Single Event Upset Hardening by 'hijacking' the multi-VT flow during synthesis

Roland Weigand

February 04, 2013

Design Automation Conference User Track

European Space Agency
Microelectronics Section

Author and Affiliation

Author:
Roland Weigand

Affiliation:
European Space Agency, Microelectronics Section

Address:
Keplerlaan 1, PO box 299
2201AG Noordwjik, The Netherlands

Phone: +31 71 565 32 98

E-mail: Roland.Weigand a.t esa.int

Abstract

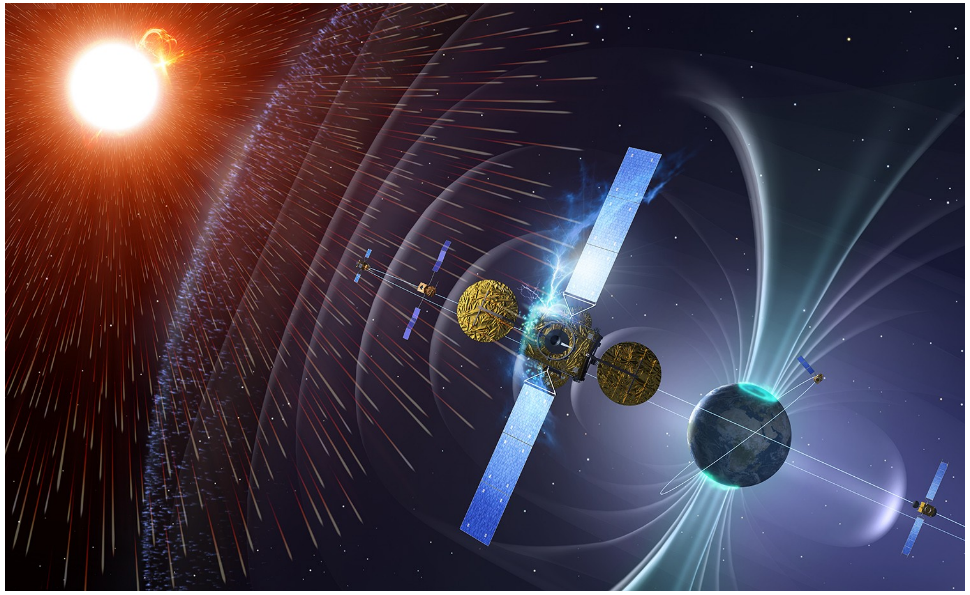
The objective of this paper is to present a new optimisation strategy when it comes to selecting Single-Event-Upset (SEU) hard flip-flops (FF) in a design. SEU are random bit flips in storage cells (RAMs and FF), induced by particles (ions, protons, neutrons) contained in cosmic rays and solar winds. These are of major concern for space applications, but increasingly, with technology downscaling, also for ground applications. SEU protection can be achieved by adding redundancy to the circuit, for example by using dedicated SEU 'hard' FF. These FF however have a price in terms of area, timing and power, and using only these hard FF, the simplest approach, may be a show-stopper. Moreover, this 'all-hard' approach often provides a level of protection, in terms of error rate, which is beyond the actual mission requirements, hence we are overdoing things. A workaround is to use the hard FF cells only for part of the design. The question is how to select the hard and the soft FF. Ideally, this should be according to functional criticality, using soft cells for those FF which are less likely to disturb the overall application of the chip. Such a criticality analysis however is often difficult to implement.

As an alternative, or even in addition to the functional criticality analysis, we propose to use an automatic selection of hardening targets based on timing considerations: hard cells are slower, so we should select the soft FF to relax the critical paths. At the same time we would like to hard-limit the overall percentage of soft FF to have clear bounds to the statistical error rate of the chip, which is a design constraint for space applications. The analogy with the multi-VT flow becomes apparent: we have pairs of cells (SVT/LVT), the LVT are faster, but have higher cost (leakage power). In our analogy (SVT = hard-FF, LVT = soft FF), the soft FF are faster, but have a higher error rate. We demonstrate that with small modifications in the .lib files, it is possible to use the multi-VT synthesis to optimise and trade-off timing performance against SEU hardening. Some limitations were detected, namely how to transfer this strategy into the backend flow, and how to optimise the 'real' leakage power, if the library parameter is 'hi-jacked' for the SEU error rates.
As a possible solution, a script-based approach will be explored in the future. We may also raise the question whether EDA vendors could offer additional optimisation features, allowing to assign custom parameters to the library and custom functions to the overall cost function used during optimisation.

Outline

- Single Event Effects (SEE) in Space Applications [1]
  - Cosmic Rays, Heavy Ions, Single Event Upsets (SEU)
  - SEU hard flip-flops
  - Calculating Error Rates
- Classical Approach for SEU Hardening
  - Hardening of FF selected by criticality analysis or fault injection
  - Hardening of all FF (brute force): high cost (area, power, timing)
- Proposed Approach: Automatic Partial Hardening
  - Optimisation of hardening vs. timing, area and power
  - Such feature does not exist → 'hijacking' multi-VT flow
  - Implementation: patching .lib file, synthesis scripts
  - Examples, results – and limitations
- Conclusion
  - Need for a dedicated custom optimisation feature in EDA tools

Single Event Effects (SEE) in Space



Cosmic rays or solar winds cause ions, protons, neutrons

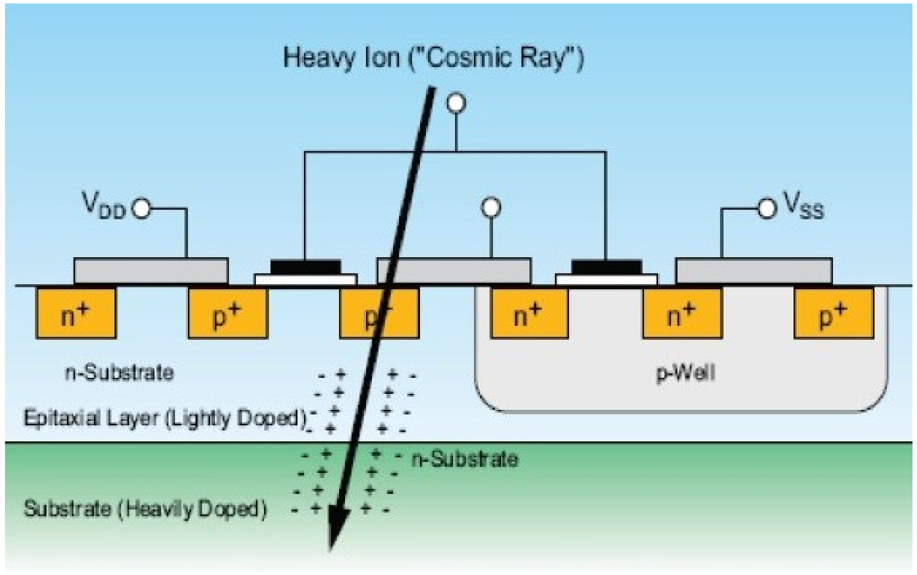
On earth, most particles are filtered by the atmosphere, yet some effects are observed on planes, and, with technology downscaling, even on ground

Particle impacts cause charge generation in the semiconductor

Glitches entering internal nodes of a latch/FF may cause bit-flips

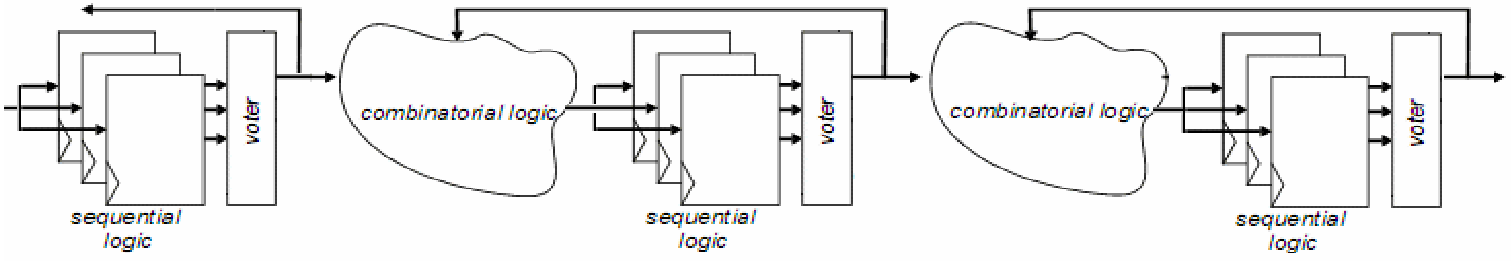
→

the **Single Event Upset (SEU)**



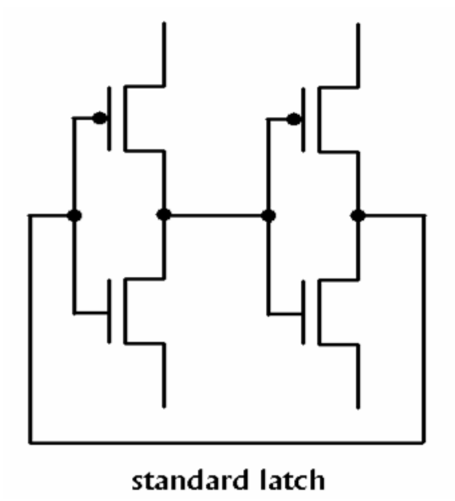
SEU Hardened Latches and Flip-Flops

- Protection by (Triple-Modular) Redundancy (TMR)



TMR design flow, previously presented in DAC User Track 2009 [2]

- Protection by dedicated SEU-hard storage cells

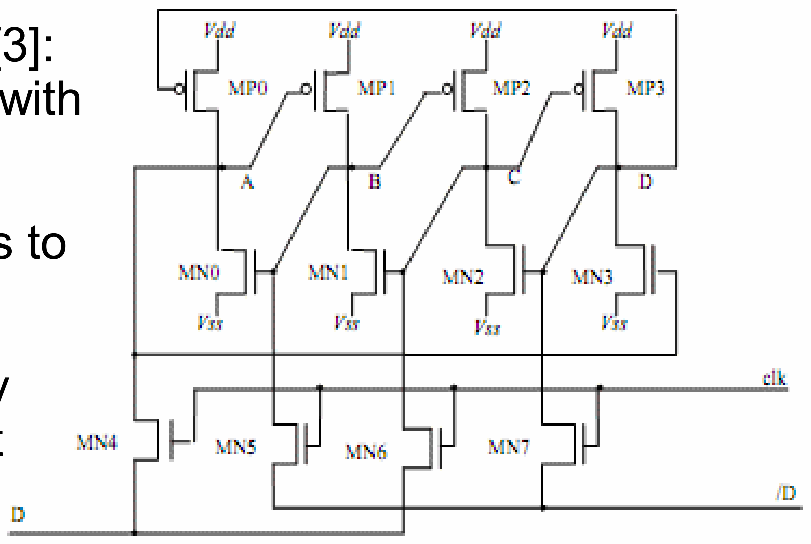


Example: the DICE latch [3]: duplicated internal nodes with bidirectional feedback

Needs glitch on two nodes to cause a bit-flip

- A space ASIC cell library usually contains both, soft (=normal) and hard cells

- Increased area, power and delay (c/w soft FF)



Calculating Single Event Error Rates

- 100% hard impossible, a residual error rate (ER) remains
  - Goal: ER ≤ probability of the system to fail for other reasons
  - Goal: ER should be low compared to the mission duration
- Chip error rates usually calculated in 'errors / device / day':

$$\mathbf{ER = \Sigma\, p_i * a_i} \quad \text{(sum over all bits in the chip)}$$

p_i = bit error rate (= probability for one bit to flip)
a_i = probability of bit flip to cause a functional error

- Assuming only two classes of bits: soft and hard flip-flops, and assuming a_i = 1 (to get worst case error rate):

$$\boxed{\mathbf{ER = p_s * N_s + p_h * N_h}}$$

p_s , p_h = soft/hard FF error rate; N_s , N_h = number of soft/hard FF

Selective Hardening of FF

- Use SEU hard FF only for a subset, selected by a criticality analysis, identifying those cells which are most likely to cause application failures (those cells with highest a_i), e.g.
  - Use hard FFs for state machines (e.g. the PC and condition flags of a microprocessor)
  - Use soft FFs in data paths (e.g. in a spread-spectrum data path, bit-flips only cause negligible noise)
- FPGA Fault injection can be used to determine critical FF [4]

Selective hardening, based on criticality analysis, is the best approach to find the trade off between "reliability ↔ resource use ↔ performance"

- Criticality analysis is often tedious and not always convincing for PA people who like to see 'simple' probability calculations

Defining all FF as critical is often the easiest solution...

SEU Hardening – Brute Force Method

- "Brute force" SEU hardening: Use **<u>only</u>** hard FF

$$ER = p_s * N_s + p_h * N_h \quad \text{becomes} \quad ER = p_h * N_{ff}$$

with $N_s = 0$ and $N_h = N_{ff}$ (total # of FF in design)

- Best radiation tolerance, easy assessment / implementation

→ set_dont_use {my_space_asic_lib/dff*}

- Huge overhead: Hard FF are 2-3x as large as soft FF
  - Increased area, power and delay (= lower QoR)
  - Designs may hit the complexity ceiling and/or miss the performance and power goals
  - Often leads to over-protection
- Goal: find a trade-off between Error Rate and QoR

What about partial hardening without criticality analysis?

Proposed: Automatic SEU Hardening

- Timing driven automatic selection of hardening targets
  → by default, use the larger and slower hard cells,
  → on critical paths, use smaller and faster soft cells
- Formula remains applicable: $ER = p_s * N_s + p_h * N_h$
- Which algorithm to use for the automatic selection?
- Dedicated SEU protection features are scarce in EDA tools, but a similar algorithm is **multi-VT leakage optimization**
  - LVT (fast, high leakage) <=> soft-FF (fast, higher error rate)
  - SVT (slow, low leakage) <=> hard-FF (slow, low error rate)
- Implementation principle
  - Annotate p_s and p_h as leakage power to the single-VT libraries
  - Define pairs of soft-/hard-FF as LVT-/SVT pairs
  - Multi-VT synthesis with lvth_percentage $(= N_s / N_{ff})$ as a hard constraint to achieve the desired error rate (ER)

Implementation(1): Patching .lib files

soft flip-flop

```
cell (dff) {
  cell_footprint : "dff";
  area : 62.72;
  cell_leakage_power : 0;
```

functionally equivalent hard FF

```
cell (hdff) {
  cell_footprint : "hdff";
  area : 235.2;
  cell_leakage_power : 0;
```

annotate error rate of soft-FF as cell_leakage_power

define (fake) foot-print equivalence dff <–> hdff

annotate error rate of soft-FF

```
cell (dff) {
  cell_footprint : "dff";
  area : 62.72;
  cell_leakage_power : 0.7;
  threshold_voltage_group : LVT;
```

```
cell (hdff) {
  cell_footprint : "dff";
  area : 235.2;
  cell_leakage_power : 0.0009;
```

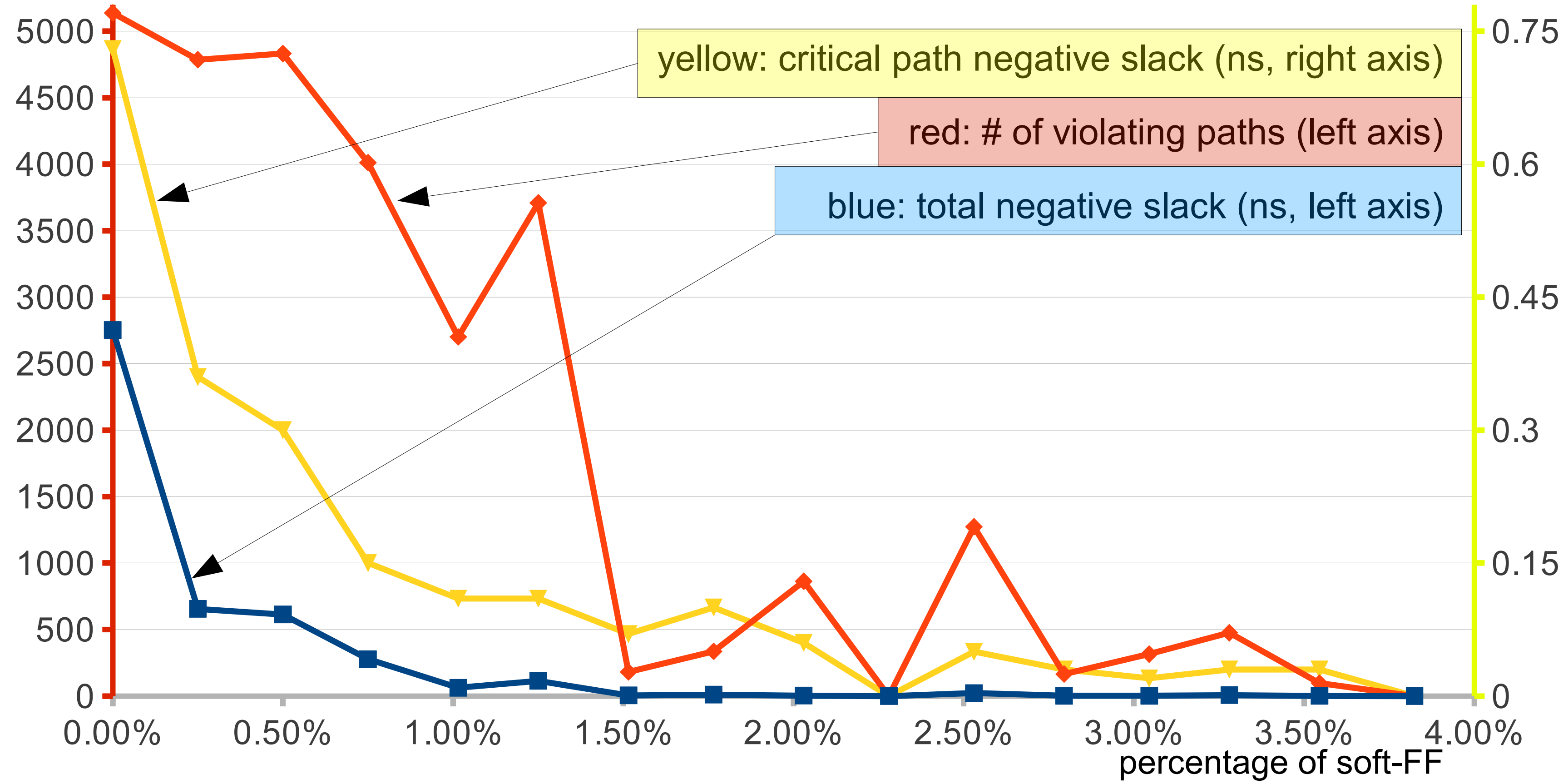
assign the soft-FF (faster but higher error rate) to the LVT group

Implementation(2): Synthesis script

```
    # 1st synthesis using brute-force method:
    # use only hard flip-flops = disallow soft flip-flops
    set_dont_use [find lib_cell xyz_lib/d*]
    set_dont_use [find lib_cell xyz_lib/s*]
    set_dont_use [find lib_cell xyz_lib/l*]
    compile_ultra -timing_high_effort_script -scan


    # group sequential and combinatorial cells in separate blocks
    group [find reference h*] -design_name allff -cell_name iallff
    characterize iallff

    # 1st re-compile: only flip-flops, allow all seq. cells, apply hard lvth %
    current_design allff
    remove_attribute [find lib_cell ATC18RHA_CELL_slow_1p65v_145c/*] dont_use
    set_multi_vth_constraint -type hard -lvth_groups "LVT" -lvth_perc ${PERC}
    set_max_leakage_power 1000000000
    compile_ultra -incremental > reports/compile-${PERC}.log

    # 2nd re-compile: FF's are now dont_touch
    current_design leon
    set_multi_vth_constraint -reset
    set_dont_touch allff
    compile_ultra -incremental -timing_high_effort_script

 }
```

Results

- DUT: LEON2 processor [5], simple config (no caches), ~ 7000 FFs, clock target 7 ns, DC G-2012.06-SP4, Atmel 180nm single-VT space libraries [6]
- Significant critical path improvement with ~ 1% of soft-FF. Gain limited by combinatorial path length. Some oscillations observed.



Problems and limitations

- lvth_percentage constraint is calculated over the whole design, including those cells which do not have a LVT equivalent → we would like to consider flip-flops only
- DC commands cause warning messages 'will be obsolete in a future version' → will multi-VT still be supported in the future?
- DC optimization strategy: lvth_percentage is not really a hard constraint. During compile, the percentage may go far beyond the specified value, leading to a different implementation of combinatorial logic. During final phase, the % is fixed, but the final timing result may be worse than with all hard-FF. Workaround: perform multi-VT optimization only on the FFs.
- Pairs of soft-/hard-FF are not truly footprint equivalent. This is likely to pose problems in a physical flow. In our example, a fixed wire-load-model was used.
- With lvth_percentage of typically 1% or less, there is little improvement of area / power (most FFs remain hard cells)

Future Work

- Use automatic hardening in physical synthesis and backend, possibly in a MMMC ECO flow. Soft/hard FF's, are NOT footprint equivalent, this may cause trouble in physical tools.

- Use the knowledge we already have of the design:
  Can we combine multi-VTH synthesis with hard constraints?
  E. g. some critical flops MUST be hard, and others SHALL be soft because we already protect them by other means: e.g. Error Detection And Correction (EDAC).

- We have hijacked the multi-VT flow with a single-VT library. What to do with 'real' multi-VT libraries? How to combine 'real' leakage optimisation and automatic selection of soft-/hard flip-flops?

Conclusion

- Single Event Effects cause bit upsets to our designs
  - in space, but increasingly also on ground applications
- Designs can be hardened by using special flip-flop cells
  - hard cells are slower, larger and consume more power
  - using only hard cells is sometimes not possible / desirable
- Proposed automatic timing driven selection of hardening targets by 'hijacking' the multi-VT synthesis flow
- Some limitations of the proposed method can be overcome by additional constraints
- As next step, a script-based approach will be developed to be used also in back-end
- Message to EDA vendors:
  A custom optimization feature could be useful.

References / Links

[1] ESA Handbook on Techniques for Radiation Effects Mitigation in ASICs and FPGAs,
http://microelectronics.esa.int/handbook/

[2] Design of a Single Event Effect fault tolerant microprocessor for space using mainstream commercial EDA tools
http://microelectronics.esa.int/papers/DAC2009-SEU-TolerantMicroprocessor-RolandWeigand-Slides-v2.pdf
http://microelectronics.esa.int/papers/DAC2009-SEU-TolerantMicroprocessor-RolandWeigand-Paper-v2.pdf

[3] T. Masson and R. Ferrant, Memory insensitive to disturbances, US Patent 5,570,313
http://www.freepatentsonline.com/5570313.pdf

[4] Workshop on Fault Injection & Fault Tolerance in space FPGAs, Sep. 11, 2009
http://www.esa.int/TEC/Microelectronics/SEMV57KIWZF_0.html

[5] The LEON2(-FT) Microprocessor
http://en.wikipedia.org/wiki/LEON
http://www.esa.int/TEC/Microelectronics/SEMUD70CYTE_0.html
http://www.iaik.tugraz.at/content/research/vlsi/archive/isec/downloads/packages/leon2-1.0.32-xst.tar.gz

[6] Atmel ATC18RHA 180 nm ASIC library for space applications
http://www.atmel.com/devices/ATC18RHA.aspx