Process Scheduling for Performance Estimation and Synthesis of Hardware/Software Systems

Petru Eles[1], Krzysztof Kuchcinski[1], Zebo Peng[1],
Alexa Doboli[2] and Paul Pop[1]

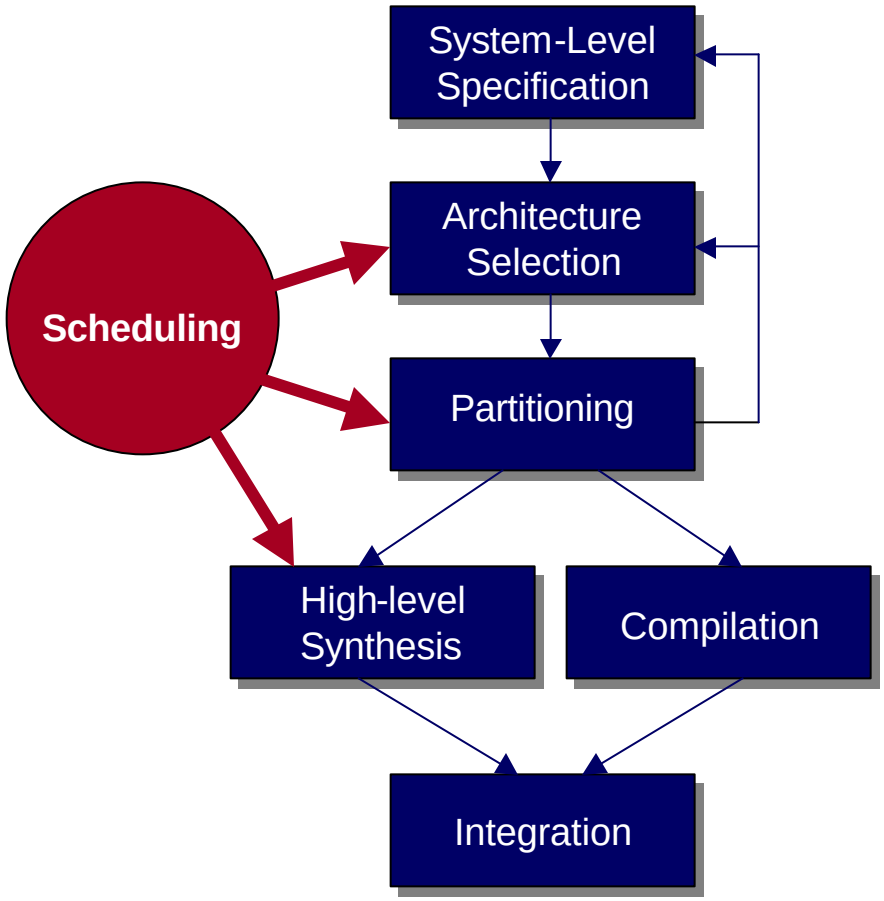
[1]Dept. Of Computer and Information Science
Linkoping University
Sweden

[2]Computer Science and Engineering Department
Technical University Timisoara
Romania

Outline

- Motivation
- Problem Formulation
- The Conditional Process Graph
- The Scheduling Strategy
- The Schedule Table
- Generation of The Schedule Table
- Experimental Results
- Conclusions

Motivation



Heterogeneous architecture

- programmable processors
- hardware components
- shared buses
- local and shared memories

Process interaction captures

- data flow
- flow of control

Problem Formulation

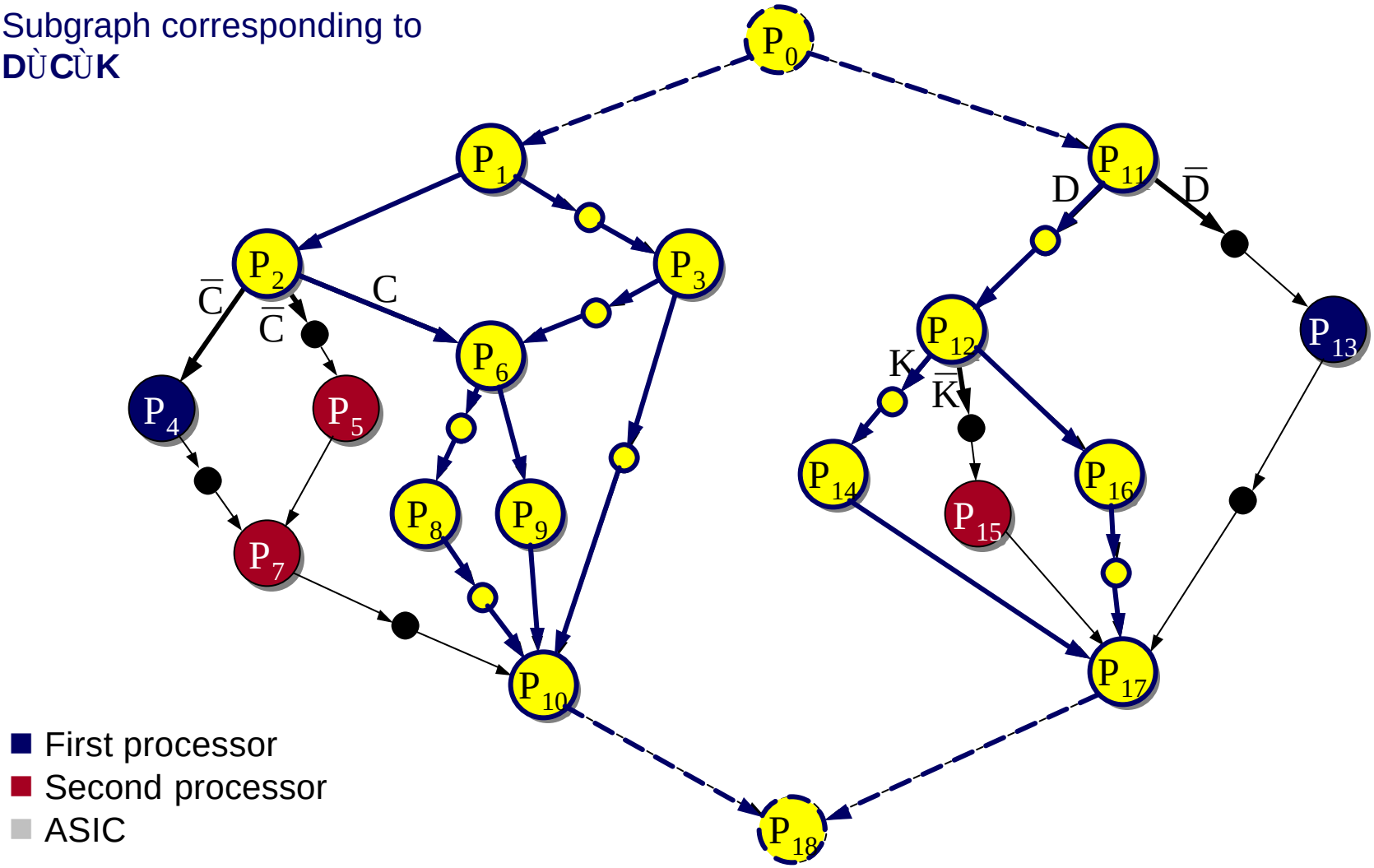
- Generic architecture:
  programmable processors, hardware components (ASICs),
  shared buses, local and shared memories
- Each process is assigned to a (programmable or hardware) processor.
- Each communication channel which connects processes assigned to different processors is assigned to a bus.
- Each process or communication task is characterized by a certain execution time.

Goals

- To derive a worst case delay by which the system completes execution, so that this delay is minimized.
- To generate the schedule which guarantees this delay.

The Conditional Process Graph

Subgraph corresponding to
DÙCÙK



The Scheduling Strategy

- The values of conditions are unpredictable.
- At a certain moment during execution, when the values of some conditions are known, they have to be used in order to take the best possible scheduling decisions.
- For each individual path there is an optimal schedule of the processes, which produces a minimal delay.

The Scheduling Strategy

1. Scheduling of each individual alternative path.
2. Merging of the individual schedules and generation of the schedule table.

The Schedule Table

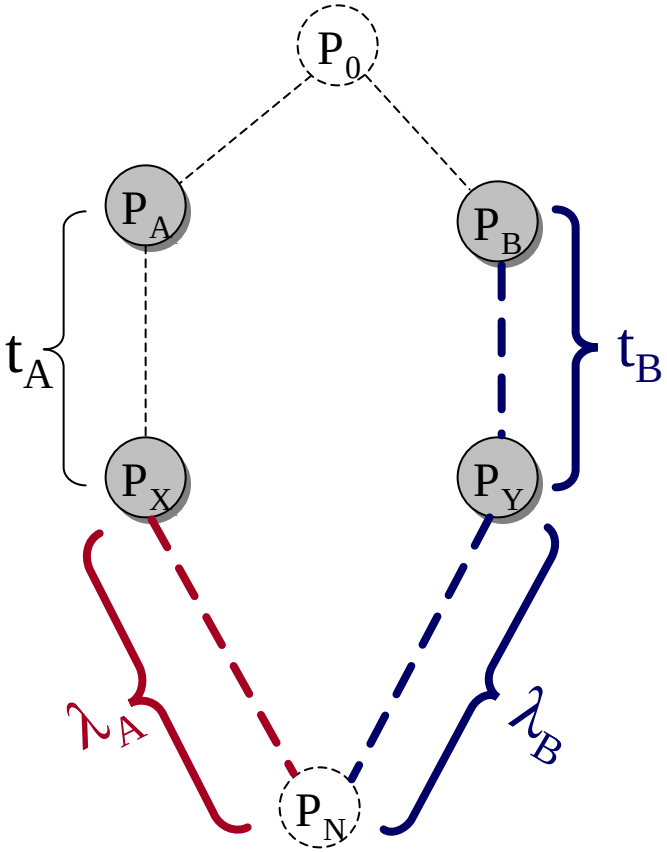
	true	D	D∧C	D∧C∧K	D∧C∧K	D∧C	D∧C∧K	D∧C∧K	D	D∧C	D∧C
P_1	0										
P_2	3										
P_{10}				34	34		26	26		34	26
P_{11}	0										
P_{14}					35			24			
P_{17}				29	37		30	26		22	24
P_{18} (1→3)	3										
P_{19} (2→5)						9					10
P_{20} (3→10)				28	20		21	21		22	18
D	6										
C		7							7		
K			15			15					

Scheduling of The Alternative Paths

- Derive a minimal static schedule for a directed, acyclic polar graph. Allocation and execution time of processes is given.

1. List scheduling based heuristic.
   - Critical Path,
   - Urgency Based and
   - *Partial Critical Path* priority functions.
2. Branch-and-Bound based algorithm.
   - Branching,
   - Selection and
   - Bounding rules.

Partial Critical Path Scheduling



Critical Path Scheduling

$$l_{PA} = t_A + l_A$$

$$l_{PB} = t_B + l_B$$

Partial Critical Path Scheduling

$$L_{PA} = \max(T\_curr + t_A + l_A,\ T\_curr + t_A + t_B + l_B)$$

$$L_{PB} = \max(T\_curr + t_B + l_B,\ T\_curr + t_B + t_A + l_A)$$

Select the alternative with the smaller delay:

$$L = \max(L_{PA}, L_{PB})$$

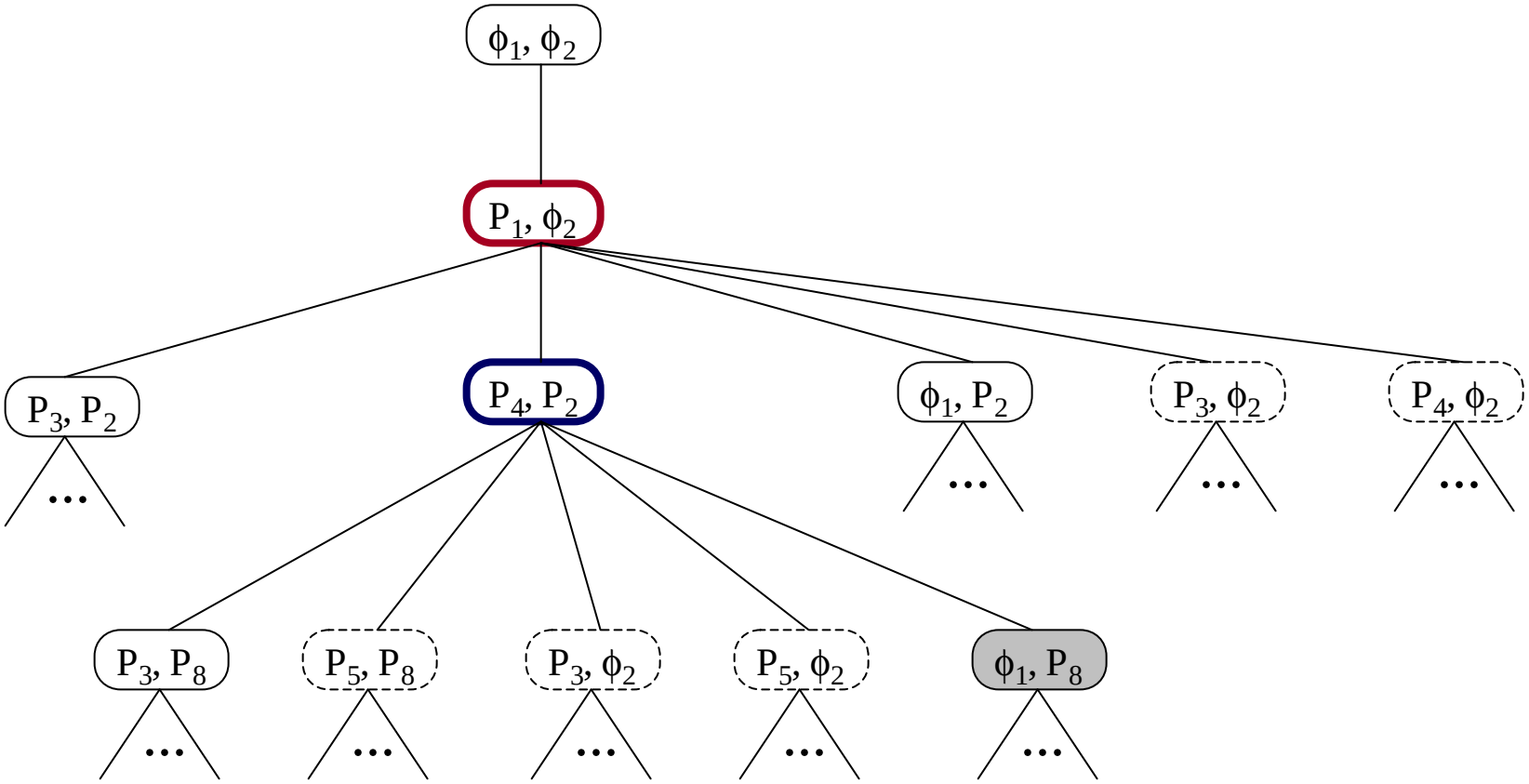
$$l_A > l_B \Rightarrow L_{PA} < L_{PB}$$
$$l_B > l_A \Rightarrow L_{PB} < L_{PA}$$

Use λ as a priority criterion.

Branch and Bound Scheduling

- Branching Rule
  Generates new states starting from a given parent state.
  Generates children as a result of a scheduling decision.
  It might let a processor idle*, even if there are ready processes on it.*
- Selection Rule
  From which state should we continue the branching operation?
  From the state which has the highest PCP priority.
- Bounding Rule
  Should exploration continue further?
  Three bounding levels:
  - *fast estimation of two weaker bounds,*
    if bounding with them doesn't succeed:
  - *third bound based on a partial traversal of the process graph*

Branch and Bound Scheduling (cont'd)



Graphs Used for Experiments

- Number of Graphs: 1250
  250 for each dimension of 20, 40, 75, 130, 200 nodes.
- Graphs Structure:
  Random and regular (trees, groups of chains).
- Architecture:
  1 ASIC, up to 10 processors and up to 3 buses.
- Mapping:
  Random and using simple heuristics.

Experimental Results

Average percentage deviation from the optimal schedule lengths.

- Urgency Based Priority: 4.73%
- Critical Path Priority: 4.69%
- Partial Critical Path Priority: 2.35%

- Averages are similar for the five graph dimensions.
- Deviations for individual graphs are in the range (0%, 47.74%).

Experimental Results (cont'd)

Percentage of final (optimal) results obtained with the BB algorithm.

time limit (s)	20 processes	40 processes	75 processes	130 processes	200 processes
0.04	91.6%	0.0%	0.0%	0.0%	0.0%
0.08	95.6%	54.0%	0.0%	0.0%	0.0%
0.3	98.8%	83.6%	66.4%	0.0%	0.0%
1	99.6%	87.6%	77.6%	56.8%	0.0%
3	**99.6%**	**89.6%**	**79.2%**	**70.8%**	**51.0%**
5	100%	90.4%	79.2%	72.0%	62.1%
60	100%	92.4%	80.8%	76.8%	71.5%
1800	100%	96.8%	84.8%	80.4%	79.5%

Experimental Results (cont'd)

Percentage deviation of PCP Schedule from intermediate results obtained with BB.

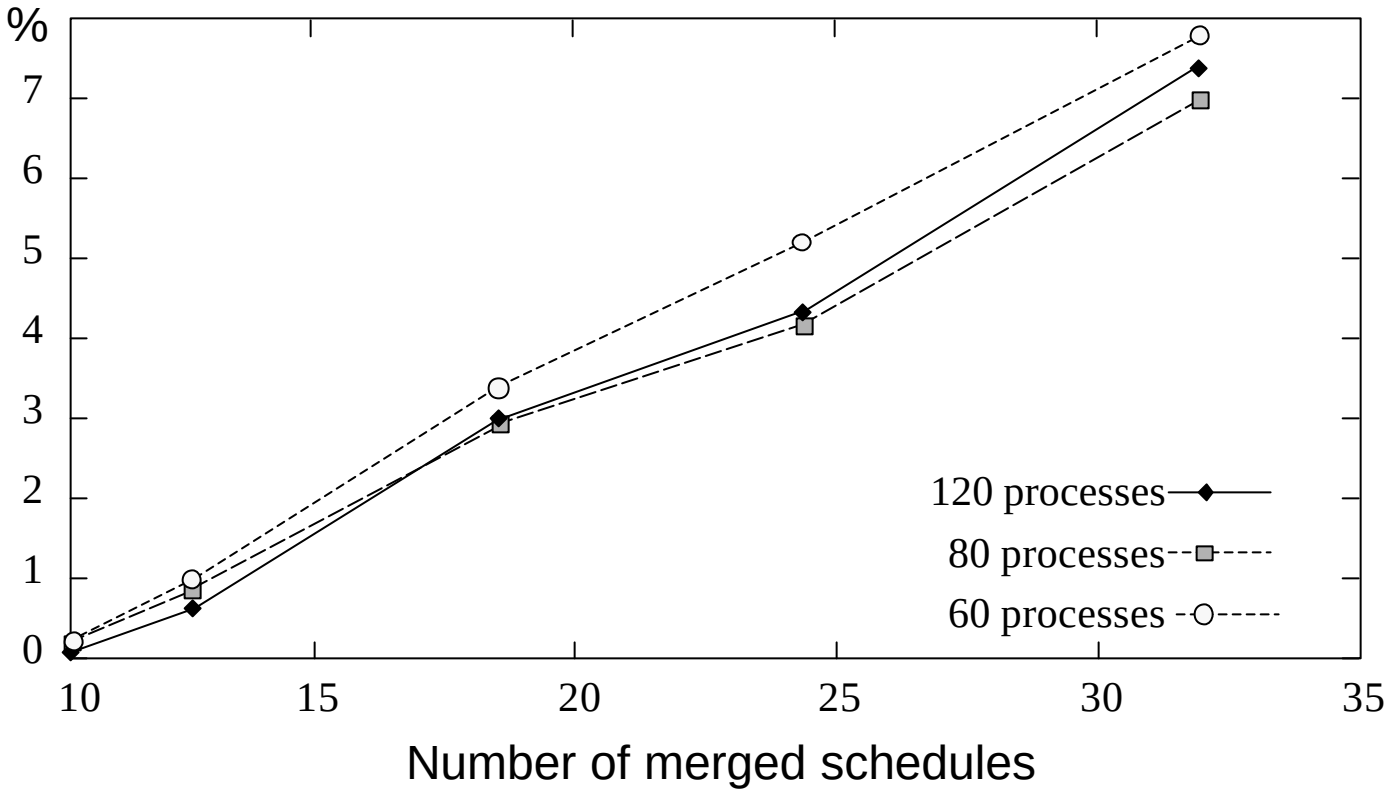
time	40 processes		75 processes		130 processes		200 processes	
	average	maximum	average	maximum	average	maximum	average	maximum
1	1.94%	17.65%	2.25%	16.11%	2.71%	10.31%	0%	0%
5	1.67%	5.50%	2.45%	16.11%	**2.76%**	**8.11%**	1.99%	21.10%
60	1.48%	4.92%	2.68%	19.01%	2.96%	10.73%	2.13%	10.58%
300	1.39%	4.26%	2.96%	19.01%	3.04%	13.73%	2.50%	12.75%

The Table Generation Algorithm

- Start times of processes are fixed in the table according, with priority, to the schedule of that path which produces the longest delay;
- The start time of a process is placed in a column headed by the conjunction of condition values known at that time on the respective processor;
- Conflicts have to be avoided at table generation.

Experimental Results (cont'd)

Percentage increase of the worst case delay relative to the delay of the longest path.



- Real-life example: F4 level of ATM protocol layer.

Conclusions

- An approach to process scheduling for the synthesis of embedded systems.
- Process level representation which captures both data flow and the flow of control.
- A schedule table is generated.
  The worst case delay is minimized.
- Evaluation based on experiments using a large number of graphs generated for experimental purpose as well as real-life examples.