gem5-Approxilyzer: An Open-Source Tool for Application-Level Soft Error Analysis

Radha Venkatagiri*, Khalique Ahmed*, Abdulrahman Mahmoud, Sasa Misailovic, Darko Marinov, Christopher Fletcher, Sarita Adve

University of Illinois at Urbana-Champaign

* Equal contribution first authors

Supported by NSF and by the Applications Driving Architectures (ADA) Research Center

Motivation

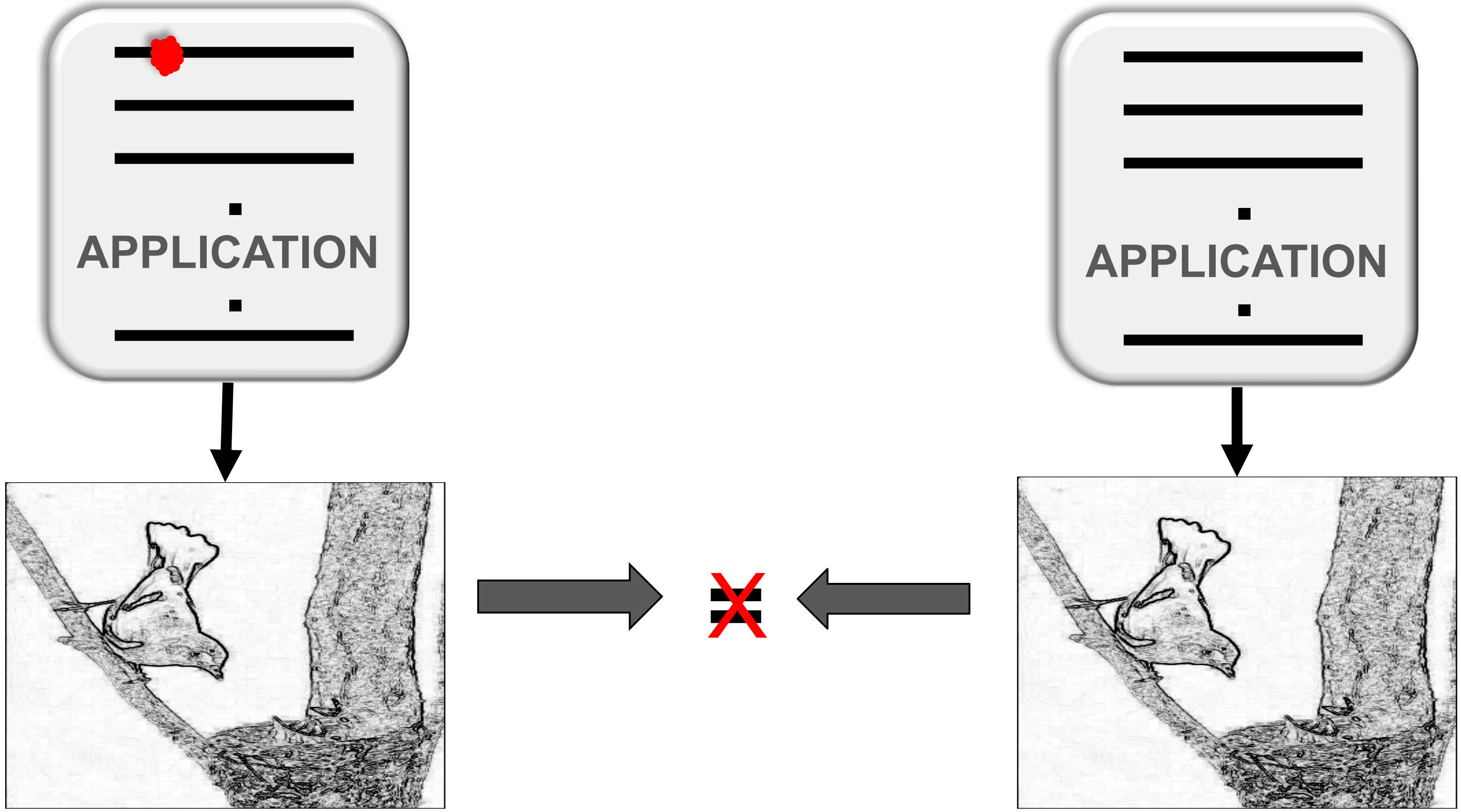
- **Modern systems increasingly susceptible to soft errors**
  - **Too expensive to protect against all errors**

- **Systems that allow controlled errors becoming popular**
  - **Approximate computing, low-cost less-than-perfect resiliency solutions**

Fundamental question: How do soft errors affect program output?

Error Outcome (of Single Error)

Sobel **Single error injection**

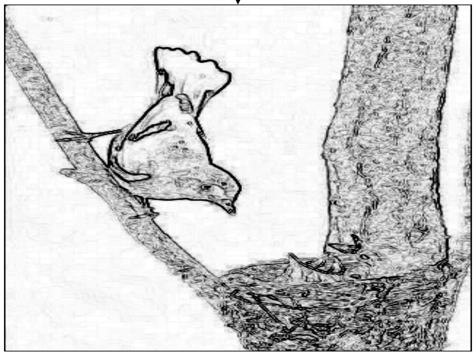


Output Corruption!

Quantifying Output Quality

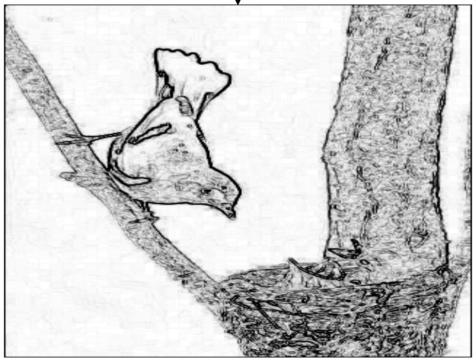
Sobel

APPLICATION



Single error injection

APPLICATION



Output Corruption!

Quality Metric (domain specific)

Image difference (rmse)

7% Quality degradation

Is Output Quality Acceptable?

Sobel

Single error injection

APPLICATION

Quality Metric
(domain specific)

Quality Threshold = 10%

APPLICATION

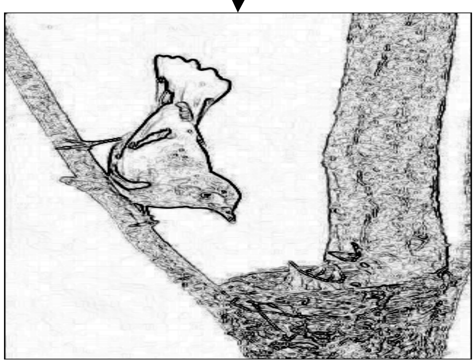
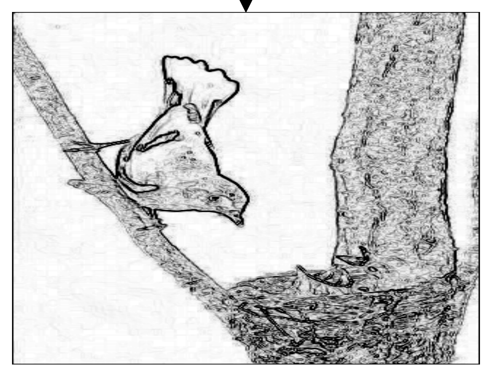


Image difference
(rmse)

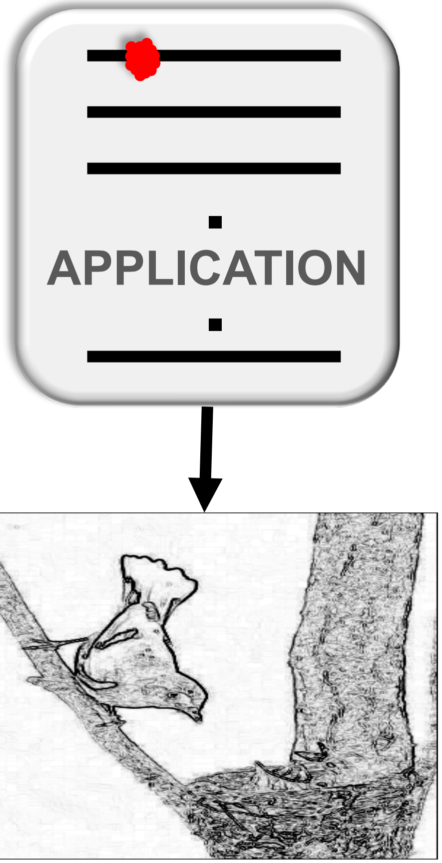
7%
Quality
degradation



User-Acceptable Output Corruption!

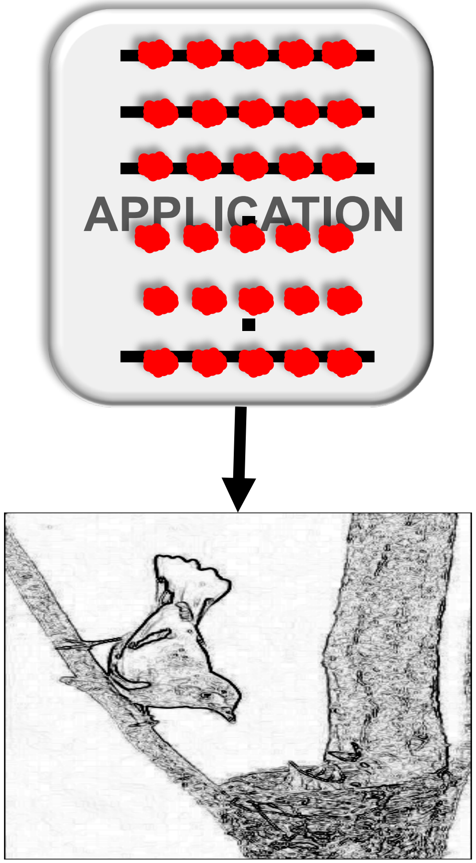
Error Outcome (of Single Error)

Sobel



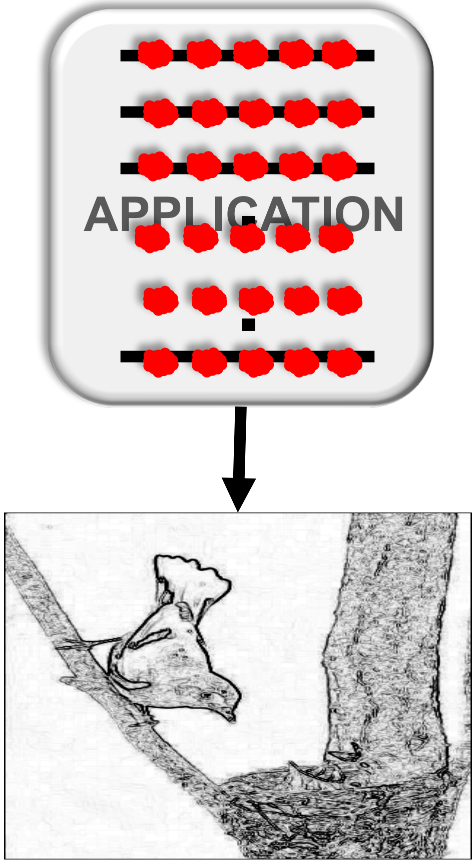
Error Outcome (of All Errors)

Sobel Filter



Application-Level Error Analysis (Holy Grail)

Sobel Filter

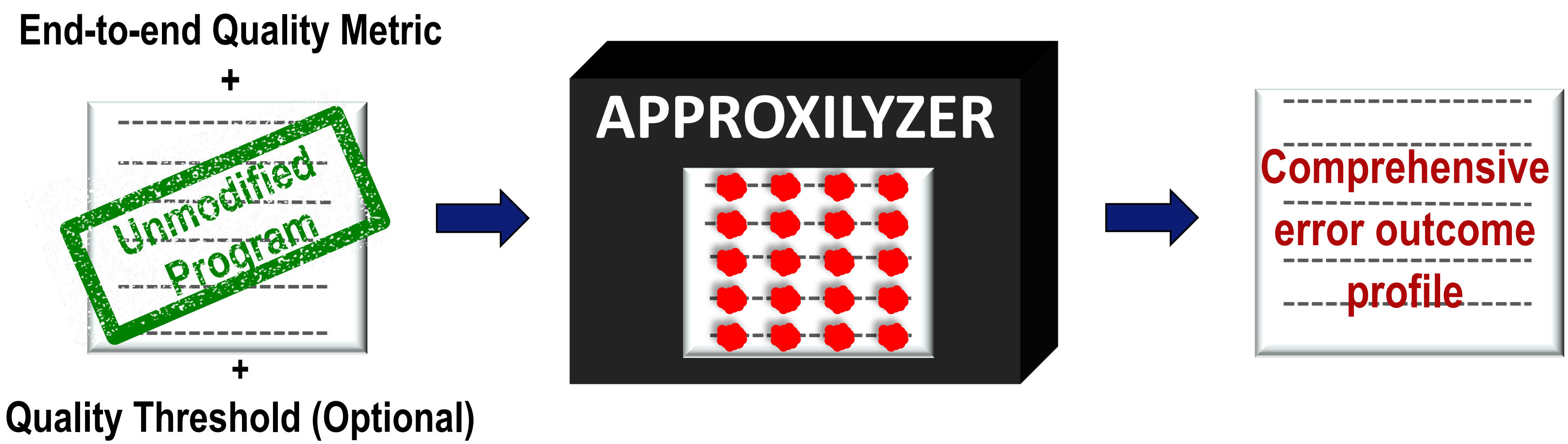


Ideal Application Error Analysis:

- Comprehensive
  - All errors (for given error model)
- Accurate
  - Precisely calculate output quality
- Automatic
  - Minimal programmer burden
- Cheap
  - Many error injections = expensive!

Approxilyzer [Micro 2016]

- **Approxilyzer: Tool to determine impact of error on program output**
  - **Minimal programmer burden, general-purpose, accurate, comprehensive**
  - **Program analysis + (relatively few) error injections**
  - **Error Model: Single bit transient errors in operand register + dynamic instructions**

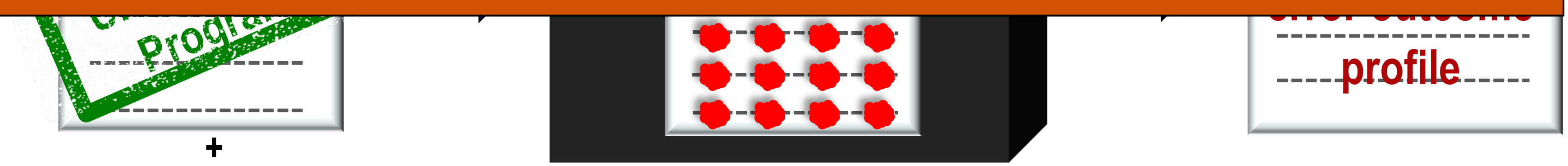


R. Venkatagiri et al., "Approxilyzer: Towards a systematic framework for instruction-level approximate computing and its application to hardware resiliency", MICRO 2016

Approxilyzer

- **Approxilyzer: Tool to determine impact of error on program output**
  - **Minimal programmer burden, general-purpose, accurate, comprehensive**
  - **Program analysis + (relatively few) error injections**

- **Prior implementation has limitations**
  - **Proprietary simulator (Wind River Simics)**
  - **Restricted to SPARC ISA**



\+

Quality Threshold (Optional)

R. Venkatagiri et al., “Approxilyzer: Towards a systematic framework for instruction-level approximate computing and its application to hardware resiliency”, MICRO 2016

Contributions: gem5-Approxilyzer

- gem5-Approxilyzer: Fully open-source implementation of Approxilyzer
  - *https://github.com/rsimgroup/gem5-Approxilyzer*
  - Built using open-source gem5 simulator
  - Support for x86; non-trivial engineering effort
  - Can be extended to other ISAs

- Show effectiveness/accuracy of Approxilyzer technique for x86
  - Two orders of magnitude reduction in error injections (over naïve techniques)
  - Determines error outcomes with high accuracy (97% on average)

- Error Analysis of application under different ISAs (x86 and SPARC)
  - Error profile of same application significantly different under x86 & SPARC
  - Static instructions that are approximable/resilient vary significantly by ISA

Outline

- Introduction
- (gem5-) Approxilyzer interface & techniques
- gem5-Approxilyzer Use Cases
- Results
- Conclusion

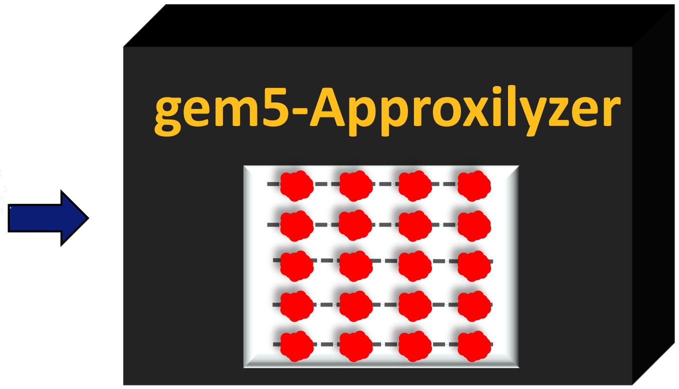
gem5-Approxilyzer: Inputs

End-to-end Quality Metric
(domain-specific)

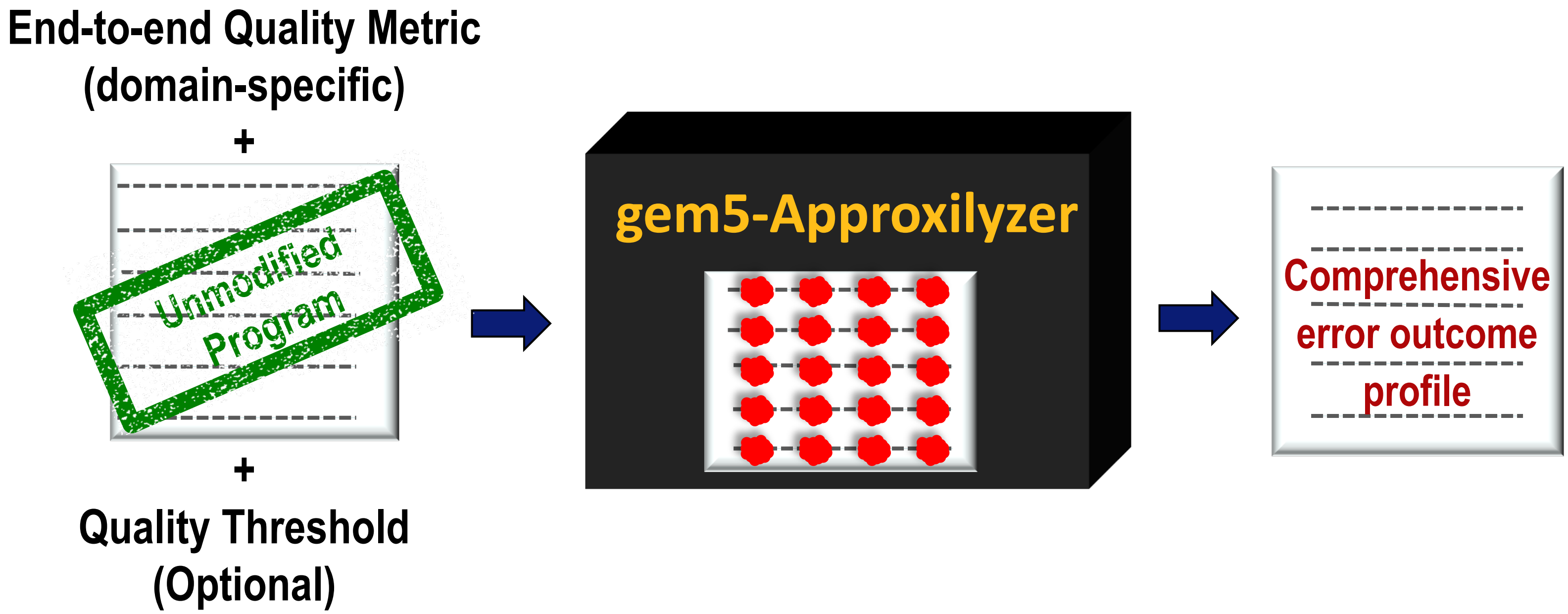
+

+

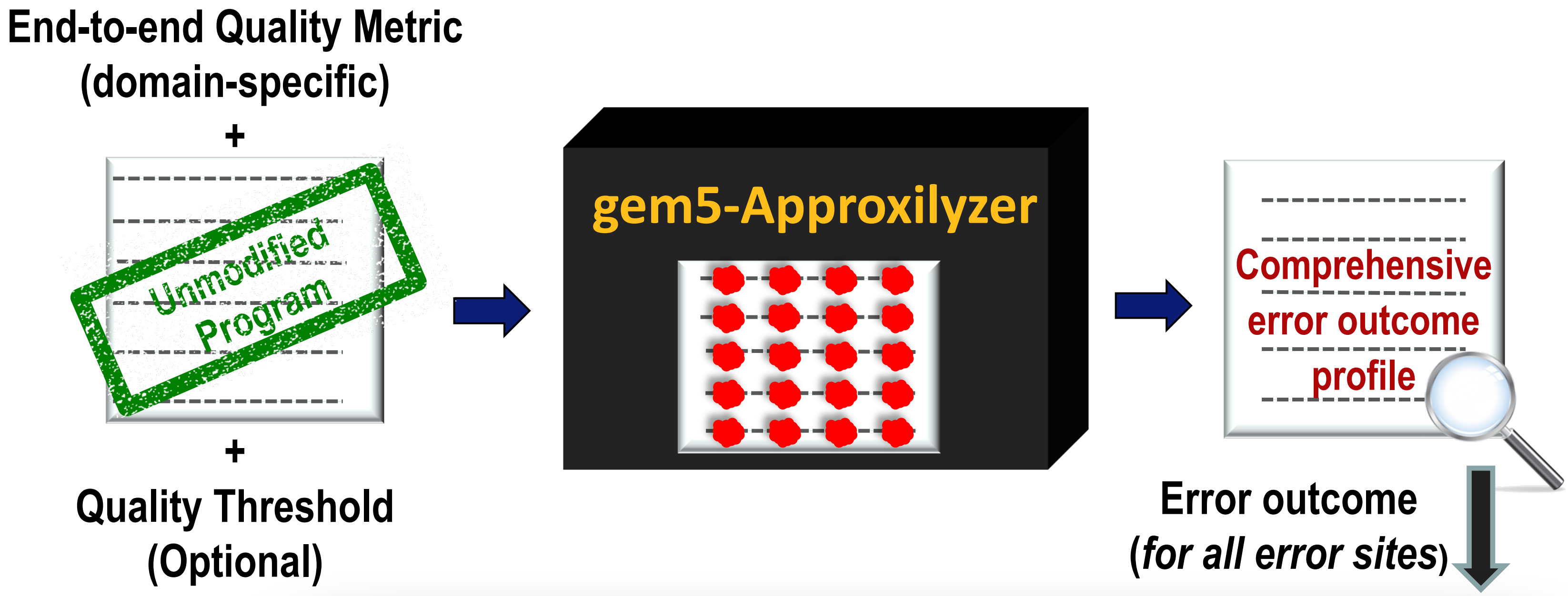
Quality Threshold
(Optional)



gem5-Approxilyzer: Output



Comprehensive Error Outcome Profile

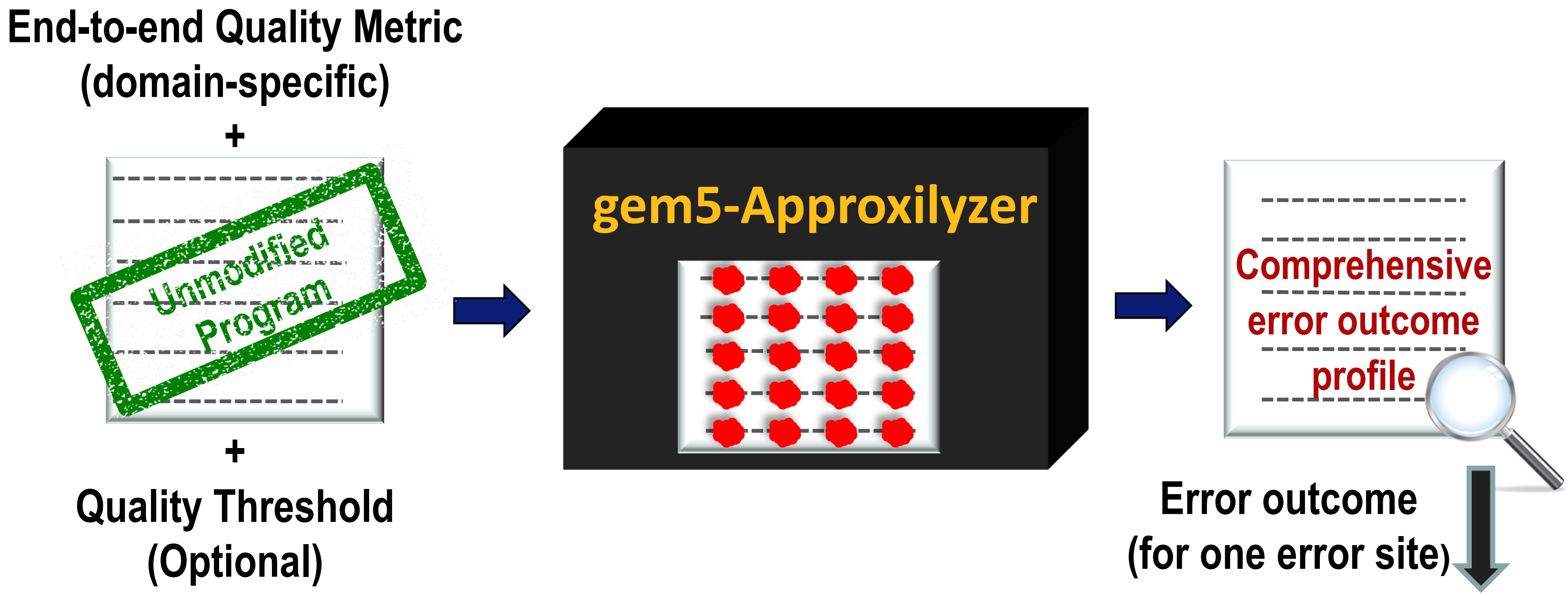


radha — venktgr2@veena-server:/shared/workspace/venktgr2/GEM5_Approxilyzer/Approxilyz...

```
0x400995, 594769813038500, r8, 14, Integer, Source :: SDC-Maybe:0.0218
```

379628,1 99%

Error Outcome for One Error Site

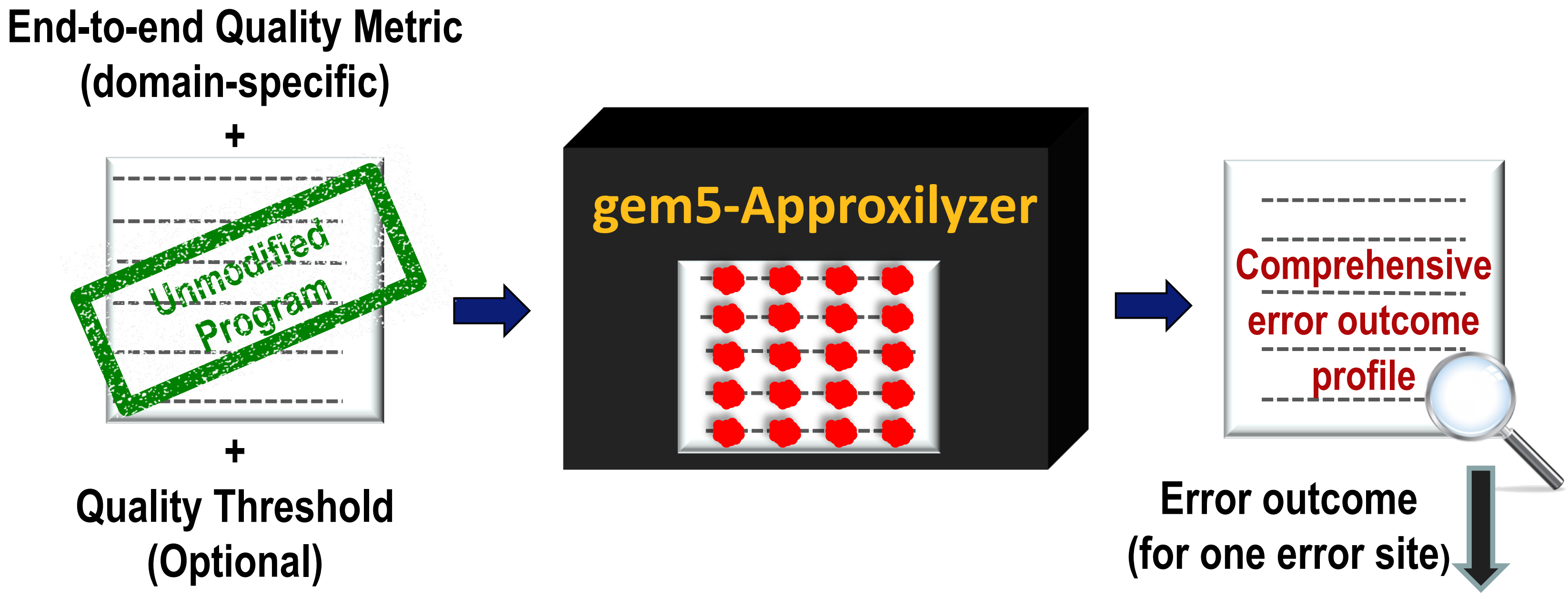



```
0x400995, 594769813038500, r8, 14, Integer, Source :: SDC-Maybe:0.0218
```

Error Site Description

Error Outcome

Error Outcome for One Error Site

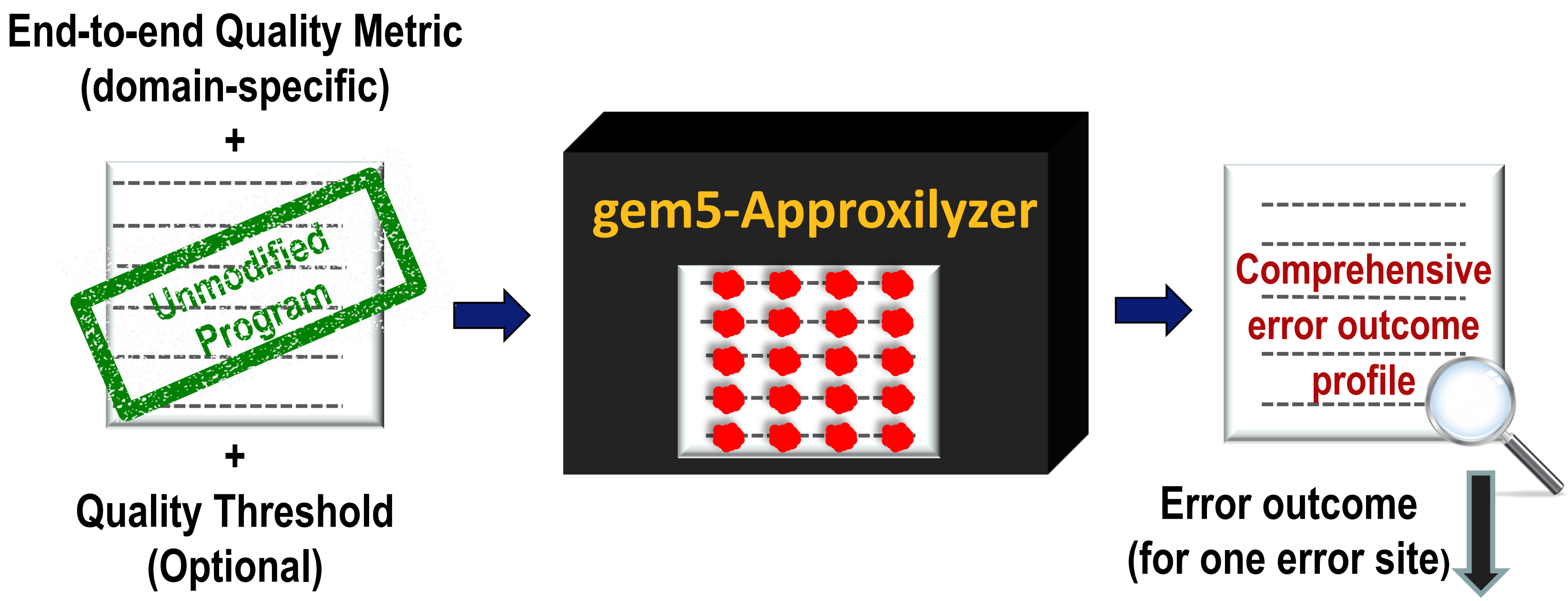


Error Model: Single bit errors in operand registers of dynamic instructions

`0x400995, 594769813038500, r8, 14, Integer, Source : SDC-Maybe:0.0218`

Error Site Description

Error Outcome for One Error Site

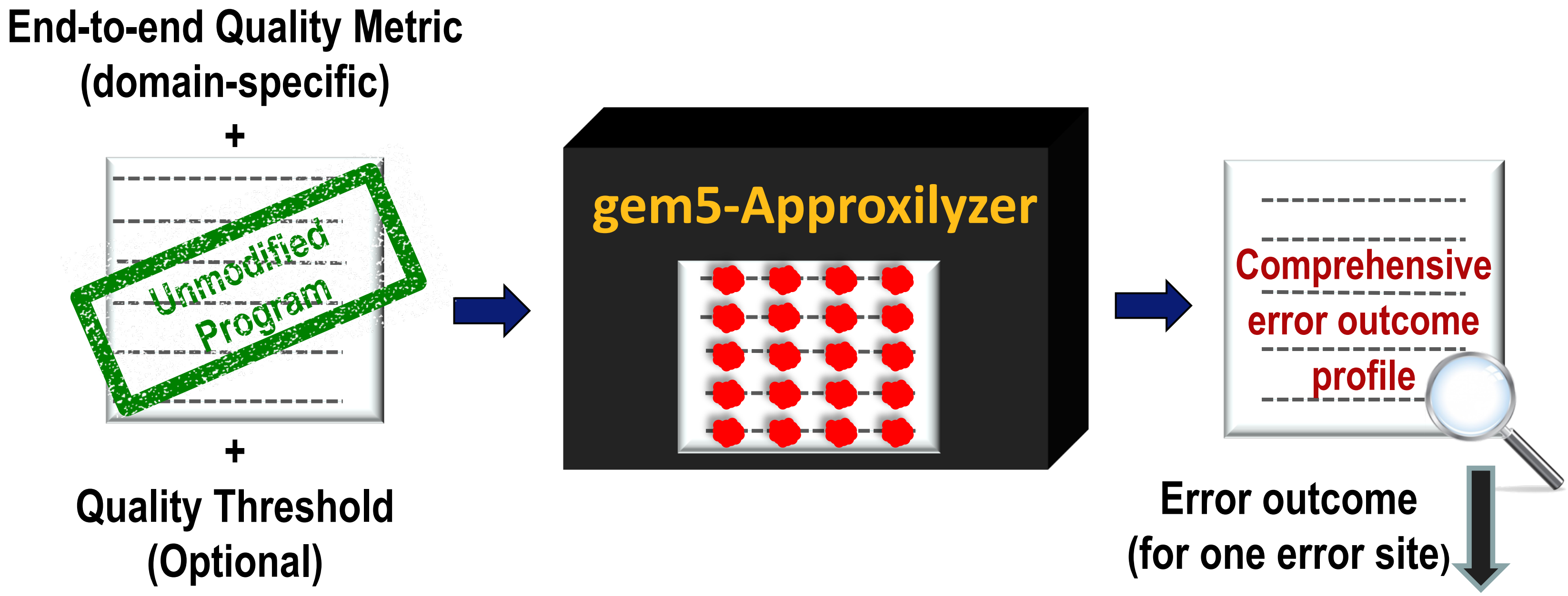


Error Site: Dynamic instruction + Operand Register + Register Bit

`0x400995, 594769813038500, r8, 14, Integer, Source :: SDC-Maybe:0.0218`

PC + Cycle = Dynamic instruction

Error Outcome for One Error Site

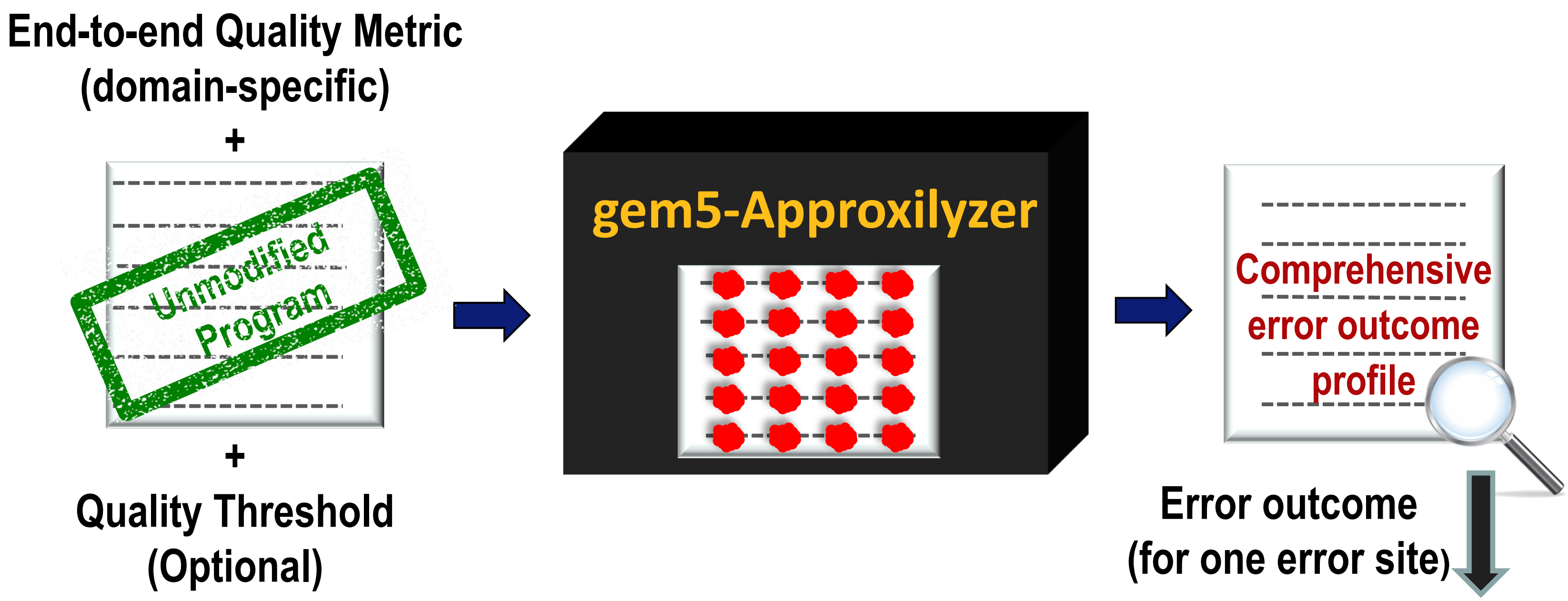


Error Site: Dynamic instruction + Operand Register + Register Bit

```
0x400995, 594769813038500, r8, 14, Integer, Source :: SDC-Maybe:0.0218
```

Register Name **Register Bit**

Error Outcome for One Error Site

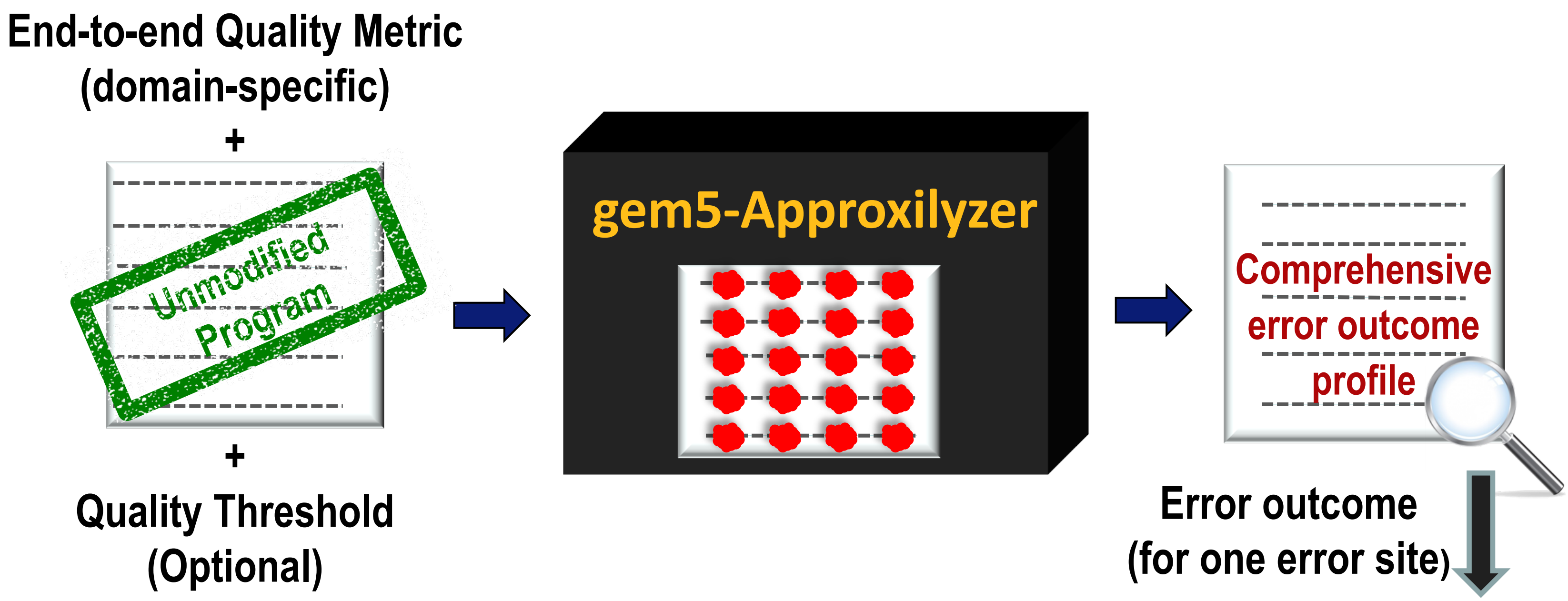


Error Site: Dynamic instruction + Operand Register + Register Bit

```
0x400995, 594769813038500, r8, 14, Integer, Source :: SDC-Maybe:0.0218
```

Register Type **Operand Type**

Error Outcome for One Error Site

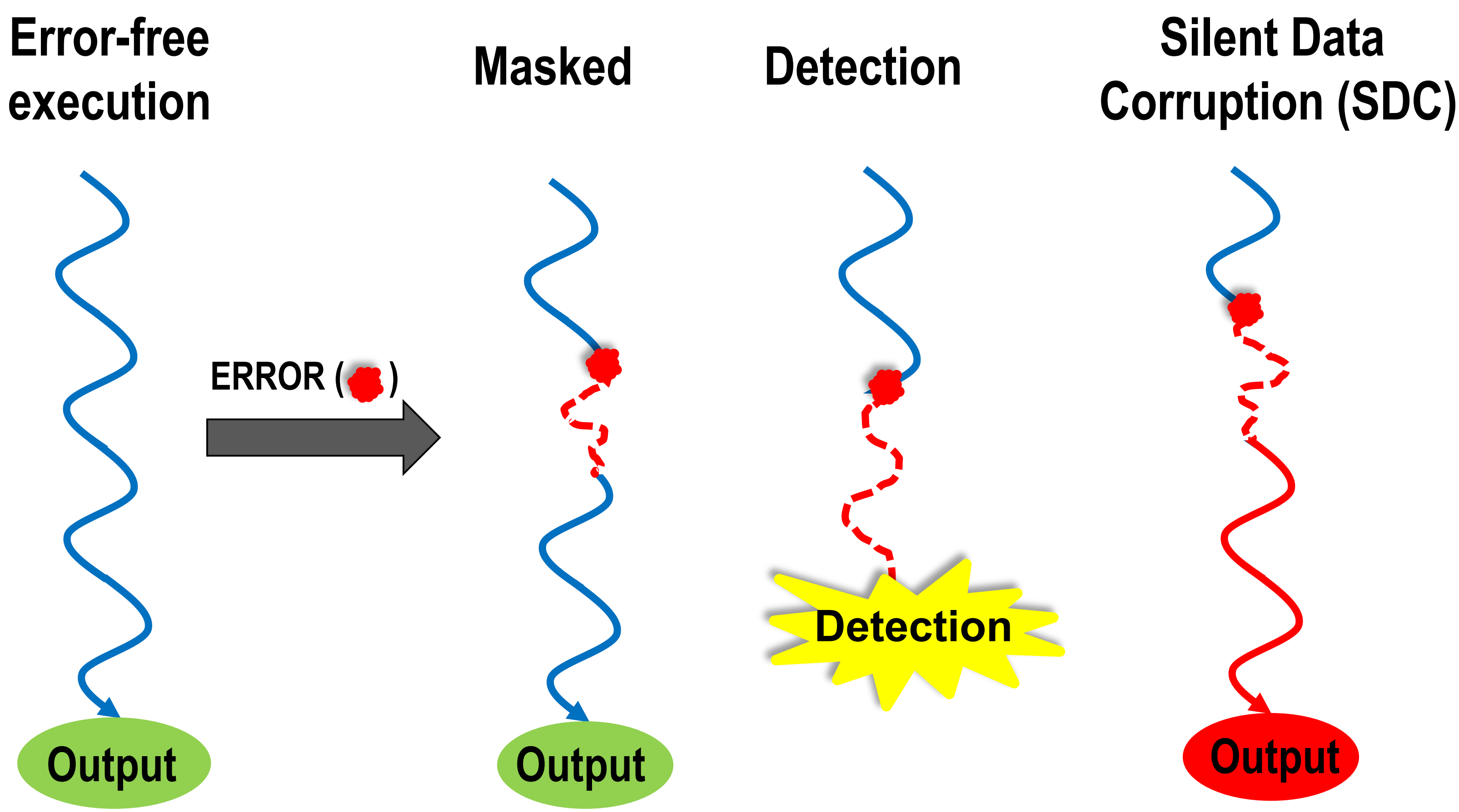


Error Outcome: Impact of an error, at this error site, on program output

```
0x400995, 594769813038500, r8, 14, Integer, Source : SDC-Maybe:0.0218
```

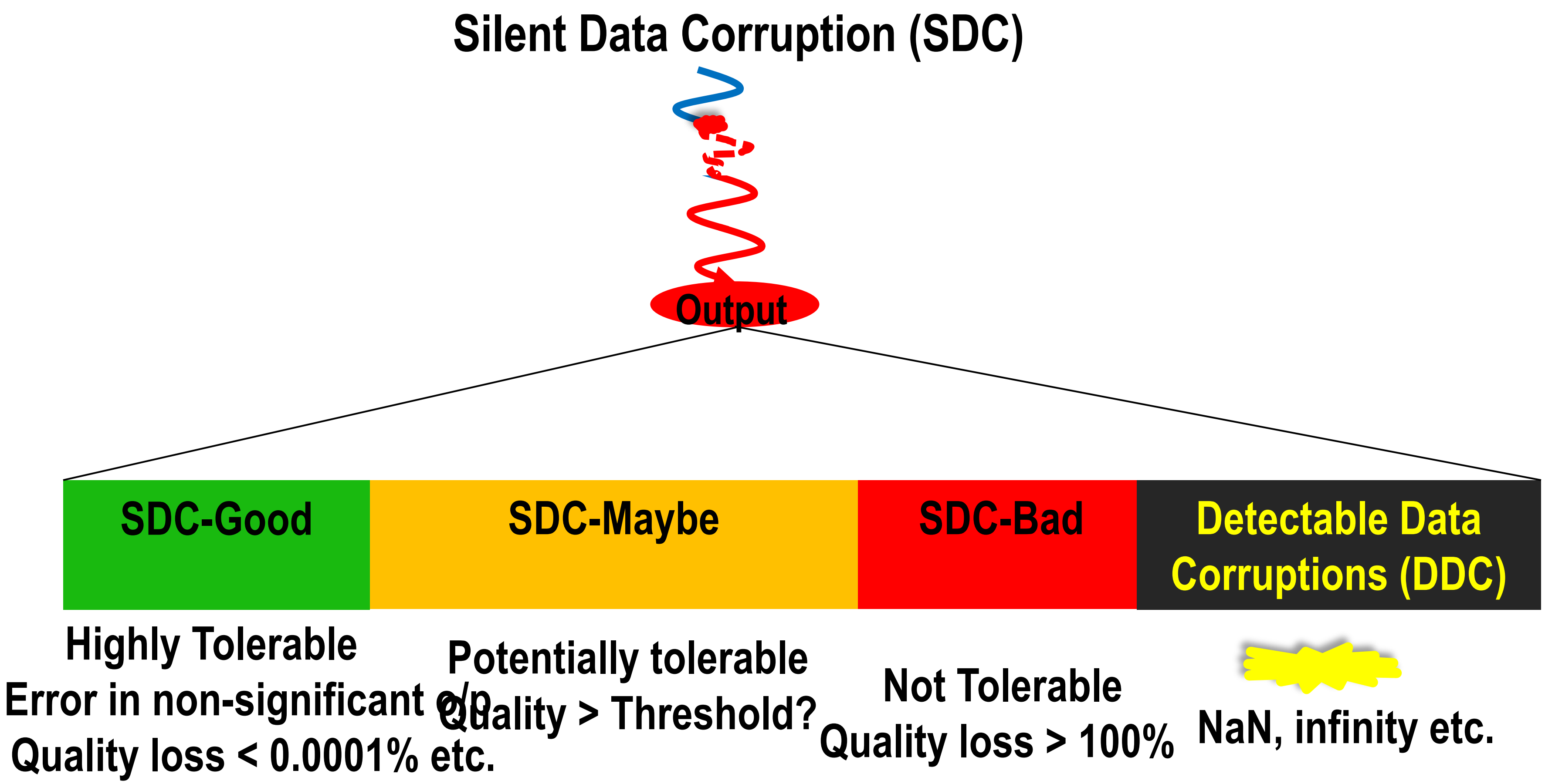
Error Outcome

Error Outcome Categories for Single Bit Errors



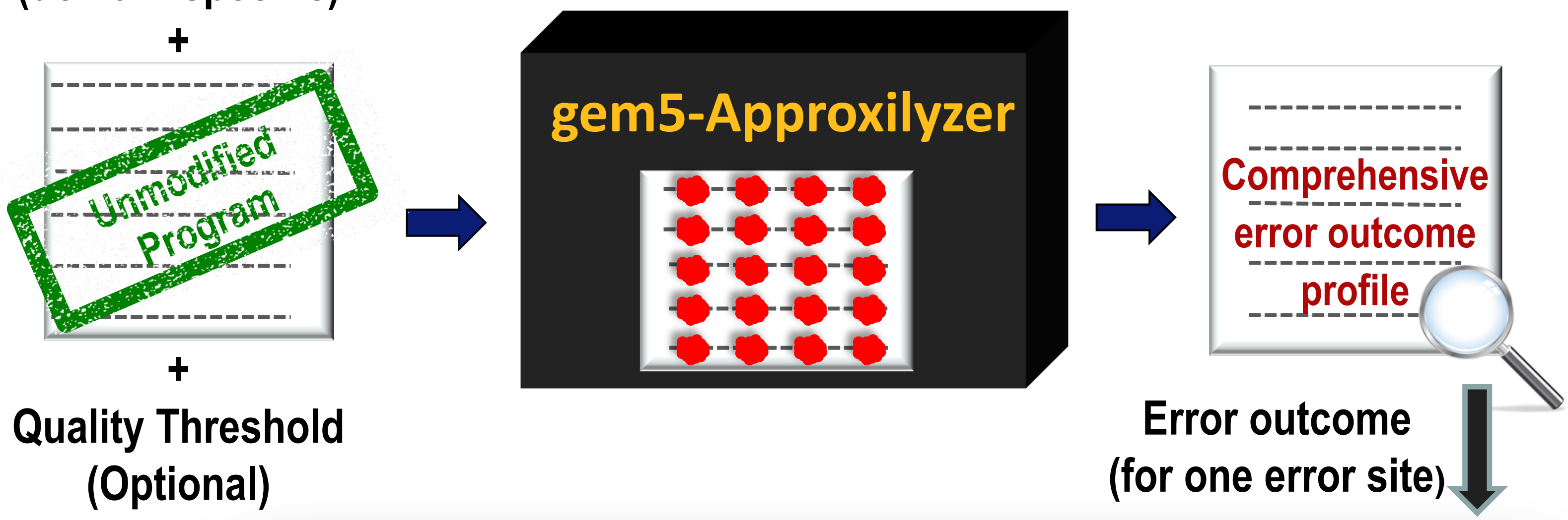
Quality Aware Error Categorization

- Not all output corruptions are bad
  - SDCs can be differentiated based on the quality of data corruptions produced



Error Outcome for One Error Site

End-to-end Quality Metric
(domain-specific)
+



+
Quality Threshold
(Optional)

Error outcome
(for one error site)

radha — venktgr2@veena-server:/shared/workspace/venktgr2/GEM5_Approxilyzer/Approxilyz...

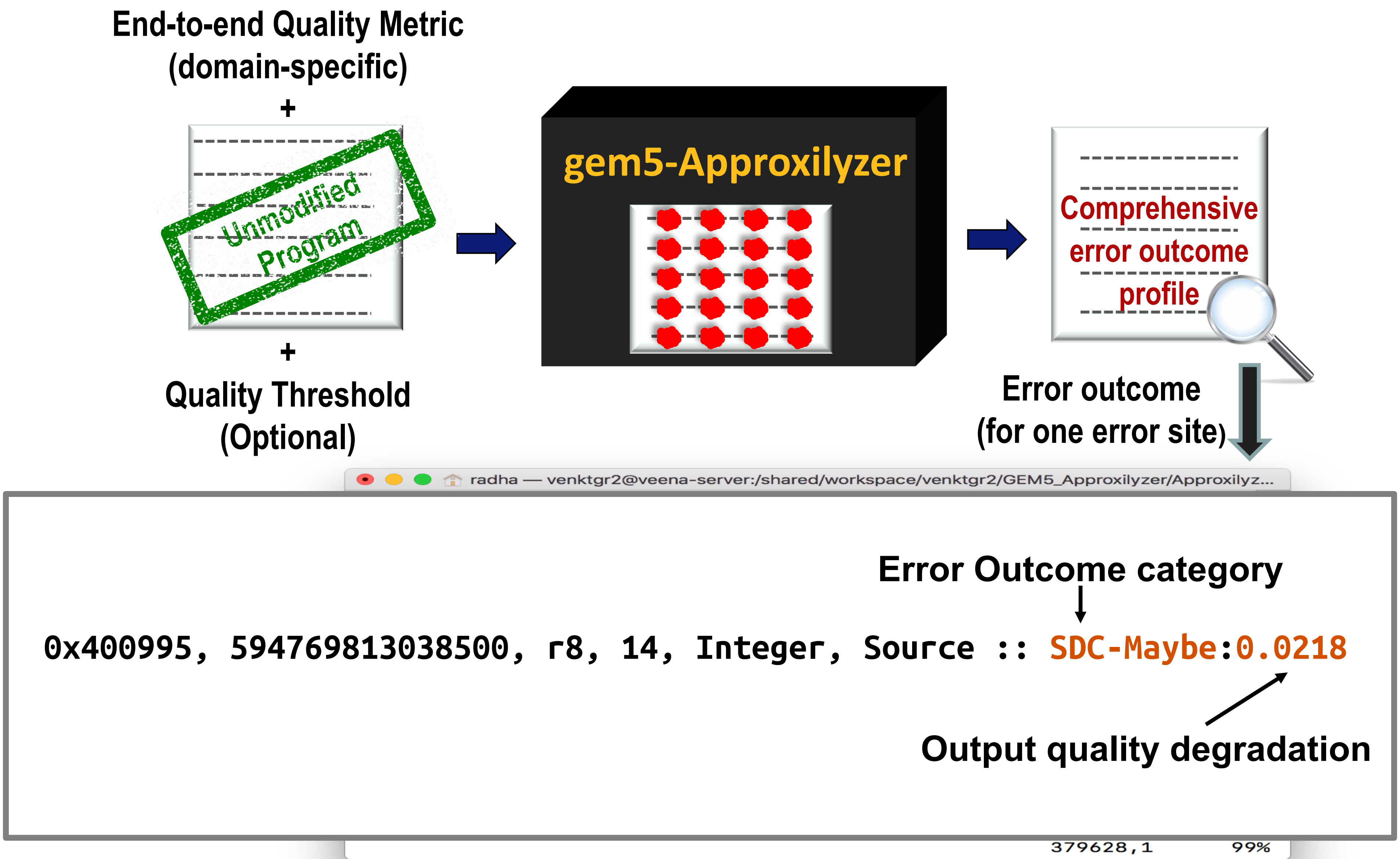
Error Outcome: Impact of an error, at this error site, on program output

```
0x400995, 594769813038500, r8, 14, Integer, Source : SDC-Maybe:0.0218
```

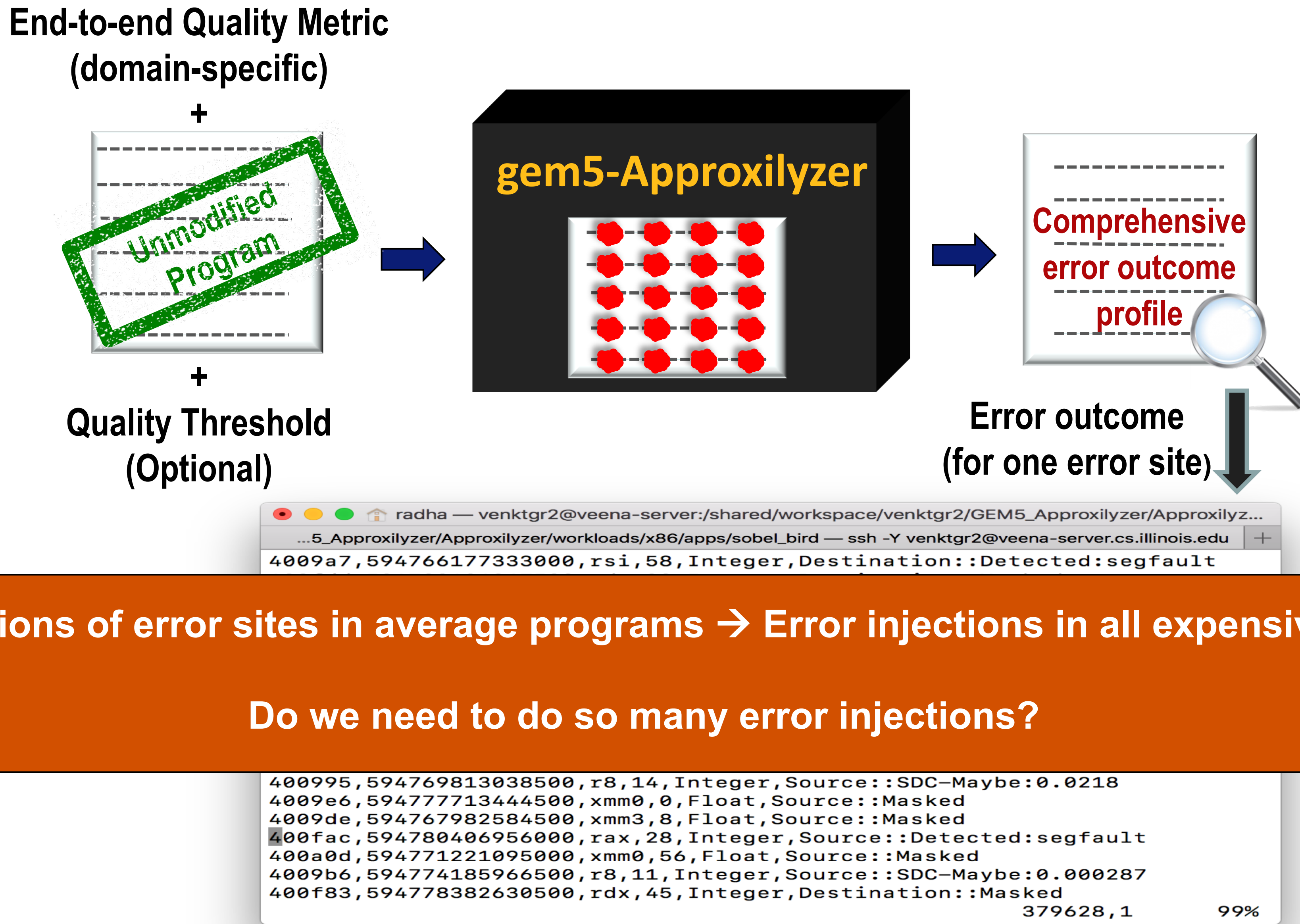
Error Outcome

379628,1 99%

Error Outcome for One Error Site



Error Injections in All Error Sites Expensive



Billions of error sites in average programs → Error injections in all expensive!

Do we need to do so many error injections?

Error Injection Pruning

- Error injection pruning
  - Removes redundant error injections
- Error injection pruning technique 1 : Known outcome pruning
  - No injection required when error outcome is known apriori

Known Outcome Pruning: Address Bound

- **Error injection pruning**
  - Removes redundant error injection
- **Error injection pruning technique 1 : Known outcome pruning**
  - No injection required when error outcome is known apriori

```
mov    (%ebx),    %eax
```

Address in ebx = 0x4ac8

Known Outcome Pruning: Address Bound

- **Error injection pruning**
  - **Removes redundant error injection**
- **Error injection pruning technique 1 : Known outcome pruning**
  - **No injection required when error outcome is known apriori**
  - **Address bound pruning**
    - **Errors outside address range result in Detected outcomes**

(Error)

mov (%ebx), %eax

Address in ebx = 0x10004ac8

Detection

Outside addressable range => Segmentation Fault

Known Outcome Pruning: Def-Use

- Error injection pruning
  - Removes redundant error injection
- Error injection pruning technique 1 : Known outcome pruning
  - No injection required when error outcome is known apriori
  - Address bound pruning
    - Errors outside address range result in Detected outcomes
  - Def-use pruning
    - Injection in a def ⇔ injection at first use
    - Only one needs to be explored by error injection

`mov (%ebx), %eax`	**Definition of register eax**
`add %eax, %ecx`	**First use of register eax**

Known Outcome Pruning: Def-Use

- **Error injection pruning**
  - Removes redundant error injection
- **Error injection pruning technique 1 : Known outcome pruning**
  - No injection required when error outcome is known apriori
  - Address bound pruning
    - Errors outside address range result in Detected outcomes
  - Def-use pruning
    - Injection in a def ⇔ injection at first use
    - Only one needs to be explored by error injection

```
mov   (%ebx),   %eax        Definition of register eax

add   %eax,   %ecx          First use of register eax
```

Known Outcome Pruning: Def-Use

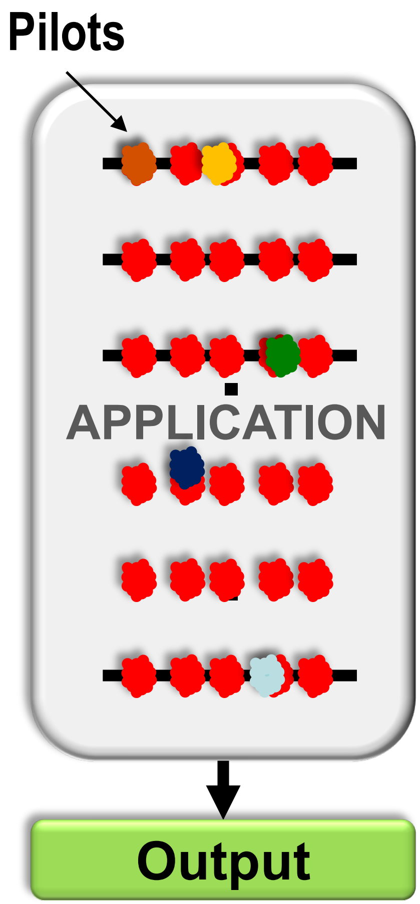
- Error injection pruning
  - Removes redundant error injection
- Error injection pruning technique 1 : Known outcome pruning
  - No injection required when error outcome is known apriori
  - Address bound pruning
    - Errors outside address range result in Detected outcomes
  - Def-use pruning
    - Injection in a def ⇔ injection at first use
    - Only one needs to be explored by error injection

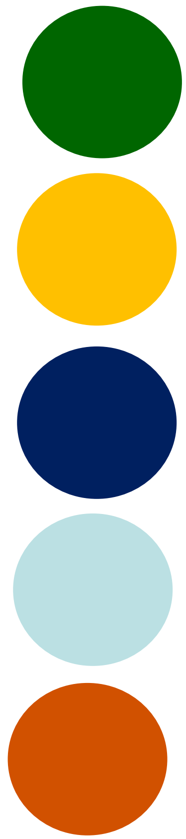
`mov (%ebx), %eax`	**Definition of register eax**
`add %eax, %ecx`	**First use of register eax**

Error injection Pruning: Equivalence

- **Error injection pruning**
  - **Removes redundant error injection**
- Pruning technique 1 : Known outcome pruning
  - No injection required when error outcome is known apriori
  - Address bound pruning
    - Errors outside address range result in Detected outcomes
  - Def-use pruning
    - Injection in a def ⇔ injection at first use
    - Only one needs to be explored by error injection
- **Pruning technique 2: Equivalence based pruning**
  - **Errors are equalized based on heuristics (control + data flow)**
  - **Only single error injection per equalized set**

Error injection pruning: Equivalence





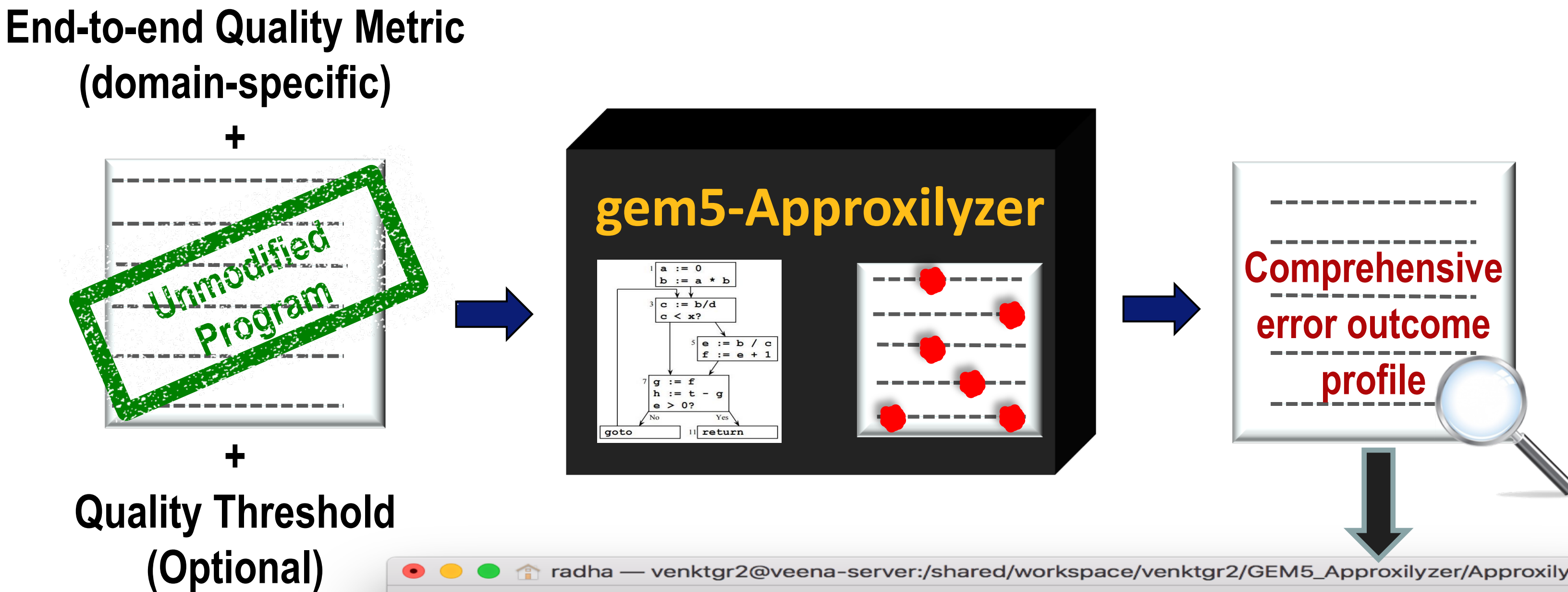
Equivalence Classes (using data + control flow heuristics)

Inject error in Pilot

Pilot outcome = Outcome of all errors in class

Insight: Errors flowing through similar control+data paths produce similar outcomes

Comprehensive Error Profile with Few Injections



radha — venktgr2@veena-server:/shared/workspace/venktgr2/GEM5_Approxilyzer/Approxilyz...

...5_Approxilyzer/Approxilyzer/workloads/x86/apps/sobel_bird — ssh -Y venktgr2@veena-server.cs.illinois.edu

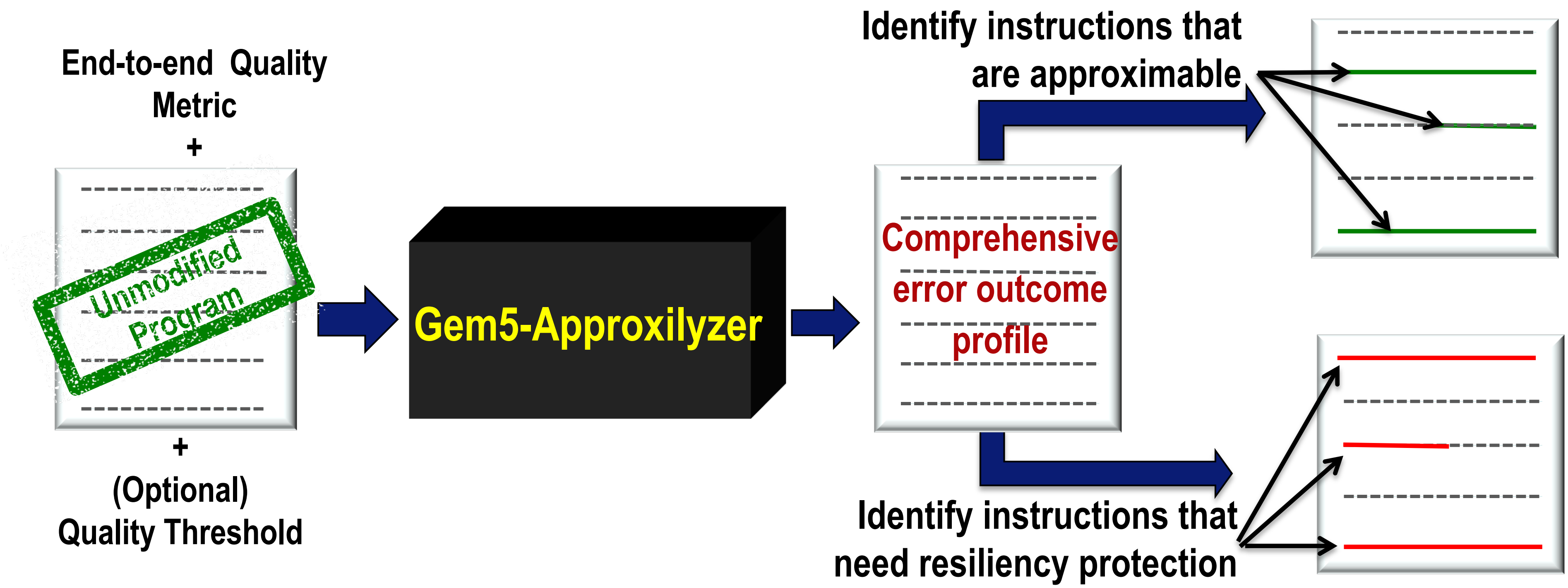
```
4009a7,594766177333000,rsi,58,Integer,Destination::Detected:segfault
400fdd,594780406967500,edx,20,Integer,Destination::Masked
400970,594764472062500,eax,17,Integer,Destination::SDC—Maybe:0.000638
400990,594758560610500,rcx,25,Integer,Source::Detected:segfault
400fd2,594778156118500,r8,29,Integer,Destination::Masked
4009a3,594780407066500,rsi,34,Integer,Source::Detected:mem_bounds
400990,594778236611000,rdi,32,Integer,Source::Detected:mem_bounds
4009ee,594758339776500,xmm0,26,Float,Source::Masked
400f28,594744890787000,ecx,21,Integer,Destination::Masked
400a29,594773068794500,xmm0,28,Float,Destination::Detected:mem_bounds
40099b,594777409518000,xmm0,55,Float,Source::Masked
400fd2,594753646272500,r8,60,Integer,Destination::Masked
400995,594769813038500,r8,14,Integer,Source::SDC—Maybe:0.0218
4009e6,594777713444500,xmm0,0,Float,Source::Masked
4009de,594767982584500,xmm3,8,Float,Source::Masked
400fac,594780406956000,rax,28,Integer,Source::Detected:segfault
400a0d,594771221095000,xmm0,56,Float,Source::Masked
```

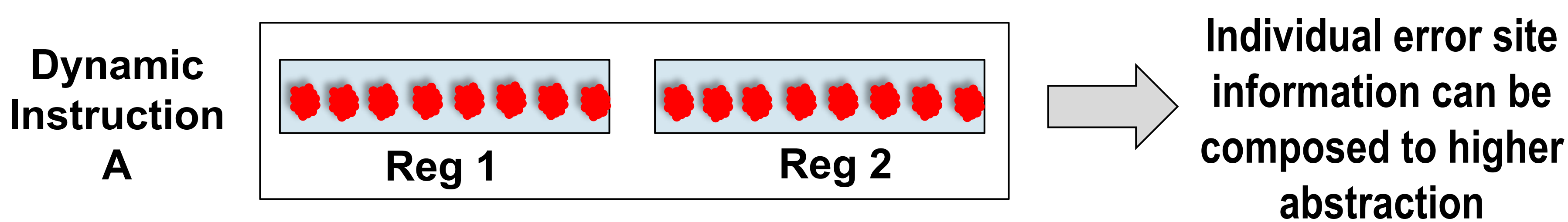
Few error injections to predict the outcome of virtually all errors

Outline

- Introduction
- (gem5-) Approxilyzer interface & techniques
- **gem5-Approxilyzer Use Cases**
- Results
- Conclusion

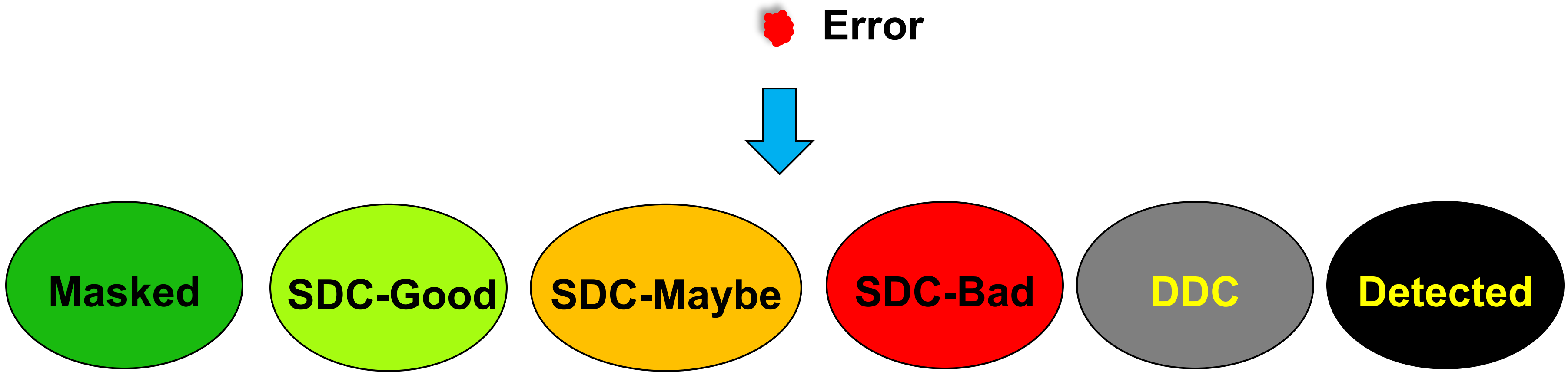
Use Cases of gem5-Approxilyzer



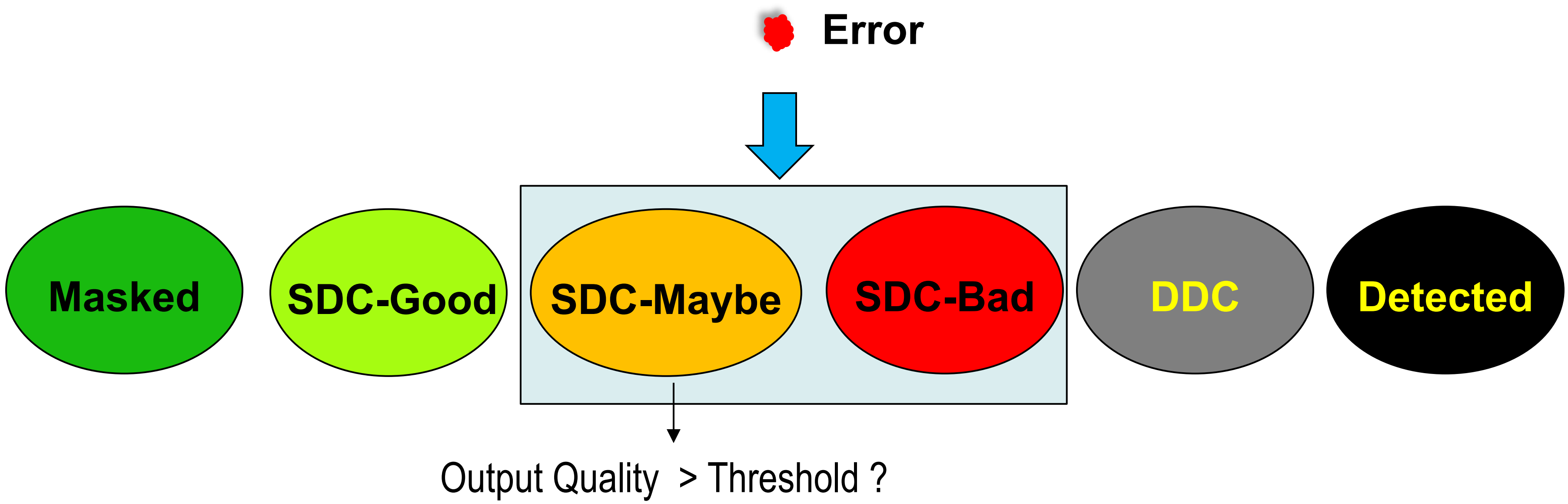


Error Outcome Categorization by Use Case

Error

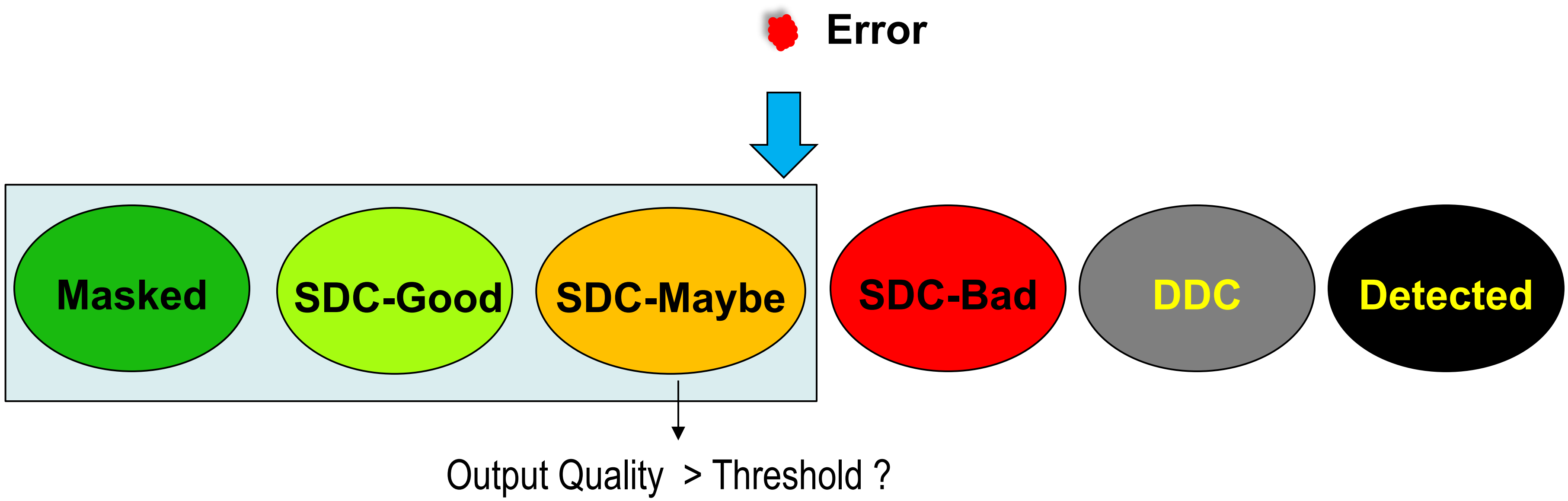


Use Case: Resiliency



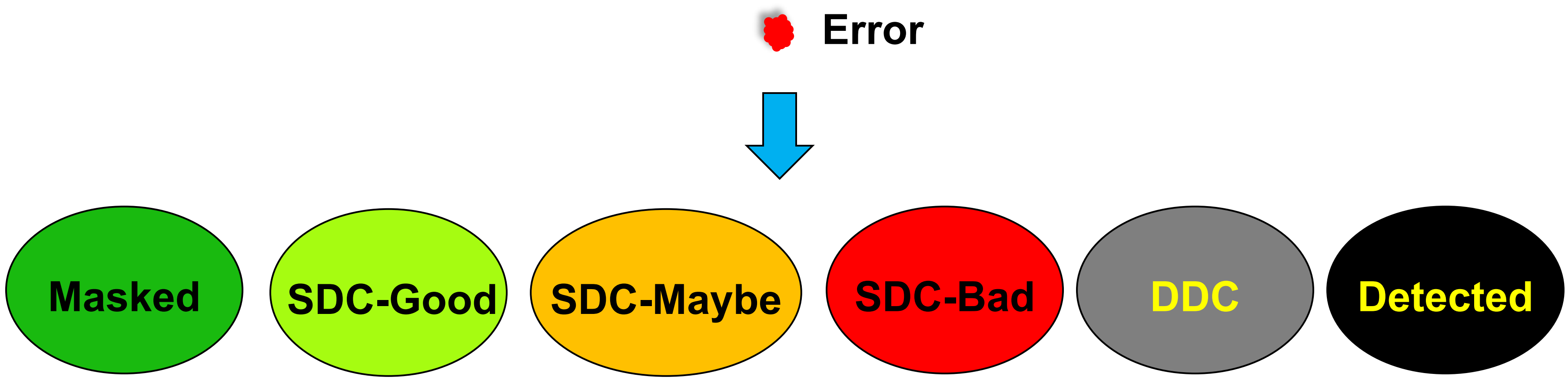
Which class of errors need resiliency protection?

Use Case: Approximate Computing

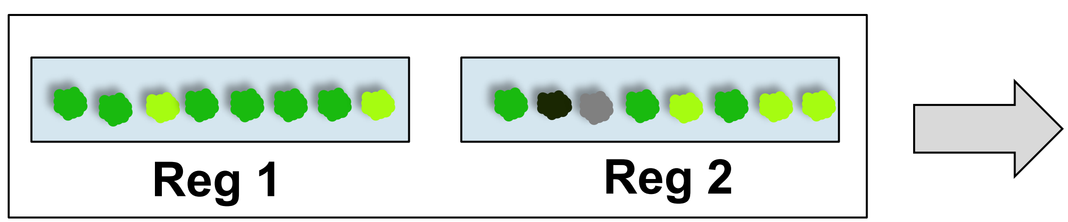


Which class of errors may be approximable?

Error Outcome Composition



Dynamic Instruction A



No resiliency protection

Is not approximable

Outline

- Introduction
- (gem5-) Approxilyzer interface & techniques
- gem5-Approxilyzer Use Cases
- **Results**
- Conclusion

Effectiveness of Error Pruning Techniques

How effective are the error pruning techniques in reducing error injections?

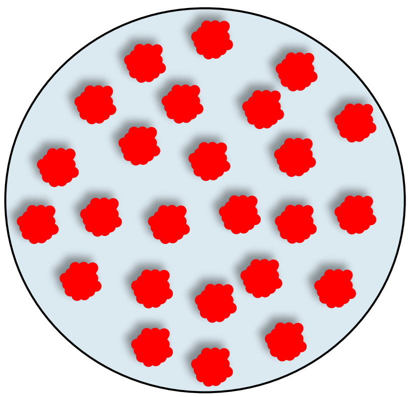
Application	Error Sites		Pruned Error Sites (%)
	Total (Before Pruning)	**Remaining (After Pruning)**	
Blackscholes	232K	100K	**56.77 %**
Swaptions	10,300K	720K	**93.01 %**
LU	1,200K	268K	**77.91 %**
FFT	4,400K	215K	**95.05 %**
Sobel	85,300K	300K	**99.65 %**

Up to two orders of magnitude reduction in error injections

Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

Equivalence class (EC)

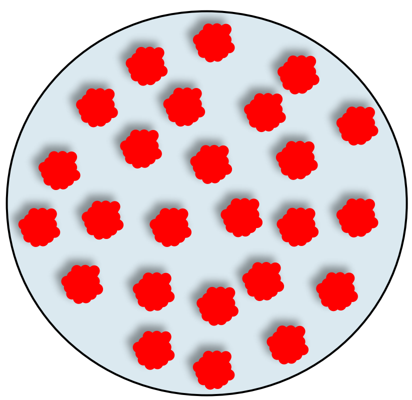


 **Pilot (Representative error-site from EC)**

Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

Equivalence class (EC)

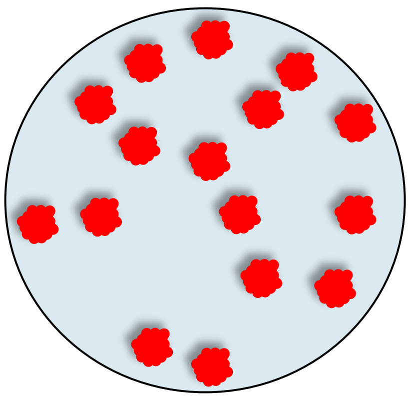


 **Pilot = SDC-Maybe with quality Q**

Validation of Equivalence Heuristics

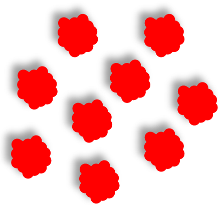
- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

Equivalence class (EC)



Pilot = SDC-Maybe with quality Q

Population (Random sample of error-sites from EC)

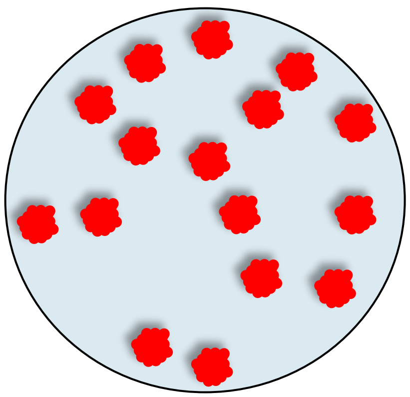


Statistical sample - 99% confidence

Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

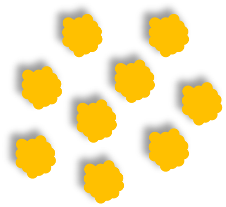
Equivalence class (EC)



Pilot = SDC-Maybe with quality Q

100% validation accuracy

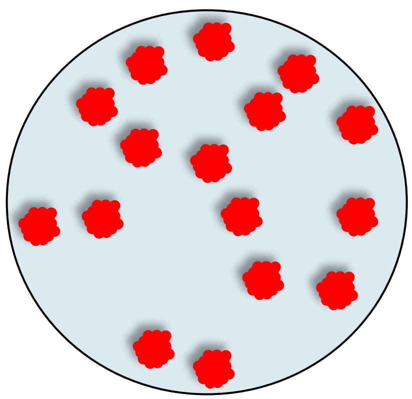
All in Population = SDC-Maybe with quality Q



Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

Equivalence class (EC)



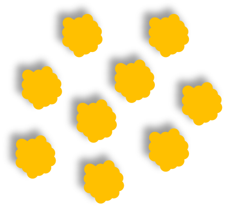
Pilot = SDC-Maybe with quality Q

⇩

100% validation accuracy

⇧

All in Population = SDC-Maybe with quality Q +/- δ

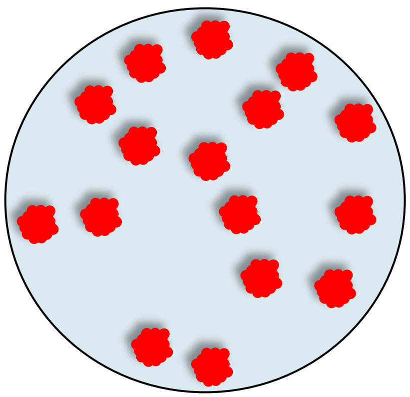


Flexible quality parameter → δ = 2%

Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

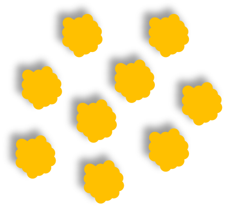
Equivalence class (EC)



Pilot = Approximable

100% validation accuracy

All in Population = Approximable



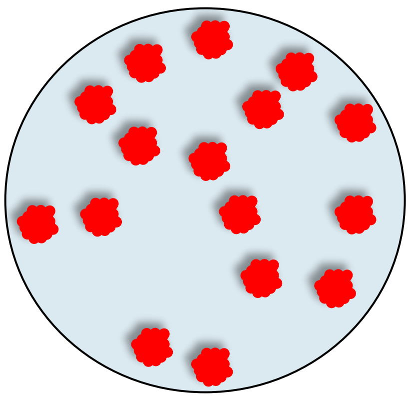
Flexible quality parameter → δ = 2%

Quality threshold = 5%

Validation of Equivalence Heuristics

- **Heuristics used to build equivalence classes need validation**
  - **Does the pilot accurately represent its equivalence class?**

Equivalence class (EC)



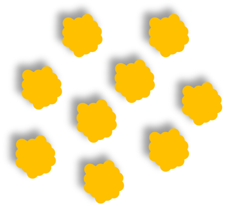
Pilot = Approximable

⇩

100% validation accuracy

⇧

All in Population = Approximable



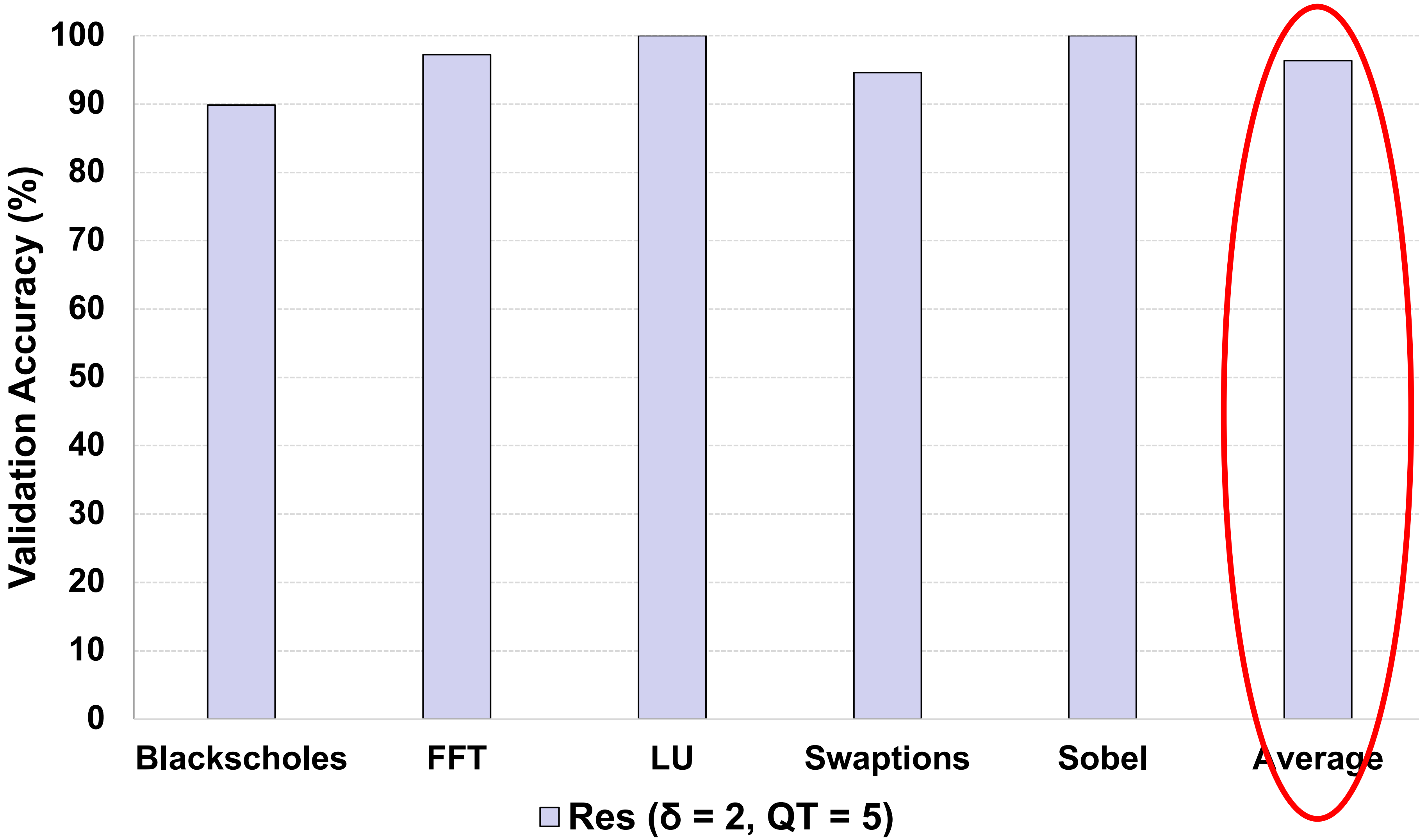
Flexible quality parameter → δ = 2%

Quality threshold = 5%

Equivalence classes selected at random (99% confidence) for validation

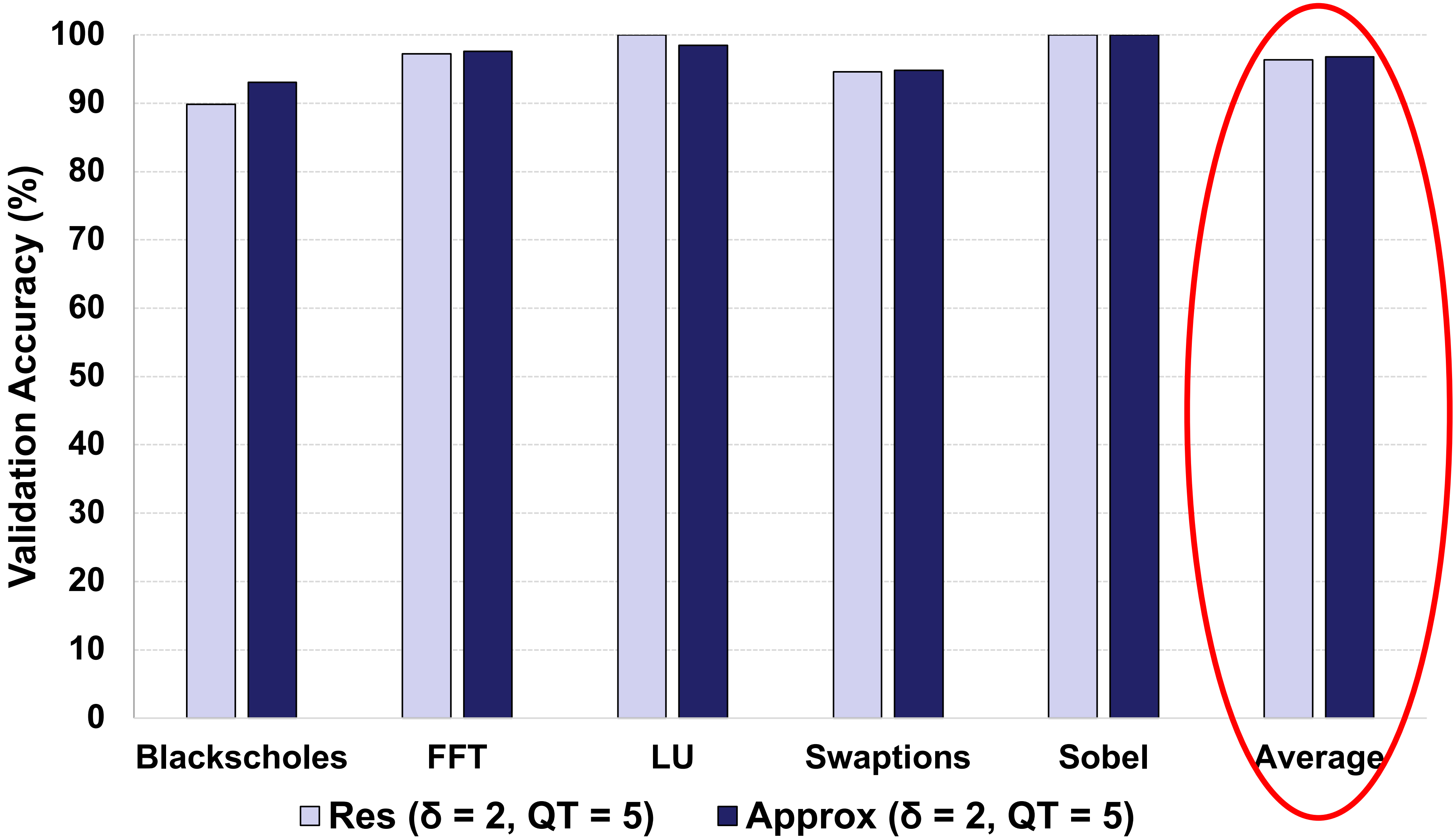
~ 1.6 million error injections

Accuracy of equivalence class heuristics



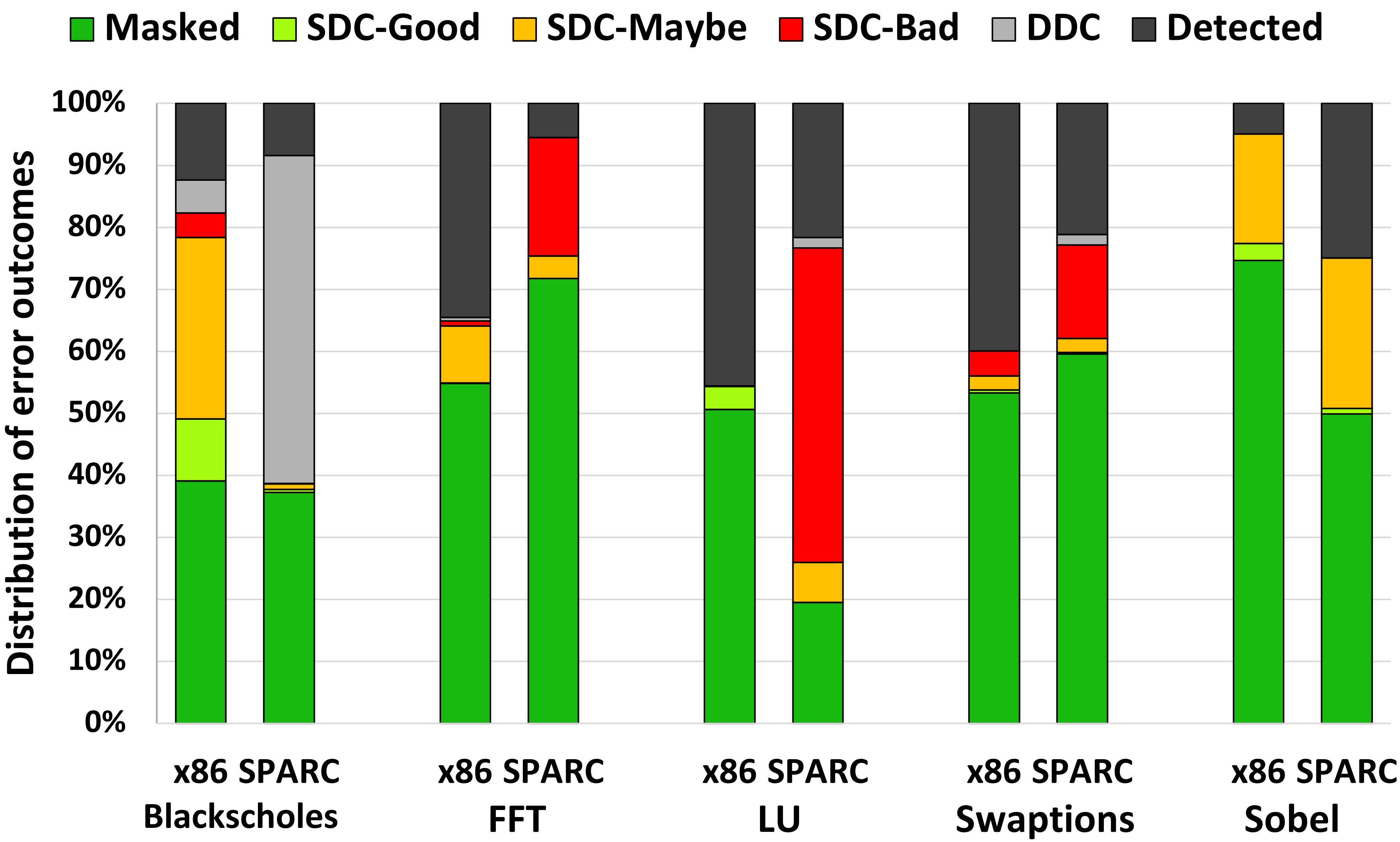
High validation accuracy (97% on average) across workloads

Accuracy of equivalence class heuristics



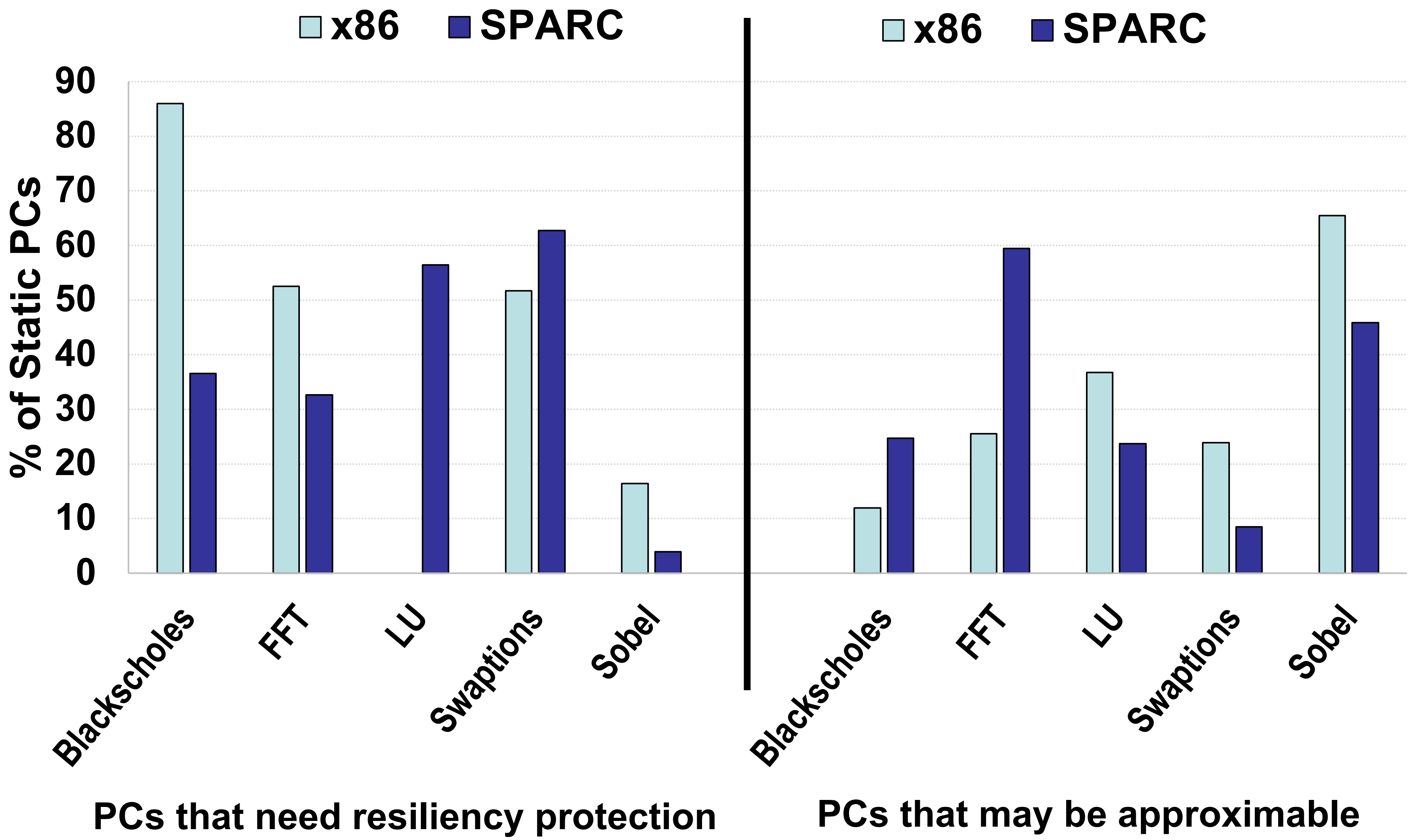
High validation accuracy (97% on average) across workloads

Error Outcomes profiles for different ISAs



Error profiles for the same workloads vary significantly for x86 vs SPARC

Comparison of static PCs by use case



Instructions for resiliency protection and approximation vary based on ISA

Conclusions

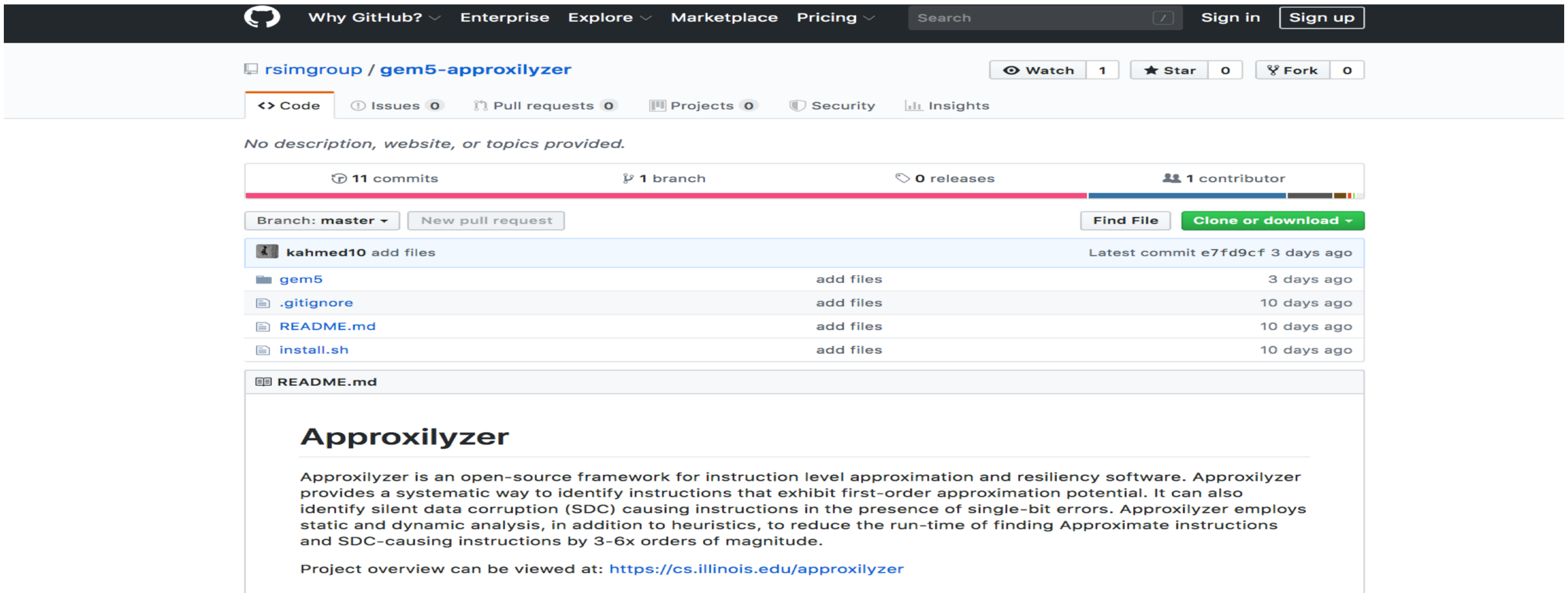
- gem5-Approxilyzer: Open-source tool for comprehensive error analysis
  - *https://github.com/rsimgroup/gem5-Approxilyzer*
  - Built using open-source gem5 simulator
  - Support for x86; can be extended to other ISAs

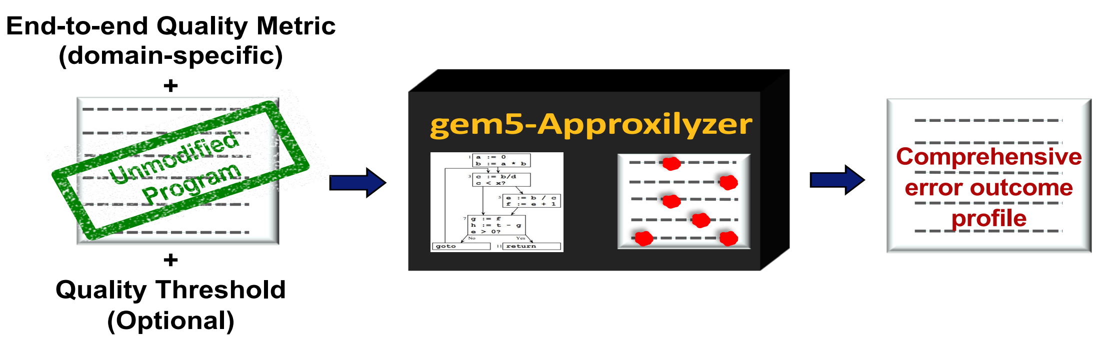
- Show effectiveness/accuracy of Approxilyzer technique for x86
  - Two orders of magnitude reduction in error injections
  - Determines error outcomes with high accuracy (97% on average)

- Error Analysis of application under different ISAs (x86 and SPARC)
  - Error profile of same application significantly different under different ISAs
  - Static instructions that are approximable/resilient vary significantly by ISA

gem5-Approxilyzer

https://github.com/rsimgroup/gem5-Approxilyzer





Backup Slides

gem5-Approxilyzer end-to-end flow

App

1 Application compiled to x86 --> Generate simplified dynamic gem5 trace

2 Error site pruning --> Generate list of pilots for error injection

3 Perform error injections --> Calculate error outcome category and quality

4 Assign outcome to population --> Generate error outcome profile

Gem5-Approxilyzer Pruning by Technique

TABLE I

BENCHMARKS, INPUTS, AND ERROR-SITE PRUNING BY TECHNIQUE (C: CONTROL-EQUIVALENCE, S: STORE-EQUIVALENCE, C+S+K: TOTAL PRUNING USING CONTROL, STORE, AND KNOWN-OUTCOME TECHNIQUES)

Application	**Input**	**Error Sites**		**Pruned Error Sites (%)**
		Total	**Remaining**	
Black-scholes [37]	21 options	232K	100K	C: 12.24 S: 9.45 C+S+K: 56.77
Swaptions [37]	1 option 1 simulation	10.3M	720K	C: 52.47 S: 7.85 C+S+K: 93.01
LU [38]	16x16 matrix 8x8 blocks	1.2M	268K	C: 23.49 S: 22.72 C+S+K: 77.91
FFT [38]	2^{20} data points	4.4M	215K	C: 43.99 S: 21.50 C+S+K: 95.05
Sobel [3]	81x121 pixels	85.3M	300K	C: 62.74 S: 20.94 C+S+K: 99.65