Implementing an Automatic Functional Test Pattern Generation for Mixed-Signal Boards in a Maintenance Context

Bertrand Gilles, Laurent Tchamnda Nana, Valérie-Anne Nicolas

Implementing an Automatic Functional Test Pattern Generation for Mixed-Signal Boards in a Maintenance Context

Bertrand GILLES Laurent NANA Valérie-Anne NICOLAS
EA3883 – LISyC – Département Informatique – Université de Bretagne Occidentale
20 avenue Le Gorgeu – CS 93837 – 29238 Brest Cedex 3 France
{gilles|nana|vnicolas}@univ-brest.fr

Abstract

In this paper, we present in the context of mixed-signal board maintenance testing, language and implementation aspects of an automatic functional test pattern generation approach. The goal is to help maintenance test engineers. Our modeling method for mixed-signal boards and their components is presented and the languages proposed for modeling are described. The implementation of our test data generation process based on constraint logic programming is discussed. The application to a simple board is shown and the tool Copernicia developed in the context of our work is presented.

Keywords: Maintenance testing, modeling, mixed-signal boards, automatic test pattern generation.

1. Introduction

Various test methods and techniques have been developed for circuit test [1] at the different stages of product life-cycle, mainly at design and production levels. Nevertheless, few interest has been thrown into maintenance testing which has its own specificity. In particular, functional testing is needed in order to check behaviors. In the case of mixed-signal boards, maintenance testing currently relies mainly on the expertise of test engineers. Methods and tools for automating the full or part of the testing process are needed to help them in their task.

Our work aims at providing such solutions. In this paper, we focus on language and implementation aspects in the approach we proposed in [2] for the modeling of mixed-signal boards and the automatic generation of test data for such boards.

We first present our modeling approach and focus on the modeling languages and test data generation. The application to a simple board is then shown and the tool we are currently developing is described. The paper ends by conclusions and future works.

2. Modeling and testing approach

The automation of the testing data generation process of mixed-signal boards requires a structured modeling approach. We consider two hierarchical levels of modeling: *Board level* and *block level* [2]. Both are used for our automatic test pattern generation (ATPG). The board is broken down into a set of interconnected blocks. Each block has an associated *functional model* which describes its behavior and a *test model* which specifies how the block can be efficiently tested.

The test of the board is achieved by testing each block individually using its associated test model. Test patterns for a block (BTP) are then generated by carrying out the transition coverage of its test model. Since the block under test is often embedded within the board, without any test access mechanism, the functional models of adjacent blocks are used for forward propagation to primary outputs (PO) and backward propagation to primary inputs (PI). During these propagations, BTP have to satisfy the constraints associated to the adjacent functional models in order to compute the final test patterns. Finally, a board test data set is the union of test patterns for all the blocks of the board.

3. Modeling languages

We propose three modeling languages: a *board description language* associated to the board level, a *block description language* and a *transition language* associated to the block level. We describe these languages in the next subsections.

3.1. Board description language

As mentioned in section 2, the board description language describes the board as a set of interconnected blocks, as depicted in Figure 1. In addition to the building blocks of the board, some external blocks are needed to model connections between the PI/PO of the board and an automatic test equipment (ATE): external

sources which supply input signals (blocks S1 and S2) and output measurement points (blocks m1 and m2).

Blocks are analog, digital or mixed-signal, and may have several inputs and outputs. Oriented links denote data exchanges (signals) between components.

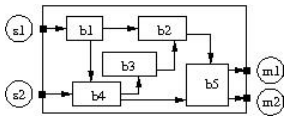

Figure 1. A board description

3.2. Block description language

The block description language is based on communicating finite state machines (CFSM). CFSM are often used for modeling systems which imply communicating processes and is well suited for our needs. Another advantage of using it is that it is well known by test engineers. Two CFSM interact when one CFSM produces an output that is placed in the input queue of the other.

Each block is described by one or more CFSM, depending on the complexity of the functionality of the block. As a consequence, the set of blocks of a board is represented by a set of CFSM.

Since communications are involved, CFSM transitions are decorated with labels. A label expresses synchronization conditions with blocking receptions of signals and/or non blocking sending of signals between two or more CFSM. Constraints on signals involved into communications are also expressed. Each label is written with the transition language presented in the next subsection.

3.3. Transition language

In this section, we present the grammar of the transition language. The main improvement from previous work is that we can express more complex analog signals and constraints. The grammar is described with production rules. A rule has the following form: A ::= w, where A and w represent respectively the left and the right hand sides of the rule. The rules are given below:

Transition : := Recv $\rightarrow$ *Constraints Send*
| *Recv* $\rightarrow$ *Send*
| *Constraints Send* | *Recv* | *Send*

Recv : := Recv;RecvFromAutomaton
| *RecvFromAutomaton*

RecvFromAutomaton : := Identifier ? SignalIds

SignalIds : := SignalIds , Identifier | *Identifier*

Constraints : := Constraints && Constraint
| *Constraint*

Constraint : := Arithmetic relOp Arithmetic

RelOp ::= <=|>=|<|>|==|!=

Arithmetic : := Arithmetic + Term
| *Arithmetic - Term*
| *Term*

*Term ::= Term * Factor*
| *Term / Factor*
| *Factor*

Factor ::= Real
| *(Arithmetic)* | *Identifier(ParamList)*
| *Identifier* | *SignalParam*

ParamList ::= ParamList, Arithmetic | Arithmetic

SignalParam : := Identifier . Attribute

Send : := [SendToAutomata] | ε

SendToAutomata ::= SendToAutomata ;SendToAutomaton
| *SendToAutomaton*

SendToAutomaton : := Constraints : Identifier ! SignalFunctors | Identifier ! SignalFunctors

SignalFunctors : := SignalFunctors , SignalFunctor
| *SignalFunctor*

SignalFunctor : := sig (Identifier, Arg, Arg, Arg)

Arg : := Identifier | Real

The possible values of the attribute appearing into the "signalParam" definition depend on the type of signal. For example, the attributes *max*, *freq* and *phase* are defined for sine signals and correspond respectively to the magnitude, the frequency and the phase of such signals.

4. Implementation of the test data generation

We have chosen constraint logic programming (CLP) for the test data generation process. CLP is very relevant for test data generation since it makes it possible to represent test data in a symbolic way, using ranges of values. These ranges of values deal efficiently with analog and digital data representations in a uniform way. Ranges of test data are computed for reaching the test requirements. Final test data are instantiated by the user.

The set of CFSM describing the board behavior is translated into Prolog predicates. The labels of transitions expressed in the transition language identify a set of constraints which are associated to the

behavior of a block. These constraints are translated and solved using CLP, leading to the test patterns.

On the implementation point of view, communication between CFSM Prolog models is achieved using dynamic predicates. Their use makes it possible to implement the synchronization conditions mentioned in section 3.2 (blocking receptions).

In the next section, we present an application of our modeling and testing approach to a simple case study.

5. Application to an example of board test

For the sake of simplicity, we consider a board which is a simple first order high-pass analog filter. Figure 2 shows this board modeled by the board description language where block *S*, *MP and F* represent respectively an analog source, an analog measurement point and the considered filter.

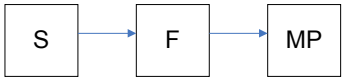

Figure 2. The board level description

5.1. Input/output modeling

Figure 3 shows the functional models with internal label representation of the source and the measurement point respectively expressed with the block description language. For each CFSM, the transitions are numbered from 1 to n, n being the total number of transitions. For a given automaton, each transition is internally represented as follows: *CFsmName::T#k {L}* where *k* is the kth transition of CFSM *CFsmName* and *L* is the label expressed with the transition language. By convention, CFSM name is the same as block name.

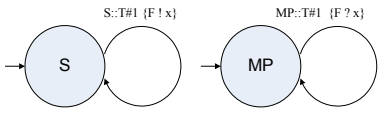

Figure 3. The source and measurement point functional models

S::T#1{ F ! x } stands for a sending of signal *x* to CFSM *F* and *MP::T#1{F ? x }* stands for the reception of signal *x* from CFSM *F*. The number of the transition is 1 in both cases since each CFSM only has one transition.

5.2. Filter modeling

As we explained in section 2, the functional model of the filter describes its behavior. It is depicted in Figure 4. Thus, *F::T#1{L1} where L1* is

S ? x -> x.type == sine

expresses that when a sine signal is received from CFSM S, the corresponding transition is crossed. Then the output signal is sent to CFSM MP thanks to *F::T#2{L2}* where L2 is

[V == (x.max / sqrt(1 + square (Fc)/square(F0))) && F0 == x.freq && Phi0 == (x.phase + arctan(Fc/F0)) : MP ! sig(sine,V,F0,Phi0)]

Fc is the filter cutoff frequency. The magnitude, the frequency and the phase of the output sine signal are represented respectively by *V*, *F0*, and *Phi0*. The first part of this label (delimited by the ':' symbol) expresses the constraints applied to the output sine signal according to the input signal characteristics. In the second part, the functor *sig* represents the output signal of the filter sent to CFSM MP. As we can see, this functional model describes the transfer function of the filter (attenuation and phase shift).

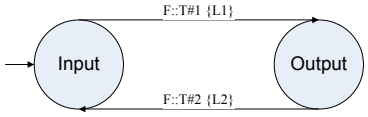

Figure 4. The filter functional model

In the other hand, the test model of the filter describes the way it can be efficiently tested (Figure 5). The model integrates test engineers skills. Thus, *F::T#1 {L1}* and *F::T#2 {L2}* where *L1* is

*S ? x -> x.type == sine && x.max > 2.0 && x.max < 10.0 && x.freq == (10.0*Fc) && x.phase == 0.0*

and *L2* is

[F0 == x.freq && V <= (x.max + d1) && V >= (x.max - d1) && Phi0 <= d2 && Phi0 >= - d2 : MP ! sig(sine,V,F0,Phi0)]

describe the filter response in the bandwidth.

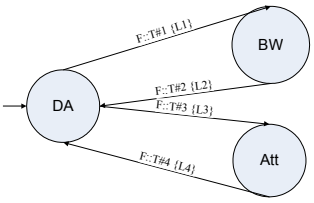

Figure 5. The filter test model

5.3. Board test data generation

The following code shows the translation of the transition F::T#1{L1} of the filter test model into Prolog predicates, according to the approach described in section 4:

```
reception_constraints(1,1,(X_type,X_max,X_frq,X_phi)) :-
cutoff_frequency(Fc), X_type $= 1, X_max $> 2.0, X_max $< 10.0,
X_frq $= 10.0 * Fc, X_phi $= 0.0.
transition(1,1,1,2,[(X_type,X_max,X_frq,X_phi)|L],L,[action(bandw
ith)],Comm,La) :-
channel(2,1,(X_type,X_max,X_frq,X_phi)), !,
Comm is 1,    La = [],
retract(channel(2,1,(X_type,X_max,X_frq,X_phi)) :-
carac_signal(2,(X_type,X_max,X_frq,X_phi))), !,
reception _constraints(1,1,(X_type,X_max,X_frq,X_phi)).
transition(1,1,1,2,[(_,_,_,_)|L],L,[action(bandwith)],Comm,La) :-
Comm is 0, La = [2].
```

The predicate *reception_constraints* implements the constraints on the received signal described in section 5.2. The blocking reception of this signal is implemented by the dynamic predicate *channel* as mentioned in section 4.

Covering the transitions of the test model of the filter as mentioned in section 2, we obtain two test data TD_1 for the bandwidth test and TD_2 for the cutoff frequency test:

```
TD1 = ( In   = sig(sine, 2.0 .. 10.0,
                   10000.0 .. 10000.0, -0.0 .. 0.0),
        Out = sig(sine, 1.9 .. 10.1,
                   10000.0 .. 10000.0, -0.1 .. 0.1))
TD2 = ( In  =  sig(sine, 2.0 .. 10.0,
                   1000.0 .. 1000.0, -0.0 .. 0.0)
         Out = sig(sine, 1.214 .. 7.271,
                   1000.0 .. 1000.0, 0.585 .. 0.985))
```

where *In* represent the primary input test signal applied and *Out* the primary output signal when the filter cutoff frequency is equal to 1000 Hz and tolerances d1, d2 in F::T#2{L2} are equal to 0.1.

6. The tool Copernicia

We are currently developing a tool named *Copernicia* that implements our modeling and testing approach. The graphical user interface (GUI) is written in C++ with the ILOG Views graphic library. We use CLP and the *ECL^{i}PS^e* solver [3] for the generation of the test data. Figure 6 shows the modeling of the board considered in section 5. The tool provides two user modes: board modeling and board test data generation. In the main panel, with the modeling mode, the upper left white background window represents the board level modeling and grey background windows represent CFSM at the block level modeling. The lower white background window (output window) gives information about the internal modeling of the board. With the board test data generation mode, the user can in a friendly way click a block to generate its test data that are written in the output window. He can also click a state or a transition on block models in order to generate test data for a particular behavior.

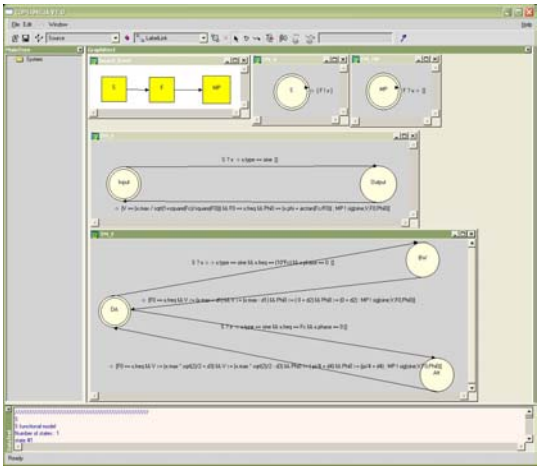

Figure 6. The tool Copernicia

7. Conclusion and future works

In this paper, we have presented our hierarchical modeling and testing approach for automatic generation of test data for mixed-signal boards. This approach is based on functional testing due to the particularity of maintenance testing.

We have proposed a graphical language for the high level description of the board. The graphical nature of the language aims at making it more convivial for users. We have also proposed a modeling approach based on CFSM and a transition language for the description of board behavior as well as the description of test strategy. CFSM are well known by testing engineers, so the use of the proposed solution should be easy for them. An implementation of our approach using CLP has been presented and its application to a simple board made of a single filter has been shown. The tool Copernicia, which implements the solutions proposed, has been presented. Future works include the application of the approach to more complex boards in order to validate or exhibit its limits.

References

[1] Bushnell Michael L., Agrawal Vishwani D. “Essentials of Electronic Testing for Digital, Memory and Mixed-signal VLSI”. Springer (2000).

[2] Gilles B., Nicolas V.-A., Lemarchand L., Marcé L., Castel B. “Towards a New Modelling of Mixed Signal Boards For Maintenance Testing”. Proceedings of the 11th IEEE International Mixed-Signals Testing Workshop, IMSTW'05 (2005).

[3] Cheadle A.M., Harvey W., Sadler A.J., Schimpf J., Shen K., Wallace M.G. “Eclipse: An introduction”. Technical Report IC-Parc-03-1, IC-Parc, Imperial College London (2003).