MIX Register-Master Packages
Developer Reference

Date: $Date: 2005/12/13 13:21:11 $ Version: $Id: MIX_Reg_Development_doc.lyx,v 1.1 2005/12/13 13:21:11 lutscher Exp $

Introduction

This document is a brief overview of the packages that have been added to MIX to process configuration register information. It is intended as an introduction to the developer, who wants to add capabilities to MIX regarding register description input formats, data structures, and output views.

The configuration register information is intrinsically hierarchical. Starting from the chip/device (could also start from an SoC or board-level point of view), it contains a register space that contains domains (mapped to base-addresses of the device) that contains registers (mapped to addresses) that contain fields (mapped to bit-positions within registers). An object has a definition and an instance. The definition/object concept is not supported by the register-master sheet but implemented in the data structure defined here to be open for future applications.

Contents

1 Classes

In Perl, classes are packages. The configuration register information is stored internally in objects. This data structure is independent of the input format or output view. In fact, it is independent of the other MIX packages and can thus be integrated into another script/tools environment if necessary.

1.1 Overview

In general, the data members can be accessed by methods that carry the name of the data member. E.g.
`$ref = $o_reg->domains()`
returns the data member `domain` of the `$o_reg` object. For setting a data member, the method can be called with a hash as parameter, e.g.
`$o_reg->device("VGCH");`

Every class implements a constructor method `new()` and a `display()` method for debugging. The `new()` method takes a hash as parameter that sets data members (keys of the hash), e.g.
`$o_reg->new("device" => "VGCH");`.
The class may also implement a destructor method `DESTROY()`.

1.2 Class `Micronas::Reg`

The `Reg` class contains global information about a configuration register space. Table 1 lists the most important class methods and data members.

Table 1: `Reg` Members

Class method or data member	Description
device	string: symbolic identifier for chip/device

domains	reference to list of hashes of the form
	`(`
	`'domain' => <Micronas::RegDomain object>,`
	`'baseaddr' => <baseaddr>`
	`)`

global	reference to hash containing Class data (static data)
init(%hinput)	initialize object with input; depending on inputformat defined in %hinput, calls appropriate function to fill register space object; note: NOT automatically called by constructor; adds to the object, does not clear it; input: hash, depends on inputformat but has at least key "inputformat"
generate_view($view)	generate a view of the register space: this is the *hook function* to call view-generating-methods in other packages/modules
<other>	see in files ->chapter 4

1.3 Class `Micronas::RegDomain`

The `RegDomain` class contains information related to a register domain. Table 2 lists the most important class methods and data members.

Table 2: `RegDomain` Members

Class method or data member	Description
id	scalar: incremented with every new object
name	string: instance name of register domain
definition	string: definition name of domain
addrmap	reference to list of hashes of the form ('reg' => <Micronas::RegReg object>, 'offset' => <address_offset>) This maps the registers into an address map. Currently only one addressmap is supported; If more are needed, the domain object has to be copied.
regs	reference to list of all `Micronas::RegReg` objects of domain
fields	reference to list of all `Micronas::RegField` objects of domain
find_reg_by_address_first-($offset)	returns the first `Micronas::RegReg` object (or undef) that is mapped to the given address offset
<other>	see in file ->chapter 4

1.4 Class `Micronas::RegReg`

The `RegReg` class contains information related to a hardware register. Table 3 lists the most important class methods and data members.

Table 3: `RegReg` Members

Class method or data member	Description
id	scalar: incremented with every new object
name	string: instance name of register
definition	string: definition name of register
attribs	hash of the form ('<attrib_name>' => <attrib_value>, ...,) Attribute names of registers are e.g. `size`, `usedbits` etc.
fields	reference to list of hashes of the form ('field' => <Micronas::RegField object>, 'pos' => <field_position>) where <field_position> is the bit position of the field in the register.
get_reg_init()	returns the init (reset) value of a register computed from all its fields
<other>	see in file ->chapter 4

1.5 Class `Micronas::RegField`

The RegField class contains information related to a bit-field. Table 4 lists the most important class methods and data members.

Table 4: `RegField` Members

Class method or data member	Description
id	scalar: incremented with every new object
name	string: instance name of bit-field
definition	string: definition name of bit-field
attribs	hash of the form ('<attrib_name>' => <attrib_value>, ...,) Attribute names of registers are e.g. `size`, `usedbits` etc.
reg	parent object
<other>	see in file ->chapter 4

2 View Concept

A *view* is a representation of the register space (or domain) and is the output format of the Register packages. The hook function to implement new views is `Micronas::Reg::generate_view($view)` whereas `Micronas::Reg::global` contains the list of currently supported views.

3 Hooks to MIX

MIX tries to read a register-master sheet and calls `Micronas::Reg::parse_register_master()`. This calls the constructor for a new `Reg` object, calls the `init()` method to parse the register-master sheet and fill the data structure, and calls the `generate_view()` method.

MIX provides an API to create a hierarchy of HDL files. If the view is an HDL view (to create register-shells or similar representations), the Reg object calls this API to create the modules. The API has been wrapped by simple methods in `RegViews.pm`.

4 Files

All Files are located in the <mix_root>/lib/perl/Micronas directory.

Table 5: Files

File	Description
`mix_0.pl`	MIX main program
`Reg.pm`	Main package containing Reg class, input format reader and hooks to MIX main program
`RegDomain.pm`	Package containing RegDomain class
`RegReg.pm`	Package containing RegReg class
`RegField.pm`	Package containing RegField class
`RegViews.pm`	Main package containing methods to generate views; in particular HDL views; extends Reg class
`RegViewE.pm`	Package to generate *e*-code view
`RegViewSTL.pm`	Package to generate STL view
`MixUtils/RegUtils.pm`	Contains non-OO utility methods