CS137: Electronic Design Automation

Day 8: February 4, 2004

Fault Detection

Today

- Faults in Logic
- Error Detection Schemes
- Optimization Problem

Problem

- Gates, wires, memories:
  - built out of physical media
  - may fail

Device Physics

- Represent a 1 or 0 with charge
  - On a gate, in a memory
- Charge may be disrupted
  - α-particle
  - Ground bounce
  - Noise coupling
  - Tunneling
  - Thermal noise
  - Behavior of individual electrons is statistical

DRAMs

- Small cells
- Store charge dynamically on capacitor
- Store about 50,000 electrons
- Must be refreshed
  - Data leaks away through parasitic resistance
- α-particle can be 1,000,000 carriers?

System Reliability

- Device fail with Probability: P_{fail}
- Have N components in system
- All must work for device to work
- $P_{sys} = (1-P_{fail})^N$

$$P_{sys} = 1 - N \times P_{fail} + \binom{N}{2} \times P_{fail}^2 - \binom{N}{3} \times P_{fail}^3 + \ldots$$

System Reliability

$$P_{sys} = 1 - N \times P_{fail} + \binom{N}{2} \times P_{fail}^2 - \binom{N}{3} \times P_{fail}^3 + \ldots$$

- If $N \times P_{fail} << 1$
  - $N \times P_{fail}$ dominates higher order terms…

$$P_{sys} \approx 1 - N \times P_{fail}$$

System Reliability

$$P_{sys} \approx 1 - N \times P_{fail}$$

- $P_{sysfail} \approx N \times P_{fail}$

Modern System

- 100 Million → 1 Billion Transistors
  - Not to mention wiring…
- > GHz = > 1 Billion Transitions / sec.
- N = 10^{18} per second…

$$P_{sys} \approx 1 - N \times P_{fail}$$

As we scale?

$$P_{sys} \approx 1 - N \times P_{fail}$$

- N increases
- Charge/gate decreases
  - Less electrons
  - Higher probability they wander
  - Greater variability in behavior
- Voltage levels decrease
  - Smaller barriers
- Greater variability in device parameters

→P_{fail} increases

Exacerbated at Nanoscale

- Small numbers of dopants (10s)
  - High variability
- Small numbers of electrons (10-1000s?)
  - High variability
  - Highly susceptible to noise
- Small number of molecules
  - May break, decay…

What do we do about it?

- Tolerate faulty components
- Detect faults
  - Not do anything bad
  - Try it again
    - If statistically unlikely error,
      - high likelihood won't recur.
- …Focus on detection…

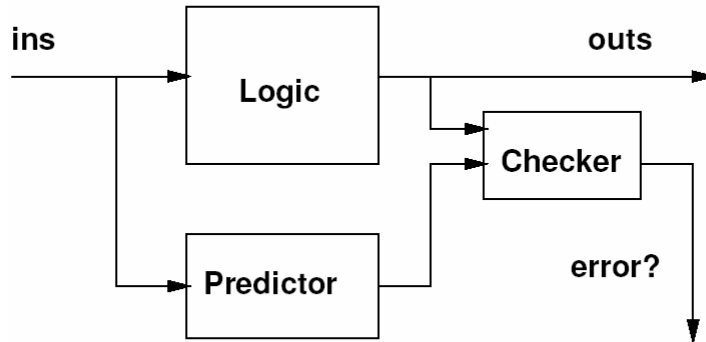
Detect Faults

- **Key Idea:** redundancy
- Include enough redundancy in computation
  - Can tell that an error occurred

What kind of redundancy can we use?

- Multiple copies of logic
- Compute something about result
  - Parity on number of outputs
  - Count of number of 1's in output

Error Detection



What do we protect against?

- Any *n* errors
  - Worst-case selection of errors

Single Error Detection

- If P_{fail} small:
  - No error: $(1-P_{fail})^N \approx 1-N\times P_{fail}$
  - One error: $N\times P_{fail} \times(1-P_{fail})^{N-1} \approx N\times P_{fail}$
  - Two errors: $[N\times(N-1)/2] \times(P_{fail})^2\times(1-P_{fail})^{N-1}$
- Probability of an error going undetected
  - Goes from $\approx N\times P_{fail}$
  - to $\approx (N\times P_{fail})^2$
  - For: $N\times P_{fail} << 1$

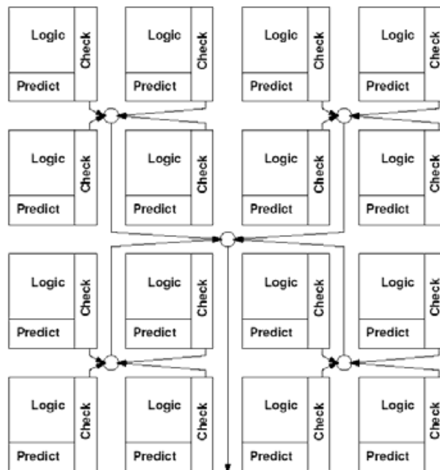
Detection Overhead

- Correction and detection circuitry increase circuit size.
- $N_{detect} > N_{logic}$
- $N_{detect} = c\ N_{logic}$
- Probability of an error going undetected
  - Goes from $\approx N\times P_{fail}$
  - to $\approx (c\times N\times P_{fail})^2$
  - Want: $c^2 << 1/(N\times P_{fail})$

Reliability Tuning

- Want $N \times P_{fail}$ small
  - Want: $(c \times N \times P_{fail})^2$ very small
- Idea:
  - Guard subsystems independently
  - Make N_{sub} suitably small
  - Smaller probability there is a double error localized in this small subsystem

Guarding Subsystems



Composing Subsystems

- $P_{sysundetect} = (N_{sys}/N_s)\ P_{subundetect}$
- $P_{subundetect} = (c \times N_s \times P_{fail})^2$
- $P_{sysundetect} = (N_{sys}/N_s)\ (c \times N_s \times P_{fail})^2$
- $P_{sysundetect} = N_{sys} \times N_s \times (c \times P_{fail})^2$
- Extermes:
  - $N_s = N_{sys}$
  - $N_s = 1$

Problem

- Generate logic capable of detecting any single error

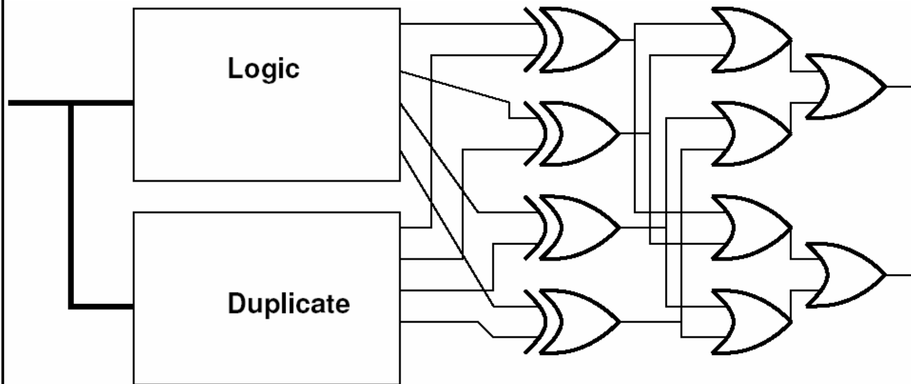
Terminology

- **Fault-secure:** system never produces incorrect code word
  - *Either* produces correct result
  - *Or* detects the error
- **Self-testing:** for every fault, there is some input that produces an incorrect code word
  - That detects the error

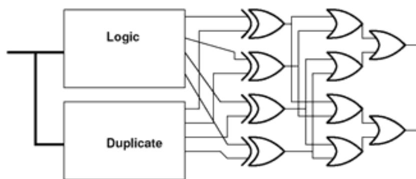
Terminology

- **Totally Self Checking:** system is both *fault-secure* and *self-testing*.

Duplication

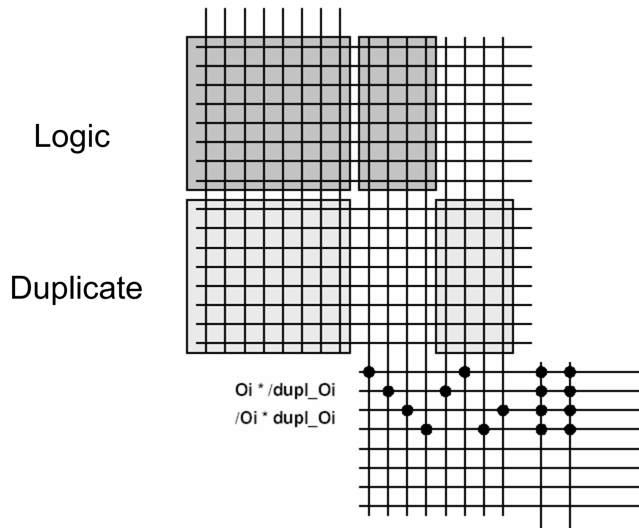


Duplication

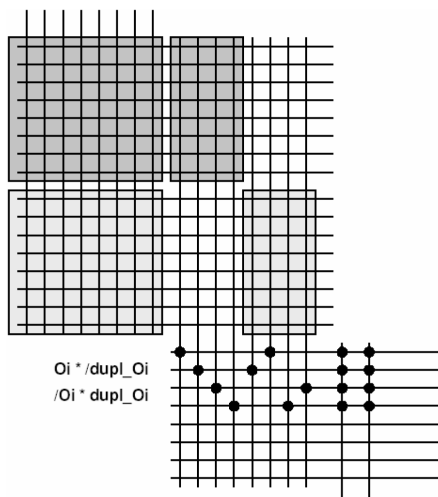


- N original gates
- Duplicate: + N
- O outputs
  - O xors
  - O/2 × 2 × 2 ors
- O<N
- 2<c<5

Duplication with PLA



PLA Duplication



- N product terms in original
- N in duplicate
- 2 O product terms for matching
- O<=N
- 2<c<4

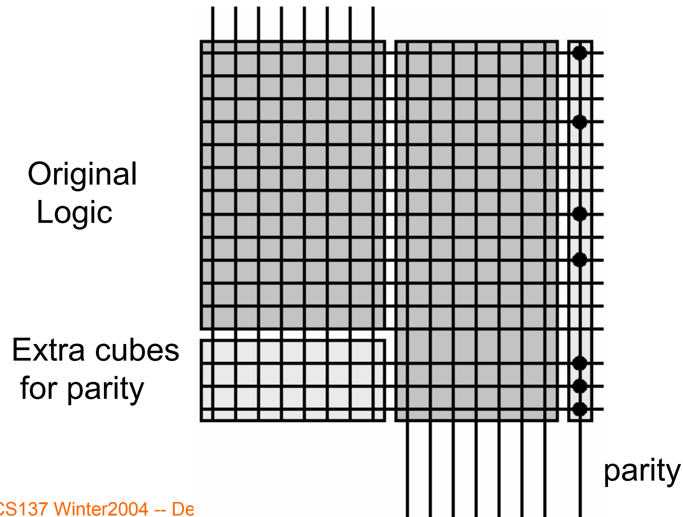
Can we do better?

- Seems like overkill to compute twice?

Idea

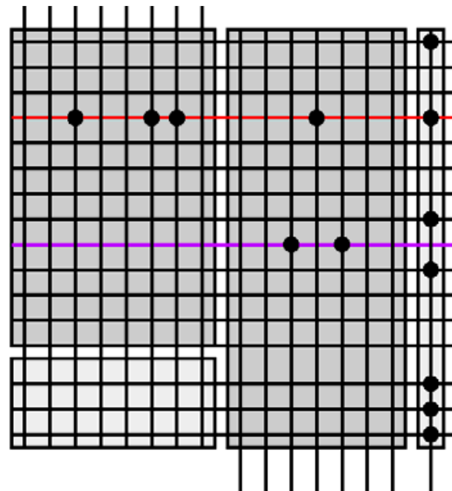
- Encode so outputs have some checkable property
  - E.g. parity

Will this work?



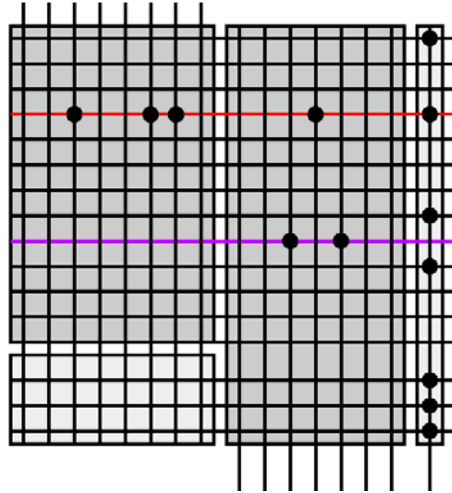
Problem

- Single fault may produce multiple output errors



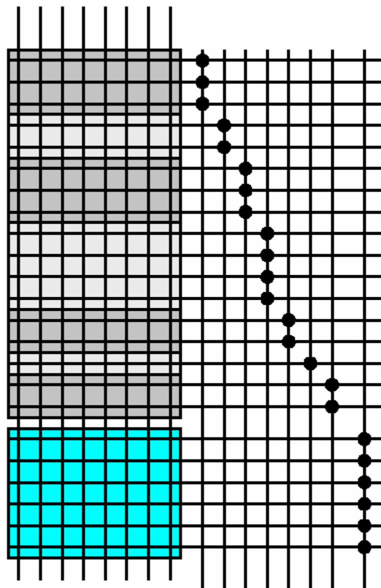
How Fix?

- How do we fix?



No Logic Sharing

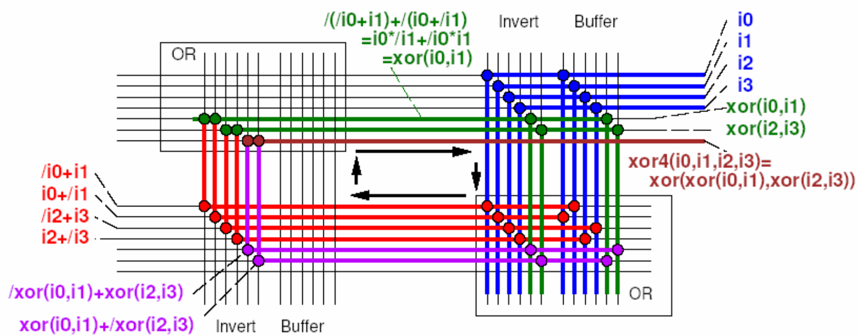
- No sharing
- Single fault effects single output



Parity Checking

- To check parity
  - Need xor tree on outputs/parity
  - $[(O+1)/2]\times2\times2 = 2(O+1)$ xors
- For PLA
  - xor would blow up
  - Wrap multiple times
  - 2 product terms per xor
  - $4\times O$ product terms

nanoPLA Wrapped xor



Note: two planes here just for buffering/inversion

Better or Worse than Dual?

- Depends on sharing in logic
- Typical results from Mitra [ITC2002]

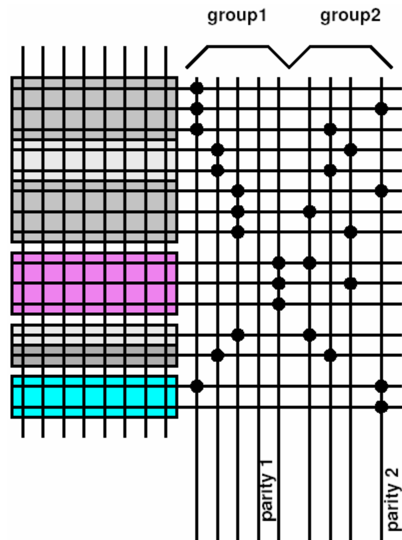
Circuit	Identical Duplex	Diverse Duplex	Parity	Berger Code	Bose - Lin
Z5xp1	**822**	836	840	1335	1068
inc	743	751	**692**	854	807
squar5	507	485	**465**	627	570
ex5.20	646	649	**593**	815	755
misex1	412	423	**367**	468	488
sao2	754	787	**748**	983	864
rd73	**474**	480	683	853	763
rd84	**642**	684	971	1135	1056

Can we allow sharing?

- When?

Multiple Parity Groups

- Can share with different parity groups
- Common error flagged in both groups



Better or Worse than Dual?

- Typical results from Mitra [ITC2002]

Circuit	Identical Duplex	Diverse Duplex	Parity	Berger Code	Bose - Lin
Z5xp1	**822**	836	840	1335	1068
inc	743	751	**692**	854	807
squar5	507	485	**465**	627	570
ex5.20	646	649	**593**	815	755
misex1	412	423	**367**	468	488
sao2	754	787	**748**	983	864
rd73	**474**	480	683	853	763
rd84	**642**	684	971	1135	1056

(parity here includes sharing)

Project Assignment

- Assignments #3 & #4
  - Out on Monday
- Provide an algorithm for identifying parity groups
  - Keep single error detection property
  - Minimize pterms

Admin

- Assignment #2 due Friday

Big Ideas

- Low-level physics imperfect
  - Statistical, noisy
- Larger devices → greater likelihood of faults
- Redundancy
- Self-checking circuits