

MIX – Micronas Interconnect Spec expander

Introduction

Management Summary

Overview

Table of contents:

MIX – Micronas Interconnect Spec expander.....	1
Introduction.....	1
Management Summary	1
Overview	1
Table of contents:	1
Installation	2
Getting Started	2
A simple example	2
Mix it!	3
You get what you typed.....	5
Details	7
Common excel worksheet properties.....	7
HIER sheet properties	7
HIER columns details	8
Special HIER sheet properties.....	9
CONN sheet details	9
CONN columns details	9
Special CONN sheet signals.....	9
%OPEN% aka. open	9
Constants	9
Generics and Parameters	9
Predefined and user macros.....	9
CONN sheet macros	10
Generator statements.....	11
IO sheet.....	11
I2C sheet.....	11
VI2C sheet.....	11
Alarm clock example	11
MIX converter man page	11
Synopsis	11
Command line switches.....	11
Runtime options and configuration	12
Misc features	12
-delta mode	12
Intermediate Excel Sheet	13
Alarm clock example	13
Core logic.....	13
IO logic.....	13
Other examples	13

Known Bugs and limitations	13
Issue tracking: CADNET -> Issue tracking -> CAD Software -> MIX.....	13
Links.....	14
MIX paper and documentation.....	14
Micronas internal	14
Others.....	14
Used Software	14

Installation

The MIX toolset uses the [Perl](#) scripting language. Please install a recent version of Perl on your workstation, e.g. the freely available [ActiveState Perl](#) from K:\PROJECTS\MIX\PROG\ActivePerl-5.6.1.633-MSWin32-x86.msi.

Apart from that no installation is required.

Start MIX from the network drive like described in the examples below. That will also make sure, that you are using the most recent release automatically. After the Perl installation, you can run MIX. immediately. Open a command shell on your desktop workstation

Start -> Ausführen -> cmd

and type

K:\Projects\MIX\PROG\mix_0.pl foo.xls

Caveat: Currently MIX is available on MS-Windows, only!

Getting Started

To receive useful results, MIX reads in various input spreadsheet data. At least you will need to prepare a description of your design hierarchy (HIER) and the connectivity sheet (CONN). Optionally MIX will convert a input/output sheet IO, listing input/output pads and iocells and how these are linked to the core logic into appropriate hierarchy and connection lists.

A simple example

To understand the usage of the various tables and options, a very simple example is shown and extended step by step. The simple example just has two components **a** and **b**, which are connected by the signal **sigfoo**. The two instances have a common parent **chip**

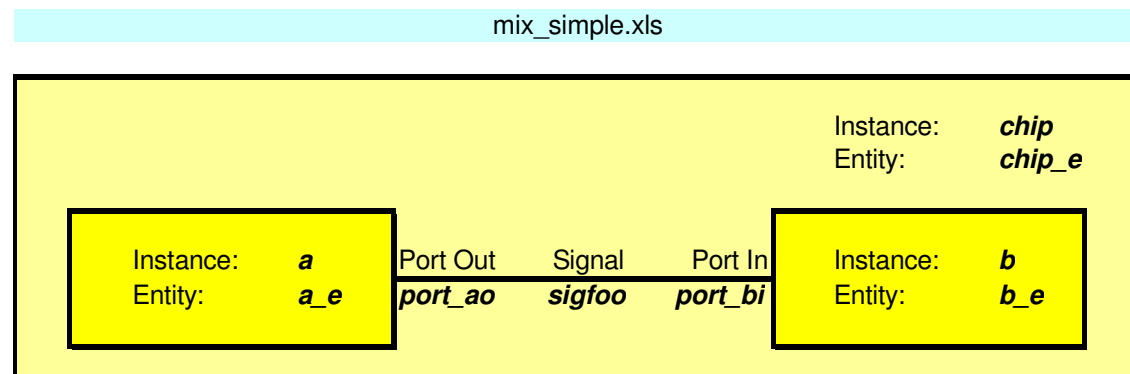


Image 1: Simple mix_simple.xls

The equivalent description of this simple design in MIX is made up from a worksheet **HIER**.

::ign	::gen	::variants	::parent	::inst	::entity	::config	::comment
# Mix SimpleExample, V0.1, 20030630, hierachy							
		Default	TESTBENCH	chip	chip e	chip e rtl conf	
		Default	chip	a	a e	a e rtl conf	
		Default	chip	b	b e	b e rtl conf	

The first row defines the table headers names. The names have to be in the form “::NAME”. Several of the columns are required, some are optional and you can define additional columns on your own. For HIER sheets the “::inst” column is the primary key. One design element will be generated for each new name of an ::inst row. If a name is defined several times, these lines will be overloaded or summarized, depending on built-in rules of the MIX converter.

The “::ign” column has to come first. If it starts with a #, the rest of this row will be ignored. All other columns can be added in arbitrary order.

The second required worksheet is **CONN**:

::ign	::gen	::bundle	::class	::clock	::type	::high	::low	::mode	::name	::out	::in	::descr	::comment
# Mix SimpleExample, V0.1, 20030630, hierachy													
		Chip	Data	Clk	std_ulogic			S	sigfoo	a/port_ao	b/port_bi	Simple	

As you see, this worksheet also starts with the table header definition line. The primary field is the “::name” column. The “::in” and “::out” columns are used to define the drivers and loads for the signals.

Mix it!

Run the MIX converter tool in the directory the excel spread sheet is stored in:

```
$ cd \work\MIX\doc\parts\simple
$ K:\Projects\MIX\PROG\mix_0.pl mix_simple.xls
```

This reads in the design description and evaluates the various sheets. It creates output files with an intermediate excel design description (mix_simple-mixed.xls) and the same data in a internal format (mix_simple.pld). A log file (mix_0.pl.log) and the HDL output files are written in the same run. Image 2 shows a screenshot of the mix_simpe.xls conversion. Only the most important errors and warnings are written to the screen, while a lot of information will be written to the log file. Search for the keywords “ERROR” and “WARNING” to verify proper conversion.

```

C:\WINNT\System32\cmd.exe

H:\work\MIX\doc\parts\simple>K:\Projects\MIX\PROG\mix_0.pl mix_simple.xls
#####
##### mix_0.pl <Revision Revision: 1.12 >
#####
##### MIX
#####
03/06/30 14:28:15 1336: INFO: Not all lowercase in new inst element TESTBENCH!
03/06/30 14:28:15 1336: file mix_simple.pld already exists! Will be overwritten

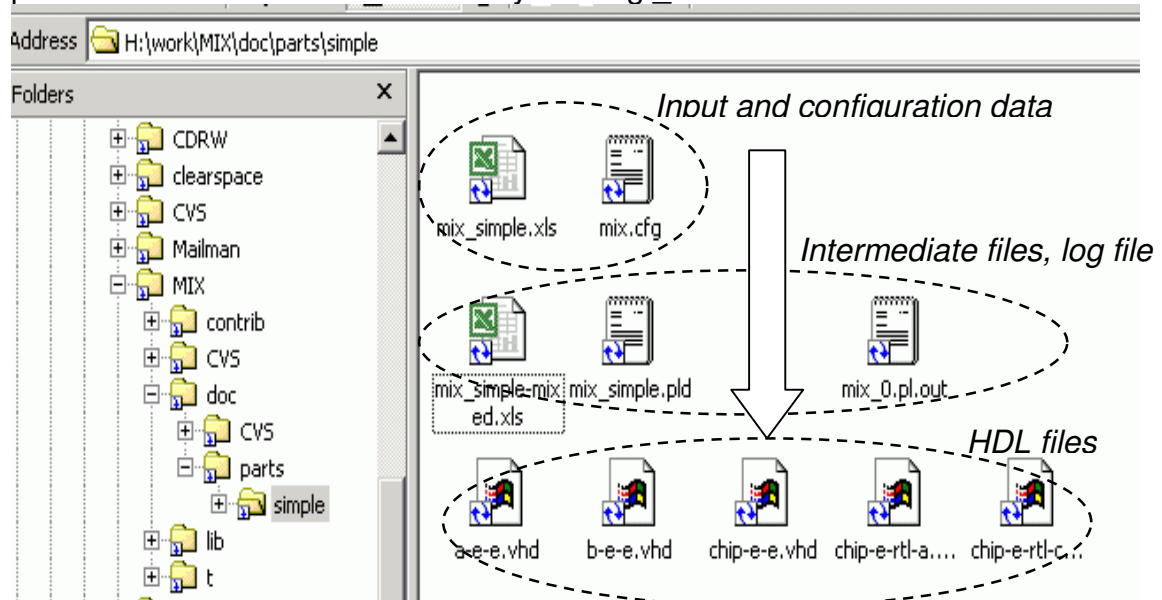
03/06/30 14:28:15 1336: File mix_simple-mixed.xls already exists! Contents will
be changed
03/06/30 14:28:15 1336: Rotating 3 old sheets of CONN!
03/06/30 14:28:15 1336: File mix_simple-mixed.xls already exists! Contents will
be changed
03/06/30 14:28:15 1336: Rotating 3 old sheets of HIER!
03/06/30 14:28:16 1336: ===== SUMMARY =====
03/06/30 14:28:16 1336: SUM: Summary of checks and created items:
03/06/30 14:28:16 1336: SUM: checkforce 0
03/06/30 14:28:16 1336: SUM: checkunique 0
03/06/30 14:28:16 1336: SUM: checkwarn 0
03/06/30 14:28:16 1336: SUM: conn 1
03/06/30 14:28:16 1336: SUM: errors 0
03/06/30 14:28:16 1336: SUM: inst 8
03/06/30 14:28:16 1336: SUM: warnings 0
03/06/30 14:28:16 1336: SUM: Number of parsed input tables:
03/06/30 14:28:16 1336: SUM: conf 0
03/06/30 14:28:16 1336: SUM: hier 1
03/06/30 14:28:16 1336: SUM: conn 1
03/06/30 14:28:16 1336: SUM: io 0
03/06/30 14:28:16 1336: SUM: vi2c 0

H:\work\MIX\doc\parts\simple>

```

Image 2: Screenshot mix_simple conversion

All output files are stored in the current working directory. Old versions of the output files are overwritten. Except the log file that is appended by each converter run. The intermediate excel description file keeps a history of previous HIER and CONN sheets by rotating _N extended worksheet names.



You get what you typed

MIX generates various HDL files defined by the input data. If you select VHDL (also the default language) as output description for a hierarchy element, each element results in an appropriate entity, architecture and configuration description. By default MIX writes one file for all entities, one for all architecture and one for all configuration descriptions. Those file names are derived from the last excel input file name by stripping of the .xls extension and attaching a -e.vhd, -a.vhd and -c.vhd.

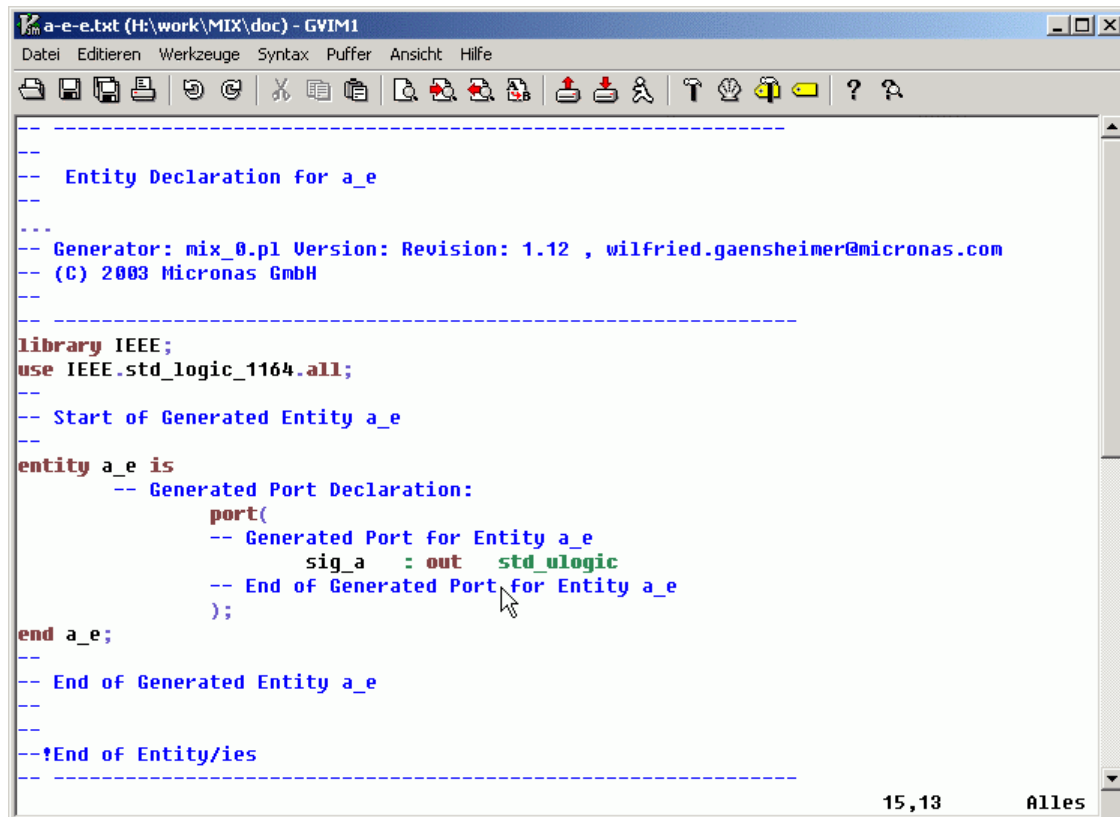
Here the working directory of the simple example contains a file mix.cfg, which is the convenient storage for MIX run-time configuration options. The lines "MIXCFG outarch ARCH", "MIXCFG outenty ENTY", and "MIXCFG outconf CONF" switch MIX outputs to separate files for each entity, architecture and configuration. In this case the file names are defined by the element name:

Type	Basename	Extension	Example
Entity	::entity-column	-e.vhd	<i>chip-e-e.vhd</i>
Architecture	::entity-column	-rtl-a.vhd	<i>chip-e-rtl-a.vhd</i>
Configuration	::conf-column	-c.vhd	<i>chip-e-rtl-conf-c.vhd</i>

"_" in the element names are converted to -.

By default MIX does not write output data for leaf blocks (instances which are not parent for other instances). Adding a line like "MIXCFG generate.output.arch leaf" into the mix.cfg file changes that.

The generated output files contain head, body and footer sections. See the screenshots of the file a-e-e.vhd and chip-e-rtl-a.vhd for examples of an entity and an architecture definition.

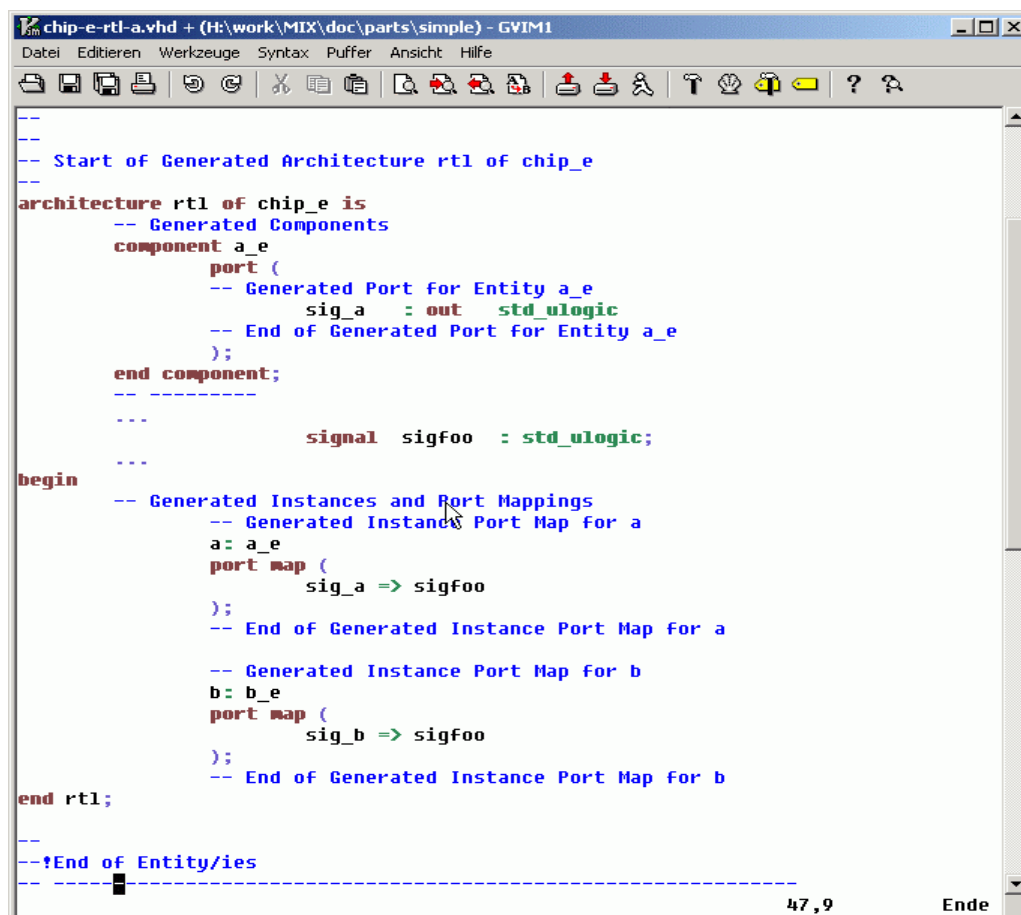


The screenshot shows a text editor window titled "a-e-e.txt (H:\work\MIX\doc) - GVIM1". The menu bar includes "Datei", "Editieren", "Werkzeuge", "Syntax", "Puffer", "Ansicht", and "Hilfe". The toolbar contains various icons for file operations and editing. The main text area displays the following VHDL code:

```
-----  
-- Entity Declaration for a_e  
--  
-- Generator: mix_0.pl Version: Revision: 1.12 , wilfried.gaensheimer@micronas.com  
-- (C) 2003 Micronas GmbH  
--  
-----  
library IEEE;  
use IEEE.std_logic_1164.all;  
--  
-- Start of Generated Entity a_e  
--  
entity a_e is  
    -- Generated Port Declaration:  
    port(  
        -- Generated Port for Entity a_e  
        sig_a : out std_ulogic  
        -- End of Generated Port for Entity a_e  
    );  
end a_e;  
--  
-- End of Generated Entity a_e  
--  
--!End of Entity/ies  
-----
```

The status bar at the bottom right shows "15,13" and "Alles".

Image 3: Entity a-e-e.vhd



The screenshot shows a text editor window titled "chip-e-rtl-a.vhd + (H:\work\MIX\doc\parts\simple) - GVIM1". The menu bar and toolbar are the same as in Image 3. The main text area displays the following VHDL code:

```
--  
-- Start of Generated Architecture rtl of chip_e  
--  
architecture rtl of chip_e is  
    -- Generated Components  
    component a_e  
        port (  
            -- Generated Port for Entity a_e  
            sig_a : out std_ulogic  
            -- End of Generated Port for Entity a_e  
        );  
    end component;  
    --  
    signal sigfoo : std_ulogic;  
    --  
begin  
    -- Generated Instances and Port Mappings  
    -- Generated Instance Port Map for a  
    a: a_e  
    port map (  
        sig_a => sigfoo  
    );  
    -- End of Generated Instance Port Map for a  
    -- Generated Instance Port Map for b  
    b: b_e  
    port map (  
        sig_b => sigfoo  
    );  
    -- End of Generated Instance Port Map for b  
end rtl;  
--  
--!End of Entity/ies  
-----
```

The status bar at the bottom right shows "47,9" and "Ende".

Image 4: architecture chip-e-rtl-a.vhd

Details

Common excel worksheet properties

All excel worksheets parsed by MIX share some common properties.

There needs to be a header line consisting only of keywords with leading double colon. All data before the header line is ignored. Only the first header line will be evaluated. Data in columns with no header or malformed headers will be ignored.

Commonly understood table headings are `::ign` and `::comment`. The `::ign` column is special, because it needs to be the first column of a sheet. If a cell in the `::ign` column starts with a #, the complete row is ignored.

The `::comment` column can contain user or program generated comments for a given row. It's data will be appended to it's contents as it appears.

MIX reads the excel cell values. This is what you see, not the real contents of the cells. Thus all excel formulas can be used to define the cell values.

A lot of predefined *text macros* are understood and converted by MIX. A text macro is made up by a name surrounded by % signs. Retrieve a complete list of macros with the `-listconf` command line switch and grep all lines starting with "MIXCFG macro". The table %macro% gives a list of predefined macros, their default settings and whether this macro can be used by the user. You can use text macros inside of any cell.

::ign	::gen	::vars	::part	::inst	::lang	::entity	::config	::comment
#MixSimpleExample, MacroExpansion, V01, 20030630, hierarchy								
		Default	TESTBENCH	chip	V-DL	%inst%_e	%inst%_e rtl_conf	
		Default	chip	a	V-DL	%inst%_e	%inst%_e rtl_conf	
		Default	chip	b	%LANGUAGE%	%inst%_e	%inst%_e rtl_conf	

A second type of macros are available to reference other columns of a like through `::NAME%`. This will be replaced by the contents of the `::NAME` column of this row. `::NAME` has to be defined for the current worksheet, obviously. In the above example `::inst%_e` will evaluate to `chip_e`, `a_e` and `b_e` accordingly. `%LANGUAGE%` becomes `vhdl`. Macro expansion happens just before the intermediate design data is written out, after evaluation of all input data. Recursive macro expansion is not implemented. Macros in primary keys like signal and instance names are evaluated at signal or instance creation time.

HIER sheet properties

The hierarchy of a design is defined in the HIER sheet. By default the HIER sheet is named HIER, but you can set the configuration value `hier.xls` to a Perl regular expression. MIX will then consider all worksheets which names match the perl regular expression, to be HIER definitions. In that case you are free to use different header definitions on different sheets.

The HIER sheets require at least the columns marked man. (mandatory) in the following table.

Column name	Description	Default value	Req.	Example
::ign	Ignore line	<empty>	man.	# comm.
::gen	Generator and match	<empty>	man.	see below

::variants	Variant selector	Default	opt.	Var1
::parent	Instance name of this instances parent cell	W_NO_PARENT	man.	chip
::inst	Instance name, primary key!	n/a	man.	a_i1
::lang	Language definition	VHDL	opt.	vhdl
::entity	Entity name	W_NO_ENTITY	man.	a
::config	Configuration name	W_NO_CONFIG	man.	a_rtl_conf
::shortname	Short name	<empty>	opt.	text
::use	Additional, project specific libraries	<empty>	opt.	padlib.foo
::comment	Comment field	<empty>	opt.	text

Internally the keywords ::debug, ::hierarchy, ::skip and ::default are used. Please do not use them. Apart from that you are free to add columns of your own. User defined columns are usable in %::NAME% macro expansion and are listed in the intermediate design data output.

Only the ::inst column has to contain a value in each row (which could be a %::NAME% macro, though). All other columns will receive more or less reasonable default values in case they are left empty.

HIER columns details

- ::gen** If a cell here is not empty, the line will be considered as generator. See description of generator statements below.
- ::variants** Select a line depending on the –variant command line switch. If –variant VAR is set, only lines whose ::variant cell contains the keyword VAR, Default or empty, are selected and read in. Several variants may be given in one cell, separated by “,”. Without specifying the –variant switch, the “Default” and empty ::variant cells are read in and evaluated.
- ::inst** Defines the instance name. If the same name appears in several rows, the resulting row will be overloaded from all input rows. The exact behaviour depends on the column name. Some are concatenated, some are replaced.
- ::lang** HDL language selection, case insensitive. If this column is omitted or empty, VHDL output is generated. The default value can be changed by means of the macro.%LANGUAGE% macro. Currently only VHDL and Verilog are supported (XXX: Verilog output still t.b.d.).

Additional columns:

- ::arch** If no ::arch column is given, architecture will default to "rtl" (configuration hier.field::arch.[3] = rtl; cannot be changed globally)
- ::use** Add project specific libraries and work packages to the HDL description files. See the ::use details below.

Special HIER sheet properties

%TOP%

CONN sheet details

CONN columns details

Special CONN sheet signals

Special signal names:

%LOW%, %HIGH%, %LOW_BUS%, %HIGH_BUS%

%OPEN% aka. open

Used the %OPEN% signal to leave some pins of module `bar port a open` MIX has no knowledge about ports. Everything is defined in terms of signals and instances. Use the "open" pseudo signal to define the extra pins.

E.g.: wire foo/a port to bar/a port. bar/a has extra pins.

signal_a, ::high 7, ::low 0, ::out foo/a ::in bar/a

%OPEN%, ::high 1, ::low 0, ::out -, ::out bar/a(9:8)

XXX write this in spreadsheet XXX

You could also force the ::in pins to high or low, instead, e.g. use %HIGH%, %HIGH_BUS%, %LOW

or %LOW_BUS%. Or a constant.

Constants

Generics and Parameters

Predefined and user macros

The following table contains a list of predefined macros. Some of them are defined at run time (e.g. the current date), some are for internal purposes only. macros marked with a yes in the "User" column can be set freely by the user on the command line or in the mix.cfg configuration file.

Macro Name	Default Value	Description	User
%0%	mix_0.pl	Program name	<i>run time</i>
%ARGV%	K:\Projects\MIX\PROG\mix_0.pl – listoncf	Program arguments	<i>run time</i>
%BUFFER%	buffer		yes
%BUS_TYPE%	std_ulogic_vector		yes
%CONST%	__CONST__		yes
%DATE%	Mon Jun 30 16:22:41 2003-06-30	Current date	<i>run time</i>
%DEFAULT_MODE%	S	Signal default mode	yes
%EMPTY%		Empty string	yes
%GENERIC%	__GENERIC__		-
%H%	\$	internal	-
%HIGH%	mix_logic1	Name of logic high value	yes

%HIGH_BUS%	mix_logic1_bus	Name of logic high value bus	yes
%HOME%	H:\		-
%IOCELL_SELECT_PORT%	select		yes
%IOCELL_TYPE%	__E_DEFAULT_IOCELL__		-
%IOCR%	\n		-
%LANGUAGE%	vhdl		yes
%LOW%	mix_logic0		yes
%LOW_BUS%	mix_logic0_bus		yes
%NULL%	0		-
%OPEN%	open		-
%PAD_CLASS%	PAD		yes
%PAD_TYPE%	__E_DEFAULT_PAD__		-
%PARAMETER%	__PARAMETER__		-
%PROJECT%	NO_PROJECT_SET		yes
%SIGNAL%	std_ulogic		yes
%SPACE%			yes
%TAB%	\t		-
%TBD%	__W_TO_BE_DEFINED		-
%TOP%	__TOP__		yes
%UNDEF%	ERROR_UNDEF		-
%UNDEF_1%	ERROR_UNDEF_1		-
%UNDEF_2%	ERROR_UNDEF_2		-
%UNDEF_3%	ERROR_UNDEF_3		-
%UNDEF_4%	ERROR_UNDEF_4		-
%USER%	wig		<i>run time</i>
%VERILOG_TIMESCALE%	'timescale 1ns / 1ns;		yes
%VERSION%	Revision: 1.12		-
%VHDL_NOPROJ%	-- No project specific VHDL libraries		-
%VHDL_USE%	-- No project specific VHDL libraries		yes
%VHDL_USE_ARCH%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes
%VHDL_USE_CONF%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes
%VHDL_USE_DEFAULT%	library IEEE;\nuse IEEE.std_logic_1164.all;\n		yes
%VHDL_USE_ENTY%	%VHDL_USE_DEFAULT%\n%VHDL_USE%		yes

The default values can be changed by using

```
-conf macro.%THIS_MACRO%=my_value
```

command line switch. New macros can be defined the same way.

Alternatively a line like

```
MIXCFG macro.%THIS_MACRO% my_value
```

to the mix.cfg configuration file will achieve the same result.

CONN sheet macros

not text macros, but MH, MD and MX generators:

MH: Macro header: needs to match against the MX line (\$X defines variables)

MD: Body of macro. Will be inserted after variable replacement.

MX: Triggers macro expansion. Will be matched against all MH lines known.

Defines variable values.

Generator statements

IO sheet

I2C sheet

VI2C sheet

Alarm clock example

MIX converter man page

Synopsis

Command line switches

- out OUTPUTFILE.ext defines output filename and type
- outenty OUT-e.vhd|ENTITY|COMB
 - Write all entities into OUT-e.vhd.
 - If argument is ENTITY, each entity will be written into a file called *entityname-e.vhd*.. (The exact naming depends on changeable rules).
 - If argument is COMB, entity, architecture and configuration will all be written into one file called *entityname.vhd*
- outarch OUT-rtl-a.vhd|ARCH|COMB
 - See description of outenty option.*
- outconf OUT-c.vhd|CONF|COMB
 - See description of outenty option.*
- combine
 - write entity, architecture and configuration into one file for each entity. Shortcut for setting -out[enty|arch|conf] to COMB individually.
- dir DIRECTORY
 - write intermediate, internal and backend data into the given DIRECTORY. By default MIX writes to the current working directory
- top TOPCELL
 - use TOPCELL as top. Default is TESTBENCH or daughter of TESTBENCH.
- adump
 - dump internal data in ASCII format, too (debugging, use with small data set).
- variant VAR1
 - Select VAR1 from the HIER worksheet.
- conf key.key.key=value
 - Overload \$EH{key}{key}{key} with *value* or add a new configuration variable.
- listconf
 - Print out all available/predefined configurations options

-delta

Output will be compared against previous runs.

-sheet SHEET=MATCH

SHEET can be one of "hier", "conn", "vi2c". MATCH is a perl regular expression.

Add your options here

"Standard" options:

my @stdopts = qw(help|h! verbose|v! quiet|q! nobanner! debug:i

makeopts=s@ gmakeopts=s@);

Caveat: the -h option will not work on MS-Windows

Runtime options and configuration

Runtime configuration is controlled by (increasing precedence):

- ❑ built-in default values
- ❑ mix.cfg files
 - if found \$HOME, \$PROJECT and/or in current directory.
 - Format is: MIXCFG name.of.conf value
- ❑ CONF sheet found in input xls files
- ❑ command line switch
 - conf foo.bar=value
- ❑ dedicated command line options

MIX reads in mix.cfg configuration files in the following locations:

1. \$ENV{HOME}
'HOMEDRIVE', 'HOMEPATH', 'USERPROFILE' or 'C:\'
(only from the first matching location)
2. \$ENV{PROJECT}
3. . (cwd)

Order: HOME / HOME / cwd() / -conf (last wins)

Misc features

-delta mode

Do not change output files, but report number of changes. Adds extra sheets DIFF_CONN and DIFF_HIER (and old versions of them) to the intermediate output. The FOO.pld internal output gets overwritten, though. All messages are appended to mix_0.pl

If a new HDL-file needs to be created by the changes, the -delta mode is not be applied for that file, but it is generated as new. If you never have written a FOO-mixed.xls intermediate output, there is no HIER or CONN sheet generated.

By adding the following two lines to your configuration (e.g. ./mix.cfg), delta mode will be the default:

```
MIXCFG output.generate.delta 1
MIXCFG output.delta sort,remove
```

See the configuration option description for more details.

-nodelta switches delta mode off, then.

Intermediate Excel Sheet

Format of intermediate xls sheet will be kept as is as long as the number and order of columns is unchanged.

MIX saves three old versions of the generated HIER and CONN sheets. The worksheets names are rotated by a trailing _ and number.

Output Redirection

MIX writes all output into the current working directory. By using the

`-dir DIRECTORY`

option, you can set the output directory for all intermediate, internal and HDL files to DIRECTORY. Absolute path names defined by other options (e.g. –out) are not changed, though. If DIRECTORY does not exist it will be created.

To have separate output directories, use the mix.cfg file:

```
MIXCFG output.path HDLDIRS
MIXCFG internal.path MIXINTERNAL
MIXCFG intermediate MIXINTERMEDIATE
```

writes internal, intermediate and HDL files to the given pathes. All directories have to exist and will not be created.

Alarm clock example

Core logic

IO logic

Other examples

Known Bugs and limitations

MS-Win/UNIX end-of-line issue:

Some EDA tools are not able to cope with the different end-of-line (CR vs LF/CR) of UNIX and MS-Windows. Use

```
$ module load freeware; recode "pc..lat1" *.vhd
```

to fix that after transferring data.

Issue tracking: CADNET -> Issue tracking -> CAD Software -> MIX

If you find unexpected or buggy behavior, please issue a trouble ticket. Don't forget to add a short description. The test case should contain all source files and also log files, the exact command line switches and configuration files applied.

Please provide a comprehensive description of issues found including all required input data and command line switch to allow fast debugging and fixing.

Links

MIX paper and documentation

EDP_2003_final_030331.pdf

MIX_Specification.xls (see <\\Galaxy\Development\PROJECTS\MIX>)

MIX_Intro.ppt (see <\\Galaxy\Development\PROJECTS\MIX>)

Micronas internal

Micronas HDL coding guidelines.pdf

Others

Resources

Downloads for:

- IO examples
- Standard bus examples
- NAND Tree
- BS example
- misc usefull macros

Used Software

Perl

Kommodo

Vim

WinCVS