Agile Frequency Scaling for Adaptive Power Allocation in Many-core Systems Powered by Renewable Energy Sources

Xiaohang Wang, Zhiming Li, Mei Yang, Yingtao Jiang, Masoud Daneshtaleb and Terrence Mak

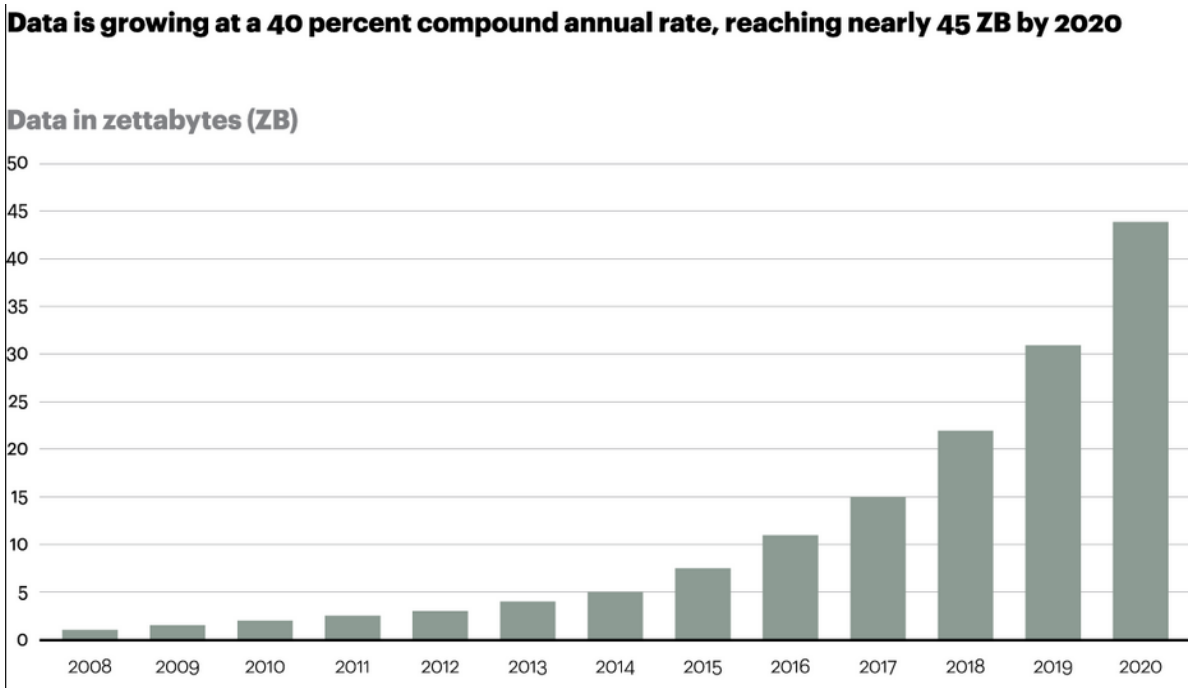
Guangzhou Institute of Advanced Technology, Chinese Academy of Science

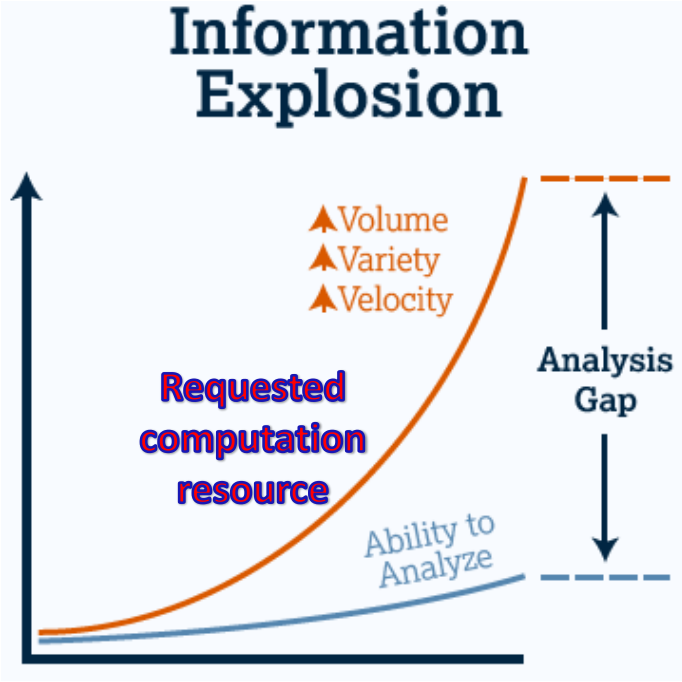
Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

Background

- Big data → request for computation resource





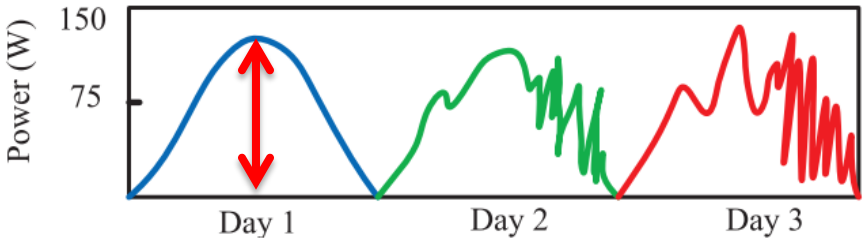
Background

- Blindly increasing the computation resource → explosive increase in power consumption and $$$
  - Power hungry !

Background

- So how to reduce the power consumption and save money ?
  - 1. renewable energy based computing

Power budget varies



  - 2. power budgeting to improve the power efficiency (GFLOPS/Watt)

Background

- Problem to solve and the challenges
  - Optimize the performance over a given power budget.
  - Challenge 1: large solution spaces.
    - 16-core, each can run at 4 frequency levels → 4^{16} choices
  - Challenge 2: should be fast and prompt enough to track the power budget variation

Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

Related work

- Power allocation at core level [1], system level [2], NoC [3], etc.
- Techniques: DVFS [2], power gating [4], etc.
- Shortcomings
  - Heuristic-based, ad-hoc: sub-optimal
  - Linear/ convex programming: High run time overhead and might consume much power
  - Poor scalability

[1] Li et al, HPCA'06
[2] Ma et al, PACT'12
[3] Sharifi et al, PACT'12
[4] Reda et al, MICRO 2012

So what we propose ?

- A dynamic programming network (DPN) based power allocation method
  - Using a hardware circuit to solve the problem
  - Globally optimal solutions
  - Can allocate power for multiple applications
  - Very fast (linear complexity) and low overhead (in terms of both area and power consumption)

Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

Models

- Suppose *Q* applications
- Power model

$$P = \sum_{i=1}^{n} \alpha_i \cdot C_i \cdot f_i \cdot V^2 = \sum_{i=1}^{n} b_i \cdot f_i$$

- Performance model
  - $Cycle = g_{cycle}(f_1, \ldots, f_{Nq})$
  - We find the *ln*() function is a good approximator

$$\ln Cycle = \sum_{i=1}^{N_q} a_i \cdot \sqrt{f_i}$$

Problem formulation

- Problem
  - Can be converted to the knapsack problem by dropping the *In* notation

$$\max \Pi = \sum_{i=1}^{n} a_i \cdot f_i$$

subject to

$$\sum_{i=1}^{n} b_i \cdot f_i + P_l \leq P_{in}$$

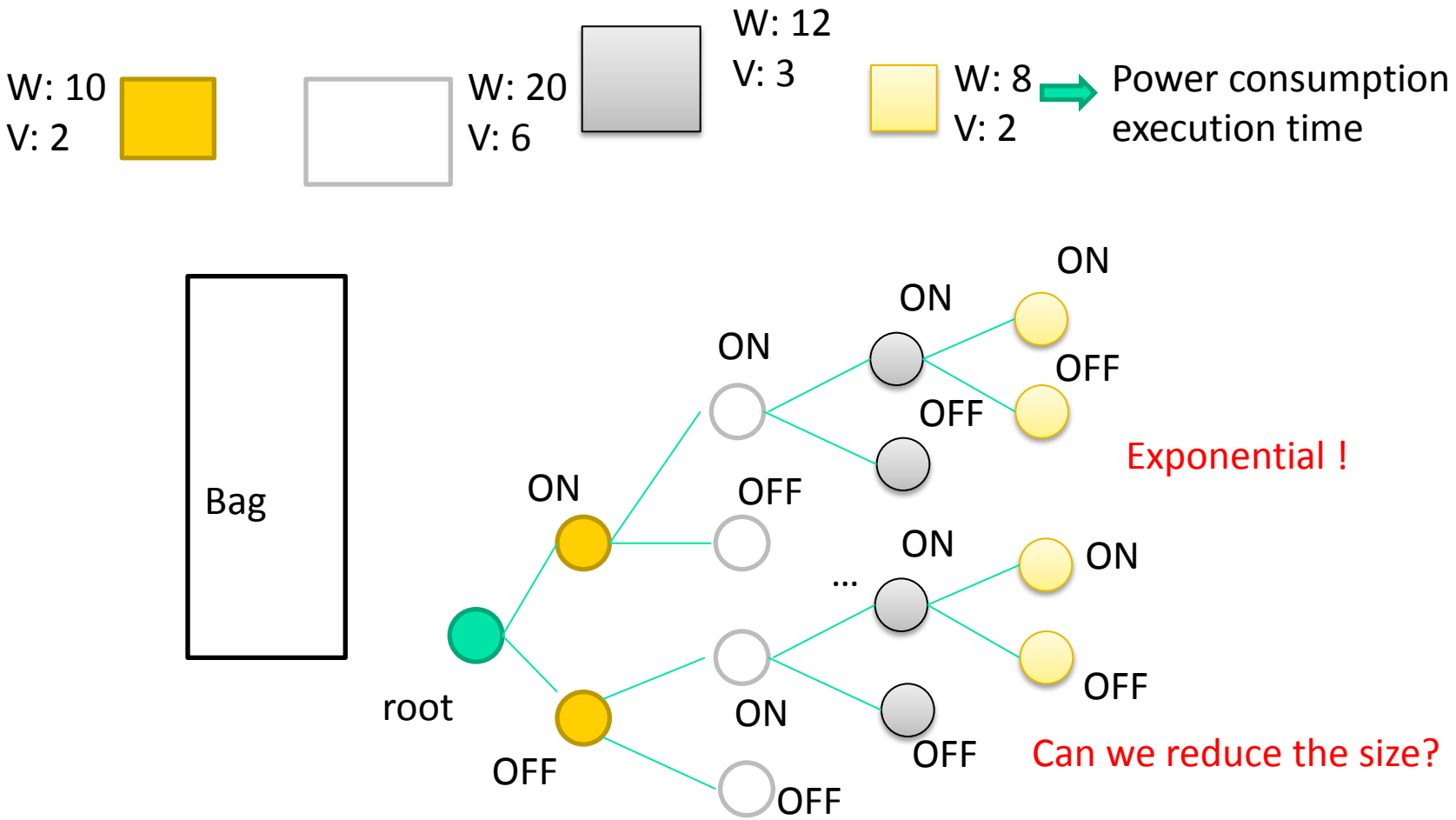
for each $f_i \in \{\tau_1, \ldots, \tau_m\}$

Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

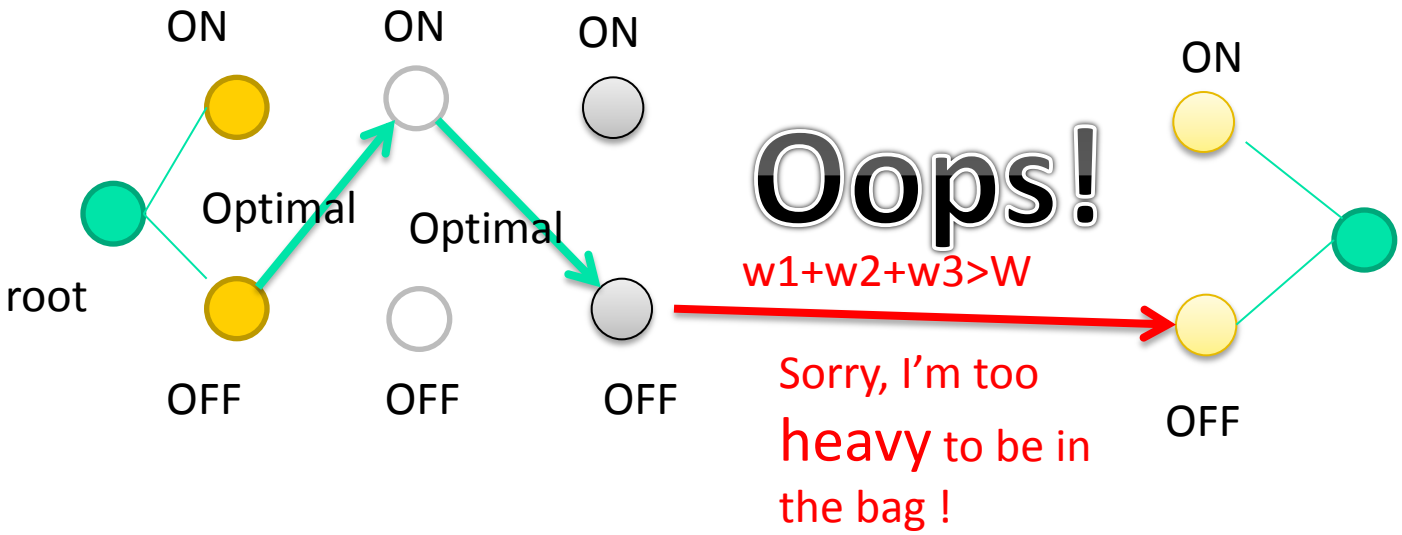
The proposed algorithm
Exhaustive approach

- The knapsack problem



The proposed algorithm
Native approach

- Let's try



Not Markovian!
Let's convert it to be Markovian

The proposed algorithm

Type -1 DPN

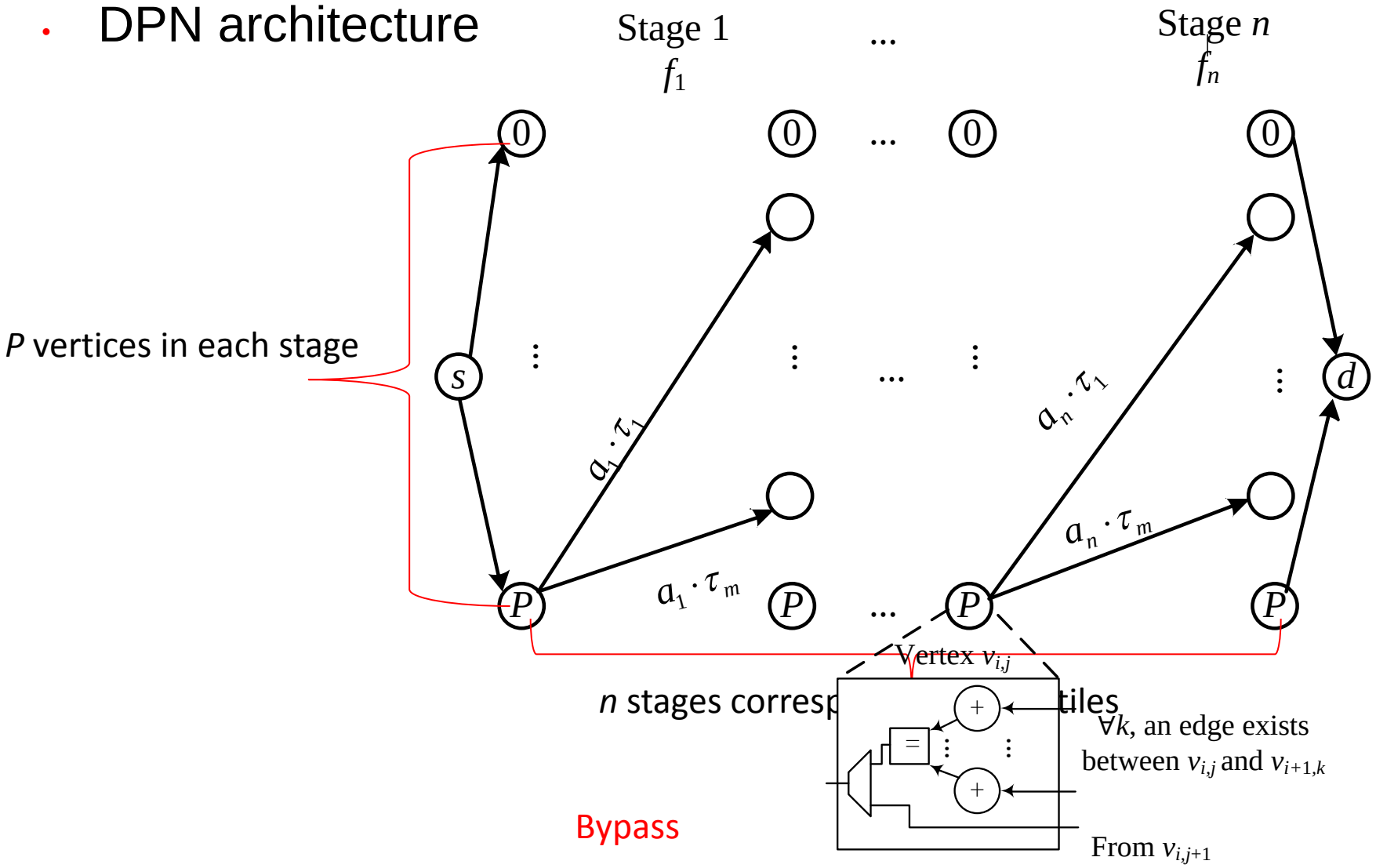
- Dynamic programming network DPN(V, E)
  - V: DP value $V(v_{i,p})$: the max value of assigning f_i given a power budget of p,
  - E: each vertex at stage i is connected to at most m vertices in the next stage i+1.
    - An edge exists between two vertices, $v_{i,p}$ and $v_{i+1,q}$ if $p - q$ = $b_i \cdot \tau_l$ for $1 \le l \le m$

$$C\left(v_{i,p}, v_{i+1,q}\right) = \begin{cases} a_{i+1} \cdot \tau_l, & \text{if } p - q = b_i \cdot \tau_l \\ -\infty, & \text{otherwise} \end{cases}$$

The proposed algorithm

Type -1 DPN

- DPN architecture



The proposed algorithm

Type -1 DPN

- DPN traversal
  - Each node selects an output edge with

$$V(v_{i,p},d)=\max_{\forall q}\{C(v_{i,p},d)+V(v_{i+1,q},d),V(v_{i,p},d)\}$$

The proposed algorithm

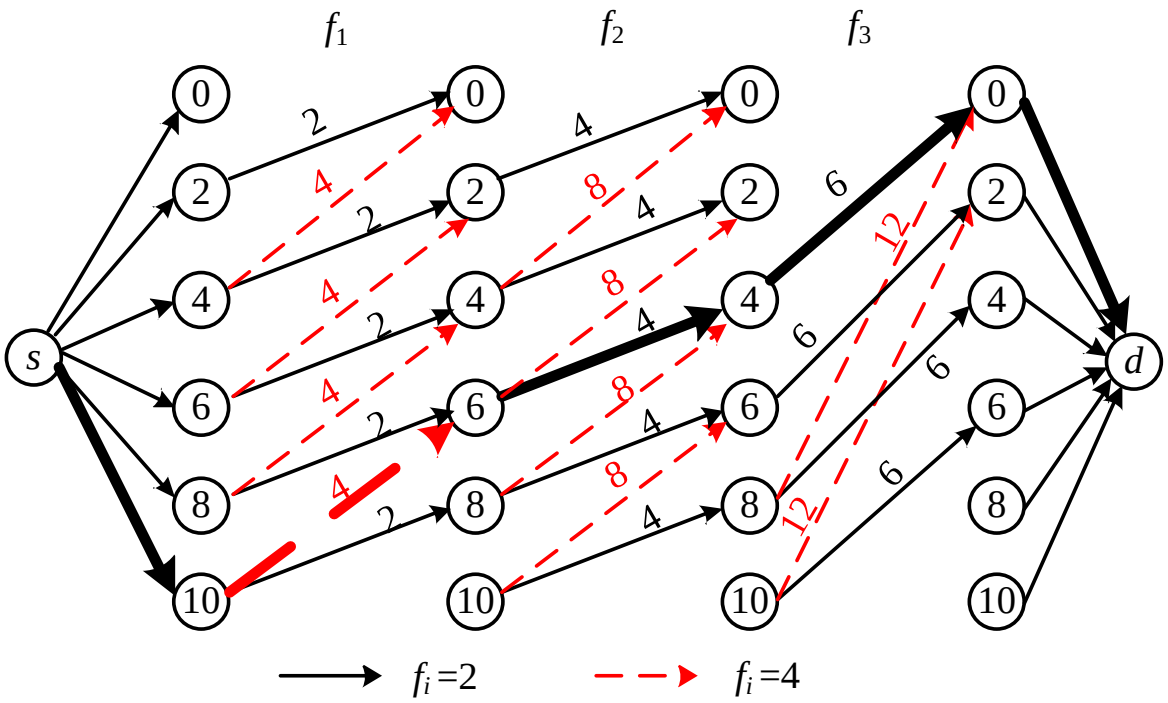
Type -1 DPN

- An example

max: $Perf = f_1 + 2f_2 + 3f_3$

Problem : $P = f_1 + f_2 + 2f_3 \leq 10$

$f_i \in \{2, 4\}$



Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

Evaluation

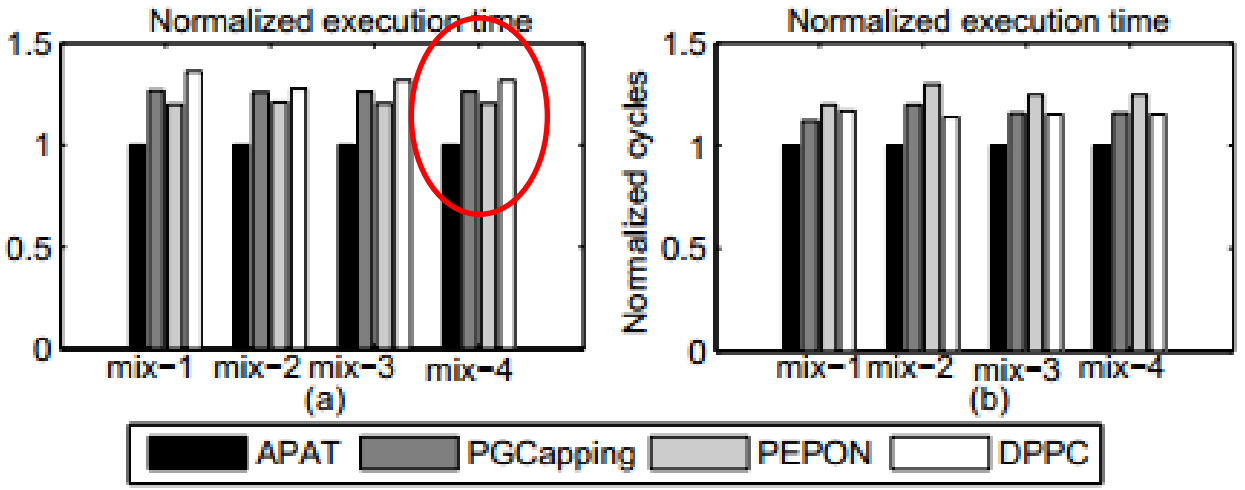
- Setup
  - 8x8
  - Com
    - P
    - P
    - D

Number of processors	64
Fetch/Decode/Commit size	4 / 4 / 4
ROB size	64
L1 D cache (private)	16KB, 2-way, 32B line, 2 cycles, 2 ports, dual tags
L1 I cache (private)	32KB, 2-way, 64B line, 2 cycles
L2 cache (shared)	64KB slice/node, 64B line, 6 cycles, 2 ports
Frequencies available	1GHz, 800MHz, 500MHz, 330MHz
On-chip network parameters	
NoC flit size	72-bit
Data packet size	5 flits
Meta packet size	1 flit
NoC latency	router: 2 cycles, link: 1 cycle
Number of VC in NoC	4
NoC buffer size	5x12 flits

T'12
PACT'12
TC' 2013

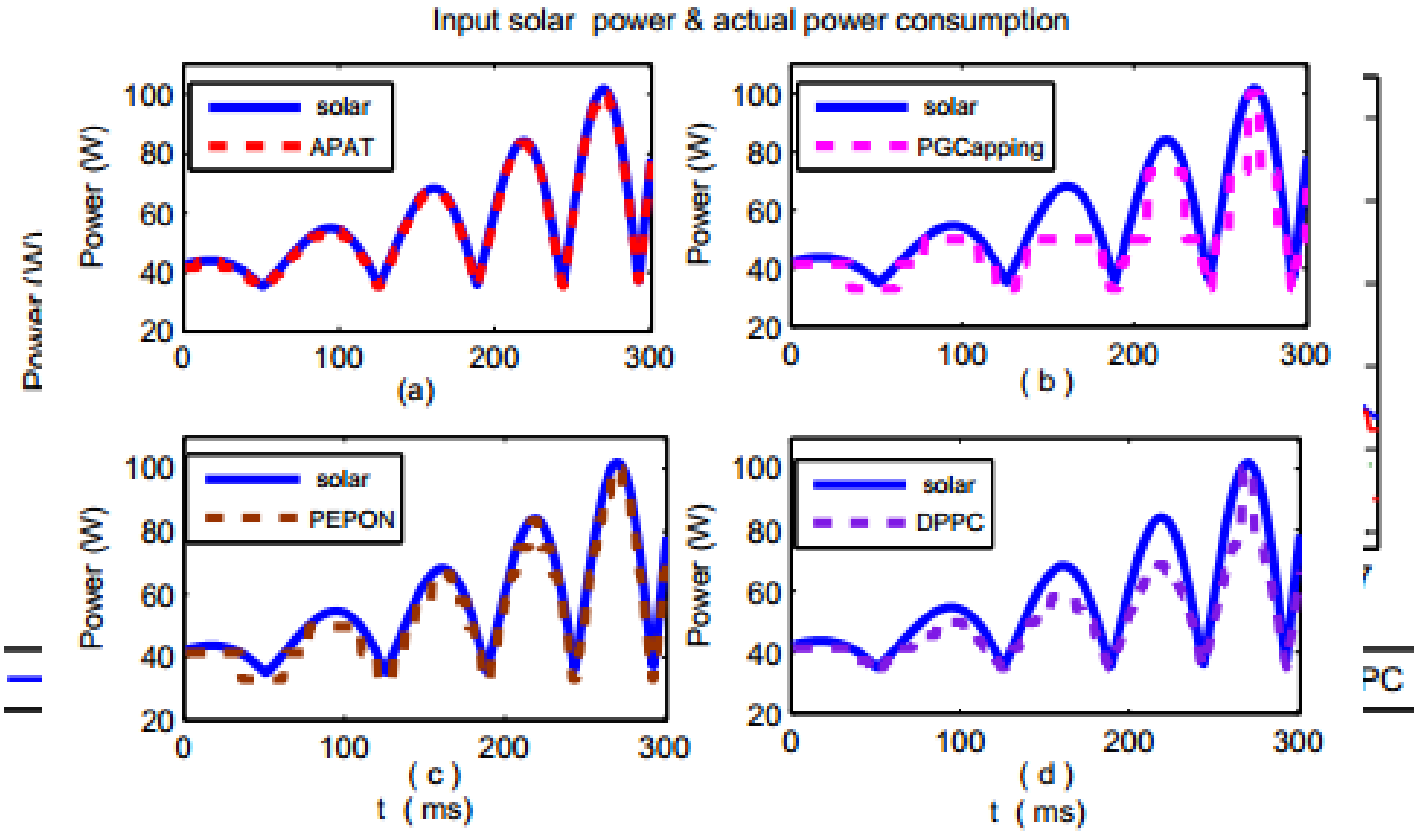
Evaluation

- Performance comparison
  - Reduces 26 %, 20%, 30 % execution time over PGCapping, PEPON, DPPC given power budget = 90W



Evaluation

- Run time adaptiveness to power budget variation
  - E ion)
    ir



  - The other three have high run time overhead and cannot match the rapid change in power budget

Evaluation

- Cost analysis
  - Area and power consumption of the DPN is 0.84 % and 0.27 % of the network-on-chip.
  - Running time: 2*n* cycles, where *n* is the network size
    - For a 64-core system, it's 128 cycles.
    - Other approaches: 1M or more cycles

Outline

- Background
- Related work
- Models and problem formulation
- The proposed algorithm
- Evaluation
- Conclusion

Conclusion

- The power allocation problem is formulated as a constraint optimization problem
- Dynamic programming is used to solve the problem. A HW circuit is used to accelerate the computation, with linear time complexity
- It can achieve better performance (lower execution) time over a power budget
- It has low running time and area overhead

THANKS
ANY QUESTIONS?

Backup slides

- Hmm, what's the type -2 DPN?
- Do I get more time ?

Conclusion
Extensions

- Clock gating instead of frequency scaling (submitted to DAC)
- Use auction models and support switching off to further reduce power consumption (accepted by DATE)
- Optimal power allocation and path selection for NoC

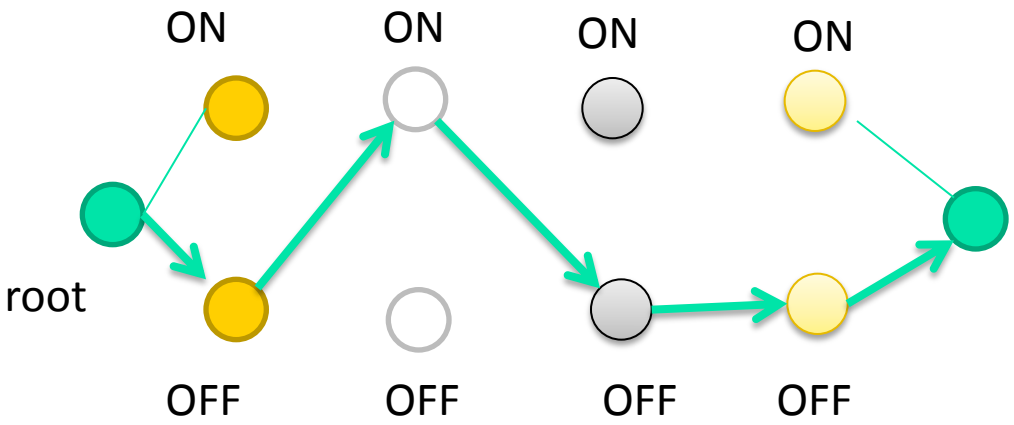
Type -2 DPN

- The shortcoming of the type-1 DPN
  - Storage O(NP), P is the power budget
  - What if P = 100 Watt?
  - Can we reduce the storage to O(NM), M is the allowable frequency levels #?

Type -2 DPN

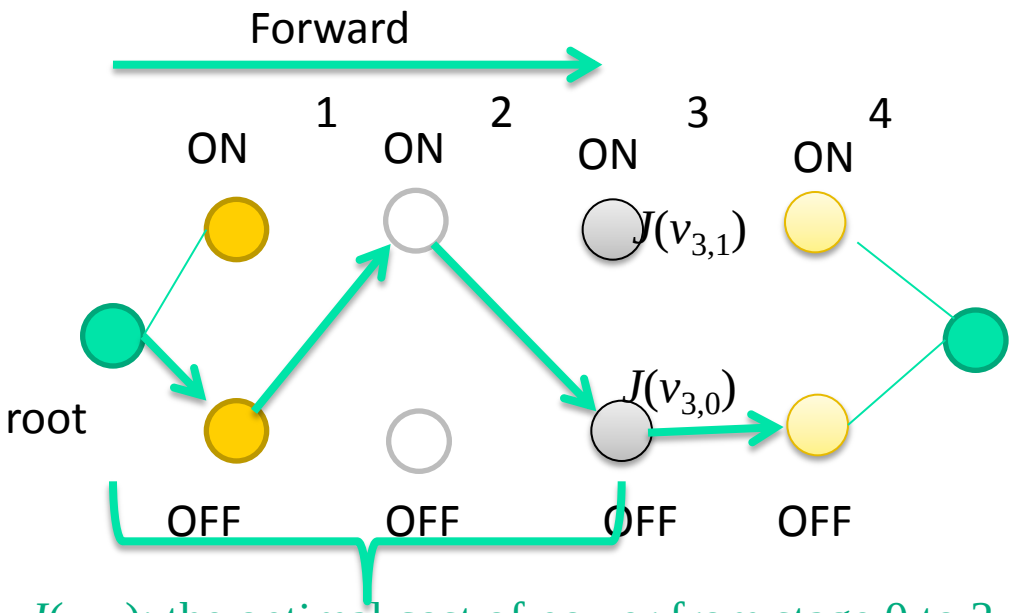
- How ?
  - Pass 1: an optimal path w.r.t. the power consumption
  - Pass 2: an optimal path w.r.t. the performance (value)

Can we simply put M vertices in each stage, instead of P vertices as in Type -1?



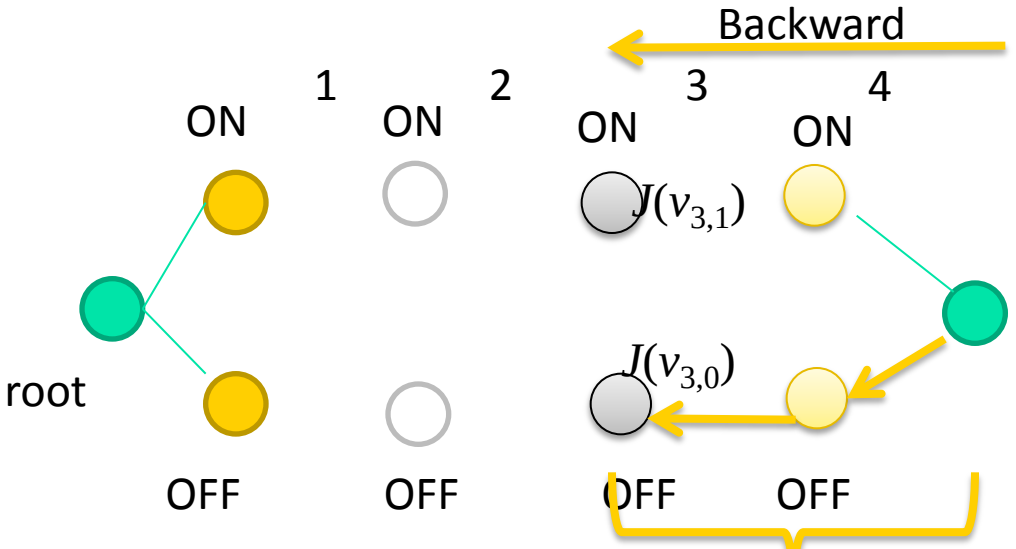
Type -2 DPN

- DPN:
  - V: $v_{i,j}$: a tile i with frequency set to be j.
  - E: each edge connects two vertices $v_{i,j}$ and $v_{i+1,l}$ in stage i+1
- Pass 1: additional DP value $J(v_{i,j})$: the optimal cost-to-go function w.r.t. power consumption from stage S to i. Forward traversing



Type -2 DPN

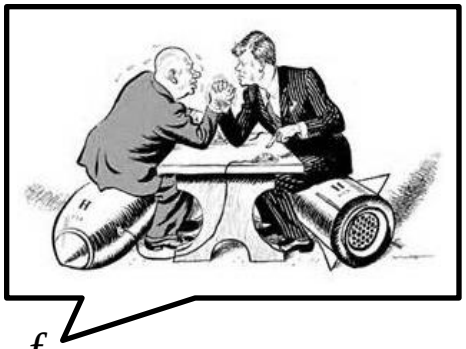
- Pass 2: an optimal path w.r.t. the performance (value)
  - $V(v_{i,j})$: the optimal performance from stage N back to i
  - $g(v_{i,j})$: the optimal power cost from stage N back to i
  - Constraint: $J(v_{i,j}) + b_i \cdot f_i + g(v_{i,j}) \le P$, $b_i f_i$: the power consumption of tile i
- Select the optimal edge among those that confirms to the constraint



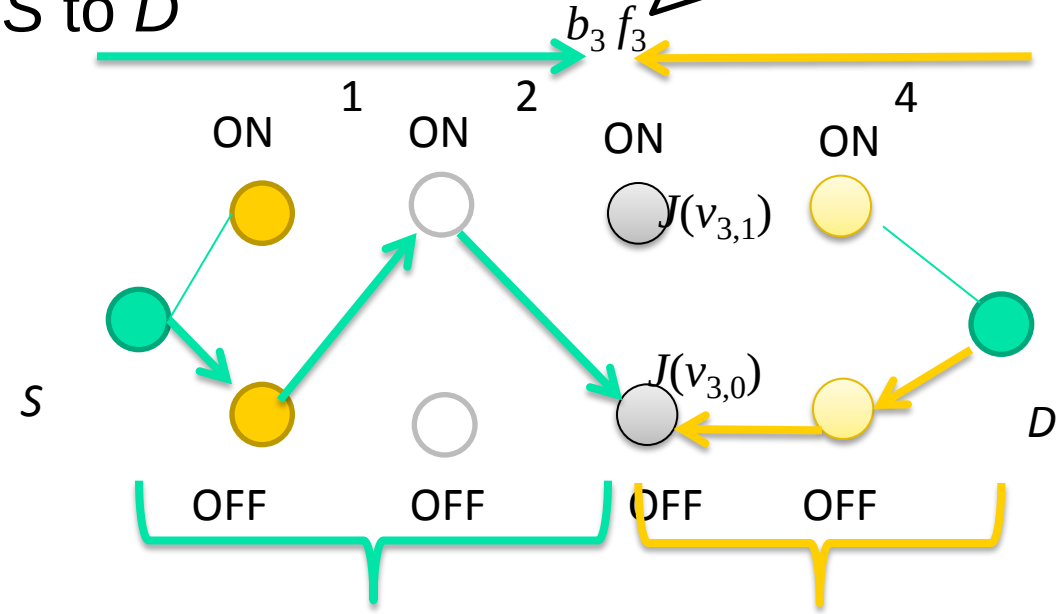
$V(v_{3,0})$: the optimal performance from stage 5 to 3
$g(v_{3,0})$: the optimal power cost from stage 5 to 3

Type -2 DPN

- What is the trick ?
  - Represent the power constraint of the full path from *S* to *D*



Where west meets east



$J(v_{3,0})$: the optimal cost of power from stage 0 to 3

$g(v_{3,0})$: the optimal cost of power from stage 5 to 3

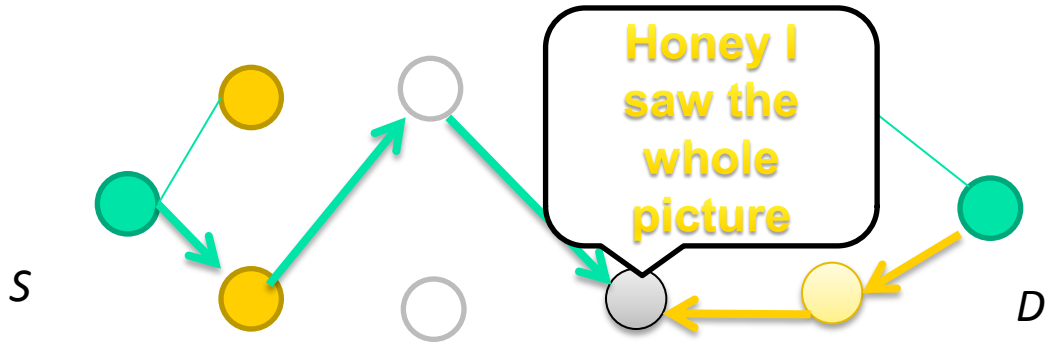
The cost of power of the path from *S* to *D* thru $v_{3,0}$ should be less than the total power budget *P*

$$J(v_{3,0}) + b_3 f_3 + g(v_{3,0}) \leq P$$

Type -2 DPN

- What's the trick ?
  - Two passes, the first pass finds the optimal power cost from stage *S* up to stage *i*, *J*(),
  - In the second pass, backward, finds the optimal power cost from stage D back to *i*, *g*()
  - So, *J*() + power cost of *i* + *g*() = the power cost of the full path from S to D thru *i*.
  - Now, we can find the optimal performance paths among the paths that confine to the above constraint.

Keep the full path constraint everywhere!



Type -2 DPN

- In general, if there are Q sets of constraints,
- Q forward passes with $J_q(v_{i,j})$ (in parallel)
- A backward pass, with $g_q(v_{i,j})$
- Even constraints of higher order, e.g., $b_i f_i^2$, $c_i f_i^3$
- $0 \leq q \leq Q$

$$\max \; Perf = \sum_{i=1}^{N} a_i \cdot f_i$$

$$subject\ \text{to} \sum_{i=1}^{N} b_i \cdot f_i \leq P$$

$$\vdots$$

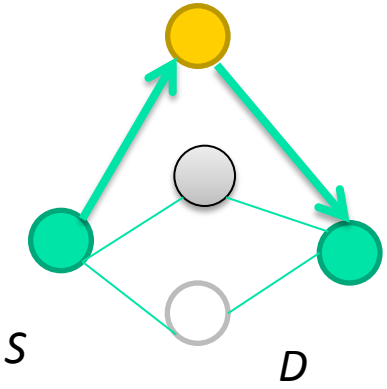
$$\sum_{i=1}^{N} z_i \cdot f_i \leq Z$$

Q constraints

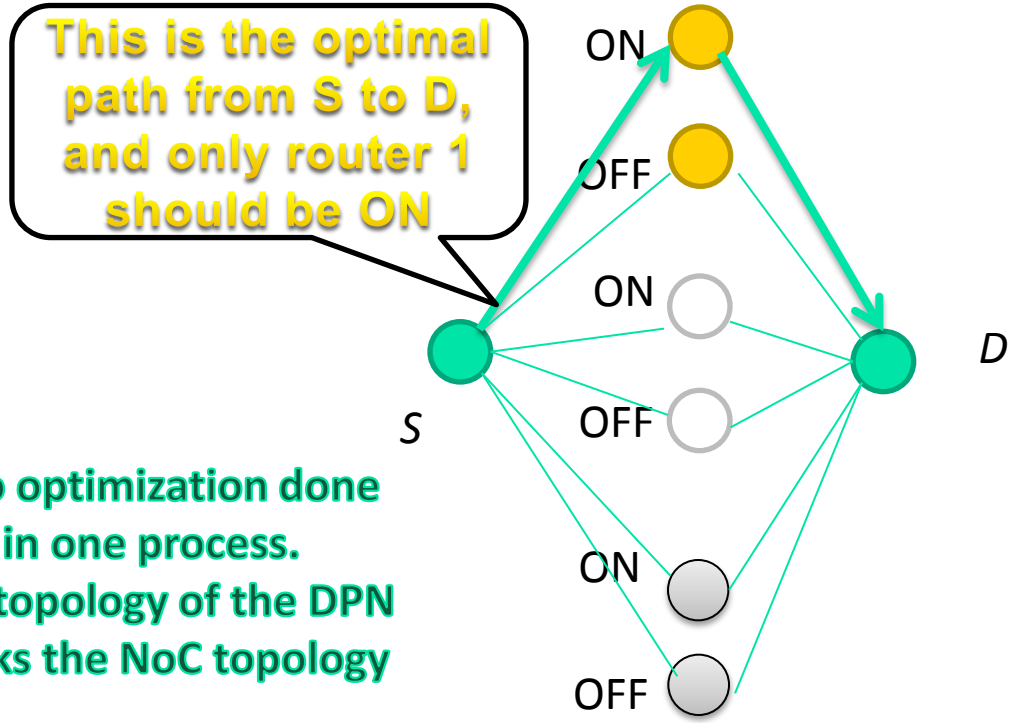
$$\text{for each } f_i \in \{F_1, \ldots, F_M\}$$

Type -2 DPN

- An application
  - Optimal decision for both router power allocation (by frequency scaling or ON/OFF) **AND** routing path selection simultaneously



Given a power budget, allocate to the routers optimally AND find the optimal path



Two optimization done in one process. The topology of the DPN tracks the NoC topology