Design of a Single Event Effect fault tolerant microprocessor for space using mainstream commercial EDA tools

Roland Weigand, European Space Agency, Roland.Weigand@esa.int
Jean Edelin, Atmel Aerospace, Jean.Edelin@atmel.com

Introduction

Integrated circuits (IC) used in a space environment are exposed to cosmic radiation, causing several adverse effects in the IC. Total Ionising Dose (TID) effects cause a slow degradation of threshold voltages and carrier mobility, leading to increased leakage currents and reduced circuit speed. These effects are mostly mitigated by process variants, adequate geometries and guard-rings in library cells, and by taking sufficient design margins (derating), anticipating an end-of-life degradation. They do not affect the HDL to GDS design flow and are not further discussed here. The second group of radiation effects, called Single Event Effects (SEE), are caused by the interaction of cosmic radiation (heavy ions) with the semiconductor material, generating electron-hole pairs leading to voltage peaks (glitches) within the drains of CMOS transistors. With decreasing capacitance of circuit nodes in advanced technologies, the SEE sensitivity increases.

SEE effects can be mitigated by various techniques. Direct modifications of the manufacturing process or of the existing standard cell library are often unaffordable, and SEE hardening has to be introduced “by logic design”, within the HDL source code, in the gate level netlist, or during the physical (layout) design. “Hardening by design” (HBD) techniques usually involves some form of spatial or temporal redundancy. But adding redundancy for SEE mitigation is often in conflict with the built-in algorithms of existing mainstream EDA tools, designed to optimise away any redundancy and delays in the design. Also is verification and test of fault-tolerant designs not a trivial task, because redundancy may hide real defects.

The Atmel AT697 Microprocessor [1], designed in the Atmel ATC18RHA [2] 180 nm ASIC technology, is based on the LEON2-FT microprocessor IP core [4], and various HBD techniques were applied. Based on this real-life experience, the objective of this paper is to report on implications of HBD in a standard cell ASIC design flow, discussing compatibility, tricks and pitfalls in conjunction with the use of commercial EDA tools. As dedicated EDA tool development for our niche market is not affordable, problems have to be solved with appropriate scripting and constraints.

SEE mitigation, well known to the aerospace community, is of increasing interest also for non-aerospace designs, in particular for high-reliability applications, such as automotive, medical or network controllers. It is therefore believed that our contribution is of general interest for the auditorium of the DAC User track.

Single Event Upsets in Flip-Flops

If a SEE induced glitch occurs within a storage point, a flip-flop (FF) or a memory cell of the circuit, it immediately causes a static bit flip, called Single Event Upset (SEU), which may lead to malfunctioning. Some standard cell libraries [2] provide specific SEU hardened FF (HDFF). These HDFF, usually based on internal redundancy such as the DICE cell (Drawing 3), can be easily introduced into the design flow e.g. by direct instantiation in the HDL source code, by constraints during synthesis, or by replacement in the netlist after synthesis. The easiest way is to synthesise the whole design only using HDFF, setting a *dont_use* constraint on all non-hardened library FF. But HDFF are usually larger and slower than the standard FF, and optimisation tools naturally try to replace them by their faster and smaller non-hardened equivalents. Care must be taken to properly constrain all optimisation steps within the HDL to GDS flow, including also the re-synthesis during layout design. During one real-life development, it was found only during the tape-out review that all HDFF had disappeared.

If no hardened cells are available, then SEU protection of FF can be done by Triple Modular Redundancy (TMR) at FF level. Each FF is replaced by three instances connected to the same data input, while their output is connected to a majority voter (Drawing 1). One SEU error per TMR triplet will be corrected and does not affect the functionality. To avoid accumulation of errors over time, the use of flip-flops with local enable MUX must be avoided, and, when clock gating is used, periodic activation of the clock for refresh cycles must be ensured. The TMR FF can be inferred from an appropriate description within the HDL code, or by replacement of FF with TMR macros (3 FF and voter) in the netlist. Both techniques have been applied during our development [1]. Implementation in HDL is straightforward, provided sequential logic can be easily located in the HDL design, as is the case for example with the “two process method” described in [3], applied in the LEON2-FT VHDL core [4]. It must be ensured then that “register merging” features in synthesis tools are disabled. But if FF inference is scattered in the source code, as it is sometimes the case in commercial IP cores not developed to a suitable coding style, identifying and replacing all FF in the source code can be a cumbersome task. Smart scripting (*grep, perl, awk, ...*) can help, but in the end, a thorough verification of the modifications is required, making sure that TMR is properly implemented, while functionality is not affected. For these designs, replacement at netlist level might be more adequate. Netlist editing however impedes a simple, straightforward top-down synthesis flow, and it causes a conflict during formal verification HDL versus netlist (1 FF is replaced by 3).

Single Event Transients in Combinatorial Logic

Radiation induced glitches in the combinatorial logic are commonly called Single Event Transients (SET). A SET in a

synchronous design is only harmful, when it happens to reach the data input of a FF during the sensitive clock transition, it can then cause a bit flip or metastability within this FF, and therefore malfunctioning. The sensitivity to SET in combinatorial logic is therefore proportional to the clock frequency. In developments with low clock rates, the effect is hardly measurable compared to the effect of SEU in FFs. In our project however, with a target frequency of 100 MHz in 180 nm technology, the effect was considered to be non-negligible, and mitigation against SET in combinatorial logic was required. Special care must be taken also of global asynchronous nets (clocks/resets).

The scheme depicted in Drawing 2 [5], developed for SRAM based reprogrammable FPGA, where the structure of the logic itself may be subject to SEE induced modification, also protects against SET. Its implementation in HDL source code however is very delicate, because elimination of combinatorial redundancy is the most elementary optimisation feature of any EDA tool, very specific coding style and constraints are required if we want the redundancy to survive synthesis. The combinatorial TMR is furthermore very expensive in terms of resources.

The SET protection scheme chosen in our project is shown in Drawing 4. Beyond the use of the above mentioned TMR triplet in the whole design, the clock net was split in three separate clock trees, each domain is driving one of the FF per TMR triplet. At the origin of the three clock trees, delay cells are inserted, causing a skew δ from one clock to another. The SET pulse, arriving at the common D input, can upset only one of the FF (Q1 in this case), and this upset will be corrected by the majority voter. The temporal filtering only works, if the value of δ is large enough compared to the width of the SET pulse. As a side effect of temporal filtering, the slack available for combinatorial logic will be reduced by $2*\delta$, effectively reducing the operating frequency, and on the other hand, the clock skews can produce many hold violations, depending on the value of δ. The choice of δ is therefore a trade-off between SET protection (the higher the better) and the performance of the chip, in our project, a value of around 500 ps (typical conditions) was chosen. This protection scheme has several implications onto the design flow:

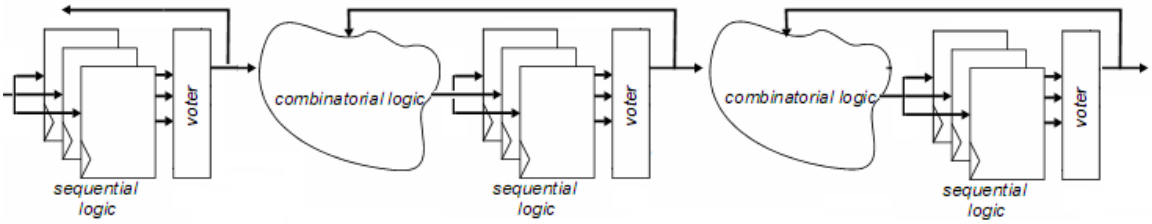
- SET protection can be spoiled by additional skews on the data path between D1, D2, D3, or different latencies of the three clock trees, compensating the desired skew δ. A solution to this problem would be to perform a coherent clock tree synthesis (CTS) and layout of the the three TMR domains, achieving close vicinity placement of the three FF per TMR triplet and of the buffers in the three clock trees. As such an option is not foreseen in the P&R and CTS tool, coherence needs to be improved by appropriate constraints to the CTS. Nevertheless, a thorough evaluation of the result needs to be done in post-layout timing analysis, determining locally for each FF (i = 1, 2, 3) in a TMR triplet the arrival times of clock $t_{ck}(i)$ and data $t_d(i)$. The goal is that the differences $[\ t_{ck}(2) - t_d(2)\ ] - [\ t_{ck}(1) - t_d(1)\ ]$ and $[\ t_{ck}(3) - t_d(3)\ ] - [\ t_{ck}(2) - t_d(2)\ ]$ are as close as possible to δ.
- During hold fix, as shown in Drawing 5, the back-end tool tries to fix the violation without affecting any other path, it inserts hold-buffers just before the violated FF (FFB3 in this case). A SET arriving at the common D input will be delayed by the same amount as the clock, and the temporal filtering effect is destroyed. The solution in our project was to group each TMR triplet in a separate design unit and set *dont_touch* attributes for the hold fixing phase. The hold buffers are then inserted as shown in Drawing 6, ensuring same delay for all three FF within a TMR triplet. It should be noted that this creates several ten-thousands of design units and constraints, which sometimes may cause tool problems such as crashes or excessive memory usage.
- The massive occurrence of hold violations in our design led to a very high number of hold buffers. Despite area in our pad-limited chip was not critical, this was undesired, because the amount buffers is not easy to integrate in a late stage of the back-end flow, and they may lead to local re-layout, degrading the max/setup performance. The hold buffers also increase power consumption, and the cross-section for SEE, thus reducing SET fault tolerance. Min/hold timing analysis showed that a vast majority of the violated paths start and end in the same TMR triplet with the same phase (non-inverted). This is a hold/enable functionality, typically used for configuration registers. In our design [4], there was massive use of hold FF, because the execution unit of the microprocessor core has a global hold signal, allowing to stall operation e.g. in case of cache miss. The short path between FF output and its own input is hold-violated as soon as $\delta > [$ T(FF clock-to-output) + T(Voter) + T(hold MUX) – T(hold guard band)] . Several trials with different value for δ and for the hold guard-band (HGB) showed that there was indeed a threshold for a massive increase of hold buffering. These local (same-phase) feedback paths do not need to be hold-fixed, because an enable-FF in the hold state has no transition on its output susceptible to cause metastability. It was therefore considered to use *false_path* constraints. But the option was discarded, because there was uncertainty on how to identify the paths which are safe to disable. A toggle flip-flop for example, looks very similar, the only difference is that data polarity is inverted, but hold fixing of toggle flip-flops must certainly not be omitted. Another concern was the stability of the tools with ten-thousands of *false_path* constraints. It was finally decided to trade-off between SET protection (value of δ), cost/ performance (number of buffers), and production test yield (HGB), and select an appropriate set of δ and HGB.
- One solution which could solve the problems described here would be to define the TMR cell from Drawing 4 (excluding the delay cells) as a library cell. After consultation with EDA tool vendors it appeared impossible to write a library description of a FF with three concurrent clock inputs that layout tools can handle. A bussed clock tree synthesis tool would be required to accompany such cell.

Conclusion

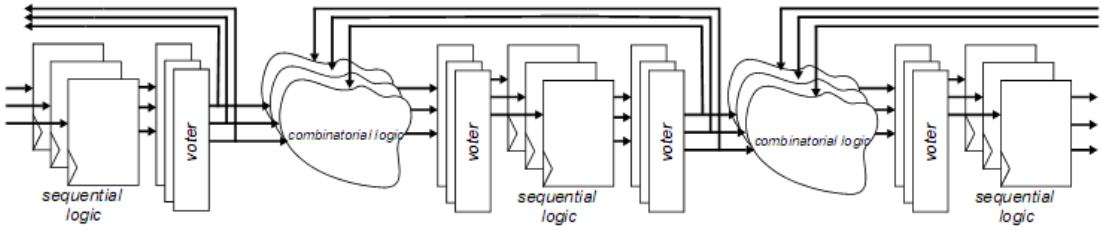
This paper presents Single Event fault tolerant design techniques applied for the Atmel AT697 rad-hard microprocessor, discussing in particular some of the conflicts encountered, when the built-in optimisation algorithms of EDA tools try to remove the redundancy features which are desired in space products, but increasingly also in other high-rel applications. Radiation testing was used to confirm the efficiency of the fault protection approach. More aspects will be shown in the presentation, such as the protection of asynchronous resets, fault-tolerance of on-chip RAM blocks, verification of designs with redundancy by simulation and by formal verification.

Links and Drawings

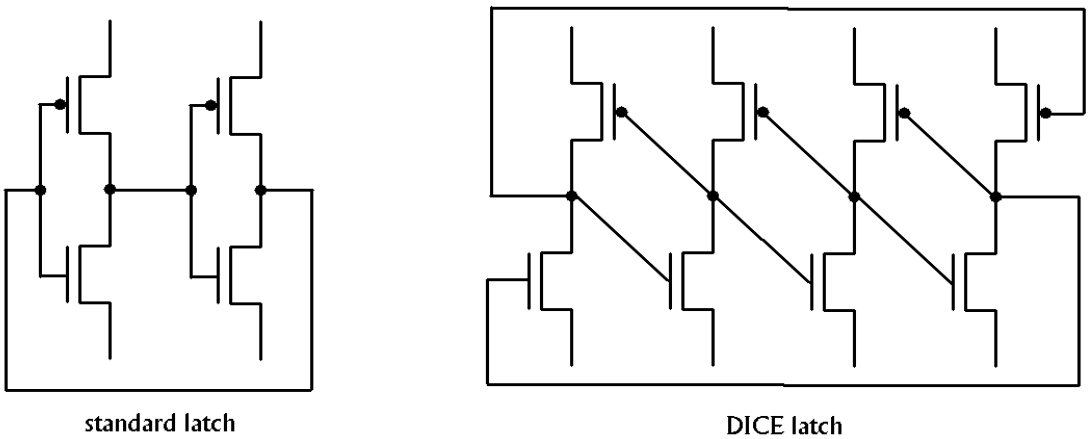
[1] AT697 page at Atmel http://www.atmel.com/dyn/products/product_card.asp?part_id=3178
[2] ATC18RHA page at Atmel http://www.atmel.com/dyn/products/product_card.asp?part_id=2318
[3] A structured VHDL design method, Jiri Gaisler, http://www.gaisler.com/doc/vhdl2proc.pdf
[4] The LEON2-FT IP core http://www.esa.int/TEC/Microelectronics/SEMUD70CYTE_0.html
[5] Functional Triple Modular Redundancy (FTMR), Sandi Habinc, December 2002
http://microelectronics.esa.int/techno/fpga_003_01-0-2.pdf



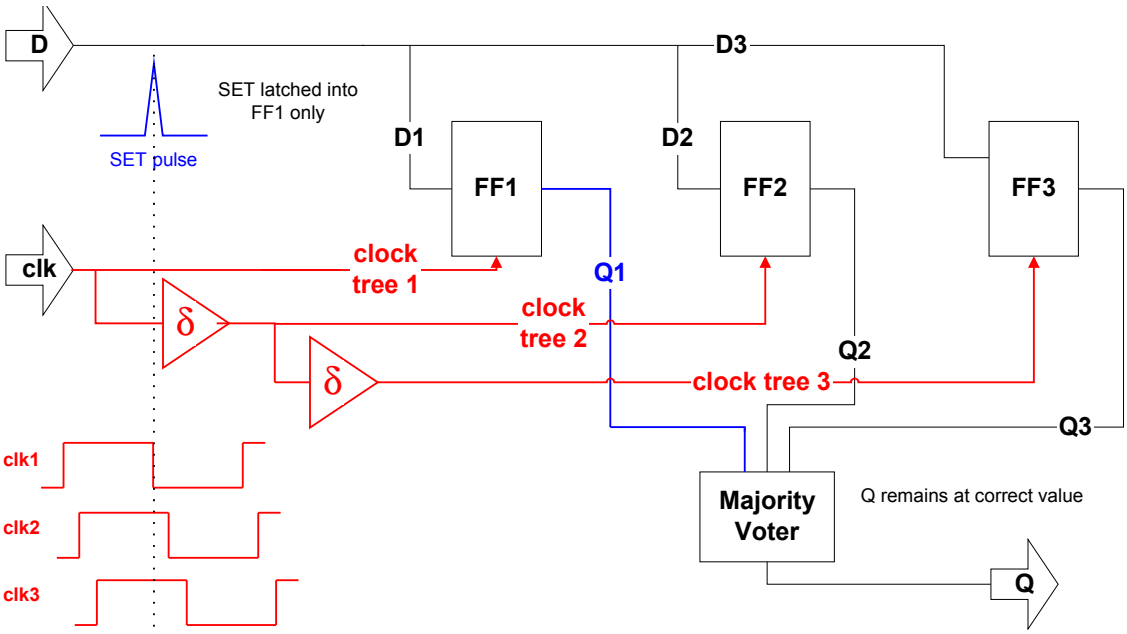
Drawing 1: TMR flip-flops



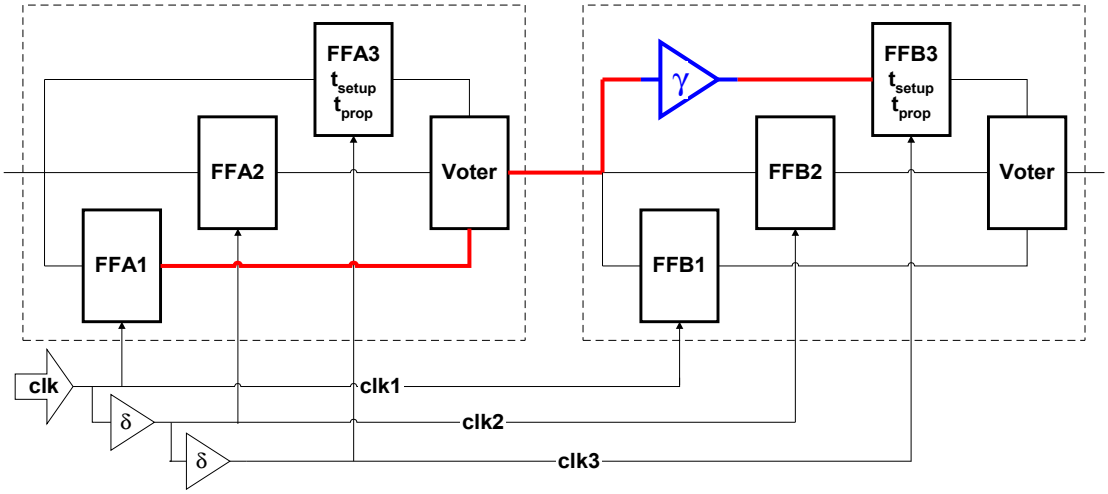
Drawing 2: TMR for combinatorial logic



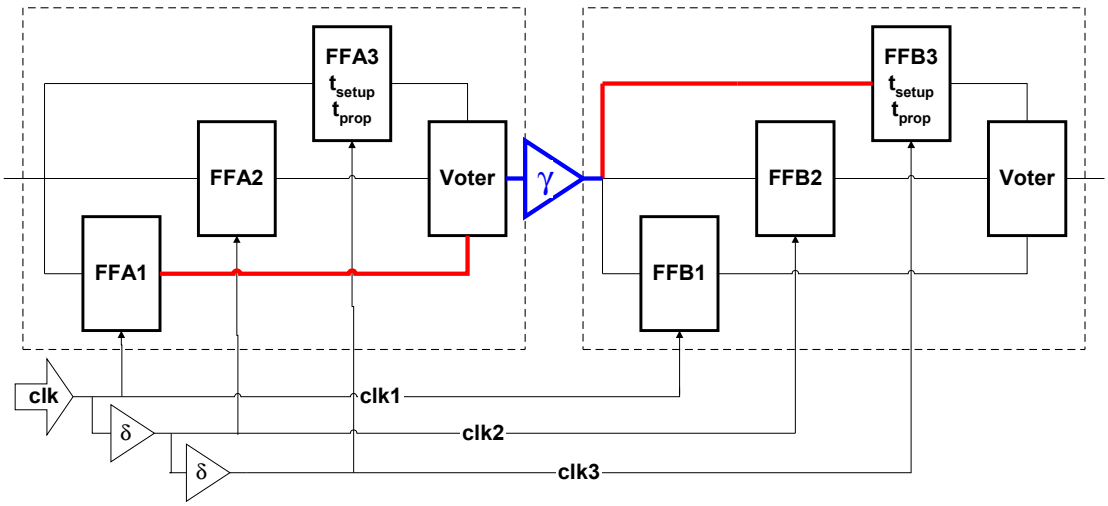
Drawing 3: Normal and DICE latch



Drawing 4: TMR flip-flop with temporal SET filtering



Drawing 5: Wrong hold fix



Drawing 6: Correct hold fix