RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Efficient Flip-Flop Designs for SET/SEU Mitigation with Tolerance to Crosstalk Induced Signal Delays

-Aditya Jagirdar, Roystein Oliveira
Dept. of Electrical & Computer Engineering,
Rutgers University
Advisor: Dr. Tapan J. Chakraborty,
Alcatel-Lucent Technologies

Outline

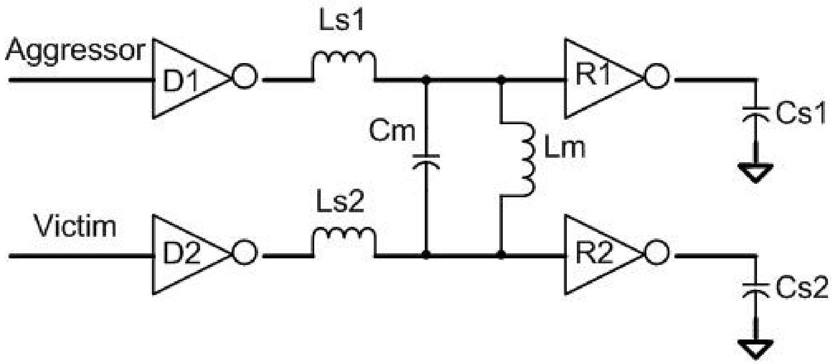
- Introduction to Soft Errors & Crosstalk
- Reliability issues
- Prior work
- Proposed designs (Design 1 & Design 2)
- Design validation
- Comparison with current designs
- Performance limitations & penalties
- Conclusion

Soft Errors

- "Soft errors induced the highest failure rate of all other reliability mechanisms combined." – Robert Baumann (TI)
- Charged particles impinging on circuit node disrupt existing node voltage
- Generated transient voltage may cross noise margins on receiving gate resulting in erroneous operation
- Many factors influence this behavior
- Fabrication technology, nature of circuit surrounding the impact site, supply voltage and intensity of radiation are main factors. Resulting effect manifested in two forms
  - Single Event Upset (SEU), Single Event Transient (SET)
- Critical charge ($\boldsymbol{Q_{crit}}$) required to affect node voltage decreases significantly with decreasing feature sizes

Interconnect Crosstalk

- Inductive + Capacitive coupling leads to crosstalk
- Increasing operational frequencies combined with decreasing interconnect lengths cause dominant inductive coupling
- Severity depends on signal transition times, interconnect spacing, operational frequency and driver sizing
- Decreasing supply voltages lead to lower noise margins resulting in more voltage disruptions at receiving end

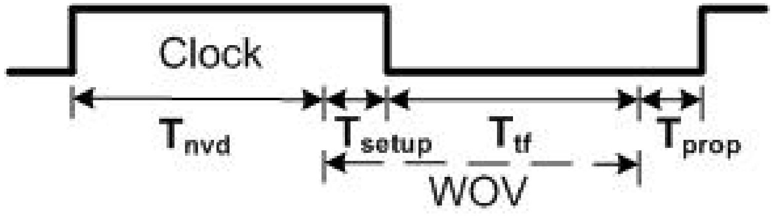


SEU/SET Effects

- A particle hit may result in either an SEU/SET depending on the position of affected node.
  - Regenerative feedback results in an SEU
  - Propagation results in an SET
- $Q_{transferred} > Q_{crit}$, particle hit on an internal node of storage element results in a bit flip
- SET must arrive at input during latching window of a storage element to corrupt data
- Period for which a latch is susceptible to particle hits is called Window of Vulnerability (***WoV***)

Window of Vulnerability (WoV) for a Latch

- Transparent when Clock=1, Opaque when Clock=0
- T_{nvd} –Period for which latch is transparent
- T_{setup}- Setup time
- T_{tf}- Period during which a hit on the internal node can flip the stored bit value
- T_{prop}- Time required for the error to propagate to latch output

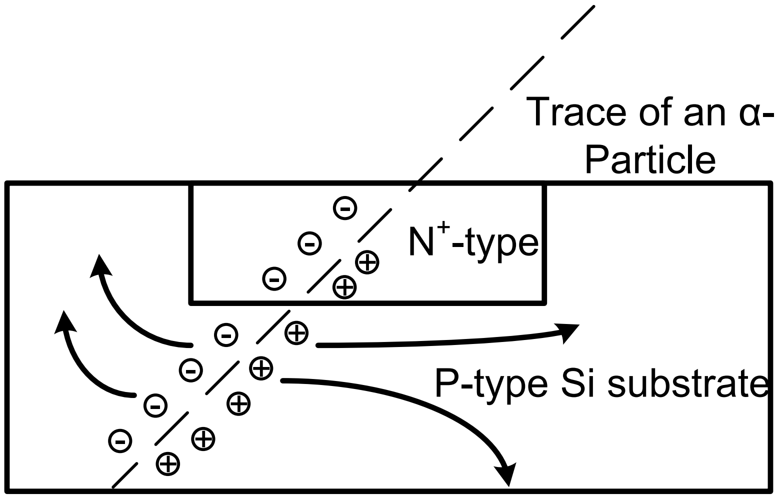


Need for Immunity from SEU/SETs

- Soft Error Rate (SER) measured in Failure in Time (FIT), 1 FIT= 1 System-level error every 10^9 hours

- Sigma-6 compliant systems (network routers, storage systems) - downtime penalty in millions

- Shrinking feature sizes /lower supply voltages aggravate this problem.



Prior Work

- Cha H. and Patel J.H., ’Latch design for transient pulse tolerance’, ICCD,1994
- Hass K.J. *et al*., ’Mitigating single event upsets from combinational logic’, NASA Symposium on VLSI Design, 1998
- D. Mavis and P. Eaton. ‘Soft Error Rate Mitigation Techniques for Modern Microcircuits’, IRPS,2002
- Mitra *et al*., ‘Robust System Design with Built-In Soft Error Resilience’, IEEE Computer, 2005
- Roy *et.al*, ‘Low-overhead design of soft-error-tolerant scan flip-flops with enhanced-scan capability’, ASPCDA,2006
- Breuer *et al*., “Test Generation in VLSI circuits for Crosstalk noise”, ITC, 1998
- Dey *et al*., “On-line Testing of Multi-source Noise-induced Errors on the Interconnects and Buses of System-on-Chips.”, ITC,2002

Drawbacks of Existing Solutions

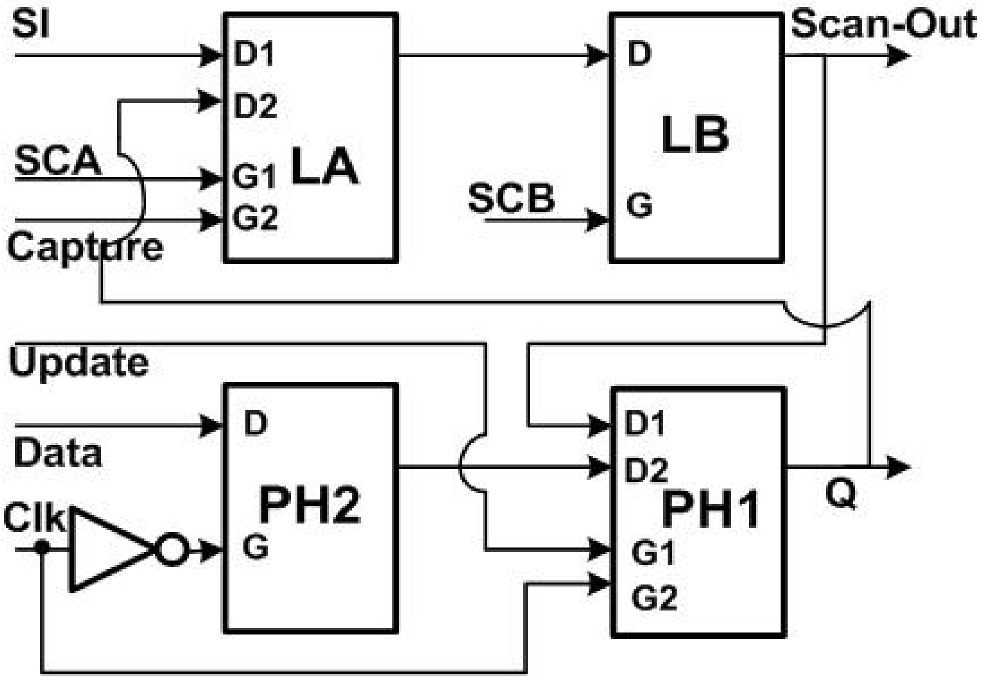
- Use of error correcting codes involves high hardware overhead and timing penalty
- ***Triple Modular Redundancy*** (TMR) involves high hardware & power overheads
- Designs developed mostly perform error blocking and only for a subset of the ***WoV***. No error correction attempted.

- Concurrent error correction mechanisms need to satisfy the following
  - Account for the entire WoV
  - Avoid a single point of failure

- For testing Xtalk sites, algorithm efficiency greatly reduces with increasing circuit size
- Detection based on wire inductance, accurate extraction commercially unfeasible

The Basic Scan Flip-Flop (BSFF)

- PH2/PH1-System Master/Slave, LA/LB-Scan Master/Slave
- Data, Clk used during functional mode of operation
- SI-Scan Input, SCA/SCB-Scan clocks, Capture-used to load system response, Update-loads test bit to PH1



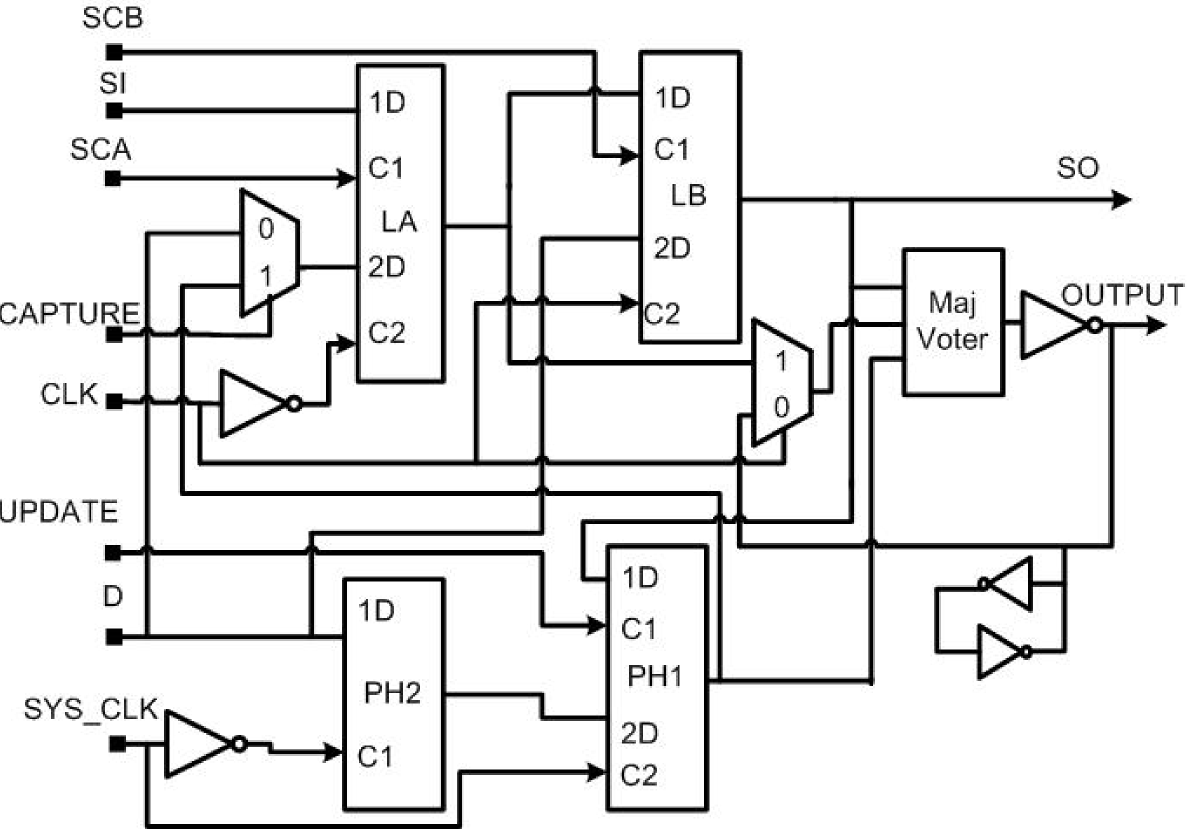
- BSFF vulnerable to
  - Incoming Transients (SET)
  - Hit on internal nodes (SEU)

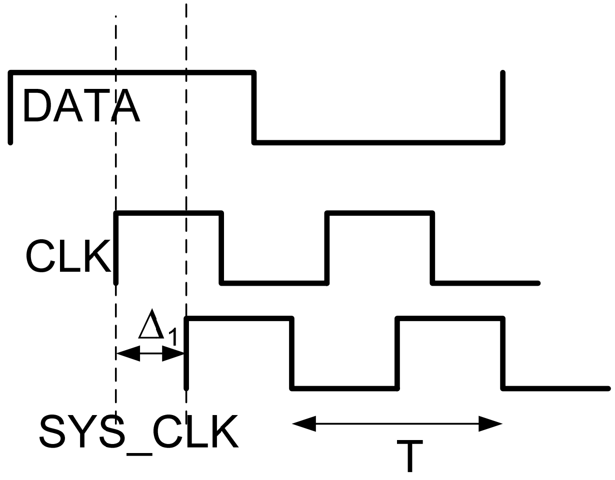
Our Work

Ideas Behind our Proposed Design

- 1 SET/SEU assumed per clock cycle
- Latches LA & LB idle during functional mode store redundant data
- Concurrent error detection/correction targeted
- Must provide flexible tradeoff between operational frequency & soft-error tolerance with minimum hardware overhead
- Modified design must be fully testable

Crosstalk & SEU Tolerant Flip-Flop (XSEUFF1)





Working of XSEUFF1

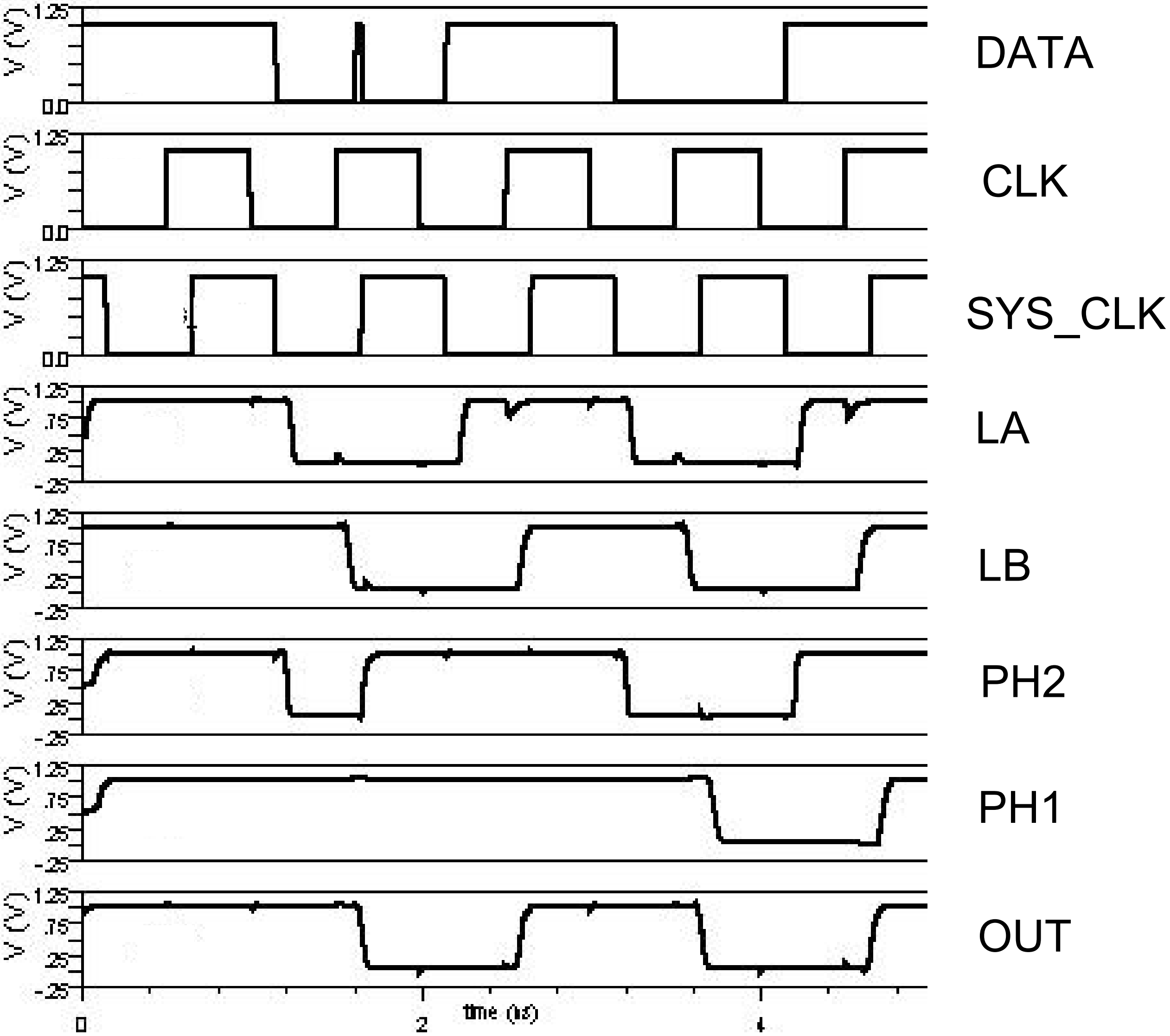
- A synchronous signal ‘Clk’ (advanced w.r.t ‘Sys_Clk’)-gating signal for scan master/slave (LA/LB)

- Data loaded first into ‘LA’ then ‘PH2’ and finally ‘LB’

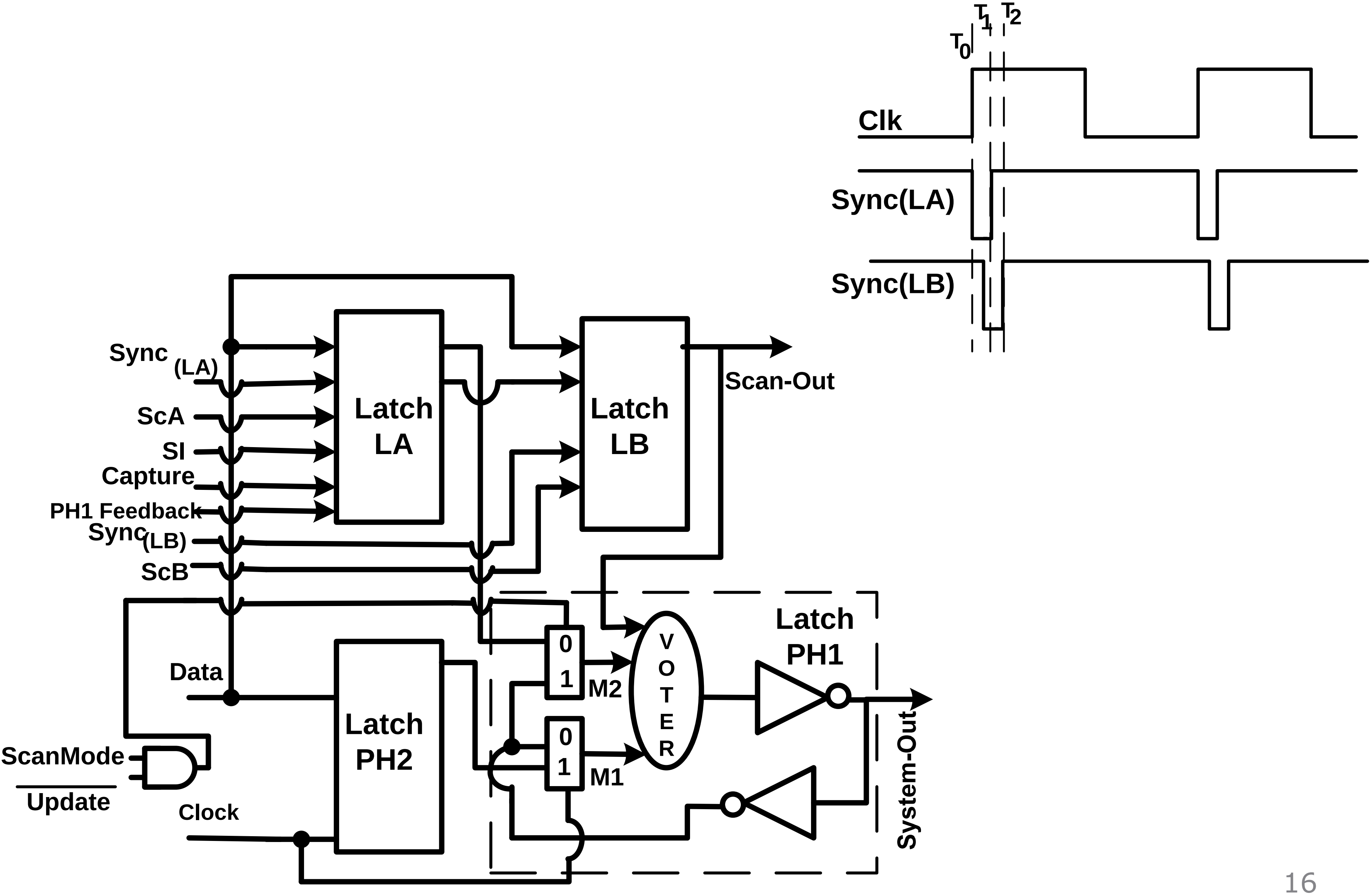
- Majority Voter (MV) loads result to keeper on output node while Clk=1-Result: Immunity from SET
- When Clk=0, keeper/LB/PH1 feed MV, protected from SEU
- SET tolerance parameter (Δ_1) is based upon the desired ***Width of Maximum Tolerable Transient*** pulse (***WMTT***)

- $\Delta_1 = t_{setup} + WMTT + t_{hold}$

Response of the XSEUFF1



The XSEUFF2


T_0
T_1
T_2
Clk
Sync(LA)
Sync(LB)
Sync (LA)
ScA
SI
Capture
PH1 Feedback
Sync(LB)
ScB
Data
ScanMode
Update
Clock
Latch LA
Latch LB
Scan-Out
Latch PH2
0
1
M2
0
1
M1
VOTER
Latch PH1
System-Out

Working of the XSEUFF2

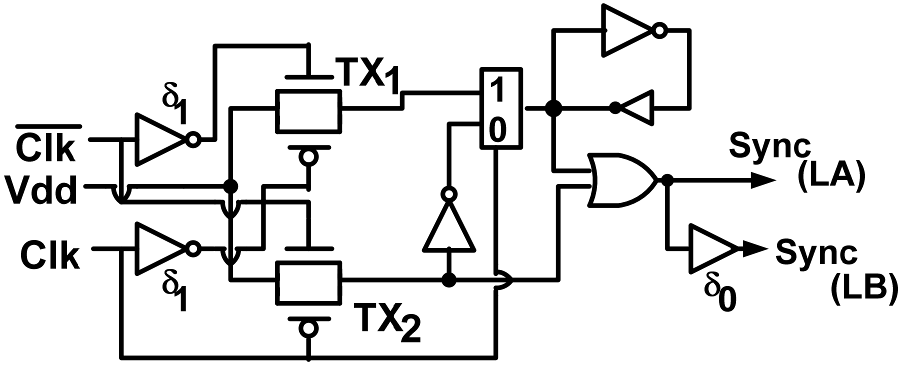
- Two synchronous signals, '$\text{Sync}_{(LA)}$' & '$\text{Sync}_{(LB)}$' control latches LA & LB respectively
- Latch PH1 modified to incorporate MV. LA and LB continuously vote on bit stored in modified PH1 when $\text{Sync}_{(LA)}=1$ & $\text{Sync}_{(LB)}=1$ (immunity from SEU)
- LA, PH1, LB vote when respective control signals are low (immunity from SETs)
- Lower transistor overhead, but need a signal generator
- Is capable of handling an SET + SEU within the same clock cycle under certain conditions

Signal Generator for the XSEUFF2

- Signals ‘$\text{Sync}_{(LA)}$’ & ‘$\text{Sync}_{(LB)}$’ generated locally

- 1 generator feeds many flip-flops, number varies with design

- Delays δ_1 & δ_0 control tolerance of XSEUFF2



Tolerance Limits of the XSEUFF2

- $WMTT = \{T_2 - t_{setup} + max(t_{cd(LB),r}, t_{cd(LB),f})\} - \{T_0 + t_{hold} - min(t_{cd(PH2),r}, t_{cd(PH2),f})\}$

- Capable of tolerating signal delays due to crosstalk effects

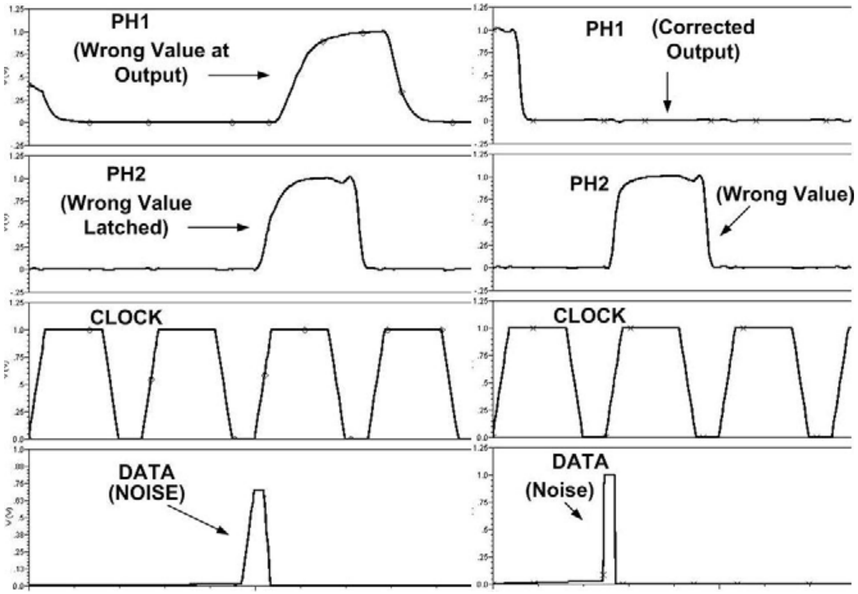
- Maximum tolerable delay (Δ_{Max}) depends on width of pulses on $Sync_{(LA)}$ and $Sync_{(LB)}$

- $\Delta_{Max} = \{T_1 - t_{setup} + max(t_{cd(LA),r}, t_{cd(LA),f})\} - \{T_0 - t_{setup}\}$

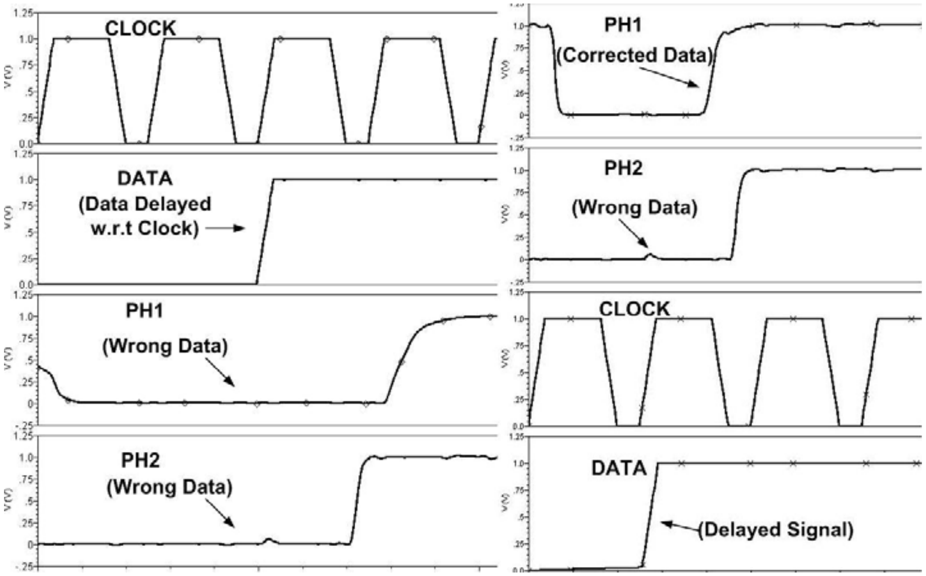
- $t_{cd(LB),r}$, $t_{cd(LB),f}$ – rise/fall times of muxes feeding data inputs of latch LB

- T_0, T_1 & T_2 arrival instants of active edges of signals Clk, $Sync_{(LA)}$ & $Sync_{(LB)}$ respectively

Response of the XSEUFF2



BSFF Vs XSEUFF2
(Noise Pulses)



BSFF Vs XSEUFF2
(Signal Delays)

Comparisons between various schemes

Type of Flip-Flop	Transistor Overhead	Timing Penalty	Power Overhead	Error Correction
BSFF (Mitra *et al*.)	1	1	1	NO
EBSFF (Mitra *et al*.)	1.13	1.23	2.77	NO
ETSFF (Mitra *et al*.)	1.15	1.01	2.13	NO
EBSHFF (Roy *et al*.)	0.91	1.25	1.72	YES
XSEUFF1	1.54	1	2.7	YES
XSEUFF2	1.37	1.25	1.95	YES

Overheads for ISCAS ’89 Benchmarks

Circuit	**XSEUFF1(%)**	**XSEUFF2(%)**
S5378	29.3	20.2
s9234	21.8	15
s13207	32.1	22.1
s15850	27.3	18.9
s35932	31.3	21.6
s38417	29.9	20.6
s38584	27	18.6
Average	**28.4**	**19.6**

Limitations & Conclusion

- XSEUFF1 & XSEUFF2 offer flexible tradeoffs between reliability and performance
- XSEUFF1 has lower timing penalty, but higher hardware/power overheads, XSEUFF2 has relatively lower hardware/power overheads, but higher timing penalty
- Need for designs tolerant to SEU/SET increasing with shrinking technology
- Better packaging offers higher SEU/SET immunity, cost increases significantly. Hence, solution must lie within the design.
- SEU immune instances can be used in conjunction with a system-level tolerant scheme to optimize performance + tolerance

Thank You!