Multiple Architecture Modeling Design Method for Mixed Signal and Multi Domain System Simulation - First Solutions

Michael Schlegel, Göran Herrmann, Dietmar Müller

Faculty of Electrical Engineering and Information Technology,
Chemnitz University of Technology,
Reichenhainer Str. 70, D-09126 Chemnitz, Germany

e-mail: michael.schlegel@infotech.tu-chemnitz.de

ABSTRACT

During the design process of MEMS digital, analogue electrical and non-electrical models at different abstraction levels may appear. The abstraction levels of the interfaces depend on the abstraction levels of the models. If system models are developed within the scope of a system design it might have been necessary to modify the interfaces of the system and component models at every design step. The aim of the work presented in this paper is to present a new methodical approach which makes it possible to keep the interface of a component unchanged if the abstraction level of the component has changed. This paper is an abridged version of a poster, which gives an overview of problems which occur when analogue and digital interfaces at different abstraction levels are connected. The poster describes a practicable solution for these problems and its realization in VHDL-AMS.

KEYWORDS

design methodology, MEMS design, top down design, unified interface, modeling

1 INTRODUCTION

System design using a top down strategy normally starts with a system model at high level of abstraction. Abstraction of the components of the system model decrease during the design process. The hardware description language VHDL allows to describe more than one architecture for one interface (entity) when designing digital systems. So it is possible to handle architectures with different abstraction levels for one component. Now this feature is also available for the design of heterogeneous systems by means of the extension of VHDL to VHDL-AMS (VHDL Analogue and Mixed Signal). The design of different architectures for one component has become common usage since the introduction of VHDL. But if this feature of VHDL should be used to describe components at different abstraction levels then the problem may occur that the models at different abstraction levels use different interfaces.

In the following sections a new methodical approach called Multi Architecture Modeling (MAM) is presented. If this new

approach is used when designing the first abstract models in a top down design process then these models can be replaced later by refined models without any modification to their interfaces or the interfaces of the system model. As a constraint to this new methodical approach, the modeling overhead should be as little as possible in comparison with the customary modeling.

2 PROBLEMS AND THEIR SOLUTIONS

2.1 Digital interfaces

Digital interfaces usually use `SIGNALS`. The abstraction level has an influence on the used datatype. E. g. the datatype integer is used on functional or algorithmic level and the datatype `bit_vector` or a vector of multi value logic is used on register transfer level. A conversion between these types is still possible simply by a datatype conversion function. So two approaches are conceivable – an interface of datatype `integer` or `bit_vector`. The approach using datatype `integer` may cause the following properties:

- small overhead
- additional delta cycles
- If the interface contains vectors of multi value logic then a loss of information of the states X, Z etc. can not be avoided.

The better approach is to use the datatype bit_vector or a vector of multi value logic. In this case the following properties will result:

- small overhead
- additional delta cycles
- If the interface contains vectors of multi value logic then a loss of information of the states X, Z etc. exists only as long as models with higher abstraction level are in the system model. This loss of information may be avoided by additional `SIGNALS` handling these states inside the component model, but then a larger overhead will appear.

2.2 Analogue electrical and non-electrical interfaces

Abstract analogue components are often described as functional blocks. The interface objects may be `QUANTITIES`. They represent time and value continuous information. In opposition to this, detailed analogue models are described as conservative nets. In this case the interface is a `TERMINAL`. The `TERMINALS` consist of an `ACROSS` value (e. g. voltage) and a `THROUGH` value (e. g. current) and represent a conservative node which meet Kirchhoffs' law. `TERMINALS` can not be connected directly to `QUANTITIES`. So until now the system model has to be modified when the interface has changed. This takes a lot of time and may cause errors in the model. So either one `TERMINAL` or two `QUANTITIES` (one for the `ACROSS` and one for the `THROUGH` value) are necessary for a common interface object.

Approach 1: Usage of two `QUANTITIES` as interface

Component at high abstraction level:

Approach with MAM

```
PORT(QUANTITY q1i,q1u:IN real;
     QUANTITY q2i,q2u:OUT real);

q2u==F(q1u,q1u'dot,t);
q2i==0.0;
```

conventional approach

```
PORT(QUANTITY q1:IN real;
     QUANTITY q2:OUT real);

q2==F(q1,q1'dot,t);
```

Component at low abstraction level:

Approach with MAM

```
PORT(QUANTITY q1i,q1u:IN real;
     QUANTITY q2i,q2u:OUT real);

q1i + q2i==0.0;
q1u - q2u==q1i * R;
```

conventional approach

```
PORT(TERMINAL t1, t2:
              electrical);

QUANTITY u ACROSS i
         THROUGH t1 to t2;

i==u / R;
```

Properties of this approach:

- In this approach abstract models using MAM need a second simultaneous statement (equation) which is technically necessary but functionally redundant.

- In the model on low abstraction level Kirchhoffs' law must be modeled explicitly by the designer. This is complicated and may produce errors.
- Instability of simulation may be caused.

Approach 2: Usage of one TERMINAL

Component at high abstraction level:

Approach with MAM

```
PORT(TERMINAL t1,t2:
             electrical);

QUANTITY q1 ACROSS  t1;
QUANTITY q2 ACROSS
        q3 THROUGH t2;

q2==F(q1,q1'dot,t);
```

conventional approach

```
PORT(QUANTITY q1:IN real;
     QUANTITY q2:OUT real);

q2==F(q1,q1'dot,t);
```

Component at low abstraction level:

Approach with MAM

```
PORT(TERMINAL t1,t2:
             electrical);

QUANTITY u ACROSS
        i THROUGH t1 to t2;

i==u / R;
```

conventional approach

```
PORT(TERMINAL t1,t2:
                electrical);

QUANTITY u ACROSS
       i THROUGH t1 to t2;

i==u / R;
```

Properties of this approach:

- small overhead
- easy to use

2.3 Digital/analogue interfaces

Even in pure digital systems it may happen that the behavior of components of the system has to be modeled with analogue elements or equations because their exact timing is relevant. The digital components on high abstraction level usually use SIGNALS, detailed models with analogue behavior use TERMINALS. SIGNALS handle time discrete information whereas TERMINALS handle time continuos information. So analogue to digital (A/D) and digital to analogue (D/A) converters are necessary. These converters have to be simple to model and they must not have side effects on the behavior of the model itself. Another question is, which kind of interface object has to be used. As it is shown in section 2.2, the use of terminals is unavoidable in order to describe the analogue behavior correctly. On the other hand, SIGNALS carry the digital information including the events controlling the digital simulator. Therefore it is necessary to transmit these events also by the interface. A solution would be to work with double interfaces using a SIGNAL and a TERMINAL for one port. So every component has to provide the output information as TERMINAL and SIGNAL. But this procedure is very complicated and may cause problems at bi-directional ports.

Therefore the interface should be only one TERMINAL. But it must be considered to provide the digital information on the TERMINAL in a way the following digital component can restore the events from the TERMINAL correctly, and also an analogue component has to handle the same information correctly.

The output must be modeled as a simplified real output driver to guarantee that an analogue input can handle the values correctly. For a SIGNAL of datatype real the output driver may be modeled as follows:

```
ENTITY outdrv IS
  PORT(SIGNAL s_in:IN real;
       TERMINAL t1:electrical);
END;

ARCHITECTURE behav OF outdrv IS
  TERMINAL t2:electrical;
  QUANTITY u_r ACROSS i_r THROUGH t2 TO t1;
  QUANTITY u_out ACROSS i_out THROUGH t2;
BEGIN

  u_out==s_in;
  u_r==i_r*R_out;
  BREAK ON s_in;

END;
```

The parameter R_out (output resistance) may either be set to technology dependent or independent values.

For the input of the receiving model at high abstraction level the following A/D converter may be used:

```
ENTITY input IS
  PORT (TERMINAL in1:electrical);
END ENTITY;

ARCHITECTURE behav OF input IS
  QUANTITY u_in ACROSS in1;
  SIGNAL s_in_r:real:=0.0;
  CONSTANT delta: real:=0.1;
  SIGNAL x1,x2: boolean;
BEGIN

  x1<=u_in'ABOVE(s_in_r+delta);
  x2<=u_in'ABOVE(s_in_r-delta);
  s_in_r<=u_in WHEN x1 OR (NOT x2);

END ARCHITECTURE;
```

It must be ensured to set `hmin` (parameter for the smallest time step allowed of the analog solver) less than `delta/u_in'dot`, where `u_in'dot` is the largest value of the derivation over time of the `QUANTITY` `u_in`. This A/D converter requires a correct implementation of the `ABOVE`-Statement in the simulator. If this is not the case then it might be possible to work with an value `hmax` (parameter for the largest allowed time step of the analog solver) smaller than `delta/u_in'dot`. The constant `delta` may be used to modify the accuracy of the converter.

A correct recognition of events is possible in most cases on interfaces using datatypes `bit`, `boolean` or vectors of these types. The recognition of events on `SIGNALS` of type `integer` or `real` is not possible in any case. Especially when using floating point types, events may get lost or additional events may appear. But these disadvantages should not cause any problems when working with synchronous designs using a clock signal.

If the input logic is working with a clock signal then the `ABOVE` statement may not be needed by using the following process:

```
PROCESS(clk)
BEGIN
  IF clk'EVENT and clk'LAST_VALUE='0' THEN
                          -- L-H Slope on clk
    s_in_r<=u_in;
  END IF;
END PROCESS;
```

Components on low abstraction level (modeled as analogue behavior) may work without an additional driver or receiver.

3. CONCLUSION AND OUTLOOK

If a component is using the interface which is necessary on low abstraction level already at the highest level in a "top down" design then the insertion of components on different abstraction levels into the system model can be done easily (see second approach in section 2.1 and 2.2). This is also true when the design by a "bottom up" method starts on low abstraction level and the later developed models for system simulation are using the same interfaces as the detailed models.

The replacement of digitally modeled components by analogue modeled components may also be done easily when `TERMINALS` are used as interface objects. The necessary drivers and receivers may be stored in a library so they can be accessed without the need to rewrite them. So the overhead in modeling is limited. It is still not possible to resolve multiple drivers at a bus by a resolution function. This feature will be realized soon for signals of type `bit`, `boolean` and `std_ulogic`.

The main disadvantage of this new approach is that the interfaces of all components must be known in every detail even at the beginning of the design process when doing a design by a "top down" method. But also in this case it can be recommended to use this method as far as the interfaces are known, because it is easier to add or modify a few ports than the whole interface. For large, pure digital designs e. g. [5] or designs with automatically synthesized interfaces e. g. [6] this method cannot be recommended. Possibilities to remove this problem will be explored in the next steps of evaluation of this approach. If the interfaces

are known then the MAM approach may help to reduce errors in system design and it seems to be a powerful approach for improving the cooperation between component and system designers when developing MEMS.

The overhead for component modeling on high abstraction level is limited to a conversion of the interface inside the model. This conversion is done in digital components by a datatype conversion function, in analogue components by an additional definition of a branch `QUANTITY` and in analogue-digital mixed interfaces by a special driver and receiver for the digitally modeled component. So there is only a small modeling overhead in the design process of abstract models. The MAM does not cause any significant overhead in the models at low abstraction level. The methodical approach of the MAM is in conformity with the Language Reference Manual of VHDL-AMS [1] which could be proved by first simple examples.

This new methodical approach to design components of heterogeneous systems on different abstraction levels with uniform interfaces must be evaluated by future designs. Therefore the design of a vibration sensor system using a sensor array – in the scope of the SFB379 (collaborative research center) – will be done using the MAM method. The focus will here be set on the practical feasibility of this method.

ACKNOWLEDGMENTS

The work presented here has been done in project A2 "System Design" of the SFB 379, which is funded by the German Science Foundation (Deutsche Forschungsgemeinschaft DFG).

REFERENCES

[1] The Institute of Electrical and Electronics Engineers, Inc.: *IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes)*. IEEE Std 1076.1, 1999. - ISBN 0-7381-1640-8

[2] Rosenberger, R.; Huss, S. A.: *A Systems Theoretic Approach to Behavioral Modeling and Simulation of Analog Functional Blocks.* IEEE/ACM Design, Automation and Test Conference Europe, Paris, 1998

[3] Hanna, J. P.; Hillman, R. G.: *A Common Basis for Mixed-Technology Micro-System Modeling.* International Conference on Modeling and Simulation of Microsystems MSM 99, San Juan, Puerto Rico, U.S.A., Apr. 19-21, 1999, pp. 679-682, ISBN 0-9666135-4-6

[4] Schlegel, M.; Müller, D.: *Gesamtsystemsimulation mit VHDL-AMS am Beispiel der Rundumprojektion mit 2D- Mikrospiegelarray*. Scientific Reports, Journal of the University of Applied Science Mittweida, 14th International Scientific Conference Mittweida, Nr. 10, 2000, Mittweida, Germany, 08.-11. November 2000, ISSN 1437- 7624, pp. 87-94

[5] Siegmund R.; Kretzschmar, C.; Müller, D.: *Adaptive Partial Businvert Coding for Power-Efficient Data Transfer over Wide System Busses.* XIII International Symposium on Integrated Systems and Circuit Design SBCCI 2000, Manaus (Brazil), September 2000, 18-23, 2000, ISBN 0-7695-0843-X

[6] Siegmund, R.; Müller, D.: *Synthesis of System-on-Chip Communication Architectures from Interface-based System Specifications.* ITG Workshop Mikroelektronik für die Informationstechnik, 20.-21. November 2000, Darmstadt, in: ITG-Fachbericht Nr. 162, ISBN 3-8007-2586-X