A Data-Parallel, Hardware-Accelerated Monte Carlo Framework for Quantifying Risk using Probabilistic Circuits

Arjun Earthperson

Department of Nuclear Engineering
North Carolina State University

August 4, 2025

Education

- **MS, Nuclear Engineering**, North Carolina State University (2023)
  Thesis: "Integrating Dual Error Propagation into Dynamic Event Trees to Support Fission Battery Probabilistic Risk Assessments"
- **BS, Electrical Engineering**, University of California, Los Angeles (2017)
  Capstone: "Integer Hardware Optimizations on TI-C6000 DSPs for Low-Power IoT Applications"

Work Experience

- **Intern**, Idaho National Laboratory (Summer 2021)
  Project: Coupled OpenPRA's OpenEPL engine with EMRALD.
- **Programmer**, The B. John Garrick Institute for the Risk Sciences, UCLA (2018–2020)
  Developed the Hybrid Causal Logic and Phoenix human reliability assessment web applications.

Awards

- **1st Place Graduate Student Winner**, ASME SERAD Student Safety Innovation Challenge (2023)
  Paper: Introducing OpenPRA: A Web-Based Framework for Collaborative Probabilistic Risk Assessment
- **Graduate Merit Award**, College of Engineering, NCSU (Summer 2025)

Software

[1] *a-earthperson/mcSCRAM*. Data-Parallel Monte Carlo probability estimation, forked from SCRAM. URL: `https://github.com/a-earthperson/mcSCRAM`.

[2] *openpra-org/inverse-canopy*. Tensorflow-based library for learning PRA model parameters. URL: `https://github.com/openpra-org/inverse-canopy`.

[3] *openpra-org/monorepo*. End-to-end PRA modeling and quantifcation solution. URL: `https://github.com/openpra-org/openpra-monorepo`.

[4] *openpra-org/openepl-engine*. Probablistic model checking on Dual-Error Propagation Models. URL: `https://github.com/openpra-org/openepl-engine`.

[5] *openpra-org/PRAcciolini*. PRA model meta-conversion utility. URL: `https://github.com/openpra-org/PRAcciolini`.

Journal Articles

[1] **Arjun Earthperson**, Mihai A. Diaconeasa. "Integrating Commercial-Off-The-Shelf Components into Radiation-Hardened Drone Designs for Nuclear-Contaminated Search and Rescue Missions". en. In: *Drones* 7.8 (2023). Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 528. ISSN: 2504-446X. DOI: `10.3390/drones7080528`. URL: `https://www.mdpi.com/2504-446X/7/8/528` (visited on 12/03/2024).

[2] **Arjun Earthperson**, Courtney M. Otani, Daniel Nevius, Steven R. Prescott, Mihai A. Diaconeasa. "A combined strategy for dynamic probabilistic risk assessment of fission battery designs using EMRALD and DEPM". en. In: *Progress in Nuclear Energy* 160 (2023), p. 104673. ISSN: 0149-1970. DOI: `10.1016/j.pnucene.2023.104673`. URL: `https://www.sciencedirect.com/science/article/pii/S0149197023001087` (visited on 03/30/2023).

[3] Elaheh Rabiei, Lixian Huang, Hao-Yu Chien, **Arjun Earthperson**, Mihai A Diaconeasa, Jason Woo, Subramanian Iyer, Mark White, Ali Mosleh. "Method and software platform for electronic COTS parts reliability estimation in space applications". en. In: *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability* (2021), p. 1748006X2199823. ISSN: 1748-006X, 1748-0078. DOI: `10.1177/1748006X21998231`. URL: `http://journals.sagepub.com/doi/10.1177/1748006X21998231` (visited on 03/25/2021).

Conference Papers

[1] Egemen Aras, **Arjun Earthperson**, Hasibul Rasheeq, Asmaa Salem Farag, Stephen Wood, Jordan Boyce, Mihai Diaconeasa. "Introducing OpenPRA's Quantification Engine: Exploring Capabilities, Recognizing Limitations, and Charting the Path to Enhancement". In: Las Vegas, NV: American Nuclear Society, 2024. DOI: `doi.org/10.13182/T130-43362`.

[2] **Arjun Earthperson**, Egemen Aras, Asmaa Salem Farag, Mihai Diaconeasa. "Towards a Deep-Learning based Heuristic for Optimal Variable Ordering in Binary Decision Diagrams to Support Fault Tree Analysis". In: Las Vegas, NV: American Nuclear Society, 2024.

[3] **Arjun Earthperson**, Priyanka Pandit, Mihai Diaconeasa. "Implementing Multiple Control Paths in the Dual Error Propagation Graph for Stochastic Failure Analysis of Digital Instrumentation and Control Systems". In: Las Vegas, NV: American Nuclear Society, 2024. DOI: `doi.org/10.13182/T130-43400`. URL: `https://epubs.ans.org/?a=56419`.

[4] **Arjun Earthperson**, Priyanka Pandit, Mihai Diaconeasa. "Quantifying Safety System Unavailability Margins via Inverse Estimation of Event Sequence Frequencies". In: Las Vegas, NV: American Nuclear Society, 2024.

[5] Priyanka Pandit, **Arjun Earthperson**, Mihai Diaconeasa. "Laying the Foundations for the Development of a Probabilistic Model Checking Method to Quantify Common Cause Failure Parameters in Digital Instrumentation and Control". In: Las Vegas, NV: American Nuclear Society, 2024. DOI: `doi.org/10.13182/T130-43603`.

[6] Priyanka Pandit, **Arjun Earthperson**, Mihai Diaconeasa. "Leveraging Inverse Uncertainty Quantification to Enhance the Resilience of Nuclear Power Plant Construction Duration Estimation". In: Las Vegas, NV: American Nuclear Society, 2024. DOI: `doi.org/10.13182/T130-43602`.

[7] Asmaa Salem Farag, Amir Afzali, Yasir Arafat, Matt Loszak, **Arjun Earthperson**, Mihai Diaconeasa. "Aalo Atomics' High Level Licensing Strategy". In: Las Vegas, NV: American Nuclear Society, 2024.

[8] Asmaa Salem Farag, S Wood, **Arjun Earthperson**, Egemen Aras, Jordan Boyce, Mihai Diaconeasa. "Evaluating PRA Tools for Accurate and Efficient Quantifications: A Follow-Up Benchmarking Study Including FTREX". In: Las Vegas, NV: American Nuclear Society, 2024. DOI: doi.org/10.13182/T130-43377.

[9] Egemen Aras, Asmaa Farag, **Arjun Earthperson**, Mihai Diaconeasa. "Method of Developing a SCRAM Parallel Engine for Efficient Quantification of Probabilistic Risk Assessment Models". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 134–140. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41054. URL: https://www.ans.org/pubs/proceedings/article-53973/ (visited on 04/02/2024).

[10] Egemen Aras, Asmaa Farag, **Arjun Earthperson**, Mihai Diaconeasa. "Methodology and Demonstration for Performance Analysis of a Probabilistic Risk Assessment Quantification Engine: SCRAM". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 452–459. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41053. URL: https://www.ans.org/pubs/proceedings/article-54005/ (visited on 04/02/2024).

[11] **Arjun Earthperson**, Egemen M. Aras, Asmaa S. Farag, Mihai A. Diaconeasa. "Introducing OpenPRA: A Web-Based Framework for Collaborative Probabilistic Risk Assessment". In: *Volume 13: Research Posters; Safety Engineering, Risk and Reliability Analysis*. New Orleans, Louisiana, USA: American Society of Mechanical Engineers, 2023, V013T15A014. ISBN: 978-0-7918-8770-7. DOI: 10.1115/IMECE2023-111708. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2023/87707/V013T15A014/1196307 (visited on 04/02/2024).

[12] **Arjun Earthperson**, Priyanka Pandit, Mihai Diaconeasa. "Introducing Multiple Control Paths in the Dual Error Propagation Graph for Stochastic Failure Analysis of Digital Instrumentation and Control Systems". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 209–218. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41248. URL: https://www.ans.org/pubs/proceedings/article-53981/ (visited on 04/02/2024).

[13] Asmaa Farag, Egemen Aras, **Arjun Earthperson**, S. Wood, Jordan Boyce, Mihai Diaconeasa. "Preliminary Benchmarking of SAPHSOLVE, XFTA, and SCRAM Using Synthetically Generated Fault Trees with Common Cause Failures". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 40–49. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41031. URL: https://www.ans.org/pubs/proceedings/article-53964/ (visited on 04/02/2024).

[14] Priyanka Pandit, **Arjun Earthperson**, Han Bao, Mihai Diaconeasa. "Analyzing Hardware and Software Common Cause Failures in Digital Instrumentation and Control Systems Using Dual Error Propagation Method". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 12–21. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41062. URL: https://www.ans.org/pubs/proceedings/article-53961/ (visited on 04/02/2024).

[15] Priyanka Pandit, **Arjun Earthperson**, Daniel Nevius, Mihai Diaconeasa. "Quantifying the Likelihood of Nuclear Supply Chain Shortage Risk". en. In: *18th International Probabilistic Safety Assessment and Analysis (PSA 2023)*. Knoxville, TN: American Nuclear Society, 2023, pp. 668–674. ISBN: 978-0-89448-792-7. DOI: 10.13182/PSA23-41229. URL: https://www.ans.org/pubs/proceedings/article-54029/ (visited on 04/02/2024).

[16] Molly Prins, Thomas M. O'Connell, **Arjun Earthperson**, Yahya A. Alzahrani, Mihai A. Diaconeasa. "Leveraging Probabilistic Risk Assessment and Machine Learning for Safety and Cost Optimization in HAZMAT Transportation". In: *Volume 13: Research Posters; Safety Engineering, Risk and Reliability Analysis*. New Orleans, Louisiana, USA: American Society of Mechanical Engineers, 2023, V013T15A012. ISBN: 978-0-7918-8770-7. DOI: 10.1115/IMECE2023-114273. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2023/87707/V013T15A012/1196286 (visited on 04/02/2024).

[17] Egemen M. Aras, Asmaa S. Farag, **Arjun Earthperson**, Mihai A. Diaconeasa. "Benchmark Study of XFTA and SCRAM Fault Tree Solvers Using Synthetically Generated Fault Trees Models". In: *Volume 9: Mechanics of Solids, Structures, and Fluids; Micro- and Nano-Systems Engineering and Packaging; Safety Engineering, Risk, and Reliability Analysis; Research Posters*. Columbus, Ohio, USA: American Society of Mechanical Engineers, 2022, V009T14A016. ISBN: 978-0-7918-8671-7. DOI: 10.1115/IMECE2022-95783. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2022/86717/V009T14A016/1157443 (visited on 02/15/2023).

[18] Bineh Ndefru, Karthik Sankaran, Theresa Stewart, Ali Mosleh, **Arjun Earthperson**, Natalie Zawalick. "Risk-Informed Decision-Making Tool for Covid-19 Community Behavior and Intervention Scenario Assessment". English. In: *Proceedings of the 16th International Conference on Probabilistic Safety Assessment and Management (PSAM)*. Vol. 3. Honolulu, Hawaii, USA: Curran Associates, Inc., 2022. ISBN: 978-1-71386-375-5. URL: https://www.iapsam.org/PSAM16/papers/KS45-PSAM16.pdf.

[19] Courtney Mariko Yang Hui Otani, Robby Christian, Steven R Prescott, Mihai Diaconeasa, **Arjun Earthperson**. "Probabilistic Methods for Cyclical and Coupled Systems with Changing Failure Rates". English. In: *Report Number: INL/CON-22-66839-Rev000*. Research Org.: Idaho National Lab. (INL), Idaho Falls, ID (United States), 2022. URL: https://www.osti.gov/biblio/1885929.

[20] **Arjun Earthperson**, Mihai A. Diaconeasa. "Verification Study of the Nuclear PRA for the Mars 2020 Mission Following Accidental Orbital Re-Entry". In: *Volume 13: Safety Engineering, Risk, and Reliability Analysis; Research Posters*. Virtual, Online: American Society of Mechanical Engineers, 2021, V013T14A019. ISBN: 978-0-7918-8569-7. DOI: 10.1115/IMECE2021-71359. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2021/85697/V013T14A019/1133290 (visited on 02/04/2022).

[21] Priyanka Pandit, **Arjun Earthperson**, Alp Tezbasaran, Mihai A. Diaconeasa. "A Quantitative Approach to Assess the Likelihood of Supply Chain Shortages". In: *Volume 13: Safety Engineering, Risk, and Reliability Analysis; Research Posters*. Virtual, Online: American Society of Mechanical Engineers, 2021, V013T14A023. ISBN: 978-0-7918-8569-7. DOI: 10.1115/IMECE2021-73696. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2021/85697/V013T14A023/1133260 (visited on 02/04/2022).

[22] Priyanka Pandit, Alp Tezbasaran, **Arjun Earthperson**, Mihai A. Diaconeasa. "Evaluating the Implementation of Distributed Ledger Technology for the Licensing and Regulation of Nuclear Power Plants". In: *Volume 8B: Energy*. Virtual, Online: American Society of Mechanical Engineers, 2021, V08BT08A016. ISBN: 978-0-7918-8564-2. DOI: 10.1115/IMECE2021-71730. URL: https://asmedigitalcollection.asme.org/IMECE/proceedings/IMECE2021/85642/V08BT08A016/1132953 (visited on 02/04/2022).

Research Story in One Slide

1. **Problem.** Exact PRA quantification crumbles beyond a few hundred gates; industry still waits hours–days for large models.
2. **Idea.** Treat the *entire* PRA—event trees & fault trees—as one probabilistic DAG and evaluate it via massively–parallel Monte-Carlo.
3. **Enablers.** Hardware-native voting gates, a Monte-Carlo–oriented compilation pipeline, and GPU-resident bit-packed kernels.
4. **Evidence.** Model compression; sub-percent error on $\sim 10^3$-event graphs in <5 s on a laptop GPU.
5. **Impact.** Opens path to real-time, high-fidelity risk insights and lays groundwork for dynamic & correlated extensions.

Why Large PRA Models Still Hurt

- Combinatorial Explosion. Minimal cut-set enumeration scales $\mathcal{O}(2^n)$; full-core reactor PRA now couples ~ 10^2 event trees and 10^3 fault trees ($\approx 10^5$ basic events).
- Reliance on Approximations. Truncation, bounding (min–cut upper bound) and rare–event heuristics trade rigour for tractability.
- Turn-around Times. State-of-practice tools still report **hours to days** for full quantification on commodity CPUs.

Data-Parallel Hardware is Waiting

- Consumer GPUs $> 10^{12}$ **integer ops/s** at < 100 W.
- Dedicated bit-manipulation units (e.g. popcount, VNNI) ideal for Boolean evaluation.
- PRA logic is embarrassingly parallel yet *under-utilizes* this silicon.

Device	**Integer Ops/s**
NVIDIA A100 GPU	$\sim 2 \times 10^{12}$
RTX 3060 Laptop	$\sim 3 \times 10^{11}$
8-core CPU	$\sim 8 \times 10^{8}$
Single CPU core	$\sim 1 \times 10^{8}$

Peak 32-bit integer throughput (vendor specs)

Key Insight

PRA probability estimation is analogous to forward inference in a feed-forward neural network :: same hardware, different algebra.

Research Questions

1. How can **all** event– and fault–tree dependencies be embedded in a single probabilistic DAG amenable to high-throughput evaluation?
2. Which **data-parallel Monte-Carlo algorithms** best leverage modern GPUs for Boolean circuits with $\geq 10^5$ nodes?
3. What compilation and data-structure transformations minimize *graph size* and *arithmetic intensity* without breaking Monte-Carlo unbiasedness?
4. How can convergence be certified in real time for probabilities spanning 10^{-0}–10^{-8}?
5. What extensions—variance reduction, discrete-event simulation—are needed for ultra-rare events and time-dependent risk?

This Dissertation Contributes

1. **Unified Risk Graph.** Formalized PRA models as probabilistic circuits, prove semantic equivalence.
2. **Hardware–Native Gate Set.** Population-count kernels for *k*-of-*n* and cardinality gates :: exponential graph compression.
3. **MC-Oriented Knowledge Compilation.** Optimizes PDAGs for throughput.
4. **Bit-Parallel Monte-Carlo Engine.** SYCL kernels achieving massive parallelism.
5. **Rigorous Convergence Criteria.** Multiobjective, with formal error bounds.
6. **Domain Extensions.** Common-cause failures, importance measures, and an importance-sampling prototype for rare events.
7. **Open-Source Release & Benchmarks.** Reproducible evaluation on 43 Aralia models; code under permissive license.

Research Objective 1

Compile PRA Models into Probabilistic Circuits

Objective 1 — Unifying Risk Logic as a Probabilistic DAG

- Map event–tree branching and fault–tree gates into a single **probabilistic directed acyclic graph (PDAG)**.
- Retain exact Boolean semantics while exposing hardware-native operations (AND/OR, *k*-of-*n*, XOR).
- Provide a substrate for compilation, bit-parallel evaluation, and future dynamic extensions.

The Triplet Definition of Risk

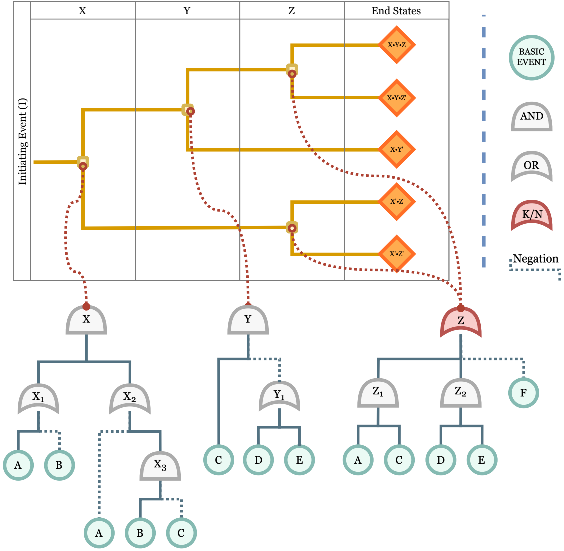
- Define risk as a set of triplets, each representing:
  1. What can go wrong? (S_i)
  2. How likely is it to happen? (L_i)
  3. What are the consequences? (X_i)

- $$R = \{\langle S_i, L_i, X_i \rangle\}_c, \tag{1}$$

  c represents completeness in enumerating *all* relevant scenarios.

Scenario S_i Modeling in PRA

- Each scenario unfolds from initiating events (IEs), followed by conditional branching events.
- Fundamental goal: assign probabilities to these scenarios and assess resulting outcomes (e.g., core damage, large release).
- Implementation typically uses structured diagrams such as:
  - Event Trees (ETs): forward chaining from IE to various end states.
  - Fault Trees (FTs): top-down decomposition to basic events (component failures).



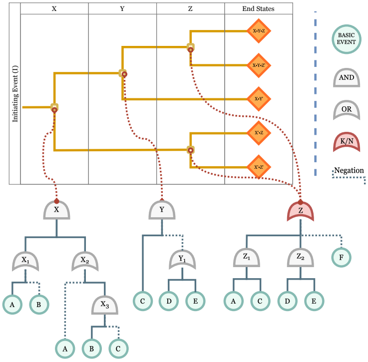
Variable	Expression
X	$(A \vert B') \bullet (A' \vert (B \bullet C'))$
Y	$C \bullet (D \vert E)'$
Z	$kn[(A \bullet C), (D \bullet E), F']$

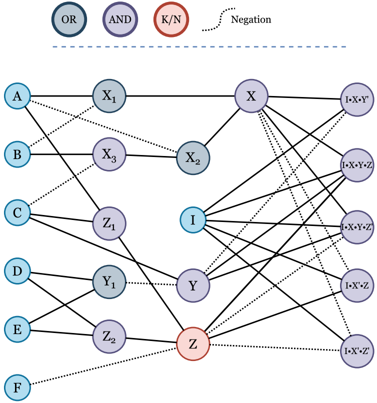
Table: Unsimplified Boolean expression for each Top Event

Small, but non-trivial structure:

- Basic events are shared.
- Some gate outputs are negated.
- Event Z is a (k=2) of n=3 gate.

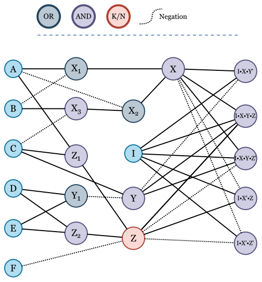
Compile a Directed Acylic Graph (DAG) from Logic Model





Compile **once**, evaluate millions of times:

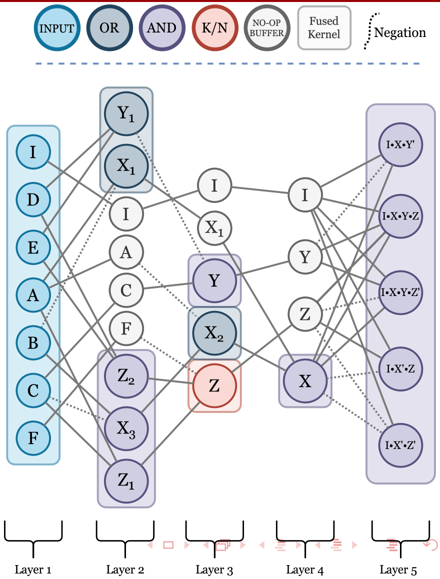
- Layered topological order for memory coalescing.
- Preserve *k*-of-*n* gates to avoid exponential blow-up.
- Replace linear scans with hash-indexed containers.



Refinements:

- Partition into layers.
- Vectorize for SIMD by fusing similar ops.
- Feed-forward only: improves caching.

Still not probabilistic: inputs, outputs are bitpacked INT64 tensors.



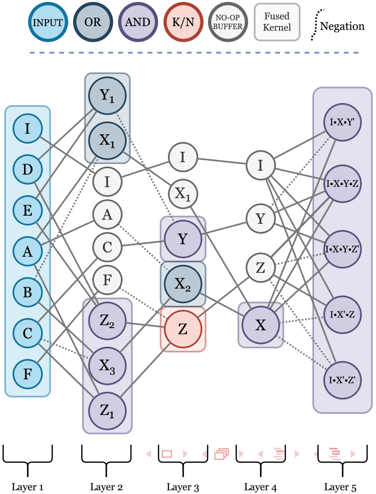
Querying the Compiled Knowledge Graph

The Simplest Type of Query: Eval(G)

- Set the inputs [on/off].
- Observe the outputs [on/off].

Can be used as a building block for an embedding ML model.

But just how fast is this?



From Logic to High-Throughput Evaluation

Next: How do we process **millions of scenarios per second**?

Hardware-native kernels :: *Research Objective 2*

Research Objective 2

Develop data-parallel methods for evaluating Boolean circuits

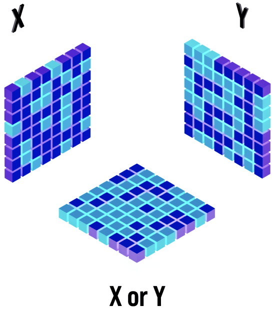
Boolean Truth Table – Single Bit

X	Y	AND	OR	XOR	NAND	NOR	XNOR
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	0
1	0	0	1	1	1	0	0
1	1	1	1	0	0	0	1

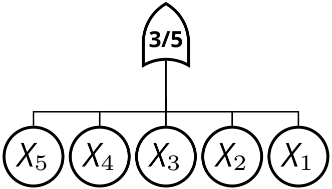
- Classical gate evaluation operates *bit-by-bit*. Throughput $\propto$ number of Boolean operations.
- Perform **64** of these truth-table lookups in one machine instruction.

Extending to a 64-Bit Word

- Pack 64 independent Bernoulli trials into one byte.
- Bitwise primitives act *independently* on every bit position.
- Hardware native instructions guarantee ns latency.



What is a *k*-of-*n* (Voting) Gate?

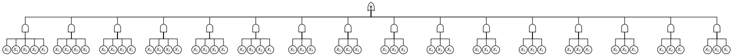


Example: $k = 3$, $n = 5$

- Outputs 1 iff at least *k* of *n* inputs are 1 (majority / threshold logic).
- Classic PRA models expand this gate into basic AND/OR primitives –> leads to combinatorial blow-up.

Naïve Expansion => Combinatorial Explosion

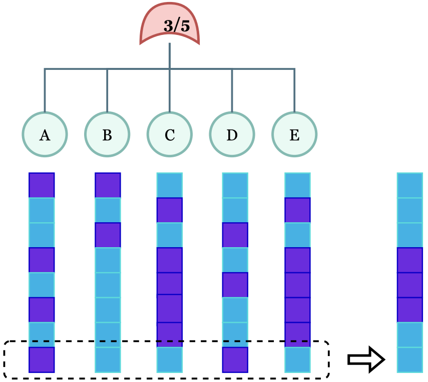
- Expansion = OR of every subset with $k, \ldots, n$ true inputs.
- For $n = 5$, $k = 3$: $\binom{5}{3} = 10$ conjunction clauses => 26 total gates after binary-tree lowering.
- Complexity becomes $\Theta(2^n/\sqrt{n})$ at $k \approx n/2$.



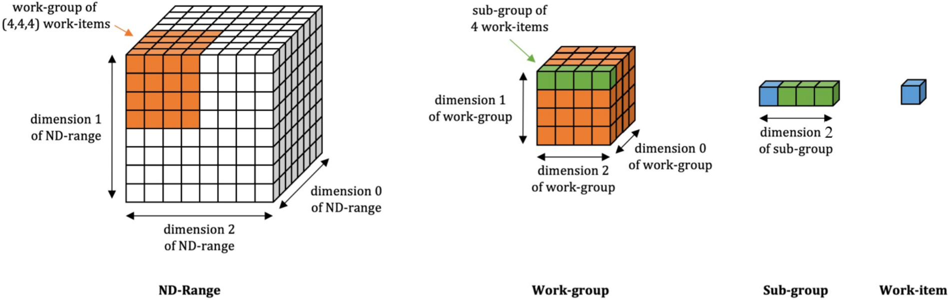
3-of-5 gate expanded to AND/OR Sum-of-Products

Hardware-Native Voting Gate (No Expansion)

- Preserve the gate as one vertex; kernel does bit-parallel population count.
- Complexity $\mathcal{O}(n)$ integer ops; counter width ≤ 8 bits for PRA fan-ins (256 inputs).
- Graph shrinks from $\Theta(2^n/\sqrt{n})$ to **1**. Huge memory launch savings.

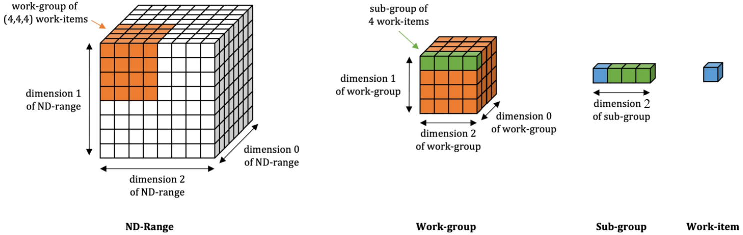


SYCL Execution Model in a Nutshell



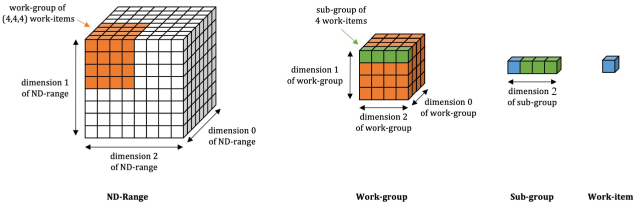
SYCL Execution Model in a Nutshell

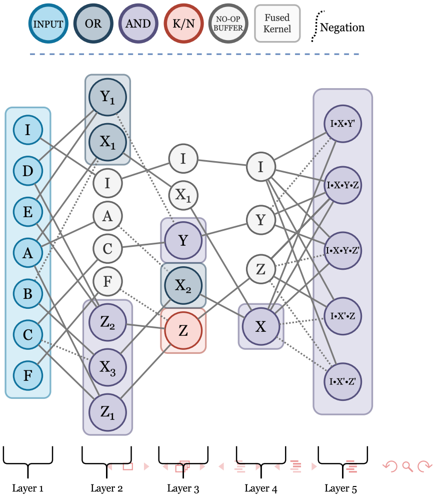
Host	submits `queue.submit()` with a `kernel_name`.
ND-Range	⟨ global $3 \times$ local ⟩ defines grid.
Work-Group	maps to CUDA block OpenCL work-group.
Sub-Group	(warp/wavefront) gives warp-level shuffle & ballot ops.
Device USM	used for persistent bit-packed buffers.



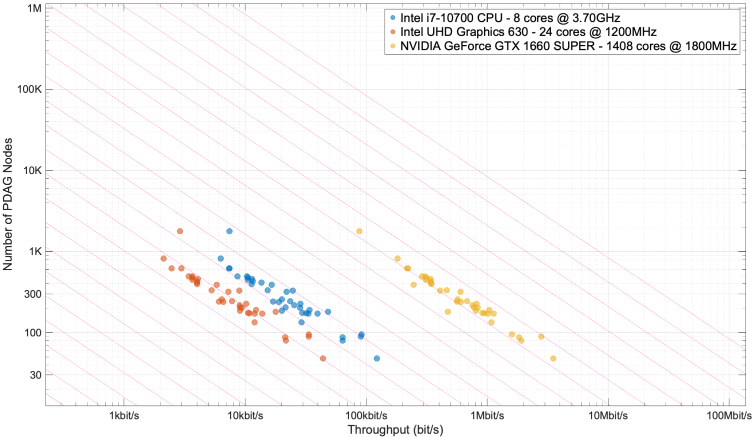
Mapping PDAG Layers to SYCL Kernels

1. Topological sort $\Rightarrow$ depth index d.
2. All nodes with depth d share *identical fan-in length*. `range<3>` := (batch, gate, bitpack).
3. One kernel per layer; gate type dispatched via template specialization.
4. Streams results to next-depth buffer in global memory.





Eval Query Performance on Generic Backends



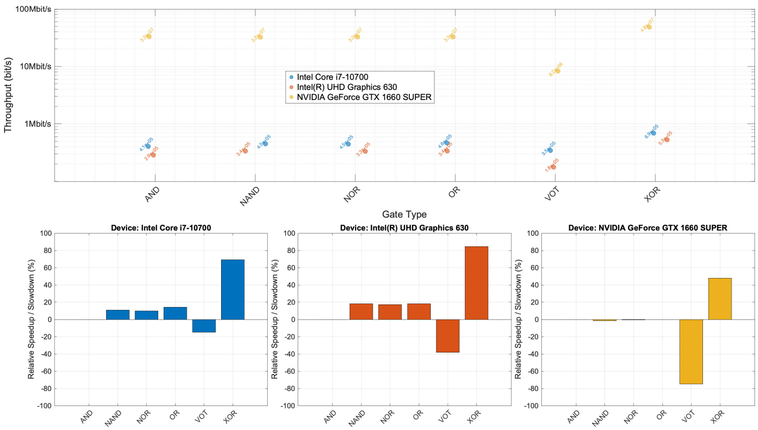
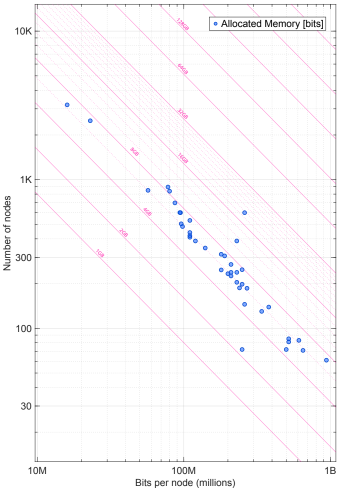


Figure: (Top) Throughput in bit/second on various backends for different gate types. (Bottom) % Relative speedup/slowdown as compared to the AND gate.

Eval Query Performance on Discrete GPUs

- Latency: 20-30 *ms* per layer.
- Throughput: Graph depth and VRAM bound (see plot).
- Benchmarked on Nvidia GTX 1660 [6GB].
- Graph sizes: from ≈ 50 to ≈ 2000 nodes.
- Evals: from 16M to 1B per node per pass.

Q: Are these enough samples to estimate the Expectation Query?



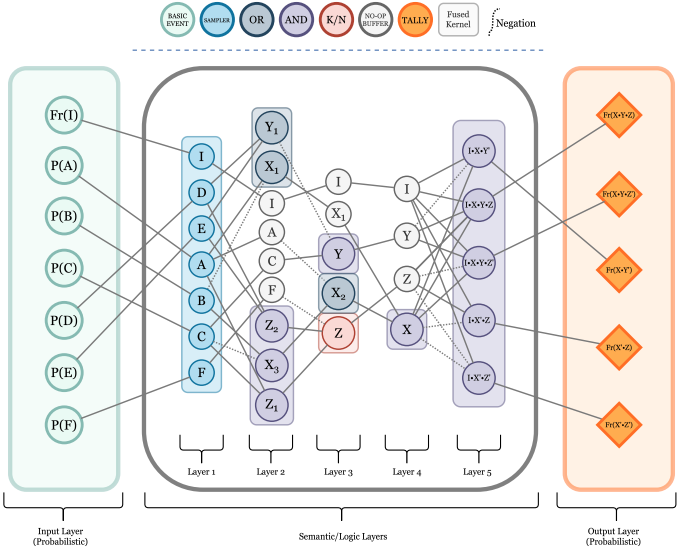
Research Objective 3

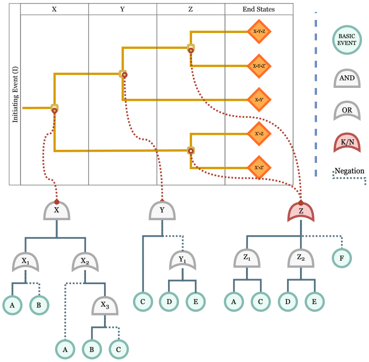
Develop data-parallel Monte-Carlo methods for estimating probabilities on unified PDAGs

Bridging Boolean Evaluation and Probabilistic Inference

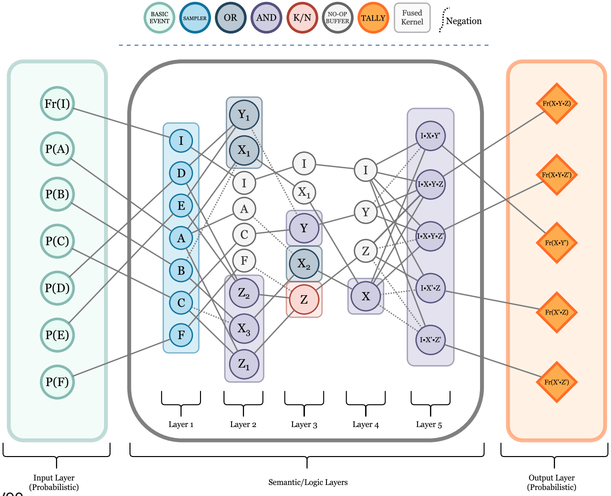
- Previous objective delivered *bit-parallel gate kernels* that evaluate a PDAG layer in $\mathcal{O}(1)$ machine words.
- To quantify risk, we must attach *probability distributions* to the PDAG leaves and propagate forward.
- Strategy: embed a Monte-Carlo engine that
  1. draws bit-packed Bernoulli samples for **all** basic events, and
  2. reuses the same gate kernels to evaluate each random draw.
- One kernel pipeline therefore suffices for **both** deterministic logic and stochastic sampling.

Back to Working Example: One Initiating Event, Three Fault Trees, Six Basic Events, Five End States





End-to-End Sampling Pipeline (one iteration)



1. **Basic-Event Kernel:** generate random draws.
2. **Gate Kernels:** evaluate PDAG layers using hardware-native logic primitives.
3. **Tally Kernel:** count number of ones, update counters, compute estimates.

Monte-Carlo Sampling over PDAGs

1. Draw $\mathbf{x}^{(i)} \sim \prod_{b \in \mathcal{B}} \mathrm{Bernoulli}(p_b)$ in **bit-packed** form.
2. Evaluate the Boolean function $F\colon \{0,1\}^{|\mathcal{B}|} \to \{0,1\}$ (root node) using the gate kernels.
3. Record $Y^{(i)} = F(\mathbf{x}^{(i)})$ (failure indicator).

After N iterations

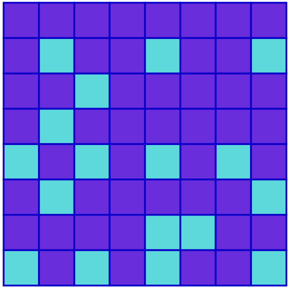
$$\widehat{P}_N = \frac{1}{N}\sum_{i=1}^{N} Y^{(i)}, \qquad \mathrm{SE}(\widehat{P}_N) = \sqrt{\frac{\widehat{P}_N(1-\widehat{P}_N)}{N}}.$$

Error shrinks as $\mathcal{O}(N^{-1/2})$; variance-reduction extensions discussed shortly.

Basic-Event Sampling Kernel

- Each basic event $b \in \mathcal{B}$ is modelled as an independent **Bernoulli** random variable $X_b \sim \mathrm{Bernoulli}(p_b)$.
- A single Monte–Carlo iteration packs $\omega = 8\,\mathrm{sizeof}(\texttt{bitpack\_t}) = 64$ independent trials into one machine word.
- For batch index b, bit–pack p, lane λ: $X_{b,p,\lambda} \sim \mathrm{Bernoulli}(p_b)$.
- The basic–event kernel uses the counter–based Philox–4×32–10 PRNG; counters are derived from (b, p, λ, t), guaranteeing reproducibility and inter–thread independence.
- After one iteration ($N = BP\omega$ trials) the sufficient statistics per basic event are $s_b = \sum_{p,\lambda} X_{b,p,\lambda}$, $\widehat{p}_b = s_b/N$,

$$\mathrm{SE}(\widehat{p}_b) = \sqrt{\widehat{p}_b(1-\widehat{p}_b)/N}.$$



P(A) = 0.25; p(A) = (17/64) ≈ 0.6525

Tally Kernel

- For a designated node v (e.g. system top event) define $Y^{(t)}_{p,\lambda} = F_v(\mathbf{X}^{(t)}_{\cdot,p,\lambda}) \in \{0,1\}$.
- **Tally kernel** executes a single SIMD `popcount` per bit-pack to obtain the per-iteration count $s^{(t)}_v = \sum_{p,\lambda} Y^{(t)}_{p,\lambda}$.
- Cumulative statistics after T iterations ($N_T = TN$ trials):

$$S_v = \sum_{t=1}^{T} s^{(t)}_v, \qquad \widehat{P}_v = \frac{S_v}{N_T}.$$

- Unbiased variance estimator

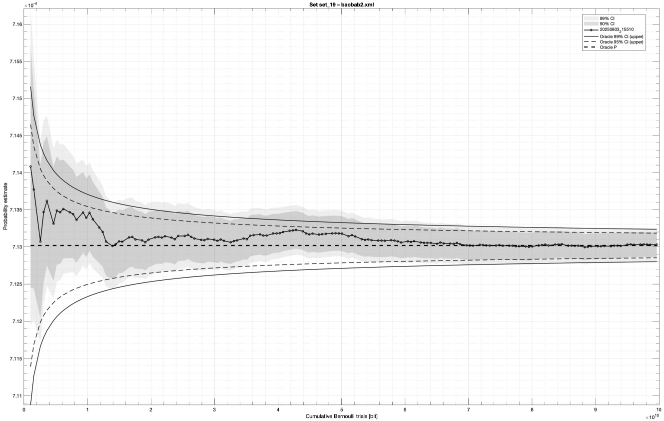
$$\widehat{\sigma}^2_v = \frac{\widehat{P}_v(1-\widehat{P}_v)}{N_T}.$$

- **Central Limit Theorem.** As $T \to \infty$

$$\frac{\widehat{P}_v - P_v}{\widehat{\sigma}_v} \xrightarrow{\mathcal{D}} \mathcal{N}(0,1).$$

When "More Samples" is Wasteful

- Fixed iteration budgets (N large and hard-wired) risk *massive* oversampling when P_v is moderate ($\approx 10^{-4}$–10^{-1}).



Research Objective 4

Develop Robust Convergence Criteria

Criterion 1: Frequentist Half-Width (Wald)

Let $\{Y^{(i)}\}_{i=1}^{N}$ be i.i.d. Bernoulli trials with $P = \Pr[Y = 1]$. The estimator $\widehat{P}_N = \frac{1}{N}\sum_i Y^{(i)}$ obeys the CLT:

$$\sqrt{N}\left(\widehat{P}_N - P\right) \xrightarrow{d} \mathcal{N}\big(0, P(1-P)\big).$$

A $(1-\alpha)$ two–sided **Wald** interval is therefore

$$\widehat{P}_N \pm z_{1-\alpha/2}\sqrt{\frac{\widehat{P}_N(1-\widehat{P}_N)}{N}}.$$

Stopping rule

Half-width: $h_N^{\text{lin}}(z) = z\sqrt{\widehat{P}_N(1-\widehat{P}_N)/N}$. Declare convergence when $h_N^{\text{lin}}/\widehat{P}_N \le \varepsilon_{\text{rel}}$.

Intuition

Shrinks a *relative* confidence band around the estimate; fast for moderate P, slow for rare events where $\widehat{P}_N \ll 10^{-4}$.

Criterion 2: Bayesian Credible Interval (Jeffreys prior)

Prior $p \sim \text{Beta}\left(\frac{1}{2}, \frac{1}{2}\right)$ is invariant under re-parameterization. After s successes and f failures the posterior is

$$p \mid \text{data} \sim \text{Beta}\left(s + \tfrac{1}{2}, f + \tfrac{1}{2}\right).$$

The central $(1 - \alpha)$ credible interval $[q_t, q_{1-t}]$ with $t = \alpha/2$ has

Stopping rule

half-width $h_N^{\text{Bayes}} = (q_{1-t} - q_t)/2$. Convergence when $h_N^{\text{Bayes}} / \widehat{P}_N \leq \varepsilon_{\text{rel}}^{\text{Bayes}}$.

Intuition

Integrates parameter uncertainty; maintains correct coverage even when $\widehat{P}_N$ is based on only a handful of observed failures (rare-event tails).

Criterion 3: Information-Theoretic Gain

Posterior entropy of Beta(α, β) is

$$H(\alpha, \beta) = \ln B(\alpha, \beta) - (\alpha - 1)\psi(\alpha) - (\beta - 1)\psi(\beta) + (\alpha + \beta - 2)\psi(\alpha + \beta).$$

After a batch $(\Delta s, \Delta f)$ the **information gain** is

$$I_{\text{batch}} = H(\alpha, \beta) - H(\alpha + \Delta s,\ \beta + \Delta f).$$

Stopping rule

Stop when $I_{\text{batch}} < I_{\min}$ bits (default 10^{-4}).

Intuition

Scale-free; halts when each new batch conveys negligible Shannon information, preventing oversampling when P is either very small or very large.

Composite Convergence Criteria

Step–1: Statistical precision per node v

$$N_{\text{req}}^{(v)} = \max\left(N_{\varepsilon}^{(v)},\ N_{\log}^{(v)},\ N_{\text{Bayes}}^{(v)},\ N_{\text{info}}^{(v)}\right)$$

where

$$N_{\varepsilon}^{(v)} := \left\lceil \frac{z^2\,\widehat{P}_v(1-\widehat{P}_v)}{(\varepsilon_{\text{rel}}\widehat{P}_v)^2} \right\rceil,$$

$$N_{\log}^{(v)} := \left\lceil \frac{z^2(1-\widehat{P}_v)}{(\varepsilon^{\log}\ \ln 10)^2\widehat{P}_v} \right\rceil,$$

$$N_{\text{Bayes}}^{(v)} := \text{Eq. (Jeffreys)},$$

$$N_{\text{info}}^{(v)} := \left\lceil (2\ln 2) I_{\min}^{-1} \right\rceil.$$

Composite Stopping Rule

Step–2: Global precision budget

$$N_{\text{req}} = \max_{v \in \mathcal{V}} N_{\text{req}}^{(v)}.$$

Step–3: Translate to iteration budget

$$T_\varepsilon = \left\lceil \frac{N_{\text{req}}}{N} \right\rceil, \qquad N = BP\omega.$$

Step–4: External user limits

$$T_{\max} \text{ (iteration cap)}, \qquad \tau_{\max} \text{ (wall-clock cap)}.$$

Stopping time

$$T = \min\{\, T_\varepsilon,\ T_{\max},\ T_\tau \,\}, \qquad T_\tau = \min\{t : \ \tau(t) \geq \tau_{\max}\}.$$

Intuition

- Statistical criteria guarantee *precision*. T_ε dominates when resources are ample.
- Rule is conservative: run halts *as soon as any* limit is met.

On-the-Fly Diagnostics & Accuracy Metrics

Additional diagnostics when `true P` is known (for debugging).

Metric	Formula
Absolute error	$\Delta_v = \lvert\widehat{P}_v - P_v\rvert$
Relative error	$\delta_v = \Delta_v / P_v$
Mean-squared error	$\mathrm{MSE}_v = (\widehat{P}_v - P_v)^2$
Linear half-width	$h_v^{\text{lin}} = z\sqrt{\widehat{P}_v(1-\widehat{P}_v)/N}$
Log half-width	$h_v^{\log} = \dfrac{z\widehat{\sigma}_v}{\widehat{P}_v \ln 10}$
Bayesian half-width	$h_v^{\text{Bayes}} = \dfrac{q_{1-t} - q_t}{2}$
Information gain	$I_{\text{batch}} = H(\alpha, \beta) - H(\alpha + \Delta s,\ \beta + \Delta f)$
z–score	$z_v = \dfrac{\widehat{P}_v - P_v}{\widehat{\sigma}_v}$
χ^2 goodness-of-fit	$\chi_v^2 = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_0 - E_0)^2}{E_0}$

Research Objective 5

Case Studies and Benchmarks

Overview: Aralia Dataset

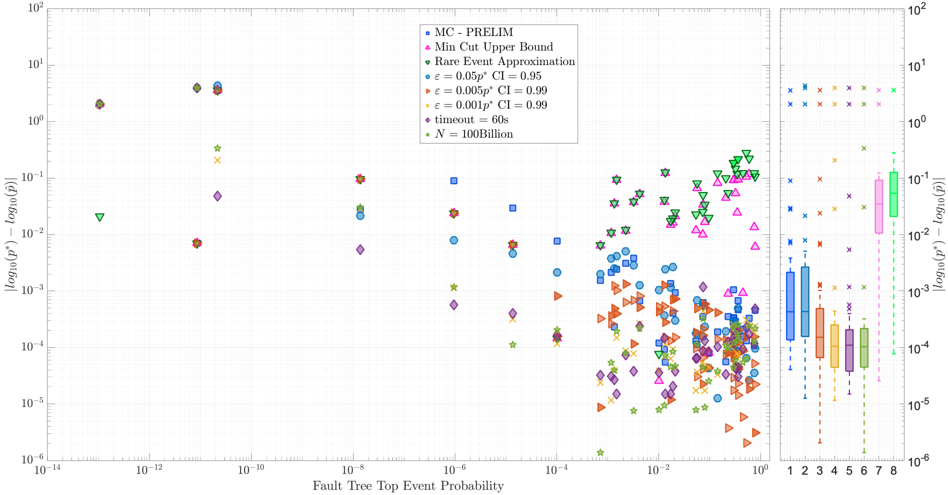
- **Dataset Composition:** The Aralia collection consists of 43 distinct fault trees, each with varying numbers of basic events (BEs), gate types (AND, OR, K/N, XOR), and minimal cut-set counts.
- **Diverse Problem Sizes:** Small trees (e.g. 25–32 BEs) through large models with over 1,500 BEs.
- **Wide Probability Range:** Top-event probabilities spanning from rare events near 10^{-13} to fairly likely failures with probability above 0.7.
- **Model Variability:** Some trees are primarily AND/OR, others incorporate more advanced gates (K/N, XOR, NOT), providing thorough coverage of typical (and atypical) fault tree logic structures.

#	Fault Tree	Basic Events	Logic Gates					Minimal Cut Sets	Top Event Probability
			Total	AND	K/N	XOR	NOT		
1	baobab1	61	84	16	9	-	-	46,188	1.01708E-04
2	baobab2	32	40	5	6	-	-	4,805	7.13018E-04
3	baobab3	80	107	46	-	-	-	24,386	2.24117E-03
4	cea9601	186	201	69	8	-	30	130,281,976	1.48409E-03
5	chinese	25	36	13	-	-	-	392	1.17058E-03
6	das9201	122	82	19	-	-	-	14,217	1.34237E-02
7	das9202	49	36	10	-	-	-	27,778	1.01154E-02
8	das9203	51	30	1	-	-	-	16,200	1.34880E-03
9	das9204	53	30	12	-	-	-	16,704	6.07651E-08
10	das9205	51	20	2	-	-	-	17,280	1.38408E-08
11	das9206	121	112	21	-	-	-	19,518	2.29687E-01
12	das9207	276	324	59	-	-	-	25,988	3.46696E-01
13	das9208	103	145	33	-	-	-	8,060	1.30179E-02
14	das9209	109	73	18	-	-	-	8.20E+10	1.05800E-13
15	das9601	122	288	60	36	12	14	4,259	4.23440E-03
16	das9701	267	2,226	1,739	-	-	992	26,299,506	7.44694E-02
17	edf9201	183	132	12	-	-	-	579,720	3.24591E-01
18	edf9202	458	435	45	-	-	-	130,112	7.81302E-01
19	edf9203	362	475	117	-	-	-	20,807,446	5.99589E-01
20	edf9204	323	375	106	-	-	-	32,580,630	5.25374E-01
21	edf9205	165	142	30	-	-	-	21,308	2.09351E-01

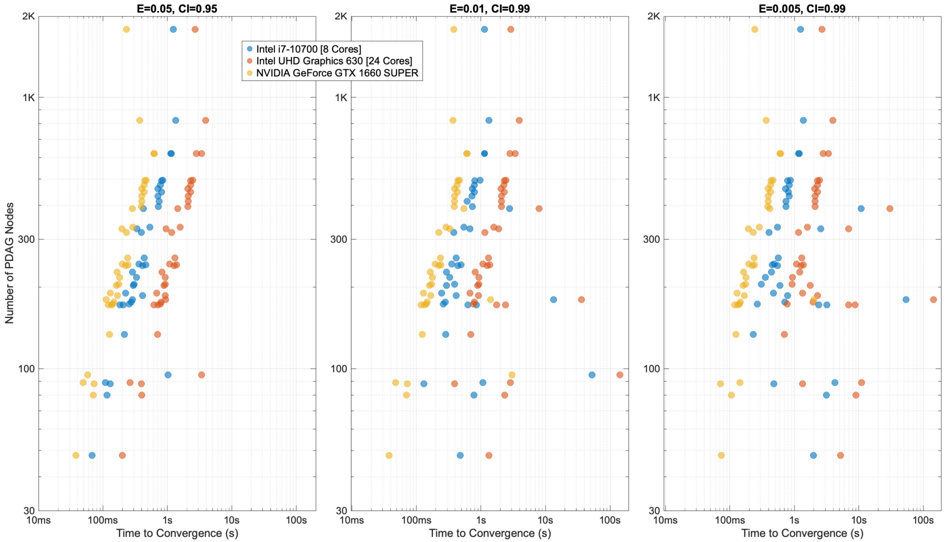
22	edf9206	240	362	126	-	-	-	385,825,320	8.61500E-12
23	edfpa14b	311	290	70	-	-	-	105,955,422	2.95620E-01
24	edfpa14o	311	173	42	-	-	-	105,927,244	2.97057E-01
25	edfpa14p	124	101	42	-	-	-	415,500	8.07059E-02
26	edfpa14q	311	194	55	-	-	-	105,950,670	2.95905E-01
27	edfpa14r	106	132	55	-	-	-	380,412	2.09977E-02
28	edfpa15b	283	249	61	-	-	-	2,910,473	3.62737E-01
29	edfpa15o	283	138	33	-	-	-	2,906,753	3.62956E-01
30	edfpa15p	276	324	33	-	-	-	27,870	7.36302E-02
31	edfpa15q	283	158	45	-	-	-	2,910,473	3.62737E-01
32	edfpa15r	88	110	45	-	-	-	26,549	1.89750E-02
33	elf9601	145	242	97	-	-	-	151,348	9.66291E-02
34	ftr10	175	94	26	-	-	-	305	4.48677E-01
35	isp9601	143	104	25	1	-	-	276,785	5.71245E-02
36	isp9602	116	122	26	-	-	-	5,197,647	1.72447E-02
37	isp9603	91	95	37	-	-	-	3,434	3.23326E-03
38	isp9604	215	132	38	-	-	-	746,574	1.42751E-01
39	isp9605	32	40	8	6	-	-	5,630	1.37171E-05
40	isp9606	89	41	14	-	-	-	1,776	5.43174E-02
41	isp9607	74	65	23	-	-	-	150,436	9.49510E-07
42	jbd9601	533	315	71	-	-	-	150,436	7.55091E-01
43	nus9601	1,567	1,622	392	47	-	-	unknown	unknown

Benchmarking Setup: Hardware and Environment

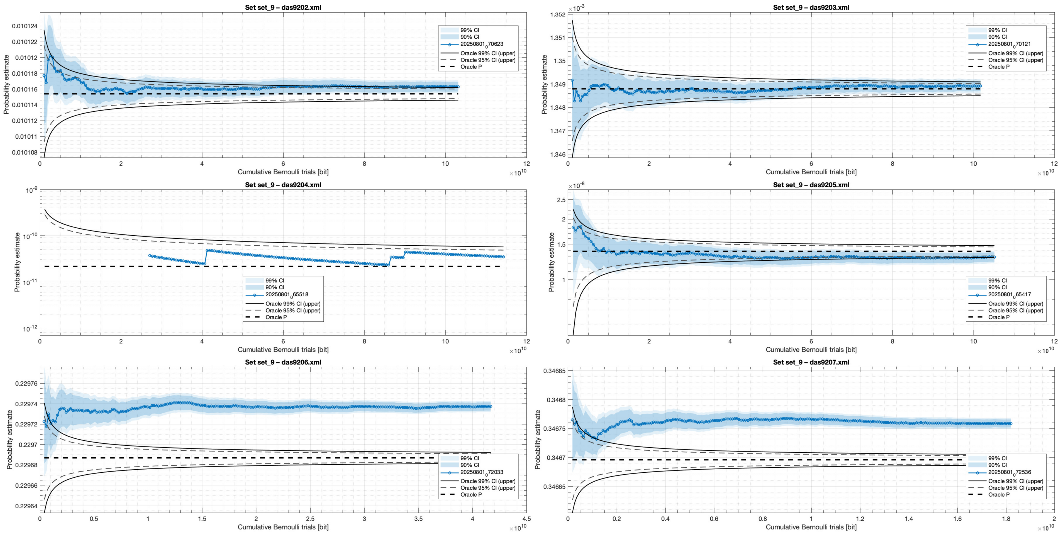
- **Target Hardware:**
  - GPU: NVIDIA® GeForce GTX 1660 SUPER (6 GB GDDR6, 1,408 CUDA cores).
  - CPU: Intel® Core™ i7-10700 (2.90 GHz, turbo-boost, hyperthreading).
- **Software Stack:**
  - SYCL-based (AdaptiveCpp/HipSYCL), with LLVM-IR JIT for kernel compilation.
  - Compiler optimization at `-O3` for efficient code generation.
  - Repeated runs (5+) to mitigate transient variations.
- **Measured Time:** Includes entire wall-clock duration, from host-device transfers and JIT compilation to final result collection.


MC - PRELIM
Min Cut Upper Bound
Rare Event Approximation
$\varepsilon = 0.05p^*$ CI = 0.95
$\varepsilon = 0.005p^*$ CI = 0.99
$\varepsilon = 0.001p^*$ CI = 0.99
timeout = 60s
N = 100Billion
$|log_{10}(p^*) - log_{10}(\hat{p})|$
Fault Tree Top Event Probability
10^{-14} 10^{-12} 10^{-10} 10^{-8} 10^{-6} 10^{-4} 10^{-2} 10^{0}
10^{2} 10^{1} 10^{0} 10^{-1} 10^{-2} 10^{-3} 10^{-4} 10^{-5} 10^{-6}
1 2 3 4 5 6 7 8

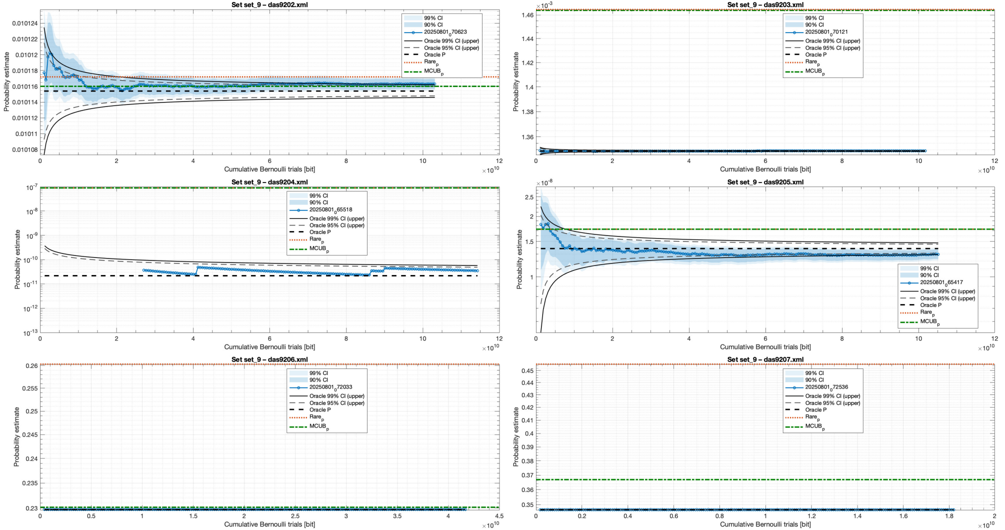
Accuracy vs. Time to Convergence

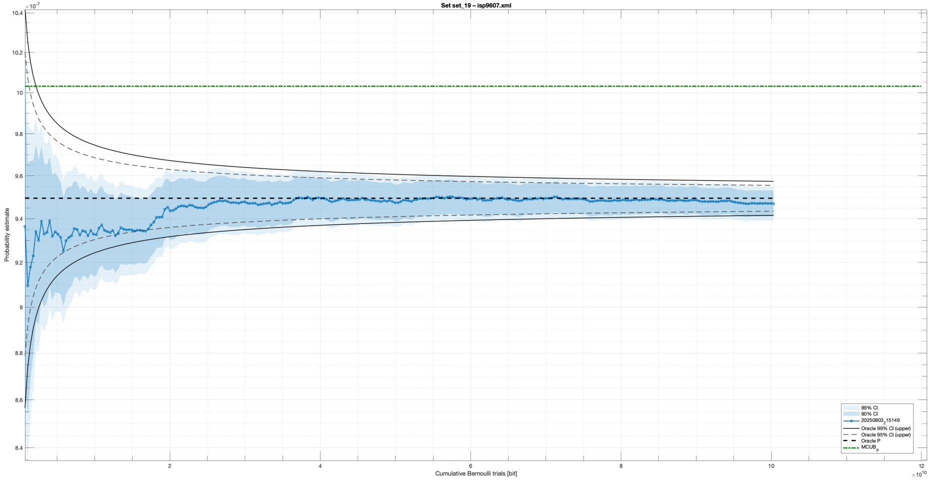


Convergence Runs

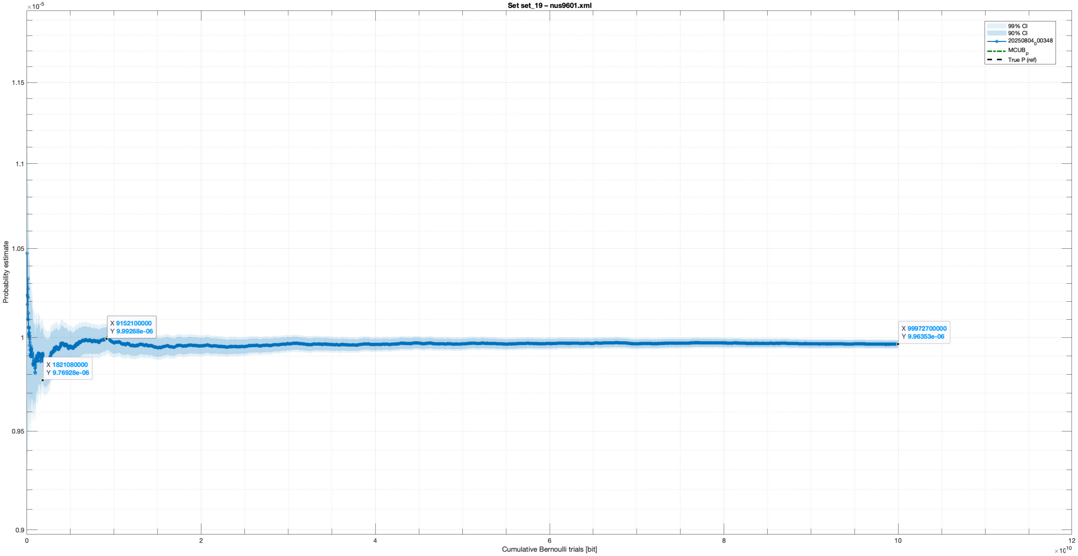


Convergence Runs, Comparison with MCUB, REA




Set set_19 – isp9607.xml
×10^-7
10.4
10.2
10
9.8
9.6
9.4
9.2
9
8.8
8.6
8.4
Probability estimate
2
4
6
8
10
12
×10^10
Cumulative Bernoulli trials [bit]
99% CI
90% CI
20250803_2 15149
Oracle 99% CI (upper)
Oracle 95% CI (upper)
Oracle P
MCUB_p

Convergence Free Run



#	Fault Tree	Mean Absolute Error - log(P)				Runtime [sec]
		REA	MCUB	Monte Carlo		
1	baobab1	$1.451\,56 \times 10^{-4}$	$1.451\,56 \times 10^{-4}$	$7.618\,80 \times 10^{-3}$	2.5×10^{8}	0.262
2	baobab2	$6.486\,28 \times 10^{-3}$	$6.347\,05 \times 10^{-3}$	$1.544\,36 \times 10^{-3}$	2.5×10^{8}	0.209
3	baobab3	$1.215\,09 \times 10^{-2}$	$1.167\,01 \times 10^{-2}$	$2.248\,43 \times 10^{-4}$	2.4×10^{8}	0.259
4	cea9601	$9.361\,95 \times 10^{-2}$	$9.322\,07 \times 10^{-2}$	$2.418\,02 \times 10^{-3}$	1.2×10^{8}	0.262
5	chinese	$1.087\,42 \times 10^{-2}$	$1.063\,54 \times 10^{-2}$	$2.146\,01 \times 10^{-3}$	9.4×10^{8}	0.277
6	das9201	$1.266\,49 \times 10^{-1}$	$1.227\,65 \times 10^{-1}$	$5.499\,63 \times 10^{-5}$	2.3×10^{8}	0.279
7	das9202	$7.727\,43 \times 10^{-5}$	$2.575\,96 \times 10^{-5}$	$1.202\,32 \times 10^{-4}$	5.2×10^{8}	0.295
8	das9203	$3.590\,19 \times 10^{-2}$	$3.559\,35 \times 10^{-2}$	$2.317\,68 \times 10^{-4}$	5.2×10^{8}	0.292
9	das9204	$1.680\,86 \times 10^{-1}$	$1.680\,87 \times 10^{-1}$	$1.134\,95 \times 10^{-1}$	6.1×10^{8}	0.292
10	das9205	$9.638\,25 \times 10^{-2}$	$9.637\,25 \times 10^{-2}$	$2.761\,90 \times 10^{-2}$	3.3×10^{9}	0.958
11	das9206	$5.435\,61 \times 10^{-2}$	$8.896\,60 \times 10^{-4}$	$3.515\,48 \times 10^{-4}$	2.0×10^{8}	0.269
12	das9207	$1.184\,86 \times 10^{-1}$	$2.454\,92 \times 10^{-2}$	$1.365\,19 \times 10^{-4}$	9.5×10^{7}	0.282
13	das9208	$4.128\,08 \times 10^{-2}$	$3.819\,68 \times 10^{-2}$	$9.340\,17 \times 10^{-5}$	2.5×10^{8}	0.307
14	das9209	$2.112\,42 \times 10^{-2}$	$1.702\,45 \times 10^{1}$		-	-
15	das9601	$5.292\,85 \times 10^{-2}$	$5.191\,22 \times 10^{-2}$	$6.671\,74 \times 10^{-4}$	1.1×10^{8}	0.256
16	das9701	$5.028\,04 \times 10^{-2}$	$3.375\,65 \times 10^{-2}$	$6.229\,78 \times 10^{-4}$	2.3×10^{7}	0.273
17	edf9201	$1.480\,12 \times 10^{-1}$	$5.361\,82 \times 10^{-2}$	$2.889\,06 \times 10^{-4}$	1.8×10^{8}	0.315
18	edf9202	$1.071\,81 \times 10^{-1}$	$6.059\,76 \times 10^{-3}$	$4.539\,00 \times 10^{-4}$	7.8×10^{7}	0.271
19	edf9203	$2.221\,46 \times 10^{-1}$	$1.172\,93 \times 10^{-1}$	$3.279\,93 \times 10^{-4}$	8.0×10^{7}	0.302
20	edf9204	$2.795\,31 \times 10^{-1}$	$1.055\,91 \times 10^{-1}$	$1.314\,16 \times 10^{-4}$	8.7×10^{7}	0.298
21	edf9205	$9.943\,39 \times 10^{-2}$	$4.462\,60 \times 10^{-2}$	$5.601\,46 \times 10^{-5}$	1.9×10^{8}	0.284
22	edf9206	$6.987\,97 \times 10^{-3}$	$7.077\,75 \times 10^{-3}$			-

Research Objective 6

Develop Importance Measures, Extend Common-Cause-Failure Analysis

Why Compute Importance Measures?

- PRA stakeholders ask: *“Which components matter most?”*
- Classical sensitivity indices (Birnbaum, Fussell–Vesely, RAW, RRW) quantify the change in top-event probability when basic-event reliabilities shift.
- Goal: embed these metrics **inside** the MC engine — no extra cut-set enumeration, no separate gate passes.

Classical Definitions (per basic event *i*)

Measure	Analytical Definition
Birnbaum (MIF)	$\mathrm{MIF}_i = \dfrac{\partial P_{\text{top}}}{\partial p_i}$
Critical (CIF)	$\mathrm{CIF}_i = \mathrm{MIF}_i \, \dfrac{p_i}{P_{\text{top}}}$
Diagnostic (DIF)	$\mathrm{DIF}_i = \dfrac{\Pr\{X_i = 1 \wedge Z = 1\}}{p_i P_{\text{top}}}$
RAW / RRW	Risk achievement / reduction worth — counterfactual probabilities when X_i forced to 1 or 0

Key insight: All measures reduce to combinations of three sample proportions:

$$\widehat{p}_i = \frac{s_i}{n}, \quad \widehat{p}_0 = \frac{s_0}{n}, \quad \widehat{p}_{0,i} = \frac{s_{0,i}}{n}.$$

MC Estimation with Minimal Sufficient Statistics

Per-iteration tallies

$$s_i \leftarrow s_i + \mathrm{popcount}(X_i), \quad s_0 \leftarrow s_0 + \mathrm{popcount}(Z), \quad s_{0,i} \leftarrow s_{0,i} + \mathrm{popcount}(X_i \,\&\, Z).$$

Only **one** extra bitwise `AND` + `popcount` per basic event.

Estimator example (Birnbaum):

$$\widehat{\mathrm{MIF}}_i = \frac{\widehat{p}_{0,i} - \widehat{p}_0 \widehat{p}_i}{\widehat{p}_i (1 - \widehat{p}_i)}.$$

MC Estimation with Minimal Sufficient Statistics

Convergence Guarantee

Each counter is a sum of $n = TN$ independent Bernoulli trials:

$$\sqrt{n}\,(\widehat{p} - p) \xrightarrow{d} \mathcal{N}(0,\, p(1-p)).$$

By the delta method the importance estimators inherit $\mathcal{O}(n^{-1/2})$ variance.

Same composite stopping rule (half-width, credible interval, info-gain) thus applies without modification.

Motivation: Common-Cause Failures

- Independent failure assumption breaks down when components share hidden dependencies (environment, manufacturing defects, etc.).
- Neglecting CCFs can under-predict top-event probability by orders of magnitude.
- Goal: incorporate parametric CCF models *without* disrupting the bit-parallel Monte-Carlo pipeline.

Graph Construction for a CCF Group $\mathcal{C}$

1. **Scale independent leaves.** Each basic event $c_i \in \mathcal{C}$ keeps an independent probability $(1 - \lambda_{\text{ccf}})\, p_{c_i}$.
2. **Insert CCF trigger variable** $S_{\mathcal{C}}$ with failure probability $\lambda_{\text{ccf}} = \sum_{k \geq 2} \pi_{\mathcal{C},k}$.
3. **CCF-root gate** $G_{\mathcal{C}} = S_{\mathcal{C}} \vee c_1 \vee \cdots \vee c_m$ routes either independent or common shock to the rest of the graph.
4. (Optional) **Multiplicity shadow gates** $H_{\mathcal{C}}^{(k)}$ realize models that distinguish the number k of simultaneously failed components.

Key property

All new nodes are *gates*; the set of basic events is unchanged, so the Monte-Carlo sampling kernel remains untouched.

MC Treatment and Convergence Guarantees

Sampling logic

- Leaves $\{S_{\mathcal{C}}, c_1, \ldots, c_m\}$ are sampled *independently*. Correlation enters only via logic connectivity.
- Same bit-packing kernel, same PRNG counters – zero performance overhead.
- Trigger variable firing forces simultaneous bits in the shadow gate; implemented by one extra OR reduction.

Unbiasedness Variance

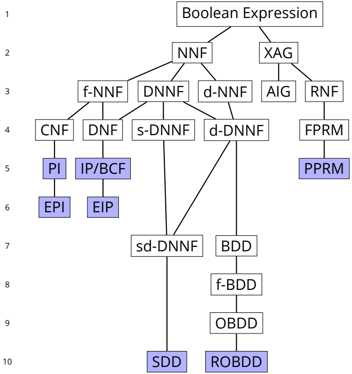
Indicator Z of any top-level node is still a Bernoulli random variable.

$$\mathrm{Var}[Z] = P(1-P), \qquad P = \Pr[Z=1].$$

Therefore all proofs for half-width, Bayesian interval, and information-gain criteria remain valid **unchanged**. CCF merely shifts P.

Take-away: CCF groups integrate into MC with *zero* sampler modification; convergence diagnostics from Research Objective 4 apply verbatim.

Hierarchy of compiled target languages. Blue nodes represent canonical forms.



Acronym	**Full form**
NNF	Negation Normal Form
XAG	XOR-And-Inverter Graph
AIG	And-Inverter Graph
ANF/RNF	Algebraic/Ring Normal Form
f-NNF	Flat Negation Normal Form
DNNF	Decomposable Negation Normal Form
d-NNF	Deterministic Negation Normal Form
FPRM	Fixed Polarity Reed-Muller
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
s-DNNF	Smooth/Structured Decomposable Negation No
d-DNNF	Deterministic Decomposable Negation Normal F
sd-DNNF	Smooth/Structured Deterministic Decomposable
PPRM	Positive Polarity Reed-Muller
PI	Prime Implicate
IP	Prime Implicant
BCF	Blake Canonical Form
EPI	Essential Prime Implicate
EIP	Essential Prime Implicant
BDD	Binary Decision Diagram
f-BDD	Free/Read-Once Binary Decision Diagram
OBDD	Ordered Binary Decision Diagram
SDD	Sentential Decision Diagram
RoBDD	Reduced Ordered Binary Decision Diagram

Two Very Different Compilation Objectives

Exact-Inference KC

- Goal: produce a *tractable* normal form where queries such as model counting, SAT, or entropy can be answered in $\mathcal{O}(|G|)$.
- Constraints: decomposability, determinism, smoothness $\Rightarrow$ DNNF, SDD, ROBDD, etc.
- Trade-off: often *expands* the graph (CNF $\rightarrow$ OBDD may grow exponentially).

Monte-Carlo KC

- Goal: *maximize throughput* of bit-parallel sampling while keeping the estimator unbiased.
- Constraints: none beyond logical equivalence; decomposability/determinism *not required*.
- Opportunity: aggressively **compress** the PDAG, collapse voting sub-trees, and flatten deep gate stacks to minimise kernel launches.

Enter: Naive MC-Specific KC Pipeline (Levels $0 \ldots 8$)

1. **L0 Baseline** – parse PDAG; keep *k*-of-*n* and XOR unexpanded.
2. **L1 Null/Negation** – eliminate vacuous gates, absorb single negations.
3. **L2 Definition Coalescing** – merge replicas, detect modules.
4. **L3 Boolean Optimization** – distributivity detection, Shannon expansion.
5. **L4–L5** – condensed passes that, in exact KC, would be repeated; here executed once until fixed-point.
6. **L6 Specialization** – map $\mathrm{VOT}(k/n)$ to hardware-native gate *instead of* DNF explosion.
7. **L7 Negation Pushdown** – only when it reduces bitwise polarity ops.
8. **L8 Final Coalescing** – stop when further compression <1

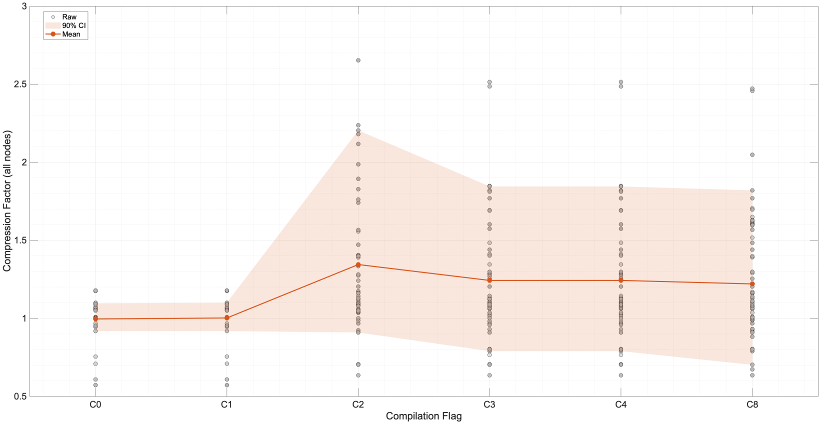
Result

Median gate count shrinks by **1.3×**; worst-case fan-in compression up to $2^n/\sqrt{n}$. Compile wall-time drops 186× compared to recursive normalizer.

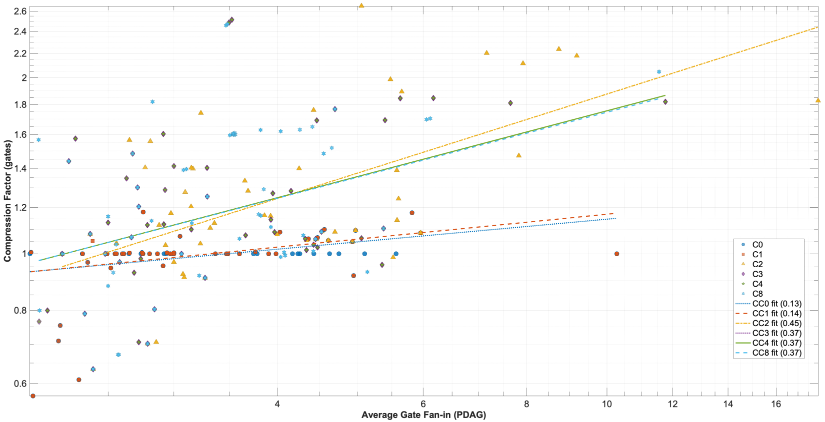
Why "Relaxed" KC Works for MC

- **Throughput**: fewer gates $\Rightarrow$ fewer kernel launches and lower memory traffic. MC cost is $\mathcal{O}(|G|)$ per iteration: shrink $|G| \Rightarrow$ linear speed-up.
- **Estimator Unbiasedness**: every transformation is a logical equivalence; Monte-Carlo estimator $\widehat{P}$ remains unbiased $\left(\mathbb{E}[\widehat{P}] = P\right)$.
- **Variance**: graph compression leaves Bernoulli variance $P(1 - P)$ unchanged; half-width formulas from Research Objective 4 stay valid.

Compression Microbenchmark



Compression Microbenchmark



Why Importance Sampling?

- Plain MC requires $\mathcal{O}(P^{-1}(1-P))$ trials to observe even a *single* failure when $P \ll 10^{-6}$. Wall-time grows to hours.
- Importance Sampling (IS) *biases* the basic-event probabilities to force rare scenarios to occur more frequently, then re-weights samples to remain unbiased.
- Objective: integrate IS into the existing bit-parallel kernel with minimal memory and arithmetic overhead.

Self-Normalized Importance Sampling (per trial)

Biasing rule

$$q_i = \mathrm{clip}(c\, p_i,\ \varepsilon,\ 1-\varepsilon), \qquad c > 1.$$

Draw $X^* \sim \mathrm{Bern}(q_i)$ *independently*. Likelihood ratio for one component:

$$\ell_i(X_i^*) = \frac{p_i^{X_i^*}\,(1-p_i)^{1-X_i^*}}{q_i^{X_i^*}\,(1-q_i)^{1-X_i^*}}.$$

Trial weight $L = \prod_i \ell_i(X_i^*)$.

Point estimator

$$\widehat{P}_{\mathrm{IS}} = \frac{\sum_{k=1}^{N} L^{(k)} Z^{(k)}}{\sum_{k=1}^{N} L^{(k)}} \quad \text{(self-normalized)}.$$

Implementation Hooks

- Store q_i and two 32-bit pre-computed ratios p_i/q_i, $(1-p_i)/(1-q_i)$.
- Extra multiplication per basic event to accumulate $\log L$; fused into existing sampling loop.
- One additional reduction per batch for ΣL and ΣLZ.

Variance Reduction and Stopping Rule

- Classical MC variance: $\sigma^2_{\text{MC}} = P(1-P)/N$.
- IS variance:

$$\sigma^2_{\text{IS}} = \frac{1}{N}\left(\mathbb{E}_q[L^2 Z] - P^2\right) \ll \sigma^2_{\text{MC}} \quad \text{if bias factor } c \text{ well chosen.}$$

- For tilt factor $c = 10^3$ and $P = 10^{-8}$, empirical runs show 100–200× variance reduction.
- **Convergence guarantee**—because $\widehat{P}_{\text{IS}}$ is still a sample mean of i.i.d. weighted Bernoullis, the CLT and all three statistical half-width criteria from Research Objective 4 apply *unchanged* with $\widehat{\sigma}_{\text{IS}}$.

Practical Tip

Auto-tune c via pilot runs that seek weight coefficient of variation $\mathrm{CV} \approx 1$; implemented as a future work item.

Limitations & Future Work

Where the Framework Still Falls Short

- **Ultra-rare probabilities** ($P < 10^{-8}$) require a mature importance sampling engine.
- **Hardware dependence**: peak throughput assumes wide SIMD + high memory bandwidth; CPU-only runtimes are $\sim$100× slower.
- **Static logic only**: time-dependent repair, mission phases, or feedback loops not yet supported.
- **Limited dependence models**: beyond CCF, no general copula or Bayesian-network sampling engine.
- **Manual kernel tuning**: batch/bitpack geometry must be tuned per device family; no auto-tuner.
- **Validation scope**: full-scale industry PRA (e.g. Generic PWR) not yet replicated end-to-end.

Future Work Roadmap

Near-Term (1 yr)

- Adaptive variance-reduction palette: stratified, antithetic, multi-level MC.
- GPU-resident auto-tuner for kernel geometry and memory layout.
- Weighted variance estimator + half-width for importance sampling.
- Extended benchmark suite (Aralia + Generic PWR).

Mid-Term (3 yr)

- Discrete-event simulation back-end for mission-time and repair modeling.
- Copula / Bayesian-network engine for correlated uncertainties.
- Gradient-based calibration via Boolean auto-diff + SGD.
- Streaming mode for real-time risk dashboards.

Ultimate vision: a portable, self-optimizing risk-analytics platform delivering sub-second updates on enterprise-scale models.

The End

Monte Carlo Sampling

- Rather than summing or bounding all combinations of failures, *simulate* random draws of $\mathbf{X}$.
- Each Monte Carlo iteration:
  1. Sample $x_1, x_2, \ldots, x_n \overset{\text{i.i.d.}}{\sim} \prod p(x_i)$.
  2. Evaluate the Boolean function $F(\mathbf{x})$ (cost is just logical gate evaluation).
  3. Collect whether $F(\mathbf{x}) = 1$ (failure) or 0 (success).
- Repeating for many samples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ yields a *sample average* estimate of the probability.
- Benefits:
  - Bypasses explicit inclusion-exclusion expansions.
  - Straightforward to parallelize (evaluate each draw in separate threads or blocks).

Estimator for the Expected Value (i.e., Probability)

- A Boolean function $F(\mathbf{x})$ can be viewed as an indicator function: $F(\mathbf{x}) \in \{0, 1\}$.
- The event $\{F(\mathbf{X}) = 1\}$ has probability $\mathbb{E}[F(\mathbf{X})]$.
- **Monte Carlo estimator:**

$$\widehat{P}_N = \frac{1}{N} \sum_{i=1}^{N} F(\mathbf{x}^{(i)}),$$

  where each $\mathbf{x}^{(i)}$ is a random draw from the input distribution.
- By the Law of Large Numbers,

$$\lim_{N \to \infty} \widehat{P}_N = \Pr\left[F(\mathbf{X}) = 1\right], \quad \text{almost surely.}$$

- Error decreases at rate $\mathcal{O}(1/\sqrt{N})$, analyzed via the Central Limit Theorem.

Boolean Functions: Basic Concepts

- Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n Boolean variables, each $x_i \in \{0, 1\}$.
- A *Boolean function* is any map $F(\mathbf{x}) : \{0,1\}^n \rightarrow \{0,1\}$.
- Example: If F encodes "system fails," then $F(\mathbf{x}) = 1$ signifies a failure mode, where $\mathbf{x}$ captures component states.
- Modeling perspective:
  - AND, OR, NOT, k-of-n gates allow composing complex logic.
  - Each F can be evaluated deterministically if we know $\mathbf{x}$.

Exact Probability Estimation: Inclusion-Exclusion

- Suppose each x_i has a probability $p_i = \Pr[x_i = 1]$, assuming independence.
- We want $\Pr[F(\mathbf{x}) = 1]$, which is

$$\Pr\left[F(\mathbf{X}) = 1\right] = \sum_{\mathbf{x}\in\{0,1\}^n} F(\mathbf{x}) \prod_{i=1}^{n} \left[p_i^{x_i}(1-p_i)^{1-x_i}\right].$$

- For sets of events, using the *inclusion-exclusion principle*:

$$\Pr\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{k=1}^{n} (-1)^{k+1} \sum_{1\le i_1<\cdots<i_k\le n} \Pr\left(E_{i_1} \cap \cdots \cap E_{i_k}\right).$$

Approximation Methods: REA and MCUB

For large n, exact enumeration of subsets is exponential, making it impractical for large Boolean circuits.

- **Rare-Event Approximation (REA):**
  - Assumes each event has small probability $p_i \ll 1$.
  - Overlaps (intersections of multiple failures) are deemed negligible.
  - Probability of the union $\approx \sum_i \Pr[E_i]$, ignoring higher-order terms.

Approximation Methods: REA and MCUB (cont.)

- **Min-Cut Upper Bound (MCUB):**

$$\Pr\left[\bigcup_{C\in\{\mathrm{MCS}\}} C\right] \leq \sum_{C\in\{\mathrm{MCS}\}} \prod_{b\in C} p_b, \tag{2}$$

  - Interprets each minimal cut set (MCS) as a distinct mechanism for failure.
  - Sums (over)estimate total failure if MCSs share components.
  - Often used as a conservative bound in safety analyses.

- Both methods reduce complexity but can misestimate the true probability when events are not truly rare or heavily intersect.

Boolean Derivatives: Definition and Interpretation

- **Boolean Derivative Concept:** For a Boolean function $F(\mathbf{x})$ with $\mathbf{x} = (x_1, \ldots, x_n)$, the derivative with respect to x_i is defined via XOR:

$$\frac{\partial F}{\partial x_i} = F(x_i = 0, \mathbf{x}_{-i}) \oplus F(x_i = 1, \mathbf{x}_{-i}),$$

  where $\oplus$ denotes the exclusive-OR operation, and $\mathbf{x}_{-i}$ are all variables except x_i.
- **Interpretation:**
  - $\frac{\partial F}{\partial x_i}(\mathbf{x}) = 1$ whenever *flipping* x_i changes the value of F under the specific configuration $\mathbf{x}_{-i}$.
  - Captures *sensitivity*: if $\frac{\partial F}{\partial x_i}$ rarely equals 1, then F is robust to changes in x_i.

Extension to Monte Carlo Estimation of Boolean Derivatives

- **Key Idea:** Estimate $\mathbb{E}[\,\partial F/\partial x_i\,]$ by sampling random configurations $\mathbf{x}^{(s)}$ of the Boolean inputs, then checking how F changes when x_i is flipped.
- **Sampling Procedure:**
  1. Draw $\mathbf{x}^{(s)} = (x_1^{(s)}, \ldots, x_n^{(s)})$ from the distribution of interest.
  2. Form $\mathbf{x}^{(s)} \oplus \mathbf{e}_i$ by flipping the ith coordinate.
  3. Compute:

$$\frac{\partial F}{\partial x_i}\big(\mathbf{x}^{(s)}\big) \;=\; F\big(\mathbf{x}^{(s)}\big) \;\oplus\; F\big(\mathbf{x}^{(s)} \oplus \mathbf{e}_i\big).$$

- **Insight:**
  - Sensitivity and importance analysis using sampling methods.
  - Gradient computation opens a path towards learning-based tasks.

Avoiding Inclusion-Exclusion via Monte Carlo

- Exact expansions for large circuits require enumerating all subsets of failing components or gates, which is computationally huge.
- In contrast, *Monte Carlo* draws a sample $\mathbf{x} \in \{0, 1\}^n$ and directly evaluates $F(\mathbf{x})$ without enumerating *all* subsets.
- Each run picks a single draw of failed components from the distribution. After many runs, the frequency of $F = 1$ approximates its probability.
- Results:
  - No exponential blow-up in the number of terms.
  - Straightforward extension to complex gate structures, correlated variables.
  - Parallelizable on modern CPU/GPU architectures.

Data-Parallel Implementation using SYCL

Data-Parallel Monte Carlo for Boolean Circuits:

- Simultaneous evaluation of *all* intermediate gates, success, and failure paths.
- Relax coherence constraints - arbitrary shapes with NOT gates permitted.
- Vectorized bitwise hardware ops for logical primitives (AND, OR, XOR, etc.)
- Specialized treatment of k/n logic, without expansion.
- Simultaneous use of all available compute - GPUs, multicore CPUs.