Probability Estimation Using Monte Carlo Simulation of Boolean Logic on Hardware-Accelerated Platforms

PSA 2025
19th International Probabilistic Safety Assessment and Analysis

Arjun Earthperson
PhD Candidate, PRA Group

NC STATE UNIVERSITY

EMBEDDED TOPICAL CONFERENCE AT THE ANS 2025 ANNUAL CONFERENCE

Industry Response to Emergent AI/ML Workloads

Heavy Investment in Data-Parallel Hardware

- GPUs, tensor cores provide high throughput for integer operations.
  - Apple M4 Neural Engine: ≈ 38 TOPS (int8).
  - Nvidia RTX 4090: ≈ 1000 TOPS.
- Current-gen consumer hardware already supports specialized ops (Intel AMX).

Designed for Inference on Massive Models

- $\approx 10^9$ parameters on mobile devices (e.g., Gemma 2B).
- $\approx 10^{12}$ parameters on HPC/cloud (LLaMa 4 at 400 Billion).

Comparatively,

- Largest (public) PRA models: $\approx 10^6$ parameters (Generic PWR).
- However, PRA model quantification is deeply recursive.

Overview and Highlights I

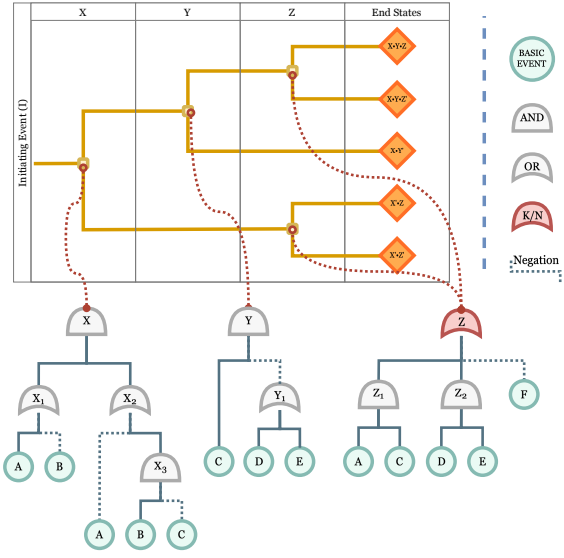
High-Throughput Boolean Logic Evaluation (aka Eval Query)

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

Overview and Highlights II

Probability Estimation using Eval Query (aka Expectation Query)

- Estimate probabilities for *all* events in the PRA model simultaneously.
- No need to compute the minimal cut sets or prime implicants.
- Streamable: Solve the entire PRA model iteratively, approx 0.3 seconds per iteration, regardless of model size. More iterations needed for complex models.



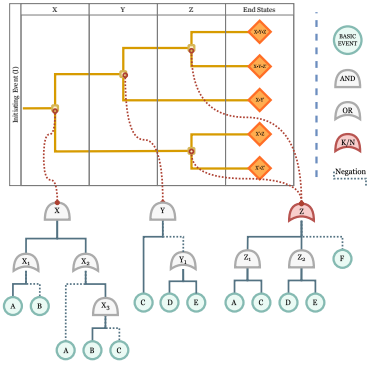
Variable	Expression
X	$(A\|B') \bullet (A'\|(B \bullet C'))$
Y	$C \bullet (D\|E)'$
Z	$kn[(A \bullet C), (D \bullet E), F']$

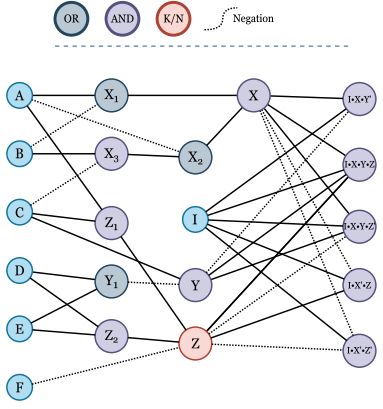
Table: Unsimplified Boolean expression for each Top Event

Small, but non-trivial structure:

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

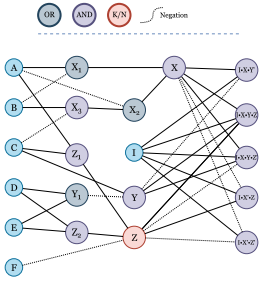
Compile a Directed Acylic Graph (DAG) from Logic Model





Start with an Arbitrary Topological Ordering:

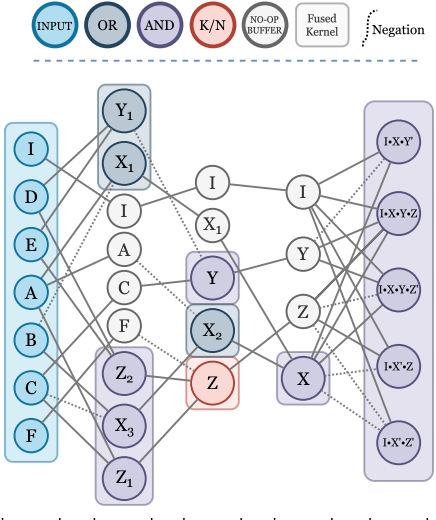
- Aim for succinctness.
- prefer HW-native ops.
- e.g. don't expand k/n.



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.

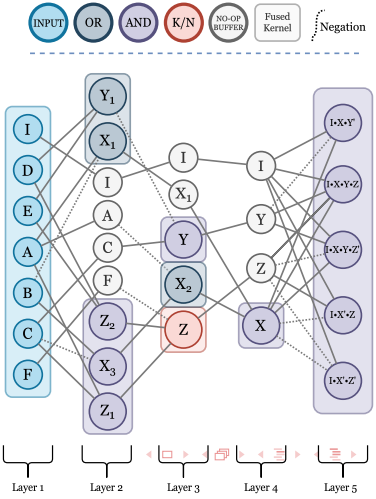


Querying the Compiled Knowledge Graph

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 it?

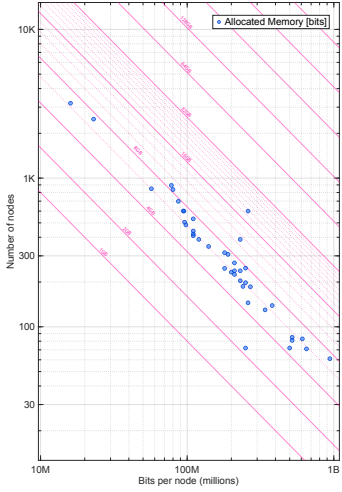


Eval Query on the Compiled Knowledge Graph

Eval Query Performance on GPUs:

- Latency: 200-300 *ms* per graph pass.
- Throughput: 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?



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:**

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

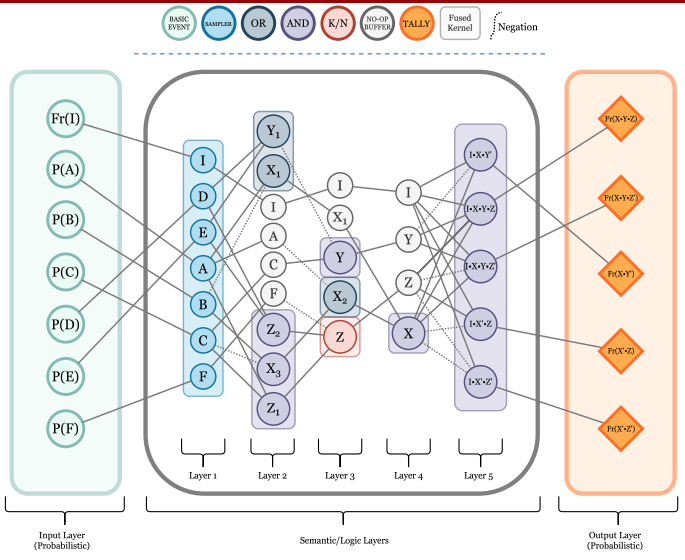
  where each $\mathbf{x}^{(i)}$ is a random draw from the input distribution.
- By the Law of Large Numbers,

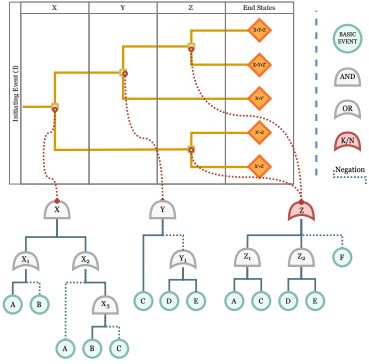
$$\lim_{N \to \infty} \widehat{P}_N \;=\; \Pr\big[F(\mathbf{X}) = 1\big], \quad \text{almost surely.}$$

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

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).


BASIC EVENT
SAMPLER
OR
AND
K/N
NO-OP BUFFER
TALLY
Fused Kernel
Negation
Fr(I)
P(A)
P(B)
P(C)
P(D)
P(E)
P(F)
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Fr(X·Y·Z)
Fr(X·Y·Z')
Fr(X·Y')
Fr(X'·Z)
Fr(X'·Z')
Input Layer (Probabilistic)
Semantic/Logic Layers
Output Layer (Probabilistic)


X
Y
Z
End States
Initiating Event (I)
X·Y·Z
X·Y·Z'
X·Y'
X'·Z
X'·Z'
BASIC EVENT
AND
OR
K/N
Negation

Preliminary Case Study

Overview: Aralia Dataset

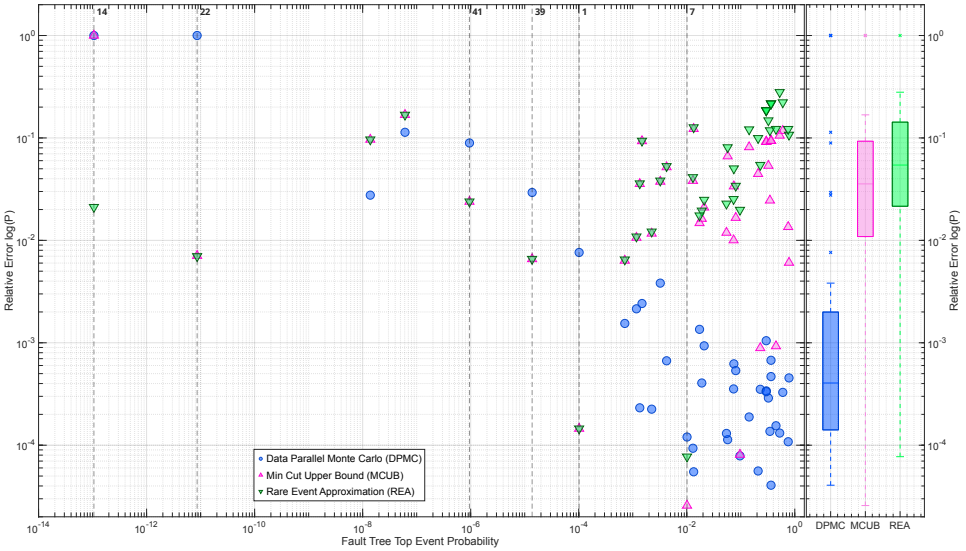
- **Dataset Composition:** The Aralia collection consists of 43 trees.
- **Diverse Problem Sizes:** Small trees (e.g. 25–32 basic events) through mid-sized models with over 1,500 BEs.
- **Wide Probability Range:** Top-event probabilities spanning from rare events near 10^{-13} to fairly likely failures 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.

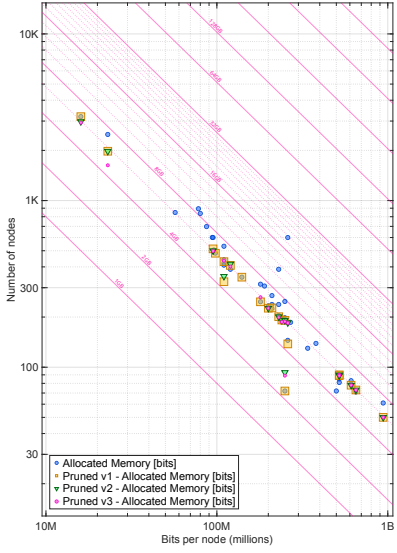
Benchmarking Setup: Hardware and Environment

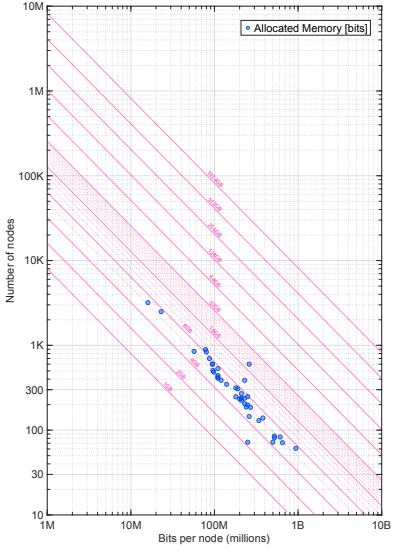
- **Target Hardware:**
  - GPU: NVIDIA® GeForce GTX 1660 SUPER (6 GB GDDR6, 1,408 CUDA cores).
  - CPU: Intel® Core™ i7-10700 (2.90 GHz, turbo-boost, hyperthreading).
- **Software Stack:**
  - SYCL-based (AdaptiveCpp/HipSYCL), with LLVM-IR JIT for kernel compilation.
  - Compiler optimization at `-O3` 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.

Monte Carlo Execution and Implementation

- **Objective:** Compute TOP event probabilities for all 43 trees.
- **Sampling Strategy:**
  - Single pass per fault tree, generating as many samples as fit in 6 GB GPU memory.
  - 128-bit Philox4x32x10 pseudo-random number generator, parallel threads.
- **Bit-Packing Optimization:**
  - Each group of 64 Monte Carlo outcomes stored in a single 64-bit word.
  - Enables vectorized instructions (e.g. `popcount`) and reduces memory I/O.
- **Data Types:**
  - Tallies in 64-bit integers.
  - Probability accumulations in double precision (64-bit float).


14
22
41
39
1
7
Relative Error log(P)
Fault Tree Top Event Probability
Data Parallel Monte Carlo (DPMC)
Min Cut Upper Bound (MCUB)
Rare Event Approximation (REA)
DPMC
MCUB
REA


10K
1K
300
100
30
Number of nodes
10M
100M
1B
Bits per node (millions)
Allocated Memory [bits]
Pruned v1 - Allocated Memory [bits]
Pruned v2 - Allocated Memory [bits]
Pruned v3 - Allocated Memory [bits]


10M
1M
100K
10K
1K
300
100
30
10
Number of nodes
1M
10M
100M
1B
10B
Bits per node (millions)
Allocated Memory [bits]

Limitations:

- Brute-force/naive Monte Carlo struggles when sampling rare-events.
  - Implement importance sampling: WIP.
- Brute-force/naive Monte Carlo is a poor strategy for sampling correlated events.

Next Steps:

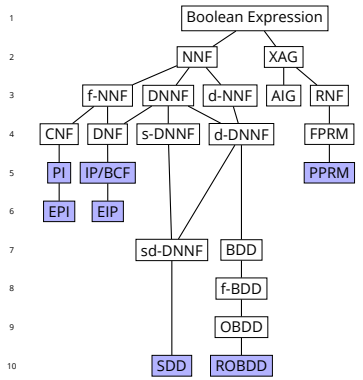
- Benchmark on larger (G-PWR, G-MHTGR) models.

Future Work:

- Embedding models from Knowledge Graph, for semantic representation.
- Gradient computation on Knowledge Graph.
- Minimal cut set computation from Knowledge Graph.

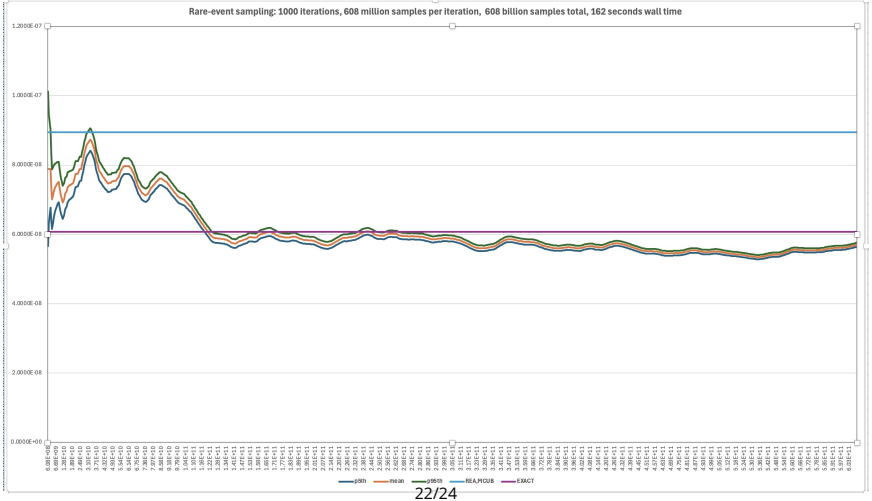
The End

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 Normal
d-DNNF	Deterministic Decomposable Negation Normal Form
sd-DNNF	Smooth/Structured Deterministic Decomposable Neg
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

Convergence over 1000 iterations, Aralia das9204



#	Fault Tree	Basic Events	Logic Gates					Minimal Cut Sets	Top Event Probability
			Total	AND	VOT	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	