16. Circuit Pitfalls, Resilient Systems

Jacob Abraham

Department of Electrical and Computer Engineering
The University of Texas at Austin

VLSI Design
Fall 2020

October 22, 2020
Bad Circuit 1

- **Circuit**
  - 2:1 multiplexer

![Circuit Diagram]

- **Symptom**
  - Mux works when selected D is 0 but not 1
  - Or fails at low $V_{DD}$
  - Or fails in SFSF corner

- **Principle**: Threshold drop
  - $X$ never rises above $V_{DD} - V_t$
  - $V_t$ is raised by the body effect
  - The threshold drop is most serious as $V_t$ becomes a greater fraction of $VDD$

- **Solution**: Use transmission gates, not pass transistors
Bad Circuit 2

- Circuit
  - Latch

![Diagagram of D latch](image)

- Symptom
  - Load a 0 into Q
  - Set $\phi = 0$
  - Eventually Q spontaneously flips to 1

- Principle: Leakage
  - $X$ is a dynamic node holding a value as charge on the node
  - Eventually, subthreshold leakage may disturb charge

- Solution: Staticize node with feedback
  - Or, periodically refresh node (this requires a fast clock, and is not practical for processes with big leakage)
Bad Circuit 3

- **Circuit**
  - Domino AND gate

- **Symptom**
  - Precharge gate \((Y = 0)\)
  - Then evaluate
  - Eventually \(Y\) spontaneously flips to 1

- **Principle:** Leakage
  - \(X\) is a dynamic node holding value as charge on the node
  - Eventually subthreshold leakage may disturb charge

- **Solution:** Keeper
Bad Circuit 4

- **Circuit**
  - Pseudo-nMOS OR

- **Symptom**
  - When only one input is true, \( Y = 0 \)
  - Perhaps only happens in SF corner

- **Principle: Ratio Failure**
  - nMOS and pMOS fight each other
  - If the pMOS is too strong, nMOS cannot pull \( X \) low enough

- **Solution: Check that ratio is satisfied in all corners**
**Bad Circuit 5**

**Circuit**
- Latch

**Symptom**
- Q stuck at 1
- May only happen for certain latches where input is driven by a small gate located far away

**Principle: Ratio failure (again)**
- Series resistance of D driver, wire resistance, and transmission gate gate must be much less than weak feedback inverter

**Solution: Check relative strengths**
- Avoid unbuffered diffusion inputs where driver is unknown
**Bad Circuit 6**

- **Circuit**
  - Domino AND gate

```
\[ \phi \quad Y \quad Z \\
A \quad X \quad B
```

- **Symptom**
  - Precharge gate while \( A = B = 0 \), so \( Z = 0 \)
  - Set \( \phi = 1 \)
  - \( A \) rises
  - \( Z \) is observed to sometimes rise

```
\[ V_X = V_Y = \frac{C_Y}{C_X + C_Y} V_{DD} \]
```

- **Principle:** Charge sharing
  - If \( X \) was low, it shares charge with \( Y \)

- **Solution:** Limit charge sharing
  - Safe if \( C_Y >> C_X \)
  - Or, precharge node \( X \) too
Circuit
- Dynamic gate + latch

Symptom
- Precharge gate while transmission gate latch is opaque
- Evaluate
- When latch becomes transparent, X falls

Principle: Charge sharing
- If Y was low, it shares charge with X

Solution: Buffer dynamic nodes before driving transmission gate
Circuit

- Latch

Symptom

- Q changes while latch is opaque
- Especially if D comes from a far-away driver

Principle: **Diffusion Input Noise Sensitivity**

- If $V_D < -V_t$, transmission gate turns on
- Most likely because of power supply noise or coupling on D

Solution: **Buffer D locally**
Circuit
- Anything

Symptom
- Some gates are slower than expected

Principle: Hot Spots and Power Supply Noise
Noise

Sources
- Power supply noise/ground bounce
- Capacitive coupling
- Substrate coupling
- Charge sharing
- Leakage
- Noise feedthrough

Consequences
- Increased delay (for noise to settle out)
- Or incorrect computations

Source: electronicproducts.com
Line-to-substrate coupling
Electromigration

- “Electron wind” causes movement of metal atoms along wires
- Excessive electromigration leads to open circuits
- Most significant for unidirectional currents (DC)
  - Depends on current density $J_{dc}$ (current/area)
  - Exponential dependence on temperature
  - Black’s Equation:
    $$MTTF \propto \frac{e^{\frac{E_a}{kT}}}{J_{dc}^n},$$
    where $E_a$ is the activation energy (empirically determined by stress testing at high temperatures), and $n$ is typically 2
- Typical limits: $J_{dc} < 1 - 2 \ mA/\mu m^2$
Self Heating

- Current through wire resistance generates heat
  - Oxide surrounding wires is a thermal insulator
  - Heat tends to build up in wires
  - Hotter wires are more resistive, slower
- Self-heating limits AC current densities for reliability

\[ I_{rms} = \sqrt{\frac{\int_{0}^{T} I(t)^2 \, dt}{T}} \]

- Typical limits: \( J_{rms} < 15 \text{ mA/\mu m}^2 \)

Self heating a problem for SOI circuits and 3-D systems

Modeling self heating, Silvaco
Latchup: positive feedback leading to $V_{DD} - \text{GND}$ short
- Major problem for 1970s CMOS processes before it was well understood
- Avoid by minimizing resistance of body to GND/$V_{DD}$
- Use plenty of substrate and well taps
Latchup risk greatest when diffusion-to-substrate diodes could become forward-biased
Surround sensitive region with guard ring to collect injected charge
Overvoltage

- High voltages can damage transistors
  - Electrostatic discharge (ESD)
  - Oxide arcing
  - Punchthrough
  - Time-dependent dielectric breakdown (TDDB)
    - Accumulated wear from tunneling currents
- Requires low $V_{DD}$ for thin oxides and short channels
- Use ESD protection structures where chip meets real world

Transient suppression device specifications for automotive applications

Source: dev.emcelettronica.com
Hot Carriers

- Electric fields across channel impart high energies to some carriers
  - These “hot” carriers may be blasted into the gate oxide where they become trapped
  - Accumulation of charge in oxide causes shift in $V_t$ over time
  - Eventually $V_t$ shifts too far for devices to operate correctly
- Choose $V_{DD}$ to achieve reasonable product lifetime
  - Worst problems for inverters and NORs with slow input rise time and long propagation delays

Source: Kiethley Application Note 2535
Bias Temperature Instability

Even when no carriers moving from source to drain, the gate voltage can cause charges to migrate into the insulating gate oxide.

Phenomenon is partly reversible.

Charges leave the oxide after the gate voltage is removed.

Source: Keane and Kim, IEEE Spectrum, May 2011
NBTI Mechanism and Degradation

- Hole interaction with oxide causes Si-H bonds to break
- Change is positive charge density in traps increases $V_t$
- When stress is removed, H atoms diffuse back to interface and anneal the broken bond
- DC stress causes much shorter lifetime

Source: Peters, Semiconductor International, March 1, 2004
Transistor Aging and Failure Prediction

Guardband violation due to transistor aging

Example of an aging sensor

Agarwal et al., VTS 2007
Oxide Breakdown

Mechanism

- Voltage across the gate can also cause electrically active defects within the oxide layer
- The defects can trap charges
- If enough of the charges accumulate, they can create a short, causing a catastrophic failure of the transistor

Source: Keane and Kim, IEEE Spectrum, May 2011
Static CMOS gates are very robust
  - Will settle to correct value if designed with sufficient margins and if you wait long enough
Other circuits suffer from a variety of pitfalls
  - Tradeoff between performance and robustness
Very important to check circuits for pitfalls
  - For large chips, you need an automatic checker
  - Design rules aren’t worth the paper they are printed on unless you back them up with a tool
Industry Estimates of Expected Lifetimes of Chips

Classes of Dependable Systems

- Systems Designed for Very Long Life
  - Spacecraft with multiyear missions, inaccessible systems
  - Techniques: Replication (spares), error coding, monitoring, shielding

- Safety-Critical Systems
  - Flight control computers, nuclear-plant shutdown, medical monitoring, automobile braking control
  - Techniques: Replication with voting, time redundancy, design diversity

- High-Availability Systems
  - Telephone switching centers, server farms, banking systems, e-commerce
  - Techniques: Hardware and Information redundancy, backup schemes, hot-swap, recovery

- Consumer Products?
  - PCs, PDAs, smart phones
  - Techniques: parity checks for memories, intrusion tolerance, virus detection, low cost of replacement
Historical Perspective

Dionysius Lardner

“The most certain and effectual check upon errors which arise in the process of computation, is to cause the same computations to be made by separate and independent computers; and this check is rendered still more decisive if they make their computations by different methods,” Dionysius Lardner, “Babbage's calculating engine,” *Edinburgh Review*, vol. 59, no. 120, pp. 263–327, 1834.

Key Papers in 1956

Moore and Shannon, “Reliable circuits using less reliable relays,” *Bell System Technical Journal*

Reliable Relay Networks

Shannon, 1956

Can be applied to MOS transistors

- A single open or short of a transistor (relay) will be masked by the network
- Many multiple faults will also be tolerated
Inspired by 1956 von Neumann paper for logic implemented in nanotechnologies

**Approach**

- Similar to NMR, but voting carried out in a *bundle*
- Executive stage – performs operations
- Restorative stage – reduces degradation caused by errors from the executive stage, acting as output “amplifier”
One way of detecting (and correcting) errors in data transmission and storage, is to encode data, with a subset of the words being code words.

Reasonable errors will change a code word to a non-code word, and the errors will be detectable.

Errors which transform one code word into another will not be detectable.

“Error Models” relate likely physical faults to the errors that they could cause.

Distance between two code words is the number of distinct changes needed to change one code word into the other.

Example: Parity codes.
Distance Properties

The Hamming Weight of a vector, $X$, $W(X)$ is the number of non-zero components of $X$

The Hamming Distance between two vectors, $X$ and $Y$, $d(X,Y)$, is the number of components in which they differ.

The minimum distance of a code is the minimum of Hamming distances between all pairs of code words.

To detect $d$-bit errors, need a code with distance $d + 1$, to correct $d$-bit errors, need a code with distance $2d + 1$.

Example:
Self-Checking Circuits – encoded inputs and outputs, output checker
Boeing 777 Primary Flight Computer

- THREE IDENTICAL CHANNELS: LEFT, CENTER, RIGHT
- THREE DISSIMILAR LANES IN EACH CHANNEL: ONE IN COMMAND, TWO FUNCTIONING AS MONITORS
- TWO PFCs IN THE E/E BAY, ONE PFC FORWARD OF THE FORWARD CARGO DOOR

![Diagram of Boeing 777 Primary Flight Computer](image.png)

**LEFT PFC**

- LANE 1
  - POWER SUPPLY
  - MICROPROCESSOR AMD 29050
  - ARINC 629 INTERFACES

- LANE 2
  - POWER SUPPLY
  - MICROPROCESSOR MOTOROLA 68040
  - ARINC 629 INTERFACES

- LANE 3
  - POWER SUPPLY
  - MICROPROCESSOR INTEL 80486
  - ARINC 629 INTERFACES

**CENTER PFC**

- L  
- C  
- R

**RIGHT PFC**

- L  
- C  
- R

Flight Controls ARINC 629 Data Buses
Reliability, Availability, Safety

- **Reliability (R(t))**
  - Conditional probability that a system provides continuous proper service in the interval [0,t] given that it provided desired service at time 0
  - Simple Reliability function (exponential): \( R(t) = e^{-\lambda t} \)
  - Constant *Failure Rate* \( \lambda \)

- **Mean Time to Failure, MTTF**
  - \( MTTF = \int_0^\infty R(t)dt \)
  - For an exponential reliability function, \( MTTF = \frac{1}{\lambda} \)

- **Availability A(t)**
  - Fraction of time that system is in the operational state (providing service) during the interval [0,t]
  - Function of both failure rate (\( \lambda \)) and repair rate (\( \mu \))
  - Steady-State Availability, \( A = \frac{MTTF}{MTTF + MTTR} = \frac{\lambda}{\lambda + \mu} \)

“Markov Chain” for a simple system with repair
View a system as providing a **service**

Faults $\implies$ Errors $\implies$ Failures

- **Fault**: an anomalous physical condition
- **Error**: an incorrect logic value as a consequence of the fault
- **Failure**: the condition where the system does not provide the expected service
Are “Fault Tolerance” and “Resilience” the Same?

**Fault Tolerance**
- Errors (due to faults) detected and corrected, fault located, reconfiguration around faulty unit
- System designed to tolerate classes of faults
- User does not see anything wrong (except perhaps an additional delay)
- Service does not suffer any down time

**Resilience**
- User may see errors during the service, but the final results are correct
- System requires on-line error detection, but may use checkpoints, retry, etc., to achieve resilience
- Ability to deal with “unknown” faults
- Service may be down intermittently
Example of Resilience – Single Engine Airplane

From Malibu Jetprop pilot’s operating handbook

- If loss of power occurs at altitude, trim the aircraft for best gliding angle (90 KIAS) and look for a suitable field.
- At best glide angle, no wind, with the engine stopped and the propeller feathered, the aircraft will travel approximately 2 miles for each thousand feet of altitude.
Achieving Resilience

Start with fault-free hardware
- Testing after manufacturing
- On-line tests to detect wearout and degradation

Detection is key
- Detect errors in results of computations
- Application-level results are, ultimately, what are important

Ensure correct results at the application level
- Appropriate checks at different levels of the design
- High-level checks tend to have lower overheads
Reduce the cost of fault tolerance by looking at computations at a higher level.

**Algorithm-Based Fault Tolerance (ABFT),** (Huang and Abraham, 1984)

- Encode data at a high level (application level).
- Design algorithm to operate on encoded input data and produce encoded output data.
- Distribute computation tasks among multiple computation units, so that failure of a unit affects only a portion of the output data, enabling the correct data to be recovered.

Very general fault model: A computation unit can produce any arbitrary logical output under failure.

Communication paths checked using coding techniques.
Checksum Calculations in High Performance Computing

ABFT applied to DGEMM

Source: Bosilca, Delmas, Dongarra and Langou, 2008.
Performance Under Failure

Performance (GFLOPS/sec/proc) of PBLAS PDGEMM, ABFT BLAS PDGEMM (0 failure), and ABFT BLAS PDGEMM (1 failure)
Error Resilience in Non-Linear Control Systems

Observed state data over different time samples

\[ \nu[t - n_h], \nu[t - n_h + 1], \nu[t - 2][t - 1], \nu[t], \nu[t + 1] \]

Time
Samples

\( (t - n_h), (t - n_h + 1), (t - 2)(t - 1), t, (t + 1) \)

Prediction Function
\( F_c(\cdot) \)

Mapped Predictive Check State (MPCS) \( \nu_c[t + 1] \)

Prediction Function
\( F_m(\cdot) \)

Mapped Information State (MIS) \( \nu_m[t + 1] \)

\[ + \]

Error Signal \( e[t + 1] \)
Lower costs, improved stopping distances

Electromechanical brakes (vs. hydraulic)
Brake by Wire algorithm executing on embedded processor

Result of simulation with no errors injected

\[ \Omega_i \text{ (in rad/s)} \]

\[ \text{Time (in s)} \]

0 1 2 3 4 5

0 10 20 30 40 50 60 70 80 90
When transient error is detected, results of the control loop (output to actuator) ignored for a few cycles till no error is seen.
Constraints on Embedded Systems

Myriad of Intelligent systems
- Cost, power consumption constraints
- In critical applications, *resiliency* is very important

Example: self-driving cars
- 100 Million lines of code for software, sensing and actuation
- 64 TOPS for cognition and control functions
Need a broader definition of resilience

Classic definitions of resiliency have been narrow

- Focused on hardware failures
- What about design bugs?
  - Duplication (such as in ISO 26262) will not be sufficient
- Resilience to external attacks?

ECE Department, University of Texas at Austin
Lecture 16. Circuit Pitfalls, Resilient Systems
Jacob Abraham, October 22, 2020
**Logic bugs**

- Verification is dominating the design cycle
- Unlikely that all design bugs are caught before deployment
- Diversity is necessary to deal with design bugs

**Design margins**

- Effects of real bugs are not easy to duplicate (in many cases, error latencies of many millions (or billions) of cycles)
- Gray: concepts of **Bohr bugs** (repeatable) versus **Heisenbugs** (not seen to be repeatable)

**Bugs and design margins could be exploited by an attacker**
Errors Not Repeatable – “Heisenbugs”

How to debug Heisenbugs

Don’t look at the screen!!!!!
Dealing with Security – Very Different From Dealing with Physical Faults or Errors

Attacks are **Intentional**

- Faults and Errors related to design or physical causes are **systematic** or **random**
- Attacks are **deliberate**
  - Initiated by a clever adversary
Security Attacks

Hardware Trojans
- Malicious modification of designs
- Example of analog circuitry modifying a digital chip – extremely difficult to identify
- Design diversity may be a solution

External attacks
- Classic work (Abadi) suggested control flow checking to detect execution of undesired code
- Effects of attacks could include modification of data, execution sequences, denial of service, etc.
  - Require data checks in addition to control-flow checks
  - Need to detect DoS attacks during operation – example, shutting down GPS system (or spoofing GPS position)
Example: Attacks on Automobiles

Automotive Bus – CAN

- Controller Area Bus (CAN) – robust vehicle bus standard
- Allows applications in microcontrollers and devices to communicate with other applications without a host computer

Attacks on CAN bus

- Allows the attacker to control the operation of an automobile over a WiFi network
- Attack was facilitated by the sharing of critical functions with automobile entertainment system
IC Design Process with Possible Attacks

Trojans Could be Extremely Difficult to Detect

Analog Trojan in a Digital System

- Fabrication-time attack with trigger in the analog domain
- Based on charge accumulating on a capacitor from infrequent events inside the processor
- Very small area, and low impact on power and timing
**Control Flow Deviation Detection for Application Level Security**

**Attacks subvert the control flow of the software**
- Insert control-flow checks in the code (particularly useful for embedded software)
- Run-time signatures and checks can be inserted automatically during compile time

**Implementation**
- Signature update instructions inserted at the beginning and end of each function, as well as before and after the call instructions
- Illegal branches will result in signature mismatches

**Proposed in 2005 (Abadi)**
Types of Attacks

- Control hijacking attacks
- Reverse engineering
- Malware
- Injected crafted packets or inputs
- Eavesdropping
- Brute-force search attacks
- Attacks during normal usage
Effects of Attacks

- Denial of service
- Code execution
- Integrity violation
- Information leakage
- Illegitimate access
- Financial loss
- Degraded level of protection
- Miscellaneous
Android Pixel 2 Security Module

Tamper Resistant Hardware

- Discrete chip separate from the SoC, with its own flash, RAM
- Can control its own execution, and is robust against side channel information leakage attacks
- Loads its OS and software directly from internal ROM and flash, and controls all updates
- Resilient against fault injection and side channel attacks

Source: X. Xin, Android Developers Blog, 13 November 2017