Requirements to a physical design support tool for microsystem technology

Andreas Wagener, Jens Popp, Kai Hahn, Rainer Brück
Institute of Microsystem Technology, University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany
E-mail: wagener@rs.uni-siegen.de

KEYWORDS

Microsystems, MST, MEMS, process configuration, process flow, process management, process optimisation, design flow, design verification, databases, EJB, Java WEB Start.

ABSTRACT

This paper describes the requirements and the realisation concepts for a design tool supporting the design stages related to the fabrication process of microsystem technology.

Based on investigations performed by the authors the needs of potential users in the MEMS (Micro-Electro-Mechanical Systems) designer community were specified. Analysing current design flows, methods and tools are presented that support the concurrent and interdependent specification of physical design as well as process sequences. Furthermore an appropriate general software architecture for new approaches in microsystems electronic design automation (EDA) software is presented.

INTRODUCTION

Recent developments and products containing microsystem technology give a clear indication of the importance of this domain. Micromachining provides the interfaces between real world parameters and microelectronic information processing. Using microstructure fabrication methods result generally in smaller products with higher performance at lower costs. Often only MEMS can meet the functional specification of applications. In this context microsystems or at least microstructured components contribute a substantial added value in many innovative products such as medical equipment, home office applications or automobiles. Market analysists expect MEMS to be the key technology for the next decade.

The design task for microstructures can be split into different levels. On the higher levels designers specify the functionality of a complete microsystem or of its components using various kinds of simulation such as FEM (finite elements method). On the lower levels, the physical design levels, the technology related issues become more and more important. Unlike in microelectronics there is a strong dependency between the layout design and the fabrication technology. Many design properties (e.g. the size of structures in the third dimension) can only be realised choosing specific process parameters like materials, process steps or process resources.

EDA in microsystem technologies requires a complete set of new software tools, both for layout and process flow. The process flow is static in microelectronics and can be summarized in the design rules. In microsystem design we need all approaches and tools known from microelectronics. Because the process flow is now application specific it is essential to include the process step configuration to the layout tools.

In the past some prototypes of software dealing with this problem were introduced [Gogoi 1994, Hahn 1999]. It turned out that the requirements of the engineers were not sufficiently met by these tools. For obtaining a practical and usable tool it is inevitable to talk with engineers and analyse their real needs. An intensive investigation phase formed the start of the project. The requirements to a physical design support tool for microsystem technology are summarized in three different domains:

- Supporting adequate workflows based on the MEMS-specific design flows
- Matching the functional requirements by providing tools like process flow editors or specific database structures
- Providing the appropriate software environment for an easy access from different platforms

The paper will describe in detail the requirements as well as the implications with regard to the realisation of the new tools.

DESIGN FLOW

In microsystem technology you can roughly distinguish between three design flows: Top-Down, Bottom-Up, and Meet-in-the-Middle. Each design flow has its own purpose, which will be described below.

The Top-Down approach is used for developing “standard” applications. In the beginning the engineer analyses the requirements of the microsystem. With the aid of these requirements he can build up a behavioural model (schematic) of the system. The model is simulated and crosschecked with the predefined requirements. If it is necessary the model will be refined. This procedure is iterated until the model matches the requirements. The next step is the synthesis of the model. The behavioural model has to be transponed into a 3D-model of the system. In some cases software tools can do this automatically. But mostly it has to be done by hand. In both cases – automatically or manually – a component library is required. This library has to be very comprehensive to get adequate results. Especially the design and structure of the different layers from different materials is very difficult to handle.

After synthesis the 3D-model is simulated with FEM (finite elements method). The results of the simulation helps the

designer finding the critical component structures in the model. Possibly the model has to be improved or the schematic has to be refined.

At last, when the 3D-model is correct, the process steps of the fabrication have to be specified. Normally the majority of steps are determined by the choice of materials in the synthesis. But the combination of the different materials and process steps to produce the layers can cause a lot of problems.

For the development of new processes or process variations for new products engineers use the Bottom-Up approach.

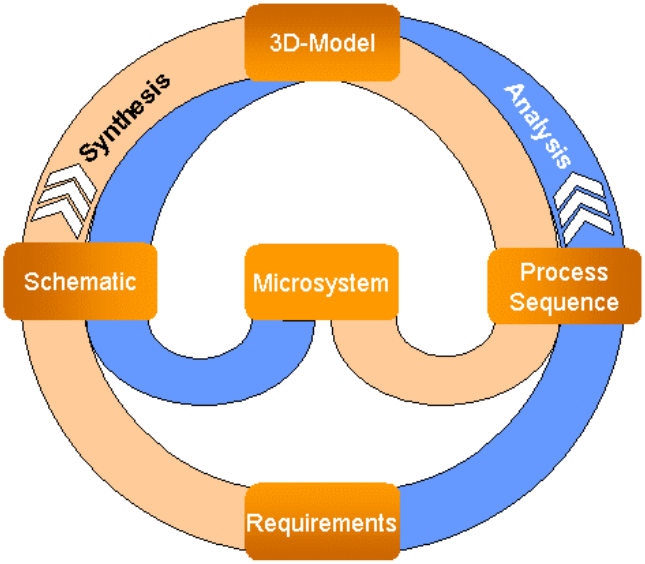

Figure 1: Design Flow Model

This is the standard method for process engineers. Driven by the need for a new process parameter studies are undertaken. Usually existing processes are stepwise varied in one or more parameters. The settings as well as the results are all logged manually. This procedure continues until the desired process characteristics are met. Actually, this heuristic approach is more appropriate for designing processes than complete microsystems. For process steps development, the approach often stops at this point of the design flow. The currently less common version is proceeding and analysing the developed process sequence. Based on the process analysis, the cross sections and the mask information a 3D-model of the target structure is generated, which will then go through a new FEM simulation. Obviously several iterations are necessary to get a satisfying 3D-model that fulfils all requirements. The output of the analysis of the 3D-model is a behavioural model of the microsystem.

Meet-in-the-Middle is a combination of the two other approaches. As described before, the purpose of developing new processes can be seen as a kind of service for supporting the microsystem designer with a large library of processes. Especially in research projects not only the system and its behaviour will be developed anew. Because of the usage of new materials or new combinations of materials the parallel development of the system and the fabrication processes is essential. On one hand the behaviour of the system is specified on the other hand suitable processes and process sequences are developed. Combining the specification and the results of the process design leads to the 3D-model required for verification.

Our work focuses on a support tool for the Bottom-Up approach, because there is the biggest need for effective support.

CAD WORKFLOW

The workflow for developing microsystems is - as described before - a highly iterative process. Especially the lack of ready-to-build and standard components in microsystem technology forces the engineers and developers using trial-and-error-methods, as described in detail in the cycle model [Hahn 1999]. To decrease the number of cycles and thus increase the time to market the design flow must be optimised.

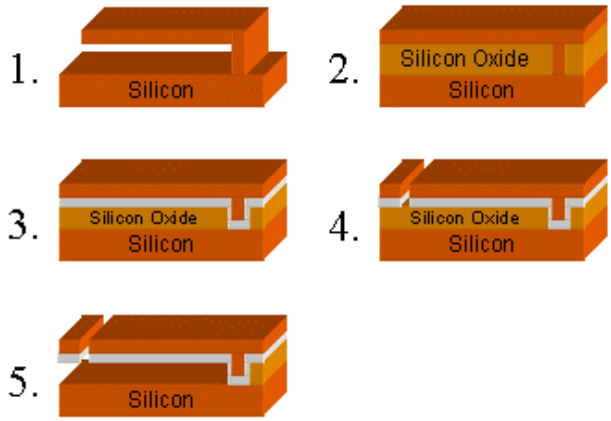

Figure 2: Workflow Example

As example a supposable workflow with a supporting software is shown in figure 2. The engineer has a vague idea of the system derived from the requirements. He sketches a draft of the system to elaborate the process sequence. In figure 2.1 there is an example of such a draft: A silicon cantilever on a silicon substrate. The software recognizes the airgap under the cantilever and suggests a sacrificial layer; in this case silicon oxide (SiOx) is used. The problem of the airgap is solved but now there is a stiction problem, since silicon does not adhere on SiOx very well. The software recommends a stiction layer. Now the layer sequence in figure 2.3 is producible. At last the sacrificial layer has to be removed. This can be achieved by etching a gap (figure 2.4) and isotropical etching of the SiOx.

SIMULATION

Having a look at the workflow described before three views on the system can be seen. The layer view shows the structure and cross-section (x-y-coordinates) of the system and the arrangement of material layers. The process flow view shows the sequence of process steps to fabricate these layers, and the mask view provides the lateral view (x-z-coordinates) to the layer and the process steps.

These three views together describe the microsystem sufficiently and must be managed properly. Providing three different editors this can be achieved. Those editors can be combined to get to a set of consistent and linked representations of the system. Every modification in one view automatically reflects a change to the other views.

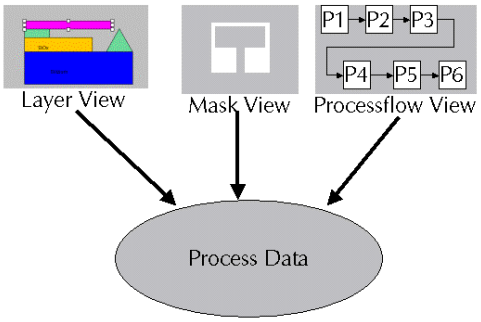

Figure 3: Model -View

Such requirements can be fulfilled by maintaining the view independent of the design data. Each view uses the same data pool (Figure 3). Specific algorithms simulate the behaviour of each process step and map it to the layer view. Vice-versa a change in the layer view effects changes in one or more process steps. The algorithms utilise a knowledge base, which is mandatory and can be realised by a database as described below. The editors in combination with the database must support versioning due to the highly iterative development process. The benefit of versioning is not only the chance to undo changes but also the possibility of reusing the intermediate steps for other projects.

Each change of the flow has an impact on the consistency of the process flow. While simulating the process sequence the consistency can be checked. Also the profile of the cross section changes (Figure 2). To assure that the processes and thus the microsystem complies with the requirements the profile must be simulated, too. Using the knowledge and rules stored in databases the consistency and the profile can be checked and the user gets feedback about the changes. The feedback looks like the feedback described in the CAD workflow example.

Indeed the quality of simulation depends on the available process information. The more information about processes and interactions is stored the better the forecast of the real profile becomes. With the increasing knowlegdge base also the possibility to have several sequence solutions for one microsystem becomes more likely. This can be an advantage in this respect of optimising the process flow. Setting preferences the sequence can be optimised by cost, production time, or specific material parameters like stress or thickness.

SOFTWARE ARCHITECTURE

A key issue for each software project is the system architecture and the target platform. Both issues influence the decision on software tools and programming language.

In the case of MEMS design more often than not many people have to work together to create a good product. For instance there are the engineers who are responsible for the machines, the mask designer and the process designer. All of them have to have access to a common base of data and tools. There is also a wide variety of hard- and software platforms in use – various Windows and Unix Systems. To make it even more complicated the same user may use different systems at different locations depending on the work he has to do.

Another important point is the security of data. Some companies outsource their production to foundries due to the enormous costs linked with manufacturing MEMS. On the one hand the foundries want to minimise costs and have to find partners on the other hand they do not want to share their knowledge about the processes. Keeping the knowledge in a database within the foundry would solve this problem. Via distributed software clients could access check algorithms that utilise the data. So it would be possible for costumers to check their design without knowing too much about the processes.

Taking all these points into account a distributed client-server-architecture is necessary. To minimize the porting overhead for the different platforms the programming language of choise is Java.

Looking at client-server programming among the different concepts in use the three- or multi-tier architectures became prominent in the last few years. Basically such systems are divided into a database, an application server and a front-end or client software. The database is the common pool of data for all applications. Normally a relational or (lately) object-relational database is used. Due to the wide standardisation efforts in this area the database management system is exchangeable. Purely object-oriented databases lack this standardisation and were therefore not chosen for the project. The second layer is the application server. The server is responsible for the retrieval and processing of the data. All (or most) of the application logic should be located here. The final layer is the front-end or client application. The task of this part is the presentation of the processed data to the user. Only a small part of logic should be situated in the client application. It is possible to split up the tiers (for instant multiple layers for application logic). The following paragraphs will describe each part.

As mentioned above it is necessary to store a large amount of data during the process development because fewest interactions between parameters in MEMS processing can be described with an adequate formalism. To achieve acceptable time in storing and retrieving this data a professional database management system is mandatory. While every database vendor has its speciality, the common base of how to store and retrieve data and how to map a database model is standardised for relational databases. This enables the interchange ability of databases in multi-tier architecture. The object oriented databases still lack the standardisation. Once they reach the maturity of relational databases they will become a real alternative for multi-tier architectures due to the fact that most data to be stored are modelled as objects in an object-orientated language.

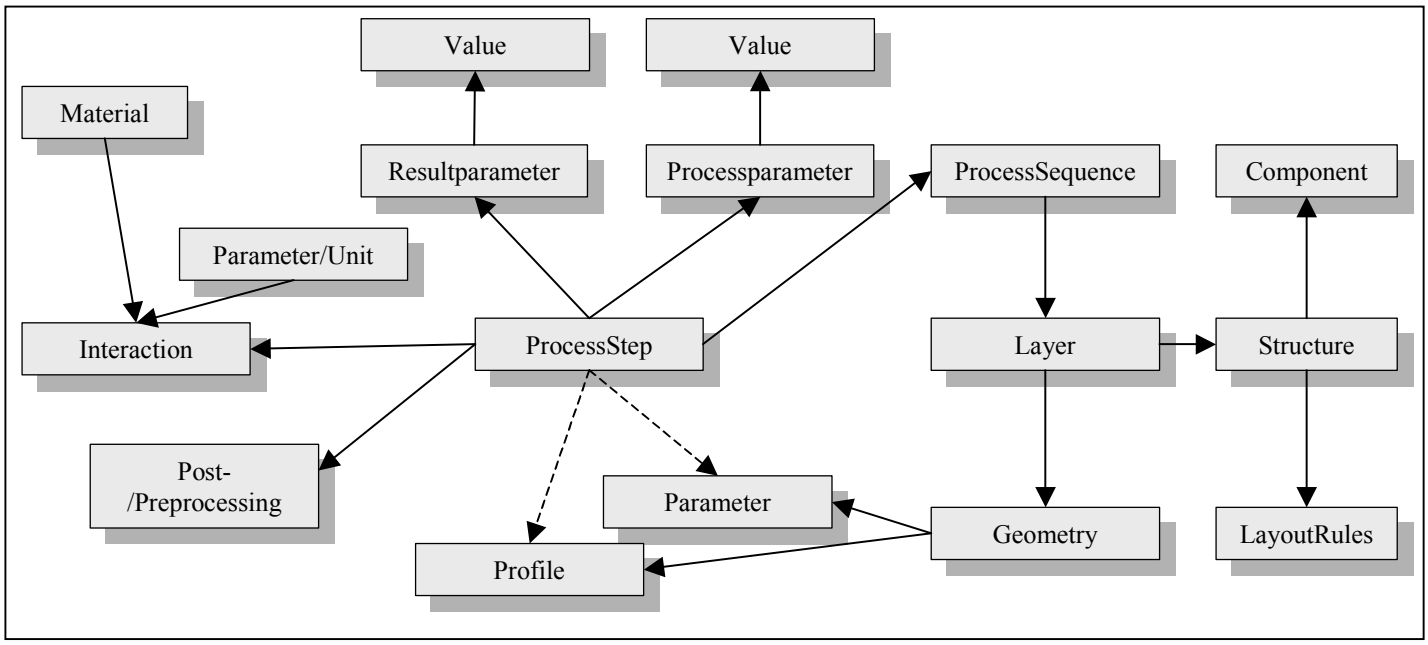

Figure 4: ER-diagram

To map data from real world into databases the so-called ER-Model (Entity-Relationship) is used. It describes entities as representation of real world objects and the relationship between those. An extract of the ER-diagram used to represent the problem domain can be seen in Figure 4. The whole model has more then 50 tables.

The second layer in the three-tier architecture is the application server. The main field of usage of application servers today is e-commerce. This is due to the fact that application servers deliver a high performance environment for business logic. Basically an application server is a container and management system for software modules for instance Enterprise Java Beans (EJB). The database connection, the lifecycle of the software components and the communication with other application servers or client software are managed by the server.

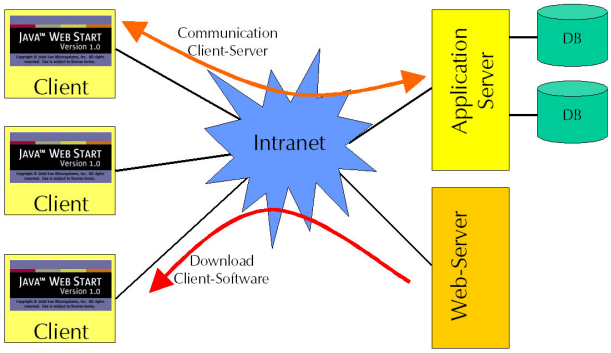

Figure 5: Distribution

The basic idea behind the use of application server with EJBs is to provide an infrastructure that is scalable, transactional and multi-user secure. With growing demands new servers can be added. Software written using this technology is written once and can be deployed on any server (at least if they comply with the specification). Standard components (e.g. equation solvers) can thus be reused in different application areas without the need for rewriting it.

The front end or client software is necessary to communicate with the user. In common software is not static but will be continuously improved. Updates and upgrades change the source code of an application. Since the software will be used in enterprises it is obviously that a lot of installations of the client can be found. All these installations have to be kept up-to-date. There are different possibilities to achieve this goal. A new kind of technology provided by Sun: Java Web Start [Sun 2002] was chosen. Web Start combines the advantages of an application and a browser applet but do not adopt the disadvantages of them. The distribution of the client-software is managed by a web-server (Figure 5). The user opens a web-page and downloads a XML-file with detailed information about the software. This XML-file will be checked and interpreted automatically by the Web Start client, which is included in the newest Java Development Kit (JDK). Now Web Start knows the location where the Software can be found. The client will be downloaded. Here is another advantage of Web Start. The user needs not to install the software manually. The code is cached inside the Web Start environment but appears like a stand-alone application. Each time the software is started a lookup is made. This lookup checks the version of the software against the version in the web. If the software in the web is newer than the installed it will be downloaded automatically.

USER INTERFACE

Another problem you have to deal with on client side is the user interface. To achieve acceptance among the target user an intensive user needs analysis was made. In the first step the storage of process data was the main field of interest.

Since the specification of new process flows is often insufficiently supported by CAD tools, standard office tools like word processing, spreadsheets, drawing software and presentation tools are in use. These tools are stand-alone solutions and not adequate to archive complex process descriptions. Most important they provide no common interface to database systems. In this sense the tools are less proper for knowledge gathering and for automatic knowledge exchange with tool support. On the other hand this software is easy to use and the users are familiar with it.

To make a transition to new software as easy as possible the user interface has to have a familiar look and feel. Representation of data in tables and some of the function of spreadsheet software (like presenting data in diagrams) will allow the user to adapt easily to the new environment.

DATA EXCHANGE

The aim of the physical design tool presented is to compose process flows and to extract valuable information (stress, temperature budget, layout rules) for further use in CAD or FEM tools. To enable this a standard exchange format is needed.

Like Java as a platform independent programming language, XML (Extensible Markup Language) [W3C 2002] is an application independent language for describing structured data. XML is a flexible and powerful markup language. It provides a simple but standard way to describe and delimit data. There are many tools available to parse, check and create XML files. One of the reasons why XML is so popular is surely its flexibility. XML serves as a meta language and provides mechanisms to build up powerful descriptions. XML's great advantage is its self-describing nature. The information is grouped and bordered by tags. Also people who were not familiar with XML before can read the listings very easily.

As a subset of XML we propose a process description markup language PDML for microsystems [Wagener 2002]. Like XML PDML is actually object-oriented but as mentioned above approaches to use an object-oriented database were discarded for different reasons. A first version of PDML has been presented in [Kleinert 2002, Wagener et. al. 2002].

CONCLUSION

Using the described design flow, CAD workflow and software architecture a new kind of tool can be created. This software complements tools for system analysis and layout of a MEMS product. Supplying the existing CAD tools with a kind of dynamic process libraries can improve the overall design process and decrease the time to market. The proposed methods of software engineering make the software scalable to user needs and easier to maintain. With the new standard exchange format for process design the interoperability is ensured. Data extracted from the tool can be used for layout check, FEM and other simulation tools.

REFERENCES

Brück, R and K. Hahn. 1997. "An Approach to Layout and Process Verification for Microsystem Physical Design. Microsystem Technologies". *In Proceedings of the 1997 Microsystem Technologies conference*. 82-90.

Brück, R. and K. Hahn. 1998. "MEMS Process/Design and Optimization using Economical Constraints". *In Proceedings of the 1997 Microsystem Technologies conference*, 479-484.

Brück, R. and C. Schumer. 1999. "Internet MEMS Design Tools based on component technology". *In Proceedings of the 1999 Design, Test and Microfabrication of MEMS/MOEMS conference*.

Gogoi, B.; R. Yuen, and C. H. Mastrangelo. 1994. "The automatic synthesis of planar fabrication process flows for surface micromachined devices". *In Proceedings of the 1994 IEEE Micro Electromechanical Systems Conference, Oiso, Japan.*

Hahn, K. 1998. "Die Prozessbeschreibungssprache LIDO-PDL, Handbuch zur Sprachbeschreibung". Research Reports, University of Dortmund.

Hahn, K. 1999. "Methoden und Werkzeuge zur fertigungsnahen Entwurfsverifikation in der Mikrotechnik". Ph. D. Thesis. Department 12, University of Siegen.

Kleinert, A. 2002. "Entwicklung einer Prozessbeschreibungssprache für die Mikrosystemtechnik auf Basis von XML". Diploma Thesis. Department 12, University of Siegen.

Schneider, C.; R. Brück, K. Hahn, et. al. 1999. "Component Based Distributed Design Tools for Microsystem Technology". *In Proceedings of Sensor 1999. Nuremberg, Germany.*

Wagener, A. 2001. "System- und Anforderungsanalyse für ein Prozessdesign-Werkzeug in der Mikrosystemtechnik". Diploma Thesis. Department 12, University of Siegen.

Wagener, A.; J. Popp, K. Hahn et. al. 2002. "PDML – A XML-Based Process Description Language". *In Proceedings of the 2002 European Concurrent Engineering Conference (ECEC), Modena, Italy.*

World Wide Web Consortium (W3C). 2002. "Extensible Markup Language (XML)". *http://www.w3c.org/XML/. (August 2002)*

Sun Microsystems Inc. 2002. "Java Web Start Homepage". *http://java.sun.com/products/javawebstart/. (September 2002)*