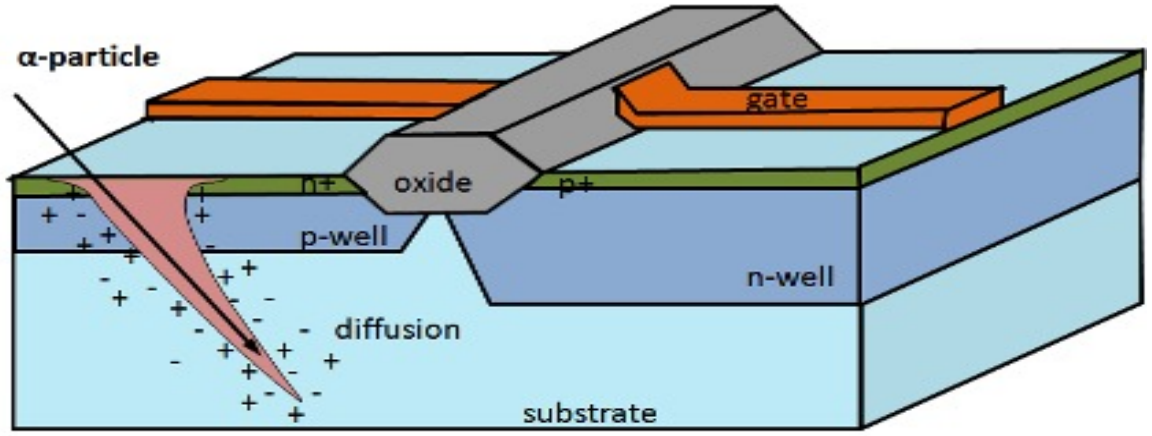
Peppa-X: Finding Program Test Inputs to Bound Silent Data Corruption Vulnerability in HPC Applications

Md Hasanur Rahman, Aabid Shamji, Shengjiang Guo, Guanpeng Li

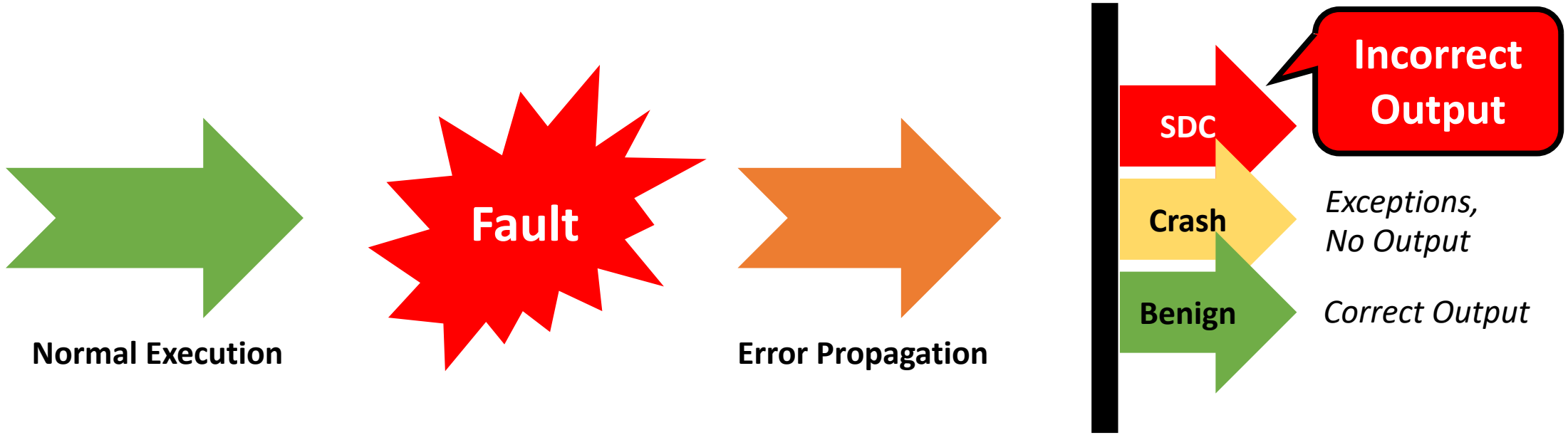
Motivation: Soft Error



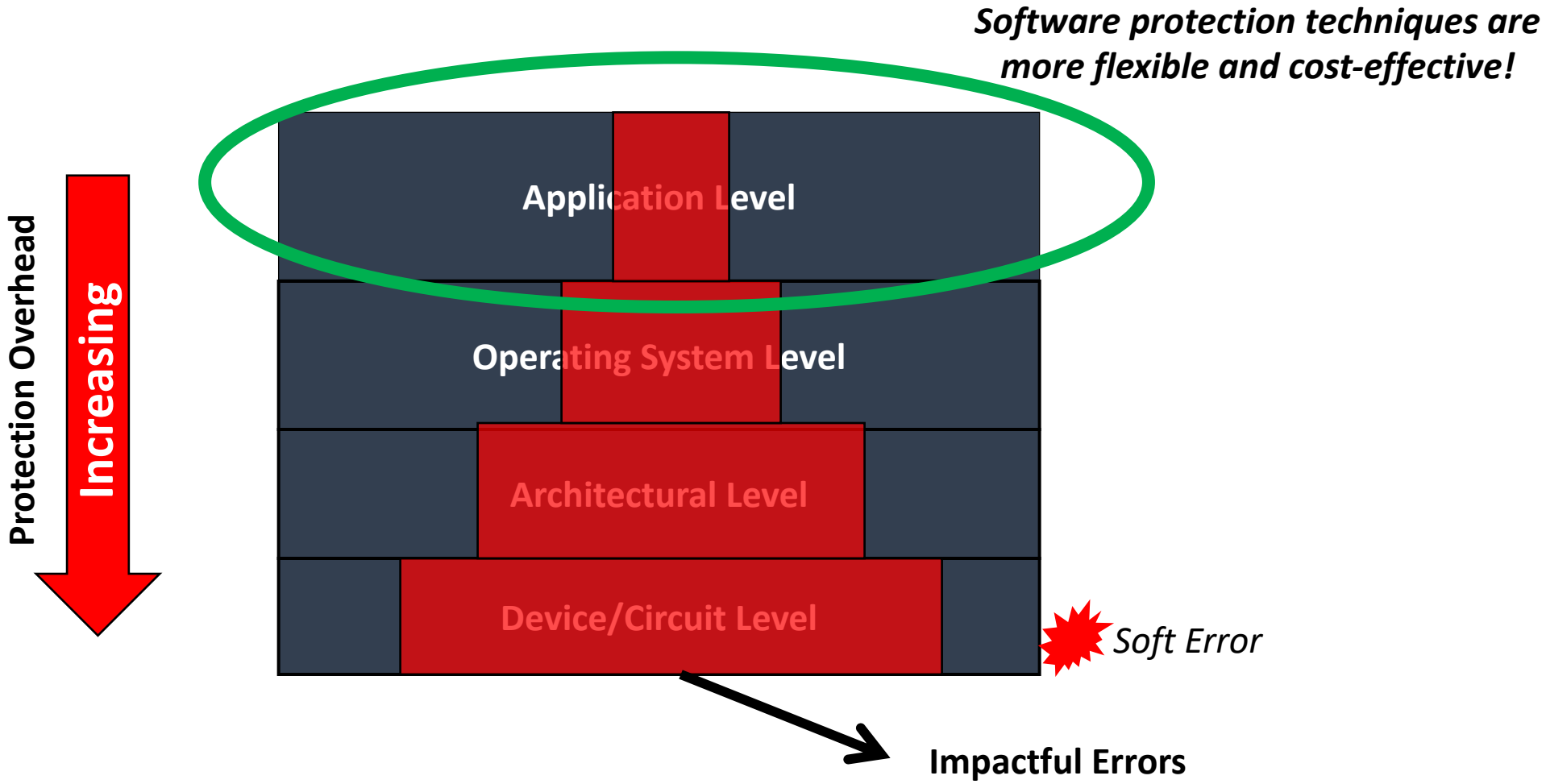
Soft errors becoming more common in processors!

S S

Silent Data Corruption (SDC)



Software Solutions



Existing Works

- Pruning FI space
  - Us
  - e.g
- Heuristics-based error estimation
  - e.g., [DSN'12]
- Syste
  - e.g

Problem

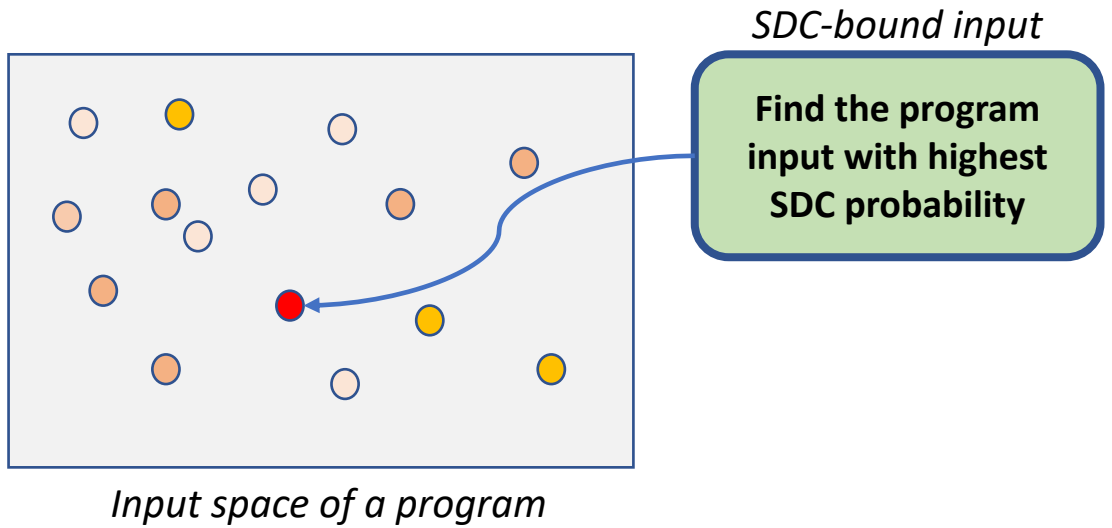
- They all focus on single input of a program!
  - Default reference input

Challenge

State space explosion in the program input space

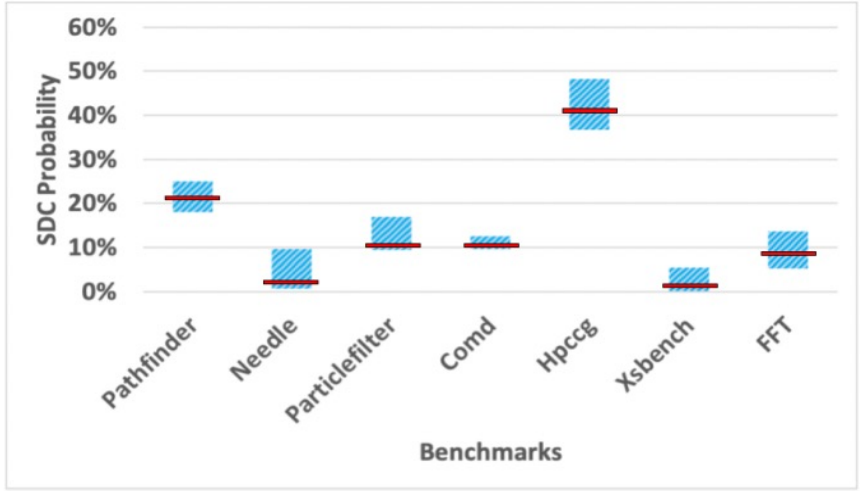
Our Goal

- Bound SDC probability of a program across multiple inputs
  - High accuracy
  - High efficiency
  - In an automated way



Initial Study

- SDC probability of a program across multiple inputs
  - Vary in a large range
  - Range up to 11.45% for Hpccg
- Code covera
  - Research
  - No correla
- SDC probabilities of individual instructions
  - Though vary, the ranking is stable



Range of Overall SDC prob. Across Multiple Inputs
f default reference input

Insight

Some instructions are always vulnerable to SDCs regardless of input changes

	Hpccg	Xsbench	FFT
	0.00	0.38	0.00

rage and Program SDC Prob. across Multiple Inputs

Pathfinder	Needle	Particlefilter	CoMD	Hpccg	Xsbench	FFT
0.92	0.79	0.90	0.90	0.96	0.59	0.77

Correlation between Rankings of Per-Instruction SDC Prob. across Multiple Inputs

Our Approach: Overview

- Find th **Challenge #1** t that executes the

  vulne

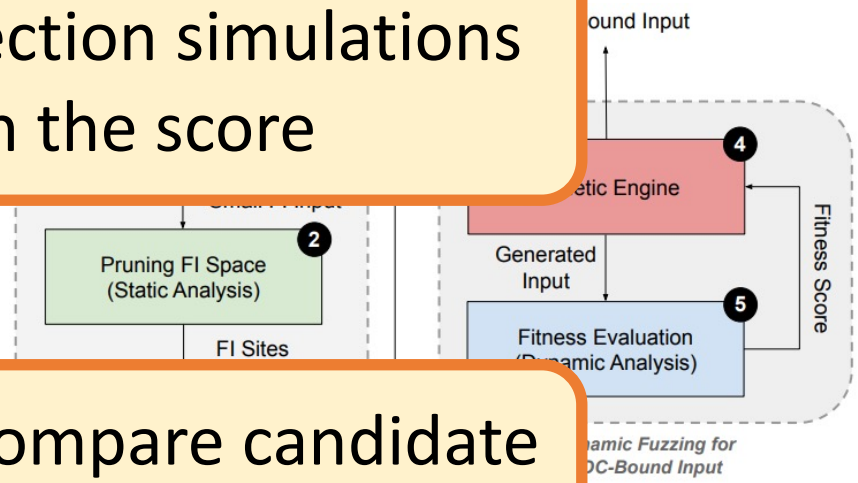
  thus obtaining higher SDC probability

  Challenge #2

  - Ass
  - Use
  - Find the SDC-bound input!

Need to conduct extensive fault injection simulations for every instruction to assign the score

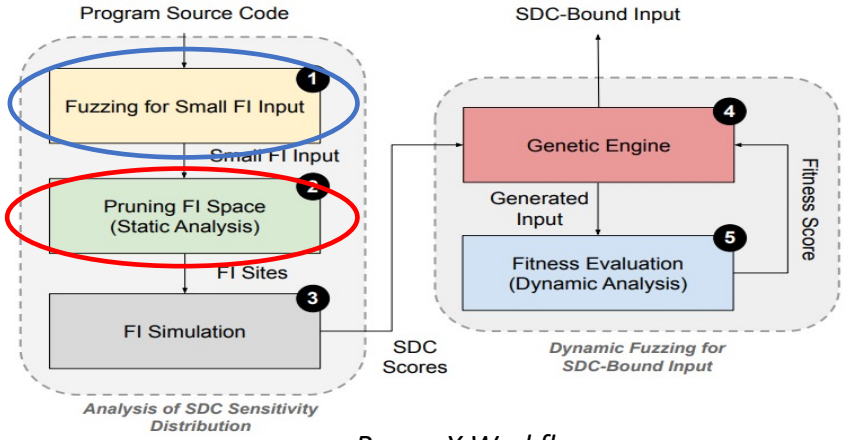
Need to know SDC probability to compare candidate inputs to make optimization decisions by GA



kflow

Challenge #1: Deriving SDC Score

- Fuzz for small FI input
  - Small workload yet equal code coverage
  - FI simulations becomes fast
- Avoid FI simulations for all instructions
  - Reduce FI space by applying pruning
    - Use static dataflow dependency analysis
- Static dataflow dependency analysis
  - Instructions within same static data dependency shows similar SDC probabilities
    - e.g., avg FI space reduced to 49%



Peppa-X Workflow

BB167

```
...
%168 = load i32* %k...          ; ID1562; SDC: 0.8%
%169 = add nsw i32 %168, 1...   ; ID1563; SDC: 0.4%
...
%171 = icmp eq i32 %169...      ; ID1565; SDC: 100.0%
...
```

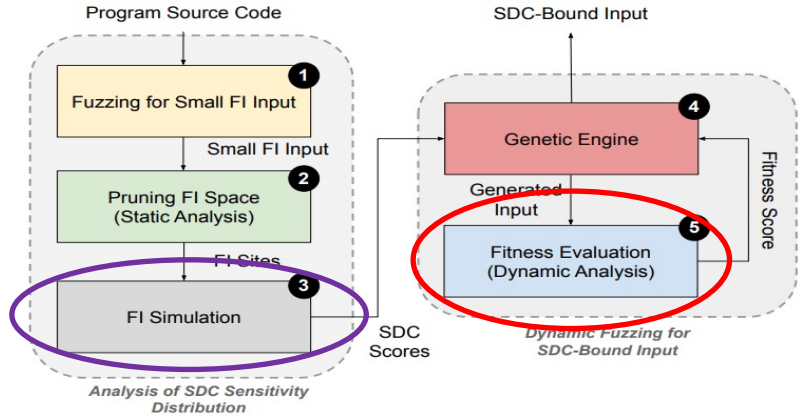
Code Example of Pruning FI space in CoMD

Pathfinder	Needle	Particlefilter	CoMD	Hpccg	Xsbench	FFT	Avg
25.49%	51.40%	46.35%	58.44%	58.69%	49.22%	55.64%	49.32%

FI-space pruning ratios

Challenge #2: Fitness Function in Fuzzing

- Avoid repetitive statistical FIs to rank each generated candidate input by GA
  - Assign score to each static instruction
    - Conduct FI simulations to only those instructions from pruned FI space.
  - Accumulate scores of executed instructions during program execution
    - An estimate for SDC probability of a program input



Peppa-X Workflow

Experimental Setup

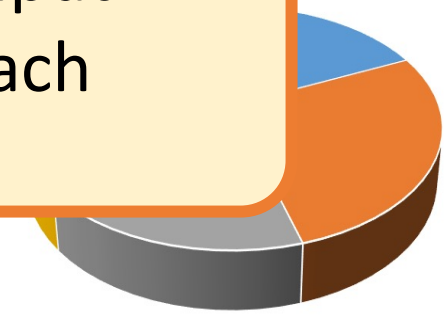
- Fault M
  - Fa
- LLFI –
  - Ra
  - Accurate to simulate soft error and evaluate SDCs [DSN'17]
  - 1000 s to evaluate SDC for each given input
- Benc
  - 7 c

Baseline

- Generate random input to find SDC-bound input
- Inject faults to calculate SDC probability of each random input

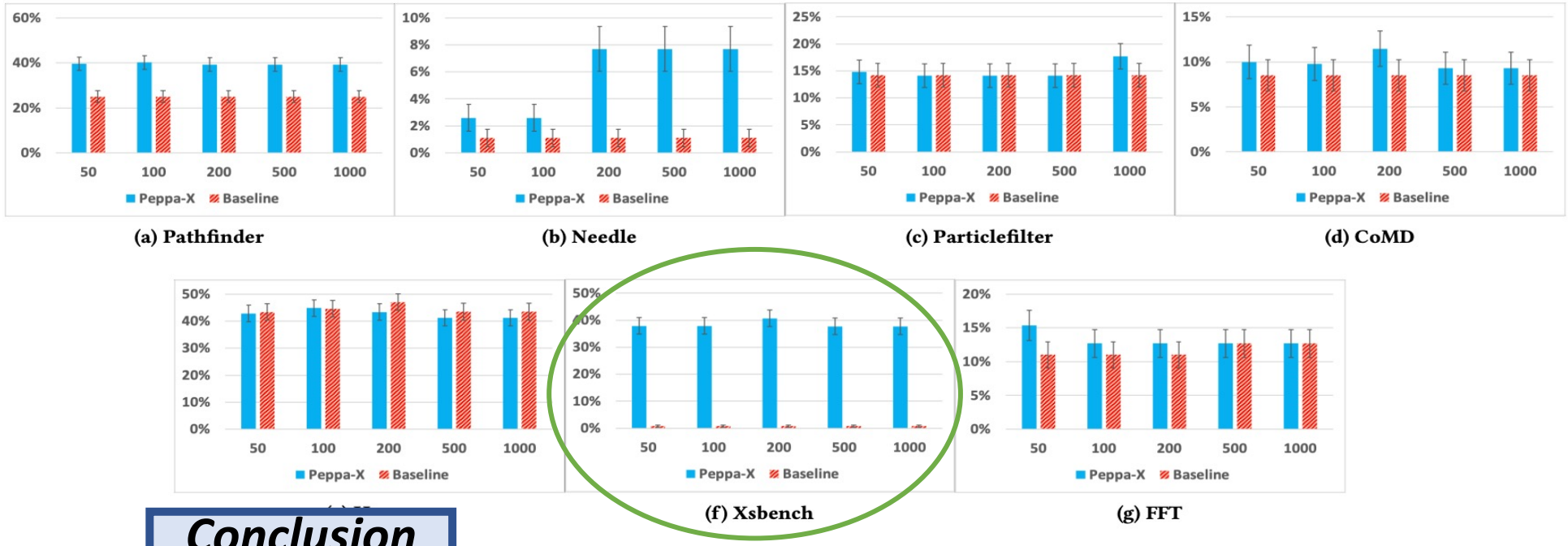
Metrics

- Accuracy
- Efficiency



cation Domains

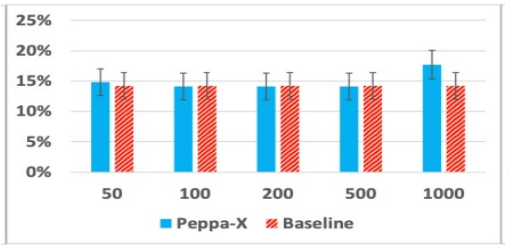
Evaluation: Accuracy



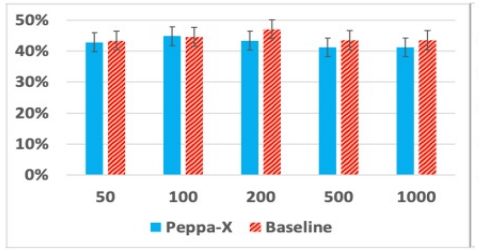
Conclusion

- Peppa-X finds inputs that have much higher SDC probabilities than Baseline at the time budgets of selected generations
  - e.g., Xsbench: 37.9% by Peppa-X while only 0.7% by Baseline

Evaluation: Accuracy



(c) Particlefilter

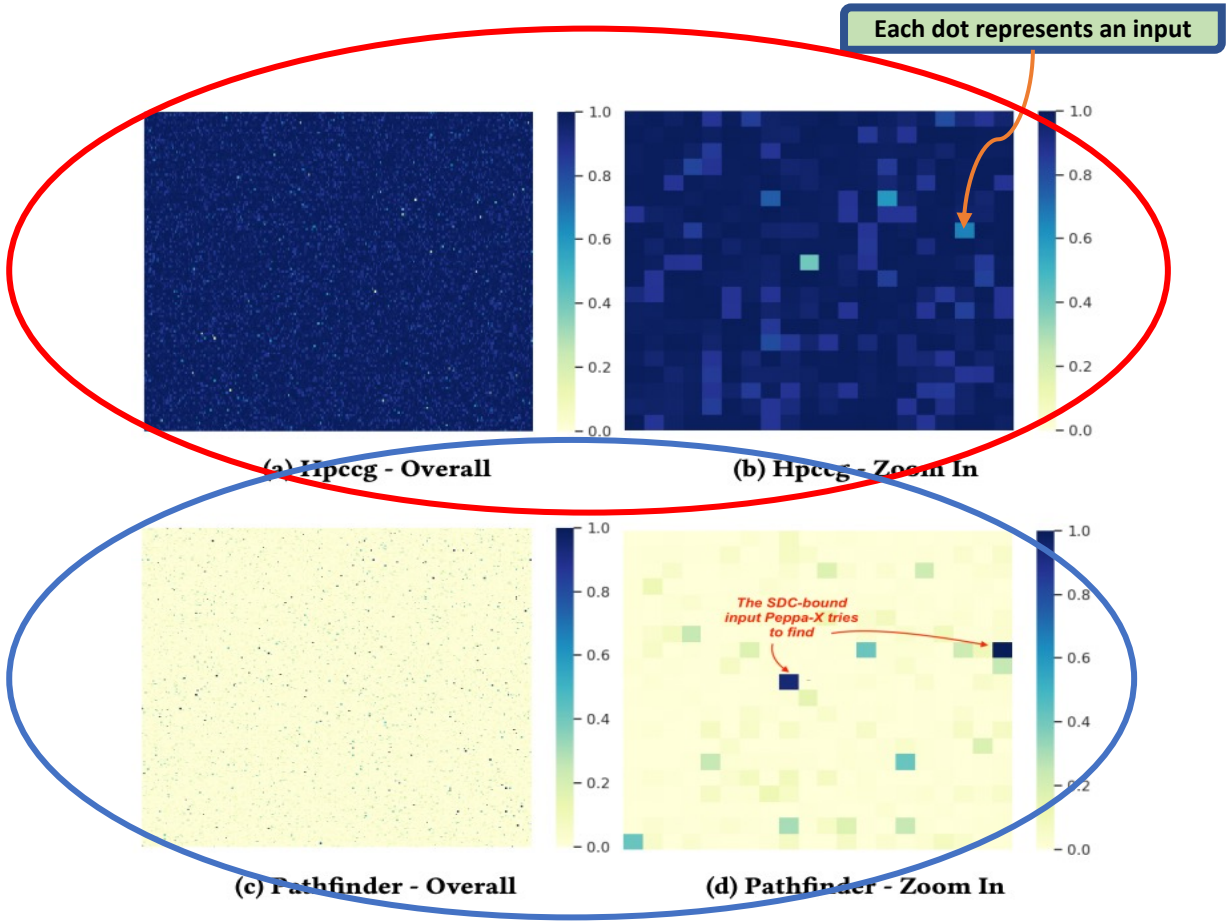


(e) Hpccg

Baseline performs as good as Peppa-X for few cases!

Evaluation: Accuracy

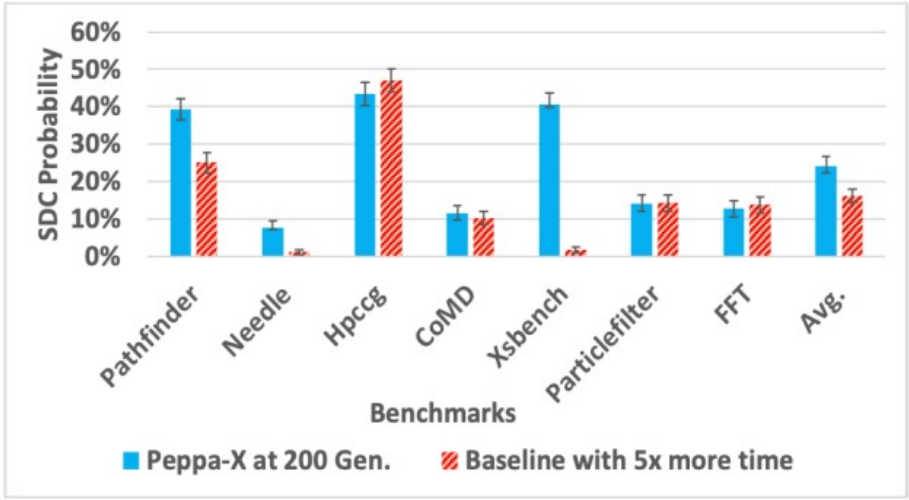
- The darker the color, the higher the SDC Probability
- Most colors are dark for Hpccg
  - A randomly sampled input leads to higher SDC probability
    - Easy task for Baseline!
- Most colors are light for Pathfinder
  - Difficult for Baseline to find SDC-bound inputs



Heat Maps of SDC vulnerability distribution in the input space of Hpccg and Pathfinder

Evaluation: Efficiency

- What if we let baseline run for 5x more time than Peppa-X at 200 generations?
- Why to choose 200 generations?
  - Program SDC probabilities are mostly saturate[illegible]ations



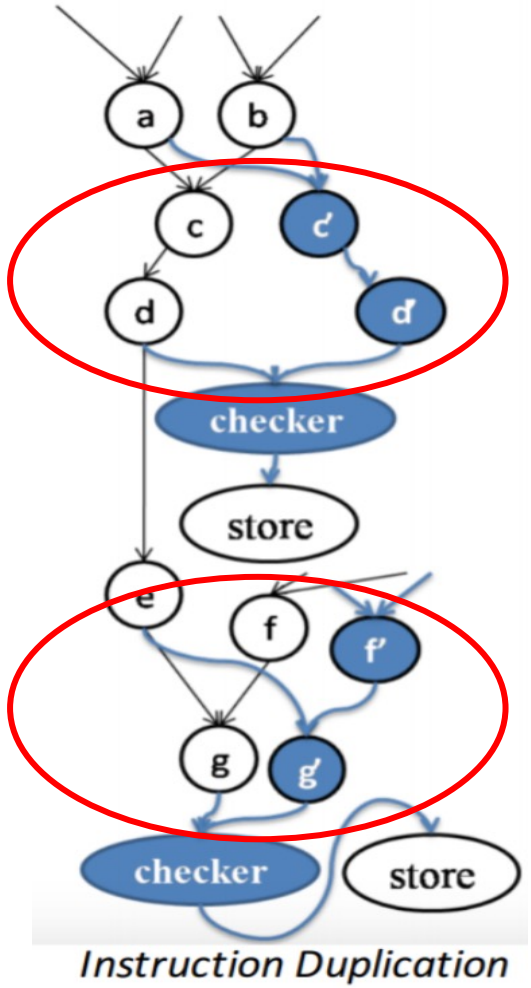
Program SDC probabilities bound by Peppa-X at 200 Generation and Baseline with 5x More Search Time (Y-Axis: SDC Prob. , X-Axis: Benchmarks)

Conclusion

Baseline is still unable to perform as good as Peppa-X

Use Case: Stress Test Selective Inst. Duplication

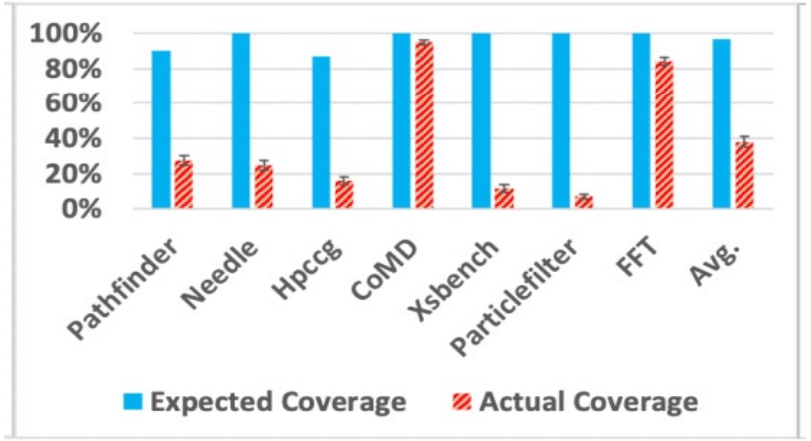
- Only a small amount of instructions being responsible for majority of SDCs
- Duplicate only those instructions by applying 0-1 knapsack
  - Cost → performance overhead of an instruction if duplicated
  - Benefit → SDC coverage by that duplicated instruction



Instruction Duplication

Use Case: Stress Test Selective Inst. Duplication

- Run the protection with default reference input and get expected SDC coverage
- Run the protected program with SDC-bound input
  - Inject faults w t
  - Measure ac



ction level (Y-Axis: SDC Coverage, Benchmarks)

Conclusion

- Protection is compromised!
  - Avg expected coverage is 96.63%
  - Avg actual coverage is only 38.02%

Conclusion

- Peppa-X is both accurate and efficient to identify SDC-bound inputs
  - Only one time cost for FI simulations
  - Leveraging static and dynamic analysis
- Baseline cannot find such SDC-bound inputs even with 5x more search time
  - Need extensive FI simulations to evaluate each random input
  - Not practical as FI takes long time!
- Our tool is open-sourced: https://github.com/hasanur-rahman/Peppa-X

Md Hasanur Rahman (Hasan)
University of Iowa
mdhasanur-rahman@uiowa.edu
https://hasanur-rahman.github.io/