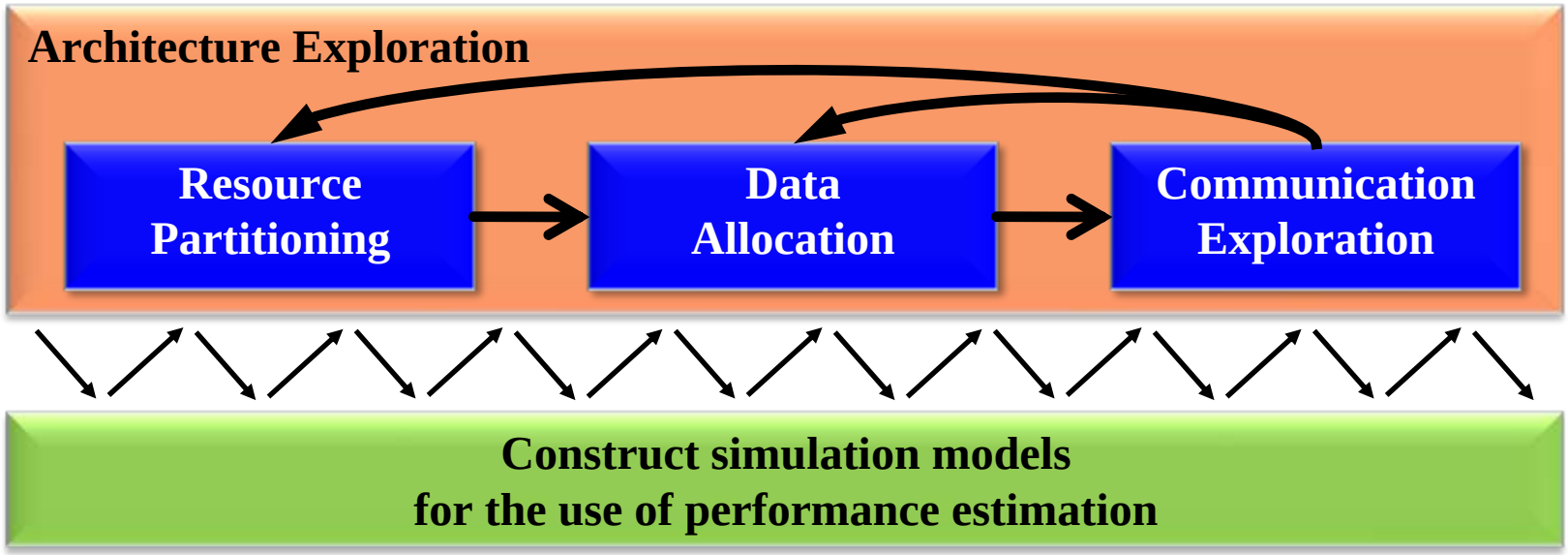
A Fast and Effective Dynamic Trace-based Method for Analyzing Architectural Performance

***Yi-Siou Chen, Lih-Yih Chiou, Hsun-Hsiang Chang**
Dept. of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan, 70101, R.O.C.



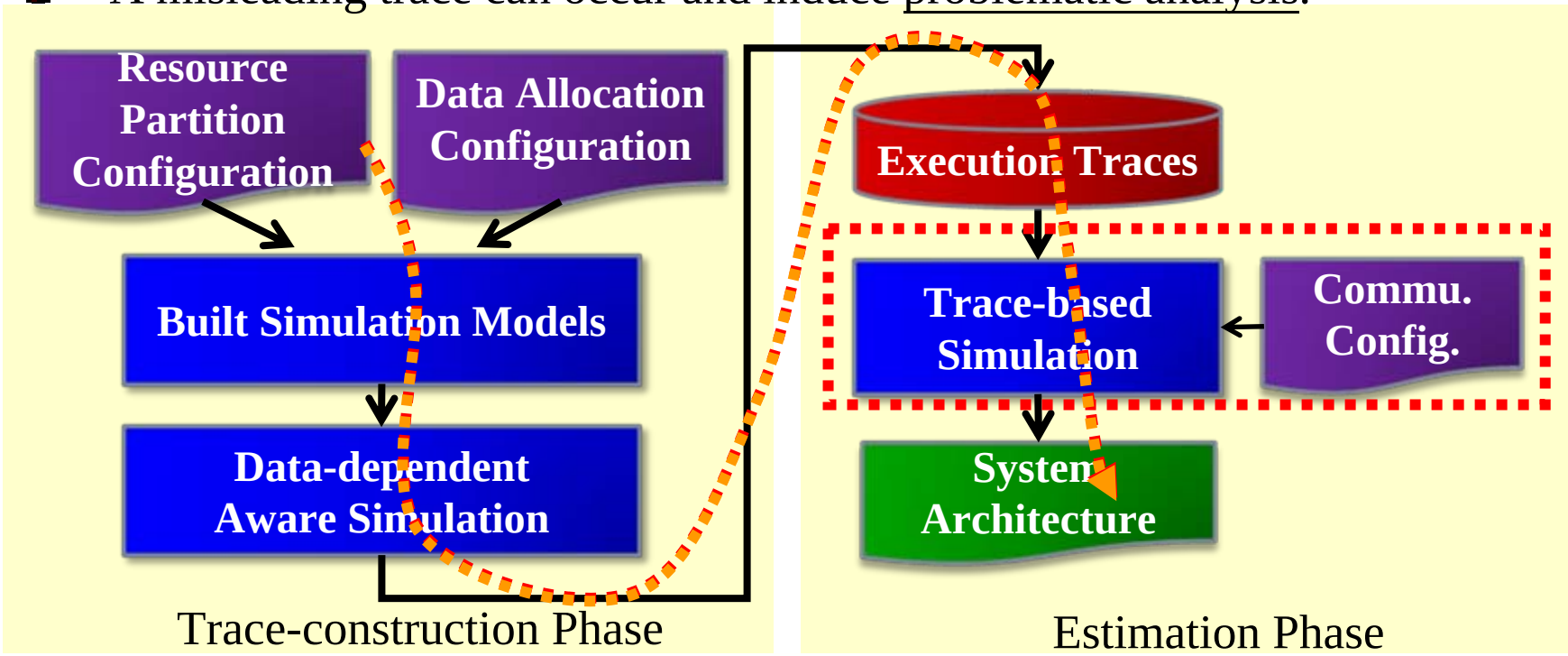
Introduction

- Designers must spend a tremendous amount of time in system remodeling for performance estimation for each alternative solution in a huge design space.
- This work is motivated by the needs to reduce remodeling effort and quickly estimate system quality in various resource partition, data allocation, and communication architecture exploration.

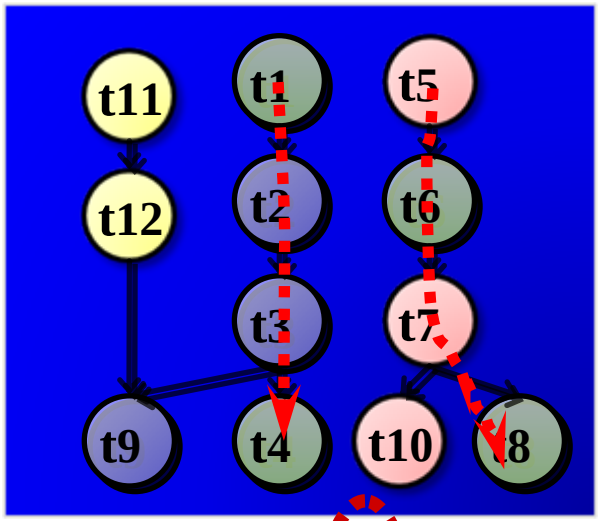


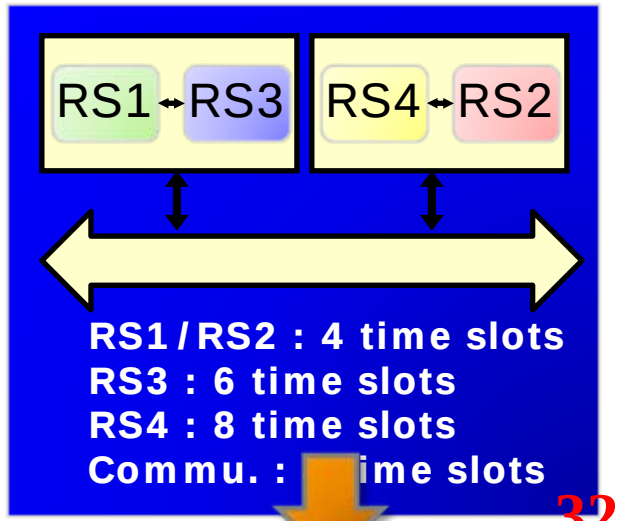
Static Trace-Based Approach (1/2)

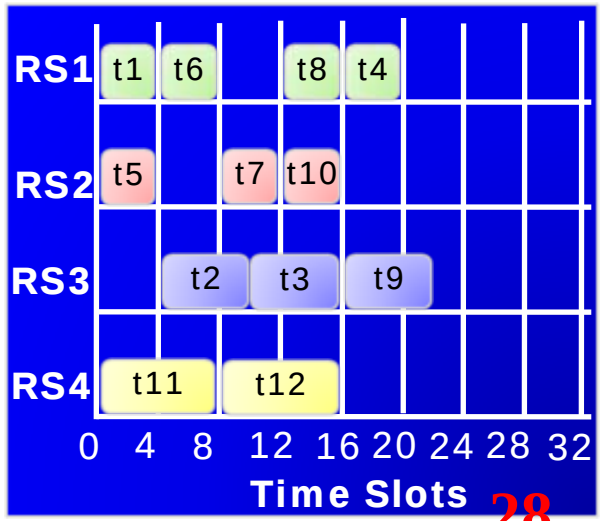
- This approach generates traces using data-dependent aware simulation without communication latency and estimates system performance based on a simulation of the traces.
- The effort needed for trace reconstruction becomes a bottleneck in the whole design process.
- A misleading trace can occur and induce problematic analysis.

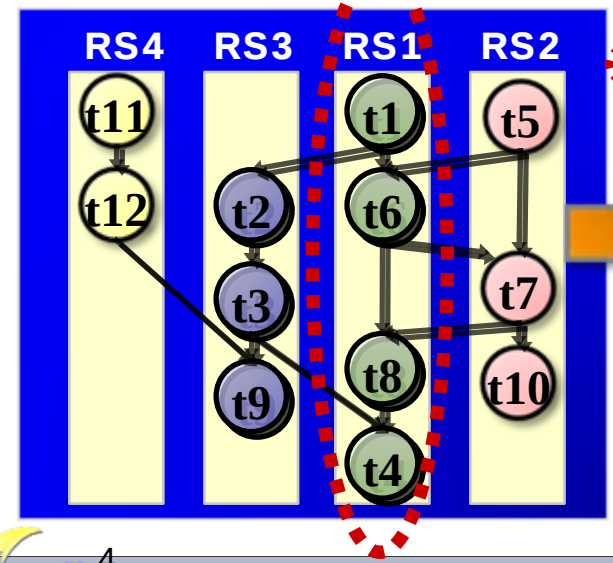


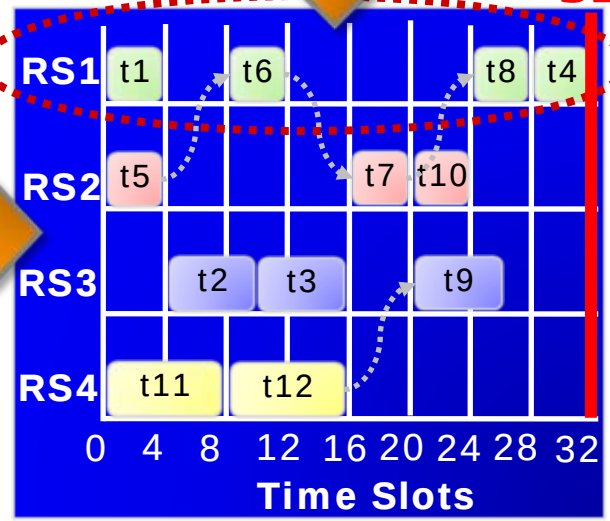
Static Trace-Based Approach (2/2)

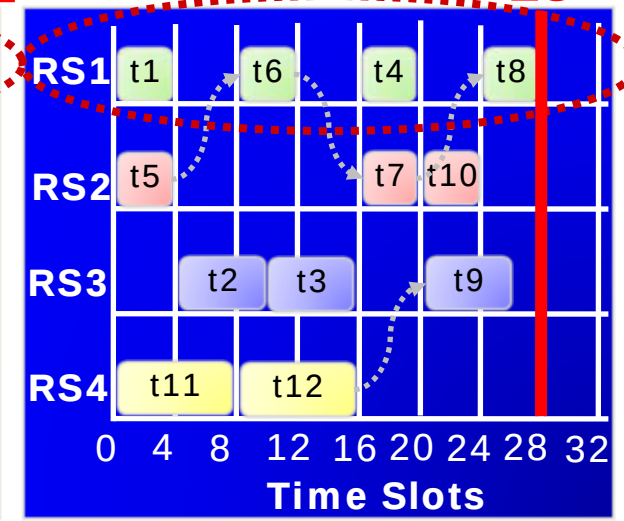






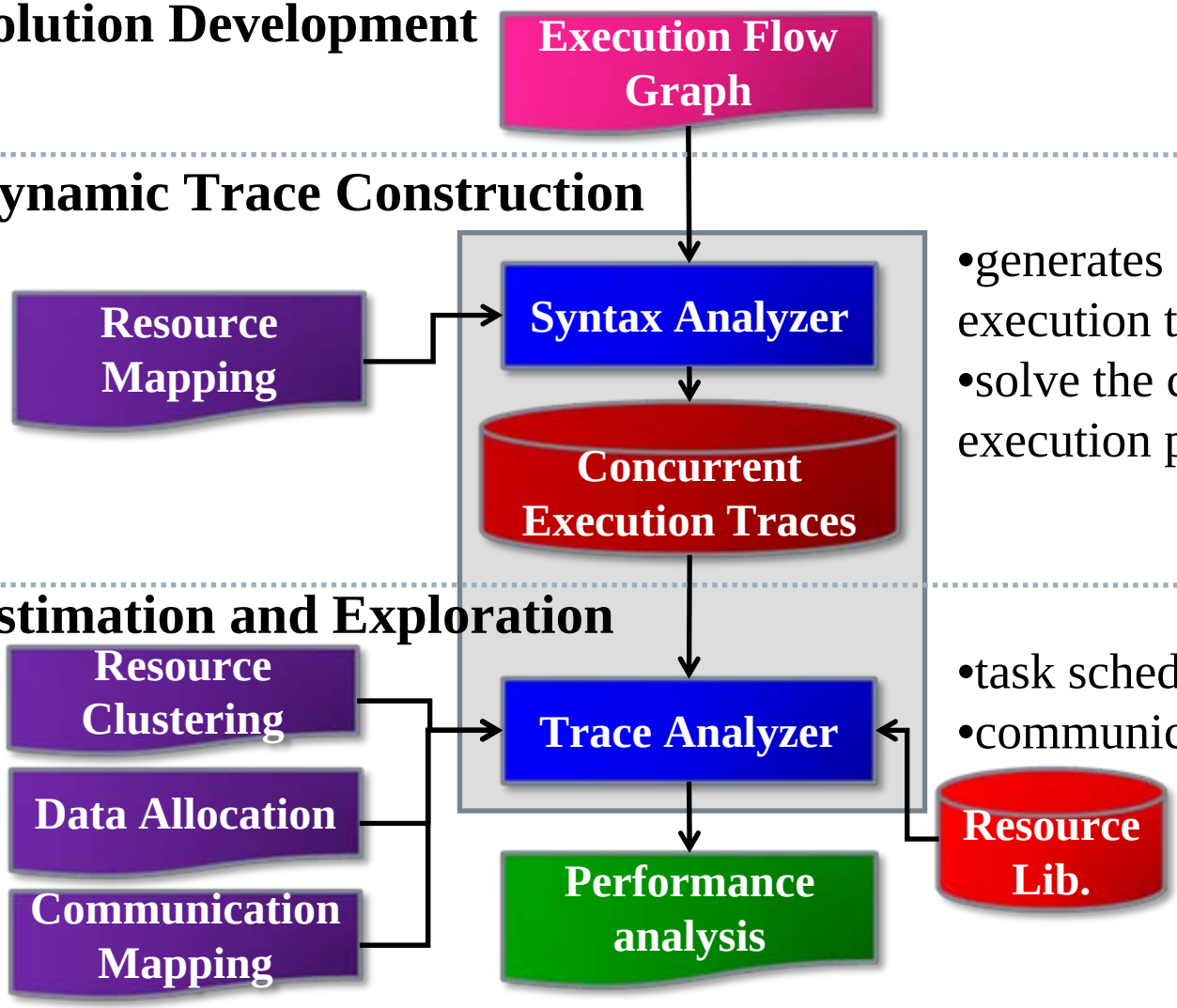






Proposed Dynamic Trace-Based Approach

- **Solution Development**
- **Dynamic Trace Construction**
- **Estimation and Exploration**

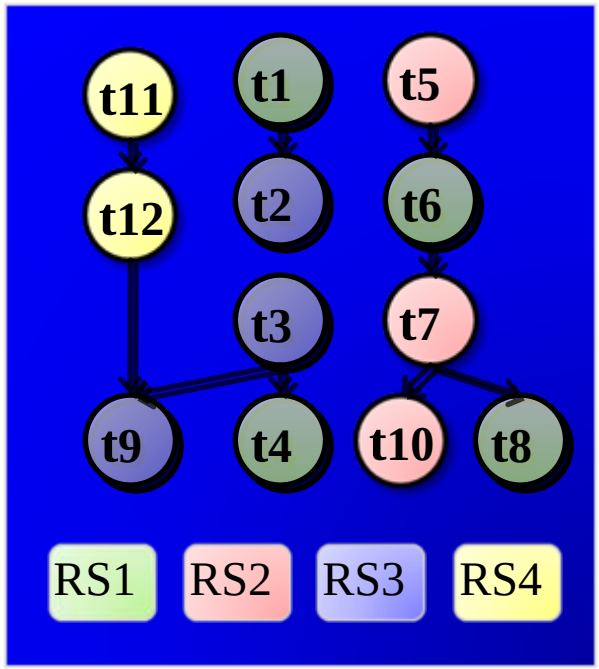


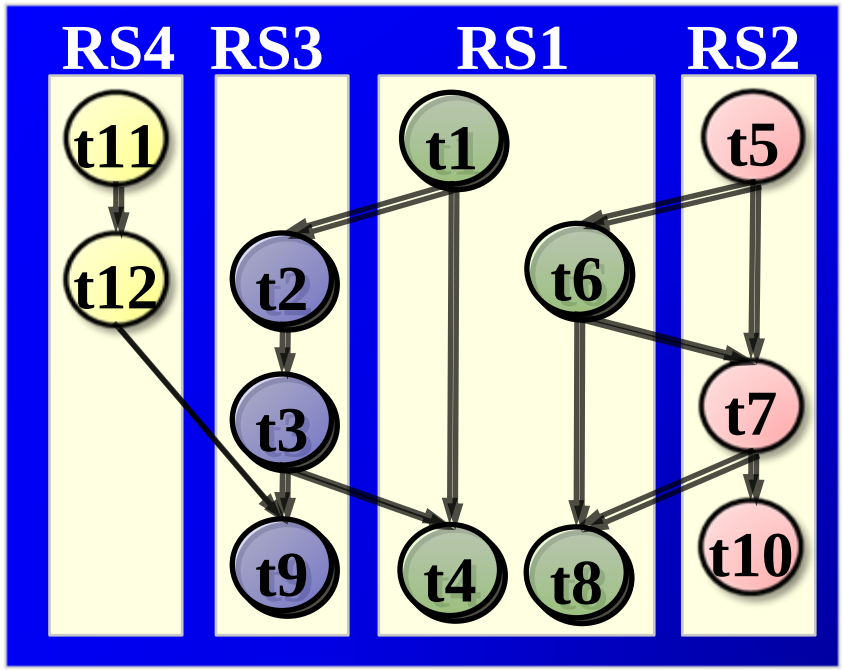
- generates concurrent execution traces automatically
- solve the competing-for-execution problem

- task scheduler
- communication scheduler

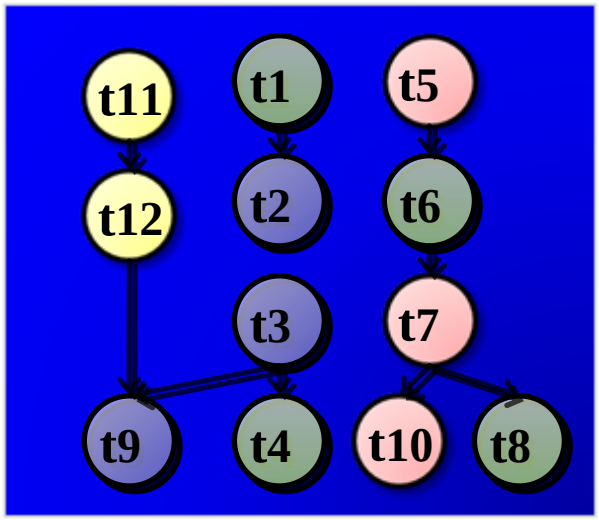
Concurrent Execution Traces

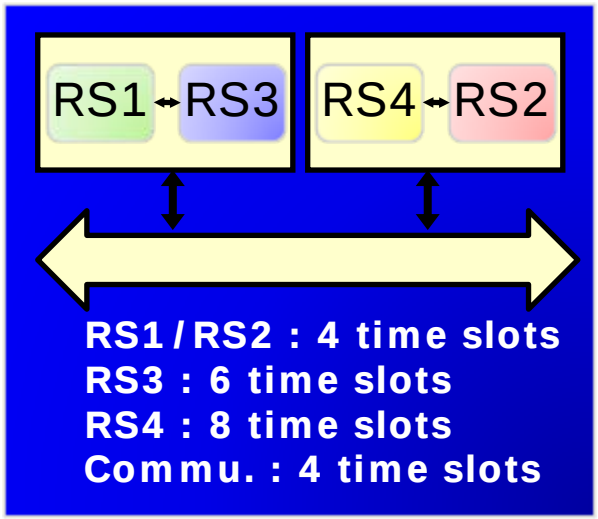
- The vertex set $V_j=\{v \in V: RM(v)=j\}$, where $RM(v)=j$ indicates that vertex v is mapped onto resource r_j. We assume $v_p, v_q \in V_j$; thus, a trace is generated when a path exists from v_p to v_q in $EFG(V,E)$, i.e., v_p and v_q are executed dependently.

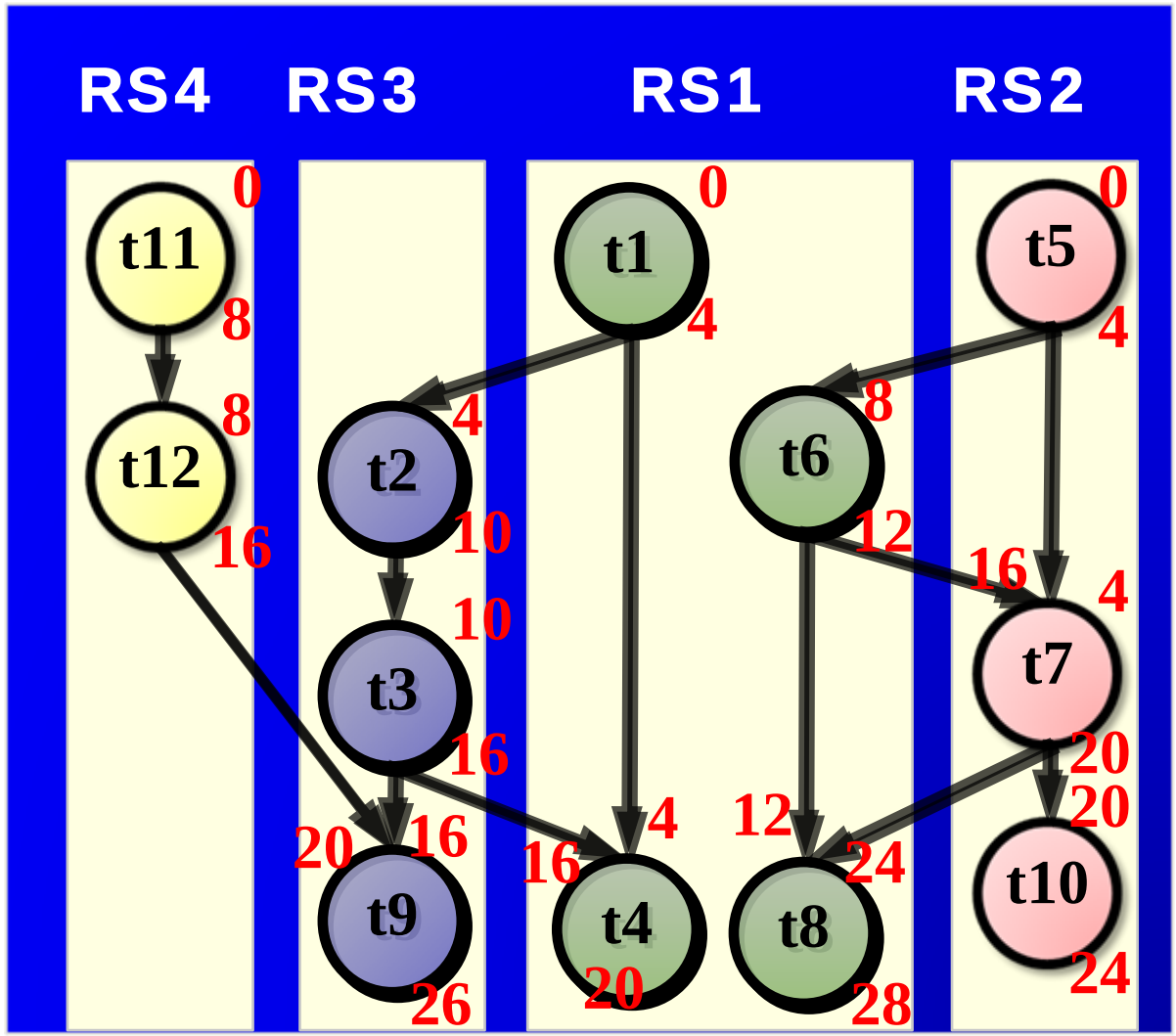




Trace analyzer



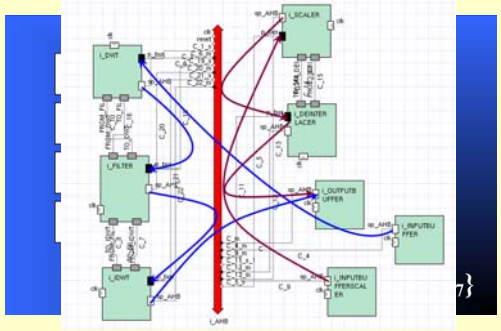




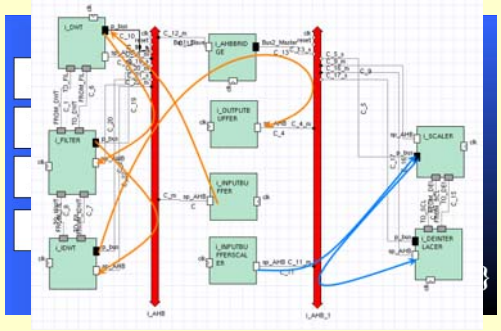
Experimental Results

- Image processing application
  - Compare with the results of bus-functional simulation using CoWare.
  - Five architectures, two bus priority settings, three frames sizes

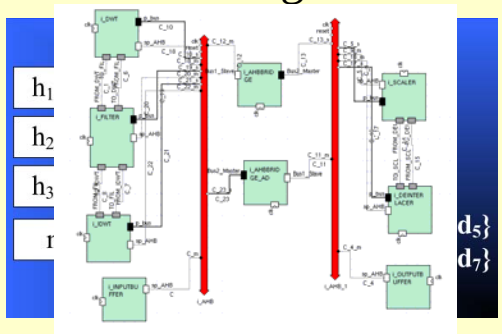
➢Arch1: Resource partition



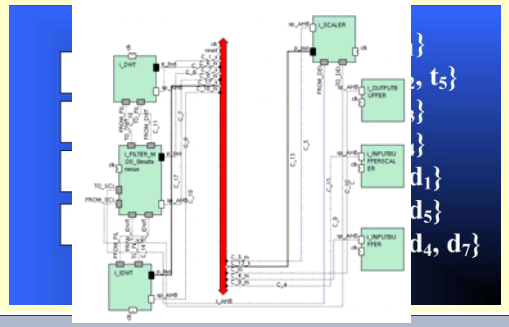
➢Arch2: Split bus



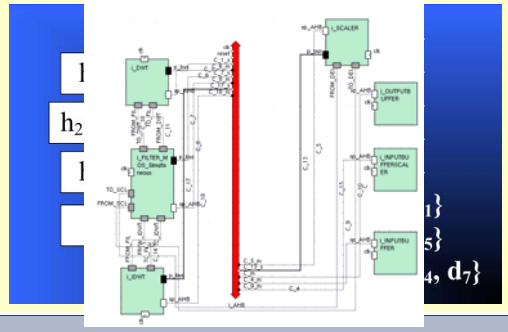
➢Arch3: Change data allocation



➢Arch4: Competing for execution

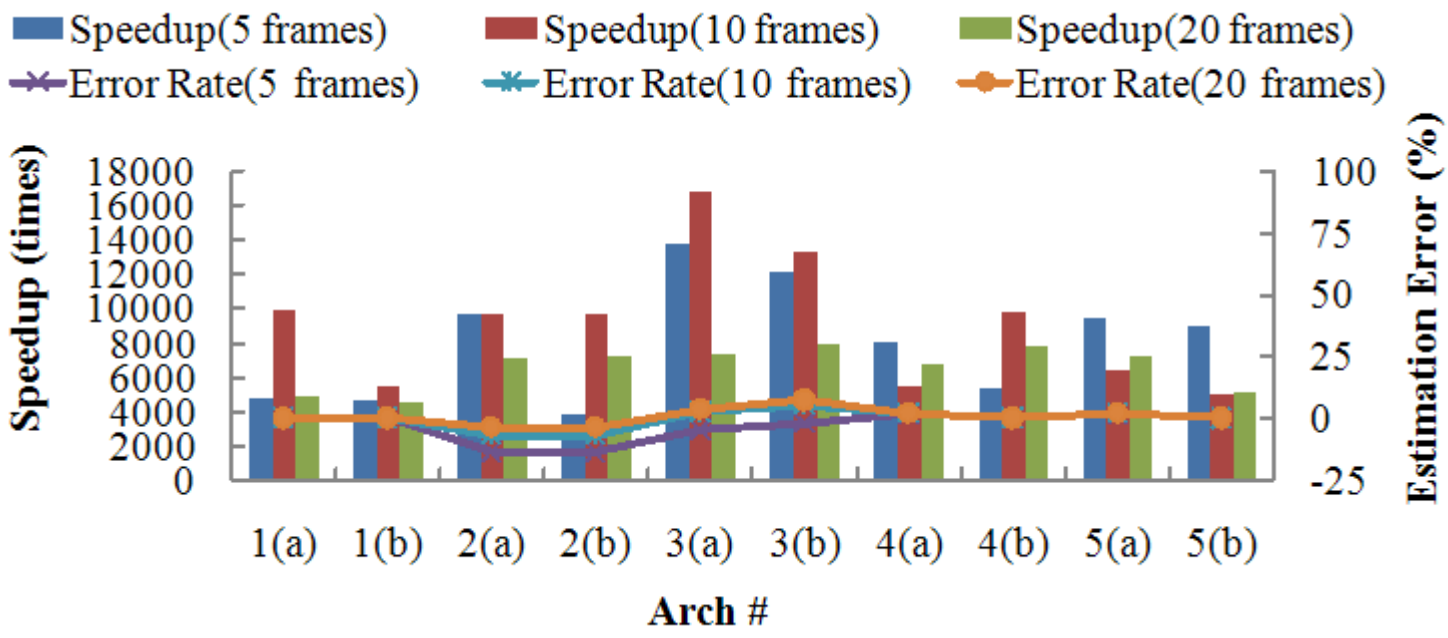


➢Arch5: Change the clustering of resources



Experimental Results

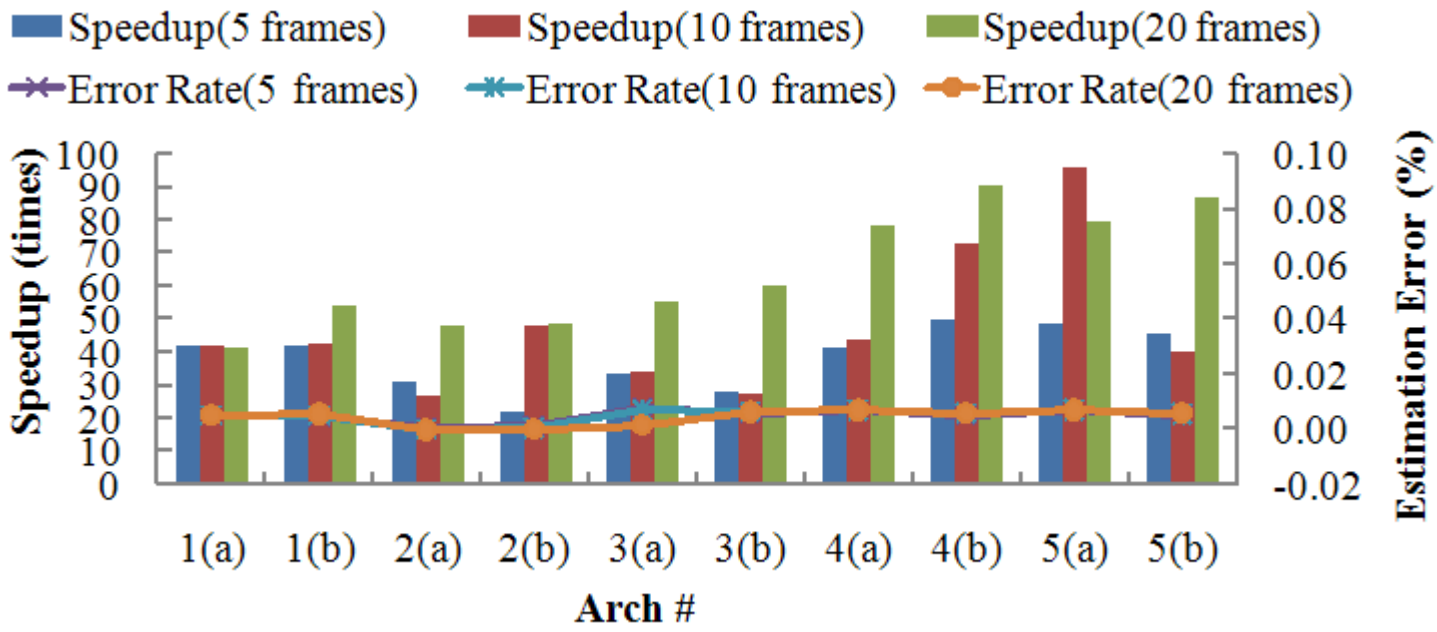
- The estimate by the APDT approach without a burst mode deviated from that by CoWare by 2.9% on average, but the simulation time was 7921 times faster.



➢ w/o burst mode

Experimental Results

- The estimation error rate was 0–0.008% and the approach with a burst mode was on average 50 times faster than CoWare.



➢ w/i burst mode

Conclusions

- The APDT approach uses the concurrent execution trace to ensure the proper order of task executions, and allows designers to explore architecture with various designs without remodeling the system for trace reconstruction.
- Experimental results demonstrate that the APDT approach is faster than the bus functional-level simulation on CoWare with minor estimation deviation.
- In the future, an efficient exploration algorithm will be developed with the aid of the APDT approach.