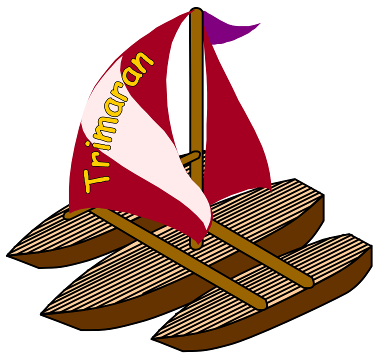


The MDES User Manual

Contents

1 Introduction

The machine description model (MDES) in TRIMARAN allows the user to develop a machine description for the HPL-PD family of processors in a high-level language, which is then translated into a low-level representation for efficient use by the compiler. The high-level language allows the specification of detailed execution constraints in an easy-to-understand, maintainable, and retargetable manner. The low-level representation is designed to allow the compiler to check execution constraints with high space/time efficiency. The purpose of this section of the TRIMARAN manual is to provide the user with an understanding of the role of MDES in TRIMARAN with an emphasis on creating/modifying/using machine descriptions. We shall also describe the compiler's interface to the machine description modules and, briefly, the data-structures internal to the compiler corresponding to the machine descriptions. Where necessary, explanations are illustrated through examples.

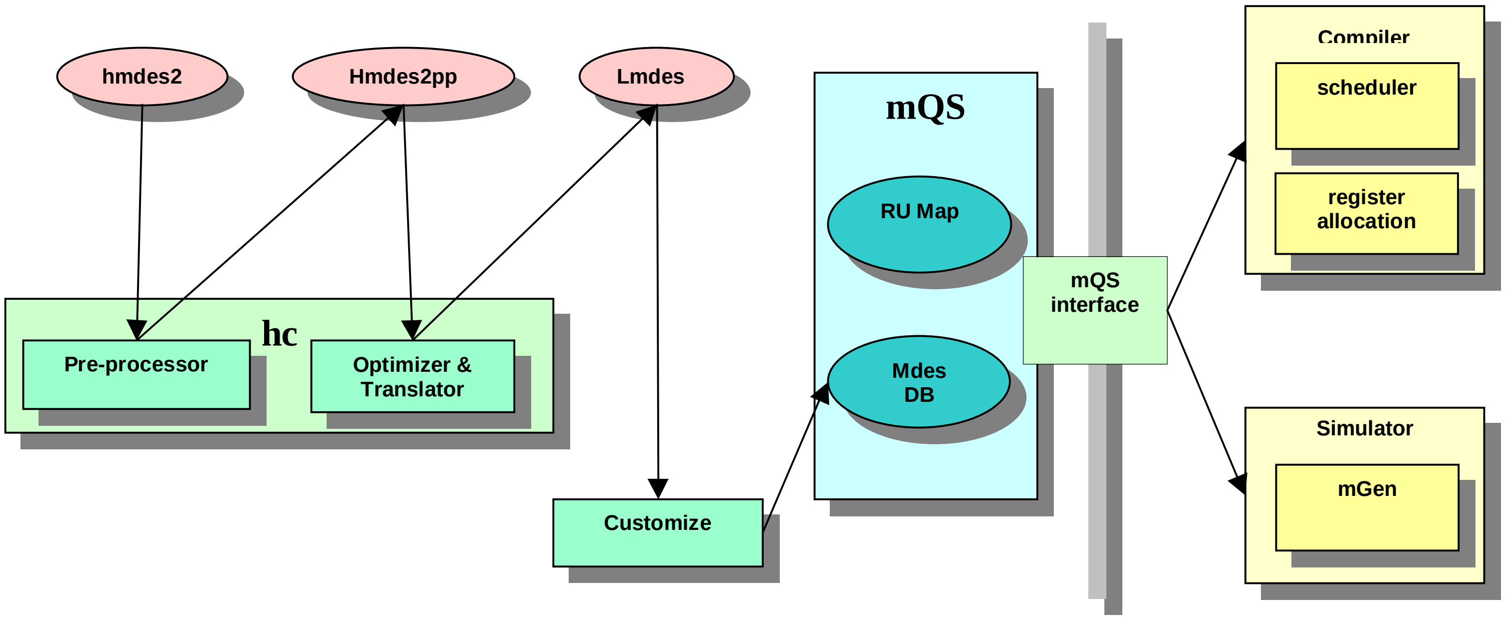


Figure 1: MDES in TRIMARAN

The MDES infrastructure in TRIMARAN is shown in Figure 1. The target (HPL-PD processor) is described in a general relational database description language called MD language [3]. Though the MD language allows for a variety of representations, the HPL-PD machine descriptions are restricted to follow a particular format called HMDES (High-level Machine Description) version 2 [2]. After the macro processing and compilation of the high-level description (*P.hmdes2* in Figure 1), the corresponding low-level specification, expressed in LMDES version 2(P.lmdes2 in Figure 1), is loaded within the compiler using the *customize* module that reads the textual LMDES specification and builds the internal data-structures of the MDES database. The information contained within the machine description database is made available to various modules of the TRIMARAN infrastructure through a query interface called mQS. Subsequent sections of this manual describe these steps in greater detail.

2 MDES Related Files in TRIMARAN

The standard organization of the **mdes module related** files within the TRIMARAN distribution is shown in Figure 2.

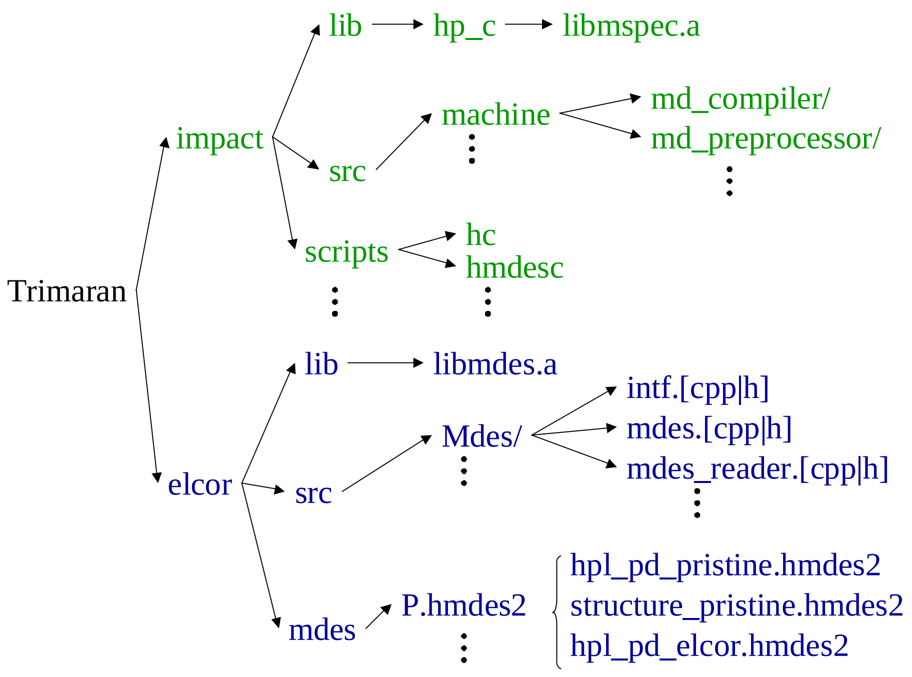


Figure 2: MDES Related Files in TRIMARAN

The Machine Description File

Each instance of HPL-PD processor is described using **four** files in textual format. In Figure 2, these four files corresponding to an example HPL-PD processor (P) are the **top-level** file **P.hmdes2** and the three files #included by it : **structure_pristine.hmdes2**, **hpl_pd_pristine.hmdes2**, and **hpl_pd_elcor.hmdes2**. Currently, the user can vary the machine by modifying parameters within the top-level file (here **P.hmdes2**) and the rest of the compiler modules will track those changes.

Programs that Process Machine Description Files

As mentioned in Section 1, the high-level machine description files are for the convenience of the user. The compiler itself understands and interprets a low-level description. The scripts/programs which are used to convert the high-level descriptions *(*.hmdes2*) to low-level descriptions (**.lmdes2*) are listed below. Please refer to Figure 2 for the location of these files in the TRIMARAN distribution.

1. hc : The main script to convert **.hmdes2* to **.lmdes2*.

2. hmdesc : This script does the same job as hc, however, the generated **.lmdes2* file is customized for the IMPACT front-end's interface to the MDES module.

3. **md_processor, md_compiler, lmdes2_customizer** : These are the main binaries which actually do the pre-processing and compilation of MDES files are called by the **hc, hmdesc** scripts.

<u>Files Specifying the Compiler's Interface to MDES</u>

In the current vision of TRIMARAN, the front-end (IMPACT) and the back-end (ELCOR) source files containing the interface functions to the MDES module are distinct. These files are likely to be unified in future releases of TRIMARAN.

1. inft.* : The mQS interface functions are defined here.
2. **Impact Interface Files** : To be added later.

<u>MDES Module Sources/Libraries</u>

- **Elcor Side MDES Sources**

  1. **mdes.*** The internal data-structures created (MDES database) are defined here.
  2. **mdes_reader.cpp** The functions that load the **.lmdes2* file are defined here.
  3. **libmdes.a** This library includes the object files of the ELCOR side of the MDES source modules.

- **Impact Side MDES Sources**

  1. **TRIMARAN_HOME/impact/src/machine** This directory contains all the sources related to the *.hmdes2 to **.lmdes2* conversion.

  2. *libmspec.a* This library contains the object modules for the impact related MDES files.

3 Structure of the Machine Description File

Here we discuss the structure of the **.hmdes2** files only. As mentioned before, four files describe a particular instance of a HPL-PD processor (say P). The top-level file (**P.hmdes2**) includes the other three (**structure_pristine.hmdes2, hpl_pd_pristine.hmdes2 and hpl_pd_elcor.hmdes2**). The union of the information contained in these files is illustrated in Figure 3. The machine structure is described as a hierarchy of types called *sections*. The top-level section called **Elcor_Operation** defines a list of generic operations which are described in terns of **Operation** and **Elcor_Operation_Flag** sections. Each **Operation** in turn is composed of a list of a list of **Scheduling_Aleternatives**. Each **Scheduling_Alternative** section contains pointers to the actual architectural opcode (**Operation_Format**), the resource usage pattern for that opcode (**Reservation_Table**) and the latency characteristics (**Operation_Latency**). Further details about the structure of the machine description can be found in the references (please see Section 7). Brief notes on the contents of these files are given below.

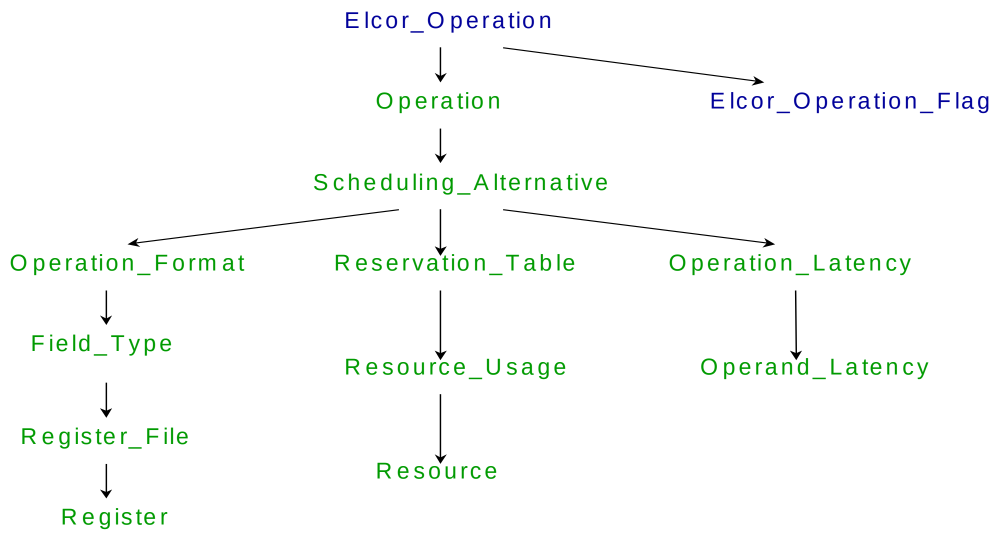


Figure 3: Structure of the Information Contained in a HMDES File

1. **P.hmdes2** : This file list various parameters:

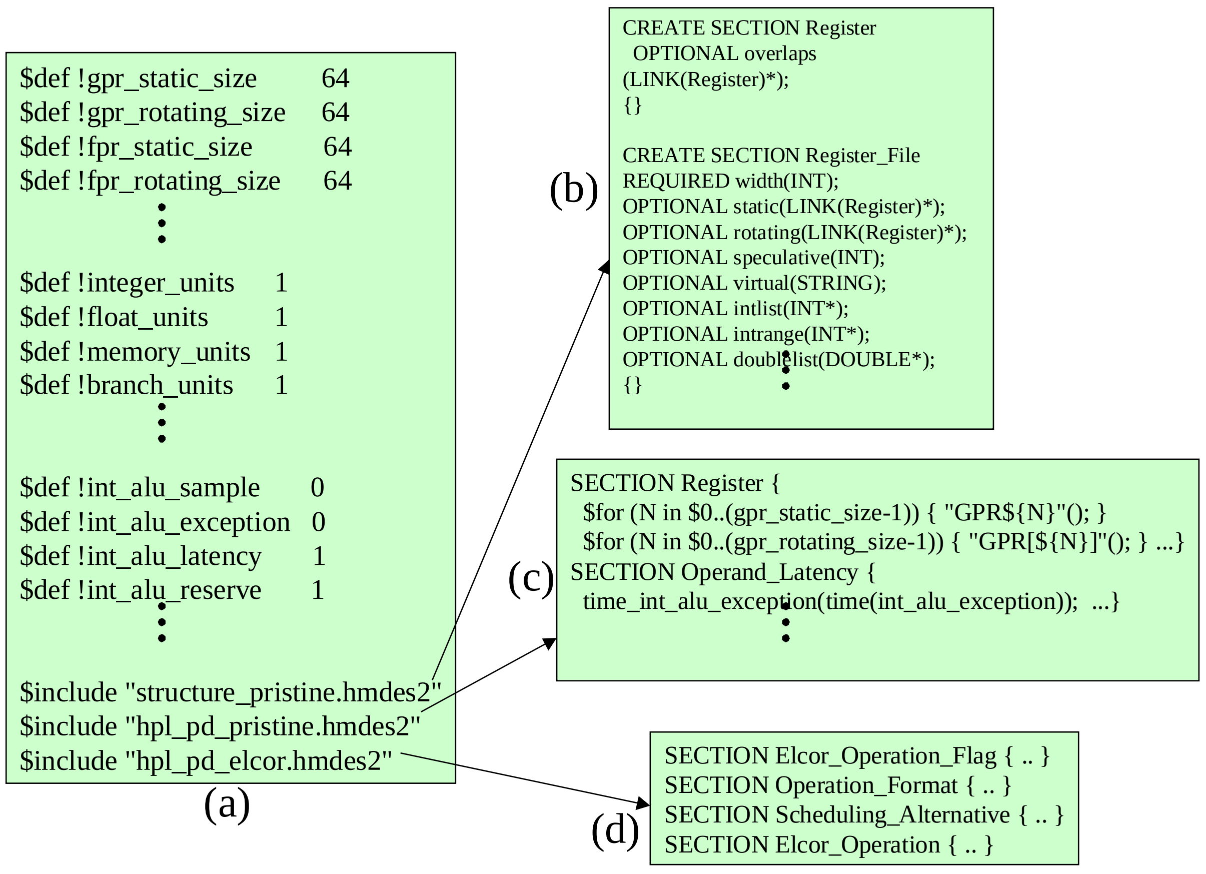
   - Register file sizes.
   - Number of different types of functional units.
   - Latency parameters for different operation groups.

   Portions of a sample file are shown in Figure 4(a).

2. **structure_pristine.hmdes2** : The definitions of all types (sections) (Figure 4(b)) are listed in this file.

3. **hpl_pd_pristine.hmdes2** : This file describes the current version of HPL-PD processor by instantiating most of the sections ("filling the schema") defined in structure_pristine.hmdes2. Portions of a sample file are shown in Figure 4(c).

4. **hpl_pd_elcor.hmdes2** : The sections Elcor_Operation and Elcor_Operation_Flag and a few others which contain ELCOR specific information also and hence described in a separate file (Figure 3(d)).



```
$def !gpr_static_size      64
$def !gpr_rotating_size    64
$def !fpr_static_size      64
$def !fpr_rotating_size    64
              •
              •
              •
$def !integer_units     1
$def !float_units       1
$def !memory_units      1
$def !branch_units      1
              •
              •
              •
$def !int_alu_sample       0
$def !int_alu_exception    0
$def !int_alu_latency      1
$def !int_alu_reserve      1
              •
              •
              •
$include "structure_pristine.hmdes2"
$include "hpl_pd_pristine.hmdes2"
$include "hpl_pd_elcor.hmdes2"
```

```
CREATE SECTION Register
  OPTIONAL overlaps
(LINK(Register)*);
{}

CREATE SECTION Register_File
REQUIRED width(INT);
OPTIONAL static(LINK(Register)*);
OPTIONAL rotating(LINK(Register)*);
OPTIONAL speculative(INT);
OPTIONAL virtual(STRING);
OPTIONAL intlist(INT*);
OPTIONAL intrange(INT*);
OPTIONAL doublelist(DOUBLE*);
{}
                  •
                  •
```

```
SECTION Register {
  $for (N in $0..(gpr_static_size-1)) { "GPR${N}"(); }
  $for (N in $0..(gpr_rotating_size-1)) { "GPR[${N}]"(); } ...}
SECTION Operand_Latency {
  time_int_alu_exception(time(int_alu_exception));  ...}
                  •
                  •
```

```
SECTION Elcor_Operation_Flag { .. }
SECTION Operation_Format { .. }
SECTION Scheduling_Alternative { .. }
SECTION Elcor_Operation { .. }
```

Figure 4: An HMDES File

4 Modifying and Compiling HMDES Files

Modifying machine descriptions : The machine description aspects which can be varied without requiring any change to the compiler or other modules of TRIMARAN are those described in the **top-level** machine description file[1]. Currently, the machine attributes that can be changed are

1. Register file sizes ({gpr|fpr|pr|btr}-{static|rotating|_size).
2. Number of functional units ({integers|float|memory|branch}_units).
3. Operation latency values (for example int_alu_sample, etc.).

Compiling machine descriptions : The only command to use is hc. The format usage for the command is :

hc *top-level-file*.hmdes2

4.1 An Example

A typical usage is shown in Example 1. *Note that the same can be accomplished through the TRIMARAN GUI. Please refer to the GUI documentation for details.*

Example 1 Modifying and Compiling HMDES Files

```
>  cd $MY_MDES_FILES_DIR
>  ls
-rw-r-r-1 ilpdev 12917 Apr 1 18:29 hpl_pd_elcor.hmdes2
-rw-r-r-1 ilpdev 39621 Apr 1 19:14 hpl_pd_pristine.hmdes2
-rw-r-r-1 ilpdev 6101  Mar 2 18:27 structure_pristine.hmdes2
-rw-r-r-1 ilpdev 2556  Mar 31 1997 P.hmdes2

;; Edit some of the allowed parameters (listed above). For example
;; one could change the number of integer_units,
   memory_access_latency etc.
>  emacs P.hmdes2
;; compile the *P.hmdes2 file
>  hc P.hmdes2
>  ls

-rw-r-r-1 ilpdev 12917 Apr  1 18:29 hpl_pd_elcor.hmdes2
-rw-r-r-1 ilpdev 39621 Apr  1 19:14 hpl_pd_pristine.hmdes2
-rw-r-r-1 ilpdev 6101  Mar  2 18:27 structure_pristine.hmdes2
-rw-r-r-1 ilpdev 2630  Apr 1 20:24 1997 P.hmdes2
-rw-r-r-1 ilpdev 214411 Apr 1 20:27 1997 P.lmdes2

;; Notice that the P.lmdes2 file has been created as a result of
;; compiling P.hmdes2
;; To use the new machine description file update the lMDES_file_name
;; parameter. This parameter is defined in the IO_DEFAULTS parameter
;;file

>  emacs $ELCOR_HOME/elcor/parms/IO_DEFAULTS

;; run a compilation
```

[1] A section of this top-level hmdes2 file is shown in Figure3(a)

5 External Interface to MDES

5.1 mQS : mdes Query System

Information contained within the MDES database should be accessed through a query interface called the mQS. The files pertinent to this interface are mentioned in Section2. The mQS interface is composed of four sets of queries:

1. Opcode, Operand parameter queries.
2. Register parameter queries.
3. Resource Usage (RU) manages queries.
4. Resource Minimum Schedule Length (RMSL) manager queries.

These queries along with brief descriptions are listed in Table 1. Please refer to the source files in the distribution for exact declarations.

5.2 Using mQS : Example 1

Calculate Res MII Using RMSL Queries : Given an if-converted, back-substituted, load-store eliminated graph of intermediate code (ELCOR IR), the given function in Example 2 Computes an integer value representing the maximum initiation interval (II) possible, given the resource bounds. It proceeds by calculating the maximum schedule length of the operations performed on each class of resource. This estimate is used in *modulo-scheduling* module of the Elcor ILP-optimizer[2]. This portion of the source code is from $*ELCOR_HOME/src/ModuloScheduler.cpp.*

Example 2 Calculate ResMII Using RMSL Queries

```
int ModuloScheduler :: calculate_ResMII(){
        char iopat [MAXIOPATLEN];
        Real_op_filter  real_op_filter;
        RMSL_alloc();
        for (Region_all_ops opi(kernel, &real_op_filter); opi != 0; oopi++) {
          Op*op = *opi;
          if (el_opcode_to_string_map.is_bound(oop->opcode())) {
                  op->build_io_pat(iopat);
                  RMSL_nextop(el_opcode_to_string_map.value(op->opcode()), iopat);

          }
        }
        int resmii = RMSL_value();
        RMSL_dealloc();
        return resmii;
}
```

[2] For details about the algorithm, please see [6]

5.3 Using mQS : Example 2

Trimaran Simulator : The simulator[3] obtains the target (HPL-PD) processor register-file sizes using a sequence of calls as shown in Example 3. This code is obtained from *simu_el_processor.cpp* file of the simulator sources.

Example 3 mQS Queries in the Simulator

```
char *file;
file =  regfile_to_char(GPR);
regs.gpr_stat_size = MDES_reg_static_size(file);
file = regfile_to_char(FPR);
regs.fpr_stat_size = MDES_reg_static_size(file);
```

[3] Information regarding the HPL-PD simulator is given in a separate chapter of the TRIMARAN documentation

Table 1 mQS : The MDES Query System

1. Opcode and Operand Parameters

MDES_src_num(*opcode*)	; return number of input operands
MDES_dest_num(*opcode*)	; return number of output operands
MDES_is_predicated(*opcode*)	;
MDES_has_speculative_version(*opcode*)	;
MDES_flow_time_io(*opcode,iodesc,port*)	; These functions are used to obtain the latency parameters
MDES_anti_time_io(*opcode,iodesc,port*)	; of any given operand. The exception time applies to entire
MDES_branch_latency(*opcode*)	; operation rather than to a particular operand. For an opcode
MDES_priority(*opcode*)	; qualified opn, the numbers are exact.

2. Register Parameters

MDES_reg_names(list&regnames)	; number of distinct compiler register sets supported
MDES_reg_overlaps(*regname1, regname2*)	; there is a structural overlap between the files?
MDES_reg_static_size(*regfile*)	; number of static registers in the register file
MDES_regrotating_size(*regfile*)	; number of rotating registers in the register file
MDES_reg_width(*regfile*)	; register width in bits
MDES_supports_rot_reg(*regfile*)	; register file supports rotating registers?
MDES_reg_has_speculative_bit(*regfile*)	; register file supports speculative registers?
MDES_reg_is_allocatable(*regfile*)	; literal registers are not allocatable

3. Resource Usage Manager Queries

RU_alloc_map(*max_length*)	; These functions alloc, delete, print and init the interal resource
RU_delete_map()	; usage map as requested by scheduler. Initialization specifies
RU_print_map(*stream*)	; whether the map is to be used for modul-scheduling or not
RU_init_map(*is_modulo, ii*)	; ii is the initiation interval (integer)
RU_get_next_nonconfl_alt(*opcode,iodesc,time*)	; Cycle through next available non-conflicting alternative
RU_place_alt(*opcode,time*)	; commit the chosen alternative
RU_remove_alt(*opcode, time*)	; unschedule the placed instruction
RU_ge_conflicting_ops()	; If the scheduler decides to unschedule, this fn gives all the
	; scheduled opns that conflict with any alternative of the
	; current access equivalent opn set if scheduled at the current cycle

4. RMSL Queries

RMSL_alloc()	; These functions allocate, deallocate and initialize
RMSL_dealloc()	; the internal resource counters for computing RMSL
RMSL_init()	;
RMSL_nextop(*opcode, iodesc*)	; Accumulates the resource counts for the best alternative
RMSL_value()	; returns the currently accumulated RMSL

6 Limitations/Bugs

1. None.

7. Further Reading

Some references for further details:

- Basic philosophy behind the design of MDES was proposed in the MICRO-29 paper [4].
- MD language details and the external file format of MDES files can be found in Gyllenhaal's Master's Thesis[3]. A complete specification of the HMDES format for representing HPL-PD and other multi-issue processors can be found in [2].
- The structure of the machine description is motivated by a machine description driven compilation methodology for VLIW processors advocated in [5, 7].
- The TRIMARAN tutorial [1] contains two thirty minutes slide presentations describing the MDES infrastructure as implemented in TRIMARAN.

8. References

[1] ReaCT-ILP Group. Trimaran tutorial. December 1997.

[2] J.C. Gyllenhaal, W.W. Hwu, and B.R. Rau. Hmdes version 2.0 specification. Technical Report IMPACT-96-3, Urbana Champagne, March 1996.

[3] John c. Gyllenhaal. Machine description… Master's thesis, University of Illinois, Urbana Champagnem, December 1993.

[4] John C. Gyllenhaal, Wen Mei W. Hwu, and B. Ramakrishna Rau. Optimization of machine descriptions for efficient use. In *Proceedings of the 29th Annual International Symposium on Microarchtecture*, pages 349-358, Paris, France, December 2-4, 1996. IEEE Computer Society TC-MICRO and ACM SIGMICRO.

[5] S. Aditya, V. Kathail and B. R. Rau. Elcor's Machine Description System: Version 3.0. HPL Technical Report HPL-98-128. Hewlett-Packard Laboratories, July 1998.

[6] B. Ramakrishna Rau. Iterative modulo scheduling: An algorithm for software pipelining loops. In *Proceedings of the 27th Annual International Symposium on Microarchitecture*, pages 63-74, San Jose, California, November 30-December 2, 1994. ACM SIGMICRO and IEEE Computer Society TC-MICRO.

[7] B. R. Rau, V. Kathail and S. Aditya. Machine-Description Driven Compilers for VLIW Processors. HPL Technical Report HPL-98-40. Hewlett-Packard Laboratories, March 1998.