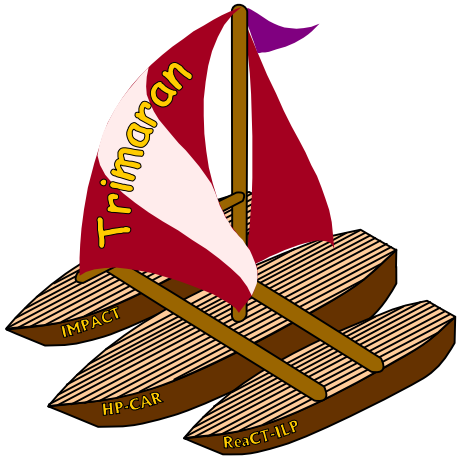



# An Overview of the Trimaran Compiler Infrastructure

# Infrastructure Goals

- 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

# Infrastructure Support

The infrastructure is comprised of the following components:

- 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)

# Infrastructure Support (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.

# Infrastructure Support (cont)

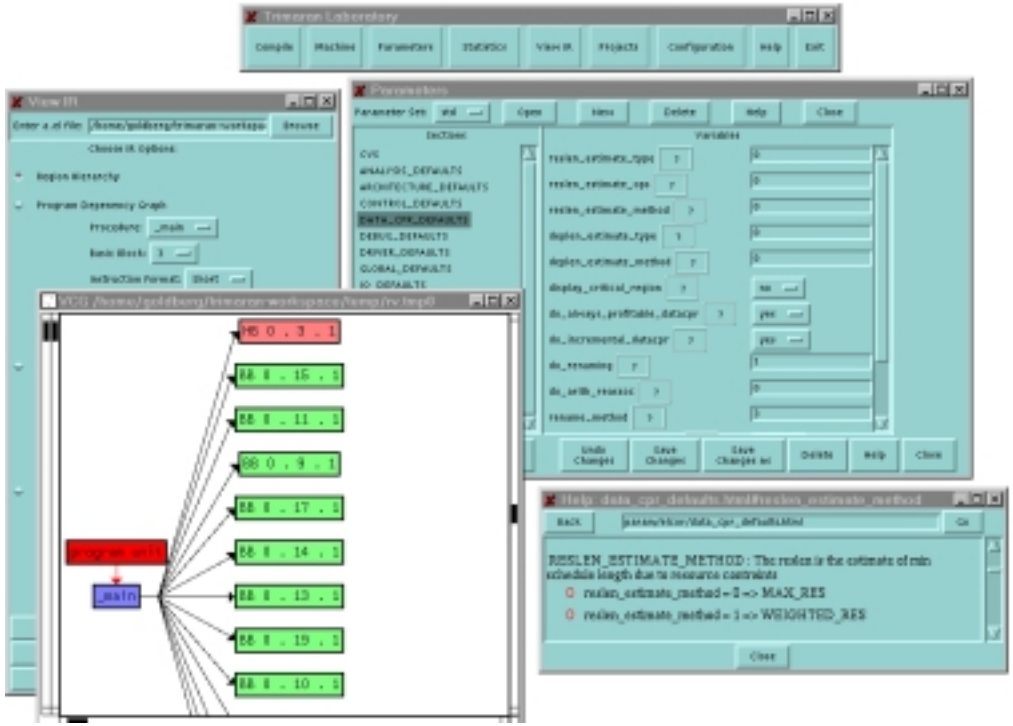
- An extensible IR (intermediate program representation)
  - Has both an internal and textual representation, with conversion routines between the two. The textual language is called Rebel.
  - Supports modern compiler techniques by representing control flow, data and control dependence, and many other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

# Infrastructure Support (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.

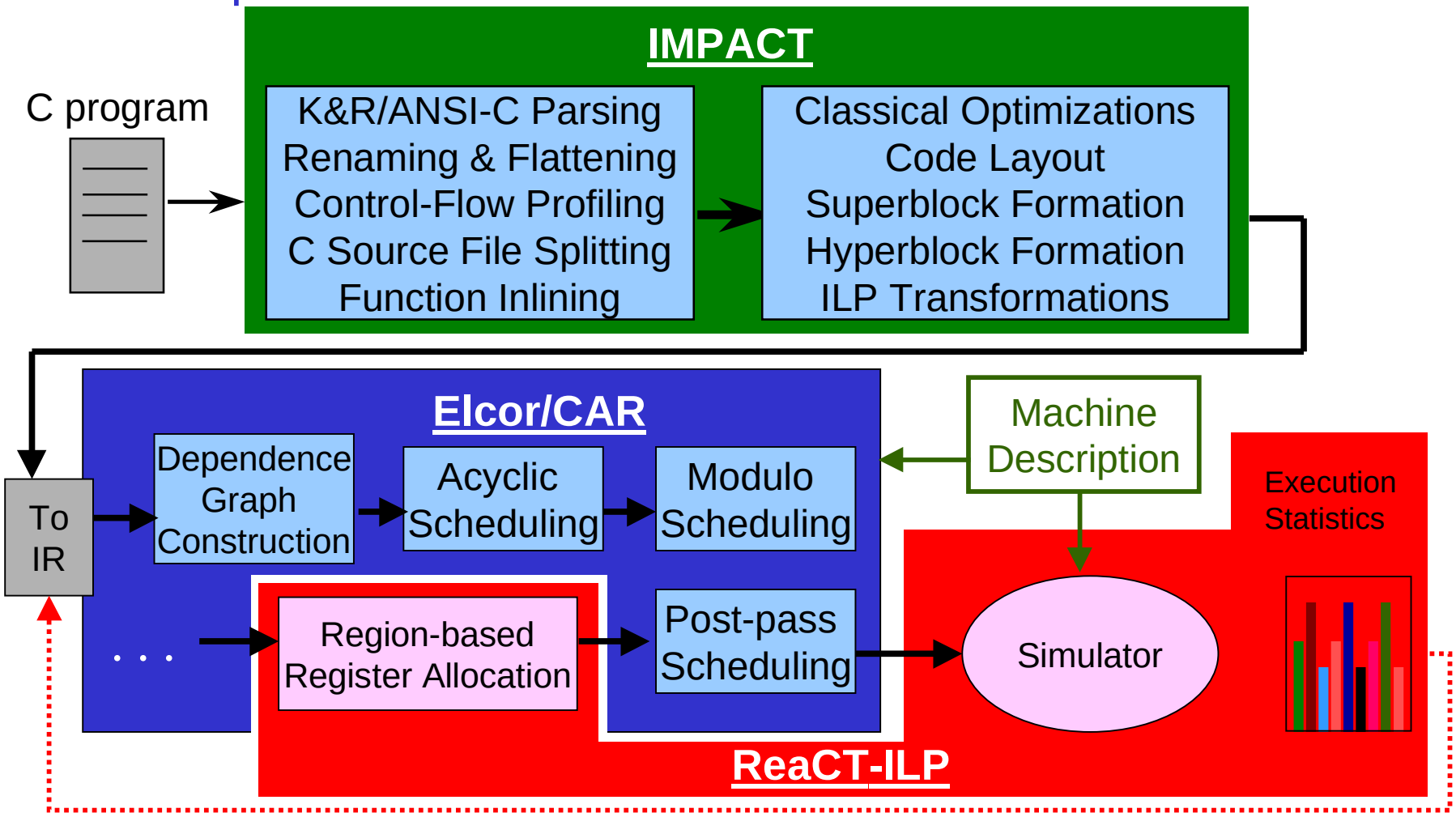
# Infrastructure Support (cont)

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



# System Organization

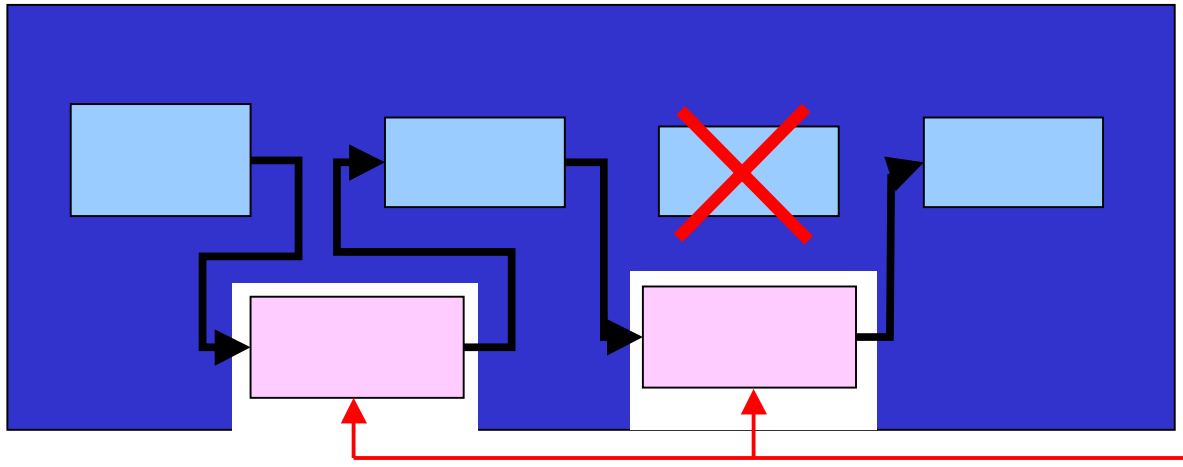
- A compiler researcher's view of the infrastructure:



# The research process

The infrastructure is used for designing, implementing, and testing new compilation modules to be incorporated into the back end.

– These phases may augment or replace existing ILP optimization modules.




– New modules may be the result of research in scheduling, register allocation, program analysis, profile-driven compilation, etc.

- For example, NYU has added a region-based register allocator.

# Why use Trimaran?

- It is especially geared for ILP research
- It provides a rich compilation framework
  - Parameterized ILP architecture (HPL-PD)
  - Machine description language
  - Single intermediate program representation
    - provides mechanism for representing wide range of program information
  - Cycle-level execution simulation
    - provides run-time information for profile-driven compilation

# More reasons…

- The framework is populated with a large number of existing compilation modules
  - provides leverage for new compiler research
  - supports meaningful experimentation, rather than simply running  toy programs.
  - Full compilation and execution path already exists
- There's a commitment on our part to releasing a robust, tested, and documented software system.

# Case Study

- Here's a data point on the usability of Trimaran:
  - We implemented a sophisticated region-based register allocator in the back end.
  - 2 person-months implementation time + 1 person-month testing and debugging
    - Once familiar with infrastructure (several more months)
    - Very short development time for a real register allocator in a serious compiler.

# The Trimaran Tutorial

- The full Trimaran Tutorial has been given at:
  - IEEE Conference on Parallel Architectures and Compiling Techniques (PACT), Paris, October 1998.
  - IEEE Symposium on Microarchitecture (MICRO-31), Dallas, December 1998
  - ACM SIGPLAN Symposium on Programming Language Design and Implementation (PLDI'99), Atlanta, May 1999.

# Since the Release...

- The Trimaran Web Site has been visited more than 5000 times.
- The Trimaran system has been downloaded to over 900 sites.
  - The 50mb system is currently ported to HP-UX.
  - A Linux port is due in mid-June.
  - A Solaris port is planned.
  - Simulator is being improved to provide measurements of cache performance.
- Embodies over 100 person-years of work.