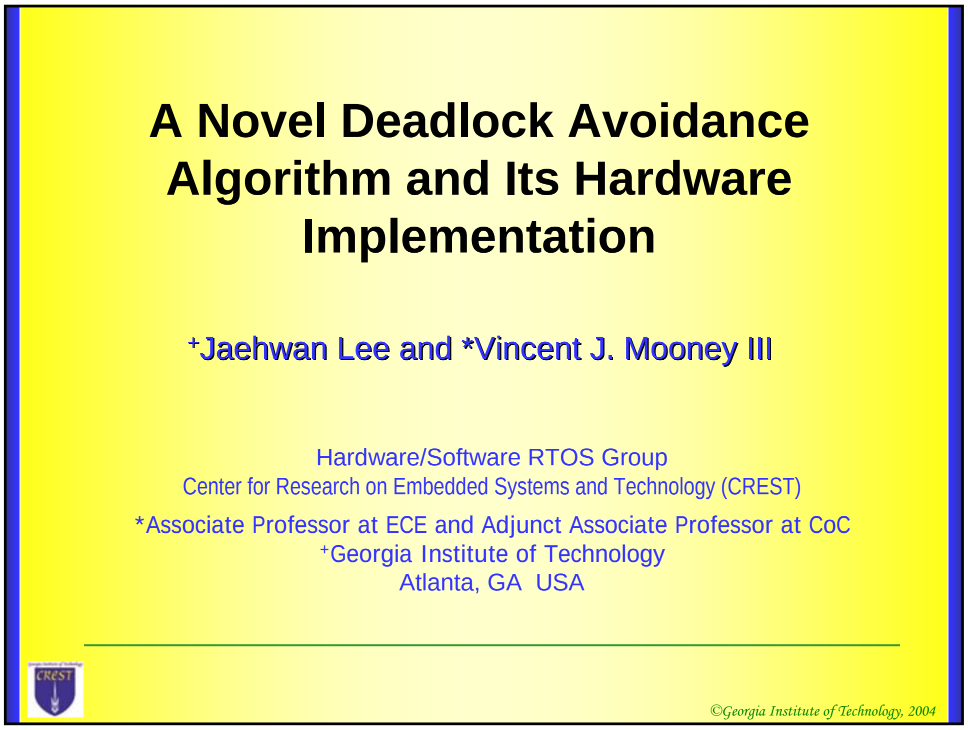
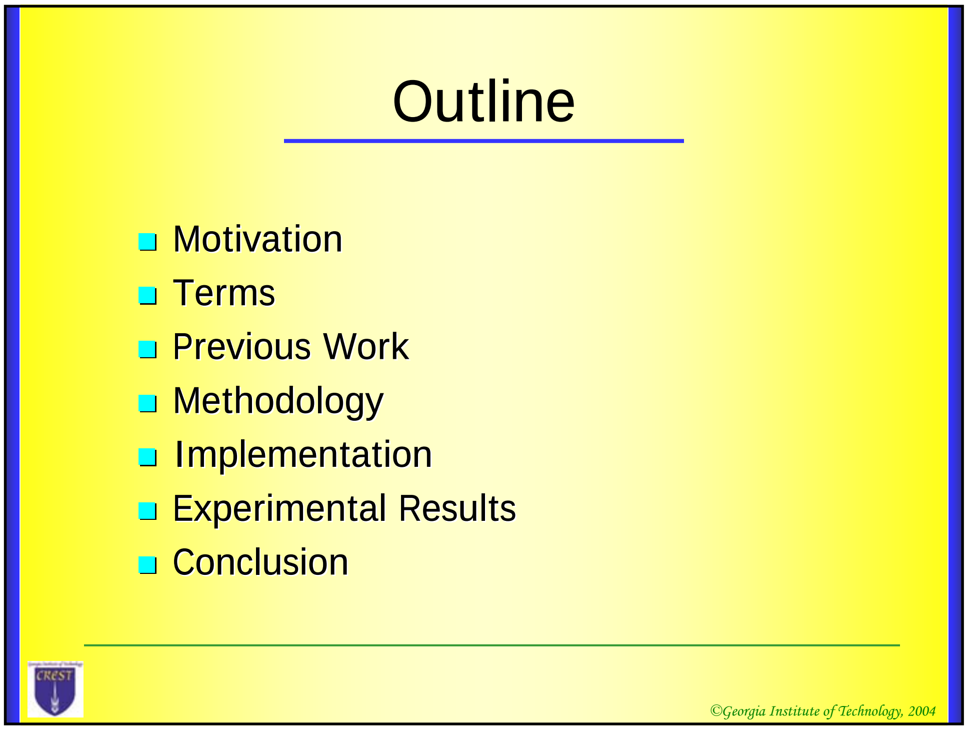


A Novel Deadlock Avoidance Algorithm and Its Hardware Implementation

[+]Jaehwan Lee and *Vincent J. Mooney III

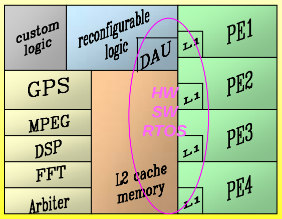
Hardware/Software RTOS Group
Center for Research on Embedded Systems and Technology (CREST)
*Associate Professor at ECE and Adjunct Associate Professor at CoC
[+]Georgia Institute of Technology
Atlanta, GA USA



Outline

Motivation
- Technology Trends

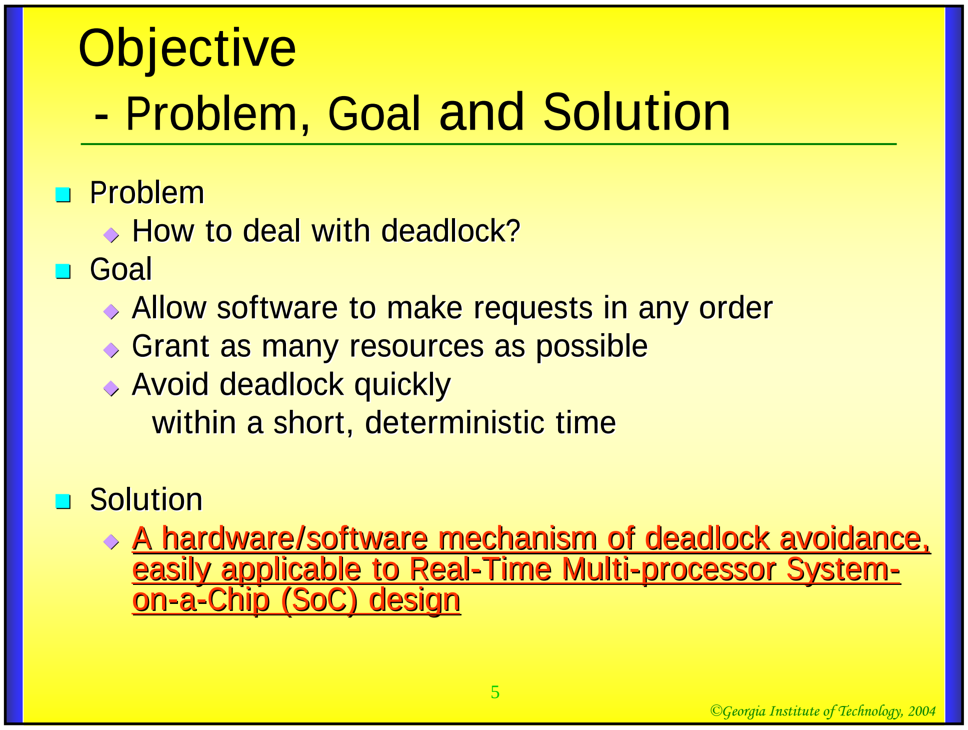
- Future SoC's
  - Multiple heterogeneous processors (tens of processes)
  - Multiple on-chip hardware resources
    - DSP, FFT, MPEG, GPS, Shared Memory, etc
  - Current example
    - Xilinx Virtex-II Pro FPGAs include multiple PowerPC processors
- Processes in such an SoC
  - Dynamically request and use resources
  - May end up in deadlock
- Current embedded system or single processor system
  - Typically ignored today



SoC: System on Chip

Motivation
- Does deadlock really matter?

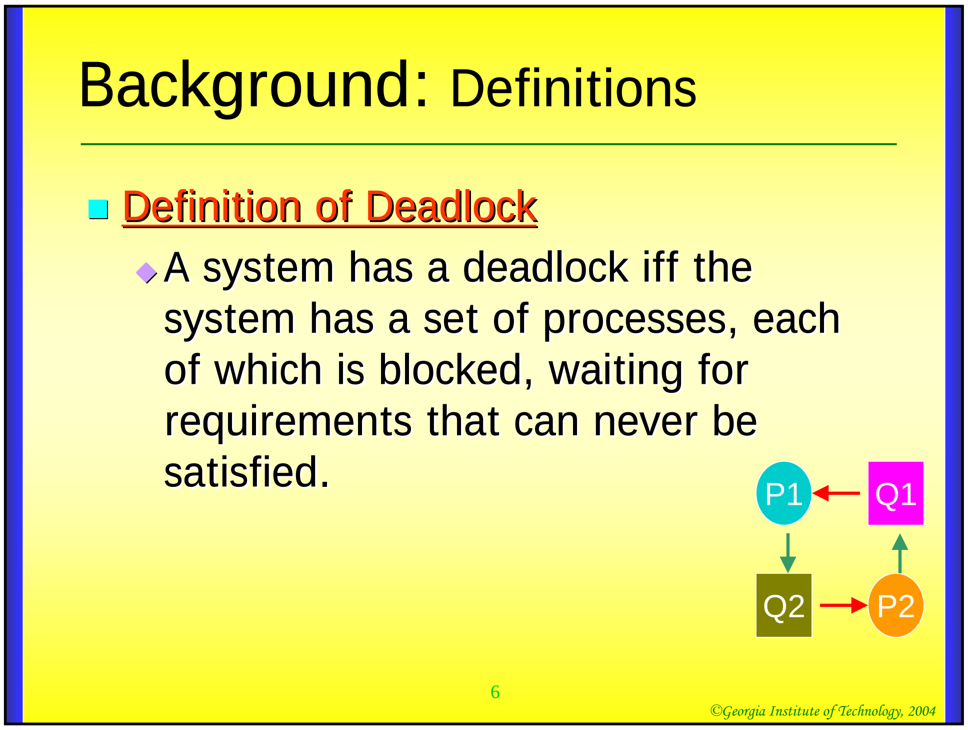
- Examples of future real-time systems
  - Human-like robot with multiple processes
    - Deadlock
      - Damage
      - People get injured
      - Law-suits
  - Mars Rover
    - Deadlock
      - Loss of millions of dollars
      - Time loss
  - Industrial control
  - Cars
- Even if low probability of deadlock, prefer no deadlock whatsoever



Objective
- Problem, Goal and Solution

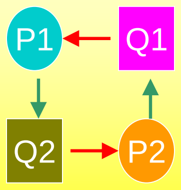
- Problem
  - How to deal with deadlock?
- Goal
  - Allow software to make requests in any order
  - Grant as many resources as possible
  - Avoid deadlock quickly
    within a short, deterministic time
- Solution
  - A hardware/software mechanism of deadlock avoidance, easily applicable to Real-Time Multi-processor System-on-a-Chip (SoC) design

5

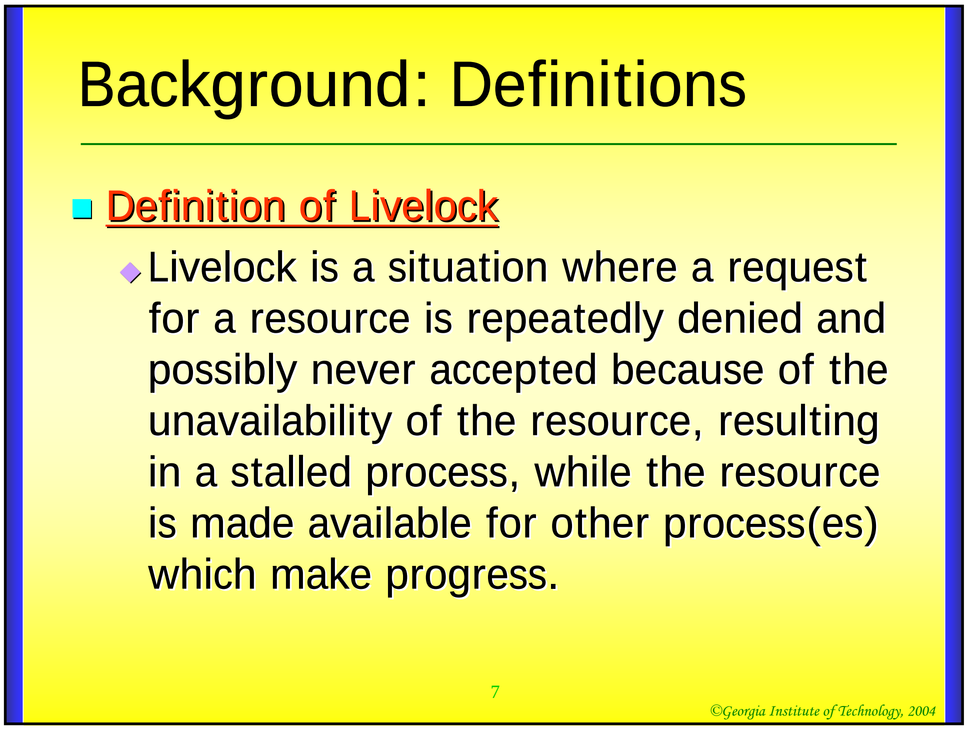


Background: Definitions

- Definition of Deadlock
  - A system has a deadlock iff the system has a set of processes, each of which is blocked, waiting for requirements that can never be satisfied.

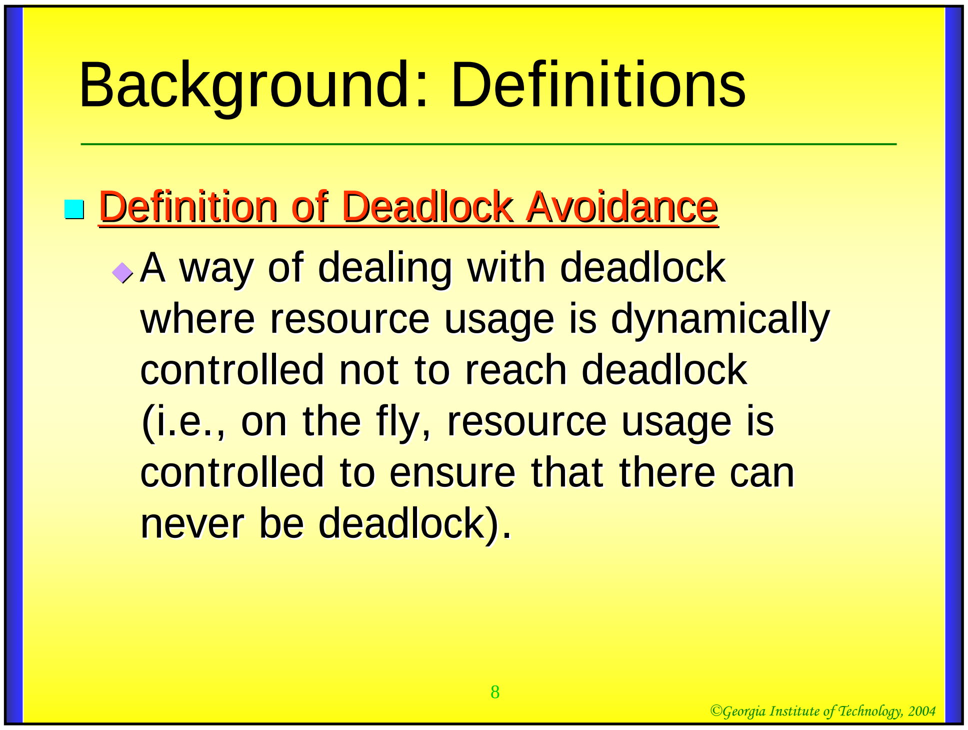


6



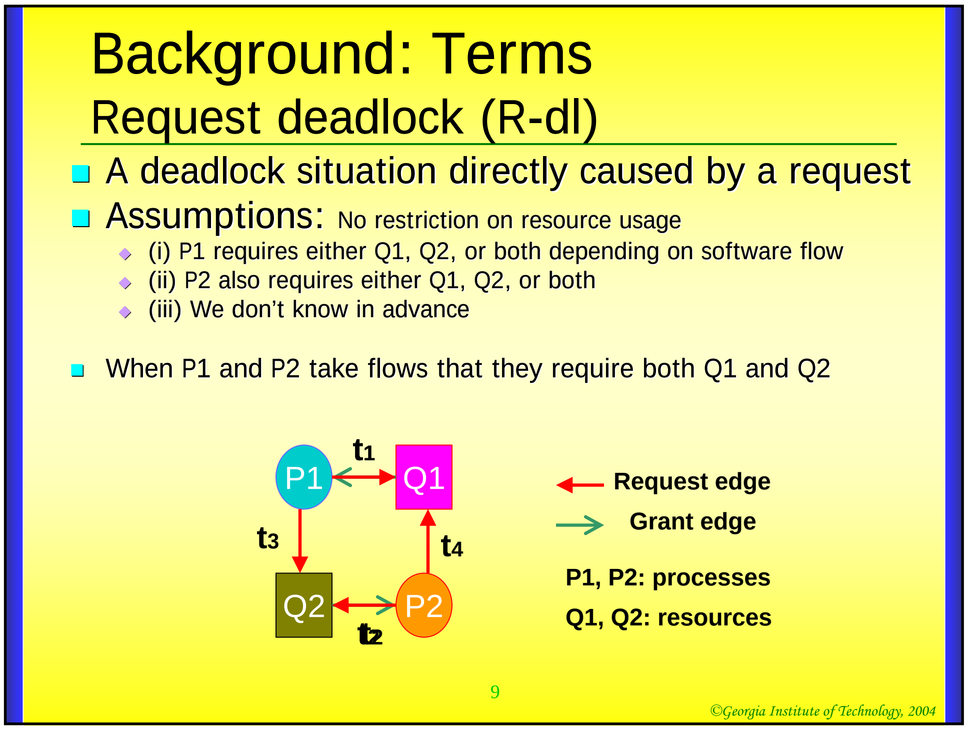
Background: Definitions

- <u>Definition of Livelock</u>
  - Livelock is a situation where a request for a resource is repeatedly denied and possibly never accepted because of the unavailability of the resource, resulting in a stalled process, while the resource is made available for other process(es) which make progress.



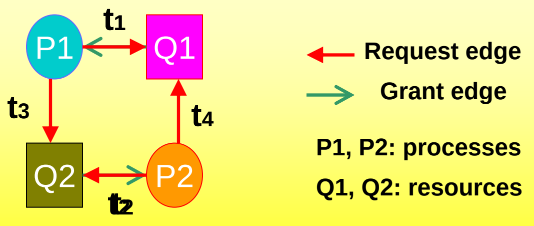
Background: Definitions

- <u>Definition of Deadlock Avoidance</u>
  - A way of dealing with deadlock where resource usage is dynamically controlled not to reach deadlock (i.e., on the fly, resource usage is controlled to ensure that there can never be deadlock).



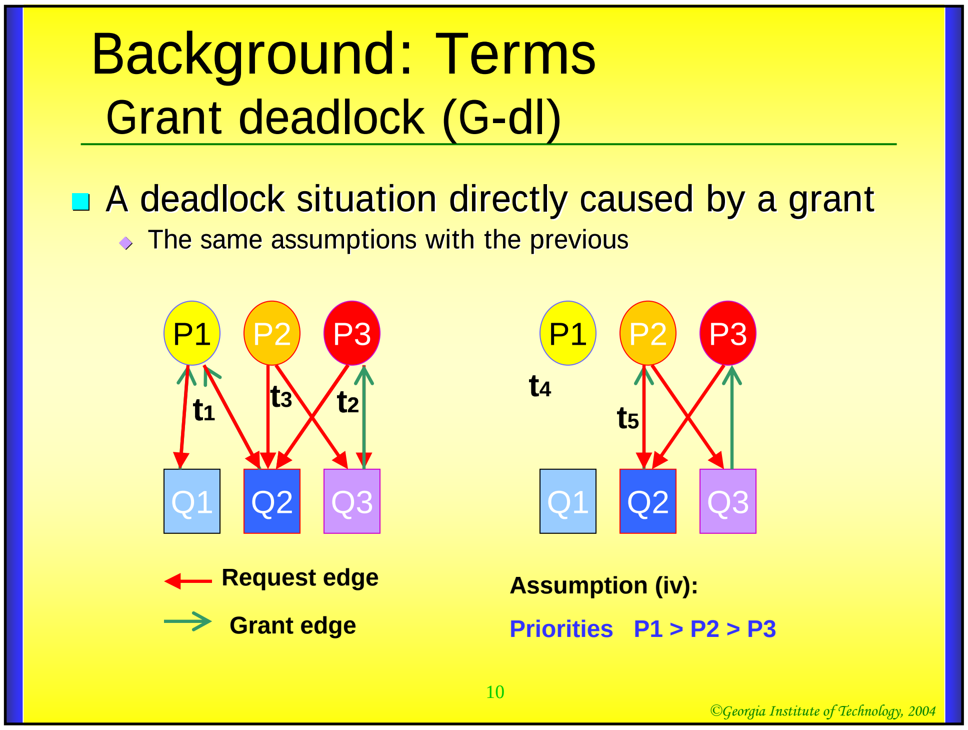
Background: Terms
Request deadlock (R-dl)

- A deadlock situation directly caused by a request
- Assumptions: No restriction on resource usage
  - (i) P1 requires either Q1, Q2, or both depending on software flow
  - (ii) P2 also requires either Q1, Q2, or both
  - (iii) We don't know in advance
- When P1 and P2 take flows that they require both Q1 and Q2



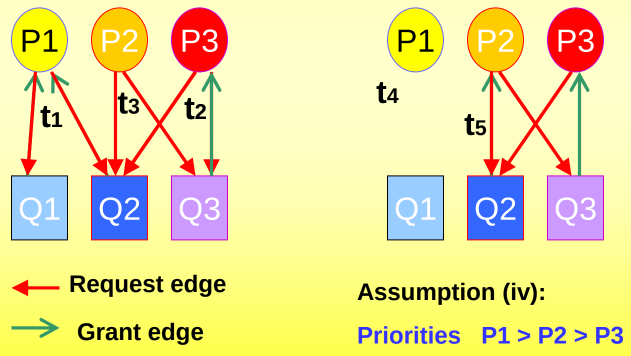
9

©Georgia Institute of Technology, 2004



Background: Terms
Grant deadlock (G-dl)

- A deadlock situation directly caused by a grant
  - The same assumptions with the previous

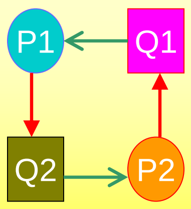


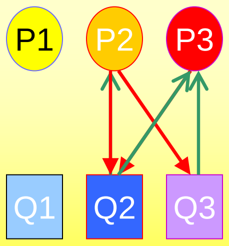
10

©Georgia Institute of Technology, 2004

Differentiation between R-dl and G-dl

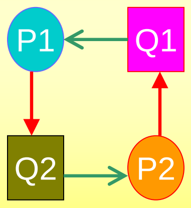
- **Reason**
  - Some actions can only be taken for either R-dl or G-dl. (E.g., G-dl could have been avoided by granting Q2 to P3 instead of P2 in the previous G-dl example.)

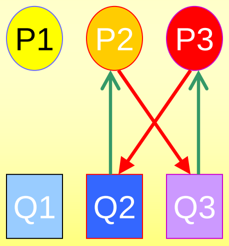




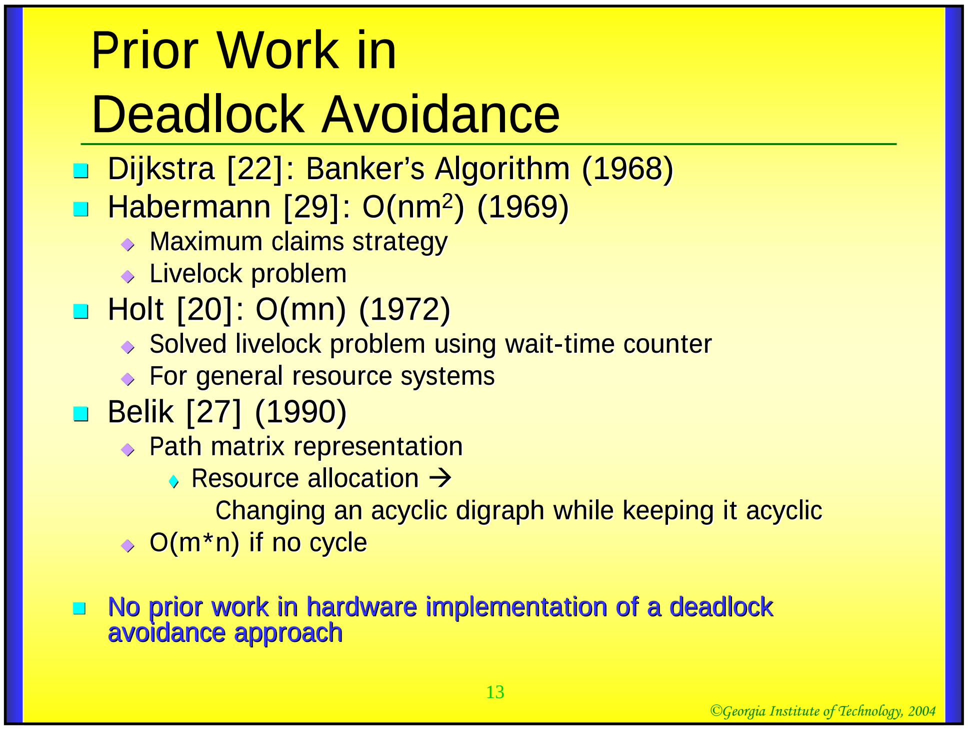
How to find deadlock - Deadlock and Cycle Relation

- Deadlock $\Rightarrow$ $\exists$ Cycle(s) in a RAG
- Cycle(s) $\Rightarrow$ $\exists$ Deadlock





- RAG : Resource Allocation Graph



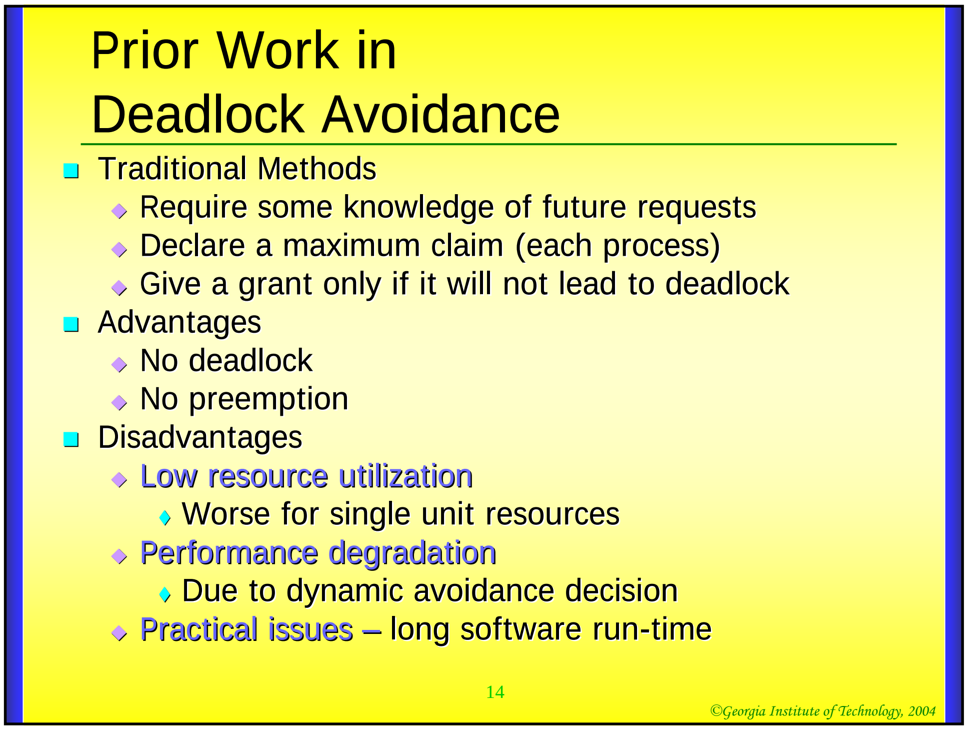
Prior Work in Deadlock Avoidance

- Dijkstra [22]: Banker's Algorithm (1968)
- Habermann [29]: O(nm^2) (1969)
  - Maximum claims strategy
  - Livelock problem
- Holt [20]: O(mn) (1972)
  - Solved livelock problem using wait-time counter
  - For general resource systems
- Belik [27] (1990)
  - Path matrix representation
    - Resource allocation $\rightarrow$
      Changing an acyclic digraph while keeping it acyclic
  - O(m*n) if no cycle

- No prior work in hardware implementation of a deadlock avoidance approach

13

©Georgia Institute of Technology, 2004

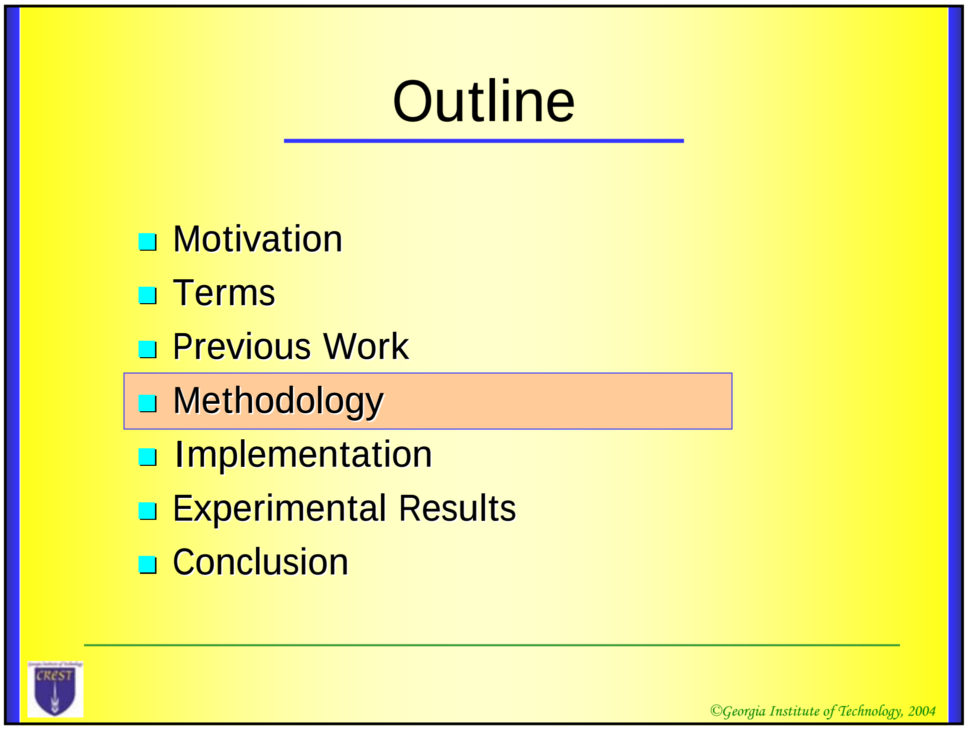


Prior Work in Deadlock Avoidance

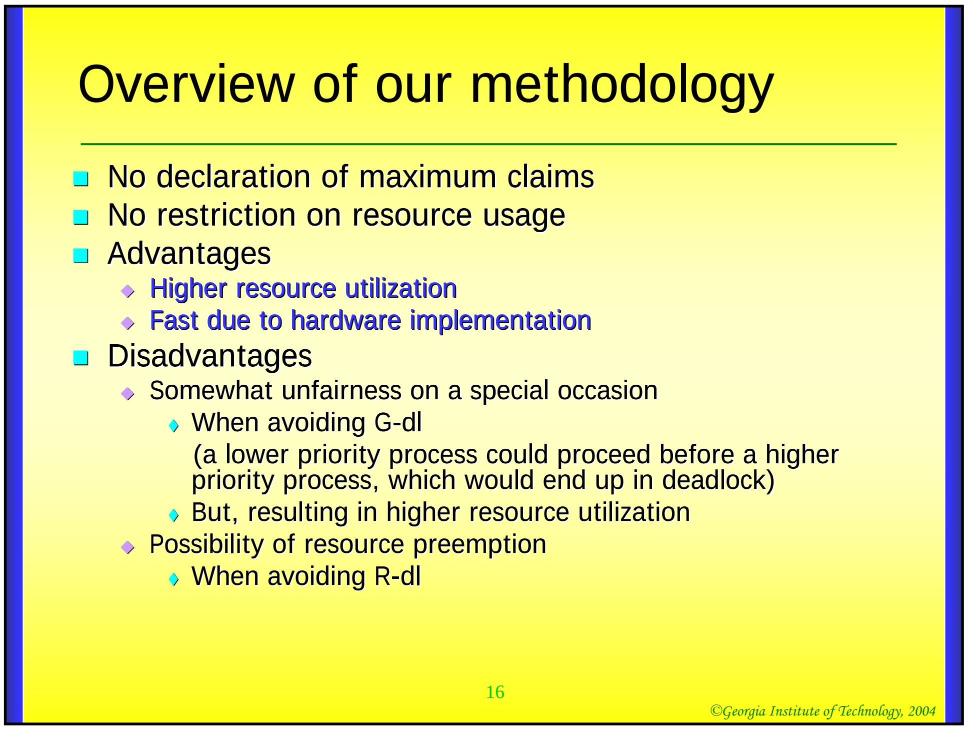
- Traditional Methods
  - Require some knowledge of future requests
  - Declare a maximum claim (each process)
  - Give a grant only if it will not lead to deadlock
- Advantages
  - No deadlock
  - No preemption
- Disadvantages
  - Low resource utilization
    - Worse for single unit resources
  - Performance degradation
    - Due to dynamic avoidance decision
  - Practical issues – long software run-time

14

©Georgia Institute of Technology, 2004



Outline

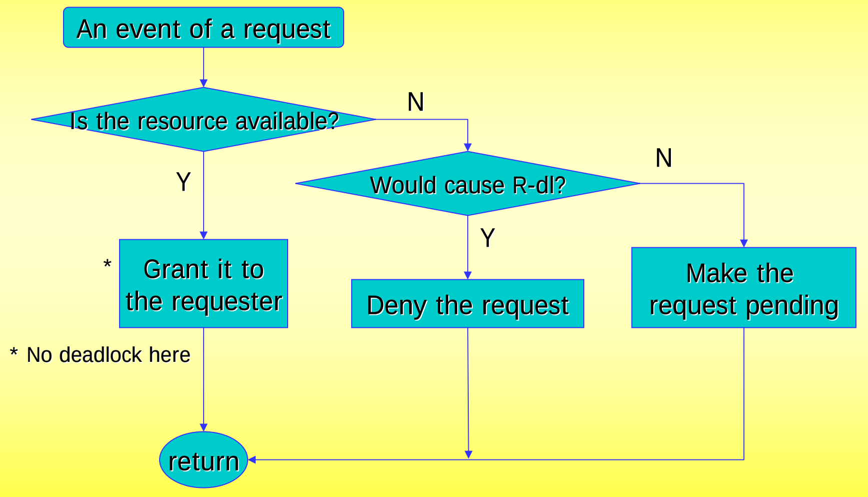


Overview of our methodology

- No declaration of maximum claims
- No restriction on resource usage
- Advantages
  - Higher resource utilization
  - Fast due to hardware implementation
- Disadvantages
  - Somewhat unfairness on a special occasion
    - When avoiding G-dl
      (a lower priority process could proceed before a higher priority process, which would end up in deadlock)
    - But, resulting in higher resource utilization
  - Possibility of resource preemption
    - When avoiding R-dl

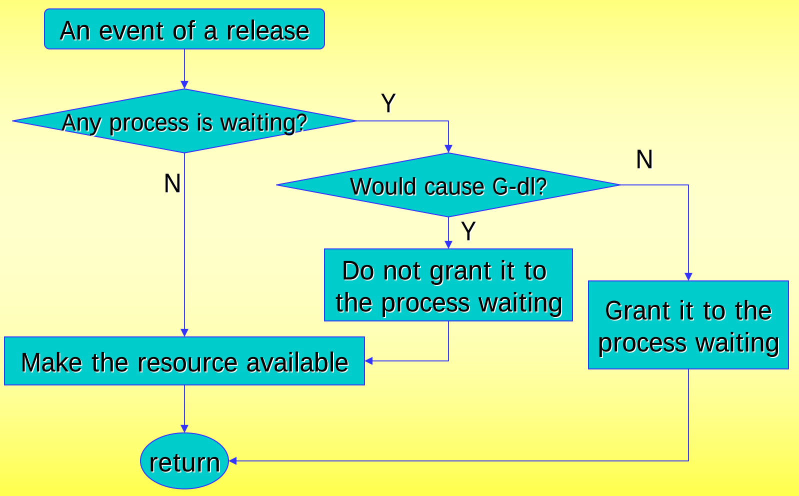
16

Deadlock Avoidance Algorithm (DAA) 1
- Basic approach



17

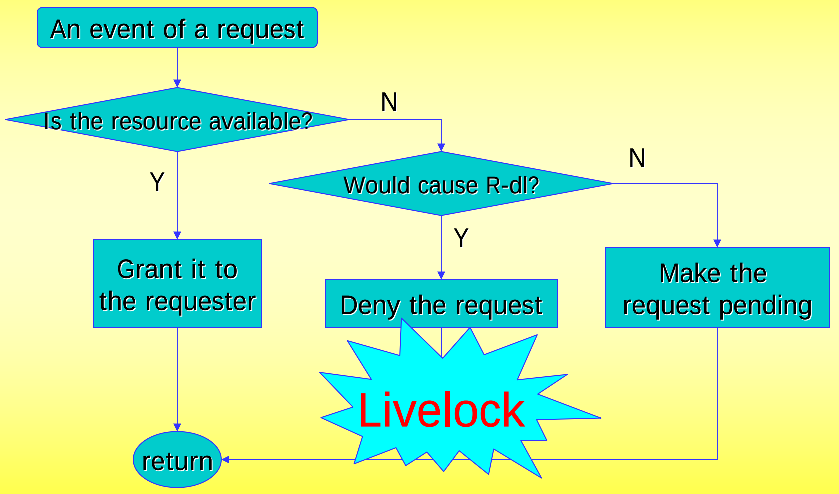
DAA 1 (cont'd)



An event of a release
Any process is waiting?
Y
N
Would cause G-dl?
N
Y
Do not grant it to the process waiting
Grant it to the process waiting
Make the resource available
return

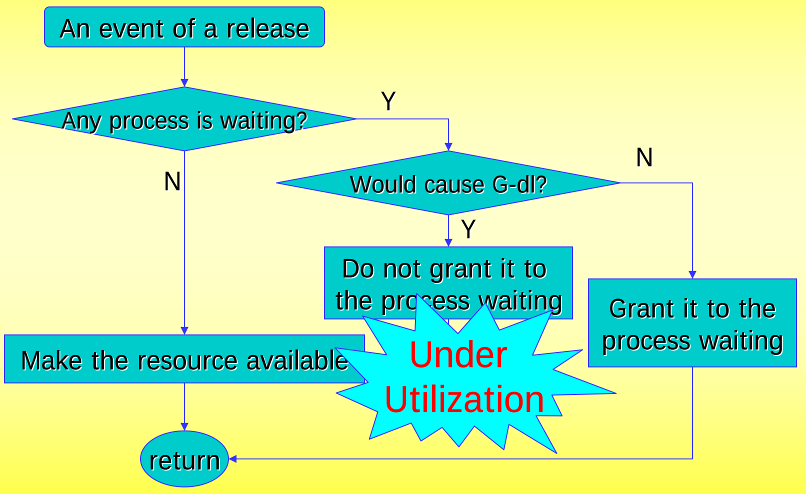
18

Problem of Deadlock Avoidance Algorithm (DAA) 1



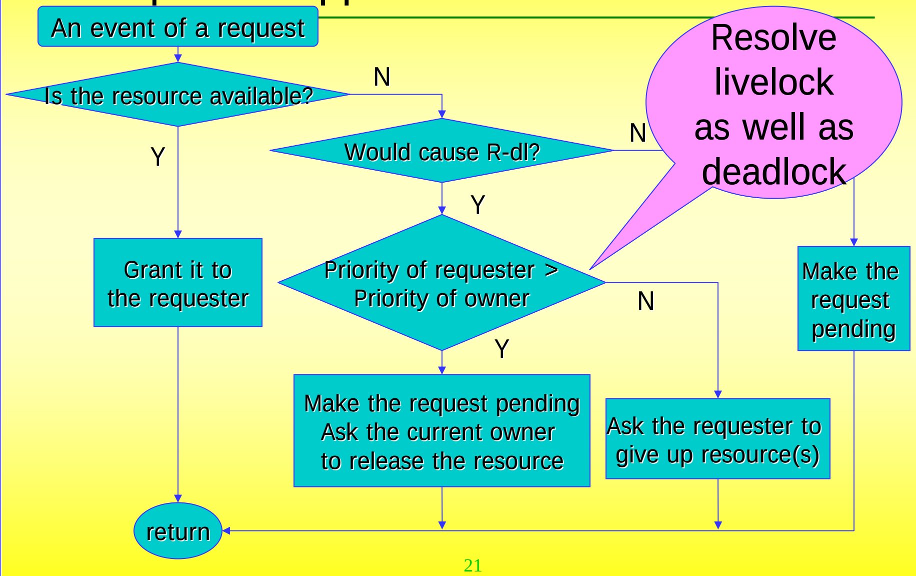
19

Problem of DAA 1 (cont'd)


An event of a release
Any process is waiting?
Y
N
Would cause G-dl?
N
Y
Do not grant it to the process waiting
Grant it to the process waiting
Make the resource available
Under Utilization
return

20

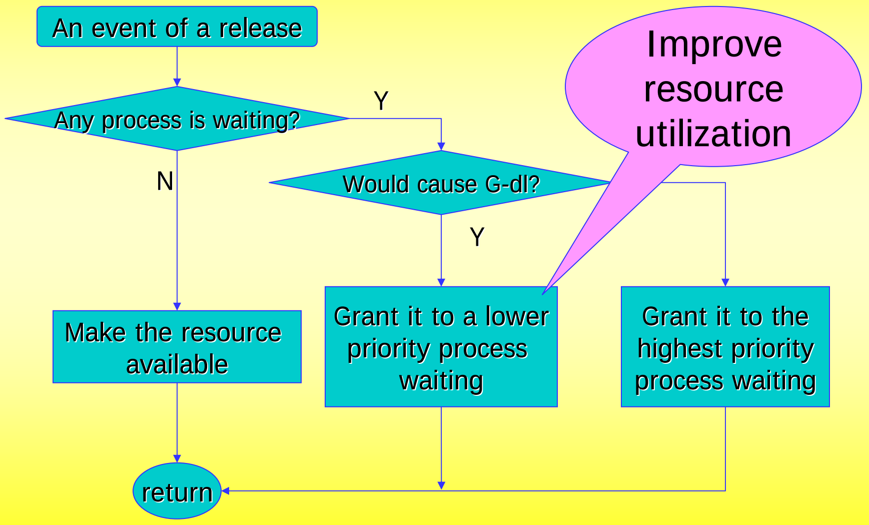
Deadlock Avoidance Algorithm (DAA) 2 - Improved Approach


An event of a request
Is the resource available?
N
Y
Would cause R-dl?
N
Y
Grant it to the requester
Priority of requester > Priority of owner
N
Y
Make the request pending
Make the request pending Ask the current owner to release the resource
Ask the requester to give up resource(s)
return
Resolve livelock as well as deadlock

21

©Georgia Institute of Technology, 2004

DAA 2 (cont'd)


An event of a release
Any process is waiting?
Y
N
Would cause G-dl?
Y
Make the resource available
Grant it to a lower priority process waiting
Grant it to the highest priority process waiting
return
Improve resource utilization

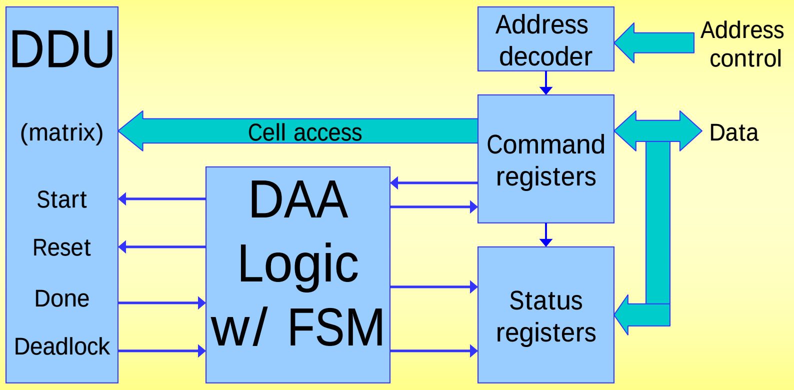
22

©Georgia Institute of Technology, 2004

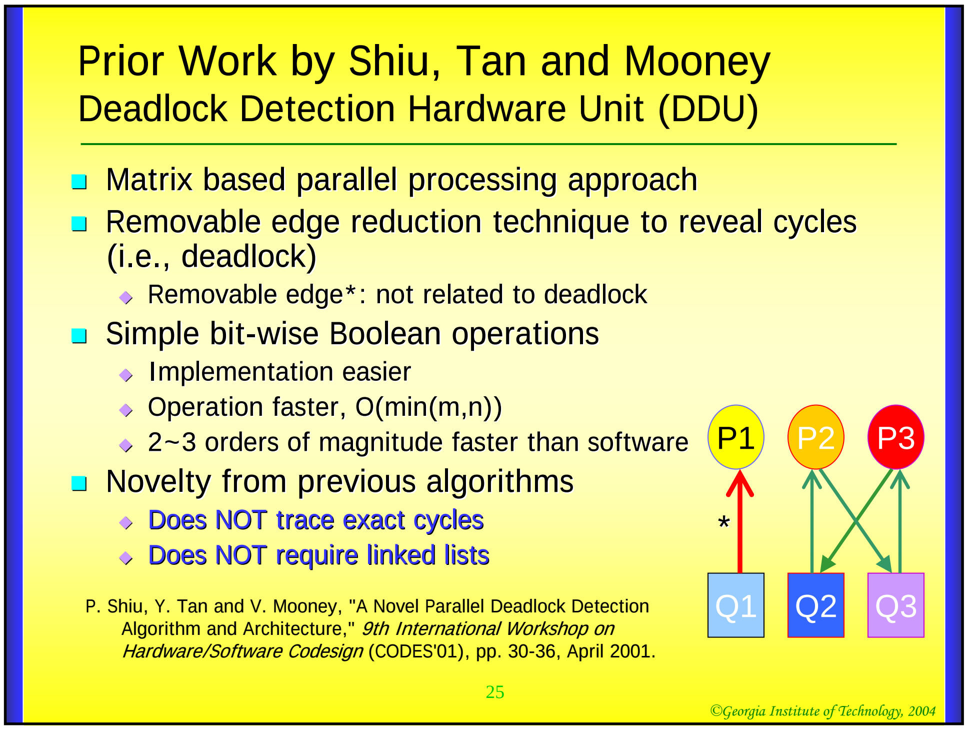
Outline

- Motivation
- Terms
- Previous Work
- Methodology
- Implementation
- Experimental Results
- Conclusion

Architecture of the DAU

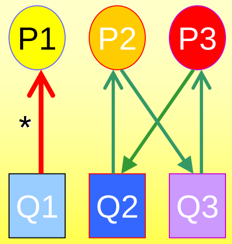


DDU: Deadlock Detection Unit
FSM: finite state machine



Prior Work by Shiu, Tan and Mooney
Deadlock Detection Hardware Unit (DDU)

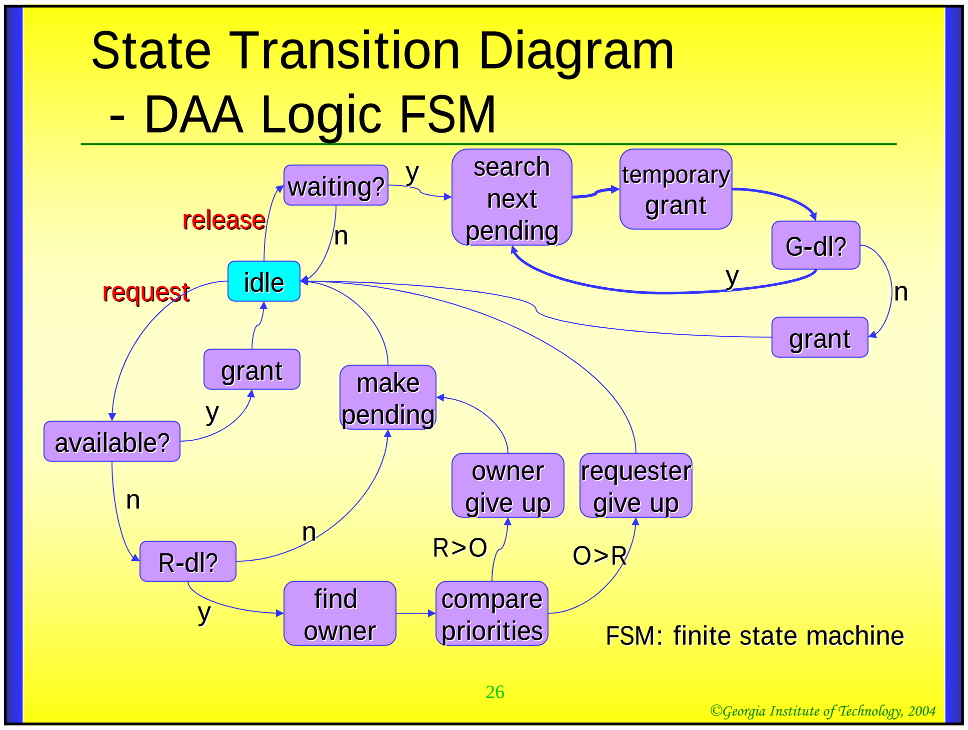
- Matrix based parallel processing approach
- Removable edge reduction technique to reveal cycles (i.e., deadlock)
  - Removable edge*: not related to deadlock
- Simple bit-wise Boolean operations
  - Implementation easier
  - Operation faster, O(min(m,n))
  - 2~3 orders of magnitude faster than software
- Novelty from previous algorithms
  - Does NOT trace exact cycles
  - Does NOT require linked lists



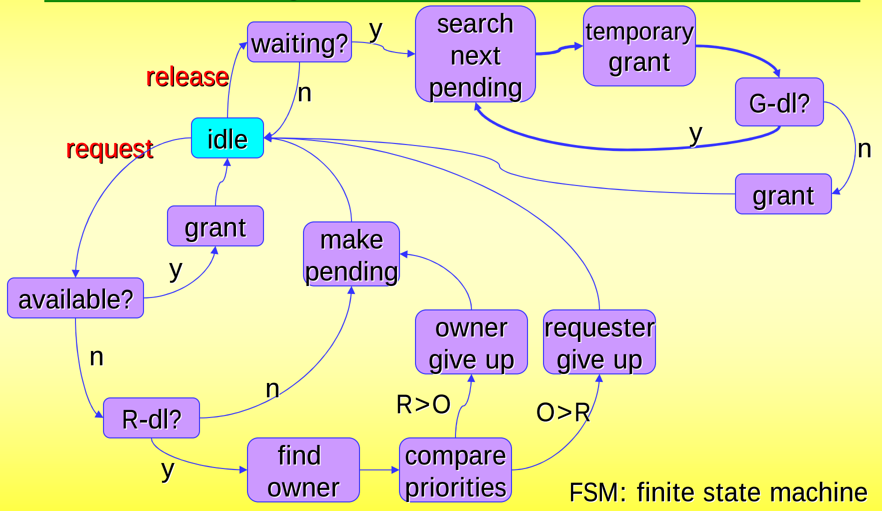
P. Shiu, Y. Tan and V. Mooney, "A Novel Parallel Deadlock Detection Algorithm and Architecture," *9th International Workshop on Hardware/Software Codesign* (CODES'01), pp. 30-36, April 2001.

25

©Georgia Institute of Technology, 2004



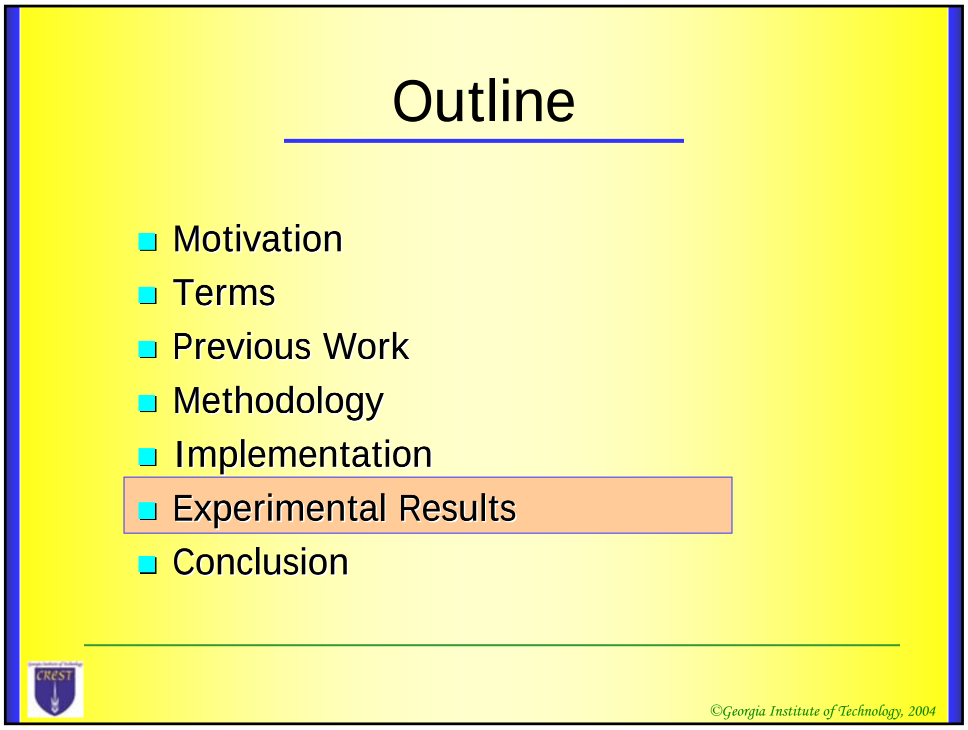
State Transition Diagram - DAA Logic FSM



FSM: finite state machine

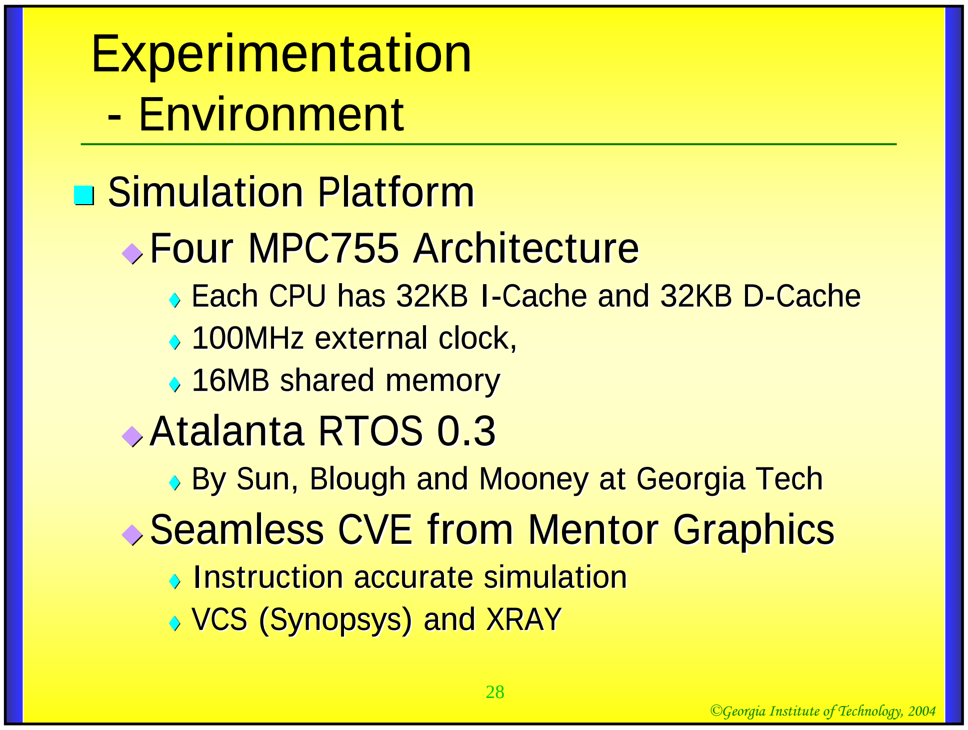
26

©Georgia Institute of Technology, 2004



Outline

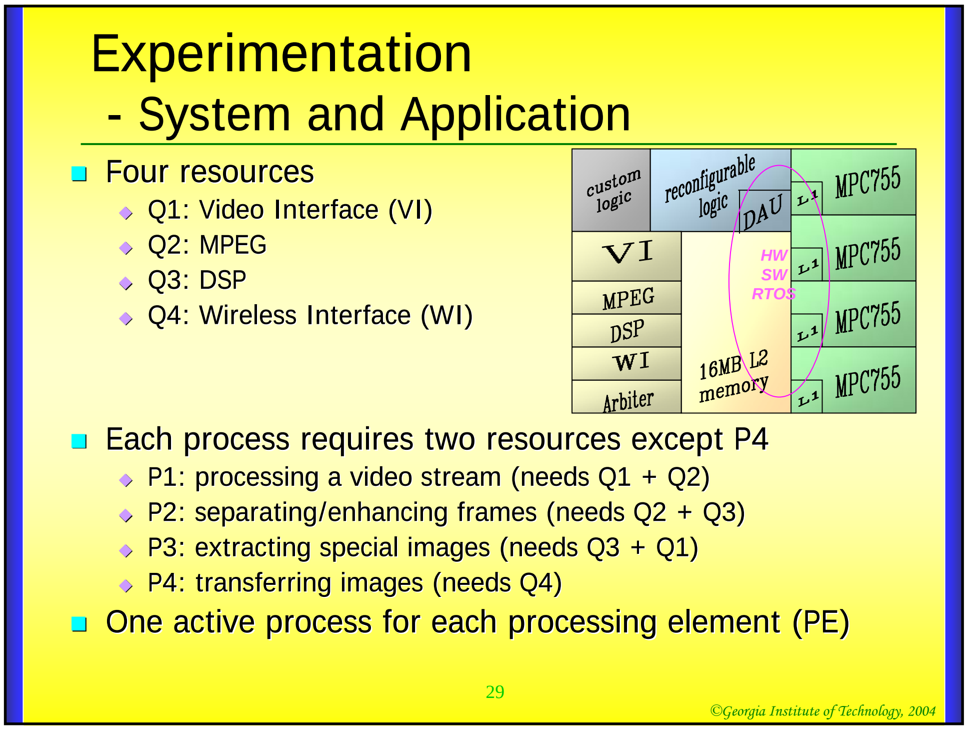
- Motivation
- Terms
- Previous Work
- Methodology
- Implementation
- Experimental Results
- Conclusion



Experimentation
- Environment

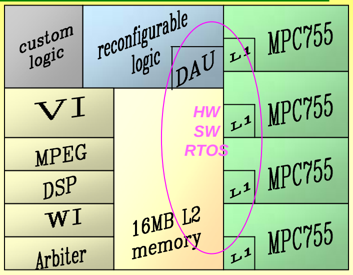
- Simulation Platform
  - Four MPC755 Architecture
    - Each CPU has 32KB I-Cache and 32KB D-Cache
    - 100MHz external clock,
    - 16MB shared memory
  - Atalanta RTOS 0.3
    - By Sun, Blough and Mooney at Georgia Tech
  - Seamless CVE from Mentor Graphics
    - Instruction accurate simulation
    - VCS (Synopsys) and XRAY

28



Experimentation - System and Application

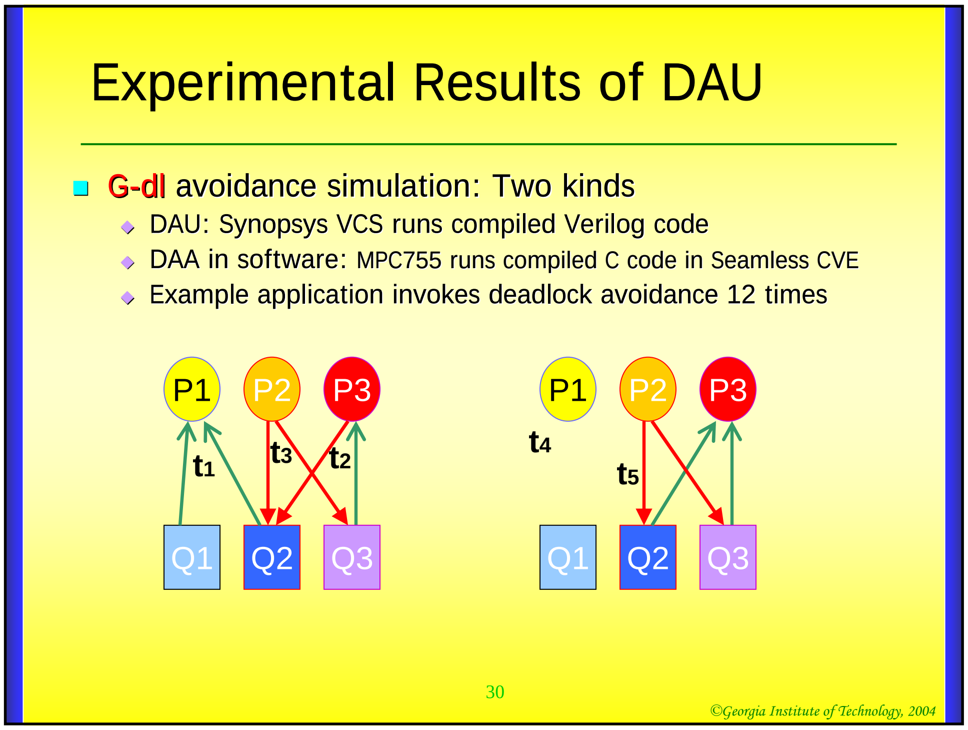
- Four resources
  - Q1: Video Interface (VI)
  - Q2: MPEG
  - Q3: DSP
  - Q4: Wireless Interface (WI)



- Each process requires two resources except P4
  - P1: processing a video stream (needs Q1 + Q2)
  - P2: separating/enhancing frames (needs Q2 + Q3)
  - P3: extracting special images (needs Q3 + Q1)
  - P4: transferring images (needs Q4)
- One active process for each processing element (PE)

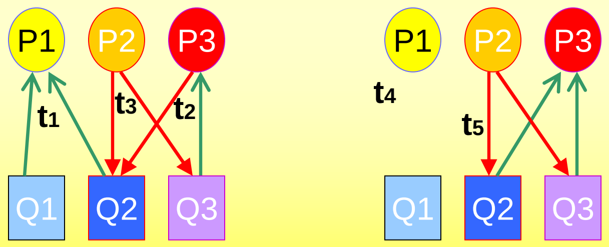
29

©Georgia Institute of Technology, 2004



Experimental Results of DAU

- G-dl avoidance simulation: Two kinds
  - DAU: Synopsys VCS runs compiled Verilog code
  - DAA in software: MPC755 runs compiled C code in Seamless CVE
  - Example application invokes deadlock avoidance 12 times



30

©Georgia Institute of Technology, 2004

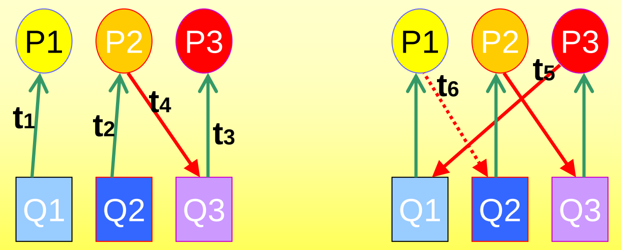
Experimental Results of DAU (cont'd)

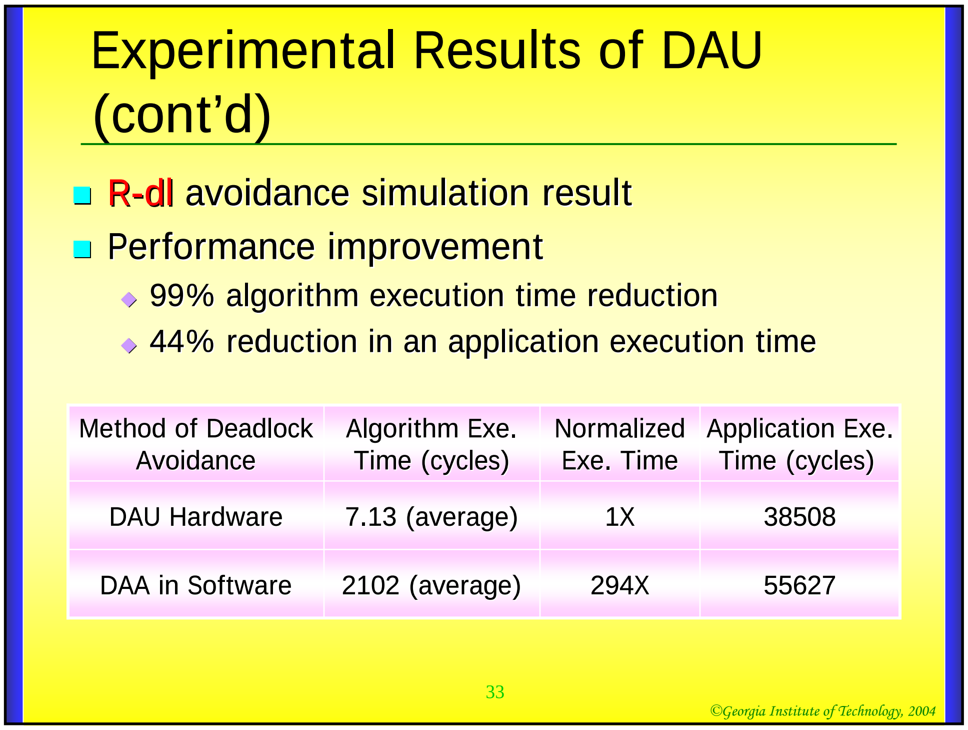
- G-dl avoidance simulation result
- Performance improvement
  - 99% algorithm execution time reduction
  - 37% reduction in an application execution time

Method of Deadlock Avoidance	Algorithm Exe. Time (cycles)	Normalized Exe. Time	Application Exe. Time (cycles)
DAU Hardware	7 (average)	1X	34791
DAA in Software	2188 (average)	312X	47704

Experimental Results of DAU (cont'd)

- R-dl avoidance simulation: Two kinds
  - DAU: Synopsys VCS runs compiled Verilog code
  - DAA in software: MPC755 runs compiled C code in Seamless CVE
  - Example application invokes deadlock avoidance 14 times

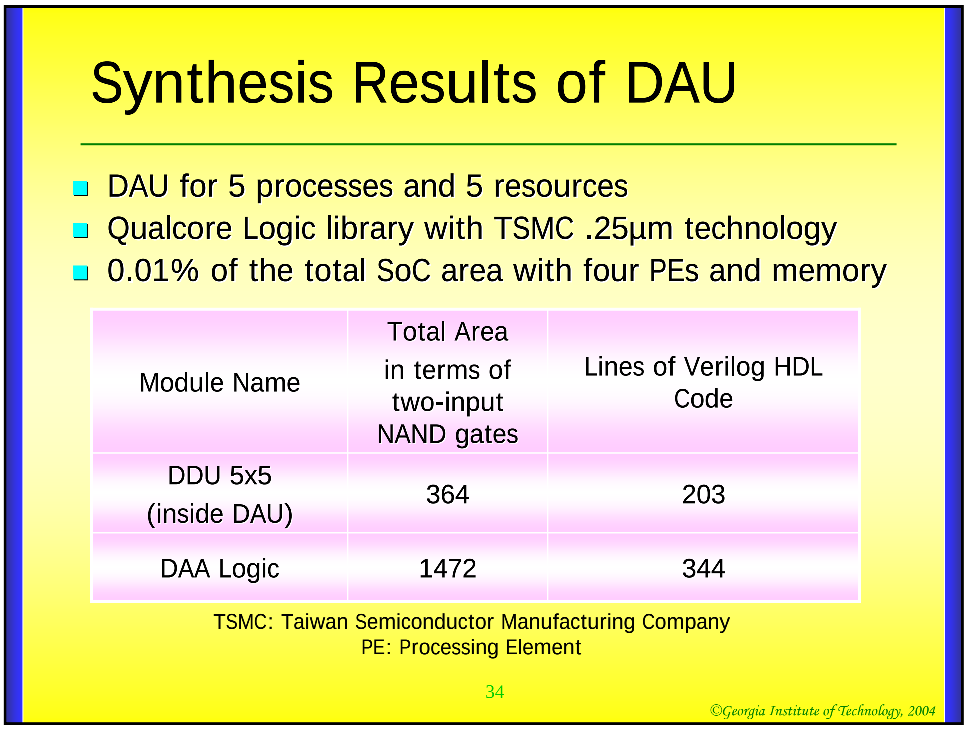




Experimental Results of DAU (cont'd)

- R-dl avoidance simulation result
- Performance improvement
  - 99% algorithm execution time reduction
  - 44% reduction in an application execution time

Method of Deadlock Avoidance	Algorithm Exe. Time (cycles)	Normalized Exe. Time	Application Exe. Time (cycles)
DAU Hardware	7.13 (average)	1X	38508
DAA in Software	2102 (average)	294X	55627

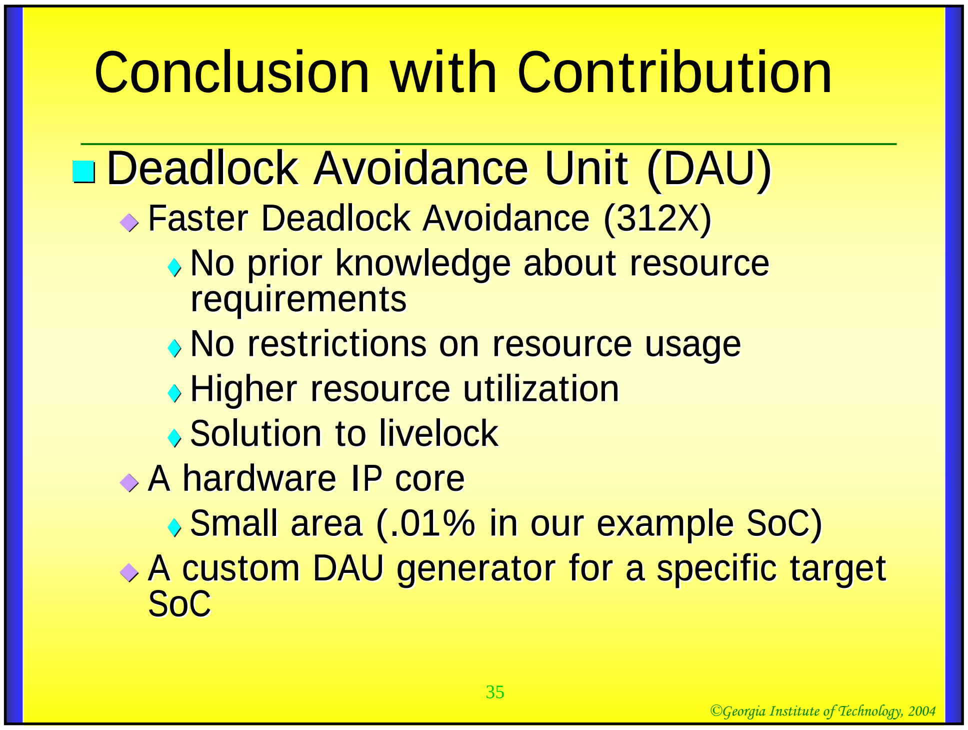


Synthesis Results of DAU

- DAU for 5 processes and 5 resources
- Qualcore Logic library with TSMC .25µm technology
- 0.01% of the total SoC area with four PEs and memory

Module Name	Total Area in terms of two-input NAND gates	Lines of Verilog HDL Code
DDU 5x5 (inside DAU)	364	203
DAA Logic	1472	344

TSMC: Taiwan Semiconductor Manufacturing Company
PE: Processing Element



Conclusion with Contribution

- Deadlock Avoidance Unit (DAU)
  - Faster Deadlock Avoidance (312X)
    - No prior knowledge about resource requirements
    - No restrictions on resource usage
    - Higher resource utilization
    - Solution to livelock
  - A hardware IP core
    - Small area (.01% in our example SoC)
  - A custom DAU generator for a specific target SoC