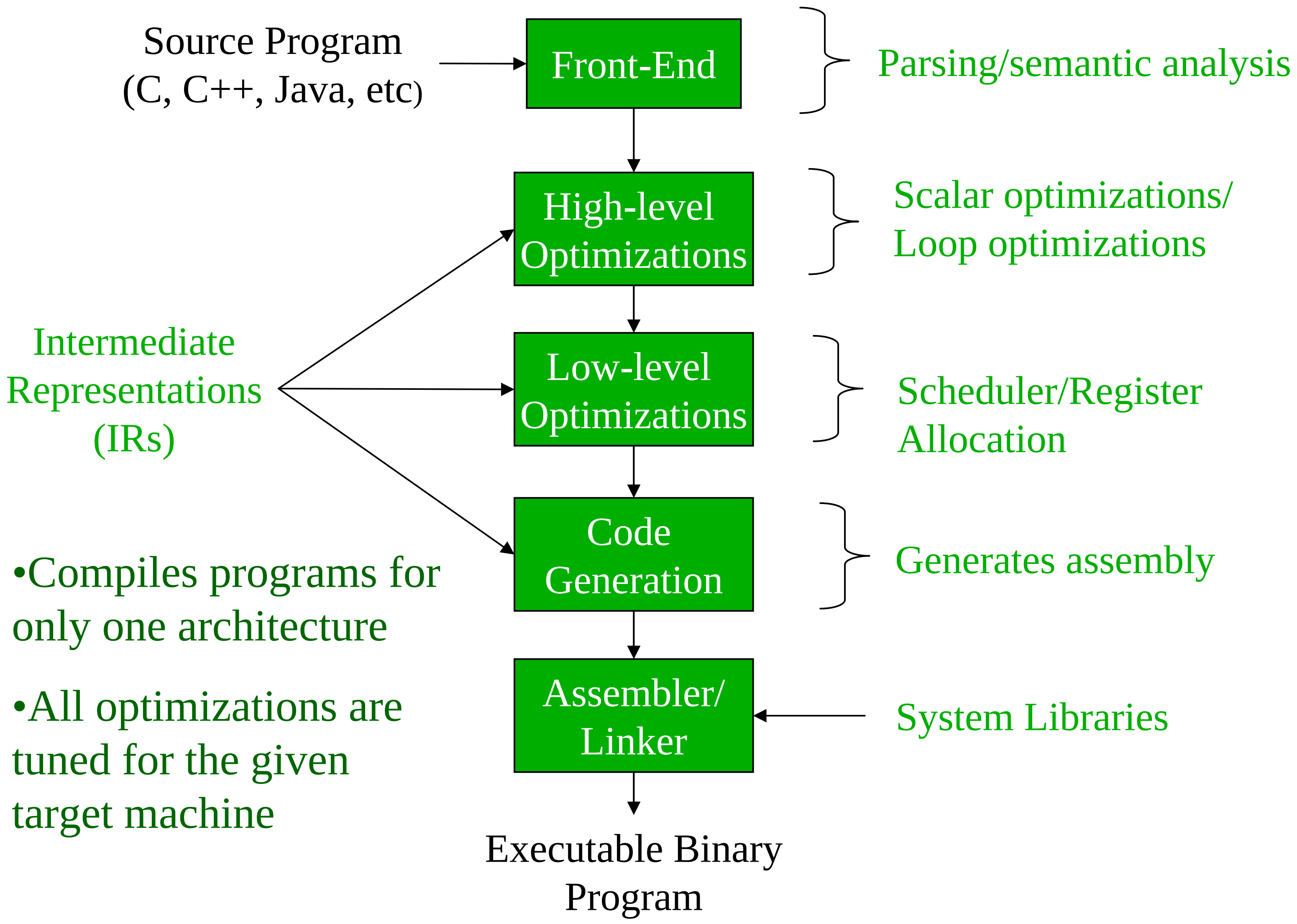
Topic 3
Introduction to Trimaran(w/ Lab)

What is Trimaran?

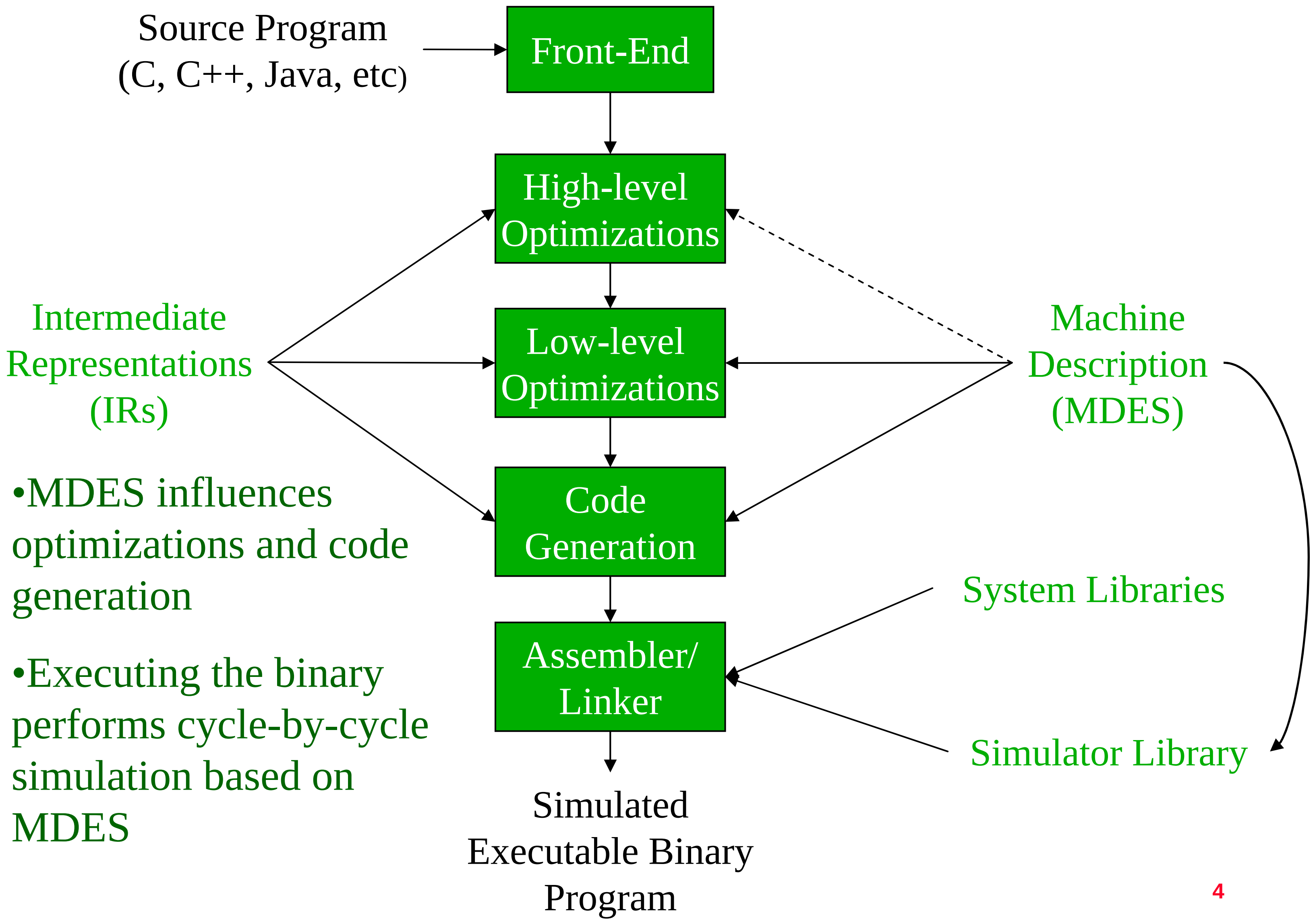
- A compilation system
  - A full-blown C compiler for the HPL-PD ISA
  - A cycle-by-cycle machine simulator + cache simulator
  - Analysis tools
- Uses HPL-PD a parameterized VLIW ISA.
  - We will discuss this in detail in the next lecture
- Compiles for target architectures specified by a machine description language.
  - Can compile optimized code for a variety of VLIW and Superscalar architectures

Compiling a Program



- Compiles programs for only one architecture
- All optimizations are tuned for the given target machine

A Retargetable Compiler + Simulator

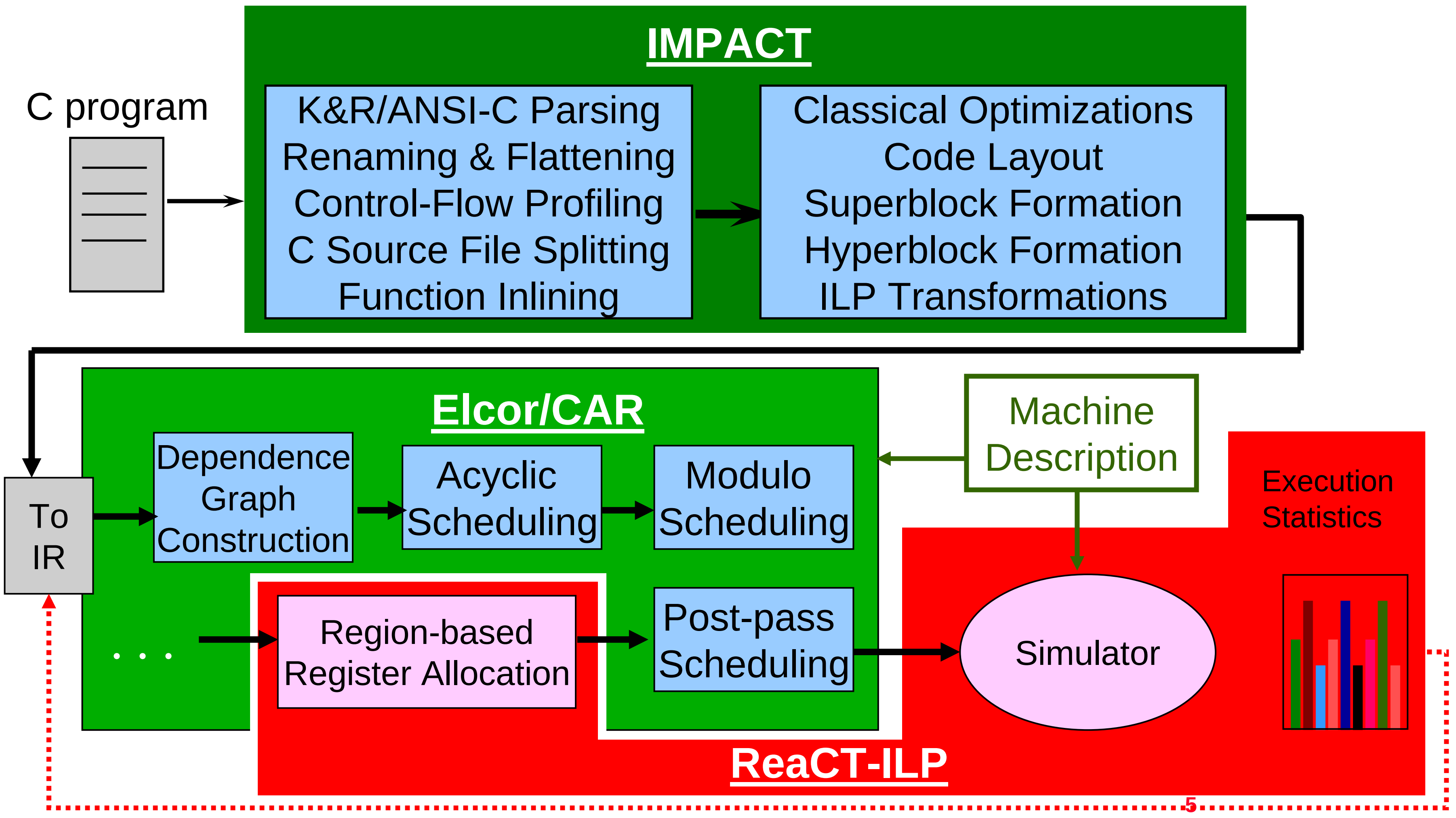


•MDES influences optimizations and code generation

•Executing the binary performs cycle-by-cycle simulation based on MDES

Trimaran System Organization

- A compiler researcher's view of the system:



Trimaran Goal

- To provide a vehicle for implementation and experimentation for state of the art research in compiler techniques for instruction-level parallel architectures.
  - Currently, the infrastructure is oriented towards Explicitly Parallel Instruction Computing (EPIC) architectures.
    - **But can also support compiler research for Superscalar architectures.**
  - Primarily, “back-end” compiler research
    - **instruction scheduling, register allocation, and machine dependent optimizations.**

Terms and Definitions

- ILP (Instruction-Level Parallelism)
  - more than one operation issued per clock cycle within a single CPU
- EPIC (Explicitly Parallel Instruction Computing)
  - ILP under compiler control
    - **A single instruction may contain many operations**
    - **Compiler determines operation dependences and specifies which operations may execute concurrently**

Trimaran Components

- Trimaran is composed of the following:
  - A machine description language, HMDES, for describing ILP architectures.
  - A parameterized ILP Architecture called HPL-PD
    - **Current instantiation in the infrastructure is as a EPIC architecture**
  - A compiler front-end for C, performing parsing, type checking, and a large suite of high-level (i.e. machine independent) optimizations.
    - **This is the IMPACT module (IMPACT group, University of Illinois)**

Trimaran Components (cont)

- A compiler back-end, parameterized by a machine description, performing instruction scheduling, register allocation, and machine-dependent optimizations.
  - **Each stage of the back-end may easily be replaced or modified by a compiler researcher.**
  - **Primarily implemented as part of the ELCOR effort by the CAR Group at HP Labs.**
  - **Augmented with a scalar register allocator from the ReaCT-ILP group at NYU.**

Trimaran Components (cont)

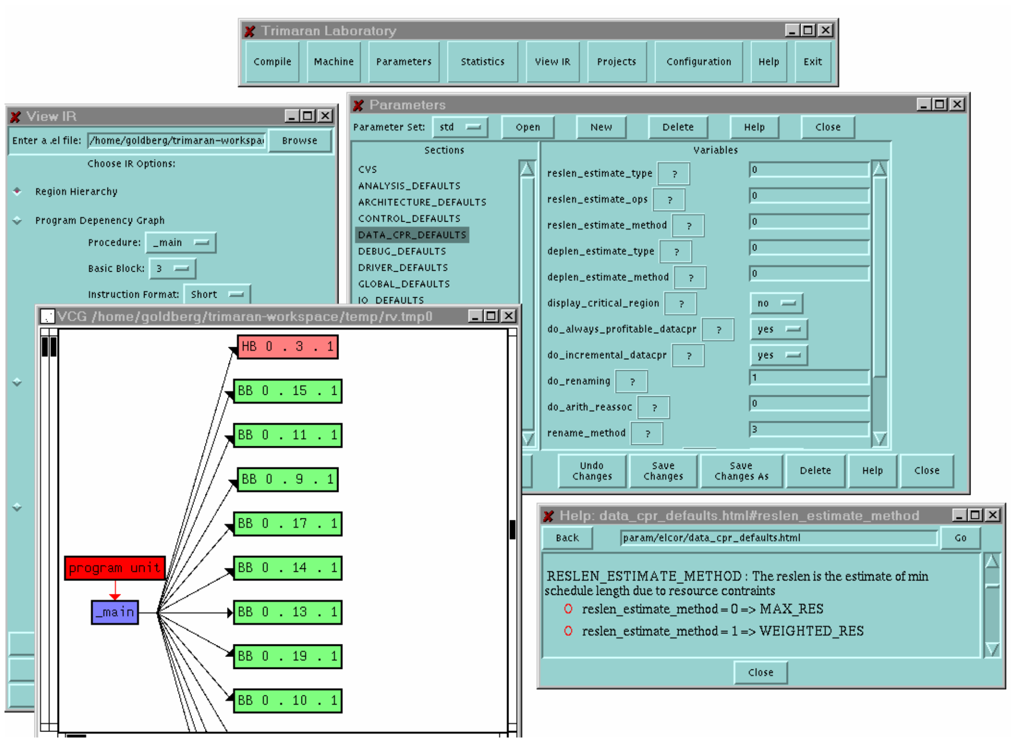
- An extensible IR (intermediate program representation)
  - Has both an internal and textual representation, with conversion routines between the two. The textual language is called Rebel.
  - Supports modern compiler techniques by representing control flow, data and control dependence, and many other attributes.
  - Easy to use in its internal representation (clear C++ object hierarchy) and textual representation (human-readable)

Trimaran Components (cont)

- A cycle-level simulator of the HPL-PD architecture which is configurable by a machine description and provides run-time information on execution time, branch frequencies, and resource utilization.
  - **This information can be used for profile-driven optimizations, as well as to provide validation of new optimizations.**
  - **The HPL-PD simulator was implemented by the ReaCT_ILP group at NYU.**

Trimaran Components (cont)

– An Integrated graphical user interface (GUI) for configuring and running the Trimaran system.



Enter a .el file: /home/goldberg/trimaran-workspa
Procedure: _main
Basic Block: 3
Instruction Format: Short

Parameter Set: std

CVS
ANALYSIS_DEFAULTS
ARCHITECTURE_DEFAULTS
CONTROL_DEFAULTS
DATA_CPR_DEFAULTS
DEBUG_DEFAULTS
DRIVER_DEFAULTS
GLOBAL_DEFAULTS
IO_DEFAULTS

reslen_estimate_type	0
reslen_estimate_ops	0
reslen_estimate_method	0
deplen_estimate_type	0
deplen_estimate_method	0
display_critical_region	no
do_always_profitable_datacpr	yes
do_incremental_datacpr	yes
do_renaming	1
do_arith_reassoc	0
rename_method	3

param/elcor/data_cpr_defaults.html

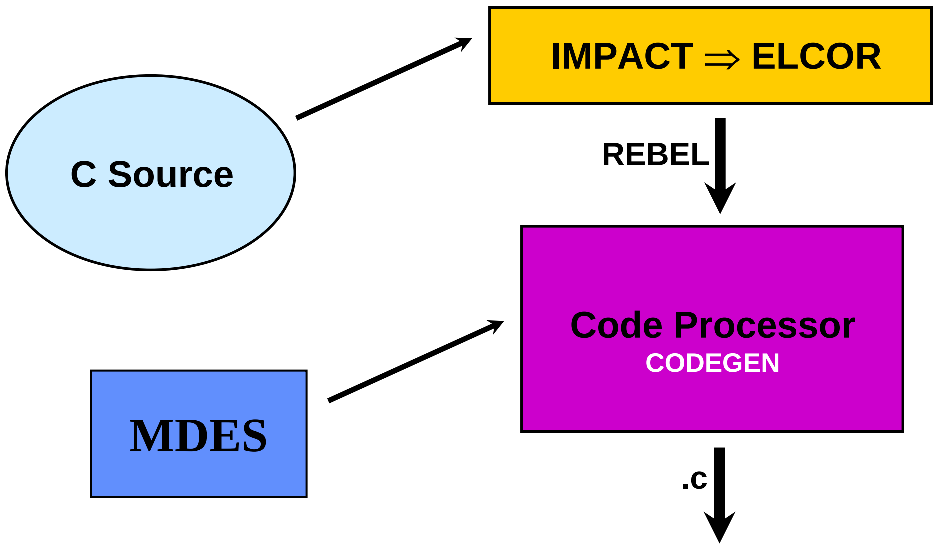
RESLEN_ESTIMATE_METHOD : The reslen is the estimate of min schedule length due to resource contraints
- reslen_estimate_method = 0 => MAX_RES
- reslen_estimate_method = 1 => WEIGHTED_RES

Introduction to Simulator Support in Trimaran

The Framework

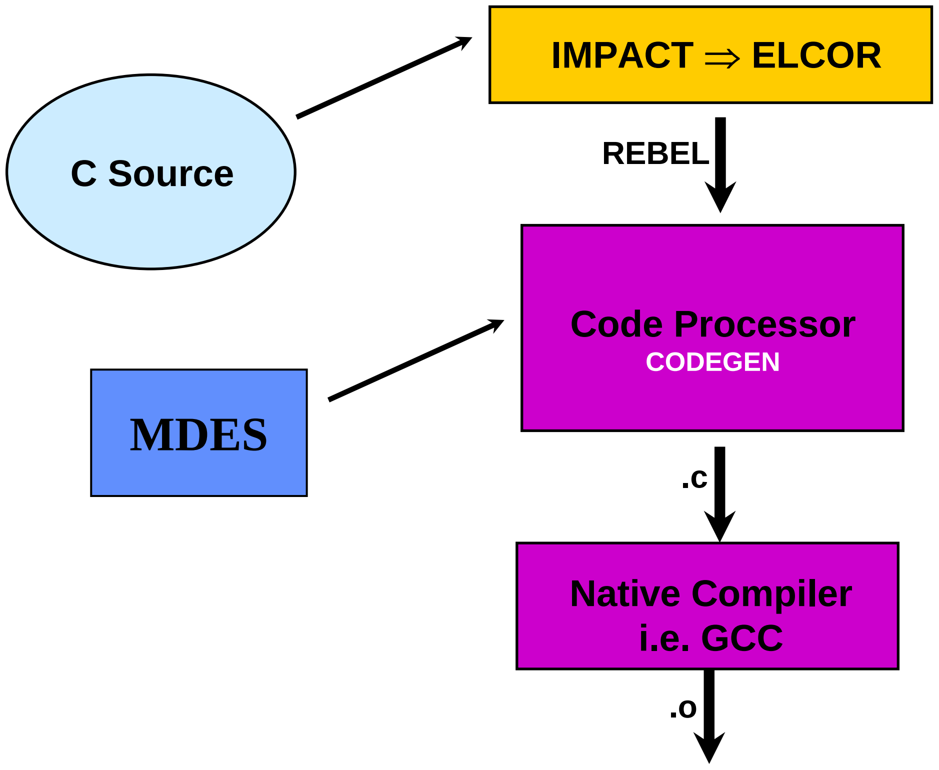
- The goals of the HPL-PD simulation framework are
  - Emulate the execution of the generated REBEL code on a virtual HPL-PD processor
  - Have the ability to adapt to changes in the machine description
  - Generate *accurate* run-time information
    - **Execution clock cycles**
    - **Dynamic control flow and call trace**
    - **Address trace**
    - **Average number of operations executed per cycle**

Simulator: Overview



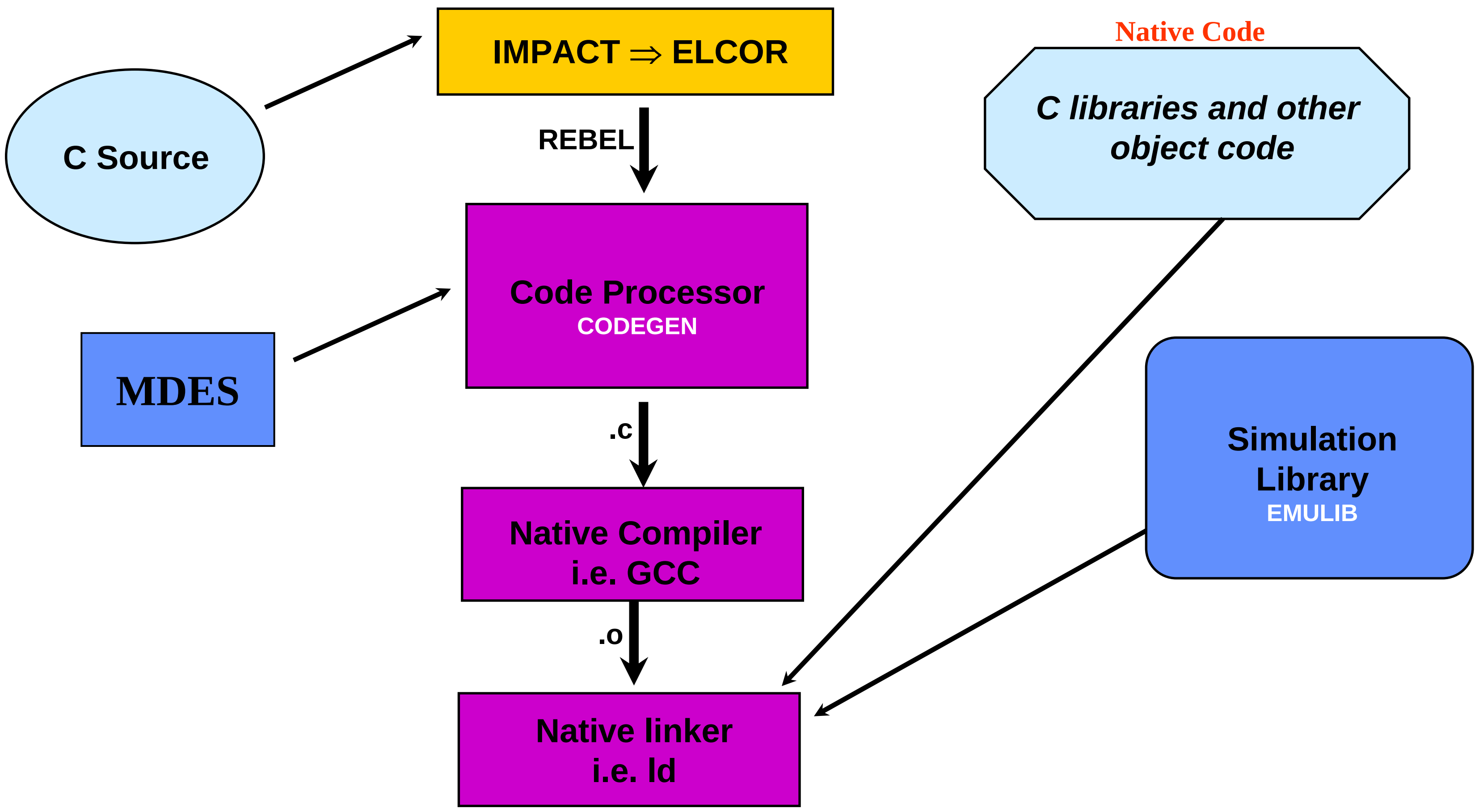
- The code processor translates the REBEL input to a *C* equivalent
- The output of CODEGEN is a collection of *pseudo-executable* low-level *C* files

Simulator: Overview



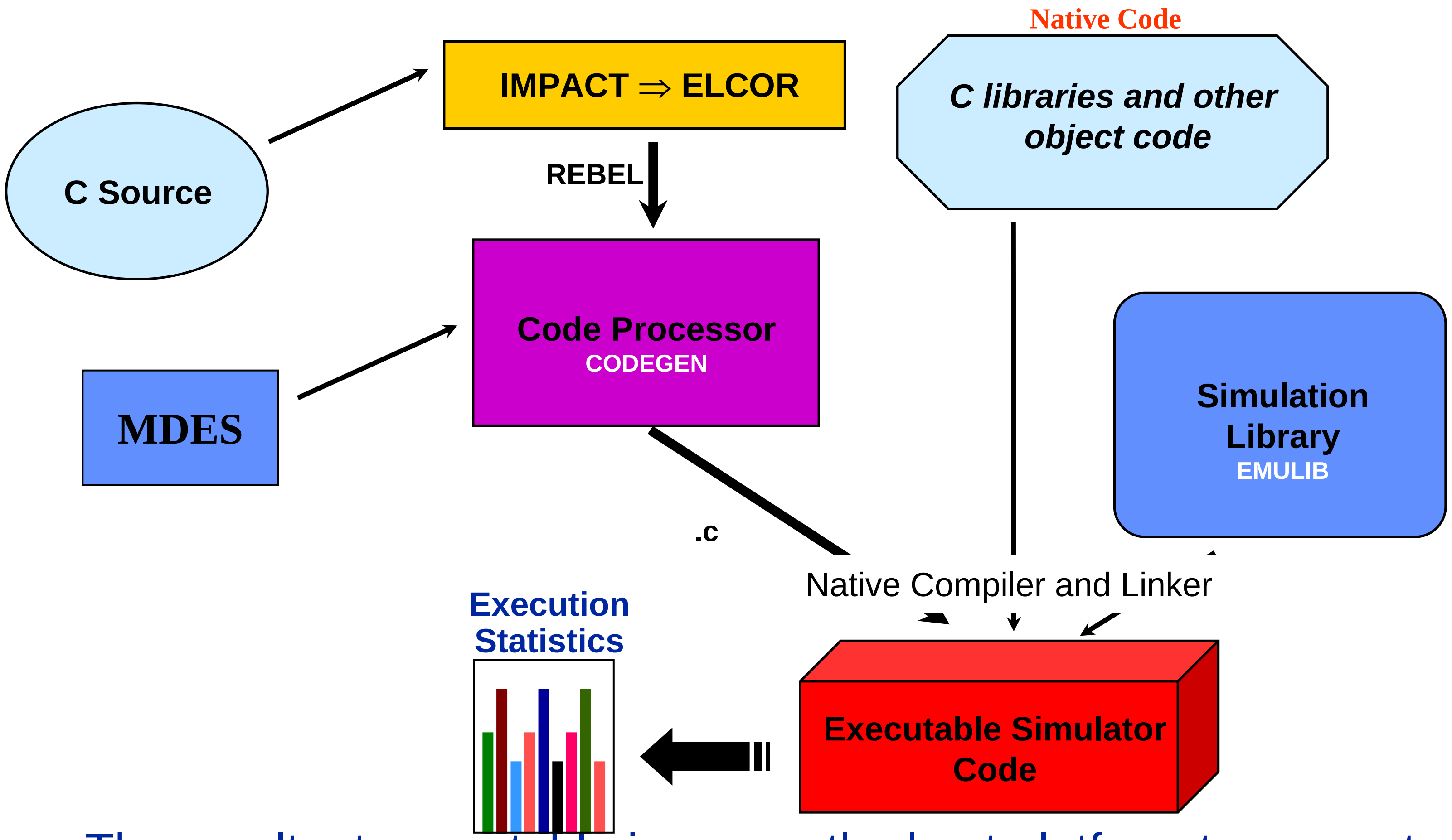
- A native compiler (i.e. GCC) is used to generate the equivalent machine code
  - Collection of .o files

Simulator: Overview



- The object files are linked with the *Simulation Library* and any other native libraries

Simulator: Overview



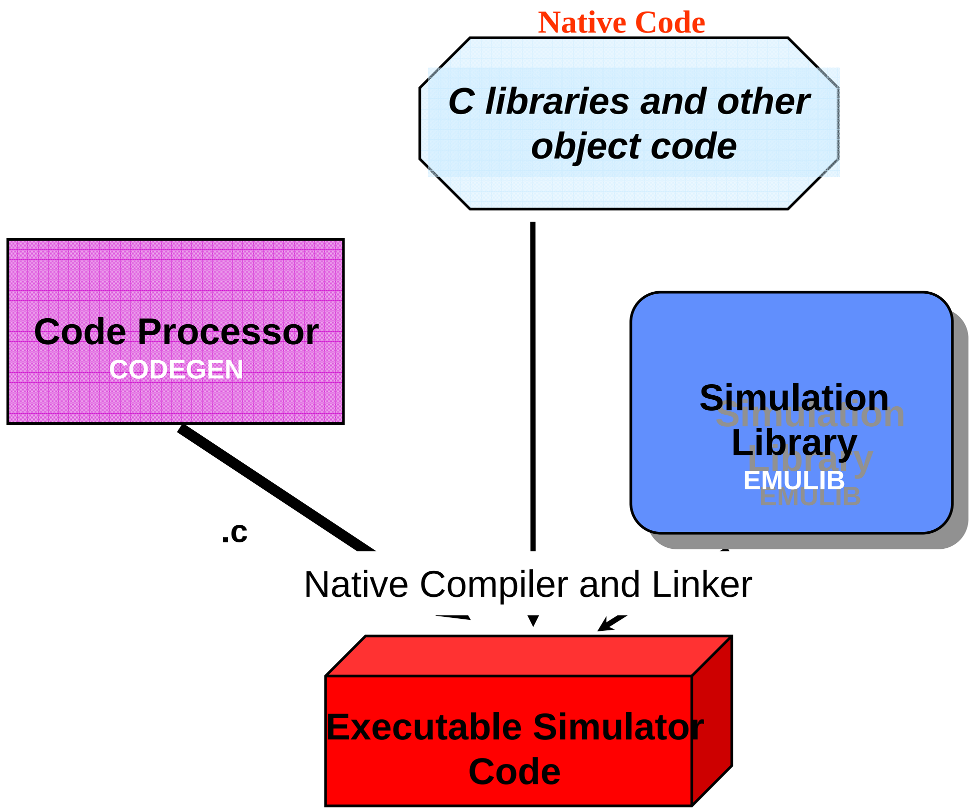
- The resultant executable is run on the host platform to generate statistics and dynamic profile information

Simulator: Codegen

- Codegen
  - Input: REBEL code that is generated by Elcor
  - Output (one for each function of the benchmark):
    - **Benchmark.simu.c file wrapper functions for emulating the assembly code**
    - **Benchmark.simu.c.tbls file for declaring the tables of assembly operations**
    - **Benchmark.simu.c.inc file contains global declarations and the statistics tracking data structures**
- The .tbls and .inc files are #included into the .c file.

Simulator: Compilation

- EMULIB
  - Collection of files corresponding to operation types:
    - **PD_load_store_ops.c**
    - **PD_move_ops.c**
    - **PD_int_arith_ops.c**
    - **Etc.**
  - These files contain a function for each variation of the HPL-PD operations.
  - These functions emulate the operation during the simulation
  - Other files support the register files, function call mechanics, statistics collection, etc.


Native Code
C libraries and other object code
Code Processor
CODEGEN
Simulation Library
EMULIB
.c
Native Compiler and Linker
Executable Simulator Code

Configuring the Target Machine

Target Machine Description

- Trimaran includes an advanced Machine Description facility, called Mdes, for describing a wide range of ILP architectures.
- It consists of
  - A high-level language, Hmdes2, for specifying machine features precisely
    - **functional units, register files, instruction set, instruction latencies, etc.**
    - **Human writable and readable**
  - A translator converting Hmdes2 to Lmdes2, an optimized low-level machine representation.
  - A query system, mQS, used to configure the compiler and simulator based on the specified machine features.

Target Machine Configuration

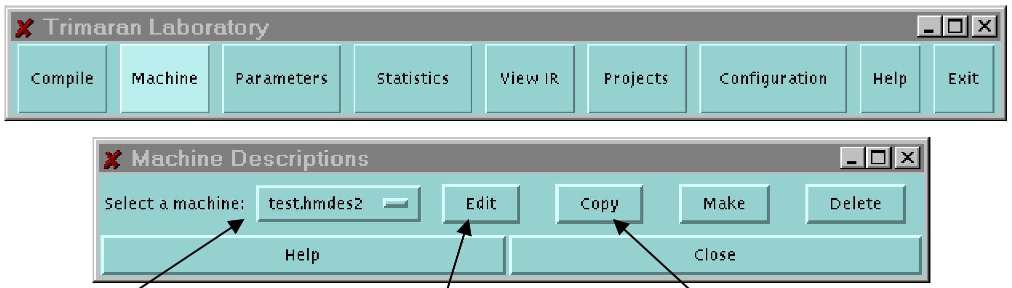
- Generally, it is expected that Trimaran users will make modest changes to the target machine configurations
  - within the HPL-PD architecture space
  - using the Trimaran graphical user interface (GUI) to modify an existing machine description rather than write a new one from scratch.
  - Very easy to change
    - **number and types of functional units**
    - **number of types of register files**
    - **instruction latencies**

Machine Descriptions

- There are two issues that a researcher must consider:
  - How can the features of a target machine be modified so that the changes are reflected in the code generated by the compiler and in the machine being simulated during execution?
    - **Most common features can be changed via the GUI**
    - **Extensive modifications can be specified via an Hmdes2 description.**
  - How can a new compiler module, implemented by a researcher, determine the features of the target machine?
    - **The mdes Query System, mQS**

Using the Trimaran GUI to configure an HPL-PD machine

- The Trimaran system is delivered with a full Mdes description of several machines in the HPL-PD architecture space.
  - Machine features within the HPL-PD space are easily modified using the Trimaran GUI.
    - **Functional units, register files, instruction latencies**



Choose a machine configuration

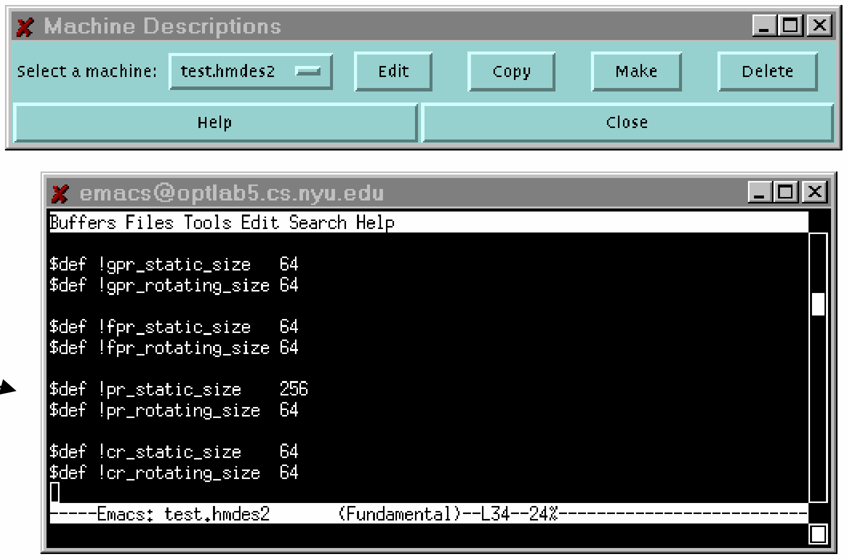
Modify the machine configuration

Create a new machine configuration based on an existing one

Using the Trimaran GUI to configure an HPL-PD machine (cont)

- When the Edit button is clicked, an Emacs window opens a configuration file for editing.
  - This file is read by the Hmdes2 preprocessor and translator to create a new Lmdes2 machine description.
- Changes to the configuration file are reflected in the target machine when the Make button is clicked.

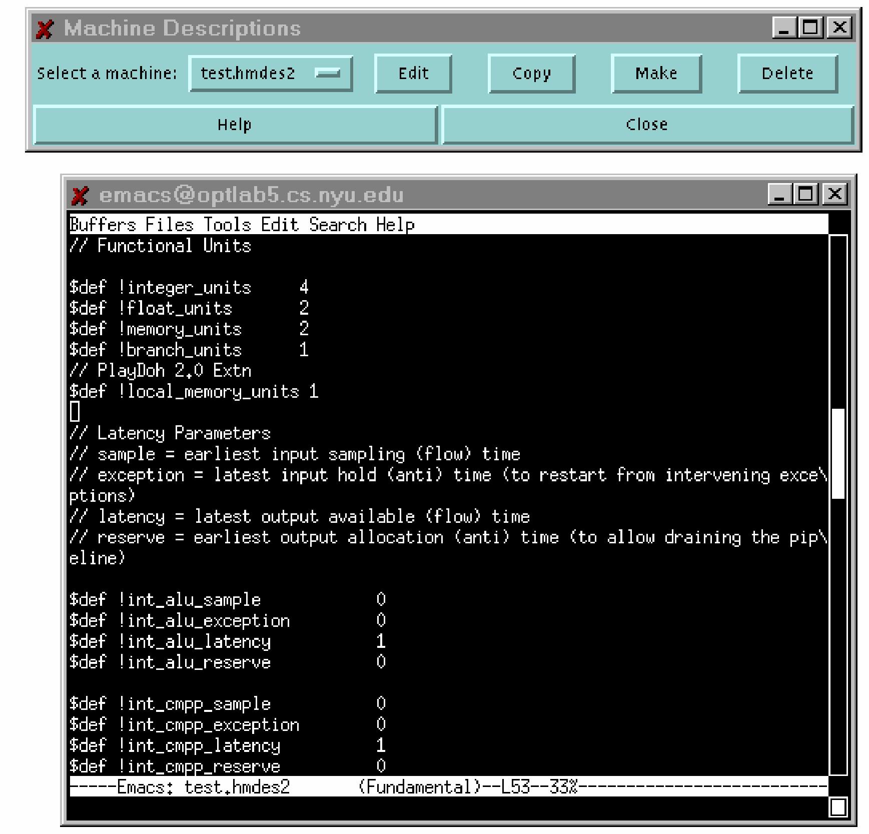
This portion of the file specifies the number of static and rotating registers in the various register files.



```
$def !gpr_static_size   64
$def !gpr_rotating_size 64

$def !fpr_static_size   64
$def !fpr_rotating_size 64

$def !pr_static_size    256
$def !pr_rotating_size  64

$def !cr_static_size    64
$def !cr_rotating_size  64

----Emacs: test.hmdes2      (Fundamental)--L34--24%-----------------------
```

Using the Trimaran GUI to configure an HPL-PD machine (cont)

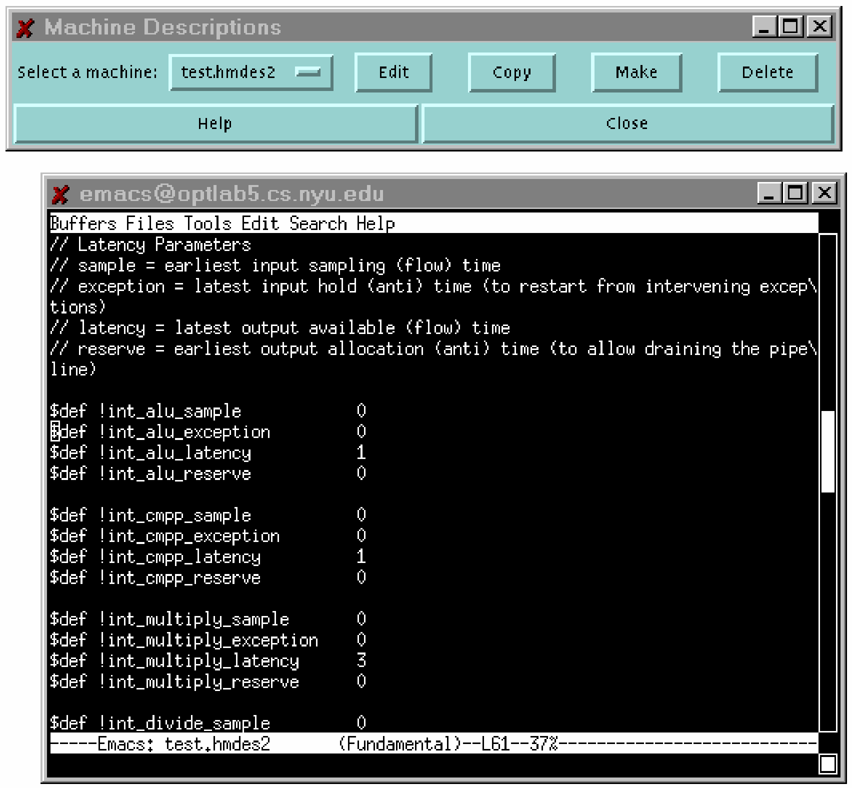
- With a few keystrokes, the configuration of functional units is changed
  - From an essentially sequential machine (very few functional units)
  - To a highly parallel machine.



```
// Functional Units

$def !integer_units    4
$def !float_units      2
$def !memory_units     2
$def !branch_units     1
// PlayDoh 2.0 Extn
$def !local_memory_units 1

// Latency Parameters
// sample = earliest input sampling (flow) time
// exception = latest input hold (anti) time (to restart from intervening exce\
ptions)
// latency = latest output available (flow) time
// reserve = earliest output allocation (anti) time (to allow draining the pip\
eline)

$def !int_alu_sample            0
$def !int_alu_exception         0
$def !int_alu_latency           1
$def !int_alu_reserve           0

$def !int_cmpp_sample           0
$def !int_cmpp_exception        0
$def !int_cmpp_latency          1
$def !int_cmpp_reserve          0
----Emacs: test.hmdes2       (Fundamental)--L53--33%--------------------------
```

Using the Trimaran GUI to configure an HPL-PD machine (cont)

- The machine configuration changes via the GUI can be quite detailed.
  - In this case, the precise latencies of operations can be modified.
    - **When the input registers are sampled.**
    - **When the value in the output register is available.**
    - **Etc.**



```
// Latency Parameters
// sample = earliest input sampling (flow) time
// exception = latest input hold (anti) time (to restart from intervening excep\
tions)
// latency = latest output available (flow) time
// reserve = earliest output allocation (anti) time (to allow draining the pipe\
line)

$def !int_alu_sample          0
$def !int_alu_exception       0
$def !int_alu_latency         1
$def !int_alu_reserve         0

$def !int_cmpp_sample         0
$def !int_cmpp_exception      0
$def !int_cmpp_latency        1
$def !int_cmpp_reserve        0

$def !int_multiply_sample     0
$def !int_multiply_exception  0
$def !int_multiply_latency    3
$def !int_multiply_reserve    0

$def !int_divide_sample       0
-----Emacs: test.hmdes2       (Fundamental)--L61--37%---------------------------
```

Describing a Machine Using Hmdes2

- If more extensive changes to a machine need to be made than can be handled in the GUI, the user can describe the machine using Hmdes2.
  - High-level machine description language
- There is a limit, however, to the extent that a machine can be modified and still be the target for the Trimaran compiler, and be simulated using the Trimaran simulator.
  - The machine must remain in the HPL-PD architecture space.
  - The instruction set cannot be significantly changed.
- The GUI is the recommended method for modifying the target machine.
  - However, Hmdes2 is a very interesting mechanism…