

PAP: Power Aware Partitioning of Reconfigurable Systems

Rabi Mahapatra
Texas A&M University
College Station, TX 77843



Outline

- Introduction
- Related Work
- PAP: Power Aware Partitioning
- MPAP: PAP for multifunctional systems
- Experiments
- Summary



Introduction

- HW/SW Codesign: Key Issues
 - Partitioning
 - Synthesis
 - Co-simulation
- Partitioning problem : **Non-trivial**
 - Application - 100 tasks , 3 different HW/SW implementations
 $(2*3)^{100}$ possible partitioning solutions



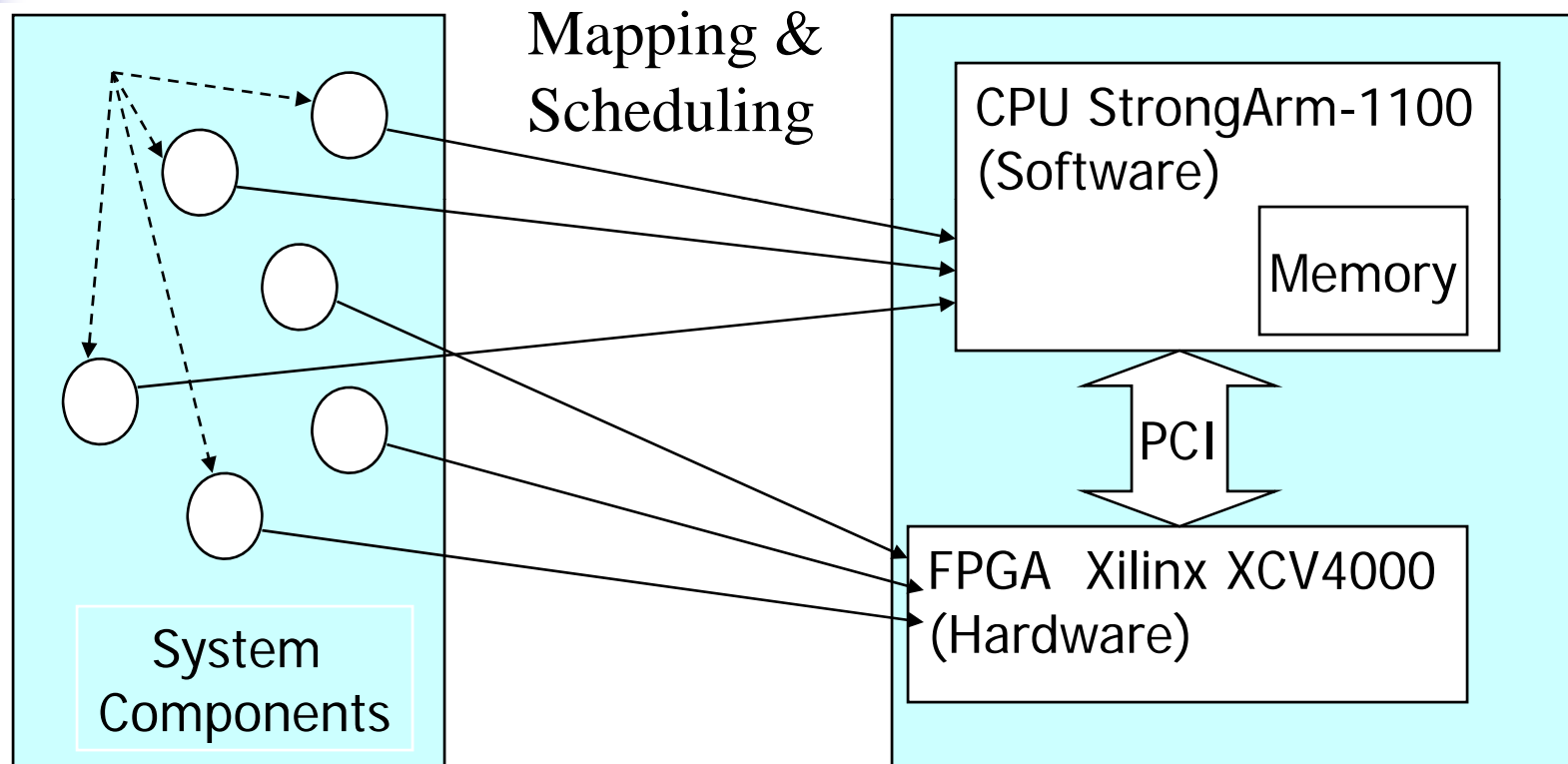
Objective

- Given (Inputs)
 - Application(s) descriptions (system level)
 - Target Architecture (CPU, FPGA, P_{\max} , Ah_{total})
 - Task's metrics (P_s , T_s , P_h , T_h , A_h)

Determine suitable partitioning framework that will map and schedule the application(s) on target architecture so as to meet

- The Deadline & Power Constraints

Partitioning



System
Description

System
Architecture



Related Work

- **Heuristic Based**
 - Asawaree Kalavade and P.A. Subramanyam 1998
“Global Criticality/Local Phase (GCLP) Heuristic”
 - System Power not considered
- **Iterative improvement techniques**
 - Huiqun Liu and D.F. Wong 1998
“Integrated Partitioning & Scheduling (IPS) algorithm”
 - Uniform SW and negligible HW execution times
 - No power consideration
- **Power-Aware Scheduling**
 - J. Liu, P.H. Chou, N. Bagherzadeh and F. Kurdahi 2001
“Power-Aware Scheduling using timing Constraints”
 - Use initial schedule assumption – may be inflexible



Contributions

- Considered power as important constraint during partitioning step, (in hybrid systems)
- Concurrent Mapping and Scheduling of tasks with non-uniform execution times – for Real-Time Applications,
- Used Reconfigurable systems for performance tuning through task migration



PAP Algorithm Overview

- Iterative improvement technique.
- Initial mapping: All Software
- Every iteration, one software task is selected for hardware mapping
 - Tasks *mobility* indices
 - Task Selection Routine
- Reschedule the tasks
- Schedule is verified to see if it meets its timing and power requirements.



Task Mobility

- Parallelism
- Schedule Dependent
- Time Interval (E_i, L_i) defined by mobility is used to schedule task i in hardware

- E_i is the earliest possible start time in HW

$$E_i = \max_{k \in \text{pred}(i)} (\eta(k))$$

$\text{pred}(i)$ is the immediate predecessor set of task i

$\eta(k)$: start time of task k



Task Mobility Contd.

- L_i is the latest possible finish time of task i in HW

$$L_i = \min_{k \in \text{succ}(i)} (\eta(k) - ts_i)$$

$\text{succ}(i)$ is the immediate successor set of task i

ts_i is the execution time of task i in SW

- Task Mobility of task i $\mu(i)$ is determined as follows:

$$\mu(i) = 1, L_i > E_i$$

$$0, L_i = E_i$$



Task Selection Routine

N_s : Set of software tasks in application

S.1 Rank the tasks in N_s in the order of decreasing software execution times ts_i

S.2 Compute the *mobility* $\mu(i)$ for all $i \in N_s$

S.3 If $\mu(i) = 0$ for all $i \in N_s$
Task i with maximum execution time ts_i is selected

Else

Task $i \in N_s$ with maximum execution time ts_i and *non-zero* mobility is selected



Definition: Time Valid Schedule

- T_{exec} : The finish time of a single iteration of the application
- $T_{\text{exec}} = \max (\eta(i) + t_i),$ for all $i \in N$
N is the set of tasks in the application
- Schedule: Time-Valid
If $T_{\text{exec}} \leq D,$ D is the application deadline



Power Valid (Definitions)

- Power Profile (P_{σ})
 - $P_{\sigma}(t) = \sum P(i)$, for all $i \in$ set of active tasks at time instant t
- Power Spike
 - $P_{\sigma}(t) > P_{\max}$
- Power-Valid
 - $P_{\sigma}(t) \leq P_{\max}$, $0 \leq t \leq T_{\text{exec}}$



Communication Model

- 32 bit 33 MHz PCI

- Delay Computation

P.V. Knudsen and Jan Madsen, 1998.

$$t_{\text{comm}} = \frac{AC + \frac{CC * N_{\text{sample}}}{N_{\text{bus}}}}{F}$$

- Power Dissipation

J.Buck, S. Ha, E.A. Lee, and D.G. Messerschmit, April 1994.

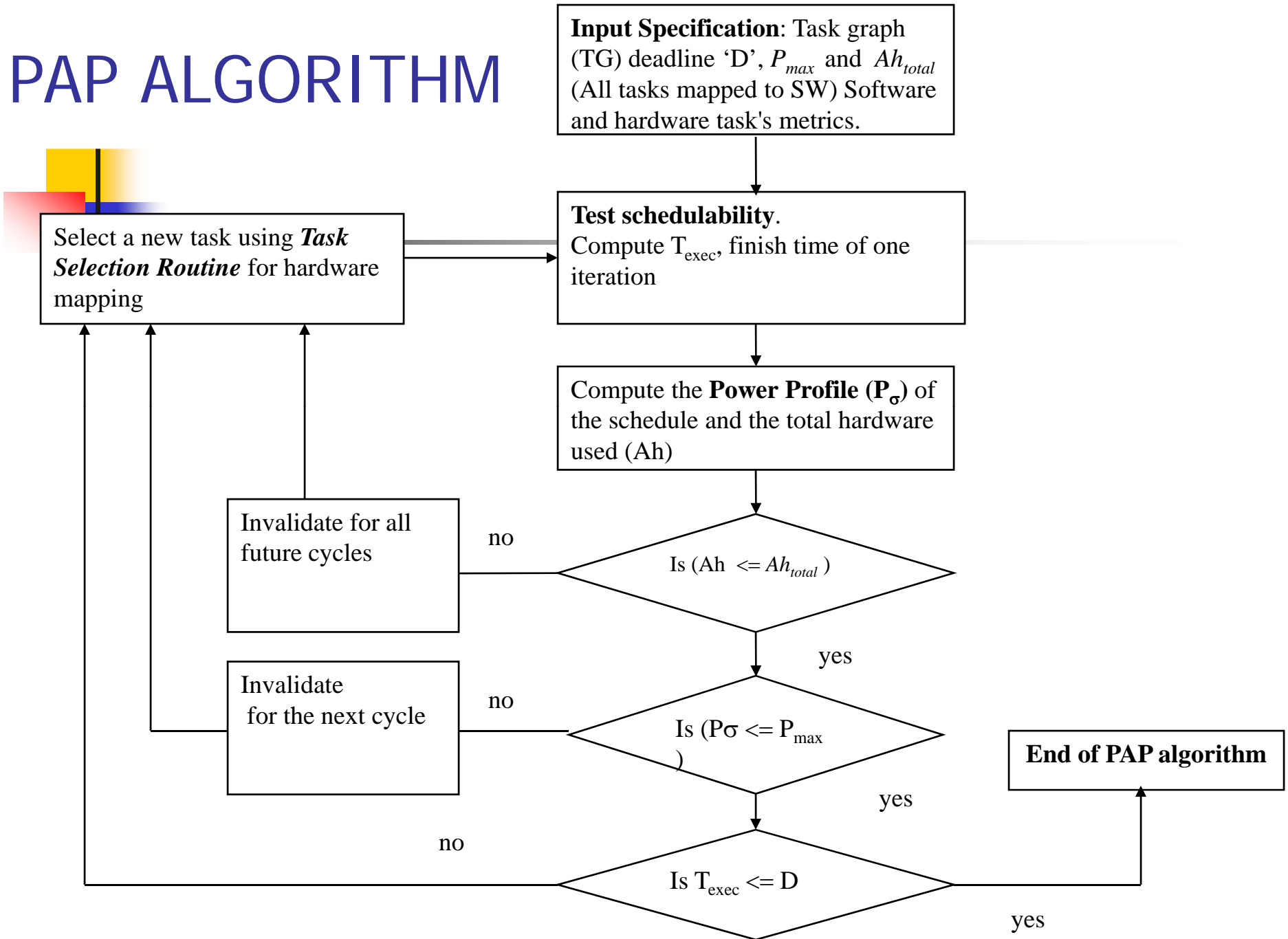
$$P_{\text{bus}} = \frac{1}{2} \times C_{\text{bus}} V^2 mn$$



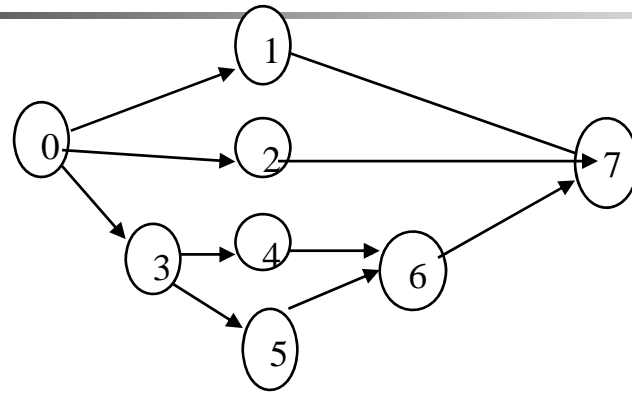
Scheduling the Bus communication

- No bus conflict is assumed.
- The execution of the hardware task and its communications should lie within the interval defined by its mobility.

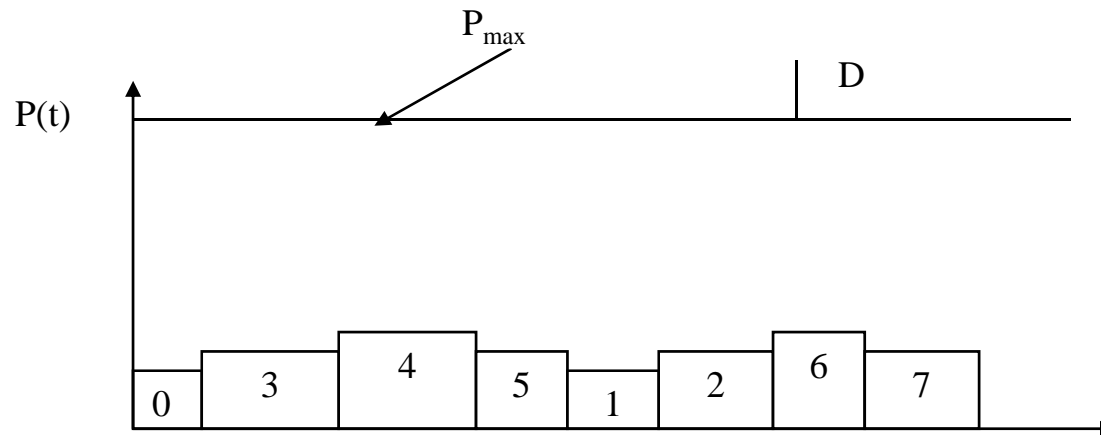
PAP ALGORITHM



Example of PAP algorithm

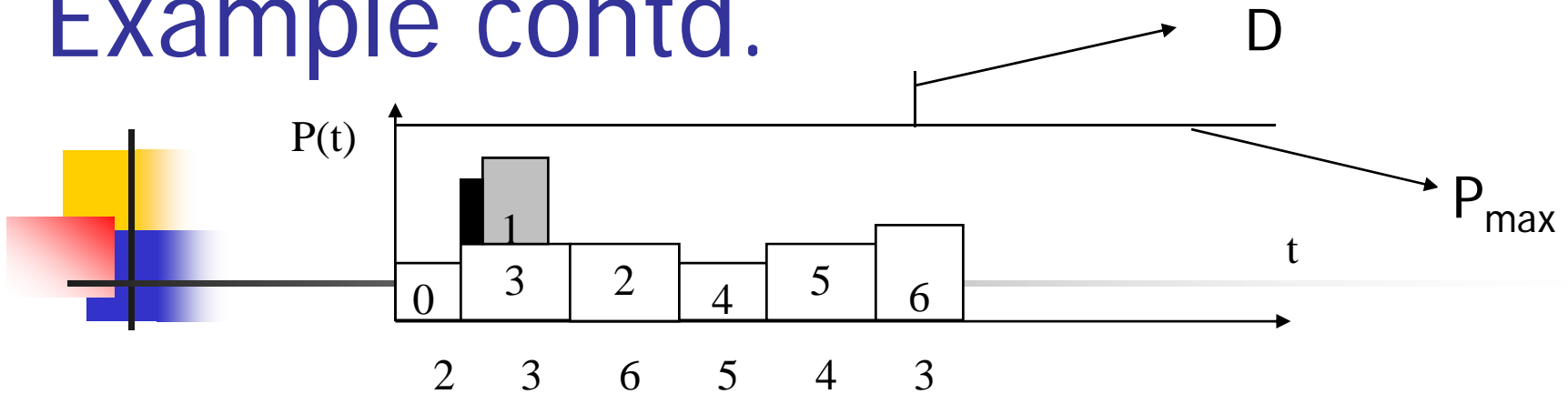


Application specified as a task graph

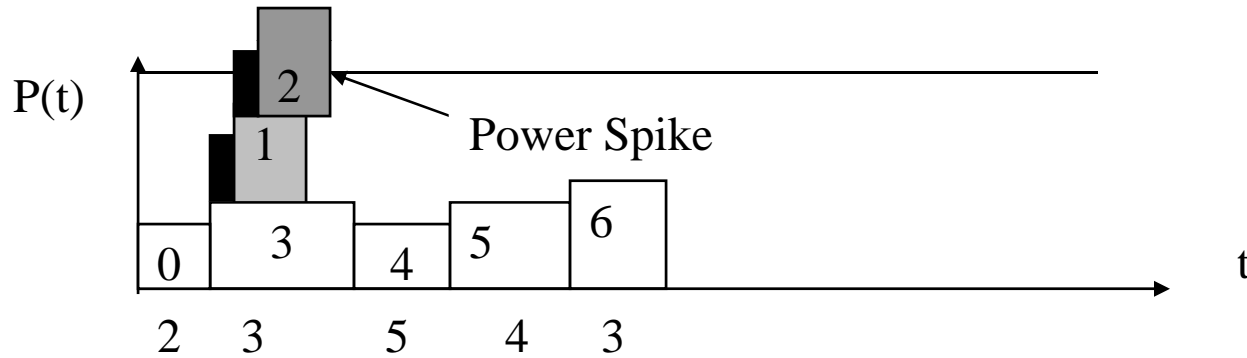


a. Initial schedule on CPU (all software)

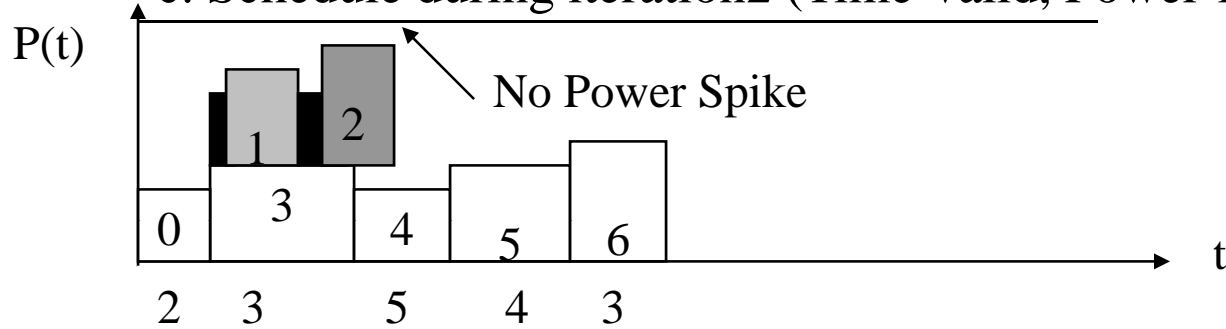
Example contd.



b. Schedule after iteration 1



c. Schedule during iteration 2 (Time-valid, Power-invalid)



d. Schedule after iteration 2 (Time-valid, Power-valid)



Partitioning of Multifunctional Systems

- Multifunctional systems- Support a set of applications.
- Set of active applications - Combined task graph (CTG).
- PAP extended to include information
 - Similar tasks
 - Hardware re-use
- Modified PAP applied to CTG



Application Criticality

- The set of active applications $\{A_1, A_2, \dots, A_n\}$ is ordered based on the criticalities.

- $$AC_i = \frac{T_{CTG}}{D_i}$$

T_{CTG} : Finish time of a single iteration of the CTG

D_i : Deadline of Application A_i



Modified Task Selection Routine

- All software tasks of CTG labeled with self and shared priorities.
- **Self-Priority:** Information about parallelism within 'own' application
- **Shared-Priority:** Information about similar tasks across the set of applications and hardware re-use.
- **Combined-priority:** Task selection index



Self-Priority: Computation

- S.1** Compute the *mobility* $\mu(i)$ for all $i \in N_s$, N_s is set of software tasks in application A_k
- S.2** Determine $N_{s1} \in N_s$, set of all software tasks with non zero mobility.

Similarly $N_{s2} \in N_s$, set of all software tasks with zero mobility.
- S.3** Initialize counter $Count = 0$



Self-Priority Contd.

S.4 Extract task i , $i \in N_{s1}$ with maximum execution time t_{si}

S.4.1 Compute $SeP(i) = \frac{N_s - Count}{N_s}$ for all $j \in N_s$

S.4.2 Increment *Count*

S.4.3 Remove task i from N_{s1}

S.4.4 Go to Step S.4

S.5 Extract task i , $i \in N_{s2}$ with maximum execution time t_{si}

S.5.1 $SeP(i) = \frac{N_s - Count}{N_s}$ for all $j \in N_s$

S.5.2 Increment *Count*

S.5.3 Remove task i from N_{s2}

S.5.4 Go to Step S.5



Shared-Priority Computation

- Num_i - Total Number of hardware implementations of similar tasks of task i in current iteration.

- The shared-priority $ShP(i) = \frac{Num_i}{\max Num_j}$ for all $j \in N_s$

N_s : Set of Software tasks of application A_k

MPAP Algorithm



Inputs: Set $\{A_1, A_2, \dots, A_n\}$, Deadlines, Ah_{total} and P_{max}

Outputs: Time and Power valid schedules for the set of applications

S.1 Set of applications is aggregated to form a single task graph CTG.
All tasks are initially mapped to software.
Schedule is assumed to be *Power-Valid*



MPAP contd.

- S.2** The Application Criticalities for $\{A_1, A_2, \dots, A_n\}$ are computed.

- S.3** Application with maximum application criticality is considered first.

- S.4** Task selected - Modified Task Selection Routine
Test Schedulability & Power Profile
Repeat for other applications in the ordered set $\{A_1, A_2, \dots, A_n\}$.



MPAP Contd.

S.5 If all applications have time and power-valid schedules

Terminate Algorithm

Else

Repeat from step S.2



MPAP: Complexity

- Task's mobility computation: $O(N)$
- The self and combined priorities: $O(N)$
- Sorting: $O(N \log N)$
- \therefore Modified task selection routine: $O(N \log N)$ time.
- Rescheduling takes $O(N)$ time.

- Initial all software schedule: $O(N^2)$
- At most N iterations
- Therefore, MPAP algorithm: $O(N^2 \log N)$ time

Case Studies



- Applications: 8 kHz 16-QAM Modem and DTMF Codec
- Specified in CGC domain of the Ptolemy system

- SW Processor: StrongARM SA-1100
- SW Estimates:
 - Timing and Power using JouleTrack (MIT)

- HW Resource: Xilinx-Virtex2 (XCV4000).
- Estimates: Xilinx ISE 4.2 simulator
 - Timing and Area using PAR
 - Power using XPower



Experiment1: PAP Vs Extensive Search

- Case Studies: 16-QAM and DTMF Codec
 - Periodic Deadline (D): 800 μ s.
- Applied PAP for 3 different P_{\max} (8W, 6W, 2W)
- Performed Extensive search for $P_{\max} = 8W$

Table1: Results from the PAP algorithm and the extensive search

Example	Method	Power (W)	Finish Time (μ s)	Search Time (sec)
16-QAM Modem	PAP	8	773	0.7
		6	780	0.7
		2	903	0.7
16-QAM Modem	Extensive Search	8	671	15310
DTMF Codec	PAP	8	791	0.8
		6	791	0.8
		2	966	0.8
DTMF Codec	Extensive Search	8	685	22160



Experiment 1: Results

- $P_{\max} = 6W, 8W$: Time-valid and Power-valid schedules
- $P_{\max} = 2W$: Time-invalid schedule for both cases.
- PAP Vs Extensive search
 - Comparable finish times for both case studies (for same hardware utilization)
 - Partitioning time (0.7 sec) is very low compared to 15K sec for 16-QAM Modem



Experiment2: MPAP(Self) Vs MPAP(Combined)

- Applied MPAP (self priorities) without hardware sharing for both case studies ($P_{\max} = 8W$)
- Applied MPAP (combined priorities) with hardware sharing for both case studies ($P_{\max} = 8W$)
- Compared the Hardware logic utilization (# of slices in the FPGA)



Table2: Total Hardware Area for the MPAP(self) and MPAP(combined) algorithms when applied to the 16-QAM Modem and DTMF Codec

Application/s	Algorithm	# of Slices
16-QAM and DTMF	MPAP (no sharing)	991
16-QAM and DTMF	MPAP (Combined)	803

- 23 % saving in hardware logic



Benefits of PAP/MPAP in RC Environment

- Admit and block applications for power and performance (task migration)
- QoS control for extended battery life



Summary

- Efficient concurrent Partitioning and Scheduling algorithm for reconfigurable systems has been proposed to meet power and timing constraints.
- Multifunctional Partitioning Algorithm : Area Efficient solution.
- Rapid estimation because proposed PAP/MPAP algorithm's run time is low.
- Suitable for dynamically changing set of applications.

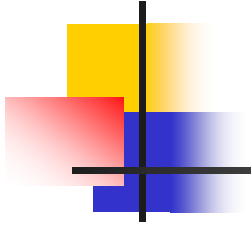


Future Work

- Understand the heuristic's behavior with more experiments
- Extend the scheme to distributed embedded systems.
- Adopt V/F scaling in CPU and F-scaling selectively in FPGA.



Questions ?



Thank You