Describing Integrated Power Electronics Modules using STEP AP210

Yingxiang Wu

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Dr. Jan Helge Bøhn, Chair

Dr. Dushan Boroyevich, Co-chair

Dr. James Armstrong, Member

December 16, 2003

Blacksburg, Virginia

Keywords: IPEM, Design Automation, STEP, AP 210, Software Integration

Describing Integrated Power Electronics Modules using STEP AP210

Yingxiang WU

ABSTRACT

The software environment for power electronics design is comprised of tools that address many interrelated disciplines including circuits design, physical layout, thermal management, structural mechanics, and electromagnetics. This usually results in a number of separate models that provide various views of a design, each of which is usually stored separately in proprietary formats. The problem is that the relationships between views (e.g., the circuit design that defines the functional connectivity between components, and the physical layout that provides physical paths to implement connections), are not explicitly captured. This makes it difficult to synchronize and maintain data consistency across all models as changes are made to the respective views.

This thesis addresses this problem by describing power electronics modules using STEP AP210, the STandard for the Exchange of Product data, Application Protocol 210; which has been designated as ISO 10303-210. A multidisciplinary model was implemented for an integrated power electronics module (IPEM). It consists of two views of the IPEM: a functional network definition of the IPEM, and a physical implementation that satisfies the functional connectivity requirements. The relationships between these two views are explicitly recorded in the model. These relationships allow for the development of a method which verifies whether the connectivity data in both views are consistent. Finally, this thesis provides guidance for deploying STEP AP210 to unify multidisciplinary data resources during the design of integrated power electronics.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all the people who helped me to finish this thesis. In particular, I would like to thank:

- Dr. Jan Helge Bøhn, my advisor, for his constant support and guidance, and for all his great effort to help me complete this thesis;
- Dr. Dushan Boroyevich, my co-advisor, for his support, challenge, and valuable suggestions;
- Dr. James Armstrong, committee member, for joining the committee. I had the pleasure of take Dr. Armstrong's VHDL class;
- Thomas Thurman at Rockwell Collins for his insight knowledge of STEP. Without his help, this thesis would not have been completed;
- The Center of Power Electronics Systems (CPES) in the Department of Electrical and Computer Engineering at Virginia Tech for providing excellent resources; and
- The talented students and colleagues at CPES for their help and friendship.

This work was supported primarily by the ERC Program of the National Science Foundation under Award Number EEC-9731677. I would like to thank PACE Program, Ansoft Corporation, Avanti Corporation, and Engineous Software for providing the software licenses.

Finally, I want to thank my family for their continuous love, patience and endless supports.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Designing power electronics systems is a complex process and involves interactions between many engineering domains: electrical circuits, electro-magnetics, thermal dynamics, and material and structure mechanics. A traditional design process usually consists of a sequence of design and test iterations, progressing from one discipline to another with hardware prototyping and testing. The use of modern computer-aided design and engineering (CAD/CAE) tools has significantly reduced the prototyping and testing costs. However, these tools are usually mono-disciplinary, and coordination and knowledge sharing between them are labor intensive and time consuming. As a result, the design automation and optimization level in power electronics industry is much less advanced than in many other high-tech industries.

To improve power electronics design, an integrated multidisciplinary design methodology has been proposed in the Center for Power Electronics Systems (CPES) [Boroyevich03]. This approach relies on two things: multidisciplinary modeling for engineering analysis, and integration of existing commercial CAD/CAE tools. Multidisciplinary modeling is needed to capture the inter-disciplinary relationships concerning the components and their packaging, which is needed to further advance the state of power electronics. The engineering analyses based on these multidisciplinary models are traditionally performed within stand-alone CAD/CAE tools, which are designed for specific engineering domains and rarely talk to each other. It is important to integrate these tools so that data can be shared electronically and the multidisciplinary design-analysis iteration can be automated to facilitate design optimization.

This integrated multidisciplinary methodology has been applied to the design of the integrated power electronics modules (IPEMs) within CPES. Figure 1.1 illustrates the four basic steps in the IPEM integrated design process [Chen01]: 1. three dimensional (3D) solid modeling of the power module geometry; 2. electromagnetic lumped parameter extraction of the interconnects and packaging; 3. electrical circuit modeling and simulation; and 4. thermal modeling and finite element (FE) analysis of the power module.

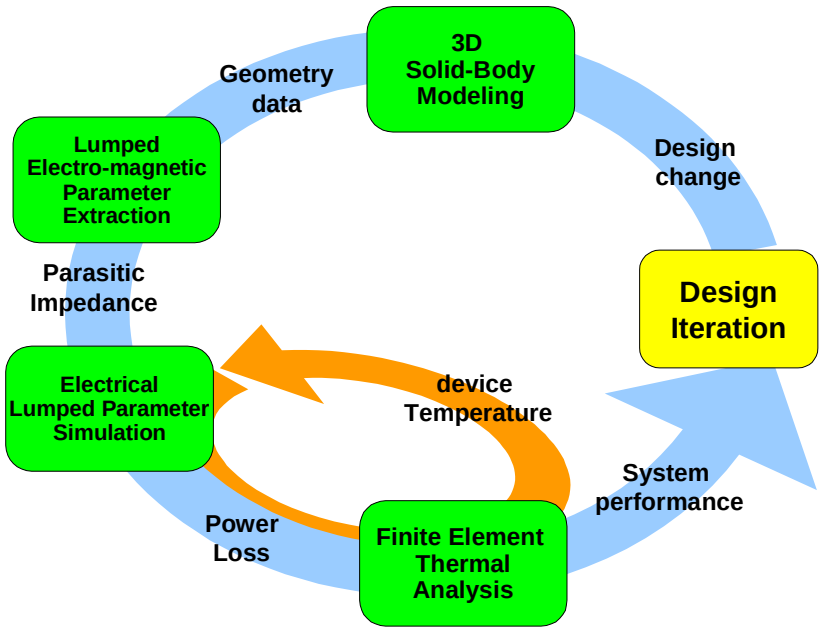


Figure 1.1 IPEM integrated design and analysis process.

First, the detailed 3D geometry model of the power module is constructed in a Mechanical CAD (MCAD) system. This solid model is shared by both the thermal analysis software for constructing the FE model and the electro-magnetic field software for the extraction of electro-magnetic parameters. The extracted parameters are used in a circuit simulation tool to model the properties of the interconnect in the circuit simulation model of the IPEM. The device power dissipations calculated by the circuit simulation

tool are then provided to the thermal analysis tool to evaluate temperature distribution within the module. The temperatures of the devices are then iteratively sent back to the circuit simulation tool to re-evaluate the power dissipation. Chapter 3 of this thesis will describe an integrated software system in which the data is shared electronically among these CAD tools in both standard and proprietary formats. This system allows the design-analysis iterations to be performed automatically and facilitates IPEM design optimization.

1.1 PROBLEMS STATEMENT AND OBJECTIVE

The problem currently faced by the design of IPEMs is that the models for a specific IPEM design are scattered among several CAD/CAE tools in different proprietary formats, and the relationships between these various views (e.g., the circuit design that defines the functional connectivity between components, or the physical layout that provides physical paths to implement connections) are not explicitly captured in its models. When changes appear in one view, it is difficult to synchronize the data in other views to satisfy the constraints between them. It is also difficult, from the file-bookkeeping point of view, to manage these scattered model files because of the lack of configuration management data (e.g., the version of the model and its application area).

The objective of this thesis is to provide a solution to construct a centralized data model for the IPEM in which multiple views of the module are provided and the relationships between these views are explicitly captured. This model can be used as a unified data resource where data of various levels of abstraction can be extracted for CAD systems across different domains, and the captured relationships between views can be used to verify model consistency whenever design data changes.

1.2 SOLUTION OUTLINE

The solution to this problem, as proposed in this thesis, is to describe the multidisciplinary aspects of the IPEM using STEP AP 210 [ISO01]. STEP is the STandard for the Exchange of Product data designated as ISO 10303 [ISO94], and AP 210 is the Application Protocol for electronic assembly, interconnection, and packaging design. STEP AP 210 provides the data structures for exchanging information between application processes during the requirements definition, design, analysis, and manufacturing phases of electromechanical products.

This thesis demonstrates the use of AP 210 by providing two different views for a Generation II IPEM (Gen-II IPEM) [Barbosa02] and explicitly describing the relationships between these two views. A photograph of this IPEM is shown in Figure 1.2(a). This thesis implements a simplified model (Figure 1.2(b)) which has been used in several electro-mechanical analyses of the Gen-II IPEM [Boroyevich03][Pang02].

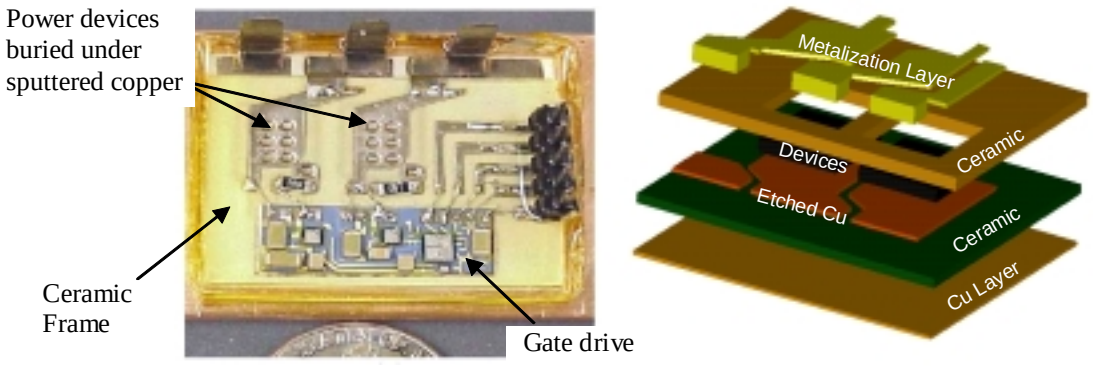


Figure 1.2. The 2nd Generation IPEM and a simplified model in explored view.

Through the modeling of this Gen-II IPEM, this thesis verified STEP AP210's capability for supporting the embedded power structure in Gen-II IPEMs [Liang01]. Gen-II IPEMs differ from other power electronics modules in the way the

interconnections to the power devices are constructed. The power devices are soldered onto a Direct Bonded Copper (DBC) board and buried in a ceramic frame with the terminal area exposed through via holes. The interconnection from the device terminals to other circuits are then created by depositing thin layers of copper onto the ceramic surface, the vias, and the device terminal pads. STEP AP210 provides a specific data structure (within the scope of the design of the interconnect module) for describing the physical realization of these interconnects and the network they implement.

This thesis uses the provided data structure to model the embedded power structure shown in Figure 1.2(b). The scope of this modeling includes the functional decomposition of the IPEM, the physical component layout and interconnection implementation, and the relationships between the functional and physical views. First, usage views of the IPEM and its constituents are provided, which define external views and interfaces to the module and the components. Second, a functional design view and a physical design view are defined for the IPEM, which is decomposed into material stack-up of the IPEM substrate, embedded devices, and interconnects. The mappings from the module terminals to the component terminals are then defined to provide explicit traceability from the outside of the IPEM to its inner details.

Finally, the relationships across the functional and the physical views are explicitly modeled to establish the constraints between the functional connectivity requirement and the physical topological implementation. A method is then developed based on these relationships to verify the model's consistency. The AP 210 model is populated with data and stored in a single file using the STEP Part 21 format. This model demonstrates the mapping from the concepts in the IPEM domain to the concepts in

STEP AP 210, and it provides guidance for future implementation of AP 210 to support IPEM design.

1.3 THESIS ORGANIZATION

The remainder of this thesis presents the details of the integrated software system and the implementation of a Gen-II IPEM model in STEP AP210. Specifically, it consists of the following:

Chapter 2 provides background information on STEP and AP 210. It first reviews the data exchange need for the IPEM design environment. Then it describes the basic structure of STEP and AP 210. Finally, it presents a general procedure for implementing STEP AP210, its current state of development, and the implementation software tools that are available.

Chapter 3 presents the integration software system for the multidisciplinary design and analysis of IPEM. It discusses the typical IPEM design process, the software tools, and the details of the integrated software system. The application of this integrated system is demonstrated through two parametric studies of an IPEM.

Chapter 4 presents the detail data structure of the Gen-II IPEM primitive model in STEP AP210. First, the mapping from the IPEM domain to AP 210 is discussed. Then, the modeling requirements of the embedded power technology in the IPEM and the supporting data structure in AP 210 are discussed. The AP 210 model is populated with data for the functional and the physical views of the IPEM and the relationships between these two views. Finally, a method to verify the consistency of the model connectivity data based on these relationships is presented.

Chapter 5 presents conclusions and gives suggestions for future work.

CHAPTER II

DATA SHARING AND OVERVIEW OF STEP AP 210

The exchange of product design information and analysis results between various engineering disciplines has become one of the most prevalent requirements in today's power electronics industry. The integrated IPEM design and analysis requires the sharing of many aspects of the product among different application tools, including 3D geometry, material properties, circuit parameters, and thermal performance. These data are either exchanged in standard formats or proprietary plain text files. As the scopes of the standards vary significantly from one discipline to another, some product data are lost during the exchange process.

This chapter reviews briefly the data sharing needs and the supporting data formats in the integrated power electronics design process. It then introduces STEP AP 210, its implementation process, and the current development in this area. It also provides a brief description about the tools that are used for STEP implementation and the documentation convention that is used in this thesis.

2.1 DATA SHARING IN THE INTEGRATED IPEM DESIGN

The integrated IPEM design and analysis process requires many different models of the IPEM. First, solid geometry models are needed to support the three-dimensional (3D) structures in IPEM device interconnects and packaging. These models can be of great detail for manufacturing purposes, or they can consist of simplified geometric features for thermal, electro-magnetic, or structural mechanics analyses. To avoid repeated modeling of the geometry and to reduce the chance of inconsistent geometric models, it is best to

have one detailed 3D model of the IPEM that is shared by various CAD/CAE systems, and from which the simplified geometry models for analysis can be generated via defeaturing.

The geometry model can be exchanged between CAD/CAE applications in many formats. For some applications, the exchange can be done in proprietary formats through direct translators between applications. For instance, the FLO/MCAD from Flomerics and Maxwell from Ansoft can import geometry from mechanical computer aided design (MCAD) software (such as Pro/ENGINEER, CATIA, or AutoCAD) directly into its native format without going through a neutral format translator. The advantage of this type of exchange is that the translation quality is usually reliable, and parametric features of the model are often preserved so that the imported geometry can be manipulated parametrically. However, this approach is too expensive to implement in general because for *N* CAD systems this would require *N*(N-1)* directly linked translators. Consequently, in most cases, such direct links do not exist. A widely accepted alternative is therefore to translate the geometry into neutral standard formats such as IGES or STEP AP 203.

The Initial Graphics Exchange Specification (IGES) is neutral exchange format for 2D or 3D CAD models, drawings, or graphics, and it is the first specification to establish information structures to be used for the digital representation and communication of product definition data [IGES96]. IGES data is organized as geometric and non-geometric entities, and it is represented in an application-independent ASCII clear-text format, to and from which the native representation of a specific CAD/CAM system can be mapped. IGES is widely supported by MCAD/ECAD systems for geometry exchange; though, a major drawback of this standard is the ambiguity of its

data definitions, which, in many cases, has caused discrepancies between the IGES translators from various systems. The interim solution to this problem is the use of IGES flavoring that specify how ambiguous data is interpreted by the IGES translator. As a result, a user must know in advance the target system that the exported geometry will be imported into and "tweak" the exporting translator accordingly to successfully exchange the geometry.

The STandard for the Exchange of Product model data (STEP) [ISO94-1] is a demonstrably superior technology to IGES. STEP is an international standard for product data representation and exchange and is designated as ISO 10303. Its purpose is to provide a common form for unambiguous representation and exchange of product data among application processes, throughout the life cycle of a product. Figure 2.1 illustrates the business use of STEP. STEP recognizes that the representational needs of various domains differ, and it therefore provides a series of application protocols (APs) to describe the life cycle context in which the product data was created, and for which discipline the data is applicable. The currently most implemented AP is STEP AP 203 [ISO94-2], which addresses the exchange of configuration controlled 3D product definition data in the mechanical engineering world.

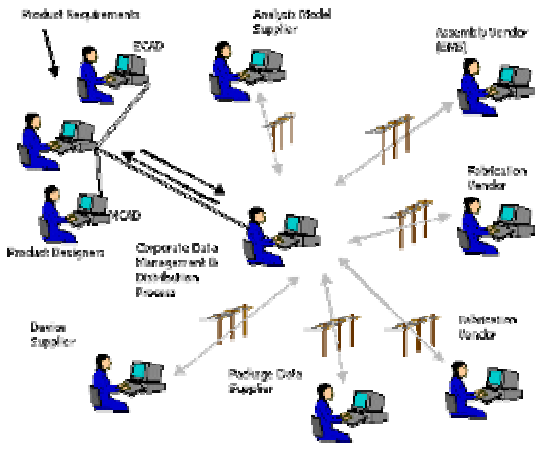


Figure 2.1 Business use of STEP [PDES01-1].

STEP AP 203 is supported by the majority of MCAD systems as well as some ECAD systems, such as Maxwell Q3D that rely on 3D modeling. Data that can be exchanged include 3D part geometries, assemblies and bills of material, and design materials, all of which can be exported and imported as AP 203 entities described in ASCII clear text files using the STEP Part 21 format [ISO94-3]. Our experience with STEP AP 203 translators supports the claim that STEP AP 203 provides a more reliable geometry translation than IGES does.

The integrated IPEM design process also requires the modeling and sharing of the IPEM's electrical characteristics among the applications. The electro-magnetic analysis of the IPEM generates inductance and capacitance matrixes for the IPEM interconnects and packaging. These data are used to construct a simulation model for the IPEM, which is used in a testing circuit in a circuit simulation tool to evaluate the IPEM's performance. These data are stored in a vendor-proprietary format, such as Saber MAST for simulation, or in some commonly used formats, such as SPICE netlist for sharing. Although STEP AP 203 does not support the description of electrical characteristics, it is within the scope of STEP AP 210 [ISO01] to address this need in the context of data exchange of electro-mechanical products between MCAD and ECAD systems.

2.2 SCOPE OF STEP AP 210

STEP Application Protocol 210 (AP 210) has been developed to support the data representation and exchange of product data for electronic assembly, interconnection, and packaging design. The scope of AP 210 is quite broad: It spans from the definition of devices and parts, to that of printed circuit boards (PCBs) and assemblies (PCAs); from the device functionality and port definition, to the assembly functional network design decomposition; from the functional requirements, to the physical implementations; and from analysis models, to manufacturing planning data (Figure 2.2).

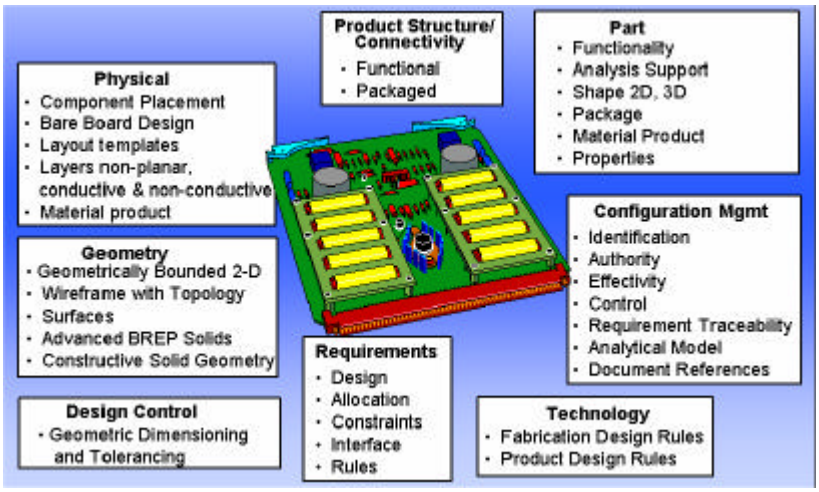


Figure 2.2 Scope of AP 210 [PDES01-1].

On the ECAD side, AP 210 has the capability to support all the data that is described using the Gerber, IDF, GenCAM, ODB++, and EDIF file formats; while on the MCAD side, it takes AP 203 as a subset to support the sharing of the 3D part and assembly data. More importantly, AP 210 facilitates electrical and mechanical collaborative design of complex products by providing detailed associativity between views, which is exactly what is needed to build relationships between models in an integrated IPEM design model.

Due to the broad scope of AP 210 (it contains more than 900 individual concepts and 95 higher-level units of functionality), AP 210 is a much more complex standard than AP 203, which defines 41 individual concepts and 14 units of functionality. Consequently, the implementation of AP 210 requires more attention to the mapping between the concepts in the target system and the concepts in AP 210. The next section reviews the basic structure and implementation considerations of STEP AP 210.

2.3 STEP STRUCTURE AND IMPLEMENTATION OF AP 210

In addition to the STEP Application Protocols (APs) that define the context, the scope and the related STEP resources for a designated application area, there are other STEP Parts that, together with APs, form the complete STEP standard [ISO94-1]. These Parts are divided into Description Methods, Integrated Resources, Implementation Methods, and Conformance Tools (Figure 2.3).

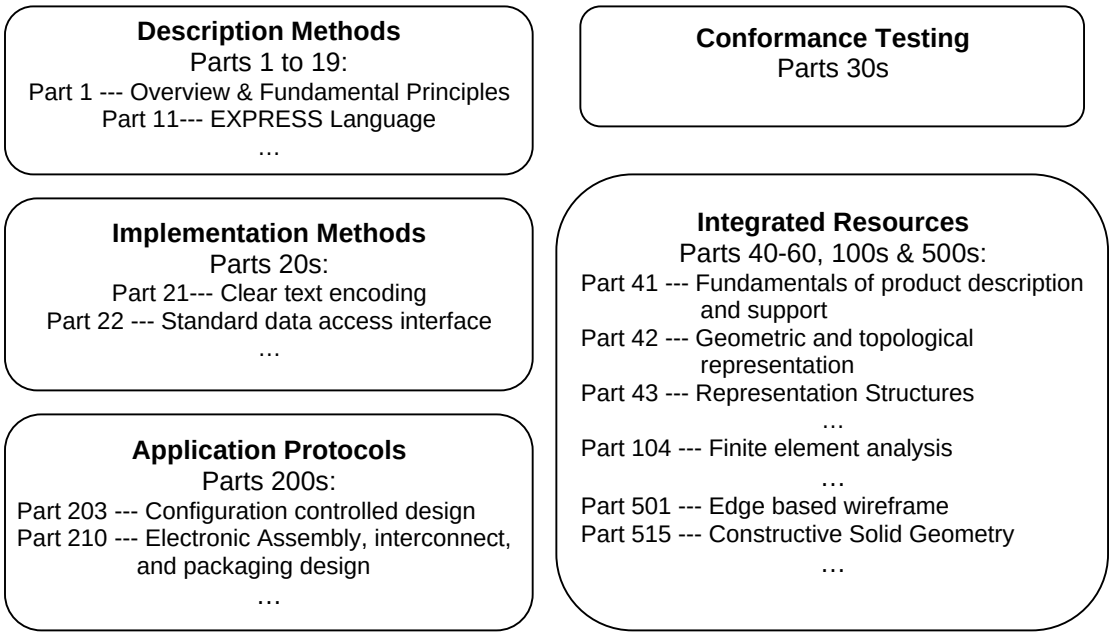


Figure 2.3 Parts of STEP are divided into five categories.

The Description Methods (Parts 1~19) define such items as the EXPRESS language [ISO94-4] as the fundamental description method of STEP. EXPRESS is an object-flavored information-model specification language, and it enables the writing of formal information models that are understandable by both humans and computers. EXPRESS is the formal language that is used to describe the data structures and constraints in the information models in STEP. For documentation purpose, this thesis uses EXPRESS-G, which is the formal diagrammatic form of EXPRESS language, to represent the data structures in AP 210 that are used for IPEM modeling.

The Integrated Resources in STEP consist of resource constructs that are product data descriptions written in EXPRESS. These resource constructs are considered the basic building blocks of STEP and are divided into two groups: integrated generic resources, which consist of generic entities that are used as needed by the APs across the entire spectrum of STEP; and integrated application resources, which contain entities that have slightly more context than the generic entities. An AP is developed by selecting the appropriate resource constructs from among the integrated resources, and placing additional constraints on them to meet the product information requirements for the specific application domain. This process is known as the interpretation of the integrated resources.

Each Application Protocol has two separate data models. One is the Application Reference Model (ARM), which describes the product information requirements of the specific application domain; and the other is the Application Interpreted Model (AIM), which results from the interpretation of the integrated resources mentioned above. The ARM is formulated using domain terminologies on the basis of extensive research into

the activities common to the domain, while the AIM describes the data using STEP-terminology names and structures.

In the ARM, the smallest information unit is the Application Object (AO), which represents a unique concept in the application domain that need to be described. AOs are grouped as Units of Functionality (UoFs) to form all the data that is needed for exchanging information within the application context. Based on the UoF that it falls into, an AO is mapped to one or more entities in the AIM through the mapping table contained in the AP. These AIM entities fulfill the information requirements defined by the given AO.

In general, the AIM of an AP is described using EXPRESS, and the ARM is described using descriptive text. AP 210, however, provides an informative EXPRESS model for its ARM in addition to the normative text description. Hence, while there are a few constraints and rules between the AOs that are missing from the ARM EXPRESS, the ARM EXPRESS provides a complete structural definition of the ARM. Implementing a product model using this ARM EXPRESS therefore helps in understanding the mapping between the AOs and attributes in the ARM, and the entities and attributes used in the target system.

Therefore, to examine AP 210's ability to model an IPEM, and to understand the mapping from the IPEM applications to the AP 210 ARM, this thesis implements an IPEM model using the data schema defined in AP 210 ARM EXPRESS.

STEP provides two forms of implementations. One is the STEP physical files defined by Part 21 [ISO94-3], the Clear Text Encoding of the Exchange Structure. In this form, the STEP file is populated with product data according to the AP's EXPRESS

entity definitions, and is formatted as described in Part 21. This is the commonly used method to exchange product data between two or more application systems (e.g., STEP AP 203 translators in many CAD systems). The other form is a shared database, which can be accessed by client systems through the Standard Data Access Interface (SDAI) [ISO94-5]. This form is more efficient when the product data is shared concurrently rather than exchanged among numerous applications. This thesis uses the first form, i.e., the STEP physical file, to store IPEM data according to the AP 210 ARM EXPRESS.

2.4 CURRENT DEVELOPMENTS IN AP 210 IMPLEMENTATION

AP 210 was published as international standard in 2001. A number of implementations have been conducted to validate the concepts and mature the standard. A list of AP 210 examples that focus on printed circuit board (PCB) layout descriptions can be found at http://www.ap210.org. They provide first-hand experiences on the implementation of AP 210 models. Several projects were conducted to investigate the data integration and tool developments in AP 210. One of these projects is the Electro-Mechanical (EM) Pilot project conducted by PDES to share STEP data between electrical and mechanical designers during the design process for printed circuit assemblies [PDES01-3]. The data exchanged were primarily physical in nature, such as board outlines, keep-out areas, and component height restrictions to prevent interference. The pilot assisted in validating AP 210 and providing usage guidance for production implementations. A prototype Intermediate Data Format (IDF) to STEP AP 210 translator, and an AP 210 viewer were developed as a result. Rockwell Collins and Boeing also initiated a similar project to evaluate the feasibility of exchanging PCA/PCB data between their two organizations using AP 210 [PDES01-1].

Commercial translators for AP 210 are in development. Currently, there are bi-directional translators available between AP 210 and three PCB CAD systems: Mentor Graphics/Board Station, Zuken/Visula (CADIF), and CadSoft EAGLE [PDES03]. Data that are translated are primarily PCB/PCA layout data, assembly information, graphics, and product data management (PDM) information. There are two prototype functional netlist translators available for online testing: EDIF and VHDL [PDES03]. Currently, only structural data are supported by these two translators. Translation of graphics in EDIF, or behavior functions in VHDL, is not supported. Also available are uni-directional translators from IDF 2.0 and CircuitCAM to AP 210 [PDES03].

2.5 OBSERVATIONS

The STEP AP 210 was initially developed for the design of PCB/PCA, though its scope covers most multidisciplinary aspects of electro-mechanical products. The IPEM products different from general PCBs/PCAs in the way power devices are packaged and connected using embedded power technology. It is therefore necessary to verify that AP 210 can support IPEM modeling: This thesis examines AP 210's support for IPEM modeling by studying the mapping from the IPEM domain to AP 210 ARM. Specifically, the following approach is pursued in this thesis:

1. Describe the embedded power structure in the Generation-II IPEMs using the AP 210 ARM EXPRESS. This demonstrates the mapping from the IPEM and its embedded power technology to the AP 210 ARM;
2. Create a functional view of the IPEM which describes the functional connectivity requirement between power devices;

3. Create a physical view of the IPEM which describes the physical features of the interconnects that implements the functional connectivity requirement; and
4. Model the relationship between the functional view and the physical view of the IPEM, and develop a method to validate the consistency of connectivity data in both views.

2.6 TOOLS AND CONVENTIONS

Due to the size and complexity of STEP AP 210, the implementation of an IPEM model in AP 210 ARM requires a tool that can process the EXPRESS schema. The most popular tool in the public domain is the Express Engine (previously call Expresso), which was initially developed at the National Institute for Standards and Technology (NIST) by Peter Denno, and is now maintained on SourceForge.net by Craig Lanning [Lanning03]. Commercial tools such as STEP Book from LKSoft and ST-Developer from STEP-Tools are also available for browsing EXPRESS schema and data populations. A complete list of EXPRESS tools and services can be found at the NIST website http://www.nist.gov/sc4/tools/express/etools98.htm. This thesis uses the EXPRESS Engine to parse the AP 210 ARM EXPRESS, and to populate and validate IPEM model data.

The ARM data structure selected for IPEM modeling is documented in this thesis using EXPRESS-G, and the IPEM data population is stored in a STEP physical file using Part 21 file format. The EXPRESS-G symbols and the conventions used for the data population diagrams in this thesis, are shown in Figures 2.4 and 2.5, respectively.

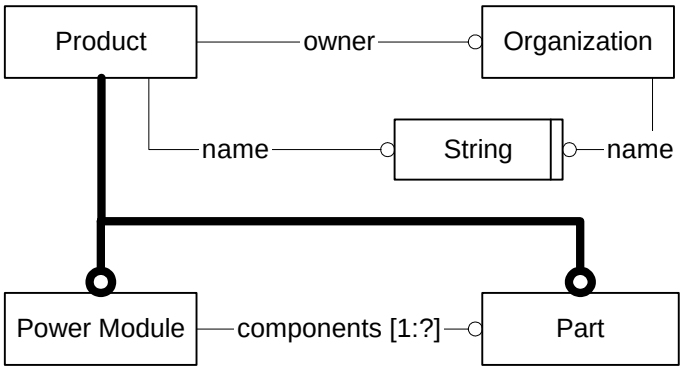


Figure 2.4 A sample model described using an EXPRESS-G diagram.

A solid rectangle in EXPRESS-G represents an entity that is defined in the model, while a solid rectangle with a double edge on the right-hand side is a predefined simple type (such as ‘real’ and ‘string’) in the EXPRESS language. Thick lines represent supertype-subtype relationships, and normal lines represent entity-attribute relationships. The subtype and the attribute are connected to the end with an open circle. Text among the line indicates the name and cardinality of the attribute. In Figure 2.4, entities *Power Module* and *Part* are subtypes of *Product*; and each *Product* has an attribute *name*, which is of predefined type *String*, and an attribute *owner*, which is of entity type *Organization*. An *Organization* also has a *name*, and a *Power Module* has one or more *Parts* as its components. Note that this model is only for demonstration of EXPRESS-G and does not represent the real data structure in STEP.

```
#1=ORGANIZATION('CPES');
#2=ORGANIZATION('Unknown');
#3=PART('Part 1', #2);
#4=PART('Part 2', #2);
#5=POWER_MODULE('IPEM',#1,(#2,#3));
```

Part 21 file

Population Diagram

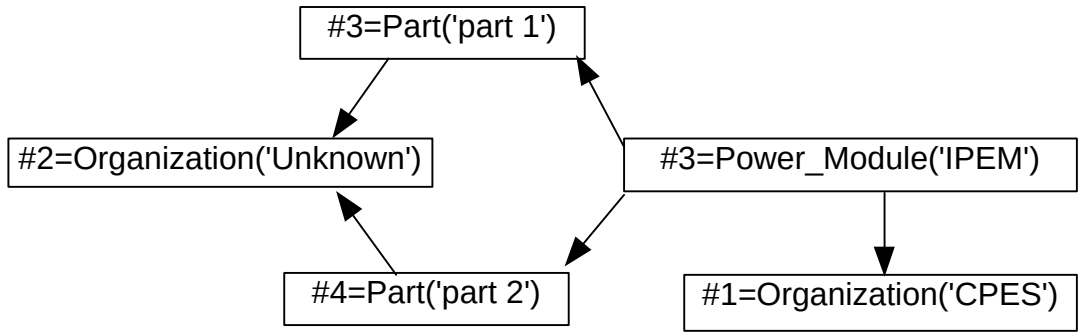


Figure 2.5 Sample data in a STEP Part 21 file, and the corresponding population diagram.

Figure 2.5 shows a data population according to the EXPRESS-G model in Figure 2.4: A *Power Module* owned by *Organization* CPES is populated as instance #5. It consists of two *Parts* (instance #3 & #4) that are owned by an unknown *Organization* (instance #2). The purpose of the population diagram is to show the dependency between the data instances. Note that the instance values in the diagram are only for documentation purposes and are not the real values that are used for the data population in the STEP Part 21 file. Readers should refer to the Part 21 file in the Appendix for the actual instance values of the IPEM model discussed in Chapter 4.

CHAPTER III

SOFTWARE INTEGRATION FOR MULTIDISCIPLINARY POWER ELECTRONICS DESIGN

The integrated software system for power electronics design presented in this thesis uses four commercial software tools: iSIGHT, I-DEAS, Maxwell, and Saber. The operation of each tool is instructed via its respective application programming interface (API). This allows each tool to be run in batch mode. The data is passed between the tools using STEP AP 203 to describe the power module geometry, PSpice to describe the circuit parameters, and plain text files to describe the design variables and analysis results. The remainder of this chapter details the configuration of each tool and the data flow within this integrated system, and is presented in journal manuscript format.

SOFTWARE INTEGRATION FOR MULTIDISCIPLINARY POWER ELECTRONICS DESIGN

Yingxiang Wu, Jonah Chen, Ying-Feng Pang,

Jan Helge Bøhn, Dushan Boroyevich, and Elaine P. Scott

Center for Power Electronics Systems
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA

Abstract

This paper presents a software integration system for automated multidisciplinary design and analysis in power electronics. This system is based on the commercial integration tool iSIGHT, and it integrates the three commercial software tools I-DEAS, Maxwell, and Saber. An in-house tool has been developed to provide a generic interface to automated thermal parametric studies in I-DEAS, and to minimize the configuration effort required for each new study. To demonstrate the flexibility and efficiency of this system, two parametric studies are presented for the design of integrated power electronics modules (IPEMs).

3.1 INTRODUCTION

Designing power electronics systems is a complex process and requires interactions between many engineering domains: electrical circuits, electro-magnetics, thermal dynamics, and material and structure mechanics. Today's computer aided design (CAD) tools for power electronics design enable complex analyses at unprecedented speeds and incorporate enormous amounts of knowledge and empirically verified expertise within their disciplines. Although the use of these tools has significantly improved the design and reduced the prototyping and testing cost, these CAD tools are usually mono-disciplinary and lack the ability to communicate with each other directly. Coordination and data sharing between them is often labor intensive and time consuming. As a result, today's design process for power electronics systems is still much less automated than in many other industries.

The Center for Power Electronics Systems (CPES) has proposed an integrated design methodology to improve the multidisciplinary aspect of the design process [Borojevich03]. It streamlines the design process in power electronics by integrating existing commercial CAD systems to perform multidisciplinary modeling and system optimization. Figure 3.1 illustrates the overall concept of this integrated design methodology. Instead of using only algebraic modeling of the multidisciplinary aspects [Boattini98], this new methodology relies on models at various levels of abstraction for multidisciplinary analyses. The CAD tools involved in each discipline are integrated so that data can be shared electronically and the tools' operations can be controlled and automated by an external system-optimization tool.

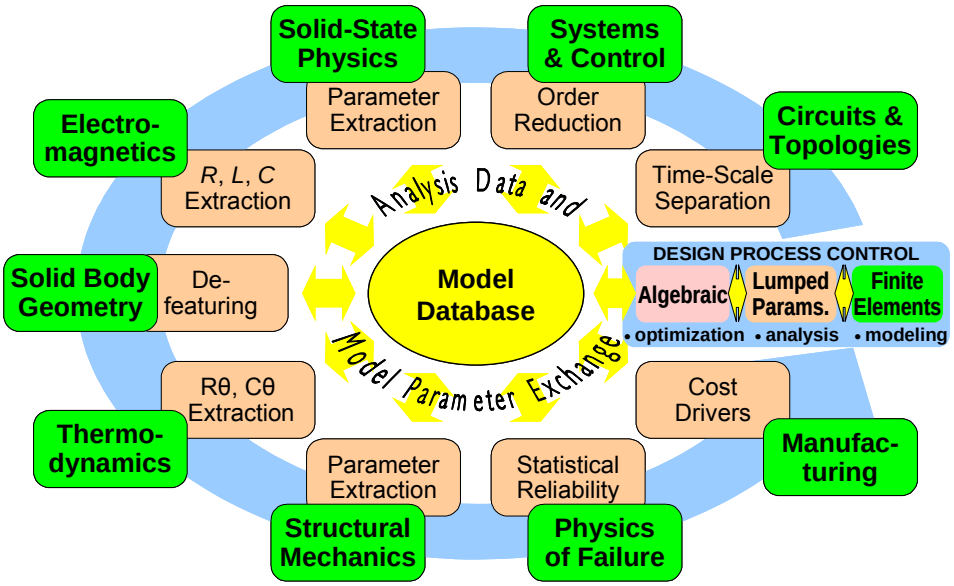


Figure 3.1. Schematic representation of the integrated design methodology: The outer boxes describe the disciplines, while the inner boxes described their reduced levels of abstraction, which are stored in the model database and processed by the design process control system.

This paper presents the software integration system that was implemented to support this integrated methodology. The flexibility and efficiency of this system are demonstrated through its application to two parametric studies of an integrated power electronics module (IPEM).

3.2 SOFTWARE INTEGRATION SYSTEM

Of the many components illustrated in Figure 3.1, four have been implemented in part to support the design of a typical power electronics module. These are the solid body modeling, electromagnetic parameter extraction, lumped parameter circuit simulation, and finite element thermal analysis [Chen01]. Figure 3.2 illustrates the integrated design system and its data flow. The system accommodates two important multidisciplinary interactions facing the design of power electronics modules: First, it accommodates sequential multidisciplinary interactions, such as how the geometric structure and materials work together to determine both the thermal performance and the structural parasitic impedance—which in turn affects the module's

electrical behavior. The outer loop in Figure 3.2 models this sequential multidisciplinary interaction. Second, it accommodates multidisciplinary co-dependencies, such as how the thermal and electrical behaviors affect one another. The electrical characteristics of the device depend on its temperature distribution; at the same time, the heat generated depends on the power loss. The inner loop in Figure 3.2 models this multidisciplinary co-dependency, which must be computed iteratively for any given geometric configuration until the results converge.

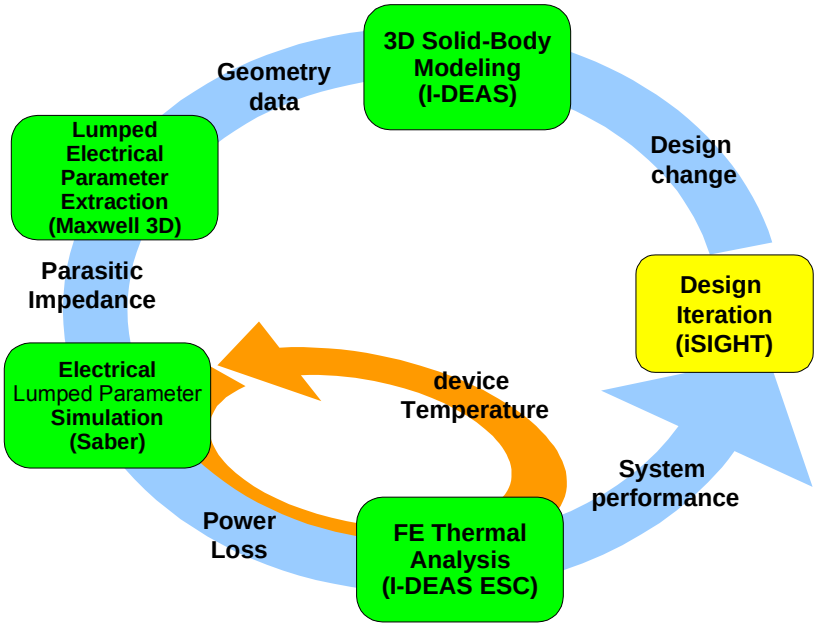


Figure 3.2. Data flow within the IPEM integrated design and analysis system.

The integrated software system is based on a set of commercial software systems that were primarily selected for their discipline-specific modeling and analysis capabilities, and for their widespread acceptance within the power electronics industry. They include I-DEAS [SDRC1] for geometric modeling, I-DEAS Electrical System Cooling (ESC) for thermal analysis, Maxwell [Ansoft] for the parasitic parameter extraction, and Saber [Avanti] for electrical circuit simulation. A supplementary selection factor was the availability of an application programming interface (API). For instance, I-DEAS ESC was chosen over Flotherm [Flomerics] because the

latter did not provide an API or another means to operate it in batch mode. These programs are tied together using the commercial integration tool iSIGHT [Engineous], to manage the design variables and the analysis process, be it for parametric studies, or for design optimization. iSIGHT controls the process by controlling the batch execution of a program, by manipulating the batch process input file, and by monitoring the corresponding output file.

To accommodate iSIGHT, each commercial software system is wrapped using a three-component structure consisting of the program controller, the input data, and the output data (Figure 3.3). The program controller is a code that interfaces with the commercial software system via its API, to control its operation during batch mode execution. The input and output data are both plain text files with a fixed format, so that iSIGHT can manipulate the program input and monitor the analysis results. The purpose of these two files is to facilitate communication with iSIGHT and not store the design data itself, such as the parametric CAD model, which might be described using I-DEAS's proprietary, binary file format. The following four subsections will therefore first describe the iSIGHT integration system, followed by its interfaces to I-DEAS, Maxwell, and Saber, respectively.

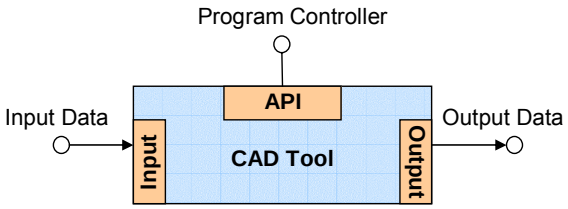


Figure 3.3. The three-component wrapper for each commercial software system (CAD Tool).

3.2.1 Process integration using iSIGHT

The IPEM design process depicted in Figure 3.2 can be easily modeled using iSIGHT's process integration functionality. Figure 3.4 shows the generic IPEM integration structure that was

created within iSIGHT to automate the IPEM design, modeling, and analysis process. The outer loop of the IPEM design process is modeled as a sequence of iSIGHT operations: (1) *ModifyGeometry*, (2) *ExtractParasiticImpedence*, and (3) a conditional while-loop, which models the inner loop of the design process, and which consists of the two operations (4) *CircuitSimulation* and (5) *ThermalSimulation*. Each operation has a set of input and/or output variables that are defined in and managed by iSIGHT. The condition to end the while-loop is that the values of output variables from *CircuitSimulation* and *ThermalSimulation* each converge respectively. This can be easily implemented by checking the difference in results between two consecutive iterations. For instance, a temperature change of less than 1^{o}C in the target device between two iterations is generally considered to signify convergence.

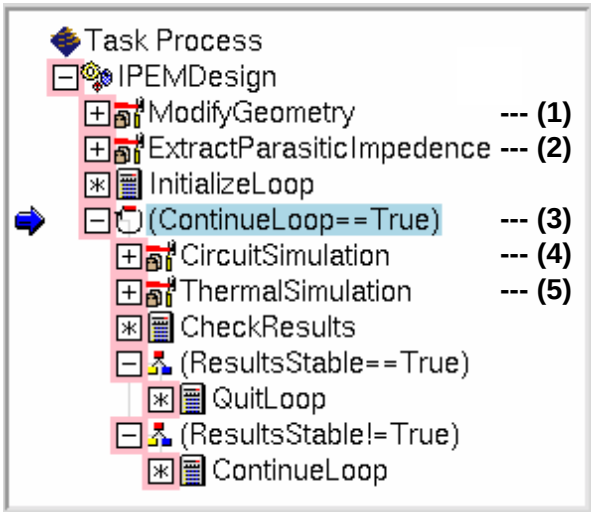


Figure 3.4. IPEM design process integration in iSIGHT consists of four operations (1,2,4,5) and one conditional loop (3).

3.2.2 Interface to I-DEAS geometry manipulation and thermal simulation

I-DEAS provides two methods to control its operation during batch mode: One is by using the I-DEAS Open Language, and the other is by using Open I-DEAS. The I-DEAS Open Language is based on scripts that contain macros, and it is suitable for applications that require simple

programming constructs [SDRC2][Osterberg95][El-Ghany00]. Open I-DEAS, on the other hand, is a full C++ application program interface (API) that allows custom programs to communicate directly with I-DEAS during runtime at the code level. It is suitable for applications that require more flexibility [SDRC3][Yoon00]. Due to the complex operations inside I-DEAS (e.g., parametric modeling, finite element meshing, thermal simulation, and post-processing), it is impractical to write generic I-DEAS macros that are reusable for topologically different geometries—especially given the non-trivial content and syntax of the I-DEAS macro files. The preferred approach is therefore to develop a program controller using C++ and the Open I-DEAS API.

A program controller using the Open I-DEAS API was therefore developed to encapsulate a set of basic operations that are needed for iterative geometry change and thermal simulation during an iSIGHT-driven optimization or parametric study. The following operations were identified to be meaningful in this regard:

- *Changing part dimensions*
- *Changing component positions*
- *Updating finite element model*
- *Exporting geometry via a STEP AP 203 file*
- *Modifying boundary conditions*
- *Generating temperature reports for given devices*

Each of these operations is represented as a separate line in a command input text file that is read by the program controller—with each line consisting of an operation id, a part or assembly name, a parameter name and value pair, and certain additional options (Figure 3.5).

```
# input parameter file
#
# Operation Part/FEM     Parameter        Parameter   Update
# ID (0-5)  Name(string) Name(string)     Value(real) FEM(0/1)
#-----------------------------------------------------------
1  FEMpart    HeatSpd_depth   0.003   0   #change dimension without FEM update
1  FEMpart    HeatSpd_Length  0.035   1   #change dimension with FEM update
2  Assembly1  HeatSpd_x       0.002       #change part assembly position
3  FEMpart    Device_power    1           #change boundary condition
4  Assembly1  HeatSpread      HeatSpd.stp #export part via STEP AP 203 file
5  FE_dir     2   Device1   Device2       #generate ESC reports for two devices
0                                         #end
```

Figure 3.5 Input text file for the generic program controller with rigid coupling to I-DEAS.

iSIGHT performs optimizations or parametric studies by reading this trivially formatted command file, manipulating its parameter values, and then initiating an iteration with I-DEAS by launching the program controller based on the current state of this command file. This process requires the manual generation of the initial command file and the corresponding I-DEAS model file. Once launched by iSIGHT, the program controller reads the command file and carries out the requested operations according to the flowchart shown in Figure 3.6. Then, depending on the operations requested in the command file, a number of output files are generated that can be parsed by iSIGHT to extract certain output values of interest. Figure 3.7 illustrates the generic setup.

As noted in Figures 3.2 and 3.4, there are two separate cases within the iSIGHT-driven process that require the launching of I DEAS: One is to modify the part geometry; the other is to perform a thermal simulation. Each of these two cases requires its own, separate input command file, and generates its own, separate set of output files.

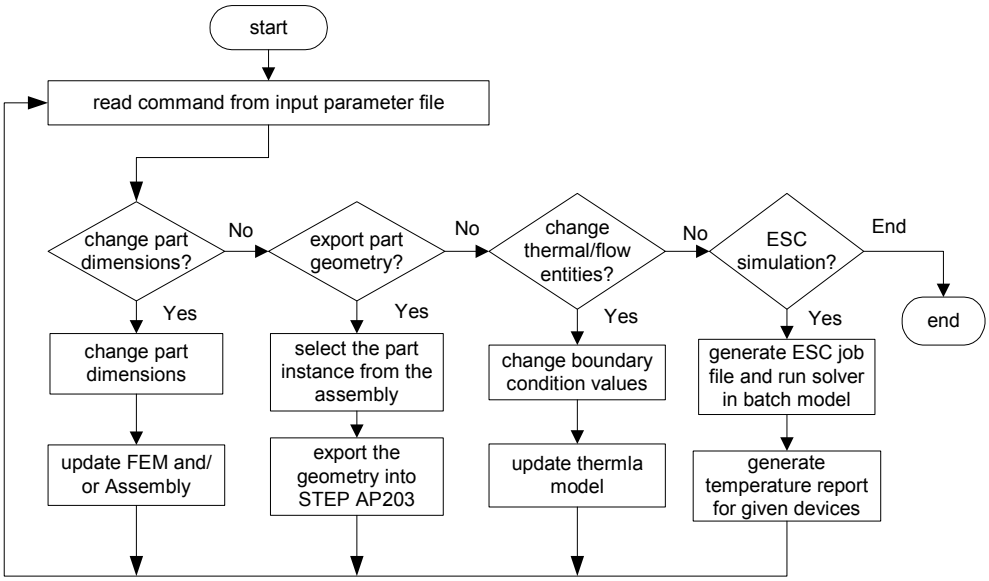


Figure 3.6. Flowchart for the generic program controller with rigid coupling to I-DEAS.

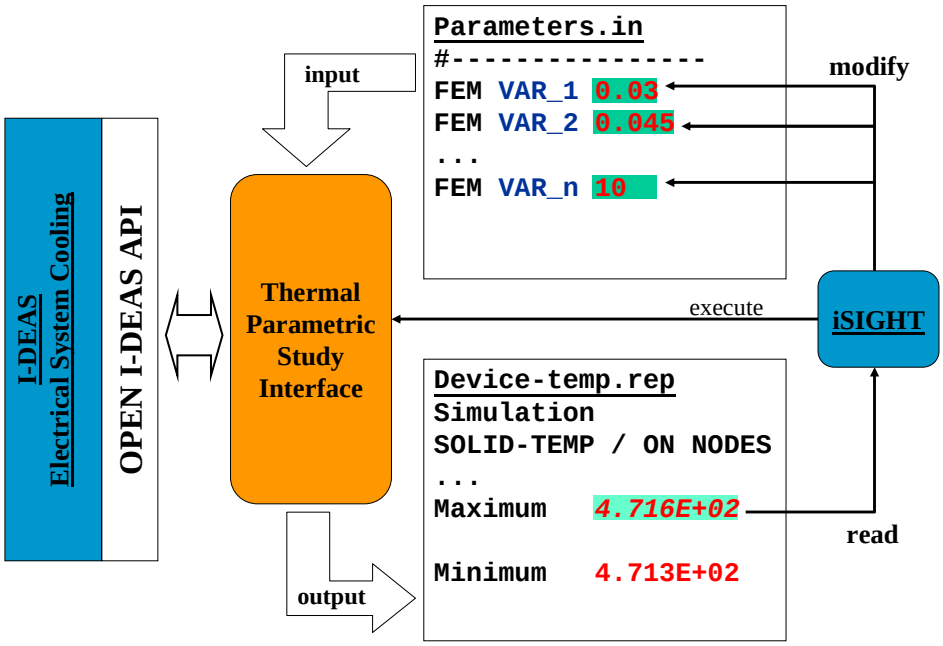


Figure 3.7. Interface to I-DEAS geometry manipulation and thermal simulation.

In the case of the iSIGHT operation *ModifyGeometry* (Figure 3.4), the input command file contains (A) the name-value pairs of the geometric parameters that iSIGHT wants to have

changed, and (B) the names of the parts that are to be exported via a STEP AP 203 part file to Maxwell for electromagnetic parameter extraction. iSIGHT determines the values of these geometric parameters based on the optimization or Design of Experiment (DOE) plan that the user has chosen. It does not, however, process the output files. Instead, it merely expects that the updated part geometry is described in a STEP AP 203 part file, which is available for subsequent use by Maxwell.

In the case of the iSIGHT operation *ThermalSimulation* (Figure 3.4), the input command file contains (A) the name-value pairs of those device power dissipations whose value depends on the results from Saber, and (B) the names of the devices whose temperature are of interest. The outputs from this operation are temperature report files that are generated by I-DEAS for the devices specified. iSIGHT parses these files to identify the maximum temperature for each device.

3.2.3 Interface to Maxwell parameter extraction

Maxwell, which is used for parasitic parameter extraction, provides only a macro-based interface for executing in batch mode. Its iSIGHT program controller therefore takes the form of a script file that contains the Maxwell macro commands for importing geometry, performing parameter extraction, and storing the parasitic impedance matrix in an output text file. iSIGHT passes this script file to Maxwell each time it launches Maxwell. Since the only changes from one iSIGHT iteration to the next, are the changes in the part geometry as described in the updated STEP AP 203 part file, which is exported by I-DEAS, there is no need for iSIGHT to modify this input script file. The input script file therefore remains static across each iSIGHT optimization or parametric study. The output from Maxwell is a text file, from which iSIGHT can extract the parasitic impedance values associated with the current part geometry.

3.2.4 Interface to Saber circuit simulation

Saber, which is used for circuit analysis, also provides only a macro-based interface for executing in batch mode. The circuit analysis is processed in two steps: First, a circuit simulation is performed using *Saber Sketch*; then a performance measurement is performed using *Saber Scope*. iSIGHT therefore makes use of two distinct input script files; one for Saber Sketch, and one for Saber Scope. These script files contain the Saber macro commands for setting up the analysis, measuring the performance, and calculating and recording the device power dissipation into a text file. iSIGHT passes these two script files to Saber each time it launches Saber to execute Sketch and Scope, respectively.

In addition to these two Saber input scripts, it is necessary to provide Saber with a circuit netlist that includes the current parasitic impedance values and the device temperatures. This text file must first be generated manually in preparation for each iSIGHT optimization or parametric study. iSIGHT will then, at each iteration, update this file with the current parasitic impedance values from Maxwell and the current device temperatures from I-DEAS. Next, it will launch Saber Sketch, followed by Saber Scope. The latter generates a text file that iSIGHT can parse to extract the device power dissipation.

3.3 PARAMETRIC MODELING AND GEOMETRY TRANSLATION

The preceding four subsections illustrate how iSIGHT can be used to integrate commercial software tools, provided that iSIGHT can (A) modify the input parameters to the tool as it runs in batch mode, and (B) read the resulting output (Figure 3.7). The user then needs to provide iSIGHT with the value range for the design parameters, the design goals, the optimization or DOE plan to be used, and a starting point in the form of a set of initial input files that are referred to as *templates*. The structure of these templates must be specified such that they are consistent

with the data they operate on. The following will discuss this last issue in more detail, both with regards to defining a parametric model in I-DEAS and sharing this geometric data with Maxwell.

iSIGHT can easily make changes to an I-DEAS solid model if this model is parametrically defined. In contrast, making changes to a non-parameterized model via iSIGHT is difficult at best. To use a parameterized model with iSIGHT, it is extremely important that the model is fully constrained. This is necessary to ensure that any changes to the driving parameters that are identified to iSIGHT, result in geometric changes as intended. In essence, this parameterization captures, to a large extent, the designer's intent. Hence, it is sound practice for the designer to verify the correctness of this capture of intent by manually modifying the parametric variables and confirming the resulting geometry.

Since the I-DEAS solid model will be used as the basis for finite element analysis (FEA), it is likewise important that the boundary conditions that will used for the FEA are fully defined in a finite element model that is fully associative with the actual solid model part geometry and not in an externally derived finite element model. This is because only a fully associative finite element model will retain these boundary conditions when the underlying part geometries change. In contrast, with an externally derived finite element model, the boundary conditions will be marked as obsolete once geometric changes are made, and thus require a manual intervention to revalidate these boundary conditions.

Hence, the above suggests the following steps be followed to manually create an initial parameterized solid model within I-DEAS for use with iSIGHT:

1. Create a fully constrained parametric solid model for each part of the power module. Note that the models' tolerances must be set to 1.0×10^{-6} meter to be consistent with the Maxwell modeler.

2. Create a finite element model for each part of the power module. Assign a unique name to each device-material combination, even if several devices use the same material. This will ensure that the temperature of each device is reported separately.
3. Create the power module assembly, and specify the constraints between the parts.
4. Create a finite element model from the assembly with the part history information included. This will create a new part that represents the module assembly. Check the history tree of this new part, and, if needed, add relations between features to fully constraint its geometry.
5. Define the boundary conditions based on the solid model and not on the finite element model.
6. Identify the parameters and boundary conditions that will be subject to change by iSIGHT, and record these variables in the command input text file as discussed in Section 3.2.2 and illustrated in Figure 3.5.

Once this parameterized solid model has been described within I-DEAS, the model needs to be exported to Maxwell for parasitic parameter extraction. Currently, the most robust approach is to translate these models from I-DEAS to Maxwell via files described using the STEP AP 203 file format; and, in doing so, translating each part-geometry separately instead of as an assembly. It is necessary to translate these part geometries individually because the Maxwell STEP translator discards any component identification information stored in the STEP file. With an assembly, the correspondence between the parts and their assigned names would then be lost, and thus make it impossible to match up the part geometries in I-DEAS with those in Maxwell. However, when the parts are translated individually, the relationship between a part geometry and its assigned name can be preserved by matching the names specified for geometry

import in the Maxwell input script file to the names specified for geometry export in the I-DEAS command input file. This way, the part geometries can be translated as an intact system of parts from I-DEAS into Maxwell for parasitic parameter extraction.

3.4 TWO CASE STUDIES

The integration system for power electronics design has been used for a number of IPEM design parametric studies. This section presents two case studies to demonstrate the efficiency of this integrated system: Case study (a) examines the effect of changing the thickness of the ceramic layer in Direct Bonded Copper (DBC) board on the electrical and thermal performance of the IPEM (Figure 3.8 (a)); whereas case study (b) examines the effect of reducing the area of the DBC copper trace (Figure 3.8 (b)). Additional case studies concerned with IPEM designs using the integrated power electronics design system can be found in [Chen01] [Pang02].

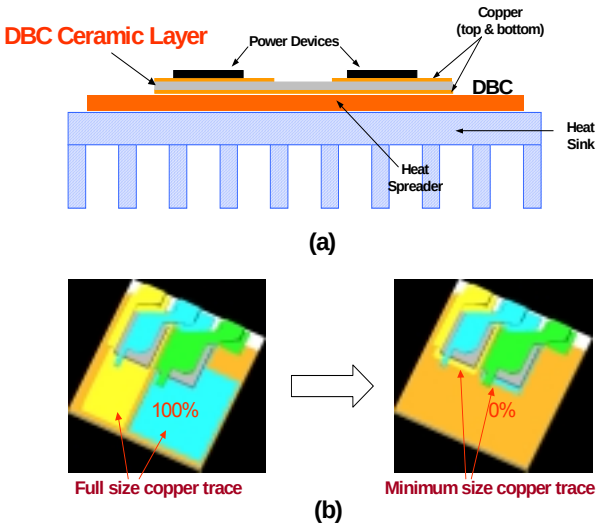


Figure 3.8. Two case studies concerning the effects on IPEM performance when (a) changing the thickness of DBC ceramic layer; and (b) reducing the area of the DBC copper trace.

In all these cases, the integration system was configured within iSIGHT as shown in Figure 3.4, following these four steps: (1) define the design variables and the analysis output in iSIGHT; (2) prepare the input templates and the output files for each operation in iSIGHT and instruct iSIGHT on how to modify the input templates and the read output files; (3) define the termination condition for the while-loop; and (4) select a parametric-study plan in iSIGHT.

For the two cases presented here, the while-loop termination condition was defined as a temperature difference between two consecutive runs of less than 1 ºC per device; and the parametric studies were controlled using the iSIGHT Design of Experiment (DOE) plan. Table 3.1 shows the variables used within iSIGHT for each of these two case studies.

Table 3.1. iSIGHT variables for IPEM parametric study cases (a) and (b)

Variables	Design Input	Analysis Output	Auxiliary Variables
Case study (a)	Ceramic_thickness	Parasitic_Impedance_Matrix Device1_ Calculated_Temperature Device2_ Calculated_Temperature Device1_Power_Loss Device2_Power_Loss Peak_Common_Mode_Current	Inner_Loop_Counter Continue_Loop Device1_Initial_Temperature Device1_Temperature_Difference Device2_Initial_Temperature Device2_Temperature_Difference
Case study (b)	Area_factor		

Figure 3.9 shows an annotated set of partial snapshots extracted from the input templates and output files that were used for case study (a), which was the study of the effect of changing the thickness of the DBC ceramic layer. In particular, it illustrates the data flow during an iSIGHT iteration:

Figure 3.9, step (a) — The initial value of the design variable, *Ceramic_thickness*, is specified during the selection of the DOE plan. During subsequent iSIGHT iterations subject to this plan, iSIGHT modifies the corresponding field in the input file for the operation *ModifyGeometry*.

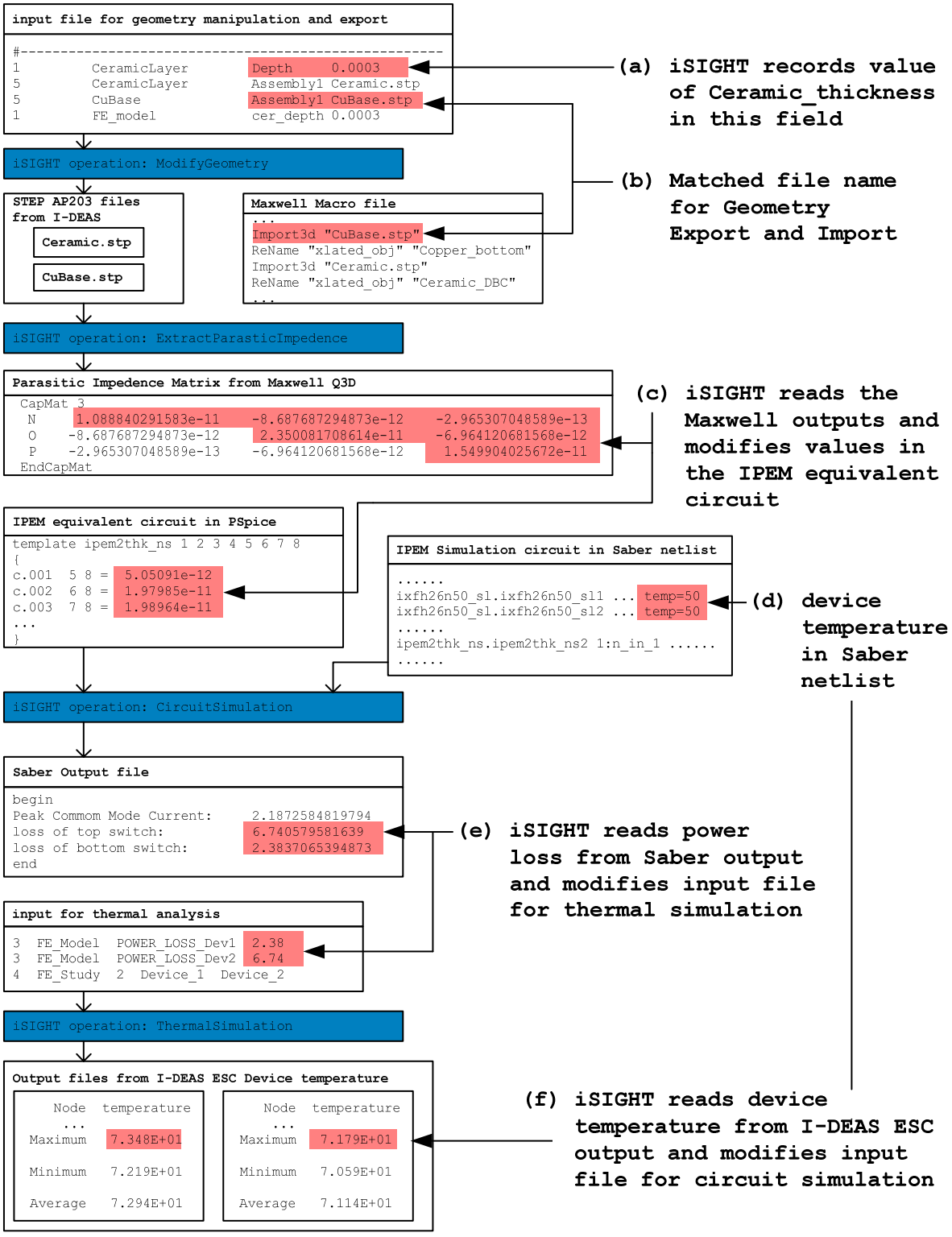


Figure 3.9. Input templates and output files for each iSIGHT operation in case study (a).

Figure 3.9, step (b) — The file names corresponding to the geometric components that are affected by this change in design variable value are identified in this input file so that they can be inserted into the Maxwell macro script. In our case, the STEP AP 203 files *Ceramic.stp* and *CuBase.stp* are generated each time the value of *Ceramic_thickness* is changed, and their names are recorded in the Maxwell macro script. Following each individual part geometry import into Maxwell, the part is renamed from the generic name *xlated_obj* to a unique name so Maxwell can precisely reference it during the subsequent parameter extraction.

Figure 3.9, step (c) — The resulting Maxwell impedance matrix is read into iSIGHT, which then adjusts the corresponding parameter values within the IPEM equivalent circuit.

Figure 3.9, step (d) — Using this equivalent circuit and the current estimate for the device temperature, Saber computes and records the peak common mode current and the power loss for each device to a text output file.

Figure 3.9, step (e) — iSIGHT reads these power loss values and records them in the input file for thermal simulation.

Figure 3.9, step (f) — I-DEAS ECS computes the revised device temperatures and records them to a text file. iSIGHT reads this file and records these revised device temperatures in the file used in step (d) above. Thus this inner loop — steps d, e, f — is iterated until the device temperatures converge. At that point, iSIGHT records the final peak common mode current, the power losses, and the device temperatures, before continuing step (a) with the next design variable value.

A typical parametric or DOE study will involve computing approximately eight to eleven data points. On a Sun Microsystems Blade100 Ultra Sparc IIe 500 with 2 Gbytes RAM, each iteration takes about 2.5 hours for an aggregate approximate 24 hours per study. Table 3.2

breaks down this computational time per iteration by operation. For the two case studies discussed here, the inner loop typically converged within two iterations. That is, each data point would typically require one *Change & Export Geometry*, one *Parameter Extraction*, two *Circuit Simulation*, and two *Thermal Simulation* operations. Other studies involving four devices have been shown to require an average of six inner-loop iterations to reach convergence [Sang03].

Table 3.2. Computational time per iteration and per operation

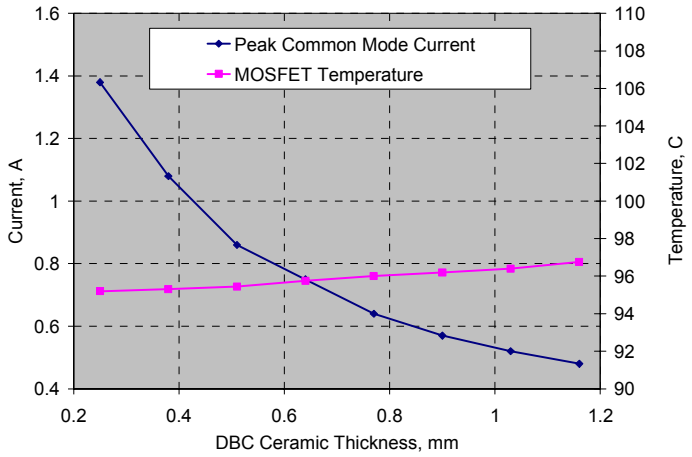
OPERATION	Change & Export Geometry	Parameter Extraction	Circuit Simulation	Thermal Simulation
COMPUTATION TIME(MINUTES)	5 – 15	30 – 40	5-15	30-50

The cumulative results for the two case studies described above are shown in Figure 3.10. Both cases show a noticeable impact on the common mode EMI current with changes in the DBC ceramic thickness or the DBC copper trace area. As expected, the common mode EMI current decreases as the DBC ceramic thickness increases (case study (a)); whereas the common mode EMI current increases as the copper trace area increases (case study (b)). Likewise, both case studies show that these two design variables have only a minimal impact on the device temperature: The MOSFET temperature increases slightly in case study (a); whereas it stays almost unaffected in case study (b). This suggests that the selection of the DBC ceramic layer thickness and copper trace area should focus on its effects on the electrical performance rather than on the thermal performance.

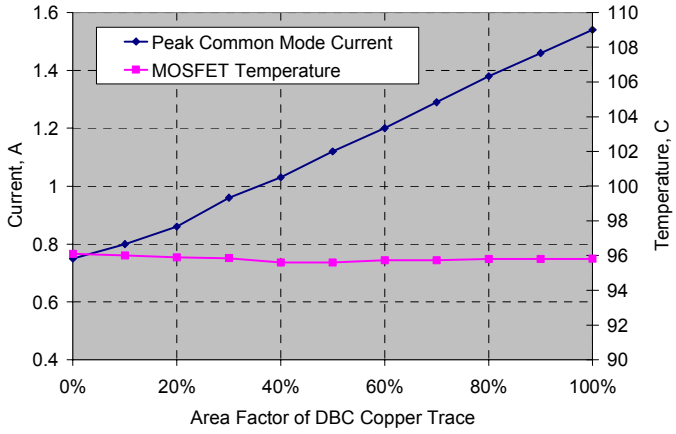
3.5 DISCUSSION

These two case studies demonstrate how parametric and DOE trade studies can be used to provide analyic insight during the design of IPEMs. Indeed, iSIGHT can easily be configured to optimize the part geometry with respect to, say, device temperatures or peak common mode

current. In either case, the automated integration process presented here represents a significant improvement over the traditional manual process of moving data betwen the software systems and manually launching each software system in turn. The manual process not only includes significant opportunities for human error, but it also typically entails significant down-time between each operation pending the attention by the human operator. Hence, it is, unfortunately, common practice today to skip the inner-loop iteration shown in Figure 3.2 due to the significant time and effort involved.



(a)



(b)

Figure 3.10 Two IPEM case studies: Trade-off between electrical and thermal performance for (a) different DBC ceramic thickness; and (b) different DBC copper trace area.

With the new integration system, this no longer is an issue. Engineers can now afford to be systematic and thorough in their analysis without taking high-risk short-cuts such as assuming that they are operating with a converged temperature and power loss. In essence, the new system provides significantly increased analytic speed, accuracy, and reduced risk of rework and design failure.

3.6 CONCLUSIONS

A flexible software integration system has been demonstrated to automate the IPEM multidisciplinary design and analysis process. It integrates four well-accepted software systems in common use within the power electronics industry: I-DEAS, Maxwell 3D, Saber, and I-DEAS ESC. The mutual dependency between power loss and device temperature has been implemented into the integration system to provide a more accurate prediction of the electrical and thermal performance for various IPEM configurations. This new system significantly reduces the design time and the risk of failure, while increasing the quality of the analysis performed.

CHAPTER IV

DESCRIBING INTEGRATED POWER ELECTRONICS MODULES USING STEP AP 210

Previous chapters described the need to create an integrated model for IPEMs where the relationships between various design views are explicitly modeled. This chapter demonstrates the procedure to describe such a model using STEP AP 210. Describing a model using AP 210 requires the study of the modeling requirements for the embedded power technologies used in the IPEM, the supporting data structure in AP 210, and the mapping from the IPEM domain to AP 210. The model description presented here contains the populated data for the functional and the physical views of the IPEM substrate, and the relationships between these two views. A method for verifying the consistency of the connectivity data based on these relationships is also presented. The remainder of this chapter details the describing of an IPEM using STEP AP 210, and it is presented in journal manuscript format.

Describing Integrated Power Electronics Modules using STEP AP 210

Yingxiang Wu[1], Jan Helge Bøhn[1], Dushan Boroyevich[1], and Thomas Thurman[2]

[1]Center for Power Electronics Systems
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

[2]Rockwell Collins
Cedar Rapids, IA 52498

Abstract

A fully integrated, multidisciplinary software environment for the design of integrated power electronics modules (IPEMs) includes a composite of multidisciplinary models of the IPEM to effectively present this product from the respective disciplines' points of view. These models are today typically stored in discrete, independent, proprietary formats. Hence, the relationships between these different views are not explicitly captured, which makes it difficult to maintain data consistency across all these models as the design evolves. The purpose of STEP AP 210 is to provide a unified, international standard format for describing such multidisciplinary models, including these different views and their relationships.

This paper presents a first, complete example of describing the multidisciplinary aspects and relationships of an IPEM within a single model using STEP AP 210. In particular, the target is the typical embedded power structure used in Generation II IPEMs, and the model description focuses on three aspects: the functional interface definition and network decomposition; the assembly decomposition and physical structure topology; and the relationships between the functional definitions and their physical implementations. A method is provided to verify that the connectivity data in the functional view is consistent with that in the physical view. STEP AP 210's ability to support IPEM modeling is thus demonstrated.

4.1 INTRODUCTION

The integrated methodology proposed by Boroyevich et al. [Boroyevich03] for the design of integrated power electronics modules (IPEMs) requires the modeling of various aspects of these modules. Each such aspect typically takes a different view to the product being designed by using a domain specific engineering application tool to create an appropriate model and store it in a separate, independent model file using a proprietary format. The relationships between these views—such as ensuring that the components that are connected in a circuit model are also connected in the corresponding physical layout model—are not explicitly captured in their respective models. It is therefore difficult to synchronize data and maintain data consistency across these views as changes are made in any one view. It is also difficult, from the file-bookkeeping point of view, to manage these scattered model files because these models do not contain common configuration management data. It is therefore desirable to develop a standard to describe a product model that includes the different views of the IPEM product and that explicitly models the relationships between these different product views, to facilitate integrated design automation and optimization of IPEM products.

This paper presents an example of describing the multiple views of the IPEM and their relationships in a single model using STEP AP 210. STEP AP 210 [ISO01] is the application protocol of ISO 10303 for representing the design of electronic assemblies, their interconnection and packaging. AP 210 focuses on the requirement definition, design, analysis, and manufacturing phases of electromechanical products, and it provides the data structures for exchanging information between application processes during each of these phases.

This paper verifies AP 210's support for IPEM modeling, and it demonstrates the use of AP 210 by implementing an AP 210 application reference model (ARM) for the embedded power substrate used in Generation II (Gen-II) IPEMs [Barbosa02]. This model consists of two different views: the functional definitions of the module interface and network decomposition, and the assembly structure decomposition and physical topology. The relationships between these two views (i.e., the mappings between functional definitions and the corresponding physical implementations) are explicitly modeled. Based on these mappings, a method has been developed to verify that the connectivity data in the functional view are consistent with that in the physical view.

The remainder of this paper is therefore structured as follows: First, the embedded power technology that is used in the Gen-II IPEMs is introduced. Next, the data structures in the AP 210 ARM that support the description of the embedded power are presented. Section 4.3 then presents how the data instances are populated for the functional and physical views. Then, finally, Section 4.4 presents a method to verify the connectivity consistency between these views.

4.2 MODELING REQUIREMENTS AND AP 210 DATA STRUCTURES

The Gen-II IPEM is a highly integrated power electronics module package that uses a planar metalization technology that is referred to as embedded power (EP) [Liang01]. Figure 4.1 shows the cross-section of the Gen-II IPEM EP substrate with the embedded power devices. At the bottom is the direct bound copper (DBC) substrate, which consists of a copper layer, a ceramic layer, and etched copper traces. The power device chips are soldered onto the copper trace and embedded into a ceramic carrier. Another layer of dielectric material covers the chips and via holes are left above device terminals to allow for connections to other circuits through

metalization layers. This new packaging method eliminates the need for wire bonds, which is beneficial from both an electrical and a thermal perspective.

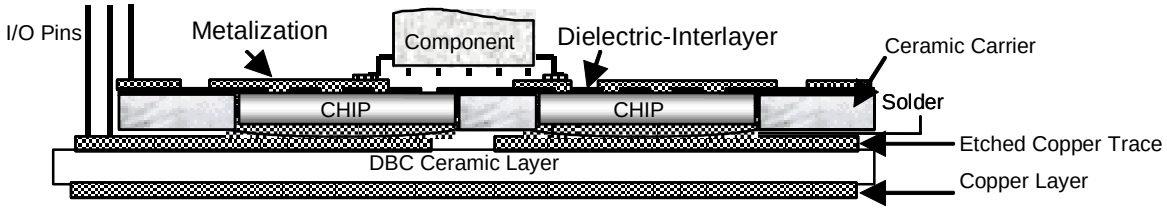


Figure 4.1. Cross-section view of Generation II IPEM.

The IPEM substrate modeled in this paper consists of the DBC substrate, two embedded power chips, the ceramic carrier, the dielectric interlayer, and the metalization layer. It does not consider the assembly of other components or circuits that are mounted on the substrate because there are other projects within the AP 210 development and validation effort that address the modeling of general print circuit assemblies [PDES03] [Smith02]. An explored view of this IPEM substrate is shown in Figure 4.2. Indeed, this particular substrate is quite well understood, as it has been subject to extensive thermal and electromagnetic analysis in the context of the Gen-II IPEM development process [Pang02].

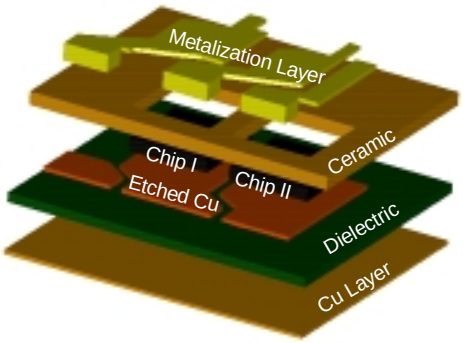


Figure 4.2. Explored view of the IPEM embedded power structure

This IPEM substrate can be described in the form of an AP 210 application reference model (ARM). The ARM is an information model, or data structure, that describes the information requirements and constraints of a specific application context, using application

objects (AO). An AO is an atomic element of an ARM that defines a unique concept of the application and contains attributes specifying the data elements of the object. For instance, using the AP 210 ARM, an implementation of an IPEM might map the physical design view of the IPEM substrate to the application object (AO) *Interconnect_module*. This AO describes, among others, the assembly structure, the physical layout, and the connections to power chip terminals through layers of conducting materials. Another example concerns the embedded power chips (Fig. 4.1). Here each power chip is described using a distinct instance of the *Bare_die_component*s AO, whose usage view definitions are defined by the common *Bare_die* AO. A usage view provides information on a part or an assembly that conveys form, fit, function, and interface specifics, but that does not include any internal design details. In this particular case, where there is only one type of power chip, the description of the power chip used is contained within single *Bare_die*, while the description of each instance of this power chip within an IPEM assembly is contained within two separate *Bare_die_component*s AOs.

To support the modeling of the Gen-II IPEM EP structure design by an *Interconnect_module*, AP 210 had to extend its ARM data structure to provide explicit description of the following two aspects: (1) the assembly location of the embedded power chips inside the EP structure; and (2) the implementation of connections to embedded devices terminals. In order to provide compatibility between two-dimensional (2D) and three-dimensional (3D) system, AP 210 uses the *Stratum* AO and the adjacent relationships between *Stratum* surfaces to model the material stack-up of an *Interconnect_module*. The interconnects within the EP structure were also described using concepts within the scope of *Stratum*. However, since the embedded devices do not belong to the *Stratum* scope, additional data

structure was needed to specify the assembly locations of the embedded devices inside the *Interconnect_module* and the connections to device terminals.

Figures 4.3 and 4.4 highlight the extended data structures for modeling the EP structure, using Express-G notation [ISO94-4]. For the assembly location of the embedded devices inside the EP structure, AP 210 now uses the boundary modeling method and relies on two AOs shown in Figure 4.3: *Component_2d_embedded_location*, and *Adjacent_stratum_surface_embedded-_component_surface_definition*. The former AO extends *Component_2d_location*, which provides the 2D placement location of the embedded component relative to the plane of the interconnect substrate, and it relies on the later AO to provide supplementary information about the vertical adjacency between the stratum surfaces and the embedded component surfaces. This allows the derivation of the 3D locations of the devices within the module from a 2D description.

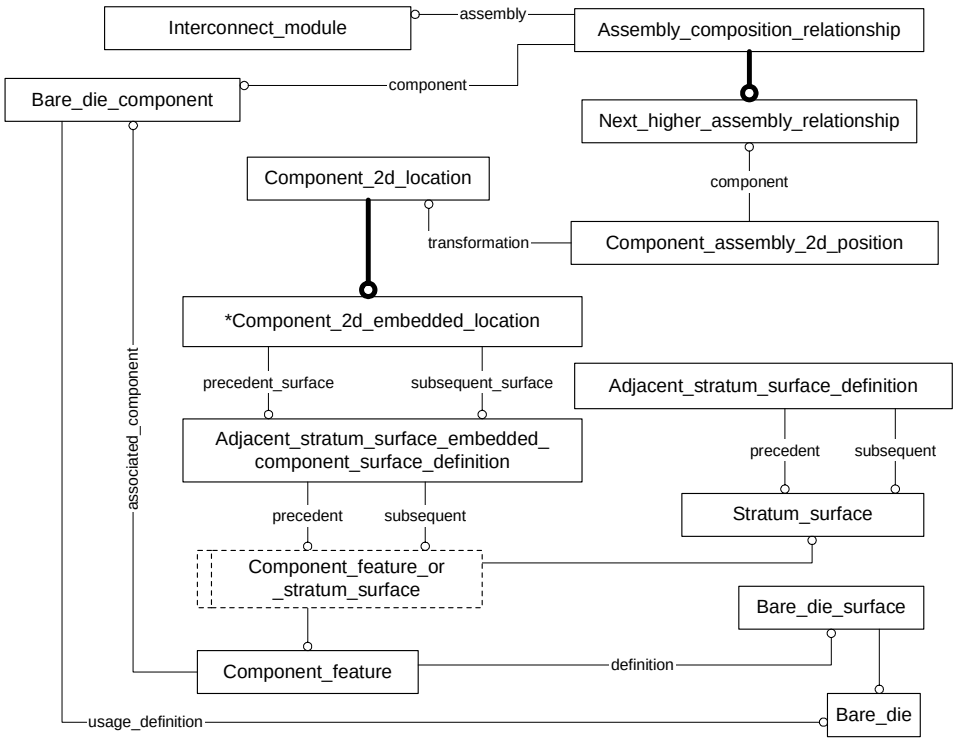


Figure 4.3. AOs for Assembly location of embedded bare die in the interconnect module.

The three major AOs that were added for the implementation of connections to the embedded terminals are shown in Figure 4.4: *Stratum_embedded_component_join_relationship*, *Direct_stratum_component_join_implementation*, and *Indirect_stratum_component_join-_implementation*. These AOs provide supports for binding an *Embedded_component_terminal* to a *Layer_connection_point* on a *Stratum*. Combined with other *Join_relationship*s defined within or between *Stratum*s, they specify a complete implementation graph for the connections between the embedded device terminals. Such a graph is referenced as a *Physical_network* in AP 210.

The details of these solutions are documented on the AP 210 STEP Enhancement and Discrepancy System as bugs #420 and #962 [NASA03]. The next section will explain in detail the resulting AP 210 ARM model for the IPEM substrate, and the population of the associated AOs.

4.3 AP 210 ARM MODEL FOR GEN-II IPEM SUBSTRATE

The physical design view of the IPEM substrate maps to the AO *Interconnect_module* in AP 210 ARM, which in this case is an assembly of two embedded power chips and several stack-up material layers. This section first provides, from the external usage point of view, the functional and physical definitions of the embedded power device and the IPEM substrate. It then describes, from the internal design view, the functional network decomposition of the IPEM substrate and its physical implementation details. Finally, mappings between the functional definitions and the corresponding physical implementations are explicitly recorded to support the verification of the connectivity consistency between views.

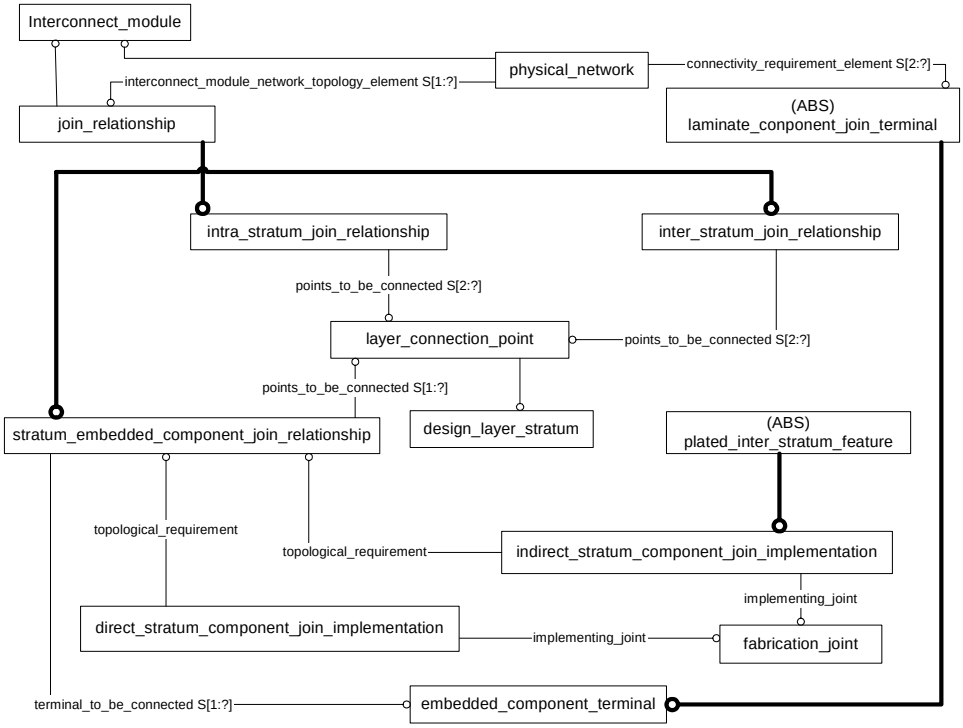


Figure 4.4. AOs for binding embedded device terminals to connection points on the design layers in the interconnect module.

4.3.1 Usage View Definitions of Embedded Power Devices

The physical usage view definition of the power chip is provided by the AO *Bare_die*. There is no design view definition of the power chip as it is not considered in our modeling. Figure 4.5 shows the AOs that are associated with the functional and physical usage view definitions of the embedded power device, and Figure 4.6 shows the populated data instances.

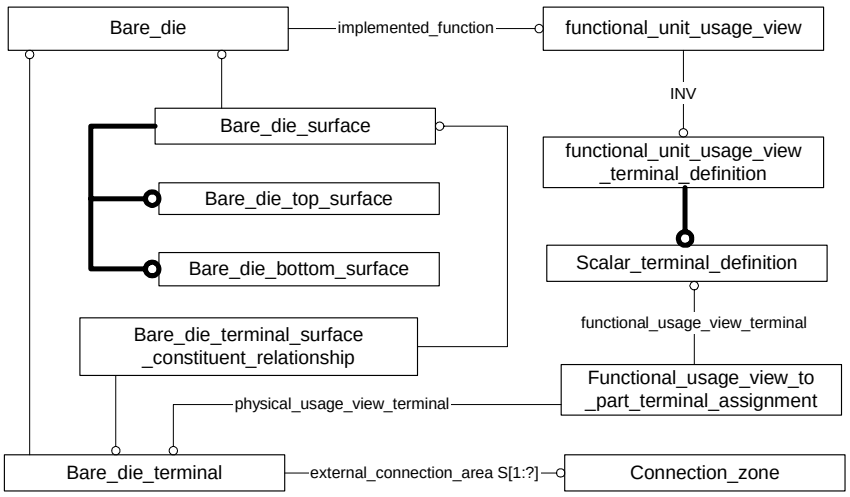


Figure 4.5. Data structure for physical and function usage view definitions of the power chip.

The power chip used in the assembly of power electronics modules has a vertical structure. The gate and source electrodes are located on the top surface while the drain electrode is on the bottom of the chip. Figure 4.6 shows the physical definition of these three terminals (*Bare_die_terminal*s #200, #220, and #290) and their functional definitions (*Scalar_terminal_definition*s #250, #260, and #320). The source electrode has multiple pads (*Connection_zone*s #190 thru #195), all of which are connected to deposited copper through *Via*s that will be described later in subsection 4.3.5. Instances #270, #280, and #300 bind the physical definition to the functional definition for the source, the drain, and the gate, respectively. Instances #140 and #150 provide definitions for the top and bottom surfaces of the chip, and will be used to specifying the embedded locations of two instances of the chip.

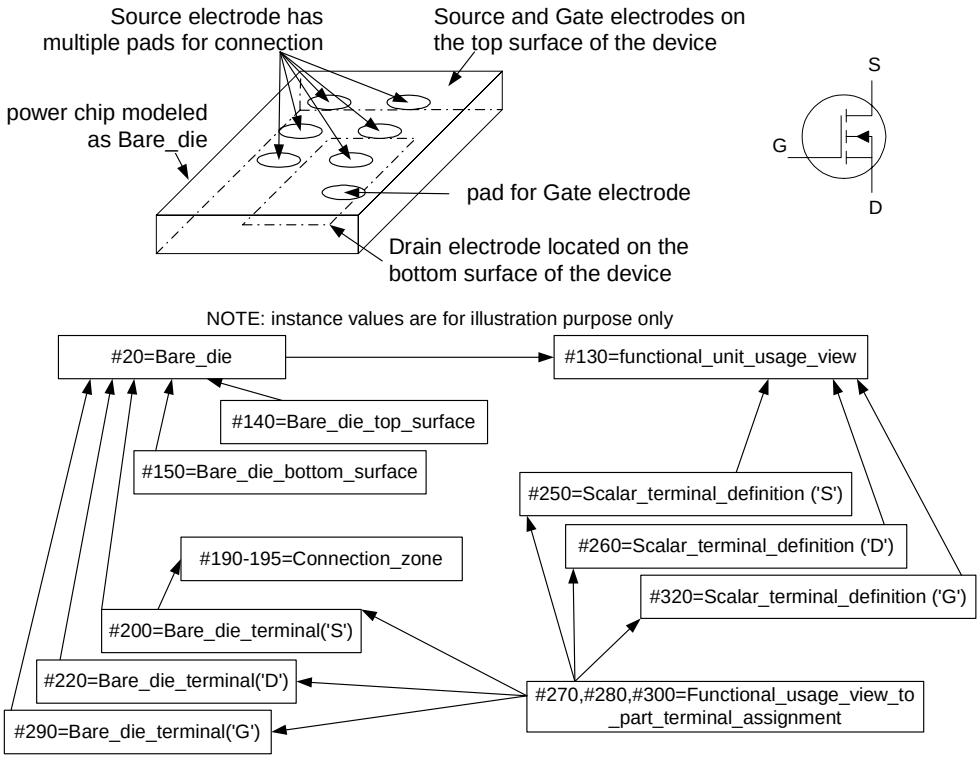


Figure 4.6. Data population for the physical and function usage view definitions of the power chip.

4.3.2 External Usage View Definitions of IPEM Substrate

The external view of the IPEM substrate has five terminals that are connected to the embedded power chips through conducting materials. The physical usage view of the IPEM substrate in AP 210 ARM is *Interconnect_module_usage_view*, and this is where the terminals to the module are defined. Figure 4.7 shows the related data structure, and Figure 4.8 shows the populated data instances. Data are populated in the same pattern as of the power chip usage view definitions.

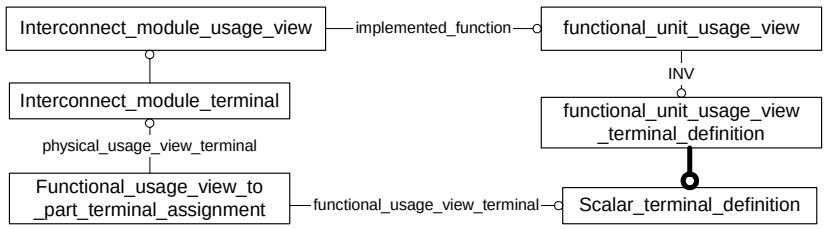


Figure 4.7 AOs for the usage view definitions of the IPEM EP substrate

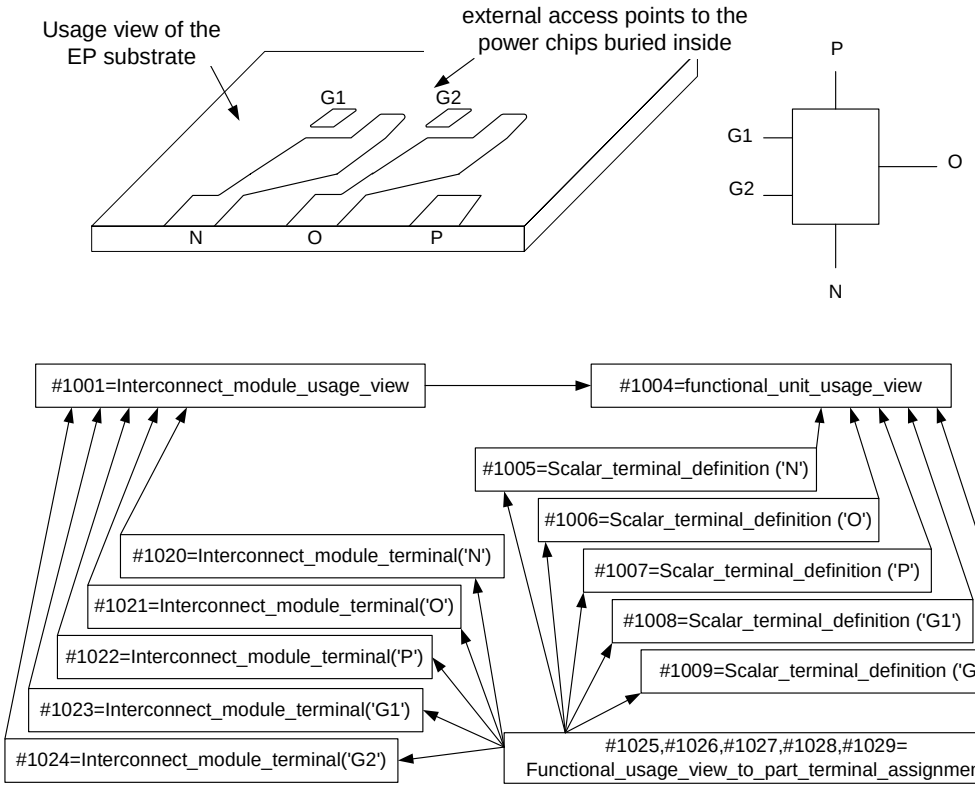


Figure 4.8. Data population for the physical and function usage view definitions of the IPEM EP substrate.

4.3.3 Internal Functional Network Decomposition of IPEM Substrate

Data in this subsection describes the structural decomposition of the IPEM substrate functional network. This is the functional design view definition of the IPEM substrate and is represented in AP 210 ARM as a *Functional_unit_network_definition.* The constituents of this design view

are *Functional_unit*s and *Functional_unit_netork_node_definitions*. A *Functional_unit* is an instance of a *Functional_unit_usage_view*, and a *Functional_unit_network_node_definition* connects the terminals of *Functional_units*. In our example, we chose MOSFET as the lowest level component in the netlist and did not decompose it further. Figure 4.9 shows the data structure for the functional design, and Figure 4.10 shows the IPEM substrate network decomposition and the populated data.

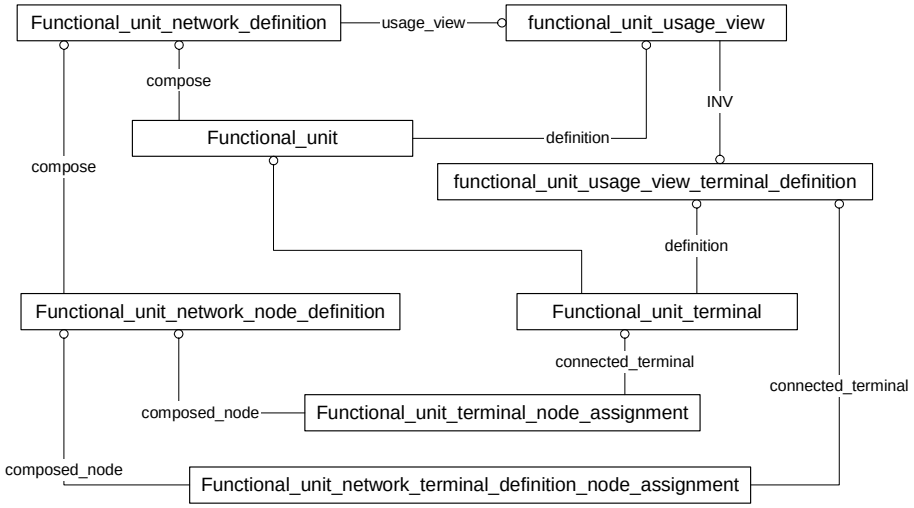


Figure 4.9. AOs for the functional design view of the IPEM EP substrate.

In this functional design view of the IPEM substrate, there are two MOSFETs (#7100 & #7200) and five internal nodes (#7300~7304) for connections. Each MOSFET is an instance of the usage view definition of the chip (#130 in Figure 4.6) with three terminal instances. The connections between terminals are specified by tying a component terminal to a node through one of the six *Functional_unit_terminal_node_assignments* (#7400~7405). The access from the external world to the inside of the substrate is provided through five Functional_unit_network_terminal_definition_node_assignment (#7500~7504) which connect the terminals of the substrate to the internal nodes and hence to the device terminals.

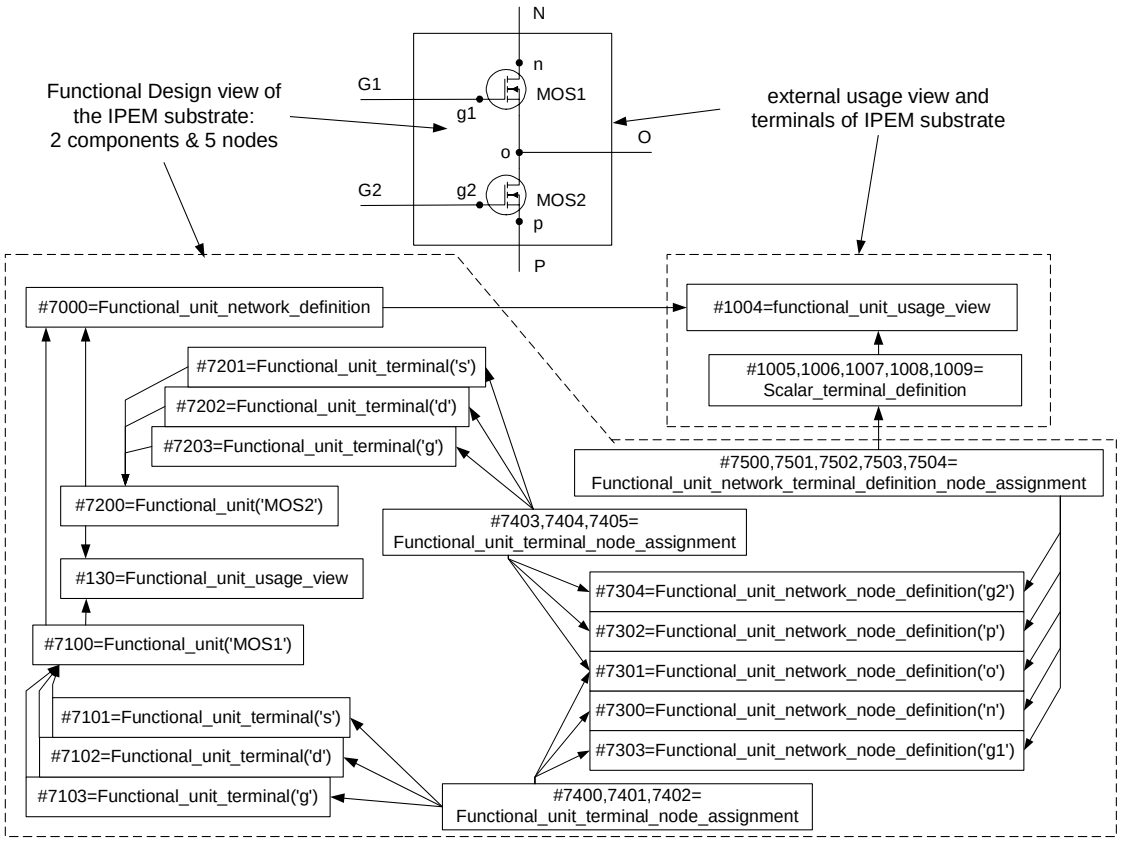


Figure 4.10. Data population for functional network decomposition of the IPEM substrate.

4.3.4 Internal Assembly View of IPEM EP Substrate

The IPEM substrate is an assembly of the DBC board, ceramic carrier, interlayer dielectric, metalization layer, and two power chips. The power chips are modeled as embedded components using *Bare_die_component*, while all layered materials are modeled as S*tratums* of the *Interconnect_module*. In our module, there are seven layers of materials (*Stratum*s): *DBC Base Cu*, *DBC Dielectric*, *DBC Etched Cu*, *Solder* layer that joins the trace and power chips, *Ceramic Frame*, *Dielectric Interlayer* and *Deposited Cu*. Each *Stratum* has two surfaces (top and bottom), and by establishing the adjacency between two *Stratum*s using *Adjacent_stratum_surface_definition* AO, the vertical location of a material layer is uniquely

identified. Figure 4.11 shows the data structure that is needed at the assembly level of the IPEM substrate in addition to those shown in Figure 4.3. Figure 4.12 shows the populated data for the material stack-up in the IPEM substrate and the placements of the embedded chips.

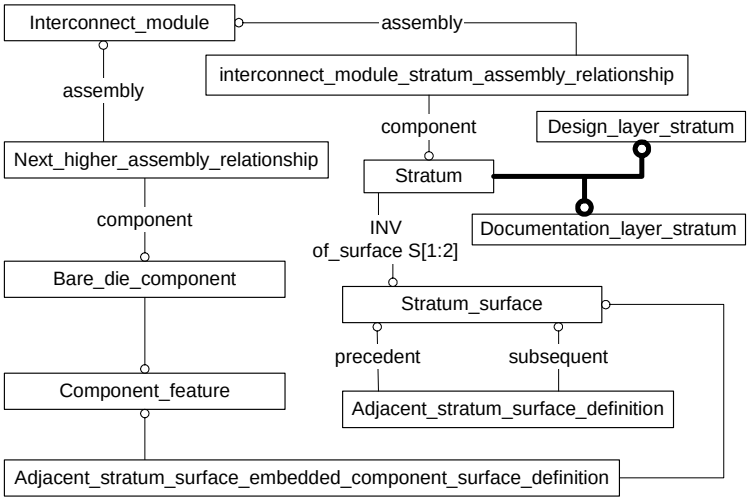


Figure 4.11. Additional AOs for the assembly of power chips and other layered materials in the design view of EP substrate.

As shown in Figure 4.12, all conducting material layers are modeled as *Design_layer_stratum* (#2050, #2090, #3100, & #4000) as required by AP 210. The dielectric layers and the ceramic layer (*Stratum* #2500, #3500, & #3700) are neither *Design_layer_stratum* nor *Documentation_layer_stratum* because their shapes are completely dependent on the shapes and features of other Stratums or embedded components. Six *Adjacent_stratum_surface_definition*s are defined between these seven *Stratums*, and two *Adjacent_stratum_surface_embedded_component_surface_definition*s are defined for each embedded power chip. In combination with the 2D locations of the chips (*Conponent_2d_embedded_location* #1042 & #1142) within the *Interconnect_module*, these data uniquely identify the 3D position of the chips.

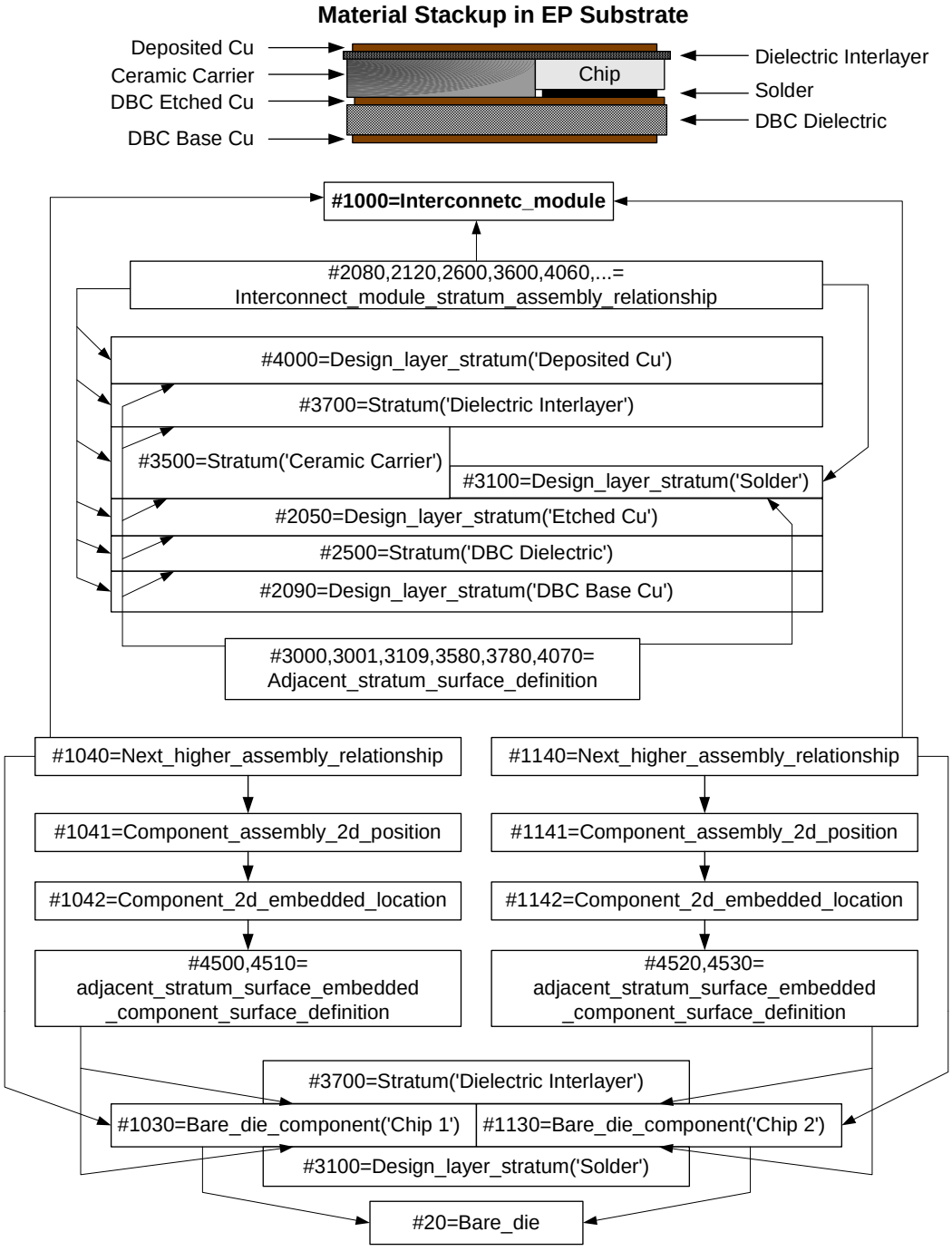


Figure 4.12. IPEM substrate assembly view structure and data population.

4.3.5 Stratum Feature and Implemented Network in the EP Substrate

Data in this subsection describes the detailed features of the *Stratums* that implement the connections to the embedded power chips. The connections from the module terminals to the power chip terminals are provided through conductors on Etched Cu and Deposited Cu layers, solder joints between copper traces and chip terminals, and copper plated vias on the Dielectric Interlayer. Figure 4.13 shows the conductive materials that compose the connections to chip terminals.

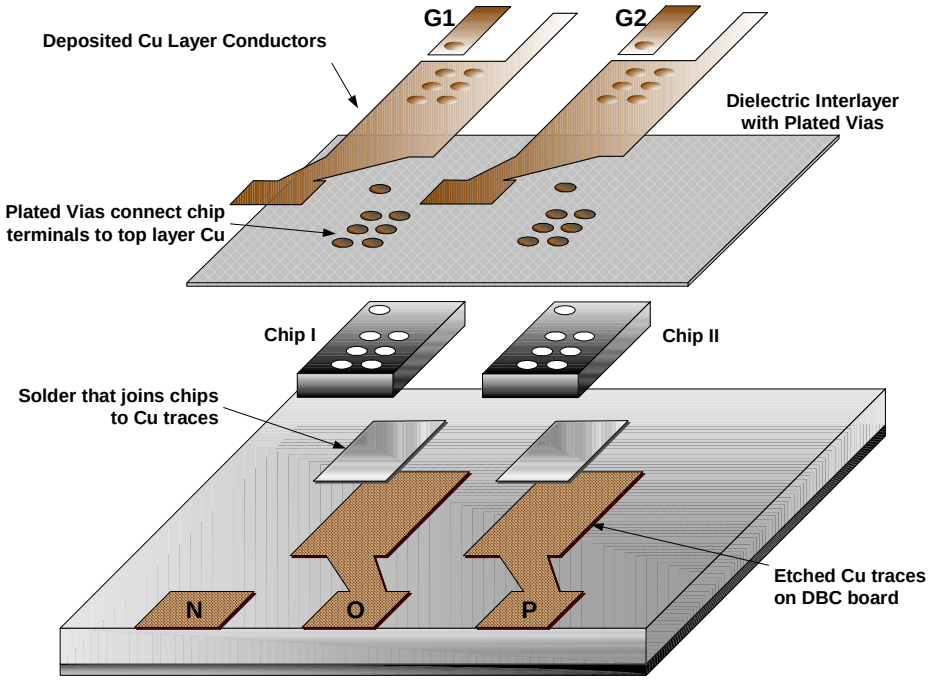


Figure 4.13. Explored view of Materials that compose interconnects between power chip terminals.

In the Etched Cu and Deposited Cu layers, each piece of copper is a *Conductor* in AP 210 ARM, which is a feature of the *Stratum* the material resides on. A *Conductor* implements the connection requirements specified by an *Intra_stratum_join_relationship* between points on that layer, and it is comprised of *Land*s and *Conductive_interconnect_element*s. The former provides

supports for connections to component terminals from the surface of the Interconnect_module, and the latter provides connections between Lands.

The Solder layer provides connections between the Etched Cu layer and the power chips' drain terminals. Each piece of solder is a *Conductive_filled_area* because there is no connection requirement (*Intra_stratum_join_relationship*) within this layer. Instead, there is an *Inter_stratum_join_relationship* that defines the connection requirement from a point on the etched Cu trace to a point on the solder, and there is a *Physical_network_supporting_stratum-_feature_conductive_join* that implements this requirement. There is also a connection requirement from the solder layer to the chip terminals defined by a *Stratum_embedded_component_join_relationship*, and there is a *Direct_stratum_component_join_implementation* that implements this requirement. These two *Join_relationship*s define the complete connection requirements from Etched Cu to the chip's drain terminal.

The conductors on the Deposited Cu layer are connected to the sources and gates of the chips through plated via holes. These vias are *Plated_conductive_blind_via*s and they implement the connection requirements defined by *Stratum_embedded_component_join_relationship*s between the chip terminals and the Deposited Cu layer.

These conductors and vias comprise five *Physical_network*s in the IPEM substrate. Each *Physical_network* is a map from a connectivity requirement between power chip terminals to the physical features that implement the requirement. A Physical_network is equivalent to a node in the functional network definition.

Figure 4.14 summarizes the data structure that are needed in addition to those shown in Figure 4.3 for the descriptions of these conductors, vias, solder joints, and physical networks they

implement. Figures 4.15 & 4.16 illustrate the populated data for the interconnections between power chip terminals and the module terminal ‘O’.

In Figure 4.15, the connection to Chip I drain terminal is provided by two *Stratum_features*: one as *Conductor* (#5100) on Etched Cu *Stratum* (#2050), the other as *Conductive_filled_area* (#5300) on Solder *Stratum* (#3100). These two features implement three join_relationships: an *Intra_stratum_join_relationship* (#5101) on Etched Cu *Stratum*, an *Inter_stratum_join_relationship* (#5301) between Etched Cu Stratum and Solder Stratum, and a *Stratum_embedded_compoment_join_relationship* (#5324) between the solder and Chip I. The connection to Chip I source terminal is provided by a conductor (#5800 in Figure 4.16) on the Deposited Cu Stratum (#4000) and six plated *Vias*. A via (#5572) connects to one of the six pads of the source terminal, and hence there needs a *Connection_zone_based_fabrication_joint* (#5876) defined between the pad and the via. All six *Vias* participate in the implementation of the *Join_relationship* (#5878) between the chip I source and the Deposited Cu. Note that Figure 4.16 shows only one *Via* populated for the connection to source terminal, other *Vias* and *Fabrication_joint*s between component terminals are omitted for clarity.

The interconnect between Chip I drain terminal, Chip II source terminal, and the 'O' terminal of the module is designated as *Physical_networks* ‘O’ (#5879) in Figures 4.15 & 4.16. This *Physical_network* references totally six *Join_relationship*s, five of which are shown in these two figures. The one that is omitted from the diagrams is the *Inter_stratum_join_relationship* between the connection point #5102 on the Etched Cu Stratum and the connection point #5802 on the Deposited Cu layer. This *Join_relationship* is implemented by a *Physicalnetwork_supporting_inter_stratum_feature* that represents the copper in that junction. Other *Physical_network*s in the module are populated in the same pattern except that each of

them involves only one chip terminal and one module terminal. These data provide complete descriptions of the physical features that implement the functional network in Section 4.3.3.

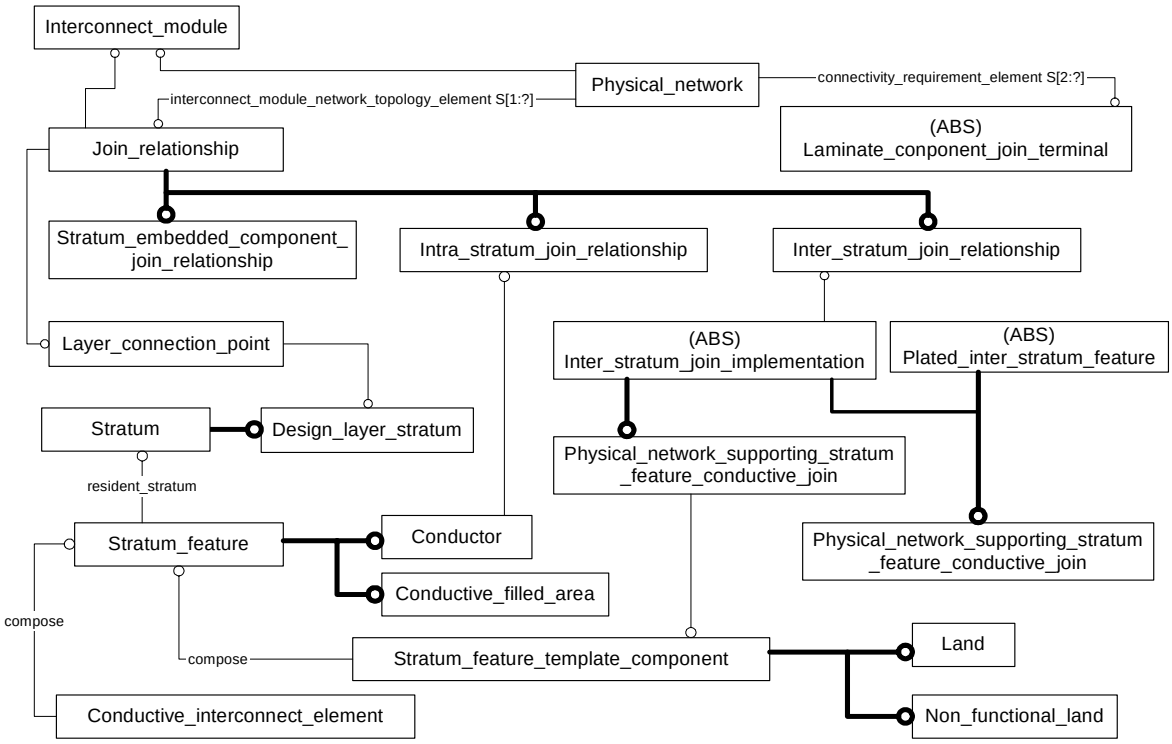


Figure 4.14. Additional AOs for the descriptions of connections to the chips' terminals.

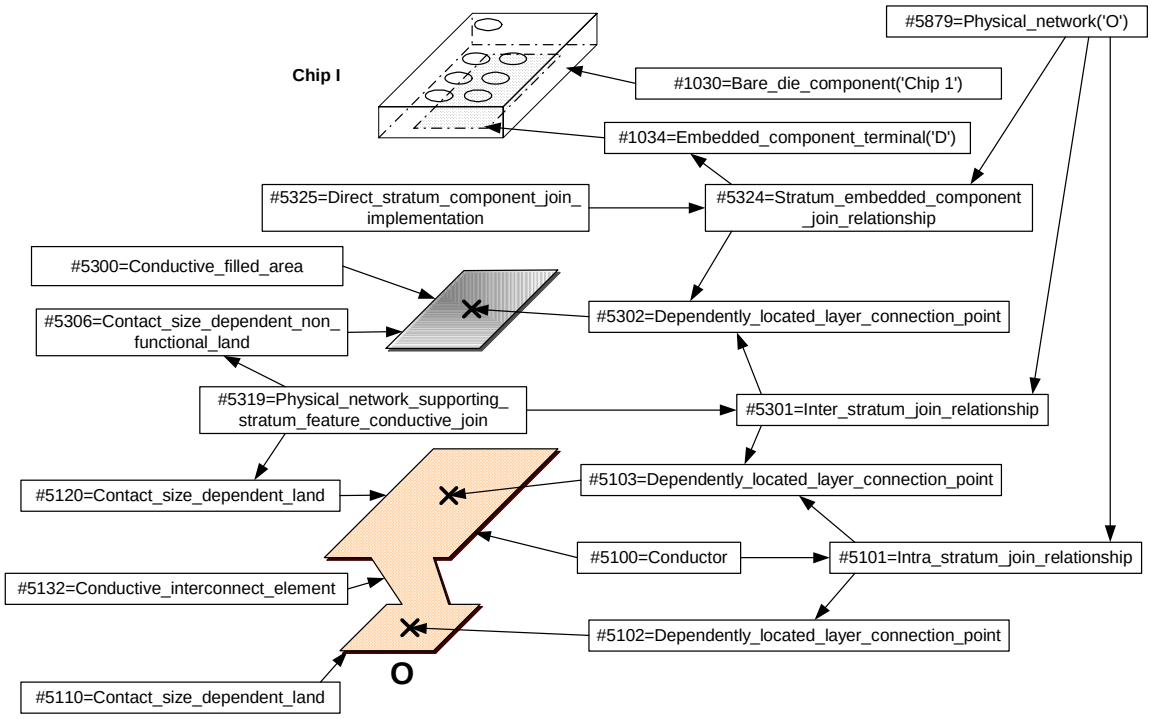


Figure 4.15. Populated data for the interconnection to Chip I drain terminal.

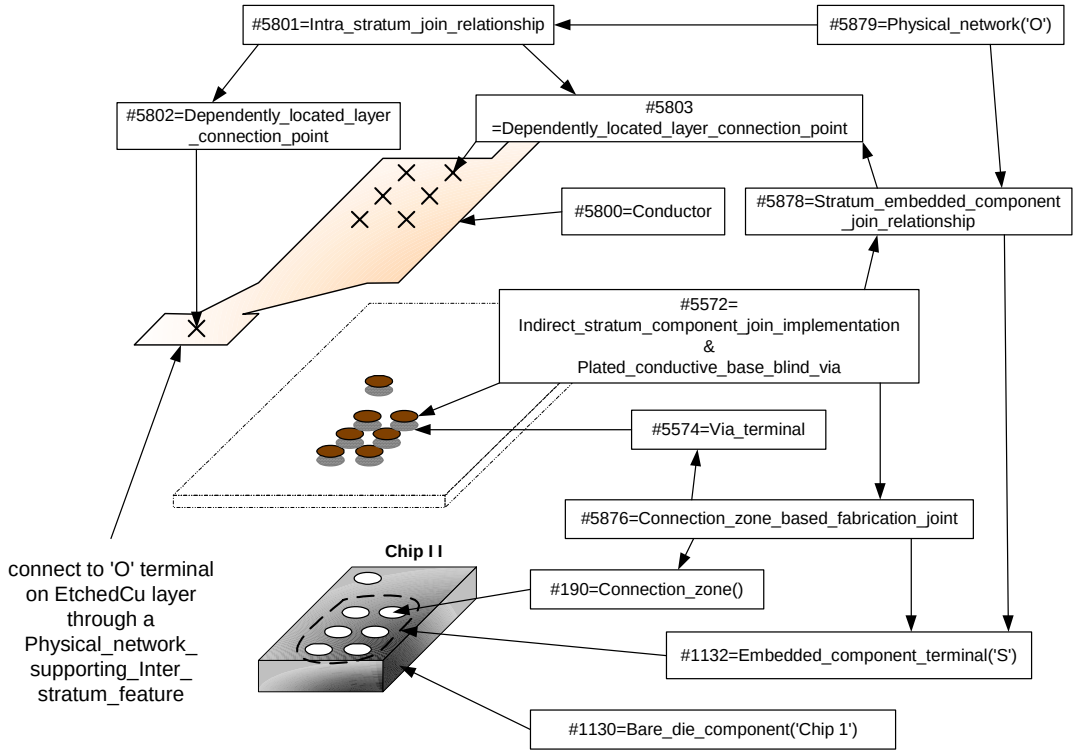


Figure 4.16. Populated data for the interconnection to Chip II Source terminal.

4.3.6 Relationships between functional network and physical implementations

Subsections 4.3.1 and 4.3.2 have shown the pin mapping relationships between the functional and physical usage view definitions of the power chip and the IPEM substrate. This subsection addresses additional relationships between the functional design view of the IPEM substrate (subsection 4.3.3) and its physical implementation (subsection 4.3.5). Specifically, these are the mappings between functional units and nodes in the design, and the physical components and networks that implement the functionalities and the nodes, respectively. Figure 4.17 shows the AOs that are needed for describing these mappings.

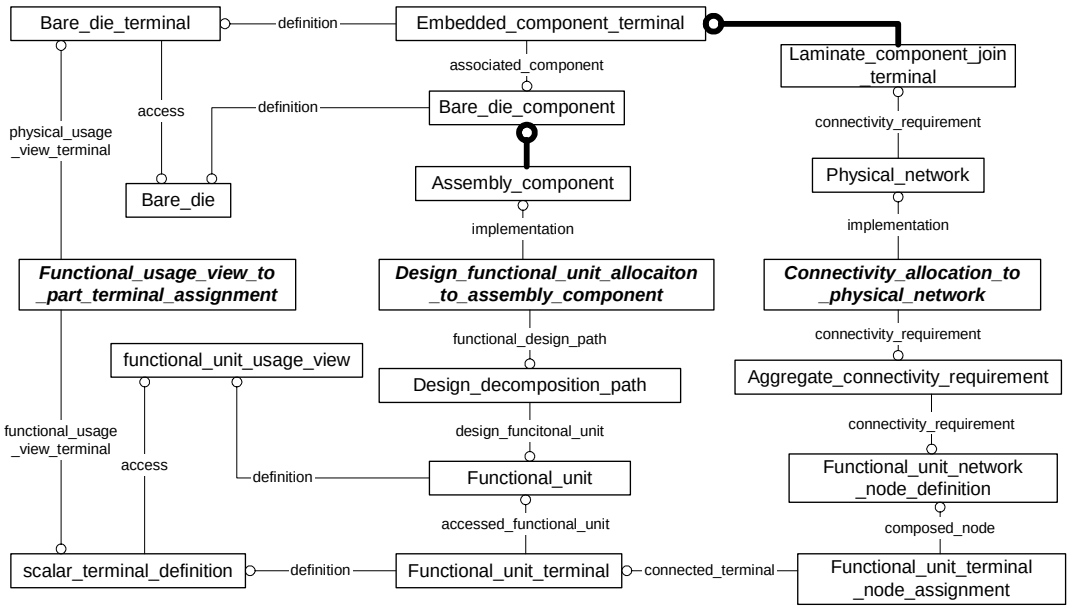


Figure 4.17. AOs for the relationships between functional design elements and physical implementations.

To map each component in the functional network to its physical implementation, a *Design_functional_unit_allocation_to_assembly_component* is populated. Figure 4.18 shows that two instances of this data type (#8100 & #8101) are populated for the mappings of two power chips. Combining with the mappings between the chip's functional terminals definition

and physical terminals definitions (instances #270, #280, & #300 in Figure 4.6), these data provide a unique map between a functional terminal instance and a physical terminal instance (e.g., the map between #1034 and #7102 in Figure 4.18 is provided through data instances #220, #260, #280, & #8100). The node to which these functional terminals are attached is mapped through AO *Connectivity_allocation_to_physical_network* to the physical network that implements the connections between the corresponding physical terminals. Figure 4.18 shows only the mapping data for node 'O' as an example. Mapping data for other nodes are populated in the same pattern but are omitted here for clarity. These relationships are used later for the verification of data consistency between those two connectivity descriptions.

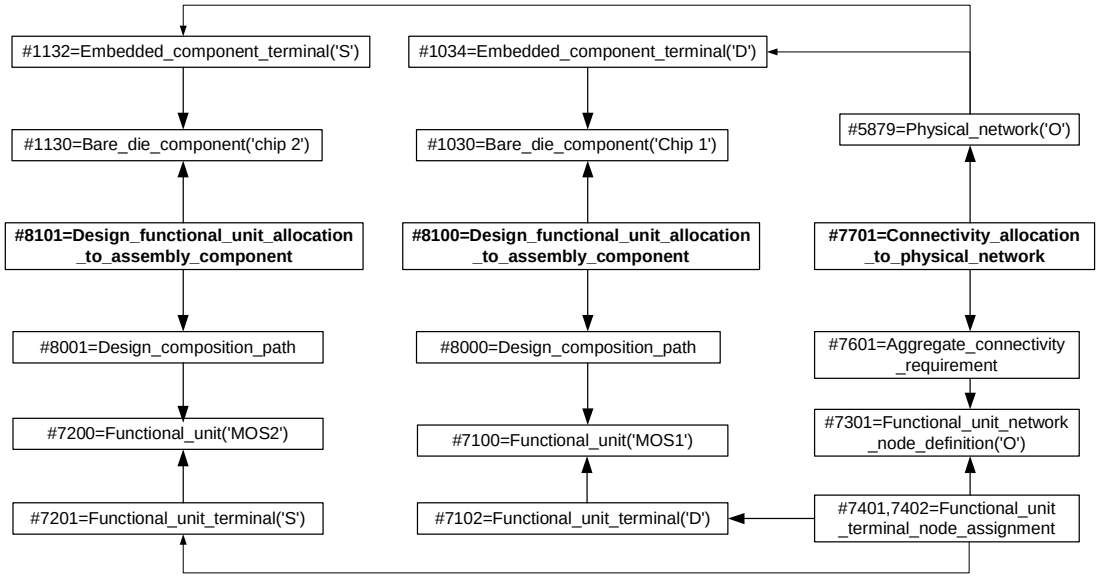


Figure 4.18. Data population for the mappings between functional units and physical components, and between functional node and physical network.

The data presented so far provides a functional and a physical usage view for the power chip and the IPEM substrate, a functional design of the IPEM substrate and a physical implementation, and the mappings in between. The next section describes a method to verify the model's consistency based on these mapping data.

4.4 IPEM ARM MODEL COHERENCE VERIFICATION

The populated IPEM ARM model provides both a functional and a physical design view of the IPEM substrate. It is necessary that the connectivity data in these two views must be consistent. Given the relationship data that was built between the functional and the physical views in Section 4.3.6, this section presents a method to verify the connectivity data consistency, which basically tracks the one-on-one mapping between functional node and physical network, builds the direct mapping between functional and physical component terminals, and confirms that the terminals which are connected functionally are also connected physically. This method consists of the following procedures:

1. Let *PN* be a list of *Physical_networks*, *FN* be a list of *Functional_unit_network_node_definitions*, and initialize both lists as empty;
2. Let *CATPN* be the list of *Connectivity_allocation_to_physical_network*s in the model and *FUTNA* be the list of *Functional_unit_terminal_node_assignment* in the model; For each allocation instance A_i in *CATPN*, do
    2.1. Assign the *Functional_unit_network_node_definition* referenced by A_i*.connectivity_requirement.connectivity_requirement* to FN_i, and assign the *Physical_network* referenced by A_i*.implementation* to PN_i. Each PN_i is the *Physical_network* that maps to the node FN_i;
    2.2. Let *FT* be a set of *Functional_unit_terminal*s, and initialize the set as empty;
    2.3. For each *Functional_unit_terminal_node_assignment* $FUTNA_i$, if $FUTNA_i$*.composed_node* = FN_i then add the *Functional_unit_terminal* referenced by $FUTNA_i$*.connected_terminal* to *FT*; *FT* is the set of terminals connected to the node FN_i;

2.4. For each functional terminal FT_j, do

2.4.1. Find a *Function_usage_view_to_part_terminal_assignment FUVTPTA* in the model such that the definition referenced by FT_j*.definition* is the same one referenced by *FUVTPTA.functional_usage_view_terminal*; and

2.4.2. Find a *Design_functional_unit_allocation_to_assembly_component DFUATAC* in the model such that the *Functional_unit* referenced by *DFUATAC.design_composition_path.design_functional_unit* is the same one referenced by FT_j*.accessed_functional_unit*; then

2.4.3. Find the *Embedded_component_terminal ET* such that the definition referenced by *ET.definition* is the same one referenced by *FUVTPTA.physical_usage_view_terminal*, and the *Assembly_component* referenced by *ET.associated_component* is the same one referenced by *DFUATAC.implementation*; this *ET* is the physical terminal that corresponds to the functional terminal FT_j;

2.4.4. If *ET* is not in the set of component_terminals referenced by PN_i*.connectivity_requirement*, then the physical connectivity implementation is not consistent with the functional connectivity requirement, and correction of this model discrepancy is then necessary;

2.4.5. Otherwise, continue step 2.4.1 with the next functional terminal in *FT* until all instances in this list are checked;

2.5. Go back to step 2.1 and proceed with the next allocation instance until all items in *CATPN* are checked. The connectivity data in both views are consistent if no discrepancy has been found.

The above procedures examine whether the physical connectivity implementation of the IPEM substrate is consistent with the functional connectivity requirement, and it can be used when changes are made to either the functional or the physical design view.

4.5 CONCLUSIONS

This paper has presented an example of describing the IPEM embedded power substrate using STEP AP 210. The AP 210 data entities that are provided for the design view description of print circuit board have been reviewed for their support of the embedded power structure. An AP 210 ARM model has been implemented for the IPEM substrate, and it consists of two views: a functional design view definition, and a physical structure that implements the functional design. By explicitly recording in the model the relationships between structural features and the functions they implement, data ambiguity has been avoided and a method was provided to verify the data consistency between the functional and physical connectivity descriptions of the interconnect module.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

Designing integrated power electronics modules (IPEM) involves interactions between many engineering domains. This thesis has presented an integrated software system that automates the iterations for the analysis and optimization of IPEM designs, and it has demonstrated the process for creating an integrated, multi-view model of the IPEM in STEP AP210. This thesis has also provided a basis for future, true implementations of STEP AP 210in the IPEM design, modeling, and analysis process.

5.1 CONCLUDING REMARKS

The integrated software system presented in this thesis consists of four commercial CAD tools (I-DEAS, Maxwell, Saber, and iSIGHT) and an in-house program that provides a generic interface to automated geometry manipulation and thermal parametric studies using I-DEAS. This system currently integrates and automates four functions for IPEM parametric design and analysis: geometry change, electromagnetic parameter extraction, circuit simulation, and thermal analysis. Each function is carried out in batch mode by customizing the CAD/CAE systems via their application programming interfaces. Data are exchanged automatically using formal and *de facto* standard file formats (STEP AP203 for geometry, and SPICE for device models) and plain text files (for design variables and analysis results).

To demonstrate the integration of IPEM models and the maintenance of data consistency between various views, a multi-view model of a Generation-II IPEM was implemented using the STEP AP 210ARM. This model consists of one functional view

that describes the connectivity requirements between devices, and one physical view that describes the physical layout and features that implement the connectivity requirements. The relationships between these two views are explicitly recorded in the model to allow for verification of model consistency based on the connectivity data. This enabled the successful validation of the AP 210ARM for describing Gen-II IPEMs, which differ from other power electronic devices with respect to packaging and connectivity due to its use of embedded power technologies.

5.2 CONTRIBUTIONS

This thesis addressed the integration of CAD tools and product models for the IPEM design. It has made the following contributions:

Automated IPEM parametric design and analysis

The integrated software system automates the design-analysis iteration process and allows for large sets of automated parametric studies to be performed in a relatively short period of time. This increased ease of investigation, including the encouragement to integrate the mutual dependency between power loss and device temperature in the system, significantly increases the probability that more accurate predictions of the IPEMs electrical and thermal performance will now be pursued.

Verification of AP210's ability to support for IPEM modeling

The implementation of a Gen-II IPEM model in AP 210required the comparison of the concepts in the IPEM applications and the concepts in the AP 210ARM. It therefore served as a test bed to verify AP210's capability to describe IPEMs. To support the embedded power structure in the Gen-II IPEM, several extensions were incorporated into AP 210by its developer. This thesis has confirmed that AP 210now supports the

functional and physical views of IPEMs and the physical implementation of the embedded power structure in particular.

Developed a basis for implementing multi-view models for IPEMs using AP210

This thesis presented the mapping from the IPEM domain to the AP 210ARM. The data structure that supports the general IPEM concepts and the embedded power structure was explained. The modeling process was demonstrated through the implementation of the functional and physical views of the IPEM and the relationships between these views. A method was developed and presented to verify the model consistency using these relationship data. This thesis thus provides the basis for future implementations of AP 210in the IPEM design environment.

5.3 FUTURE WORK

The successful use of AP 210as an exchange format in the IPEM integrated design process depends on the availability of software tools that understand AP 210data. In addition, the following issues can be addressed:

IPEM Simulation models in AP210

The IPEM model described using the AP 210ARM includes only functional and physical structural views; it does not provide any simulation models. AP 210allows for simulation models to be associated with every component in the module, including the materials that implement the interconnects. The implementation of simulation models, defined either internally using AP 210EXPRESS, or by making references to external models in other languages, can be investigated to provide guidance for the integration of IPEM simulation data.

Validation of Geometry according to connectivity data

The model verification method presented in Chapter 4 enforces that the topological definition in the physical view is consistent with the functional connectivity requirement. This does not imply that the actual geometry of the model satisfies the connectivity requirement. A method must be developed to verify the geometric data, so that for the physical features that are topologically connected, their geometric boundaries are contiguous.

CAD Tools that can integrate multiple views of a product

In Chapter 4, the two views of the IPEM are integrated via relationship data defined between them. In reality, these views can be implemented by different people using different tools. CAD tools that can cross examine multiple views of a product and provide interfaces to specify the relationships between entities from different views are therefore needed.

REFERENCES

[Ansoft] Maxwell SI, version 4.0, Ansoft Corporation, Pittsburgh, PA.

[Avanti] Saber, version 5.1, Avanti Corporation, Fremont, CA.

[Barbosa02] Barbosa, P., Lee, F.C., van Wyk, J.D., Boroyevich, D., Scott., E., Thole, K., Odendaal, H., Liang, Z., Pang, Y., Sewall, E., Chen, J., and Yang, B., "An overview of the IPEM-based modular implementation for distributed power systems," *Proc., 2002 CPES Power Electronics Seminar*, pp. 70-76, 2002.

[Boattini98] Boattini, F., Galli, R., Monti, A., and Secondo, G., "Integrated design methodology for electrical drives: a marine application experience," *Proc., IEEE IECON*, 1998, pp. 591-595.

[Borojevich03] Borojevich, D., and Chen, J.Z., "Integrated Multidisciplinary Modeling, Analysis and Design in Power Electronics," *Proc., 2003 CPES Power Electronics Seminar*, Blacksburg, VA, April 27-29, 2003, pp. 25-31.

[Chen01] Chen, J.Z., Wu, Y., Gence, C., Borojevich, D., and Bøhn, J.H., "Integrated Electrical and Thermal Analysis of Integrated Power Electronics Modules Using iSIGHT," *Proc., 2001 CPES Power Electronics Seminar*, Blacksburg, VA, April 23-25, 2001, pp. 141-145.

[El-Ghany00] El-Ghany, K.M.A., and Farag, M.M., "Expert system to automate the finite element analysis for non-destructive testing," *NDT & E International*, Elsevier, UK, September 2000, vol.33, no.6, pp.409-415.

[Engineous] iSIGHT, version 6.0, Engineous Software, Cary, NC.

[IGES96] IGES/PDES Organization, *ANS US PRO/IPO-100-1996, Initial Graphics Exchange Specification 5.3,* 1996, US PRO, North Charleston, SC.

[ISO94-1] International Standards Organization, *ISO 10303-1, STEP, Part 1: Overview and fundamental principles*, 1994, US PRO, North Charleston, SC.

[ISO94-2] International Standards Organization, *ISO 10303-203, STEP, Part 203: Configuration controlled 3D designs of mechanical parts and assemblies*, 1994, US PRO, North Charleston, SC.

[ISO94-3] International Standards Organization, *ISO 10303-203, STEP, Part 21: Implementation Methods: Clear text encoding of the physical file Exchange Structure*, 1994, US PRO, North Charleston, SC.

[ISO94-4] International Standards Organization, *ISO 10303-11, STEP, Part 11: Description methods: The EXPRESS language reference manual*, 1994, US PRO, North Charleston, SC.

[ISO94-5] International Standards Organization, *ISO 10303-203, STEP, Part 22: Implementation Methods: Standard data access interface specification*, 1994, US PRO, North Charleston, SC.

[ISO01] International Standards Organization, *ISO 10303-210, STEP, Part 210: Electronic assembly, interconnection, and packaging design*, 2000, US PRO, North Charleston, SC.

[Lanning03] Lanning, Craig, Express Engine Project, release 3.1, May 23, 2003, URL: http://exp-engine.sourceforge.net

[Liang01] Liang, Z., and Lee, F.C., “Embedded Power Technology for IPEMs packaging applications,” *Proc., APEC 2001*, pp. 1057-1061.

[NASA03] AP 210STEP Enhancement and Discrepancy System, http:\\step.nasa.gov\sparkynet, November, 2003.

[Osterberg95] Osterberg, P.M., and Senturia, S.D. “Membuilder: An Automated 3D Solid Model Construction Program For Microelectromechanical Structures,” *Proc., the 8th International Conference on Solid-State Sensors and Actuators, and Eurosensors IX*, Stockholm, Sweden, June 25-29,1995, vol. 2, pp. 21-24.

[PDES01-1] PDES, Inc., *The International Standard for Electronics and Electromechanical Design*, URL: http://ap210.aticorp.org/

[PDES01-2] PDES, Inc., “Concept of Operations for Application Protocol 210,” ME008.01.01, February 7, 2001, URL: http://pdesinc.aticorp.org/whatsnew/rec_prac/ap210_2701.pdf

[PDES01-3] PDES, Inc., Electro-Mechanical (EM) Pilot, URL: http://empilot.aticorp.org/

[PDES03] PDES, Inc., STEP for Electronics, URL: http://www.ap210.org/

[Pang02] Pang, Y.F., Chen, J.Z., Scott, E.P., and Thole, K.A., “Electrical and Thermal Layout Design and Optimization Considerations for DPS Active IPEM,” *Proc., 2002 ASME International Mechanical Engineering Congress & Exposition*, IMECE 2002, New Orleans, LA, November 17-22, 2002, paper 33778.

[Sang03] Sang, T., Wu, Y., Lee, F.C., Odendaal, W.G., and Bøhn, J.H., “An Iterative Approach for Integrated Electro-thermal Design of a Distributed Power Supply Systems Module,” *Proc., 2003 CPES Power Electronics Seminar*, Blacksburg, VA, April, 2003, pp. 304-307.

[SDRC1] I-DEAS Master Series, version 8, SDRC, Milford, OH.

[SDRC2] I-DEAS Master Series Open Architecture User's Guide, SDRC, Milford, OH.

[SDRC3] Open I-DEAS Programming Guide, SDRC, Milford, OH.

[Smith02] Gregory L. Smith, "Utilization of STEP AP 210at the Boeing Company," *Computer-Aided Design*, vol. 34, issue 14, pp. 1055-1062, 2002.

[Yoon00] Yoon, D.H., and Shaikh, F.Z., “Integrating CAD and CAM via CORBA,” *Proc., the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2000),* Edinburgh, UK, 3-7 April 2000, pp. 3-8.

APPENDIX

IPEM MODEL IN STEP PART 21 FILE

This appendix provides an example of the IPEM data is mapped into the AP 210 ARM. The data is stored in STEP physical file using Part 21 format.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((' '),'2;1');
FILE_NAME('IPEM_test_case.p21',
      '2003-4-1 T11:50:28',
      ('Y Wu'), ('CPES'),
      '2.4.5 Gustas (compiled 15:04 2002/10/22)',
      'Express Engine', '',' ');
FILE_SCHEMA(('AP210_ARM_WD1_34'));

ENDSEC;
DATA;
#1=EE_DOCUMENT(#5,$,$,#2,$,#3,$,$,(#4),'1',
      'Electronic interconnect, assembly and packaging design');
#2=EE_DOCUMENT_IDENTIFIER('document name',
      'Industrial automation systems and integration Product data
      representation and exchange Part 210 : Application Protocol:
      Electronic assembly, interconnect, and packaging design');
#3=DATE_AND_TIME(1,2001,9,12,0,0,0,.BEHIND.);
#4=PERSON('Tom',$);
#5=PERSON_AND_ORGANIZATION(#4,#6,$);
#6=ORGANIZATION('RCI',$);
#10=DATE_AND_TIME(1,2002,1,1,1,1,1,.AHEAD.);
#20=BARE_DIE((#80),'MOSFET',#30,#41,#110,#100,#10,$,$,#40,
      'conceptual design',$,$,$,$,$,$,$,#120,#130);
#30=SECURITY_CLASSIFICATION(#70,#100,#10,'',#10);
#40=EE_TEXT('Discipline id : Power Electronics');
#41=EE_TEXT('default physical unit usage view context description');
#42=EE_TEXT('default functional usage view context description');
#50=PERSON('Yingxiang Wu','');
```

```
#60=ORGANIZATION('Center for Power Electronics Systems,
      Virginia Tech','');
#70=PERSON_AND_ORGANIZATION(#50,#60,'');
#80=ORGANIZATION('UNKNOWN ORG','UNKNOWN ADDRESS');
#90=EE_PRODUCT('','Mosfet','part number 1',#80,.T.);
#91=EE_PRODUCT_RELATED_PRODUCT_CATEGORY($,$,'bare_die',#1,(#90));
#100=EE_APPROVAL('',(#80),.APPROVED.,#10);
#110=EE_PRODUCT_VERSION(#80,$,'',#90,#30,#100);
#120=SEATING_PLANE('Mosfet Seating plane',$,#20);
#130=FUNCTIONAL_UNIT_USAGE_VIEW((#80),'MOSFET',#30,#42,#110,#100,#10,$,
      $,#40,'',$);
#140=BARE_DIE_TOP_SURFACE('Bare Die top',$,#20,$,$,$);
#150=BARE_DIE_BOTTOM_SURFACE('Bare Die bottom',$,#20,$,$,$);
#160=EE_DOCUMENT_IDENTIFIER('identifier name','assigned value');
#170=EE_DOCUMENT(#80,$,$,#160,$,#10,$,$,$,'rev #',
      'ee_material document name');
#180=EE_MATERIAL('Copper',.CONDUCTIVE.,.NON_CONDUCTIVE.,
      #170,.CONDUCTIVE.);
#190=CONNECTION_ZONE($);
#191=CONNECTION_ZONE($);
#192=CONNECTION_ZONE($);
#193=CONNECTION_ZONE($);
#194=CONNECTION_ZONE($);
#195=CONNECTION_ZONE($);

#200=(BARE_DIE_TERMINAL(#180,(#190,#191,#192,#193,#194,#195),$)
      MANAGED_DESIGN_OBJECT()MINIMALLY_DEFINED_BARE_DIE_TERMINAL()
      PACKAGE_TERMINAL(#20,$,$,.DOES_NOT_INTERSECT.,#201,$,$,$,$,$,$)
      PART_FEATURE(#20,$,$)PART_TERMINAL()
      PHYSICAL_FEATURE_OR_PART_TEMPLATE()
      PRIMARY_REFERENCE_TERMINAL()SHAPE_ELEMENT('Source',$));
#201=MATERIAL_DEFINITION((#80),'COPPER',#30,#202,#203,#204,#10,$,$,#40,
      'conceptual design',#180);
#202=EE_TEXT('default material definition context description');
#203=EE_PRODUCT_VERSION(#80,$,'',#206,#30,#204);
#204=EE_APPROVAL('',(#80),.APPROVED.,#10);
#205=EE_PRODUCT_RELATED_PRODUCT_CATEGORY($,$,'material',#1,(#206));
#206=EE_PRODUCT('','Material','part number 1',#80,.T.);
```

```
#210=BARE_DIE_TERMINAL_SURFACE_CONSTITUENT_RELATIONSHIP(#140,$,#200,
      'Source on top');
#220=BARE_DIE_TERMINAL('Drain',$,#20,$,$,#180,(#240),$);
#230=BARE_DIE_TERMINAL_SURFACE_CONSTITUENT_RELATIONSHIP(#150,$,#220,
      'Drain on bottom');
#240=CONNECTION_ZONE($);
#250=SCALAR_TERMINAL_DEFINITION(#130,'S');
#260=SCALAR_TERMINAL_DEFINITION(#130,'D');
#270=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#200,#250);
#280=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#220,#260);
#290=BARE_DIE_TERMINAL('Gate',$,#20,$,$,#180,(#310),$);
#300=BARE_DIE_TERMINAL_SURFACE_CONSTITUENT_RELATIONSHIP(#140,$,#290,
      'Gate on top');
#310=CONNECTION_ZONE($);
#320=SCALAR_TERMINAL_DEFINITION(#130,'G');
#330=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#290,#320);
#500=PERSON('UNKNOWN','UNKNOWN');
#510=ORGANIZATION('UNKNOWN','UNKNOWN');
#520=PERSON_AND_ORGANIZATION(#500,#510,'');
#530=EE_APPROVAL('',(#520),.APPROVED.,#10);
#540=SECURITY_CLASSIFICATION(#520,#530,#10,'',#10);

#1000=INTERCONNECT_MODULE((#60),'IPEM substrate design view',
      #540,#1002,#1012,#530,#10,$,$,#40,'conceptual design',
      $,$,$,$,$,#1001,$);
#1001=INTERCONNECT_MODULE_USAGE_VIEW((#60),'IPEM substrate usage view',
      #540,#1003,#1012,#530,#10,$,$,#40,'conceptual_design',
      $,$,$,$,$,$,$,$,$,$,$,#1004);
#1002=EE_TEXT('default interconnect module design view
      context description');
#1003=EE_TEXT('This is default interconnect module usage
      view context description');
#1004=FUNCTIONAL_UNIT_USAGE_VIEW((#60),'IPEM substrate',
      #540,#42,#1012,#530,#10,$,$,#40,'conceptual design',$);
#1005=SCALAR_TERMINAL_DEFINITION(#1004,'N');
#1006=SCALAR_TERMINAL_DEFINITION(#1004,'O');
#1007=SCALAR_TERMINAL_DEFINITION(#1004,'P');
```

```
#1008=SCALAR_TERMINAL_DEFINITION(#1004,'G1);
#1009=SCALAR_TERMINAL_DEFINITION(#1004,'G2');
#1010=EE_PRODUCT('','IPEM substrate product','ipem gen II',#520,.F.);
#1011=EE_PRODUCT_RELATED_PRODUCT_CATEGORY($,$,'interconnect module',
      #1,(#1010));
#1012=EE_PRODUCT_VERSION(#520,$,'rev code',#1010,#30,#530);

#1020=INTERCONNECT_MODULE_INTERFACE_TERMINAL('N','DEFAULT DESCRIPTION',
      #1001,$,$,$);
#1021=INTERCONNECT_MODULE_INTERFACE_TERMINAL('O','DEFAULT DESCRIPTION',
      #1001,$,$,$);
#1022=INTERCONNECT_MODULE_INTERFACE_TERMINAL('P','DEFAULT DESCRIPTION',
      #1001,$,$,$);
#1023=INTERCONNECT_MODULE_INTERFACE_TERMINAL('G1',
      'DEFAULT DESCRIPTION',#1001,$,$,$);
#1024=INTERCONNECT_MODULE_INTERFACE_TERMINAL('G2',
      'DEFAULT DESCRIPTION',#1001,$,$,$);
#1025=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#1020,#1005);
#1026=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#1021,#1006);
#1027=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#1022,#1007);
#1028=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#1023,#1008);
#1029=FUNCTIONAL_USAGE_VIEW_TO_PART_TERMINAL_ASSIGNMENT(#1024,#1009);
#1030=BARE_DIE_COMPONENT(#20);
#1031=COMPONENT_FEATURE(#140,#1030);
#1032=EMBEDDED_COMPONENT_TERMINAL(#200,#1030,$,$,$);
#1033=COMPONENT_FEATURE(#150,#1030);
#1034=EMBEDDED_COMPONENT_TERMINAL(#220,#1030,$,$,$);
#1035=EMBEDDED_COMPONENT_TERMINAL(#290,#1030,$,$,$);
#1040=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#1030,#30,$,
      'IPEM layer 4 device 1',$);
#1041=COMPONENT_ASSEMBLY_2D_POSITION(#1070,#1042,#1080,#1040);

/* Fab_joint between die primary terminal and vias */
/* Note the copper is not in the same stratum as given by #4500 */
/* which is actually refering to the covering ceramic layer */
#1042=COMPONENT_2D_EMBEDDED_LOCATION(.T.,#1110,#4500,#4510,#5676);
#1050=LENGTH_DATA_ELEMENT(1.0E-5,'.METRE.',$);
#1060=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
```

```
      .RADIAN.,#1050,$);
#1070=ASSEMBLY_COMPONENT_2D_SHAPE(#1060,$,(#1030, #1130));
#1080=PHYSICAL_UNIT_PLANAR_SHAPE(#1060,(#1100,#1200),$,#1000,$,.ABOVE.,
      .NOMINAL_MATERIAL_CONDITION.,$,.MANUFACTURING.,
      PREDEFINED_PLANAR_PURPOSE(.DESIGN.));
#1090=DIRECTION();
#1091=DIRECTION();
#1100=CARTESIAN_POINT();
#1110=CARTESIAN_TRANSFORMATION_OPERATOR_2D(#1090,#1091,#1100,1.0);
#1130=BARE_DIE_COMPONENT(#20);
#1131=COMPONENT_FEATURE(#140,#1130);
#1132=EMBEDDED_COMPONENT_TERMINAL(#200,#1130,$,$,$);
#1133=COMPONENT_FEATURE(#150,#1130);
#1134=EMBEDDED_COMPONENT_TERMINAL(#220,#1130,$,$,$);
#1135=EMBEDDED_COMPONENT_TERMINAL(#290,#1130,$,$,$);
#1140=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#1130,#30,$,
      'IPEM layer 4 device 2',$);
#1141=COMPONENT_ASSEMBLY_2D_POSITION(#1070,#1142,#1080,#1140);

/* Fab_joint between die primary terminal and vias */
/* Note the copper is not in the same stratum as given by #4500 */
/* which is actually refering to the covering ceramic layer */
#1142=COMPONENT_2D_EMBEDDED_LOCATION(.T.,#1210,#4520,#4530,#5876);
#1190=DIRECTION();
#1191=DIRECTION();
#1200=CARTESIAN_POINT();
#1210=CARTESIAN_TRANSFORMATION_OPERATOR_2D(#1190,#1191,#1200,1.0);

/* definition of stratum start */
#2000=EE_PRODUCT('default description','DBC trace layer',
      'IPEM layer 3 trace',#520,.F.);
#2010=EE_PRODUCT_VERSION(#520,$,'rev 1',#2000,#30,#100);
#2020=LENGTH_DATA_ELEMENT(0.3,'METRE',.MILLI.);
#2030=LENGTH_DATA_ELEMENT(0.1,'METRE',.MILLI.);
#2040=DESIGN_LAYER_TECHNOLOGY(.ALL.,#180,#2020,#2030,
      'copper stratum technology',#2030,$,#2030,$,$,.POWER_OR_GROUND.);
#2050=DESIGN_LAYER_STRATUM((#60),'product def id',
      #540,#40,#2010,#530,#10,$,$,#40,'DBC copper trace',$,#2040,.T.);
```

```
#2060=STRATUM_SURFACE(#2050,.PRIMARY_SURFACE.);
#2070=STRATUM_SURFACE(#2050,.SECONDARY_SURFACE.);
#2080=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#2050,
      'IPEM layer 3 DBC trace');
#2090=DESIGN_LAYER_STRATUM((#60),'product def id',
      #540,#40,#2092,#530,#10,$,$,#40,'DBC copper ground',$,#2040,.F.);
#2091=EE_PRODUCT('default description','DBC GND layer',
      'IPEM layer 1 copper',#520,.F.);
#2092=EE_PRODUCT_VERSION(#520,$,'rev 1',#2091,#30,#100);
#2100=STRATUM_SURFACE(#2090,.PRIMARY_SURFACE.);
#2110=STRATUM_SURFACE(#2090,.SECONDARY_SURFACE.);
#2120=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#2090,
      'IPEM layer 1 DBC bottom');
#2500=STRATUM((#60),'product definition
id',#540,#40,#2502,#530,#10,$,$,#40,'DBC dielectric',$,#2530);
#2501=EE_PRODUCT('default description','DBC Dielectric',
      'IPEM layer 2 dielectric',#520,.F.);
#2502=EE_PRODUCT_VERSION(#520,$,'rev code',#2501,#30,#100);
#2510=STRATUM_SURFACE(#2500,.PRIMARY_SURFACE.);
#2520=STRATUM_SURFACE(#2500,.SECONDARY_SURFACE.);
#2530=STRATUM_TECHNOLOGY(.INTERNAL.,#2540,#2550,#2560,
      'dielectric stratum technology',$,$,$,$,$);
#2540=EE_MATERIAL('Aluminum Oxide',
      .NON_CONDUCTIVE.,.NON_CONDUCTIVE.,#170,.RESISTIVE.);
#2550=LENGTH_DATA_ELEMENT(3,'METRE',.MILLI.);
#2560=LENGTH_DATA_ELEMENT(0.3,'METRE',.MILLI.);
#2600=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#2500,
      'IPEM layer 2 DBC dielectric');
#3000=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 3-2
      DBC_Cu_Ceramic',#2070,#2510);
#3001=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 2-1
      DBC_Ceramic_Cu',#2520,#2100);

/* need a DESIGN layer for solder */
#3100=DESIGN_LAYER_STRATUM((#60),'product def
id',#540,#40,#3102,#530,#10,$,$,#40,'IPEM solder layer',$,#3103,.F.);
#3101=EE_PRODUCT('default description','IPEM Solder',
      'IPEM layer 4 solder',#520,.F.);
```

```
#3102=EE_PRODUCT_VERSION(#520,$,'rev 1',#3101,#30,#100);
#3103=DESIGN_LAYER_TECHNOLOGY(.INTERNAL.,#3110,#3104,#3105,
      'solder technology',#3105,$,#3105,$,$,.POWER_OR_GROUND.);
#3104=LENGTH_DATA_ELEMENT(0,'METRE',.MILLI.);
#3105=LENGTH_DATA_ELEMENT(0,'METRE',.MILLI.);
#3106=STRATUM_SURFACE(#3100,.PRIMARY_SURFACE.);
#3107=STRATUM_SURFACE(#3100,.SECONDARY_SURFACE.);
#3108=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#3100,
      'IPEM layer 4 solder');
#3109=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 4-3
      Solder_DBC_Cu',#3107,#2060);
#3110=EE_MATERIAL('Solder',.CONDUCTIVE.,.NON_CONDUCTIVE.,#170,
      .CONDUCTIVE.);
#3500=STRATUM((#60),'product definition id',
      #540,#40,#3502,#530,#10,$,$,#40,'Ceramic Carrier',$,#3530);
#3501=EE_PRODUCT('default descritption','Ceramic Carrier',
      'IPEM layer 4 CC',#520,.F.);
#3502=EE_PRODUCT_VERSION(#520,$,'rev code',#3501,#30,#100);
#3510=STRATUM_SURFACE(#3500,.PRIMARY_SURFACE.);
#3520=STRATUM_SURFACE(#3500,.SECONDARY_SURFACE.);
#3530=STRATUM_TECHNOLOGY(.INTERNAL.,#3540,#3550,#3560,
      'surrounding ceramic technology',$,$,$,$,$);
#3540=EE_MATERIAL('Aluminum Oxide',
      .NON_CONDUCTIVE.,.NON_CONDUCTIVE.,#170,.RESISTIVE.);
#3550=LENGTH_DATA_ELEMENT(35,'INCH',.MILLI.);
#3560=LENGTH_DATA_ELEMENT(35,'INCH',.MILLI.);
#3570=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#3500,
      'IPEM layer 4 surrounding ceramic');
#3580=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 4-3 CC_DBC_Cu',
      #3520,#2060);

/* add another layer of dieletric on top of die */
#3700=STRATUM((#60),'product definition
id',#540,#40,#3702,#530,#10,$,$,#40,'Interlayer dieletric',$,#3730);
#3701=EE_PRODUCT('default descritption','Interlayer dielectric',
      'IPEM layer 5',#520,.F.);
#3702=EE_PRODUCT_VERSION(#520,$,'rev code',#3701,#30,#100);
#3710=STRATUM_SURFACE(#3700,.PRIMARY_SURFACE.);
```

```
#3720=STRATUM_SURFACE(#3700,.SECONDARY_SURFACE.);
#3730=STRATUM_TECHNOLOGY(.INTERNAL.,#3740,#3750,#3760,
      'interlayer technology',$,$,$,$,$);
#3740=EE_MATERIAL('dielectric
material',.NON_CONDUCTIVE.,.NON_CONDUCTIVE.,#170,.RESISTIVE.);
#3750=LENGTH_DATA_ELEMENT(0,'INCH',.MILLI.);
#3760=LENGTH_DATA_ELEMENT(0,'INCH',.MILLI.);
#3770=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#3700,
      'IPEM layer 5 interlayer dieletric');
#3780=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 5-4 IntLayer_CC',
      #3720,#3510);

#4000=DESIGN_LAYER_STRATUM((#60),'product def
id',#540,#40,#4002,#530,#10,$,$,#40,'sputtered copper',$,#4010,.T.);
#4001=EE_PRODUCT('default description','Sputtered Copper',
      'IPEM layer 6',#520,.F.);
#4002=EE_PRODUCT_VERSION(#520,$,'rev code',#4002,#30,#100);
#4010=DESIGN_LAYER_TECHNOLOGY(.PRIMARY.,#180,#4020,#4030,
      'metalization technology',#4030,$,#4030,$,$,.POWER_OR_GROUND.);
#4020=LENGTH_DATA_ELEMENT(0.4,'METRE',.MILLI.);
#4030=LENGTH_DATA_ELEMENT(0.3,'METRE',.MILLI.);
#4040=STRATUM_SURFACE(#4000,.PRIMARY_SURFACE.);
#4050=STRATUM_SURFACE(#4000,.SECONDARY_SURFACE.);
#4060=INTERCONNECT_MODULE_STRATUM_ASSEMBLY_RELATIONSHIP(#1000,#4000,
      'IPEM layer 6 Sputtered copper');
#4070=ADJACENT_STRATUM_SURFACE_DEFINITION('IPEM layer 6-5
      SCu_IntLayer',#4050,#3710);

#4500=ADJACENT_STRATUM_SURFACE_EMBEDDED_COMPONENT_SURFACE_DEFINITION(
      'IPEM layer 5-4 IntLayer_Dev1',#3720,#1031);
#4510=ADJACENT_STRATUM_SURFACE_EMBEDDED_COMPONENT_SURFACE_DEFINITION(
      'IPEM layer 4-4 Dev1_Solder',#1033,#3106);
#4520=ADJACENT_STRATUM_SURFACE_EMBEDDED_COMPONENT_SURFACE_DEFINITION(
      'IPEM layer 5-4 IntLayer_Dev2',#3720,#1131);
#4530=ADJACENT_STRATUM_SURFACE_EMBEDDED_COMPONENT_SURFACE_DEFINITION(
      'IPEM layer 4-4 Dev2_Solder',#1133,#3106);

/* straum features in the trace layer */
```

```
#5000=STRATUM_FEATURE('N terminal','DEFAULT DESCRIPTION',#2050,.T.);
#5001=CONTACT_SIZE_DEPENDENT_NON_FUNCTIONAL_LAND(#5002,$,#5000,$,$);
#5002=DEFAULT_ATTACHMENT_SIZE_BASED_LAND_PHYSICAL_TEMPLATE(
      'IPEM PNO terminal template',
      'default description',$,$,$,#5003,#2040,$,#5011,#5012);
#5003=TEMPLATE_DEFINITION((#60),'ipem pno template definition',
      #540,#40,#5004,#530,#10,$,$,#40,'conceptual design');
#5004=EE_PRODUCT_VERSION(#520,$,'rev code',#5005,#30,#100);
#5005=EE_PRODUCT('default descritption','template product',
      'IPEM PNO Terminal template product',#520,.F.);
#5006=EE_PRODUCT_RELATED_PRODUCT_CATEGORY($,$,'template model',
      #1,(#5005,#5126,#5145,#5615,#5625,#5675,#5775));
#5007=NON_FUNCTIONAL_LAND_JOIN_TERMINAL(#5008,#5001,$,$,$);
#5008=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal',
      'default description',#5002,$,.SURFACE_AREA.);
#5009=NON_FUNCTIONAL_LAND_INTERFACE_TERMINAL(#5010,#5001);
#5010=LAND_TEMPLATE_INTERFACE_TERMINAL('land interface terminal',
      'default description',#5002,$,.SURFACE_AREA.);
#5011=CONNECTION_ZONE($);
#5012=CONNECTION_ZONE($);
#5013=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5001,#30,$,
      'IPEM layer 3 N terminal',$);
#5014=
COMPONENT_TERMINAL_TO_INTERCONNECT_MODULE_INTERFACE_TERMINAL_ASSIGNMENT(
#1020,#5009);

/* straum features in the trace layer */
#5100=CONDUCTOR('O terminal and trace to dev 1 drain',
      'DEFAULT DESCRIPTION',#2050,.T.,#5101);
#5101=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5102,#5103));
#5102=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5104,$,
      'connection point to O terminal',#2050,#1021,$);
#5103=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5105,$,
      'connection point to dev 1 solder',#2050,#1034,$);
#5104=CARTESIAN_POINT();
#5105=CARTESIAN_POINT();
```

```
#5107=STRATUM_FEATURE_PLANAR_SHAPE(#5108,
      (#5104,#5105,#5135,#5136,#5139,#5140,#5148,#5149,#5150,#5151),
      #5100);
#5108=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);
#5110=CONTACT_SIZE_DEPENDENT_LAND(#5002,$,#5100,$,$);
#5111=LAND_JOIN_TERMINAL(#5008,#5110,$,$,$);
#5112=LAND_INTERFACE_TERMINAL(#5010,#5110);
#5113=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5110,#30,$,
      'IPEM layer 3 O terminal',$);
#5114=
COMPONENT_TERMINAL_TO_INTERCONNECT_MODULE_INTERFACE_TERMINAL_ASSIGNMENT(
#1021,#5112);
#5120=CONTACT_SIZE_DEPENDENT_LAND(#5121,$,#5100,$,$);
#5121=DEFAULT_ATTACHMENT_SIZE_BASED_LAND_PHYSICAL_TEMPLATE(
      'land for solder attachment template',
      'default description',$,$,$,#5124,#2040,$,#5122,#5123);
#5122=CONNECTION_ZONE($);
#5123=CONNECTION_ZONE($);
#5124=TEMPLATE_DEFINITION((#60),'product def ID',
      #540,#40,#5125,#530,#10,$,$,#40,'conceptual design');
#5125=EE_PRODUCT_VERSION(#520,$,'rev code',#5126,#30,#100);
#5126=EE_PRODUCT('default descritption','template product',
      'IPEM solder attached land template',#520,.F.);
#5127=LAND_JOIN_TERMINAL(#5128,#5120,$,$,$);
#5128=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to trace',
      'default description',#5121,$,.SURFACE_POINT.);
#5129=LAND_JOIN_TERMINAL(#5130,#5120,$,$,$);
#5130=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to solder',
      'default description',#5121,$,.SURFACE_AREA.);
#5131=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5120,#30,$,
      'IPEM layer 3 dev 1 solder land',$);

#5132=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5100,#5133,#5134,#5135,#5136,#5137,#5138);

#5133=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5141,#5143);
```

```
#5134=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5142,#5143);
#5135=TRIMMED_LINE(#5148,#5149);
#5136=TRIMMED_LINE(#5150,#5151);
#5137=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5139,$,
      'start terminal point 1',#2050);
#5138=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5140,$,
      'end terminal point 1',#2050);
#5139=CARTESIAN_POINT();
#5140=CARTESIAN_POINT();
#5141=TEMPLATE_DEFINITION((#60),'start template 1',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5142=TEMPLATE_DEFINITION((#60),'end template 1',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5143=CURVE_STYLE('trace template curve style',
      .ROUND.,#5146,$,.EXTEND.,.SQUARE.,#5147);
#5144=EE_PRODUCT_VERSION(#520,$,'rev code',#5145,#30,#100);
#5145=EE_PRODUCT('default descritption','template product',
      'IPEM trace template',#520,.F.);
#5146=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5147=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5148=CARTESIAN_POINT();
#5149=CARTESIAN_POINT();
#5150=CARTESIAN_POINT();
#5151=CARTESIAN_POINT();
#5152=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5132,#30,$,
      'IPEM layer 3 O terminal trace',$);
#5153=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5132,$,
      'terminal to O',#5137);
#5154=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5132,$,
      'terminal to land',#5138);
#5155=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5100,$);
#5156=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5100,$);
#5157=FABRICATION_JOINT(#5111,#5153,#5155,#1000);
#5158=FABRICATION_JOINT(#5154,#5127,#5156,#1000);

/* straum features in the trace layer */
#5200=CONDUCTOR('P terminal and trace to dev 2 drain',
      'DEFAULT DESCRIPTION',#2050,.T.,#5201);
```

```
#5201=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5202,#5203));
#5202=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5204,$,
      'connection point to P terminal',#2050,#1022,$);
#5203=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5205,$,
      'connection point to dev 2 solder',#2050,#1134,$);
#5204=CARTESIAN_POINT();
#5205=CARTESIAN_POINT();
#5206=PHYSICAL_NETWORK(#1000,$,(#1134,#5211),(#5201,#5401,#5424),
      'PHYSICAL NODE P',$);
#5207=STRATUM_FEATURE_PLANAR_SHAPE(#5208,(#5204,#5205,#5235,#5236,
      #5239,#5240,#5248,#5249,#5250,#5251),#5200);
#5208=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);
#5210=CONTACT_SIZE_DEPENDENT_LAND(#5002,$,#5200,$,$);
#5211=LAND_JOIN_TERMINAL(#5008,#5210,$,$,$);
#5212=LAND_INTERFACE_TERMINAL(#5010,#5210);
#5213=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5210,#30,$,
      'IPEM layer 3 P terminal',$);
#5214=
COMPONENT_TERMINAL_TO_INTERCONNECT_MODULE_INTERFACE_TERMINAL_ASSIGNMENT(
#1022,#5212);
#5220=CONTACT_SIZE_DEPENDENT_LAND(#5121,$,#5200,$,$);
#5227=LAND_JOIN_TERMINAL(#5128,#5220,$,$,$);
#5229=LAND_JOIN_TERMINAL(#5130,#5220,$,$,$);
#5231=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5220,#30,$,
      'IPEM layer 3 dev 2 solder land',$);

/* WD 34 version */
#5232=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5200,#5233,#5234,#5235,#5236,#5237,#5238);

#5233=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5241,#5143);
#5234=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5242,#5143);
#5235=TRIMMED_LINE(#5248,#5249);
#5236=TRIMMED_LINE(#5250,#5251);
#5237=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5239,$,
      'start terminal point 2',#2050);
```

```
#5238=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5240,$,
      'end terminal point 2',#2050);
#5239=CARTESIAN_POINT();
#5240=CARTESIAN_POINT();
#5241=TEMPLATE_DEFINITION((#60),'start template 2',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5242=TEMPLATE_DEFINITION((#60),'end template 2',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5248=CARTESIAN_POINT();
#5249=CARTESIAN_POINT();
#5250=CARTESIAN_POINT();
#5251=CARTESIAN_POINT();
#5252=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5232,#30,$,
      'IPEM layer 3 P terminal trace',$);
#5253=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5232,$,
      'terminal to P',#5237);
#5254=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5232,$,
      'terminal to land',#5238);
#5255=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5200,$);
#5256=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5200,$);
#5257=FABRICATION_JOINT(#5211,#5253,#5255,#1000);
#5258=FABRICATION_JOINT(#5254,#5227,#5256,#1000);

/* end of trace layer feature */

/* straum features in the solder layer */
#5300=CONDUCTIVE_FILLED_AREA('Solder to dev 1','default
label',#3100,.F.);
#5301=INTER_STRATUM_JOIN_RELATIONSHIP(#1000,(#5302,#5103));
#5302=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5303,$,
      'connection point to dev 1 drain',#3100,#1034,$);
#5303=CARTESIAN_POINT();
#5304=STRATUM_FEATURE_PLANAR_SHAPE(#5305,(#5304),#5300);
#5305=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);
#5306=CONTACT_SIZE_DEPENDENT_NON_FUNCTIONAL_LAND(#5307,$,#5300,$,$);
```

```
#5307=DEFAULT_ATTACHMENT_SIZE_BASED_LAND_PHYSICAL_TEMPLATE        ('land
for device attachment template',                        'default
description',$,$,$,#5310,#3103,$,#5308,#5309);
#5308=CONNECTION_ZONE($);
#5309=CONNECTION_ZONE($);
#5310=TEMPLATE_DEFINITION((#60),'product def ID',
      #540,#40,#5311,#530,#10,$,$,#40,'conceptual design');
#5311=EE_PRODUCT_VERSION(#520,$,'rev code',#5312,#30,#100);
#5312=EE_PRODUCT('default descritption','template product',
      'IPEM device attached land template',#520,.F.);
#5313=LAND_JOIN_TERMINAL(#5314,#5306,$,$,$);
#5314=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to device',
      'default description',#5307,$,.SURFACE_AREA.);
#5315=LAND_JOIN_TERMINAL(#5316,#5306,$,$,$);
#5316=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to trace',
      'default description',#5307,$,.SURFACE_AREA.);
#5317=FABRICATION_JOINT(#5315,#5129,$,#1000);
#5318=FABRICATION_JOINT(#1034,#5313,$,#1000);
#5319=PHYSICAL_NETWORK_SUPPORTING_STRATUM_FEATURE_CONDUCTIVE_JOIN($,
      (#5320),#5306,#5120,#5301);
#5320=EE_REQUIREMENT_OCCURRENCE(#5321,$,$,(#1010),$,$,$,$,$,
      'solder requirement for device');
#5321=REQUIREMENT_DEFINITION((#60),'solder requirement',
      #540,#42,#5322,#530,#10,$,$,#40,'conceptual design');
#5322=EE_PRODUCT_VERSION(#520,$,'rev code',#5323,#30,#100);
#5323=EE_PRODUCT('default descritption','solder requirement',
      'requirement for cutout1',#520,.F.);
#5324=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1034),
      (#5302));
#5325=DIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5318,#5324);
#5326=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5306,#30,$,
      'IPEM layer 4 dev 1 land',$);

#5400=CONDUCTIVE_FILLED_AREA('Solder to dev 2','default label',
      #3100,.F.);
#5401=INTER_STRATUM_JOIN_RELATIONSHIP(#1000,(#5402,#5203));
#5402=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5403,$,
      'connection point to dev 1 drain',#3100,#1134,$);
```

```
#5403=CARTESIAN_POINT();
#5404=STRATUM_FEATURE_PLANAR_SHAPE(#5405,(#5404),#5400);
#5405=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);
#5406=CONTACT_SIZE_DEPENDENT_NON_FUNCTIONAL_LAND(#5307,$,#5400,$,$);
#5413=LAND_JOIN_TERMINAL(#5314,#5406,$,$,$);
#5415=LAND_JOIN_TERMINAL(#5316,#5406,$,$,$);
#5417=FABRICATION_JOINT(#5415,#5229,$,#1000);
#5418=FABRICATION_JOINT(#1134, #5413,$,#1000);
#5419=PHYSICAL_NETWORK_SUPPORTING_STRATUM_FEATURE_CONDUCTIVE_JOIN($,
      (#5320),#5406,#5220,#5401);
#5424=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1134),
      (#5402));
#5425=DIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5418,#5424);
#5426=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5406,#30,$,
      'IPEM layer 4 dev 2 land',$);

/* end of solder layer feature */

/* cutouts for the dies */
#5500=CUTOUT(#5501,$,.F.);
#5501=UNSUPPORTED_PASSAGE_TEMPLATE('cutout for die','',
      $,$,$,#5502,#5503);
#5502=TEMPLATE_DEFINITION((#60),'die cutout
template',#540,#40,#5504,#530,#10,$,$,#40,'conceptual design');
#5503=DEFAULT_UNSUPPORTED_PASSAGE_DEFINITION(#1000,#5506,$,.F.,$,#5507,
      'DIE CUTOUT PASSAGE TECH',$,$,$,$);
#5504=EE_PRODUCT_VERSION(#520,$,'rev code',#5505,#30,#100);
#5505=EE_PRODUCT('default descritption','template product',
      'IPEM die cutout template',#520,.F.);
#5506=INTER_STRATUM_EXTENT(#3500,#3500);
#5507=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5508=INTERCONNECT_MODULE_CONSTRAINT_REGION('','', #1000,.F.,$,
      'DIE CUTOUT',#5509,(#3500),.INTERCONNECT_MODULE_CUTOUT.);
#5509=EE_REQUIREMENT_OCCURRENCE(#5510,$,$,(#1010),$,$,$,$,$,
      'requirement for cutout location');
#5510=REQUIREMENT_DEFINITION((#60),'cutout1',#540,#42,#5511,#530,#10,$,
      $,#40,'conceptual design');
```

```
#5511=EE_PRODUCT_VERSION(#520,$,'rev code',#5512,#30,#100);
#5512=EE_PRODUCT('default descritption','requirement for cutout1',
      'requirement for cutout1',#520,.F.);
#5513=(CSG_2D_SHAPE()MANAGED_DESIGN_OBJECT()
      NON_FEATURE_SHAPE_DEFINITION(#5508,$)
      PLANAR_SHAPE()SHAPE_DEFINITION(#1060,$));
#5514=EE_PRODUCT_RELATED_PRODUCT_CATEGORY($,$,'requirement models',
      #1,(#5323, #5512));
#5515=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5500,#30,$,
      'IPEM layer 4 dev 1 cutout',$);
#5516=CUTOUT(#5501,$,.F.);
#5517=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5516,#30,$,
      'IPEM layer 4 dev 2 cutout',$);

/* VIA definition for dielectric layer and sputtered copper */
#5550=VIA_TEMPLATE('CONNECTION to die terminals',' ',
      $,$,$,#5551,#5552);
#5551=TEMPLATE_DEFINITION((#60),'die terminal via template',
      #540,#40,#5553,#530,#10,$,$,#40,'conceptual design');
#5552=DEFAULT_VIA_DEFINITION(#1000,#5555,#180,.T.,$,#5556,
      'DIE TERMINAL VIA PASSAGE TECH',$,$,#5556,$);
#5553=EE_PRODUCT_VERSION(#520,$,'rev code',#5554,#30,#100);
#5554=EE_PRODUCT('default descritption','template product',
      'IPEM die terminal via template',#520,.F.);
#5555=INTER_STRATUM_EXTENT(#4000,#3700);
#5556=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5557=VIA_TEMPLATE_TERMINAL('1','via terminal outter',#5550,$,$);
#5558=VIA_TEMPLATE_TERMINAL('2','via terminal bottom',#5550,$,(#5559));
#5559=CONNECTION_ZONE($);

#5560=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5678,#5676)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5561=VIA_TERMINAL(#5557,#5560);
#5562=VIA_TERMINAL(#5558,#5560);
```

```
#5563=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5560,#30,$,
      'dev 1 source via 1',$);



#5564=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5678,#5677)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5565=VIA_TERMINAL(#5557,#5564);
#5566=VIA_TERMINAL(#5558,#5564);
#5567=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5564,#30,$,
      'dev 1 source via 2',$);

#5568=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5775,#5774)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5569=VIA_TERMINAL(#5557,#5568);
#5570=VIA_TERMINAL(#5558,#5568);
#5571=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5568,#30,$,
      'dev 1 gate via',$);

#5572=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5878,#5876)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5573=VIA_TERMINAL(#5557,#5572);
#5574=VIA_TERMINAL(#5558,#5572);
#5575=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5572,#30,$,
      'dev 2 source via 1',$);
```

```
#5576=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5878,#5877)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5577=VIA_TERMINAL(#5557,#5576);
#5578=VIA_TERMINAL(#5558,#5576);
#5579=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5576,#30,$,
      'dev 2 source via 2',$);

#5580=(ASSEMBLY_COMPONENT(#5550)BLIND_VIA()INTER_STRATUM_FEATURE(.F.)
      INTER_STRATUM_JOIN_IMPLEMENTATION()
      LAMINATE_COMPONENT($)MANAGED_DESIGN_OBJECT()
      INDIRECT_STRATUM_COMPONENT_JOIN_IMPLEMENTATION(#5975,#5974)
      PLATED_CONDUCTIVE_BASE_BLIND_VIA()PLATED_INTER_STRATUM_FEATURE()
      PLATED_PASSAGE()VIA());
#5581=VIA_TERMINAL(#5557,#5580);
#5582=VIA_TERMINAL(#5558,#5580);
#5583=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5580,#30,$,
      'dev 2 gate via',$);

/* ADD inter_stratum_feature, and template for connect between trace and
sputter, AND JOIN_RELS, and assembly relation */
#5584=INTER_STRATUM_FEATURE_TEMPLATE('',                          'joint
from sputter cu to trace',$,$,$,#5585,$5586);
#5585=TEMPLATE_DEFINITION((#60),'template',#540,#40,#5587,#530,#10,
      $,$,#40,'conceptual design');
#5586=PASSAGE_TECHNOLOGY(#1000,#5589,#180,.T.,$,$,
      'EDGE PASSAGE TECH',$,$,$,$);
#5587=EE_PRODUCT_VERSION(#520,$,'rev code',#5588,#30,#100);
#5588=EE_PRODUCT('default descritption','template product',
      'IPEM edge connection template',#520,.F.);
#5589=INTER_STRATUM_EXTENT(#4000,#2050);
#5590=PHYSICAL_NETWORK_SUPPORTING_INTER_STRATUM_FEATURE(#5584,$,.F.,
      #5591);
#5591=INTER_STRATUM_JOIN_RELATIONSHIP(#5602,#5002);
```

```
#5592=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5590,#30,$,
      'inter_sf N',$);
#5593=PHYSICAL_NETWORK_SUPPORTING_INTER_STRATUM_FEATURE(#5584,$,.F.,
      #5594);
#5594=INTER_STRATUM_JOIN_RELATIONSHIP(#5802,#5102);
#5595=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5593,#30,$,
      'inter_sf O',$);

/* sputtered copper features follows */
/* conductor to dev 1 source */
#5600=CONDUCTOR('N terminal and trace to dev 1 source',
      'DEFAULT DESCRIPTION',#4000,.T.,#5601);
#5601=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5602,#5603,#5604));
#5602=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5605,$,
      'connection point to N terminal',#4000,#5590,$);
#5603=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5606,$,
      'connection point 1 to dev 1 source',#4000,#1032,#190);
#5604=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5607,$,
      'connection point 2 to dev 1 source',#4000,#1032,#191);
#5605=CARTESIAN_POINT();
#5606=CARTESIAN_POINT();
#5607=CARTESIAN_POINT();
#5608=STRATUM_FEATURE_PLANAR_SHAPE(#5609,(#5605,#5606,#5607,#5653,
      #5654,#5657,#5658,#5659,#5660,#5661,#5662),#5600);
#5609=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);

#5610=DEFAULT_VIA_BASED_LAND_PHYSICAL_TEMPLATE(
      'sputtered copper via  based land template',
      'default description',$,$,$,#5611,#4010,$5612,#5552,#5555,#5613);
#5611=TEMPLATE_DEFINITION((#60),
      'sputtered copper via based template definition',
      #540,#40,#5614,#530,#10,$,$,#40,'conceptual design');
#5612=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5613=LENGTH_DATA_ELEMENT(0.0,'METRE',.MILLI.);
#5614=EE_PRODUCT_VERSION(#520,$,'rev code',#5615,#30,#100);
#5615=EE_PRODUCT('default descritption','template product',
      'Sputtered copper via based land template product',#520,.F.);
```

```
#5616=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to via',
      'default description',#5610,$,.EDGE_CURVE.);
#5617=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal to trace',
      'default description',#5610,$,.SURFACE_POINT.);

#5620=LAND_PHYSICAL_TEMPLATE('IPEM PNO terminal template',
      'default description',$,$,$,#5621,#4010,$);
#5621=TEMPLATE_DEFINITION((#60),'ipem pno template definition',
      #540,#40,#5624,#530,#10,$,$,#40,'conceptual design');
#5624=EE_PRODUCT_VERSION(#520,$,'rev code',#5625,#30,#100);
#5625=EE_PRODUCT('default descritption','template product',
      'IPEM sputtered layer terminal template product',#520,.F.);
#5626=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal',
      'default description',#5620,$,.SURFACE_POINT.);
#5627=LAND_TEMPLATE_JOIN_TERMINAL('land join terminal',
      'default description',#5620,$,.SURFACE_AREA.);

/* lands to vias to dev 1 source */
#5630=VIA_DEPENDENT_LAND(#5610,$,#5600,$,$);
#5631=LAND_JOIN_TERMINAL(#5616,#5630,$,$,$);
#5632=LAND_JOIN_TERMINAL(#5617,#5630,$,$,$);
#5633=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5630,#30,$,
      'IPEM layer 6 land to dev1 source via 1',$);
#5634=VIA_DEPENDENT_LAND(#5610,$,#5600,$,$);
#5635=LAND_JOIN_TERMINAL(#5616,#5634,$,$,$);
#5636=LAND_JOIN_TERMINAL(#5617,#5634,$,$,$);
#5637=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5634,#30,$,
      'IPEM layer 6 land to dev1 source via 2',$);

/* land to IPEM terminal N */
#5640=LAND(#5620,$,#5600,$,$);
#5641=LAND_JOIN_TERMINAL(#5626,#5640,$,$,$);
#5642=LAND_JOIN_TERMINAL(#5627,#5640,$,$,$);
#5643=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5640,#30,$,
      'IPEM layer 6 land for terminal N',$);

/* trace */
/* WD 34 version */
```

```
#5650=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5600,#5651,#5652,#5653,#5654,#5655,#5656);

#5651=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5663,#5143);
#5652=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5664,#5143);
#5653=TRIMMED_LINE(#5657,#5658);
#5654=TRIMMED_LINE(#5659,#5660);
#5655=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5661,$,          'start
terminal point 3',#4000);
#5656=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5662,$,
      'end terminal point 3',#4000);
#5657=CARTESIAN_POINT();
#5658=CARTESIAN_POINT();
#5659=CARTESIAN_POINT();
#5660=CARTESIAN_POINT();
#5661=CARTESIAN_POINT();
#5662=CARTESIAN_POINT();
#5663=TEMPLATE_DEFINITION((#60),'start template 3',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5664=TEMPLATE_DEFINITION((#60),'end template 3',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5665=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5650,#30,$,
      'IPEM layer 6 trace to dev 1 source',$);
#5666=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5650,$,
      'terminal to tn',#5655);
#5667=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5650,$,
      'terminal to dev 1 source',#5656);


#5669=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5600,$);
#5670=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5600,$);
/* fab_joint between cie and land to terminal tn */
#5671=FABRICATION_JOINT(#5641,#5666,#5669,#1000);
/* fab_joint between cie and lands to vias */
#5672=FABRICATION_JOINT(#5667,#5632,#5670,#1000);
#5673=FABRICATION_JOINT(#5667,#5636,#5670,#1000);
/* fab_joint between lands and vias */
```

```
#5674=FABRICATION_JOINT(#5561,#5631,$,#1000);
#5675=FABRICATION_JOINT(#5565,#5635,$,#1000);
/* fab_joint between vias and die terminal connection zones*/
#5676=CONNECTION_ZONE_BASED_FABRICATION_JOINT(#1032,#5562,$,
      #1000,#190,#5559);
#5677=CONNECTION_ZONE_BASED_FABRICATION_JOINT(#1032,#5566,$,
      #1000,#191,#5559);
#5678=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1032),
      (#5603,#5604));
#5679=PHYSICAL_NETWORK(#1000,$,(#1032,#5007),(#5591,#5601,#5678),
      'PHYSICAL NODE N',$);


/* conductor to dev1 gate */
#5700=CONDUCTOR('G1 terminal and trace to dev 1 gate',
      'DEFAULT DESCRIPTION',#4000,.T.,#5701);
#5701=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5702,#5703));
#5702=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5705,$,
      'connection point to G1 terminal',#4000,#1023,$);
#5703=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5706,$,
      'connection point to dev 1 gate',#4000,#1035,$);
#5705=CARTESIAN_POINT();
#5706=CARTESIAN_POINT();
#5708=STRATUM_FEATURE_PLANAR_SHAPE(#5709,(#5705,#5706,#5753,#5754,
      #5757,#5758,#5759,#5760,#5761,#5762),#5700);
#5709=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);

/* lands to vias to dev 1 gate */
#5730=VIA_DEPENDENT_LAND(#5610,$,#5700,$,$);
#5731=LAND_JOIN_TERMINAL(#5616,#5730,$,$,$);
#5732=LAND_JOIN_TERMINAL(#5617,#5730,$,$,$);
#5733=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5730,#30,$,
      'IPEM layer 6 land to dev 1 gate via',$);

/* land to IPEM terminal G1 */
#5740=CONTACT_SIZE_DEPENDENT_LAND(#5620,$,#5700,$,$);
#5741=LAND_JOIN_TERMINAL(#5626,#5740,$,$,$);
```

```
#5742=LAND_INTERFACE_TERMINAL(#5627,#5740);
#5743=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5740,#30,$,
      'IPEM layer 6 land for terminal G1',$);


#5744=
COMPONENT_TERMINAL_TO_INTERCONNECT_MODULE_INTERFACE_TERMINAL_ASSIGNMENT(
#1023,#5742);

/* trace */
/* WD 34 version */
#5750=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5700,#5751,#5752,#5753,#5754,#5755,#5756);

#5751=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5763,#5143);
#5752=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5764,#5143);
#5753=TRIMMED_LINE(#5757,#5758);
#5754=TRIMMED_LINE(#5759,#5760);
#5755=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5761,$,
      'start terminal point 4',#4000);
#5756=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5762,$,
      'end terminal point 4',#4000);
#5757=CARTESIAN_POINT();
#5758=CARTESIAN_POINT();
#5759=CARTESIAN_POINT();
#5760=CARTESIAN_POINT();
#5761=CARTESIAN_POINT();
#5762=CARTESIAN_POINT();
#5763=TEMPLATE_DEFINITION((#60),'start template 4',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5764=TEMPLATE_DEFINITION((#60),'end template 4',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5765=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5750,#30,$,
      'IPEM layer 6 trace to dev 1 gate',$);
#5766=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5750,$,
      'terminal to G1',#5755);
```

```
#5767=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5750,$,
      'terminal to dev 1 gate',#5756);
#5769=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5700,$);
#5770=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5700,$);
/* fab_joint between cie and land to terminal G1 */
#5771=FABRICATION_JOINT(#5741,#5766,#5769,#1000);
/* fab_joint between cie and lands to vias */
#5772=FABRICATION_JOINT(#5767,#5732,#5670,#1000);
/* fab_joint between lands and vias */
#5773=FABRICATION_JOINT(#5569,#5731,$,#1000);
/* fab_joint between via and die gate*/
#5774=FABRICATION_JOINT(#1035,#5570,$,#1000);
#5775=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1035),
      (#5703));
#5776=PHYSICAL_NETWORK(#1000,$,(#1035,#5741),(#5701,#5775),
      'PHYSICAL NODE G1',$);

/* conductor to dev 2 source */
#5800=CONDUCTOR('O terminal and trace to dev 2 source',
      'DEFAULT DESCRIPTION',#4000,.T.,#5801);
#5801=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5802,#5803,#5804));
#5802=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5805,$,
      'connection point to O terminal',#4000,#5593,$);
#5803=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5806,$,
      'connection point 1 to dev 1 source',#4000,#1132,#190);
#5804=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5807,$,
      'connection point 2 to dev 1 source',#4000,#1132,#191);
#5805=CARTESIAN_POINT();
#5806=CARTESIAN_POINT();
#5807=CARTESIAN_POINT();
#5808=STRATUM_FEATURE_PLANAR_SHAPE(#5809,(#5805,#5806,#5807,#5853,
      #5854,#5857,#5858,#5859,#5860,#5861,#5862),#5800);
#5809=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);

/* lands to vias to dev 2 source */
#5830=VIA_DEPENDENT_LAND(#5610,$,#5800,$,$);
#5831=LAND_JOIN_TERMINAL(#5616,#5830,$,$,$);
```

```
#5832=LAND_JOIN_TERMINAL(#5617,#5830,$,$,$);
#5833=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5830,#30,$,
      'IPEM layer 6 land to dev 2 source via 1',$);
#5834=VIA_DEPENDENT_LAND(#5610,$,#5800,$,$);
#5835=LAND_JOIN_TERMINAL(#5616,#5834,$,$,$);
#5836=LAND_JOIN_TERMINAL(#5617,#5834,$,$,$);
#5837=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5834,#30,$,
      'IPEM layer 6 land to dev2 source via 2',$);

/* land to IPEM terminal O */
#5840=LAND(#5620,$,#5800,$,$);
#5841=LAND_JOIN_TERMINAL(#5626,#5840,$,$,$);
#5842=LAND_JOIN_TERMINAL(#5627,#5840,$,$,$);
#5843=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5840,#30,$,
      'IPEM layer 6 land for terminal O',$);

/* trace */

/* WD 34 version */
#5850=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5800,#5851,#5852,#5853,#5854,#5855,#5856);

#5851=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5863,#5143);
#5852=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5864,#5143);
#5853=TRIMMED_LINE(#5857,#5858);
#5854=TRIMMED_LINE(#5859,#5860);
#5855=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5861,$,          'start
terminal point 5',#4000);
#5856=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5862,$,
      'end terminal point 5',#4000);
#5857=CARTESIAN_POINT();
#5858=CARTESIAN_POINT();
#5859=CARTESIAN_POINT();
#5860=CARTESIAN_POINT();
#5861=CARTESIAN_POINT();
#5862=CARTESIAN_POINT();
```

```
#5863=TEMPLATE_DEFINITION((#60),'start template 5',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5864=TEMPLATE_DEFINITION((#60),'end template 5',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5865=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5850,#30,$,
      'IPEM layer 6 trace to dev 2 source',$);
#5866=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5850,$,
      'terminal to tn',#5855);
#5867=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5850,$,
      'terminal to dev 1 source',#5856);

/* WD 36_2 version */
#5868=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL_LINK(#5850,#5866,#5867);

#5869=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5800,$);
#5870=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5800,$);
/* fab_joint between cie and land to terminal tn */
#5871=FABRICATION_JOINT(#5841,#8666,#5869,#1000);
/* fab_joint between cie and lands to vias */
#5872=FABRICATION_JOINT(#5867,#5832,#5870,#1000);
#5873=FABRICATION_JOINT(#5867,#5836,#5870,#1000);
/* fab_joint between lands and vias */
#5874=FABRICATION_JOINT(#5573,#5831,$,#1000);
#5875=FABRICATION_JOINT(#5577,#5835,$,#1000);
/* fab_joint between vias and die terminal connection zones*/
#5876=CONNECTION_ZONE_BASED_FABRICATION_JOINT(#1132,#5574,$,
      #1000,#190,#5559);
#5877=CONNECTION_ZONE_BASED_FABRICATION_JOINT(#1132,#5578,$,
      #1000,#191,#5559);
#5878=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1132),
      (#5803,#5804));
#5879=PHYSICAL_NETWORK(#1000,$,(#1034,#1132,#5111),
      (#5101,#5301,#5324,#5594,#5801,#5878),'PHYSICAL NODE O',$);


/* conductor to dev2 gate */
#5900=CONDUCTOR('G2 terminal and trace to dev 2 gate',
      'DEFAULT DESCRIPTION',#4000,.T.,#5901);
```

```
#5901=INTRA_STRATUM_JOIN_RELATIONSHIP(#1000,(#5902,#5903));
#5902=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5905,$,
      'connection point to G2 terminal',#4000,#1024,$);
#5903=DEPENDENTLY_LOCATED_LAYER_CONNECTION_POINT(#5906,$,
      'connection point to dev 2 gate',#4000,#1135,$);
#5905=CARTESIAN_POINT();
#5906=CARTESIAN_POINT();
#5908=STRATUM_FEATURE_PLANAR_SHAPE(#5909,
      (#5905,#5906,#5953,#5954,#5957,#5958,#5959,#5960,#5961,#5962),
      #5900);
#5909=CARTESIAN_COORDINATE_SYSTEM('.METRE.',$,$,.TWO_DIMENSIONAL.,
      .RADIAN.,#1050,$);

/* lands to vias to dev 1 gate */
#5930=VIA_DEPENDENT_LAND(#5610,$,#5900,$,$);
#5931=LAND_JOIN_TERMINAL(#5616,#5930,$,$,$);
#5932=LAND_JOIN_TERMINAL(#5617,#5930,$,$,$);
#5933=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5930,#30,$,
      'IPEM layer 6 land to dev 2 gate via',$);

/* land to IPEM terminal G1 */
#5940=CONTACT_SIZE_DEPENDENT_LAND(#5620,$,#5900,$,$);
#5941=LAND_JOIN_TERMINAL(#5626,#5940,$,$,$);
#5942=LAND_INTERFACE_TERMINAL(#5627,#5940);
#5943=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5940,#30,$,
      'IPEM layer 6 land for terminal G2',$);
#5944=
COMPONENT_TERMINAL_TO_INTERCONNECT_MODULE_INTERFACE_TERMINAL_ASSIGNMENT(
#1024,#5942);

/* trace */
/* WD 34 verison */
#5950=
CONDUCTIVE_INTERCONNECT_ELEMENT_WITH_USER_DEFINED_SINGLE_TRANSITION
($,$,#5900,#5951,#5952,#5953,#5954,#5955,#5956);

#5951=TRACE_TEMPLATE('start trace xross section',$,$,$,$,#5963,#5143);
#5952=TRACE_TEMPLATE('end trace xross section',$,$,$,$,#5964,#5143);
```

```
#5953=TRIMMED_LINE(#5957,#5958);
#5954=TRIMMED_LINE(#5959,#5960);
#5955=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5961,$,          'start
terminal point 6',#4000);
#5956=EXPLICITLY_LOCATED_LAYER_CONNECTION_POINT(#5962,$,
      'end terminal point 6',#4000);
#5957=CARTESIAN_POINT();
#5958=CARTESIAN_POINT();
#5959=CARTESIAN_POINT();
#5960=CARTESIAN_POINT();
#5961=CARTESIAN_POINT();
#5962=CARTESIAN_POINT();
#5963=TEMPLATE_DEFINITION((#60),'start template 6',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5964=TEMPLATE_DEFINITION((#60),'end template 6',
      #540,#40,#5144,#530,#10,$,$,#40,'conceptual design');
#5965=NEXT_HIGHER_ASSEMBLY_RELATIONSHIP(#1000,#5950,#30,$,
      'IPEM layer 6 trace to dev 2 gate',$);
#5966=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5950,$,
      'terminal to G2',#5955);
#5967=CONDUCTIVE_INTERCONNECT_ELEMENT_TERMINAL($,#5950,$,
      'terminal to dev 2 gate',#5956);
#5969=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5900,$);
#5970=STRATUM_FEATURE_TEMPLATE_COMPONENT($,$,#5900,$);
/* fab_joint between cie and land to terminal G1 */
#5971=FABRICATION_JOINT(#5941,#5966,#5969,#1000);
/* fab_joint between cie and lands to vias */
#5972=FABRICATION_JOINT(#5967,#5932,#5970,#1000);
/* fab_joint between lands and vias */
#5973=FABRICATION_JOINT(#5581,#5931,$,#1000);
/* fab_joint between via and die gate*/
#5974=FABRICATION_JOINT(#1135,#5582,$,#1000);
#5975=STRATUM_EMBEDDED_COMPONENT_JOIN_RELATIONSHIP(#1000,(#1135),
      (#5903));
#5976=PHYSICAL_NETWORK(#1000,$,(#1135,#5941),(#5901,#5975),
      'PHYSICAL NODE G2',$);
#7000=ELECTRICAL_NETWORK_DEFINITION((#60),'IPEM substrate netlist',
      #540,#42,#1012,#530,#10,$,$,#40,'conceptual design',$,#1004);
```

```
#7100=FUNCTIONAL_UNIT(#130,'mos1',#7000,$);
#7101=FUNCTIONAL_UNIT_TERMINAL(#7100,#250);
#7102=FUNCTIONAL_UNIT_TERMINAL(#7100,#260);
#7103=FUNCTIONAL_UNIT_TERMINAL(#7100,#320);
#7200=FUNCTIONAL_UNIT(#130,'mos2',#7000,$);
#7201=FUNCTIONAL_UNIT_TERMINAL(#7200,#250);
#7202=FUNCTIONAL_UNIT_TERMINAL(#7200,#260);
#7203=FUNCTIONAL_UNIT_TERMINAL(#7200,#320);
#7300=FUNCTIONAL_UNIT_NETWORK_NODE_DEFINITION(#7000,'N');
#7301=FUNCTIONAL_UNIT_NETWORK_NODE_DEFINITION(#7000,'O');
#7302=FUNCTIONAL_UNIT_NETWORK_NODE_DEFINITION(#7000,'P');
#7303=FUNCTIONAL_UNIT_NETWORK_NODE_DEFINITION(#7000,'G1');
#7304=FUNCTIONAL_UNIT_NETWORK_NODE_DEFINITION(#7000,'G2');
#7400=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7300,#7101);
#7401=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7301,#7102);
#7402=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7301,#7201);
#7403=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7302,#7202);
#7404=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7303,#7103);
#7405=FUNCTIONAL_UNIT_TERMINAL_NODE_ASSIGNMENT(#7304,#7203);
#7500=FUNCTIONAL_UNIT_NETWORK_TERMINAL_DEFINITION_NODE_ASSIGNMENT
      (#7300,#1005);
#7501=FUNCTIONAL_UNIT_NETWORK_TERMINAL_DEFINITION_NODE_ASSIGNMENT
      (#7301,#1006);
#7502=FUNCTIONAL_UNIT_NETWORK_TERMINAL_DEFINITION_NODE_ASSIGNMENT
      (#7302,#1007);
#7503=FUNCTIONAL_UNIT_NETWORK_TERMINAL_DEFINITION_NODE_ASSIGNMENT
      (#7303,#1008);
#7504=FUNCTIONAL_UNIT_NETWORK_TERMINAL_DEFINITION_NODE_ASSIGNMENT
      (#7304,#1009);
#7600=AGGREGATE_CONNECTIVITY_REQUIREMENT(#7300,#7000);
#7601=AGGREGATE_CONNECTIVITY_REQUIREMENT(#7301,#7000);
#7602=AGGREGATE_CONNECTIVITY_REQUIREMENT(#7302,#7000);
#7603=AGGREGATE_CONNECTIVITY_REQUIREMENT(#7303,#7000);
#7604=AGGREGATE_CONNECTIVITY_REQUIREMENT(#7304,#7000);
#7700=CONNECTIVITY_ALLOCATION_TO_PHYSICAL_NETWORK(#5679,#7600);
#7701=CONNECTIVITY_ALLOCATION_TO_PHYSICAL_NETWORK(#5879,#7601);
#7702=CONNECTIVITY_ALLOCATION_TO_PHYSICAL_NETWORK(#5206,#7602);
#7703=CONNECTIVITY_ALLOCATION_TO_PHYSICAL_NETWORK(#5776,#7603);
```

```
#7700=CONNECTIVITY_ALLOCATION_TO_PHYSICAL_NETWORK(#5976,#7604);


ENDSEC;
END-ISO-10303-21;
```