MIX - Micronas Interconnect Specification Expander

Development Documentation

6.10.2003

Contents

1 Introduction

This Document will give Developers a short introduction into Micronas Interconnect Specification Expander Libraries. It will not give any help about MIX usage, if you need information about this, please refer the MIX-Userreference. The following Sections will give some functional overview and coding examples.

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 (most configuration is hold inside of it)
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

3.3 Spreadsheet Operations

The first example will show the usage of MixUtils::IO which is used for Spreadsheet read and write Operations.This code converts a Excel Spreadsheet to the Star-Office Spreadsheet format. See MixUtils Section for other supported file-formats).

```
#!/usr/bin/perl -w

use strict;
use Cwd;
use File::Basename;
use Getopt::Long qw(GetOptions);
use Pod::Text;

use vars qw($pgm $base $pgmpath $dir);

$dir = "";
($^O=~/Win32/) ? ($dir=getcwd())=~s,/,\\,g : ($dir=getcwd());

BEGIN{     #
```

```
    ($^O=~/Win32/) ? ($dir=getcwd())=~s,/,\\,g : ($dir=getcwd());

    ($pgm=$0) =~s;^.*(/|\\);;g;
    if ( $0 =~ m,[/\\],o ) { #$0 has path ...
        ($base=$0) =~s;^(.*)[/\\]\w+[/\\][\w\.]+$;$1;g;
        ($pgmpath=$0) =~ s;^(.*)[/\\][\w\.]+$;$1;g;
    } else {
        ( $base = $dir ) =~ s,^(.*)[/\\][\w\.]+$,$1,g;
        $pgmpath = $dir;
    }
}

# $pgmpath contains the programs path (this skripts path)

use lib "$pgmpath/MIX/lib/perl";     # search Libraries in Subdir MIX/lib/perl

use Log::Agent;
use Log::Agent::Priorities qw(:LEVELS);
use Log::Agent::Driver::File;

use Micronas::MixUtils;       # contains the EH Variable and other every-day stuff
use Micronas::MixUtils::IO;   # here you find input and
                              # ouput Spreadsheet-File Handling

#*****************************************************************************
# Global Variables
#*****************************************************************************

if(scalar(@ARGV)!=1) {
    logwarn "usage: xls2sxc.pl <excel-file>";
    die;
}

mix_init();

my $file = $ARGV[0];

if(not $file=~ m/.xls/) {
    $file = $file . ".xls";
}

if($ARGV[0]=~ m/.xls$/) {
    $ARGV[0]=~ s/.xls$/.sxc/;
}
elsif(not $ARGV[0]=~ m/.sxc$/) {
    $ARGV[0] .= ".sxc";
}

my $oBook = Spreadsheet::ParseExcel::Workbook->Parse($file);   # Excel Parser Module
```

```
if(!$oBook || !defined $oBook) {
    logwarn("ERROR: File <$file> not found");
    die;
}

my $sheet;
my $sname = "";
my @data = ();

my $ref;


for(my $i=0; $i<$oBook->{SheetCount}; $i++) {  # for all sheets in the file

    $sheet = $oBook->{Worksheet}[$i];    # get Sheet, number $i from Object
    $sname = $sheet->{Name};             # get the Sheets name

    @data = open_xls($file, $sname, 0);  # open $file, read Sheet $sname; no flags
    $ref = pop(@data);       # get the first sheet of the name $sname which was found
    write_sxc($ARGV[0], $sname, $ref);# write to file,
                                      # createsheet with name containing data of $ref
}
```

The number of Sheets, with the same name to keep, as many other things is conatined in the EH Variable. This Variable is initialized by the call of `mix_init()`, optional you may setup your completly own EH Variable, but you will have to look into MixUtils

3.4 Mix parsing Extensions

Writing a Mix parsing Extension is even easy as Spreadsheet operations. Mix does it's work by mapping input Information (taken from a Spreadsheet) on instances and connections. These are keept inside of two Perl-Hashes `Adding a Instance FooBar to TOPLEVEL is done by:`

```
    my %my_instance = ( '::inst' => "FooBar",
                        '::entity' => "foobar",
                        '::parent' => "TOPLEVEL", );

    add_inst(%my_instance);
```

For other settings see `MIX_Userreference`. The definition of those (called the Headers) is done in the Mix-Module `MixUtils.pm` (located in the `mix_init()` Function, inside the `EH` structure). The following Listing shows the I^2C Header definiton:

```
    'i2c' => {
        'xls' => 'I2C',
'req' => 'optional',
'parsed' => 0,
'field' => {
```

```
    #Name    =>       Inherits
    #     Multiple
    #     Required
    #   Defaultvalue
    #     PrintOrder
    #                                  0       1       2 3       4
    '::ign'  => [ qw( 0 0 1 %EMPTY%     1)],
    '::variants'=> [ qw( 1 0 0 Default      2)],
    '::dev'             => [ qw(        0       0       1       %EMPTY%     3)],
    '::sub'             => [ qw(        0       0       1       %EMPTY%     4)],
    '::interface'       => [ qw(        0       0       1       %EMPTY%     5)],
    '::dir'             => [ qw(        0       0       1       RW           6)],
    '::spec'            => [ qw(        0       0       0       NTO          7)],
    '::clock'           => [ qw(        0       0       1       %OPEN%      8)],
    '::reset'           => [ qw(        0       0       1       %OPEN%      9)],
    '::busy'            => [ qw(        0       0       0       %EMPTY%     10)],,
    '::b'=> [ qw( 0 1 1 %OPEN%      11)],
    '::init'            => [ qw(        0       0       0       0            12)],
    '::rec'             => [ qw(        0       0       0       0            13)],
    '::comment'        => [ qw( 1 1 2 %EMPTY%     14)],
    '::default'        => [ qw( 1 1 0 %EMPTY%     0)],
    'nr'=> 15,  # Number of next field to print
},
    },
```

Description: `xls` Spreadsheet name (the name `xls` is a relict from Excel-only-input-times)
`req` tells Mix if this Sheet is mandatory

The Tabular describes the characteristics of a Headers-field:
In the first column you can find the names of the Headers, as you can see these are the Keywords you maybe known from your Spreadsheets
`Inherits` not needed, can be `0`
`Multiple` if set to `1` the Column may be located more then once, internal Mix numerates them begining from `0`, naming convention is: `::<Headersname>:<number>`; if <number> is `0`: `::<Headername>`
`Required` tells Mix if this Header must be located, if not Mix will return an error
`Defaultvalue` specifies a default entry if the column is empty and mandatory
`PrintOrder` internal numeration of Headers
`nr` simply lists the number of Headers defined

To tell Mix to read in some more tabular, open `MixUtils/IO.pm` and search for the `mix_utils_open_input` Function. Here you can find one Array and some calls for every Tabular. In the following example only I^2C operation is listed (normaly this function return 4 array-references, i.e.):

```
sub mix_utils_open_input(@) {
    my @in = @_;

    my $ai2c = [];

    for my $i ( @in) {
```

```
        my $i2c;

@i2c = open_infile($i, $EH{'i2c'}{'xls'}, $EH{'conf'}{'req'});

for my $c ( $i2c) {
    $EH{'i2c'}{'parsed'}++;
    my $norm_i2c = convert_in("i2c", $c);
    select_variant( \@norm_i2c);
    push(@$ai2c, @norm_i2c);
}
    }
    return( $ai2c);
}
```

For every Spreadsheet-File `$i`, a Table named like the value of `$EH'i2c''xls'` is extracted. The last Argument of `mix_open_infile`, the value of`$EH'conf''req'`, specifies if this Table is need or not. The Function `open_infile` return a Array of I^{2}C-Tables found ($i2c[0] is the first Table found in the File `$i`). Next every I^{2}C-Tables Header-Row is matched, emtpy Cells are stiped of and the relevant Variant is selected. Header-Matching is done by `convert_in`, while `select_variant` selects the Variant. All Arrays `norm_i2c` are pushed onto `$ai2c`. The Array returned consists a Two-Dimensional Array. If there was more than one all of them are pushed together.
Next you would have to write a Parser. The Parser will take Input-Information and create hierarchical Units and Connections of it. The following example shows some part MIX's I^{2}C-Parser (for some more practical examples see `MixIOParser.pm`, `MixI2CParser.pm` or `MixParser.pm`):

```
sub parse_i2c_init($) {

    my $r_i2c = shift;

    my $ehr = $EH{'i2c'}{'field'};

    foreach my $i (@$r_i2c) {

        next if ( $i->{'::ign'} =~ m,^\s*(#|\\),); # Skip comments, just in case they sn

        for my $j ( keys %$ehr) {
    next unless ( $j =~ m/^::/ );
    if ( not defined( $i->{$j} )) {
next if( $ehr->{$j}[2] eq 0);
$i->{$j} = $ehr->{$j}[3];
    }
    elsif( $ehr->{$i}[2] && $i->{$j} eq "") {
$i->{$j} = $ehr->{$i}[3]; # Set to DEFAULT Value
    }
}
add_interface($i);
    }
    return 0;
}
```

Ass you can see the parser checks if the cells of the passed Tables are defined, if not it sets a default Value. All fields marked as ignore are dropped.
MIX uses two internal Databases, `$hierdb` and `$conndb`, for Maintaining and Processing HDL-Structures. Adding hierarchical Elements as well as Connections is done by the Functions `add_hier` and `add_conn`. The single Argument passed is a Hash-reference containing new Elements definition. The call to `add_interface` adds a new I^2C-Registerblock, defined in `$i`, to `$hierdb` and `$conndb`. The Code below is only for demonstrating further work, it's not the Real "`MIXI2CPareser.pm`" Code:

```
sub add_interface(%) {

    my $in = shift;

    my %inst;
    my %conn

    # define a new instance
    my $name = $in->{'::interface'};
    $inst->{'::inst'} = $name;
    $inst->{'::entity'} = "PREFIX_". $name;

    # add it to hierdb
    $parent = add_inst( %$inst);

    # define a  new connection
    %conn = ( '::name' => "some_reset",
              '::in' => $name . "/reset",
              '::out' => $in->{'::reset'}, );

    # add it to conndb
    add_conn(%conn);
}
```

This Code simply adds a new Instance named like the content of a "`::interface`" cell. Next the connection of a Signal "`some_reset`" is made from the out-pin specified in the cell called "`::reset`" to the in-pin of the new create instance.

4 The Package: Micronas

Here can be found some additional information about the MIX Libraries which are inseperately part of MIX. The MIX-Libraries offer all MIX capabitlites for Parsing, Checking, Processing, reading and writing Data (HDL, Spreadsheets and internal Structures).

4.1 Library: MixUtils

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. MixUtils is divided into the two Packages MixUtils and MixUtils::IO. The IO Package only contains Functions for Input-and Output Spreadsheet Operations. Supportet formats are Excel, Star-Office and CSV.

Because of the Excel-Sheetwriter Function uses OLE it's only available for Windows. All other Spreadsheet Operations can be done without any external Program or Library.

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.