# MIX - Micronas Interconnect Specification Expander

Development Documentation

6.10.2003

# Contents

**1 Introduction** **3**

**2 Installation and Setup** **3**
2.1 Requirements . . . 3
2.2 Getting from CVS . . . 3

**3 Basic Concepts** **3**
3.1 Input/Output formats . . . 3
3.2 Basical steps . . . 4

**4 The Package: Micronas** **4**
4.1 Library: MixUtils.pm . . . 4
4.1.1 mix_getopt_header, mix_getops, mix_banner, mix_usage and mix_help 4
4.1.2 mix_init and init_ole . . . 5
4.1.3 . . . 5
4.2 Library: MixParser.pm . . . 5
4.3 Library: MixIOParser.pm . . . 5
4.4 Library: MixWriter.pm . . . 5
4.4.1 generate_entities () . . . 5
4.4.2 write_entities () . . . 5
4.4.3 write_architecture () . . . 5
4.4.4 write_configuration () . . . 6
4.5 Library: MixChecker.pm . . . 6

# 1 Introduction

This Document will give Developers a short introduction into Micronas Interconnect Specification Expander. It will not give any help about MIX usage, if you need information about this, please refer the MIX-Userreference. The following Sections will explain the basic concepts of MIX and what you will need to install it(specially cvs).

# 2 Installation and Setup

## 2.1 Requirements

Because of MIX is a Perl script (using additional Packages), you will have to install a Perl 5 interpreter. You may use the Packages included in CVS or optionally, download each of your own from the Internet. If you are using Linux you might find Perl 5 already beeing installed. The Packages included in cvs where used for developing and testing MIX, if you refer problems please you these ones. If you are using Windows you should install WinCVS (most Linux-Distributions include a CVS client per default) for keeping up to date and permiting own changes.

## 2.2 Getting from CVS

Instead using a obsolete Version you should decide to get MIX from CVS. The following steps discribe what to do, if you want to get MIX via this way:

1. install a CVS client (visit http://www.wincvs.org for more information)
2. install a ssh client (visit http://lexa.mckenna.edu/sshwindows/)
3. change to the Base directory where you wan't MIX to recide
4. set a Environment variable called "CVSRSH" to "ssh"
5. enter the command: `cvs -z3 -d:ext:.....  co mix`
6. optionally: install ActiveState Perl and Komodo
7. optionally: download packages from CPan (http://cpan.perl.com)

Further informations about cvs usage could be found at: http://www.sourceforge.net
- select "Site Docs" from the section "SF.net Resources"
- visit section "F", which contains various information about CVS

# 3 Basic Concepts

## 3.1 Input/Output formats

As mentioned above you won't find here any information about "how to build Hierachical, Connectivity and IO-Sheets", because of the MIX-Userreference gives some examples how to do this. In this point please read the MIX-Userreference.

## 3.2 Basical steps

The MIX main program, called "mix_0.pl", is containing calls to functions located in the Micronas Package. These calls represent MIX's procedural flow. Each of this procedures solves some basical problems, the following steps represent sections which include similar functions.

1. Initialize the %EH variable with all the default configuration
2. Processing of command line switches, at least one argument has to be specified
3. Read the input files one by one and retrieve tables (simple convert into Arrays of Hashes)
4. Retrieve generator statements (generator, macros) from input data
5. Initialize Hierarchy Data Base and convert to internal format
6. Parse connectivity and IO sheet and convert to internal format
7. Apply connectivity macros, hierachical and connectivity generation
8. Do some internal clean up
9. Replace all occurences of %MACRO% in %hierdb% and %conndb%
10. Add connections and ports if needed (hierarchy traversal)
11. Clean up again
12. Add a list of all signals for each instance and generate entities
13. Dump intermediate data
14. Write output (entities, architecture, configuration and status) and exit

# 4 The Package: Micronas

Here can be found some additional information about the MIX Libraries which are inseperately part of MIX. Most of work is done here.

## 4.1 Library: MixUtils.pm

This package includes some utilities which are used in most of the other packages. Because of there are a lot of functions here, only the most importend are mentioned here.

### 4.1.1 mix_getopt_header, mix_getops, mix_banner, mix_usage and mix_help

These functions help us solving user interactions.
- mix_getopt_header() and mix_getops() are used to get command line options, paased by the user. - mix_banner() prints the headings MIX outputs to std every run - mix_usage() and mix_help() are needed to print the program help and/or usage

### 4.1.2 mix_init and init_ole

The function mix_init() initializes the %EH variable with all the configuration we have/need. This variable keeps all configuation information while the program is running.
The function init_ole() initializes the Microsoft OLE System, which is needed for editing with Excel sheets.

### 4.1.3

## 4.2 Library: MixParser.pm

The package provides the parsing capabilites for the MIX project. A matrix of information in some well-known format is converted into intermediate format and/or source code files.

## 4.3 Library: MixIOParser.pm

This Package provides the parsing capabilites for the MIX project. Take a matrix of information in some well-known format and convert it into intermediate format and/or source code files MixIOParser is spezialized in parsing the IO description sheet Information about IOcells and Pads will be converted to connection and hierachy and added to the HIER and CONN databases accordingly.

## 4.4 Library: MixWriter.pm

The package provides the output capabilites for the MIX project. Here you can find the following functions: *generate_ entities*, *write_ entities*, *write_ architecture* and *write_ configuration*.

### 4.4.1 generate_entities ()

Scan all of hierachy and create consistent, checked list of entities. An entity has a name and a portmap (generic map). Produced data structure will look like $entities$name$porttype|from|to|mode. Type will be generic or std_logic or std_ulogic ....
This functions relies on $hierdbinst::conn::in|::out to exist!
TODO: Usage of hierdb is not nice. Search for a better way.
TODO: Generics and inout mode ....

### 4.4.2 write_entities ()

This function writes entities into output file(s).

*ENTY* -> write each entity in a file of it's own

*COMB(INE)* -> combine entity, architecture and configuration

This is mostly important for architecture and configuration. In the both cases above the entity name is used as filename. In all other cases "*efname*" is used as entity file name (write all entities into one file of that name). In combine mode, *entity.vhd* is choosen as filename. Filename extension defaults to VHDL.

### 4.4.3 write_architecture ()

This function writes architecture into (VHDL/Verilog) output file(s).

#### 4.4.4 write_configuration ()

This function writes configurarion into (VHDL/Verilog) output file(s).

## 4.5 Library: MixChecker.pm

Ths package here provides the checking capabilites for the MIX project. Accepts the intermediate (aka. final connection and hierachy description) and check if everything against your company design guide lines. Through plug-ins you can add these checks at will.