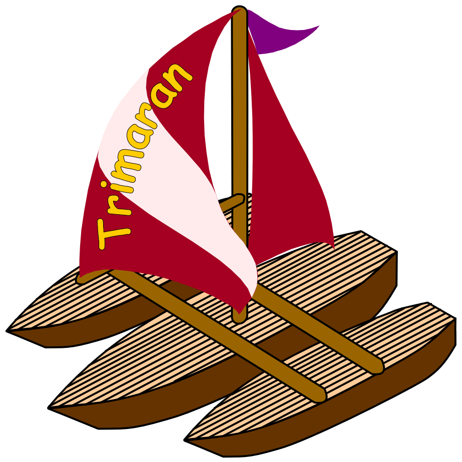


mdes Tutorial

Part II

mdes Query System (mQS) Interface

- Register file parameters

```
void MDES_reg_names(List<char*>& regnames);   // list all register files

int MDES_reg_static_size(char* regname);
int MDES_reg_rotating_size(char* regname);
int MDES_reg_width(char* regname);
Bool MDES_supports_rot_reg(char* regname);
Bool MDES_reg_has_speculative_bit(char* regname);
Bool MDES_reg_is_allocatable(char* regname); // Literal files are not
```

- Operation parameters

```
int MDES_src_num(char* opcode) ;          // excludes predicate input
int MDES_dest_num(char* opcode) ;
Bool MDES_predicated(char* opcode) ;
Bool MDES_has_speculative_version(char* opcode);
int MDES_priority(char* opcode);
```

mdes Query System (mQS) Interface (contd.)

- Latency parameters

```
void MDES_init_op_io(char* opcode, char* iodesc);
int MDES_flow_time_io(IO_Portkind portkind, int portnum); // T_r, T_s
int MDES_anti_time_io(IO_Portkind portkind, int portnum); // T_x, T_a
void MDES_branch_latency(char* opcode);                   // branch T_r
```

- Miscellaneous

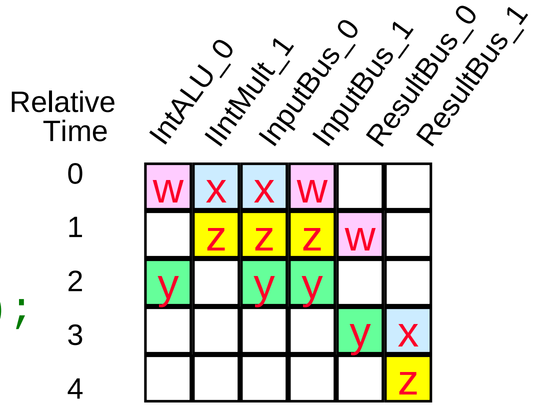
```
class MDES {
public:
  MDES(char* lmdes2_filename);         // read in mdes
  …
}

void set_current_MDES(MDES* mdes);     // install mdes
```

Resource Manager Functions

- Resource table manipulation

```
void RU_alloc_map(int maxlength);
void RU_delete_map(void);
void RU_print_map(FILE *mout);
void RU_init_map(Bool modulo, int length);
```



Relative Time	IntALU_0	IIntMult_1	InputBus_0	InputBus_1	ResultBus_0	ResultBus_1
0	w	x	x	w		
1		z	z	z	w	
2	y		y	y		
3					y	x
4						z

- Operation scheduling

```
void RU_init_iterator                  // initialize scheduling request
  (char* opcode, void* op, char* iodesc, int time);
Bool RU_get_next_nonconfl_alt
  (char** opcode, int* priority);  // return alternative, if successful
void RU_place(void);                   // commit alternative

void RU_get_conflicting_ops(Hash_set<void*>& ops);
void RU_remove(void* op, char* iodesc, int time);
void *RU_at(int time, int res_index);
```

Resource Minimum Schedule Length Estimation

- Resource table manipulation

```
void RMSL_alloc(void);
void RMSL_dealloc(void);
void RMSL_init(void);
```

Relative Time	IntALU_0	IIntMult_1	InputBus_0	InputBus_1	ResultBus_0	ResultBus_1
0	w	x	x	w	w	x
1	y	z	z	z	y	z
2			y	y		
3						
4						

- Resource lower bound estimation

```
void RMSL_nextop(char* opcode, char* iodesc);
int RMSL_value(void);     // current lower bound
```

Hmdes2: High-level Machine Description Language

- Hmdes2 is a schema expressed in DBL
- DBL: an incremental relational database description language

Section$_1$	field$_1$	field$_2$	…
record$_1$			
record$_2$			
…			

Section$_2$	field$_1$	field$_2$	…
record$_1$			
record$_2$			
…			

…

- Text Macroprocessor
  - File inclusion
  - Macro-variables, shell environment variables
  - Recursive variable replacement (textual)
  - Fixed/floating numeric expression evaluation
  - If-then-else
  - For-loop (counted and list ranges)

Hmdes2 schema for HPL-PD/Elcor

	Arch. fields	Comp. fields
Arch. records	HPL-PD pristine records	Additional Elcor properties
Comp. records	Elcor records with arch. fields	Additional Elcor records

$$\begin{tikzcd}
 & \text{Elcor\_Operation} \arrow[d] \arrow[rd] & \\
 & \text{Operation} \arrow[d] & \text{Elcor\_Operation\_Flag} \\
 & \text{Scheduling\_Alternative} \arrow[ld] \arrow[d] \arrow[rd] & \\
\text{Operation\_Format} \arrow[d] & \text{Reservation\_Table} \arrow[d] & \text{Operation\_Latency} \arrow[d] \\
\text{Field\_Type} \arrow[d] & \text{Resource\_Usage} \arrow[d] & \text{Operand\_Latency} \\
\text{Register\_File} \arrow[d] & \text{Resource} & \\
\text{Register} & &
\end{tikzcd}$$

Register

Schema

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

Example

```
SECTION Register {
  GPR0(); GPR1(); … GPR63();
  'GPR[0]'(); 'GPR[1]'(); … 'GPR[63]'();
  ...
  CR0(overlaps(PR0 … PR31));
  ...
}
```

Register File

Schema

```
CREATE SECTION Register_File
  REQUIRED width(INT);
  OPTIONAL virtual(STRING*);        // generic register types supported
  OPTIONAL static(LINK(Register)*);
  OPTIONAL rotating(LINK(Register)*);
  OPTIONAL speculative(INT);        // 0=non-spec. 1=spec.
  OPTIONAL intlist(INT*);           // literal values
  OPTIONAL intrange(INT*);
  OPTIONAL doublelist(DOUBLE*);
{}
```

Example

```
SECTION Register_File {
  RF_i(width(32) virtual(i) speculative(1)
      static(GPR0 … GPR63) rotating('GPR[0]' … 'GPR[63]'));
  LF_s(width(6) virtual(l) intrange(-32 31));
  ...
  LF_l(width(32) virtual(l));       // generic literal file (for Elcor)
  RF_u(width(0) virtual(u));        // generic bit-bucket (for Elcor)
}
```

Field Type (I/O set)

Schema

```
CREATE SECTION Field_Type
  OPTIONAL compatible_with(LINK(Field_Type)*);
  OPTIONAL regfile(LINK(Register_File));
{}
```

Example

```
SECTION Field_Type {
  FT_i(regfile(RF_i));
  FT_c(regfile(RF_c));
  FT_l(regfile(LF_l));
  ...
  FT_icl(compatible_with(FT_i FT_c FT_l));
  ...
}
```

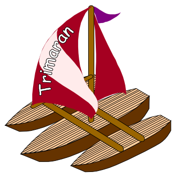
Operation Format (I/O descriptor)

Schema

```
CREATE SECTION Operation_Format
  OPTIONAL pred(LINK(Field_Type)*);
  OPTIONAL src(LINK(Field_Type)*);
  OPTIONAL dest(LINK(Field_Type)*);
{}
```

Example

```
SECTION Operation_Format {
  OF_intarith2(pred(FT_p) src(FT_icl FT_icl)
               dest(FT_ic));
  OF_intcmppred(pred(FT_p) src(FT_il FT_il)
                dest(FT_p FT_p));
  ...
}
```



Resource

Schema

```
CREATE SECTION Resource
{}
SECTION Resource
  OPTIONAL display(INT);  // display resource during printing
{}
```

Example

```
SECTION Resource {
  i0(); i1(); f0(); m0(); b0();
}
SECTION Resource {        // incremental field addition
  i0(display(1)); i1(display(1));
  f0(display(1)); m0(display(1));
  b0(display(1));
}
```

Resource Usage

Schema

```
CREATE SECTION Resource_Usage
  REQUIRED use(LINK(Resource));
  REQUIRED time(INT INT*);     // use same resource at
{}                             // various times
```

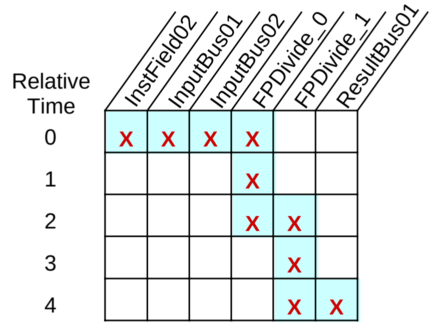
Example

```
SECTION Resource_Usage {
  RU_i0(use(i0) time(0));
  RU_i1(use(i1) time(0));
  ...
}
```

Reservation Table

Schema

```
CREATE SECTION Reservation_Table
  REQUIRED use(LINK(Resource_Usage)*);
{}
```



Relative Time	InstField02	InputBus01	InputBus02	FPDivide_0	FPDivide_1	ResultBus01
0	x	x	x	x		
1				x		
2				x	x	
3					x	
4					x	x

Example

```
SECTION Reservation_Table {
  RT_null(use());         // null reservation for dummy ops
  RT_i0(use(RU_i0));
  RT_i1(use(RU_i1));
  ...
}
```

Operand Latency

Schema

```
CREATE SECTION Operand_Latency
  REQUIRED time(INT*);          // use same operand at
{}                              // various times
```

Example

```
SECTION Operand_Latency {
  time_null(time(0));           // null reservation for dummy ops
  time_int_alu_sample(time(0));
  time_int_alu_exception(time(0));
  time_int_alu_latency(time(1));
  time_int_alu_reserve(time(0));
  ...
}
```

Operation Latency

Schema

```
CREATE SECTION Operation_Latency
  OPTIONAL dest(LINK(Operand_Latency)*);        // $T_r$
  OPTIONAL src(LINK(Operand_Latency)*);         // $T_s$
  OPTIONAL pred(LINK(Operand_Latency)*);        // $T_s$
  OPTIONAL exc(LINK(Operand_Latency));          // $T_x$ (one for all inputs)
  OPTIONAL rsv(LINK(Operand_Latency)*);         // $T_a$
  OPTIONAL sync_dest(LINK(Operand_Latency)*);   // $T_r$ (for sync ports)
  OPTIONAL sync_src(LINK(Operand_Latency)*);    // $T_s$ (for sync ports)
{}
```

Example

```
SECTION Operation_Latency {
  OL_int(dest(time_int_alu_latency … time_int_alu_latency)
         src(time_int_alu_sample … time_int_alu_sample)
         pred(time_int_alu_sample)
         exc(time_int_alu_exception)
         rsv(time_int_alu_reserve … time_int_alu_reserve)
         sync_dest(time_int_alu_sample time_int_alu_sample)
         sync_src(time_int_alu_sample time_int_alu_sample));
}
```

Scheduling Alternative

Schema

```
CREATE SECTION Scheduling_Alternative
  REQUIRED format(LINK(Operation_Format)*);
  REQUIRED resv(LINK(Reservation_Table));
  REQUIRED latency(LINK(Operation_Latency));
{}
```

Example

```
SECTION Scheduling_Alternative {
  SA_intarith2_int_i0(format(OF_intarith2)
     resv(RT_i0) latency(OL_int));    // integer unit 0
  SA_intarith2_int_i1(format(OF_intarith2)
     resv(RT_i1) latency(OL_int));    // integer unit 1
  ...
}
```

Operation

Schema

```
CREATE SECTION Operation
  REQUIRED alt(LINK(Scheduling_Alternative)*);
{}
```

Example

```
SECTION Operation {
  'addw.0'(alt(SA_intarith2_int_i0));
  'addw.1'(alt(SA_intarith2_int_i1));
  ...
}
SECTION Operation {                 // Elcor dummy ops
  'dummy_branch.0'(alt(SA_dummy_null_null));
  'control_merge.0'(alt(SA_dummy_null_null));
  ...
}
```

Elcor Operation Flag

Schema

```
CREATE SECTION Elcor_Operation_Flag
{}
```

Example

```
SECTION Elcor_Operation_flag {
  NULL           ();
  INT            ();
  FLOAT          ();
  MEMORY         ();
  BRANCH         ();

  NOSPEC         ();
  SPEC           ();
  ...
}
```

Elcor Operation

Schema

```
CREATE SECTION Elcor_Operation
  OPTIONAL op(LINK(Operation)*);
  OPTIONAL flags(LINK(Elcor_Operation_Flag)*);
{}
```

Example

```
SECTION Elcor_Operation {
  ADD_W(op('addw.0' 'addw.1') flags(INT SPEC));
  ...
  dummy_branch(op('dummy_branch.0')
               flags(NULL NOSPEC));
  control_merge(op('control_merge.0')
                flags(NULL NOSPEC));
  ...
}
```