Lemma 1 Ltd.
c/o Interglossa
2nd Floor
31A Chain St.
Reading
Berks
RG1 2HX

ClawZ
—
System Test and Evaluation Report

Version:	8.2
Date:	14 February 2002
Reference:	DAZ/RPT016
Pages:	6

Prepared by:	R.B.Jones
Tel:	+44 1344 642507
E-Mail:	RBJones@RBJones.com

0 DOCUMENT CONTROL

0.1 Contents

0.2 Document Cross References

[1] LEMMA1/DAZ/INT517. *ClawZ - Diagram Translator Test Cases.* R.B. Jones, Lemma 1 Ltd., `rbjones@rbjones.com`.

[2] LEMMA1/DAZ/PLN031. *ClawZ Extensions — Proposal.* R.B. Jones, Lemma 1 Ltd., `rbjones@rbjones.com`.

0.3 Changes History

Issue 7.1 March 2001 release (real ClawZ).

Issue 8.1 January 2002 release (complete rewrite, ClawZ extensions).

Issue 8.2 February 2002 release (amendment to evaluation information concerning naming and ordering).

0.4 Changes Forecast

None.

1 GENERAL

1.1 Introduction

This document is one of the deliverables from the ClawZ extensions project, placed by DERA Malvern with **Lemma 1 Ltd.**. For the relevant proposal see [2].

It provides a report on the system test and evaluation activities, and is based on evaluating the translator using the system tests specified in [1].

1.2 Overview

Under this contract a substantial increment in the functionality of ClawZ has been implemented, involving a major change to the architecture of the software.

The following topics are addressed in the rest of this report:

Test Results Short summary of the results of the system test activity.

Evaluation General notes on the changes, in the light of initial evaluation.

2 TEST RESULTS

All tests go through ClawZ with the current full metadata set, demonstrating satisfactorily all the new features, as well as confirming prior functionality. The principle new features are:

- Artificial subsystems.
- Masked Subsystems
- Block References
- Synthesis of bus creators and selectors and of mux and demux blocks.
- Generation of metadata for libraries.

For details of which tests address these items see [1].

The tests have also been run with all metadata for mux and demux removed, to see to what extent the new provision for synthesis of specifications for these blocks could replace the previous treatment by library instantiation.

3 EVALUATION

Limited evaluation has been undertaken. The results suggest that more intensive scrutiny of some aspects of the new functionality may be desirable.

3.1 Ordering and Naming

In the final extension to this contract work was commissioned in this area.

This has resulted in a specific order being defined for the declarations in a subsystem schema, and in changes to and stablisation of the algorithm for mapping Simulink blocknames and paths into Z. Global names are now systematically compounded from local names, and consequently the facility enabling users to override the algorithm and chose their own Z names has been modified to operate only on local names in specified contexts.

No problems with these new features have been identified by the system test, though some problems in formatting and parsing exotic names (e.g. containing "/" characters) which were present in previous versions of ClawZ were discovered and eliminated during the development.

The previous practice of writing name-mapping information to the log file has been retained, though this does sometimes result in disruption of the information by garbage collection messages.

3.2 Masked Subsystems

The implementation of masked subsystem has been more thorough than had originally been expected, since finding a reasonable compromise short of this proved difficult.

Our evaluation has found no problems with this new facility, but the following points are noted:

- Subsystems containing uses of free local variables are compiled with parameters to permit the values to be passed to them. Normally such variables are introduced by masked subsystems and would not occur in the top level system of a model. However, if a block reference is made to a subsystem in a library which contains free local variables (which is supported but not recommended), it is possible that the reference occurs in the referring model outside the context of an appropriately masked subsystem. In that case the top level system will be translated as an abstraction expecting the values of these free variables to be supplied to it in a binding. There are three courses of action which may be undertaken in this case.

    1. change the library subsystem referred to so that the passage of the variables in question is explict rather than implicit (i.e. include all the local variables used in the subsystem as masked variables of the subsystem).
    2. include the variables as masked variables in some subsystem of the referring model enclosing the reference.

3. supply manually a calling schema which invokes the top level system schema supplying the required values as parameters.

- There are a number of more or less arbitrary choices to be made in generating Z. These choices have consequences which are not readily assessed at this or earlier stages in the development, in particular for ease or cost of reasoning or other kinds of processing of the resulting specifications. It is possible that the choices we have made (for example, the use of horizontal schema declarations in abstractions and theta terms) may require some additional proof automation. The have been chosen with suitability for automation and legibility of specification in mind, and can be adjusted if they are found not to be ideal. Types where required have always been rendered as "U", which assists in automated inference, but could be rendered more specifically in some cases without adverse consequences (or in others at the cost of additional proof obligations).

3.3 Block Synthesis

Block synthesis is dependent upon ClawZ being able to deduce sufficient information about the structure of the signals attached to the ports of the block whose specification is to be synthesized.

The present synthesis algorithms are relatively conservative, and generally expect that input signal structures are fully resolved. The ability to infer signal structure is however limited. It depends on information either supplied encoded in the metadata (giving information about blocks translated by library instantiation) or on information hard coded into ClawZ about certain kinds of block.

The compromise realised at this release fails to infer sufficient information for synthesis to take place in circumstances which do not prevent library instantiation, and therefore does not completely replace, even for the supported block types, the use of library instantiation. The use of synthesis is analogous to the use of a macrogenerator, and by comparison with library instantiation (which is analogous to a subroutine call) results in larger specifications. Some practical experience will be necessary before an optimal mix of intantiation and synthesis is reached, and this experience will probably yield ideas for further development of the synthesis capability (or the library instantiation mechanisms).

The evaluation has considered possible causes of failure of synthesis while translating the current tests with no Mux/Demux metadata (thus inhibiting library instantiation and forcing an attempt at synthesis).

It is thought that the following factors contribute:

1. lack of information about the structure of signals on top level system input ports.
2. insufficient information about the structure of signals coming from library blocks.
3. conservative synthesis code, which may require more information than is strictly necessary.

It is not easy to assess the relative practical significance of these two factors. They could be eliminated or mitigated by the following enhancements:

- Provision for the user to supply information about the structure of signals on input ports on the top level system.
- Extension of the PortTypes metadata parameter syntax to allow full bus structure to be specified, and changes to metadata generation to provide this information when translating libraries.
- More aggressive synthesis code.

It should be noted that the incompleteness of signal type analysis in ClawZ, by comparison with the obvious capabilities of Simulink, is attributable to the limited knowledge by ClawZ of the characteristics of the Simulink Library blocks. The information available to ClawZ now comes from two sources:

1. the library metadata
2. hard coded in signal analysis and signal propagation code

The former source is more extensive in its coverage of the library, but more limited in its expressiveness. The latter, though unlimited in expressiveness, and open ended in its extent, at present only covers bus constructors and selectors, and Mux/DeMux.

Concerning aggressiveness of synthesis code, there are choices to be made which affect the reliability of ClawZ in producing consistent and correct translations. Library instantiation, under the control of the user can be used to effect translations whose output is incorrect or even inconsistent. To match the coverage of the present library instantiation facilities might only be possible by coding into the synthesis an unsafe level of aggressiveness. For example, Demux blocks are always instantiated by the current library as if the width of the input signal was the same as the number of output ports. The existing library facility is unable to detect when this condition does not hold, and in such cases the resulting translation would be incorrect.

Under the contract extension a facility to dump to a file the results of signal-type inference has been added.

3.4 Block Reference and Library Translation

The block reference feature is working as anticipated within agreed limits (e.g. that modifications to blocks referred to are not supported). Generation of the required metadata is subject to some awkwardness due to the possibility that multiple runs of ClawZ might prove necessary. This only occurs during the translation of libraries with internal block references.