Path-Based Edge Activation for Dynamic Run-Time Scheduling

Vincent J. Mooney III

Assistant Professor
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA USA

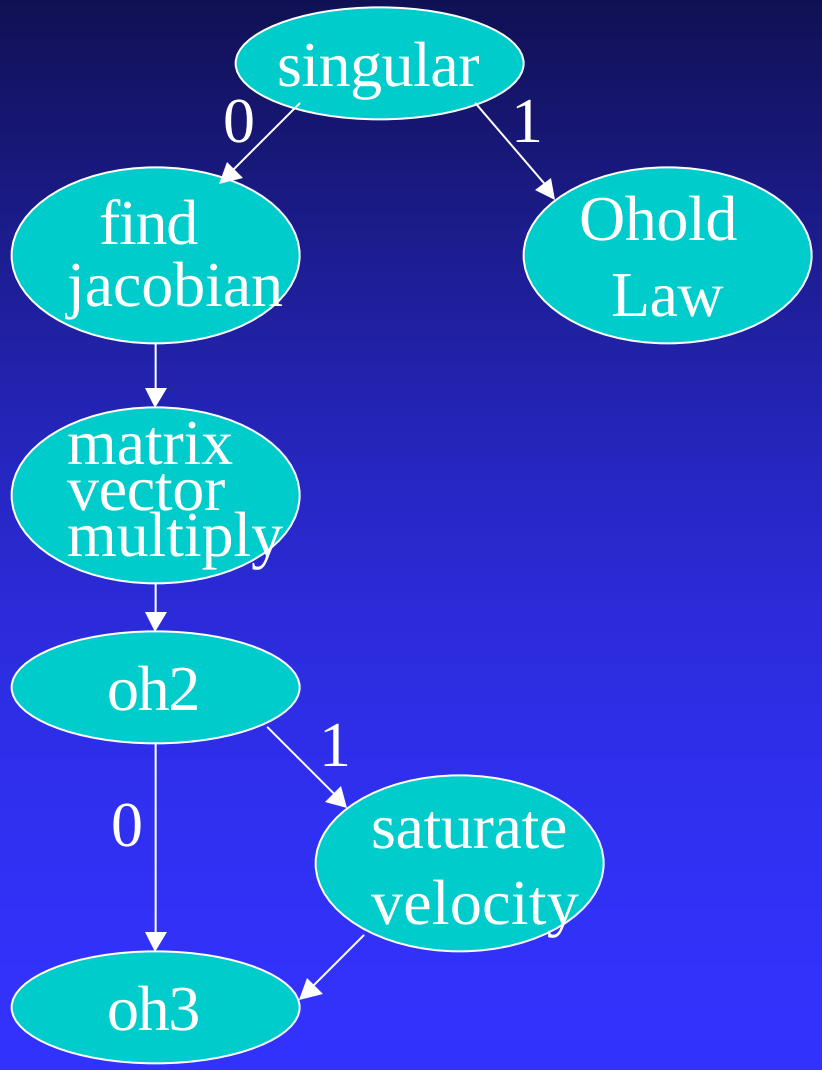
Outline

- Motivation
- Previous Work
- Path-Based Edge Activation
- Example
- Synthesis Flow
- Experimental Results
- Future Work

Motivation

- Dynamic Hard-Real-Time Systems
- Previous work by author limited to DAGs
- Application examples have control flow
- Extend run-time system to handle CDFG

Robotics Example: Concurrent Control Laws



Previous Work

- "Scheduling of Conditional Process Graphs for the Synthesis of Embedded Systems," Eles, et. al., DATE, 1998.
- "Hardware/Software Co-Design of Run-Time Systems," Ph.D. thesis, Stanford, 1998.
- "Hardware/Software Co-Design of Run-Time Schedulers for Real-Time Systems," to appear in Design Automation of Embedded Systems.

Conditional Process Graphs: Figure 4 (page 136), Processor pe_2

a) Optimal schedule of the path corresponding to D^C^K

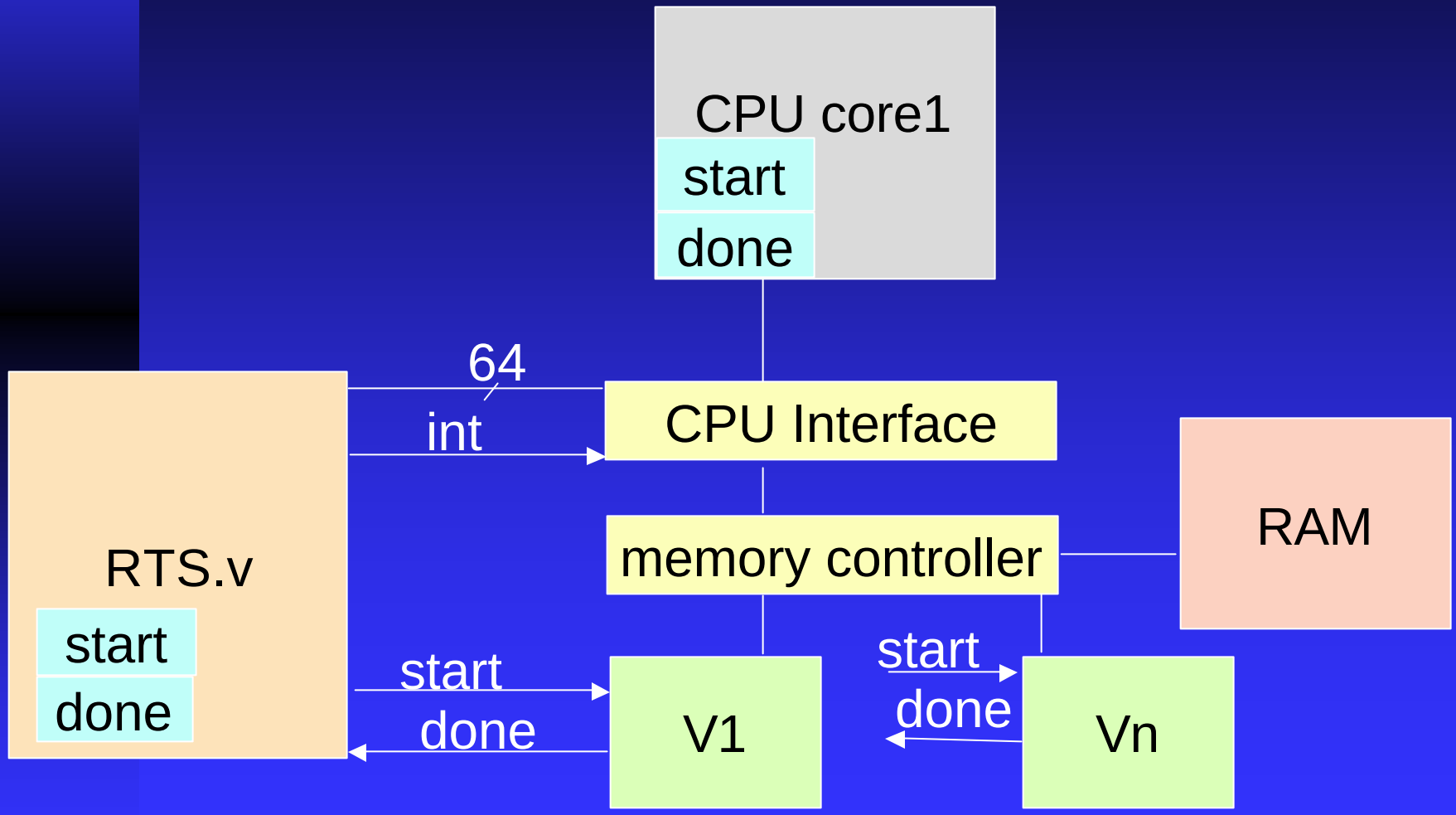
b) Optimal schedule of the path corresponding to D^C^*K'*

c) Adjusted schedule of the path corresponding to D^C^K

Conditional Process Graphs

- Conditionals (e.g., D, C, K) are broadcast to all processing elements
- Activation times (start times) for tasks fixed based on values of conditionals (or subset of conditionals)
- Focus on handling late arriving conditionals
- In case where all conditionals are ready at the beginning, *schedule merging* may result in known suboptimal solution

Previous Work (author)



Task Control

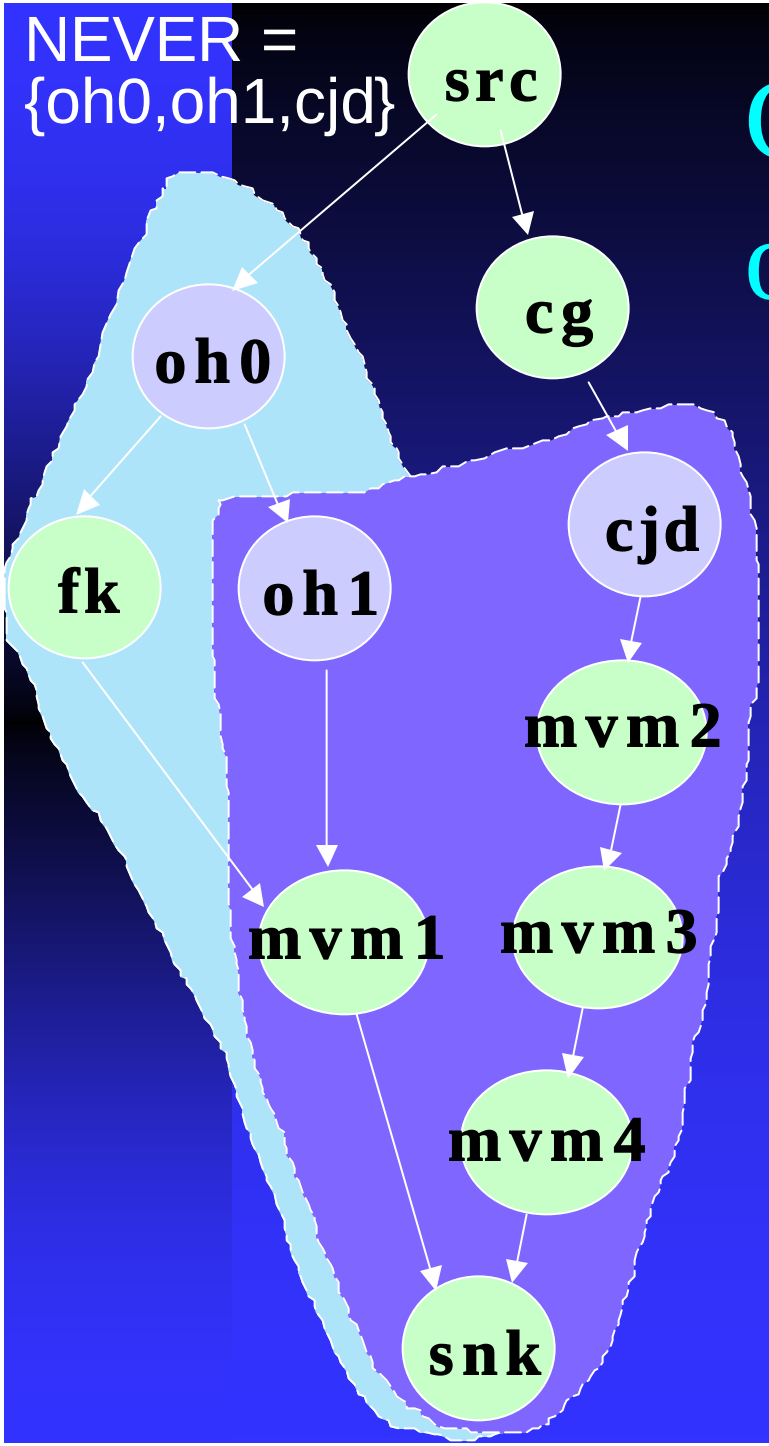
- Associate *start* and *done* event with each task
- Control of hardware tasks
  - *start* signal (bit)
  - *done* signal (bit)
- Control of software tasks
  - *start* vector encapsulates all sw *start* events
  - *done* vector encapsulates all sw *done* events

Run Time Scheduler Implementation

- Start with control flow of hw- and sw-tasks
- Hardware implementation:
  - put FSM corresponding to the control flow
    - cycle based semantics
    - can predictably satisfy hard real-time constraints
- Software implementation:
  - preemptive static priority scheduler
    - can execute different threads
    - keeps track of which threads are suspended
  - direct execution of software tasks by ISR
    - all sw tasks run to completion (no suspension)
- Mixed implementation can leverage advantage of hardware and software

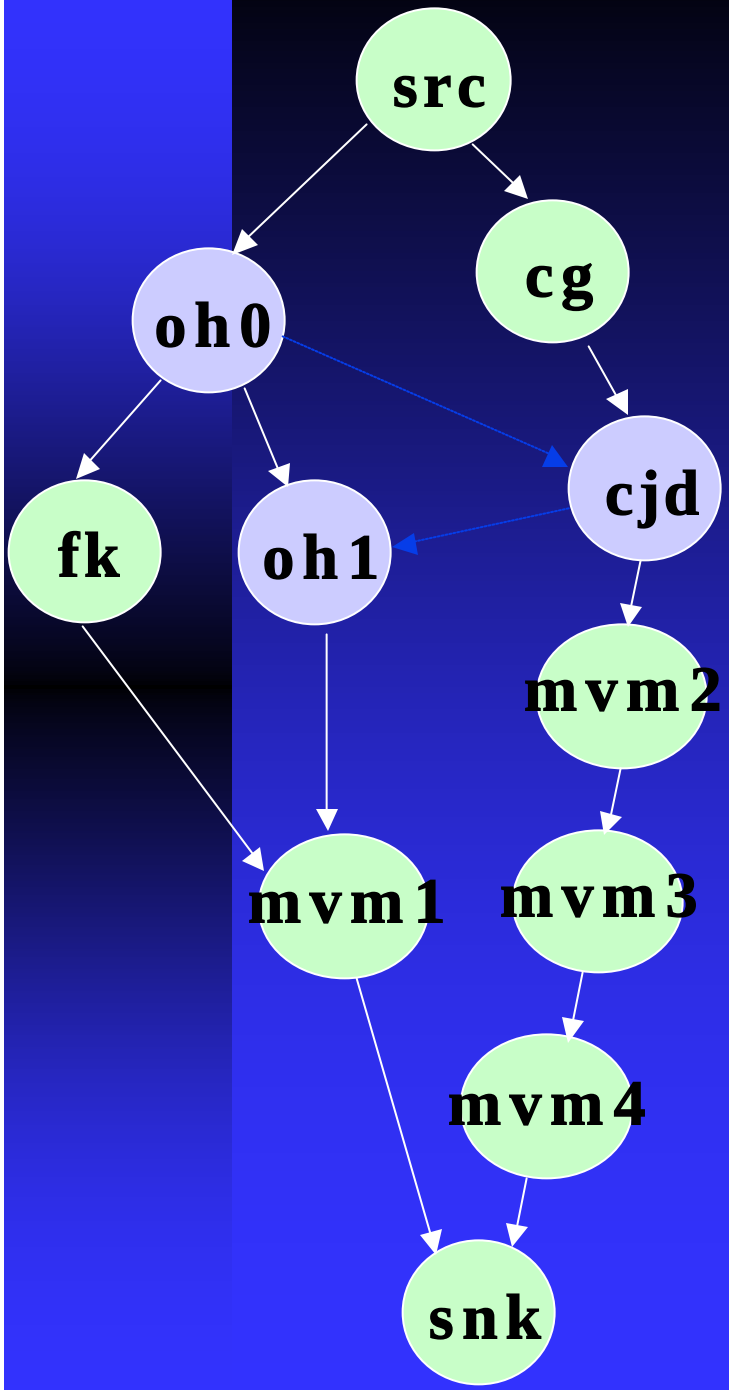
Constructive Heuristic on DAG

NEVER = {oh0,oh1,cjd}



t_3 \ x_{3k}	$f_3(t_3,x_{3k})$		$f_3^*(t_3,x_{3k})$	x_{3k}^*
	oh1,snk	cjd,snk		
oh0	24,020	∞	24,020	oh1,snk
oh1	∞	43,812	43,812	cjd,snk
cjd	35,012	∞	35,012	oh1,snk

X_2^* = {(oh0,oh1,snk),(oh1,cjd,snk), (cjd,oh1,snk)}



Constructive Heuristic Scheduling Algorithm: Result

Final Result:

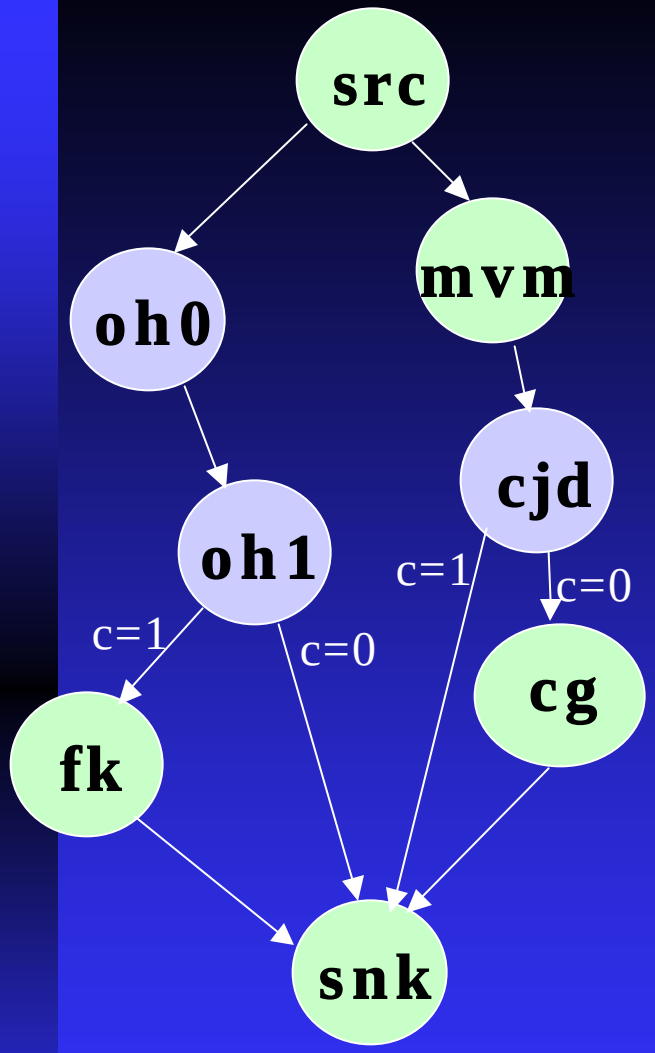
---

oh0 -- priority 1
cjd -- priority 2
oh1 -- priority 3

WCET: 39,012

Path-Based Edge Activation

- Extend scheduling to handle CDFG, not just DAG
- Conditional edges
  - active only if a particular path chosen
  - a path is defined by a set of values of conditional choices in the CDFG
- For each path, insert conditional edges to minimize WCET
  - assumption: conditional values evaluated early enough for all conditional edge insertions



Example

task	hw/sw	wcet(cycles)
cg	hw	11,000
oh0	sw	2,554
oh1	sw	20,581
fk	hw	11,500
cjd	sw	14,878
mvm	hw	4,400

NEVER = {oh0, oh1, cjd}

No static order can achieve better than a WCET of 49,013

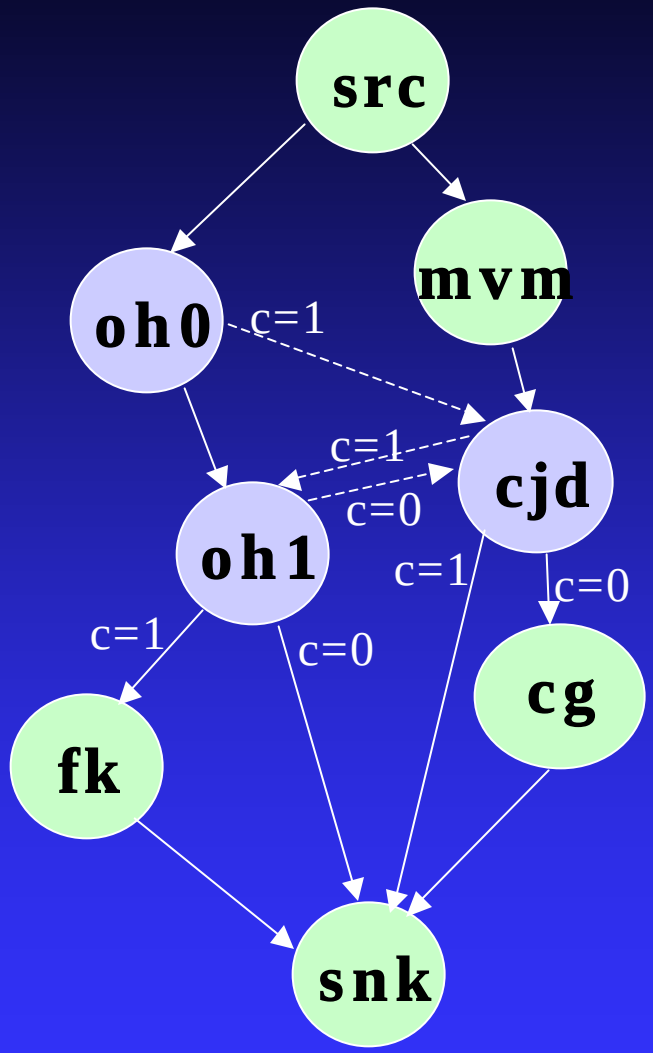
Centralized Control

- Done signals arrive to hardware run-time scheduler (no broadcast)
- Dynamic ordering of software tasks must be done by hardware run-time scheduler
- Use *hardware-driven software execution*
- ISR executes a software task
  - adv.: fast
  - disadv.: software tasks not interruptable

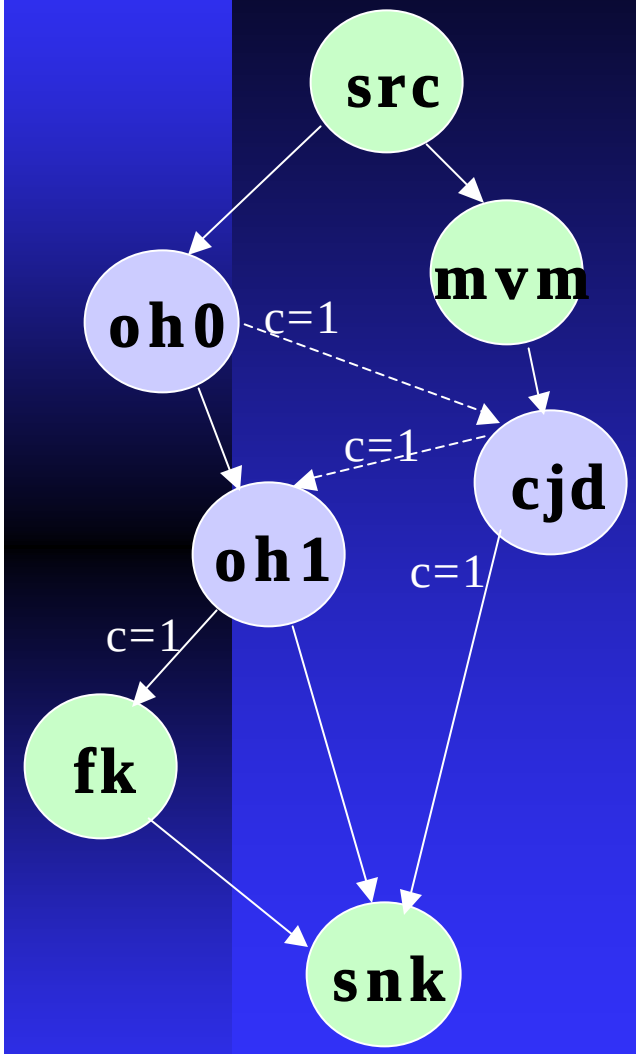
Scheduling Assumptions

- A CDFG represents the set of tasks
  - limited number of paths
- One rate constraint for the graph
- A NEVER set specifies mutually exclusive sw-tasks
- Each sw-task, once started, runs to completion
  - limits solution space
- Hw-sw communication accounted for
  - in task WCET
  - as a separate task
- Interrupts come only from the hw run-time sched.

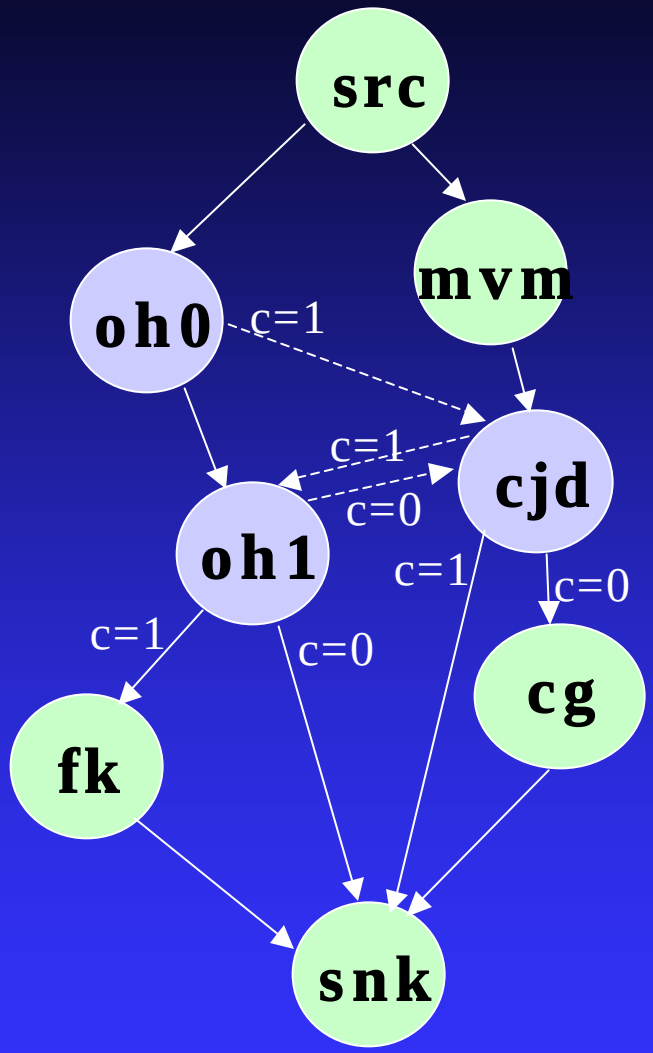
CDFG



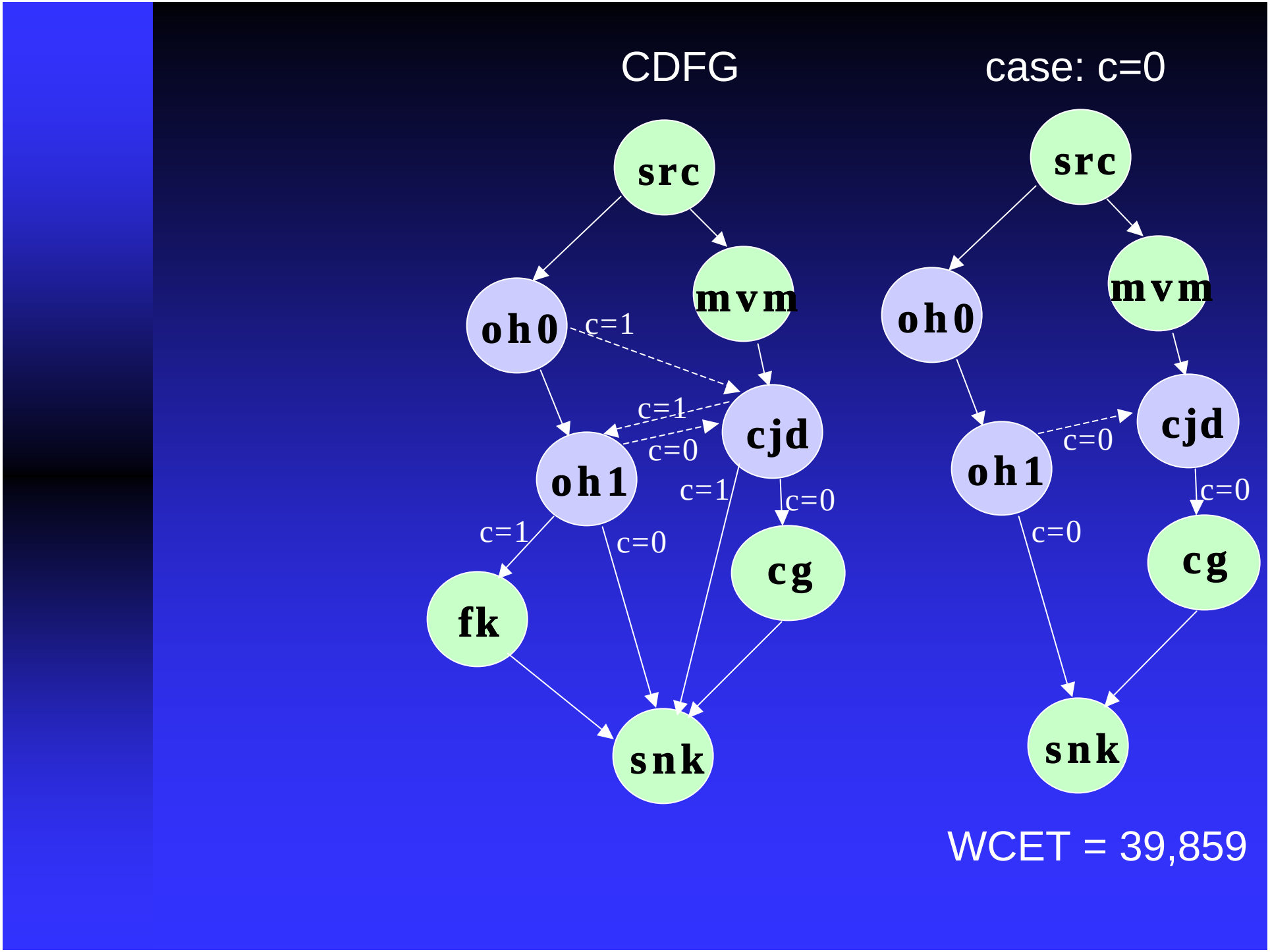
case: c=1

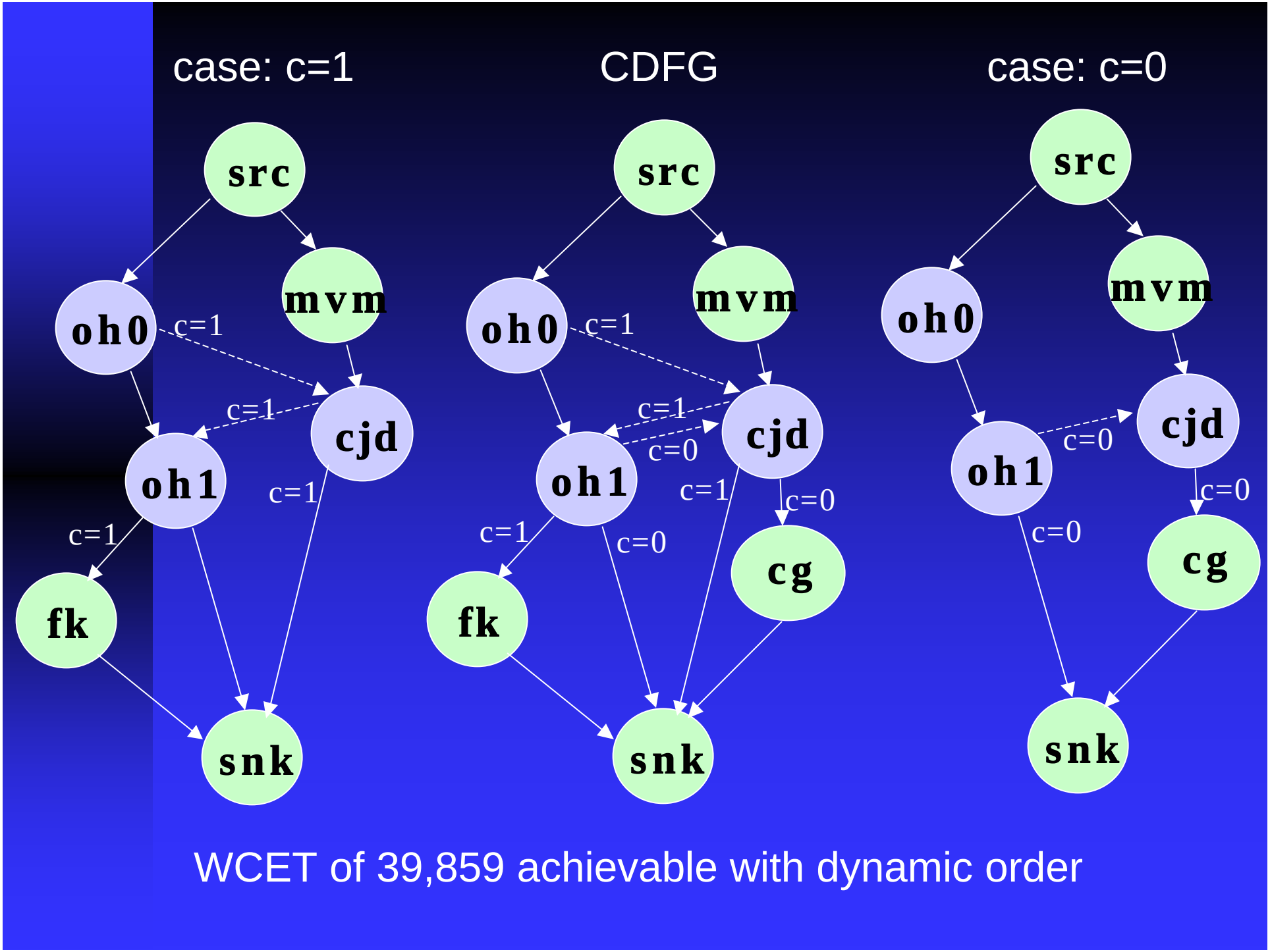


CDFG



WCET = 38,013


CDFG
case: c=0
src
mvm
oh0
c=1
cjd
c=1
c=0
oh1
c=1
c=0
c=1
c=0
fk
cg
snk
src
mvm
oh0
cjd
c=0
oh1
c=0
c=0
cg
snk
WCET = 39,859

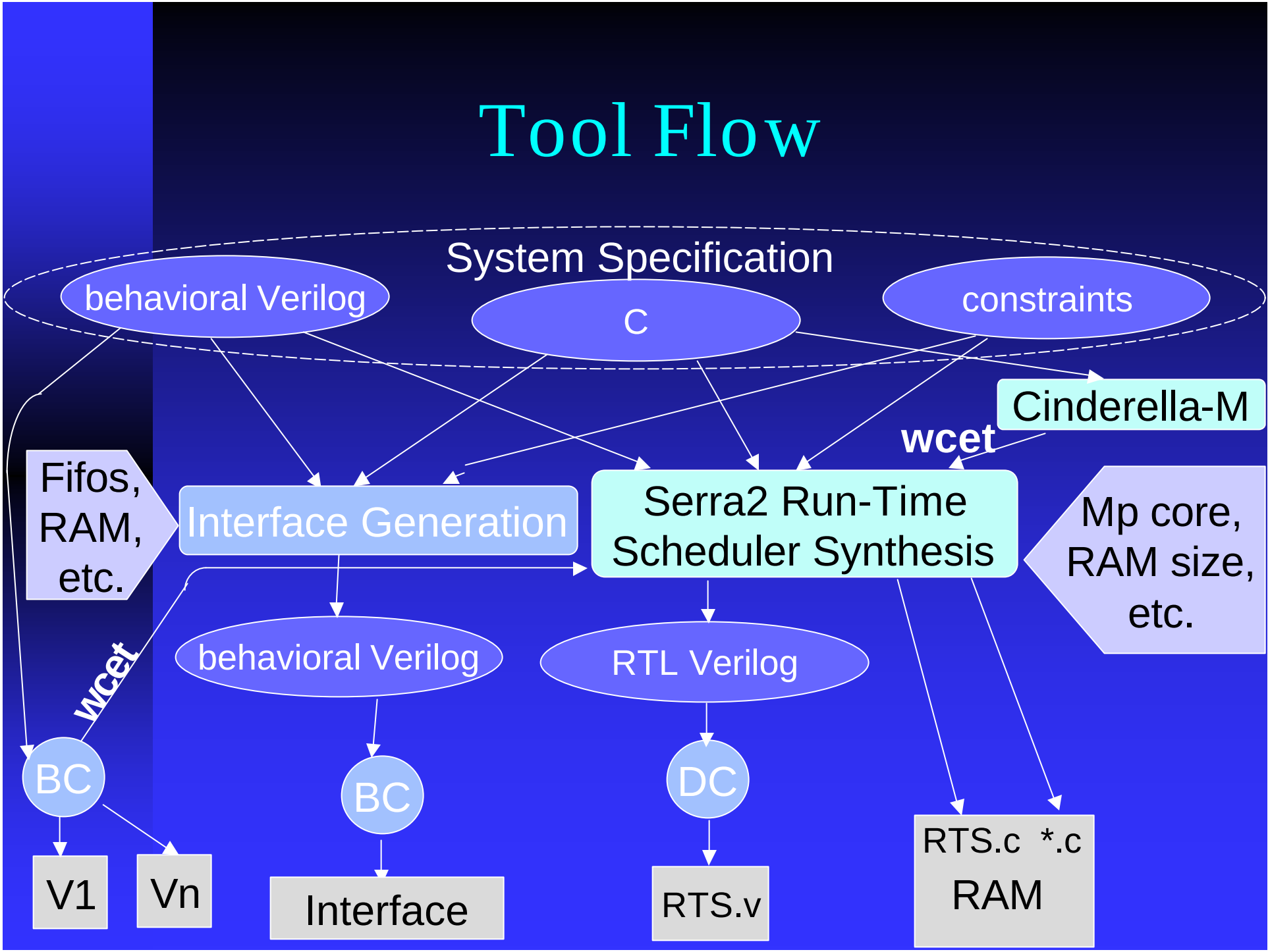


WCET of 39,859 achievable with dynamic order

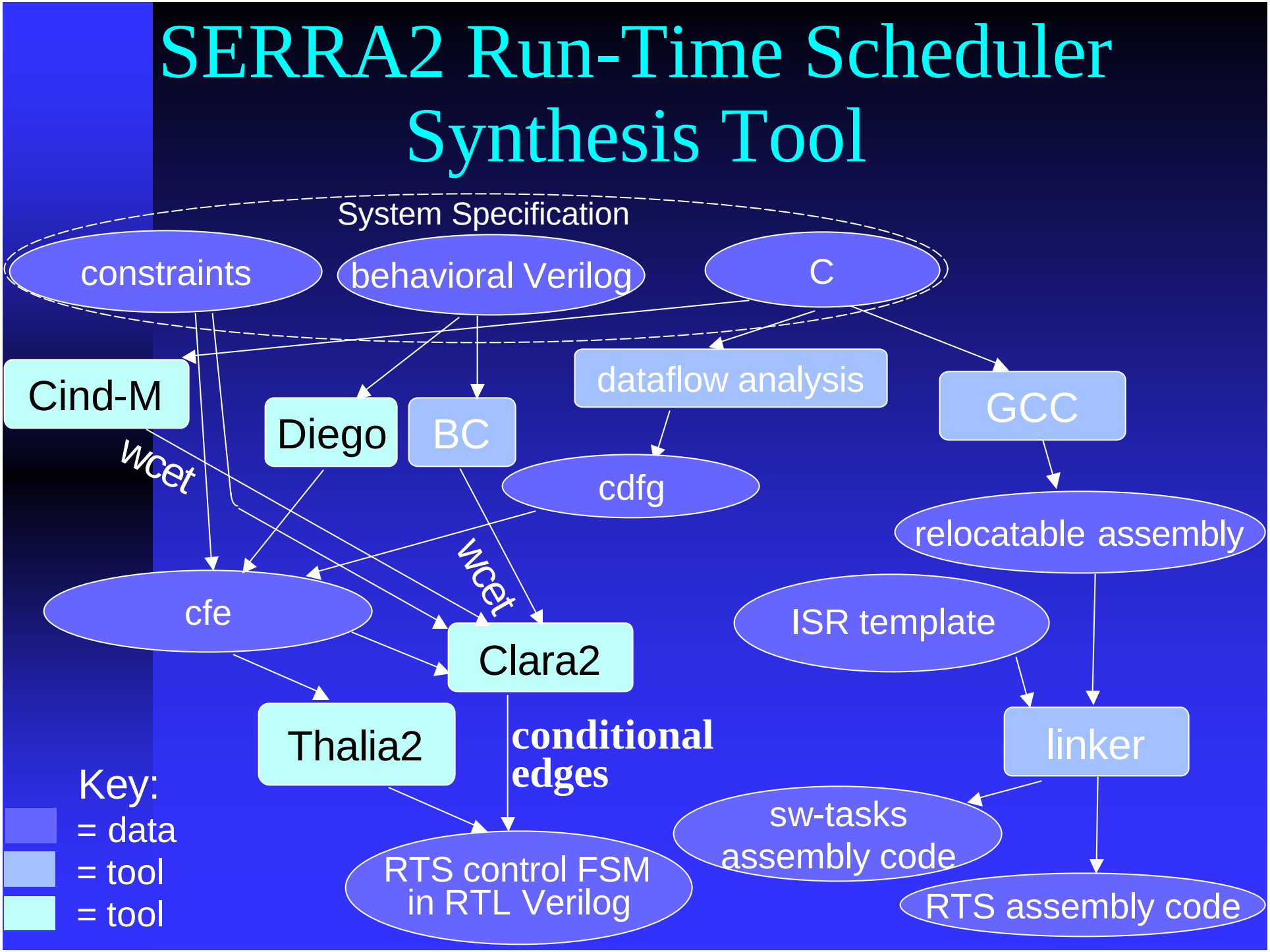
Algorithm

Solve_order(CDFG,NEVER)
beginmodule
  foreach *path* determined by a unique set of conditional values
  begin
    DAG = subset of CDFG determined by *path*
    Schedule DAG using *constructive heuristic scheduling*
    Add conditional edges to enforce DAG schedule
  end
endmodule

Tool Flow


System Specification
behavioral Verilog
C
constraints
Cinderella-M
wcet
Fifos, RAM, etc.
Interface Generation
Serra2 Run-Time Scheduler Synthesis
Mp core, RAM size, etc.
wcet
behavioral Verilog
RTL Verilog
BC
BC
DC
V1
Vn
Interface
RTS.v
RTS.c *.c
RAM

SERRA2 Run-Time Scheduler Synthesis Tool



Example and Experimental Results

Software Task	# Lines of C	# Lines Asmbly	WCET
cjd	286	1177	14878
oh0	90	237	2554
oh1	693	3263	20581
int-ser-rtn	N/A	26	20

Hw-task	# Lines V	Area	WCET
mvm	629	33645	4400
fk	2362	42168	11500
cg	2897	59587	11000
rtsched-hw	484	413	99701

- Hw-tasks written in Verilog for BC, use LSI 10K library
- Verilog model of MIPS core with interrupts
- 19% decrease in WCET: 39859 (49013)
- Used VCS™ to verify result

Future Work

- Extend to handle late arriving conditionals
- Extend to allow interruptable software tasks