Component-Based System for Designing Power Supply of no Expert Knowledge Required

Heidi H. T. Yeung N. K. Poon Joe C. P. Liu
PowerELab Limited
1/F Tech. Innovation & Incubation Bldg., Hong Kong University, Pokfulam Rd., Hong Kong
Email: heidi@powerelab.com , nkpoon@eee.hku.hk , cpliu@eee.hku.hk

Abstract – This paper introduces the Component based architecture (CBA) employed in a software to design the switching mode power supply (SMPS). Theoretically, there is a unique mapping between a real converter and its behavior. Based on a known circuit topology, the proper design is underlining in the combination of components and its physical configuration. Neither abstract electrical model nor domain knowledge is expected from the user. Besides, this architecture turns optimizing the abstract parameters into component selection.

I. INTRODUCTION

Generally, the technical works are divided into 2 types:

A) Technological Research
– Doing things wrong before becoming right
B) Product Development
– Doing things right first time and all the time

The tools for Technological Research are generic and subtle to handle all configurations and tolerate any fault during the experiments. The results from the tools maybe abstract and no solution is guaranteed. For developing a product, at least one solution is expected from the tools for a particular specification. The solution from the tools must be specific, focused and leaves no room for uncertainty.

Many SMPS companies resort to use Technological Research tools for Product Development. Much expert knowledge and effort is required on bundling up a pool of solutions. Unfortunately, the inexperienced engineers often fail to use the tools correctly. It leads that the varied design quality with individual engineer's judgment. Consequently, the experience earned within a company is difficult to accumulate to improve the design.

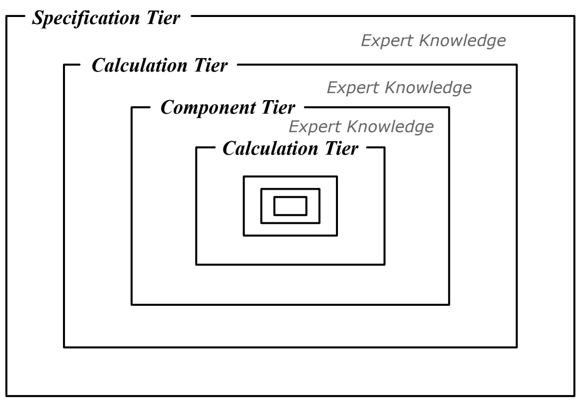
Some software is developed to standardize the design procedures of power supply. This kind of methods is rather expensive not only from the license fee, the working hours engineer spent to understand the theories behind and get used to the domain driven procedures also laid down. In short, most conventional software cannot meet the stringent demand.

This paper introduces a design methodology to produce good design without expert knowledge from user. A new concept is developed and employed in SMPS design software.

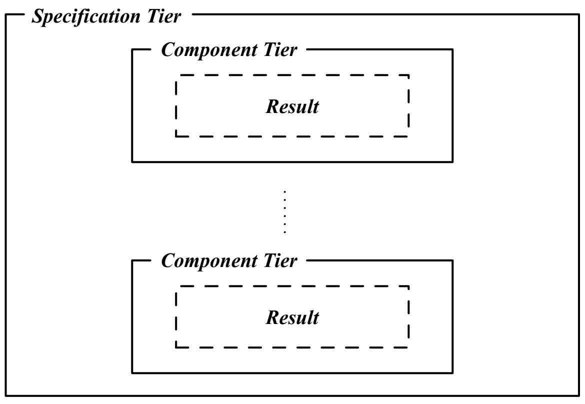
II. CONCEPTS

A. *Recursive designing procedures*

Fig. 1a depicts the design procedures used by most engineers in SMPS design. The first *Specification Tier* represents the conversion of customers' specification to engineering design parameters. Expert knowledge is required for choosing a circuit or known topology. The next layer is the *Calculation Tier* that computes and estimates all corresponding values to be used in the circuit at a subjective base. For example, winding ratio, inductance etc. are acquired from expert knowledge, rules from engineering cookbook or working tradition.



(a) Recursive designing procedures



(b) Parallel designing procedure

Fig. 1: Illustration map of designing SMPS

The reasons for subjective choice in a rational circuit design are caused by the *non-homogeneous system* of the equations found in the specification. The information provided is always insufficient to solve those equations. Engineer needs to add subjectively more relations to ensure number of variables in the circuit equal to the relations. This subjective addition is called the experienced design.

In *Component Tier*, the most appropriate components are chosen from those design values. It is not always possible to find the perfect set of components matching the calculated values. Therefore, the engineers use their expert knowledge to choose the "right" components. In some cases, the deviation between the chosen components and the expected values such as the leakage inductance, *TR*, or *ESR* etc are too large. It leads to next Calculation Tier in order to provide more hints for next Component Tier. Such iterative processes might converge if the engineers are smart and lucky enough. Otherwise, they may stick with these two tiers for a long time in the diverging situation and never commit to the results.

The root cause for the second situation is the involvement of expert knowledge from each individual in each design and decision step. Synthetic mistake and the capability of engineers affect the entire production of the SMPS design directly and critically.

To prevent synthetic mistake in calculation, computer simulator is often employed. However, it might not be quite helpful when the engineer is not capable to give the correct input arguments to the simulator. The possibility of having diverging situation does not reduce much.

B. Parallel designing procedures

A better design concept [1] is introduced to a new SMPS design tool. It serializes the design procedures from the recursive iterations and avoids expert knowledge from user while inputting parameters. From Fig. 1b, there is no Calculation Tier in such software architecture and the Component Tiers are in parallel.

Starting from Specification Tier, the first and most important input from the user is the detail specification such as the known circuit topology, input voltage range, expected output voltage, output current etc. In the Component Tier, the user has to select the modeled real components into the circuit. The initial set of components is generated according to the specification and some simple rules. It may not be the best design but at least one solution is provided. The result, likes the rating of the design, is estimated by deriving the chosen components for a particular circuit. As all variables in the circuit are known, the corresponding relationships can be described by the circuit theory. No expert knowledge is required to solve the set of relations.

The process will continue until the best result from these parallel blocks or component combinations is found. Hence, the convergence of design is guaranteed. The expert knowledge is needed once only to convert the discrete components to results while designing the software. The user may not be an expert in the field but more knowledge from the user helps increasing the speed and accuracy while using this design tool.

To improve the SMPS design, we need to maximize the efficiency. The best solution is searched from a set of parallel blocks determined by a cost function. System optimization is a well-known process. No matter the search method is, e.g. Full Search or Genetic Algorithm [3], it requires no expert knowledge from the user.

As a result, the proposed approach is suitable to be implemented to computer software. It should provide the user interface of selecting the circuits and their corresponding components. An optimized result according to engineering requirement is found out after pressing the optimizing button.

III. From Analogy To Discrete

As mentioned in Section II, the proposed design employs a parallel structure to eliminate the recursive process between component selection and calculation. To facilitate this structure, a new set of object functions with the component as the input arguments is illustrated in Fig. 2a. These functions embrace all the abstract equations under the new structure.

The conversion from the abstract continuous model to a discrete approach is shown in Fig. 2b. The result of this particular configuration is described by an object function with discrete component type as the input arguments.

In fact, all components in a SMPS can be regarded as object functions. The outermost object function is understood as the entire power converter itself. The innermost object function is a single electrical part in the converter. We call this approach Component Based Architecture (CBA) [2].

Once the conversion is done, the total number of input arguments in discrete Component Based Architecture is

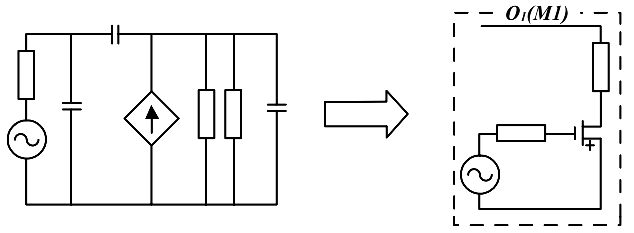
$$f_0\left(L_m, L_k, T_j\right) \rightarrow O_0\left(XF\right)$$

$$\left.\begin{cases} f_1\left(R_{ds}, T_j\right) \\ f_2\left(t_r, t_f, V\right) \end{cases}\right\} \rightarrow O_1\left(M1, Rg_{M1}\right)$$

$$\vdots$$

$$f_n\left(V_f, T_j\right) \rightarrow O_m\left(Do1\right)$$

(a) Turning continuous functions to discrete object functions



(b) Turning continuous model to real component

Fig. 2: Transforming abstract continuous model to real discrete object

smaller than that in the abstract continuous model. Thus, the complexity of solving an optimized solution will be greatly reduced. This approach shortens time required to find the optimized solution and it does not cause infinite loop even though no solution from the specification is obtained from the abstract continuous model approach.

We have so far discussed how to select the component to form the solution. Besides, there are another important attributes in this CBA. It is the configuration.

In general, the physical configuration of a power converter is known namely, 1) Circuit, 2) Component placement, 3) PCB layout. Once these three items are defined, the corresponding result or performance of the SMPS can be estimated accordantly shown as Fig. 3. The best solution must be in one of the combinations.

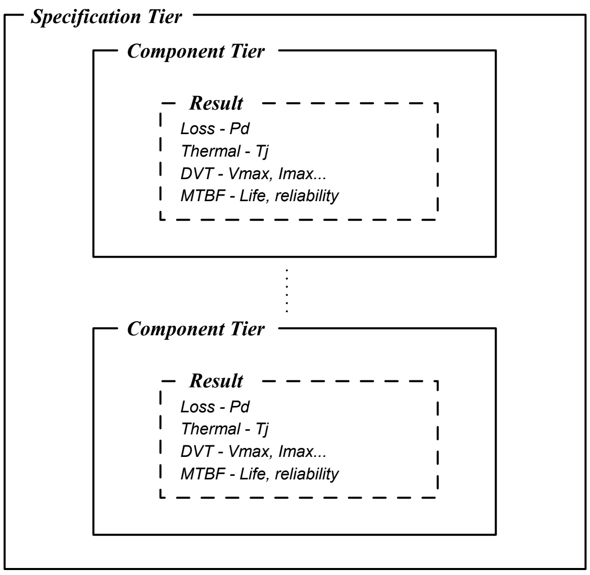


Fig. 3: Detail result of using CBA approach

With a set of object functions and configuration information, we can find out all the necessary information for the combination of real component and real configuration. No pre-determined assumption is needed.

For example, someone may assume the losses in order to predict the junction temperature at each component. In fact, the CBA structure eventually produces the losses and junction temperature at the same time. All results are generated by their own combination of component and configuration. The result is not intended as other initial criteria for another calculation for changing the design.

IV. Real Component and Configuration

In order to implement the CBA software, we have to overcome the first barrier of defining each real component, e.g. resistor, capacitor, MOSFET, diode, etc, into the object function. It is not difficult for most components. The software provides interface to input the raw data of components from specification. All values of the equivalent model are calculated from the raw data with corresponding object functions. No interface for inputting these values directly is allowed. Nonetheless, there is another challenge from the custom-made components such as transformers.

A second barrier is the configuration issue in SMPS. After a circuit design is chosen, only half of the job is done. For instance, the PCB assembly affects the final performance significantly especially the thermal performance. The traditional recursive iteration method treats the junction temperature of a component as an input argument. Besides choosing a proper component from Component Tier, the external factors like attachment of heat sink and component placement influent the final junction temperature. It is a paradox that we need the junction temperature to calculate the losses, but the loss itself is greatly dependent on the temperature. It leads deadlock dependence.

To allow the engineers making their own custom component, e.g. transformer, and do the component placement on the PCB, The user interface is needed. However, most concurrent tools or user interface does not exactly use CBA approach. For example, some engineers spend a lot of time to construct a transformer winding method by using some magnetic design tool. He looks for a very low winding resistance transformer under the blueprint but too much leakage is resulted finally or vice versa. Design and redesign is still unavoidable.

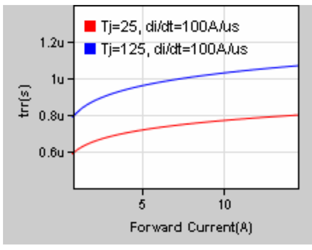
In short, the proper user interface is a necessary condition for CBA approach. Installing several tools into a computer and linking them together do not mean CBA. Our belief is to utilize the concept of CBA in developing software.

The software is built according to the CBA philosophy to prove the approach.

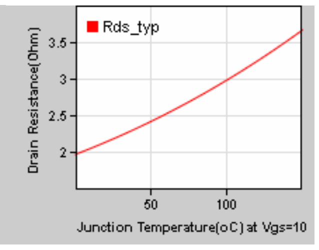
V. From Concept to Implementation

A. Building standard component

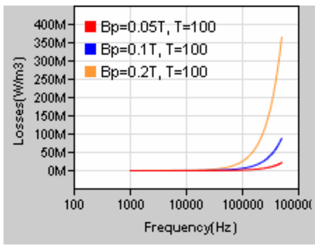
The requiring data of the components are inputted through



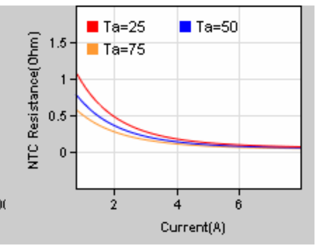
(a) *trr* VS *di/dt* of a rectifier



(b) *Rds* VS *Ti* of a MOSFET



(c) *Loss* VS *freq* of a ferrite material



(d) *R* VS *I* of a power NTC

Fig. 4: Implementing component and non-linear characteristic

the user interface. The characteristics are retrieved and modeled with object functions. In general, the methods inside the object functions can be a close form relationship, iterative relationship or some finite element analysis result. The non-linear behaviors of all basic components are properly described in terms of the relationship against voltage, current, frequency and temperature.

Fig. 4a is the non-linear characteristic of the reverse recovery time of a rectifier. Fig. 4b illustrates the non-linear characteristic of the channel resistance of a MOSFET against junction temperature. Figure 4c depicts core losses of a magnetic material against frequency. Fig. 4d shows the resistance of a power NTC Thermistor against root-mean-square (*rms*) current.

Other physical parameters such as dimensions are also captured for thermal simulation. These properties are ready to use in database once the user chooses particular component.

B. *Building custom components*

Transformer is a typical custom component. It varies widely and takes a very important role in affecting the performance of a SMPS. Quite a lot of studies and ways have been done to describe the behaviors of a multi-winding and multi-layer transformer. Nevertheless, the CBA description

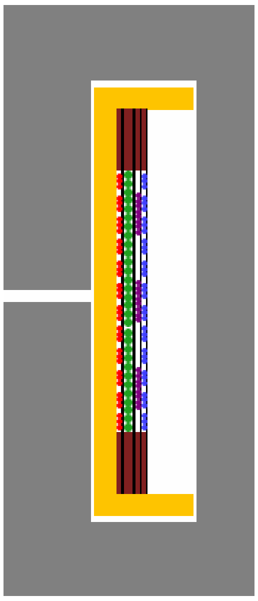


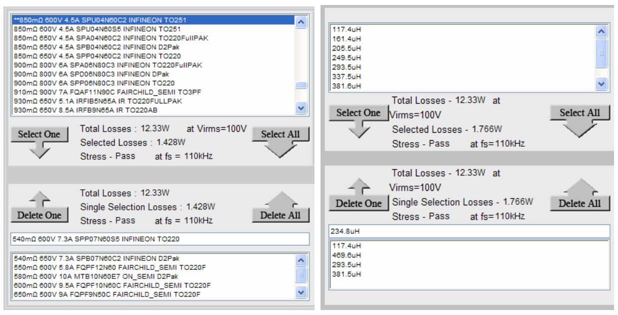
Fig. 5: Custom designed transformer GUI

of a transformer is not its equivalent model but its physical construction.

The proposed software provides a way for the user to construct the physical structure of a transformer. Fig. 5 illustrates the graphical user interface (GUI) of building the custom-made transformer. The user can choose the number of turns and select the real wires from the manufacturers such as enamel wire, triple insulated wire, or PVC wire, from the wire database. After selecting the wire, the user can also arrange the wire in a real magnetic core with custom PVC tape, creepage insulator, bobbin etc.

C. *Generic component selection page*

The components modeled in A) are selected from the Component Selection Page is the part in the circuit as Fig. 6 such as MOSFET. Besides, some numerical parameters, e.g. number of turns etc, are also regarded as component. The



```
**850mΩ 600V 4.5A SPU04N60C2 INFINEON TO251
850mΩ 600V 4.5A SPU04N60S5 INFINEON TO251
850mΩ 650V 4.5A SPA04N60C2 INFINEON TO220FullPAK
850mΩ 650V 4.5A SPB04N60C2 INFINEON D2Pak
850mΩ 650V 4.5A SPP04N60C2 INFINEON TO220
900mΩ 800V 6A SPA06N80C3 INFINEON TO220FullPAK
900mΩ 800V 6A SPD06N80C3 INFINEON DPak
900mΩ 800V 6A SPP06N80C3 INFINEON TO220
910mΩ 900V 7A FQAF11N90C FAIRCHILD_SEMI TO3PF
930mΩ 650V 5.1A IRFIB5N65A IR TO220FULLPAK
930mΩ 650V 8.5A IRFB9N65A IR TO220AB
```

Total Losses : 12.33W at Virms=100V
Selected Losses : 1.428W
Stress - Pass at fs = 110kHz

Total Losses : 12.33W
Single Selection Losses : 1.428W
Stress - Pass at fs = 110kHz

```
540mΩ 600V 7.3A SPP07N60S5 INFINEON TO220
```

```
540mΩ 650V 7.3A SPB07N60C2 INFINEON D2Pak
550mΩ 600V 5.8A FQPF12N60 FAIRCHILD_SEMI TO220F
580mΩ 600V 10A MTB10N60E7 ON_SEMI D2Pak
600mΩ 600V 9.5A FQPF10N60C FAIRCHILD_SEMI TO220F
650mΩ 500V 9A FQPF9N50C FAIRCHILD_SEMI TO220F
```

```
117.4uH
161.4uH
205.5uH
249.5uH
293.5uH
337.5uH
381.6uH
```

Total Losses - 12.33W at Virms=100V
Selected Losses - 1.766W
Stress - Pass at fs=110kHz

Total Losses - 12.33W at Virms=100V
Single Selection Losses - 1.766W
Stress - Pass at fs=110kHz

```
234.8uH
```

```
117.4uH
469.6uH
293.5uH
381.5uH
```

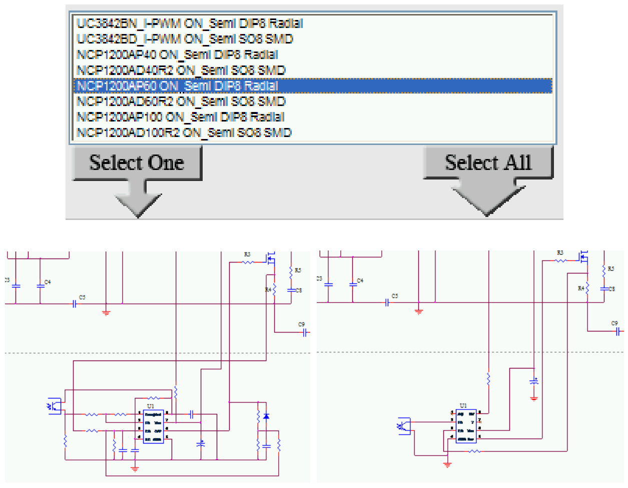
Fig. 6: Generic component selection pages

values are quantized to discrete component and supported by the generic component selection page. This allows the optimization engine searching the best solution among finite number of component combinations with concurrent sophisticated optimization method, e.g. GA.

The three main parts in the whole SMPS circuit are converter, PWM controller and feedback circuits. They can be selected as component as shown in Fig. 7.

VI. OPERATIONAL FLOWS

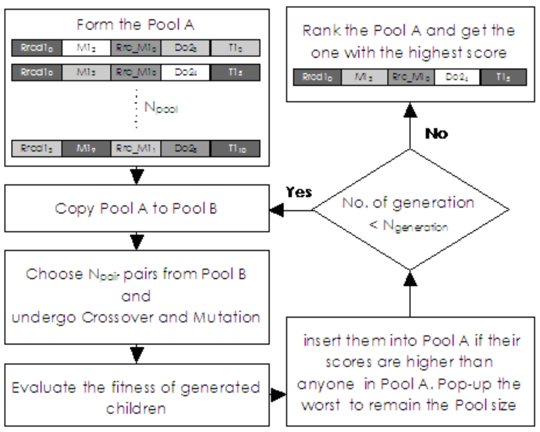
The most important merit of CBA is the discrete characteristics, which allow optimization without domain knowledge. Any generic optimization method can be adopted according to different cost function, e.g. losses, gain margin, material cost, etc.



```
UC3842BN_I-PWM ON_Semi DIP8 Radial
UC3842BD_I-PWM ON_Semi SO8 SMD
NCP1200AP40 ON_Semi DIP8 Radial
NCP1200AD40R2 ON_Semi SO8 SMD
NCP1200AP60 ON_Semi DIP8 Radial
NCP1200AD60R2 ON_Semi SO8 SMD
NCP1200AP100 ON_Semi DIP8 Radial
NCP1200AD100R2 ON_Semi SO8 SMD
```

Fig. 7: Generic component page for selecting circuits

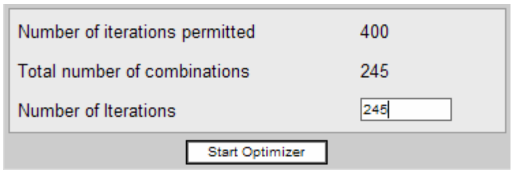
Losses or Efficiency optimization can be easily done by putting the losses calculated by the circuit simulator with Genetic Algorithm [4] as shown at Fig. 8. With limited resources in computation, the number of iterations or number of called cost functions in optimization should be controlled.



Form the Pool A
Rrcd1₆ | MI₂ | Rrc_MI₆ | Do2₃ | T1₅
Rrcd1₆ | MI₃ | Rrc_MI₆ | Do2₄ | T1₃
Npool
Rrcd1₃ | MI₇ | Rrc_MI₁ | Do2₃ | T1₁₀
Copy Pool A to Pool B
Choose Npair pairs from Pool B and undergo Crossover and Mutation
Evaluate the fitness of generated children
insert them into Pool A if their scores are higher than anyone in Pool A. Pop-up the worst to remain the Pool size
No. of generation < Ngeneration
Yes
No
Rank the Pool A and get the one with the highest score
Rrcd1₆ | MI₃ | Rrc_MI₆ | Do2₄ | T1₃

Fig. 8: Optimizer using Genetic Algorithm

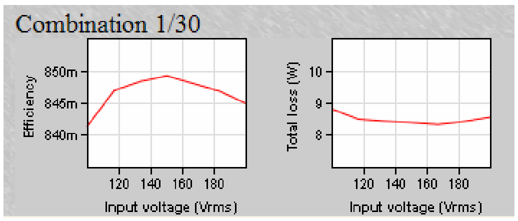
Fig. 9 shows the user interface for user to set the maximum number of iterations allowed. The more the number of iteration, the more possible the best solution is found.


Number of iterations permitted 400

Total number of combinations 245

Number of Iterations 245

(a) Optimizer panel



(b) Ranked results with different efficiencies

Fig. 9: Optimizer and ranked results

As mentioned in IV, the prediction of losses and junction temperature of component are depending to each other. CBA can easily eliminate this situation as proposed in Fig. 10. The thermal simulator is an integrated part of the whole system to estimate loss. We preset the junction temperature for all components to approximate values (e.g. 60°C for resistors and 100°C for the main transformer). Then, the losses are estimated from the preset temperature and used in simulating the temperature after the placement on PCB. The results are put into the next simulation cycle.

The convergence detector determines the saturation of temperatures as the end of simulations. The results may not always converge since the overheat components may lead chain effects on other components. Indeed, this also represents the real situation that the whole SMPS is getting burnt and following by smoke. Fig. 11 shows one of the thermal simulation results.

VII. CONCLUSION

A CBA Component Based Architecture is proposed for Switching Power Supply simulation software. It introduces a new era of designing a power supply that require engineer knowledge from close-form optimization method to component choosing skill.

This approach is suitable for Product Development purpose. Engineer can estimate the circuit performance, BOM, cost, DVT, MTBF, Life and Stress in short time. The more consistent SMPS is designed by different engineer. Getting rid of the subjective relations avoids the possibility of falling in design blind spot. The quality of a power supply hence can be improved.

The future work can be possibly done on considering different circuits as a component for the optimization purpose.

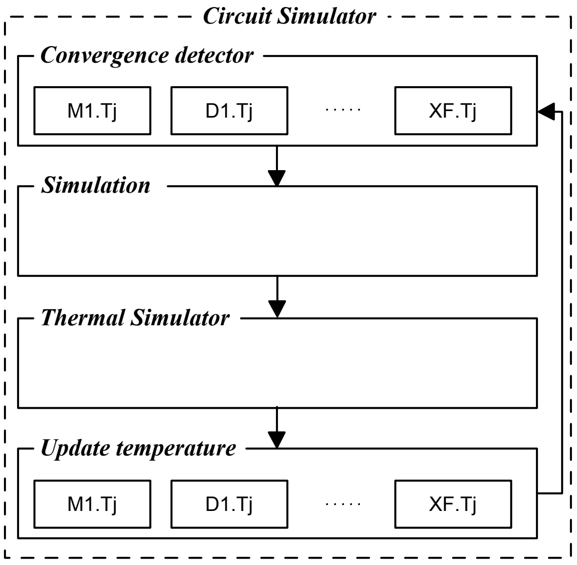


Fig. 10: Circuit simulation procedures

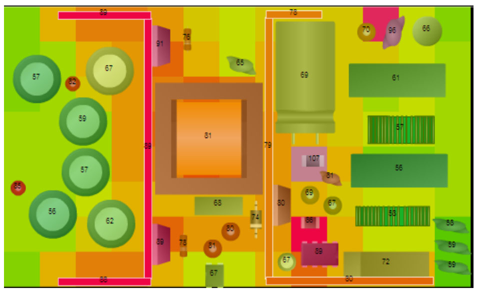


Fig. 11: Thermal Analysis GUI

ACKNOWLEDGEMENT

The pictures in this paper are captured from PowerESim (http://www.poweresim.com).

REFERENCES

[1] L. Barroca, J. Hall and P. Hall, Software Architectures Advances and Applications, Springer-Verlag London Limited, 1999.

[2] I. Crnkovic, J. A. Stafford, H. W. Schmidt and K. Wallnau, Component-Based Software Engineering, Springer-Verlag Berlin Heidelberg, 2004.

[3] D. E. Goldberg, Genetic Algorithms, Addison Wesley, 1988.

[4] D. Whitley, "A genetic algorithm tutorial", Statistics and Computing, vol. 4, pp. 65-85, 1994.