Protecting Reprogrammable Hardware with Polymorphic Circuit Variation*

**J. Todd McDonald, Yong C. Kim,
and Michael R. Grimaila**

**Center for Cyberspace Research
Air Force Institute of Technology
WPAFB, OH**

***The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government**

Outline

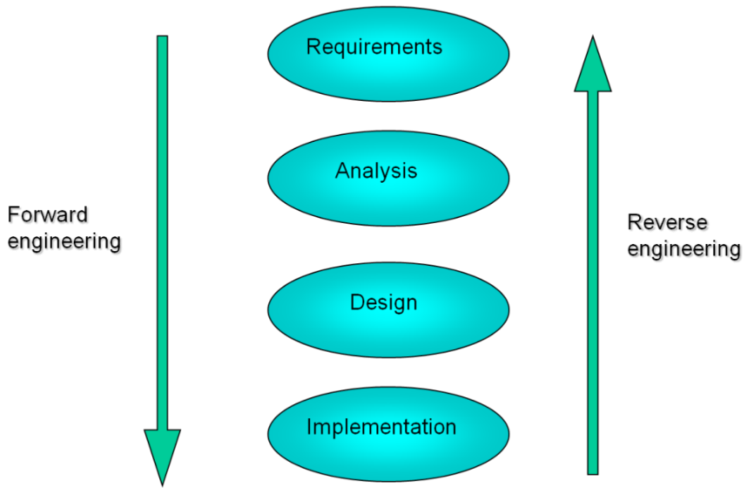
- Protection Context
- Polymorphic Variation as Protection
- Hiding Properties of Interest
- Framework and Experimental Results

Protection Context

- Embedded Systems / “Hardware”
  - Increasingly represented as reprogrammable logic (i.e., software!)
  - We used to like hardware because it offered “hard” solutions for protection (physical anti-tamper, etc.)
- Our beginning point: what happens if hardware-based protections fail?
  - Hardware protection: I try to keep you from physically getting the netlist/machine code
  - Software protection: I give you a netlist/machine code listing and ask you questions pertaining to some protection property of interest
- Protection/exploitation both exist in the eye of the beholder

Protection Context

- Critical military / commercial systems vulnerable to malicious reverse engineering attacks
  - Financial loss
  - National security risk

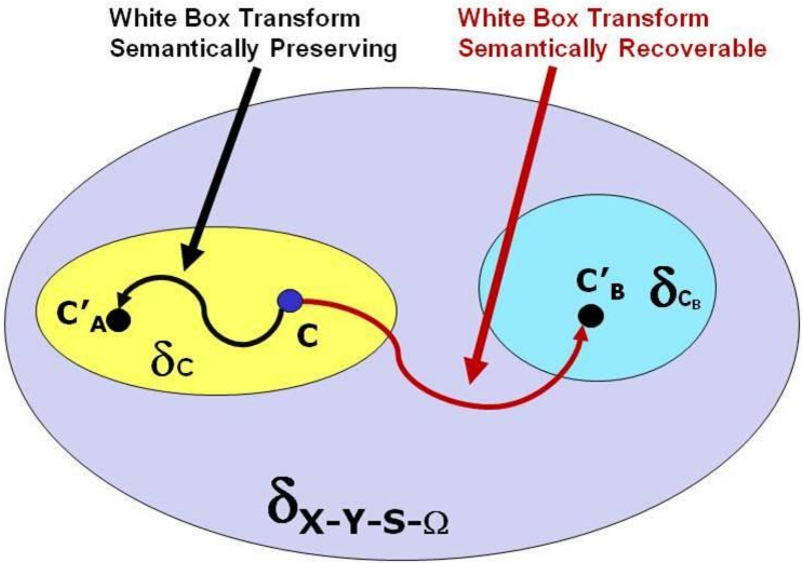


- Reverse Engineering and Digital Circuit Abstractions
  - Architectural (Behavioral)
  - Register Transfer Language (RTL)
  - Gate Level
  - Transistor Level
  - Layout

INCREASING DETAIL

Polymorphic Variation as Protection

- Experimental Approach:
  - Consider practical / real-world / theoretic circuit properties related to security
  - Use a variation process to create polymorphic circuit versions
    - *Polymorphic* = *many forms* of circuits with semantically equivalent or semantically recoverable functionality
  - Characterize algorithmic effects:
    - Empirically demonstrate properties
    - Prove as intractable
    - Prove as undecidable

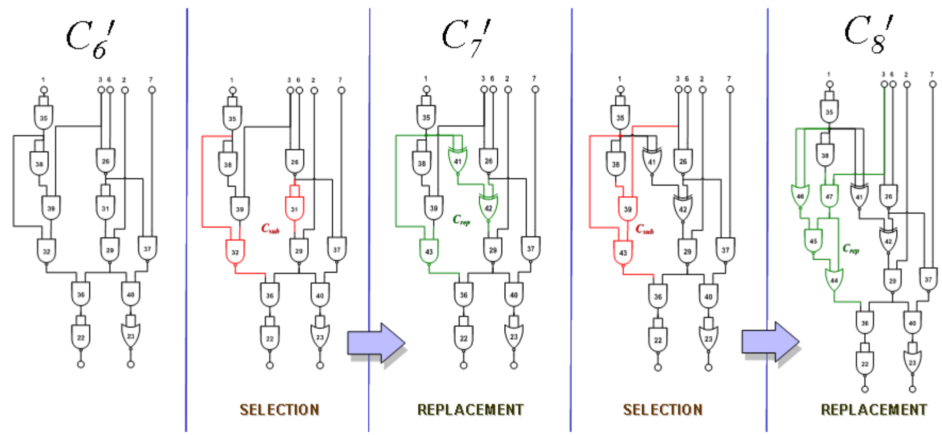


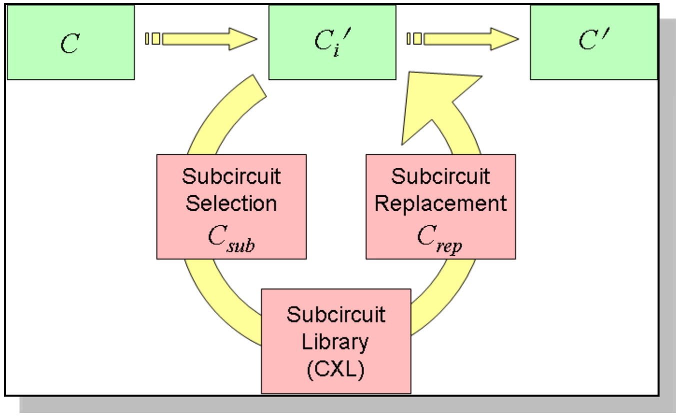
Polymorphic Variation as Protection

Algorithm and Variant Characterization:

Selection:
1) Random
2) Deterministic

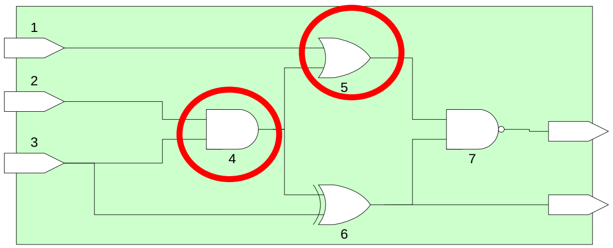
Replacement
1) Random
2) Deterministic



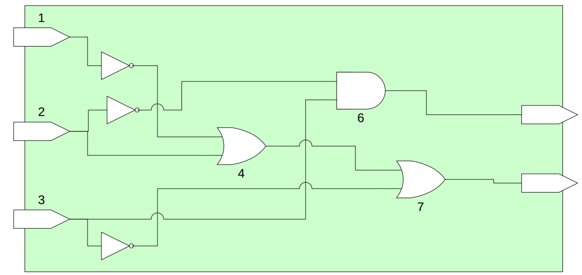


Hiding Properties of Interest

General Intuition and Hardness of Obfuscation



The ONLY true "Virtual Black Box"



X1	X2	X3	4	5	Y6	Y7
			AND(3,2)	OR(4,1)	XOR(4,3)	NAND(5,6)
0	0	0	0	0	0	1
0	0	1	0	0	1	1
0	1	0	0	0	0	1
0	1	1	1	1	0	1
1	0	0	0	1	0	1
1	0	1	0	1	1	0
1	1	0	0	1	0	1
1	1	1	1	1	0	1

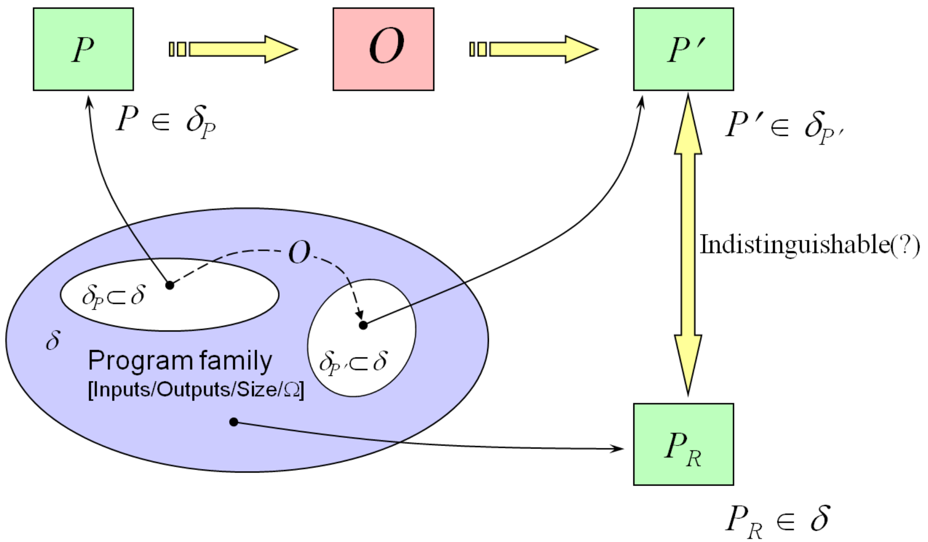
"The How"

X1	X2	X3	Y6	Y7
			XOR(4,3)	NAND(5,6)
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Semantic Behavior

Hiding Properties of Interest

- Since we can't hide *all* information leakage….
  - Can we protect intent?
    - Tampering with code in order to get specific results
    - Manipulating input in order to get specific results
    - Correlating input/output with environmental context
  - Can we impede identical exploits on functionally equivalent versions?
  - Can we define and measure *any* useful definition of hiding short of absolute proof and not based *solely* on variant **size**?



Hiding Properties of Interest

Logical View

Functional Hiding

Control Hiding

Component Hiding

Signal Hiding

Topology Hiding (Gate Replacement)

Physical Manifestation

Side Channel Properties

Framework and Experimental Results

- When does (random/deterministic) iterative selection and replacement:
  1) Manifest hiding properties of interest?
  2) Cause an adversarial reverse engineering task to become intractable or undecidable?

- What role does logic reduction and adversarial reversal play in the outcome (ongoing)

- Are there circuits which will fail despite the best variation we can produce? (yes)

Framework and Experimental Results

- Is perfect or near topology recovery useful (therefore, is topology *hiding* useful)?
  - In some cases, yes
  - Foundation for other properties (signal / component hiding)
  - For certain attacks, it is all that is required

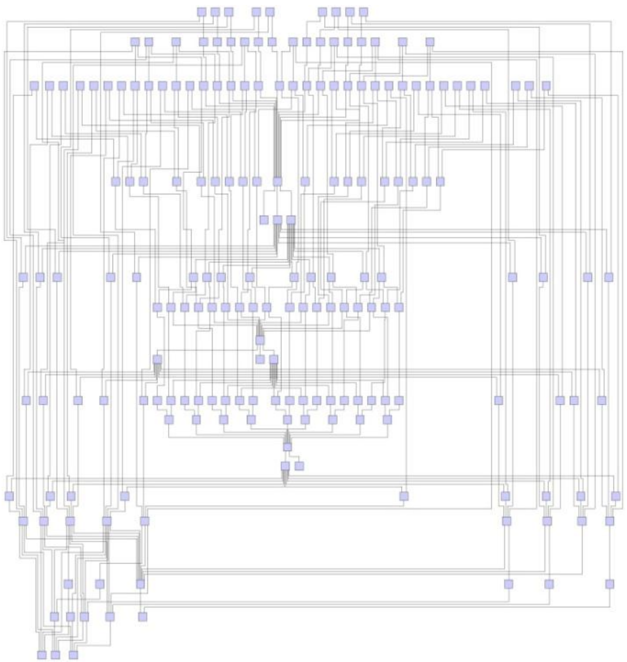
- Accomplishing topology hiding
  - Change basis type (normalizing distributions, removing all original)
  - Guarantee every gate is replaced at least once
  - Multiple / overlapping replacement = diffusion

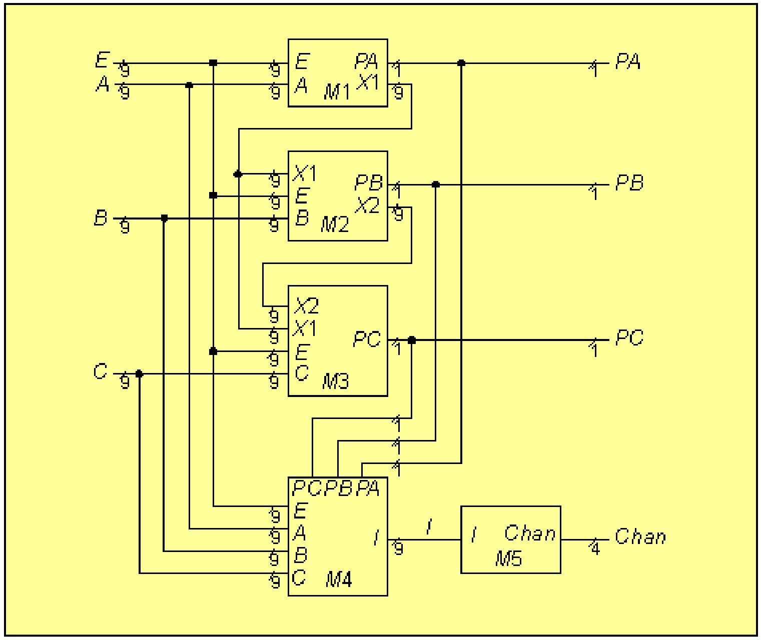
Topology:
Gate fan-in
Gate fan-out
Gate type

Experiment 1: Measuring "Replacement" Basis Change

Develop America's Airmen Today ... for Tomorrow

c432

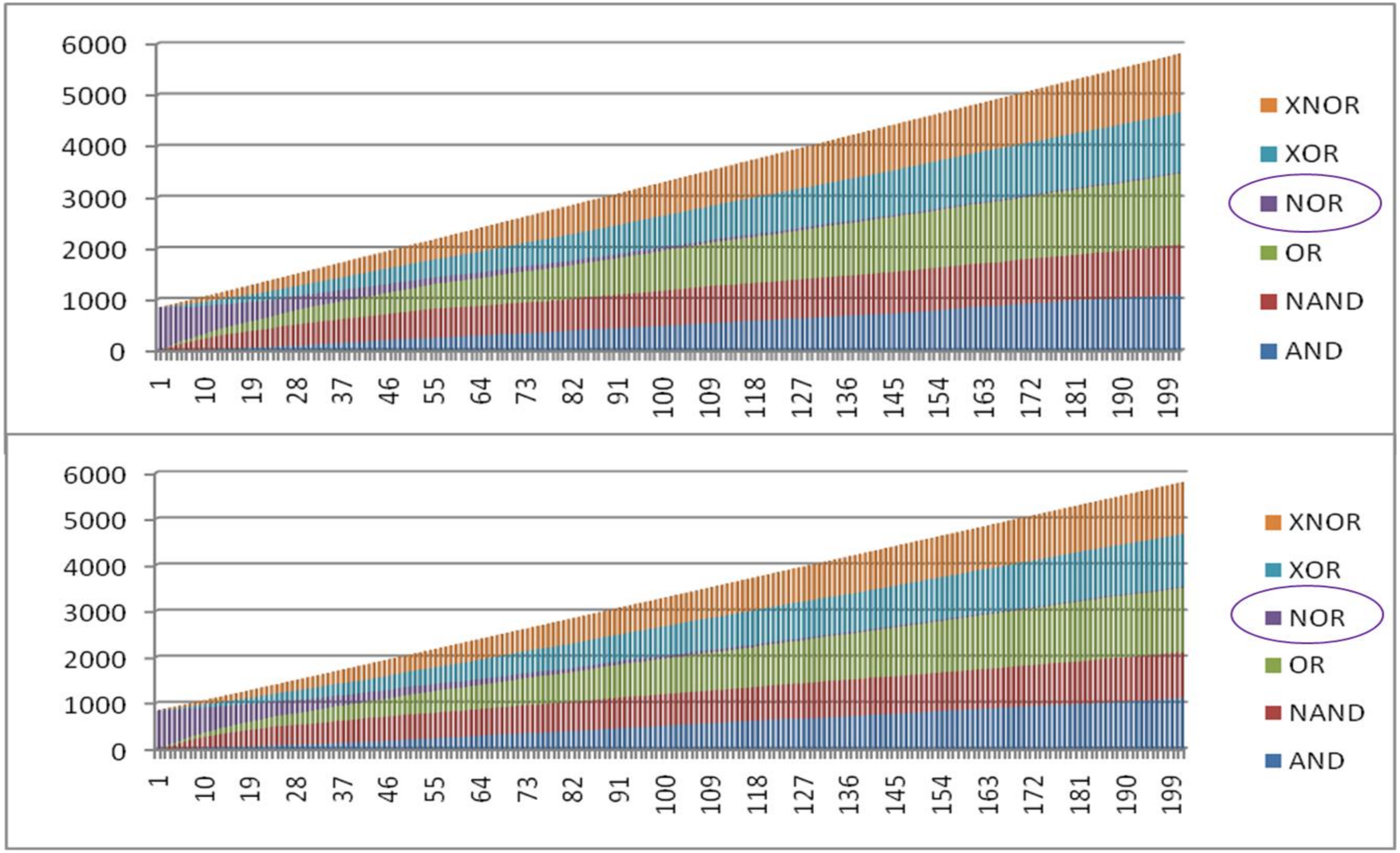




c432	**120 gates (4 ANDs + 79 NANDs + 19 NORs + 18 XORs + 40 inverters)**
Decomposed	**230 gates (60 ANDs + 151 NANDs + 19 NORs + 40 inverters)**
Decomposed NOR	**843 gates (843 NORs)**

Experiment 1a: Measuring "Replacement" Basis Change

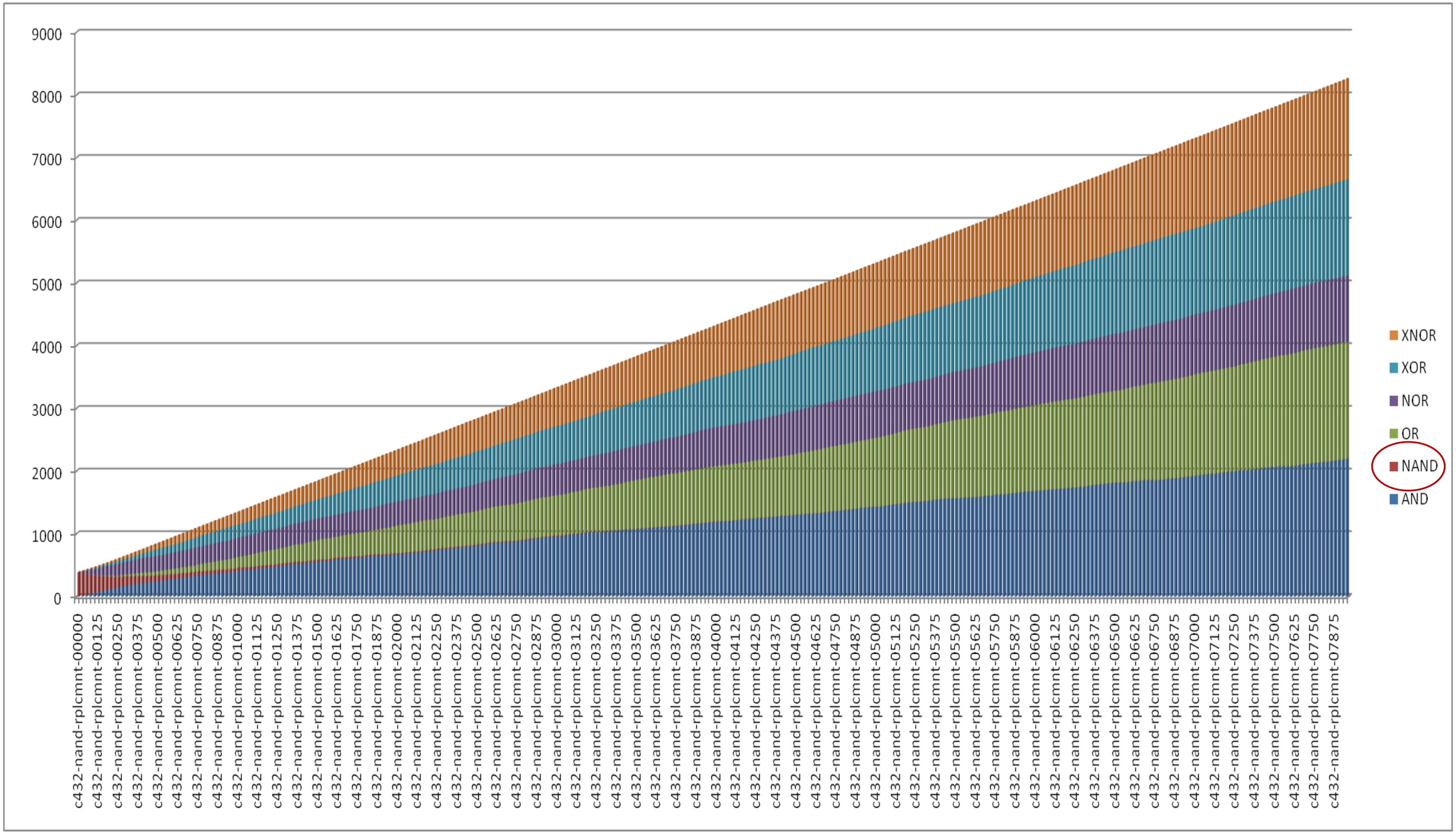
Ω = {NOR} → Ω = {AND, NAND, OR, XOR, NXOR}



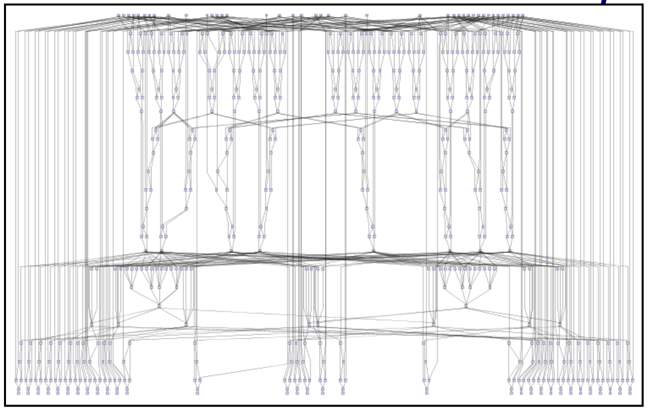
Experiment 1b: Measuring "Replacement" Basis Change

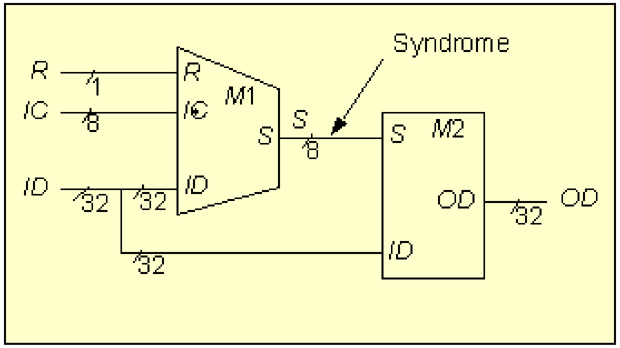
Develop America's Airmen Today ... for Tomorrow

Ω = {NAND} → Ω = {AND, NOR, OR, XOR, NXOR}



Experiment 2: Measuring "Replacement" Uniform Basis Distribution





ISCAS-85 c1355

Iterative Random Selection Algorithm:

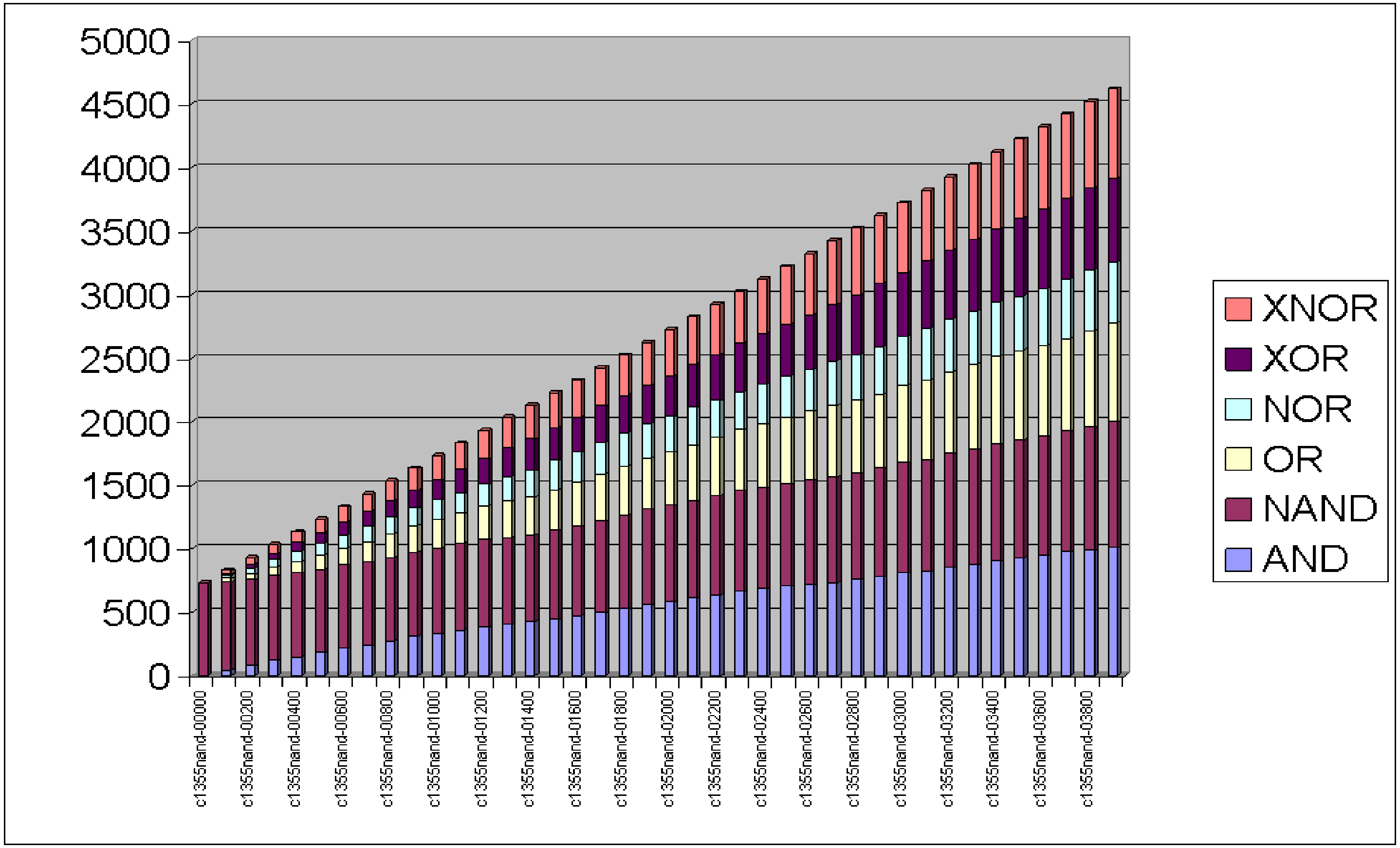
Selection Strategy:
- 5% 1) Single Gate: Random
- 75% 2) Two Gate: Random
- 5% 3) Two Gate: Largest Level
- 5% 4) Two Gate: Output Level
- 5% 5) Two Gate: Random Level
- 5% 6) Two Gate: Fixed Level

Replacement Strategy:
Random 6-GATE Basis

C1355	**506 gates (56 ANDs + 416 NANDs + 2 ORs + 32 buffers + 40 inverters)**
Decomposed	**550 gates (96 ANDs + 416 NANDs + 6 ORs + 32 buffers + 40 inverters)**
Decomposed NAND	**730 gates (730 NANDs)**

Experiment 2: Measuring "Replacement" Uniform Basis Distribution

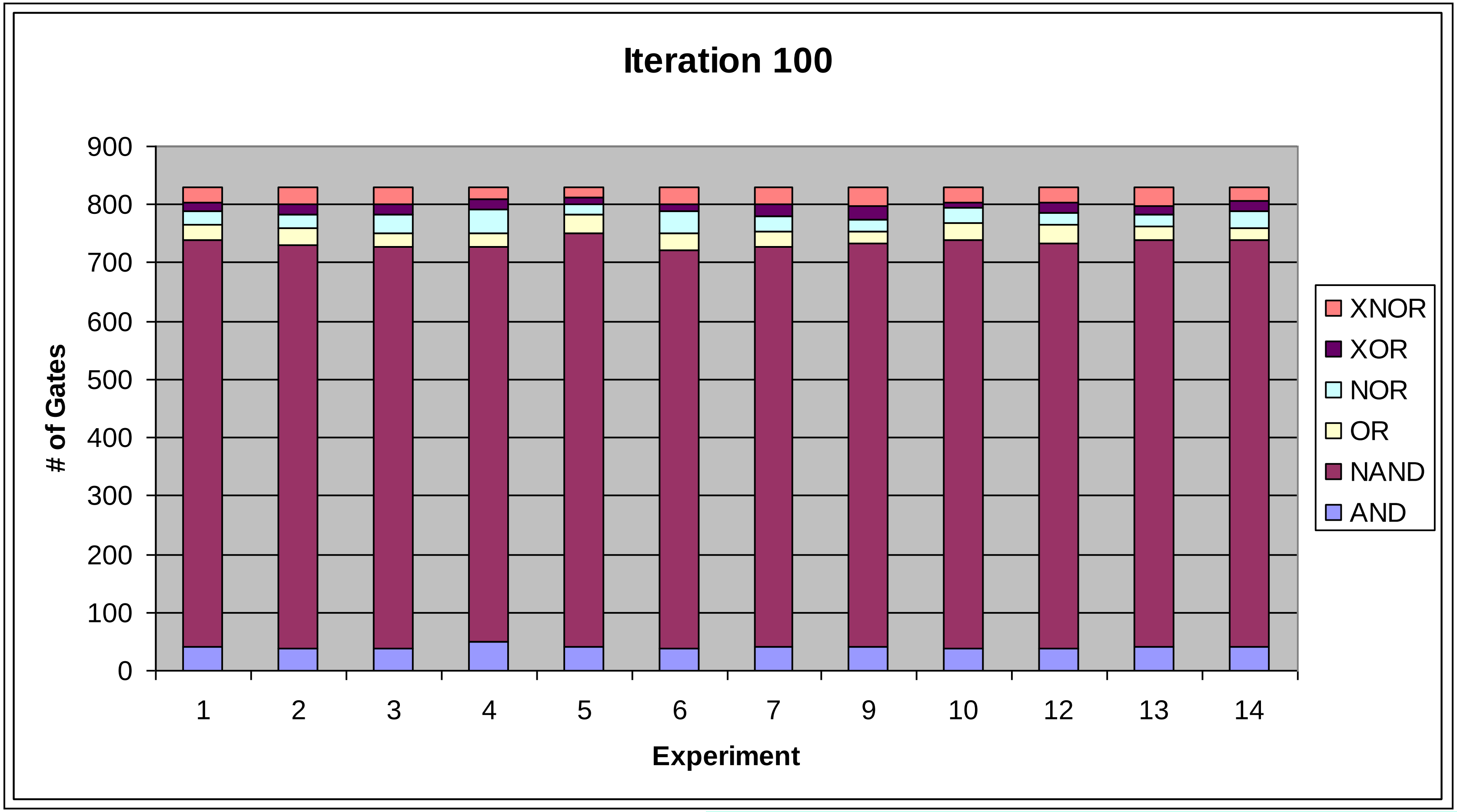
Ω = {NAND} → Ω = {AND, NAND, OR, NOR, XOR, NXOR}



"Single 4000 Iteration Experiment"

Experiment 2: Measuring "Replacement" Uniform Basis Distribution

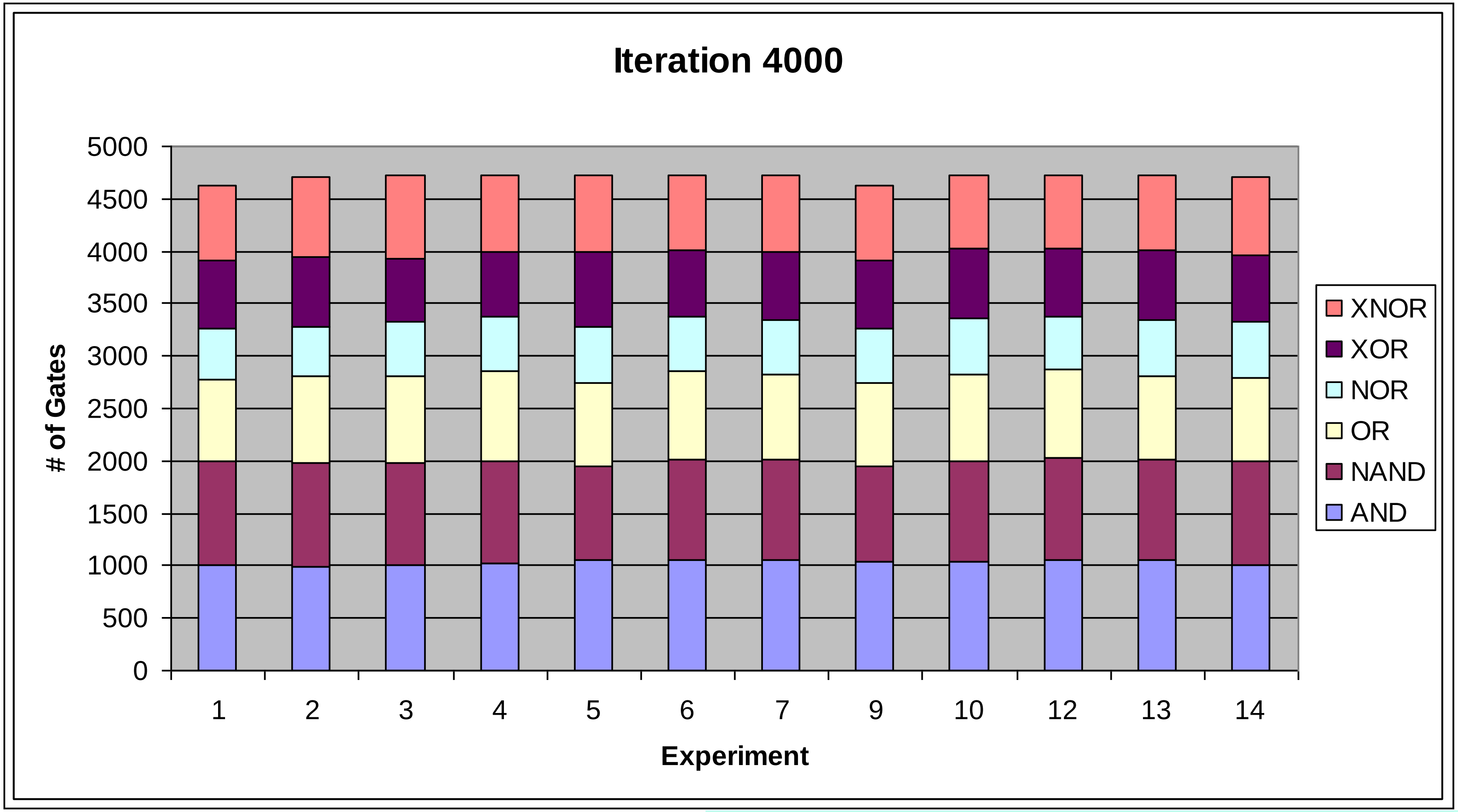
Ω = {NAND} → Ω = {AND, NAND, OR, NOR, XOR, NXOR}



"Multiple 4000 Iteration Experiments"

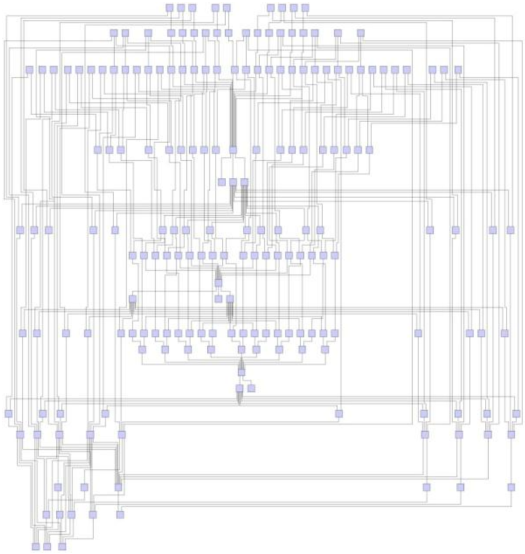
Experiment 2: Measuring "Replacement" Uniform Basis Distribution

Ω = {NAND} → Ω = {AND, NAND, OR, NOR, XOR, NXOR}



"Multiple 4000 Iteration Experiments"

Experiment 3: Measuring "Replacement" Smart Random Selection



ISCAS-85 c432

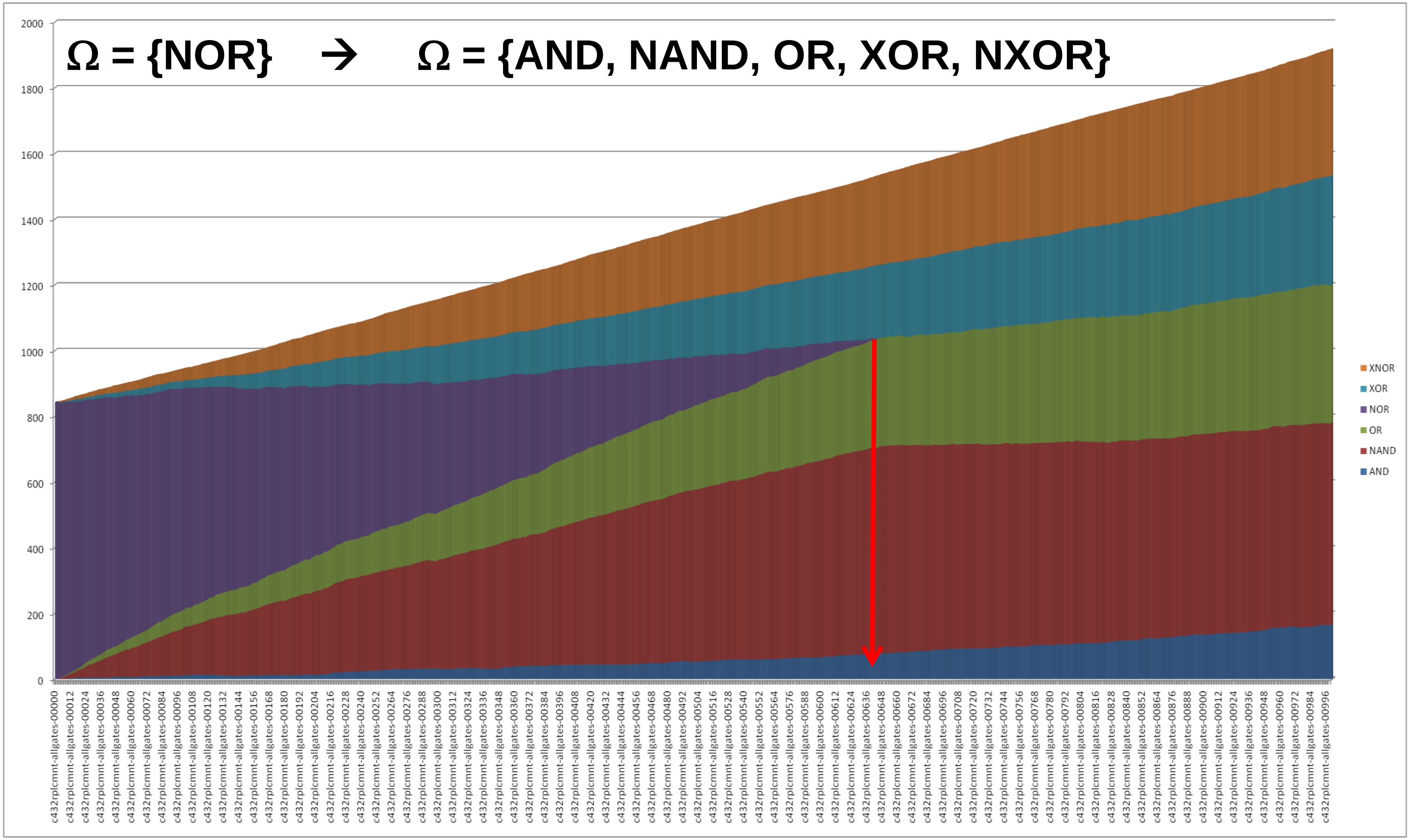
Iterative Smart Random 2-Gate Selection Algorithm:

Selection Strategy:
Smart Two Gate Random

Replacement Strategy:
Random Equivalent

Experiment 3: Measuring “Replacement” Smart Random Selection

Develop America's Airmen Today ... for Tomorrow



Things We've Learned Along the Way

- What algorithmic factors influence hiding properties the most?
  - Iteration number
  - Selection size
  - Replacement circuit generation (redundant vs. non-redundant)
- Ongoing work in:
  - Increasing selection size
  - Determinist generation
  - Integrated logic reduction
  - Formal models: term rewriting systems, abstract interpretation, graph partitioning

Questions

?

Obfuscation Comparison Models

VBB

$P_1 \Rightarrow \boxed{O} \Rightarrow O(P_1)$ **???** TT_{P1}

RPM

$P_1 \Rightarrow \boxed{O} \Rightarrow O(P_1)$ **???** P_R

$\boxed{O}$ **???**

IND

$P_1 \Rightarrow \boxed{O} \Rightarrow O(P_1)$ $P_2 \Rightarrow \boxed{O} \Rightarrow O(P_2)$

P_1, P_2 **???** $O(P_1)$, $O(P_2)$

$P_{1,} P_2 \in \delta_f$

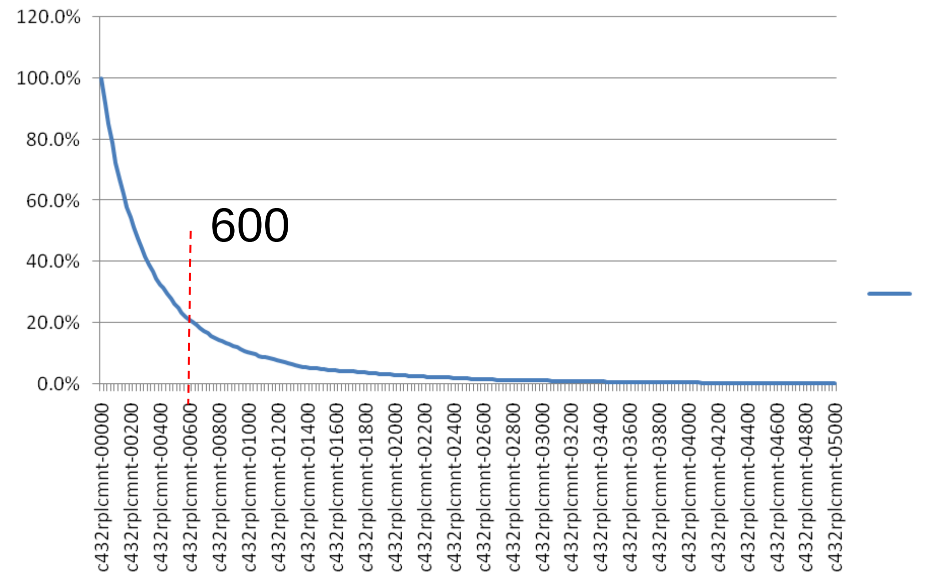
BP

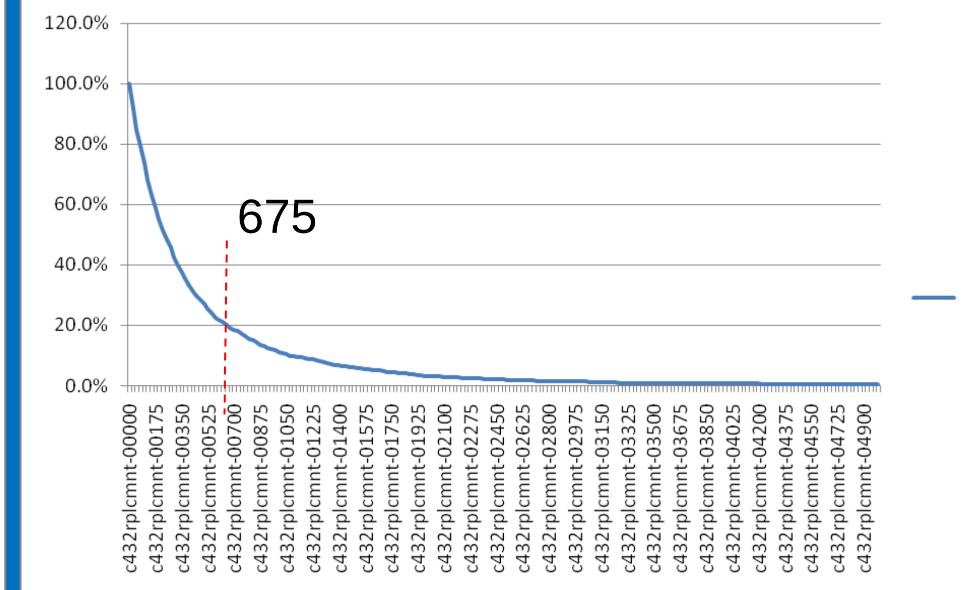
$P_1 \Rightarrow \boxed{O} \Rightarrow O(P_1)$ $O(P_1) \Rightarrow \boxed{O} \Rightarrow O(O(P_1))$

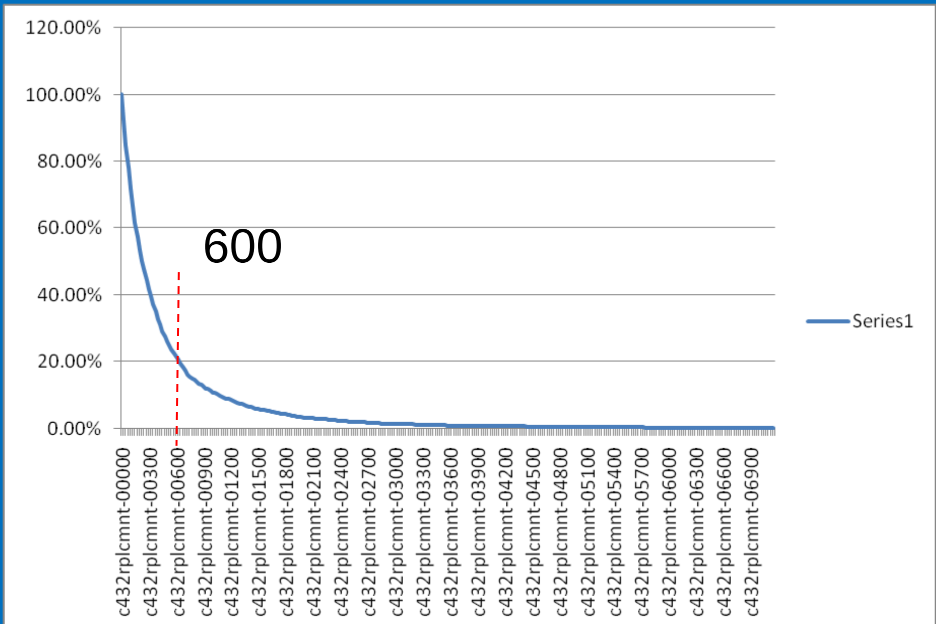
P_1 **???** $O(P_1)$ **???** $O(O(P_1))$

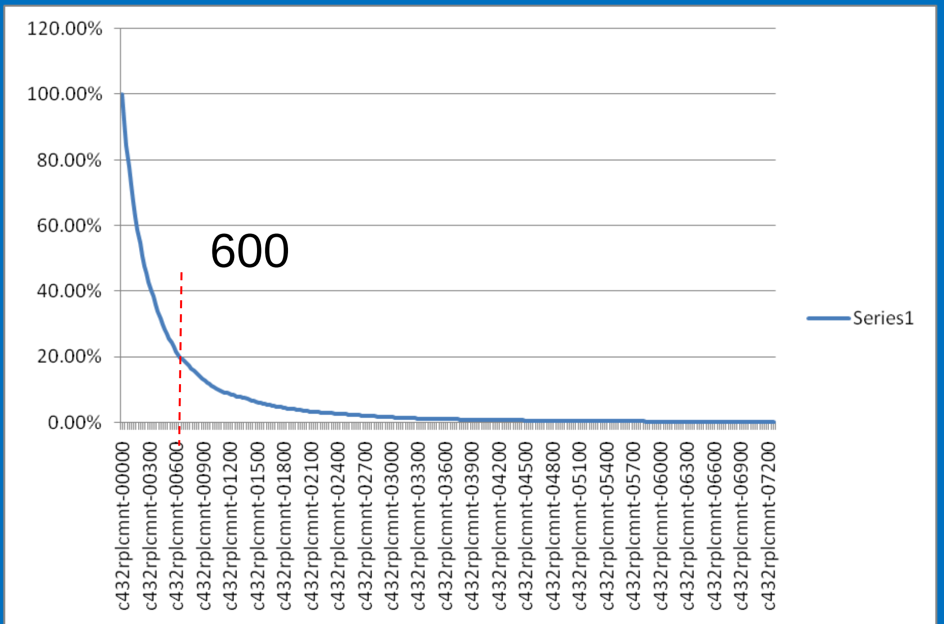
Experiment 1a: Measuring

% of ORIGINAL GATES



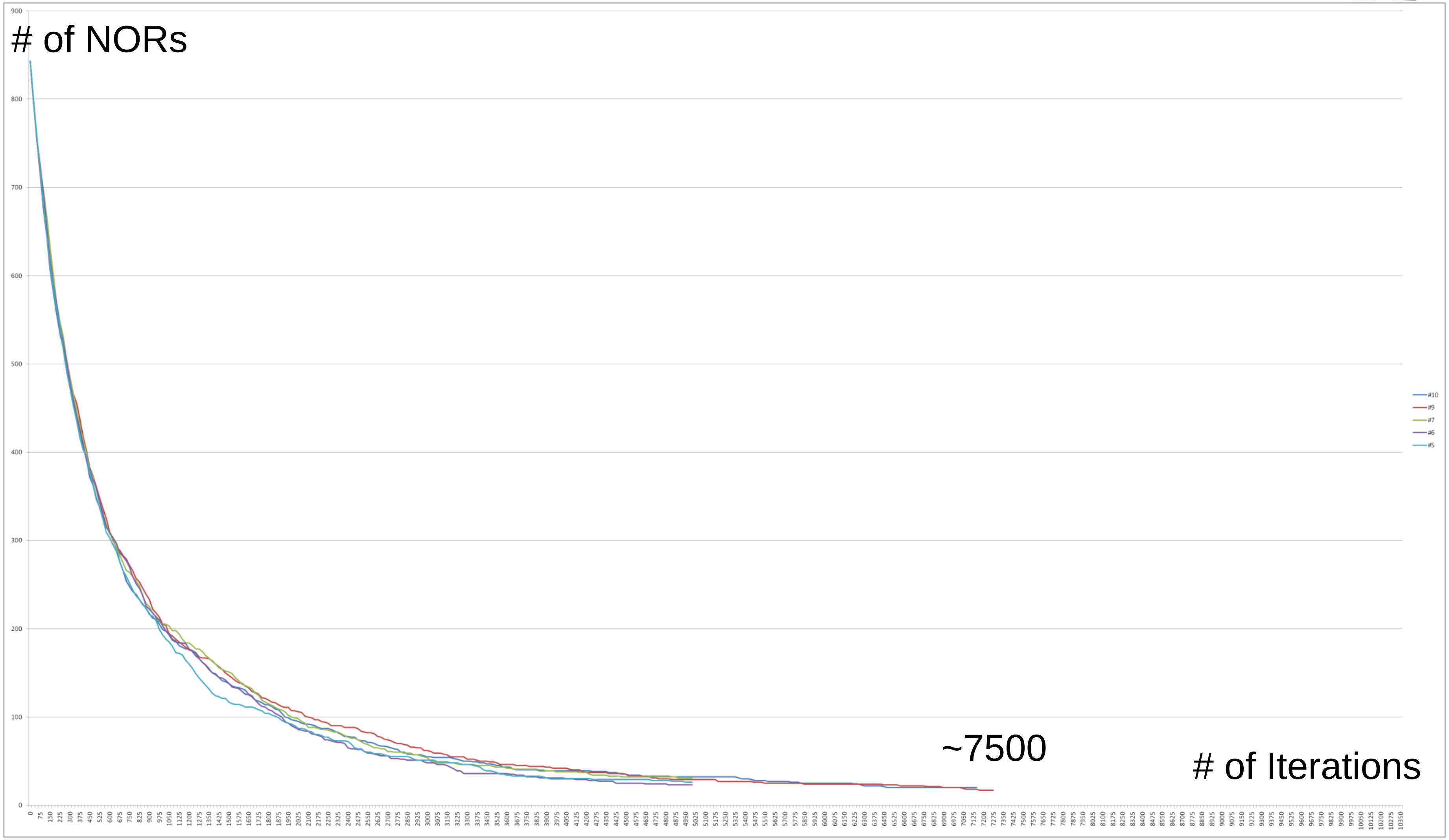






Experiment 1a: Measuring "Replacement"

Ω = {NOR} → Ω = {AND, NAND, OR, XOR, NXOR}

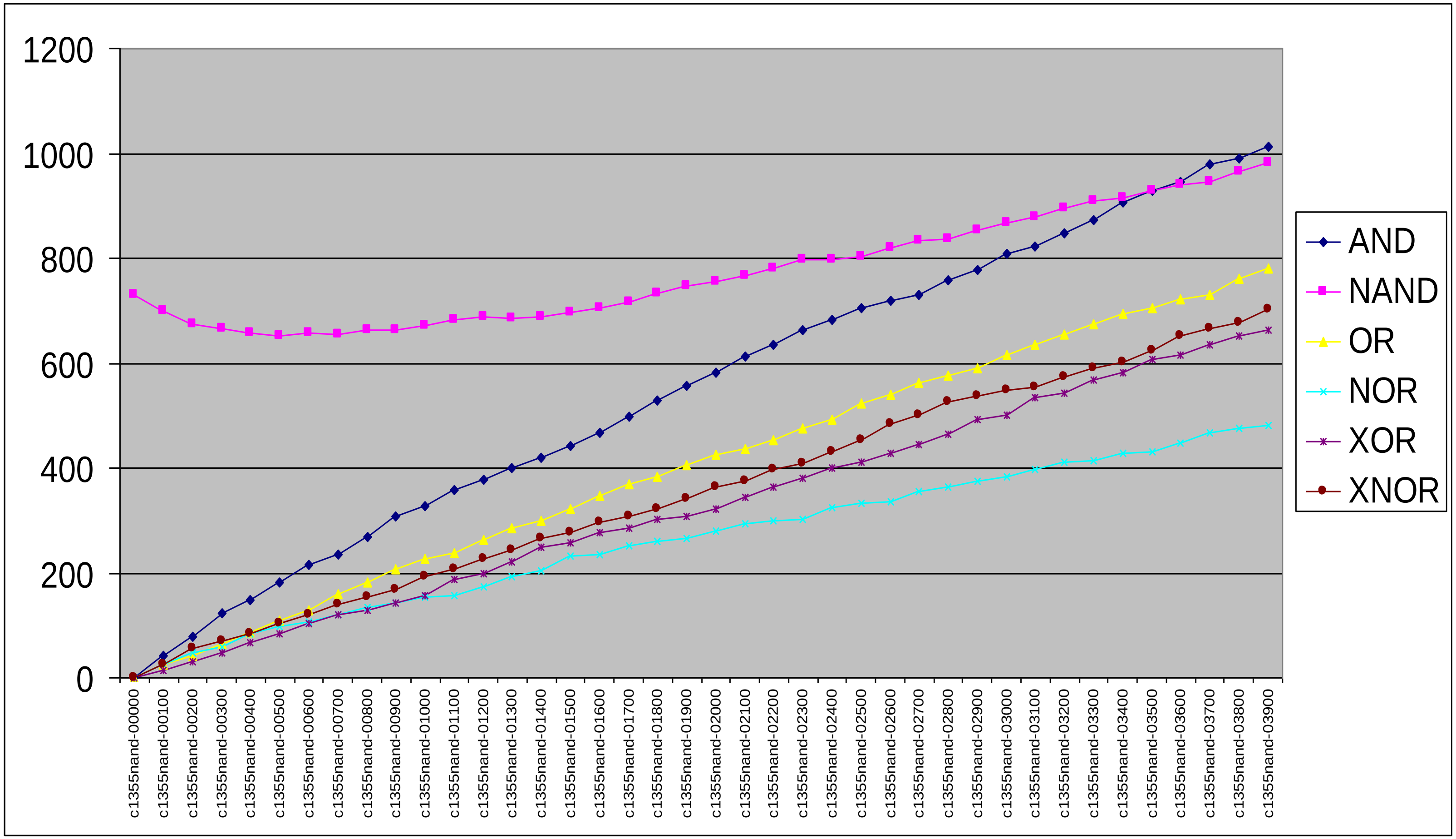


ISCAS-85 c1355

Experiment 2: Measuring "Replacement"

Develop America's Airmen Today ... for Tomorrow

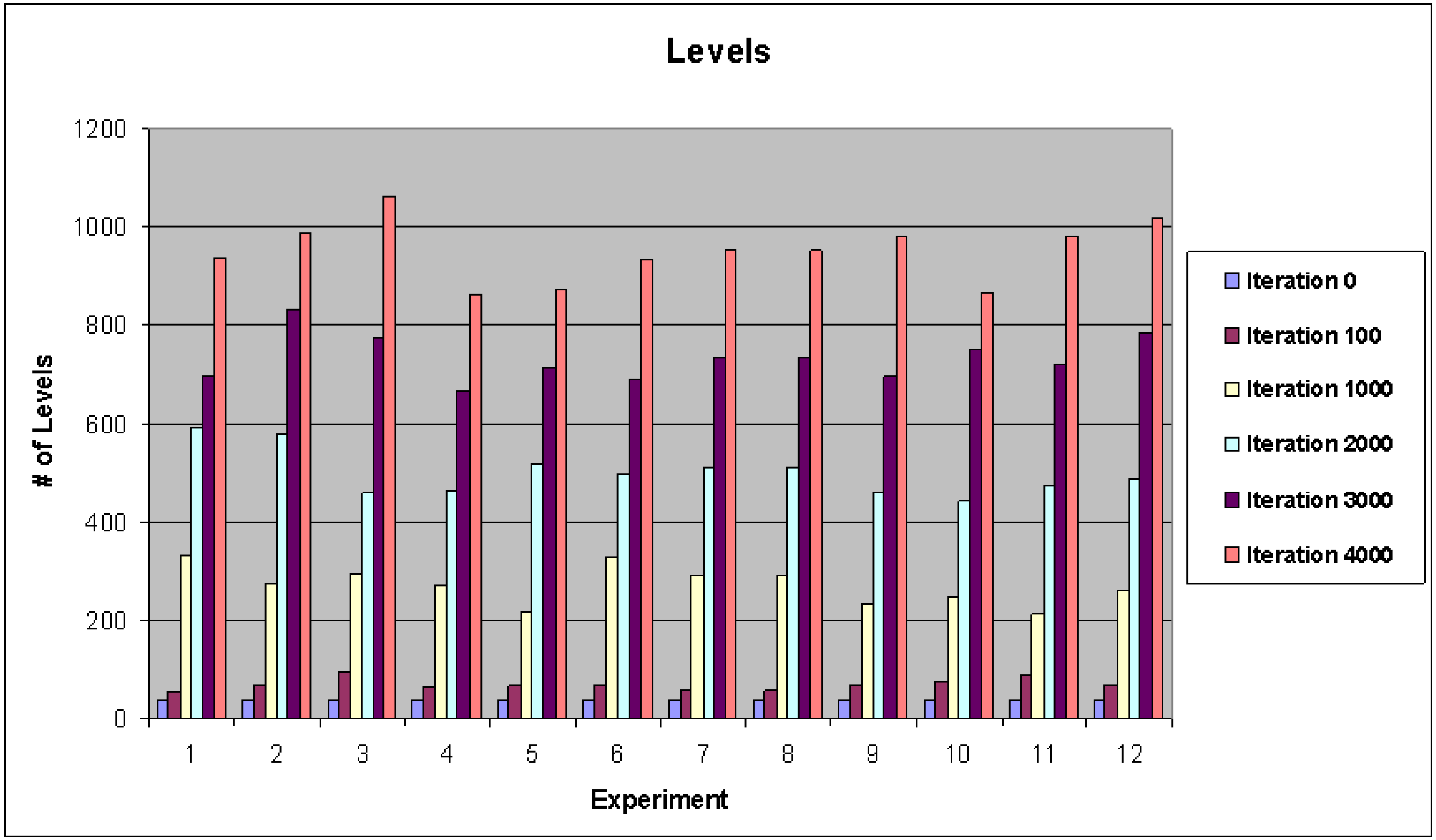
Ω = {NAND} → Ω = {AND, NAND, OR, NOR, XOR, NXOR}



"Single 4000 Iteration Experiment"

Experiment 2: Measuring "Replacement"

Ω = {NAND} → Ω = {AND, NAND, OR, NOR, XOR, NXOR}



"Multiple 4000 Iteration Experiments"