ID = ID;
                obj.RdsON = RdsON;
                obj.Coss = Coss;
                obj.Qg = Qg;
                obj.Vrvs = Vrvs;
                obj.Tj = Tj;
                obj.Idc = Idc;
                obj.Ipeak = Ipeak;
                obj.Rja = Rja;
                obj.isSMD = isSMD;
                obj.Price = Price;
            end
        end

        function powerLoss = calcPowerLoss(obj, Irms, Vds, Vcc, Fsw)
            CossVds = obj.Coss * sqrt(25/Vds);

            pSwitching = Vds^2 * Fsw * CossVds / 2;
            pGating = obj.Qg * Fsw * Vcc;
            pConduction = obj.RdsON*Irms*Irms;

            powerLoss = pConduction + pSwitching + pGating;
        end
    end
end
```

```
classdef Opto
    %Optocoupler class
    properties
        ID
        CTRmin
        CTRmax
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Opto(ID, CTRmin, CTRmax, Price)
            if nargin > 0
                obj.ID = ID;
                obj.CTRmin = CTRmin;
                obj.CTRmax = CTRmax;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef Resistor
    %resistor class
    properties
        %can be found in datasheets
        ID;
        R;
        Vmax;
        Pmax;
        Tolerance;
        isSMD;
        Price;
    end

```

```
    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Resistor(ID, R, Vmax, Pmax, Tolerance, isSMD, Price)
            if nargin > 0
                obj.ID = ID;
                obj.R = R;
                obj.Vmax = Vmax;
                obj.Pmax = Pmax;
                obj.Tolerance = Tolerance;
                obj.isSMD = isSMD;
                obj.Price = Price;
            end
        end

        function powerLoss = calcPowerLoss(obj, I)
            powerLoss = obj.R * I^2;
        end
    end
end
```

```
classdef Shunt
    %Shunt resistor class
    properties
        ID
        R
        Pmax
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Shunt(ID, R, Pmax, Price)
            if nargin > 0
                obj.ID = ID;
                obj.R = R;
                obj.Pmax = Pmax;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef TL431
    %TL431 class
    properties
        ID
        Vref
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = TL431(ID, Vref, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Vref = Vref;
                obj.Price = Price;
            end
        end
    end
end
```

```
classdef Wire
    %Wire class
    properties
        ID
        Area
        Dia
        ResPerMeter
```

```
        MaxFreq
        Price
    end

    methods
        %constructor
        %get all the parameters in the class properties
        function obj = Wire(ID, Area, Dia, ResPerMeter, MaxFreq, Price)
            if nargin > 0
                obj.ID = ID;
                obj.Area = Area;
                obj.Dia = Dia;
                obj.ResPerMeter = ResPerMeter;
                obj.MaxFreq = MaxFreq;
                obj.Price = Price;
            end
        end
    end
end
```

APPENDIX A.2 Component Library Importing Codes

```
%get components from excel sheet database into MATLAB
tDiode = xlsread("component_database.xlsx", 'diode');
tCapacitorEL = xlsread("component_database.xlsx", 'capacitorEL');
tCapacitorCER = xlsread("component_database.xlsx", 'capacitorCER');
tResistor = xlsread("component_database.xlsx", 'Resistor');
tCore = xlsread("component_database.xlsx", 'Core');
tMosfet = xlsread("component_database.xlsx", 'MOSFET');
tWire = xlsread("component_database.xlsx", 'Wire');
tShunt = xlsread("component_database.xlsx", 'Shunt');
tController = xlsread("component_database.xlsx", 'Controller');
tTL431 = xlsread("component_database.xlsx", 'TL431');
tOpto = xlsread("component_database.xlsx", 'Opto');

%initialize component count array and design choices.
componentCounts = zeros(1,16);

temp = size(tDiode);
componentCounts(1) = temp(1,1);
temp = size(tCapacitorEL);
componentCounts(2) = temp(1,1);
temp = size(tCapacitorCER);
componentCounts(3) = temp(1,1);
temp = size(tResistor);
componentCounts(4) = temp(1,1);
temp = size(tCore);
componentCounts(5) = temp(1,1);
temp = size(tMosfet);
componentCounts(6) = temp(1,1);
temp = size(tWire);
componentCounts(7) = temp(1,1);
temp = size(tShunt);
componentCounts(8) = temp(1,1);
temp = size(tController);
componentCounts(9) = temp(1,1);
temp = size(tTL431);
componentCounts(10) = temp(1,1);
temp = size(tOpto);
componentCounts(11) = temp(1,1);
%create empty component arrays
Diodes = Diode.empty();
CapacitorELs = CapacitorEL.empty();
CapacitorCERs = CapacitorCER.empty();
Resistors = Resistor.empty();
MOSFETs = MOSFET.empty();
Cores = Core.empty();
Wires = Wire.empty();
Shunts = Shunt.empty();
Controllers = Controller.empty();
TL431s = TL431.empty();
Optos = Opto.empty();
```

```
51
52 %transfer component variables into their objects
53 for c = 1:componentCounts(1)
54     Diodes(c) = Diode(tDiode(c,1),tDiode(c,2),tDiode(c,3),tDiode(c,4), ...
55     tDiode(c,5),tDiode(c,6),tDiode(c,7),tDiode(c,8),tDiode(c,9), ...
56     tDiode(c,10), tDiode(c,11), tDiode(c,12));
57 end
58
59 for c = 1:componentCounts(2)
60     CapacitorELs(c) = CapacitorEL(tCapacitorEL(c,1),tCapacitorEL(c,2),...
61         tCapacitorEL(c,3), tCapacitorEL(c,4), tCapacitorEL(c,5),...
62         tCapacitorEL(c,6),tCapacitorEL(c,7),tCapacitorEL(c,8),...
63         tCapacitorEL(c,9), tCapacitorEL(c,10), tCapacitorEL(c,11),...
64         tCapacitorEL(c,12), tCapacitorEL(c,13));
65 end
66
67 for c = 1:componentCounts(3)
68     CapacitorCERs(c) = CapacitorCER(tCapacitorCER(c,1),tCapacitorCER(c,2),...
69         tCapacitorCER(c,3),tCapacitorCER(c,4), tCapacitorCER(c,5),...
70         tCapacitorCER(c,6),tCapacitorCER(c,7),tCapacitorCER(c,8));
71 end
72
73 for c = 1:componentCounts(4)
74     Resistors(c) = Resistor(tResistor(c,1),tResistor(c,2),tResistor(c,3),...
75         tResistor(c,4), tResistor(c,5),tResistor(c,6),tResistor(c,7));
76 end
77
78 for c = 1:componentCounts(5)
79     Cores(c) = Core(tCore(c,1),tCore(c,2),tCore(c,3),tCore(c,4), tCore(c,5),...
80         tCore(c,6), tCore(c,7), tCore(c,8), tCore(c,9),tCore(c,10),...
81         tCore(c,11), tCore(c,12));
82 end
83
84 for c = 1:componentCounts(6)
85     MOSFETs(c) = MOSFET(tMosfet(c,1),tMosfet(c,2),tMosfet(c,3),tMosfet(c,4), ...
86     tMosfet(c,5),tMosfet(c,6),tMosfet(c,7),tMosfet(c,8),tMosfet(c,9), ...
87     tMosfet(c,10), tMosfet(c,11));
88 end
89
90 for c = 1:componentCounts(7)
91     Wires(c) = Wire(tWire(c,1),tWire(c,2),tWire(c,3),tWire(c,4), ...
92     tWire(c,5),tWire(c,6));
93 end
94
95 for c = 1:componentCounts(8)
96     Shunts(c) = Shunt(tShunt(c,1),tShunt(c,2),tShunt(c,3),tShunt(c,4));
97 end
98
99 for c = 1:componentCounts(9)
100    Controllers(c) = Controller(tController(c,1),tController(c,2),...
101        tController(c,3),tController(c,4), tController(c,5), tController(c,6));
102 end
103
104 for c = 1:componentCounts(10)
105    TL431s(c) = TL431(tTL431(c,1),tTL431(c,2),tTL431(c,3));
106 end
107
108 for c = 1:componentCounts(11)
109    Optos(c) = Opto(tOpto(c,1),tOpto(c,2),tOpto(c,3),tOpto(c,4));
```

CURRICULUM VITAE

Name Surname: Giray Balcı

Place and Date of Birth: Bornova 09.04.1992

E-Mail: giraybalci1108@gmail.com

EDUCATION:

- **B.Sc.:** 2016, Istanbul Technical University, Faculty of Electrical and Electronics Engineering, Electrical Engineering Department
- **M.Sc.:** 2019, Istanbul Technical University, Graduate School of Engineering and Technology, Electrical Engineering Program

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2016-2019 R&D Engineer, Arçelik A.Ş.