Systematic Architecture Exploration based on Optimistic Cycle Estimation for Low Energy Embedded Processors

Ittetsu Taniguchi*, Murali Jayapala**, Praveen Raghavan**, Francky Catthoor**, Keishi Sakanushi*, Yoshinori Takeuchi*, and Masaharu Imai*

*Graduate School of IST, Osaka University, Japan

**Nomadic Embedded Systems, IMEC vzw., Belgium

Outline

- Introduction
- Reconfigurable AGU Model
- Architecture Exploration Problem
- Feature of Solution Space
- Optimistic Cycle Estimation
- Experimental Results
- Conclusion and Future Work

Introduction

- High performance and low energy consumption for nomadic embedded systems
  - Memory access consumes “time” and “energy”
    - Especially for scratch pad memory (SPM)
- High level compiler optimization
  - Removing unnecessary memory access
  - Improving memory access locality
- Next overhead: Address calculation
  - Address generation unit (AGU) for complex address calculation

Architecture exploration for address generation unit

Motivation

```
for (x=1; x<=N-2; ++x) {
  for (y=1; y<=M-2; ++y) {
    for (k=-1; k<=1; ++k) {
      A[x][y]+=B[x+k][y]*C[abs(k)];
      A[x][y] /= tot;
    }
  }
}
```

Before high level compiler optimization

More than 60% of calculations are “Address Calculation”

```
for (y=0; y<=M+2; ++y) {
  for (x=0; x<=N+2; ++x) {
    if (x>=0 && x<N && y>=1
                          && y<=M-2) {
      D[x%3] = B[(y*N+x)%8704+
        (y*N+x)%8704*16384+7680];
    }
    if (x-1>=1 && x-1<=N-2
                  && y>=1 && y<=M-2) {
      for (k=-1; k<=1; ++k) {
        A[x][y] += D[(x-1+k)%3]
                          *C[abs(k)];
      }
    }
    A[x][y] /= tot;
  }
}
```

After high level compiler optimization

Related Work for Address Generation Unit

- Mathew et al., 2004.
  - Address generation and loop acceleration for VLIW processors
  - Limitation:
    Only 2D affine address equation: A[**i*P+Q**][**j*R+S**]

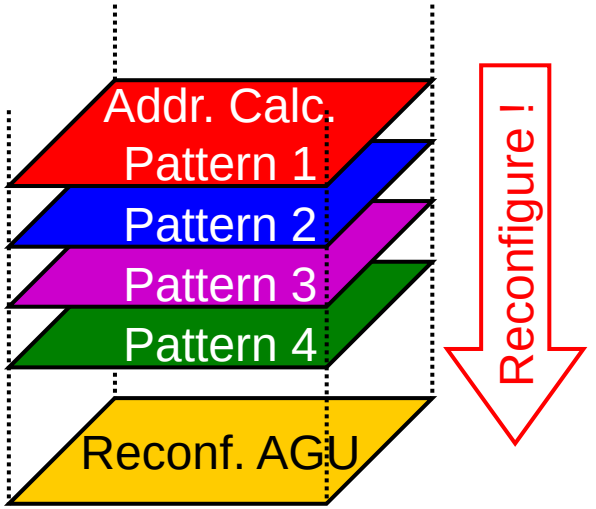
```
…
for (i=0; i<N; i++) {
    for (i=0; i<N; i++) {
        tmp[i] = Images[Macroblock*(ImageMCRef-1)*(IMG_ROWS+IMG_ROWS/2)*IMG_COLS
                        + IMG_ROWS*IMG_COLS + (v*M + j)*IMG_COLS/2 + (h*N+i)];
        …
    }
}
```

Address calculation after strong optimization!!!

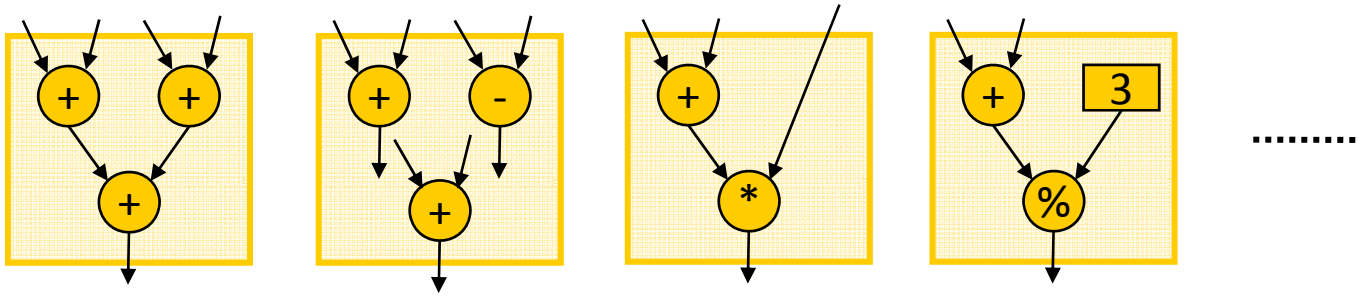
More general address generation unit for VLIW processors

Reconfigurable AGU

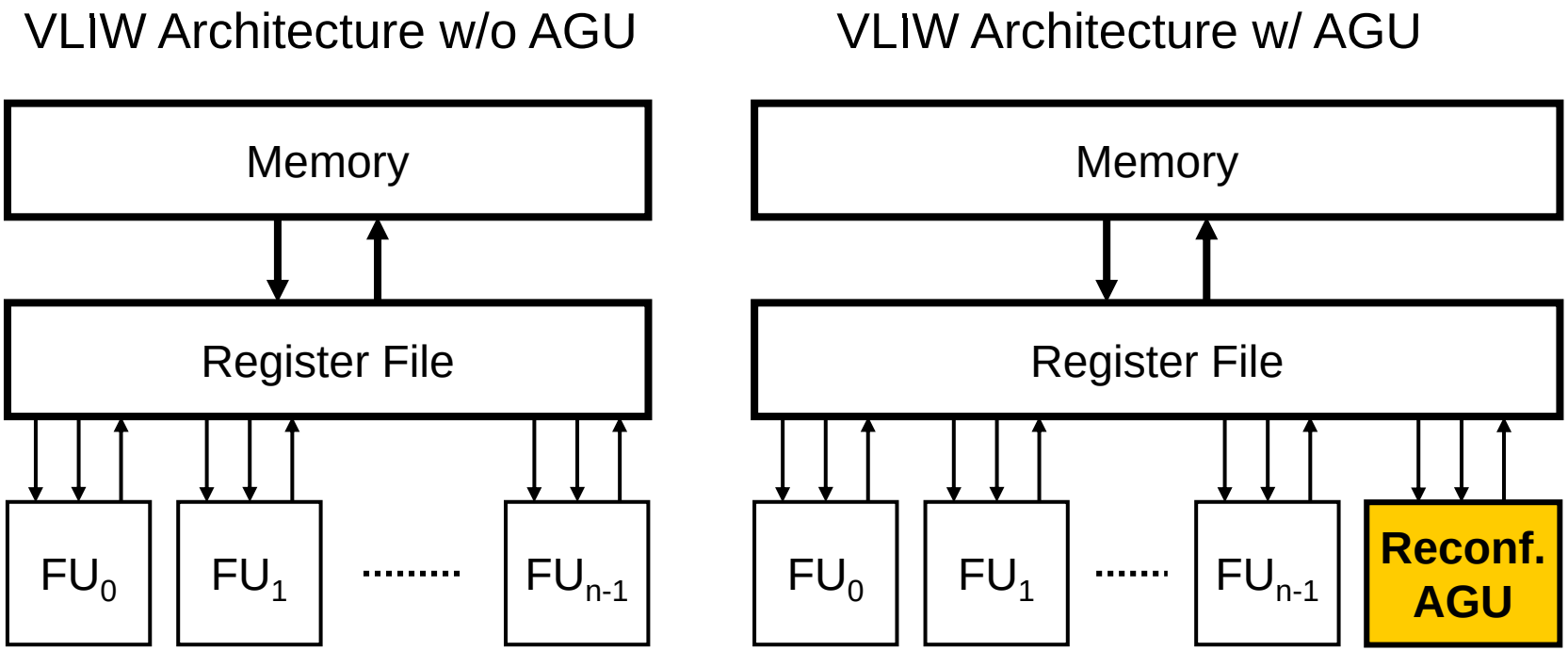
- Address calculation can be divided into some patterns !!!
- Reconfigurable AGU realizes one pattern on it
- Effective calculation by changing pattern
  - Reconfigurable AGU changes its function dynamically



Example: Address calculation patterns

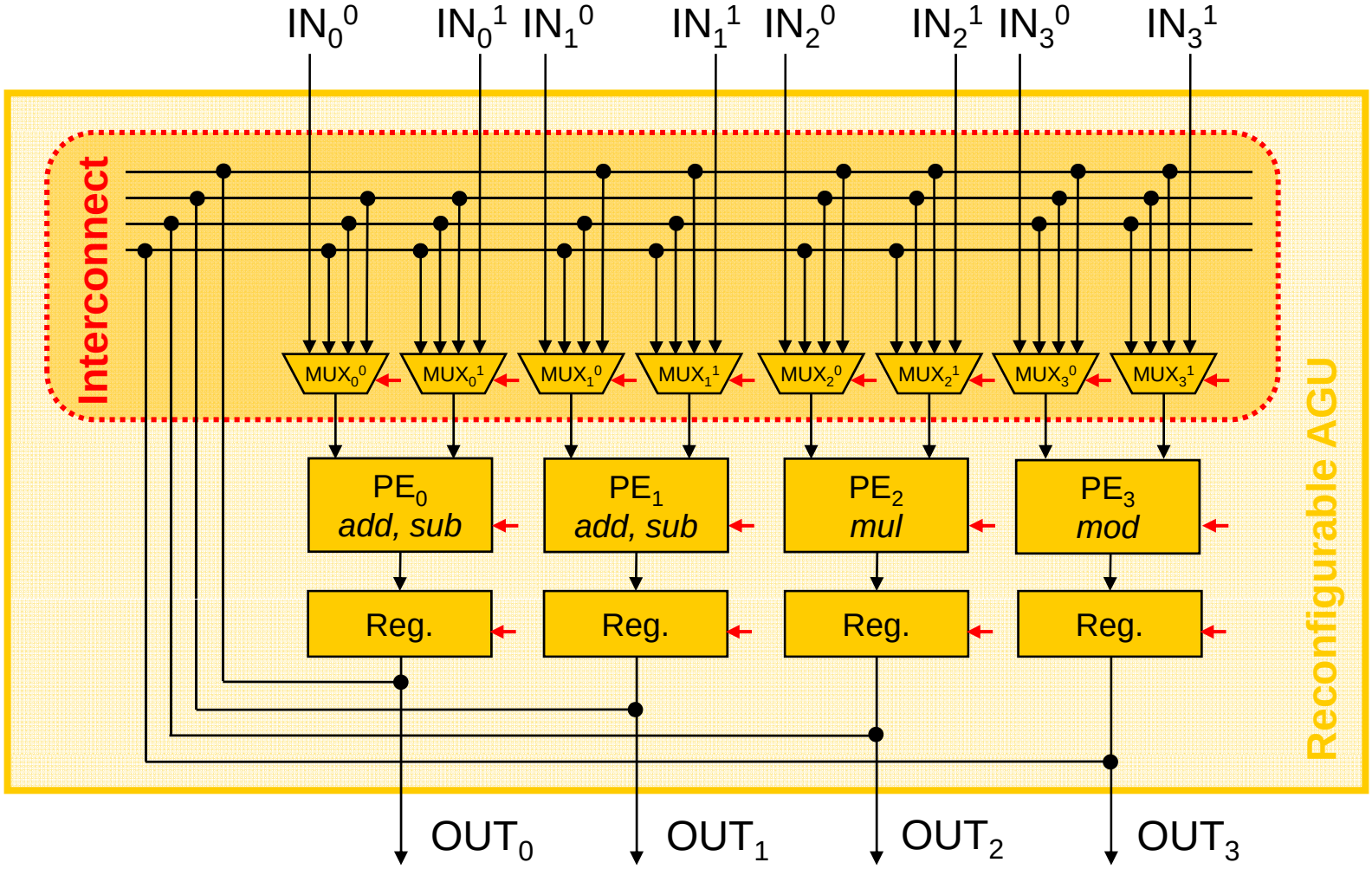


Reconfigurable AGU from Architectural View



AGU as dedicated functional unit for address calculation

Reconfigurable AGU Model



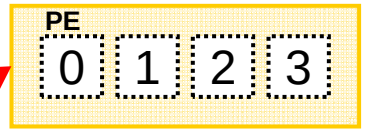
Parameters: #PE, Instruction assignment for each PE

➔ How to evaluate each AGU?

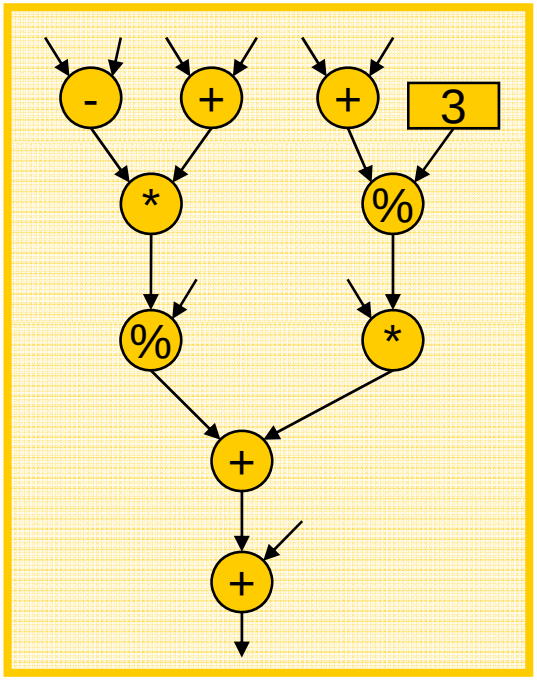
AGU Mapping as Performance Evaluation

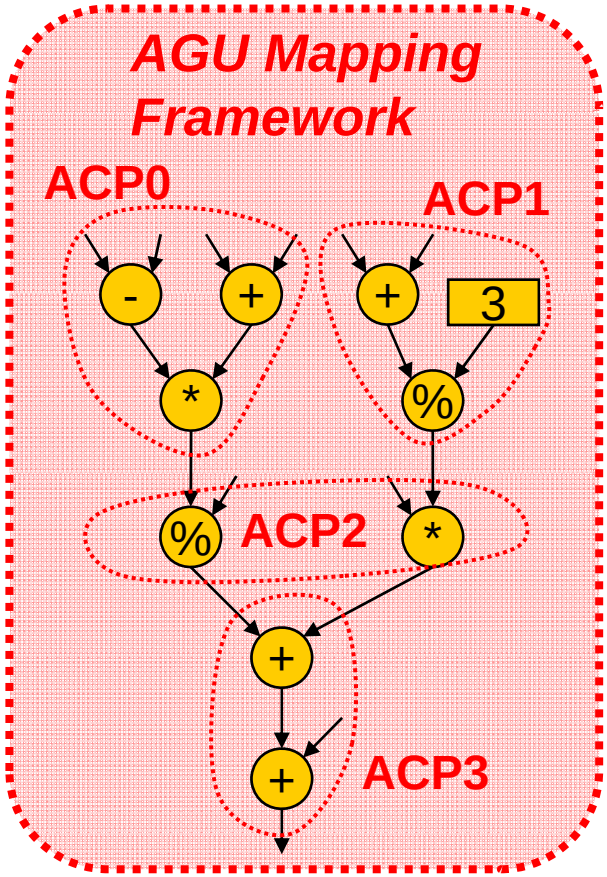
Target Reconfigurable AGU Spec.

```
#PE=4;
PE0={('+',1cycle),('-',1cycle)};
PE1={('+',1cycle),('-',1cycle)};
PE2={('*',3cycle)};
PE3={('%',30cycle)};
```

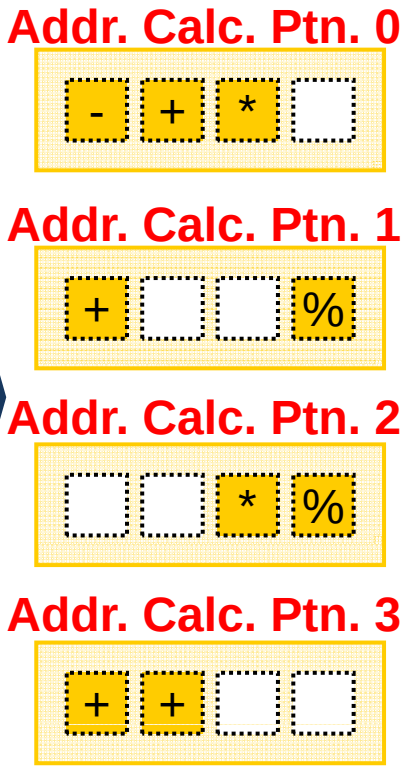


Address Calculation DFG





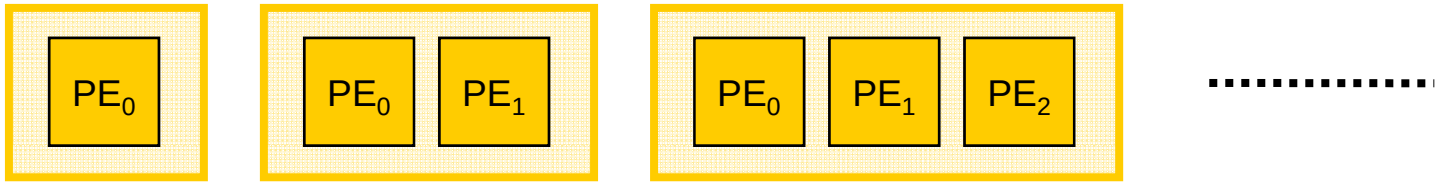
Mapping Results and Exec. Cycle



➔ **How to explore the best AGU effectively???**

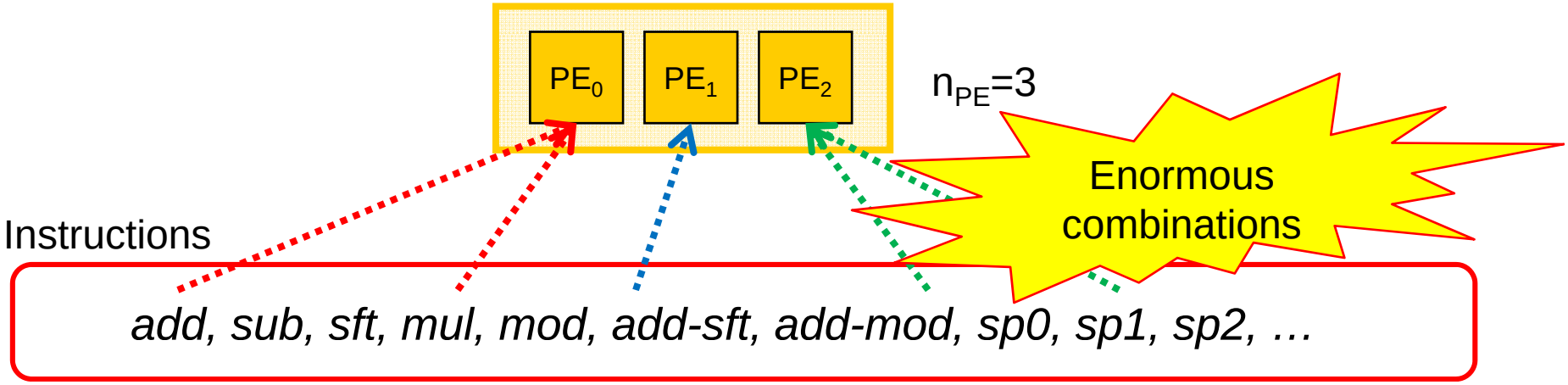
Variety of Reconfigurable AGU Model

- Number of PEs



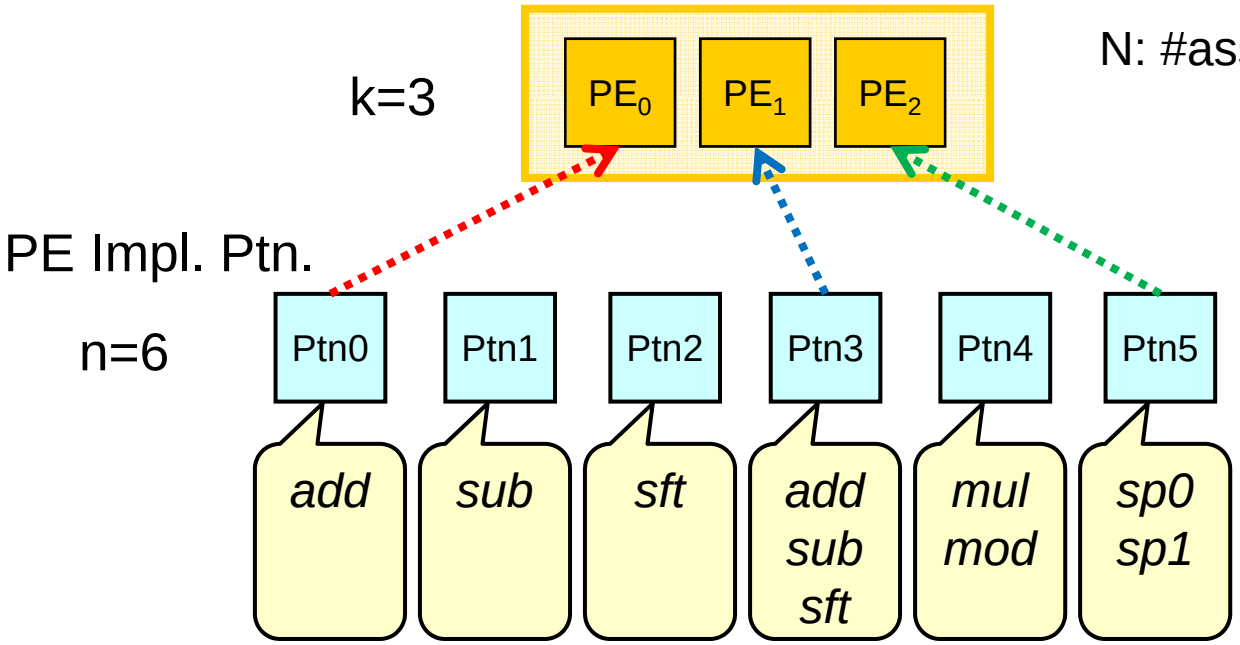
n_{PE}=1 n_{PE}=2 n_{PE}=3

- Instruction assignment for each PE
  - A lot of instructions including special instructions



Assignment of PE Implementation Pattern

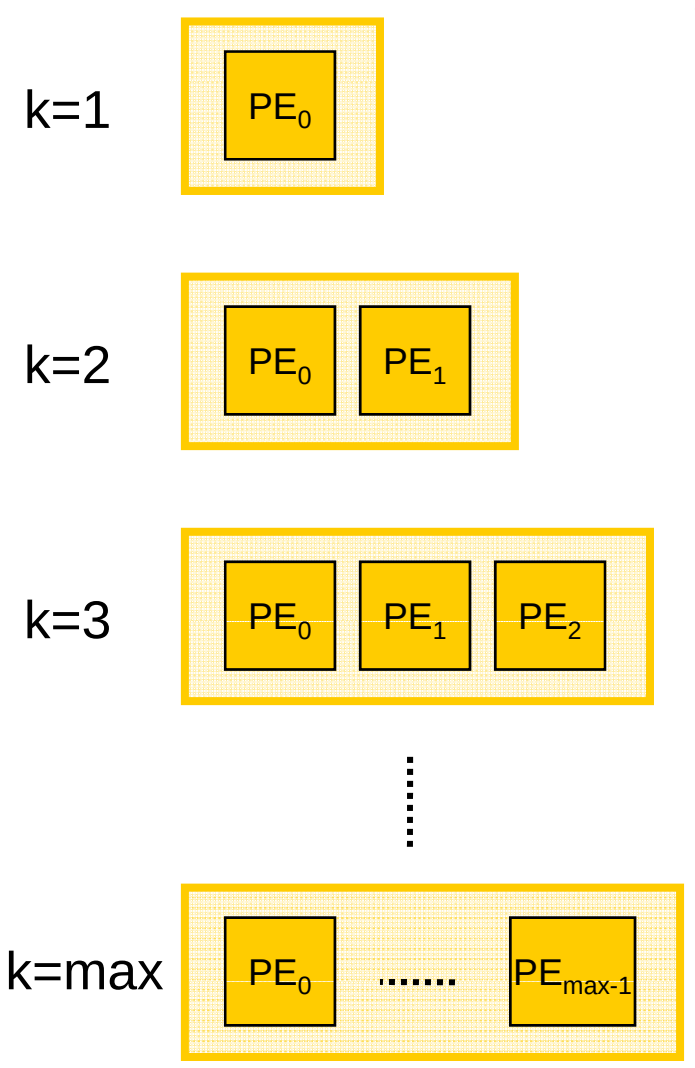
- NOT assign instructions directly, but assign set of instructions to each PE
  - Set of instructions = PE implementation pattern



N: #assignment of PE Impl. Ptn.

$$N = {}_nH_k = \frac{(n+k-1)!}{k!(n-1)!}$$

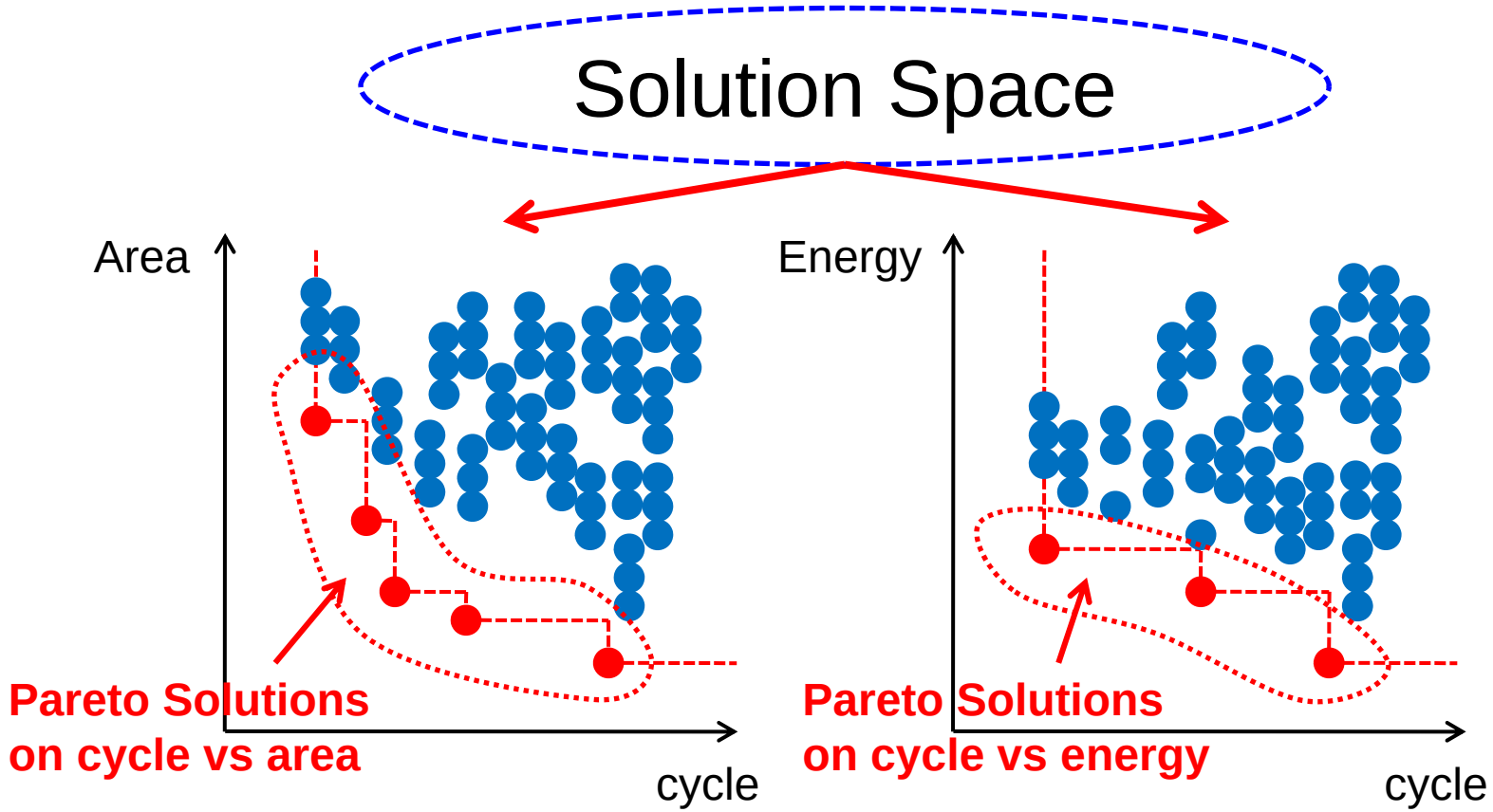
Solution Space



$$|S| = \sum_{k=1}^{\max} {}_n H_k = \sum_{k=1}^{\max} \frac{(n+k-1)!}{k!(n-1)!}$$

S: Solution space
n: #PE Implementation Pattern
max: Limitation of PE

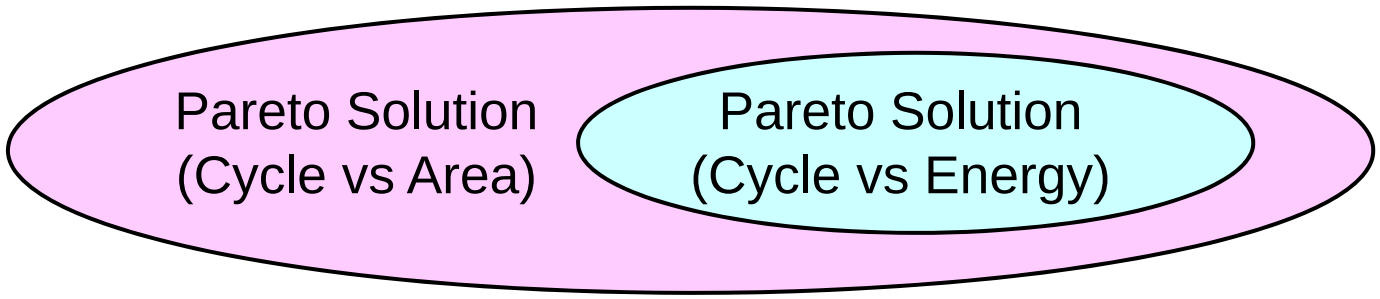
Architecture Exploration Problem



***Architecture Exploration Problem*:**
For given application, to find Pareto solutions from supposed solution space.

Set of Cycle vs Area Pareto solutions gives you everything!

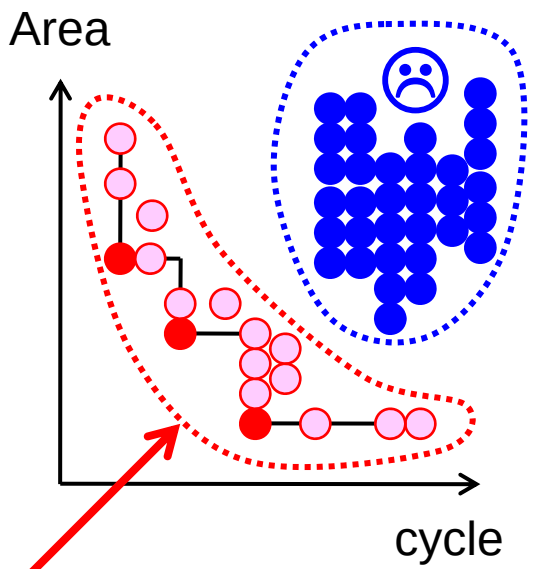
- Set of cycle vs energy Pareto solutions becomes a subset of cycle vs area Pareto solutions under the given energy model and following assumptions:
  - Leakage energy is non-negligible
  - Clock & power gating schemes are not applied
  - Voltage and frequency remain constant



Concentrate on only cycle vs area Pareto solutions, and you will get both!!!

How to obtain cycle vs area Pareto solutions fast?

- Evaluate only promising solutions
  - Performance evaluation is time consuming part in architecture exploration kernel
- Rough cycle estimation to find promising solutions on cycle vs area
  - Area estimation is easy !



New metric to find promising solutions on cycle vs area

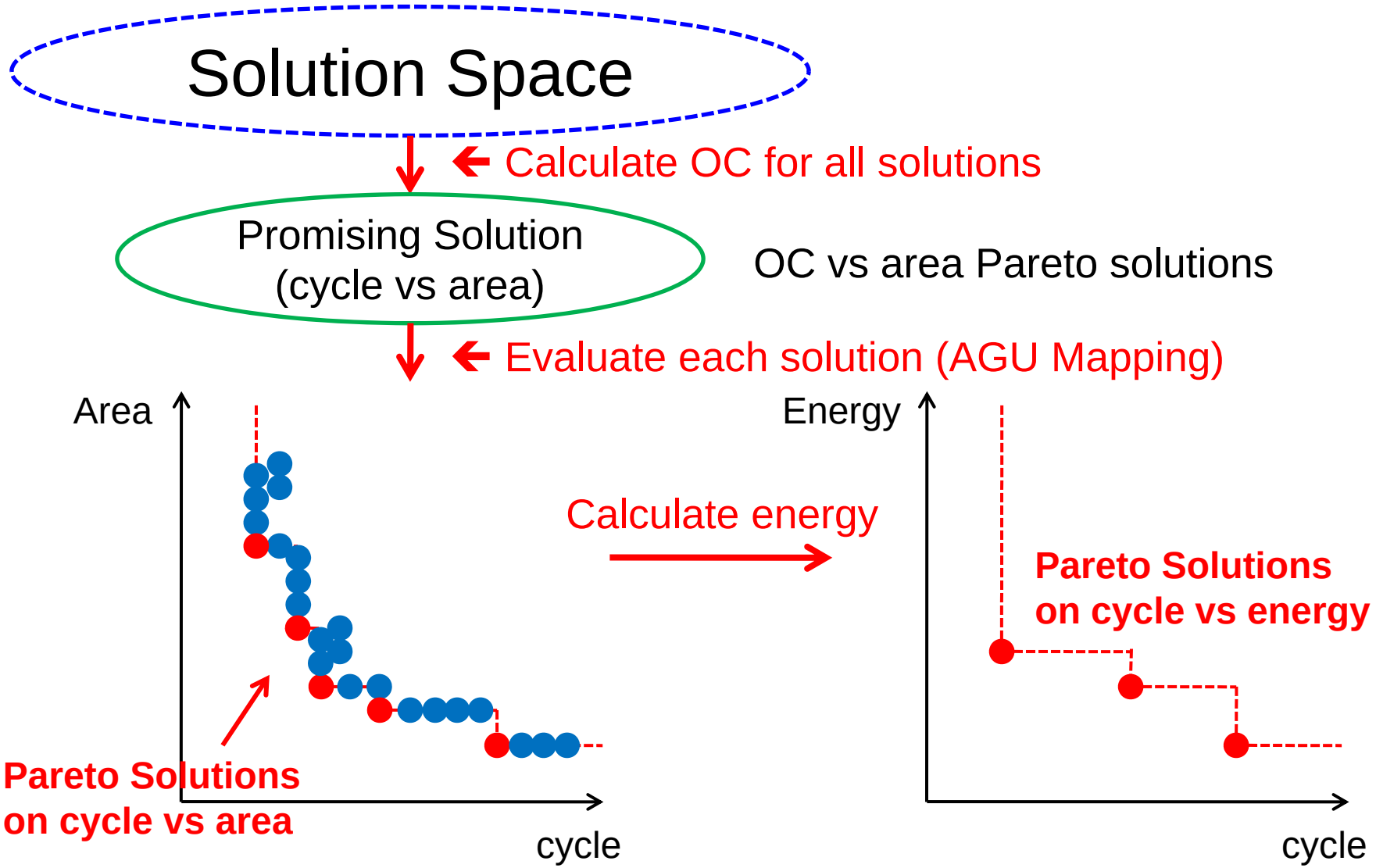
Optimistic Cycle (OC) as rough cycle estimation

- I: a set of instructions
- n(inst): #instruction inst in a given application
- latency(inst): latency of instruction inst
- para(inst): #PE which can execute instruction inst

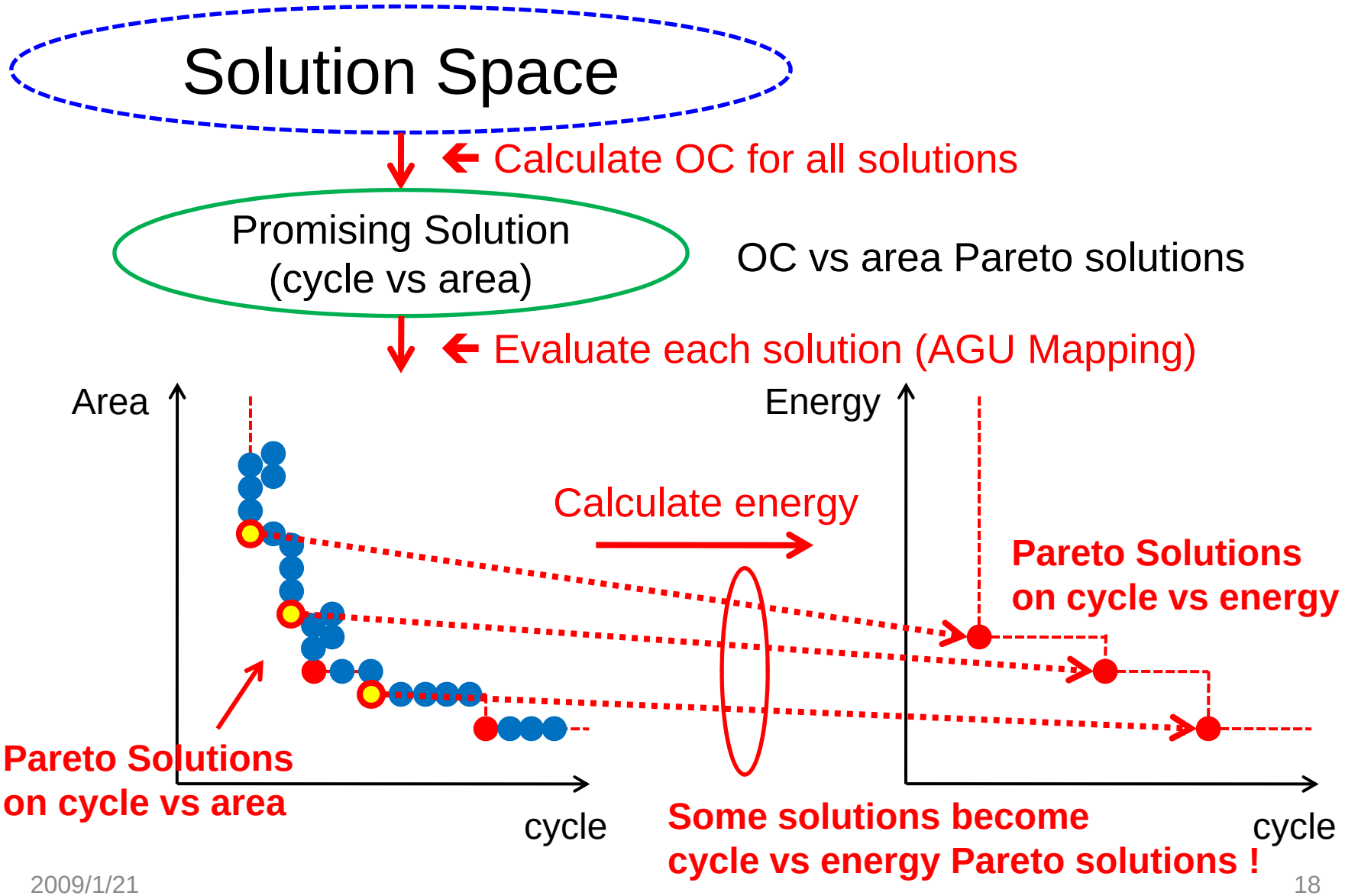
$$OC = \sum_{\forall inst \in I} \frac{n(inst) \cdot latency(inst)}{para(inst)}$$

Cycle estimation w/o any dependency for each instruction

Algorithm


Solution Space
Calculate OC for all solutions
Promising Solution
(cycle vs area)
OC vs area Pareto solutions
Evaluate each solution (AGU Mapping)
Area
cycle
Pareto Solutions
on cycle vs area
Calculate energy
Energy
cycle
Pareto Solutions
on cycle vs energy

Algorithm


Solution Space
Calculate OC for all solutions
Promising Solution
(cycle vs area)
OC vs area Pareto solutions
Evaluate each solution (AGU Mapping)
Area
cycle
Pareto Solutions
on cycle vs area
Calculate energy
Energy
cycle
Pareto Solutions
on cycle vs energy
Some solutions become
cycle vs energy Pareto solutions !

Comparison of Exploration Time

On PentiumD 2.8GHz, 2GB Mem.

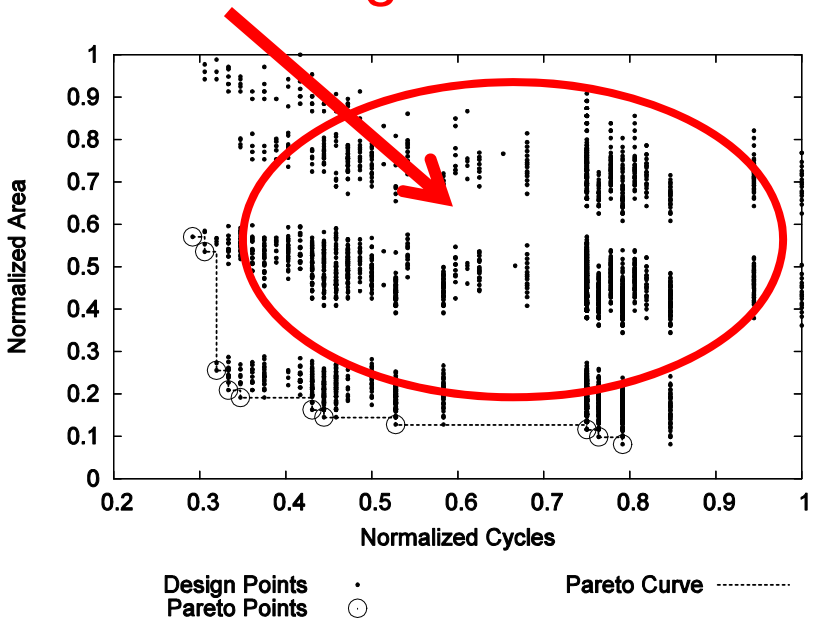
	#Solutions applied SA base evaluation Exhaustive / OC	Exploration Time [sec] Exhaustive / OC
Handmade	1909 / 38	12384 / 236
Cavity	2387 / 35	10341 / 157
Motion	1120 / 12	14790 / 136
QSDPCM	3586 / 23	46356 / 282

164 times faster !!!

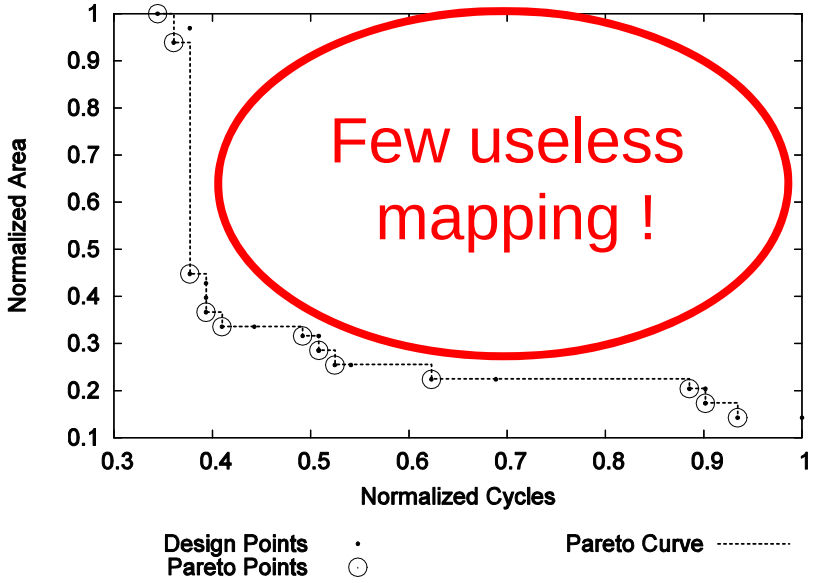
Exploration time is drastically reduced because of less mapped solutions !!!

Why so fast ? – Exhaustive vs Systematic Cycle vs Area Tradeoff -- QSDPCM

Time consuming useless evaluation !



Exhaustive Search



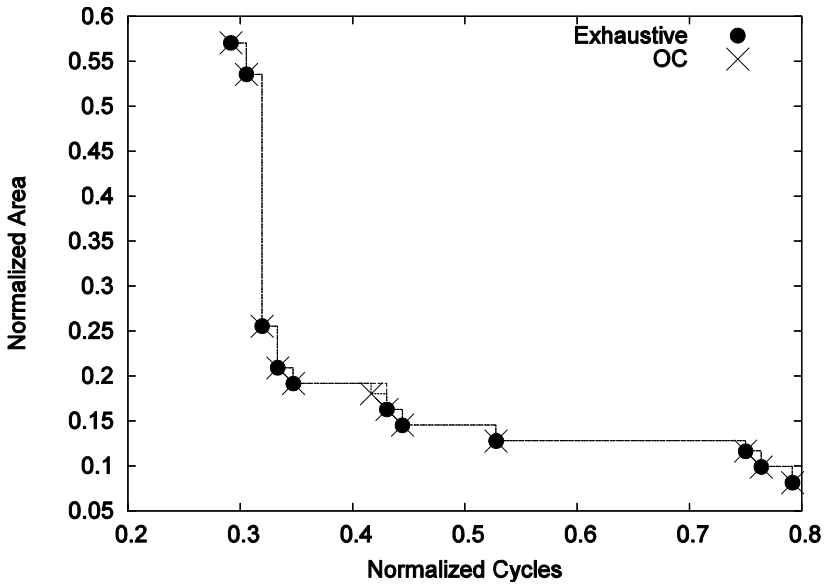
Search by Proposed Method

ONLY promising solutions are mapped!!!

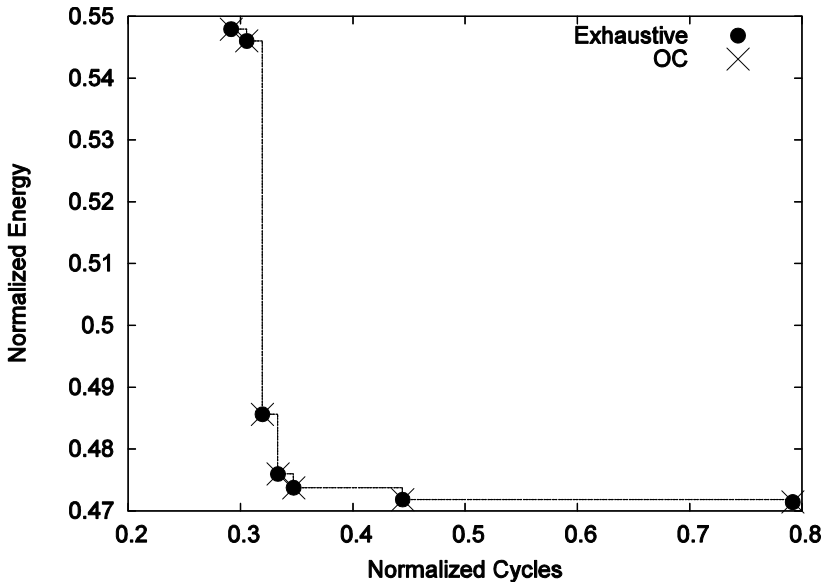
Comparison of Pareto Curves: QSDPCM

5 min !!!

Cycle vs Area



Cycle vs Energy

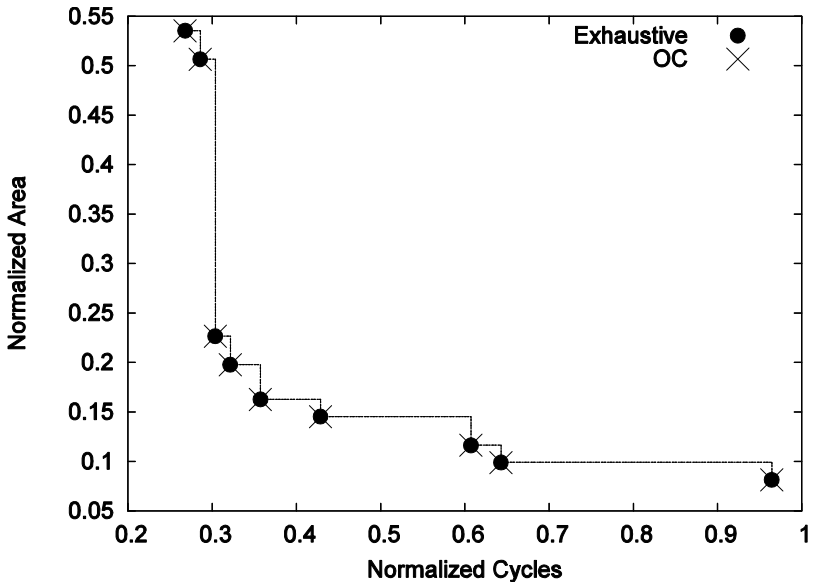


Pareto curves are completely overlapped !!!

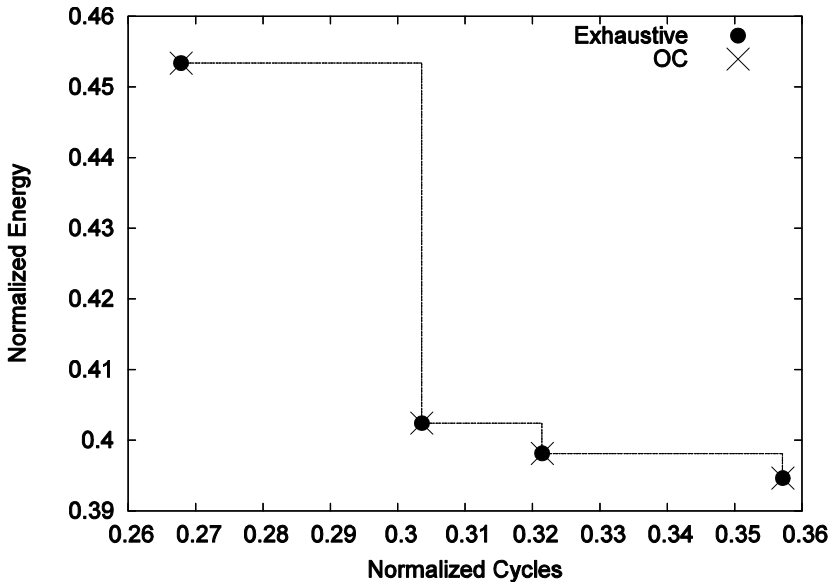
Comparison of Pareto Curves: Motion

2.5 min !!!

Cycle vs Area



Cycle vs Energy



Pareto curves are completely overlapped !!!

Conclusion and Future Work

- Conclusion
  - Systematic architecture exploration for reconfigurable AGU
    - Feature of solution space
    - Optimistic cycle estimation
  - 164 times faster architecture exploration
- Future Work
  - More accurate energy estimation
    - Assume clock and power gating scheme
  - Architecture exploration from exploded solution space