University of Arkansas, Fayetteville

ScholarWorks@UARK

Electrical Engineering Undergraduate Honors Theses

Electrical Engineering

5-2017

A Manufacturer Design Kit for Multi-Chip Power Module Layout Synthesis

Jonathan Main
University of Arkansas, Fayetteville

Follow this and additional works at: https://scholarworks.uark.edu/eleguht

Part of the Electrical and Computer Engineering Commons, and the Software Engineering Commons

Citation

Main, J. (2017). A Manufacturer Design Kit for Multi-Chip Power Module Layout Synthesis. *Electrical Engineering Undergraduate Honors Theses* Retrieved from https://scholarworks.uark.edu/eleguht/54

This Thesis is brought to you for free and open access by the Electrical Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Electrical Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact scholar@uark.edu, uarepos@uark.edu.

A Manufacturer Design Kit for Multi-Chip Power Module Layout Synthesis

An undergraduate honors college thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Electrical Engineering with Honors

by

Jonathan H. Main

April 2017
University of Arkansas

Abstract

The development of Multi-Chip Power Modules (MCPMs) has been a key factor in recent advancements in power electronics technologies. MCPMs achieve higher power density by combining multiple power semiconductor devices into one package. The work detailed in this thesis is part of an ongoing project to develop a computer-aided design software tool known as PowerSynth for MCPM layout synthesis and optimization. This thesis focuses on the definition and design of a Manufacturer Design Kit (MDK) for PowerSynth, which enables the designer to design an MCPM for a manufacturer's fabrication process.

The MDK is comprised of a layer stack and technology library, design rule checking (DRC), and layout versus schematic checking. File formats have been defined for layer stack and design rule input, and import functions have been written and integrated with the existing user interface and data structures to allow PowerSynth to accept these file formats as a form of input. Finally, an exhaustive DRC function has been implemented to allow the designer to verify that a synthesized layout meets all design rules before committing the design to manufacturing. This function was validated by running DRC on an example layout solution using two different sets of design rules.

Acknowledgements

I would like to take this opportunity to recognize some of the people who have helped me through the endeavor of completing this thesis and my undergraduate education at the University of Arkansas.

First, I am immensely thankful for the guidance of Dr. Alan Mantooth, who has not only served as my advisor for this thesis but also as a mentor throughout my time in college. I also want to thank Tom Vrostos and Dr. Yarui Peng for providing valuable guidance, technical expertise, and perspective on this project. I would like to acknowledge Shilpi Mukherjee, Quang Le, and Tristan Evans as well for their collaboration on this project. It has been a pleasure to work alongside them.

I also want to recognize the students and professionals who helped me get started with building a background on this thesis topic through personal interviews: Dr. Matt Francis, Paul Koch, Dr. Alex Lostetter, Chris Farnell, Sayan Seal, and Yuzhi Zhang.

I would like to thank the National Science Foundation's Center for Power Optimization of Electro-Thermal Systems (POETS) for funding this project.

Finally, I can't imagine my undergraduate experience without the support of my family and friends. They have been a constant source of love and encouragement through the ups and downs of college, and I truly wouldn't be who I am today without them.

Table of Contents

I. Introduction 1

II. Background 4

A. MCPM Structure 4

B. The MCPM Design Process 5

C. PDKs for IC Design 6

III. Definition of an MDK 8

IV. Software Design and Implementation 9

A. Layer Stack and Technology Library 9

B. Design Rules 14

C. Layout Versus Schematic 19

V. Results 21

VI. Conclusions and Future Research 25

VII. References 28

List of Figures

Figure 1: Catastrophic failure of an MCPM [1] 2

Figure 2: Simplified Structure of an MCPM [1] 5

Figure 3: Graphical Representation of PDK Components and the IC Design Process. 7

Figure 4: Previous Main Window GUI with Module Stack Input Fields 10

Figure 5: Excel Template for Layer Stack File Input 11

Figure 6: CSV Format of Layer Stack File 12

Figure 7: Main Window GUI with Layer Stack Import Option 13

Figure 8: Technology Library Editor/Wizard GUI 14

Figure 9: Previous Process Design Rules Editor GUI 15

Figure 10: Excel Template for Design Rules File Input 16

Figure 11: CSV Format of Design Rules File 16

Figure 12: Process Design Rules Editor GUI with Import Option 17

Figure 13: Solution Viewer UI with Run DRC Option 18

Figure 14: Layout Solution with Export Options 20

Figure 15: Default Design Rules for Exhaustive DRC Validation 22

Figure 16: Example Layout Solution for Exhaustive DRC Validation 23

Figure 17: Exhaustive DRC Pass Notification 24

Figure 18: Extreme Design Rules Used for Exhaustive DRC Validation 24

Figure 19: Exhaustive DRC Fail Notification 25

Figure 20: Exhaustive DRC Fail Console/Log Message 25

I. Introduction

Power semiconductor devices play a vital role in many of today's technological developments, including electric vehicles, aerospace technologies, and alternative energy. These devices control the flow of electric power between electronic systems and other components such as batteries, electric motors, and energy sources. Power electronics designers aim to combine these devices into high-performance systems that are inexpensive, small in size, fast, efficient, and reliable [1].

Recent advances in power electronics technologies have focused on increasing the switching frequency, allowing the power semiconductor devices to modulate the flow of power at a faster rate. The size of passive components such as capacitors, inductors, and transformers can be reduced at higher frequencies, which reduces cost and module size while increasing efficiency. However, there are several major design challenges associated with increasing the switching frequency. The effects of electrical parasitics increase switching losses, which can negate efficiency gains. Parasitics and electromagnetic interference (EMI) can also create problems with signal integrity and timing, as well as dangerous over-voltage conditions. Additionally, smaller module designs with higher power density place many heat-producing devices close together, which causes issues with heat dissipation. These problems can hinder module performance, reduce operating lifetime, or even cause catastrophic failure of the system as shown in Figure 1 [1].

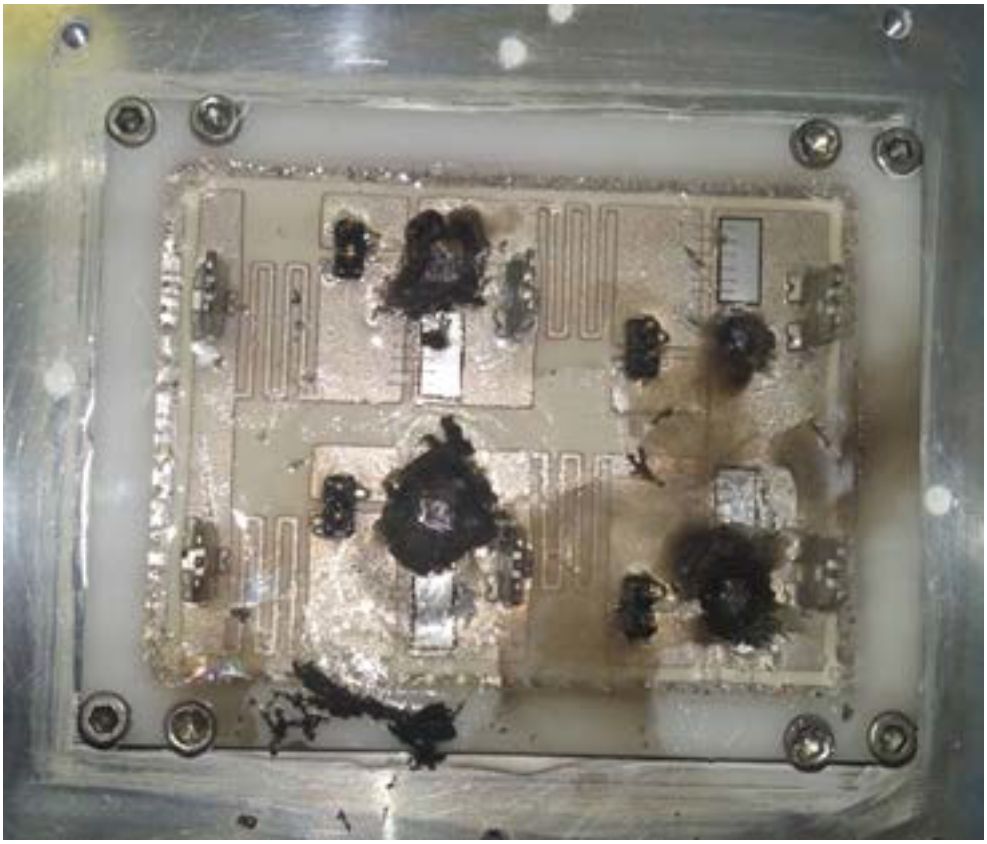


Figure 1: Catastrophic failure of an MCPM [1]

To mitigate these issues and push the boundaries of performance and reliability, power electronics designers have developed multi-chip power modules (MCPMs). MCPMs combine multiple power semiconductor devices into a single package. This approach reduces the effects of electrical parasitics and increases power density by packing devices close together with short interconnects between them [2]. MCPM design methodologies focus on minimizing parasitics and EMI effects while effectively dissipating heat for a high-power density module.

The current MCPM design process is largely a manual task that requires an experienced designer to go through multiple design iterations. For each iteration, the designer must model the electrical, thermal, and mechanical performance of the design and check that it meets both design

specifications and manufacturing process design rules. This process is costly and time-consuming, and it typically doesn't result in the most optimal layout possible [3].

The work detailed in this thesis is part of an ongoing effort to develop an electronic design automation (EDA) software tool for power module layout synthesis. This tool, known as PowerSynth, aims to aid the MCPM designer through automated layout synthesis and optimization. This project is led by Dr. Alan Mantooth and sponsored by the National Science Foundation's Center for Power Optimization of Electro-Thermal Systems (POETS).

PowerSynth utilizes multi-objective optimization algorithms to generate MCPM layouts that are optimized for both electrical and thermal performance. It relies on fast electrical and thermal models that can quickly approximate the electrical and thermal characteristics of a power module with small sacrifices in accuracy. These models are detailed in [4]. The designer inputs the intended circuit topology, components, materials, and other necessary information into PowerSynth. After running PowerSynth, the designer is presented with an array of layout solutions reflecting the trade-offs between optimal electrical and thermal performance. The designer can then choose the layout solution best suited to the intended application and export the design for further analysis or manufacturing.

The work described in this thesis focuses on the definition and design of a Manufacturer Design Kit (MDK) for PowerSynth. Like Process Design Kits (PDKs) used in integrated circuit (IC) design, an MDK contains everything an MCPM designer needs to design a power module with a manufacturer's manufacturing process. It aims to represent the boundaries of the MCPM manufacturing process to the designer in order to ensure manufacturability, improve reliability, and maximize efficiency of MCPM designs.

This thesis consists of five chapters, including this introduction. Chapter 2 provides background information on power module structure, power module design, and PDKs. Chapter 3 outlines the definition, purpose, and components of an MDK. Chapter 4 details the software design and implementation of an MDK for PowerSynth. Chapter 5 presents results from validating some of the MDK functions. Finally, this thesis concludes with a summary of the work and discussion of future research possibilities in Chapter 6

II. Background

A. MCPM Structure

An MCPM consists of a series of layers and components as shown in Figure 2. The baseplate at the bottom of the structure primarily acts as a heat spreader to help dissipate heat from the power devices. Above the baseplate is a substrate, which consists of a bottom metal layer, a top layer of interconnect, and a dielectric layer between them. The interconnect traces are etched according to the desired circuit topology in order to connect the power devices. A solder-like material referred to as substrate attach bonds the bottom of the substrate to the top of the baseplate. Power devices, which are typically unpackaged die, are bonded to the top of the substrate by another solder-like material referred to as die attach. Power bond wires may connect power die to other traces, and package leads are attached to traces for external connections such as power, ground, and output.

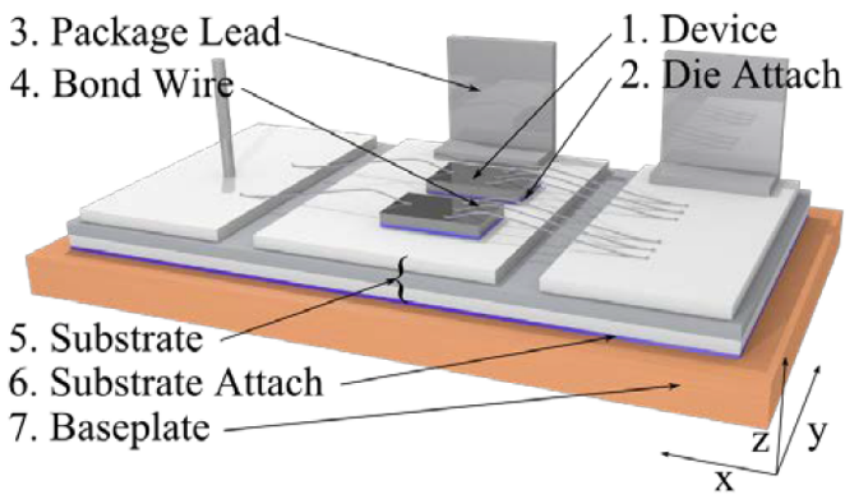


Figure 2: Simplified Structure of an MCPM [1]

The MCPM structure shown in Figure 2 is a simple example of the structure currently handled by PowerSynth. New MCPM structures are emerging that seek to further improve power density, electrical performance, and thermal management by using a more diverse range of designs. These approaches include bond wire-less modules and three-dimensional designs that may stack multiple components, substrates, baseplates, heatsinks, or other layers on either side of the design. Designers are also developing MCPMs that incorporate discrete components, integrate gate driver circuits into the module package, and utilize non-rectangular traces and routing. PowerSynth currently does not incorporate these design types, but they are areas of future research and development.

B. The MCPM Design Process

The basic process of designing MCPMs follows a similar pattern to other circuit design techniques, including those used in integrated circuit (IC) design. The designer begins with a set

of specifications, which may include electrical, thermal, and mechanical characteristics as well as manufacturing process design rules and additional constraints. The designer draws a schematic of the intended electrical circuit, which defines its electrical connectivity including components and external connections. The schematic is typically stored as a text file called a netlist. Simulation programs such as SPICE can be used to analyze the schematic and predict the electrical behavior of the circuit. After completing the schematic, the designer develops the physical layout of the circuit, including placement of the components, routing of the traces between them, and design of the inner layers of the board or substrate. Once physical layout is finalized, the designer may conduct additional electrical, thermal, and mechanical analyses and fine-tune the design before committing it to manufacturing. Modern design methodologies rely on EDA software tools to assist the designer and manage design files.

C. PDKs for IC Design

PDKs provide important links between EDA software and IC design manufacturing technologies and processes [5]. A PDK consists of files and libraries that help to streamline the design process and enable the designer to consider the manufacturing process and its implications throughout the IC design procedure.

From a broader perspective, a PDK assists in the design of ICs by representing the entire manufacturing process to the designer. In a personal interview with the author, Paul Koch provided a visual depiction of the components of a PDK and their relationships with various stages of the IC design process. This graphic is shown in Figure 3 with the PDK components highlighted in yellow. The IC design process can be thought of as two different "views" – the behavioral view and the physical view. The behavioral view refers to the electrical behavior and

schematic of the circuit, while the physical view perspective focuses on the physical design and layout. A PDK assists the designer in navigating the IC design process and keeping the design consistent across both views.

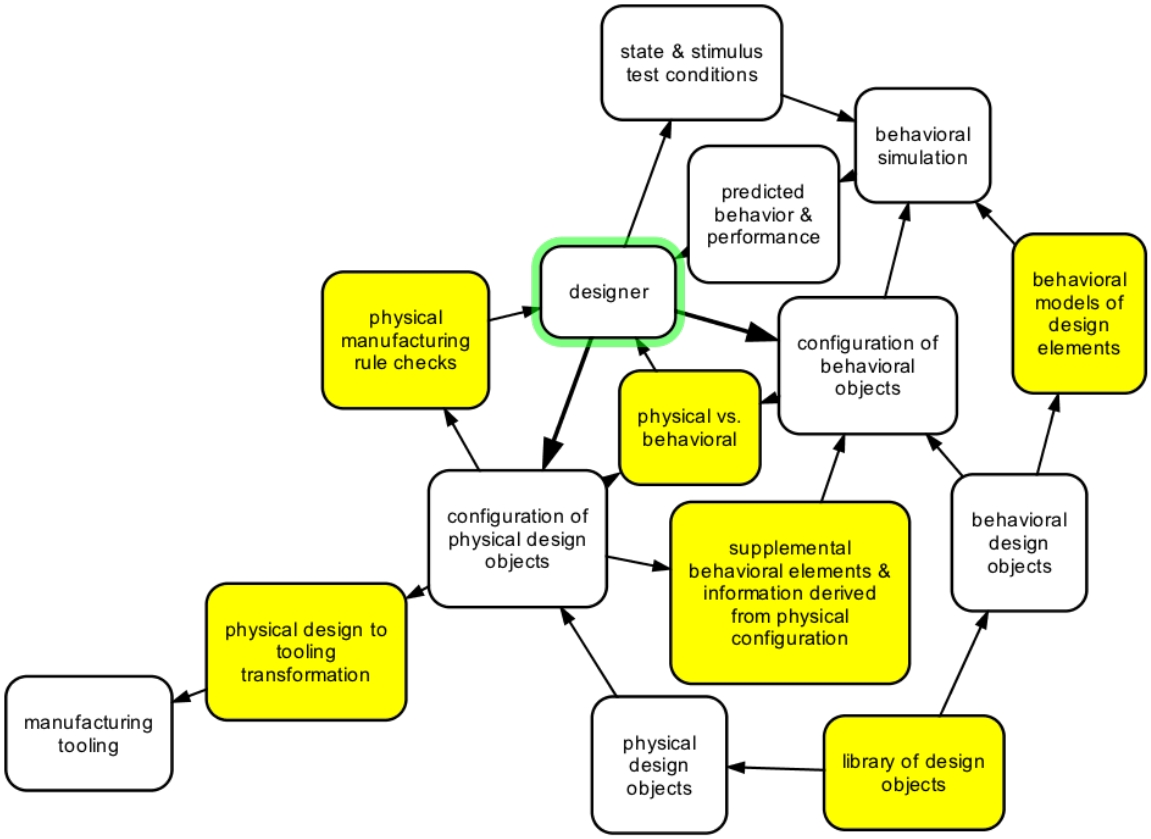


Figure 3: Graphical Representation of PDK Components and the IC Design Process.

A PDK assists the designer first by providing a library of design objects. These objects include behavioral design objects, such as SPICE models, and physical design objects that may contain information about an object's physical dimensions and properties. The designer utilizes behavioral models in simulators like SPICE to predict the performance of the IC. Once the designer begins configuring the physical design objects in a layout, a PDK allows the designer to

check that the layout meets physical manufacturing tolerances and specifications. A PDK can also extract additional behavioral elements that are derived from the physical layout and feed these elements back into the behavioral side of the design process. Parasitic extraction is an important aspect of this, as it allows the designer to analyze electrical parasitics introduced by the layout and perform additional behavioral simulations considering their effects on performance. PDKs also include checks between the physical and behavioral design aspects to ensure that the physical layout matches the intended circuit schematic. This feature is referred to as layout vs schematic (LVS) checking. Finally, a PDK may contain data about the manufacturing tooling used to fabricate the circuit. This allows the designer to more easily transfer the physical design into the necessary format for fabrication.

III. Definition of an MDK

An MDK models the MCPM manufacturing process for EDA software tools. In many ways an MDK follows the same concepts as a PDK, but applies them to MCPM design rather than IC design. The contents of an MDK are contained in files, data structures, libraries, and software functions. The end goals of an MDK are to assist with the design of more efficient and reliable high performance MCPMs and to ensure that these designs are manufacturable. The components of an MDK are organized into three categories: Layer Stack and Technology Library, Design Rules, and LVS.

A layer stack contains data about the various layers in an MCPM design. It stores the order of the layers as well as the name, type, and dimensions of each layer. The layer stack may also contain layer properties or other relevant data. A few simple examples of layer types are baseplates, metal, dielectric, and interconnect layers. Other layer types may be necessary to

model more complex MCPM designs, as will be discussed later in this thesis. The technology library contains data about the materials and devices used in power module designs. This data can be shared between different projects and applications.

Design rules are a set of constraints that ensure that a module design can be successfully fabricated. These rules are primarily based on fabrication process tolerances and thus obtained from manufacturers. Specific design constraints typically vary between different foundries and processes. Design rule checking (DRC) is an important function in EDA software as it allows the designer to analyze a layout to find and correct any design rule violations.

Finally, LVS assists with maintaining consistency between the behavioral and physical sides of module design. Parasitic extraction enables a designer to examine the electrical parasitics of the physical MCPM layout. An MDK should facilitate the extraction and export of parasitics information so the designer can conduct electrical analyses considering parasitic effects and revise the layout as necessary to improve expected module performance. Like its PDK counterpart, the LVS component of an MDK also compares the module layout with the schematic to ensure that the physical design matches the intended circuit schematic.

IV. Software Design and Implementation

A. Layer Stack and Technology Library

PowerSynth allows the designer to input layer stack data in the Module Stack section of the main window GUI, which is shown in Figure 4. The designer can select a technology library file for the material properties and type in values for dimensions and other layer properties. PowerSynth checks that the module stack input data is valid and then feeds this data into layout synthesis.

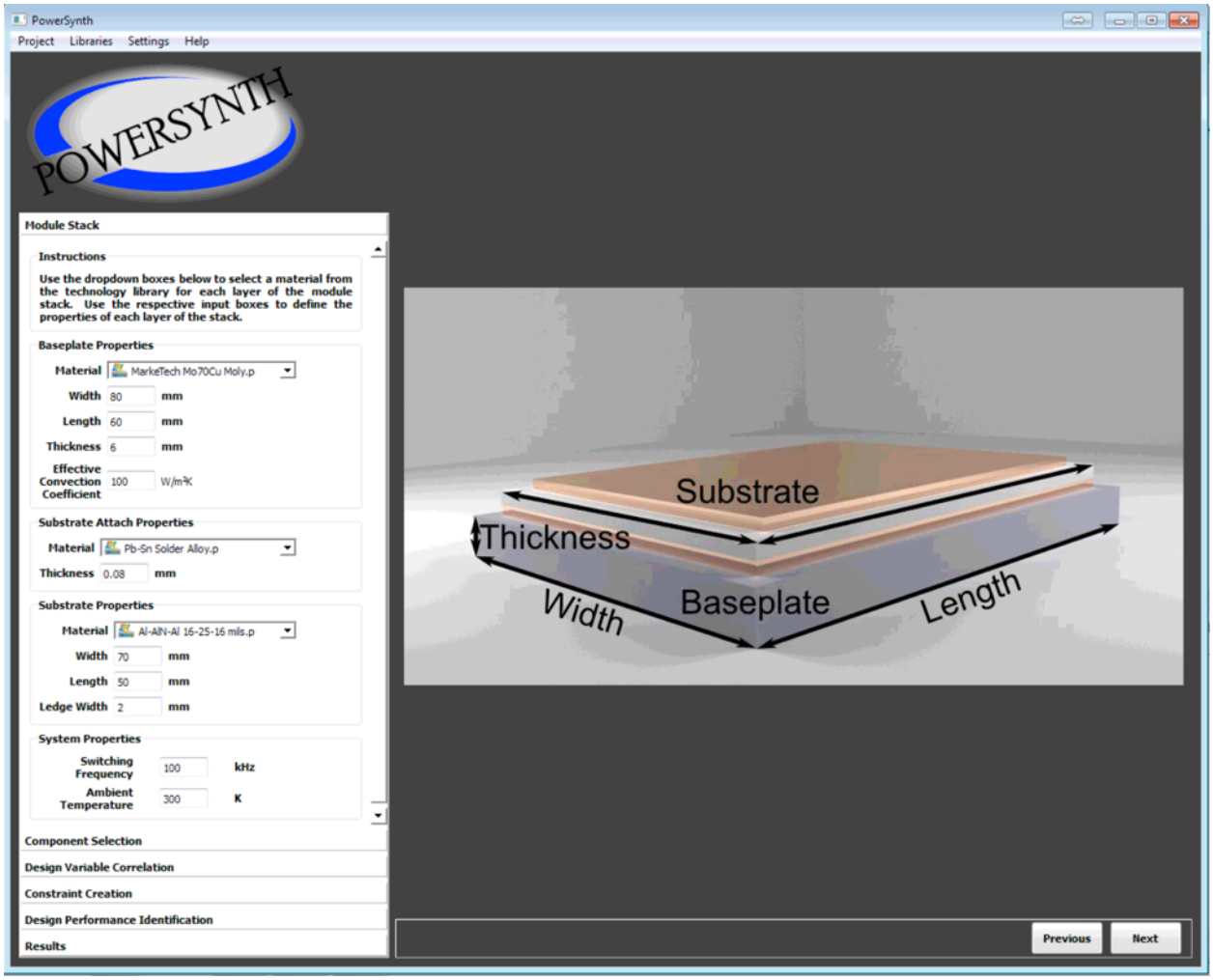
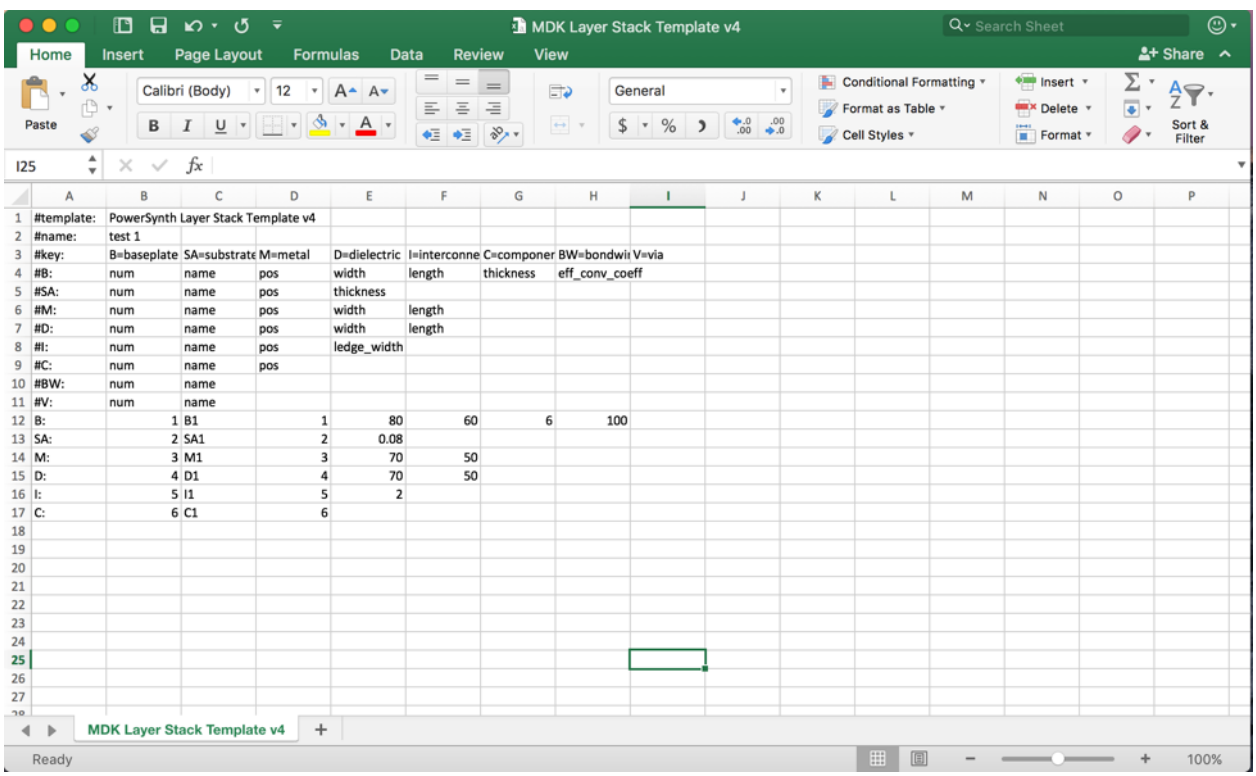


Figure 4: Previous Main Window GUI with Module Stack Input Fields

Currently, PowerSynth is designed for a layout structure described in section 2.1 of this thesis. This structure limits designs to one baseplate and one substrate and does not support three-dimensional module designs that stack multiple layers on either side of the design, gate driver circuits, or non-rectangular traces and routing.

While the present implementation is limited to this structure, a layer stack file format has been defined that is generalized enough to allow future developers to use it for later versions of PowerSynth that may support a more diverse range of layout structures. The layer stack file

stores layer information such as layer names, positions, dimensions, and properties where applicable. This file is defined in comma-separated values (CSV) format primarily because this format can be easily parsed. CSV files are also supported by Microsoft Excel, which can serve as a GUI for the designer to enter layer stack data into the file. An Excel template has been defined to assist the designer in creating or modifying a layer stack file. This template, shown in Figure 5, is designed for easy parsing and simplicity so the designer can easily enter rules data and future developers can easily modify or add new layers. The CSV format for this template is shown in Figure 6.



	A	B	C	D	E	F	G	H	I
1	#template:	PowerSynth Layer Stack Template v4							
2	#name:	test 1							
3	#key:	B=baseplate	SA=substrate	M=metal	D=dielectric	I=interconne	C=componer	BW=bondwir	V=via
4	#B:	num	name	pos	width	length	thickness	eff_conv_coeff	
5	#SA:	num	name	pos	thickness				
6	#M:	num	name	pos	width	length			
7	#D:	num	name	pos	width	length			
8	#I:	num	name	pos	ledge_width				
9	#C:	num	name	pos					
10	#BW:	num	name						
11	#V:	num	name						
12	B:	1	B1	1	80	60	6	100	
13	SA:	2	SA1	2	0.08				
14	M:	3	M1	3	70	50			
15	D:	4	D1	4	70	50			
16	I:	5	I1	5	2				
17	C:	6	C1	6					

Figure 5: Excel Template for Layer Stack File Input

```
#template:,PowerSynth Layer Stack Template v4
#name:,test 1
#key:,B=baseplate,SA=substrate attach,M=metal,D=dielectric,I=interconnect,C=components,BW=bondwire,V=via
#B:,num,name,pos,width,length,thickness,eff_conv_coeff
#SA:,num,name,pos,thickness
#M:,num,name,pos,width,length
#D:,num,name,pos,width,length
#I:,num,name,pos,ledge_width
#C:,num,name,pos
#BW:,num,name
#V:,num,name
B:,1,B1,1,80,60,6,100
SA:,2,SA1,2,0.08
M:,3,M1,3,70,50
D:,4,D1,4,70,50
I:,5,I1,5,2
C:,6,C1,6
```

Figure 6: CSV Format of Layer Stack File

An import button, shown in Figure 7, has been added to the Module Stack section of the main window GUI to allow the designer to import layer stack data from a layer stack file. The import function called by the button allows the user to select a layer stack CSV file and then parses the CSV file to extract layer data. This data is automatically analyzed to determine if it is valid and compatible with the current module structure supported by PowerSynth. If the imported layer stack data is valid and compatible, it is used to populate the GUI fields in the Module Stack section of the main window. PowerSynth can then use this data for layout synthesis.

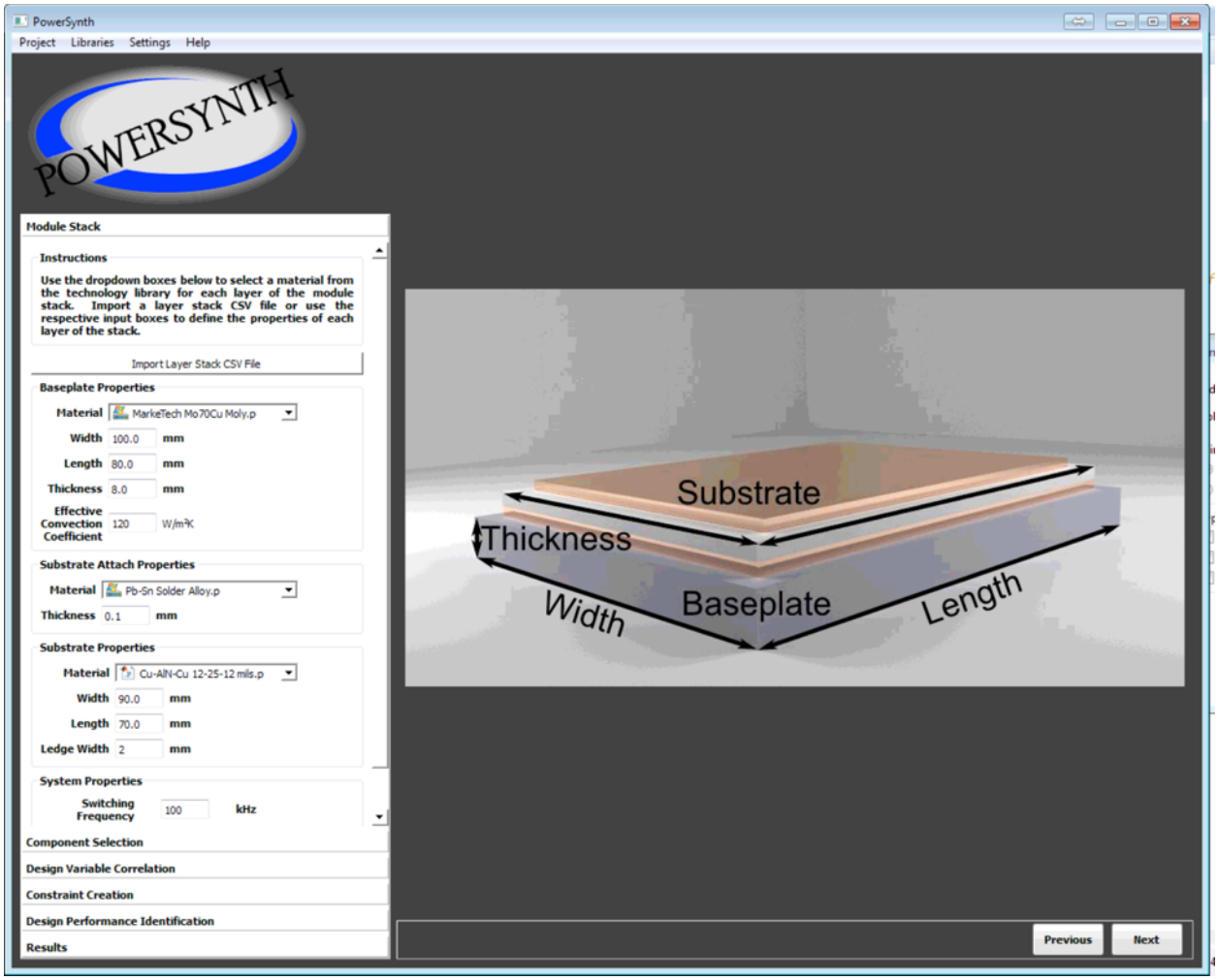


Figure 7: Main Window GUI with Layer Stack Import Option

A technology library structure was developed in PowerSynth prior to the work described in this thesis. This is implemented primarily through a Technology Library Editor (TLE), which provides an interface for the designer to easily access and make changes to the technology library. The GUI for the TLE is wizard format as shown in Figure 8. More information on the design of the TLE can be found in [1].

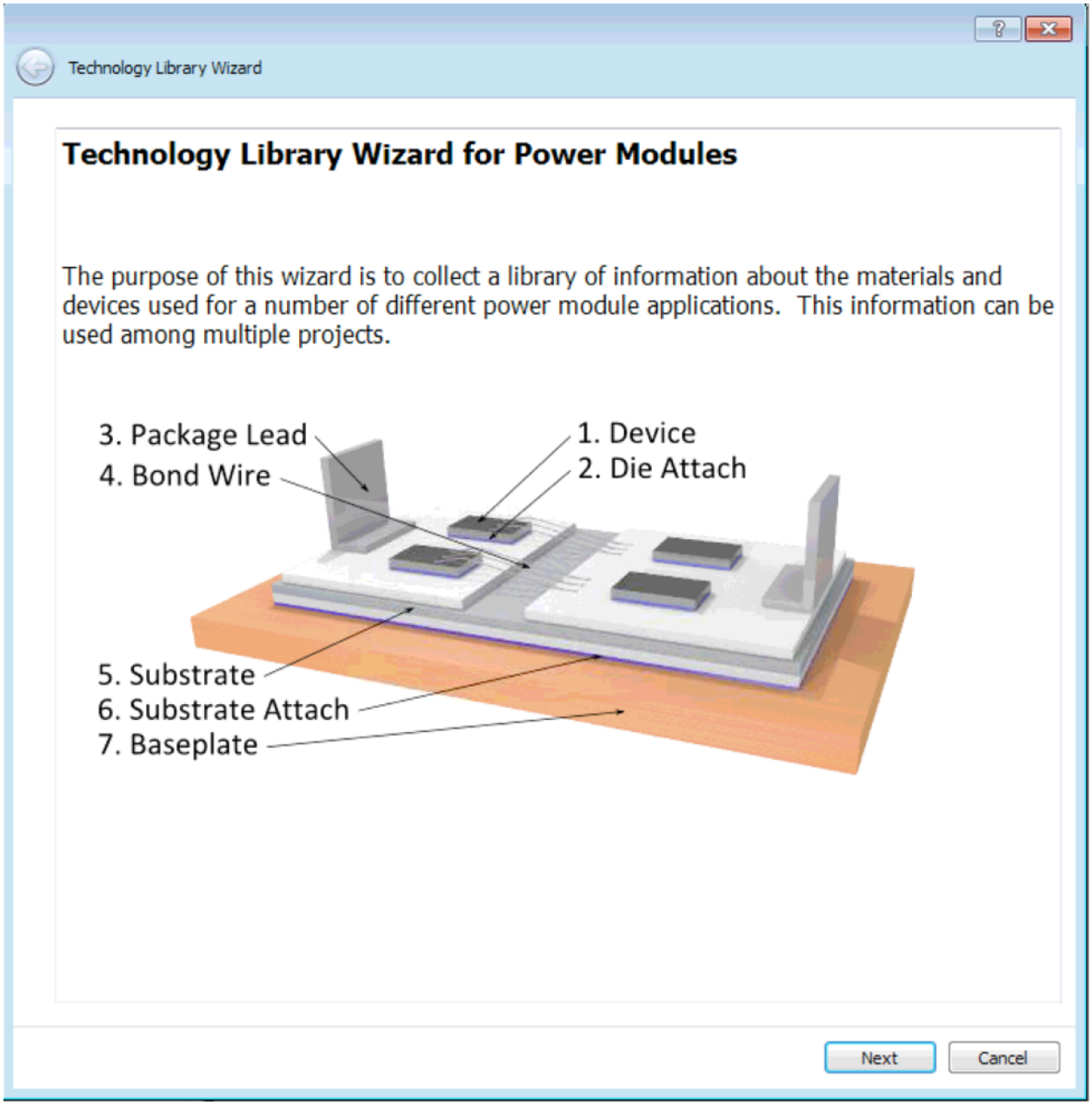


Figure 8: Technology Library Editor/Wizard GUI

B. Design Rules

Prior to the work described in this thesis, PowerSynth allowed the designer to enter process design rules through a GUI window as shown in Figure 9. These values are saved to a process design rules data structure for the current project. Design rule checking was implemented

by using the design rules as constraints on layout optimization. The optimization algorithm evaluates each possible layout generated and assigns it a value. Layouts that violate design rules were given an extremely large “undesirable” value so that the optimizer would eliminate these layouts from the potential solution set.

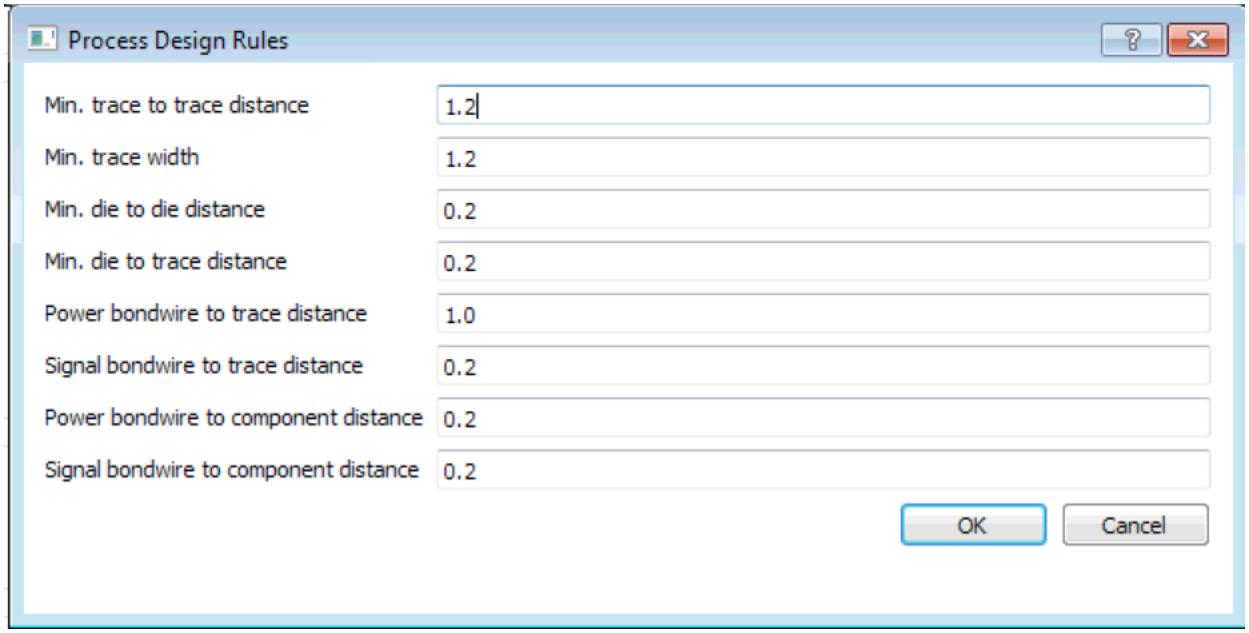
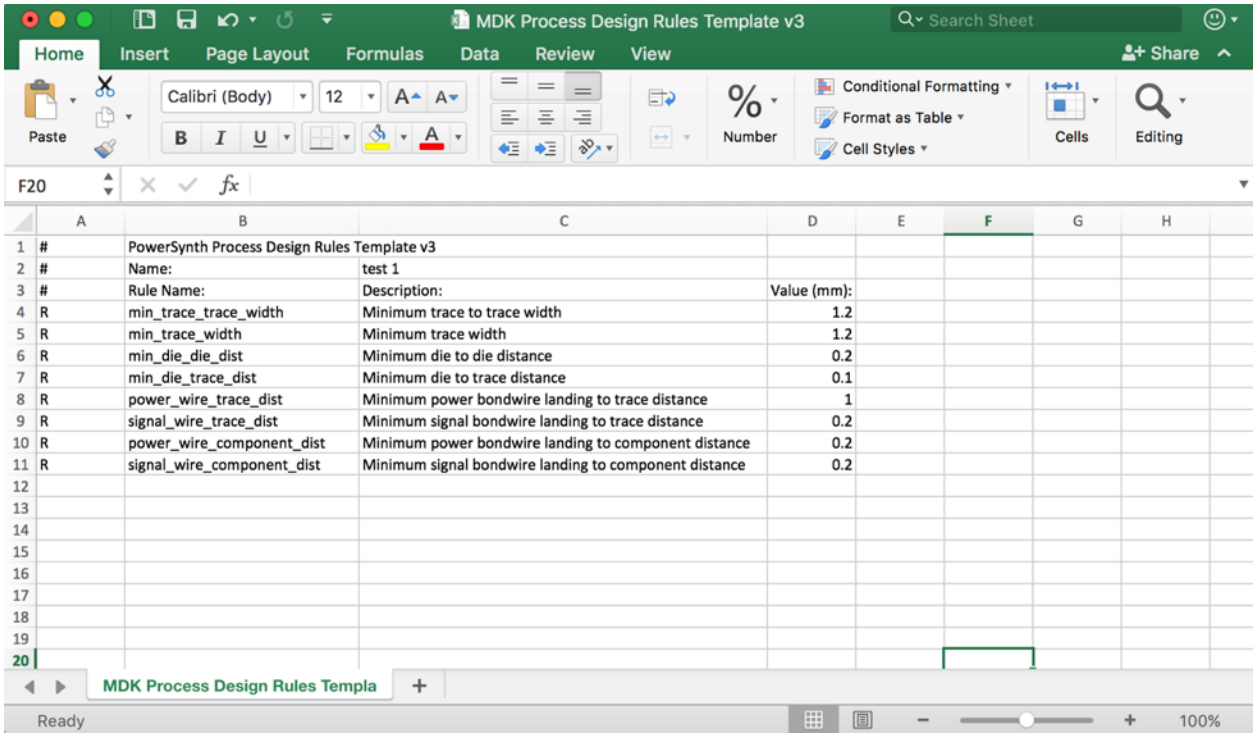


Figure 9: Previous Process Design Rules Editor GUI

The MDK includes a new design rules file format to store process design rules data. Like the layer stack file, the design rules file is defined in a CSV format and Microsoft Excel can be used as a GUI for the designer to enter layer stack data in to the file. An Excel template has been defined to assist the designer in creating or modifying a rules file. This template, shown in Figure 10, is designed for easy parsing and simplicity so the designer can easily enter rules data and future PowerSynth developers can easily add new rules if needed in the future. The CSV format for this template is shown in Figure 11.



	A	B	C	D
1	#	PowerSynth Process Design Rules Template v3		
2	#	Name:	test 1	
3	#	Rule Name:	Description:	Value (mm):
4	R	min_trace_trace_width	Minimum trace to trace width	1.2
5	R	min_trace_width	Minimum trace width	1.2
6	R	min_die_die_dist	Minimum die to die distance	0.2
7	R	min_die_trace_dist	Minimum die to trace distance	0.1
8	R	power_wire_trace_dist	Minimum power bondwire landing to trace distance	1
9	R	signal_wire_trace_dist	Minimum signal bondwire landing to trace distance	0.2
10	R	power_wire_component_dist	Minimum power bondwire landing to component distance	0.2
11	R	signal_wire_component_dist	Minimum signal bondwire landing to component distance	0.2

Figure 10: Excel Template for Design Rules File Input

```
#,PowerSynth Process Design Rules Template v3
#,Name:,test 1
#,Rule Name:,Description:,Value (mm):
R,min_trace_trace_width,Minimum trace to trace width,2.4
R,min_trace_width,Minimum trace width,2.4
R,min_die_die_dist,Minimum die to die distance,0.4
R,min_die_trace_dist,Minimum die to trace distance,0.4
R,power_wire_trace_dist,Minimum power bondwire landing to trace distance,2
R,signal_wire_trace_dist,Minimum signal bondwire landing to trace distance,0.4
R,power_wire_component_dist,Minimum power bondwire landing to component distance,0.4
R,signal_wire_component_dist,Minimum signal bondwire landing to component distance,0.4
```

Figure 11: CSV Format of Design Rules File

An import button, shown in Figure 12, was added to the Process Design Rules Editor GUI to allow the designer to import rules from a process design rules file. The import function called by the button allows the user to select a design rules CSV file and then parses the CSV file and populates the GUI fields for rule values from the rules data extracted from the CSV file.

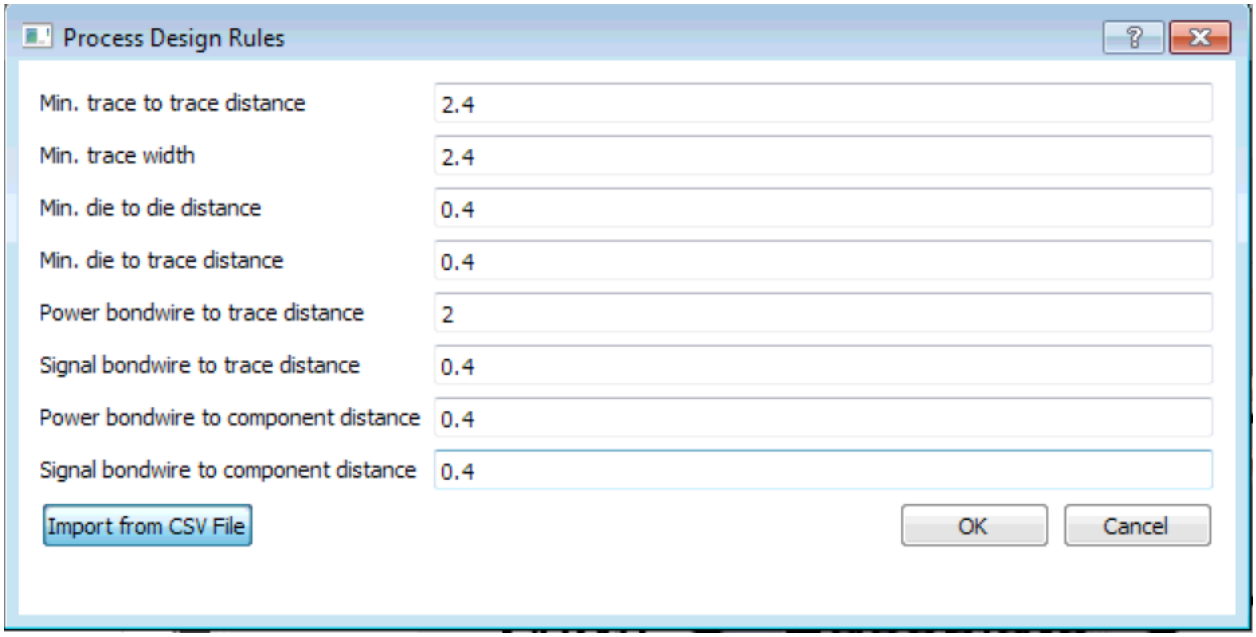


Figure 12: Process Design Rules Editor GUI with Import Option

The design rule optimizer constraint functionality discussed at the beginning of this section is maintained in the MDK implementation. In addition to the optimizer constraints, an exhaustive DRC function has been implemented to ensure that layout solutions pass all design rules. This function is run outside the optimization loop and serves as a final pass/fail DRC test for a layout solution. The exhaustive DRC function runs from the Solution Viewer UI as shown in Figure 13, outside the optimization loop, and notifies the designer if the layout passes DRC or if it has DRC violations.

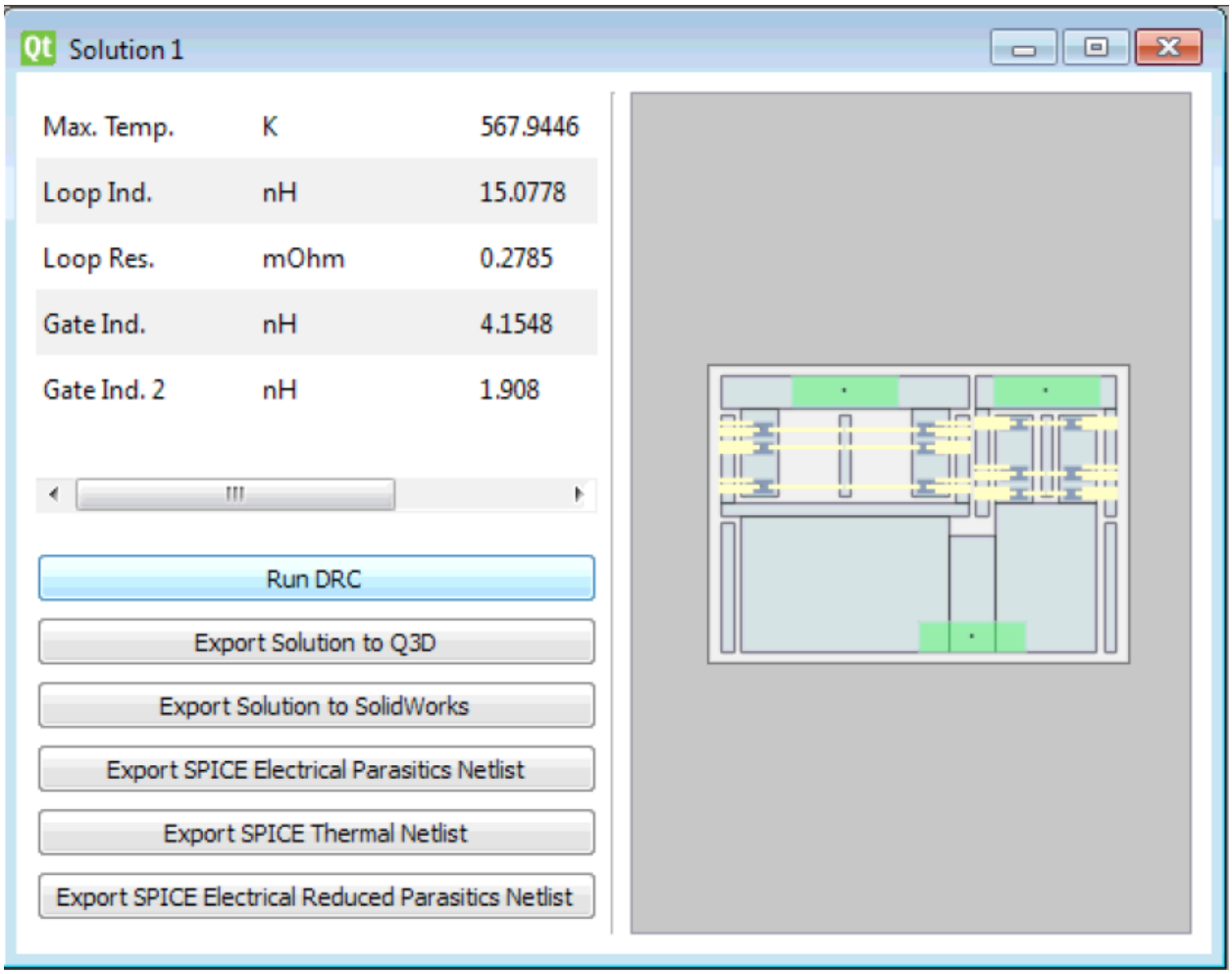


Figure 13: Solution Viewer UI with Run DRC Option

C. Layout Versus Schematic

Some LVS aspects of an MDK for PowerSynth were implemented prior to the work described in this thesis, but they are discussed here since they are part of the MDK. This section focuses on the two primary components of LVS for an MDK – parasitic extraction and connection checks.

As described previously in this thesis, parasitic extraction allows the designer to back-annotate schematics with electrical parasitics and perform simulations to compare expected performance with initial simulations. This allows the designer to further analyze and revise the design with consideration given to parasitics before committing to manufacture. The role of PowerSynth in parasitic extraction focuses on estimating the parasitics of a layout solution and providing the designer with the means to easily extract and export this data to industry-standard simulation tools like SPICE.

PowerSynth's electrical models can quickly calculate the parasitics for each layout solution. PowerSynth then allows the designer to export a netlist of the circuit with parasitic components added, including parasitic resistances, inductances, and capacitances. As discussed previously, a netlist is a text file describing the components and connectivity of the circuit. It is a standard format used by SPICE and other simulation tools. Thus, the designer can use PowerSynth to extract the parasitics of a layout solution and export them to SPICE for further analysis.

In addition to exporting an electrical netlist with parasitics, the designer can also use PowerSynth to export a thermal equivalent netlist. This type of netlist models the generation, flow, and dissipation of heat using electrical circuit components as an analog for thermal model

components [6]. For example, voltage is analogous to temperature and current is analogous to heat flow. A current source can be used to model a device emitting heat, which then flows through a network of thermal resistances and capacitances until it reaches ambient temperature outside the module, which can be represented by a voltage source. This allows the designer to simulate heat dissipation through a module using electric circuit simulators like SPICE and perform further thermal analysis of the layout.

Functions for exporting a SPICE electrical parasitics netlist or a SPICE thermal netlist are connected to buttons in the Solution Window GUI as shown in Figure 14.

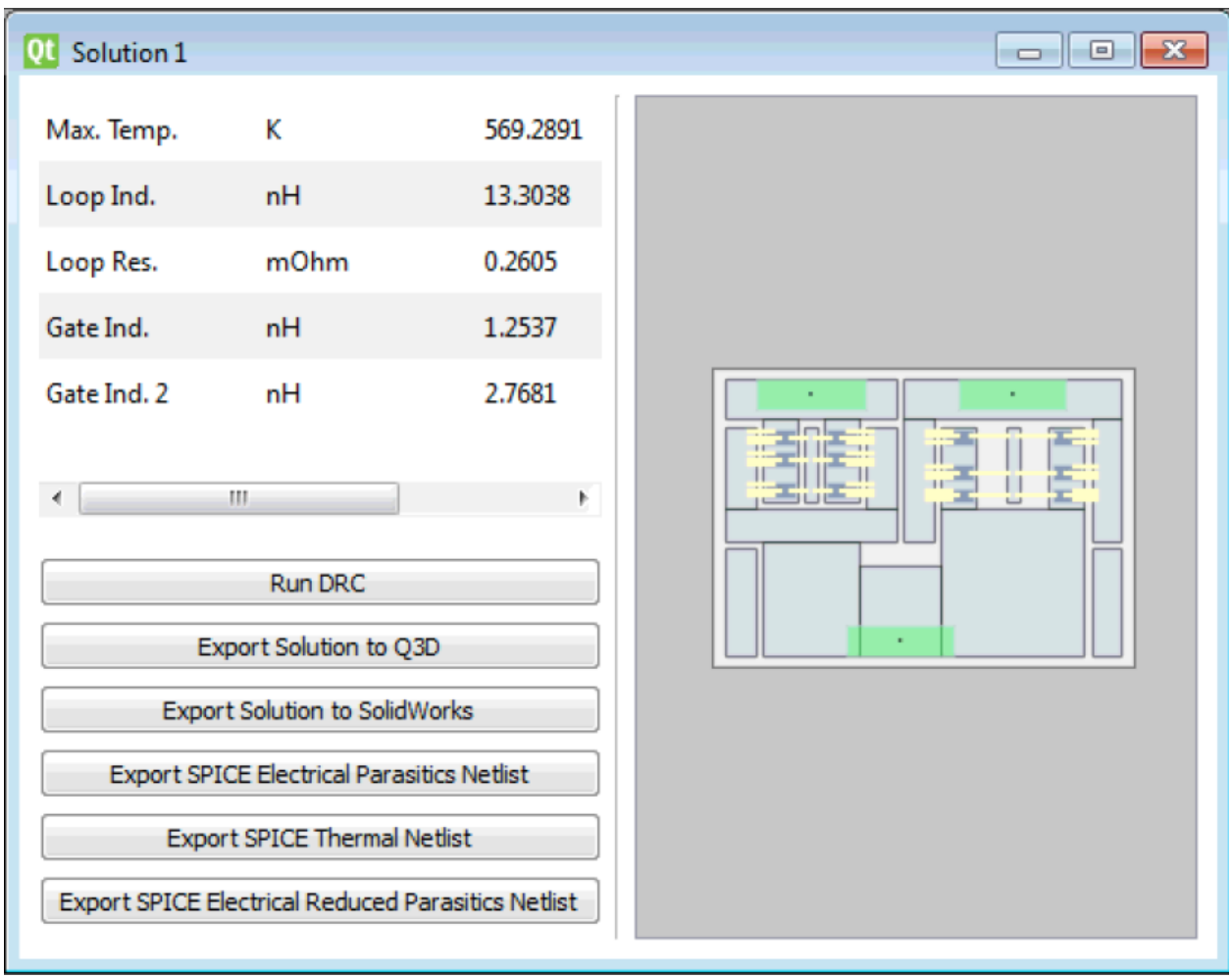


Figure 14: Layout Solution with Export Options

PowerSynth is designed to address connection checks through layout synthesis. By automatically generating the physical layout from a circuit netlist or simple topology, the software can be designed to ensure that layouts are "correct by construction," matching the intended circuit network.

V. Results

The results from testing layout synthesis in PowerSynth, including optimization constrained by DRC, can be found in [1]. As discussed previously, the DRC implemented in the optimization loop aims to eliminate generated layouts that violate design rules by giving them an extremely large "undesirable" value that is evaluated by the optimization algorithm. In contrast, the exhaustive DRC function is a strict pass/fail test that directly notifies the designer if a layout solution contains any design rule violations. This section focuses on validation of the exhaustive DRC function.

In order to test the exhaustive DRC function, PowerSynth was used to synthesize a set of layout solutions based on the R&D 100 award-winning MCPM design mentioned in [3]. The optimizer was constrained by the default set of design rules shown in Figure 15. A random solution shown in Figure 16 was selected from the solution browser.

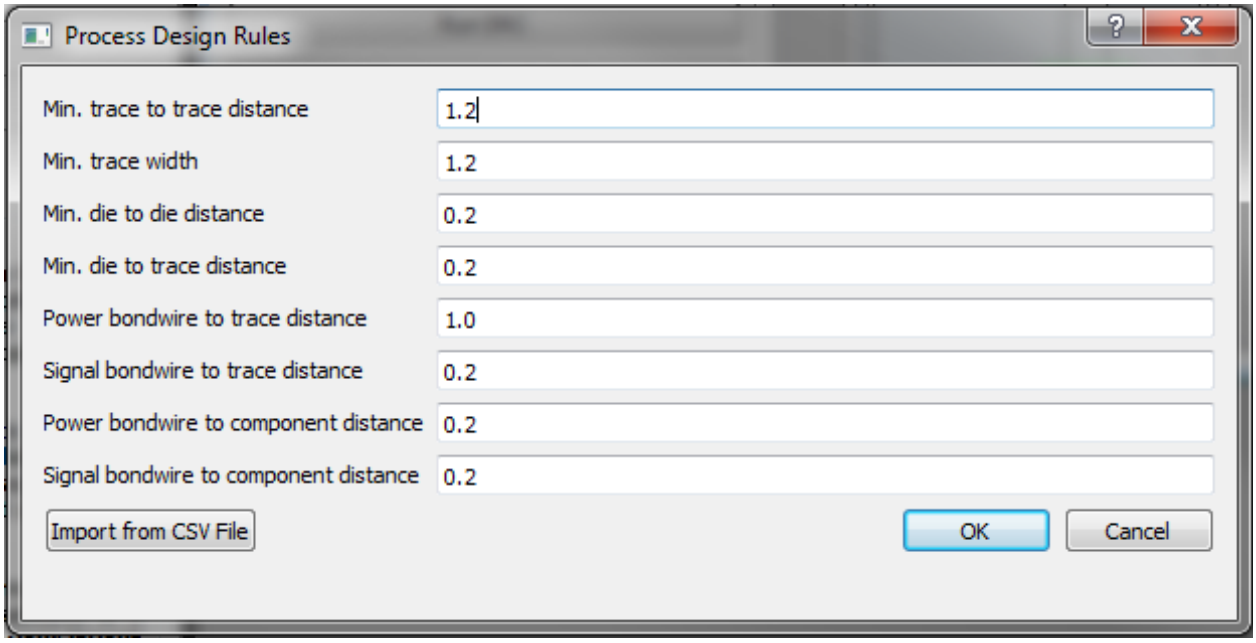


Figure 15: Default Design Rules for Exhaustive DRC Validation

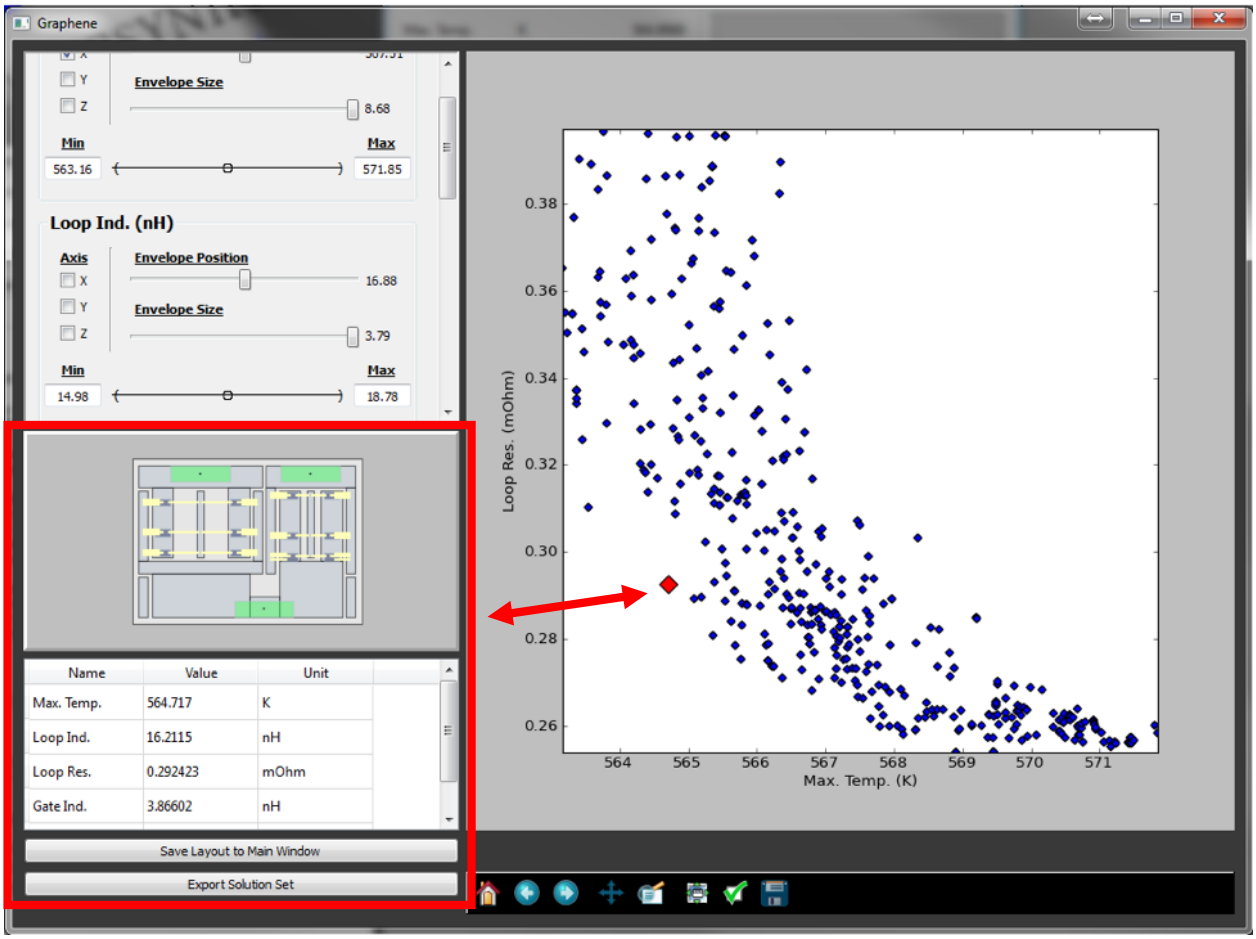


Figure 16: Example Layout Solution for Exhaustive DRC Validation

After saving the layout solution shown in Figure 16 to the main window, the exhaustive DRC function was run using the same default design rules that were used for optimization. The message box shown in Figure 17 appeared, showing that the layout passed all DRC as expected.

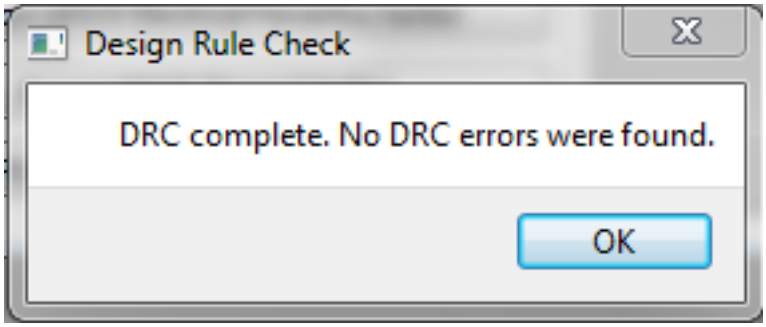


Figure 17: Exhaustive DRC Pass Notification

Next, the design rules were changed to an extreme set of values using the design rules editor dialog, as shown in Figure 18. The exhaustive DRC function was then run on the example layout using the extreme values. As expected, the message box shown in Figure 19 appeared, showing that the layout failed to pass DRC. As indicated in this notification, the designer can then check the console or log for details of the identified DRC violations. An example console or log message for this case is shown in Figure 20.

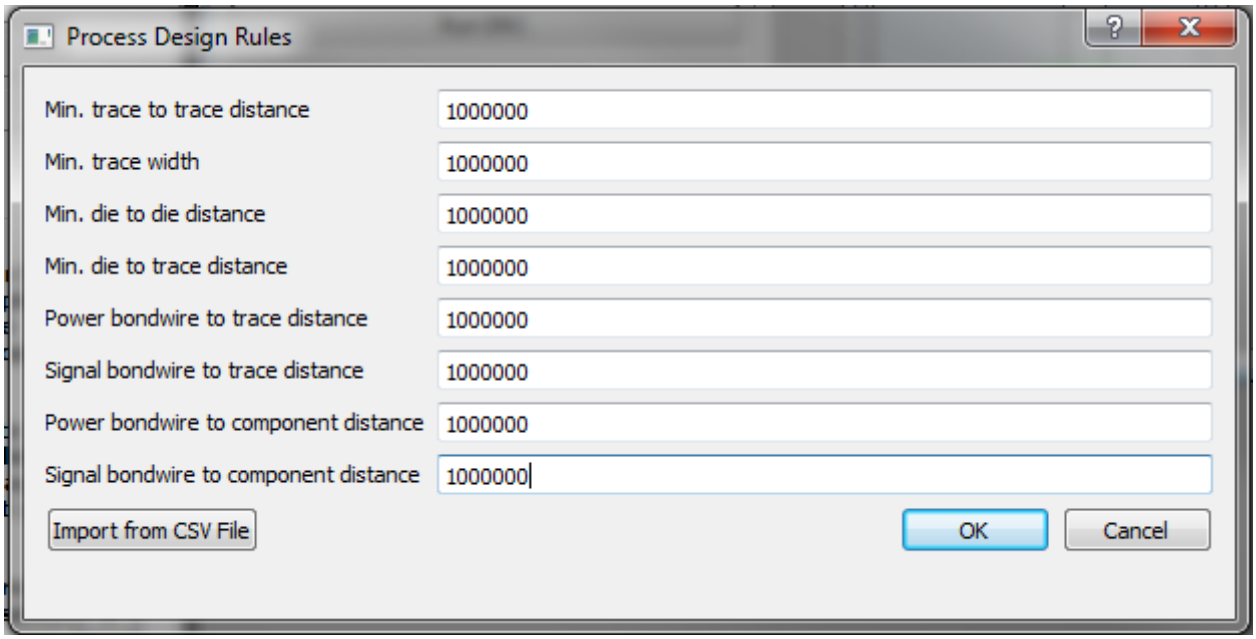


Figure 18: Extreme Design Rules Used for Exhaustive DRC Validation

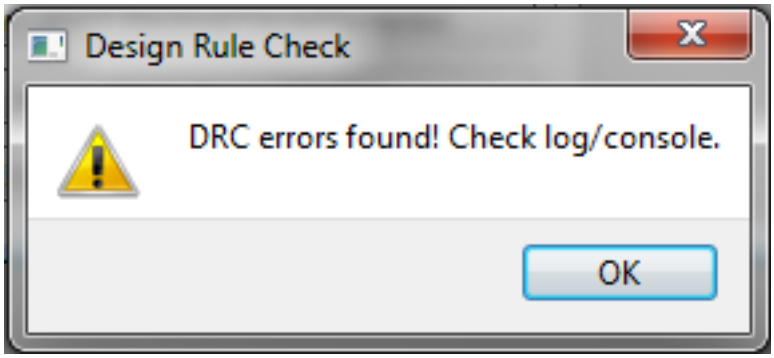


Figure 19: Exhaustive DRC Fail Notification

```
----- Running exhaustive DRC -----
Using design rules:
Min trace trace width = 1000000.0
Min trace width = 1000000.0
Min die die dist = 1000000.0
Min die trace dist = 1000000.0
Power bondwire to trace dist = 1000000.0
Signal bondwire to trace dist = 1000000.0
Power bondwire to comp dist = 1000000.0
Power bondwire to comp dist = 1000000.0
Traces less than Min_Width 35999514.8282
Devices are not overlapping trace 4.80001727989e+13
Two or more components are overlapping 760.32
Bondwires are not contacting trace 234.644624274
Bondwires are intersecting other layout components 43.9162595425
4.80002087995e+13 DRC errors found.
```

Figure 20: Exhaustive DRC Fail Console/Log Message

VI. Conclusions and Future Research

The work described in this thesis resulted in the definition of an MDK and its software design and implementation in PowerSynth. An MDK has been defined as a series of files, data structures, and software interfaces that allow an MCPM designer to design a module for a manufacturing process. An MDK consists of a layer stack and technology library for layer dimensions and properties, DRC, and LVS checking.

The MDK design for PowerSynth discussed in this thesis allows the designer to create design rules or the layout stack in a CSV file and import this data into PowerSynth for MCPM layout synthesis. The designer can input material properties through a technology library editor interface, which allows these properties to be saved in a technology library for use across different projects. Design rules are used as a constraint in layout synthesis and optimization so PowerSynth can automatically generate layouts that conform to these rules. The designer can also perform an exhaustive DRC test after layout synthesis to ensure that a layout solution does not violate any design rules before committing it to manufacturing. The exhaustive DRC function has been validated by running DRC on a synthesized layout using two different sets of design rules. These functions and others are integrated into the user interface and connected to the necessary data structures for PowerSynth to perform layout synthesis.

Future research building on this project could work toward generalizing the MDK for a wider array of MCPM design structures, including bond wire-less and three-dimensional structures. The data structures in PowerSynth that handle layer data could also be restructured using polymorphism to further modularize the source code. For example, a generic layer class could be defined, and classes representing different types of layers like metal or dielectric could extend the generic layer class. Input checking for the layer stack import, design rule import, and design rule editor could be improved to make these features more robust. Export functions could be added to allow the user to save layer stack data or design rules from PowerSynth to a CSV file. Finally, the exhaustive DRC feature could be improved to provide the designer with more detailed and user-friendly feedback about DRC violations. This could be accomplished by displaying a visual representation of the layout solution with indicators showing where DRC violations were found, which would better enable the designer to fix DRC errors before

finalizing the design for manufacturing. Finally, a feature could be implemented to directly compare a synthesized layout to the source netlist. While the current layout synthesis process does not change the netlist other than extracting layout parasitics, future work may include the additional of passive or active components during layout synthesis. A direct comparison between the generated layout and source netlist would then be more valuable to the designer.

VII. References

[1] B. W. Shook, "The Design and Implementation of a Multi-Chip Power Module Layout Synthesis Tool," M.S. thesis, Dept. Elect. Eng., Univ. of Arkansas, Fayetteville, AR, 2014.

[2] R. K. Ulrich and W. D. Brown, *Advanced Electronic Packaging*, 2nd ed. Piscataway, NJ: IEEE Press, 2006, ch. 1, sec. 1.4, pp. 15-17.

[3] P. N. Tucker, "SPICE Netlist Generation for Electrical Parasitic Modeling of Multi-Chip Power Module Designs," Undergraduate honors thesis, Dept. Elect. Engr., Univ. of Arkansas, Fayetteville, AR, 2013.

[4] Z. Gong, "Thermal and Electrical Parasitic Modeling for Multi-Chip Power Module Layout Synthesis," M.S. thesis, Dept. Elect. Engr., Univ. of Arkansas, Fayetteville, AR, 2010.

[5] Y. Li et al., "A complete Process Design Kit verification flow and platform for 28nm technology and beyond" in *2012 IEEE 11th International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, 2012 © IEEE. doi: 10.1109/ICSICT.2012.6467921

[6] S. Pearson and A. Laprade, "Tips and Tricks to Get More Out of Your SPICE Models," in *Fairchild Power Seminar*, 2007.