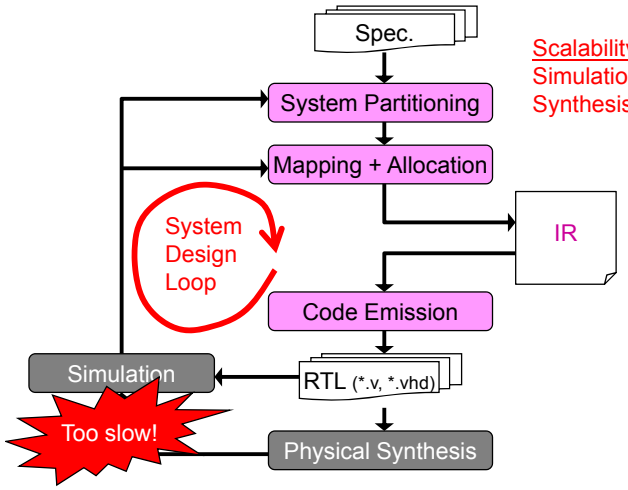
Slack Analysis in the System Design Loop

Girish Venkataramani — Carnegie Mellon University, The MathWorks

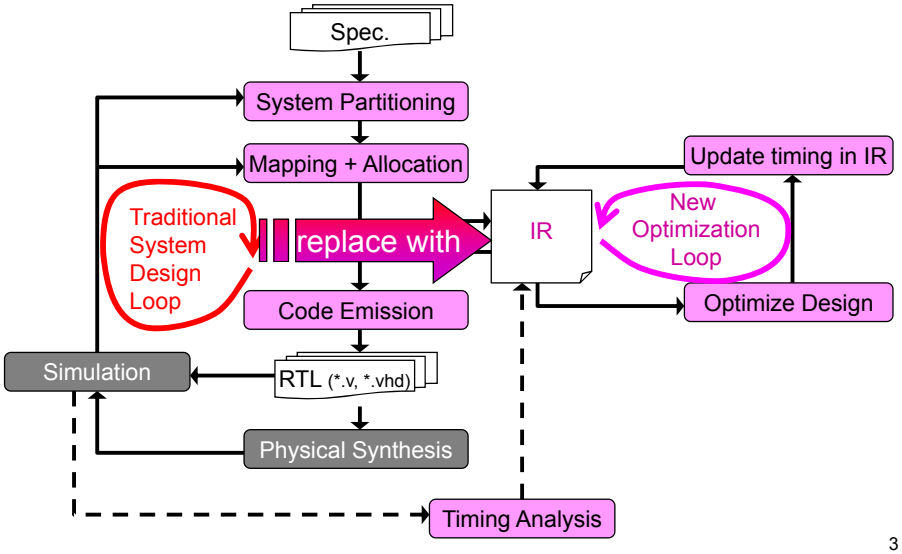
Seth C. Goldstein — Carnegie Mellon University

Typical System Design Flow

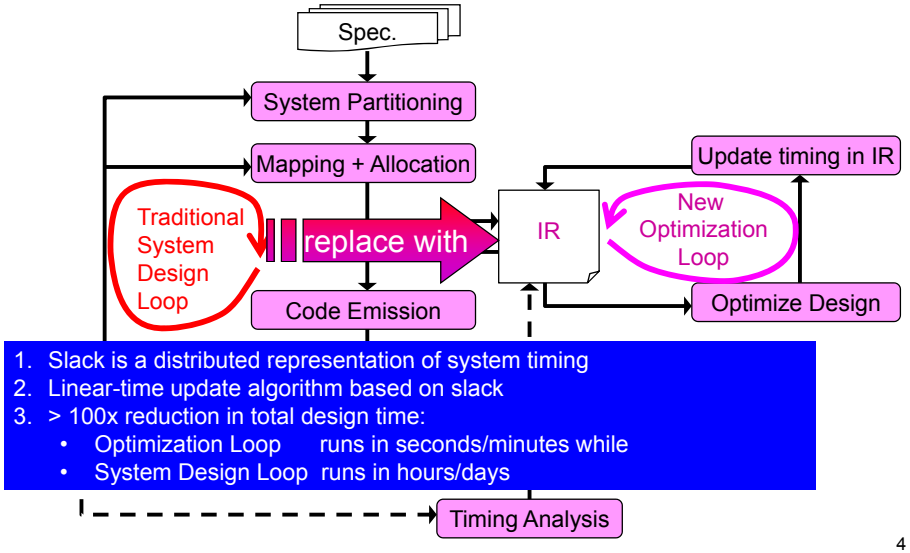


Scalability Issues:
Simulation takes minutes to hours
Synthesis takes many hours

Proposed Design Flow



Key Contributions



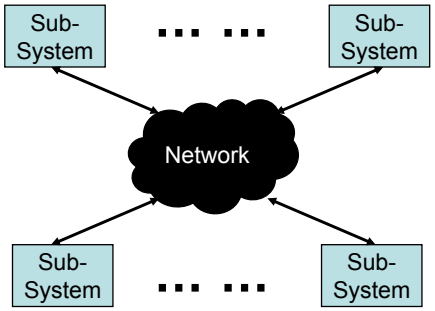
1. Slack is a distributed representation of system timing
2. Linear-time update algorithm based on slack
3. > 100x reduction in total design time:
   - Optimization Loop runs in seconds/minutes while
   - System Design Loop runs in hours/days

Outline

- Motivation
- Timing Metrics
  - Cycle time
  - Slack: An alternative view of cycle time
- Slack Update Algorithm
- Experimental Evaluation
- Conclusions

The Intermediate Representation (IR)

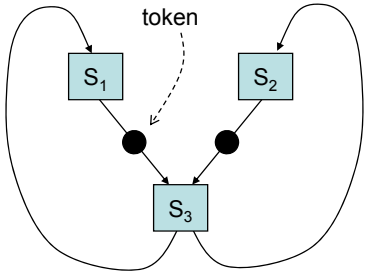
- Models a dynamic system
- Concurrent sub-systems
  - PE, FSM, S/W, Memory
- Communication between sub-systems based on pre-defined protocols
  - FIFOs, NoC, shared bus



Transaction-Level Modeling (TLM), [Cai, ISSS 03]
Adopted by System-C, Bluespec, Balsa, Tangram

Marked Graphs

- Model dynamic system interactions
- Events and transitions
  - Event: An edge acquires a token
  - Transition: Node consumes inputs and generates outputs
- Encode the communication protocols



Timing Analysis of IR

- Time Separation between Event (TSEs)
  - TSE between consecutive firings of same event in steady state is the mean cycle time
- Mean Cycle Time, *CT*

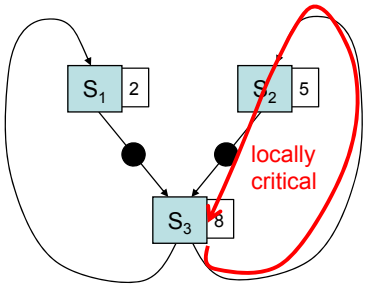
$$CT = \max_{\forall C_i} \left(\frac{C_i}{N_i}\right)$$

where: C_i is latency of a graph cycle,
N_i is number of tokens on cycle

- Computing *CT* is about $O(|E|^3)$ complexity [Dasdan 04]

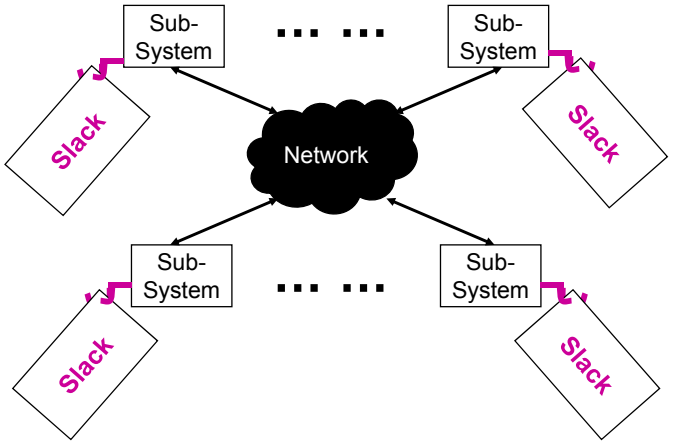
Slack as a Timing Metric

- Distributed representation of cycle time
  - Different type of TSE
  - Defined on each (input edge, node) pair
  - How early this input arrives
    - Slack(S1, S3) = 3
    - * Slack(S2, S3) = 0
    - Slack(S3, S1) = 0
    - Slack(S3, S2) = 0
- Zero-slack input is locally critical



- Longest chain of zero-slack events yields the critical cycle or the Global Critical Path (GCP)
  - Cycle time = Latency of the GCP
  - Given slack values, computing cycle time has linear complexity

Slack is an Annotation on the IR

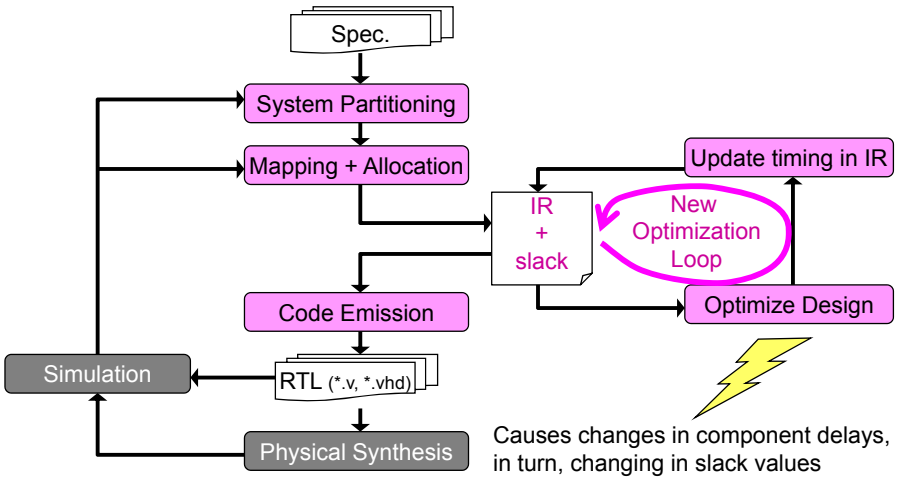


Helps in discovering hotspots and applying optimizations

Outline

- Motivation
- Timing Metrics
- Slack Update Algorithm
- Experimental Evaluation
- Conclusions

Optimizations Change the IR

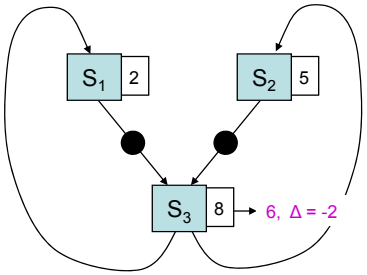


Causes changes in component delays, in turn, changing in slack values

Need to update slack on each change

Problem Description

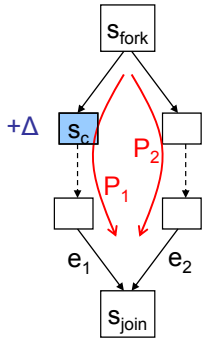
- Given a graph model and its current slack values, compute new values of slack when latency of a node changes by a given Δ



Insight behind Update Algorithm

- Slack is also latency difference of two branches of a re-convergent fork-join

- If delay of node, s_c, increases by Δ
  - Update slack in surrounding re-convergent fork-joins
  - Propagate change globally



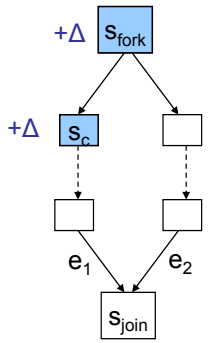
Assume $\delta(P_1) > \delta(P_2)$,
$Di = \delta(P_1) - \delta(P_2)$
$Slack(e_1, s_{join}) = 0$
$Slack(e_2, s_{join}) = Di$

Update (let $\Delta \leq Di$):
$Slack(e2, s_{join}) = Di + \Delta$

Insight behind Update Algorithm

1. If there is a path from change point to every input of s_{join}, then no change in slack

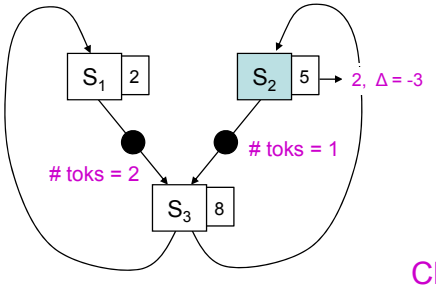
2. If not, then slack changes occur



Easy for acyclic graphs, but what about scc graphs?

Insight for Cyclic Graphs

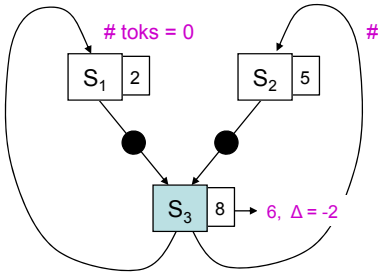
- Use token knowledge
  - Count tokens from change point to every input
  - If value is equal for all inputs, then no change in slack values



Change in slack exists

Insight for Cyclic Graphs

- Use token knowledge
  - Count tokens from change point to every input
  - If value is equal for all inputs, then no change in slack values



No change in slack

Algorithm Summary

- Initially, find # tokens between every pair of nodes in the graph
  - Problem formulated as a flow lattice
  - Invoked once, complexity is $O(|M_0|\ |V|)$
- After inducing every change in graph
  - Compute slack change at each node
  - Propagate new change to neighboring outputs
  - Overall complexity is $O(|V|)$

Outline

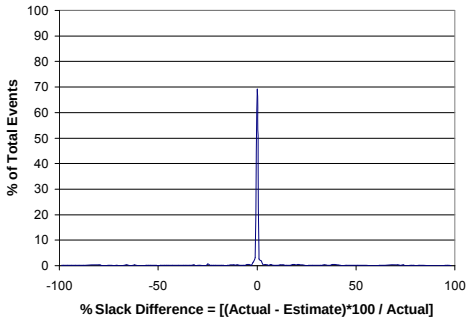
- Motivation
- Timing Metrics
- Slack Update Algorithm
- Experimental Evaluation
- Conclusions

Experimental Setup

- Slack Update loop incorporated into CASH compiler [ASPLOS 04, DAC 07]
  - Synthesizes asynchronous circuits from ANSI-C programs
- Applied three different optimizations
  - SM: Slack Matching [ICCAD 06]
  - OC: Operation Chaining [ICCAD 07]
  - ASU: Heterogeneous Pipeline Synthesis [Async 08]
- Benchmarks: Fifteen frequently executed kernels from Mediabench suite, [Lee 97]
- All results are post-synthesis mapped to ST Micro 180nm library

Absolute Accuracy

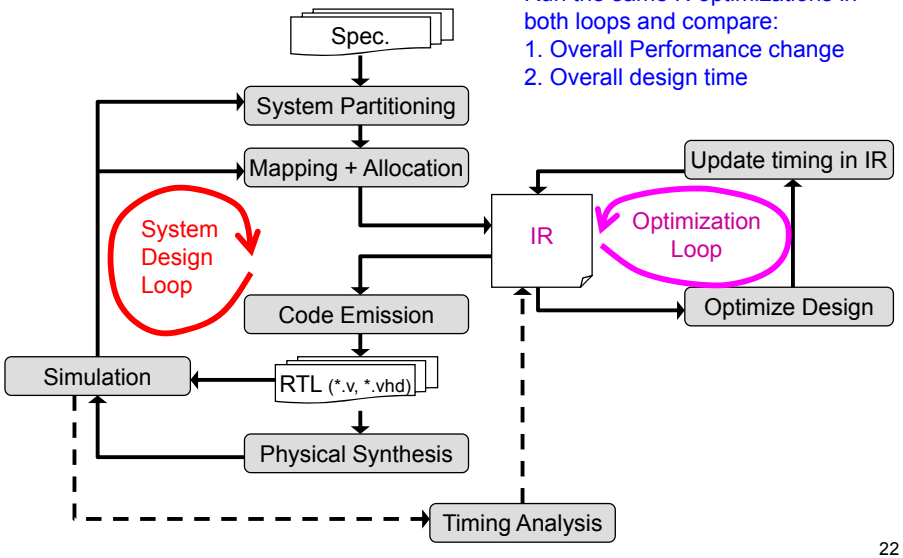
- After SM, compare computed values of slack against actual values of slack for `adpcm_d`



- Close to 100 changes applied (algo invoked for each change)
- 1.2x performance speedup
- Update inaccuracy due to unknown latency values during circuit transformation

Design Loop Experiments

Run the same N optimizations in both loops and compare:
1. Overall Performance change
2. Overall design time



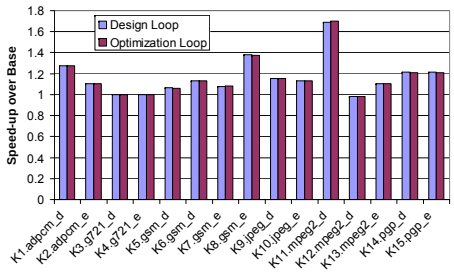
Design Loop Experiments

- Three optimization sequences
  1. SM-ASU: Slack matching followed by Heterogenous latch insertion
  2. ASU-SM
  3. SM-ASU-OC

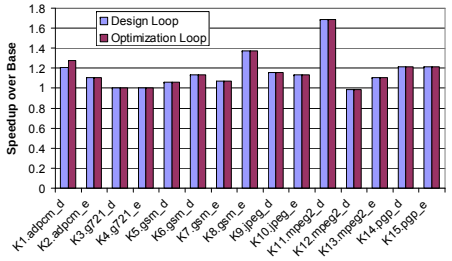
- Compare final circuit timing and loop traversal (design) times between the two methodologies

Design Loop Experiments

SM-ASU

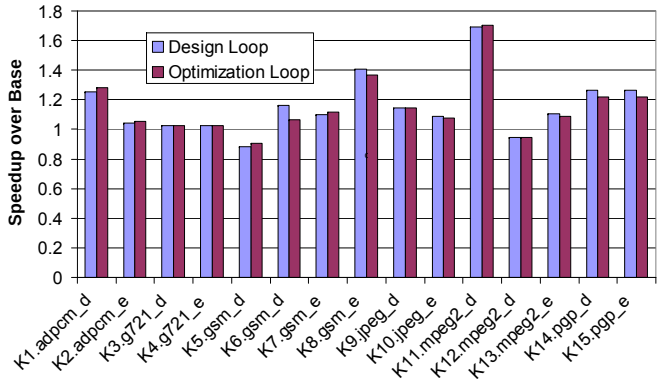


ASU-SM



- About ~500 circuit changes, on average
- Design Loop time: 0.5 – 4 hours
- Optimization loop: 10 - 100 seconds

Design Loop Experiments: SM-ASU-OC



- About ~1000-3000 circuit changes, on average
- Design Loop time: 1 – 10 hours
- Optimization loop: 20 - 200 seconds

300x Speedup

Outline

- Motivation
- Timing Metrics
- Slack Update Algorithm
- Experimental Evaluation

- Conclusions

Conclusions

- Slack update algorithm to speed up design loop in TLM-based workflows
- Use slack to track system-level timing changes
- Orders of magnitude reduction in design time at negligible loss in accuracy
- New optimizations loop enables scalability in iterative system design flows