Solution Optimizer for SMPS Applications

Master Thesis

Submitted by

Witoon Sintanavevong
Information and Media Technologies
Student ID 28103

Supervised by

Prof. Dr. rer. nat. Ralf Möller
Hamburg University of Technology
Software Systems Institute (STS)

Rainer Kling
Infineon Technologies AG
Power Management and Supply
AIM PMD CP M2

Munich, Febraury 2006

Declaration

Hereby I declare that I am the author of this thesis, titled “Solution Optimizer for SMPS Applications“. All literally or content related quotations from other sources are clearly pointed out and no other sources rather than the declared ones were used.

Munich, Febraury 2006
Witoon Sintanavevong

Acknowledgement

This report describes the Master thesis work that I have done during August 2005 to January 2006 at Infineon Technologies AG, Munich as a part of the Master of Science in Information and Media Technologies (IMT) at Hamburg University of Technology, Hamburg, Germany.

First of all, I would like to thank my family for the great continuous support, providing me an opportunity to study here in Germany.

I am very grateful to Prof. Dr. Ralf Möller for supervising this thesis as well as devoting his valuable time giving precious suggestions.

To my supervisor, Rainer Kling, for offering me an opportunity to do this project as well as supervising the work and reviewing the report.

To Harald Zöllinger, Dirk Ahlers, Michael Herfurth, Wolfgang Frank and other colleagues at Infineon for all supports and advises in electronic field in which I had no experience before.

To my friends for cheering me up in difficult times.

To myself for working hard.

This project could not be successful without the great support and cooperation from individuals and their willingness to share their expertise and knowledge and to devote their precious time to discuss related topics.

Table of Contents

Declaration i
Acknowledgement ii
Table of Contents iii
Table of Tables iv
Table of Figures v
Table of Listings vi
Chapter 1 Introduction 1
1.1 Objectives 2
1.2 Infineon Technologies AG 2
1.3 Structure of this Thesis 3
Chapter 2 Switched-Mode Power Supply 5
2.1 Basic principle of the switched-mode power supplies 6
2.2 Switched-mode power supply topologies 8
2.2.1 The feed forward converter 8
2.2.2 The flyback converter 10
2.3 Switched-mode power supplies with several output voltages 12
2.4 Selection criteria for switched-mode power supplies 12
2.5 Power factor 13
2.6 Circuit principles for the PFC stages 15
Chapter 3 Related Technologies 21
3.1 .NET Reflection 21
3.2 Web Service 22
Chapter 4 Optimization Strategy 23
4.1 Problem Characteristics 23
4.2 Problem Modeling 24
4.3 Algorithms for Optimization 25
Chapter 5 System Design 27
5.1 System Architecture 27
5.2 Software Architecture 28
5.3 Object Models 28
5.3.1 Topologies and modes 29
5.3.2 Products 30
5.3.3 Optimizers 31
5.4 Attribute Class 31
5.5 Data Structures 33
5.5.1 Limit Values 33
5.5.2 IPS Structure 33
5.5.3 IPS Warning 34
5.5.4 Rubycon Capacitor Structure 35
5.5.5 Multiple Outputs Structure 36
5.5.6 Output Structure 36
5.5.7 Default Value Structure 37
5.5.8 Advanced Inputs for Optimizer 37

5.6 Products 38
5.6.1 IPS Structure Sorted List 38
5.6.2 Default Value Structure Sorted List 39
5.6.3 Bill of Material 40
5.6.4 Automatic Value Recommendation 40
5.6.5 XML Import and Export 42
5.7 Optimizers 42
Chapter 6 Database Design 45
6.1 Table Schemas 45
6.2 Stored Procedures Declaration 50
Chapter 7 Implementation 53
7.1 Offline Client 53
7.1.1 SMPS Design 54
7.1.2 Optimizer 56
7.1.3 Check New DataSet 62
7.1.4 Customize Components 63
7.2 Web Service 65
7.3 Web Application 66
Chapter 8 Conclusion 69
Bibliography 71
Appendix A IPS Design XML Schema 73
Appendix B SMPS Design Procedure 77

Table of Tables

3.1 Comparison of feed forward and flyback converters 12
3.2 Power ranges for the various converter types 13
6.1 CachedQuery table schema 45
6.2 CachedSolution table schema 46
6.3 CoolSETs table schema 46
6.4 ElecCompValues table schema 48
6.5 HeatDissipaters table schema 48
6.6 MOSFETs table schema 48
6.7 Packagings table schema 48
6.8 ProductSeries table schema 49
6.9 RubyConCaps table schema 49
6.10 TransformerCores table schema 49
B.1 Design procedure for Flyback DCM using CoolSET-F2 77

Table of Figures

2.1 Conventional power supply without stabilization 5
2.2 Conventional power supply with stabilization of the output voltage 5
2.3 Block diagram of a switched-mode power supply 7
2.4 Overview of switched-mode power supplies 8
2.5 Single-phase feed forward converter 9
a) circuit diagram, b) and c) response characteristics in steady-state operation
2.6 Flyback converter 11
a) circuit diagram, b) and c) response characteristics in steady-state operation
2.7 Sinusoidal alternating current with examples of inductive and capacitive currents 13
2.8 Current and voltage waveforms for a typical power supply input circuit 14
2.9 Amplitudes of the harmonics and current or voltage waveform for a choke PFC solution 15
2.10 Charge pump PFC circuit 16
2.11 Schematic design of a boost converter for PFC operation 16
2.12 Schematic Current waveform for free-running operation 17
2.13 Current waveform for fixed-frequency operation 18
4.1 A tree model of the problem 24
5.1 System architecture 27
5.2 Major Components 28
5.3 Topologies and modes object model 29
5.4 Products object model 30
5.5 Optimizers object model 31
5.6 Populating property list of Product 39
5.7 Default Value Structure Sorted List 40
5.8 Automatic Value Recommendation Mechanism 41
5.9 Intermediate Steps 41
5.10 XML Import Export 42
5.11 Optimizer “getTargetProducts” method flowchart 44
7.1 Offline Client Splash Screen 53
7.2 Getting a new design from scratch 54
7.3 Getting a new design from an optimizer 54
7.4 SMPS design window 55
7.5 Input section of the SMPS design window 55
7.6 Data items section of the SMPS design window 56
7.7 Warning section of the SMPS design window 56
7.8 Optimizer window 57
7.9 Topology/mode dependent key input parameters 57
7.10 Multiple outputs tab page 58
7.11 Input missing notification 58
7.12 Maximum allowed junction temperature setting 59
7.13 a) Considered heat dissipaters, b) Considered transformer cores and c) Considered external MOSFET 59
7.14 Default value list 60

7.15 Neglecting non-critical warnings check box 60
7.16 Solutions tab showing the recommended optimal solution 61
7.17 Solutions tab showing all possible solutions 62
7.18 a) Check New DataSet command and b) Customize command 62
7.19 Customized heat dissipaters 63
7.20 Customized transformer cores 64
7.21 Customized external MOSFETs 64
7.22 Data update web service 65
7.23 Data update process 66
7.24 Web application 67

Table of Listings

5.1 Class Declaration in VB.Net (simplified): IPSPropertyAttribute 32
5.2 An attribute class instance 32
5.3 Class Declaration in VB.Net (simplified): LimitValues 33
5.4 Class Declaration in VB.Net (simplified): IPSStruct 34
5.5 Class Declaration in VB.Net (simplified): IPSWarning 35
5.6 Class Declaration in VB.Net (simplified): RebyConCapStruct 35
5.7 Class Declaration in VB.Net (simplified): FlyBackMultiOutputStruct 36
5.8 Class Declaration in VB.Net (simplified): OutputStruct 36
5.9 Class Declaration in VB.Net (simplified): DefaultValueStruct 37
5.10 Class Declaration in VB.Net (simplified): OptimizerAdvInputs 37
5.11 Class Declaration in VB.Net (simplified): Product 38
5.12 Class Declaration in VB.Net (simplified): OptimizerBase 42
7.1 Class Declaration in VB.Net (simplified): IPSDataUpdate 65

Chapter 1
Introduction

Nowadays, the Switched-Mode Power Supply (SMPS) plays an important role in the consumer power market. With its small size, it can be found everywhere from mobile phone chargers to Plasma TVs. It is used to convert the electricity from one voltage/current to the other voltage/current. For a simple example, a mobile phone charger converts the electricity from the plug with around 220VAC (in Europe) to 9VDC to supply the mobile phone.

Typically, the SMPS solution consists of several electronic components: a switching controller, a power MOSFET [1], a transformer, diodes, resistors, capacitors and inductors. In order to obtain the complete SMPS design, a long list of calculations has to be done in order to figure out the appropriate value for all electronic components.

Since the introduction of the Information Systems, hand calculations have been replaced by computer programs. However, those programs have just merely helped users reduce the calculation time and mistakes that could easily occur in hand calculations e.g. FlyCal[2] Calculation Software. In addition, some programs provide possibility to find the appropriate product from the given input parameters or to give a partial SMPS solution e.g. CCS[3] Calculation Software, FPS™ SMPS Toolkit[4] from Fairchild Semiconductor or PI Expert™[5] from Power Integrations.

Moreover, there are several modes and electronic network topologies giving different waveform characteristics for various applications. It is very complicated for non-expert users to understand all variables as well as to figure out which mode of which topology is appropriate for their applications. The different modes within the same topology result in minor calculation differences. On the other hand, different topologies result in major calculation differences.

Currently, if customers would like to have a complete SMPS design, they have to contact experts in the company. It takes sometime for customers to get designs. If we consider supporting the mass market in which there are lots of non-

---

[1] Metal-Oxide-Semiconductor Field-Effect Transistor

[2] an Excel based program used to calculate SMPS solution in Discontinuous Conduction Mode of the Flyback topology using Infineon products

[3] a Visual Basic based program used to find the appropriate Infineon CoolSET product for the given input parameters

[4] a Macromedia Flash based program providing step-by-step design of a fixed frequency flyback offline power supply and a quasi-resonant converter

[5] an interactive program that takes a user's power supply specifications and automatically determines the critical components (including transformer specs) needed to generate a working switch mode power supply.

expert customers but they want to buy only low amount of chips, the support cost would be very expensive and therefore not feasible. On the other hand, low responsiveness due to insufficient support staffs may lead to loss of business opportunities.

Optimization has become an interesting issue in industries so far. It has been applied in various processes in order to minimize the cost and maximize the profit. Thus, the SMPS design should be optimized to meet the customer requirements at the minimal cost. Normally, the optimization is done by experts in the company. Considering the long list of calculations, it is infeasible to manually try many combinations of electronic components. Thus, the proposed design might not be the most optimal one.

1.1 Objectives

To analyze the problem of choosing optimal electronic components in SMPS applications and design an appropriate object model. The object model has to be extensible and descriptive enough to be reused in the future. The optimization approach should be chosen to fit the scenario. User interfaces have to be implemented in order to prove the usability of the system.

1.2 Infineon Technologies AG

Infineon Technologies AG was founded in April 1999, when the semiconductor operations of parent company, Siemens AG, were spun off to form a separate legal entity. Today, Infineon has about 36,400 employees worldwide, 7.400 of them involved in research and development. In fiscal year 2005, the company achieved sales of 6.76 billion euros. The EBIT[6] came to 183 million euros[7].

Infineon designs, develops, manufactures and markets a broad range of semiconductors and complete system solutions used in a wide variety of microelectronic applications, including computer systems, telecommunications systems, consumer goods, automotive products, industrial automation and control systems, and chip card applications. Infineon Products include standard commodity components, full-custom devices, semi-custom devices, and application-specific components for memory, analog, digital, and mixed-signal applications. With a global presence, Infineon operates through its subsidiaries in the US from San Jose, California, in the Asia-Pacific region from Singapore and in Japan from Tokyo.

In financial year 2005, Infineon business is organized into three principal operating segments serving various markets in the semiconductor industry:

- Automotive, Industrial and Multimarket segment designs, develops, manufactures and markets semiconductors and complete system solutions for use in automotive, industrial and multimarket applications.

[6] Earnings Before Interest and Tax

[7] Infineon Internet source: http://www.infineon.com/cgi-bin/ifx/portal/ep/channelView.do?channelId=-65721&channelPage=%2Fep%2Fchannel%2FinformationPage.jsp&pageTypeId=17224

- Communication segment designs, develops, manufactures and markets a wide range of ICs, other semiconductors and complete system solutions for wireline and wireless communication applications.
- Memory products segment designs, develops, manufactures and markets semiconductor memory products with various packaging and configuration options and performance characteristics for standard, specialty and embedded memory applications.

1.3 Structure of this Thesis

This thesis consists of eight chapters and two appendixes. The second chapter describes fundamental of SMPS applications. However, the complete design procedure with examples is shown in the appendix B. The third chapter is dedicated for major related technologies which support the development of this project. The fourth chapter explains which optimization strategy is applicable and why it is suitable to use the certain approach. The fifth chapter describes about the system design while the XML Schema is shown in appendix A. The sixth chapter provides database table schemas as well as stored procedures declaration. The seventh chapter shows implementations which are results of this project. Finally, the last chapter concludes the whole project.

Chapter 2

Switched-Mode Power Supply

Nowadays, it can hardly imagine the lifestyle without the provision and processing activities which use electrical energy, and its supply. A host of devices in everyday use are operated either directly from the mains power grid, from the vehicle power supply in an automobile or using accumulators. The electronic circuits in modern devices in entertainment, data processing or industrial electronics are mostly supplied with direct voltages from 12 V down to below 1 V. To be able to operate these devices from the common alternating voltage mains network, or to change up the internal accumulators, a power supply is required.

Conventional power supply consists of a mains transformer for voltage reduction and galvanic isolation from the mains, a rectifier for producing a direct voltage and a bulk capacitor for voltage smoothing as shown in figure 2.1.

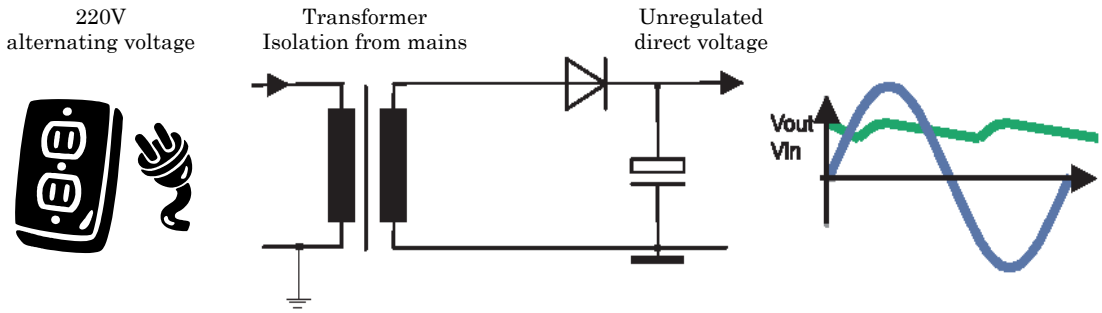


Figure 2.1: Conventional power supply without stabilization

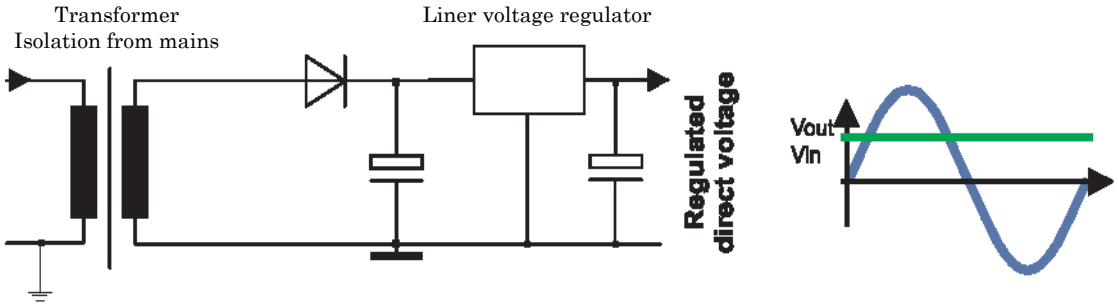


Figure 2.2: Conventional power supply with stabilization of the output voltage

A stabilized current or voltage supply, which is independent of fluctuations in the input voltage and load, is achieved by means of a stabilization section. Stabilization circuits generally have considerable losses, and call for a correspondingly large mains transformer as shown in figure 2.2. This increases the total losses, and causes heating of the device. In total, this results in an unsatisfactory efficiency, because when a mains overvoltage occurs the linear voltage regulator must absorb the entire power difference.

To an increasing extent, these conventional power packs are being superseded by so-called switched-mode power supplies. These have the following advantages:

- lower volume and weight, due to a smaller transformer and smaller capacitors
- better efficiencies, and hence less heating
- lower losses in standby mode
- the possibility of operating the power supply over a wide input voltage range (85- 245 V)
- electronic safety functions for fault situations, such as short circuits

In the age of globally marketed electronic devices, the wide possible input voltage range of these power supplies is especially necessary. As an example, a device developed in Germany may be sold and operated in countries with a 110 V mains power supply (including the USA, Japan) or in Europe or Asia (220 V mains power supply).

Ever more electronic devices are no longer disconnected from the mains by a mains switch when they are not in use (TVs, video recorders, PCs, chargers etc.). Instead, the devices are put into a standby mode, and thus consume unnecessary energy. There is thus an increasing demand for the power consumption in standby mode to be reduced, which has now become very efficiently feasible, using switched-mode power supplies.

2.1 Basic principle of the switched-mode power supplies

Switched-mode power supplies (SMPSs) are power supplies which chop the rectified and filtered mains voltage at a frequency which is significantly higher than the 50 Hz of the mains alternating current as shown in figure 2.3. By using the semiconductors exclusively as switches, only switching and forward losses arise. This is responsible for the characteristically high efficiency of a pulsed power supply by comparison with analog methods. Regulation is effected either by altering the pulse duty factor at a constant frequency, or by changing the frequency for a fixed or variable pulse duty factor. The voltage chopped in this way can be transformed to any other required voltage, and rectified. The frequency of this alternating voltage, which may be rectangular, trapezoidal or sometimes even sinusoidal, lies in a range from a kHz up to several 100 kHz. As a consequence of this high working frequency, smaller transformers with ferrite cores can be used. The ferrite core transformer serves not only to make the required voltage

conversion and provide galvanic isolation from the mains supply but also, depending on its working principle, to store the magnetic energy.

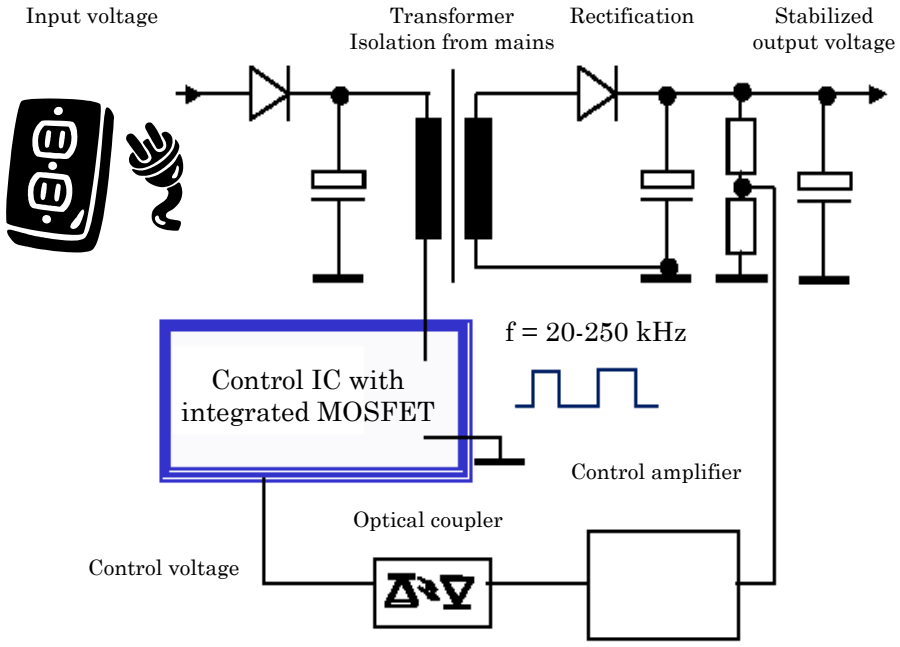


Figure 2.3: Block diagram of a switched-mode power supply

With SMPSs, the pulsing results in harmonic waves, which can cause radiated interference to radio and TV reception, as well as to communication transmissions. Legislation demands a limit on the interference signal levels for all electrical devices and systems which produce high frequency energy.

An SMPS which is powered from the alternating voltage mains and which supplies a DC voltage at its output is also called an AC-DC converter. If a direct current source (e.g. an AC-DC converter, or a battery) is connected to the input, then we speak of a DC-DC converter, or a switching regulator. If the voltage is not rectified at the output, the device is a DC-AC converter or inverter. If the SMPS is supplied from the mains, and there is no rectification on the output side, we have an AC-AC converter, i.e. an alternating current converter.

The main application fields for SMPSs are:

- Consumer electronics: TVs, DVD players, video recorders, set top boxes, satellite receivers, chargers and external power supply units.
- Electronic DP: PCs, servers, monitors, notebooks
- Telecommunications: mobile communication base stations, switching stations, mobile phone chargers
- Industrial electronics: open and closed loop control engineering, measuring instruments, auxiliary power supplies, battery chargers etc.

2.2 Switched-mode power supply topologies

The working principle of an SMPS essentially determines its characteristics and production cost. The basic circuits of the most frequently used AC/DC and DC/DC converters will now be described. A basic distinction is made between two different conversion principles: the feed forward converter and the flyback converter as shown in figure 2.4.

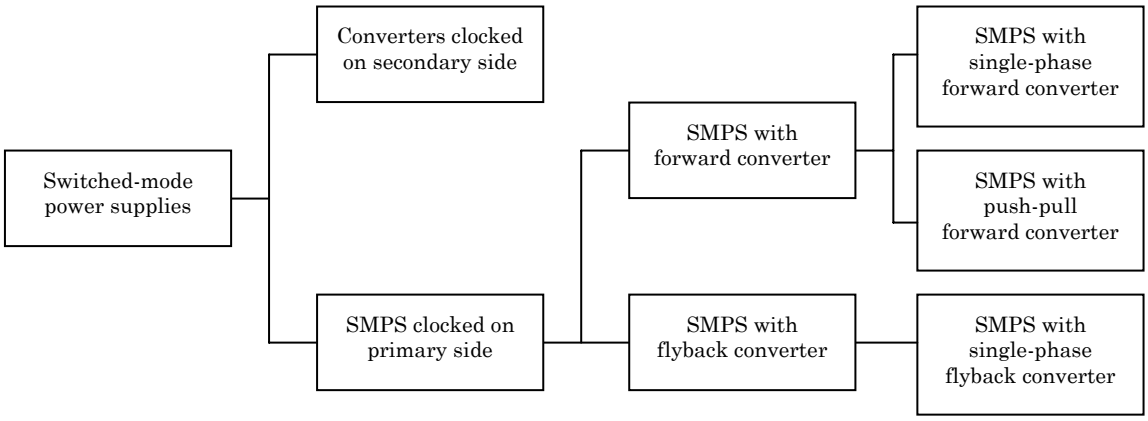


Figure 2.4: Overview of switched-mode power supplies [IFX04]

The name feed forward converter is explained by the response characteristics of the arrangement, with which an energy flow arises between the primary circuit and the secondary circuit during the semiconductor's blocking phase. On the primary side, the load current is superimposed on the magnetizing current. For this reason, conditions must be created to allow the transformer to be demagnetized again. With a single-phase feed forward converter this occurs during the semiconductor switch's conducting phase. With push-pull and bridge circuits, the conducting phase of one semiconductor switch is followed, after a short blocking phase for both semiconductor switches, by the conducting phase for the second. With a flyback converter, energy is accumulated in the converter during the conducting phase of the semiconductor switch, and is then given up on the secondary side during the blocking phase.

2.2.1 The feed forward converter

Figure 2.5 shows the circuit diagram of a single-phase feed forward converter. On its input side, this converter has a smoothing capacitor C_{IN}. This performs the functions of smoothing the rectified mains voltage, providing a low-inductance supply of the pulsed currents required by the converter, and absorbing the magnetizing current which is fed back from the transformer. The transformer of the feed forward converter has a ferrite core with no airgap in order to achieve a high magnetic coupling of the windings. The primary winding is connected to the

input voltage by a switching transistor. When the transistor is conducting, an induced voltage arises in the secondary winding with a rectangular waveform and magnitude corresponding to the transformation ratio, which produces a current flow in the secondary winding through the rectifier diode D_1 and the smoothing choke L_{OUT} on the output side as shown in figure 2.5b and 2.5c with time period T_{on}.

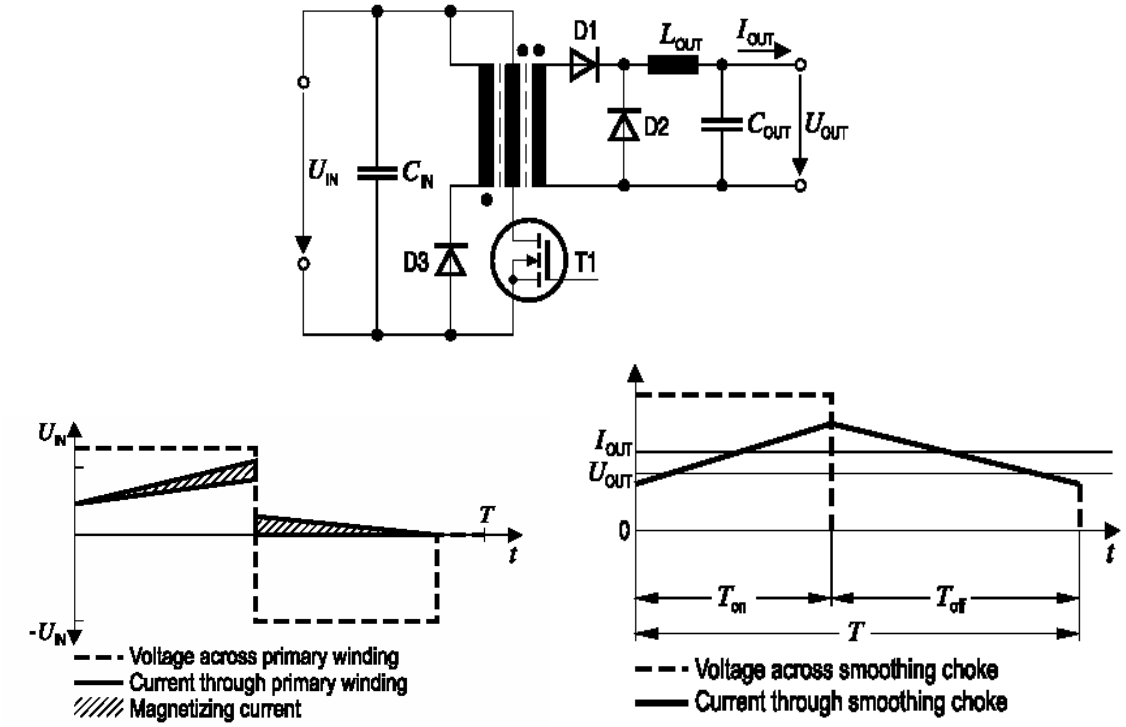


Figure 2.5: Single-phase feed forward converter [IFX04]
a) circuit diagram, b) and c) response characteristics in steady-state operation

The current in the secondary winding induces a current in the primary winding corresponding to the transformation ratio. In addition to this load current, superimposed on it in the primary winding is the so-called magnetizing current. The magnetizing current must be allowed to decay during the transistor's blocking phase. This is affected by a demagnetization winding, which serves as a third transformer winding. It has the same number of turns as the primary winding, but with a smaller conductor cross-section because only the magnetizing current flows through this winding, during the transistor's blocking phase. Compared to the primary and secondary windings, the demagnetization winding has reversed polarity, indicated in the circuit diagram by the dots at the start of each winding. The demagnetization winding is connected directly to the input voltage, via a diode D_3.

During the transistor's conducting phase, the same voltage is induced in the demagnetization winding as there is in the primary winding, so that twice the input voltage arises at the diode D_3, in its blocked direction. During the transistor's blocking phase, the energy which has been built up in the transformer core as a

result of the magnetizing current must be released again, so that the magnetizing current does not grow arbitrarily large and the ferrite core does not reach magnetic saturation. For this reason, a freewheeling diode D_2 is provided in the secondary circuit, through which the current flows on through the smoothing choke L_{OUT} when the voltage in the secondary winding drops to zero or goes negative. During this phase of operation, the diode D_1 decouples the current loop on the secondary side from the transformer. This allows the polarity in the windings to reverse.

The magnetizing current now flows back through the diode D_3 and the demagnetization winding into the smoothing capacitor on the input side. As a result, twice the input voltage is now applied to the transistor as the blocking voltage. The demagnetization of the transformer is assured if the voltage/time area, that is the area enclosed under the voltage in the primary winding along the time axis for the duration of the demagnetization, is at least equal to the corresponding area for the switched-on state. For this reason, the duration of the on-state for a single-phase feed forward converter may not amount to more than 50% of the cycle duration.

The purpose of the smoothing choke L_{OUT} is to generate a steady energy flow from the current or voltage signal, as appropriate, which arises across the secondary winding of the transformer during the switched-on period Ton, and to limit the current rise in the transformer. The smoothing capacitor C_{OUT} at the output smoothes the current ripple from the choke, and acts as an energy store during load changes. The response characteristics of the feed forward converter are described by the following formula:

$$U_{OUT} = U_{IN} \cdot \frac{n_1}{n_2} \cdot \frac{T_{on}}{T}$$

n_1 number of turns in primary winding,
n_2 number of turns in secondary winding.

2.2.2 The flyback converter

These converters also have a smoothing capacitor C_{IN} on the input side, with the functions of smoothing the rectified mains voltage, and providing a low-inductance supply of the pulsed currents required by the converter. Unlike the single-phase feed forward converter, here the magnetizing current is not fed back to the input capacitor, but to the smoothing capacitor C_{OUT} on the output side as shown in figure 2.6.

In its basic form, the flyback converter has two windings, with opposite polarization. When the switching transistor T_1 is switched on, the anode-cathode voltage for the rectifier diode D_1 is negative, i.e. no current flows in the secondary winding of the transformer. The magnetizing current flows in the primary winding. In a flyback converter, a ferrite core is used which has an air gap, as a result of which a considerably larger inductive current flows and this establishes a magnetic field in the air gap. In addition, when the transistor is switched on magnetic energy is stored in the transformer of the flyback converter (predominantly in the air gap).

When the transistor is blocked, the voltage across the windings reverses. The voltage across the secondary winding rises until the rectifier diode D_1 becomes conducting, i.e. at the value of the output voltage U_{OUT}. Because the magnetic flux in the transformer is steadily changing, at the point in time when the transistor is blocked a current, corresponding to the primary current transformed by the transformation ratio, will flow in the secondary winding. For this reason, the rectifier diode D_1 must feed directly to a capacitor C_{OUT} which is capable of accepting the high current.

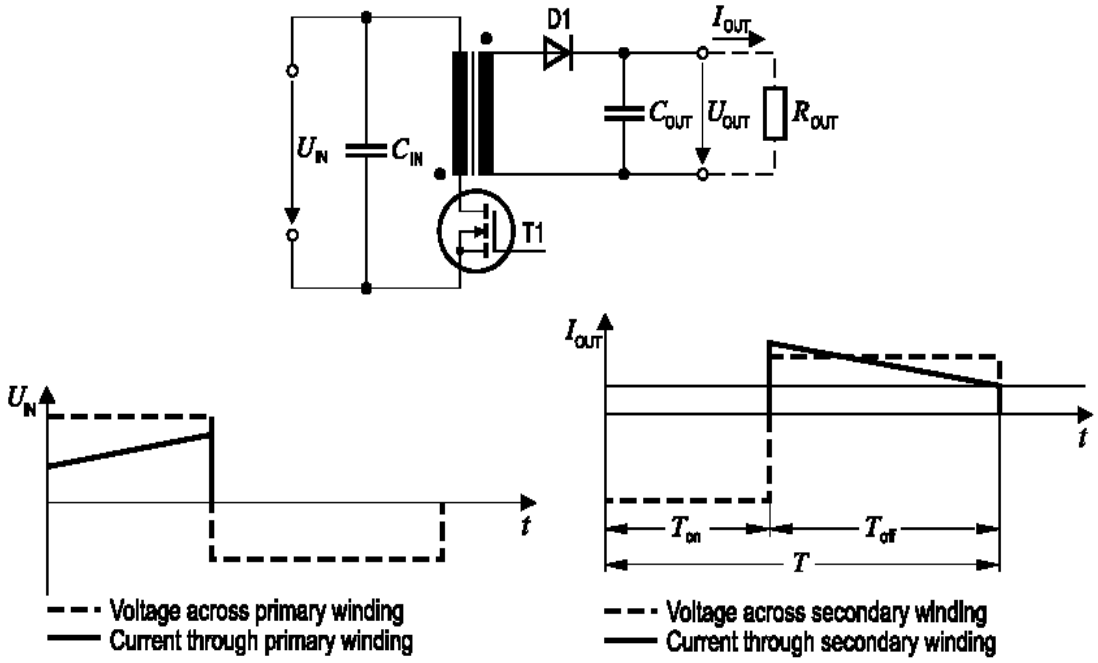


Figure 2.6: Flyback converter [IFX04]
a) circuit diagram, b) and c) response characteristics in steady-state operation

During the time the transistor is switched off, the blocking voltage applied to it will be the input voltage, plus the output voltage transformed in accordance with the transformation ratio. With the standard dimensioning, this represents somewhat more than twice the input voltage. We now distinguish between forward converters with a trapezoidal current waveform in the transformer, and those with a triangular current waveform. With the trapezoidal waveform as shown in figure 2.6b and 2.6c, the transistor is switched on again at a point in time which is before the current in the secondary winding reaches zero. An important characteristic of this form of operation is that the maximum values which occur for the current are, relative to the output current, significantly lower than when operated with a triangular current waveform.

The response characteristic for a trapezoidal current waveform is represented by the formula:

$$U_{OUT} = U_{IN} \cdot \frac{n_2}{n_1} \cdot \frac{T_{on}}{T} \cdot \frac{1}{1 - \frac{T_{on}}{T}}$$

From this it will be seen that the output voltage U_{OUT} changes if the pulse duty ratio T_{on} / T is altered. However, the relationship between the output voltage and the pulse duty ratio is not linear, but hyperbolic. This means that the output voltage will become infinitely large if the pulse duty ratio approaches close to 1. For this reason, flyback converters must not be operated without a load resistance or a closed control loop, because the output voltage, and with it also the blocking voltage for the transistor, can assume high values.

2.3 Switched-mode power supplies with several output voltages

With a switched mode power supply which has several output voltages, one of these voltages must be chosen to provide the controlled variable, as a consequence of which this will be the best regulated voltage. The regulation of the other voltages will be less exact. For this reason, it may be necessary to stabilize them by means of longitudinal regulators or pulsed secondary regulators. The last named switching regulators represent converters with a secondary clock pulse control.

2.4 Selection criteria for switched-mode power supplies

Table 3.1 provides a comparison of feed forward and flyback converters, Table 3.2 shows the powers which can be achieved by the different types of converter.

Table 3.1: Comparison of feed forward and flyback converters [IFX04]

	Feed forward converter	Flyback converter
Number of components required	Greater	Fewer
Smoothing choke and freewheeling diode	Required	Not applicable
Transformer core	Without air gap	With air gap
Magnetic coupling during switching processes	Better	Worse
Voltage excess during switching processes	Smaller	Larger
Susceptibility to influence of magnetic fields	Smaller	Larger
Current amplitudes relative to load current	Significantly smaller	Substantially larger
Pulsed current loading of components	Smaller	Larger
Interference suppression and smoothing of input and output quantities depending on	Simpler	More expensive

	Feed forward converter	Flyback converter
interference modes		
Energy flow controlled by changing the pulse duty ratio by	Varying the voltage / time integral	Storing variable proportions of energy
Creation of several strictly regulated secondary DC voltages at the same time, by adding further secondary circuits	Possible to a limited extent (choke current must be continuous)	Easily possible
Regulation dynamics of output quantities	Slower (due to choke smoothing)	Faster

Table 3.2: Power ranges for the various converter types [IFX04]

Power (W)	≤ 100	100…300	300…1000	1000…3000	≥ 3000
Single-phase flyback converter	×	×			
Single-phase feed forward converter	×	×			
Half-bridge converter		×	×		
Full-bridge converter		×	×	×	
Push-pull converter		×	×	×	×

2.5 Power factor

The power factor is the technical term for the ratio of the real power to the complex power, and takes a value between 0 and 1. The power factor is therefore defined as the cosine of the phase difference between a sinusoidal voltage and the associated current waveform, i.e. PF = cos (φ_{UI}) if the system only contains linear loads. With a purely ohmic load, e.g. an incandescent lamp or a radiator element, the voltage and current are in phase, and thus the power factor is 1. With an inductive load (e.g. an asynchronous motor) or a capacitive load, there is a phase shift and the power factor is < 1 as shown in figure 2.7.

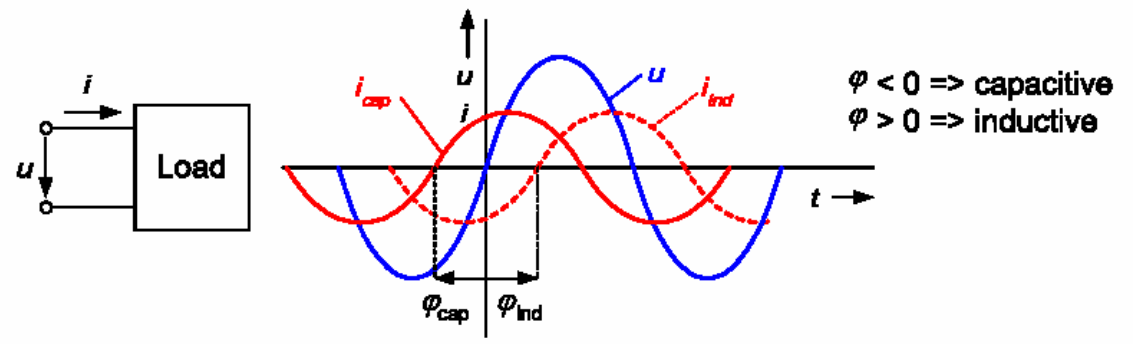


Figure 2.7: Sinusoidal alternating current with examples of inductive and capacitive currents [IFX04]

However, a power supply unit with an input rectifier and a downstream smoothing capacitor (figure 2.8) represents a non-linear load. The current which flows is pulsed, and is in phase with the input voltage provided that this input voltage is greater than the voltage across the smoothing capacitor.

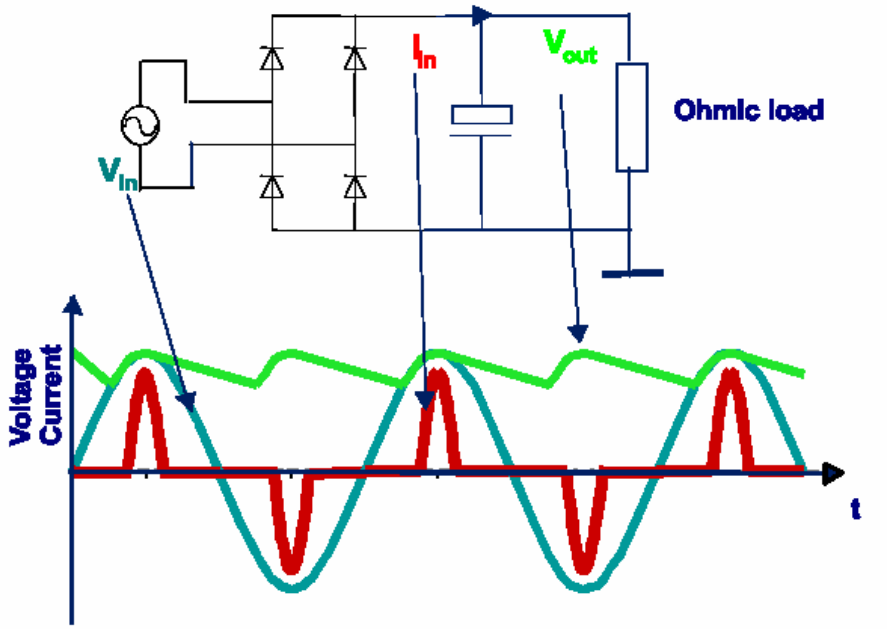


Figure 2.8: Current and voltage waveforms for a typical power supply input circuit [IFX04]

The current flow is indeed broadly in phase with the input voltage, but the spectral distribution shows a strong harmonic content. Apart from the fundamental wave, therefore, the current also has components with frequencies which are an integral multiple of the fundamental wave. These deviations from a sinusoidal current are referred to as the THD (total harmonic distortion):

$$THD = \frac{\sqrt{\left(I_{RMS}^2 - I_1^2\right)}}{I_{1RMS}^2}$$

I_1 = Effective value of the fundamental oscillation
I_{RMS} = Effective value of the total current

For a purely sinusoidal current, THD = 0. The greater the deviation is from the sinusoidal shape, the larger is the THD. The power factor (PF) is then determined, taking account of the non-linear components, as:

$$THD = \frac{\cos(\varphi_1)}{\sqrt{1+THD^2}} = \frac{I_{1RMS} \cdot \cos(\varphi_1)}{I_{RMS}}$$

where φ_1 = phase shift between the input voltage and the fundamental current wave.

From this it can be seen that if there is strong distortion of the current the power factor will be < 1. Typical values for switched mode power supplies lie around PF = 0.6 … 0.7. An example of a practical implication of the high THD value is that considerable transient currents flow in the house wiring, requiring wires with a larger cross-section. In general, the effect of a lower power factor is that power stations must make available a considerable reactive power, which is practically unused in the load. As a result, power stations cannot be operated optimally, imposing additional loads on the environment.

PFC (Power Factor Correction) circuits represent a solution to this problem, making a power factor of virtually 1 possible in the ideal situation. In 2001 the norm EN-61000-3-2 (IEC 1000-3-2), on limiting values for harmonic currents in the mains power supply, was introduced. According to this, devices such as TVs, monitors and PCs with a power consumption of > 75 W must adhere to certain limiting values for harmonics. All fluorescent lamps are covered by the norm. A similar standard applies in Japan, so far in the USA only fluorescent lamps are subject to a PFC condition.

2.6 Circuit principles for the PFC stages

Passive PFC stages: the simplest solution, and generally the least expensive, consists in inserting an appropriately sized iron-cored choke before the input rectifier of the power supply unit. In this way, power factor values of < 0.9 can be achieved and harmonic amplitudes which lie just beneath the limiting values as shown in figure 2.9.

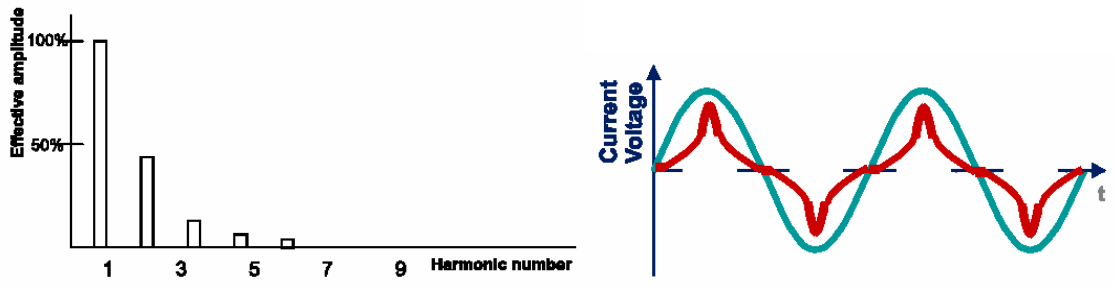


Figure 2.9: Amplitudes of the harmonics (left) and current or voltage waveform for a choke PFC solution [IFX04]

The disadvantages of this solution are the high weight and volume of the PFC choke, while the sensible power limit is at < 200 W. A similar quality is achieved by using a so-called charge pump as shown in figure 2.10.

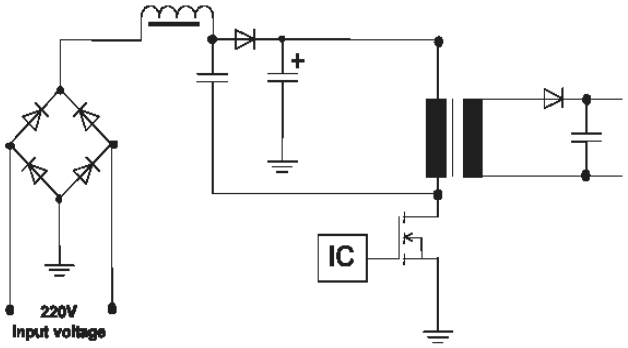


Figure 2.10: Charge pump PFC circuit [IFX04]

The choke sits behind the rectifier, and is exposed to the pulse frequency of the switched mode power supply, so that a ferrite-cored choke can be used which has significantly smaller dimensions than for an iron-cored choke. The power limit lies at about 250 W. Consequently, this solution is very well suited to applications which must satisfy the standard but for which an optimal power factor correction is not being sought. The additional weight and volume are small. Typical applications are power supplies in TVs or adapters.

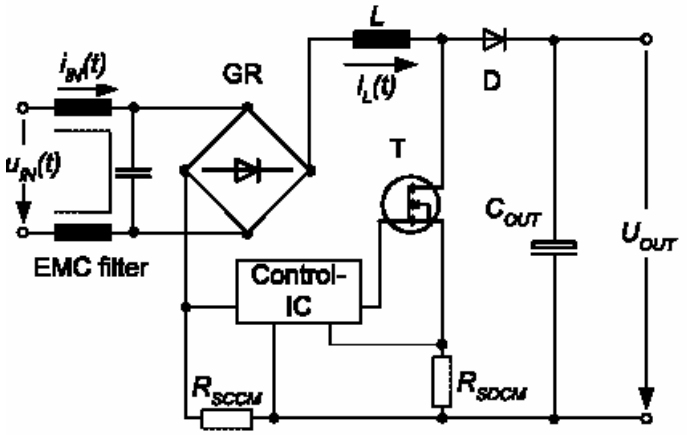


Figure 2.11: Schematic design of a boost converter for PFC operation [IFX04]

Active PFC stages: a power factor of almost 1 can be achieved using active PFC circuits. By comparison with the passive solutions, optimal power factor correction and a reduction in the harmonic content are achieve over a significantly wider range of input voltages and loads, combined with a higher efficiency. To the mains supply, the circuit ought to appear as an ohmic load. Weight and volume are similar to that of a charge pump circuit that is significantly more favorable than the iron choke solution. The active PFC circuits are connected upstream of the switched mode power supply. Basically, three topologies are used: boost converter, flyback converter and buck/boost converter. However, the most-used principle is

the boost converter (figure 2.11), which converts a sinusoidal input voltage into a direct voltage output, with a value greater than the peak value of the sine-wave voltage.

This constant high input voltage to the following switched mode power supply has the advantage that the switching transistor and the input capacitor can be made smaller than in the case of a power supply for a wide range of input voltages which does not have a PFC boost converter. In general, the input capacitor for the converter is completely omitted, because the output capacitor COUT can, as shown in figure 2.11, completely replace it. The drive is either free-running, i.e. with a variable pulse frequency, or at a fixed pulse frequency. With a free-running PFC converter, within each pulse period the boost converter choke L is first magnetized and then is always fully demagnetized. This is shown by the fact that the choke current $i_L(t)$ at the end of a pulse period has returned to 0 again. Immediately after this, a new pulse period begins, with another magnetization of the choke. The result is that a triangular current waveform $i_L(t)$ arises. The converter is thus operated right up to the point of discontinuity in the choke current, so that this mode of operation is also called discontinuous conduction mode (figure 2.12).

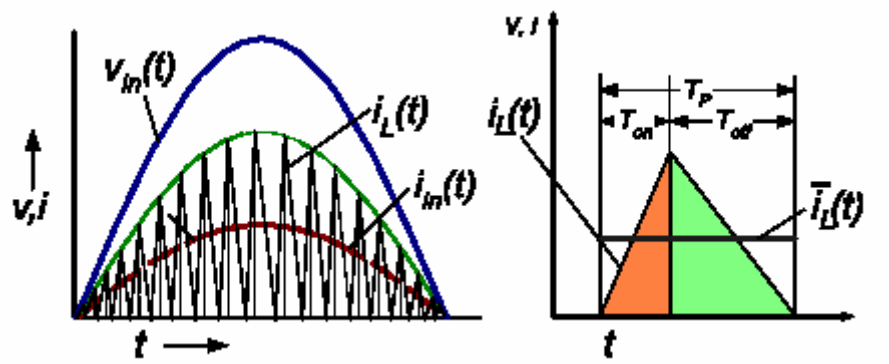


Figure 2.12: Schematic Current waveform for free-running operation [IFX04]

The time trace of the choke current over one pulse period can be described in accordance with the law of induction as:

$$i_L(t) = \frac{1}{L} \cdot \int_0^t u_{\text{in}}(t)dt = \frac{u_{\text{in}}(t)}{L} \cdot t \qquad (1)$$

$$0 \leq t \leq T_{\text{on}}$$

$$i_L(t) = I_0 - \frac{U_{\text{out}} - u_{\text{in}}(t)}{L} \cdot (t - T_{\text{on}}) \qquad (2)$$

$$T_{\text{on}} \leq t \leq T_p$$

Here, I_0 is the final value of the choke current which is reached during the magnetization period up to the time T_{on}, i.e. $I_0 = u_{in}(t) \cdot T_{on} / L$. As the duration of a pulse period, T_P, is much smaller than that of a mains supply cycle, i.e. $T_P << T_{Mains}$, one may assume as a first approximation that within any one pulse period the values of the voltages and currents are constant. Taken with (1) this gives the mean value of the choke current as:

$$\begin{aligned} \overline{i_L}(t) &= \frac{1}{T_p} \cdot \int_0^{t_p} i_L(t)\,dt = \\ &= \frac{1}{T_p} \cdot \frac{I_0 \cdot T_p}{2} = \frac{T_{on}}{2L} \cdot u_{in}(t) = \\ &= konst \cdot u_{in}(t) \end{aligned} \qquad (3)$$

Hence, within a pulse period T_p, the mean current $i_L(t)$ through the choke, which is also the input current, depends linearly on the instantaneous value of the input voltage $u_{in}(t)$. A prerequisite for this is, however, that the time T_{on} for which the MOSFET T is switched on is correspondingly constant. If one now puts an infinite number of pulse periods one after another, one gets a continuous current flow which is proportional to the input voltage.

If a PFC converter is driven using a fixed frequency, the boost converter choke is never fully demagnetized or discharged, i.e. the transistor switches into the next pulse period while there is still a choke current $i_L(t)$ flowing. The choke current is therefore trapezoidal in form and has a significant ripple. Operation in this manner with no breaks in the current is also referred to as continuous conduction mode (figure 2.13).

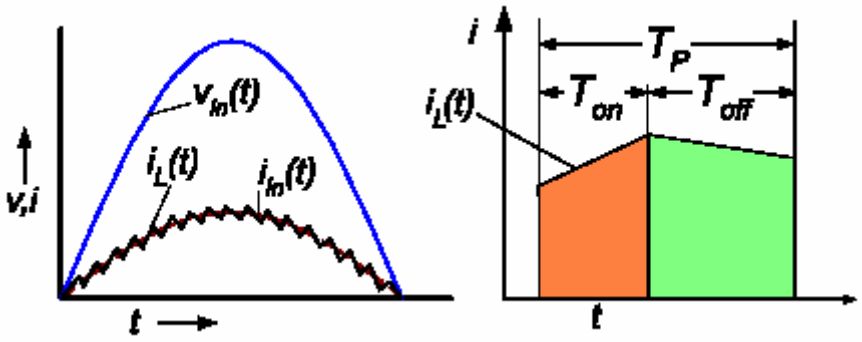


Figure 2.13: Current waveform for fixed-frequency operation [IFX04]

In contrast to the free-running PFC concept, the current between ground and the negative connection on the bridge rectifier is measured via the resistance R_{SCCM}, and is shown in figure 2.13. This measures the entire choke current. This signal is smoothed and multiplied by a signal which corresponds to the instantaneous value of the input voltage. The set value for the pulse width modulator is thus

$$i_{set}(t) = I_{AVG} \cdot k_U \cdot u_{in}(t) = konst \cdot u_{in}(t) \qquad (4)$$

The value I_{AVG} here represents the smoothed value (mean/average) of the choke current, which is assumed to be constant over a pulse period. For this reason, this type of control is also called average current control. The factor k_U is a scaling factor with the units 1/V. From equation (4) it is easy to see the linear relationship between the current set value and the input voltage. The current set value is then converted by a pulse width modulator into a drive signal for the switching transistor.

The decision as to which method is most suitable for which application depends ultimately on the system costs. Very broadly, the discontinuous conduction mode is today preferred for powers up to about 200 W, and above this the continuous conduction mode. Because of the high peak currents, the cost of interference suppression is higher for the discontinuous conduction mode, but on the other hand the requirements to be met by the diodes, in terms of blocking characteristics, are much more critical for the continuous conduction mode. For this reason, silicon carbide diodes (SiC) are especially suitable for this mode of operation.

Chapter 3

Related Technologies

This chapter gives rough information of key related technologies used in this project. There are two key technologies chosen to be explained here: .NET Reflection which plays an important role in dynamic properties access and Web Service which is now widely used to enable interoperability of softwares across a computer network.

3.1 .NET Reflection

The .NET Framework is an integral Windows component for building and running software applications and Web services. The multiple-language capability of the .NET Framework enables developers to use the programming language that is most appropriate for a given task and to combine languages within a single application. Components written in different languages can consume functionality from each other transparently, without any extra work required from the developer. Support for the .NET Framework has been announced for over twenty commercial and academic programming languages. The component-based, plumbing-free design of the .NET Framework minimizes the amount of code developers have to rewrite and maximizes potential for code reuse. As a result, developers can focus on the core business logic code. The framework is composed of the common language runtime and a unified set of class libraries

The Common Language Runtime (CLR) is responsible for run-time services such as language integration, security enforcement, and memory, process, and thread management. In addition, the CLR has a role at development time when features such as life-cycle management, strong type naming, cross-language exception handling, and dynamic binding reduce the amount of code that a developer must write to turn business logic into a reusable component.

The CLR loader manages application domains. The management includes loading each assembly; a fundamental unit of deployment, version control, reuse, activation scoping, and security permissions in .NET framework, into the appropriate application domain and controlling the memory layout of the type hierarchy within each assembly.

Assemblies contain modules, modules contain types, and types contain members. Reflection provides objects that encapsulate assemblies, modules, and types. Reflection can be used to dynamically create an instance of a type, bind the

type to an existing object, or get the type from an existing object. Invocation of the type's methods and access to the type's fields and properties can be done afterwards.

PropertyInfo class is used to discover information such as the name, data type, declaring type, reflected type, and read-only or writable status of a property, and to get or set property values.

In the .NET Reflection components, there is a class called an *Attribute* class that can be tagged to any piece of code (e.g. class, property, method, etc.) to give additional information. Object instances of this class can be read using .NET Reflection at run-time. Further detail regarding .NET reflection can be found at [MSDN05].

3.2 Web Service

A Web Service is a software system designed for supporting interoperable machine to machine interactions over a network. It has a machine-processable descriptive language, Web Services Description Language (WSDL), to describe interfaces. Interactions between machines can be done using messages as defined in the interface. The message may be enclosed in a Simple Object Access Protocol (SOAP) envelope. Typically, the message is encoded in XML-format and transmitted through HyperText Transfer Protocol (HTTP).

Computer applications developed in different languages running in different platforms could make use of the web service to interchange data over a computer network e.g. Internet in a manner like inter-process communications on a single computer. Due to the usage of open standards, softwares developed in different platforms can now interoperate smoothly and effectively without much effort.

Chapter 4

Optimization Strategy

This chapter describes characteristics of the problem discussed in this project. The problem is then modeled in a proper structure which can be solved. Algorithms for optimization are discussed. The suitable algorithm is chosen. Finally, it explains how the chosen algorithm can be realized.

4.1 Problem Characteristics

In order to generate a SMPS design, a design procedure (see example in Appendix B) has to be done. Typically, it consists of half a hundred up to over a hundred steps varying from topology to topology and mode to mode. Each step can be either an input parameter or a result from an equation. The sequence of steps is very important since previous steps have much influence on the following steps. Numbers of input parameters have to be given at the beginning along with an IC which defines additional input parameters. During the calculation, several major components have to be chosen manually. Afterwards, the other components are adjusted in order to give the preferred result. Note that the set of major electronic components e.g. IC, MOSFET is not large.

In addition, there are some constraints that need to be considered. Constraints can be categorized into two types: critical constraints and insignificant constraints. As its name, critical constraints must not be violated by all means e.g. the maximum junction temperature must not exceed 150°C. On the other hand, insignificant constraints are just warnings that should be considered e.g. the current density should not exceed 8A/mm^2. Constraint enforcement in some steps may result in changing values in the previous steps.

Since electronic components have a limited set of possible values, it is not possible to find a single component with the exact preferred value. Additionally, it would cost more money to have them in series or parallel if it is not critical. Normally, the minimal number of components is preferred. The simple solution is to choose the component with the closest value. However, as mentioned before, the chosen component value largely affects the following equations therefore the new value has to be used for the rest of the calculation to get more precise results.

Manual validation of all combinations is not feasible therefore only potential combinations are validated. Currently, figuring out whether a combination has potential or not, requires lot of experience.

4.2 Problem Modeling

The problem can be modeled as a tree as shown in figure 4.1. A leaf node represents a complete combination of electronic components. Each intermediate level represents each step choosing an electronic component. The Number of intermediate levels varies from topology to topology as well as from mode to mode. Constraint validation could be done at every intermediate level to stop traversing unnecessary leaf nodes in case that intermediate node violates some constraint.

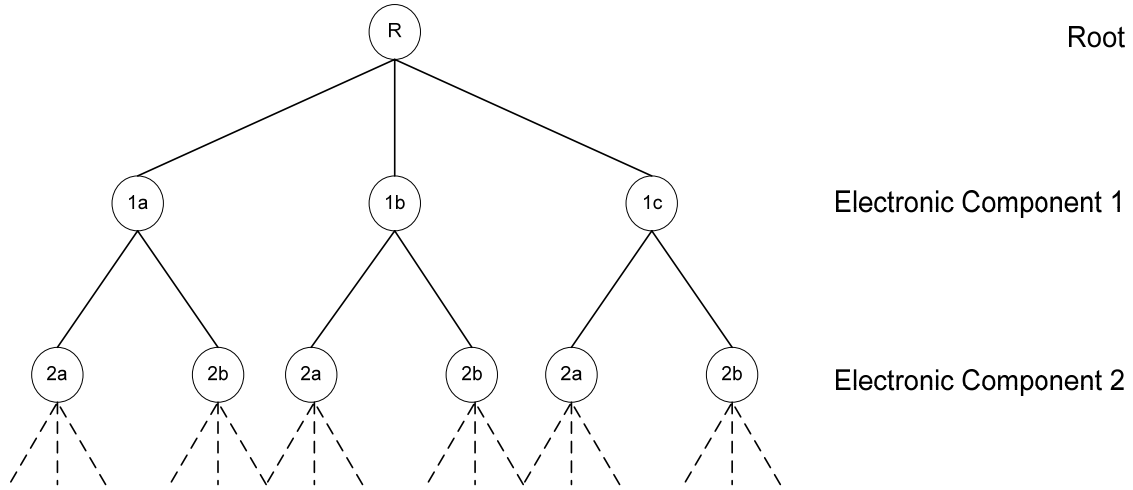


Figure 4.1: A tree model of the problem

The electronic components discussed above do not include the basic electronic components e.g. resistors, capacitors and inductors in which they can be easily addressed by finding a component with the closest value to the calculation. Therefore the number of total possible combinations can be expressed as:

$$\prod_{i=1}^{n} X_i$$

where n is the total number of major components and
X_i is the number of possible values of the major component i

As mentioned in the section 4.1, the number of possible values of each major component is not large (less than a hundred). In addition, typically only several major components involve. As a result, the number of total possible combination will not be too large. Moreover, with the constraint validation, it would reduce the number of combinations dramatically because the child nodes of the intermediate node violating some constraint will be excluded.

4.3 Algorithms for Optimization

According to [Hromkovic04], when one gets a new algorithm problem, the most reasonable approach is to look into the following robust and paradigmatic techniques whether one of them alone could efficiently solve the problem or not. The first technique is divide-and-conquer. This technique recursively breaks the given problem instance into many problem subinstances. It is easier for problem subinstances to be solved. By combining problem subinstance solutions, it leads to the solution of the original problem instance.

Dynamic programming is another technique similar to the divide-and-conquer method. They both solve problems by combining solutions of subproblems. The difference between these two techniques is that divide-and-conquer recursively breaks problem instances into subinstances and calls itself on these subinstances, while dynamic programming works in a bottom-up approach by starting with computing solutions of smallest subinstances and continuing to larger and larger subinstances until the original problem instance is solved. The main advantage of dynamic programming is that it solves a problem subinstance only once no matter how many times this problem subinstance appears. On the other hand, the divide-and-conquer may solve the same problem subinstance many times.

The next technique is backtracking which is used to solve optimization problems by a possibility exhaustive search of the all feasible solutions set. Moreover, a feasible solution is only looked once. Another technique is local search which defines a neighborhood in the all possible solutions set and then searches in the set going from a feasible solution to a neighboring feasible solution if the cost of the neighboring solution is better than the cost of the original solution. A local search algorithm stops with a feasible solution that is a local optimum according to the defined neighborhood.

The last technique, greedy algorithm, is based on a sequence of steps. Every step the algorithm specifies one parameter of a feasible solution. The name greedy comes from the way it chooses the parameters. It always chooses the most promising choice from all possibilities which then used to specify the next parameter, and no decision is reconsidered later on.

Taking all algorithms described above in to the consideration, it is found that the backtracking technique is the most suitable algorithm for solving the problem in SMPS applications domain. The divide-and-conquer and the dynamic programming are not applicable because the problem is already atomic and the sequence of calculations is very important. The sequence highly influences calculation results. The local search approach can not be applied since at the beginning, there is no information of neighboring solutions. The greedy algorithm is also not suitable due to lacking of reconsideration which easily leads to invalid solution. In addition, the complexity of the problem in SMPS applications is relatively low therefore it is feasible to do exhaustive search over all possibilities. However, with the domain-specific knowledge, it is possible to considerably reduce a number of possibilities by validating the chosen items against the defined rules for every step of item selection. If the combination is not valid, it is obvious that it must go back one step and try another combination. All valid combinations are considered as solutions.

Chapter 5

System Design

In this chapter, the system architecture, software architecture and object models are depicted. Attribute class which plays an important role in giving descriptive information about each property, is defined. Major classes are roughly explained in order to understand their features and responsibility.

5.1 System Architecture

The system architecture is illustrated in figure 5.1. Influenced by web and web services technologies, it consists of three layers: client layer, application layer and database layer. There are two possibilities to use this system: using an off-line client installed locally which periodically connects to the web service to update the locally-stored data, using a web browser to interactively access the web application.

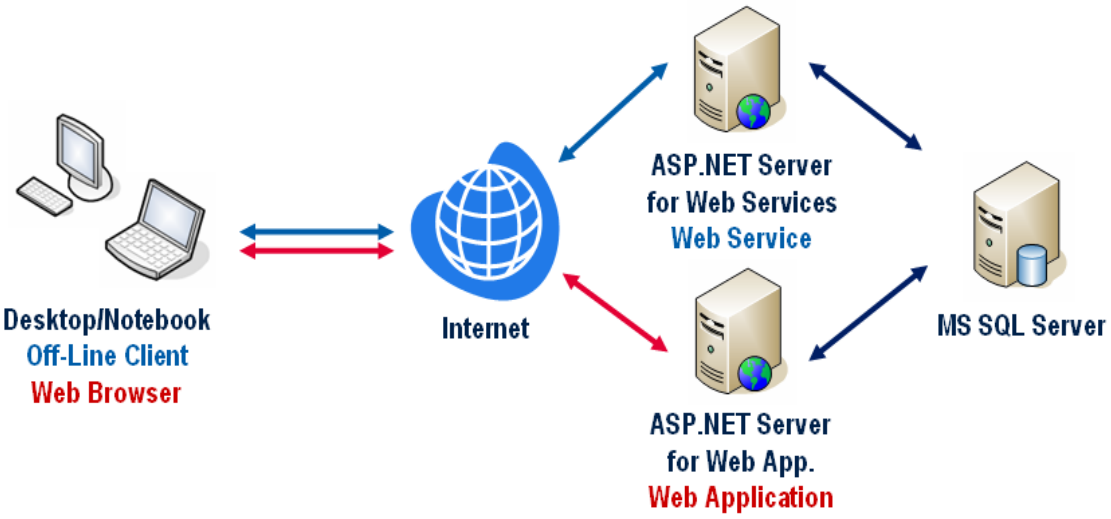


Figure 5.1: System architecture

Both approaches have their pros and cons. The off-line version is suitable for the usage without an Internet connection which is quite often in the real business scenario when salespersons visit a customer. On the other hand, the web-based version is easy to use since no local installation required and it supports various platforms of client machines. However, the web-based version has some limitations due to technology constraints. Consequently, some advanced features could not be implemented in the web-based version. Nevertheless, both approaches should be implemented in this project to fulfill different needs.

5.2 Software Architecture

The software architecture as shown in figure 5.2 consists of four major components: the core component, the off-line client component, the web application component and the web service component. The core component defines data structures, generic interfaces, generic classes and utility classes. The off-line component defines windows-based user interfaces. The web application component defines web-based user interfaces. Finally, the web service component defines web service operations.

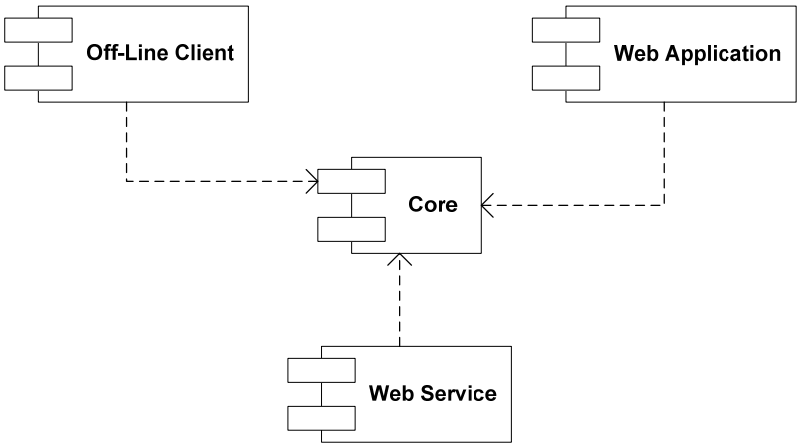


Figure 5.2: Major Components

5.3 Object Models

After analyzing products in SMPS applications e.g. CoolSET, it is found that some products can be used in different modes and topologies. However, some products can be used only in a specific mode in a specific topology. In addition, single output calculations slightly differ from multiple outputs calculations. As a result, the object model is constructed as the followings:

5.3.1 Topologies and modes

Topologies and modes are represented by interfaces. Since multiple inheritances are not allowed in .NET framework, supporting different modes and topologies to a single product could only be done via interfaces. Products are therefore represented by classes. Product classes could implement several interfaces as products could be used in different modes and topologies.

Basically, there are four levels of interfaces as illustrated in figure 5.3. The topmost level consists of only an interface namely "Topology". This interface is the base interface of all interfaces in this object model. The second level consists of interfaces representing topologies. The third level consists of sub interfaces of the second level interfaces categorized by outputs: single output or multiple outputs. This results from analyzing the characteristics of the calculations that single output shares a lot of common calculations with the multiple outputs. The lowermost level derives from the third level for different modes of operation.

Each interface defines a set of properties representing variables and equations in the calculation. For equations giving only results, read-only properties are defined. In addition, to avoid confusion and to make the calculation transparent, every recalculation of a variable generates a new variable.

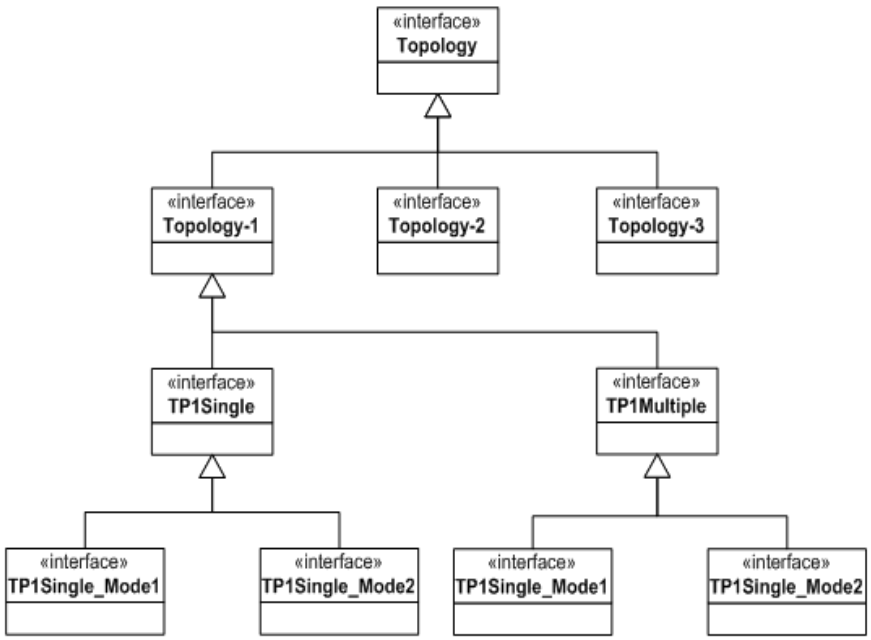


Figure 5.3: Topologies and modes object model

The number of sub interfaces varies from topology to topology. Some topology supports only a single output as well as only a single mode. With inheritance capabilities, sub interfaces could easily share common properties. This would dramatically reduce code redundancies in the implementation.

5.3.2 Products

As mentioned in the section 5.3.1, products are represented by classes. A single product may be used in different topologies and modes. Therefore, it may have to implement several interfaces. The product class should basically store the product-oriented characteristic values. Each characteristic value is represented by a property.

Normally, there are five levels of classes as depicted in figure 5.4. The top most level consists of only an abstract class namely "Product". This class is the root of all product classes in the object model. The second level consists of abstract classes representing product groups. The third level abstract classes derive from the second level classes with difference in the number of outputs: single output or multiple outputs. The fourth level consists of abstract derived classes of the third level classes representing product families. The fifth level classes derive from the fourth level classes with difference in the mode of operation.

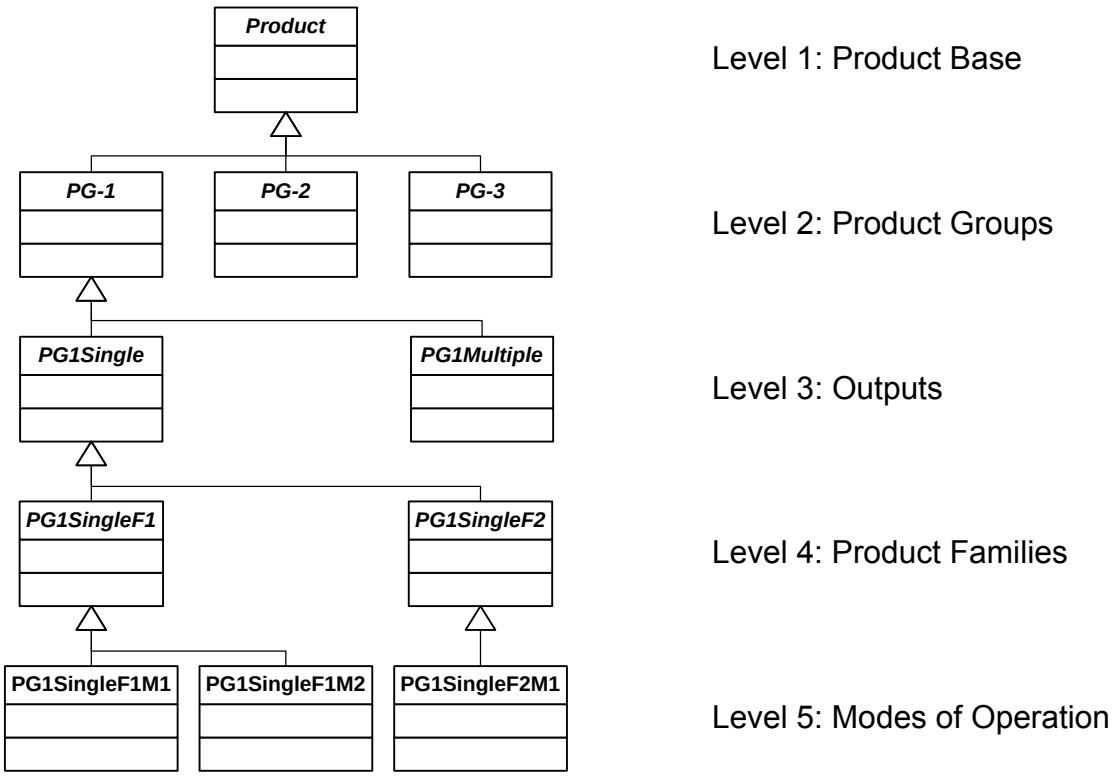


Figure 5.4: Products object model

Regarding interfaces from section 5.3.1, Second-level topology interfaces can be implemented at the second-level abstract classes. Third-level interfaces can be implemented at the third-level abstract classes matching number of outputs. Fourth-level interfaces can be implemented at the fifth-level classes.

5.3.3 Optimizers

Due to variation of optimization steps in different modes of operation, optimizers have to be varied from mode to mode as well. The object model of optimizers is defined as illustrated in figure 5.5. There are three levels of classes in the object model. The first level consists of only an abstract class namely "OptimizerBase". The second level consists of two abstract classes: OptimizerSingle for single-output solution and OptimizerMultiple for multiple-output solution. The third level classes represent modes of operation.

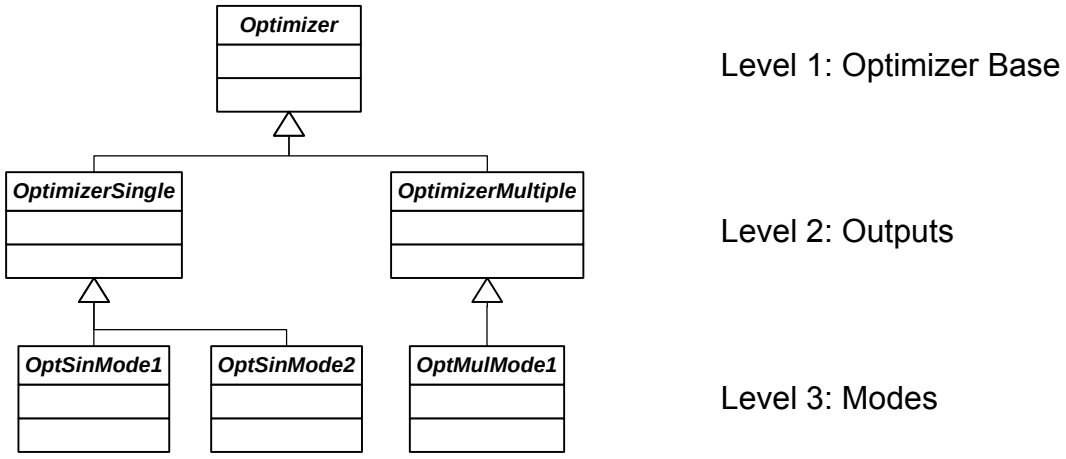


Figure 5.5: Optimizers object model

5.4 Attribute Class

Considering the large number of properties in calculations, if the description of each property has to be manually shown in user interfaces, there will be a lot of redundant work and may lead to inconsistency problem. In addition, the units of values should be consistent wherever the value is presented e.g. mA (milliampere) or A (ampere). Therefore, values should be kept internally at the base unit e.g. A (ampere), V (volt) to avoid difficulties in unit conversions during calculations. For the output e.g. shown in user interfaces, values have to be converted into a proper target unit by multiplying with a certain factor e.g. converting A to mA requires a factor of 10^3. The factor should be provided along with the target unit. The number format is another parameter that should be controlled in user interfaces e.g. two-digit decimal should be a default number format. Moreover, to give the users understanding where an equation comes from, the reference should be provided.

To provide information described in the previous paragraph, an attribute class is introduced. The attribute class namely "IPSPropertyAttribute" is defined and tagged to every property representing a variable or an equation. The simplified class declaration in VB.Net is shown in listing 5.1.

Listing 5.1: Class Declaration in VB.Net (simplified): IPSPropertyAttribute

```
<AttributeUsage(AttributeTargets.Property)> _
Public Class IPSPropertyAttribute
   Inherits Attribute

   Public ReadOnly Property Unit() As String
   Public ReadOnly Property UnitFactor() As Double
   Public ReadOnly Property Description() As String
   Public ReadOnly Property NumberFormat() As String
   Public ReadOnly Property Reference() As String
   Public ReadOnly Property IsEscalatableUnit() As Boolean

   Public Sub New(ByVal pUnit As String, _
          Optional ByVal pDescription As String = "", _
          Optional ByVal pUnitFactor As Double = 1, _
          Optional ByVal pNumberFormat As String = "0.00", _
          Optional ByVal pReference As String = "", _
          Optional ByVal pIsEscalatableUnit As Boolean = False)
End Class
```

An example of the attribute class instance is shown in listing 5.2 where an attribute class instance describes information of the "VACmin" property in the "FlybackTopology" interface. The first parameter is string-type specifying the unit. The second parameter gives a description of the property. The third parameter refers to the unit factor however in this example the default value of one is applied. The fourth parameter defines the number format. Here is "0" meaning that the value will be shown without any decimal point. The default value of this parameter is "0.00" meaning value will be shown with two decimal points. The fifth parameter which is missing in this example, gives a reference. An empty string is a default value for this parameter. The sixth parameter specifies whether the unit of this value can be automatically escalated or not e.g. 1000mΩ to be 1Ω.

Listing 5.2: An attribute class instance

```
<IPSProperty("V", "Min. AC Input Voltage (0 if DC source)", , "0")>
Property VACmin() As Double
```

The attribute class plays an important role in user interfaces since it provides descriptive information of each property shown in user interfaces. This makes the system dynamic and transparent. It also centralizes the descriptive information of properties to be in one place. If some information needs to be changed, it can be easily modified once and everywhere using this information will be affected. With this approach, it dramatically reduces redundant work required to manually provide descriptive information of all properties in user interfaces.

5.5 Data Structures

Several data structures have been introduced in order to support the system. Since the object-oriented concept is applied, classes are chosen to represent data structures. There are twelve classes defined as the following.

5.5.1 Limit Values

This data structure is dedicated for the electrical characteristic values of products which often consist of minimum, typical and maximum values. Values are stored in Double-type variables. Sometimes only two of three values exist therefore each value should be tagged whether the value is present or not. The problem comes from the fact that null value cannot be applied to basic data type e.g. Double. Consequently, an additional Boolean-type variable is introduced for each value. In addition, to ease the duplication of values, the "CopyFrom" sub-routine is defined. The simplified class declaration in VB.Net is shown in listing 5.3.

Listing 5.3: Class Declaration in VB.Net (simplified): LimitValues

```
Public Class LimitValues

   Public Property IsMinValueDefined() As Boolean
   Public Property MinValue() As Double
   Public Property IsTypicalValueDefined() As Boolean
   Public Property TypicalValue() As Double
   Public Property IsMaxValueDefined() As Boolean
   Public Property MaxValue() As Double

   Public Sub CopyFrom(ByVal pLV As LimitValues)
End Class
```

5.5.2 IPS Structure

Due to the vitality of the sequence of equations and difficulties in ordering properties using .NET Reflection, a list of items for user interfaces has to be constructed. The IPS structure represents each item or row to be shown in user interfaces. To avoid redundancy and inconsistency, the information of the property which already described by the attribute class as well as the value of the property are not kept within this structure. The IPS structure can easily link to the target property via the "PropertyInfo" class of .NET Reflection. Additional information required for user interfaces are provided in this structure. The simplified class declaration in VB.Net is shown in listing 5.4.

An integer-type "Sequence" property contains the sequence number of the item. The "GroupName" property is used for grouping items. The "IsResult" property will be set to true if the item represents a result. To refer back to the owner object of the item which is required in .NET Reflection, the "OwnerObject"

property is provided. Obviously, the "PropInfo" property is the link to the "PropertyInfo" class instance of the target property. The "IPSProp" property is provided here giving a direct link to the target attribute class in order to reduce complexity due to the hierarchical structure of product classes. If the item represents limit values, the enumeration must be given to know which sub-value is referred to the item. The "OutputIndex" property is only used in multiple outputs solution.

Listing 5.4: Class Declaration in VB.Net (simplified): IPSStruct

```
Public Class IPSStruct

   Public Property Sequence() As Integer
   Public Property GroupName() As String
   Public Property IsResult() As Boolean
   Public Property OwnerObject() As Object
   Public Property PropInfo() As PropertyInfo
   Public Property IPSProp() As IPSPropertyAttribute
   Public Property LimitValuesEnum() As LimitValuesEnums
   Public Property OutputIndex() As Integer

   Public Sub New(ByVal pSequence As Integer, _
                  ByVal pGroupName As String, _
                  ByVal pIsResult As Boolean, _
                  ByVal pOwnerObject As Object, _
                  ByVal pPropInfo As PropertyInfo, _
                  ByVal pIPSProp As IPSPropertyAttribute, _
                  Optional ByVal pLV As LimitValuesEnums = _
                                        LimitValuesEnums.notDefined, _
                  Optional ByVal pOutputIndex As Integer = 0)
   Public Sub New(ByVal pSequence As Integer, _
                  ByVal pGroupName As String, _
                  ByVal pIsResult As Boolean, _
                  ByVal pOwnerObject As Object, _
                  ByVal pPropInfo As PropertyInfo, _
                  Optional ByVal pLV As LimitValuesEnums = _
                                        LimitValuesEnums.notDefined, _
                  Optional ByVal pOutputIndex As Integer = 0)
End Class
```

5.5.3 IPS Warning

When rules for validation are violated, warnings will be raised. The "IPSWarning" class serves this purpose. It consists of 5 properties. String-type Code, Description and Resolution properties are self understandable. The Boolean-type "IsCritical" property refers that the warning is critical or not. If so, it cannot be neglected. The "TargetPropertyName" property keeps the name of the property in which the problem can be solved. The simplified class declaration in VB.Net is shown in listing 5.5.

Listing 5.5: Class Declaration in VB.Net (simplified): IPSWarning

```
Public Class IPSWarning

    Public Property WarningCode() As String
    Public Property WarningDescription() As String
    Public Property WarningResolution() As String
    Public Property IsCritical() As Boolean
    Public Property TargetPropertyName() As String

    Public Sub New(ByVal pWarningCode As String, _
                   ByVal pWarningDescription As String, _
                   ByVal pWarningResolution As String, _
                   Optional ByVal pIsCritical As Boolean = False, _
                   Optional ByVal pTargetPropertyName As String = _
                        Nothing)
End Class
```

5.5.4 Rubycon Capacitor Structure

The special capacitor used for some point in the circuit is determined by not only the capacitance value but also some other electrical characteristics e.g. ripple current. Therefore, a special structure has to be provided. The simplified class declaration in VB.Net is shown in listing 5.6.

Listing 5.6: Class Declaration in VB.Net (simplified): RubyConCapStruct

```
Public Class RubyConCapStruct
        <IPSProperty("V", "Rated Voltage")> _
        Public Property RatedVoltage() As Double
        <IPSProperty("µF", "Capacitance Value", (10 ^ 6), "0")> _
        Public Property Capacitance() As Double
        <IPSProperty("", "Size (Diameter x Length)")> _
        Public Property DxL() As String
        <IPSProperty("A", "Max. Permissible Ripple Current")> _
        Public Property MaxIRipple() As Double
        <IPSProperty("Ohm", "ESR @ 20°C / 100kHz", , "0.000")> _
        Public Property ESRAt20C() As Double

        Public Sub New(ByVal pRatedVoltage As Double, _
                       ByVal pCapacitance As Double, _
                       ByVal pDxL As String, _
                       ByVal pMaxIRipple As Double,
                       ByVal pESRAt20C As Double)
End Class
```

5.5.5 Multiple Outputs Structure

Since some equations depend on each output, the multiple outputs structure is designed for handling variables and equations which are dependent on an output. The simplified class declaration in VB.Net of the multiple outputs structure for Flyback topology is shown in listing 5.7. However, the topology-related properties and sub-routines are not included in the listing.

Listing 5.7: Class Declaration in VB.Net (simplified): FlyBackMultiOutputStruct

```
Public Class FlyBackMultiOutputStruct

   Public ReadOnly Property OwnerTopology() As FlybackMultiple
   Public ReadOnly Property OwnerCollection() As _
                                        FlyBackMultiOutputCollection
   Public ReadOnly Property Number() As Integer
   Public ReadOnly Property FormattedNumber() As String

   Public Sub New(ByVal pOwnerTopology As FlybackMultiple, _
                  ByVal pOwnerCoolSET As CoolSET, _
                  ByVal pOwnerCollection As _
                                      FlyBackMultiOutputCollection, _
                  ByVal pNumber As Integer)

End Class
```

The "OwnerTopology" property links to the object instance of class that implements the preferred interface. In addition, the "OwnerCollection" property links to the collection which this item is stored. The "Number" property represents a sequence number of the item while the "FormattedNumber" property returns a two-digit formatted string of the "Number" property.

5.5.6 Output Structure

To support development of user interfaces where output-dependent parameters are gathered, the "Output" structure is introduced. This structure helps handling output-dependant parameters more efficiently. The simplified class declaration in VB.Net is shown in listing 5.8.

Listing 5.8: Class Declaration in VB.Net (simplified): OutputStruct

```
Public Class OutputStruct

   Public Property VOut() As Double
   Public Property IOut() As Double
   Public ReadOnly Property POutnom() As Double

   Public Sub New(ByVal pVOut As Double, ByVal pIOut As Double)
End Class
```

5.5.7 Default Value Structure

Default values play an important role in the project since one of the goals is to minimize the number of input parameters that non-expert users have to choose but not getting rid of possibility to change advanced parameters for expert users. A list of default values should be generated from the product class. The list is then shown in the user interface for approval. The approved list is later sent to an optimizer to realize in the optimization. The simplified class declaration in VB.Net is shown in listing 5.9.

Listing 5.9: Class Declaration in VB.Net (simplified): DefaultValueStruct

```
Public Class DefaultValueStruct

   Public Property Name() As String
   Public Property IPSProp() As IPSPropertyAttribute
   Public Property Value() As Double
   Public Property OutputIndex() As Integer

   Public Sub New(ByVal pPropInfo As PropertyInfo, _
                  ByVal pValue As Double, _
                  Optional ByVal pOutputIndex As Integer = 0)
End Class
```

5.5.8 Advanced Inputs for Optimizer

In addition to a list of default values, there are other advanced input parameters for an optimizer. Set of transformer cores, heat dissipaters and external MOSFETs can also be tailored. The simplified class declaration in VB.Net is shown in listing 5.10.

Listing 5.10: Class Declaration in VB.Net (simplified): OptimizerAdvInputs

```
Public Class OptimizerAdvInputs

   Public Property HDs() As Collection
   Public Property MOSFETs() As Collection
   Public Property TFCores() As Collection
   Public Property MaxAllowedTj() As Double
   Public Property DefaultValues() As DefaultValueStructSL
   Public Property NeglectNonCriticalWarnings() As Boolean

   Public Sub New(ByVal pHDs As Collection, _
                  ByVal pMOSFETs As Collection, _
                  ByVal pTFCores As Collection, _
                  ByVal pMaxAllowedTj As Double, _
                  ByVal pDefaultvalues As DefaultValueStructSL, _
                  Optional ByVal pNeglectNonCriticalWarnings As _
                                                    Boolean = False)
End Class
```

5.6 Products

The Product class is an abstract base class which all product-related classes have to inherit. It defines generic functions and declares functions that must be overridden. The simplified class declaration in VB.Net is shown in listing 5.11.

Listing 5.11: Class Declaration in VB.Net (simplified): Product

```
Public MustInherit Class Product

   Public Property Name() As String
   Public Property SortableName() As String
   Public Property Series() As String
   Public Property CanUseExtMOSFET() As Boolean
   Public Property Transformer() As TransformerCore
   Public ReadOnly Property IPSWarnings() As IPSWarningsSL
   Public Property AutoValueRecommendation() As Boolean

   Public Property CanUseHeatSink() As Boolean
   Public Property UseHeatSink() As Boolean
   Public Property UseCuArea() As Boolean

   Public Property RthCuArea() As Double
   Public Property RthHT() As Double
   Public Property RthHS() As Double
   Public Property ExtMOSFET() As MOSFET

   Public MustOverride Function getIPSStructSL() As IPSStructSL
   Public MustOverride Function getDefaultValueStructSL( _
      Optional ByVal existingDVSL As DefaultValueStructSL = Nothing, _
      Optional ByVal numberOfOutputs As Integer = 1) As _
      DefaultValueStructSL
   Public MustOverride Function getBOM() As BOMTable
   Public MustOverride Sub enforceAutoValueRecommendation( _
      Optional ByVal pKeepAVRSetting As Boolean = True)
   Protected Function getPropInfo(ByVal pName As String, _
      ByVal pPropInfos As PropertyInfo()) As PropertyInfo
   Public MustOverride Sub ExportToXML(ByVal xmlParEmt As XmlElement,
      ByVal pTopology As Type)
   Public MustOverride Sub ImportXML(ByVal xmlParEmt As XmlElement)
End Class
```

5.6.1 IPS Structure Sorted List

The IPS structure sorted list is generated for user interfaces to easily access to properties needed to be shown for a specific product. The “getIPSStructSL” method is responsible for generating the list. The list contains IPS structures, already described in section 5.5.2, sorted in a proper sequence. The structure acts as a proxy to the product object. Each IPS structure represents either a variable or an equation in the calculation which is a property in the product object. The property value will be accessed via the structure either for value request or value

manipulation. Furthermore, additional user-interface-oriented information is provided in the structure. Figure 5.6 depicts the process when a user interface populates a property list of a product class. The user interface firstly requests for an IPS structure sorted list (IPSStructSL) to the product object. The product object generates a list and returns it to the user interface. Assume that a structure X is bounded to a property X and the user interface is populating an item X based on the structure X. The user interface gathers user-interface-oriented information directly from the structure. However, for the IPS attribute and the value of the property X must be gathered from the property itself via the structure.

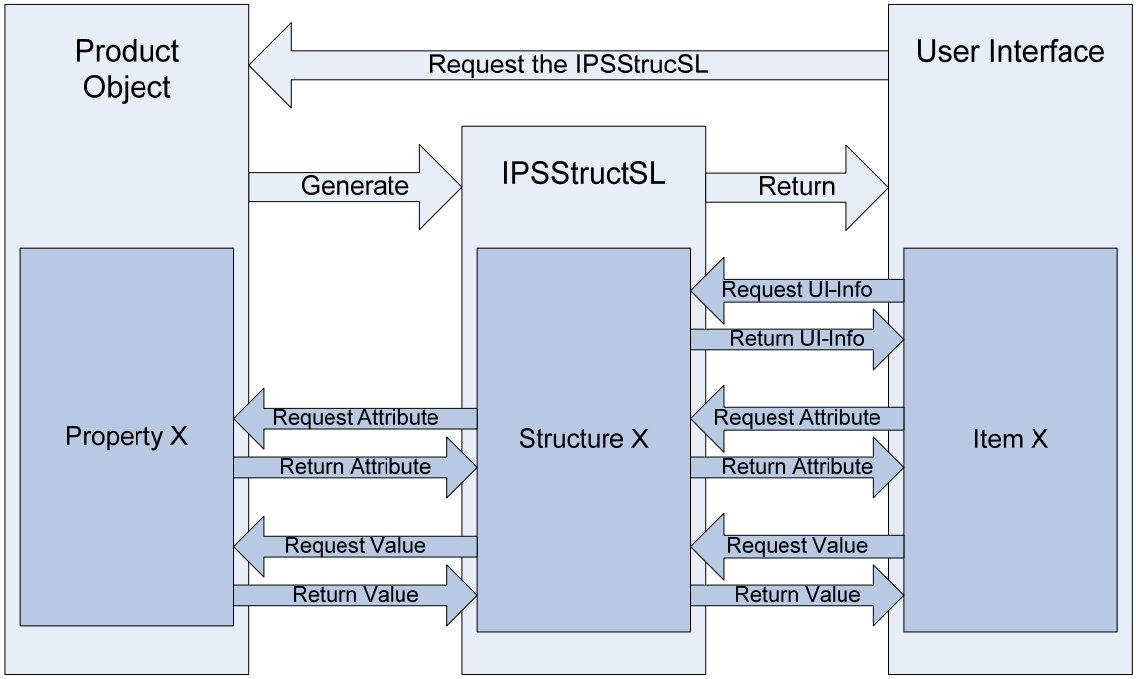


Figure 5.6: Populating property list of Product

5.6.2 Default Value Structure Sorted List

As explained in section 5.5.7, the default value structure plays an important role in balancing between the simplicity of input parameters for non-expert users and the flexibility of advanced input parameters manipulation for expert users. The default value structure sorted list (DefaultValueStructSL) is generated by invoking the product object's "getDefaultValueStructSL" method. The list contains all necessary default values required for an optimizer. Figure 5.7 shows the process in which the structure is generated. The structure may be manipulated by expert users. Finally, it is sent to the optimizer along with other input parameters. In addition, it might happen that several products capable of operating in the same mode are considered in an optimizer. For this case, the union set of default values of all considered products must be provided therefore the method must be able to support adding non-existing default values to the existing list.

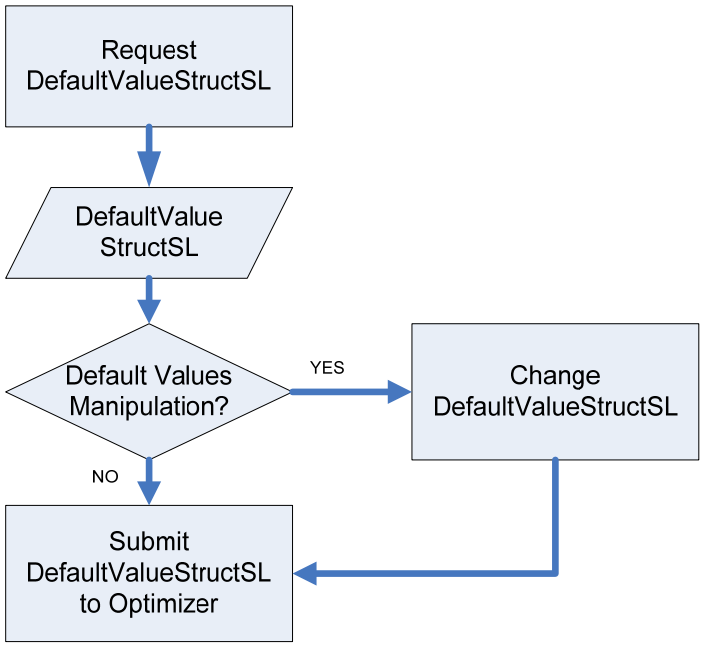


Figure 5.7: Default Value Structure Sorted List

5.6.3 Bill of Material

The bill of material (BOM) is a list of electronic components used in an SMPS solution. It describes the type of components, value, rating as well as the number of each component required. Users can easily take the bill of material to build up a real circuit board. The BOM can be generated by invoking the "getBOM" method of the product class.

5.6.4 Automatic Value Recommendation

The automatic value recommendation is a mechanism to automatically find appropriate values for some electronic components e.g. finding an available resistor with the resistance close to the preferred value. This can be done by either table look up or iterating through all possibilities. An equation generating automatic value recommendation is represented by a read-only property. When it gets a request to return a result from an equation and the automatic value recommendation flag is set, the target variable will be changed by defined rules. Figure 5.8 shows the flowchart of the automatic value recommendation mechanism.

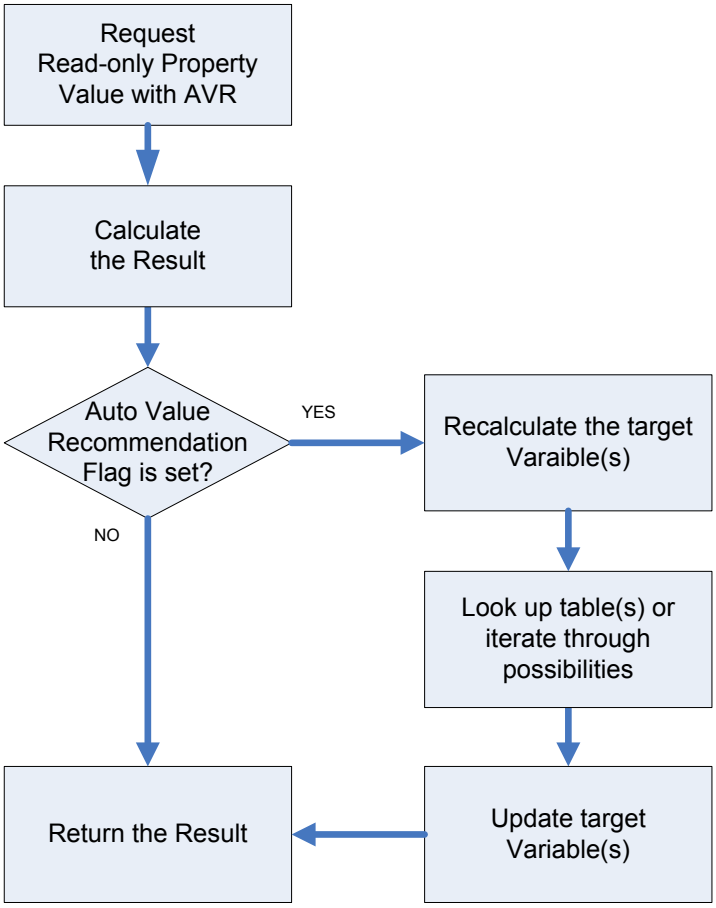


Figure 5.8: Automatic Value Recommendation Mechanism

Considering a sequence of equations, it must make sure that intermediate steps of automatic value recommendation are invoked properly in a correct sequence in order to get correct final results as shown in figure 5.9. This can simply be done via the "enforceAutoValueRecommendation" method. This method ensures that all intermediate steps are invoked in a proper sequence. Moreover, if one variable is changed, all other dependant variables and equations will be affected. Since there is no automatic mechanism to notify user interfaces, it must be assured that user interfaces have refreshed all affected values.

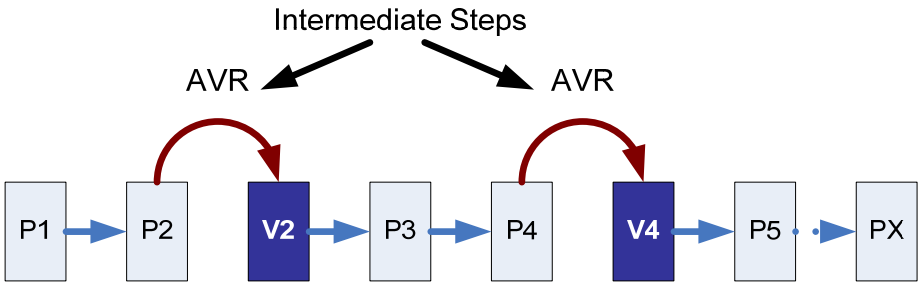


Figure 5.9: Intermediate Steps

5.6.5 XML Import and Export

The product class is also considered as a solution class in this project. Therefore it should be able to be exported to and imported from an XML-based file for storing the solution. The generic serialization of .NET Framework could not apply so the customized serialization has to be done. The "ExportToXML" method is used for serialization. On the other hand, the "ImportXML" is used to recover the solution from an XML-based file. To avoid the mismatched structure problems, the XML schema validation, which is one of infrastructure provided by .NET framework, is used as shown in figure 5.10. As a result, the serialization and deserialization processes are controlled by the XML schema validation using the XML schema defined in Appendix A.

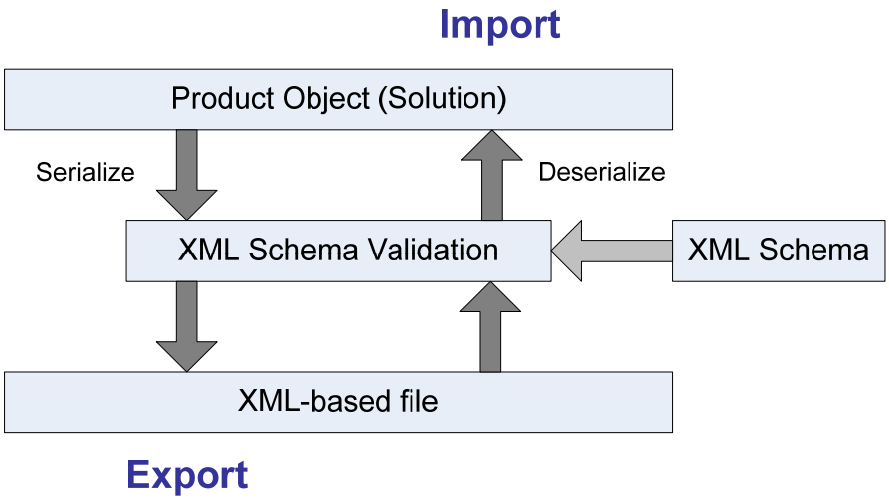


Figure 5.10: XML Import Export

5.7 Optimizers

The "OptimizerBase" class is the base class of all optimizer classes implemented in this project. It defines common properties as well as common methods. The simplified class declaration in VB.Net is shown in listing 5.12.

Listing 5.12: Class Declaration in VB.Net (simplified): OptimizerBase

```
Public MustInherit Class OptimizerBase

   Public Property Conn() As SqlConnection
   Public Property CSMGR() As CoolSETManager
   Public Property TCMGR() As TransformerCoreManager
   Public Property MFMGR() As MOSFETManager
   Public Property HDMGR() As HeatDissipaterManager
   Public Property ProgressBar() As System.Windows.Forms.ProgressBar
   Public Property OptAdvInputs() As OptimizerAdvInputs
```

Listing 5.12: Class Declaration in VB.Net (simplified): OptimizerBase (Cont.)

```
   Public Property VACmin() As Integer
   Public Property VACmax() As Integer
   Public Property VDCmin() As Integer
   Public Property VDCmax() As Integer
   Public Property Ta() As Integer
   Public Property EffPre() As Double

   Protected Sub New(ByVal pCSmgr As CoolSETManager, _
                     ByVal pTCmgr As TransformerCoreManager, _
                     ByVal pMFmgr As MOSFETManager, _
                     ByVal pHDmgr As HeatDissipaterManager, _
     Optional ByVal pOptAdvInputs As OptimizerAdvInputs = Nothing, _
     Optional ByVal pConn As SqlConnection = Nothing, _
     Optional ByVal pgb As System.Windows.Forms.ProgressBar = Nothing)

   Public Function getTargetProducts(ByRef rTargetProductSL As _
                                     SortedList) As TargetProductTable
   Protected Function getQueryID(ByRef pIsNewQuery As Boolean) As Long
   Protected Overridable Sub fillSQLCmdParameters(ByVal CmdParams As _
                                                  SqlParameterCollection)
   Protected Function VerifySolution(ByVal cs As CoolSET, Optional _
                            ByVal pHDname As String = Nothing) As Boolean
   Protected MustOverride Function getNewCoolSET(ByVal series As _
                                                 String) As CoolSET
   Protected MustOverride Function getTargetCoolSETs() As SortedList
   Protected MustOverride Function getTopology() As Type
   Protected MustOverride Function getMode() As String
   Protected Function getDefaultOptAdvInputs() As OptimizerAdvInputs
   Protected MustOverride Function getDefaultValueSL() As _
                                                   DefaultValueStructSL
   Protected Overridable Sub applyDefaultValues(ByVal cs As CoolSET)
End Class
```

Several properties must be supplied in order to instantiate an optimizer object. They are object instances of "CoolSETManager", "TransformerCore-Manager," "MOSFETManager" and "HeatDissipaterManager". The "OptimizerAdv-Inputs" object is optional only if default values have been manipulated or the set of considered components has been changed. For the online optimization, it is necessary to supply a database connection in order to gather and update cached queries and cached solutions. In addition, the Optimizer can support showing the active progress of the process on the "Progressbar" object during optimization process for Windows-based client.

The "getTargetProducts" method returns a sorted list of possible solutions through the pass-by-reference variable "rTargetProductSL" and the returned "TargetProductTable" object. The table is mainly used for displaying solutions summary in a DataGrid control of user interfaces. On the other hand, the sorted list contains all solution object instances for getting calculation detail. The flowchart of the method is depicted in figure 5.11.

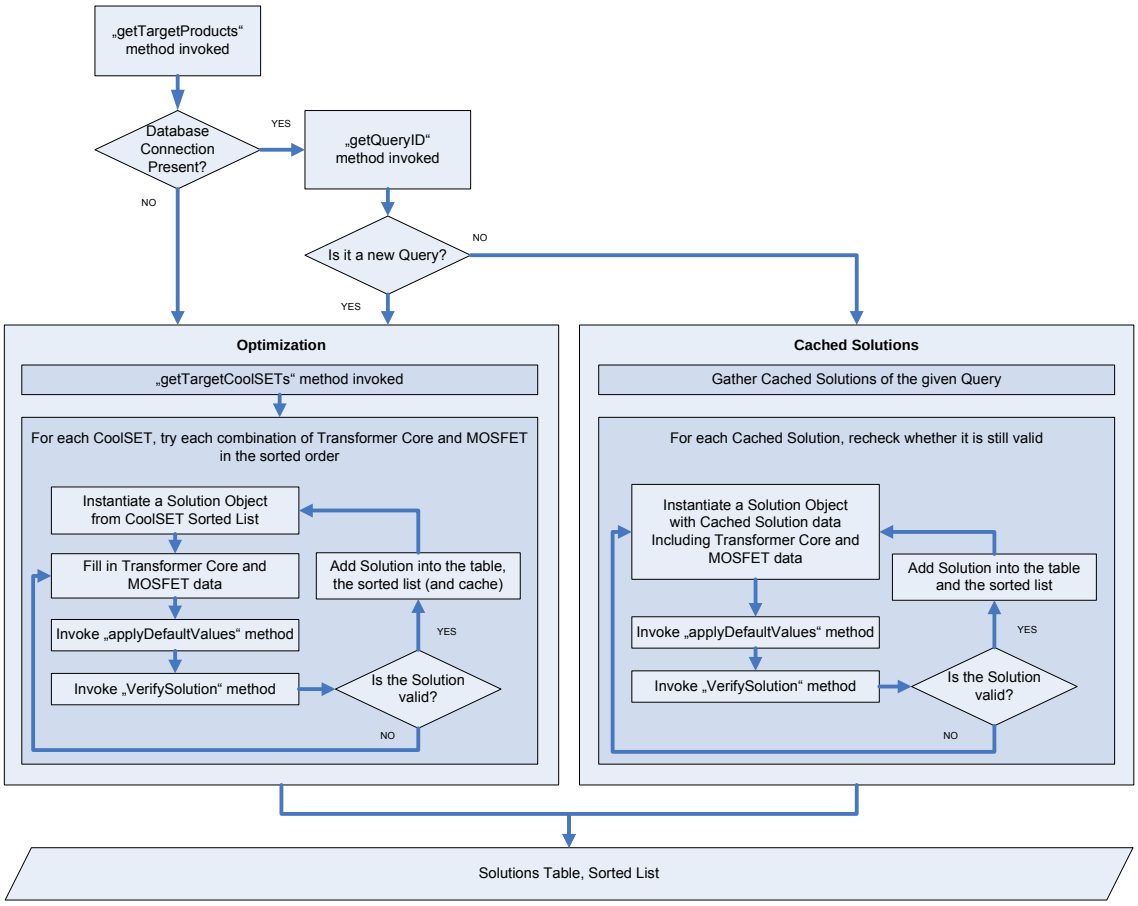


Figure 5.11: Optimizer “getTargetProducts” method flowchart

For the online optimization, the method first checks whether the query already exists in the cache or not by invoking the protected “getQueryID” method. If the given query does not exist in the cache, it will be inserted getting a new query identifier as a result. In contrast, the existing query identifier is returned.

If it is a new query, the optimization has to be done. Firstly, a sorted list of CoolSETs is gathered by invoking the “getTargetCoolSETs” method. A Solution object is instantiated from each CoolSET object within the sorted list. It is to be tested with each combination of considered transformer cores and MOSFETs. However, with the knowledge of the characteristics of the SMPS application domain, it does not make sense to use oversize components unless it is necessary. Therefore, if the first transformer core and MOSFET which make the solution valid are found, the iteration stops for that CoolSET. The protected “applyDefault-Values” method is called to fill out default values to the solution object. The solution object is then verified via the “VerifySolution” method. Unless the solution has an error, the solution is added to the solutions sorted list, solutions table and, in case of online optimization, the solution is also added into the cache. On other hand, if cached solutions are used, they are loaded into the solutions list and solution objects are reconstructed.

Chapter 6

Database Design

Database is an efficient data storage sharing among multiple users. In this project, the database is used to store all information on the server side e.g. product information. However, for client computers which the database connection is not available, the off-line partial copy of the database in the form of DataSet is used. The off-line partial copy of the database is generated via Web Service which queries data from the online database.

In this project, MS SQL Server 2000 was chosen to be a DBMS[8]. With its stored procedure capabilities, security measures can be easily realized. Direct table access should be avoided therefore most of operations have to be done via stored procedures. User roles can be introduced to categorize users into groups which have different access rights. Typically, from external access the database should be read only i.e. no data manipulating operation permitted. However, for caching known requests and solutions, there should have a special role named "CachedQueryManipulator" which can only manipulate cache tables.

There are ten tables and eleven stored procedures introduced in this project. Schemas of those tables as well as the declaration of those stored procedures are shown in the following sections. Note that, table schemas refer to MS SQL Server 2000 data types.

6.1 Table Schemas

Table 6.1: CachedQuery table schema

PK [9]	Field Name	Data Type	Size	Description
●	QueryID	bigint	-	Query Identifier (auto increment)
	Mode	varchar	10	Topology/Mode of operation
	InputType	varchar	2	Input type (AC/DC)
	VINmin	float	-	Minimum input voltage
	VINmax	float	-	Maximum input voltage
	Ta	float	-	Ambient temperature
	f	float	-	Switching Frequency
	fOSC2	float	-	Second Oscillation Frequency
	OutputPower	float	-	Output Power in Watt

[8] Database Management System

[9] Primary Key

	VOut01	float	-	Output Voltage 1
	IOut01	float	-	Output Current 1
	VOut02	float	-	Output Voltage 2
	IOut02	float	-	Output Current 2
	VOut03	float	-	Output Voltage 3
	IOut03	float	-	Output Current 3
	VOut04	float	-	Output Voltage 4
	IOut04	float	-	Output Current 4
	VOut05	float	-	Output Voltage 5
	IOut05	float	-	Output Current 5
	VOut06	float	-	Output Voltage 6
	IOut06	float	-	Output Current 6
	VOut07	float	-	Output Voltage 7
	IOut07	float	-	Output Current 7
	VOut08	float	-	Output Voltage 8
	IOut08	float	-	Output Current 8
	VOut09	float	-	Output Voltage 9
	IOut09	float	-	Output Current 9
	VOut10	float	-	Output Voltage 10
	IOut10	float	-	Output Current 10
	QueryDT	datetime	-	Data and Time Query Created

Table 6.2: CachedSolution table schema

PK	Field Name	Data Type	Size	Description
●	QueryID	bigint	-	Query Identifier **(FK[10] CachedQuery)**
●	Seq	int	-	Sequence of Solutions in a Query
	SortableCoolSET	varchar	20	Sortable CoolSET name
	CoolSET	varchar	20	CoolSET name
	SortableMOSFET	varchar	20	Sortable CoolMOS name
	CoolMOS	varchar	20	CoolMOS name
	TFCore	varchar	20	Transformer Core name
	HeatDissipater	varchar	20	Heat Dissipater name

Table 6.3: CoolSETs table schema

PK	Field Name	Data Type	Size	Description
●	Name	varchar	20	CoolSET name
	SortableName	varchar	20	Sortable CoolSET name
	Series	varchar	20	Product Series name **(FK ProductSeries)**

[10] Foreign Key

	CanUseExtMOSFET	bit	-	Flag specifying whether this CoolSET can use an external MOSFET or not
	f	float	-	Switching Frequency
	Packaging	varchar	20	Packaging of the CoolSET **(FK Packagings)**
	Vcsth_min	float	-	Minimum Peak Current Limitation
	Vcsth_typ	float	-	Typical Peak Current Limitation
	Vcsth_max	float	-	Maximum Peak Current Limitation
	Vbr_dss	float	-	Drain Source Breakdown Voltage
	IvccST_typ	float	-	Typical Start-up Current
	IvccST_max	float	-	Maximum Start-up Current
	IvccIAG_typ	float	-	Typical Inactive Gate Current
	IvccIAG_max	float	-	Maximum Inactive Gate Current
	IvccAG_typ	float	-	Typical Active Gate Current
	IvccAG_max	float	-	Maximum Active Gate Current
	VCCon_min	float	-	Minimum VCC Turn-On Threshold
	VCCon_typ	float	-	Typical VCC Turn-On Threshold
	VCCon_max	float	-	Maximum VCC Turn-On Threshold
	VCCoff_typ	float	-	Typical VCC Turn-Off Threshold
	VCCHY_min	float	-	Minimum Turn-On/Off Hysteresis
	VCCHY_typ	float	-	Typical Turn-On/Off Hysteresis
	VCCHY_max	float	-	Maximum Turn-On/Off Hysteresis
	VREF_min	float	-	Minimum Trimmed Reference Voltage
	VREF_typ	float	-	Typical Trimmed Reference Voltage
	VREF_max	float	-	Maximum Trimmed Reference Voltage
	Av_min	float	-	Minimum PWM-OP Gain
	Av_typ	float	-	Typical PWM-OP Gain
	Av_max	float	-	Maximum PWM-OP Gain
	VFB_min	float	-	Minimum Operating Feedback Voltage
	VFB_max	float	-	Maximum Operating Feedback Voltage
	RFB_min	float	-	Minimum Feedback Resistance
	RFB_typ	float	-	Typical Feedback Resistance
	RFB_max	float	-	Maximum Feedback Resistance
	Rsoftstart_min	float	-	Minimum Soft-Start Resistance
	Rsoftstart_typ	float	-	Typical Soft-Start Resistance
	Rsoftstart_max	float	-	Maximum Soft-Start Resistance
	Vsoftstart1_min	float	-	Minimum Activation Limit of Overload & Open Loop Detection
	Vsoftstart1_typ	float	-	Typical Activation Limit of Overload & Open Loop Detection
	Vsoftstart1_max	float	-	Maximum Activation Limit of Overload & Open Loop Detection
	fOSC	float	-	Oscillation Frequency
	tSOFTS	float	-	Soft-Start time
	VOPOVP	float	-	Typical Output Over-Voltage Protection
	VSOC	float	-	Typical Soft Over Current Control
	GmOTA1	float	-	Typical OTA1 Transconductance Gain

	GmOTA2	float	-	Typical OTA2 Transconductance Gain
	RDSON25C_typ	float	-	Typical On-Resistance at 25°C
	RDSON25C_max	float	-	Maximum On-Resistance at 25°C
	Coer	float	-	Eff. Output Capacitance (Energy-Related)
	RthJA	float	-	Thermal Resistance Junction-Ambient
	RthJC	float	-	Thermal Resistance Junction-Case

Table 6.4: ElecCompValues table schema

PK	Field Name	Data Type	Size	Description
●	ELevel	varchar	5	Level of E-Table
●	EValue	float	-	Possible Electronic Component Value

Table 6.5: HeatDissipaters table schema

PK	Field Name	Data Type	Size	Description
●	Name	varchar	20	Heat Dissipater name
	RthHS	float	-	Thermal Resistance of Heat Sink
	RthCuArea	float	-	Thermal Resistance of Copper Area

Table 6.6: MOSFETs table schema

PK	Field Name	Data Type	Size	Description
●	Name	varchar	20	MOSFET name
	SortableName	varchar	20	Sortable MOSFET name
	Voltage	float	-	Voltage Rating
	IDAt25C	float	-	Continuous Drain Current at 25°C
	RDSonAt25C	float	-	On-Resistance at 25°C
	Coer	float	-	Eff. Output Capacitance (Energy-Related)
	RthJA	float	-	Thermal Resistance Junction-Ambient
	RthJC	float	-	Thermal Resistance Junction-Case

Table 6.7: Packagings table schema

PK	Field Name	Data Type	Size	Description
●	Packaging	varchar	20	Packaging name
	CanUseHeatSink	bit	-	Flag specifying whether this packaging can use a heat sink or not

Table 6.8: ProductSeries table schema

PK	Field Name	Data Type	Size	Description
●	Series	varchar	20	Product Series name

Table 6.9: RubyConCaps table schema

PK	Field Name	Data Type	Size	Description
●	RatedVoltage	float	-	Rated Voltage
●	Capacitance	float	-	Capacitance Value
●	DxL	varchar	10	Diameter and Length
	MaxIRipple	float	-	Maximum Ripple Current
	ESRAt20C	float	-	Equivalent Series Resistance at 20°C

Table 6.10: TransformerCores table schema

PK	Field Name	Data Type	Size	Description
●	Name	varchar	20	Transformer Core name
	Type	varchar	20	Transformer Core Type
	Material	varchar	20	Core Material
	Pmax	float	-	Maximum Power
	BW	float	-	Bobbin Width
	AN	float	-	Winding Cross Section
	Ae	float	-	Effective Magnetic Cross Section
	Bmax	float	-	Maximum Flux Density
	le	float	-	Effective Magnetic Path Length
	lN	float	-	Average Length per Turn
	PV	float	-	Relative Core Losses
	K1	float	-	Core-Specific Constant 1
	K2	float	-	Core-Specific Constant 2
	Ve	float	-	Effective Magnetic Volume

6.2 Stored Procedures Declaration

PROCEDURE: dbo.ips_cachedquery_insert

PARAMETERS:		
	@Mode varchar(10),	Topology/Mode
	@InputType varchar(2),	AC/DC Input Voltage
	@VINmin float,	Minimum Input Voltage
	@VINmax float,	Maximum Input Voltage
	@Ta float,	Ambient Temperature
	@f float,	Switching Frequency
	@fOSC2 float,	2nd Oscillation Freq.
	@OutputPower float,	Output Power
	@VOut01 float,	Output Voltage 1
	@IOut01 float,	Output Current 1
	@VOut02 float,	Output Voltage 2
	@IOut02 float,	Output Current 2
	@VOut03 float,	Output Voltage 3
	@IOut03 float,	Output Current 3
	@VOut04 float,	Output Voltage 4
	@IOut04 float,	Output Current 4
	@VOut05 float,	Output Voltage 5
	@IOut05 float,	Output Current 5
	@VOut06 float,	Output Voltage 6
	@IOut06 float,	Output Current 6
	@VOut07 float,	Output Voltage 7
	@IOut07 float,	Output Current 7
	@VOut08 float,	Output Voltage 8
	@IOut08 float,	Output Current 8
	@VOut09 float,	Output Voltage 9
	@IOut09 float,	Output Current 9
	@VOut10 float,	Output Voltage 10
	@IOut10 float,	Output Current 10
	@QueryID bigint ***output***,	Query Identifier
	@IsNewQuery bit ***output***	New Query Flag.

PURPOSE: to insert a new query into the cache getting the newly generated query identifier and New Query flag as outputs. If the given query already exists in the cache, the query identifier is returned.

PROCEDURE: dbo.ips_cachedsolution_insert

PARAMETERS:		
	@QueryID bigint,	Query Identifier
	@Seq int,	Sequence of Solutions
	@SortableCoolSET varchar(20),	Sortable CoolSET name
	@CoolSET varchar(20),	CoolSET name
	@SortableMOSFET varchar(20),	Sortable MOSFET name
	@MOSFET varchar(20),	MOSFET name

	`@TFCore varchar(20),`	Transformer Core name
	`@HeatDissipater varchar(20)`	Heat Dissipater name
PURPOSE:	to insert a new solution of the query into the cache.	

PROCEDURE: **dbo.ips_cachedsolution_list_by_id**
PARAMETER: `@QueryID bigint` Query Identifier
PURPOSE: to list all cached solutions of the given query identifier.

PROCEDURE: **dbo.ips_coolsetqrs_list**
PURPOSE: to list all Quasi-Resonant CoolSETs (Series name = "PWM-QSR").

PROCEDURE: **dbo.ips_coolsets_list**
PARAMETER: `@Series varchar(20) = NULL` Series name
PURPOSE: to list all CoolSETs with the given series name. If the series name is not given, all CoolSETs with the series name beginning with "CoolSET" will be returned.

PROCEDURE: **dbo.ips_eleccompvalues_elevel_list**
PURPOSE: to list all distinct E-levels from the ElecCompValues table.

PROCEDURE: **dbo.ips_eleccompvalues_select_all**
PURPOSE: to list all rows from the ElecCompValues table.

PROCEDURE: **dbo.ips_heatdissipaters_list**
PURPOSE: to list all rows from the HeatDissipaters table.

PROCEDURE: **dbo.ips_mosfets_list**
PURPOSE: to list all rows from the MOSFETs table.

PROCEDURE: **dbo.ips_rubyconcaps_list**
PURPOSE: to list all rows from the RubyConCaps table.

PROCEDURE: **dbo.ips_transformercores_list**
PURPOSE: to list all rows from the TransformerCores table.

Chapter 7

Implementation

This chapter shows all implementations that have been done in this project. It mainly consists of three parts: Offline Client, Web Service and Web Application. User interfaces are shown and roughly explained. In addition, Major features are pointed out.

7.1 Offline Client

The Offline client is a Windows-based application. It was developed using Microsoft Visual Basic .NET. The main purpose of the offline client is to provide the possibility to use the system offline. It is very suitable for salespersons, field application engineers as well as external customers who have no 24-hour-ready Internet connection e.g. they are discussing with customers outside the office.

Figure 7.1: Offline Client Splash Screen

Since the application was designed for both expert users and non-expert users, therefore there are two possibilities to get a solution. For expert users, a

design can be created directly from the File menu as shown in figure 7.2. The user has to choose the topology, mode of operations as well as number of output voltages. For this approach, the user has to choose every major component manually. Alternatively, a design can be automatically generated from an optimizer by invoking "New Optimizer" command as shown in figure 7.3. The Optimizer asks for key input parameters and then lists all possible designs. This approach is very simple suitable for non-expert users. However, it is also suitable for expert users by allowing default values manipulation and sets of considered components modification.

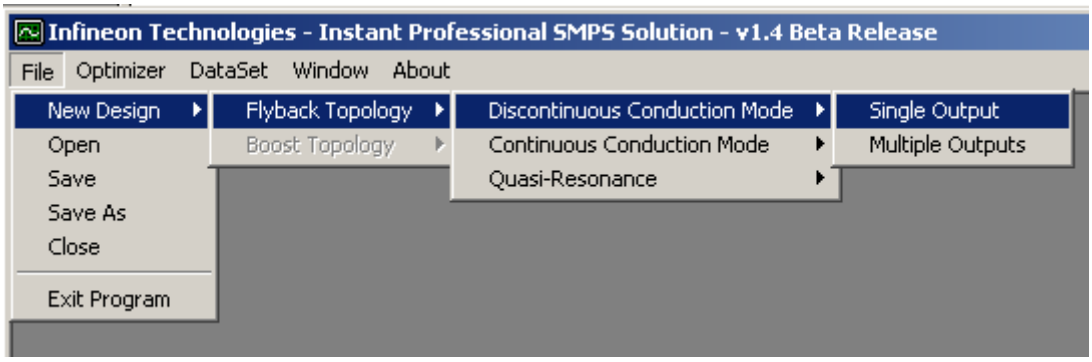


Figure 7.2: Getting a new design from scratch

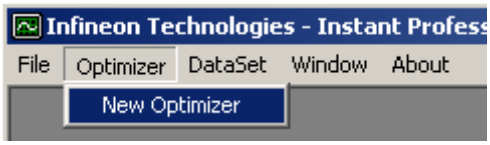


Figure 7.3: Getting a new design from an optimizer

7.1.1 SMPS Design

The SMPS design window as shown in figure 7.4 represents a design in the application i.e. one design one SMPS design window. There are three major parts in this window: input section, data items section and warning section. The input section as shown in figure 7.5 consists of combo boxes for choosing the CoolSET, Transformer Core and MOSFET. MOSFET combo box is disabled when the chosen CoolSET cannot use external MOSFET. Moreover, on the right area of the section, there are two check boxes and a button. The first check box is to specify whether the current design uses a heat sink or not. Remind that not all CoolSET can use a heat sink therefore if it cannot use a heat sink, the check box is disabled. The second check box is to specify if the copper area is used. Only a heat sink or a copper may be used at once. The export button is used to export the design to an Excel sheet. In addition, for multiple outputs design, the number of output voltages can be changed in the range of two up to ten.

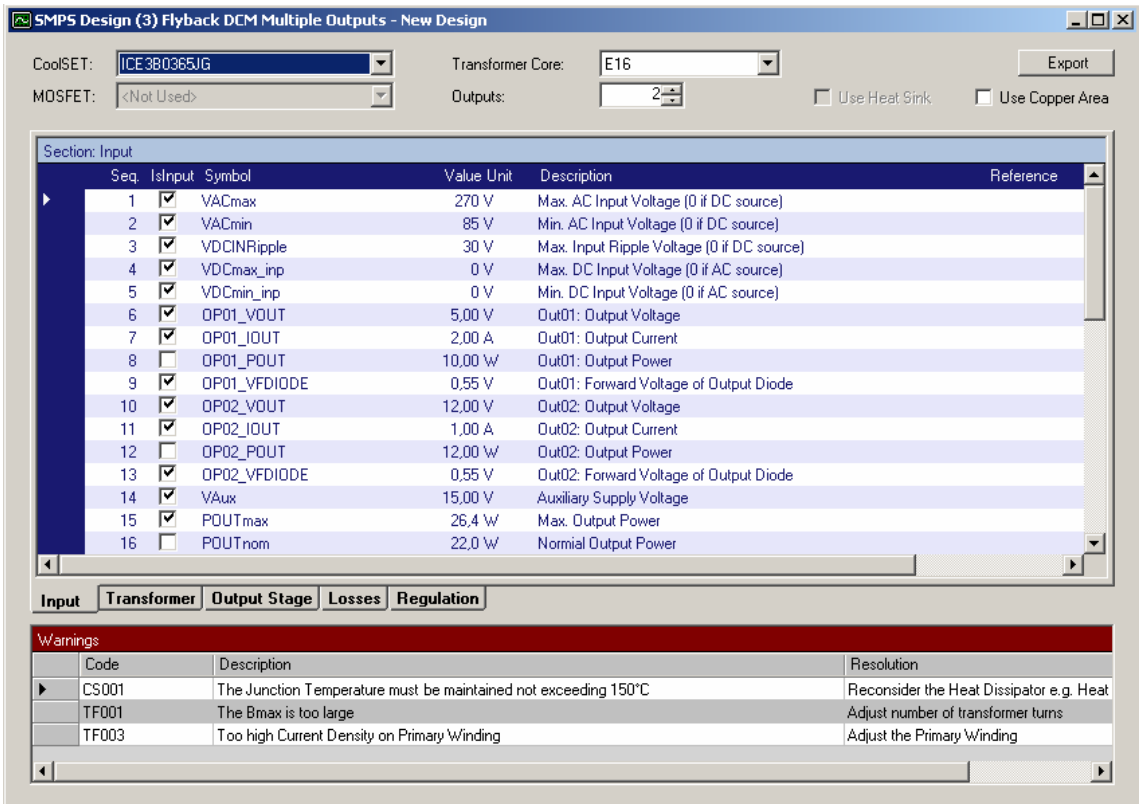


Figure 7.4: SMPS design window

Figure 7.5: Input section of the SMPS design window

The second section, the data items section, shows all data items of the design categorized in defined groups. Each group is represented by a tab page. As shown in figure 7.6, there are five groups. Each tab page consists of a data grid with data items of that group. The data grid has seven columns: sequence, input flag, symbol, value, unit, description and reference. The sequence column refers to the sequence number of the data item of the whole design calculation. The Input flag specifies whether the value of that data item can be changed or not. The symbol, value, unit and description column are self-explained. The reference column shows where the equation comes from or gives a hint on how to choose the value. The change of value will affect the data items having higher sequence number e.g. change of sequence number one's value will affect data items with sequence number two upwards. In addition, changes in the input section also affect data items as well.

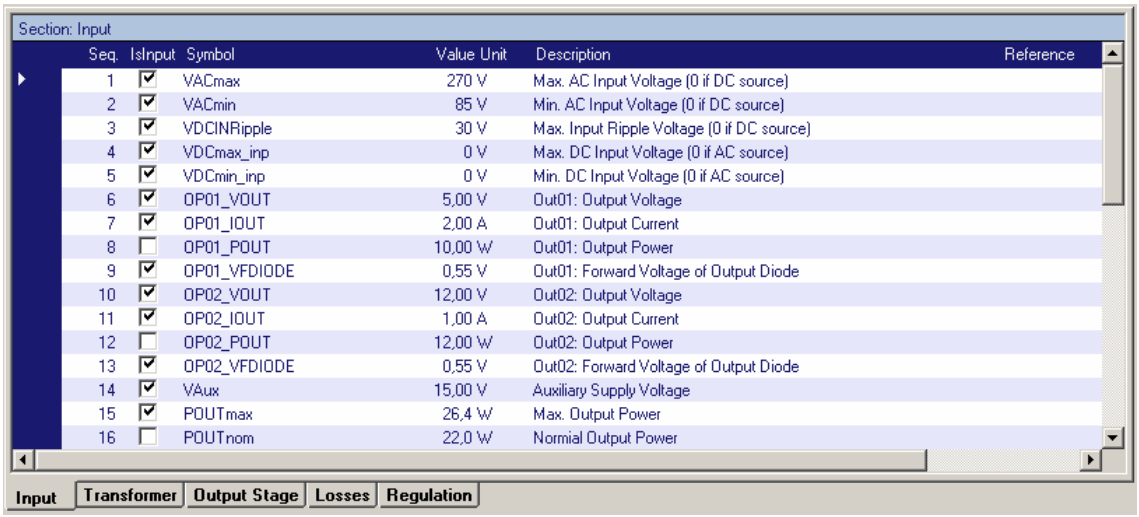


Figure 7.6: Data items section of the SMPS design window

The last section, the warning section as shown in figure 7.7, shows a list of warnings produced by the design. The list is represented by a data grid consisting of three columns: code, description and resolution. If the warning item is clicked, the data items section may point to the data item where the problem of that warning can be solved.

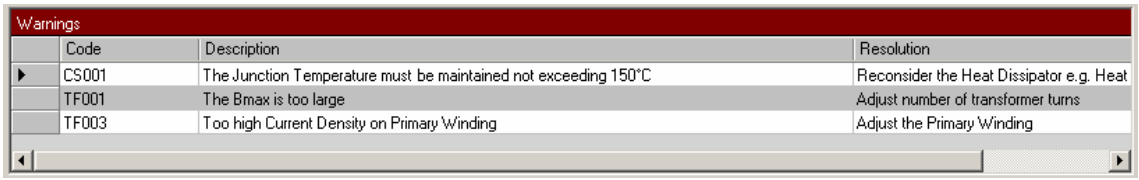


Figure 7.7: Warning section of the SMPS design window

7.1.2 Optimizer

The optimizer window is dedicated not only to non-expert users but also expert users. With the simplicity of the input process i.e. only few parameters are requested, it is very easy to use for non-expert users who have not much experience in the SMPS applications. However, it is also favorable for expert users who would like to play in detail parameters but do not want to manually try each combination of major components. The default values can be manipulated as well as sets of considered components. The optimizer window shown in figure 7.8 consists of three main tab pages: Basic Inputs, Advanced Inputs and Solutions.

The first tab, Basic Inputs, has four parts. The first part on the topmost left-hand side asks for the input voltage range which can be either alternating current (AC) or direct current (DC). The middle left-hand side part asks for the ambient

temperature. The lowermost left-hand side part is where the user can choose the topology/mode of the design. In addition, it requests additional topology/mode dependent key input parameters e.g. the Flyback DCM requires no additional topology/mode dependent key input parameters while the Flyback Quasi-Resonance shown in figure 7.9 requires two additional topology/mode dependent key input parameters.

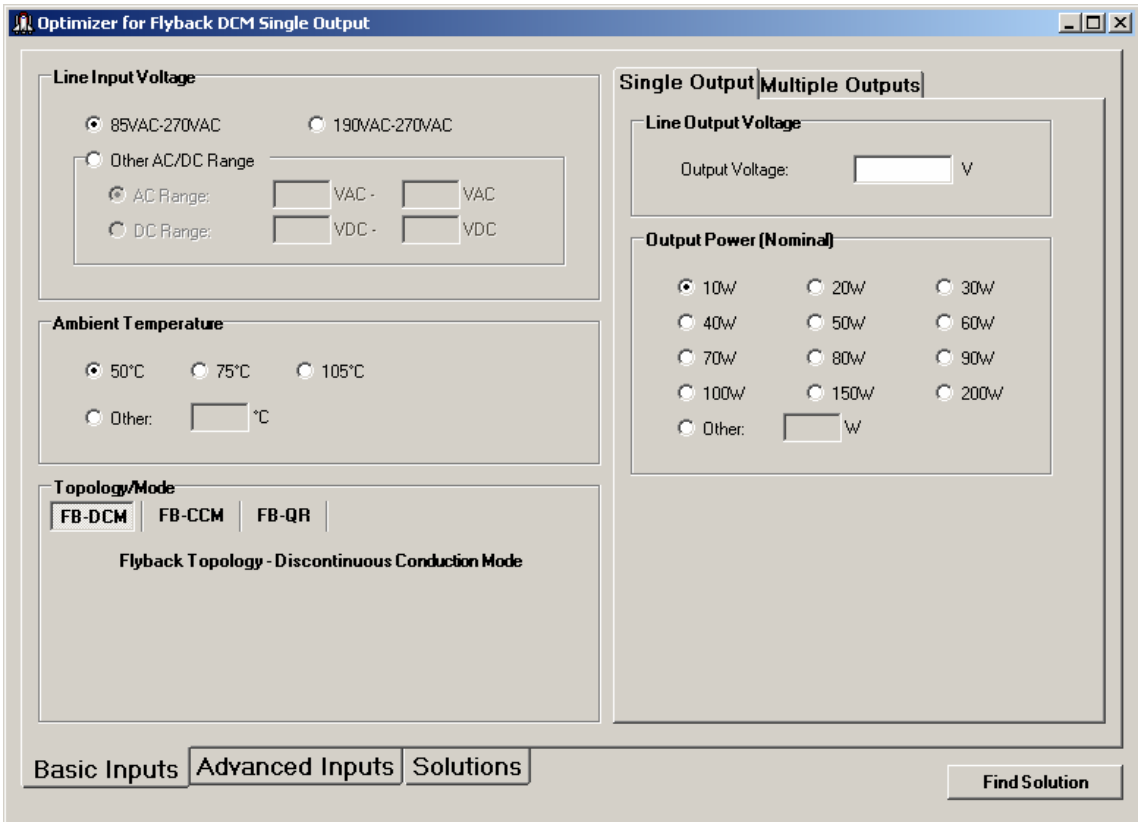


Figure 7.8: Optimizer window

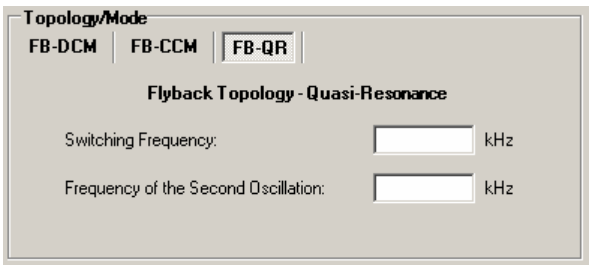


Figure 7.9: Topology/mode dependent key input parameters

The right-hand side part of the Basic Inputs consists of two tab pages: one for single output and the other for multiple outputs. The tab page also determines whether the solution will be single output or multiple outputs. For single output

tab page, it asks for the output voltage and the nominal output power. The multiple outputs tab page shown in figure 7.10 allows users to choose from two up to ten output voltages. For each output, the output voltage and the output current have to be specified. Remind that only the first output will be regulated.

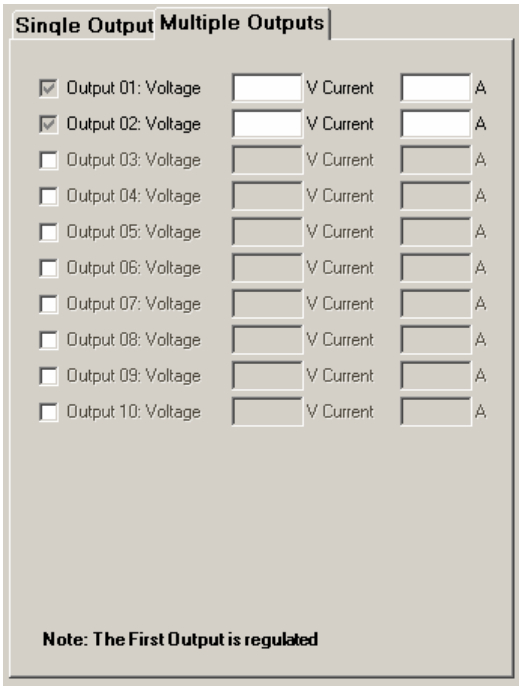


Figure 7.10: Multiple outputs tab page

Each input text box is controlled by the value validation control. All required inputs must be given before the "FindSolution" button is clicked. Some input has a valid range so the given value must be within the valid range. If there is any rule violation e.g. input missing, value out of valid range, the user will be notified by the red circle with a white exclamation symbol as shown in figure 7.11. Unless all notifications are cleared, the finding solution process will not proceed.

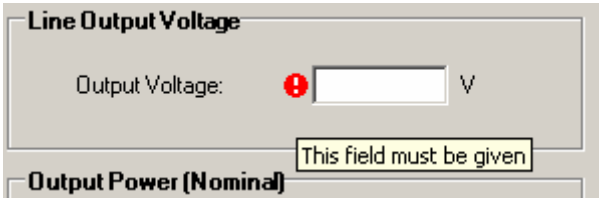


Figure 7.11: Input missing notification

The second tab, Advanced Inputs, is dedicated for expert users to play with detail parameters. It consists of five sub tab pages. The Heat Dissipater tab page has two parts. The upper part shown in figure 7.12 asks for the maximum allowed junction temperature. Typically, this value is set to 150°C however the user can decrease the limit to the lower temperature for some applications which heat is a potential concern. The lower part shown in figure 7.13a allows the user to choose the considered heat dissipaters. By default, all existing heat dissipater including customized items are selected. The "Check All" and "Uncheck All" buttons are provided for the user's convenience.

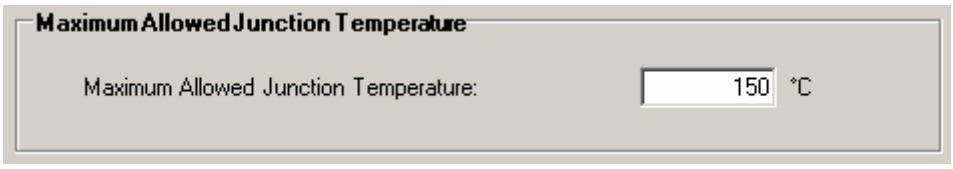


Figure 7.12: Maximum allowed junction temperature setting

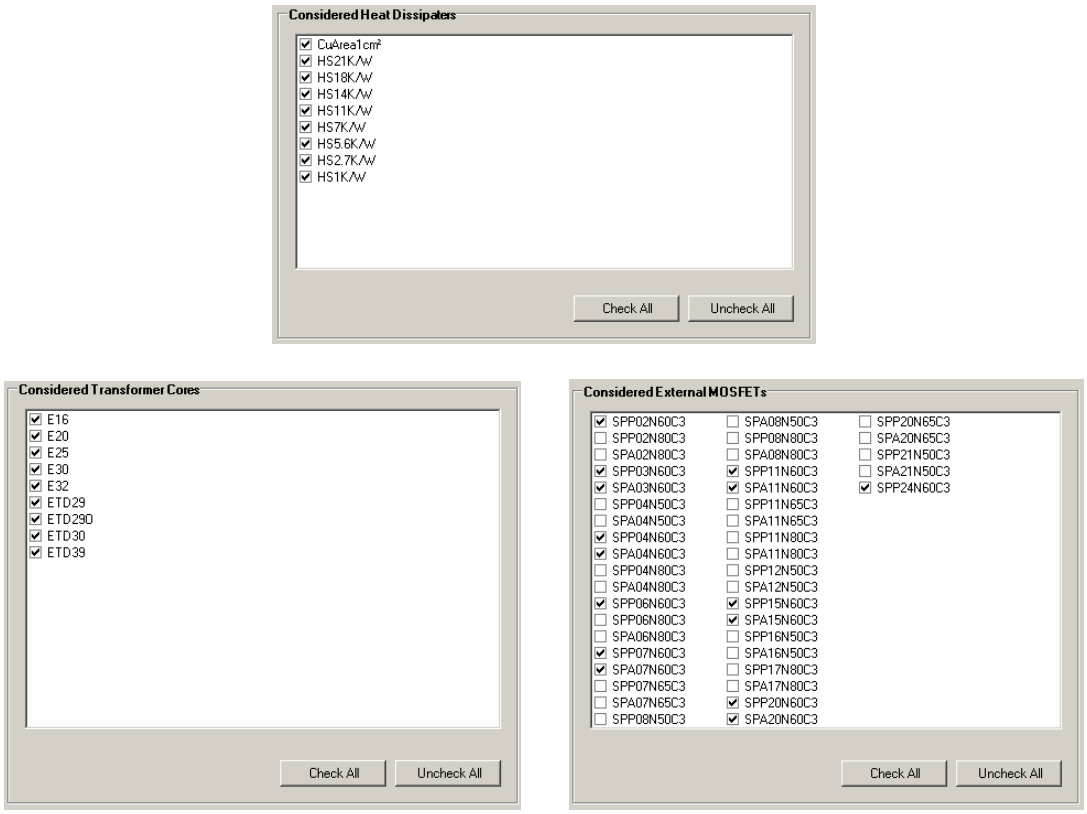


Figure 7.13: a) Considered heat dissipaters,
b) Considered transformer cores and c) Considered external MOSFETs

The Transformer Cores tab shown in figure 7.13b, allows the user to choose the considered transformer cores. Generally, all existing transformer cores including customized items are selected. The External MOSFETs tab shown in figure 7.13c enables the selection of the considered external MOSFETs. Normally, the 600V MOSFETs as well as customized items are chosen.

The Default Values tab shown in figure 7.14 allows the user to manipulate default values. All required default values for the optimizer of the chosen topology/mode are populated in the data grid. Here users have to change values with their own risk.

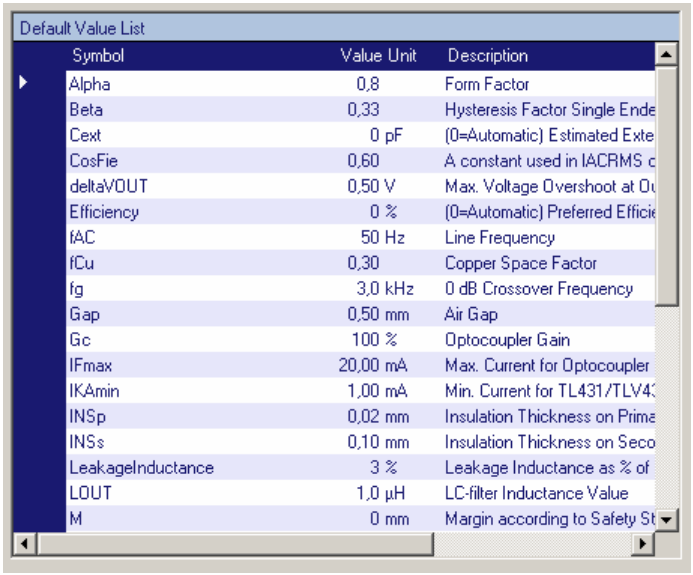


Figure 7.14: Default value list

The Warnings tab has a check box to specify whether the optimizer should neglect non-critical warnings occurred during optimization or not. This would result more solutions. It is unchecked by default and not recommended.

Figure 7.15: Neglecting non-critical warnings check box

The third tab page, solutions, shows results from the optimization. As shown in figure 7.16, the topmost area is the product filter where the user can specify the scope of interested products: all products, only hybrid products and only discrete products. Below the topmost area, there is a data grid showing the recommended optimal solution. In order to get a detail calculation, the user has to

click “Choose Solution” button. It will populate a new SMPS design window filled with the chosen solution data

Typically, the solution data grid consists of eight columns: CoolSET, Packaging, MOSFET, Heat Dissipater, Transformer Core, Maximum Junction Temperature, Total Power Losses and Expected Efficiency.

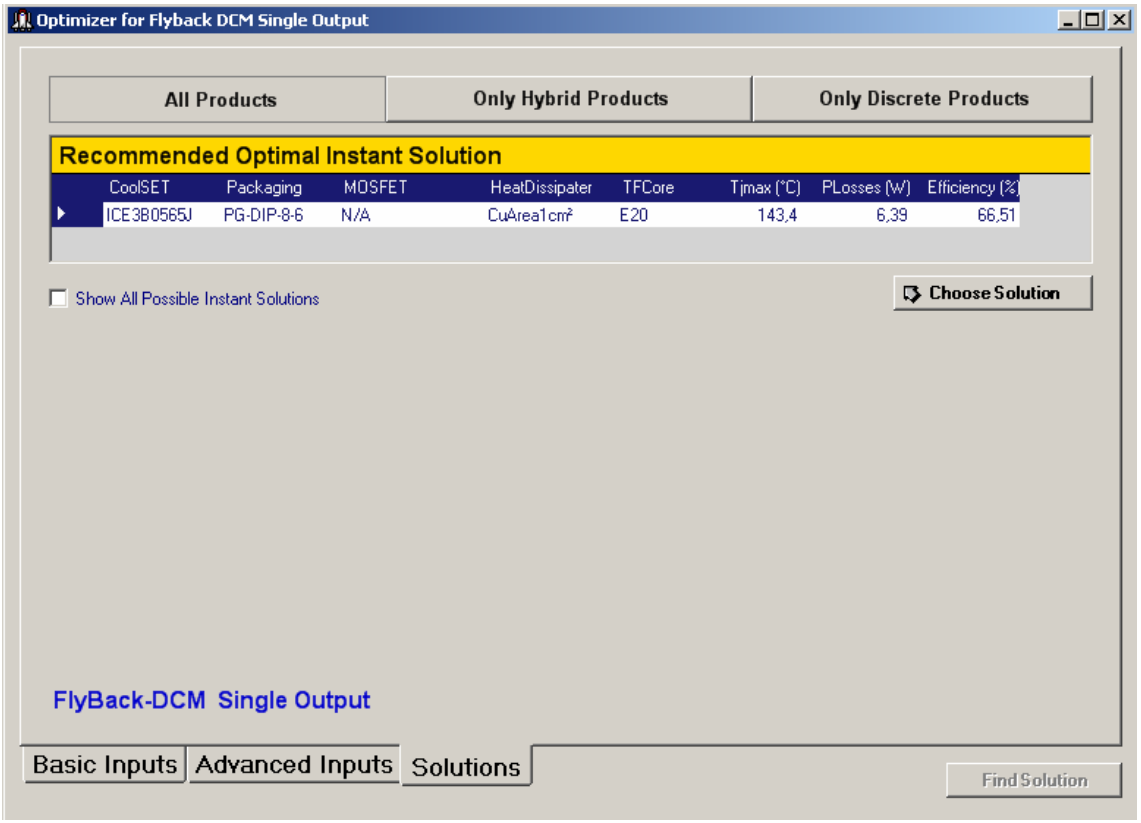


Figure 7.16: Solutions tab showing the recommended optimal solution

There is a check box “Show All Possible Instant Solutions” below the first data grid. If this check box is checked, the recommended optimal solution is hidden and the all possible solutions data grid is shown as figure 7.17.

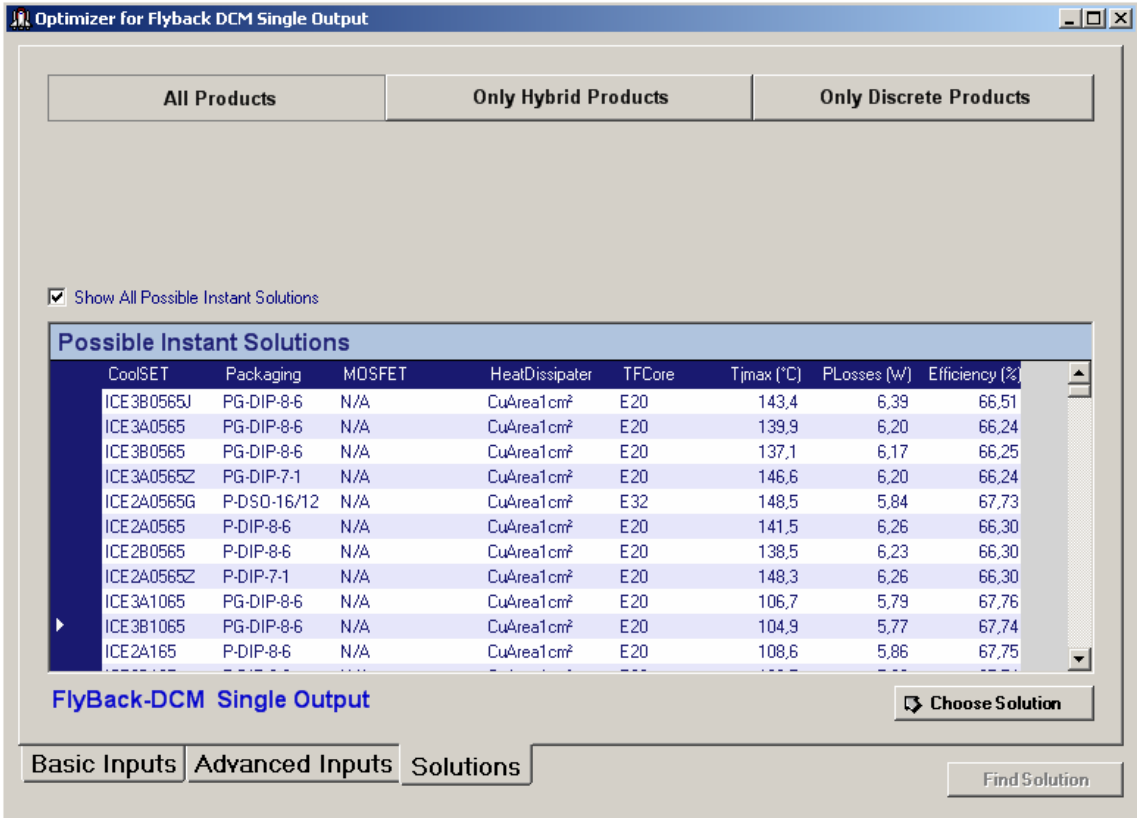


Figure 7.17: Solutions tab showing all possible solutions

7.1.3 Check New DataSet

The check new dataset command as shown in figure 7.18a is used to check whether the offline dataset is up to date or not. It first requests the new data digest from the web service supplying a challenge randomized number. After getting the data, it verifies with the existing dataset whether the digest is identical or not. If not, the new data transmission begins. The complete sequence flow can be seen in the section 7.2.

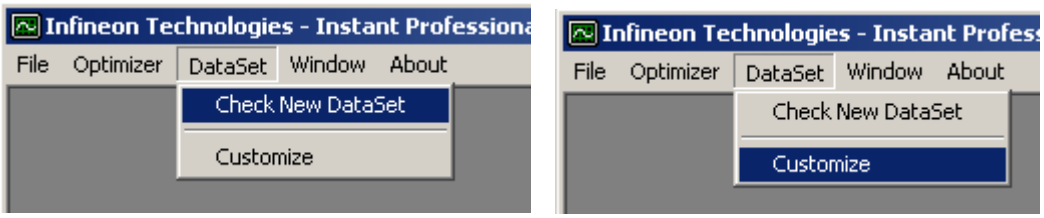


Figure 7.18: a) Check New DataSet command and b) Customize command

7.1.4 Customize Components

The customize command shown in figure 7.18b allows the user to customize heat dissipaters, transformer cores and MOSFETs. There are three tab pages in the customize window. The first tab shown in figure 7.19, heat dissipaters, provides the possibility to customize five copper areas and five heat sinks. The second tab, transformer cores shown in figure 7.20, allows the user customizing up to four transformer cores. The last tab, external MOSFETs shown in figure 7.21, lets the user customize up to four external MOSFETs.

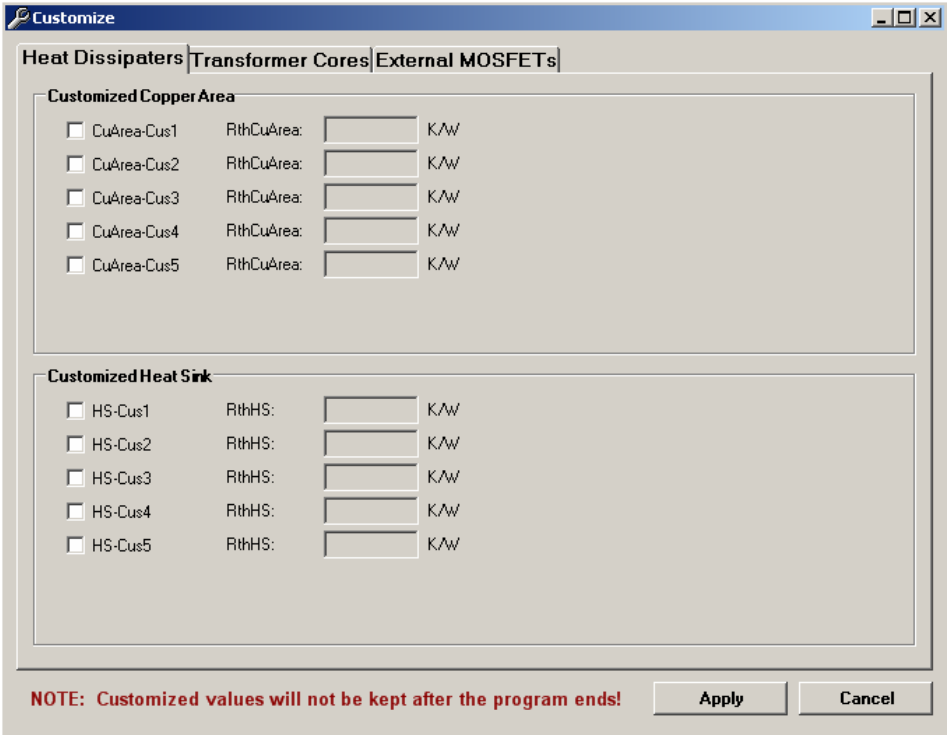


Figure 7.19: Customized heat dissipaters

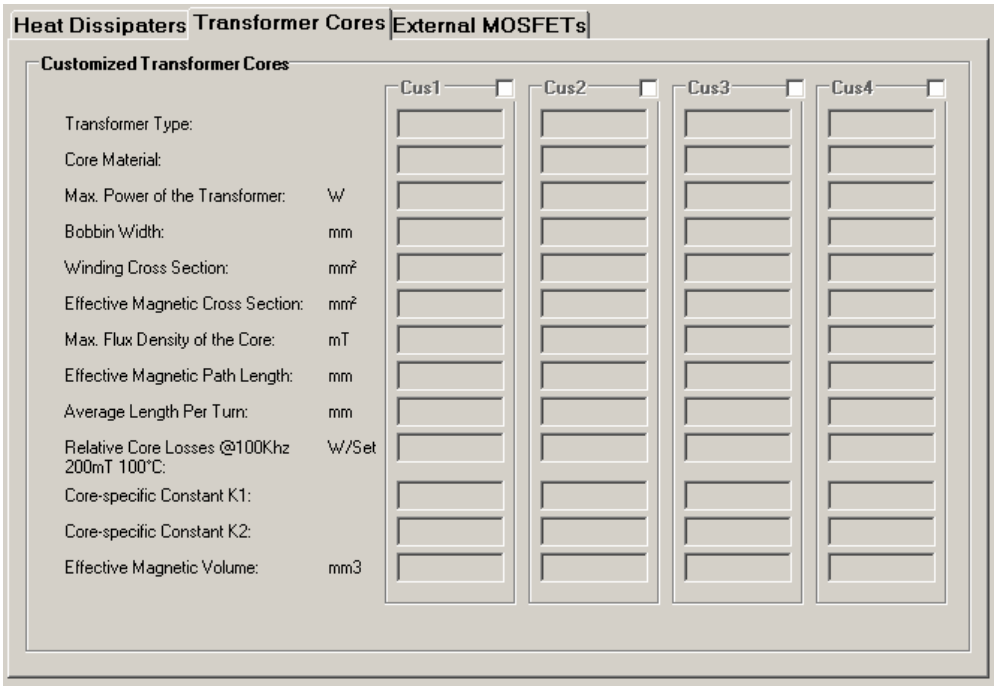


Figure 7.20: Customized transformer cores

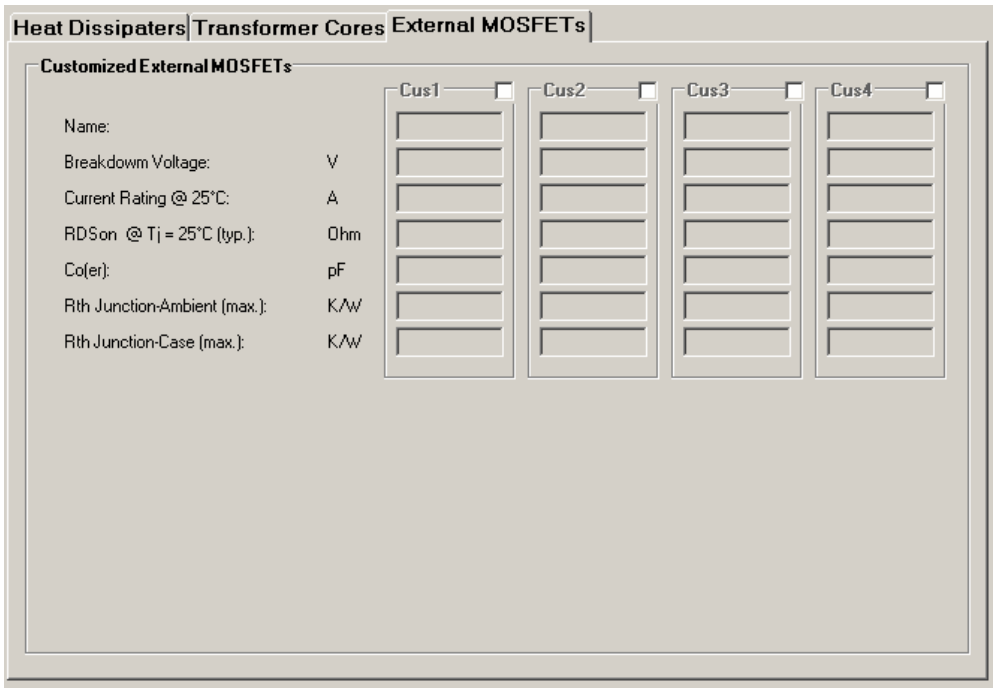


Figure 7.21: Customized external MOSFETs

7.2 Web Service

The web service shown in figure 7.22 is provided in order to periodically update the offline client dataset. The offline client invokes the web service which then gathers data from the underlying database. With the benefit of web service of encoding all transmitting data in XML-based text stream, problems with firewalls are minimal. In addition, the web service can be invoked from anywhere around the world where Internet is available. Since the web service provides a web method which is basically similar to conventional methods in term of usage, users do not have to manually download and update the dataset. Users can easily click a button and all the update process is done automatically. The simplified class declaration in VB.Net is shown in listing 7.1.

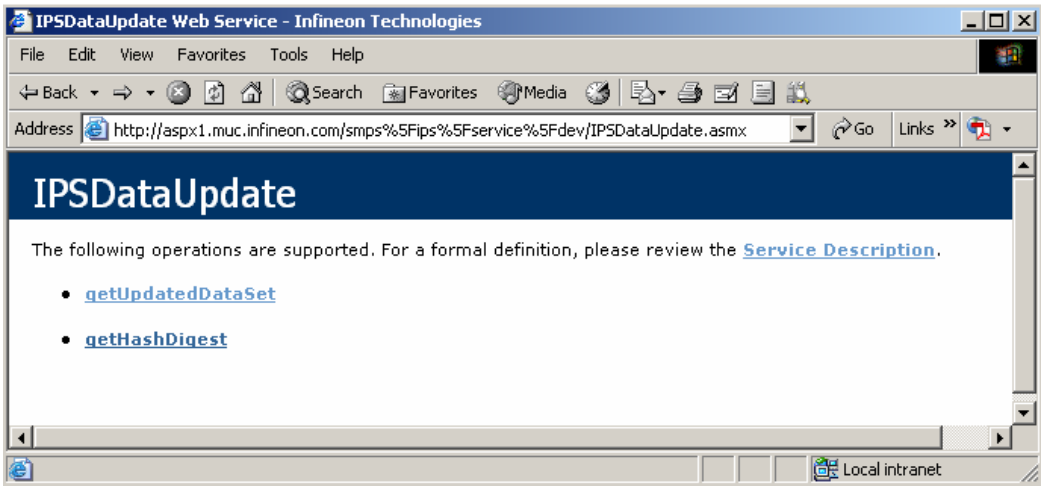


Figure 7.22: Data update web service

Listing 7.1: Class Declaration in VB.Net (simplified): IPSDataUpdate

```
<System.Web.Services.WebService( _
 Namespace:="http://www.infineon.com/IPS/IPSDataUpdate")> _
Public Class IPSDataUpdate
    Inherits System.Web.Services.WebService

   <WebMethod()> Public Function getUpdatedDataSet() As DataSet
   <WebMethod()> Public Function getHashDigest(ByVal _
                              challengingString As String) As String
End Class
```

The sequence diagram of the data update process is shown in figure 7.23. When a user requests an update by invoking the “getHashDigest” web method supplying a challenge randomized number in hexadecimal string encoded, the web service retrieves the actual data from the database. The data is then concatenated with the challenge randomized number. Afterwards, the concatenated data is

hashed using SHA-512 [11] hash function and returned to the client. The client concatenates the existing dataset with the challenge randomized number and hashes them with SHA-512 function. The client now can compare both hash digests whether they are identical or not. If not, the client invokes the "getUpdateDataSet" web method. The data is then transmitted to the client.

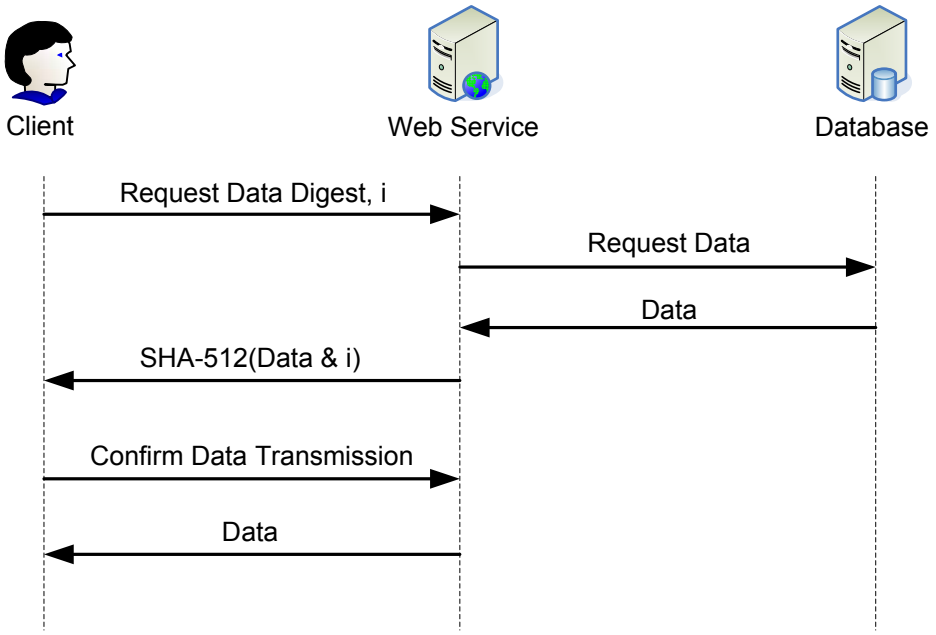


Figure 7.23: Data update process

7.3 Web Application

The functionalities of the web application are similar to the offline client. Unfortunately, due to technology limitations some features cannot be implemented in the web application. For an instance, the solutions caching mechanism has a problem with customized components and customized default values. In addition, to avoid unnecessary memory consumption on the server, the session time is introduced to limit the life time of the session data. The limited number of designs can be stored during the session. The sample screen of the web application is shown in figure 7.24. The web application is suitable for users who do not want to install an offline client into their computers. With the limitation in term of features, it might fulfill only basic queries. For advanced queries, it is recommended to use the offline client instead.

[11] Secure Hash Algorithm 512 bits

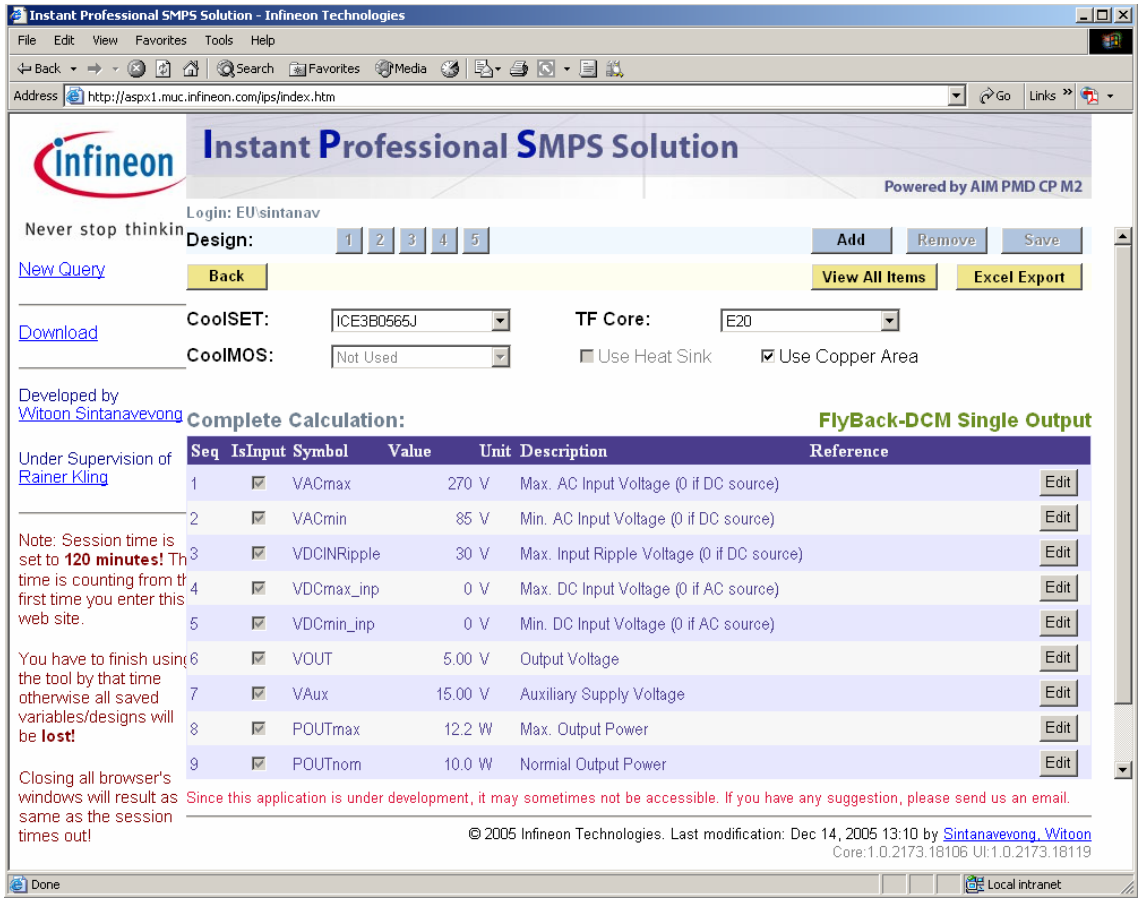


Figure 7.24: Web application

Chapter 8

Conclusion

The Solution Optimizer for SMPS Solutions greatly helps users find appropriate combination of electronic components. This could not be done effectively and efficiently in the former time when the manual selection was the only possibility. The optimizer also provides an opportunity for non-expert users who have little or no experience in SMPS applications to get a complete design without much effort.

The problem was analyzed and modeled as a tree. Each node represents a combination of major electronic components used in a solution. The leave nodes represent solutions which can be either valid or invalid. Validation rules were defined to scope down possibilities. The validation takes place for every step choosing a new node in the path. Due to the simplicity of the tree model, the backtracking approach was chosen.

The object model was designed in a dynamic and flexible manner. It focuses mainly on the controller ICs. The topologies and modes of operation are represented by interfaces in which product classes could implement. Variables and equations are represented by properties of the product class. The descriptive information of each property is provided using an attribute class enabling this information to be read dynamically at run-time via .NET reflection mechanism.

User interfaces were implemented in three parts: offline client, web service and web application. The offline client provides a possibility to do an optimization at local computer without connecting to the server. The dataset can be periodically updated via a simple update command which automatically connects to the web service. Solutions can be saved into an XML-based file. In addition, to ensure the consistency of the saved file, the XML schema validation is used. The second part, web service, provides a data update service which connects to the underlying database gathering new data. The last part, web application, provides users a possibility to find optimal solution online. The main features are similar to the offline client but due to technology limitations, some features cannot be implemented. The additional technique used in the web application is the caching of requests and solutions. This considerably helps reduce execution time which gives better responsiveness to online users.

Bibliography

[Baase00] Sara Baase and Allen Van Gelder: *Computer Algorithms: Introduction to Design & Analysis, 3rd Edition*. Addison-Wesley, USA April 2000.

[Date03] C. J. Date: *An Introduction to Database Systems, Eighth Edition*. Addison-Wesley, USA July 2003.

[Hromkovic04] Juraj Hromkovic: *Algorithmics for Hard Problems: Introduction to Cominatorial Optimization, Randomization, Approximation, and Heuristics, 2nd Edition*. Springer, Germany 2004.

[Hu05] Jing Hu: *Converter Design using the Quasi-Resonant PWM Controller ICE 2QS01, version 1.0*. Infineon Technologies, Singapore October 2005.

[IFX04] Infineon Technologies AG: *Semiconductors: Technical information, technologies and characteristic data, 2nd Edition*. Publicis Corporate Publishing, 2004.

[MSDN05] Microsoft Corporation: *.NET Framework Developer's Guide*. Microsoft Developer's Network, December 2005. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconreflectionoverview.asp

[Tarter93] Ralph E. Tarter: *Solid-State Power Conversion Handbook*. John Wiley & Sons Inc, USA 1993.

[Turley05] Paul Turley and Dan Wood: *Beginning Transact-SQL with SQL Server 2000 and 2005*. Wrox, USA October 2005.

[Zöllinger02] Harald Zöllinger and Rainer Kling: *CoolSET ICE2xXXX for Off-line switched-mode power supply, Application Note version 1.2*. Infineon Technologies AG, Germany February 2002.

Appendix A

IPS Design XML Schema

XML-based IPS design files (typically with .ips extension) can be easily controlled by the XML Schema validation to ensure the file consistency avoiding errors in reconstructing an object model of the design. Thus the XML schema is defined as the following:

```
<!--IPS Design XML Schema version 1.0
    Infineon Technologies AG, AIM PMD CP M2
    By Witoon Sintanavevong
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 id="IPSDesign" xmlns:ips="http://www.infineon.com/aim/pmd/cp/m2/ips"
 targetNamespace="http://www.infineon.com/aim/pmd/cp/m2/ips">
   <xsd:complexType name="OutputType">
      <xsd:sequence>
         <xsd:element name="Number" type="xsd:integer" />
         <xsd:element name="VOUT" type="xsd:double" />
         <xsd:element name="IOUT" type="xsd:double" />
         <xsd:element name="VFDIODE" type="xsd:double" />
         <xsd:element name="Asec_AWG" type="xsd:double" />
         <xsd:element name="INSs" type="xsd:double" />
         <xsd:element name="NPs" type="xsd:integer" />
         <xsd:element name="Ns" type="xsd:integer" />
         <xsd:element name="VCOUT" type="xsd:double" />
         <xsd:element name="COUT" type="xsd:double" />
         <xsd:element name="RESR" type="xsd:double" />
         <xsd:element name="Iacmax" type="xsd:double" />
         <xsd:element name="nc" type="xsd:integer" />
         <xsd:element name="VCLC" type="xsd:double" />
         <xsd:element name="CLC" type="xsd:double" />
         <xsd:element name="LOUT" type="xsd:double" />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="Output" type="ips:OutputType" />
   <xsd:complexType name="TransformerCoreType">
      <xsd:sequence>
         <xsd:element name="TFName" type="xsd:string" />
         <xsd:element name="Type" type="xsd:string" />
         <xsd:element name="Material" type="xsd:string" />
         <xsd:element name="Pmax" type="xsd:double" />
         <xsd:element name="BW" type="xsd:double" />
         <xsd:element name="AN" type="xsd:double" />
         <xsd:element name="Ae" type="xsd:double" />
         <xsd:element name="Bmax" type="xsd:double" />
         <xsd:element name="Gap" type="xsd:double" />
         <xsd:element name="le" type="xsd:double" />
         <xsd:element name="lN" type="xsd:double" />
         <xsd:element name="PV" type="xsd:double" />
         <xsd:element name="K1" type="xsd:double" />
         <xsd:element name="K2" type="xsd:double" />
```

```
      <xsd:element name="Ve" type="xsd:double" />
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="TransformerCore" type="ips:TransformerCoreType" />
<xsd:complexType name="MOSFETType">
   <xsd:sequence>
      <xsd:element name="MOSFETName" type="xsd:string" />
      <xsd:element name="MOSFETSortableName" type="xsd:string" />
      <xsd:element name="Voltage" type="xsd:double" />
      <xsd:element name="IDAt25C" type="xsd:double" />
      <xsd:element name="RDSonAt25C" type="xsd:double" />
      <xsd:element name="Coer" type="xsd:double" />
      <xsd:element name="Tj" type="xsd:double" />
      <xsd:element name="RthJA" type="xsd:double" />
      <xsd:element name="RthJC" type="xsd:double" />
      <xsd:element name="Alpha" type="xsd:double" />
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="MOSFET" type="ips:MOSFETType" />
<xsd:complexType name="FlyBackType">
   <xsd:sequence>
      <xsd:element name="CosFie" type="xsd:double" />
      <xsd:element name="Efficiency" type="xsd:double" />
      <xsd:element name="fAC" type="xsd:double" />
      <xsd:element name="Gc" type="xsd:double" />
      <xsd:element name="POUTmax" type="xsd:double" />
      <xsd:element name="POUTmin" type="xsd:double" />
      <xsd:element name="POUTnom" type="xsd:double" />
      <xsd:element name="VACmax" type="xsd:double" />
      <xsd:element name="VACmin" type="xsd:double" />
      <xsd:element name="VDCmax_inp" type="xsd:double" />
      <xsd:element name="VDCmin_inp" type="xsd:double" />
      <xsd:element name="VAux" type="xsd:double" />
      <xsd:element name="VDCINRipple" type="xsd:double" />
      <xsd:element name="VRmax" type="xsd:double" />
      <xsd:element name="Tj" type="xsd:double" />
      <xsd:element name="CIN" type="xsd:double" />
      <xsd:element name="Np" type="xsd:integer" />
      <xsd:element name="NAux" type="xsd:integer" />
      <xsd:element name="fCu" type="xsd:double" />
      <xsd:element name="M" type="xsd:double" />
      <xsd:element name="Ap_AWG" type="xsd:double" />
      <xsd:element name="INSp" type="xsd:double" />
      <xsd:element name="NPp" type="xsd:integer" />
      <xsd:element name="VFAux" type="xsd:double" />
      <xsd:element name="RSense" type="xsd:double" />
      <xsd:element name="LeakageInductance" type="xsd:double" />
      <xsd:element name="CClamp" type="xsd:double" />
      <xsd:element name="RClamp" type="xsd:double" />
      <xsd:element name="DeltaVOUT" type="xsd:double" />
      <xsd:element name="nCP" type="xsd:integer" />
      <xsd:element name="VCOUT" type="xsd:double" />
      <xsd:element name="COUT" type="xsd:double" />
      <xsd:element name="RESR" type="xsd:double" />
      <xsd:element name="Iacmax" type="xsd:double" />
      <xsd:element name="nc" type="xsd:integer" />
      <xsd:element name="CLC" type="xsd:double" />
      <xsd:element name="LOUT" type="xsd:double" />
      <xsd:element name="tSstart" type="xsd:double" />
      <xsd:element name="CSS" type="xsd:double" />
      <xsd:element name="CVCC" type="xsd:double" />
      <xsd:element name="Cext" type="xsd:double" />
      <xsd:element name="VF" type="xsd:double" />
      <xsd:element name="Alpha" type="xsd:double" />
      <xsd:element name="Beta" type="xsd:double" />
      <xsd:element name="p100" type="xsd:double" />
      <xsd:element name="Ta" type="xsd:double" />
      <xsd:element name="VREF_TD" type="xsd:double" />
      <xsd:element name="VFD_TD" type="xsd:double" />
      <xsd:element name="IKAmin" type="xsd:double" />
      <xsd:element name="IFmax" type="xsd:double" />
```

```
         <xsd:element name="R2" type="xsd:double" />
         <xsd:element name="fg" type="xsd:double" />
         <xsd:element name="R1" type="xsd:double" />
         <xsd:element name="R3" type="xsd:double" />
         <xsd:element name="R4" type="xsd:double" />
         <xsd:element name="R5" type="xsd:double" />
         <xsd:element name="C2" type="xsd:double" />
         <xsd:element name="C1" type="xsd:double" />
         <xsd:element ref="ips:Output" minOccurs="1" maxOccurs="unbounded" />
         <xsd:element name="IChargeC" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="RStart" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="f" type="xsd:double" minOccurs="0"
                      maxOccurs="1" />
         <xsd:element name="fOSC2" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="Rtdn1" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="Rtdn2" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="Ctdn3" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
         <xsd:element name="deltaIL" type="xsd:double"
                      minOccurs="0" maxOccurs="1" />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="FlyBack" type="ips:FlyBackType" />
   <xsd:complexType name="IPSDesignType">
      <xsd:sequence>
         <xsd:element name="CoolSETName" type="xsd:string" />
         <xsd:element ref="ips:TransformerCore" />
         <xsd:element ref="ips:MOSFET" minOccurs="0" />
         <xsd:element name="UseHeatSink" type="xsd:boolean" />
         <xsd:element name="UseCuArea" type="xsd:boolean" />
         <xsd:element name="RthCuArea" type="xsd:double" />
         <xsd:element name="RthHT" type="xsd:double" />
         <xsd:element name="RthHS" type="xsd:double" />
         <xsd:element name="Topology" type="xsd:string" />
         <xsd:choice>
            <xsd:element ref="ips:FlyBack" />
         </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="Version" type="xsd:string" use="required" />
   </xsd:complexType>
   <xsd:element name="IPSDesign" type="ips:IPSDesignType" />
</xsd:schema>
```

Appendix B

SMPS Design Procedure

Since there are so many design procedures varying from topology to topology and from mode to mode, in order to have a rough understanding of how the SMPS design could be calculated, a complete simple SMPS design procedure is shown in this section: the design procedure for fixed-frequency single-output Flyback Converter with Infineon CoolSET ICE2xXXX operating in Discontinuous Conduction Mode.

Table B.1: Design procedure for Flyback DCM using CoolSET-F2

Procedure		Example
Define Input Parameters:-		
Max. AC Input Voltage:	V_{ACmax}	264 V
Min. AC Output Voltage:	V_{Acmin}	90 V
Max. Input Ripple Current:	$V_{DCIN\ Ripple}$	30 V
Output Voltage:	V_{OUT}	16 V
Auxiliary Voltage:	V_{Aux}	12 V
Max. Output Power:	P_{OUTmax}	50 W
Nom. Output Power:	P_{OUTnom}	40 W
Min. Output Power:	P_{OUTmin}	0.5 W
Preferred Efficiency:	η_P	85 %
Optocoupler Gain:	G_c	100 %
Max. Reflected Output Voltage:	V_{Rmax}	120 V
Line Frequency:	f_{AC}	50 Hz
Fwd. Voltage of Output Diode:	V_{FDIODE}	0.85 V
Fwd. Voltage of Aux. Out. Diode:	V_{Faux}	0.85 V
Ambient Temperature:	T_a	50 °C
CoolSET:		**ICE2A0365**
Switching Frequency:	f	100 kHz
Breakdown Voltage:	V_{BR_DSS}	650 V
Typ. Peak Current Limitation:	V_{csth}	1 V
Typ. Feedback Resistance:	R_{FB}	3.7 kΩ
Typ. Trimmed Ref. Voltage:	V_{REF}	6.5 V
Min. Soft-Start Resistance:	R_{SS}	42 kΩ
Min. Act. Lim. Of OL/OPL Dect.:	V_{SS1}	5.15 V
Max. Start-up Current:	I_{VCC1}	55 µA
Max. VCC Current Act. Gate:	I_{VCC3}	9.8 mA

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Typ. VCC On/Off Hysteresis:	V_{CCHY}	5 V
Typ. VCC Turn-On Threshold:	V_{CCon}	13.5 V
Typ. Drain Source On Resistance:	$R_{DSon@25°C}$	0.45 Ω
Typ. Eff. Output Capacitance:	$C_{o(er)}$	30 pF
Max. Thermal Resistance J-A:	R_{thJA}	90 K/W
Typ. PWM-OP Gain:	A_V	3.65
Max. Operating FB Voltage:	V_{FBmax}	4.6 V
There is no special requirement imposed on the input rectifier and storage capacitor in the Flyback converter. The components will be chosen to meet the power rating and hold-up requirements.		
Max. Input Power: $P_{IN\max} = \frac{P_{OUT\max}}{\eta_P}$	**(Eq. 1)**	$P_{IN\max} = \frac{50W}{85\%} = 58.8W$
Input Diode Bride:		
Power Factor	cosφ	0.6
Input RMS Current: $I_{ACRMS} = \frac{P_{IN\max}}{V_{AC\min} \cdot \cos\varphi}$	**(Eq. 2)**	$I_{ACRMS} = \frac{58.8W}{90V \cdot 0.6} = 1.09A$
Max. DC Input Voltage: $V_{DC\max PK} = V_{AC\max} \cdot \sqrt{2}$	**(Eq. 3)**	$V_{DC\max PK} = 264V \cdot \sqrt{2} = 373.4V$
Input Capacitor:		
Min. Peak Input Voltage at no load condition: $V_{DC\min PK} = V_{AC\min} \cdot \sqrt{2}$	**(Eq. 4)**	$V_{DC\min PK} = 90V \cdot \sqrt{2} = 127.3V$
$V_{DC\min\_pre} = V_{DC\min PK} - V_{DCINRipple}$	**(Eq. 5)**	$V_{DC\min\_pre} = 127.3V - 30V = 97.3V$
Discharging time at each half-line cycle: $T_D = \frac{1}{4 \cdot f_{AC}} \cdot \left(1 + \frac{\arcsin \frac{V_{DC\min\_pre}}{V_{DC\min PK}}}{90} \right)$	**(Eq. 6)**	$T_D = \frac{1}{4 \cdot 50Hz} \cdot \left(1 + \frac{\arcsin \frac{97.3V}{127.3V}}{90} \right) = 7.77ms$
Required Energy at discharging time of Input Capacitor: $W_{IN} = P_{IN\max} \cdot T_D$	**(Eq. 7)**	$W_{IN} = 58.8W \cdot 7.77ms = 0.46W \cdot s$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
Input Capacitor (cal.): $C_{IN\_cal} = \frac{2 \cdot W_{IN}}{V^2_{DC\,min\,PK} \cdot V^2_{DC\,min\_pre}}$ **(Eq. 8a)**	$C_{IN\_cal} = \frac{2 \cdot 0.46W \cdot s}{(127.3V)^2 \cdot (97.3)^2} = 135.7\mu F$
Alternatively, a rule of thumb of choosing Input Capacitor may be applied: Input Voltage — Factor 115V — 2µF/W 230V — 1µF/W 85V…270V — 2…3µF/W	
$C_{IN\_cal} = P_{IN\,max} \cdot factor$ **(Eq. 8b)**	$C_{IN\_cal} = 58.8W \cdot 2.5\frac{\mu F}{W} = 146.45\mu F$
Select an Input Capacitor from the Epcos Databook of **Aluminum Electrolytic Capacitors** The following types are preferred: For 85°C Applications: Series B43303… 2000hrs life time B43501… 10000hrs life time For 105°C Applications: Series **B43504…** **2000hrs life time** B43505… 5000hrs life time	
Choose the Rated Voltage greater or equal to the calculated $V_{DCmaxPK}$	Since $V_{DCmaxPK}$ = 373.4V, choose 400V
Choose the Capacitance greater or equal to the calculated C_{IN_cal} from Eq. 8a	Since C_{IN_cal} = 135.7µF, choose 150µF
Input Capacitor: C_{IN}	150 µF
Recalculation after Input Capacitor chosen: $V_{DC\,min} = \sqrt{V^2_{DC\,min\,PK} - \frac{2 \cdot W_{IN}}{C_{IN}}}$ **(Eq. 9)**	$V_{DC\,min} = \sqrt{(127.3V)^2 - \frac{2 \cdot 0.46W \cdot s}{150\mu F}} = 100.5V$
Note that special requirements for hold up time, including cycle skip/dropout, or other factors which affect the resulting minimum DC input voltage and capacitor time should be considered at this point as well.	

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Transformer Design:		
Max. Duty Cycle (preliminary): $D_{\max\_pre} = \frac{V_{R\max}}{V_{R\max} + V_{DC\min}}$	**(Eq. 10)**	$D_{\max\_pre} = \frac{120V}{120V + 100.5V} = 0.544$
Peak Current of Primary Inductance (pre.): $I_{LPK\_pre} = \frac{2 \cdot P_{IN\max}}{V_{DC\min} \cdot D_{\max\_pre}}$	**(Eq. 11)**	$I_{LPK\_pre} = \frac{2 \cdot 58.8W}{100.5V \cdot 0.544} = 2.15A$
RMS Current of Primary Inductance (pre.): $I_{LRMS} = I_{LPK\_pre} \cdot \sqrt{\frac{D_{\max\_pre}}{3}}$	**(Eq. 12)**	$I_{LRMS} = 2.15A \cdot \sqrt{\frac{0.544}{3}} = 0.92A$
Primary Inductance within the limit of **Max. Duty Cycle**: $L_{P\_pre} = \frac{D_{\max\_pre} \cdot V_{DC\min}}{I_{LPK\_pre} \cdot f}$	**(Eq. 13)**	$L_{P\_pre} = \frac{0.544 \cdot 100.5V}{2.15A \cdot 100kHz} = 254\mu H$
Select Core Type and Inductance Factor (AL) from **Epcos "Ferrite Databook" or CD-ROM "Passive Components"**. Fix Max. Flux Density: Typically, $B_{max} \approx 0.2T \ldots 0.4T$ for Ferrite Cross depending on Core Material. We choose **300mT** for Material **N87**.		Select Core: **E 25/13/7** Material = N87 BW = 15.6mm A_N = 61mm² l_N = 50mm A_e = 52.5mm² K_1 = 90 l_e = 57.5mm K_2 = -0.731 P_V = 1.6W/Set @100°C, 100kHz, 200mT
Air Gap:	Gap	0.6 mm
Inductance Factor: $A_L = K_1 \cdot Gap^{K_2}$	**(Eq. 14)**	$A_L = 90 \cdot (0.6mm)^{-0.731} = 130.74nH$
Number of Primary Inductance (cal.): $N_{P\_cal} = \sqrt{\frac{L_{P\_pre}}{A_L}}$	**(Eq. 15)**	$N_{P\_cal} = \sqrt{\frac{254\mu H}{130.74nH}} = 44.08Turns$
Choose the number of Primary Turns in which makes the following conditions satisfied: - D_{max} (Eq. 26) $\leq$ 0.72 (typ. max. D_{max}) - (D_{max} (Eq. 26) + D'_{max} (Eq 27)) $\leq$ 1 - B_{max} (Eq. 28) $\leq$ B_{max} of the Transformer Core - N_{S_cal} (Eq. 16) very close to integer - Maximum N_S possible		

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
If there is no suitable value found, try to increase the Gap by 0.1mm step until it reaches 1mm. Otherwise, increase the Input Capacitor size by one step.		
Number of Primary Turns:	N_P	42 Turns
Number of Secondary Turns (cal.): $N_{S\_cal} = \frac{N_P \cdot (V_{OUT} + V_{FDIODE})}{V_{R\max}}$	**(Eq. 16)**	$N_{S\_cal} = \frac{42 \cdot (16V + 0.85V)}{120V} = 5.9Turns$
Number of Secondary Turns:	N_S	6 Turns
Number of Auxiliary Turns (cal.): $N_{Aux\_cal} = \frac{N_P \cdot (V_{Aux} + V_{FAux})}{V_{R\max}}$	**(Eq. 17)**	$N_{Aux\_cal} = \frac{42 \cdot (12V + 0.85V)}{120V} = 4.5Turns$
Number of Auxiliary Turns:	N_{Aux}	5 Turns
Eff. IC Auxiliary Supply Voltage: $V_{Aux\_cal} = \frac{N_{Aux}}{N_S} \cdot (V_{OUT} + V_{FDIODE}) - V_{FAux}$	**(Eq. 18)**	$V_{Aux\_cal} = \frac{5}{6} \cdot (16V + 0.85V) - 0.85V = 13.2V$
Eff. Primary Inductance: $L_P = N_P^2 \cdot A_L$	**(Eq. 19)**	$L_P = (42)^2 \cdot 130.74nH = 231\mu H$
Primary Peak Current before R_{Sense} chosen: $I_{LPK} = \sqrt{\frac{2 \cdot P_{IN\max}}{L_P \cdot f}}$	**(Eq. 20)**	$I_{LPK} = \sqrt{\frac{2 \cdot 58.8W}{231\mu H \cdot 100kHz}} = 2.26A$
Sense Resistor:		
The Sense Resistance can be used to individually define the max. peak current and thus the max. power transmitted. **Caution:** When calculating the max. peak current, short term peaks in output power must also be taken into consideration.		
Sense Resistor (cal.): $R_{Sense\_cal} = \frac{V_{csth}}{I_{LPK}}$	**(Eq. 21)**	$R_{Sense\_cal} = \frac{1V}{2.26A} = 442.76m\Omega$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Choose Sense Resistor from table E24 for Resistance < 1Ω and table E96 for ≥ 1Ω with the closest lower value.		
Sense Resistor:	RSense	430 mΩ
Power Rating of Sense Resistor: $P_{SR} = I_{PRMS}^2 \cdot R_{Sense}$	**(Eq. 22)**	$P_{SR} = (0.98A)^2 \cdot 430m\Omega = 0.41W$
Eff. Primary Peak Current: $I_{LPK\_SR} = \frac{V_{csth}}{R_{Sense}}$	**(Eq. 23)**	$I_{LPK\_SR} = \frac{1V}{430m\Omega} = 2.33A$
Eff. Max. Output Power: $P_{OUT\,max\_SR} = \frac{1}{2} \cdot f \cdot L_P \cdot I_{LPK\_SR}^2 \cdot \eta_P$	**(Eq. 24)**	$P_{OUT\,max\_SR} = \frac{1}{2} \cdot 100kHz \cdot 231\mu H \cdot (2.33A)^2 \cdot 85\% = 53W$
Reflected Voltage: $V_R = \frac{(V_{OUT} + V_{FDIODE}) \cdot N_P}{N_S}$	**(Eq. 25)**	$V_R = \frac{(16V + 0.85V) \cdot 42}{6} = 118V$
Max. Turn-On Duty Cycle: $D_{max} = \frac{L_P \cdot I_{LPK\_SR} \cdot f}{V_{DC\,min}}$	**(Eq. 26)**	$D_{max} = \frac{231\mu H \cdot 2.33A \cdot 100kHz}{100.5V} = 0.536$
Max. Turn-Off Duty Cycle: $D'_{max} = \frac{L_P \cdot I_{LPK\_SR} \cdot f}{V_R}$	**(Eq. 27)**	$D'_{max} = \frac{231\mu H \cdot 2.33A \cdot 100kHz}{118V} = 0.456$
Max. Flux Density: $B_{max} = \frac{L_P \cdot I_{LPK\_SR}}{N_P \cdot A_e}$	**(Eq. 28)**	$B_{max} = \frac{231\mu H \cdot 2.33A}{42 \cdot 52.5mm^2} = 244mT$
Eff. Primary RMS Current: $I_{PRMS} = \sqrt{\frac{D_{max}}{3}} \cdot I_{LPK\_SR}$	**(Eq. 29)**	$I_{PRMS} = \sqrt{\frac{0.536}{3}} \cdot 2.33A = 0.98A$
Eff. Secondary Peak Current: $I_{SPK} = I_{LPK\_SR} \cdot \frac{N_P}{N_S}$	**(Eq. 30)**	$I_{SPK} = 2.33A \cdot \frac{42}{6} = 16.31A$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Eff. Secondary RMS Current: $I_{SRMS} = \sqrt{\frac{D'_{max}}{3}} \cdot I_{SPK}$ **(Eq. 31)**		$I_{SRMS} = \sqrt{\frac{0.456}{3}} \cdot 16.31A = 6.36A$
Winding Design:		
Safety Standard Margin:	M	0 mm (use Tex/E wire if M=0)
Copper Space Factor:	f_{Cu}	0.3 (Range 0.2 … 0.4)
Eff. Bobbin Width: $BW_e = BW - (2 \cdot M)$ **(Eq. 32)**		$BW_e = 15.6mm - (2 \cdot 0mm) = 15.6mm$
Eff. Winding Cross Section: $A_{Ne} = \frac{A_N \cdot BW_e}{BW}$ **(Eq. 33)**		$A_{Ne} = \frac{61mm^2 \cdot 15.6mm}{15.6mm} = 61mm^2$
The Winding Cross Section A_N has to be subdivided according to the number of windings:		
Primary Winding	0.5	
Secondary Winding	0.45	
Auxiliary Winding	0.05	
Wire Copper Area for Primary Winding: $A_P = \frac{0.5 \cdot f_{Cu} \cdot A_{Ne}}{N_P}$ **(Eq. 34)**		$A_P = \frac{0.5 \cdot 0.3 \cdot 61mm^2}{42} = 0.22mm^2$
Wire Copper Area for Secondary Winding: $A_S = \frac{0.45 \cdot f_{Cu} \cdot A_{Ne}}{N_S}$ **(Eq. 35)**		$A_S = \frac{0.45 \cdot 0.3 \cdot 61mm^2}{6} = 1.37mm^2$
Wire Copper Area for Auxiliary Winding: $A_{Aux} = \frac{0.05 \cdot f_{Cu} \cdot A_{Ne}}{N_{Aux}}$ **(Eq. 36)**		$A_{Aux} = \frac{0.05 \cdot 0.3 \cdot 61mm^2}{5} = 0.18mm^2$
Wire Size in AWG unit can be calculated: $AWG = 9.97 \cdot (1.8277 - (2 \cdot \log(d)))$ **(Eq. 37)**		
Wire Diameter from Copper Area: $d = 2 \cdot \sqrt{\frac{A}{\pi}}$ **(Eq. 38)**		

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
Wire Diameter from AWG unit: $d = 10^{\left(\frac{1.8277}{2} - \frac{AWG}{2 \cdot 9.97}\right)}$ **(Eq. 39)**	
Wire Size in AWG unit using combination of Eq. 37 and Eq. 38: $AWG_{Pc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{A_P}{\pi}}\right)\right)\right)$	$AWG_{Pc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{0.22mm^2}{\pi}}\right)\right)\right)$ $AWG_{Pc} = 24$ diameter ≈ 0.53mm
$AWG_{Sc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{A_S}{\pi}}\right)\right)\right)$	$AWG_{Sc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{1.37mm^2}{\pi}}\right)\right)\right)$ $AWG_{Sc} = 16$ diameter ≈ 1.32mm
$AWG_{Auxc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{A_{Aux}}{\pi}}\right)\right)\right)$	$AWG_{Auxc} = 9.97 \cdot \left(1.8277 - \left(2 \cdot \log\left(2 \cdot \sqrt{\frac{0.18mm^2}{\pi}}\right)\right)\right)$ $AWG_{Auxc} = 25$ diameter ≈ 0.48mm
It is a good practice to use smaller wires in parallel instead of using one big wire. However, the following conditions should be satisfied for choosing Wire Size and Number of Parallel Wires: - $EffCuArea_X$ (Eq 40) $\leq A_X$ (Eq 34/35) - S_X (Eq 41) ≤ 8 A/mm² - $NP_X \leq 10$ - 0.18 mm $\leq$ Wire Diameter $\leq$ 0.6 mm Note: X = P/Primary or S/Secondary Winding	
Wire Size in AWG unit for Pri.: AWG_P	24
Num. of Parallel Wires for Pri.: NP_P	1
Insulation Thickness of Pri. Wire: INS_P	0.02 mm
Wire Size in AWG unit for Sec.: AWG_S	23
Num. of Parallel Wires for Sec.: NP_S	4
Insulation Thickness of Sec. Wire: INS_S	0.10 mm
Typically, the Auxiliary Winding consists of only one wire and its size is insignificant due to low current.	

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
Recalculate Wire Diameter using Eq. 39:	
$d_P = 10^{\left(\frac{1.8277}{2} - \frac{AWG_P}{2 \cdot 9.97}\right)}$	$d_P = 10^{\left(\frac{1.8277}{2} - \frac{24}{2 \cdot 9.97}\right)} = 0.51mm$
$d_S = 10^{\left(\frac{1.8277}{2} - \frac{AWG_S}{2 \cdot 9.97}\right)}$	$d_S = 10^{\left(\frac{1.8277}{2} - \frac{23}{2 \cdot 9.97}\right)} = 0.58mm$
Eff. Copper Area:	
$EffCuArea = \left(\frac{d}{2}\right)^2 \cdot \pi \cdot NP$ **(Eq. 40)**	
$EffCuArea_P = \left(\frac{d_P}{2}\right)^2 \cdot \pi \cdot NP_P$	$EffCuArea_P = \left(\frac{0.51mm}{2}\right)^2 \cdot \pi \cdot 1 = 0.21mm^2$
$EffCuArea_S = \left(\frac{d_S}{2}\right)^2 \cdot \pi \cdot NP_S$	$EffCuArea_S = \left(\frac{0.58mm}{2}\right)^2 \cdot \pi \cdot 4 = 1.04mm^2$
Current Density:	
$S = \frac{I_{RMS}}{EffCuArea}$ **(Eq. 41)**	
$S_P = \frac{I_{PRMS}}{EffCuArea_P}$	$S_P = \frac{0.98A}{0.21mm^2} = 4.7\frac{A}{mm^2}$
$S_S = \frac{I_{SRMS}}{EffCuArea_S}$	$S_S = \frac{6.36A}{1.04mm^2} = 6.1\frac{A}{mm^2}$
Wire Outer Diameter including Insulation:	
$Od = d + (2 \cdot INS)$ **(Eq. 42)**	
$Od_P = d_P + (2 \cdot INS_P)$	$Od_P = 0.51mm + (2 \cdot 0.02mm) = 0.55mm$
$Od_S = d_S + (2 \cdot INS_S)$	$Od_S = 0.58mm + (2 \cdot 0.1mm) = 0.78mm$
Max. Number of Turns per Layer:	
$NL = \left\lfloor \frac{BW_e}{Od \cdot NP} \right\rfloor$ **(Eq. 43)**	
$NL_P = \left\lfloor \frac{BW_e}{Od_P \cdot NP_P} \right\rfloor$	$NL_P = \left\lfloor \frac{15.6mm}{0.55mm \cdot 1} \right\rfloor = 28 Turns / Layer$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
$NL_S = \left\lfloor \frac{BW_e}{Od_S \cdot NP_S} \right\rfloor$	$NL_S = \left\lfloor \frac{15.6mm}{0.78mm \cdot 4} \right\rfloor = 5 Turns / Layer$
Min. Number of Layers: $Ln = \left\lceil \frac{N}{NL} \right\rceil$ **(Eq. 44)**	
$Ln_P = \left\lceil \frac{N_P}{NL_P} \right\rceil$	$Ln_P = \left\lceil \frac{42}{28} \right\rceil = 2 Layers$
$Ln_S = \left\lceil \frac{N_S}{NL_S} \right\rceil$	$Ln_S = \left\lceil \frac{6}{5} \right\rceil = 2 Layers$
Output Rectifier Diode:	
The output rectifier diodes in Flyback converters are subjected to large Peak and RMS current stress. The Values depend on the load and operating mode. The Voltage Requirements depend on the output voltage and transformer winding ratio.	
Max. Reverse Voltage: $V_{RDiode} = V_{OUT} + \left(V_{DC\max PK} \cdot \frac{N_S}{N_P} \right)$ **(Eq. 45)**	$V_{RDiode} = 16V + \left(373.4V \cdot \frac{6}{42} \right) = 69.34V$
Clamping Network:	
Clamping Voltage: $V_{Clamp} = V_{BR\_DSS} - V_{DC\max PK} - V_R$ **(Eq. 46)**	$V_{Clamp} = 650V - 373.4V - 118V = 159V$
For calculating the clamping network, it is necessary to know the leakage inductance. The most common approach is to have the Leakage Inductance value given in a percentage of the Primary Inductance. If it is known that the transformer construction is very consistent, measuring the Primary Leakage Inductance by shorting the Secondary Windings will give an exact number (assuming the availability of a good LCR analyzer)	

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Leakage Inductance Raito:	LIR	5 %
Leakage Inductance: $L_{LK} = LIR \cdot L_P$	**(Eq. 47)**	$L_{LK} = 5\% \cdot 231\mu H = 11.5\mu H$
Clamping Capacitor (cal.): $C_{Clamp\_cal} = \frac{I_{LPK\_SR}^2 \cdot L_{LK}}{(V_R + V_{Clamp}) \cdot V_{Clamp}}$	**(Eq. 48)**	$C_{Clamp\_cal} = \frac{(2.33A)^2 \cdot 11.5\mu H}{(118V + 159V) \cdot 159V} = 1.42nF$
Use table E12, find the closest higher value		
Clamping Capacitor:	C_{Clamp}	1.5 nF
Clamping Resistor (cal.): $R_{Clamp\_cal} = \frac{(V_{Clamp} + V_R)^2 - V_R^2}{0.5 \cdot L_{LK} \cdot I_{LPK\_SR}^2 \cdot f}$	**(Eq. 49)**	$R_{Clamp\_cal} = \frac{(159V + 118V)^2 - (118V)^2}{0.5 \cdot 11.5\mu H \cdot (2.33A)^2 \cdot 100kHz} = 20k\Omega$
Use table E96, find the closest value		
Clamping Resistor:	R_{Clamp}	20 kΩ
Output Capacitors:		
Output Capacitors are highly stressed in Flyback converters. Normally, capacitors are chosen based on **3 major parameters: Capacitance, Low ESR and Ripple Current Rating.**		
To calculate Output Capacitors, the Max. Voltage Overshoot in case of switching off at Max. load condition must be set.		
Max. Voltage Overshoot:	ΔV_{OUT}	0.5 V
After switching off the load, the control loop needs about 10…20 internal clock periods to reduce the duty cycle.		
Number of Clock Periods:	n_{CP}	20
Max. Output Current: $I_{OUT\,max} = \frac{P_{OUT\,max}}{V_{OUT}}$	**(Eq. 50)**	$I_{OUT\,max} = \frac{50W}{16V} = 3.13A$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Ripple Current: $I_{Ripple} = \sqrt{I_{SRMS}^2 - I_{OUT\max}^2}$ **(Eq. 51)**		$I_{Ripple} = \sqrt{(6.36A)^2 - (3.13A)^2} = 5.54A$
Output Capacitance (cal.): $C_{OUT\_cal} = \frac{I_{OUT\max} \cdot n_{CP}}{\Delta V_{OUT} \cdot f}$ **(Eq. 52)**		$C_{OUT\_cal} = \frac{3.13A \cdot 20}{0.5V \cdot 100kHz} = 1252\mu F$
We propose the Rubycon ZL Series **Aluminum Electrolytic Capacitors** for Output Capacitors due to its high ripple current and low impedance. Since the Ripple Current that each capacitor can handle is quite low comparing to the calculated Ripple Current (Eq. 51), parallel capacitors are necessary. However, regarding space, the number of parallel capacitors should not exceed three. The conditions can be summarized as the following: - Rated Voltage of Cap. ⩾ (1.45·V_{OUT}) - (I_{acR}·nc) close to I_{Ripple} - (C_{OUT}·nc) close to C_{OUT_cal} - nc ⩽ 3		We choose Rubycon ZL Series 2 x 1500µF 25V (R_{ESR}=0.018Ω, I_{acR}=2.77A)
Output Capacitor: Number of Parallel Capacitors:	C_{OUT} nc	1500 µF 2
Output Filter: The Output Filter consists of one Capacitor and one Inductor in a L-C filter topology.		
Zero Frequency of Output Capacitors and associated ESR: $f_{ZCOUT} = \frac{1}{2 \cdot \pi \cdot R_{ESR} \cdot C_{OUT}}$ **(Eq. 53)**		$f_{ZCOUT} = \frac{1}{2 \cdot \pi \cdot 0.018\Omega \cdot 1500\mu F} = 5.89kHz$
This equation is based on the assumption that all output capacitors have the same capacitance and ESR. Ripple Voltage at 1st Stage: $V_{Ripple1} = \frac{I_{SPK} \cdot R_{ESR}}{nc}$ **(Eq. 54)**		$V_{Ripple1} = \frac{16.31A \cdot 0.018\Omega}{2} = 0.15V$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
The Inductance is required to compensate the Zero Frequency caused by output capacitors:		
L-C Filter Inductance:	L_{OUT}	1 µH
L-C Capacitor (cal.): $C_{LC\_cal} = \frac{(C_{OUT} \cdot R_{ESR})^2}{L_{OUT}}$	**(Eq. 55)**	$C_{LC\_cal} = \frac{(1500\mu F \cdot 0.018\Omega)^2}{1\mu H} = 729\mu F$
We propose the Rubycon ZL Series Aluminum Electrolytic Capacitors for L-C Filter Capacitors due to its high ripple current and low impedance. The conditions can be summarized as the following: - Rated Voltage of Cap. ⩾ (1.45·V_{OUT}) - Capacitance close to C_{LC_cal}		We choose Rubycon ZL Series 680µF 25V
L-C Capacitor:	C_{LC}	180 µF
Frequency of L-C Filter: $f_{LC} = \frac{1}{2 \cdot \pi \cdot \sqrt{C_{LC} \cdot L_{OUT}}}$	**(Eq. 56)**	$f_{LC} = \frac{1}{2 \cdot \pi \cdot \sqrt{680\mu F \cdot 1\mu H}} = 6.1kHz$
Ripple Voltage at 2nd Stage: $V_{Ripple2} = V_{Ripple1} \cdot \frac{\frac{1}{2 \cdot \pi \cdot f \cdot C_{LC}}}{\frac{1}{2 \cdot \pi \cdot f \cdot C_{LC}} + (2 \cdot \pi \cdot f \cdot L_{OUT})}$	**(Eq. 57)**	$V_{Ripple2} = 0.15V \cdot \frac{\frac{1}{2 \cdot \pi \cdot 100kHz \cdot 680\mu F}}{\frac{1}{2 \cdot \pi \cdot 100kHz \cdot 680\mu F} + (2 \cdot \pi \cdot 100kHz \cdot 1\mu H)} = 1mV$
Soft-Start Capacitor:		
The voltage at the soft-start pin together with feedback voltage control the overvoltage, open loop and overcurrent protection functions.		
The Soft-Start Capacitor must be calculated in such a way that the output voltage and the feedback voltage is within the working range(V_{FB} ⩽ 4.8V) before the overcurrent threshold (typ. 5.3V) is reached.		
Soft-Start Time (cal.): $t_{SS\_cal} = (V_{OUT})^2 \cdot \frac{(nc \cdot C_{OUT}) + C_{LC}}{P_{OUTmax\_SR} - P_{OUTnom}}$	**(Eq. 58)**	$t_{SS\_cal} = (16V)^2 \cdot \frac{(2 \cdot 1500\mu F) + 680\mu F}{53W - 40W} = 72.5ms$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Soft-Start Time:	t_{SS}	73 ms
Soft-Start Capacitor (cal.): $C_{SS\_cal} = t_{SS} \cdot \frac{1}{-R_{SS} \cdot \ln\left(1-\frac{V_{SS1}}{V_{REF}}\right)}$ **(Eq. 59)**		$C_{SS\_cal} = 73ms \cdot \frac{1}{-42k\Omega \cdot \ln\left(1-\frac{5.15V}{6.5V}\right)} = 1106nF$
Use table E12, find the closet higher value		
Soft-Start Capacitor:	C_{SS}	1000 nF
VCC Capacitors:		
The VCC Capacitor needs to ensure the power supply of the IC until the power can be provided by the Auxiliary Winding. In addition, it is recommended to use a 100nF Ceramic Capacitor very close between pin 7 & 8 in parallel to the VCC Capacitor. Alternatively, an HF-type electrolytic with low ESR and ESL may be used.		
VCC Capacitor (cal.): $C_{VCC\_cal} = \frac{I_{VCC3} \cdot t_{SS}}{V_{CCHY}} \cdot \frac{2}{3}$ **(Eq. 60)**		$C_{VCC\_cal} = \frac{9.8mA \cdot 73ms}{5V} \cdot \frac{2}{3} = 95.4\mu F$
Use table E12, find the closet higher value		
VCC Capacitor:	C_{VCC}	100 µF
Start-up Resistors:		
VCC Cap. Charge-up Current:	$I_{ChargeC}$	70 µA
Start-up Resistance (cal.): $R_{Start\_cal} = \frac{V_{DC\min}}{I_{VCC1} + I_{ChargeC}}$ **(Eq. 61)**		$R_{Start\_cal} = \frac{100.5V}{55\mu A + 70\mu A} = 804k\Omega$
It is better to use two resistors in series to accomplish the start-up resistance. Use table E96, find the closest value		
Start-up Resistors (x2):	R_{Start}	402 kΩ

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Start-up Time: $t_{Start} = \frac{C_{VCC} \cdot V_{CCon}}{I_{Charge C}}$ **(Eq. 62)**		$t_{Start} = \frac{100\mu F \cdot 13.5V}{70\mu A} = 19.3s$
Note: **Before the IC can be plugged into the application board, the VCC Capacitor must always be <u>discharged!</u>**		
Losses:		
Diode Bride Forward Voltage:	V_F	1 V
Input Diode Bridge Loss: $P_{DIN} = 2 \cdot I_{ACRMS} \cdot V_F$ **(Eq. 63)**		$P_{DIN} = 2 \cdot 1.09A \cdot 1V = 2.18W$
Form Factor: Hysteresis Factor Single Ended:	α ß	0.8 0.33
Transformer Core Loss: $P_{Core} = P_V \cdot \left(\frac{f}{100kHz}\right)^{1.5} \cdot \left(\frac{B_{max}}{200mT}\right)^{2.3} \cdot \alpha \cdot \beta$ **(Eq. 64)**		$P_{Core} = 1.6W \cdot \left(\frac{100kHz}{100kHz}\right)^{1.5} \cdot \left(\frac{244mT}{200mT}\right)^{2.3} \cdot 0.8 \cdot 0.33 = 667mW$
Copper Resistivity @ 100°C:	p_{100}	0.0172 Ω mm²/m
Copper Resistance: $R_{Cu} = \frac{l_N \cdot N \cdot p_{100}}{EffCuArea}$ **(Eq. 65)**		
$R_{PCu} = \frac{l_N \cdot N_P \cdot p_{100}}{EffCuArea_P}$		$R_{PCu} = \frac{50mm \cdot 42 \cdot 0.0172 \frac{\Omega \cdot mm^2}{m}}{0.21mm^2} = 172m\Omega$
$R_{SCu} = \frac{l_N \cdot N_S \cdot p_{100}}{EffCuArea_S}$		$R_{SCu} = \frac{50mm \cdot 6 \cdot 0.0172 \frac{\Omega \cdot mm^2}{m}}{1.04mm^2} = 4.95m\Omega$
Copper Resistance Loss on Primary Side: $P_{PCu} = I_{PRMS}^2 \cdot R_{PCu}$ **(Eq. 66)**		$P_{PCu} = (0.98A)^2 \cdot 172m\Omega = 165mW$
Copper Resistance Loss on Secondary Side: $P_{SCu} = I_{SRMS}^2 \cdot R_{SCu}$ **(Eq. 67)**		$P_{SCu} = (6.36A)^2 \cdot 4.95m\Omega = 200mW$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
Total Copper Resistance Loss: $P_{Cu} = P_{PCu} + P_{SCu}$ **(Eq. 68)**		$P_{Cu} = 165mW + 200mW = 365mW$
Output Rectifier Diode Loss: $P_{DDIODE} = I_{SRMS} \cdot V_{FDIODE}$ **(Eq. 69)**		$P_{DDIODE} = 6.36A \cdot 0.85V = 5.41W$
Clamping Network Loss: $P_{Clamp} = \frac{1}{2} \cdot L_{LK} \cdot I_{LPK\_SR}^2 \cdot f \cdot \frac{V_{Clamp} + V_R}{V_{Clamp}}$ **(Eq. 70)**		$P_{Clamp} = \frac{1}{2} \cdot 11.5\mu H \cdot (2.33A)^2 \cdot 100kHz \cdot \frac{159V + 118V}{159V} = 5.44W$
Estimated Ext. Parasitic Cap.:	C_{ext}	48 pF
Junction Temperature:	T_j	125 °C
On-Resistance at Junction Temperature: $R_{DSon@T_j} = R_{DSon@25°C} \cdot \left(1 + \frac{\alpha}{100}\right)^{(T_j - 25°C)}$ **(Eq. 71)**		$R_{DSon@T_j} = 0.45\Omega \cdot \left(1 + \frac{0.8}{100}\right)^{(125°C - 25°C)} = 1\Omega$
MOSFET Losses in V_{ACmin} scenario:		
Switching Loss in V_{ACmin} scenario: $P_{SON1} = \frac{1}{2} \cdot \left(C_{o(er)} + C_{ext}\right) \cdot V_{DC\min}^2 \cdot f$ **(Eq. 72)**		$P_{SON1} = \frac{1}{2} \cdot (30pF + 48pF) \cdot (100.5V)^2 \cdot 100kHz = 39mW$
Conduction Loss in V_{ACmin} scenario: $P_{D1} = I_{PRMS}^2 \cdot R_{DSon@T_j}$ **(Eq. 73)**		$P_{D1} = (0.98A)^2 \cdot 1\Omega = 960mW$
Total MOSFET Loss in V_{ACmin} scenario: $P_{MOSFET1} = P_{SON1} + P_{D1}$ **(Eq. 74)**		$P_{MOSFET1} = 39mW + 960mW = 999mW$
MOSFET Losses in V_{ACmax} scenario:		
Switching Loss in V_{ACmax} scenario: $P_{SON2} = \frac{1}{2} \cdot \left(C_{o(er)} + C_{ext}\right) \cdot V_{DC\max PK}^2 \cdot f$ **(Eq. 75)**		$P_{SON2} = \frac{1}{2} \cdot (30pF + 48pF) \cdot (373.4V)^2 \cdot 100kHz = 544mW$
Conduction Loss in V_{ACmax} scenario: $P_{D2} = \frac{1}{3} \cdot R_{DSon@T_j} \cdot I_{LPK\_SR}^2 \cdot \left(\frac{L_P \cdot I_{LPK\_SR} \cdot f}{V_{DC\max PK}}\right)$ **(Eq. 76)**		$P_{D2} = \frac{1}{3} \cdot 1\Omega \cdot (2.33A)^2 \cdot \left(\frac{231\mu H \cdot 2.33A \cdot 100kHz}{373.4V}\right) = 259mW$
Total MOSFET Loss in V_{ACmax} scenario: $P_{MOSFET2} = P_{SON2} + P_{D2}$ **(Eq. 77)**		$P_{MOSFET2} = 544mW + 259mW = 803mW$
MOSFET Losses: $P_{MOSFET} = \max(P_{MOSFET1}, P_{MOSFET2})$ **(Eq. 78)**		$P_{MOSFET} = \max(999mW, 803mW) = 999mW$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
Heat Dissipater:	
All CoolSETs in DSO/DIP/SMD package cannot use a Heat Sink but the Copper Area is possible. Typically, Copper Area should not exceed 1cm². However, all CoolSETs in TO packages as well as external CoolMOS can use a Heat Sink.	Since ICE2A365 is a DIP-8 package, we choose a standard 1cm² Copper Area ($R_{thCuArea} \approx 10K/W$)
Thermal Resistance:	
In case NO Heat Sink $R_{th} = R_{thJA} - R_{thCuArea}$ **(Eq. 79a)**	$R_{th} = 90K/W - 10K/W = 80K/W$
In case WITH Heat Sink $R_{th} = R_{thJC} + R_{thHT} + R_{thHS}$ **(Eq. 79b)**	
where R_{thJC} : TR. Junction-Case R_{thHT} : TR. Chip-Heat Sink R_{thHS} : TR. Heat Sink-Ambient	From CoolSET/CoolMOS Datasheet Typ. 1K/W Depending on Heat Sink
Delta Temperature from MOSFET Losses: $\Delta T = P_{MOSFET} \cdot R_{th}$ **(Eq. 80)**	$\Delta T = 999mW \cdot 80K/W = 80K$
Max. Junction Temperature: $T_{j\max} = T_a + \Delta T$ **(Eq. 81)**	$T_{j\max} = 50°C + 80K = 130°C$
Max. Junction Temperature must not exceed the limitation stated in the Datasheet, typically 150°C.	
Controller Loss: $P_{Controller} = V_{Aux\_cal} \cdot I_{VCC3}$ **(Eq. 82)**	$P_{Controller} = 13.2V \cdot 9.8mA = 129mW$
Total Loss: $P_{Losses} = P_{DIN} + P_{Core} + P_{Cu} + P_{DDIODE} + P_{Clamp} + P_{MOSFET} + P_{Controller}$ **(Eq. 83)**	$P_{Losses} = 2.18W + 667mW + 365mW + 5.41W + 5.44W + 999mW + 129mW = 15.19W$
Efficiency after Losses Consideration: $\eta_L = \frac{P_{OUT\max\_SR}}{P_{OUT\max\_SR} + P_{Losses}}$ **(Eq. 84)**	$\eta_L = \frac{53W}{53W + 15.19W} = 77.72\%$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

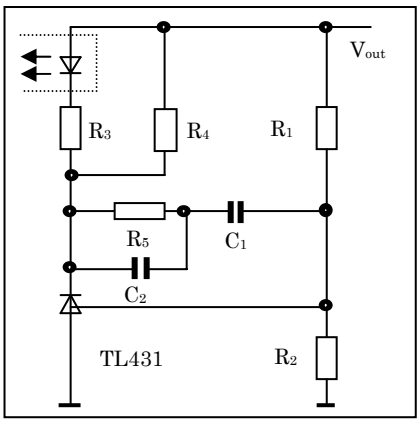
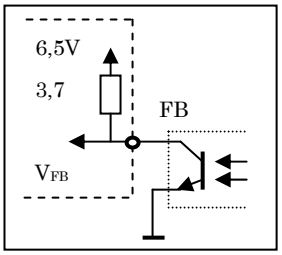
Procedure		Example
Regulation Loop:		
Reference: TL431 Optocoupler: SFH617-3		
		
TL431 Reference Voltage:	V_{REF_TL}	2.5 V
Forward Voltage of Diode:	V_{FD_TL}	1.2 V
Min. Current for TL431 Diode:	I_{KAmin}	1 mA
Max. Current of SFH617-3 Diode:	I_{Fmax}	20 mA
R_2 Value of Voltage Divider:	R_2	4.75 kΩ
Primary Side:		
Max. Feedback Current: $I_{FB\max} = \frac{V_{REF}}{R_{FB}}$	**(Eq. 85)**	$I_{FB\max} = \frac{6.5V}{3.7k\Omega} = 1.76mA$
Min. Feedback Current: $I_{FB\min} = \frac{V_{REF} - V_{FB\max}}{R_{FB}}$	**(Eq. 86)**	$I_{FB\min} = \frac{6.5V - 4.6V}{3.7k\Omega} = 0.51mA$
Secondary Side:		
R_1 Value of Voltage Divider (cal.): $R_{1\_cal} = R_2 \cdot \left(\frac{V_{OUT}}{V_{REF\_TL}} - 1 \right)$	**(Eq. 87)**	$R_{1\_cal} = 4.75k\Omega \cdot \left(\frac{16V}{2.5V} - 1 \right) = 25.65k\Omega$
Use table E96, find the closest value		
R_1 Value of Voltage Divider:	R_1	25.5 kΩ
R_3 Value to supply Opto. Diode (cal.): $R_{3\_cal} \geq \frac{V_{OUT} - \left(V_{FD\_TL} + V_{REF\_TL}\right)}{I_{F\max}}$	**(Eq. 88)**	$R_{3\_cal} \geq \frac{16V - \left(1.2V + 2.5V\right)}{20mA} = 615\Omega$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

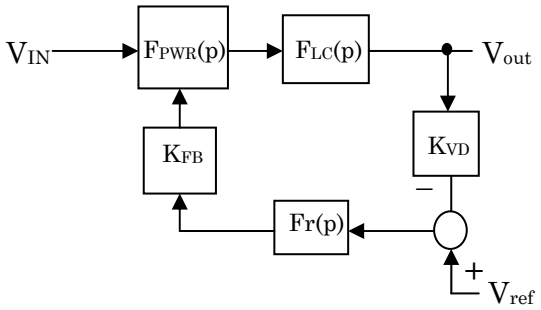
Procedure	Example
Use table E96, find the closest higher value	
R_3 Value to supply Opto. Diode: R_3	619 Ω
R_4 Value to supply TL431 Diode (cal.):	
$R_{4\_cal} \le \dfrac{V_{FD\_TL} + \left(R_3 \cdot \dfrac{I_{FB\min}}{G_c} \right)}{I_{KA\min}}$ **(Eq. 89)**	$R_{4\_cal} \le \dfrac{1.2V + \left(619\Omega \cdot \dfrac{0.51mA}{100\%} \right)}{1mA} = 1.51k\Omega$
Use table E96, find the closest lower value	
R_4 Value to supply TL431 Diode: R4	1.50 kΩ
Output Voltage from Regulation Loop:	
$V_{OUT\_RL} = \left(\dfrac{R_1}{R_2} + 1 \right) \cdot V_{REF\_TL}$ **(Eq. 90)**	$V_{OUT\_RL} = \left(\dfrac{25.5k\Omega}{4.75k\Omega} + 1 \right) \cdot 2.5V = 15.92V$
Regulation Loop Elements:	
	
Feedback Transfer Characteristic:	
$K_{FB} = \dfrac{G_c \cdot R_{FB}}{R_3}$ **(Eq. 91)**	$K_{FB} = \dfrac{100\% \cdot 3.7k\Omega}{619\Omega} = 5.98$
Gain of Feedback Transfer Characteristic:	
$G_{FB} = 20 \cdot \log(K_{FB})$ **(Eq. 92)**	$G_{FB} = 20 \cdot \log(5.98) = 15.53dB$
Voltage Divider Transfer Characteristic:	
$K_{VD} = \dfrac{V_{REF\_TL}}{V_{OUT\_RL}} = \dfrac{R_2}{R_1 + R_2}$ **(Eq. 93)**	$K_{VD} = \dfrac{4.75k\Omega}{25.5k\Omega + 4.75k\Omega} = 0.16$

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
Gain of Voltage Divider Transfer Characteristic: $G_{VD} = 20 \cdot \log(K_{VD})$ **(Eq. 94)**	$G_{VD} = 20 \cdot \log(0.16) = -15.92dB$
Zeroes and Poles of Transfer Characteristics:	
Resistance at Max. Load Pole: $R_{LH} = \frac{V_{OUT\_RL}^2}{P_{OUT\max\_SR}}$ **(Eq. 95)**	$R_{LH} = \frac{(15.92V)^2}{53W} = 4.78\Omega$
Resistance at Min. Load Pole: $R_{LL} = \frac{V_{OUT\_RL}^2}{P_{OUT\min}}$ **(Eq. 96)**	$R_{LL} = \frac{(15.92V)^2}{0.5W} = 506.89\Omega$
Poles of Power stage at Max. Load Pole: $f_{OH} = \frac{1}{\pi \cdot R_{LH} \cdot (nc \cdot C_{OUT})}$ **(Eq. 97)**	$f_{OH} = \frac{1}{\pi \cdot 4.78\Omega \cdot (2 \cdot 1500\mu F)} = 22.20Hz$
Poles of Power stage at Min. Load Pole: $f_{OL} = \frac{1}{\pi \cdot R_{LL} \cdot (nc \cdot C_{OUT})}$ **(Eq. 98)**	$f_{OL} = \frac{1}{\pi \cdot 506.89\Omega \cdot (2 \cdot 1500\mu F)} = 0.21Hz$
In order to have sufficient phase margin at low load condition, we choose the Zero Frequency of the compensation network to be at the middle between the min. and max. load poles of the power stage.	
Zero Frequency of the Compensation Net.: $f_{OM} = f_{OH} \cdot 10^{0.5 \cdot \log\left(\frac{f_{OL}}{f_{OH}}\right)}$ **(Eq. 99)**	$f_{OM} = 22.20Hz \cdot 10^{0.5 \cdot \log\left(\frac{0.21Hz}{22.20Hz}\right)} = 2.16Hz$
With adjustment of the transfer characteristics of the regulator, we want to reach equal gain within the operating range and to compensate the pole f_o of the power stage $F_{PWR}(\omega)$	
Because of the compensation of the output capacitor's zero (Eq 53), we neglect it as well as the LC-Filter pole (Eq 56). Consequently, the transfer characteristic of the power stage is reduced to a single-pole response.	

Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure	Example
In order to calculate the gain of the open loop, we have to choose the crossover frequency. We calculate the gain of the Power stage with Max. Output Power at the chosen crossover frequency.	
Zero dB Crossover Frequency: f_g	3 kHz
Transient Impedance Calculation: Transient Impedance defines the direct relationship between the level of the Peak Current and the Feedback pin voltage. It is required for the calculation of the power stage amplification. Transient Impedance: $Z_{PWM} = \frac{\Delta V_{FB}}{\Delta I_{PK}} = A_V \cdot \frac{R_{Sense}}{V_{csth}}$ **(Eq. 100)**	$Z_{PWM} = 3.65 \cdot \frac{430m\Omega}{1V} = 1.57\frac{V}{A}$
Power stage at Crossover Frequency: $\lvert F_{PWR}(f_g)\rvert = \frac{1}{Z_{PWM}} \cdot \sqrt{\frac{R_{LH} \cdot L_p \cdot f \cdot \eta_P}{2}} \cdot \left(\frac{1}{\sqrt{1+\left(\frac{f_g}{f_{OH}}\right)^2}} \right)$ **(Eq. 101)**	$\lvert F_{PWR}(f_g)\rvert = \frac{1}{1.57\frac{V}{A}} \cdot \sqrt{\frac{4.78\Omega \cdot 231\mu H \cdot 100kHz \cdot 85\%}{2}} \cdot \left(\frac{1}{\sqrt{1+\left(\frac{3kHz}{22.2Hz}\right)^2}} \right) = 0.03$
Gain of Power stage at Crossover Frequency: $G_{PWR}(f_g) = 20 \cdot \log\left(\lvert F_{PWR}(f_g)\rvert\right)$ **(Eq. 102)**	$G_{PWR}(3kHz) = 20 \cdot \log(0.03) = -30.46dB$
Transfer Characteristics:	

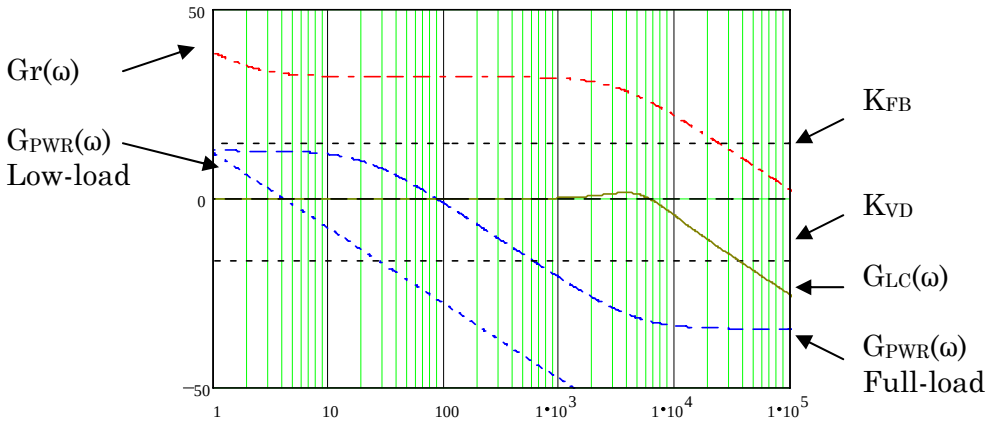


Table B.1: Design procedure for Flyback DCM using CoolSET-F2 (cont.)

Procedure		Example
At the Crossover Frequency (f_g), we calculate the Open Loop Gain: $G_{OL}(\omega) = Gs(\omega) + Gr(\omega) = 0$ **(Eq. 103)**		
With the equations for the transfer characteristics, we calculate the gain of the regulation loop at f_g: $Gs(\omega) = G_{FB} + G_{PWR} + G_{VD}$ **(Eq. 104)**		$Gs(\omega) = 15.53dB - 30.46dB - 15.92dB = -30.85dB$
Separated components of the regulator: $Gr(\omega) = 0 - Gs(\omega)$ **(Eq. 105)**		$Gr(\omega) = 0 - (-30.85dB) = 30.85dB$
R_5 Value of Compensation Net. (cal.): $R_{5\_cal} = 10^{\frac{Gr}{20}} \cdot \frac{R_1 \cdot R_2}{R_1 + R_2}$ **(Eq. 106)**		$R_{5\_cal} = 10^{\frac{30.85dB}{20}} \cdot \frac{25.5k\Omega \cdot 4.75k\Omega}{25.5k\Omega + 4.75k\Omega} = 139.64k\Omega$
Use table E96, find the closest value		
R_5 Value of Compensation Net.:	R_5	140 kΩ
C_2 Value of Compensation Net. (cal.): $C_{2\_cal} = \frac{1}{2 \cdot \pi \cdot R_5 \cdot f_g}$ **(Eq. 107)**		$C_{2\_cal} = \frac{1}{2 \cdot \pi \cdot 140k\Omega \cdot 3kHz} = 379\,pF$
Use table E12, find the closest higher value		
C_2 Value of Compensation Net.:	C_2	390 pF
C_1 Value of Compensation Net. (cal.): $C_{1\_cal} = \frac{1}{2 \cdot \pi \cdot R_5 \cdot f_{OM}} - C_2$ **(Eq. 108)**		$C_{1\_cal} = \frac{1}{2 \cdot \pi \cdot 140k\Omega \cdot 2.16Hz} - 390\,pF = 525.92nF$
Use table E12, find the closest higher value		
C_1 Value of Compensation Net.:	C_1	560 nF