# INTEGRATED WCET ESTIMATION OF MULTICORE APPLICATIONS

Dumitru Potop-Butucaru, Isabelle Puaut

# Motivation: Scalable timing analysis



- Real-time systems: complexity steadily increases
  - Hardware: Multi-core, networks-on-chips
  - Software: Parallel/concurrent software
- Safety margins used in practice after schedulability analysis are already enormous (40%-60%)
  - Further static abstraction is not a solution
- How to preserve both tractability and precision?
  - Probabilistic approaches (another form of abstraction), or
  - Use « WCET-friendly » hardware and software
    - Limit/control timing interferences due to concurrency
      - Static (off-line) scheduling, non-preemptive, etc.
      - No shared caches, LRU caches, time-triggered execution, etc.

# Static timing analysis



- 3 basic sources of imprecision:
  - Application-related:
    - Input arrival dates, data-dependent behavior
  - Mapping-related:
    - Concurrency (pipelining, buses, scheduling)
  - Analysis-related:
    - Abstraction (e.g. IPET, real-time calculus, etc.)
- Our thesis:

**Few sources of imprecision in the application and mapping allow for scalable, precise analysis**

# Reducing imprecision



- Everybody is doing it (to a point)
  - Industry: Space & time partitioning (among others)
    - Time-triggered standards: TTA, ARINC 653
    - Recent many-core chips: TilePro64, Kalray MPPA256, etc.
  - Research:
    - Precision timed architectures (PRET) – Lee, etc.
    - CompSoC, Aethereal, etc.
    - Off-line scheduling – Fohler, Eles, Sorel, etc.

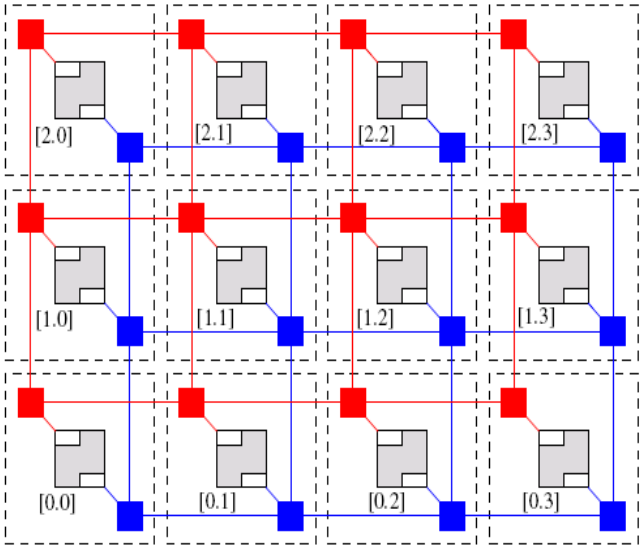
- But we do it all the way:
  - **Remove all application- and mapping-related imprecision sources that are not handled by classical WCET analysis**
    - Possibly add some back later on (future work)
    - **This paper: see that it's possible and determine the gain**

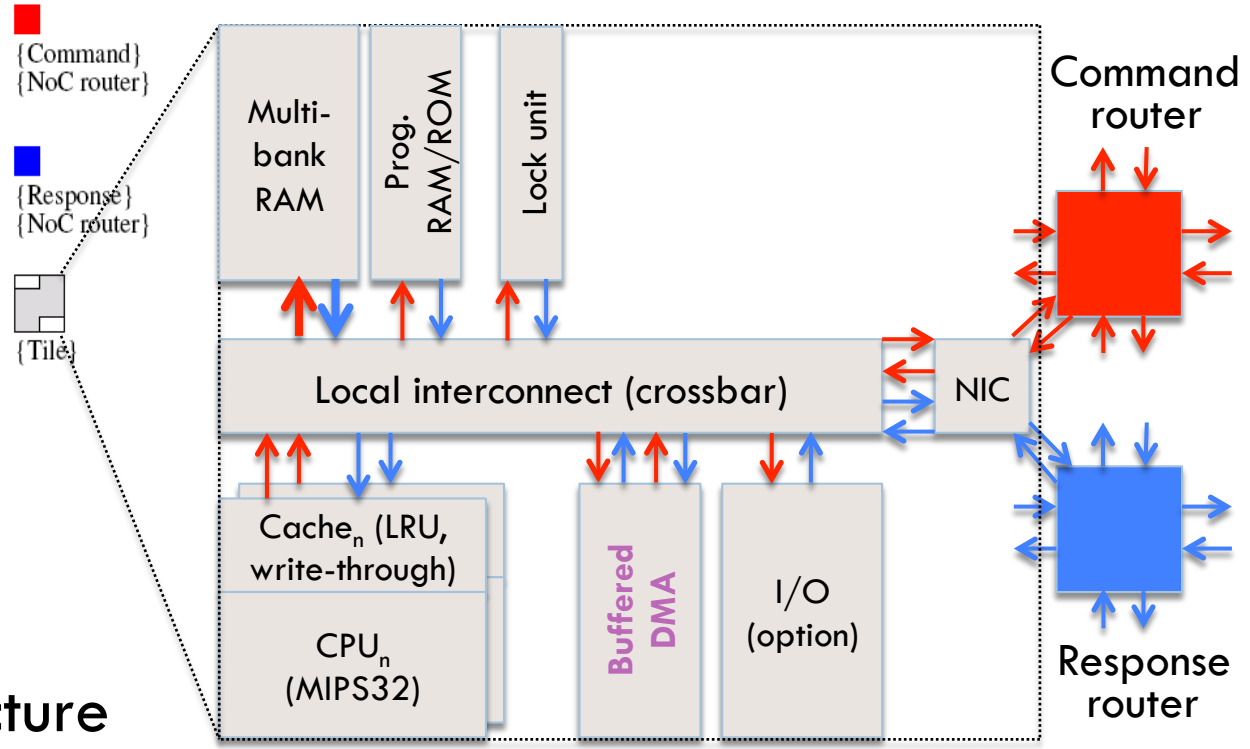
# Tiled MPSoC architecture

Djemal et al.,
DASIP 2012

Based on SoCLib
(UPMC/LIP6)



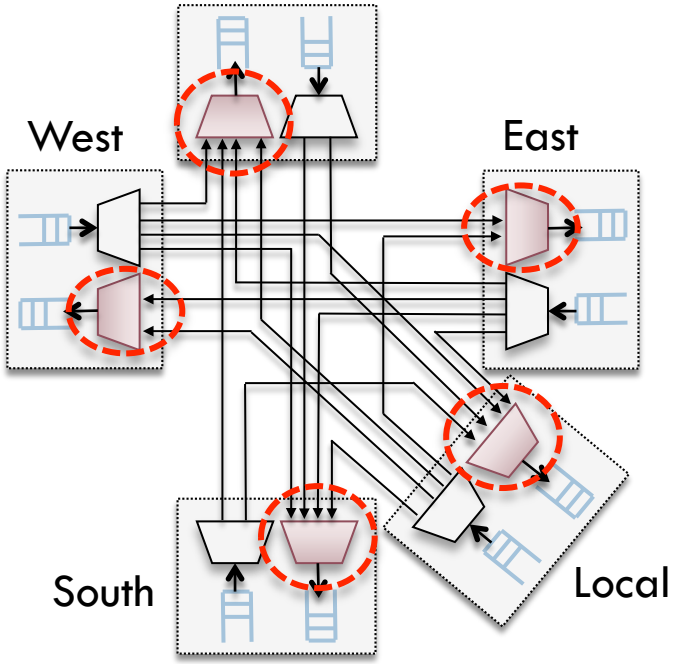


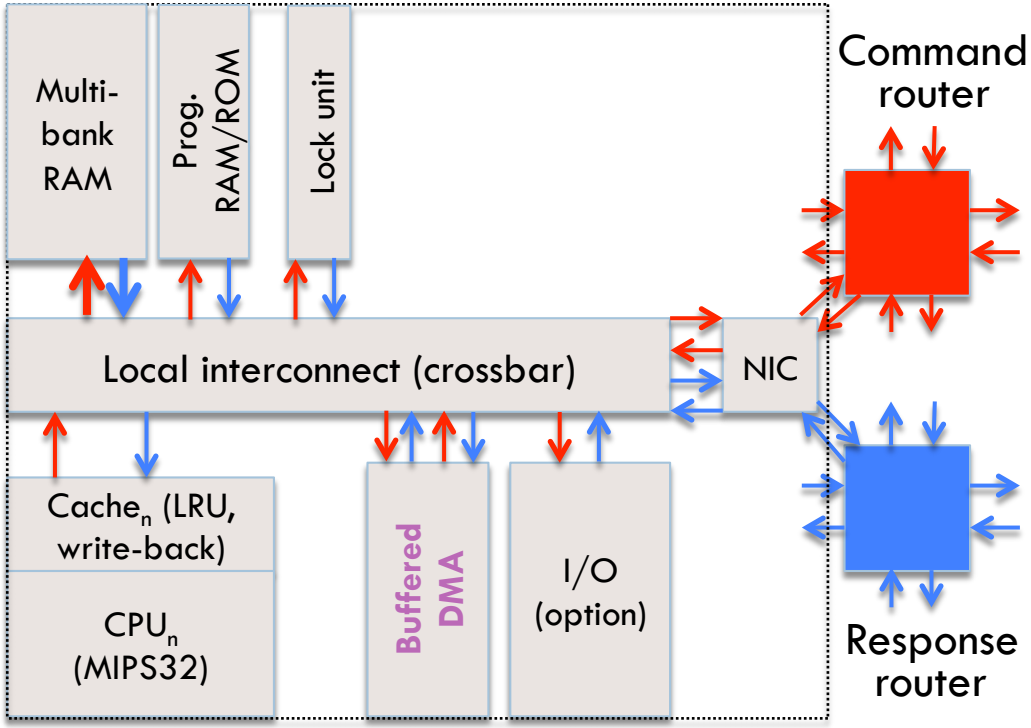




- Multi-bank RAM
- Harvard-like architecture
- Full crossbar intra-tile interconnect
- Hardware locks for synchronization (not interrupts)
- Static routing (X-first)

# Tiled MPSoC architecture






- Provide timing guarantees for inter-tile communications
  - Use of locks, programmed arbitration (others do TDMA or other types of resource reservation)
- Tool limitation: 1CPU/tile

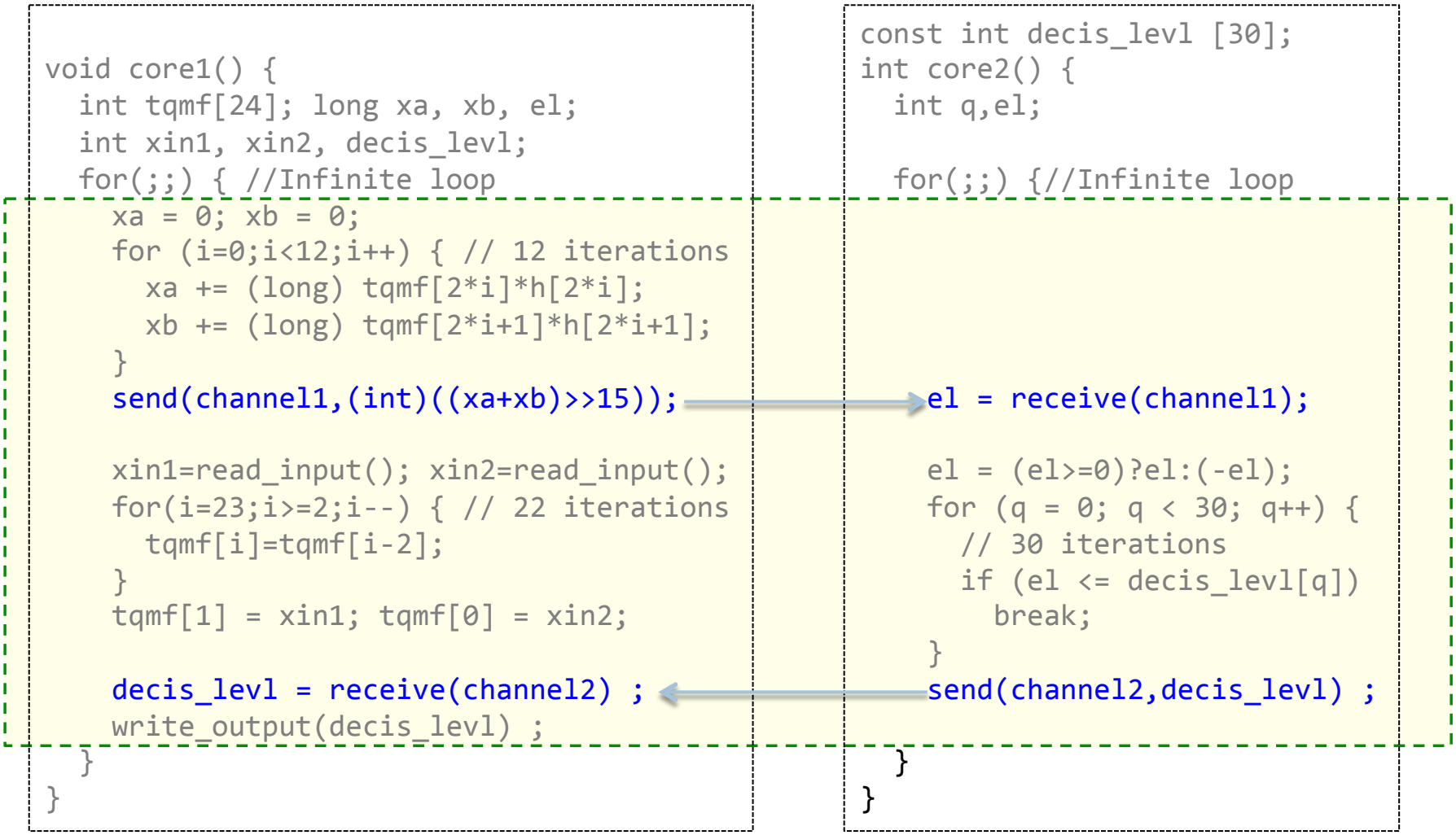
# Tiled MPSoC applications



- On each processor, sequential code
  - Non-preemptive, off-line scheduling
  - Synchronization by blocking send/recv operations
    - Lossless FIFOs
  - A.k.a. Kahn process networks (G. Kahn, 1974)
- No concurrent access to RAM banks, DMA units, NoC router outputs
  - Data allocation on memory banks, use of locks to enforce a predefined schedule
- Tool limitations
  - Sampled I/O only
  - Send/recv primitives are explicitly matched
  - Send/recv only at top level (global loop), non-conditioned

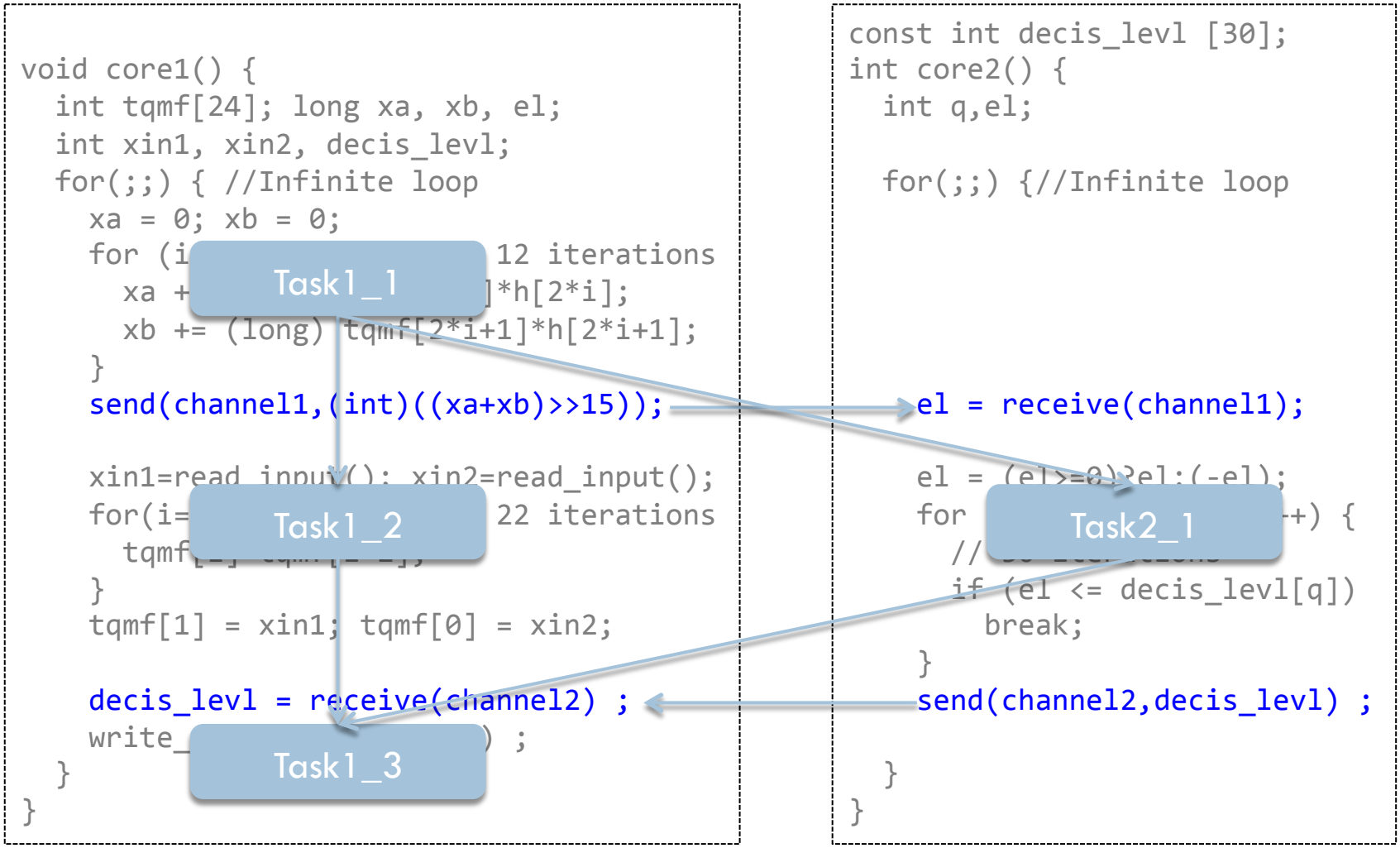
# Tiled MPSoC applications (example)





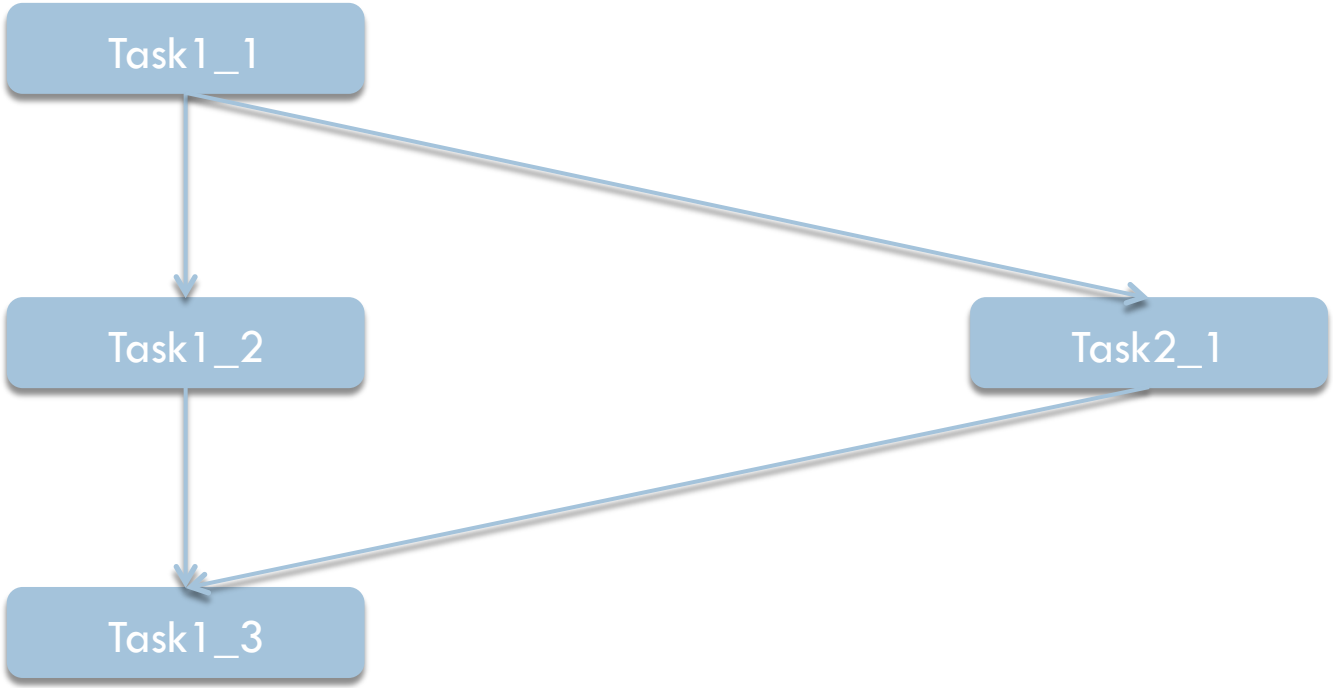
# Traditional timing analysis





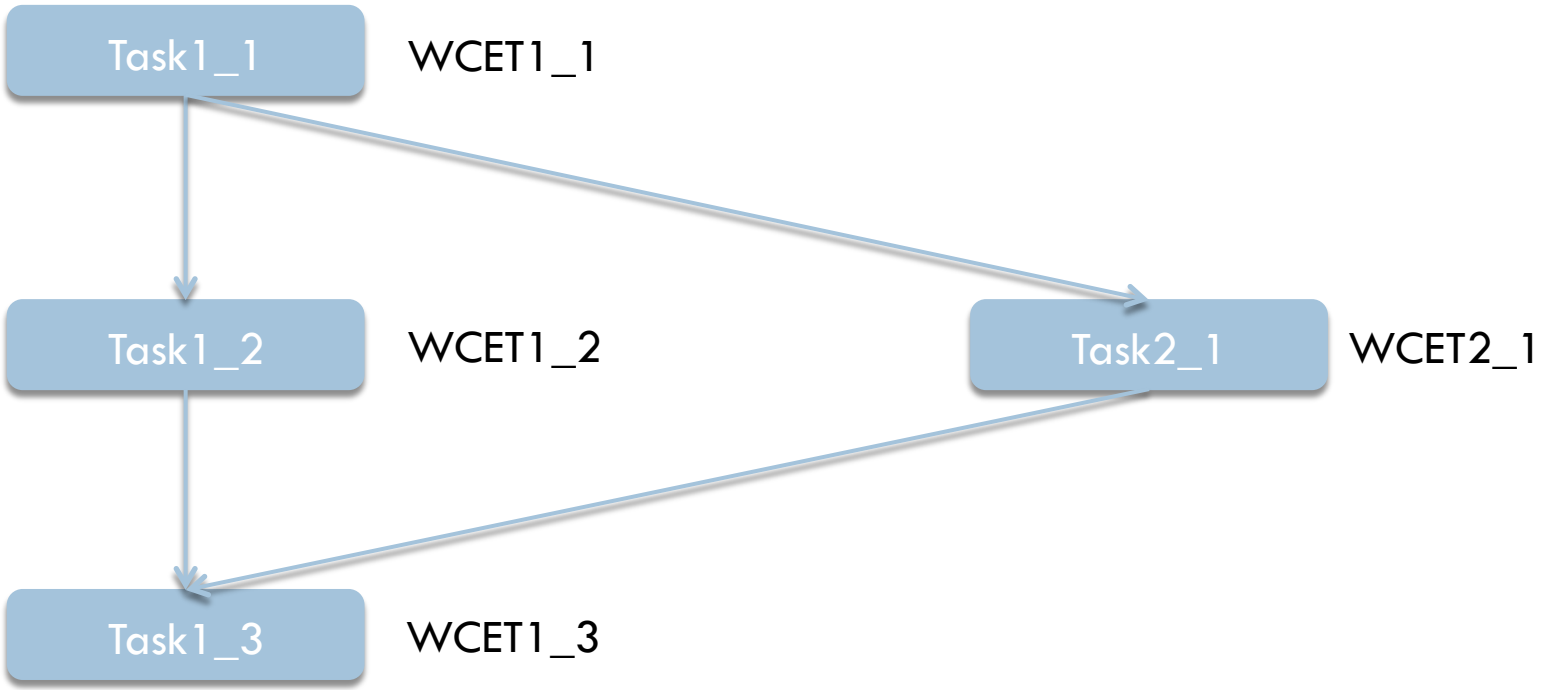
# Traditional timing analysis





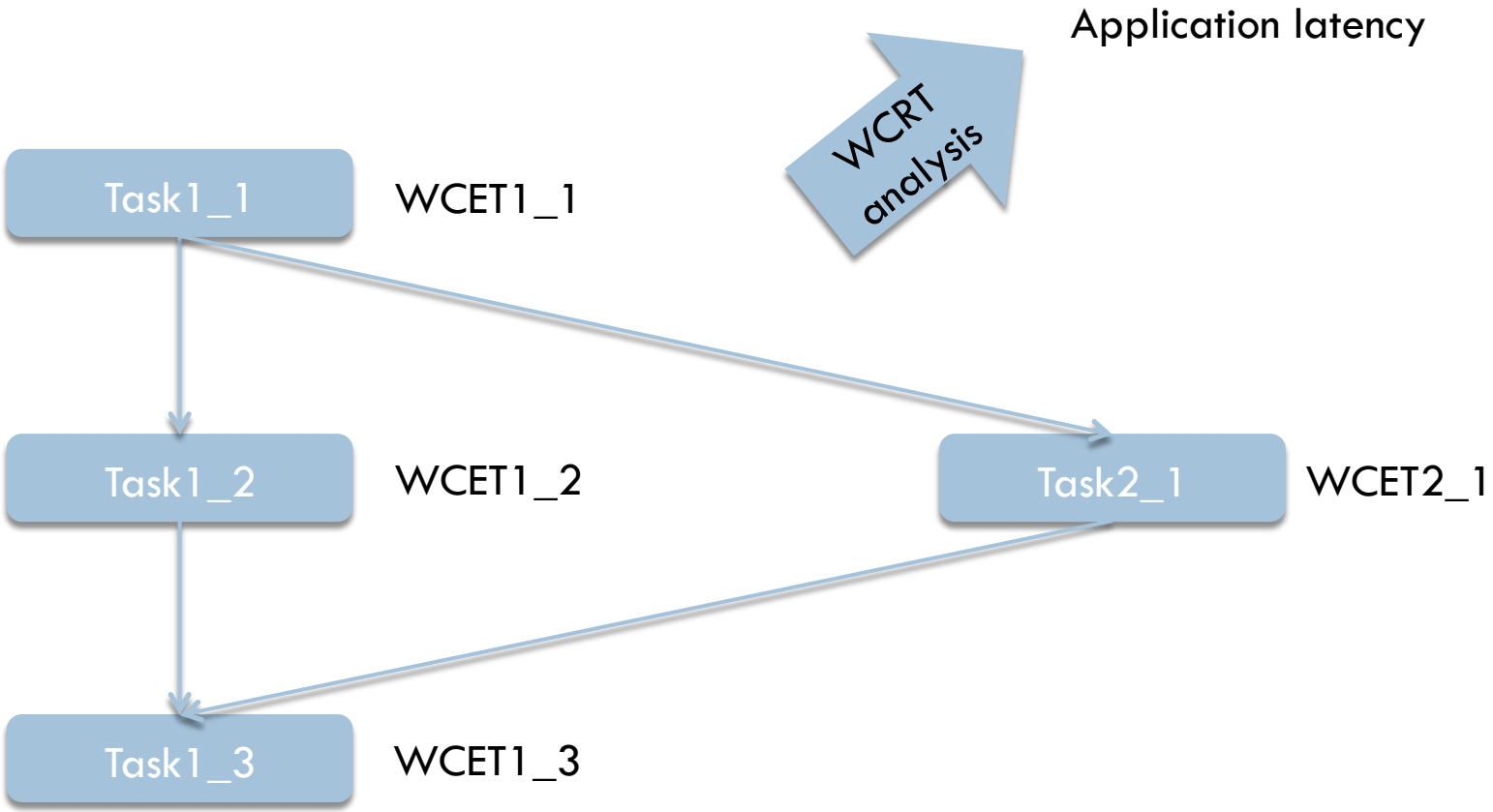
# Traditional timing analysis





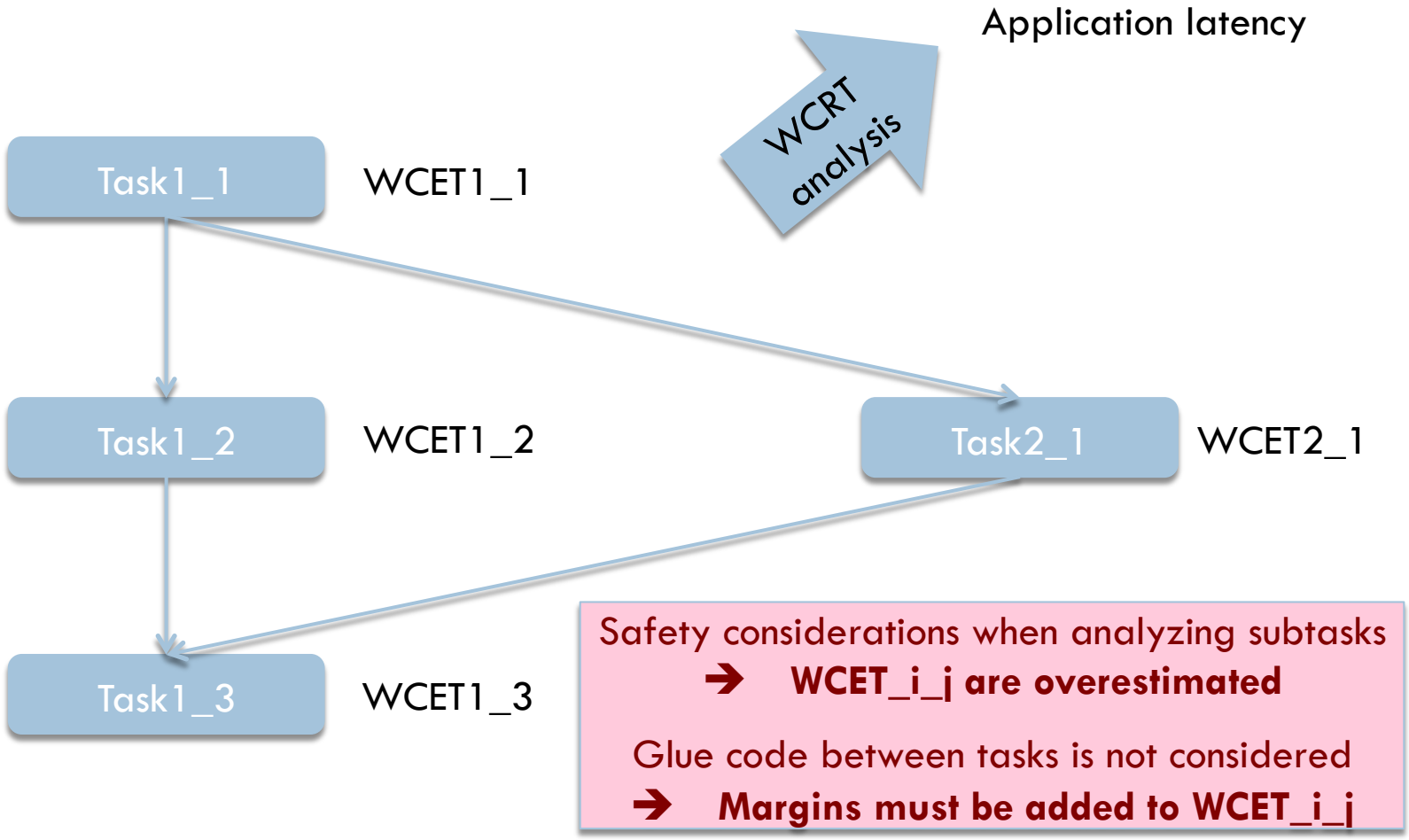
# Traditional timing analysis





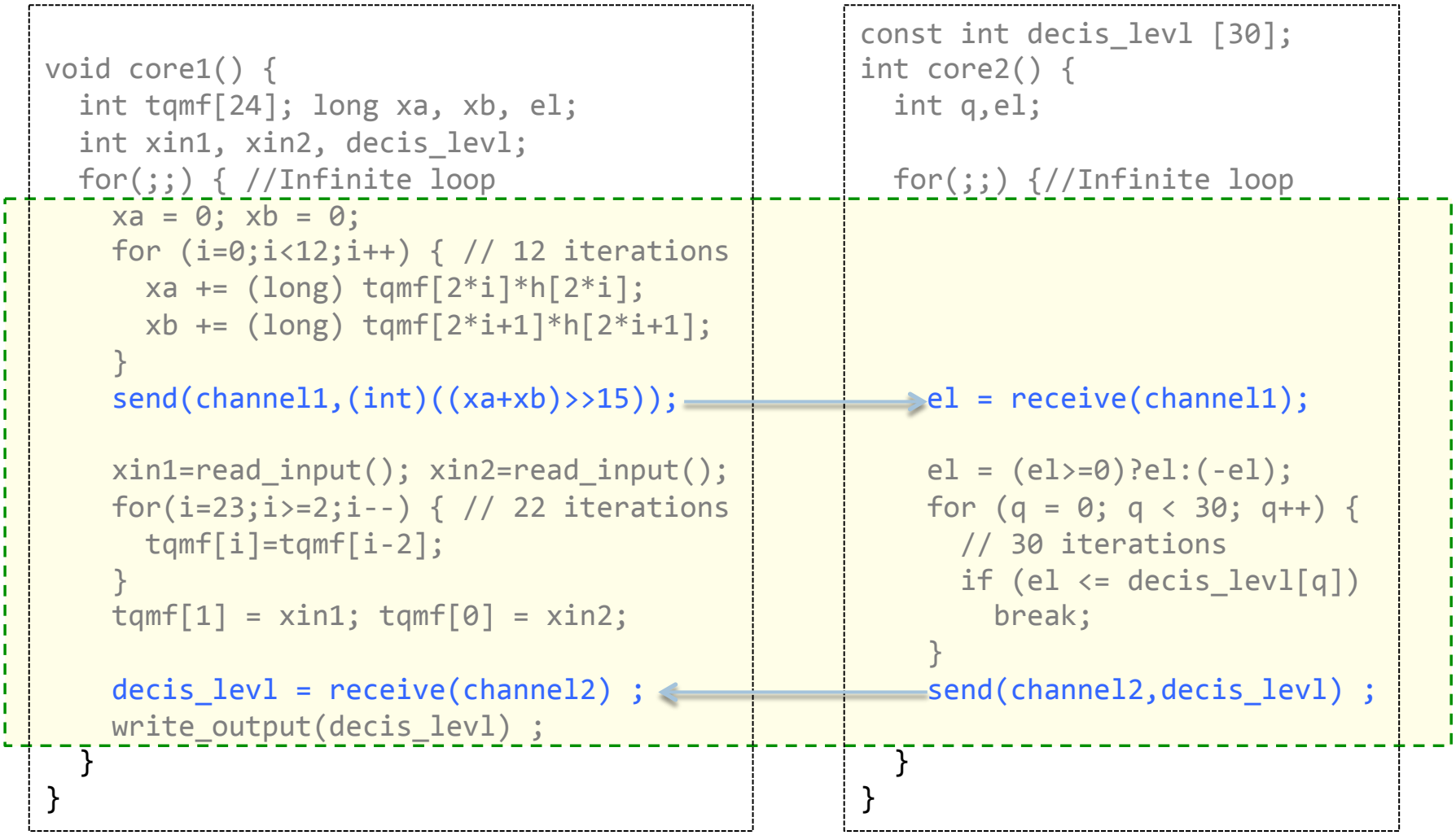
# Traditional timing analysis





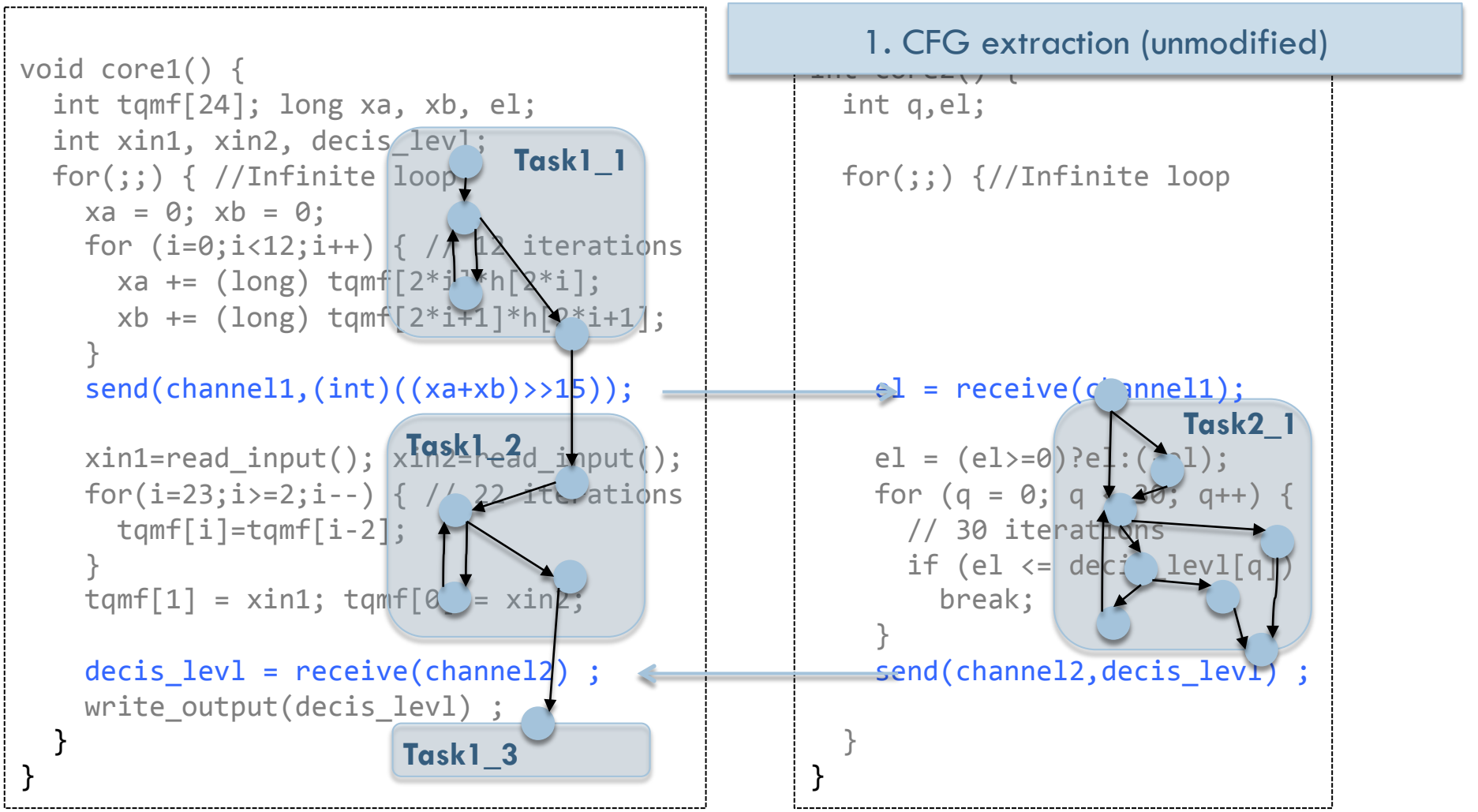
# Unified timing analysis





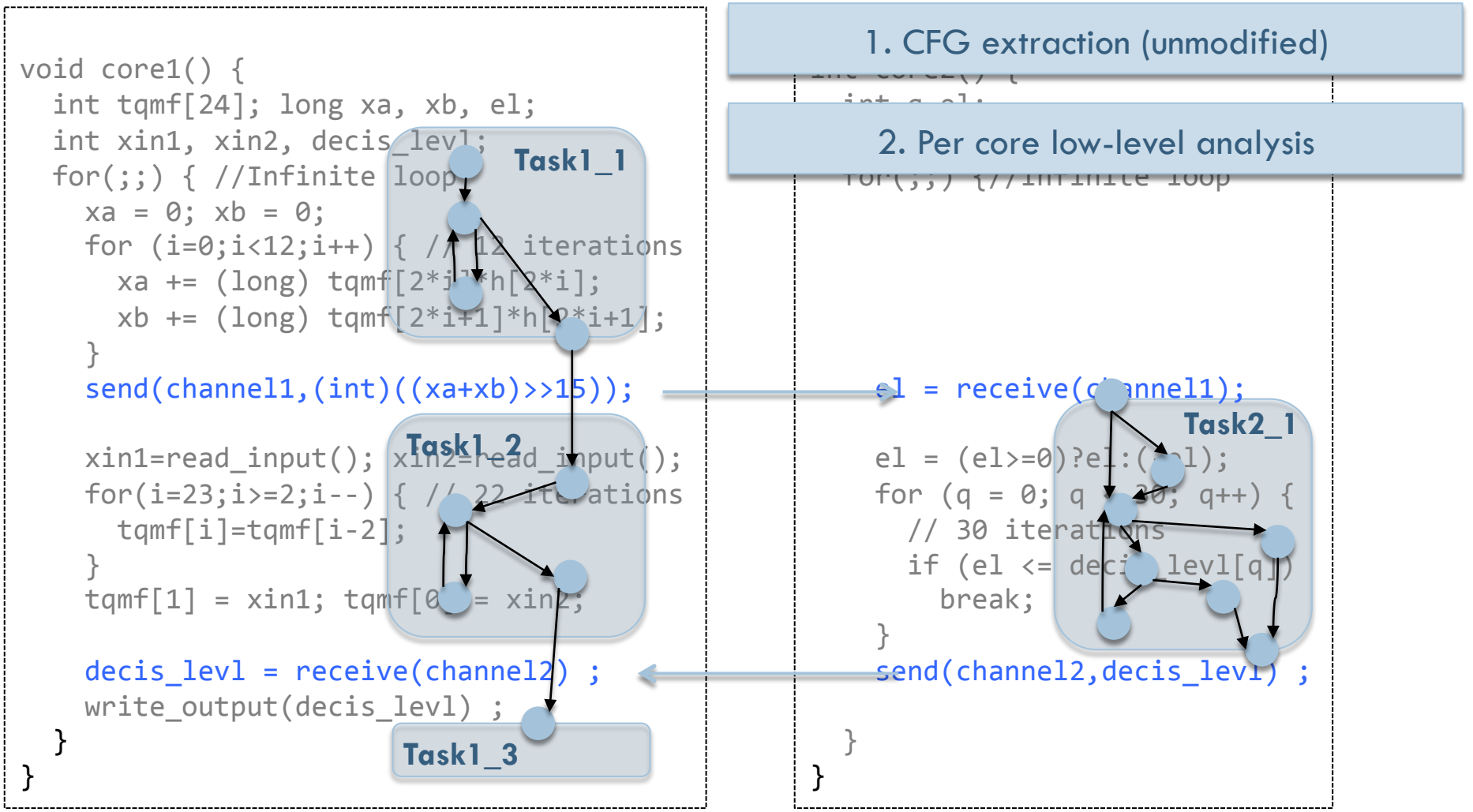
# Unified timing analysis





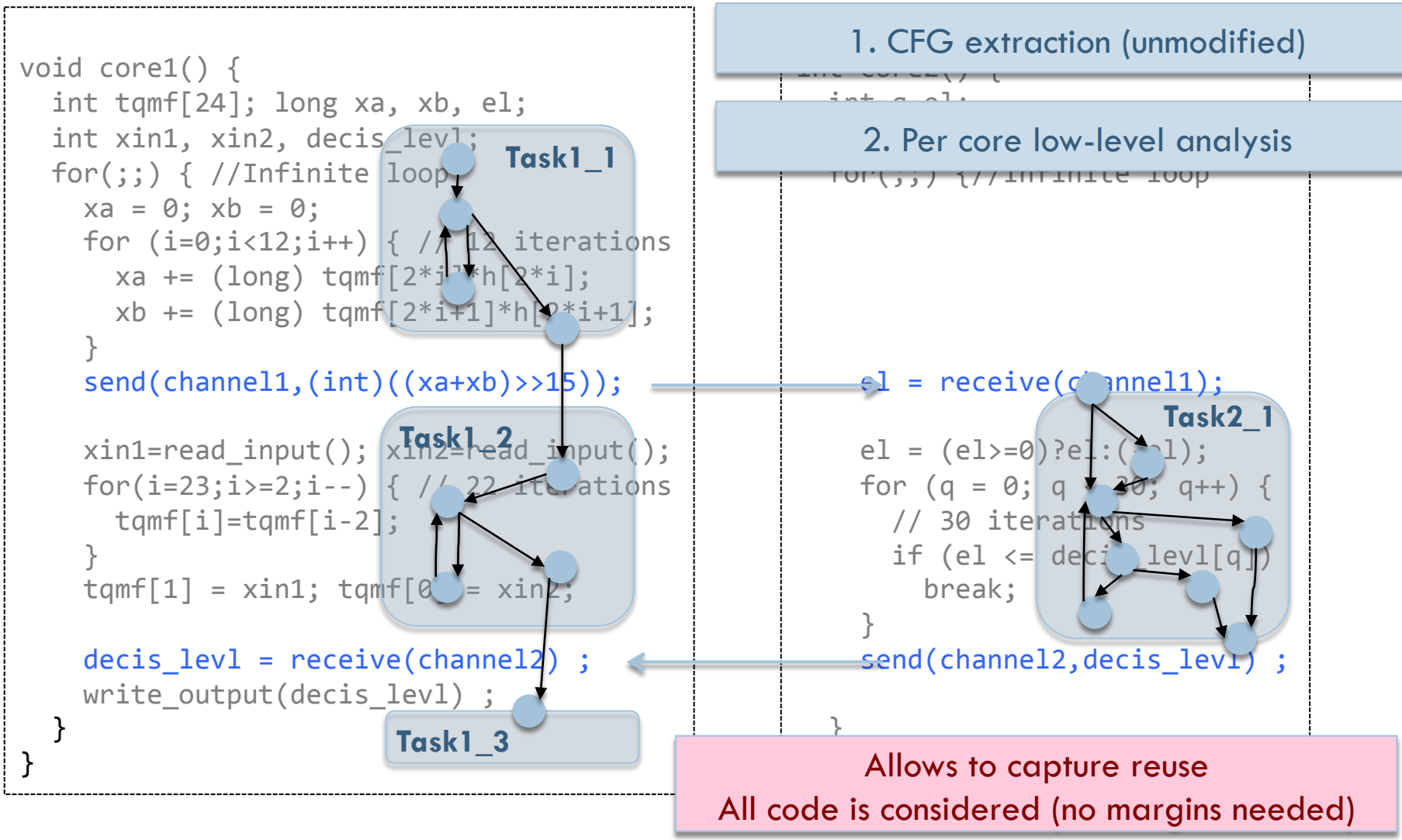
# Unified timing analysis





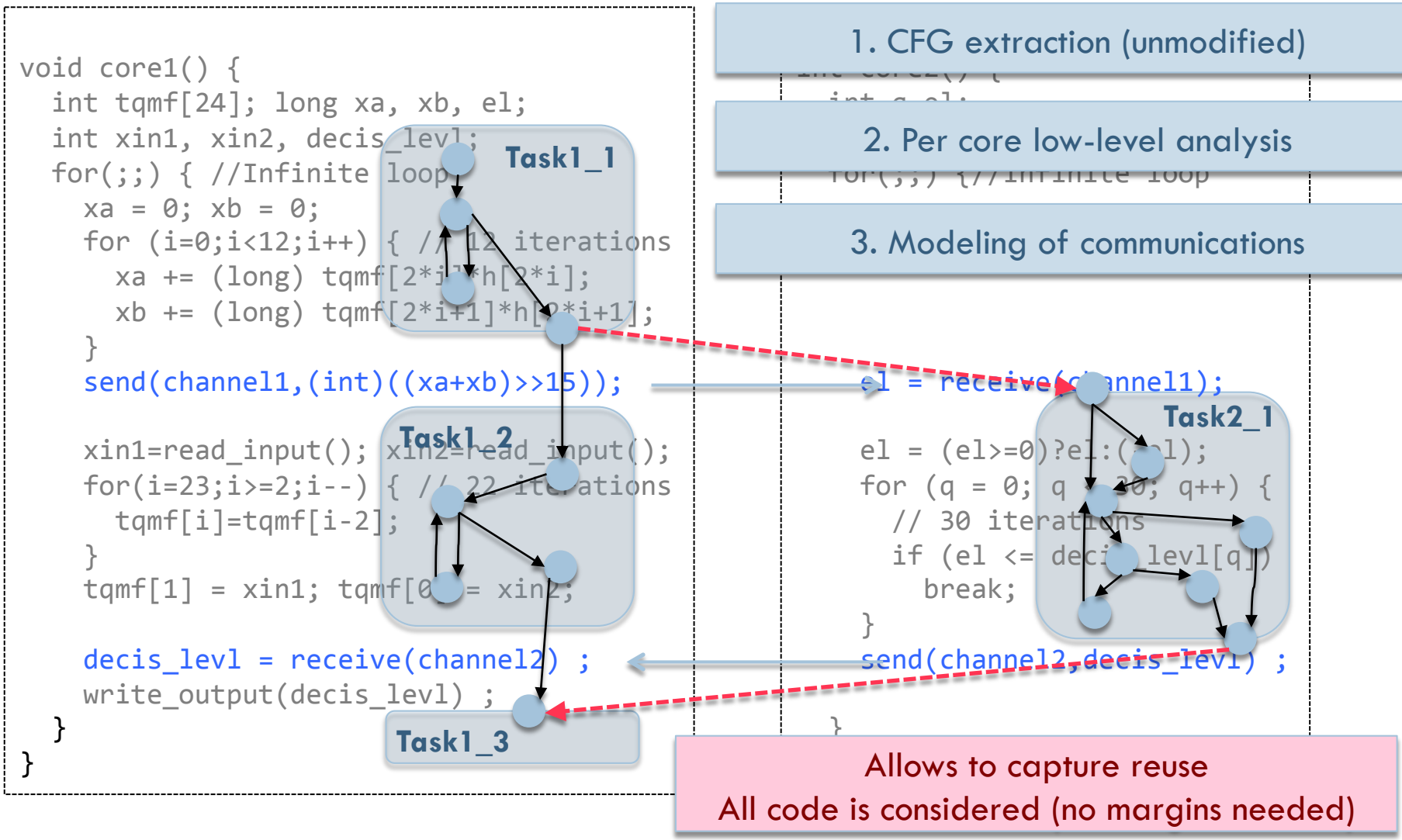
# Unified timing analysis





# Unified timing analysis





# Unified timing analysis



Flow constraints:

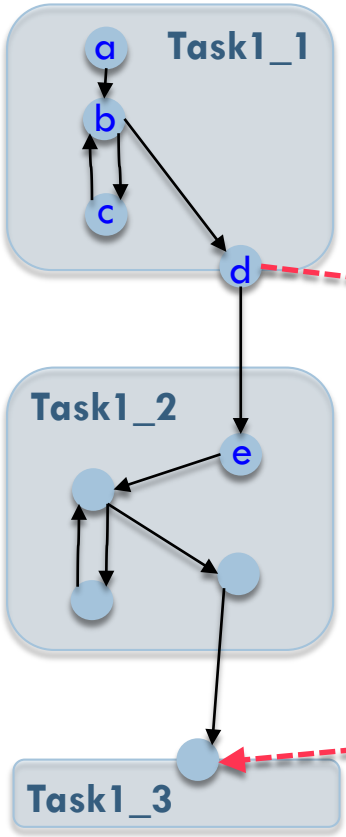
$x_b = x_{ab} + x_{cb} = x_{bc} + x_{bd}$

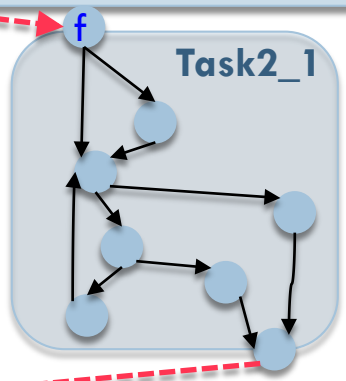
$x_d = x_{db} = x_{df} + x_{de}$

…

Objective function:

$\max(x_a * t_a + x_b * t_b + x_{df} * 250)$




1. CFG extraction (unmodified)

2. Per core low-level analysis

3. Modeling of communications

4. WCET estimation (standard IPET)

Allows to capture reuse
All code is considered (no margins needed)

# Unified timing analysis (detail, 1)



Flow constraints:

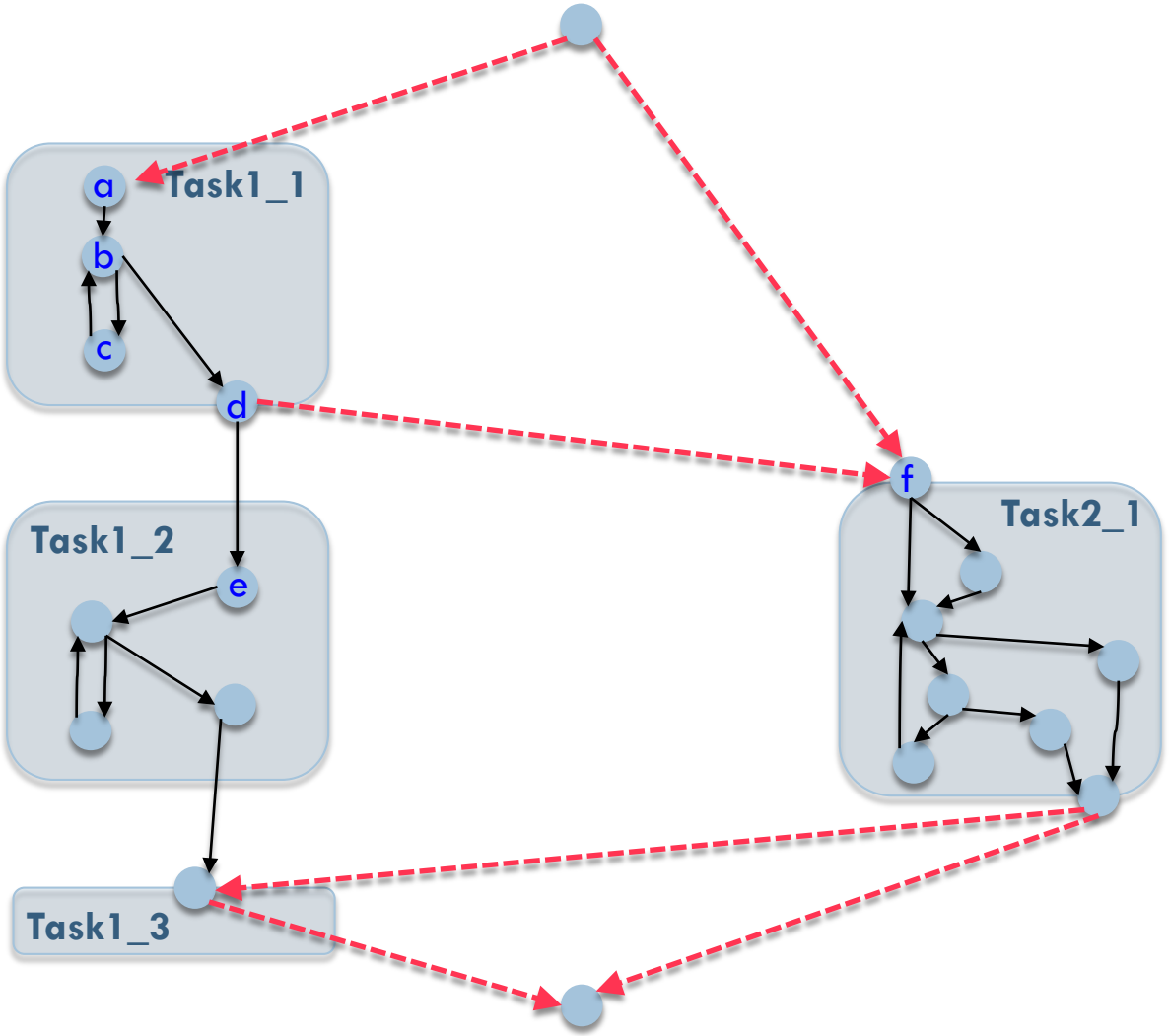
$x_b = x_{ab} + x_{cb} = x_{bc} + x_{bd}$

$x_d = x_{db} = x_{df} + x_{de}$

…

Objective function:

$\max(x_a * t_a + x_b * t_b + x_{df} * 250)$



# Unified timing analysis (detail, 2)



Flow constraints:

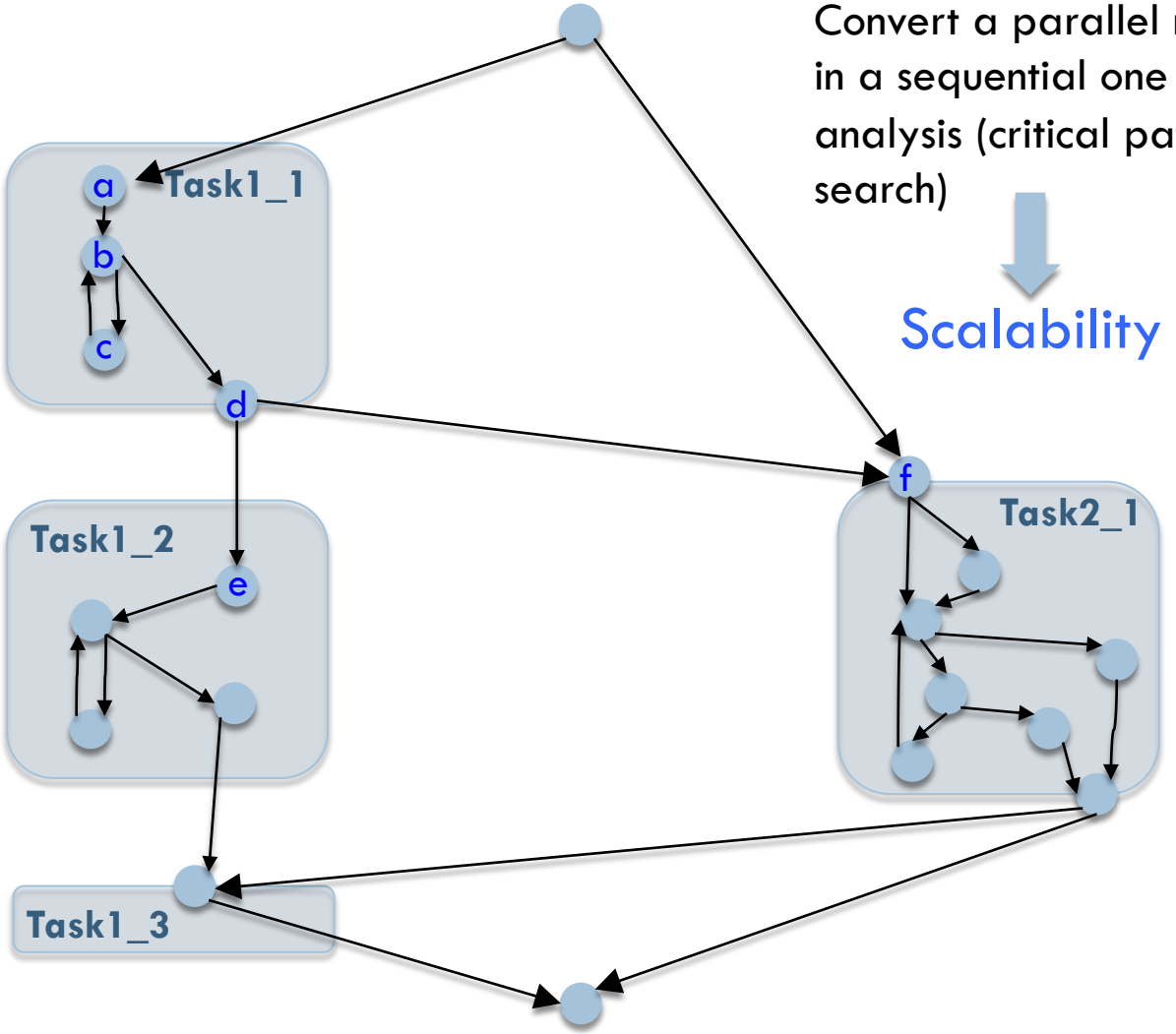
$x_b = x_{ab} + x_{cb} = x_{bc} + x_{bd}$

$x_d = x_{db} = x_{df} + x_{de}$

…

Objective function:

$\max(x_a * t_a + x_b * t_b + x_{df} * 250)$



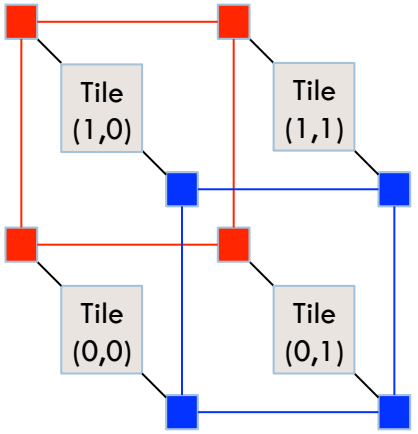

Convert a parallel model in a sequential one for the analysis (critical path search)

Scalability

# Experimental results



- Experimental setup
  - 2x2 MPSoC
    - 1 CPU/tile
      - In order execution
      - No variable time instructions
    - Cycle-accurate simulator
  - Heptane WCET analysis tool

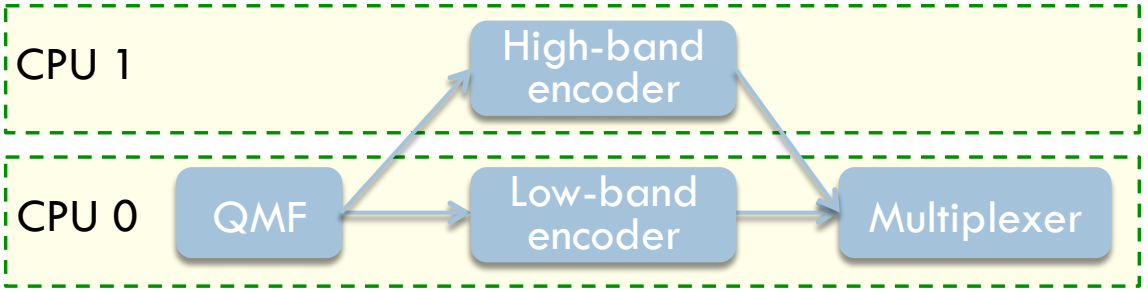



- Very accurate (precise) hardware model
  - Same number of cycles for simple single-path programs between Heptane and the SystemC simulator

# Experimental results



- Two examples, 3 configurations:
  - Adpcm
    - 2 cores
      - No (SW) pipelining
    - 4 cores
      - One operation/CPU
      - Pipelined




  - Load balancing (2 cores)
    - Simple filter, need 2 CPUs to meet throughput
    - Pipelined

# Experimental results



- Comparison with the isolated (traditional) timing analysis

| | Integrated (cycles) | Isolated (cycles) | Improvement (%) |
|---|---|---|---|
| Adpcm – 2 cores | 73563 | 101431 | 36.5% |
| Adpcm – 4 cores | 44568 | 55919 | 25.5% |
| Filter – 2 cores | 110825 | 112543 | 1.55% |

# Experimental results



- Comparison with the isolated (traditional) timing analysis

| | Integrated (cycles) | Isolated (cycles) | Improvement (%) |
|---|---|---|---|
| Adpcm – 2 cores | 73563 | 101431 | 36.5% |
| Adpcm – 4 cores | 44568 | 55919 | 25.5% |
| Filter – 2 cores | 110825 | 112543 | 1.55% |

Always an improvement
Improvement depend on the amount of reuse

# Experimental results



- Comparison with the measured execution time (typical input, single run)

| | Integrated (cycles) | Measured (cycles, typical input) | Pessimism (%) |
|---|---|---|---|
| Adpcm – 2 cores | 73563 | 64944 | 13.3% |
| Adpcm – 4 cores | 44568 | 41468 | 7.5% |
| Filter – 2 cores | 110825 | 108296 | 2.3% |

# Experimental results



- Comparison with the measured execution time (typical input, single run)

| | Integrated (cycles) | Measured (cycles, typical input) | Pessimism (%) |
|---|---|---|---|
| Adpcm – 2 cores | 73563 | 64944 | 13.3% |
| Adpcm – 4 cores | 44568 | 41468 | 7.5% |
| Filter – 2 cores | 110825 | 108296 | 2.3% |

Actual pessimism expected to be lower
Still, pessimism is reasonable

# Conclusion



- **Predictable architecture + integrated approach → static tight WCETs**
  - Scalable
    - Same complexity as IPET on a sequential program of the same size
  - Better than traditional timing analysis
    - Captures cache reuse within one core
    - No need for safety margins to account for glue code
- Future work
  - More experiments
  - More general task/architecture model
  - Closer interaction WCET – scheduling/mapping
    - Put WCET in the loop during scheduling/mapping