

---

# Fast Source-Level Data Assignment to Dual Memory Banks

Alastair Murray and Björn Franke  
Institute for Computing Systems Architecture,  
School of Informatics, University of Edinburgh

SCOPES: March 14th, 2008



# Overview

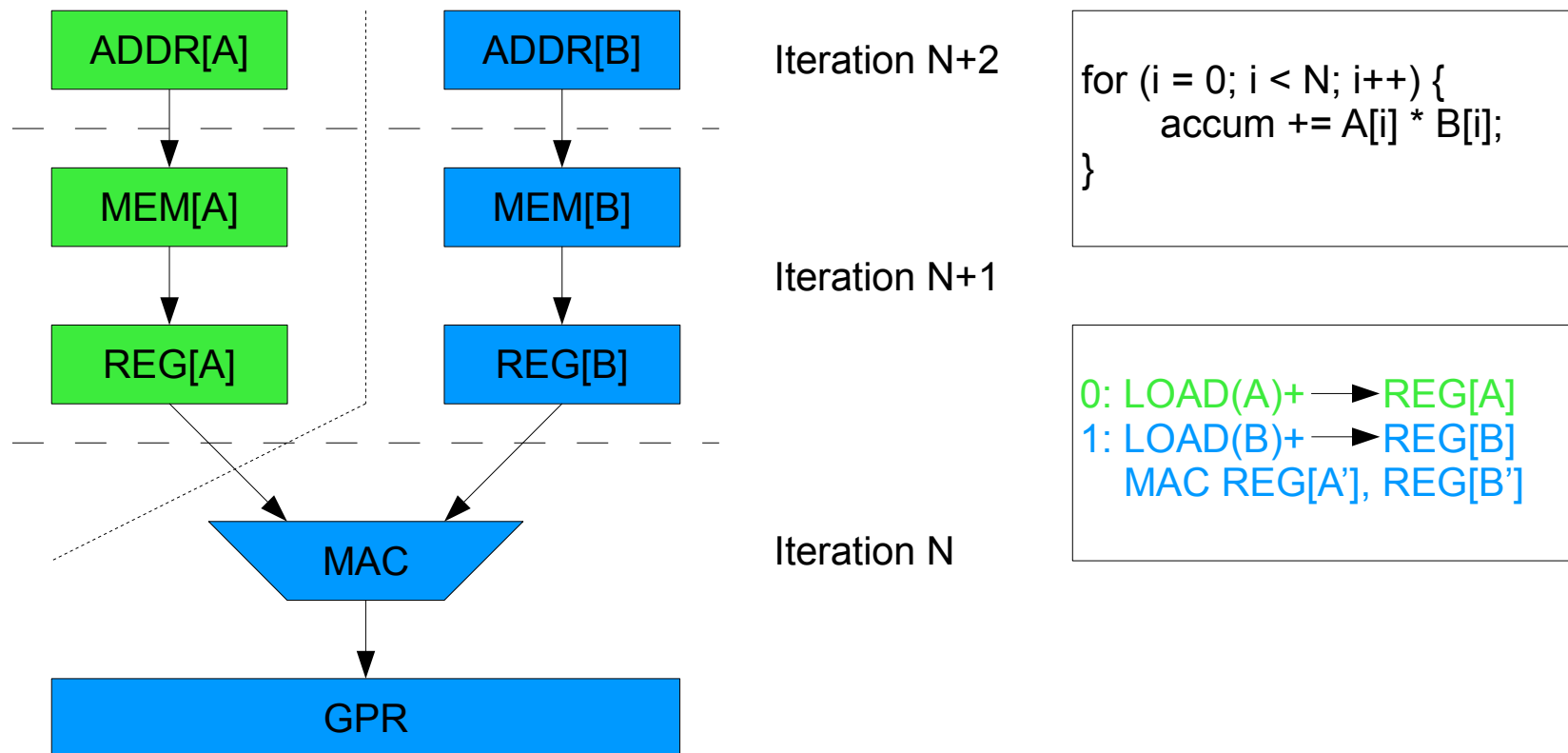
- Many DSPs have dual on-chip memories to increase bandwidth.
- Automatic methods to exploit these already exist:
  - Aren't used in Industrial DSP compilers.
  - Give little or no control to the programmer.
- A source-level solution would allow programmer interaction.
  - Creates some new issues!

---

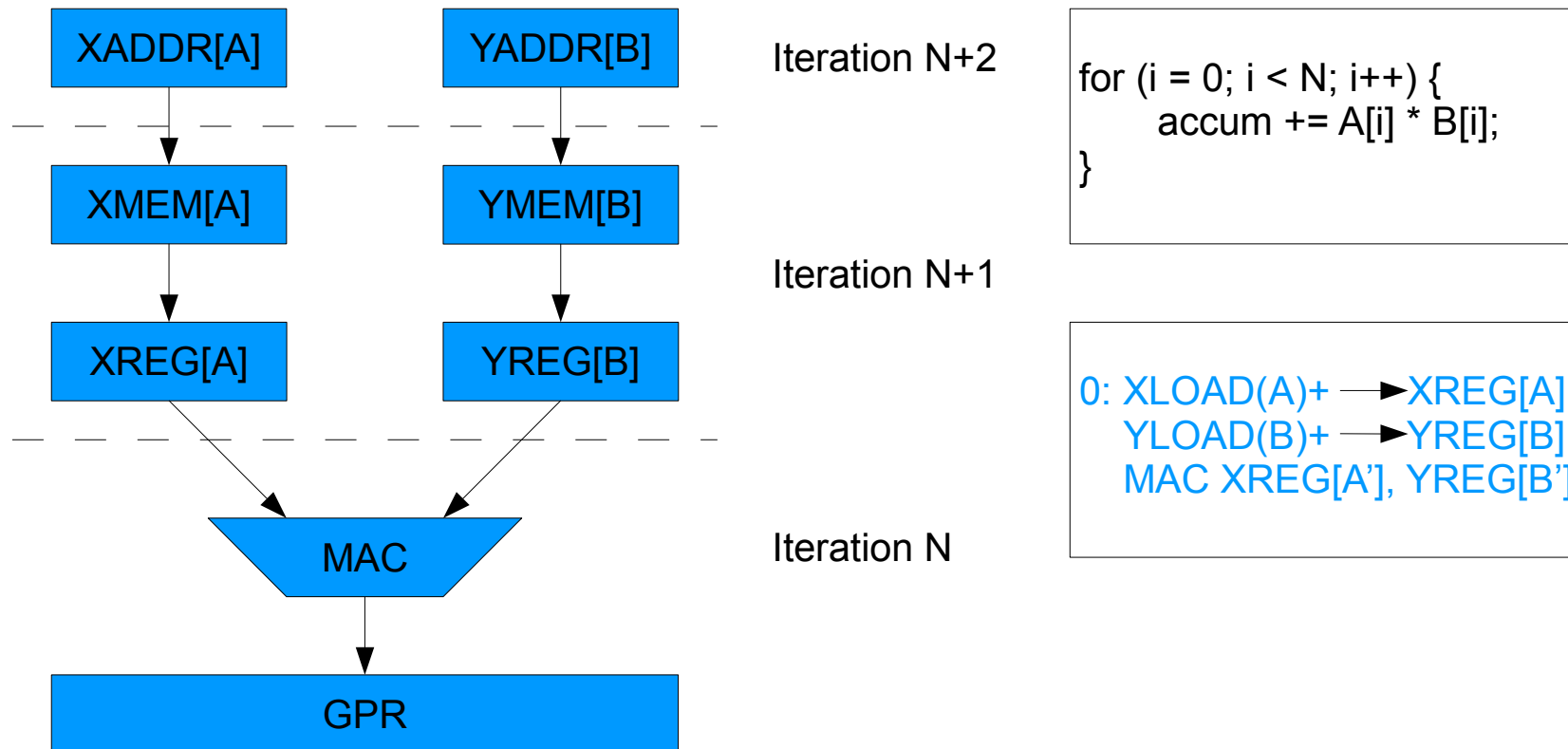
# Outline

- Dual memory bank architectures.
- Automatic assignment of variables to memory banks.
- Manual assignment of variables to memory banks.
- A new technique: Automatic assignment using DSP-C.
- A new colouring method: Soft Colouring.
- Empirical results.
- Summary.

# Dual Memory Bank Architectures



# Dual Memory Bank Architectures



---

# Automatic Assignment

- Existing solutions operate on compiler IR.
  - Mostly low-level.
  - Builds some form of interference graph.
  - Colours graph aiming to minimise interference.
- Leading existing solutions based on Integer Linear Programming.
- Seen little application in Industry.

---

# Manual Assignment

- Assign variables to memory banks via C language extensions.
  - DSP-C
  - Embedded C
- Supported by many DSP compilers.
- Syntax examples:

```
float X data[32];  
int X * Y pointer;
```

---

# Automatic DSP-C Assignment

- Our approach:
  - Create a C-to-DSP-C source-level transformation tool.
  - Have this tool perform full or partial assignment.
  - Portable to any DSP-C compiler.
- Allows programmers to place some variables manually.
- Requires additional analysis to ensure correctness:
  - Pointer aliasing.
  - Function parameters.

---

## Choosing an Assignment

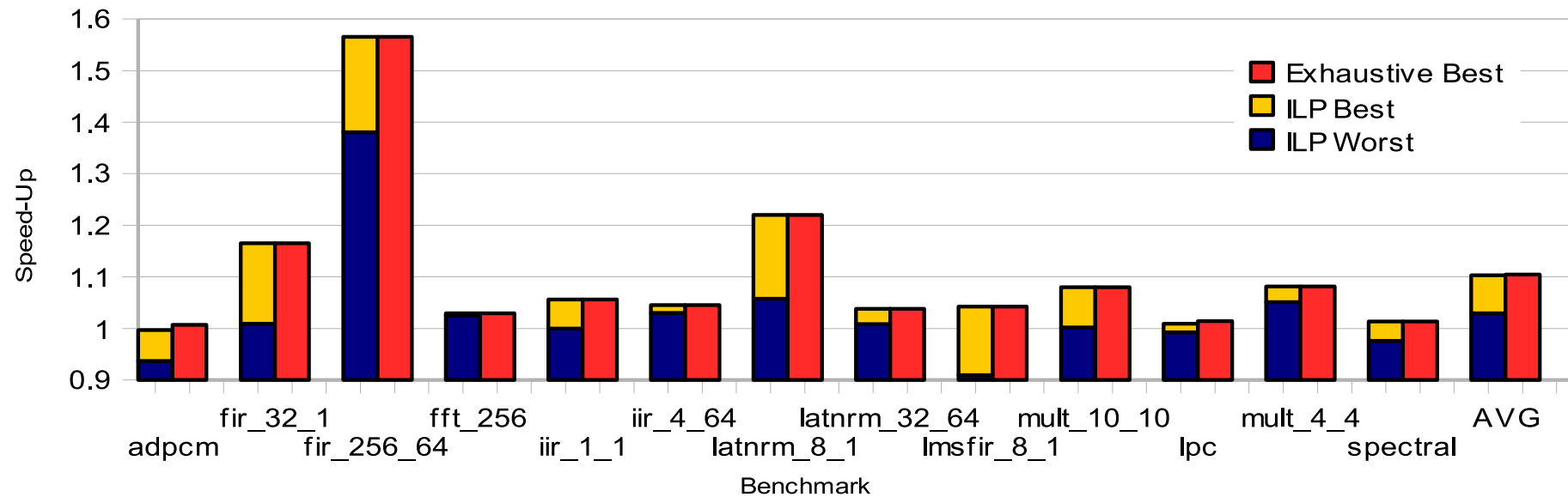
- Leading existing technique – Integer Linear Programming:
  - Gets good results.
  - Scalability issues.
  - Multiple equivalent solutions.
- Newly proposed technique – Soft Colouring:
  - Based on a distributed systems algorithm.
  - Was designed for ‘soft’ colouring constraints.
  - Not necessarily optimal.
  - Stochastic, so multiple solutions are a natural result.

---

# Soft Colouring Algorithm

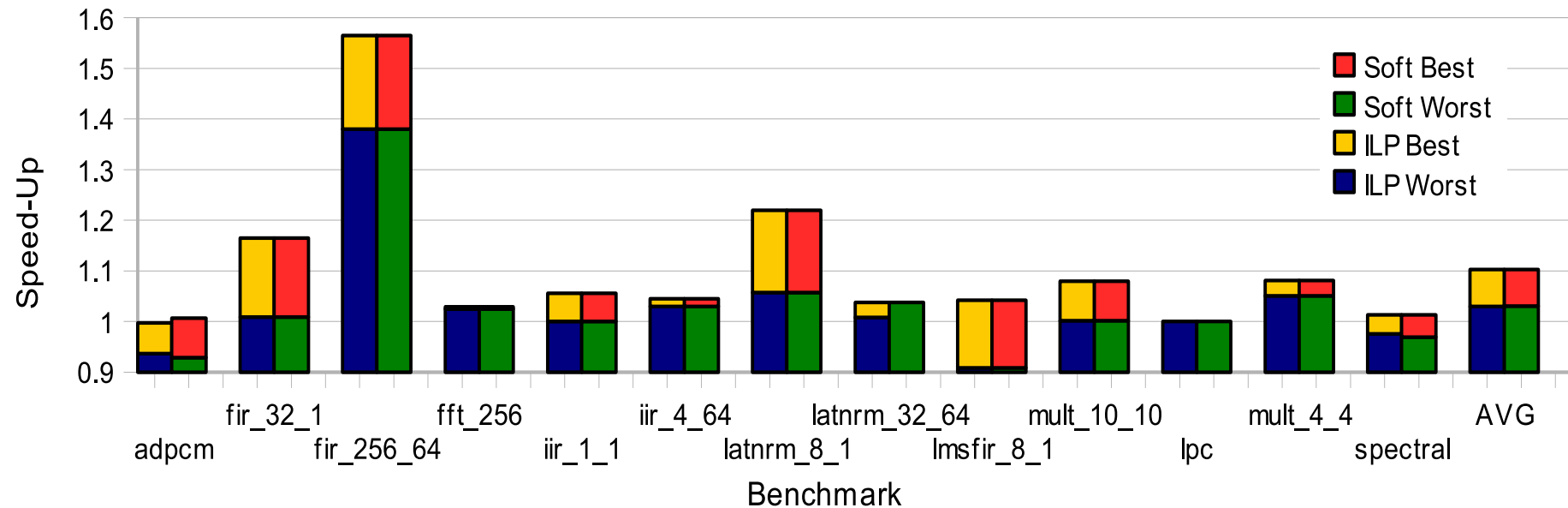
- Basic idea:
  1. Initialise every variable to a random colour.
  2. For each variable:
    3. Calculate locally optimal colour.
    4. 50% probability of changing to the locally optimal colour.
  5. While some nodes are still not locally optimal, iterate.
- Settles on some local maxima.

## Results - ILP vs Exhaustive



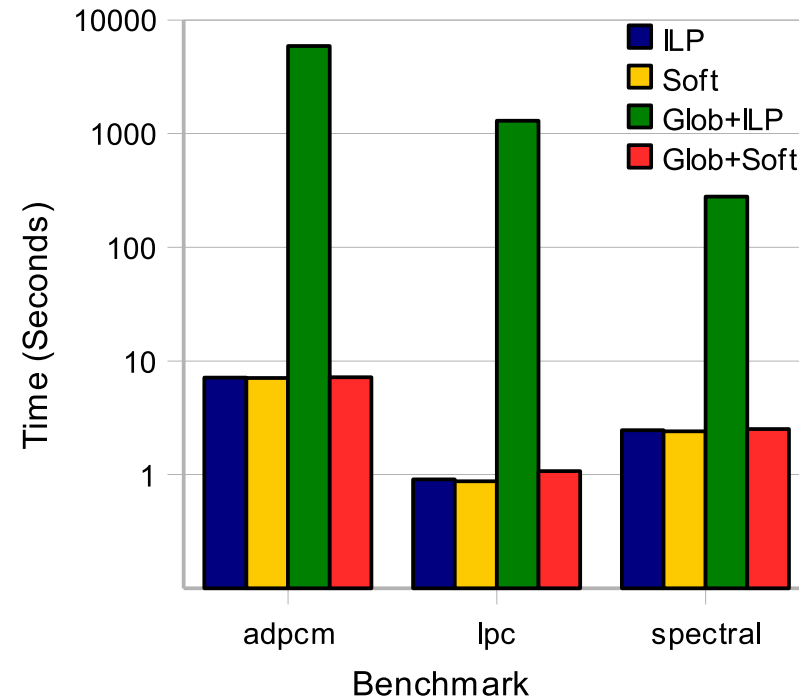
- Timings taken using an Analog Devices TigerSHARC TS-101.

## Results - ILP vs Soft Colouring



- Timings taken using an Analog Devices TigerSHARC TS-101.

## Results - Colouring Times



- Timings taken using an Intel Xeon 3GHz.

---

## Summary

- Future work:
  - Investigate different interference graph constructions to minimise range.
- Source-level assignment gives more control to the programmer.
- Easily portable.
- Leads to range of equivalent ILP solutions.
- Soft Colouring finds similar range of results but scales more efficiently.

---

**Any Questions...?**