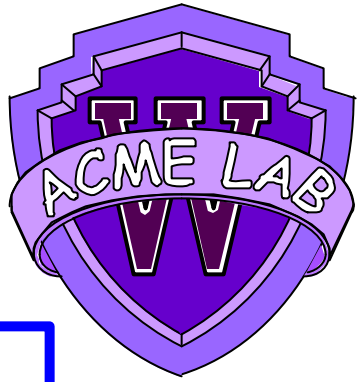
The ACME Lab
Totem and Précis Projects

Mark L. Chang

mchang@ee.washington.edu

February 26, 2002



The ACME Lab

- Adaptive Computing Machines and Emulators
- Professor Scott Hauck
- Eight graduate students
  - Mark L. Chang
  - Katherine Compton
  - Kenneth Eguro
  - Mark Holland
  - Kimberly Motonaga
  - Todd Owen
  - Shawn Phillips
  - Akshay Sharma
- http://acme.ee.washington.edu

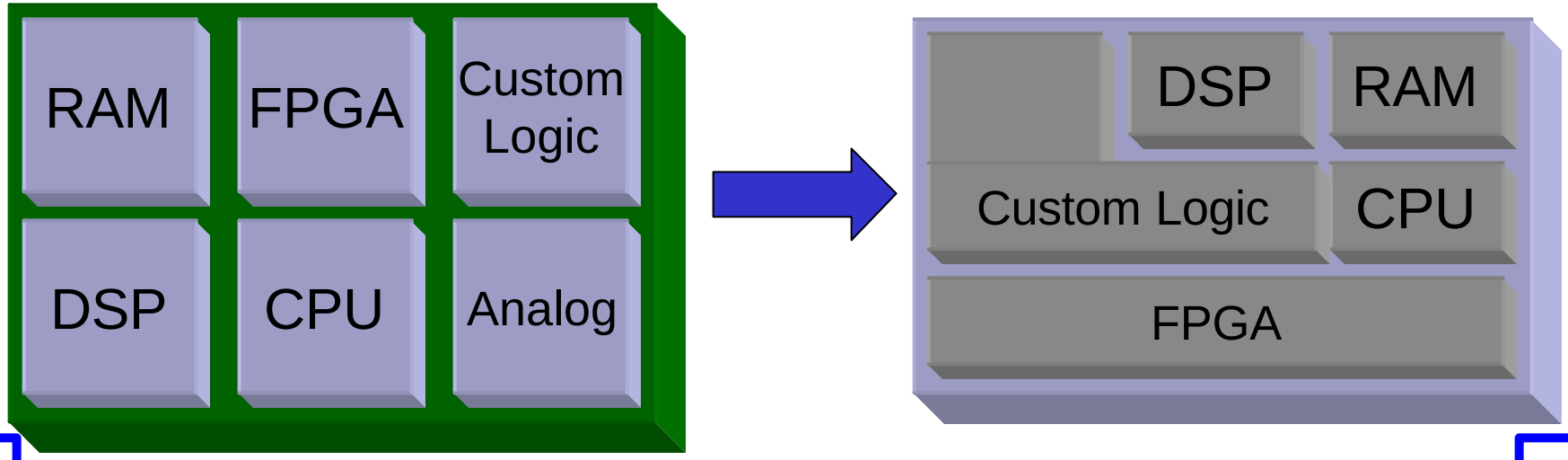
Current Research

- FPGA-based Hyperspectral Image Compression
- FPGA Configuration Management
- ALICE - ACME Labs Instant Computer
- Fast CAD
- Totem: Reconfigurable Subsystems for System-on-a-Chip
- Précis: Variable Precision Analysis

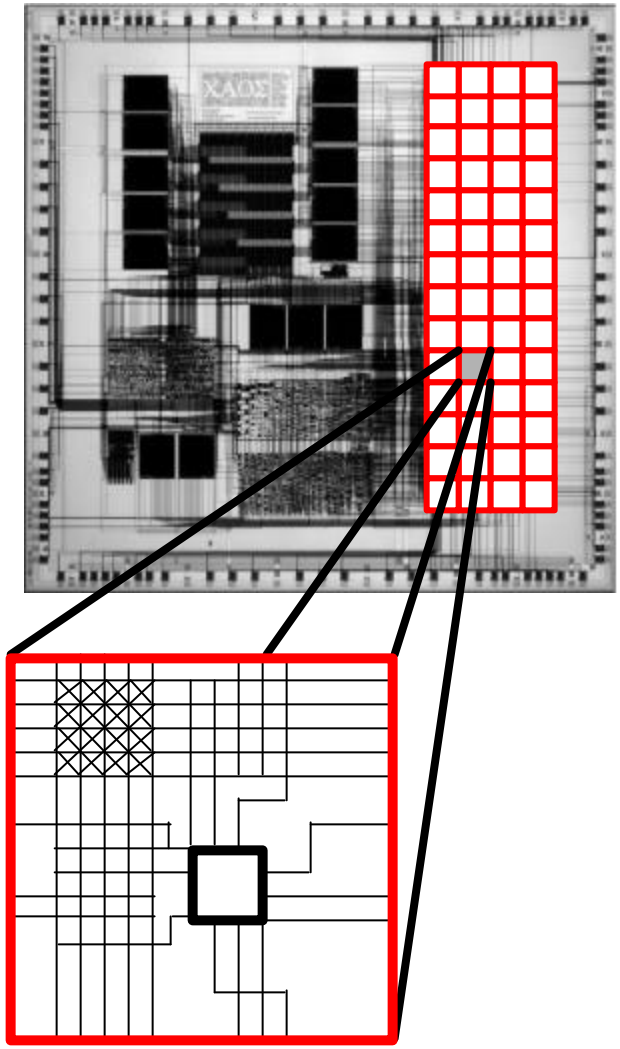
Project Overview

Reconfigurable Computing

- **Specialized hardware is commonly used for compute-intensive applications**
  - **Coprocessors (FPU, graphics, sound, …)**
  - **Accelerator boards (FPGA boards, I/O cards, …)**
- **FPGAs also perform well for many of these apps**
- **Reconfigurable logic in System-On-A-Chip**



FPGA in System-on-a-Chip

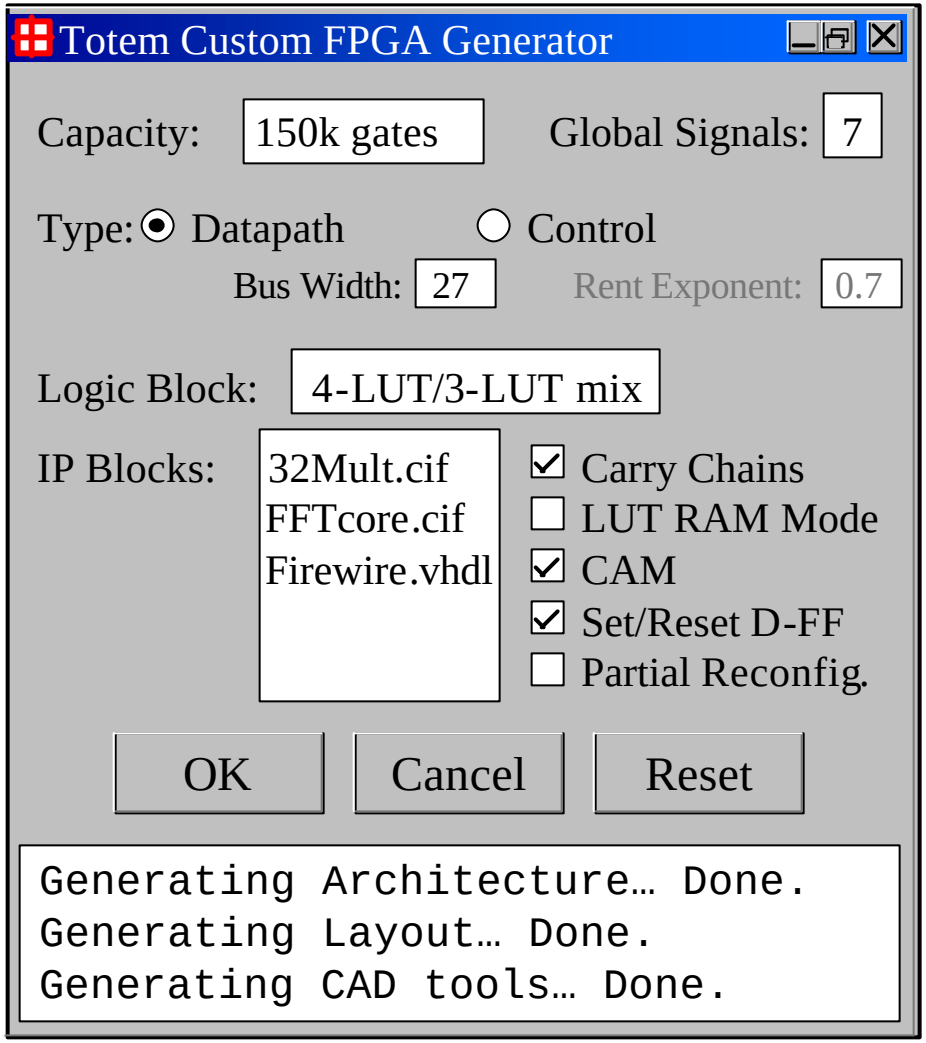


- **Fast Time-To-Market**
- **Post-Fabrication Customization**
  - **Broaden application domain**
  - **Run-time Reconfiguration**
  - **Bug Fixes**
  - **Upgrades**
- **10x-100x Worse:**
  - **Area**
  - **Performance**
  - **Power**

Domain Specific FPGAs

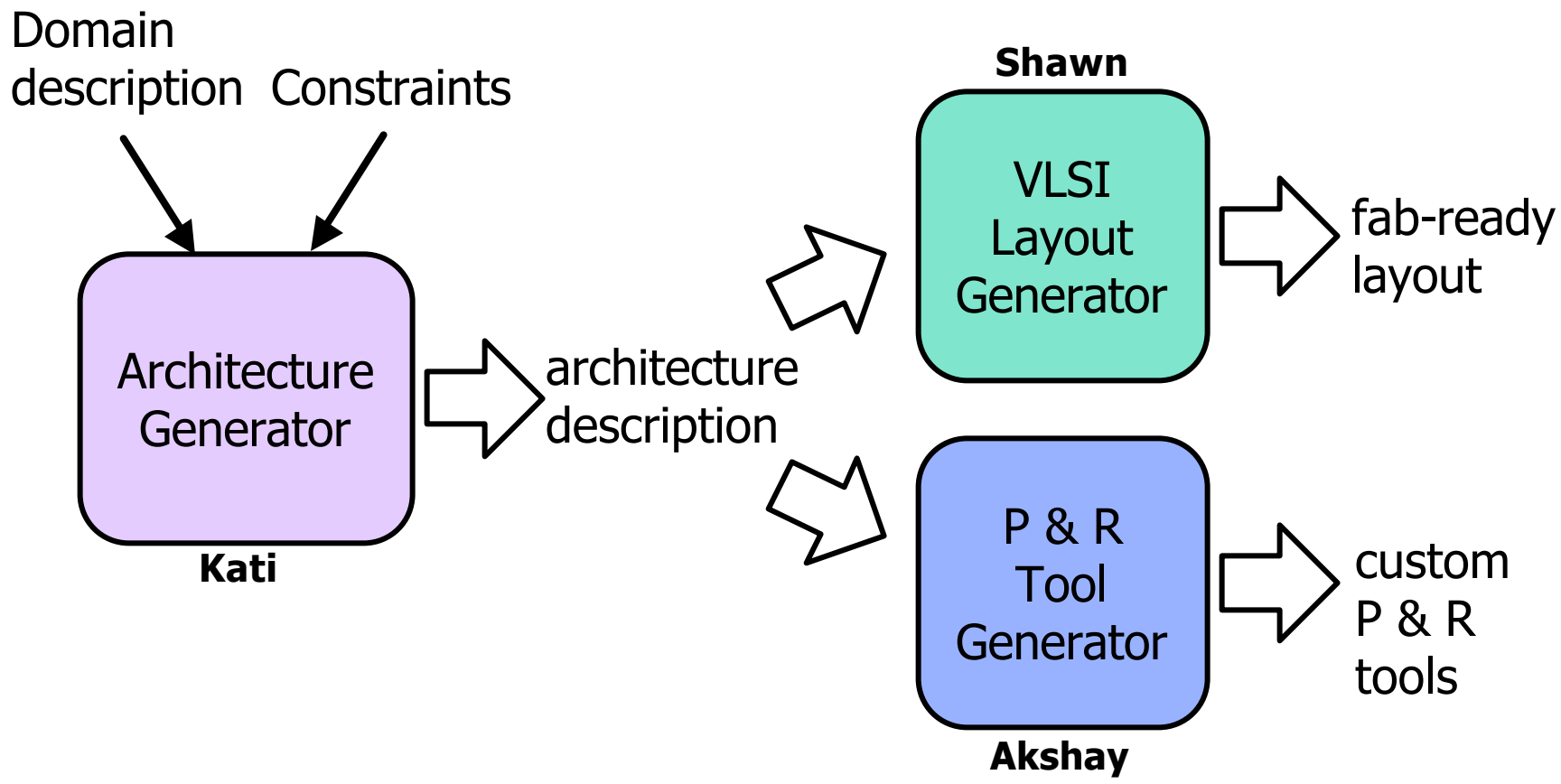
- **Modify architecture to more efficiently support a range of known applications**
  - **Higher performance**
  - **Smaller area requirements**
- **Creating architecture by hand for each domain?**
  - **Time consuming!**
  - **Makes SoC design too costly**
- **Do it *AUTOMATICALLY***
  - **Would not add significantly to time of SOC design**
  - **Fast time-to-market**

Totem Goals

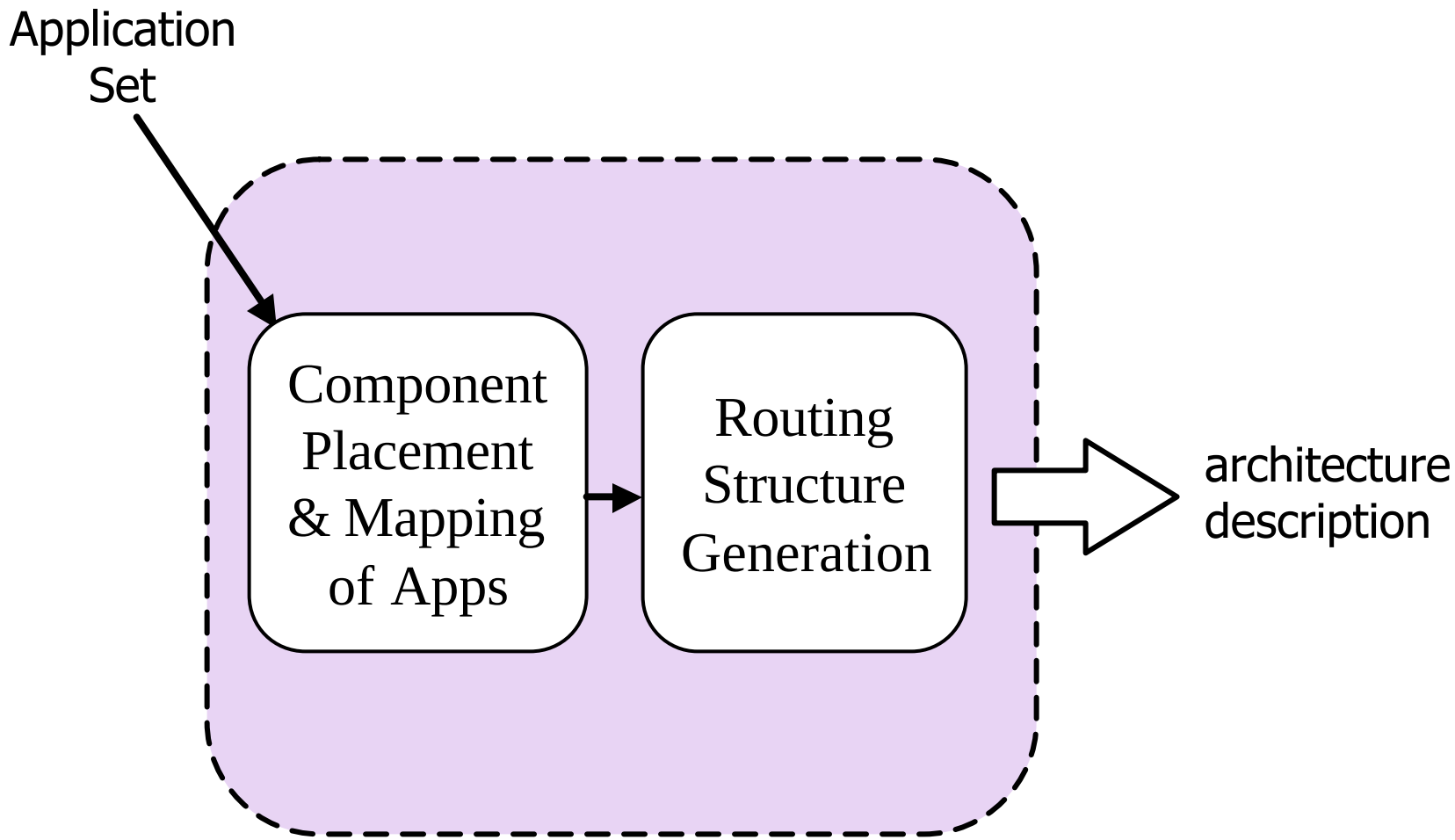


- **Speed of ASIC**
- **Flexibility of FPGA**
- **Specialized for application domains**
  - **DSP, cryptography, scientific computing, etc.**
- **Embedded in an SOC**
  - **Instead of generic reconfigurable fabric**
- **Quick automatic generation of custom reconfigurable architectures**

Totem Design Flow

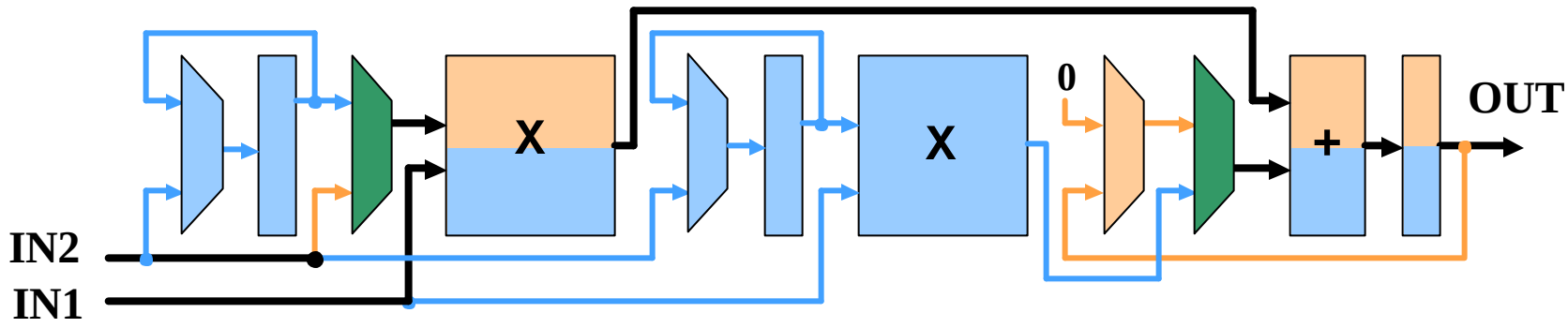


Architecture Generation



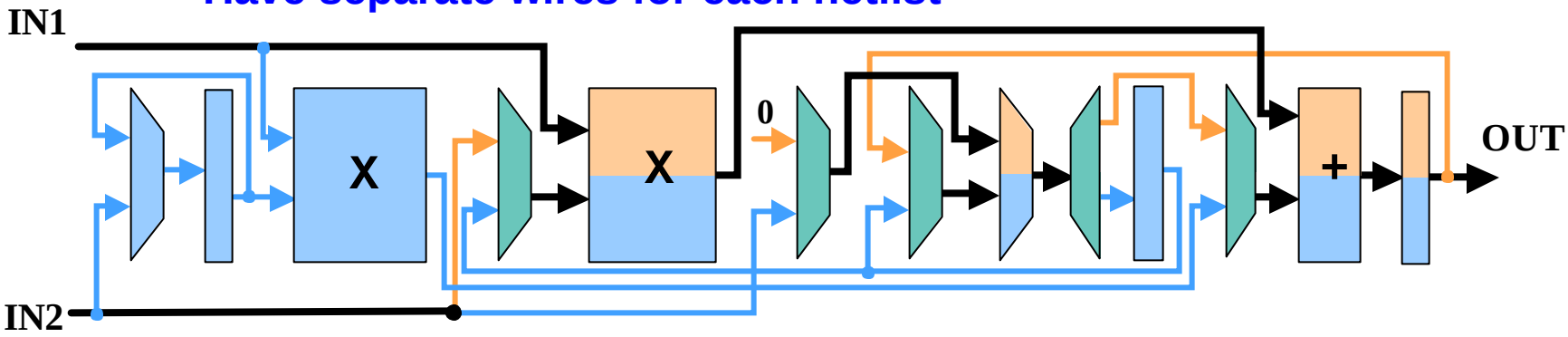
Architecture Generation

- Input several netlists for consideration
- Use simulated annealing to place physical and logical components
- Some components shared between netlists
- Muxes added to handle different needs of the netlists (shown in green)
- Some wires shared (black), others not



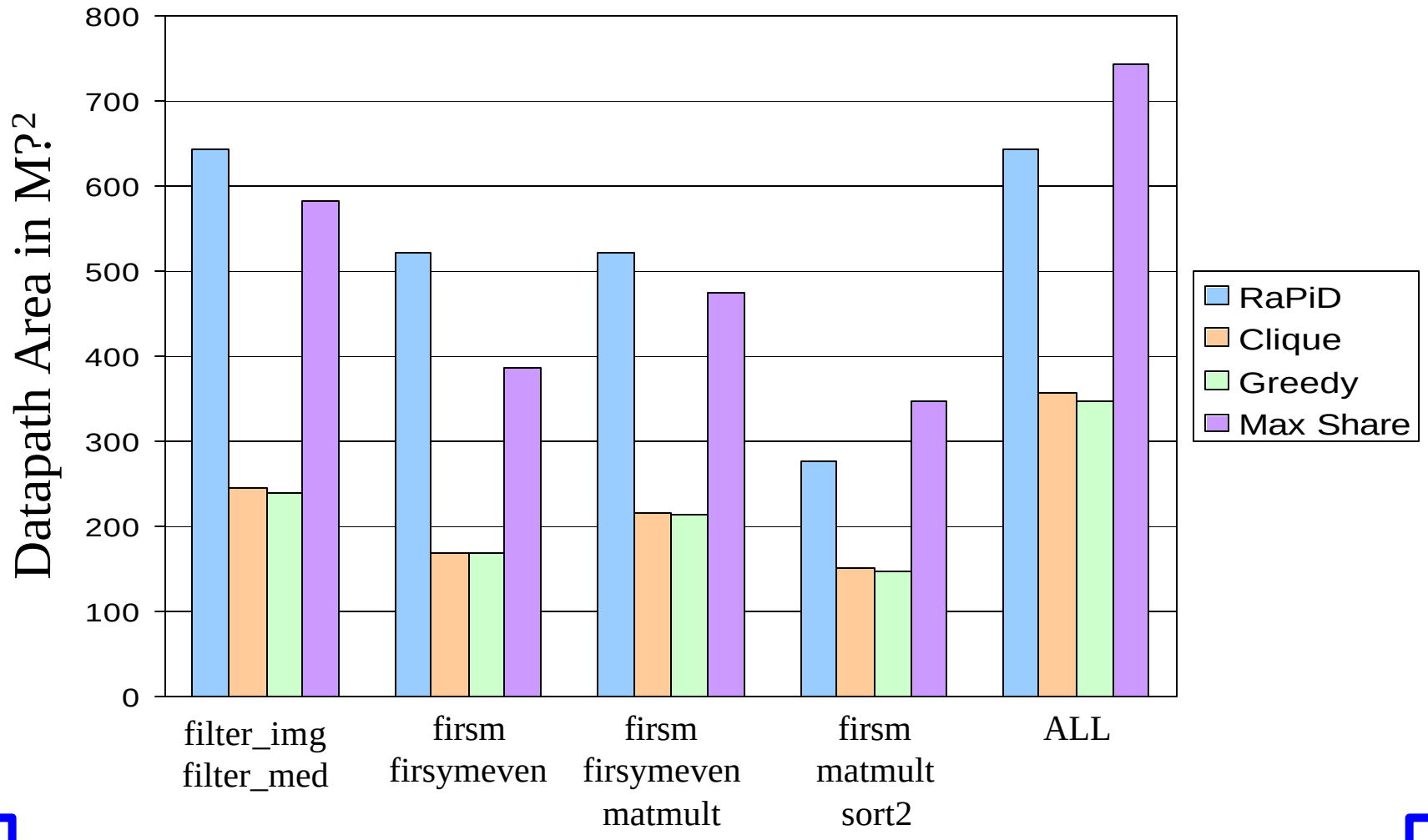
Routing Structure Generation

- Need to make *wires* for the signals
  - Share wires between netlists
  - Have separate wires for each netlist



- Compare several different algorithms
  - Maximum wire sharing (left-edge)
    - Loose approximation of “smallest” solution
  - Greedy sharing
    - Group wires together that share sources or destinations
  - Clique partitioning sharing

Area Results



Automated VLSI Layout

- With architecture description, we need to generate a layout automatically
- Standard Cell
  - Use standard cells to implement architecture description
  - Use standard tools (Cadence) to perform physical design
- Template Reduction
  - Use standard component templates (macro-cells)
  - Remove logic that isn't needed to reduce area, perform compaction
- Circuit Generator
  - Create parameterized generators for main FPGA components
    - LUTs, de/muxes, DFF, SRAM bits, etc.
  - Create complete reconfigurable fabric by abutting generated components

Place and Route Tools

- Goal is to efficiently map a logical netlist on a generated architecture
- Placement Objective:
  - Determine the position that each logic-element will occupy
  - Ensure that the router is able to find enough routing resources
  - Use simulated annealing
- Router:
  - Assigns a set of routing resources to each signal in the netlist
  - Optimization goals are generally area, delay, or a combination of both
  - Use PathFinder algorithm

Totem: The Future

- **Architecture Generation**
  - More flexible routing structures
  - 2D structures
- **Place and Route Tool Generation**
  - Pipelined routing
  - 2D architecture support
- **Automated VLSI Layout**
  - Template reduction
  - Automatic circuit generator

Précis:
Design-Time Precision Analysis

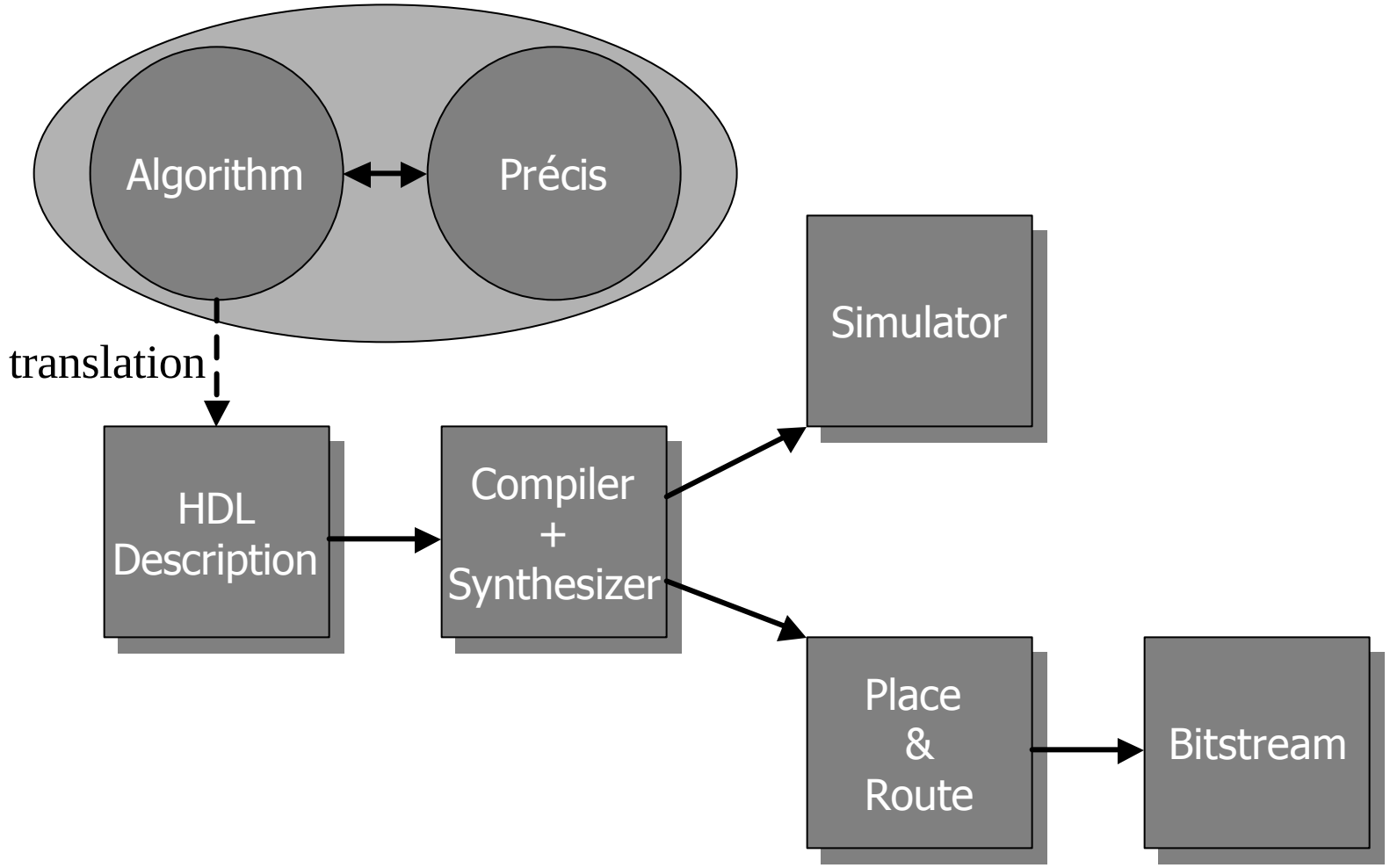
Motivation

- Mapping algorithms to FPGAs can be tough
- Coping with precision is a key difficulty in mapping to hardware
  - Too many bits == wasted resources
  - Too few bits == incorrect results
- Difficult for the non-hardware-savvy
- Need tool to help designer optimize bit-widths
- Still want to develop in a high-level description language
  - MATLAB, C, C++, Java

Motivation

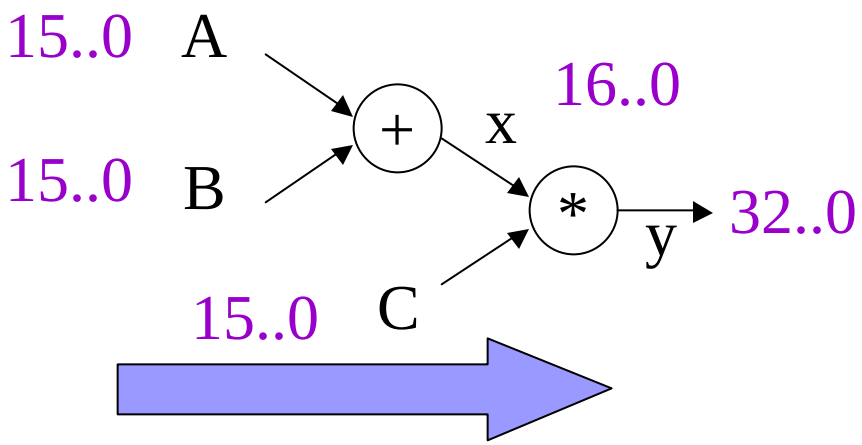
- **Few tools can automate HDL generation**
  - **C-to-Verilog, C-to-VHDL**
  - **MATCH (MATLAB to target code for embedded systems)**
  - **Proprietary systems and languages…**
- **Even fewer tools offer “Designer Aides”**
  - **Floating-point to fixed-point translation**
  - **Flexible resource utilization (e.g. memory vs. logic)**
  - **Dealing with precision issues**
- **Tool gap in *design-time* tools**

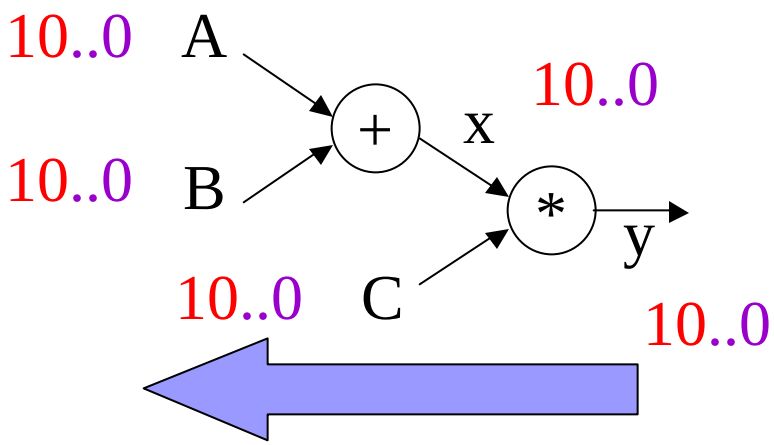
Role of Précis in Tool Flow



Précis Application

- **Supports MATLAB as an input language**
- **Provides propagation engine for precision constraints**

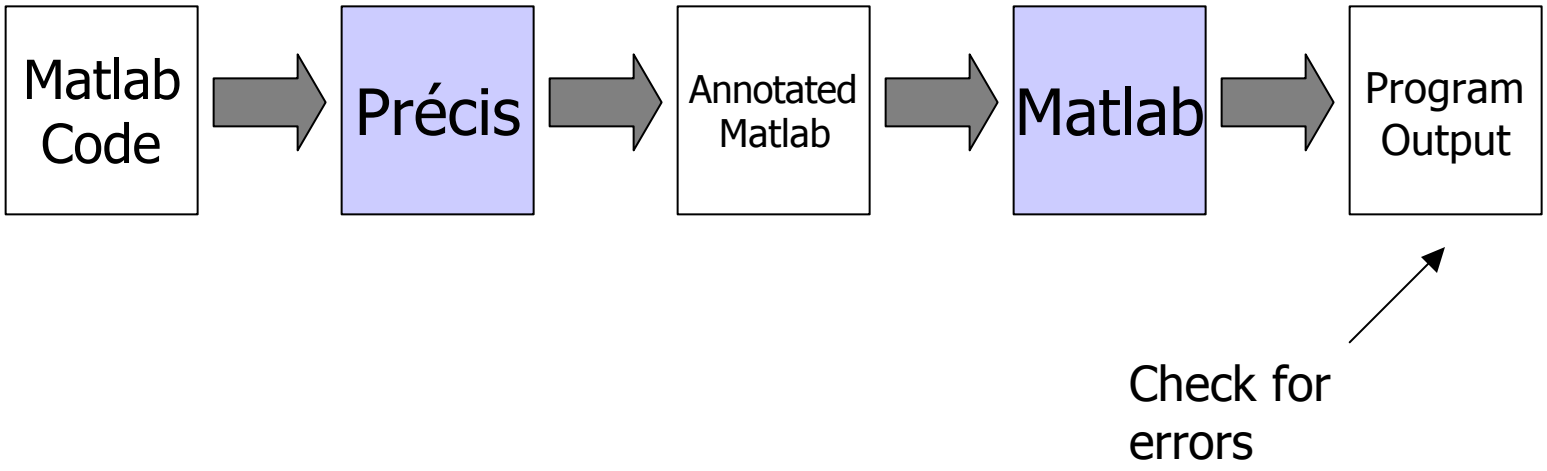





```
x = a + b
y = c + x
```

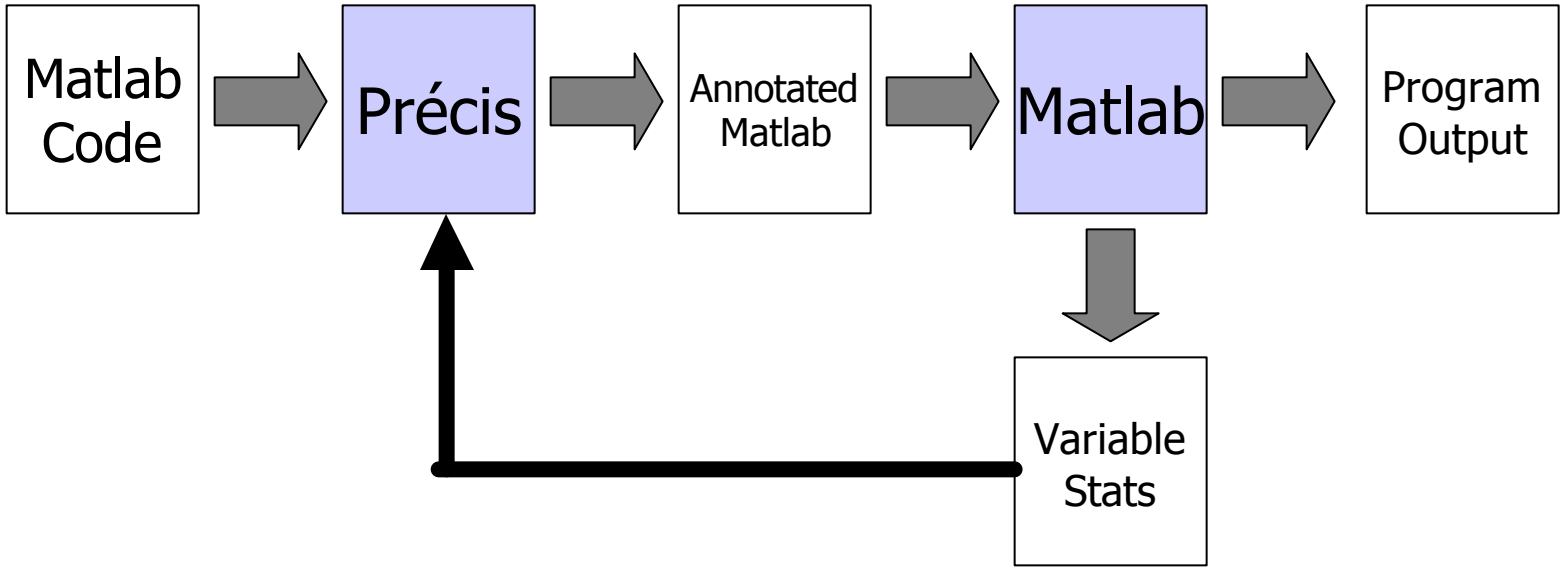
Simulation

- **Generates annotated MATLAB**
- **User may specify precision constraints on any variable**
- **Simulation demonstrates the effects of fixed-point operations which may result in rounding or truncation errors in the output**



Range Finding

- **Generates annotated MATLAB**
- **Records ranges of variables for sample data sets**
- **Results are reintegrated into Précis to allow for more investigation**



User Guidance

- Want to guide a developer's manual optimization
  - Helpful for a novice designer
  - Provides a starting point for hand-optimization
  - Allows iterative optimization of the implementation
    - Optimize until if fits and runs…
    - Optimize further if we have time/money
- We ask questions, developer answers
  - What is the algorithm
  - Known precision of variables
  - Simulation and data gathering
- Provide suggestions to the user

Slack Analysis

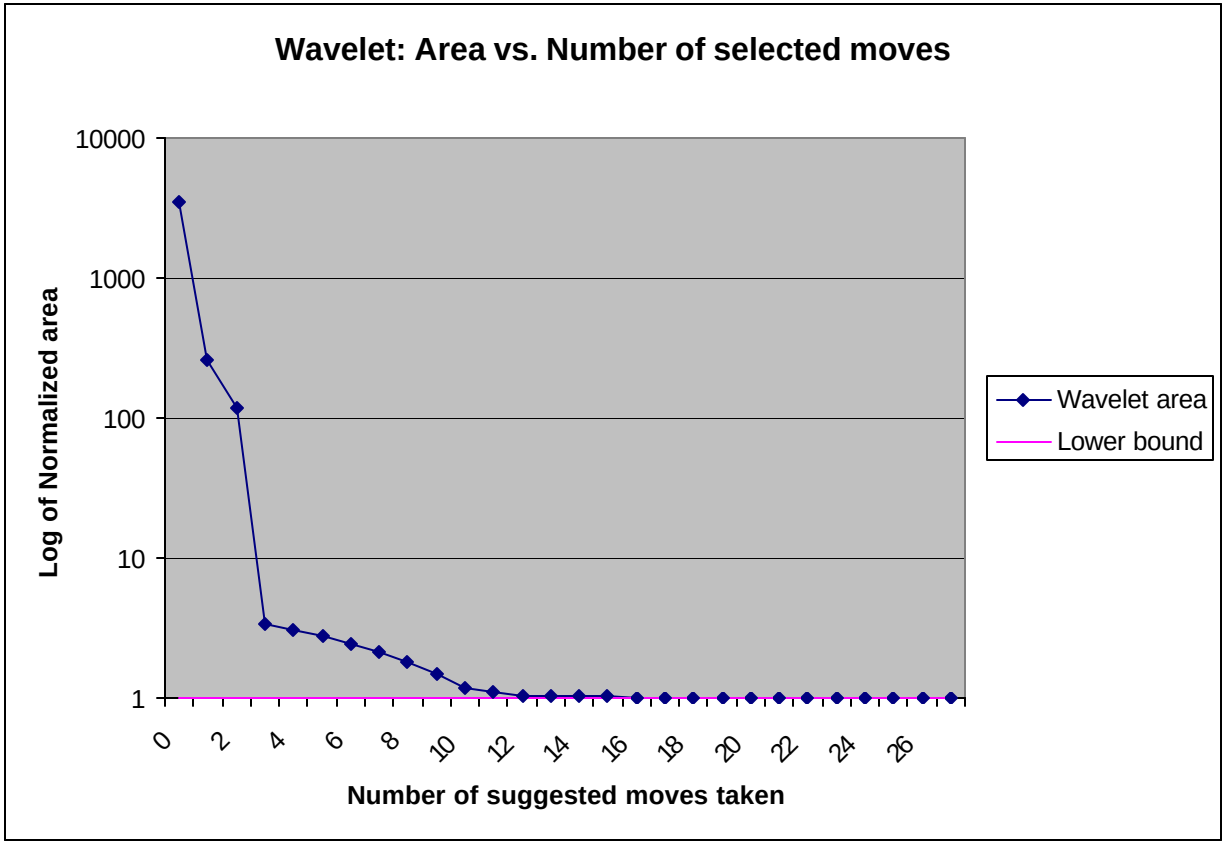
- **Propagation == upper bound**
- **Range finding == lower bound**
- ***Difference == Slack***

- **Try to identify nodes that have the greatest area impact on the final circuit implementation**
- **Present these results as an ordered list of "moves" for the user to (optionally) make**

Slack Analysis

- For each node with slack, set precision to lower bound
  - Propagate change through system
  - Calculate the gain in area for this “move”
- This creates a “tuning list” of variables to consider
- Iteratively, the user can choose to make moves and recalculate what the next move should be

Benchmark: Wavelet Transform



- 27 variables selected for slack analysis
- 3 moves to within a factor of 3 of lower bound
- 11 moves to within 10% of lower bound
- 13 moves to within 3% of lower bound

Current Research

- FPGA-based Hyperspectral Image Compression
- FPGA Configuration Management
- ALICE - ACME Labs Instant Computer
- Fast CAD
- Totem: Reconfigurable Subsystems for System-on-a-Chip
- Précis: Variable Precision Analysis