ASP-DAC 2018

Task Assignment and Scheduling in MPSoC under Process Variation: A Stochastic Approach

Behnam Khodabandeloo

Ahmad Khonsari, Alireza Majidi, Mohammad Hassan Hajiesmaili

Department of Electrical and Computer Engineering, The University of Tehran, Tehran, Iran

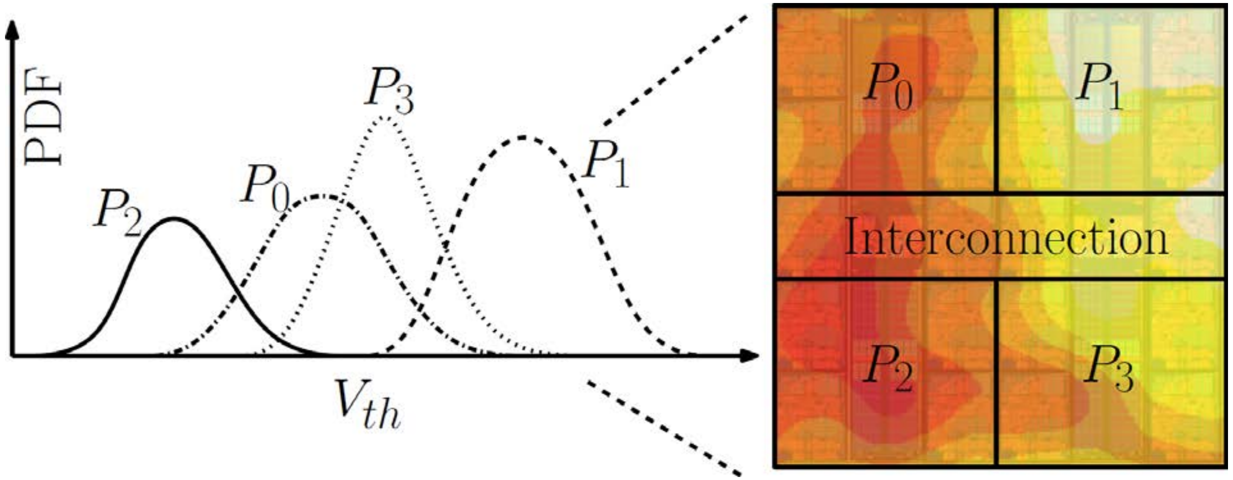
School of Computer Science, Institute for Research in Fundamental Sciences, Tehran, Iran

Outline

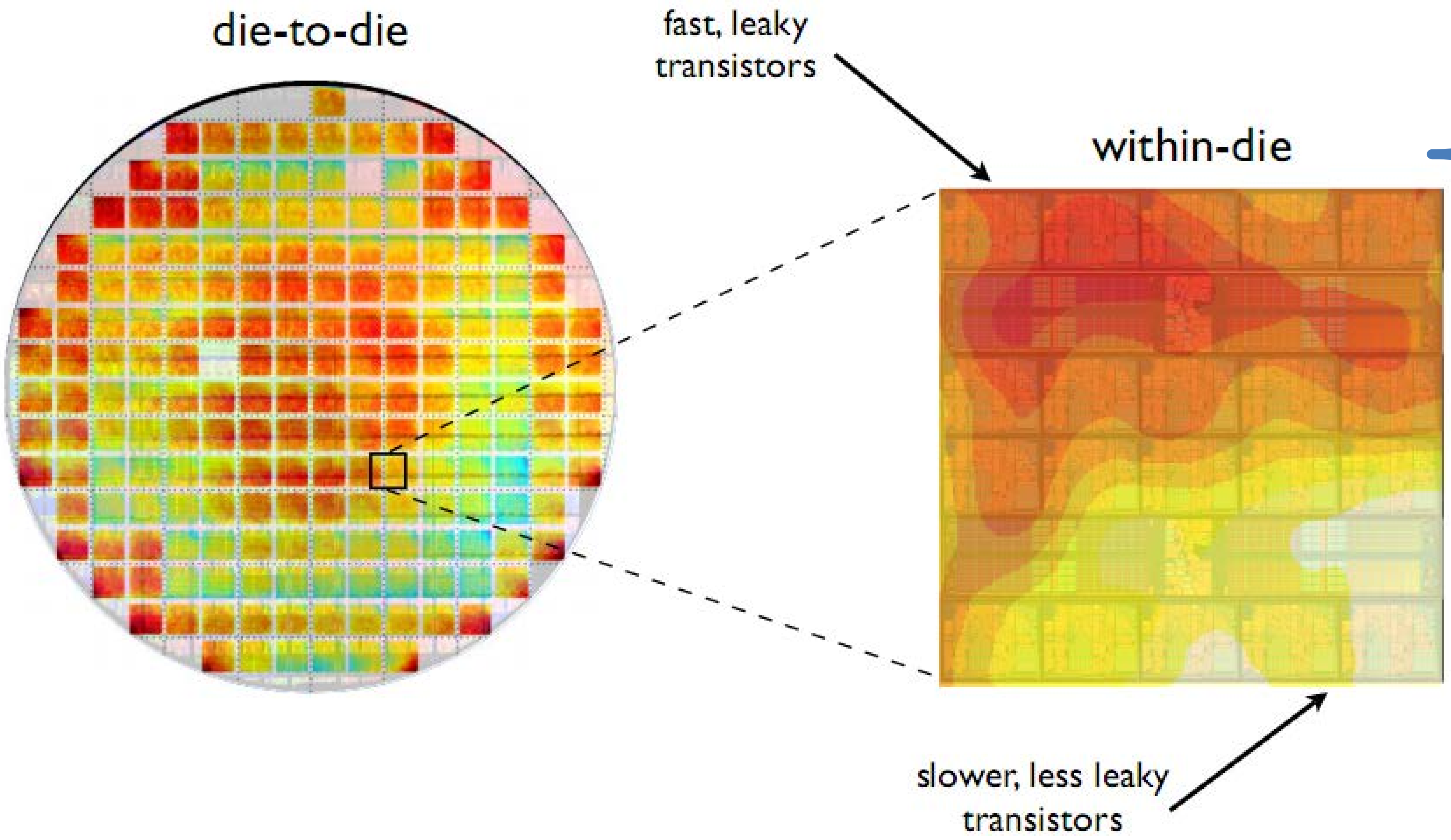
- Introduction
- Problem definition
- The proposed method
- Experimental result
- Conclusion

Process Variation

- To conceal process variation effects on system level.
  - □ Numerous works have been carried out in:
    - Logic level [1]
    - Arithmetic levels [2]
- However failure to hide its effects completely leads to variations on
  - □ Frequency
  - □ Leakage power



Process Variation Parameter Categories

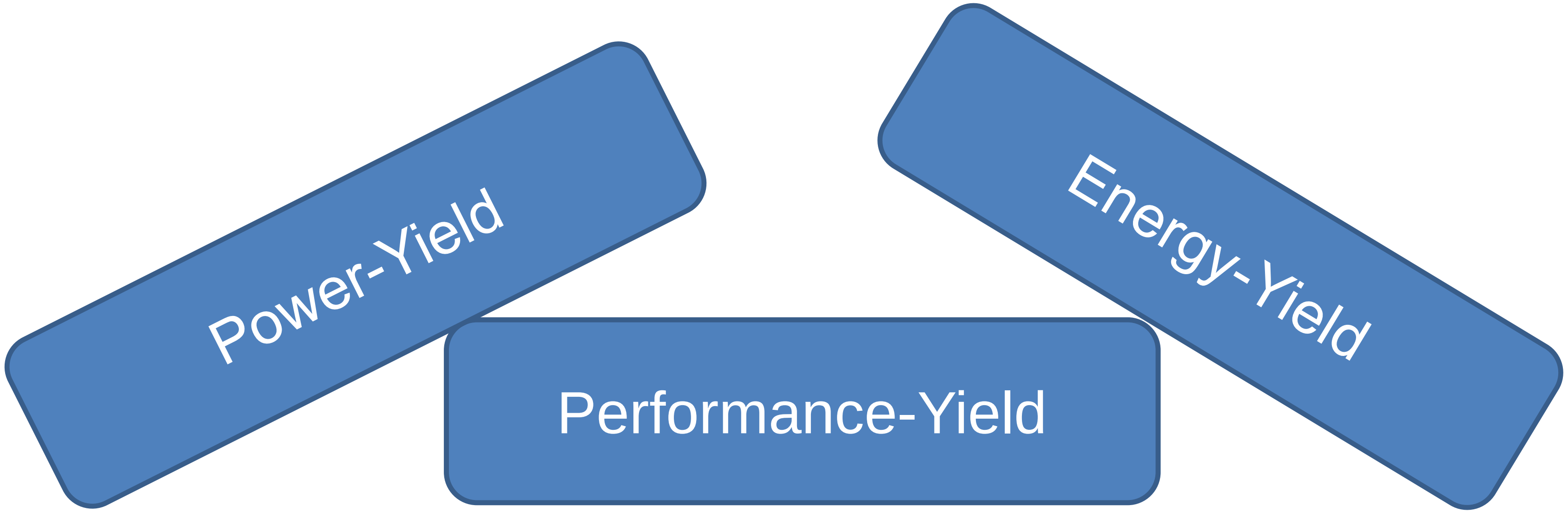


- Systematic
- Random

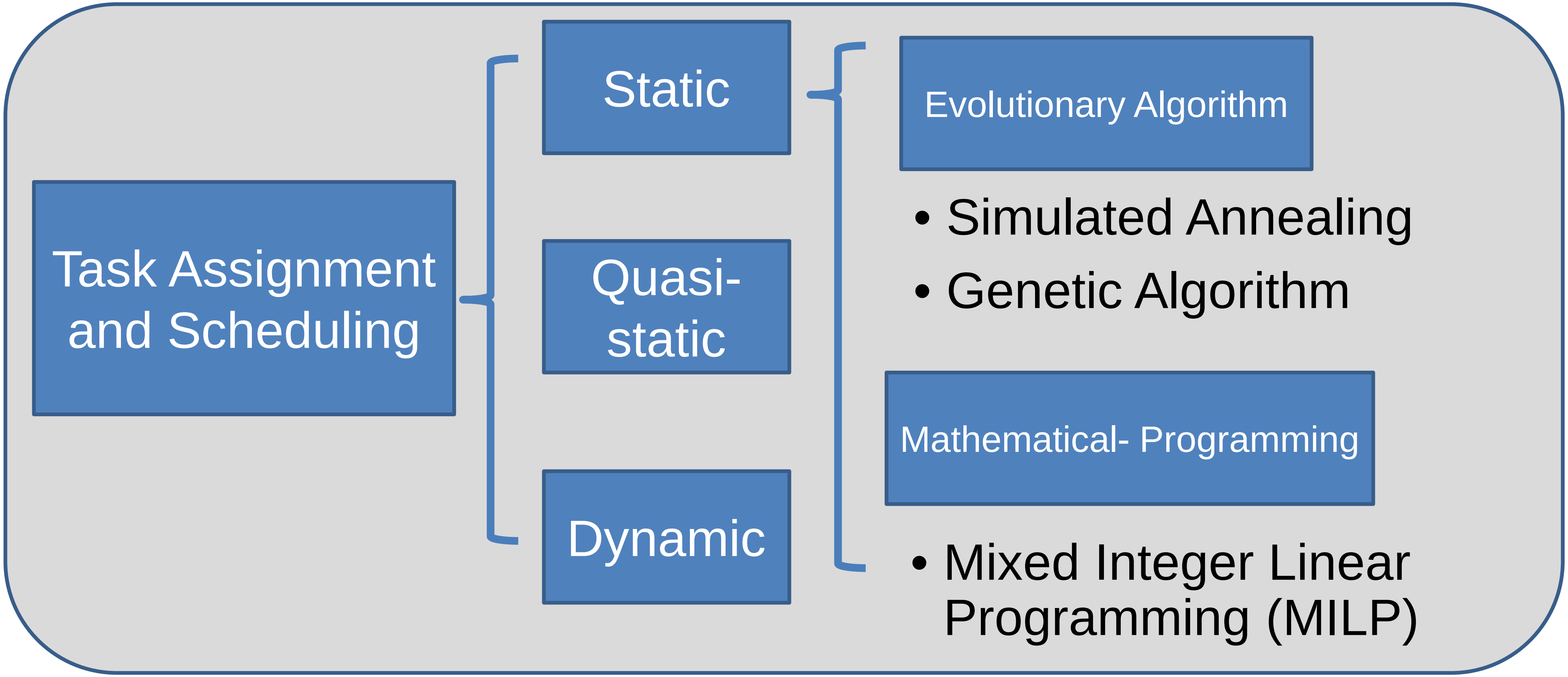
[3]

Process Variation Aware Measures

- New measures known as parametric-yields have been proposed.
- Performance-yield is defined as the probability of the generated schedule happens to meet the timing constraint of the system.



Task Assignment and Scheduling Methods



Task Assignment and Scheduling

Static

Quasi-static

Dynamic

Evolutionary Algorithm

- Simulated Annealing
- Genetic Algorithm

Mathematical- Programming

- Mixed Integer Linear Programming (MILP)

MILP-based Solutions

- To calculate parametric yields:

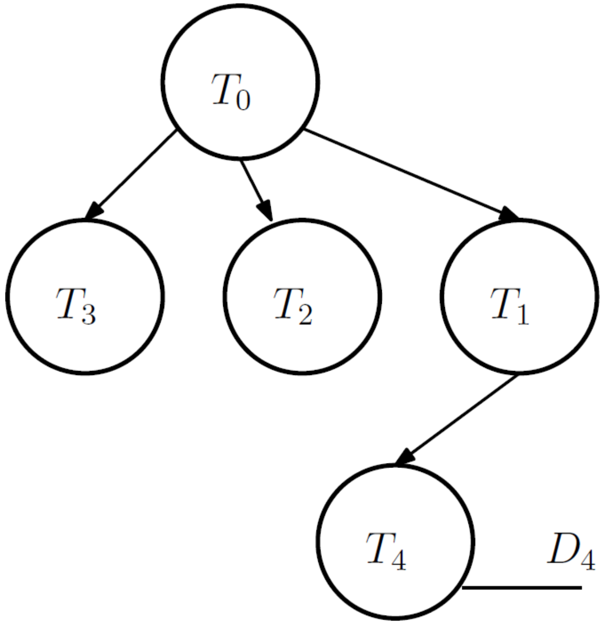
Using Monte Carlo Simulation [4]

- To generate test chips.

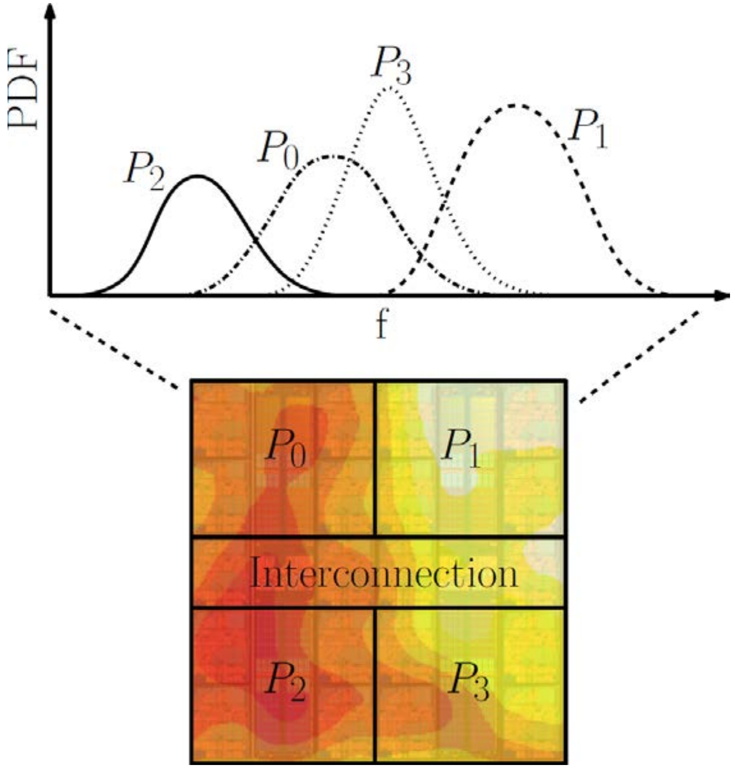
Discretizing the Probability Distribution Function (PDF) [5]

- To linearize problem constraints.

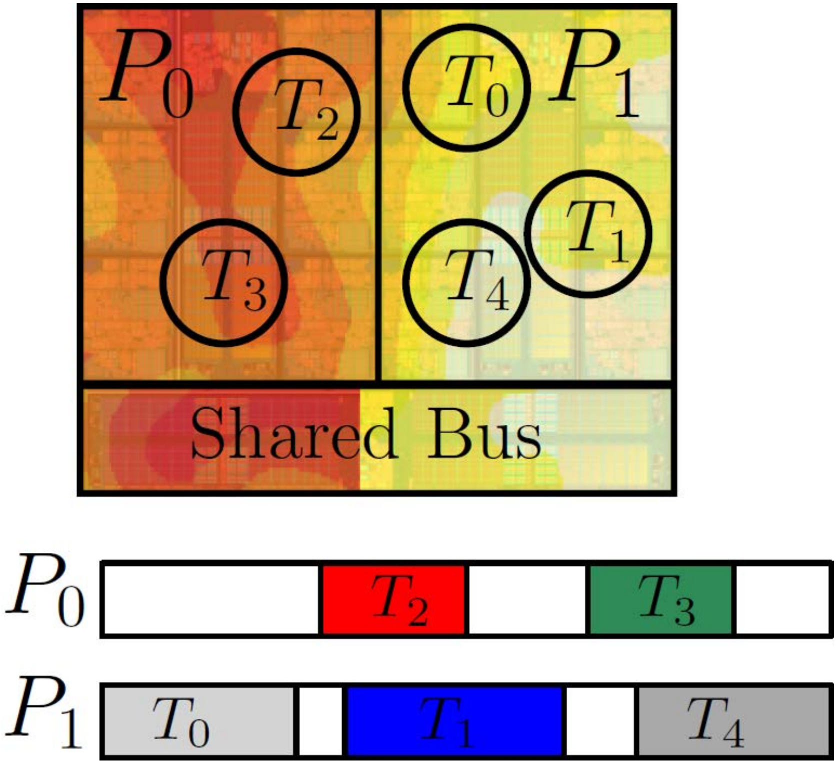
Problem Definition



Input:
Application model



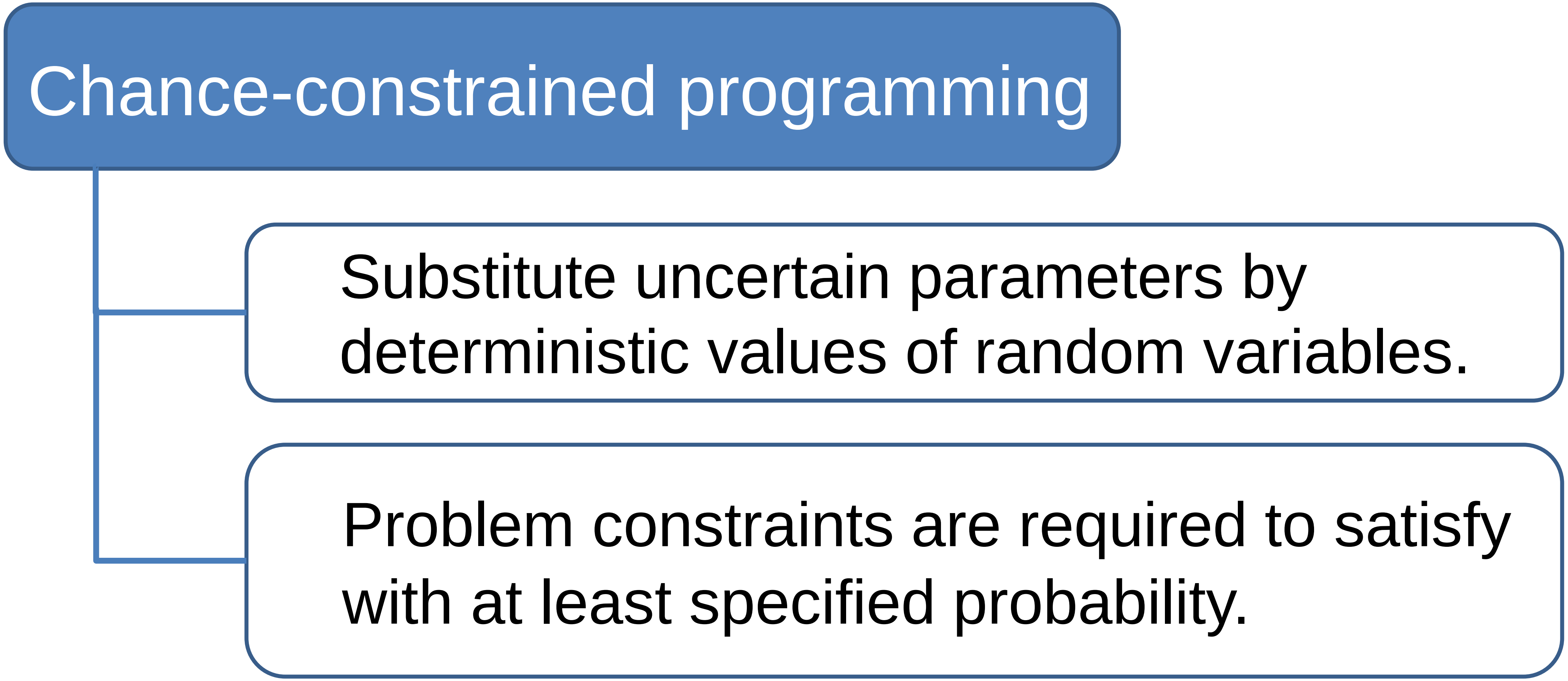
Input:
MPSoC platform model



Output:
Task Assignment and Scheduling

We assume the result task schedule is non-preemptive.

The Proposed Method

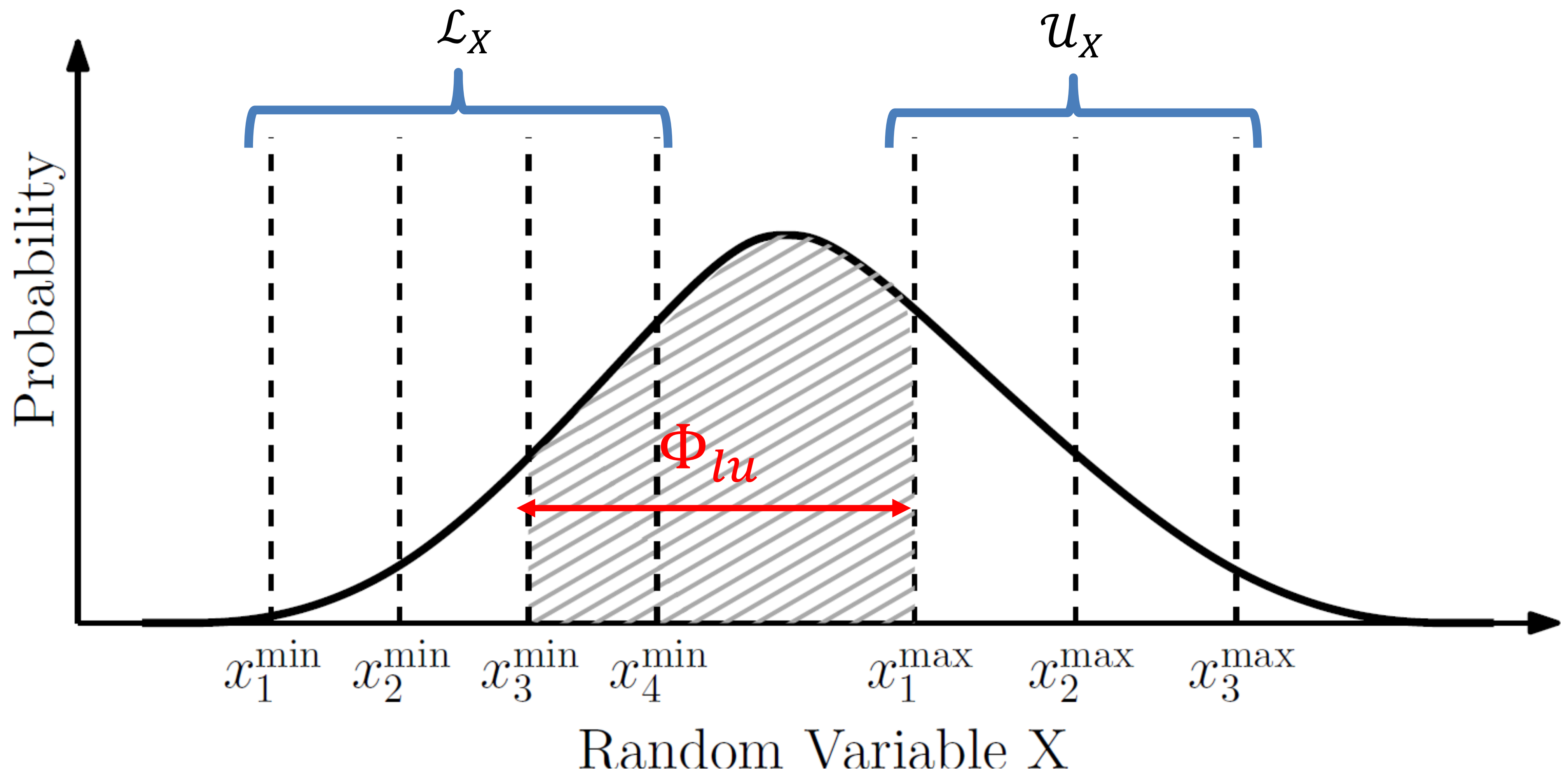


Chance-constrained programming

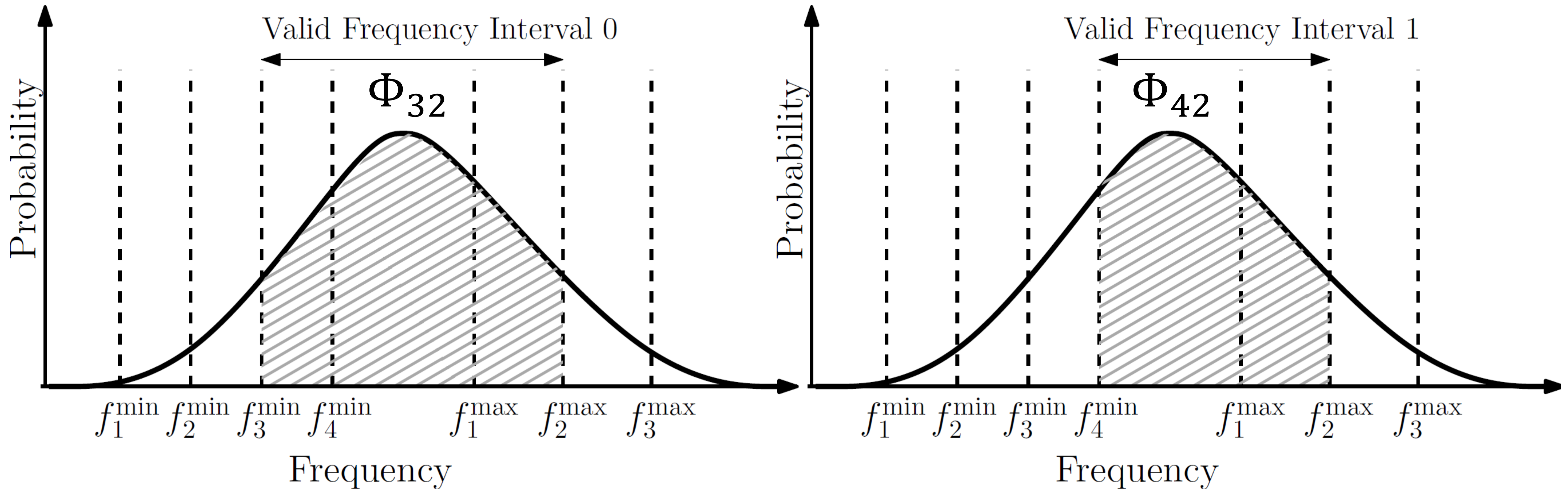
Substitute uncertain parameters by deterministic values of random variables.

Problem constraints are required to satisfy with at least specified probability.

Valid Parameter Interval (VPI)



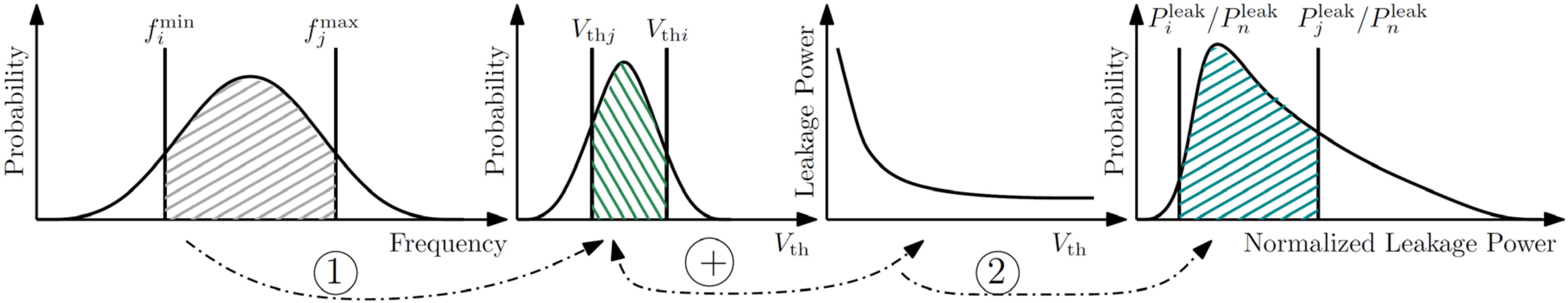
Valid Frequency Interval (VFI)


Valid Frequency Interval 0
Φ_{32}
Probability
$f_1^{\min}$ $f_2^{\min}$ $f_3^{\min}$ $f_4^{\min}$ $f_1^{\max}$ $f_2^{\max}$ $f_3^{\max}$
Frequency
Valid Frequency Interval 1
Φ_{42}
Probability
$f_1^{\min}$ $f_2^{\min}$ $f_3^{\min}$ $f_4^{\min}$ $f_1^{\max}$ $f_2^{\max}$ $f_3^{\max}$
Frequency

$$\Pr(\Phi_{32}) \geq \Pr(\Phi_{42}) \geq \rho_{th}$$

$$f_3^{\min} < f_4^{\min}$$

The proposed approach for computing the expected value of leakage power of a VFI



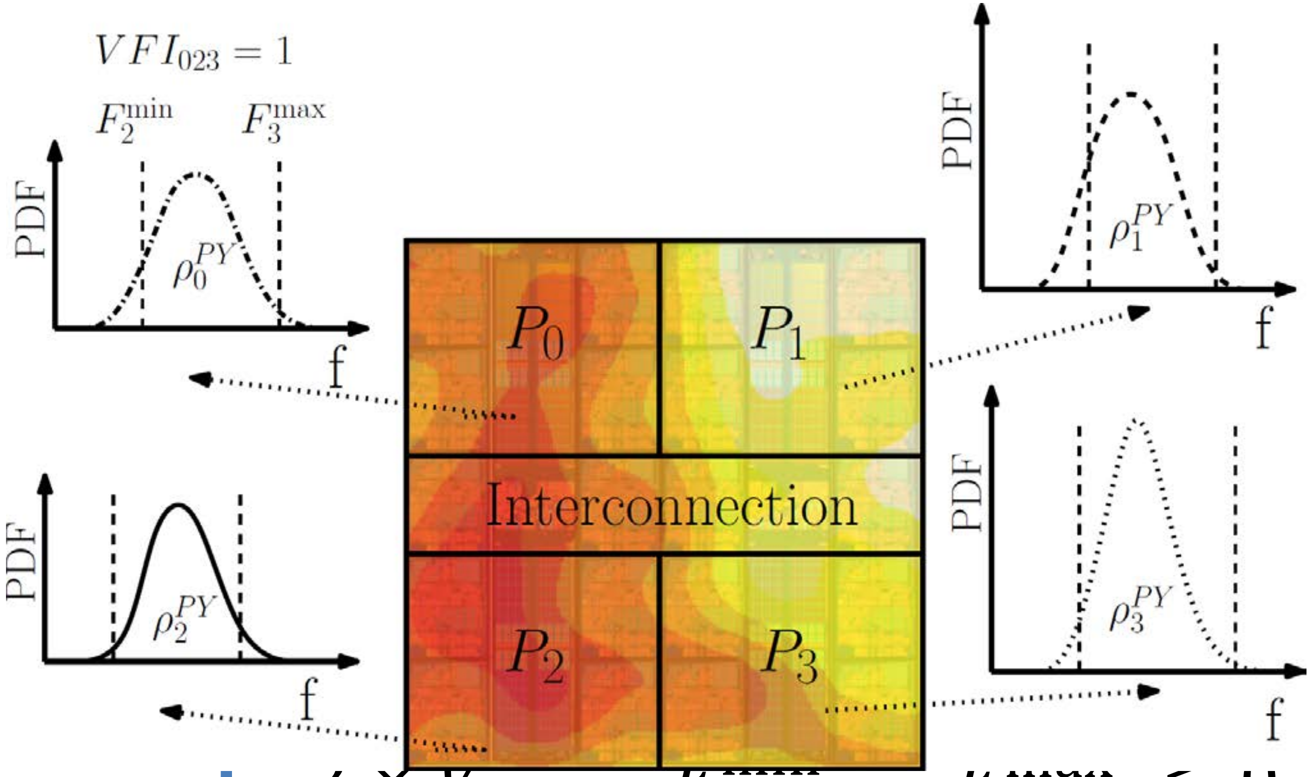
- Computing the corresponding leakage power interval from a given VFI.
  - □ (1): Finding a corresponding voltage threshold interval for a given VFI.
  - □ (2) Finding a corresponding leakage power interval for a given voltage threshold Interval.

The Problem Formulation

- *Assignment Constraints*
- *Performance-Yield Constraints*
- *Timing constraints*
- *Thermal and Power Constraints*
- *Objective Function*

Performance-yield Calculation

$$(1)\ \forall p \in \mathcal{P}, \sum_{l \in \mathcal{L}_F} F_{pl}^{\min} = 1$$



$$2 \times V_{plu} - F_{pl}^{\min} - F_{pu}^{\max} \leq 0$$

$$(4)\ \rho^{PY} = \prod_{p \in P} \rho_p^{PY}$$

$$\log \rho^{PY} = \sum_{p \in P} \log \rho_p^{PY}$$

$$\log \rho_p^{PY} = \sum_{u \in \mathcal{U}_F} \sum_{l \in \mathcal{L}_F} \log \rho_{plu}^{P} V_{plu}$$

$$\log \rho^{PY} \geq \log \rho_{th}^{PY}$$

Experimental Setup

- We use Embedded System Synthesis Benchmarks Suite (E3S).
- Floorplan configurations:
  - □ Homogenous
  - □ Heterogeneous
- Benchmark types:
  - □ Single-program benchmarks
  - □ Multi-program benchmarks

Benchmark	Tasks
Office, Office, Office	15
Automation, Networking	37
Automation, Consumer	36
Telecom, Networking	43
Telecom, Consumer	42
Telecom, Automation	54
Networking, Consumer, Office	30
Automation, Networking, Consumer	49

Experimental Setup (2)

- All required parameters of leakage power and frequency models
  - □ Extracted by SPICE under 22nm technology
- To model process variation
  - □ Using Various model [6].
- To generate the probability distribution of relative chip frequency as a function of V_{th}:
  - □ 6 FO4s in critical path
  - □ 10000 critical paths
- The MILP formulations are solved using IBM CPLEX 11.1 tools [7].

Experimental Results

- Related works for comparison
  - ✖ The formulation in [8]:
  - □ The formulation in [5]:
    - We use this formulation as baseline method.
    - To have a fair comparison
      - □ We use identical computation for power and energy in both methods.
    - This formulation does not consider temperature
      - □ We ignore the effects of this parameter in our comparison.
- Comparison Metrics:
  - □ Performance-yield (PY)
  - □ Energy-yield (EY)

Performance-Yield Comparison

- The results of performance-yield metric show 3 categories of benchmark behaviors that are dependent on the nature of the applications.
  - The first category is related to applications with a tight deadline.
  - The second category is related to applications with a relaxed deadline.
  - The third category is mainly for multi-program benchmarks with 3×3 floorplan configurations.

Performance-Yield Comparison – First Category

Benchmark	Proposed Formulation		Baseline [5]		Improvement	
	PY	Run-time(s)	PY	Run-time	PY	Run-time(X)
AutHo	90	0.342	0	182.209	90	532.775
AutHe	90	0.355	0	44.977	90	126.696
NetHo	80	0.492	0	16.010	80	32.541
NetHe	80	0.414	0	16.756	80	40.473
ConHe	31.5	5.272	0	95.463	31.5	18.107

- The proposed formulation provides improvements in both performance-yield (PY) and run-time that fall within [31.5%, 90%] and within [18X, 532X], respectively.

Performance-Yield Comparison – Second Category

Benchmark	Proposed Formulation		Baseline [5]		Improvement	
	PY	Run-time(s)	PY	Run-time	PY	Run-time(X)
ConHo	100	0.063	100	6.512	0	103.365
TelHo	100	0.686	100	81.261	0	118.456
TelHe	100	0.851	100	205.269	0	241.209
3OffHo	100	0.156	100	1.7	0	10.897
3OffHe	100	0.125	100	1.513	0	12.104

- Applications with a relaxed deadline.
- PY is equal for both methods.
- The proposed formulation improves the run-time that falls within [10X, 241X].

Performance-Yield Comparison – Third Category

Benchmark	Proposed Formulation		Baseline [5]		Improvement	
	PY	Run-time(s)	PY	Run-time	PY	Run-time(X)
AutCon	90	7.301	RoM	RoM	90	-
AutNet	64.8	17.831	RoM	RoM	64.8	-
TelCon	100	9.141	RoM	RoM	100	-
TelNet	80	220.007	RoM	RoM	80	-
TelAut	90	153.271	RoM	RoM	90	-
NetConOff	80	19.266	RoM	RoM	80	-
AutNetCon	26.2	292.532	RoM	RoM	26.2	-

RoM: Running out of memory.

- The third category which is mainly for rectangular mesh configuration with dimension size 3 produce no results for baseline due to the MILP solver running out of memory before finding a feasible solution
- The achieved improvements in PY falls within [26.2%, 100%] with an average of 74.4%.

Energy-Yield Comparison

Benchmark	Proposed Formulation		Baseline [5]		Improvement	
	EY	Run-time(s)	EY	Run-time	EY	Run-time(X)
ConHo	66	0.655	66	188.683	0	288.066
3OffHo	80	0.078	80	15.943	0	204.397
3OffHe	80	0.359	80	16.587	0	46.203
TelHo	90	35.912	90	2176.89	0	60.617
TelHe	86	23.977	86	799.845	0	33.359

- Energy-Yield (EY) is equal for both methods.
- The proposed formulation improves the run-time that falls within [33X, 288X].

Decision Variables Comparison

Variable Name	Proposed Formulation		Baseline [5]	
	Notation	Numbers	Notation	Numbers
Task Assignment	X_{ij}	$N \times M$	$PAlloc_{i,p}$	$N \times M$
Task Start Time	T_i^{st}	N	$TS_{i,a}$	$N \times L$
Task Finish Time	T_i^{fn}	N	$TE_{i,a}$	$N \times L$
Task Orders	η_{ij}	$N \times N$	$TOrder_{i,j}$	$N \times N$
Performance-Yield	F_{pl}^{min} , F_{pu}^{max}	$2 \times M \times S$	$Meet_{a_1 \ldots a_M}$	M^L

N = Task number ,M = Processor number, $S = \max(|\mathcal{L}_F|, |\mathcal{U}_F|)$, $S \sim L$

- The reasoning behind achieving better run-time of our method is due to the fact that the number of decision variables in our formulation is much less than the number of decision variables in baseline formulation [5] as shown in this Table

Conclusion

- Using the chance-constrained programming
  - □ We captured the process variation effects on the frequency and the leakage power of MPSoC processors.
- Using valid frequency interval (VFI)
  - □ The number of decision variables in our formulation is much less than the number of decision variables in baseline formulation[5].
  - □ The proposed method achieves higher PY and better run-time in comparison with baseline method.

Reference

[1] F. Wang, et al., ”Variability-driven module selection with joint design time optimization and post-silicon tuning,” Proc. DAC, 2–9, 2008.

[2] A. Agarwal, et al., Statistical timing analysis for intra-die process variations with spatial correlations, Proc. ICCAD, 900–907, 2003.

[3] http://www.cse.ohiostate.edu/ teodores/download/slides/radu-isca08-slides.pdf.

[4] K. Bhardwaj, et al., ”Power-Performance Yield optimization for MPSoCs using MILP,” Proc. ISQED, 764–771, 2012.

[5] M. Ghorbani, ”A Variation and Energy Aware ILP Formulation for Task Scheduling in MPSoC,” Proc. ISQED, 772–777, 2012.

[6] R. Teodorescu, et al., ”VARIUS: A model of parameter variation and resulting timing errors for microarchitects,” 2nd Workshop on Architectural Support for Gigascale Integration, 2007.

[7] CPLEX 11.1ILOG: http://www.ilog.com/product/cplex/, 2011.

[8] T. Chantem, et al., ”Temperature-Aware Scheduling and Assignment for Hard Real-Time Applications on MPSoCs,” IEEE Transactions on Very Large Scale Integration Systems, 19(10) 1884–1897, 2011.

Question

Thank you!