SEE Analysis and Mitigation for FPGA and Digital ASIC Devices

University of Surrey
07 December 2005

Roland Weigand
European Space Agency
Data Systems Division TEC-EDM
Microelectronics Section
Tel. +31-71-565-3298
Fax. +31-71-565-6791
Roland.Weigand[at]esa.int

Introduction

- ***The Technical and Quality Management Directorate (TEC)***
  - http://www.esa.int/techresources/index.html

  in TEC, mainly 3 sections work on SEE effects:
- ***The Space Environments and Effects Section***
  - Analysis of space environments and their effects on space systems
  - http://space-env.esa.int/index.html
- ***The Radiation Effects and Analysis Techniques Section***
  - Analysis at component level and radiation testing
  - https://escies.org/public/radiation/esa/
- ***The Microelectronics Section***
  - Availability of appropriate technologies and development methods
  - Availability of space-specific standard components and IP
  - Development support to projects
  - Analysis and mitigation of SEE at design level
  - http://www.estec.esa.nl/microelectronics/

SEU Emulation and Simulation Tools

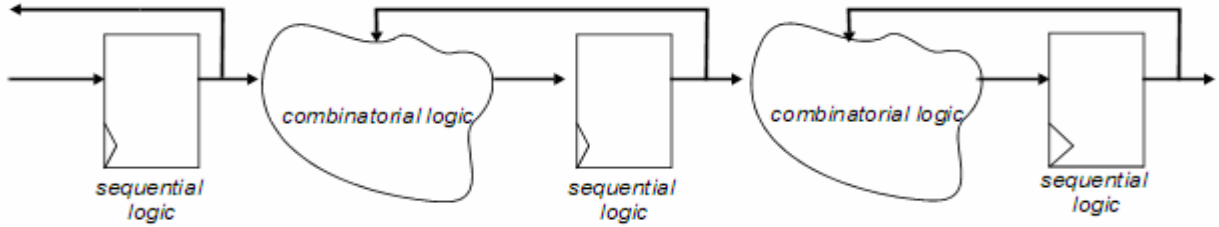
- ***FT-UNSHADES (University of Seville)***
  - SEU Validation of ASIC designs by fault injection into user flip-flops and user memory using an SRAM based FPGA
  - http://www.estec.esa.nl/microelectronics/finalreport/FT-UExcutiveSummary.pdf
- ***FLIPPER (IASF Milan)***
  - SEU Validation of designs targeting Xilinx Virtex II reprogrammable FPGAs (RFPGA) by fault injection into the configuration memory and reconfiguration logic registers
  - http://www.estec.esa.nl/microelectronics/techno/Flipper_ProductSheet.pdf
- ***The SEUs Simulation Tool (SST)***
  - A set of Perl and tcl scripts, which allows injecting SEU like faults into HDL and netlist simulations
    - » http://www.estec.esa.nl/microelectronics/asic/asic.html

SEU in reprogrammable FPGA (RFPGA)

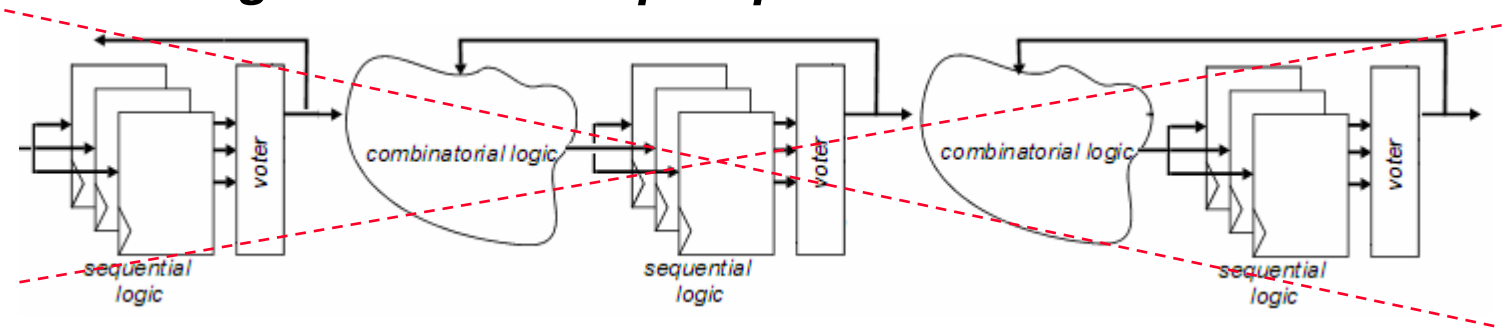
- ***Increasing interest for SRAM based RFPGA***
  - Lower NRE cost than ASIC
  - In-flight reconfiguration capability
  - High performance and complexity allowing System-On-FPGA
- ***SEU in configuration memory***
  - Affect not only user data or state (as in ASIC) …
  - … but alter the functionality of the circuit itself
  - … turn the direction of I/O pins
- ***SEU mitigation for RFPGA***
  - Configuration scrubbing or read-back and partial reconfiguration
  - Triplication of registers and combinatorial logic
  - Voting of logical feedback paths
  - Redundancy for user memory
  - Voting of the outputs
  - Triplication of I/Os

Triple Modular Redundancy for SRAM FPGA

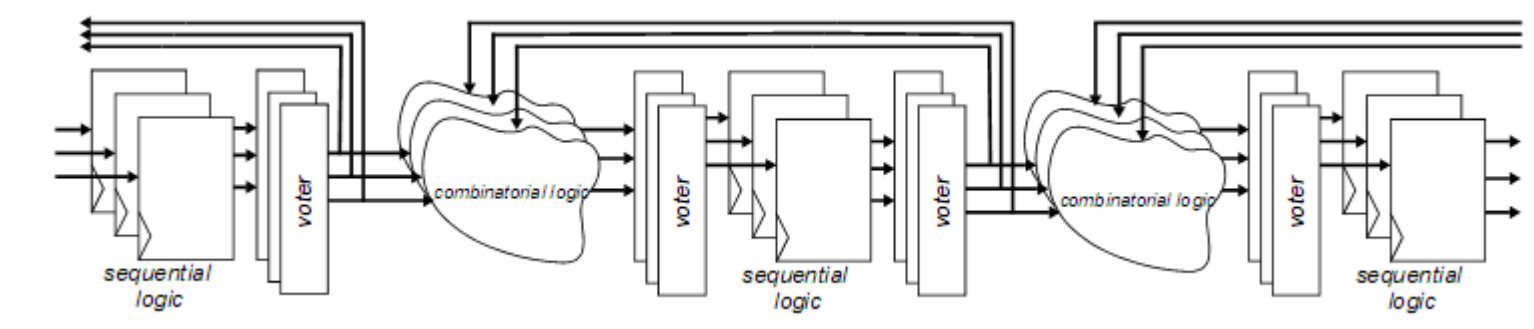
- **Non-hardened sequential and combinatorial logic**



- **TMR and single voters for flip-flops not for SRAM FPGA**



- **TMR for sequential and combinatorial logic and voters**



SEU mitigation in reprogrammable FPGA

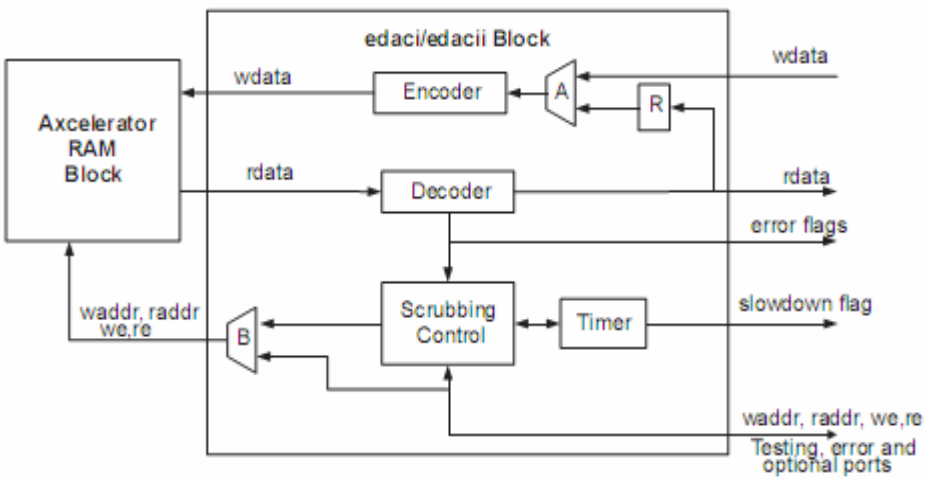
- ***SEE mitigation by design for commercial RFPGA***
  - Functional Triple Modular Redundancy (FTMR) – combinatorial and sequential triplication and voting in implemented in VHDL source code
    - » http://www.estec.esa.nl/microelectronics/techno/reprofpga.html
  - Future projects TBD: evaluate Xilinx XTMR, design a scrubbing controller IP
- ***Xilinx SEE Consortium (USA and Europe/International)***
  - "A voluntary group of organizations that have a mutual interest in the evaluation of reconfigurable FPGAs for Aerospace Applications"
    - » http://www.cad.polito.it/research/consortium.html
    - » http://www.xilinx.com/products/silicon_solutions/market_specific_devices/aero_def/capabilities/see.htm
- ***Development of SEE hardened reprogrammable FPGA***
  - Atmel AT40KEL and the next generation 200K FPGA under CNES contract
    - » http://www.atmel.com/dyn/products/product_card.asp?part_id=2766
  - Xilinx SIRF = SEU Immune Reconfigurable FPGA (RadHard-Virtex)
    - » http://klabs.org/mapld05/presento/176_bogrow_p.ppt

Protection of embedded SRAM blocks (1)

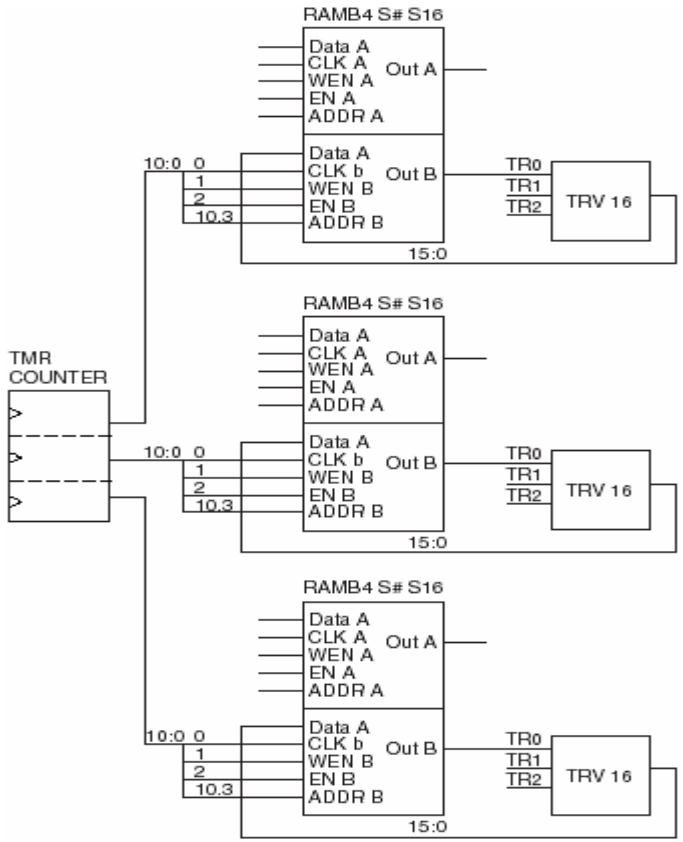
- ***EDAC = Error Detection And Correction***
  - Usually corrects single and detects multiple bit flips per memory word
  - Regular access required to preventing error accumulation (scrubbing)
  - Control state machine required to rewrite corrected data
  - Impact on max. clock frequency (XOR tree)
- ***Parity protection allows detection but no hardware correction***
  - When redundant data is available elsewhere in the system
    - » Embedded cache memories (duplicates of external memory) → LEON2-FT
    - » Duplicated memories (reload correct data from replica) → LEON3-FT
  - On error: reload in by hardware state machine or software (reboot)
- ***Proprietary solutions***
  - ACTEL core generator: http://www.actel.com/documents/EDAC_AN.pdf
    - » EDAC and scrubbing
  - XILINX XTMR: http://klabs.org/mapld05/presento/238_rezgui_p.ppt
    - » Triplication, voting and scrubbing
- ***Area overhead from 1 bit/word (parity) to > triple (Xilinx solution)***

Protection of embedded SRAM blocks (2)

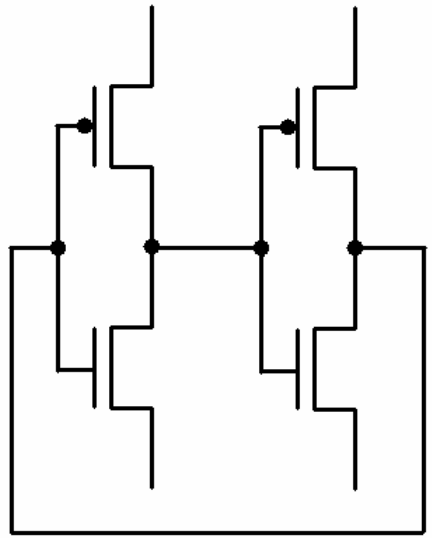
- ***EDAC protected memory (Actel)***
  - Scrubbing takes place only in idle mode (we, re = inactive)
  - Required memory width
    - » 18-bit for data bits <= 12
    - » 36-bit for 12 < data bits <= 29
    - » 54-bit for 20 < data bits <= 47



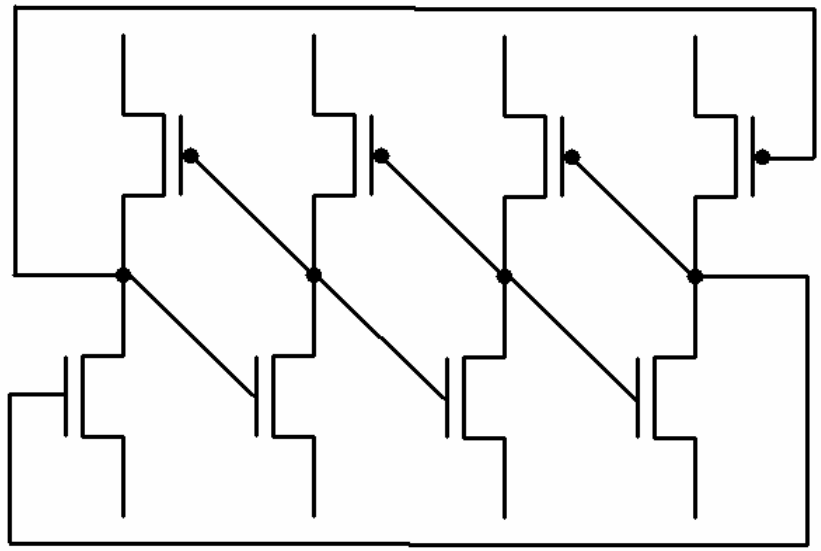
- ***Triplicated memory (Xilinx)***
  - Scrubbing in background using spare port of dual-port memory
  - Triplication against configuration upset



SEU hardening of standard library flip-flops



standard latch

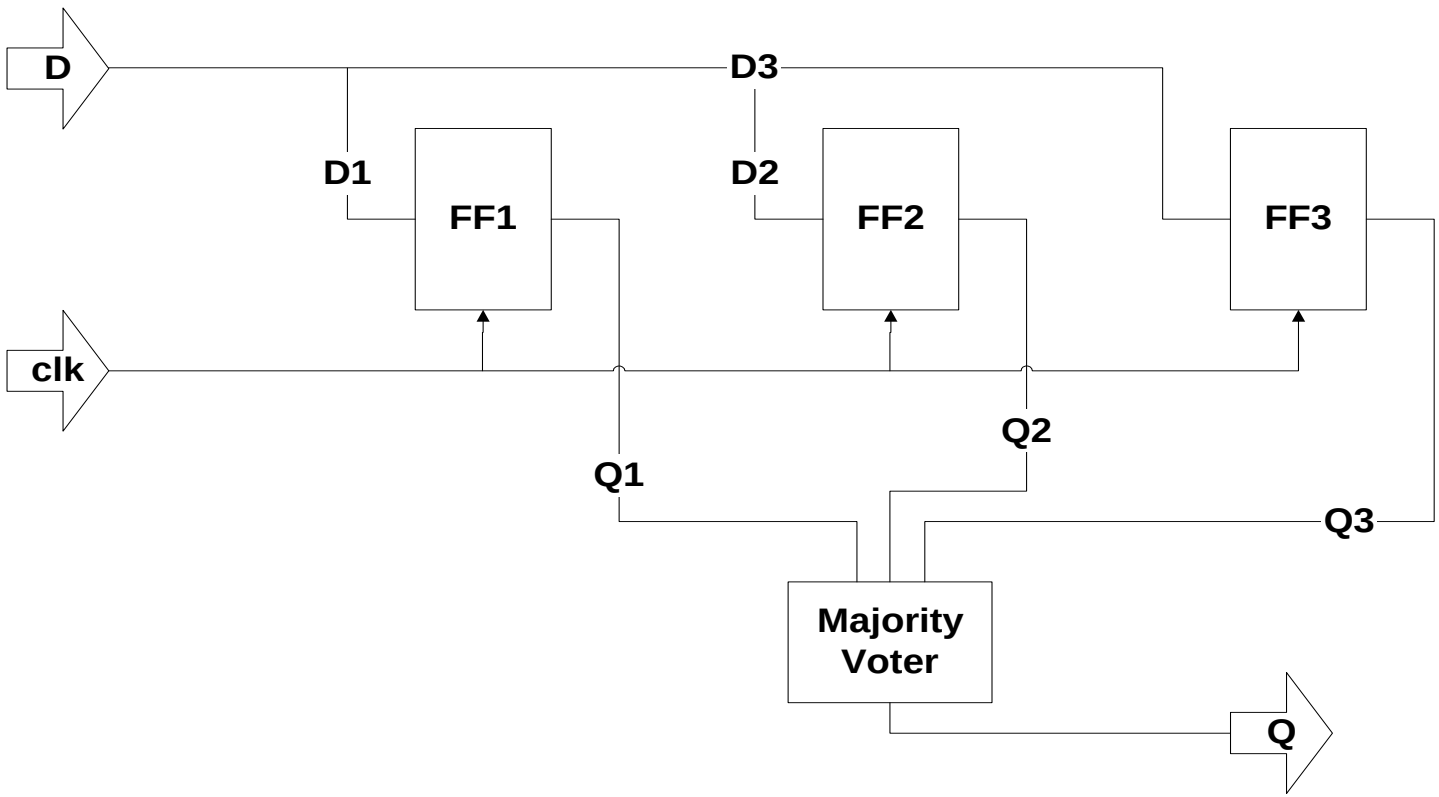


DICE latch

- *HIT = Heavy Ion Tolerant storage cell (RADECS 1993)*
- *DICE = Dual Interlocked storage CEll (1995)*
- *Many hardened libraries exist around the world*
  - **ATMEL in their 0.35 and 0.18 technologies http://www.atmel.com/**
  - **MRC Microelectronics on TSMC (0.35/0.25), UTMC/AMI, HP, NSC, Peregrine http://parts.jpl.nasa.gov/mrqw/mrqw_presentations/S4_alexander.ppt**
  - **HIREC/JAXXA http://www.hirec.co.jp - Fujitsu 0.18, OKI 0.15 SOI (NSREC2005)**
  - **ESA's DARE (Design Against Radiation Effects) library for UMC 0.18**
    - » **http://www.estec.esa.nl/microelectronics/finalreport/DareExecutiveSummary_V3.pdf**

TMR Flip-Flop with voter

- ***Hardening by Triple Modular Redundancy (TMR) flip-flops***
  - Triplication of flip-flops and combinatorial voting
  - Implemented in the RTL source code, by netlist editing or by synthesis tool
  - Overhead > x3 on flip-flops, x2 on typical designs (50% combinatorial logic)

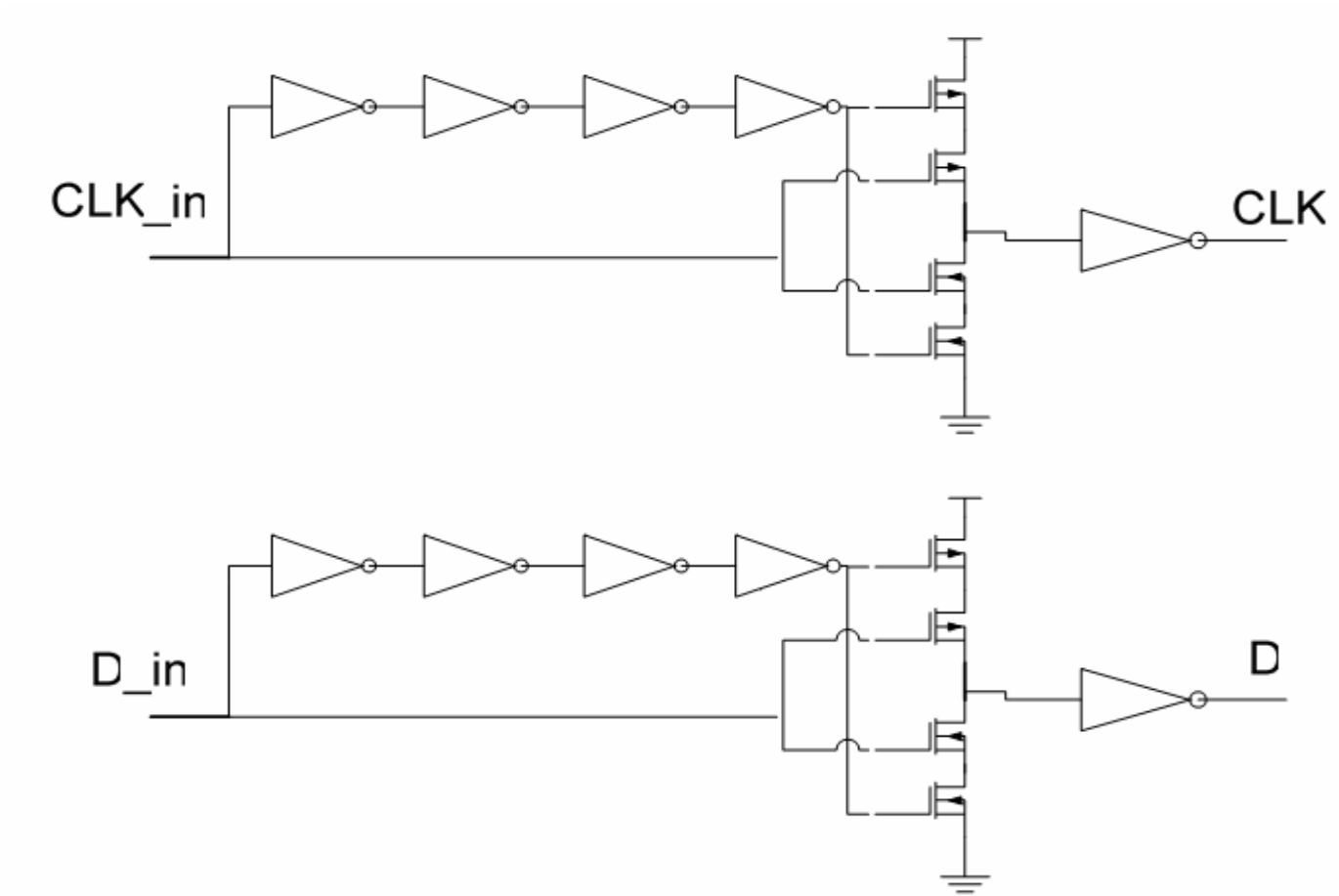


Single Event Transients (SET)

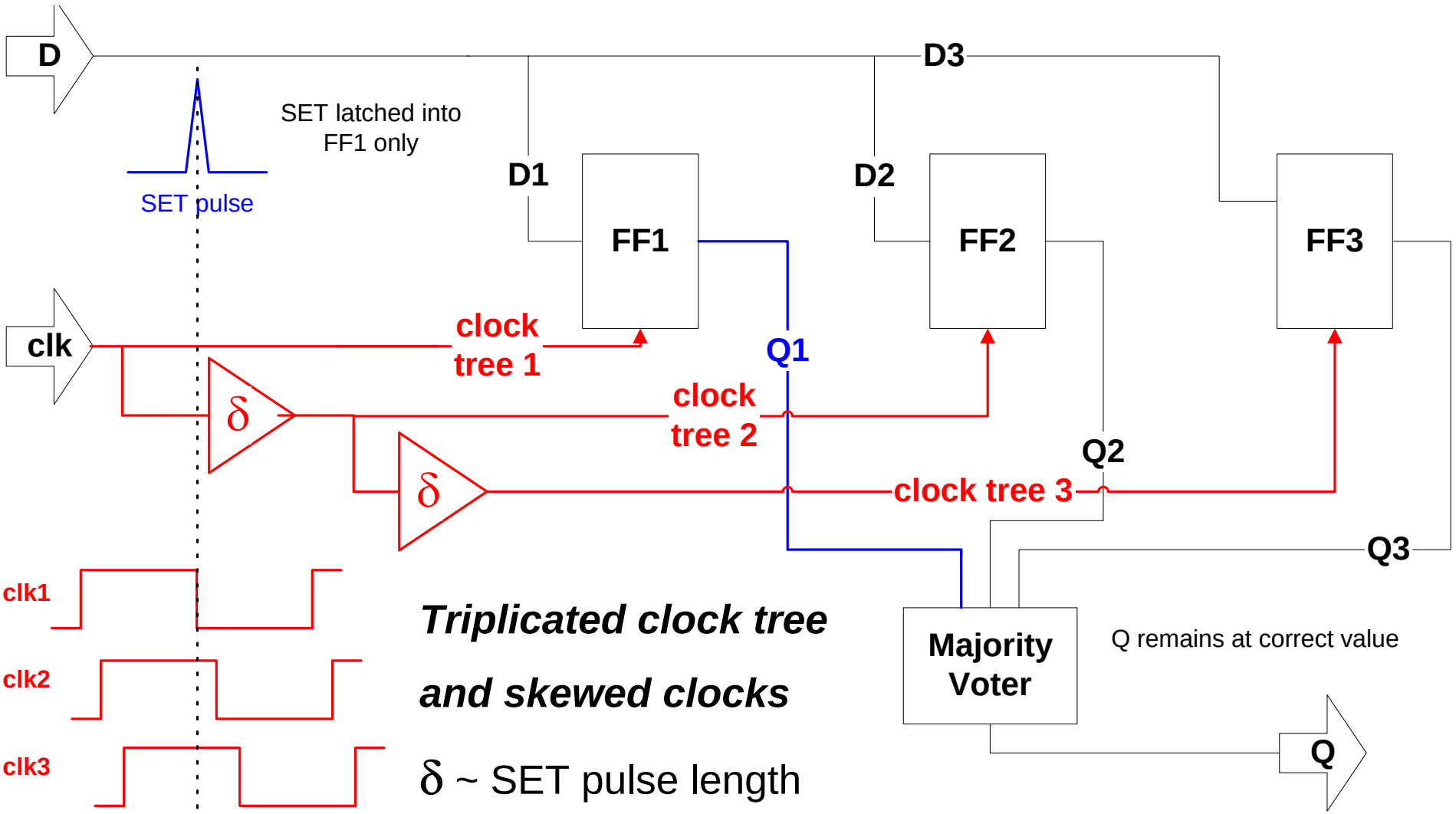
- ***Collision induced carrier generation in PN junctions***
  - Propagate as glitches in combinatorial logic
  - Latched into storage cells when arriving at data input during clock edge
  - ➔ Upset rate increases with the clock frequency
  - Main SEE in ERC32 processor (0.5 μm technology)
  - … definitely a concern in 0.18 μm and below
- ***Analysis of SET effects in simulation and radiation tests***
  - SET pulse length and amplitude are most important parameters
  - Specific test structures to catch and characterise the pulse
  - CNES contract with Atmel on SET effects in the 0.18 μm technology
- ***Mitigation of SET effects***
  - http://www.mrchsv.com/docs/Vanderbilt/Circuit%20and%20layout%20Issues.pdf
  - Propagation of complementary logic levels (“Dual Stream”)
  - Using stronger drivers and higher capacitive loads
  - Delay filtering on all flip-flop inputs (clock, data, reset)
  - Temporal Vote: Triple skewed clocks in conjunction with the TMR flip-flop
    - » Triplication of clock-like nets (including asynchronous resets)

Delay Filtering on flip-flop inputs

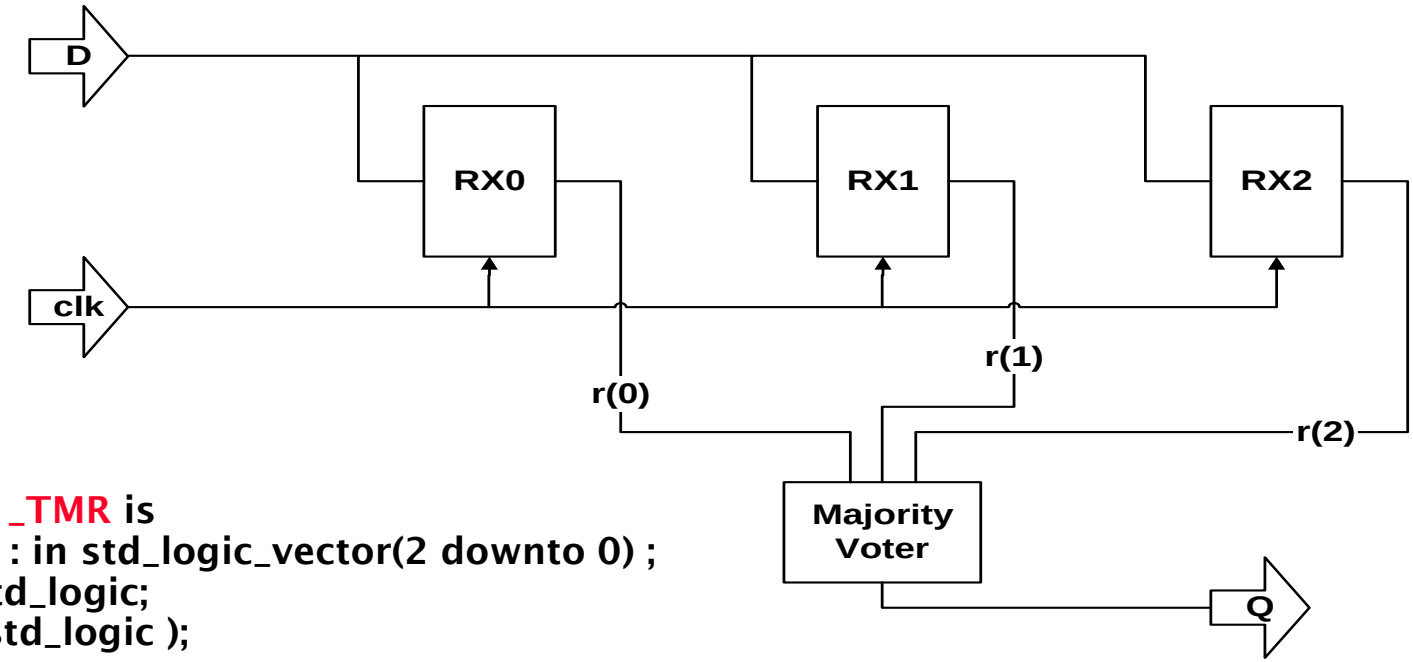
- ***Mongkolkachit, Pitsini; Bhuva, Bharat, RADECS 2003***
  - Area and timing overhead depends on SET pulse width (length of delay chain)



SET-tolerance by skewed clocks

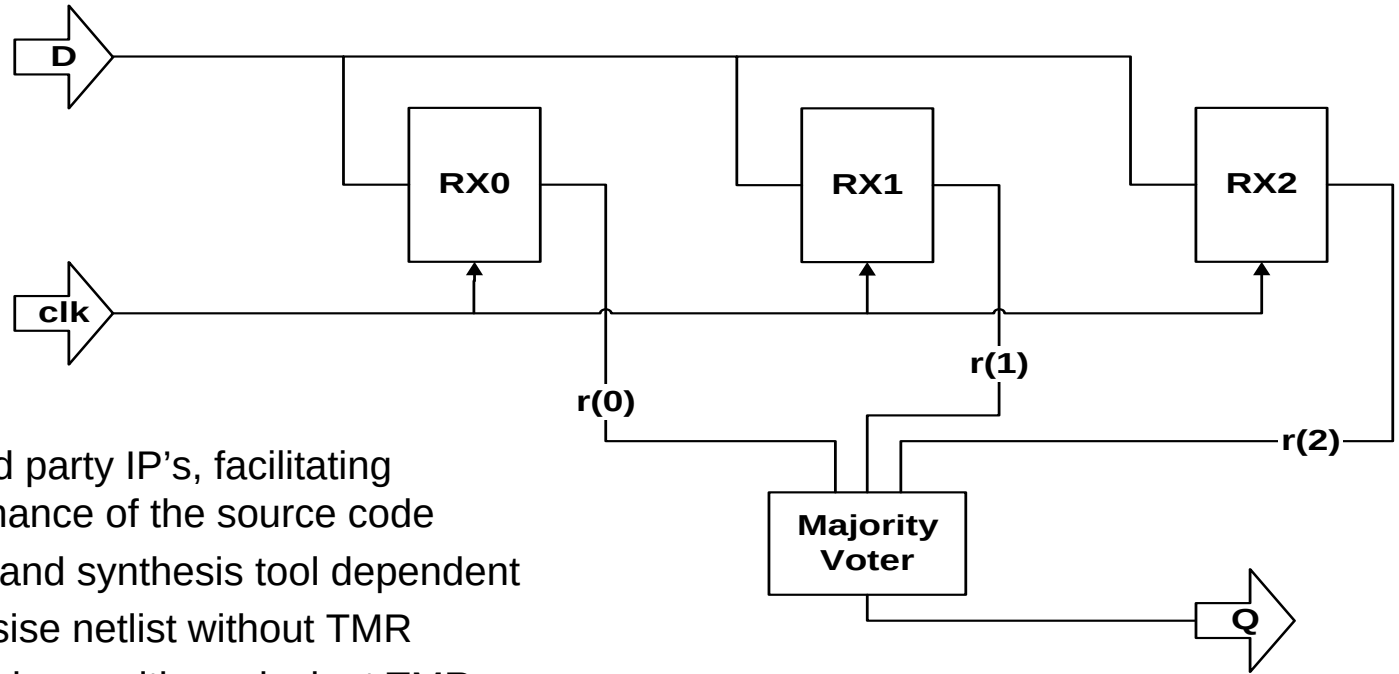


TMR insertion in VHDL source code




```
entity DFF1_TMR is
  port ( clk  : in std_logic_vector(2 downto 0) ;
    d  : in  std_logic;
    q  : out std_logic );
end;

-- One process per TMR Flip-flop
rx0 : process(clk) begin if rising_edge(clk(0)) then r(0) <= d; end if; end process;
rx1 : process(clk) begin if rising_edge(clk(1)) then r(1) <= d; end if; end process;
rx2 : process(clk) begin if rising_edge(clk(2)) then r(2) <= d; end if; end process;

-- Voting outputs
q <= (r(0) and r(1)) or (r(0) and r(2)) or (r(1) and r(2));
```

TMR insertion at gate level



- For third party IP's, facilitating maintenance of the source code
- Library and synthesis tool dependent
- Synthesise netlist without TMR
- Use package with equivalent TMR cells for all flip-flops used in the netlist
- Edit netlist to triplicate clocks (including any clock buffers/inverters), instantiate **TMR cells** instead of **standard flip-flops**
- Carefully inspect edited netlist
- Resynthesise the edited netlist

```
sed
 -e 's/CLK\(.*\) std_logic/CLK\1
     std_logic_vector(2 downto 0) /'
-e 's/bufx\(.*\)INVDL/bufx\1INVDL_TMR/'
-e 's/DFF1 port map/DFF1_TMR port map/'
-e 's/DFF2 port map/DFF2_TMR port map/'
netlist_notmr.vhd > netlist_tmr.vhd
```

Synthesis constraints for hardened flip-flops

- ***Use only hard flip-flops***
  - dont_use on all non-hard cells
  - make sure the dont_use is applied consistently at all optimisation steps
- ***Partly insert hard cells***
  - dont_use on all non-hard cells
  - synthesise design (all hard FF)
  - dont_touch on all cells to be hard
  - remove attribute dont_use
  - synthesise again
  - keep dont_touch throughout all subsequent optimisation steps
- ***Verify implementation***
  - hardened cells are larger and slower → can be easily removed by backend
  - grep in **final netlist**
  - extract all flip-flop instance- and cell names

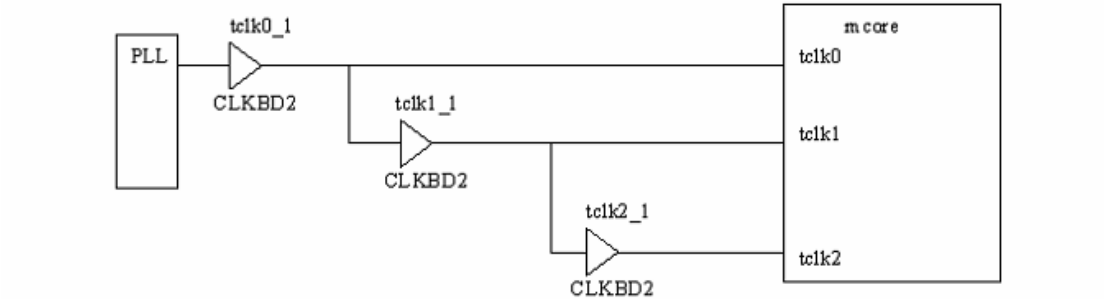
```
/* Group cells to be hardened in a separate sub-module */
seu_hard_cell_list = { InitState_reg*, InitN_reg*, InitReset_reg* };
group seu_hard_cell_list -design_name seu_hard_cells -cell_name
      seu_hard

/* Force use of SEU-hard cells instead of non-SEU-hard cells */
remove_attribute MGRT + "/HD*" dont_use
remove_attribute MGRT + "/HL*" dont_use
set_dont_use MGRT + "/DF*"
set_dont_use MGRT + "/LAT*"

/* map only the hardened flip-flops by unconstrained compile */
current_design seu_hard_cells
compile

/* remove the hierarchy around hardened flip-flops, freeze flip-flops */
current_design active_design
ungroup seu_hard
foreach(cell_name, seu_hard_cell_list)
      { set_dont_touch "seu_hard/" + cell_name; }

/* Force use of non-SEU-hard cells instead of SEU-hard cells */
remove_attribute MGRT + "/DF*" dont_use
remove_attribute MGRT + "/LAT*" dont_use
set_dont_use MGRT + "/HD*"
set_dont_use MGRT + "/HL*"

/* ... any further synthesis iterations use non-hardened flip-flops */
/*    while keeping untouched the hardened flip-flops ... */
```

Inserting triple skewed clock trees (1)

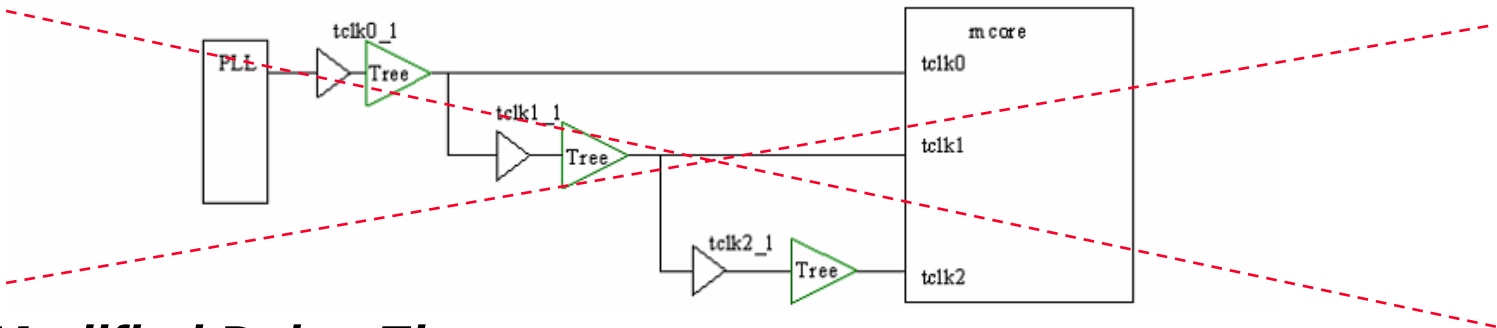
- ***Clock Tree Synthesis (CTS) is part of backend (layout) design***
  - Triple CTS critical when many clock domains exist
  - CTS guarantees only max clock skew inside a clock tree
  - Different latency (delay from source to flip-flop) in different clock trees
- ***Delay insertion at the origin of the clock trees***
  - Delays are not synthesisable
  - Instantiate delay buffers in the VHDL source code
    → speculative, needs control and adjustment in the backend process
  - ➔ In the backend process along with the CTS
- ***Delay insertion in the flip-flops***
  - Area consuming
  - Only one clock tree (easy design flow)
  - No SET mitigation in the clock tree
- ***Triplication of reset trees***
  - Asynchronous resets are clock-like signals

Inserting triple skewed clock trees (1)

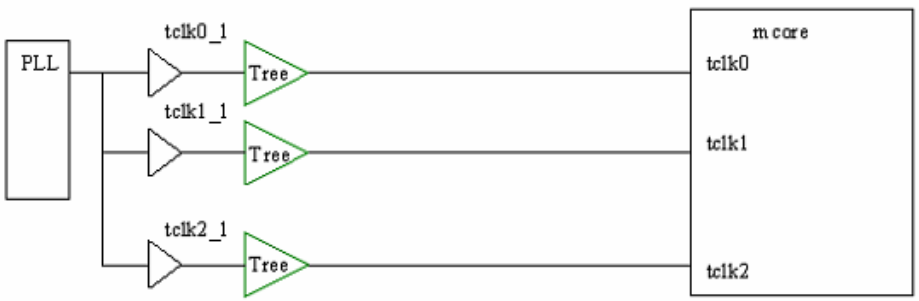
- **Delay elements in VHDL code**



- **First Clock-Tree-Synthesis (CTS)**



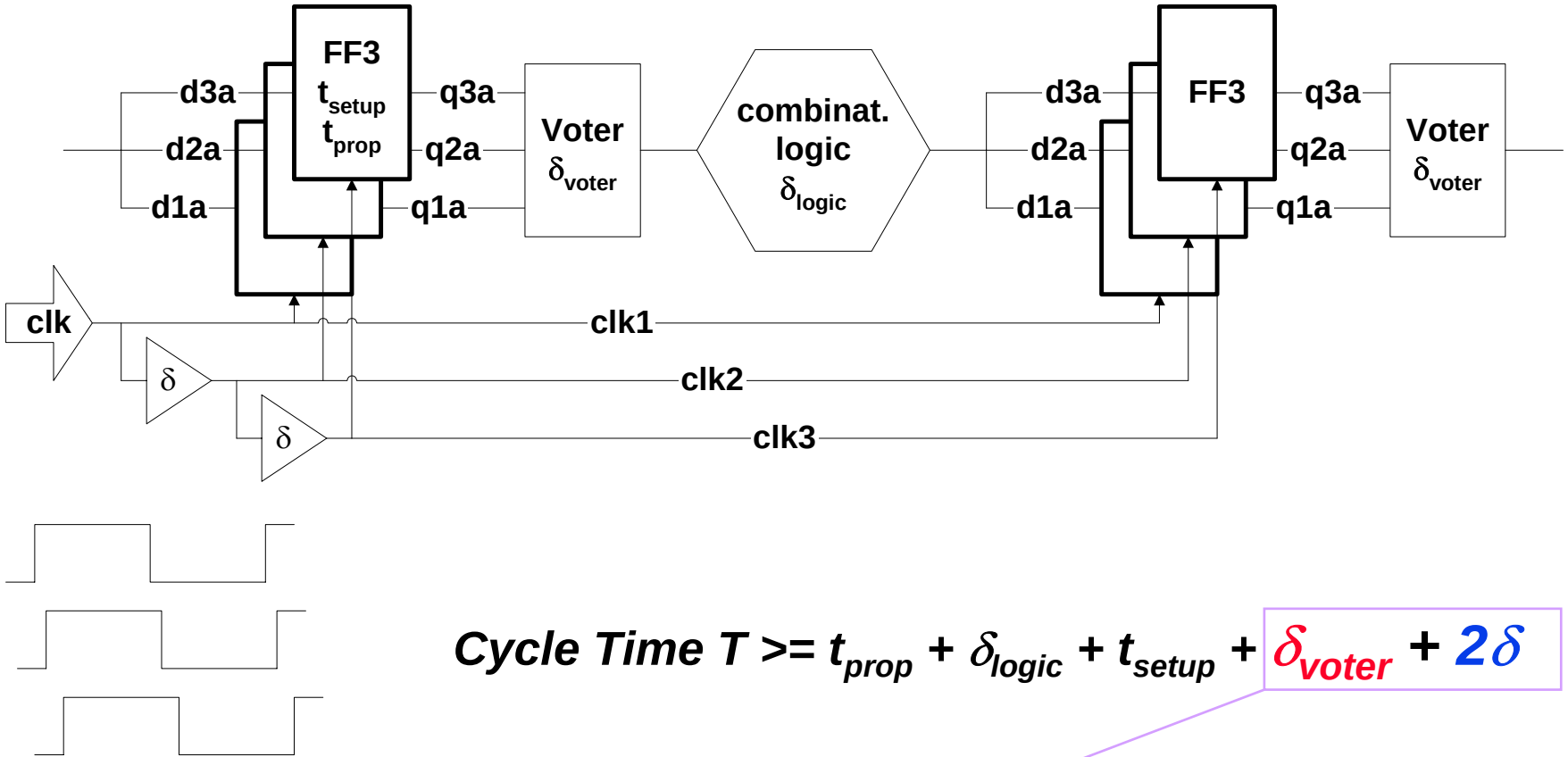
- **Modified Delay Elements**



SEU mitigation in the ASIC design flow

- ***Increased complexity affects the design flow and –results***
  - Large netlist with higher cell and node count
  - Increased run-time or even crashes of EDA tools
  - Design optimisation is less efficient
- ***Synthesis tools are designed to remove redundancy***
  - Normally, registers are not modified – but be careful …
  - … with sequential optimisation (pipelining, retiming etc.)
- ***Timing issues***
  - TMR voting and clock skewing reduces maximum speed
  - Increased area leads to higher interconnect delay
  - Clock skewing can be removed by hold-time fix
- ***Verification and test issues***
  - TMR and formal verification (1 FF in RTL → 3 FF at gate level)
  - TMR (= redundancy) affects testability in scan testing
  - Implementation of protection has to be verified at netlist level

TMR Timing Issues



$$\textit{Cycle Time } T >= t_{prop} + \delta_{logic} + t_{setup} + \delta_{voter} + 2\delta$$

TMR voters and clock skewing reduce operating frequency

Area and power overheads of hardened FF

- ***Voted TMR cells***
  - Area overhead >~ factor 3
  - Power consumption ~ factor 3
- ***SEU hardened flip-flops***
  - Area overhead factor 2 – 2.5
  - Power consumption factor 2 – 3
- ***Overhead only on flip-flops***
  - Total overhead depends on share of combinatorial and sequential logic
  - A = 3x flip-flops + 1x combinatorial

Share of flip-flops	*Area overhead*
25%	***1.5***
50%	***2***
75%	***2.5***

Synthesis description of the DARE library

State toggle power increases ~ x3

Standard DFF rise_power(li5X5) {

```
index_1("0.016, 0.064, 0.128, 0.8, 1.07") ;
index_2("0.03, 0.15, 0.75, 1.5, 3") ;
values("0.260154 0.260608 0.259797 0.262227 0.265544",\
    "0.258697 0.259304 0.258465 0.262485 0.264274",\
    "0.258899 0.259535 0.258754 0.26171 0.264257",\
    "0.259817 0.260501 0.259856 0.262175 0.265157",\
    "0.259849 0.260833 0.260201 0.262509 0.265653"); }
```

Hardened XDFF rise_power(li5X5) {

```
index_1("0.016, 0.064, 0.128, 0.8, 1.07") ;
index_2("0.03, 0.15, 0.75, 1.5, 3") ;
values("0.800729 0.800399 0.794199 0.79509 0.799814",\
    "0.789216 0.788791 0.7821 0.78533 0.78516",\
    "0.781962 0.781545 0.774982 0.777166 0.776802",\
    "0.770274 0.769896 0.763804 0.765198 0.769422",\
    "0.765816 0.76547 0.759478 0.760922 0.765386"); }
```

Clock power increases ~ x2

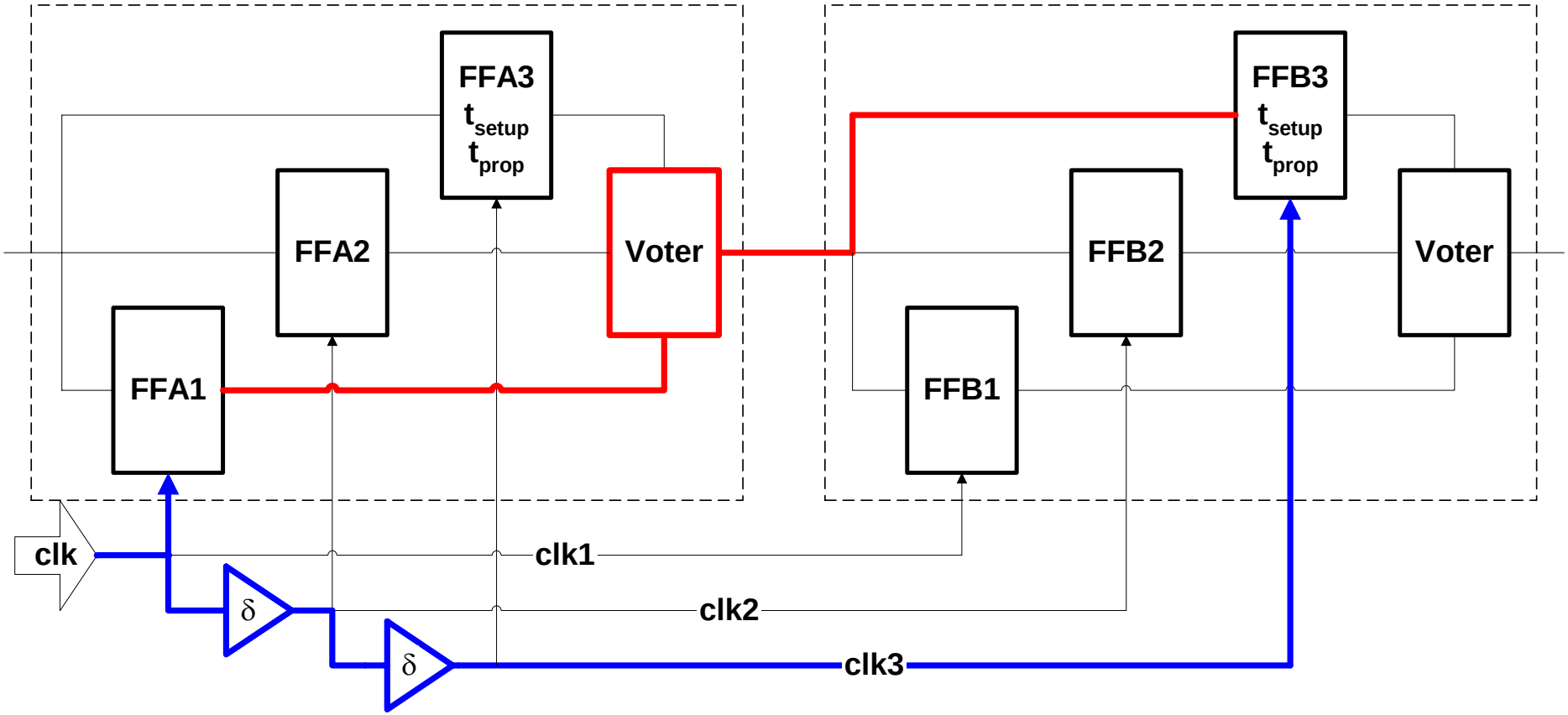
Standard DFF rise_power(i5) {

```
index_1("0.03, 0.15, 0.75, 1.5, 3") ;
values("0.09928 0.098241 0.111142 0.131959 0.180269"); }
```

Hardened XDFF rise_power(i5) {

```
index_1("0.03, 0.15, 0.75, 1.5, 3") ;
values("0.208006 0.207359 0.227548 0.26199 0.344905"); }
```

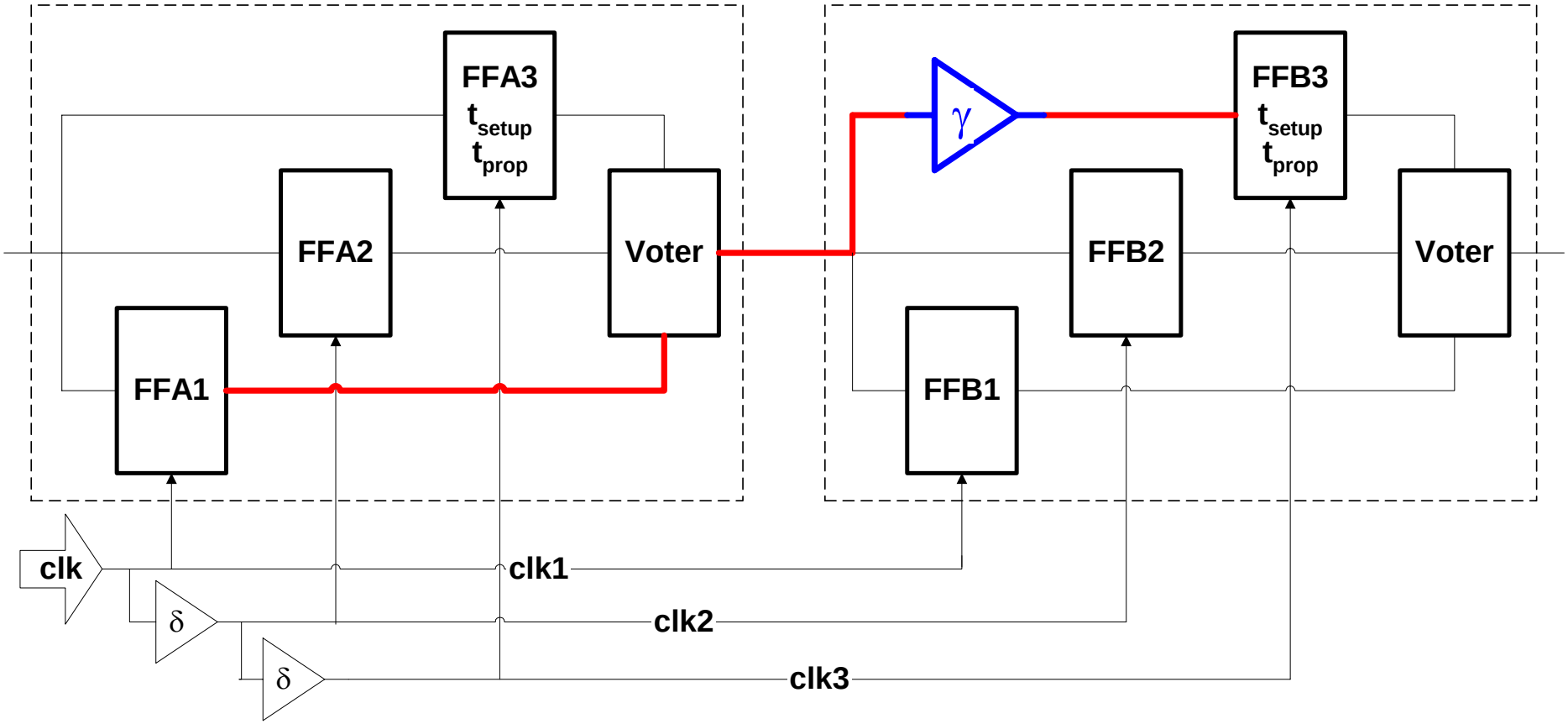
Hold violations with skewed clocks



When propagation delays (t_{prop}, voter) < (2 δ) clock skew

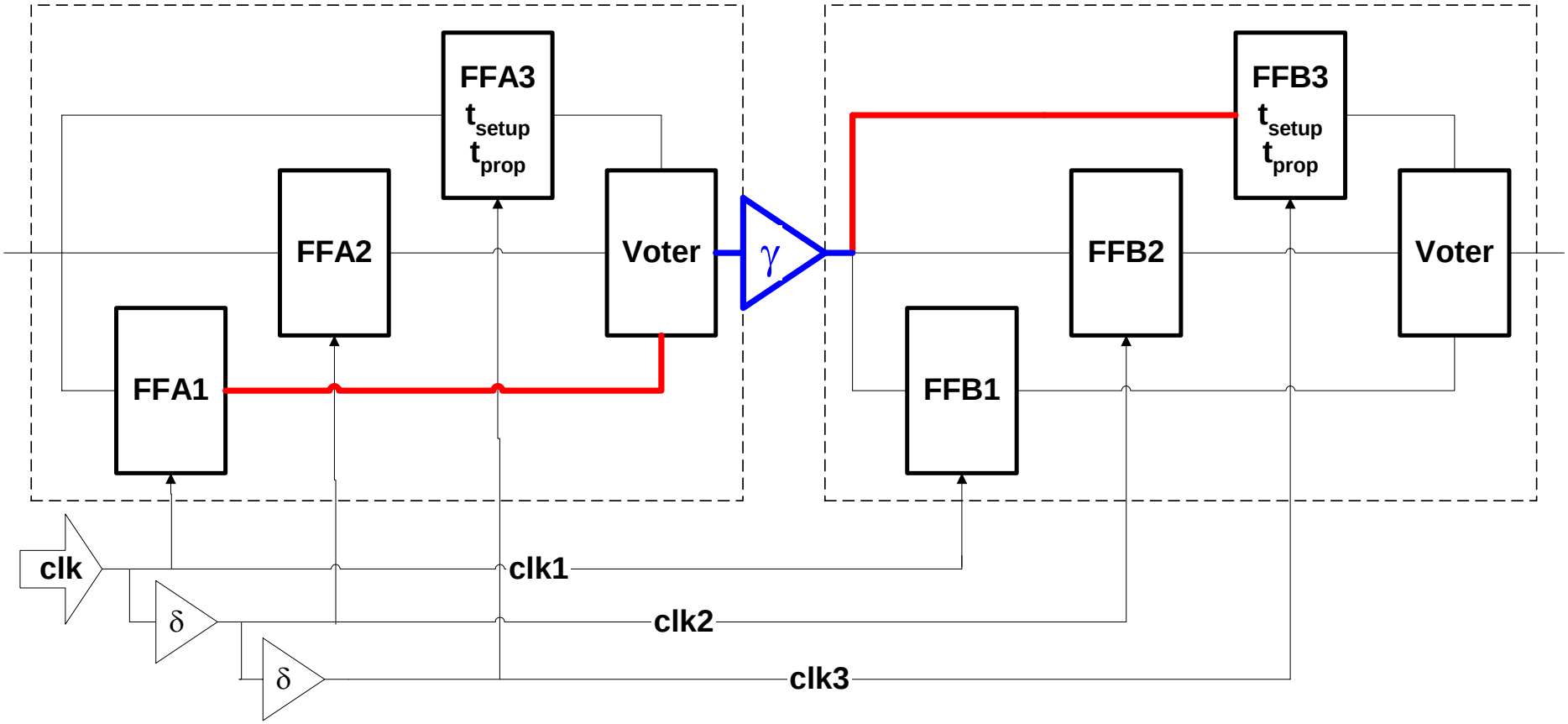
➔ hold violation FFA1 → FFB3

Wrong hold fix by EDA tool



Automatic buffer insertion by fix-hold of synthesis tool compensates clock skew → and spoils SET protection

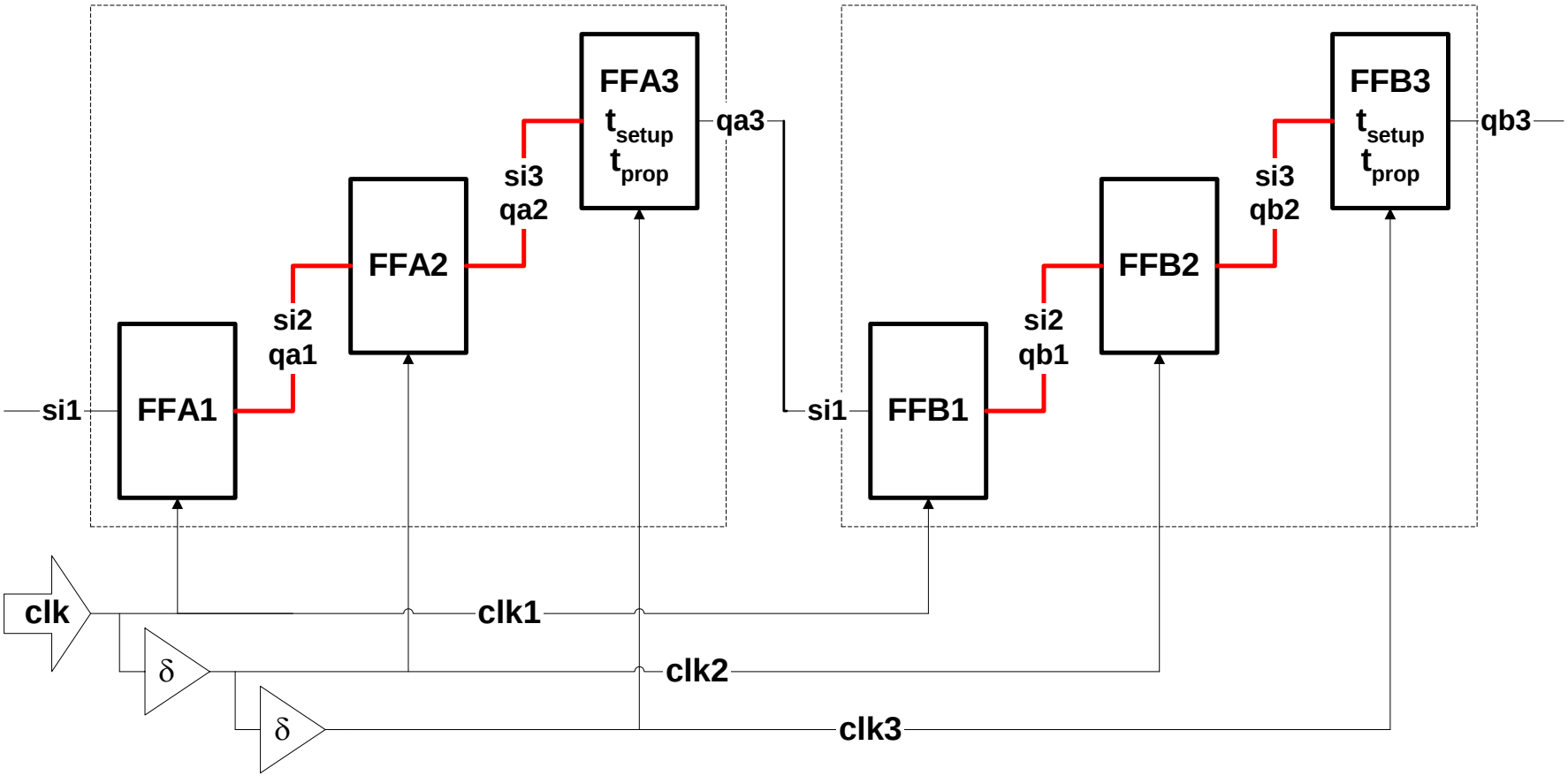
Correct hold fix



Inserting delay buffer between two entire TMR flip-flops…

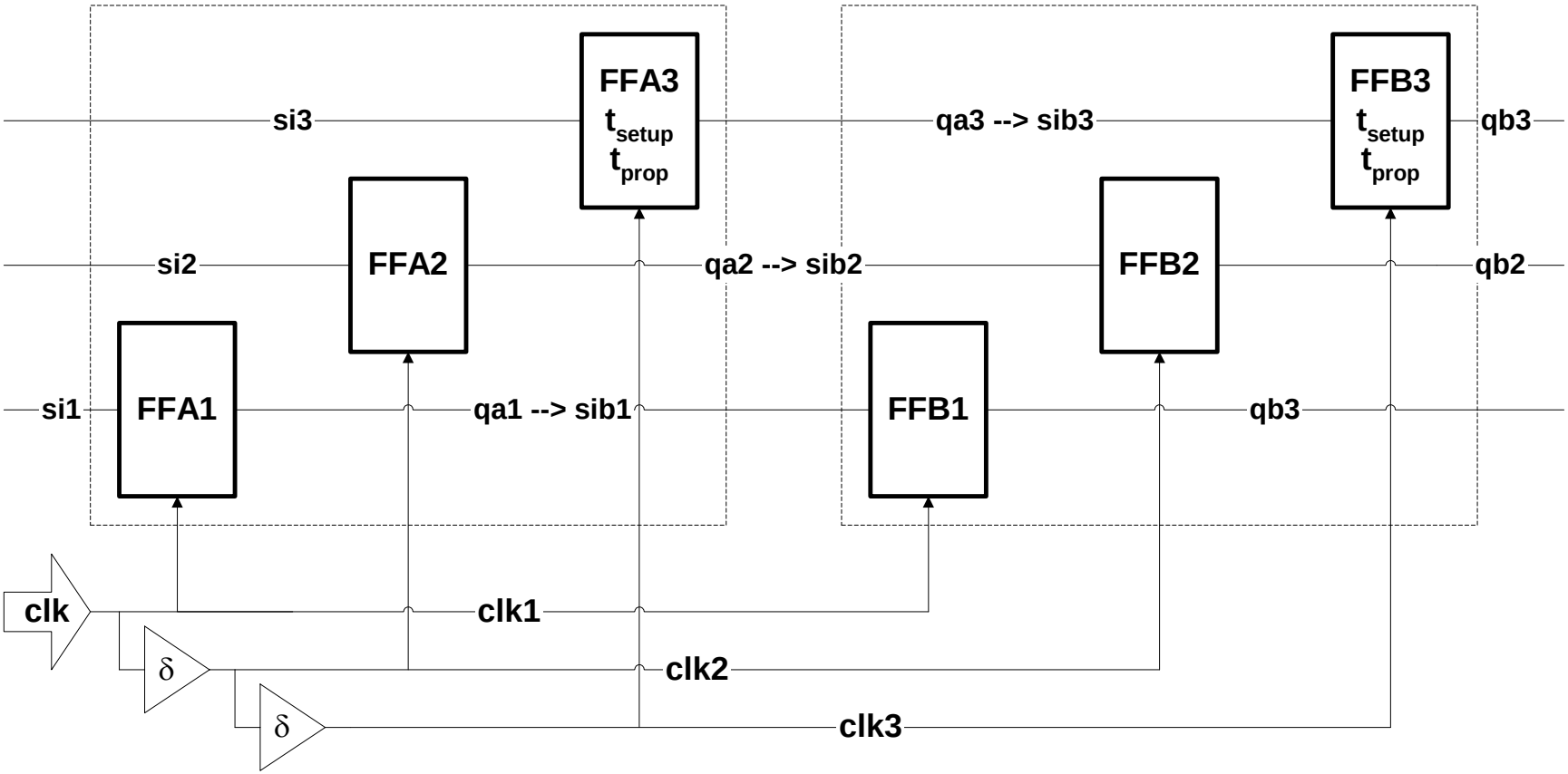
➔ SET protection through clock skew conserved

Scan Path Insertion (wrong)



Scan path routing across sub-clock domains ➔ hold violations

Scan Path Insertion (right)



Better: one scan path per sub-clock domain

may also simplify pattern generation

Future perspectives

- ***SEU protected flip-flops available for many technologies***
  - … but SET protection is currently in experimental stadium
- ***SEU and SET protected flip-flop as library cells***
  - DF-DICE http://www.isi.edu/~draper/papers/mwscas05_bhatti.pdf
- ***If not available - workaround: build SET flip-flop as macrocell***
  - Compose TMR with triple clock input out of standard library cells
  - Generate appropriate front-end synthesis library for the TMR cell
  - Replace TMR macrocells by standard cell triplet in the gate-level netlist
  - Place and Route with standard foundry design flow
- ***Advantages***
  - Can be implemented with a standard vendor library
  - No need to modify design at source code level
  - Avoids many problems with design flow and tools
- ***Issues***
  - Constraints on backend flow (freeze the SET-cell for timing and hold-fix)
  - Triple skewed clock and triple reset trees

Conclusion

- ***SEU and SET protection possible with commercial ASIC technology***
- ***Refresh/scrubbing against accumulation of (uncorrectable) upsets***
- ***Pitfalls in the design flow with commercial EDA tools***
  - Requires workarounds, scripting and proper constraining
- ***Hardened flip-flops easier to use than building TMR in source code***
  - Hardened library cells, Macrocells composed of commercial library cells
- ***But there will always be a price to pay (speed, area, power…)***
- ***Is full SEU protection always necessary?***
  - Determine upset rate of a given design (sub-function) in a given orbit
  - Determine the impact of an upset at system level
  - Apply selective use of SEU protection
- ***Contact us… http://www.estec.esa.int/microelectronics***

Questions?