# Limitations of Adaptable System Architectures for WCET Reduction

Jack Whitham

April 21st 2008

# WCET reduction

Usually, system architects aim to:

- Minimize *average case execution time* (ACET) of software: maximizing typical performance.

But hard real-time system architects aim to:

- Minimize *worst-case execution time* (WCET) of software: maximizing *guaranteed* performance.

# My topic

- WCET reduction of a program,
- preferably automatic,
- preferably using a conventional programming language.

# ACET vs WCET

ACET optimizations are relatively easy to implement:

- Performance analysis is simple: use profiling.
- Predictability isn't important provided that average performance is good.
- $\Rightarrow$ Heuristic mechanisms can be used.

# ACET vs WCET

WCET optimizations are relatively hard to implement:

- WCET analysis is tricky.
  - Must model the program.
  - Must model the CPU and system architecture.
- Predictability is important.
  - Predictability simplifies the models.
  - Predictability reduces *pessimism*.
- $\Rightarrow$ Heuristic mechanisms should be avoided.

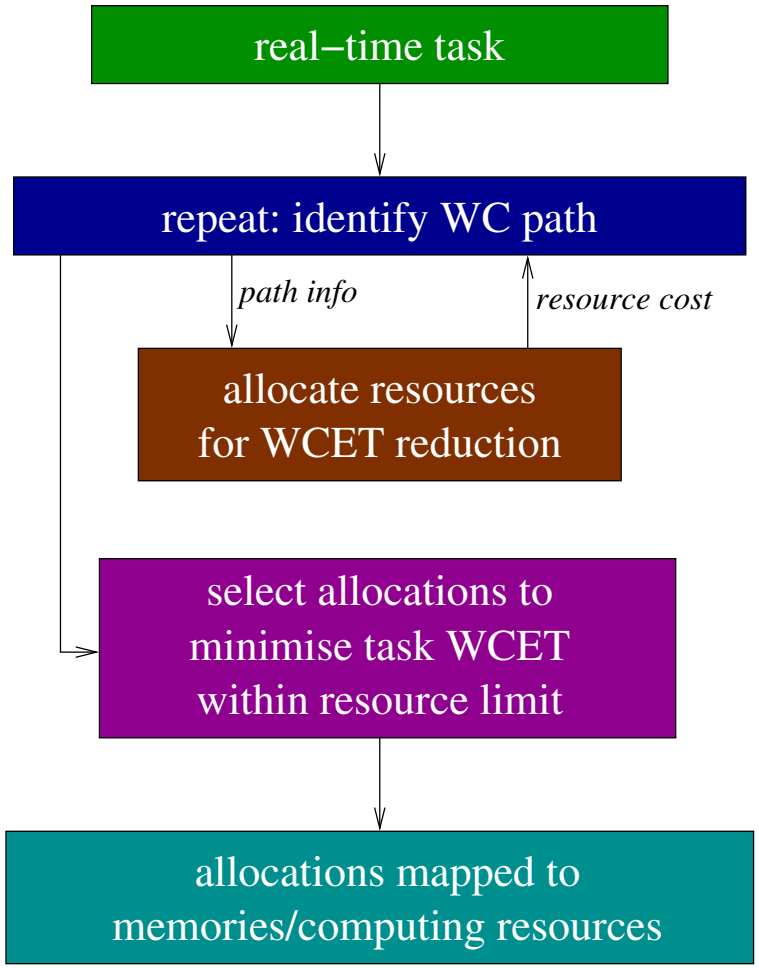
# CPU designs

Conventional CPU designs are good for ACET reduction, but not WCET reduction, because of:

- caches,
- superscalar out-of-order execution,
- branch prediction,
- generally, *clever but unpredictable techniques*.

All heuristic mechanisms! Analysis is possible but costly and pessimistic.

# Generalized WCET reduction process



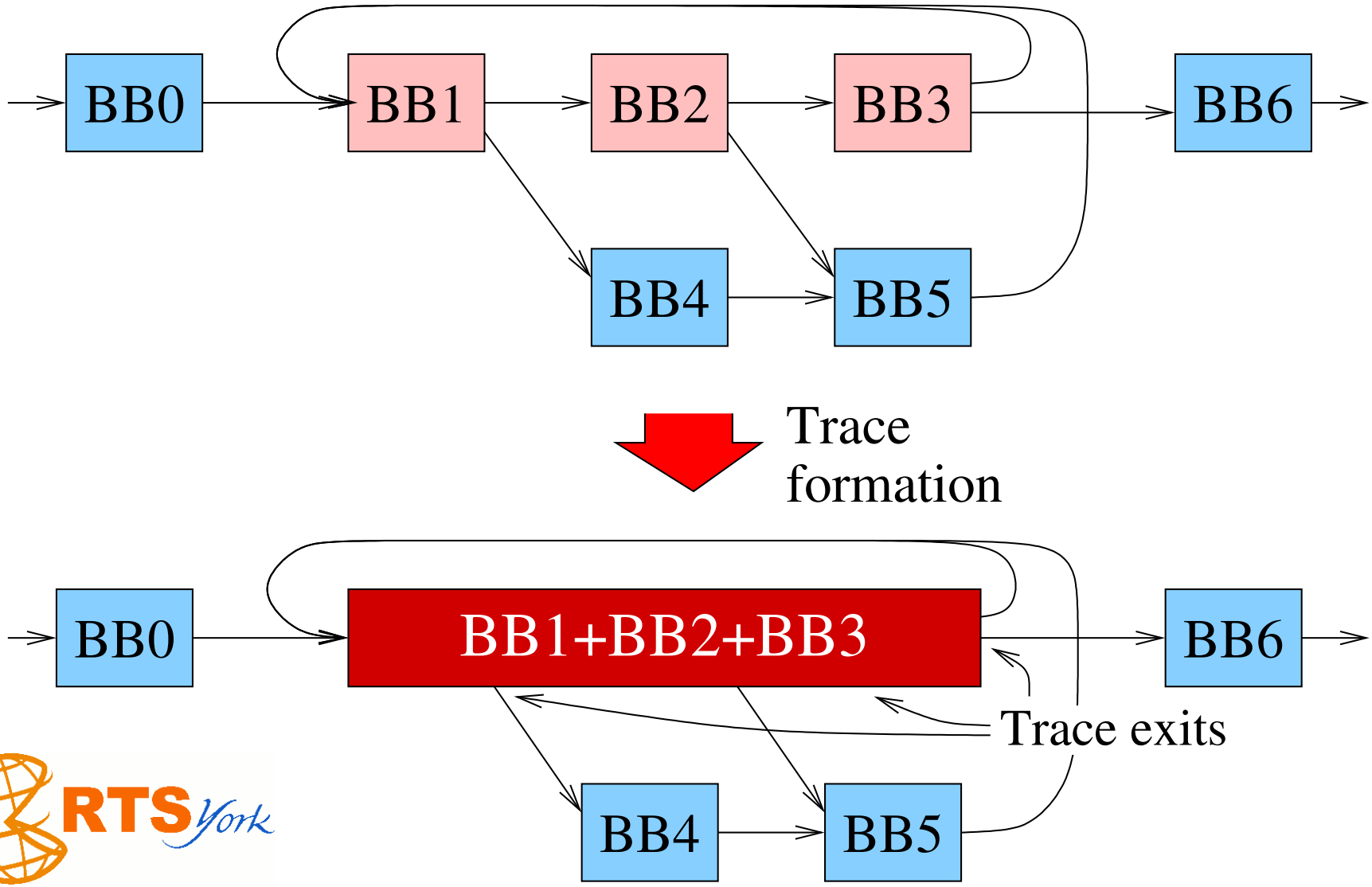
# Examples

*Adaptable and reconfigurable systems could implement predictable mechanisms to minimize WCET.*

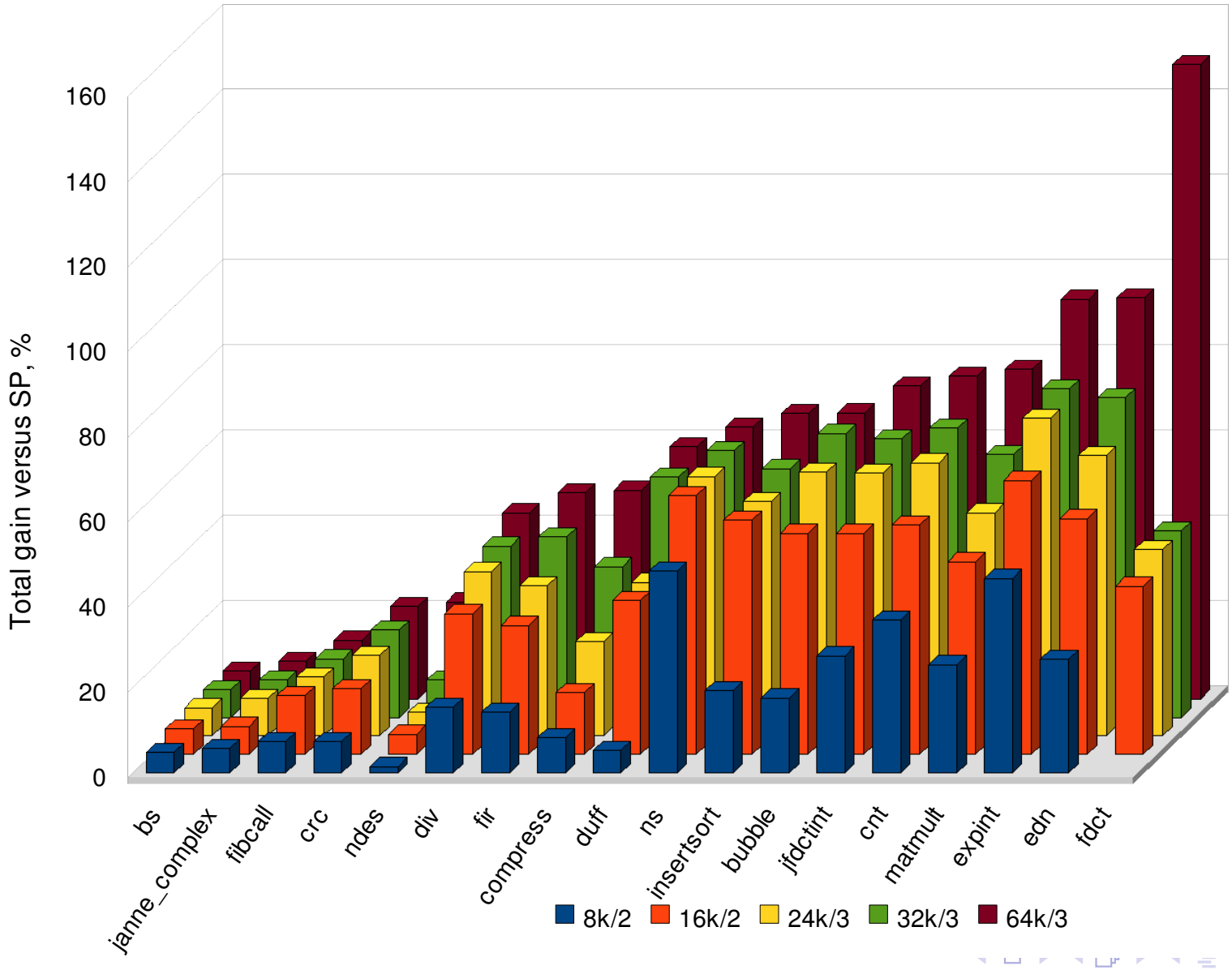
For example, code can be accelerated by:

- Co-processor modules, loaded by run-time reconfiguration.
- Scratchpad memory, loaded at run-time.
- Custom microprograms, loaded at run-time.

# Example: custom microprograms



# Results: custom microprograms



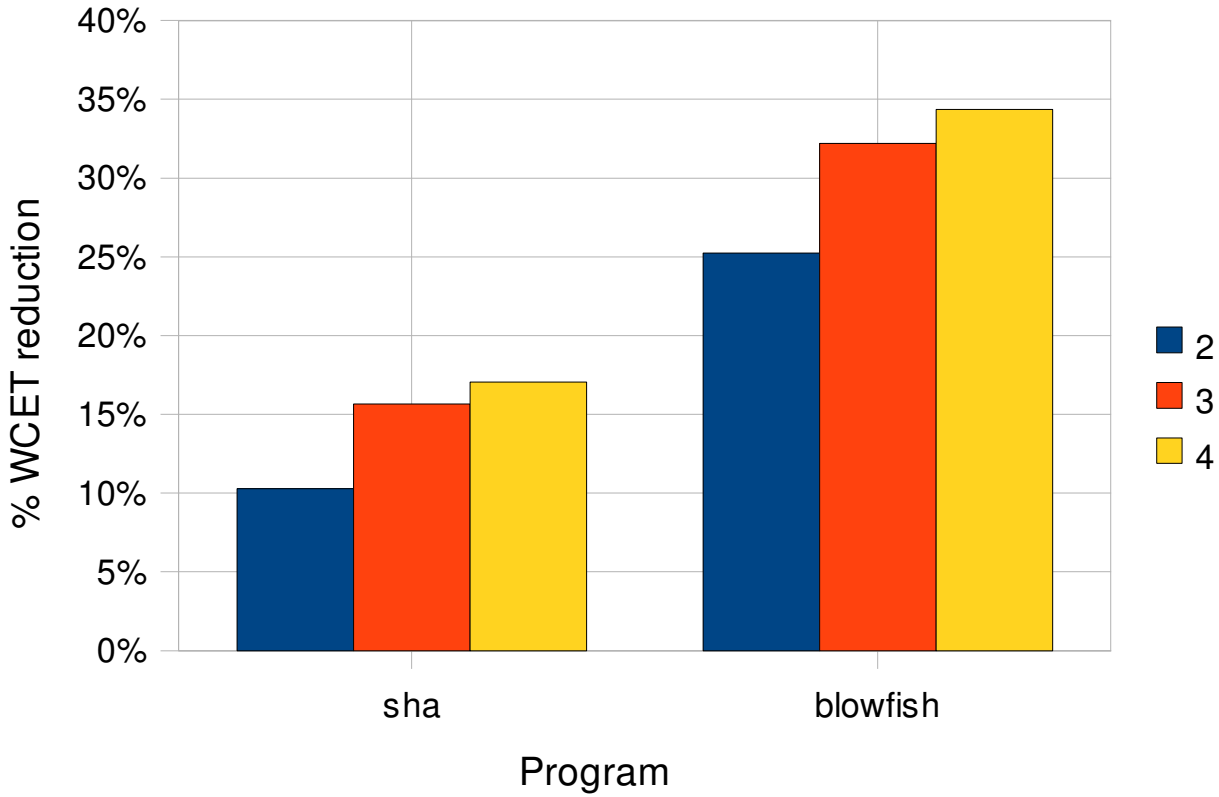
# General problems

All implementations will be subject to these limits:

1. *Instruction level parallelism* (ILP) limit.
2. Load cost limit.
3. General purpose architecture limit.

# ILP limit

Applies if you want to reduce the WCET of code written in a conventional programming language (e.g. C).



# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
2. Data dependences introduced by the compiler.
3. Data dependences introduced by the problem requirements.

# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
   - Addressed by dynamic speculation.
2. Data dependences introduced by the compiler.
3. Data dependences introduced by the problem requirements.

# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
   - Addressed by dynamic speculation.
   - Addressed by static speculation.
2. Data dependences introduced by the compiler.

3. Data dependences introduced by the problem requirements.

# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
   - Addressed by dynamic speculation.
   - Addressed by static speculation.
2. Data dependences introduced by the compiler.
   - Partly addressed by memory speculation and register renaming.
3. Data dependences introduced by the problem requirements.

# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
   - Addressed by dynamic speculation.
   - Addressed by static speculation.
2. Data dependences introduced by the compiler.
   - Partly addressed by memory speculation and register renaming.
   - Real solution: improved programming languages (e.g. support for vectorisation).
3. Data dependences introduced by the problem requirements.

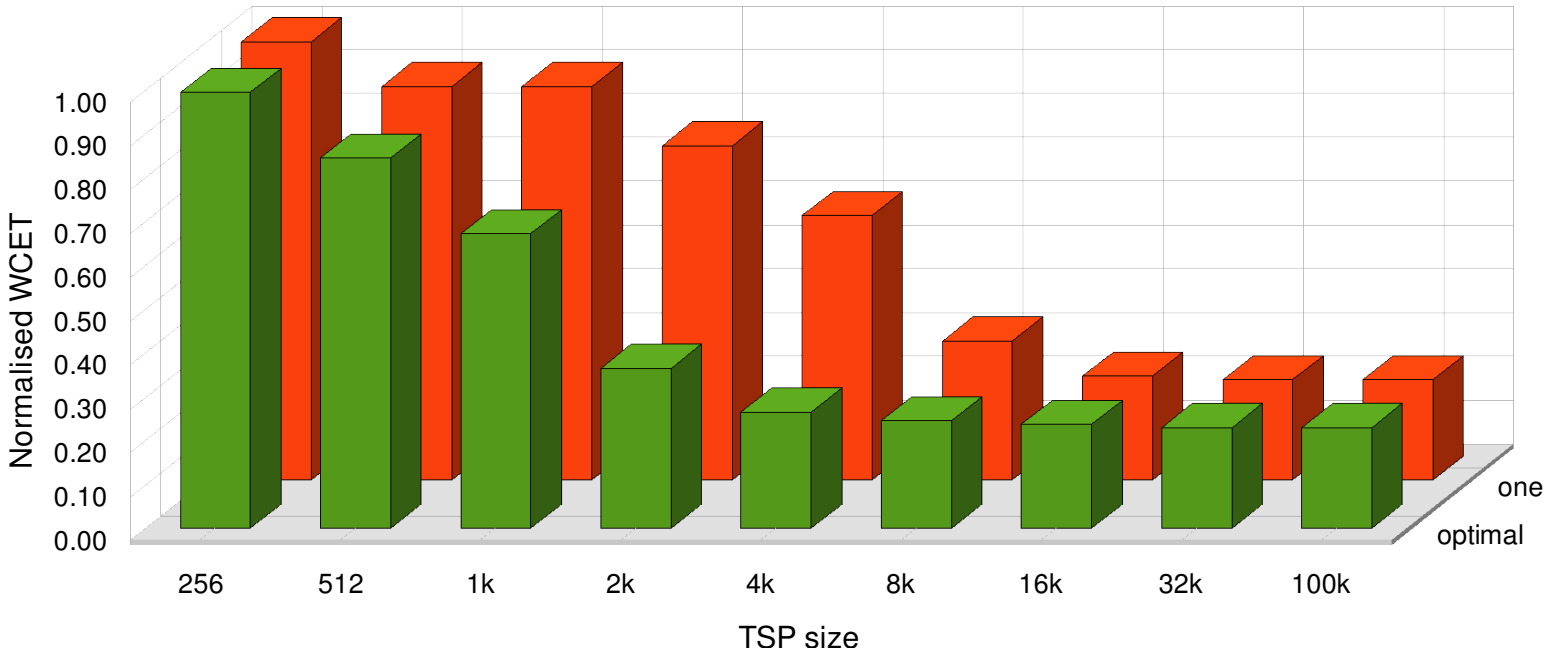
# ILP: Problems and Solutions

Caused by:

1. *Control flow* (branches).
   - Addressed by dynamic speculation.
   - Addressed by static speculation.
2. Data dependences introduced by the compiler.
   - Partly addressed by memory speculation and register renaming.
   - Real solution: improved programming languages (e.g. support for vectorisation).
3. Data dependences introduced by the problem requirements.
   - Unavoidable.

# Load cost limit

Applies if you want to load instructions (or data) into a scratchpad (or FPGA). Necessary to make best use of limited on-chip memory.



# Load cost: Problems and Solutions

Caused by:

1. Limited space in on-chip memory.

2. Cost of transferring data.

# Load cost: Problems and Solutions

Caused by:

1. Limited space in on-chip memory.
   - Addressed by dynamic loading.
2. Cost of transferring data.

# Load cost: Problems and Solutions

Caused by:

1. Limited space in on-chip memory.
   - Addressed by dynamic loading.
   - Addressed by static loading (overlaying).
2. Cost of transferring data.

# Load cost: Problems and Solutions

Caused by:

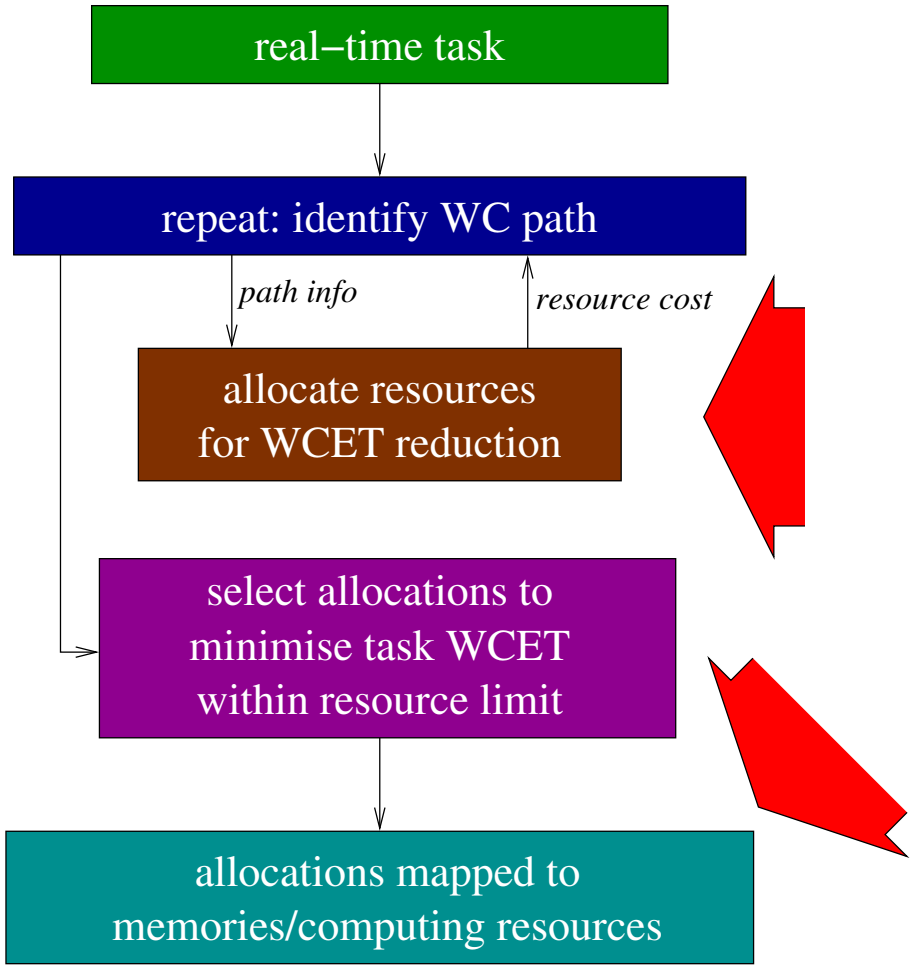
1. Limited space in on-chip memory.
   - Addressed by dynamic loading.
   - Addressed by static loading (overlaying).
2. Cost of transferring data.
   - Addressed by burst transfers.

# Load cost: Problems and Solutions

Caused by:

1. Limited space in on-chip memory.
   - Addressed by dynamic loading.
   - Addressed by static loading (overlaying).
2. Cost of transferring data.
   - Addressed by burst transfers.
   - Addressed by compression.

# General purpose architecture limit



# General purpose architecture limit

A choice: either,

- Write programs in a conventional programming language for a general purpose architecture,

or,

- Write programs that use application-specific hardware.

# General purpose architecture limit

A choice: either,

- Write programs in a conventional programming language for a general purpose architecture,
  - Limited by ILP.

or,

- Write programs that use application-specific hardware.

# General purpose architecture limit

A choice: either,

- Write programs in a conventional programming language for a general purpose architecture,
  - Limited by ILP.

or,

- Write programs that use application-specific hardware.
  - WCET reduction search is difficult (co-design).

# General purpose architecture limit

A choice: either,

- Write programs in a conventional programming language for a general purpose architecture,
  - Limited by ILP.

or,

- Write programs that use application-specific hardware.
  - WCET reduction search is difficult (co-design).
  - Manual hardware design may be required.

# Conclusion

For WCET reduction, tradeoffs exist between:

1. Conventional languages versus specialist languages.
2. Loading costs versus on-chip memory sizes.
3. General-purpose versus application specific architectures.

# Conclusion

For WCET reduction, tradeoffs exist between:

1. Conventional languages versus specialist languages.
2. Loading costs versus on-chip memory sizes.
3. General-purpose versus application specific architectures.

- My own work has explored the first two.

# Conclusion

For WCET reduction, tradeoffs exist between:

1. Conventional languages versus specialist languages.
2. Loading costs versus on-chip memory sizes.
3. General-purpose versus application specific architectures.

- My own work has explored the first two.
- There is plenty of scope for future work.

# Conclusion

For WCET reduction, tradeoffs exist between:

1. Conventional languages versus specialist languages.
2. Loading costs versus on-chip memory sizes.
3. General-purpose versus application specific architectures.

- My own work has explored the first two.
- There is plenty of scope for future work.
- Questions?