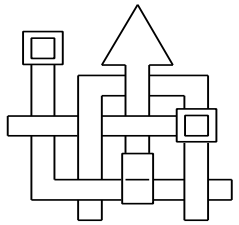
Proposal for unified system design meta flow in task-level and instruction-level design technology research for multi-media applications

Francky Catthoor, Diederik Verkest, Erik Brockmeyer
IMEC, Leuven, Belgium K.U.Leuven



IMEC

Acropolis

Work partly sponsored by the MEDEA-IWT project SMT.

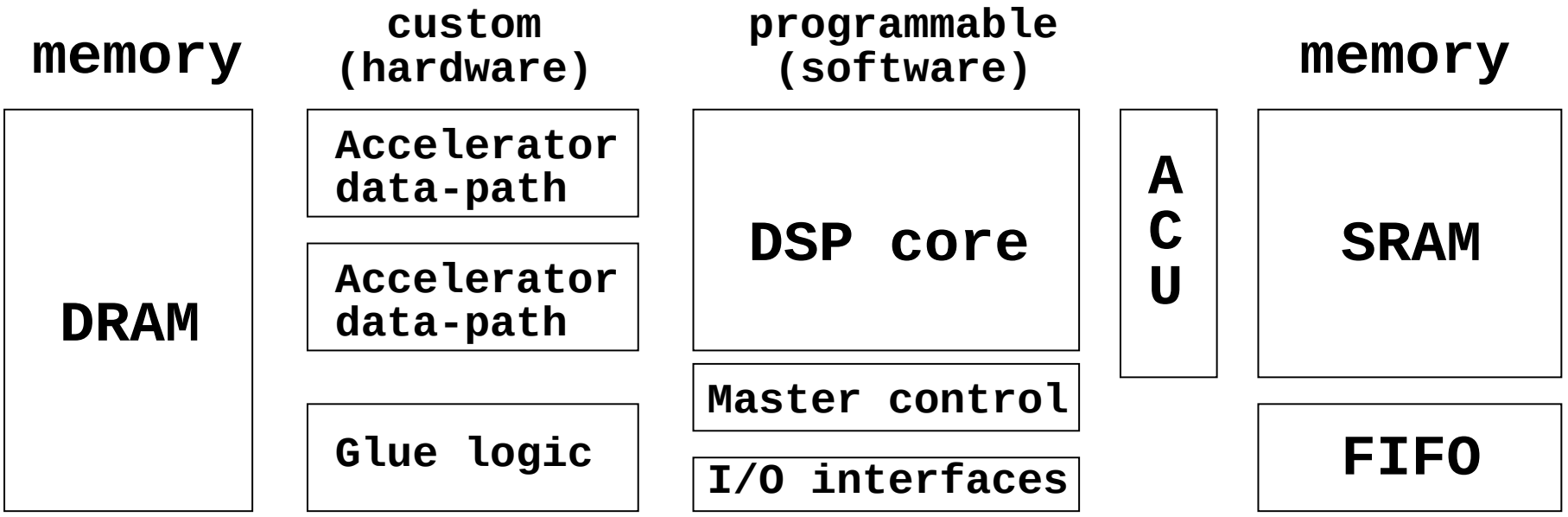
Outline

1. Context and objectives

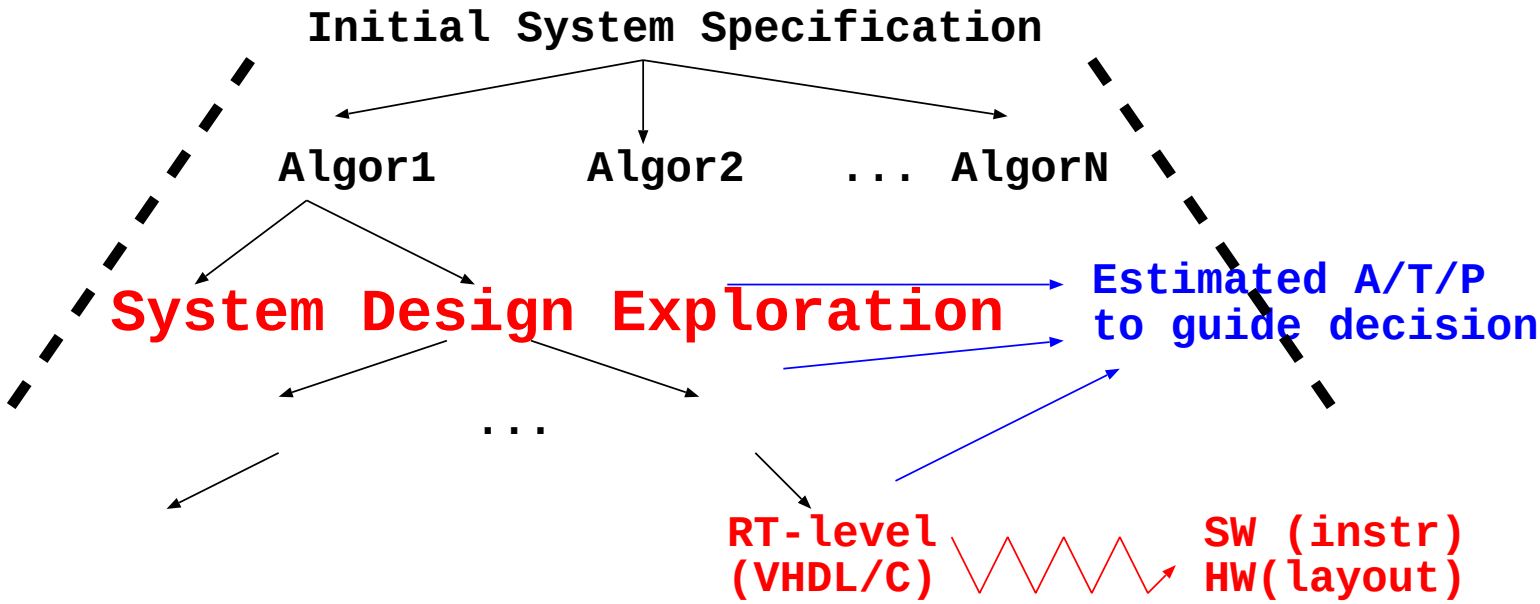
2. Illustration of major principles

3. Conclusions and perspective

Heterogeneous System Architecture



System Design Exploration: context



Target domain: subsystems in end-user telecom, multi-media processing and network components

Target users: system and architecture designers + tool support developers

Objectives and illustration

- Common terminology: in use at IMEC since begin 1997
- Simpler identification of related work and comparison: "ref library"
- Framework to identify "holes" in flow: interesting Ph.D. topics
- Remove redundancy and unnecessary overlap: inter-group cooperation i.s.o. competition!
- Reusable software motivation (both tools and kernels): e.g. Atomium kernel reuse at several DTSE levels at IMEC

Characteristics

1. Meta-flow from which many others can be derived

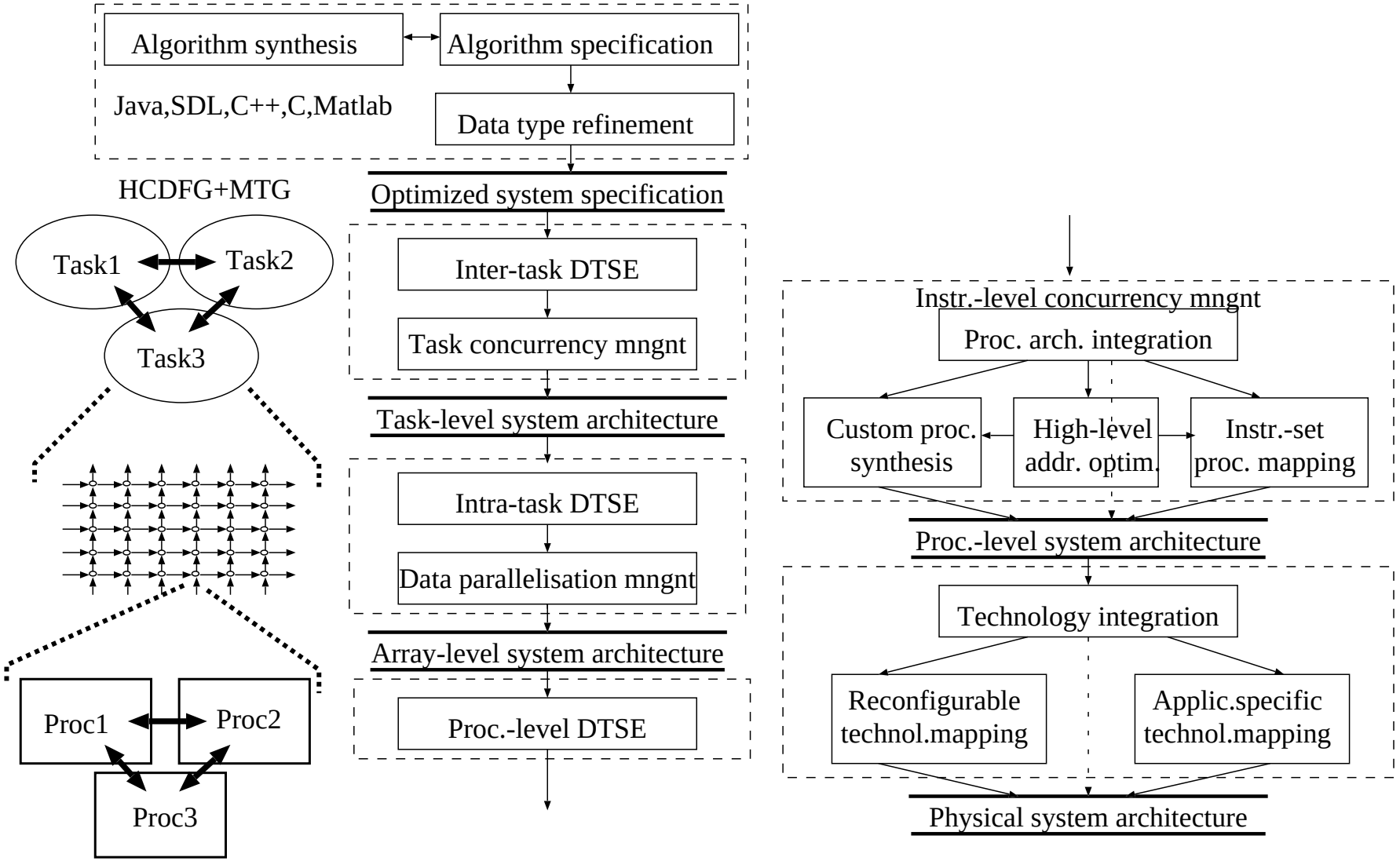
2. Independent of type of system design, as long as within electronic context

3. NOT single detailed design flow for target application domain (where coupling still exists)

4. In instantiated flows, the dependencies of the meta-flow should be obeyed but clustering and/or skipping possible

5. NOT tool flow nor incorporation of design models (e.g. data-flow vs control-flow)

Global meta flow and models



Main principles

1. distinction between *target technology platform* and *target architectural style*

2. independent choice at technology platform and instruction/processor level (no SW vs HW)

3. algorithm design, task-level concurrency, data parallelisation and instruction-level concurrency levels

4. at each concurrency level, both data transfer and storage (DTSE) issues and concurrency management issues

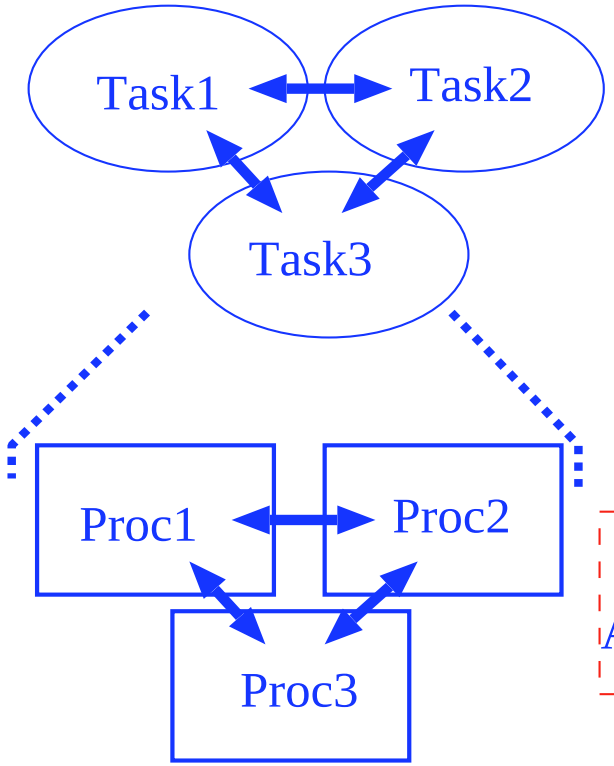
5. each subtasks incorporates constraints from higher in script and imposes extra constraints to lower subtasks (design reuse propagation)

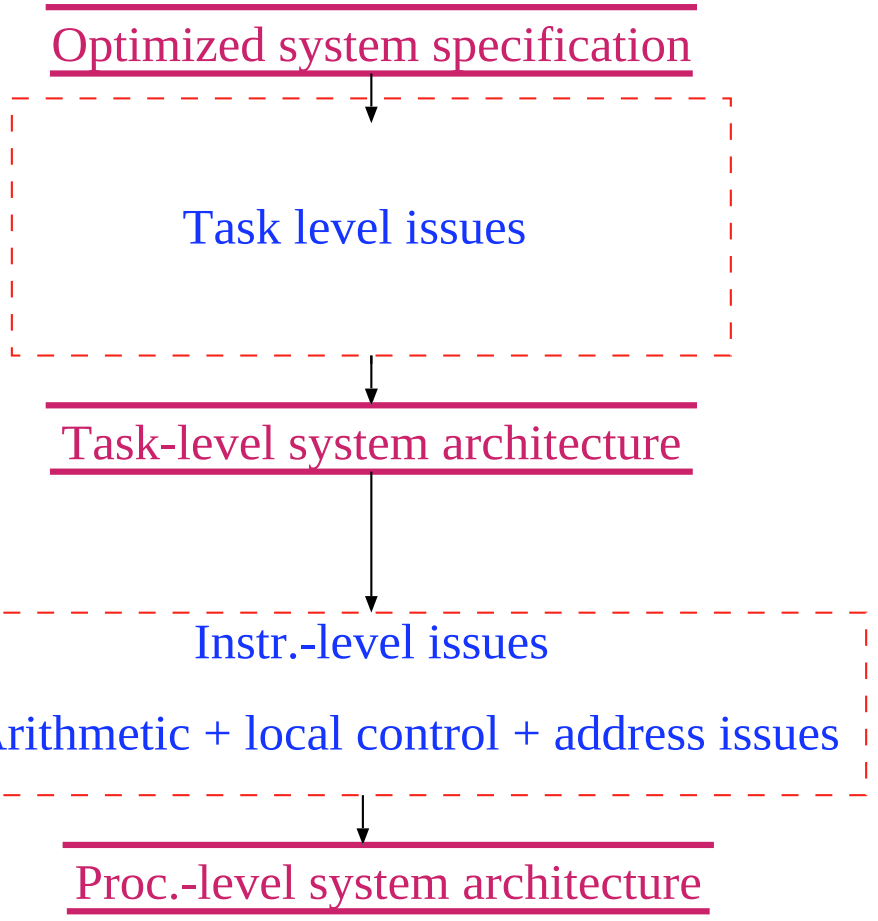
6. each subtask, typically uses *high-level estimator*s to predict

Task vs Instr level issues

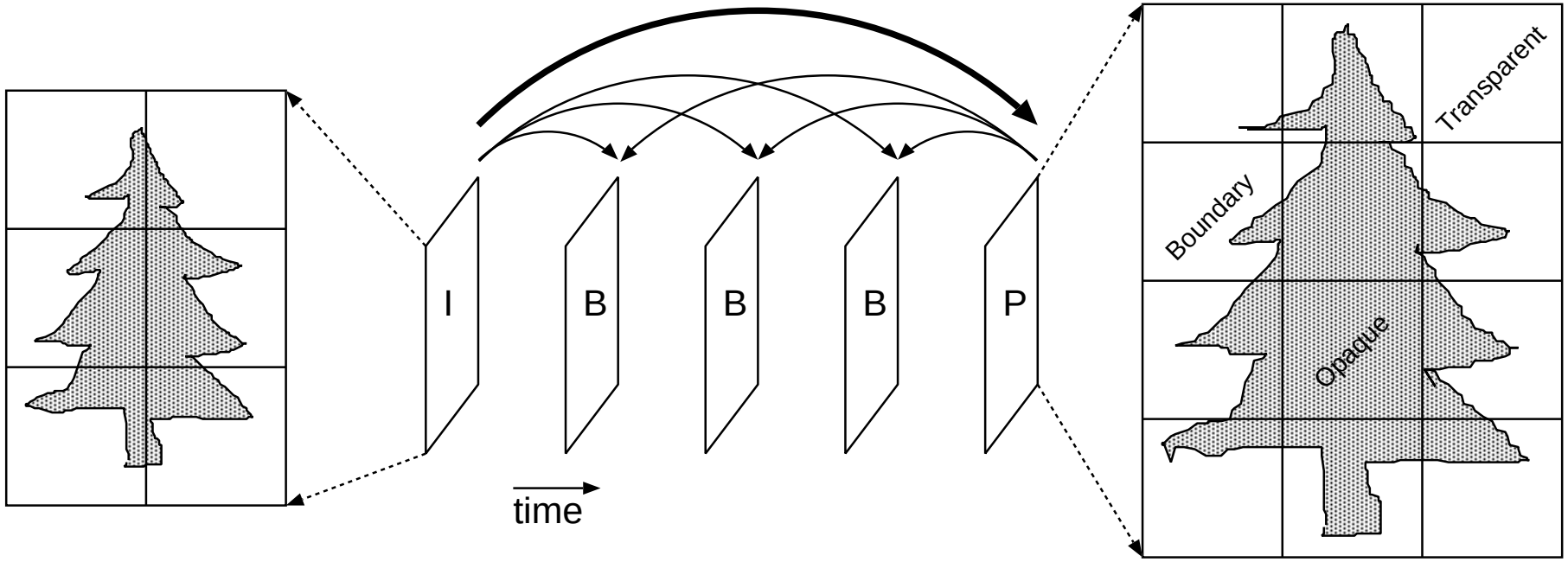
Most costly signals shared/transfered between tasks: fixed
Most costly signals/ dependencies inside tasks: incorp.

All data inside heterogeneous processor scope





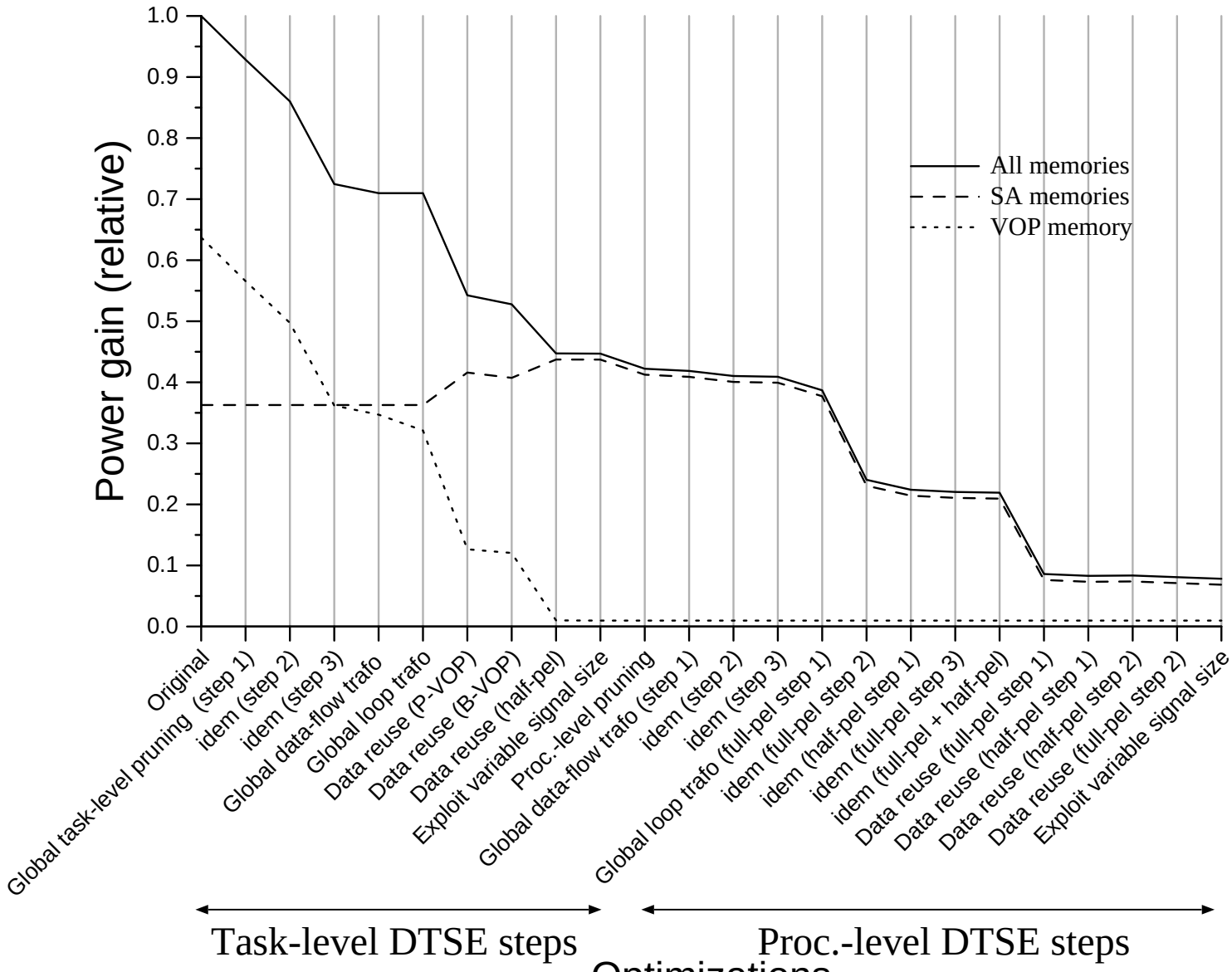
Task-level DTSE illustration: MPEG4



→MPEG-4 sequence, content based object coding

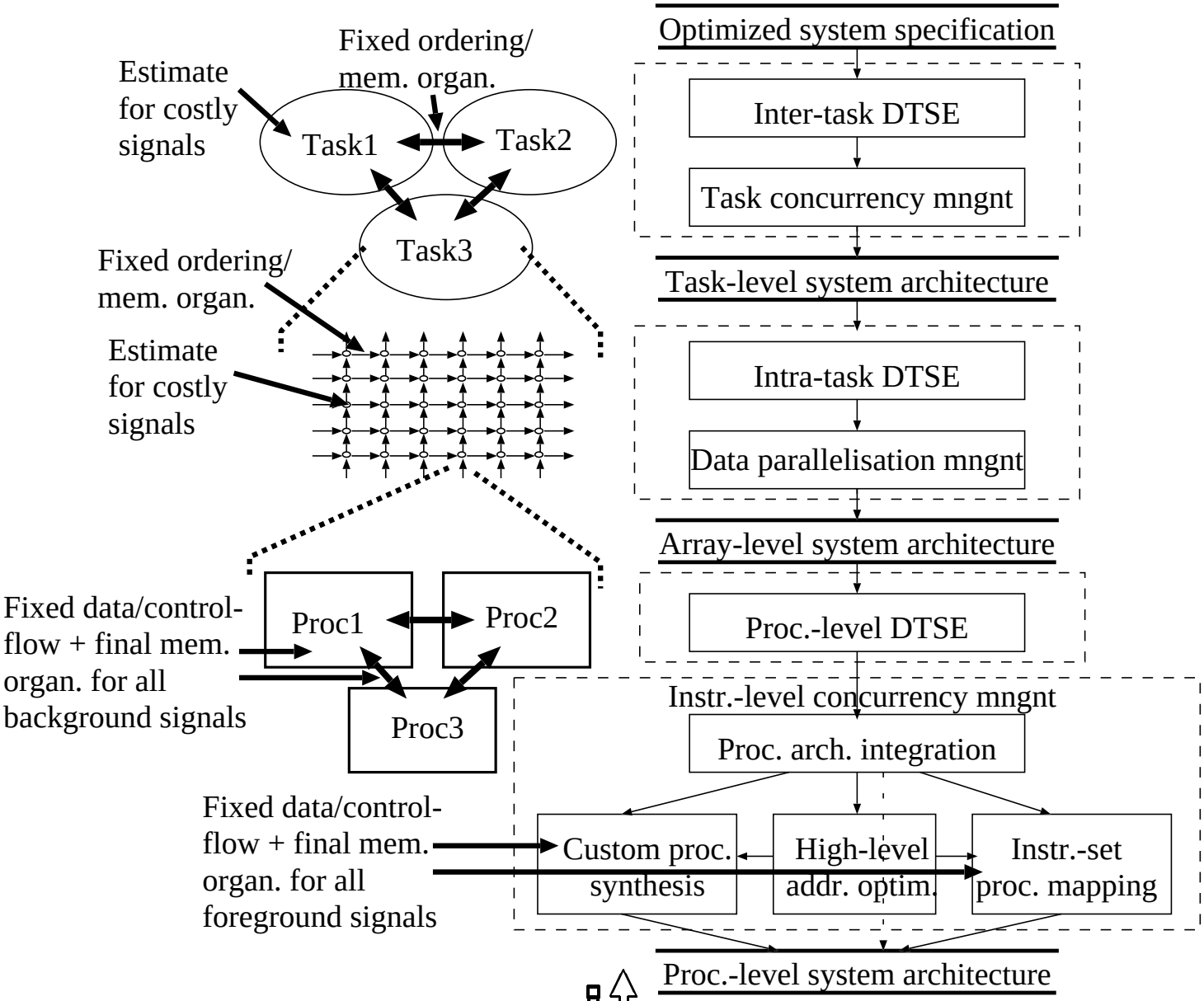
Video Object Plane (VOP) size and position can change during a sequence

MPEG4 total power gain (VOP and SA): measured

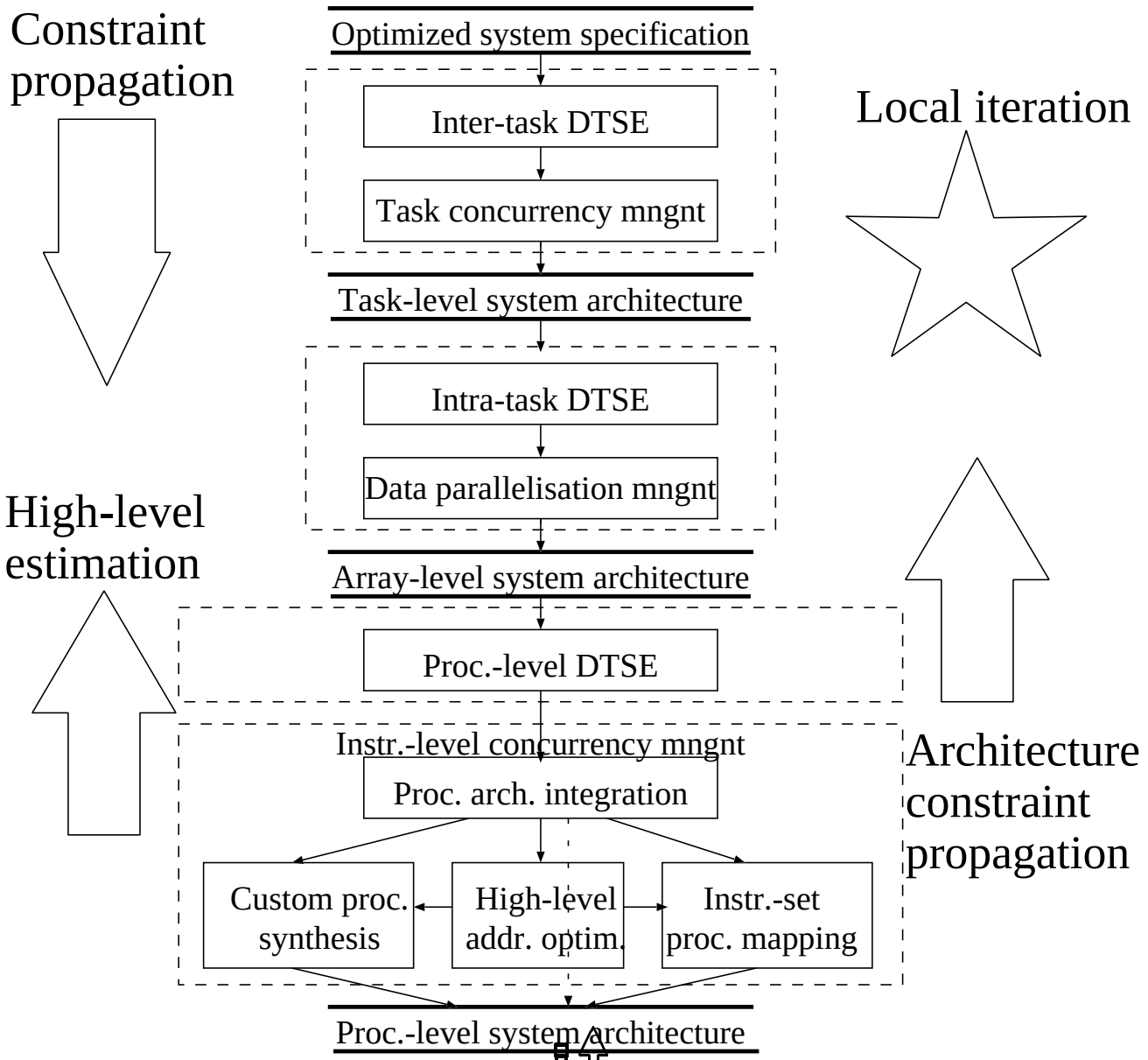


Optimizations

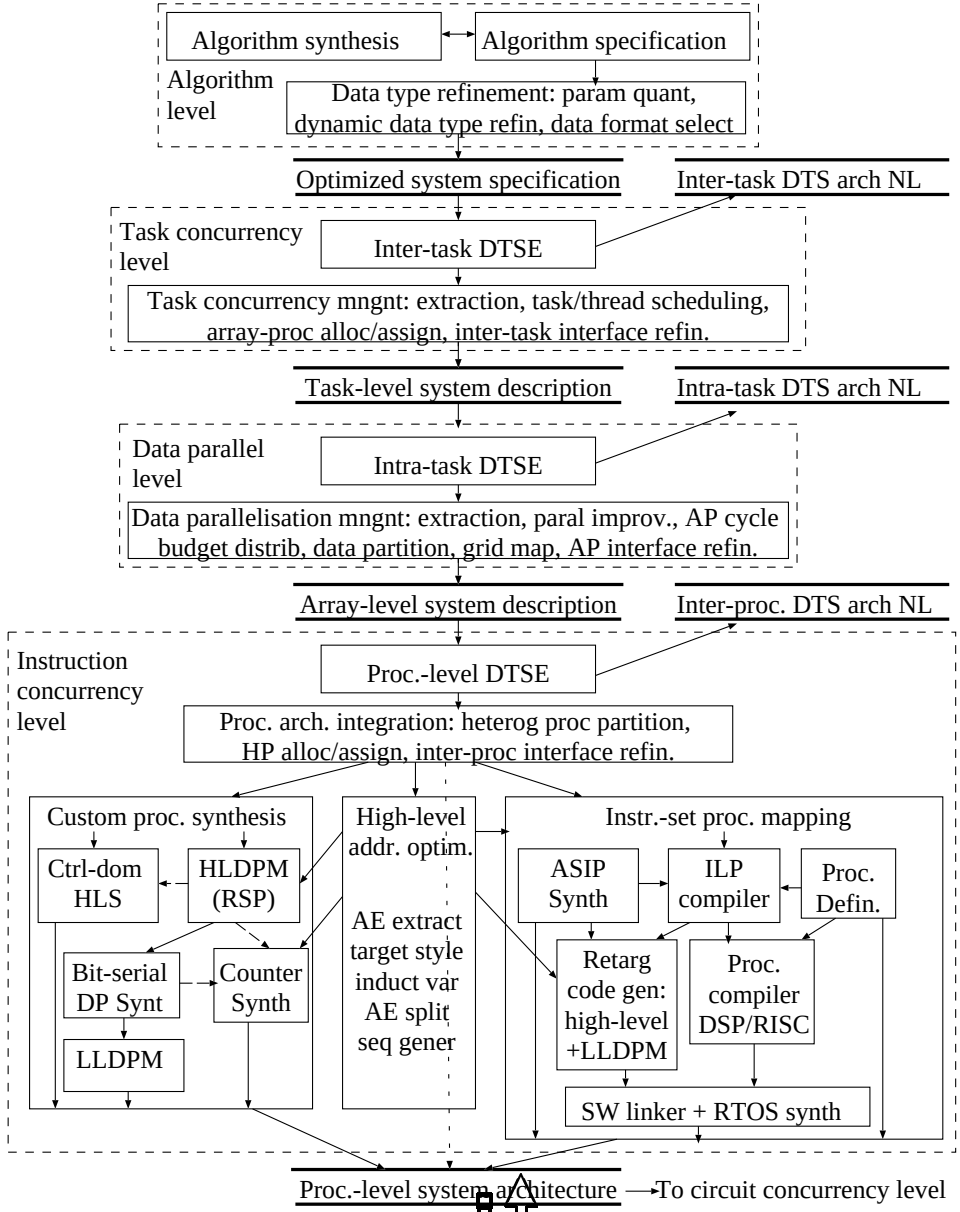
DTSE vs concurrency management issues



Iteration, constraint propagation and estimators



Global meta flow at highest layers



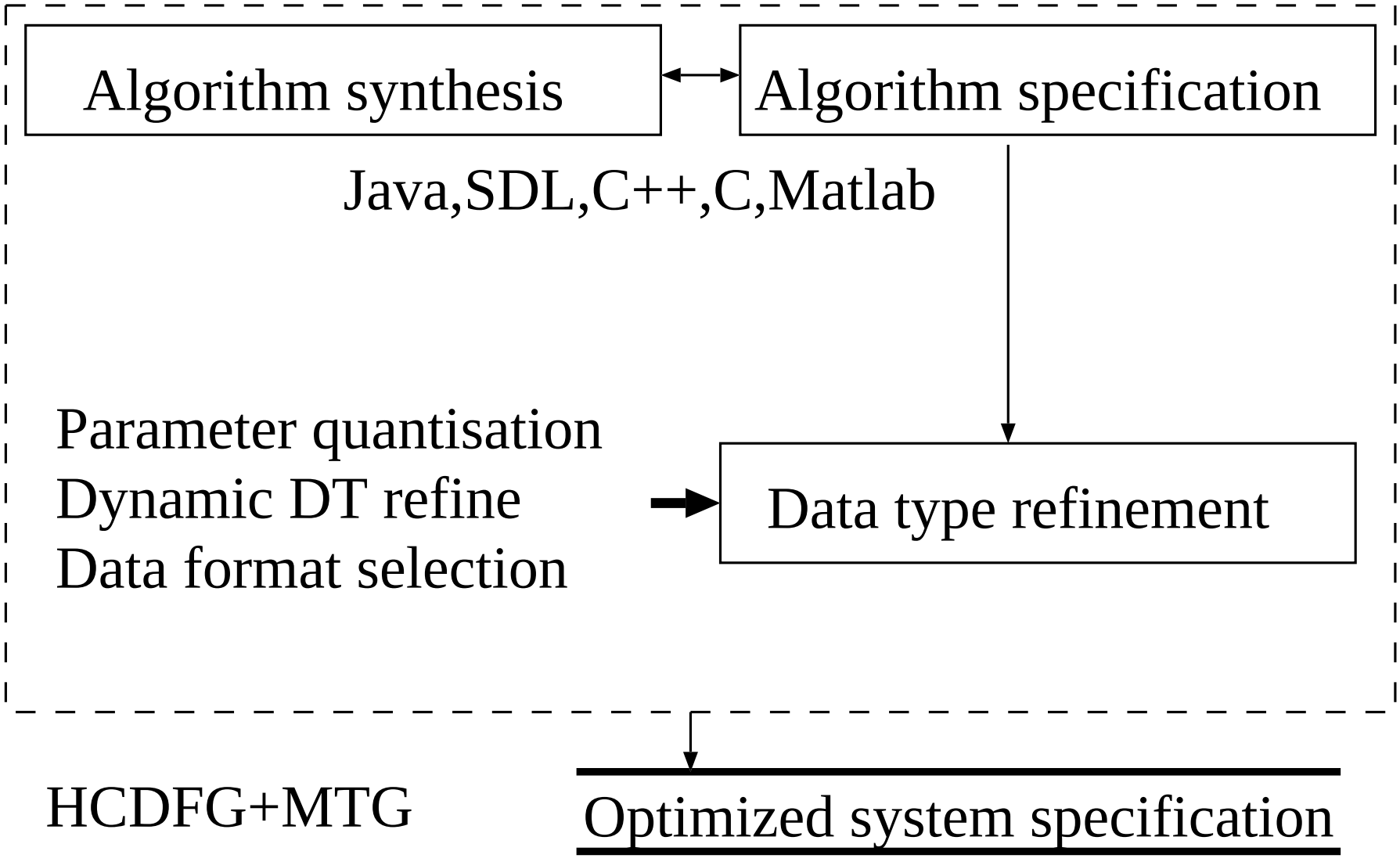
Overall conclusions on use of unified meta flow

Appears to be possible to unify known/published system design flows in electronic context in single overall "framework" which is surprisingly simple but which is still not trivial.

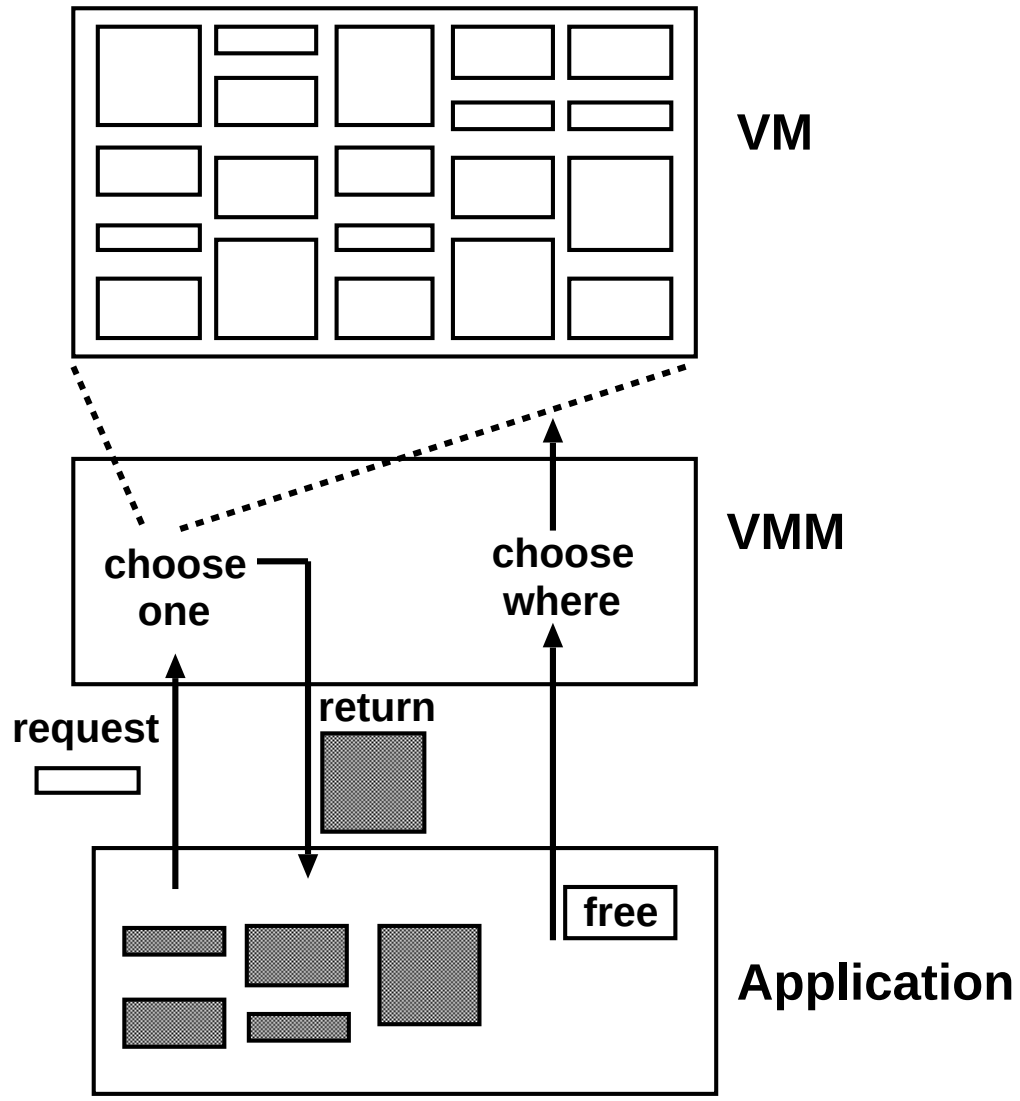
Historical distinction between "hardware" and "software" is leading to artificial and highly inefficient boundaries between different design solutions

Separation between different abstraction level: algorithm, task concurrency, data parallelism, processor concurrency, custom vs instruction-set processor and technology platform (circuit level) allows clearer and more decoupled view on crucial and complex design issues.

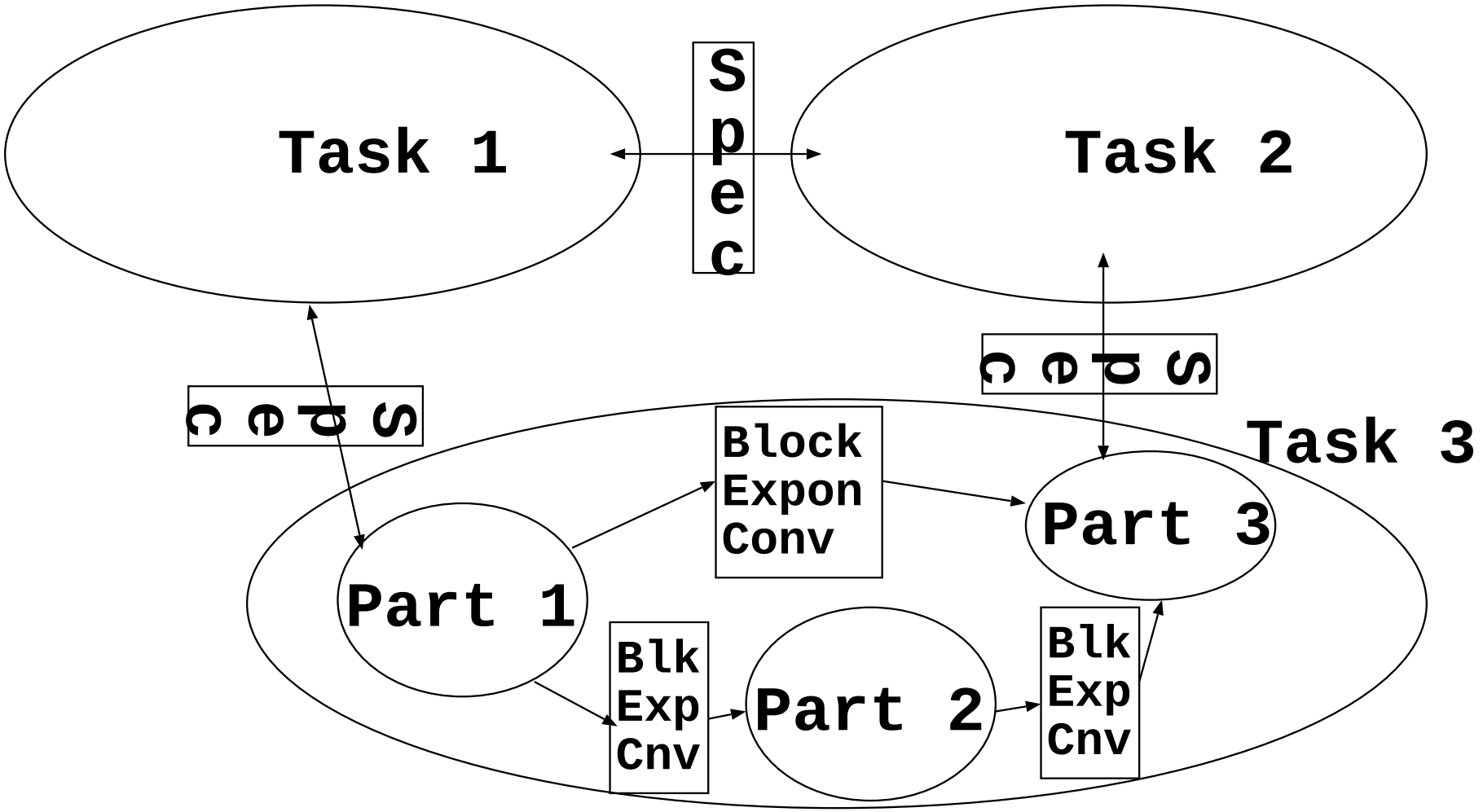
Algorithmic issues



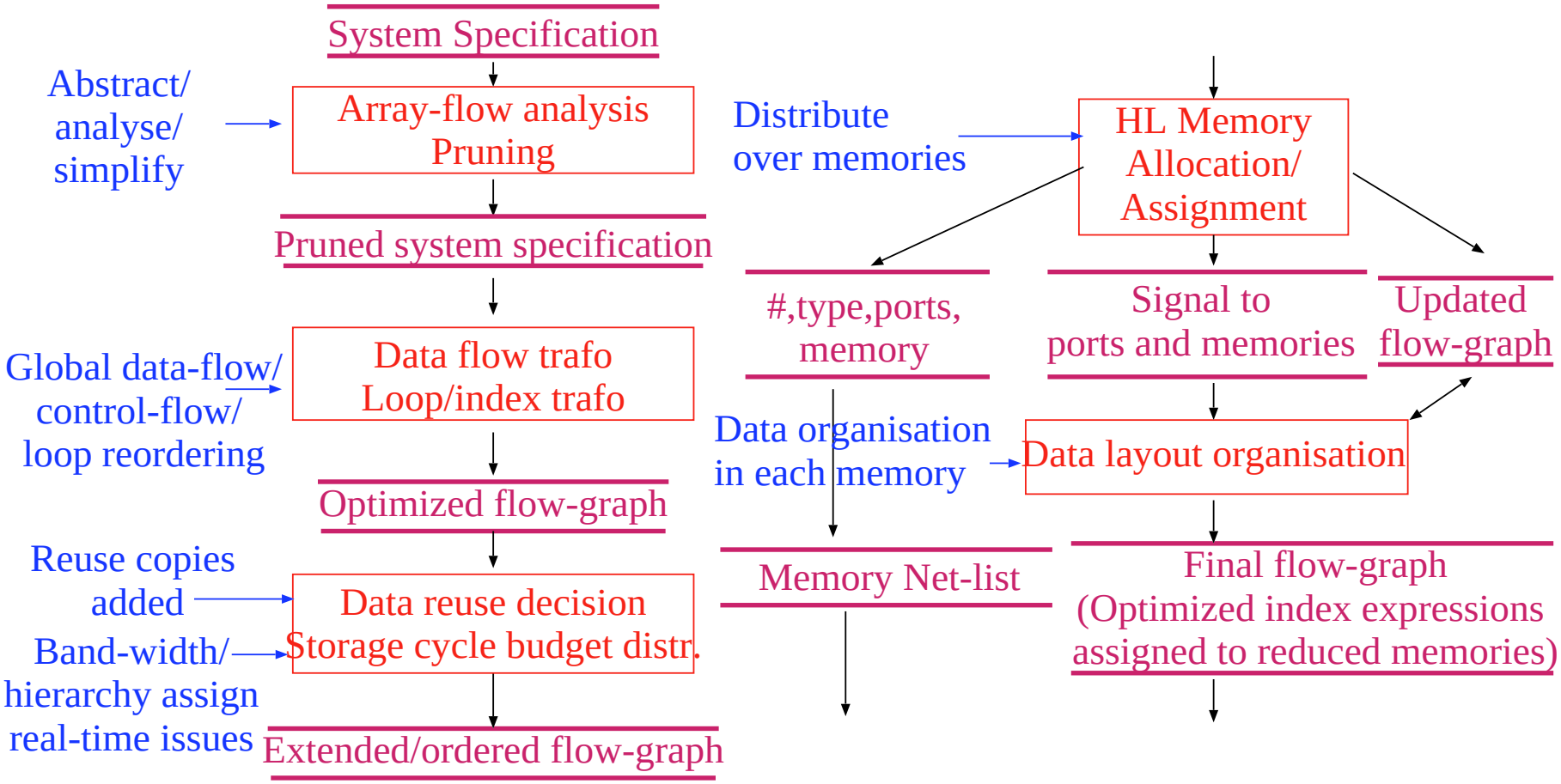
Virtual Memory Management (VMM)



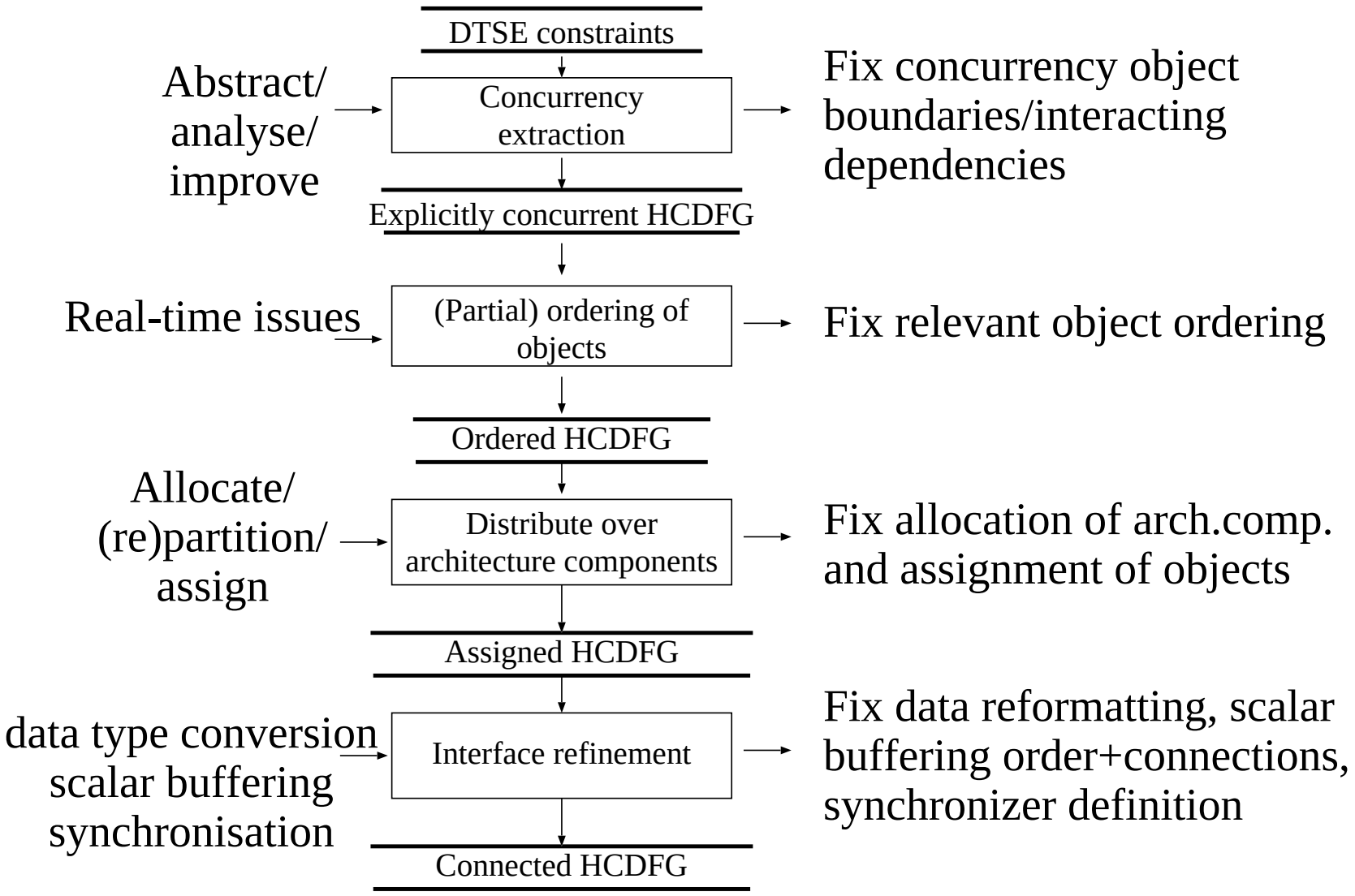
Static data type refinement: global exploration



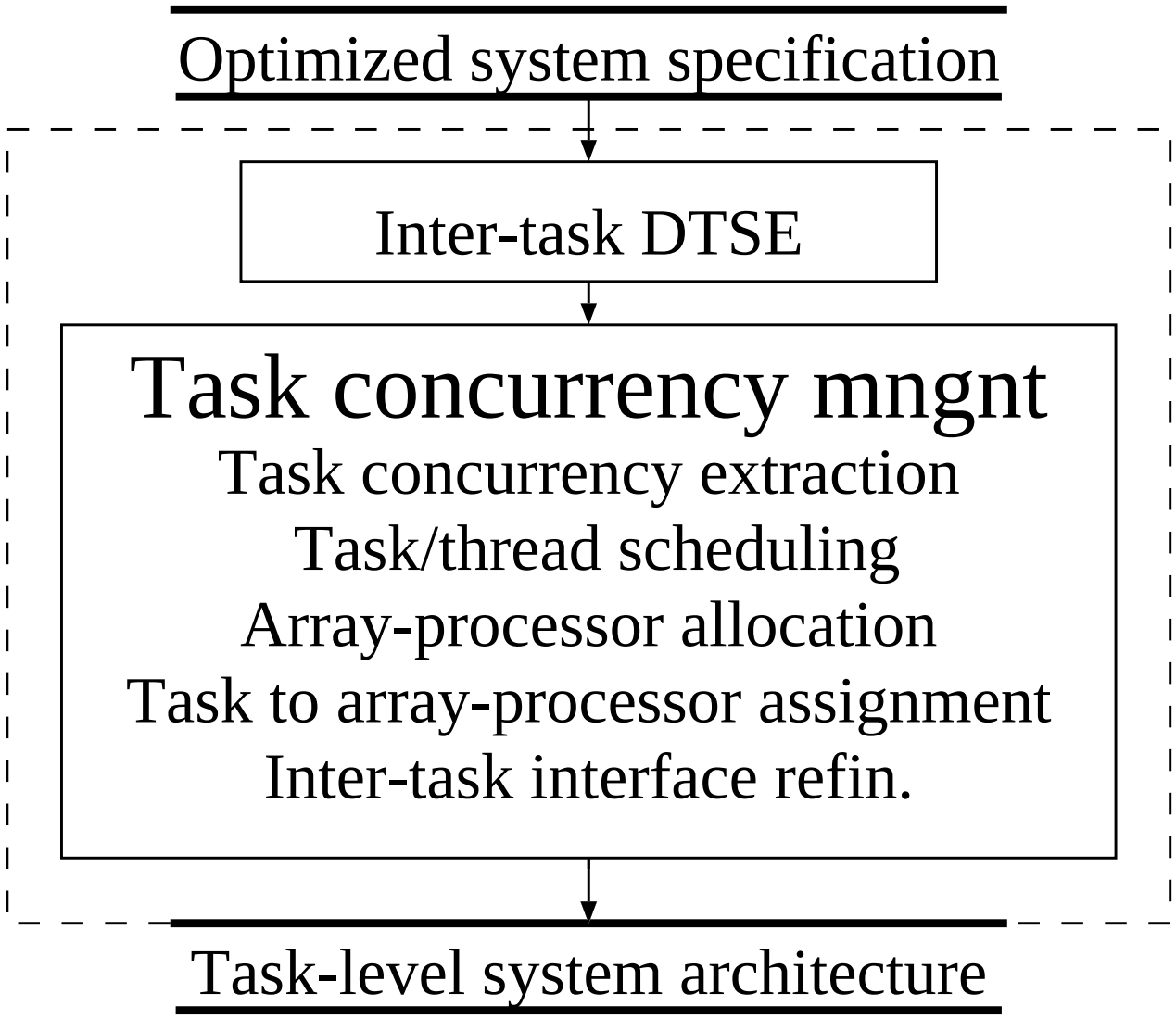
DTSE issues and constraints



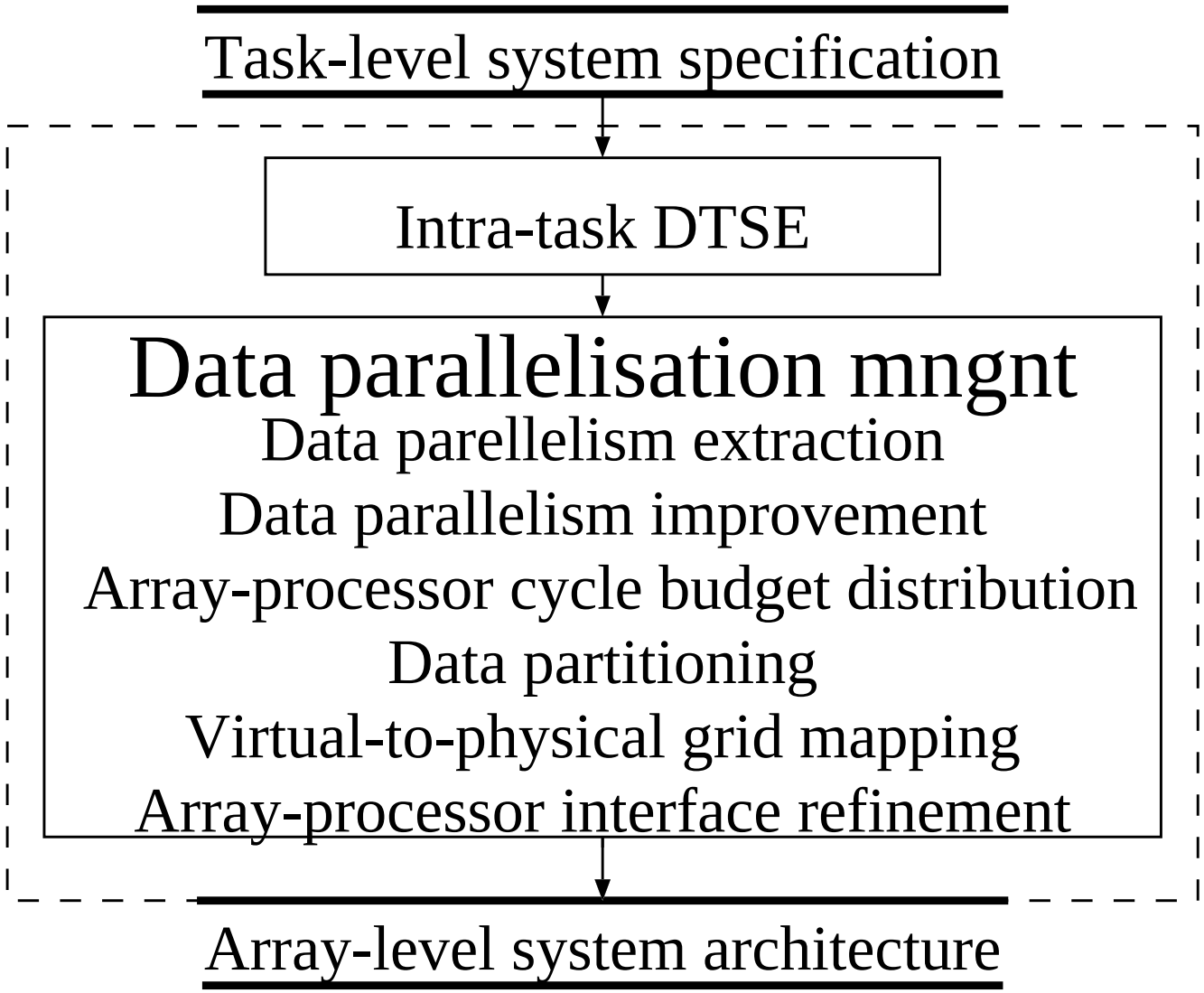
Concurrency issues and constraints



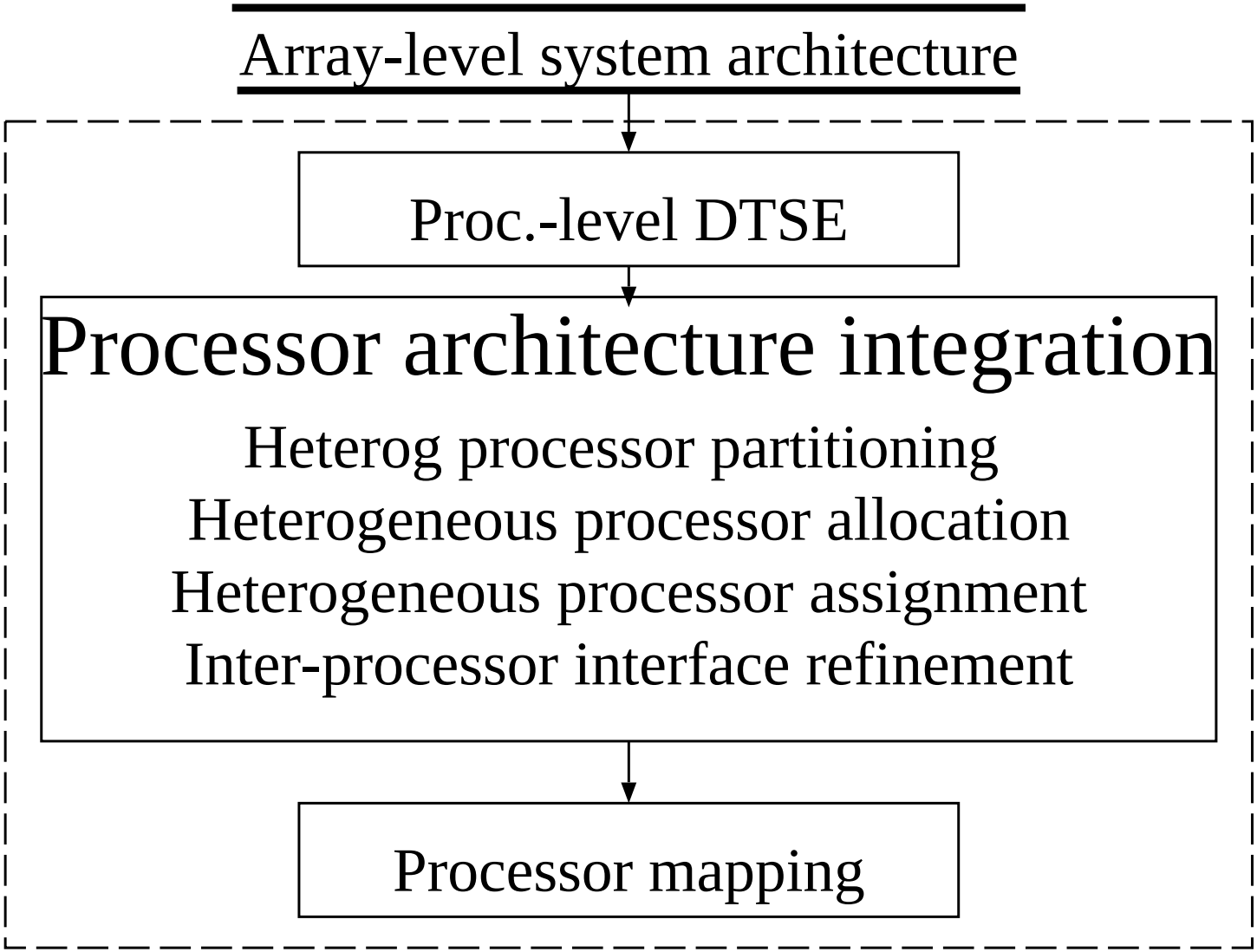
Task concurrency management flow



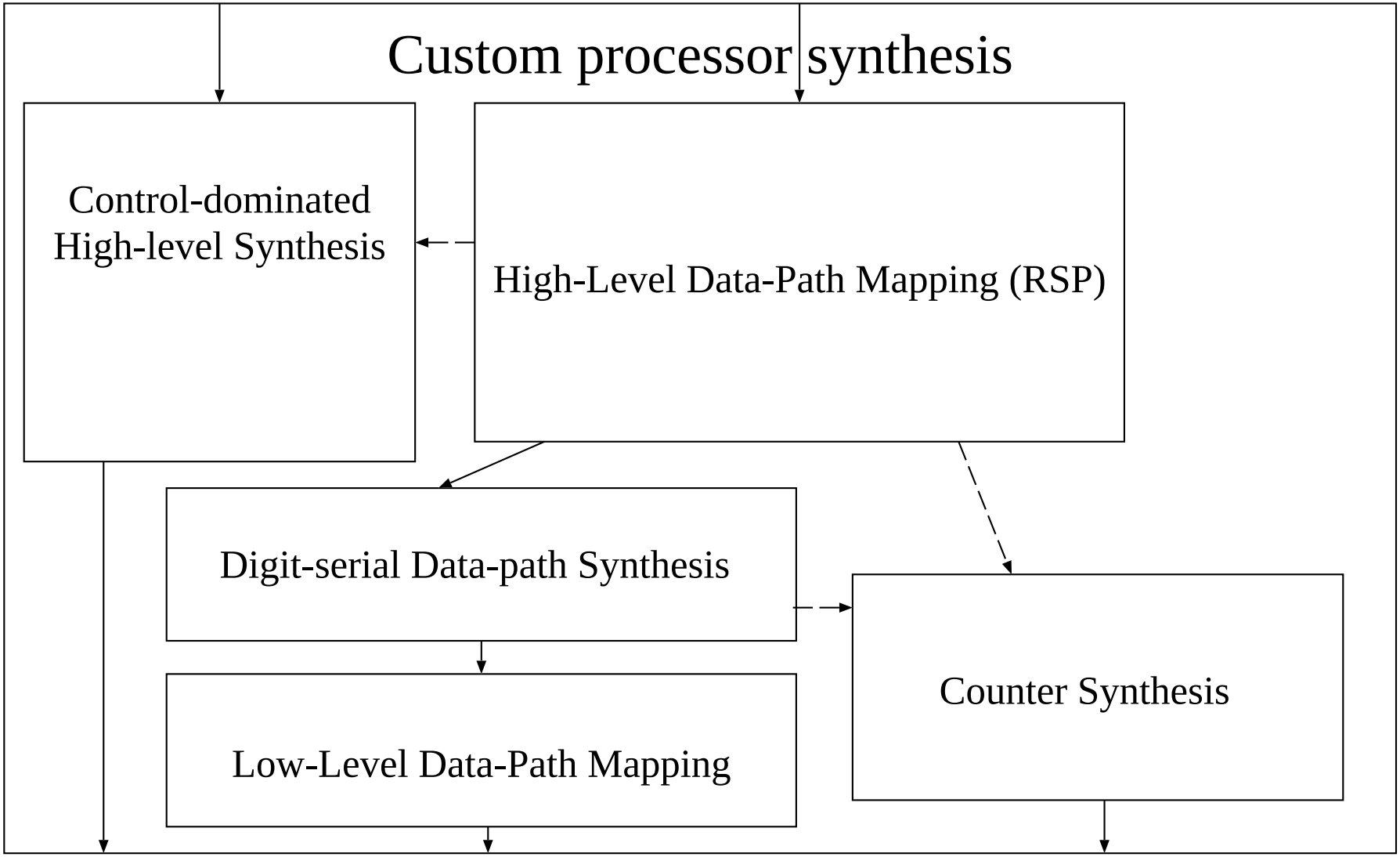
Data parallelisation management flow



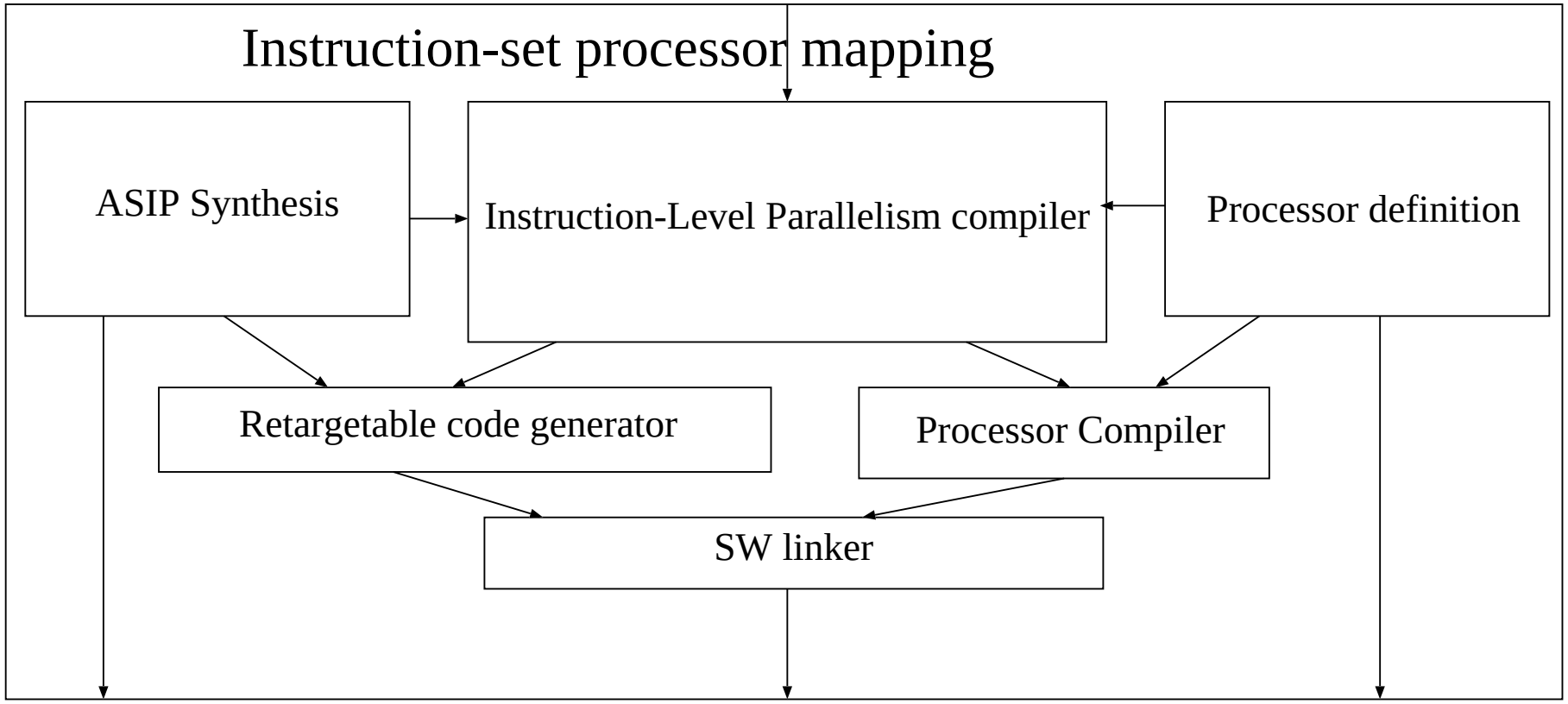
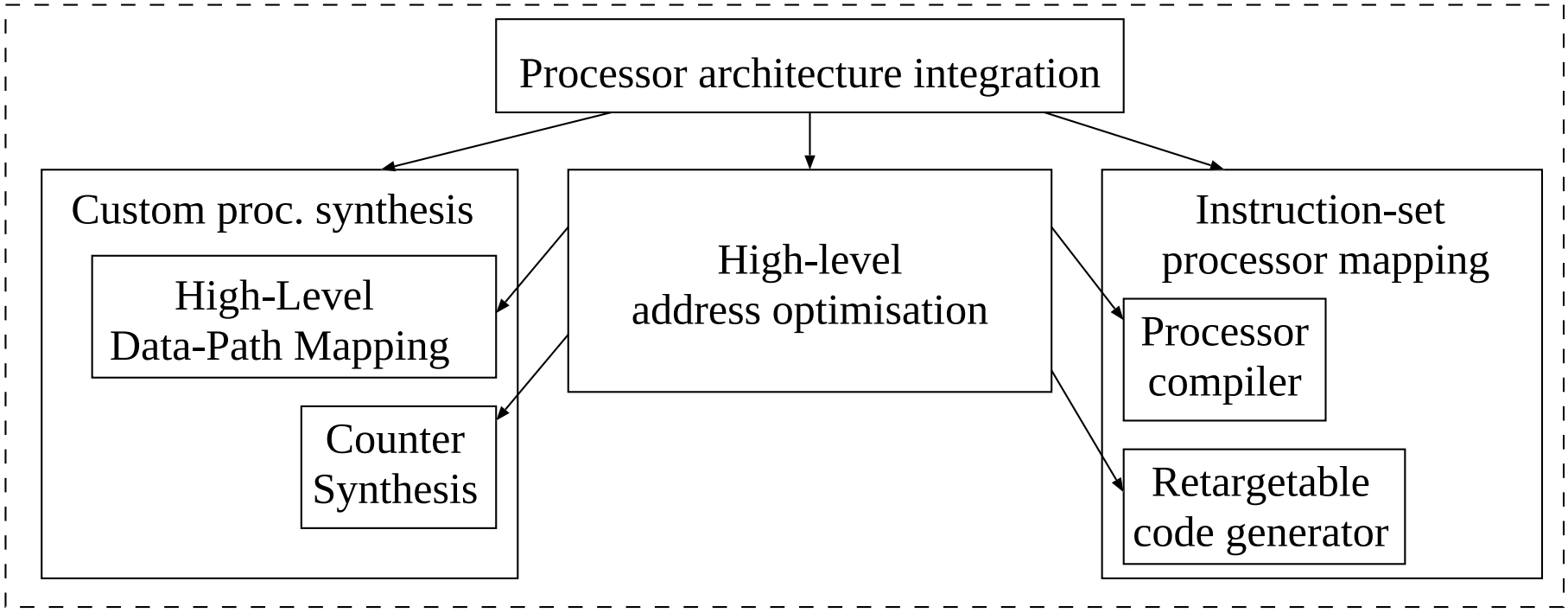
Processor architecture integration flow



Custom processor synthesis flow: simple



Instruction-set processor mapping flow: simple



High-level address optimisation flow: simple