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

Design Automation Conference
DAC 2009

Roland Weigand
European Space Agency
Roland.Weigand[at]esa.int

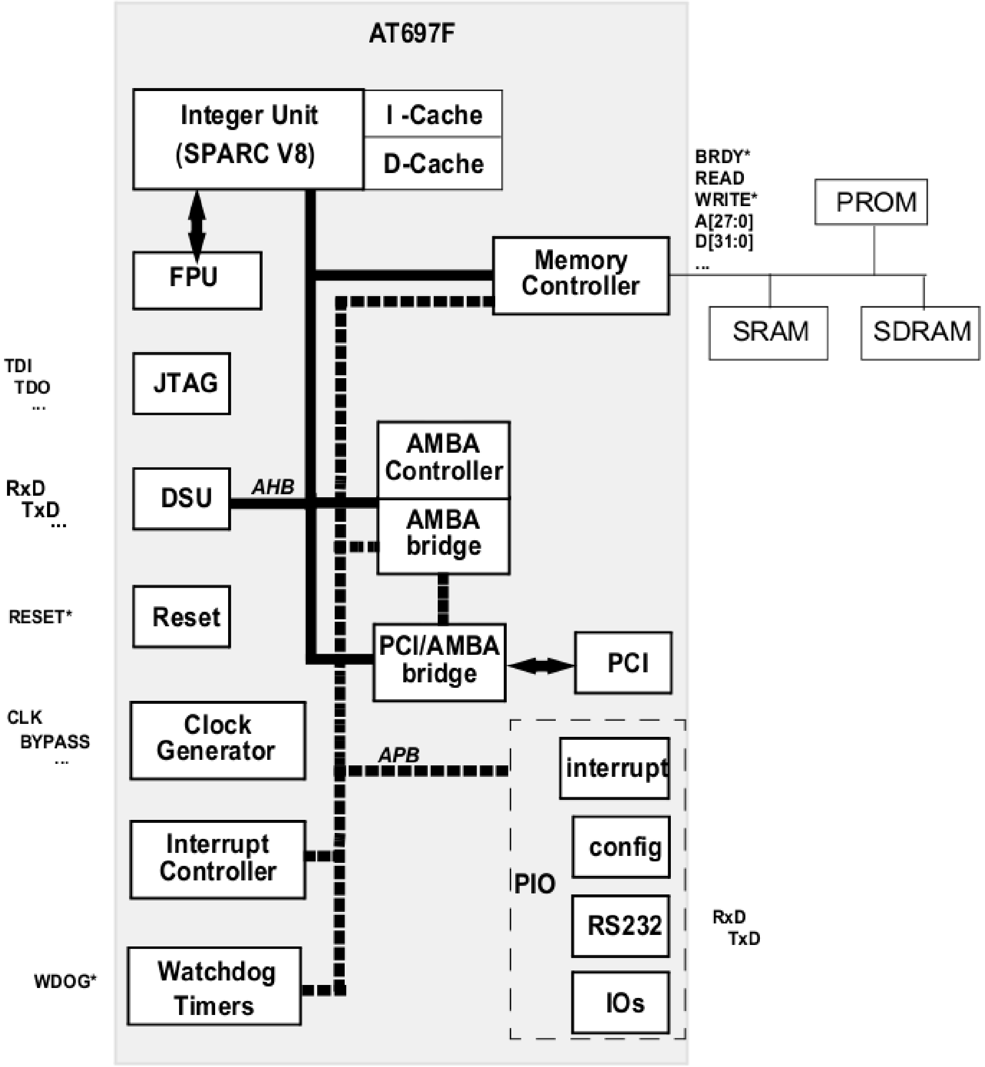
Jean Edelin
Atmel Aerospace
Jean.Edelin[at]atmel.com

Contents

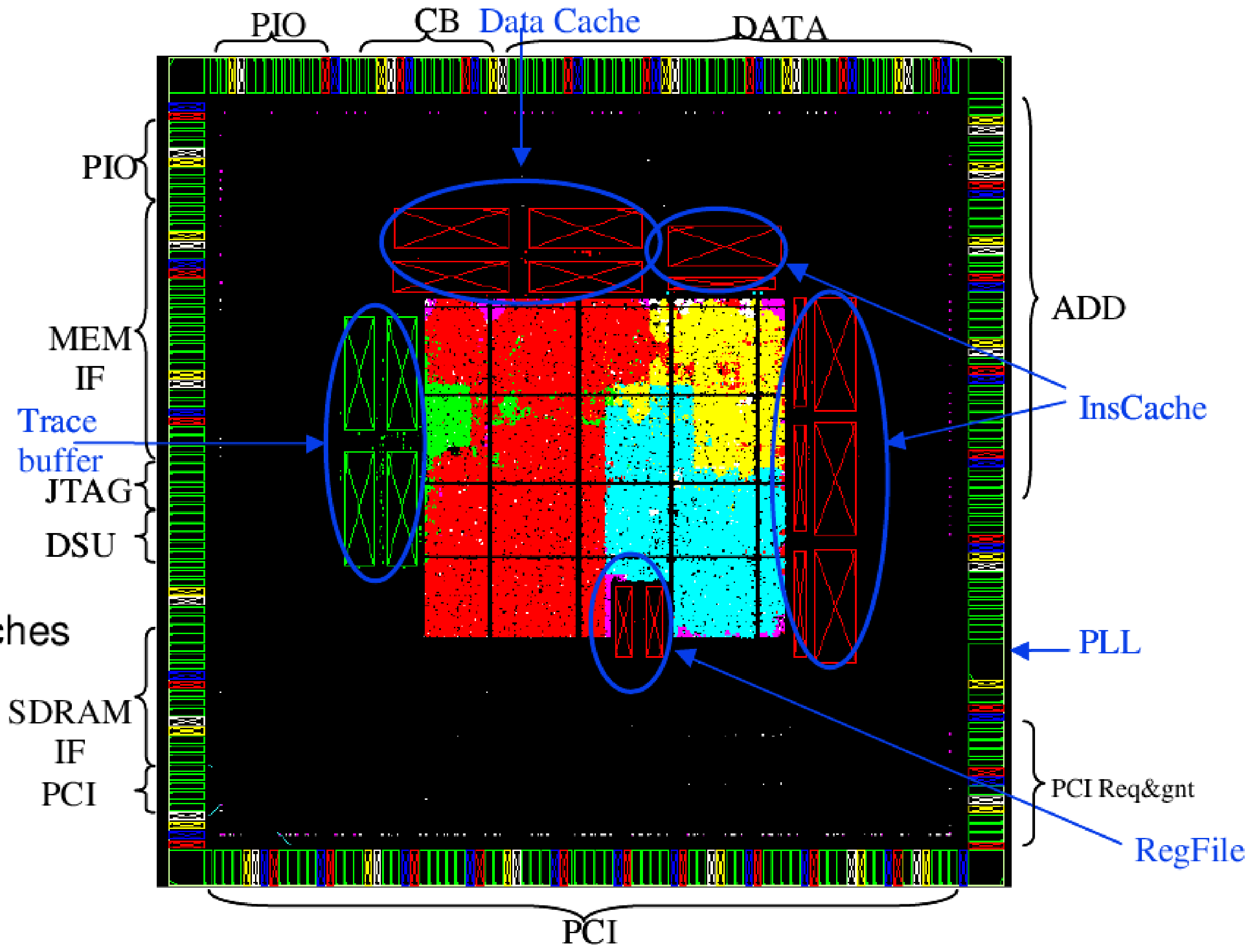
- *The AT697 SPARC V8 microprocessor*
- *Radiation effects in space components*
  - → Total Ionising Dose (TID) and Single Event Effects (SEE)
- *Mitigation of Single Event Effects*
  - → Hardened flip-flops, triple modular redundancy (TMR), glitch filtering
  - → RAM protection by parity and EDAC
- *STMR: 3 voted flip-flops with 3 phase-skewed clock trees*
- *Impact on design flow*
  - → Implementation of STMR in HDL or in netlist
  - → Clock tree synthesis (CTS)
  - → Verification
  - → Timing issues
  - → Scan path
  - → EDA tool issues
- *Overheads for STMR fault tolerant design*

The AT697 Microprocessor [1]

- *SPARC V8 Architecture*
  - → LEON2 IP core [2]
  - → IEEE 754 FPU
  - → Max. 100 MHz
- *PCI 2.2 32-bit 33 MHz*
- *SRAM/SDRAM interface*
- *Radiation tolerance*
  - → Parity/EDAC on internal and external memories
  - → Up to 300 kRad total dose
  - → SEU <= 10^{-5} error/device/day
  - → Latch-up free (70 MeV*cm²/mg)
- *Power consumption <= 1W*
- *Atmel 180 nm technology [1]*
  - → Packages: MCGA 349, QFP 256



Floorplan of the AT697



🟥 IU-FPU-caches

🟨 AMBA

🟦 PCI

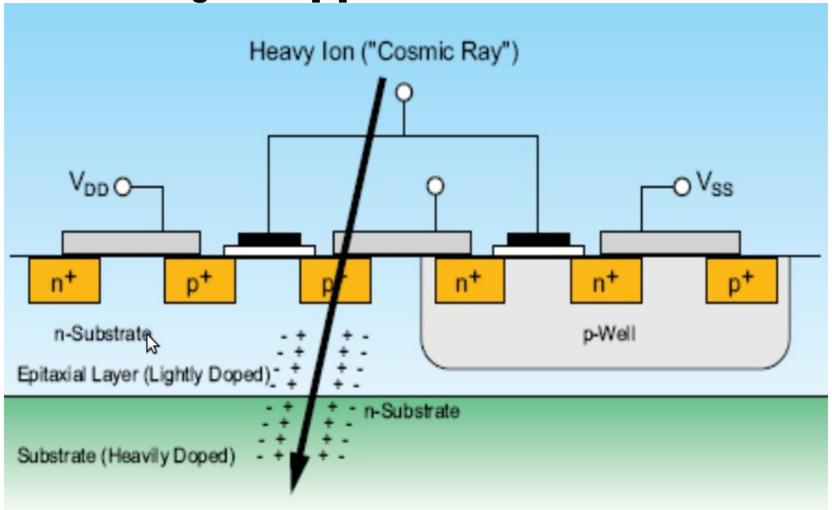
🟩 DSU

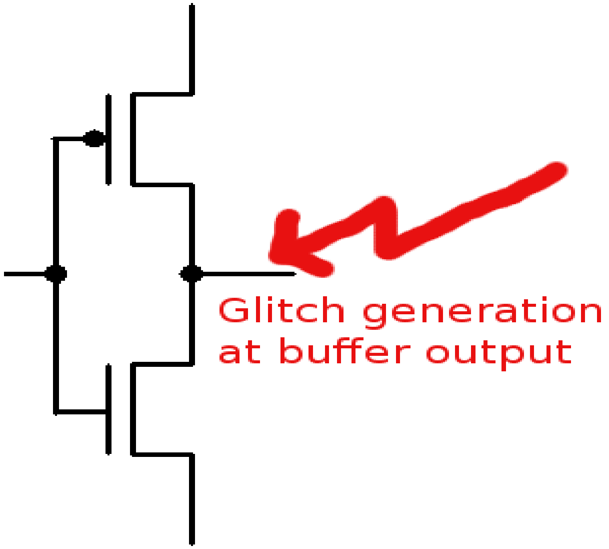
🟪 JTAG

Radiation effects in space components

- *Total Ionising Dose (TID)*
  - → Defects in the semiconductor lattice, degradation of mobility and V_{th}
  - → Reduced speed, increased leakage current at end-of-life
  - → Mitigation: process, cell layout (guardrings), design margins (derating)
- *Single Event Effects (SEE)*
  - → Electron-hole pair generation by interaction with heavy ions
  - → Glitches when carriers are caught by drain pn-junctions

Drawing from: [3]



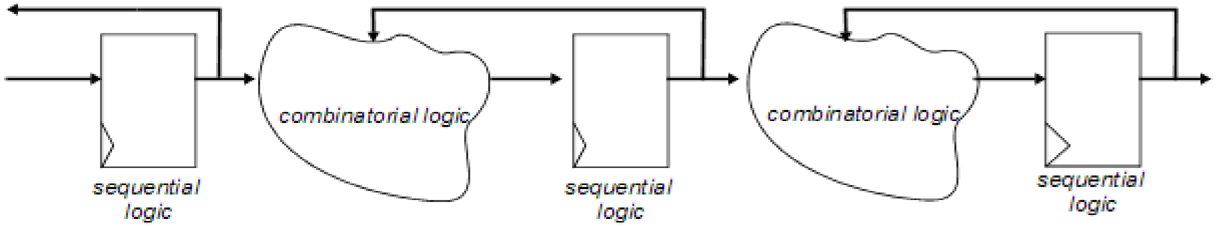


Single Event Effects

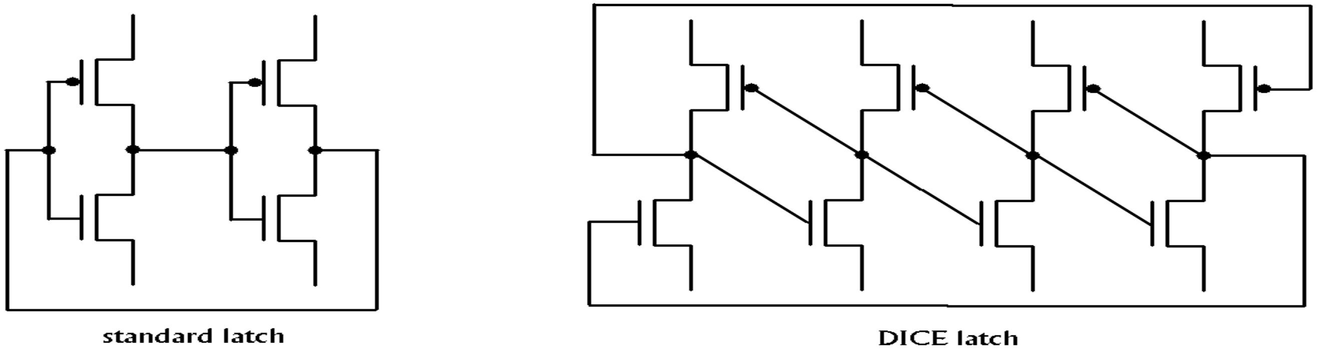
- ***Single Event Latchup (SEL)***
  - **SEE induced triggering of parasitic thyristors**
  - **Mitigation by process and library cell design**
- ***Single Event Upset (SEU) in Flip-Flops and SRAM***
  - **SEE glitch inside the bistable feedback loop of storage point**
  - **Immediate bit flip → loss of information, change of state, functional fault**
- ***Single Event Transients (SET) in clocks and resets***
  - **Glitches on clocks → change of state, functional fault**
  - **Asynchronous resets are clock-like signals**
- ***Single Event Transients (SET) in combinatorial logic***
  - **SEE glitches in combinatorial logic behave like cross-talk effects**
  - **Causes SEU when arriving at flip-flop/memory D-input during clock edge**
  - **Sensitivity increases with clock frequency**
  - **Synchronous resets are like combinatorial signals**

Mitigation of SEU in Flip-Flops

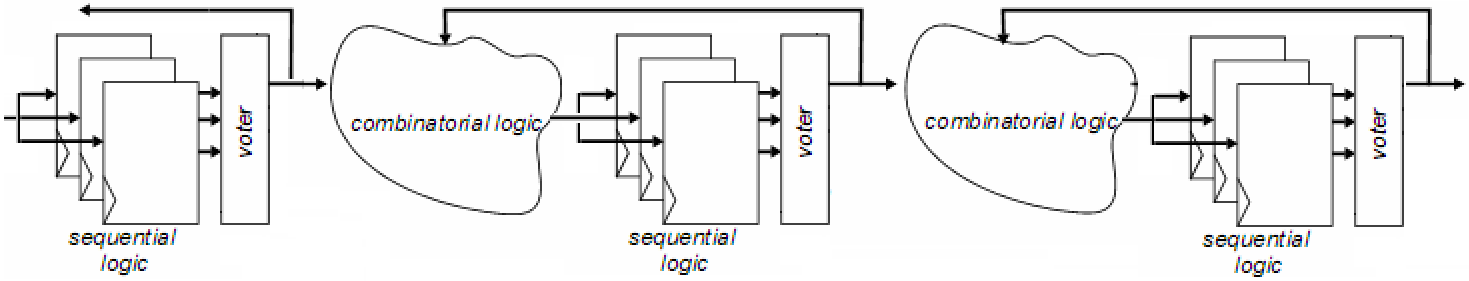
- Standard synchronous RTL design



- SEU hardened flip-flops

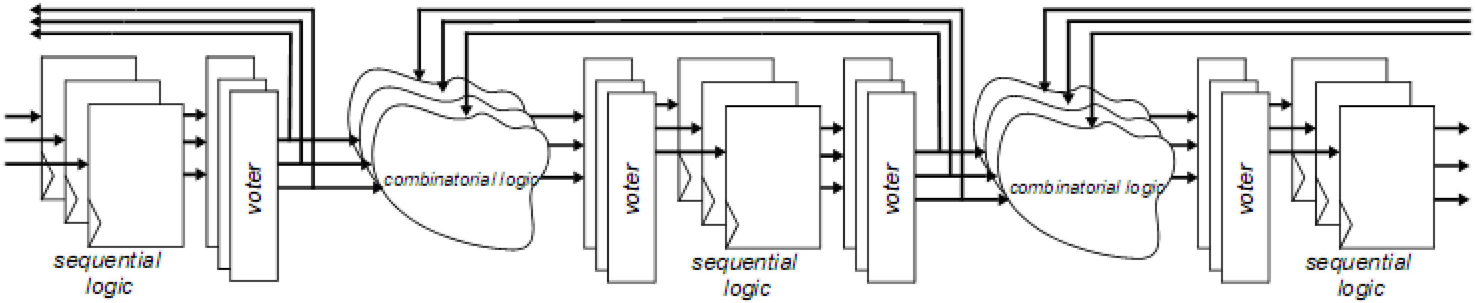


- Triple Modular Redundancy (TMR) flip-flops



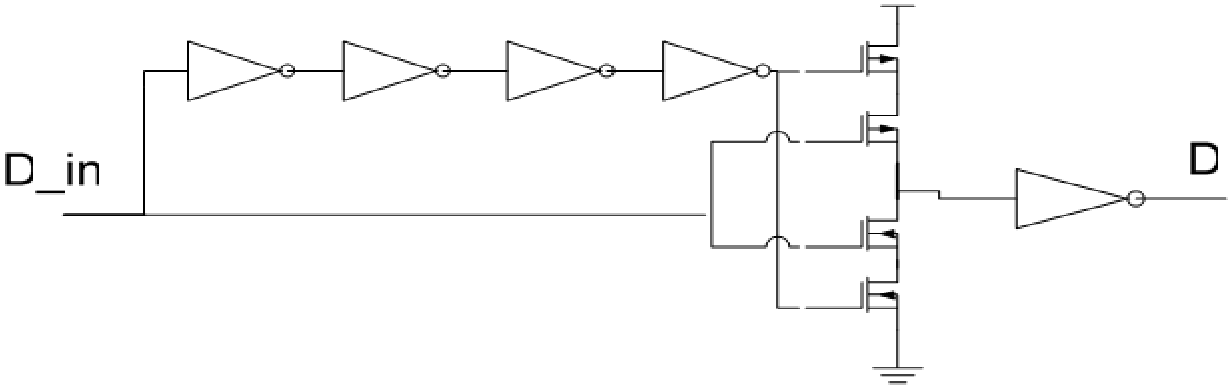
Mitigation of combinatorial SET

- ***Triple redundancy of flip-flops and combinatorial logic [4]***



- ***Glitch Filtering on all flip-flop inputs [5]***

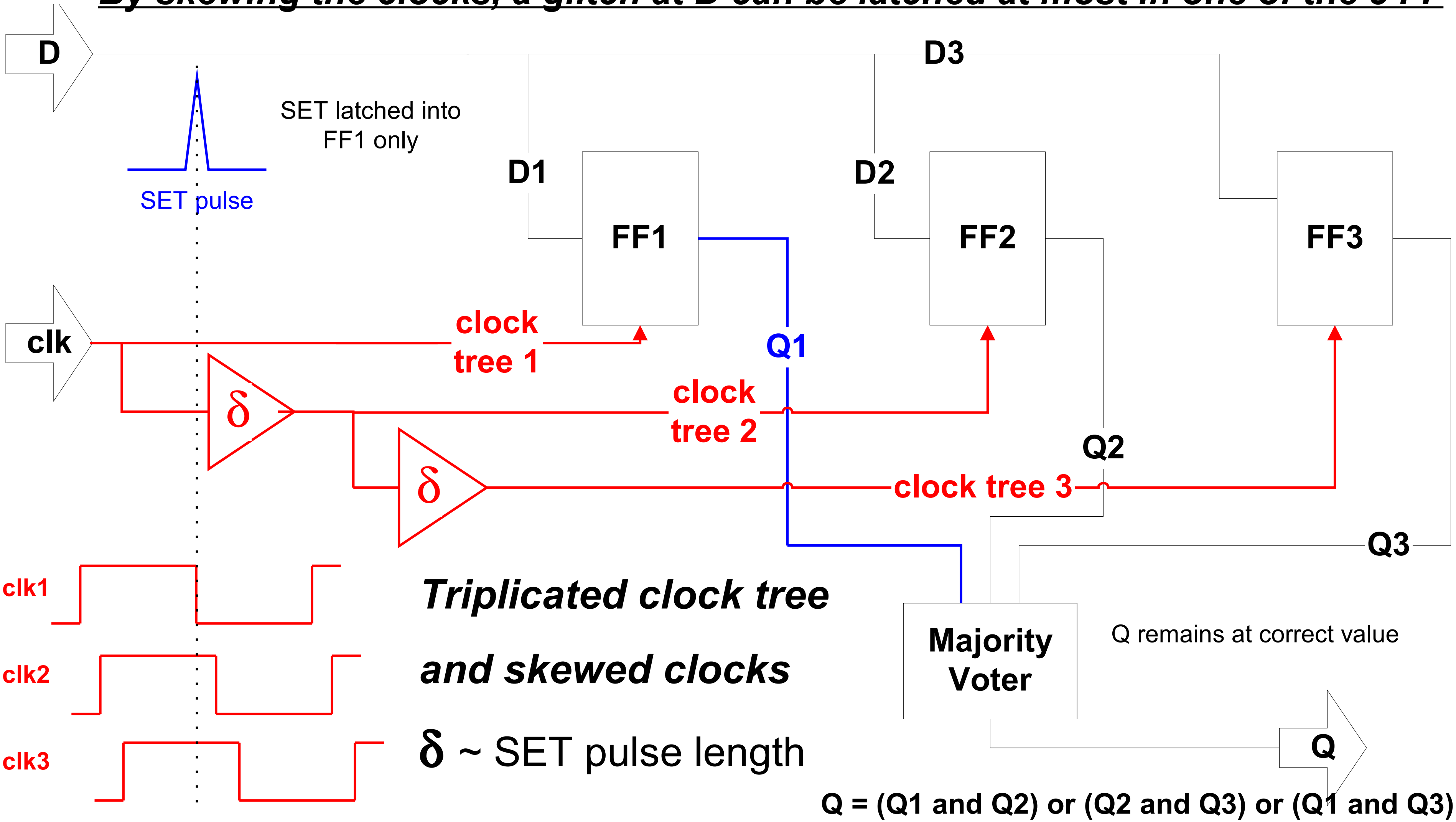
  (P. Mongkolkachit, Pitsini; Bharat Bhuva, 2003)



- ***STMR: TMR flip-flop with triple skewed clock trees***
  - ***Selected for the AT697 microprocessor, see next slide...***

STMR: TMR with triple skewed clock

By skewing the clocks, a glitch at D can be latched at most in one of the 3 FF



Triplicated clock tree and skewed clocks

δ ~ SET pulse length

Q = (Q1 and Q2) or (Q2 and Q3) or (Q1 and Q3)

Impacts on the RTL-GDS design flow

- *Insertion of STMR into the design*
  - Create TMR flip-flops in RTL or post-synthesis
  - Generation of triple skewed clock trees
- *Increased complexity affects the design flow and –results*
  - Increased cell and node count → higher tool runtime (or crashes)
  - Optimisation is less efficient, higher interconnect delay
- *Synthesis tools are designed to remove <u>redundancy</u>*
  - Don't use sequential optimisation (register merging, pipelining, retiming)
- *Timing issues*
  - TMR voters and clock skewing reduces maximum speed
  - 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

STMR insertion at RTL or gate level

STMR in VHDL

- Clock nets/ports are a vector of 3 bit
- Use the “two-process” method [6]

-- One process per TMR domain:

```
rx0 : process(clk) begin
      if rising_edge(clk(0)) then r0 <= d;
end if; end process;
rx1 : process(clk) begin
      if rising_edge(clk(1)) then r1 <= d;
end if; end process;
rx2 : process(clk) begin
      if rising_edge(clk(2)) then r2 <= d;
end if; end process;
```

-- Vote outputs

```
r <= (r0 and r1) or (r0 and r2) or (r1 and r2);
```

- Synthesis with TMR in one go
- Disallow register merging
- Structural verification required

STMR at gate level

- Used mainly for third party IP
- Library and tool dependent
- Synthesise netlist without TMR
- Create HDL package with TMR equivalent macro-cells
- Edit netlist to triplicate clocks and asynchronous resets

```
sed -e 's/CLK\(.*\) std_logic/CLK\1
std_logic_vector(2 downto 0) /'
```

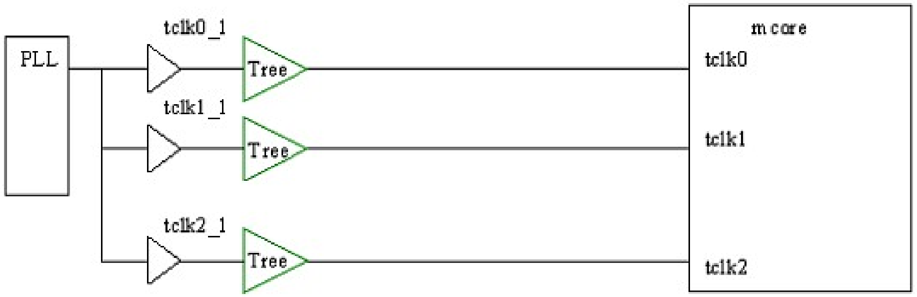
- Edit netlist replacing every flip-flop by its TMR equivalent

```
sed -e 's/DFF1/DFF1_TMR/'

sed -e 's/DFF2/DFF2_TMR/'
```

- Resynthesise the edited netlist, linking with the TMR macro-cell package
- Disallow register merging
- Structural verification required

Inserting triple skewed clock/reset trees



- **Clock Tree Synthesis (CTS) optimises skew inside a clock tree**
  - Need control over the insertion delay ($\delta 1 = \delta 2$)
  - Synthesis of several coherent trees not provided by CTS
  - Compromise: insert three distinct trees with well adjusted CTS parameters
- **Delay δ inserted at the origin of the clock trees**
  - Instantiate delay buffers in the VHDL source code for simulation
  - Model δ at synthesis by *set_ideal_latency* and *set_propagated_clock*
  - Initial value for δ is speculative → control/adjustment in backend process
- **Combinatorial logic on clock/asynchronous reset**
  - Needs to be triplicated as well

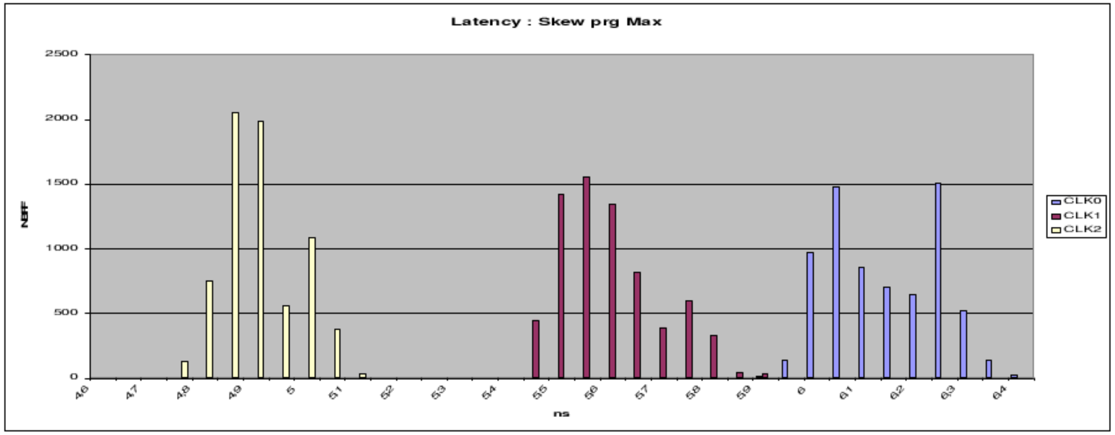
Coherent clock trees

We need to control the relative clock latency:

X

X+δ

X+2*δ

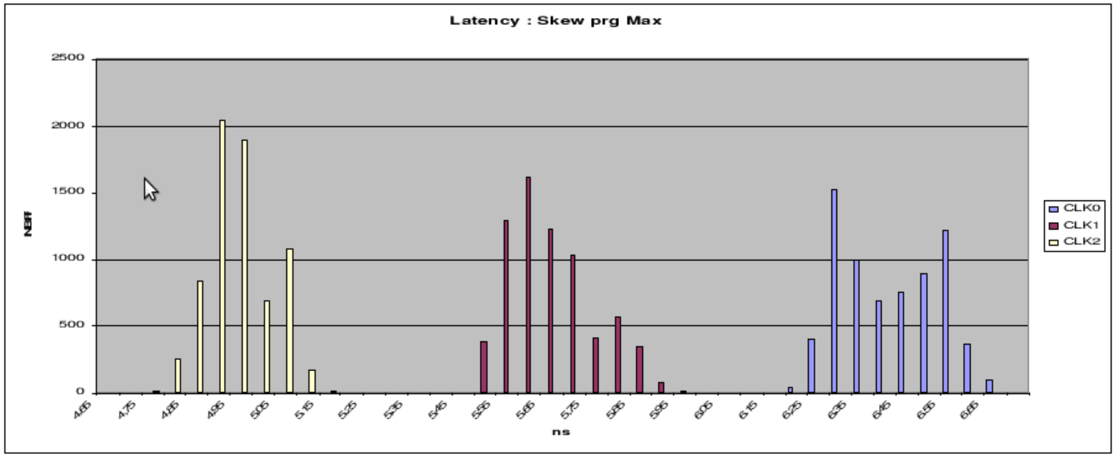


Before manual adjustment

CTS did not achieve goal

→

Manual adjustment of delay elements required



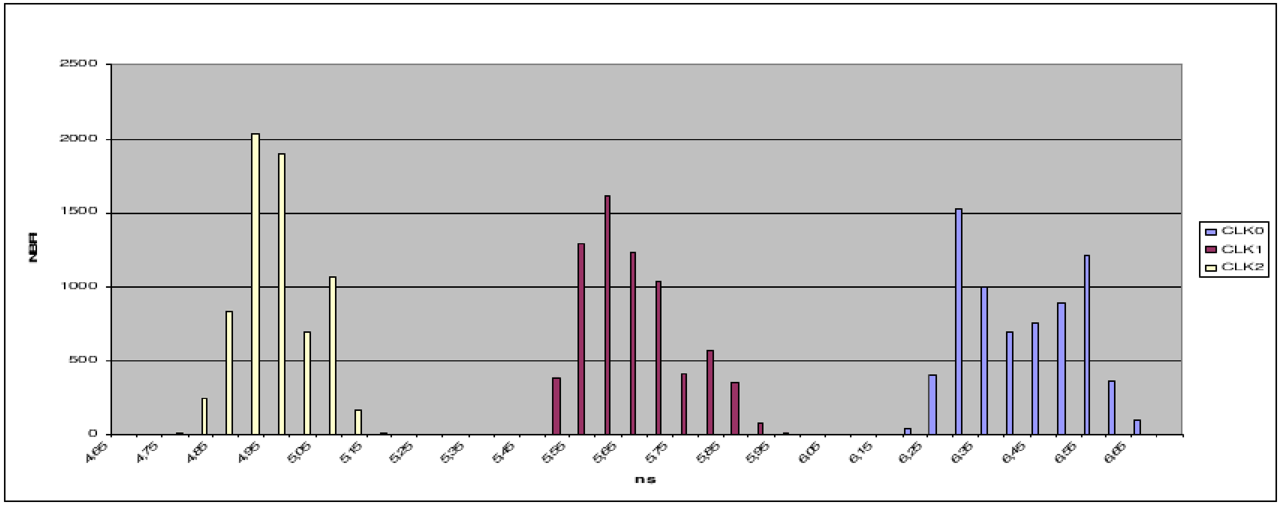
After manual adjustment

Mastering skew inside each clock tree

Above:

δ ~ 800 ps

high variance

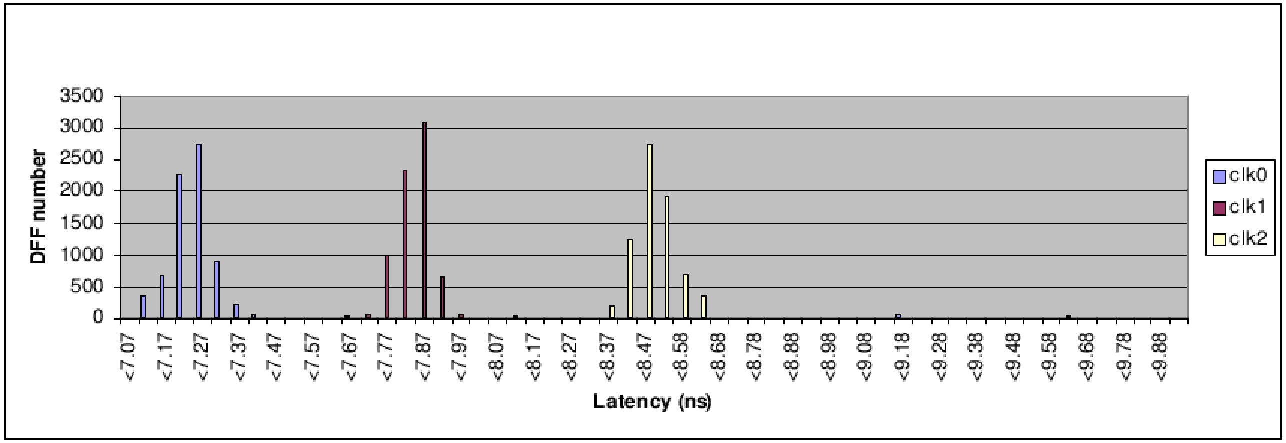


AT697E clock latencies

Below:

δ ~ 600 ps

low variance



AT697F clock latencies

Verification of STMR

- *TMR is larger and slower than normal flip-flops*
  - » Redundancy removed by logic optimisation (synthesis and back-end)
  - » TMR modified by timing optimisation
- *Defects in redundant structures do not appear at simulation*
  - » TMR simulation “works” even if only two of the three FF are correct

????????????

Verification of STMR

- *TMR is larger and slower than normal flip-flops*
  - » Redundancy removed by logic optimisation (synthesis and back-end)
  - » TMR modified by timing optimisation
- *Defects in redundant structures do not appear at simulation*
  - » TMR simulation “works” even if only two of the three FF are correct

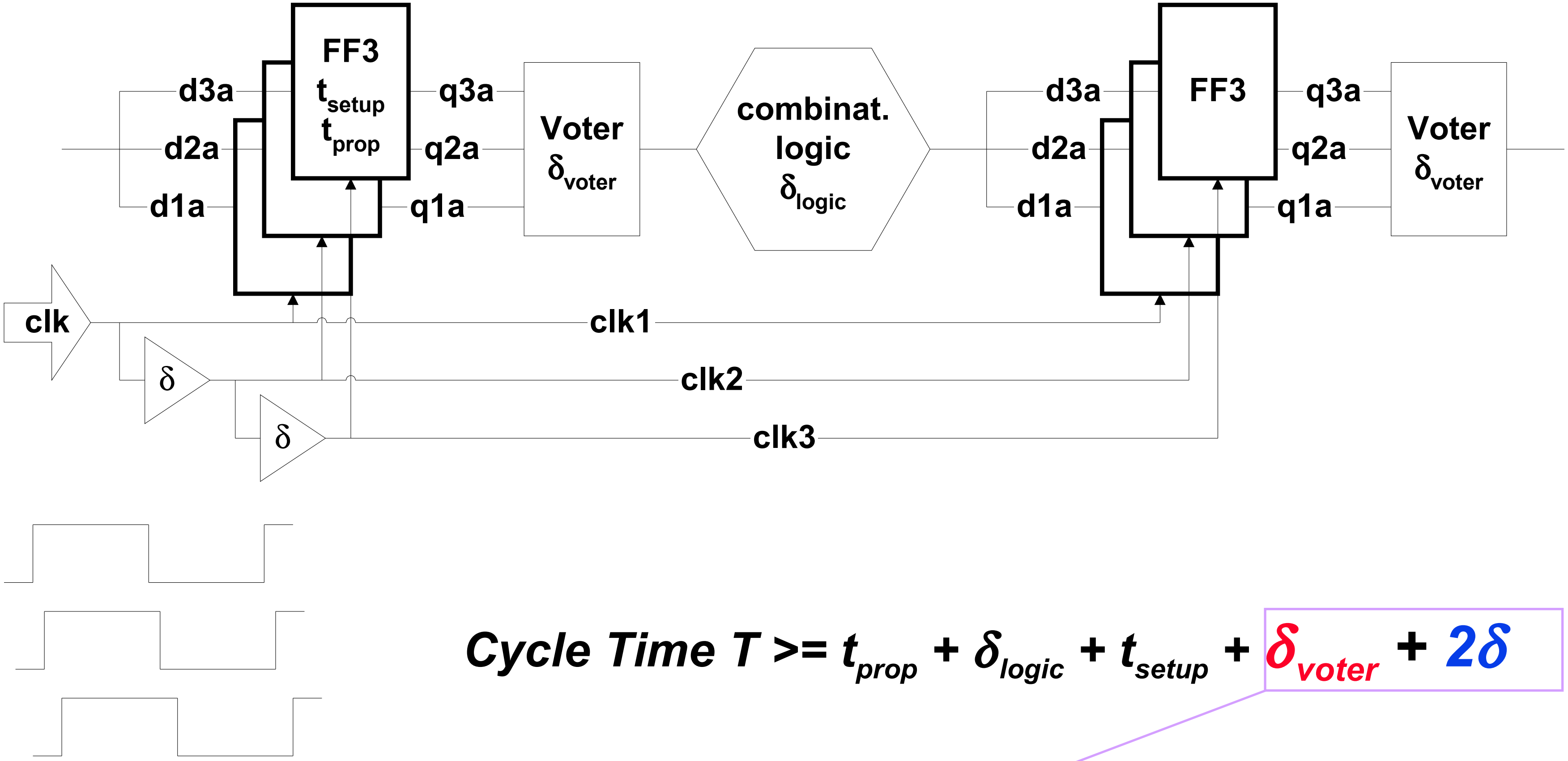
⇒⇒⇒⇒⇒⇒⇒

- *Structural and formal verification required*
  - » Presence of triple FF, correct wiring of the three clock/reset domains
  - » Parsing the netlist with scripts (grep)
  - » Increasing complexity requires formal verification tools
- *Timing analysis of clock trees*
  - » Measure insertion delay from clock root (PLL) to every flip-flop
  - » Difference between clock arrival and data arrival

Structural and Formal Verification

- *COTS formal verification tools get confused*
  - Netlist contains three FF for one described in RTL
  - Workarounds: declare equivalence of flip-flops
  - Script/constraint was provided by tool vendor
- *Structural verification of TMR*
  - Netlist parsing was used in our project
  - Formal verification, custom tool developed at ESA [7]
  - NASA/Mentor: Formal verification for TMR designs [3]
- *Fault injection*
  - Fault injection by simulation
    - Example: SST, an SEU simulation tool developed at ESA [8]
  - Fault emulation by FPGA emulation
    - Example: FT-Unshades [9]
- *Radiation Testing*
  - Expensive, and only after manufacturing

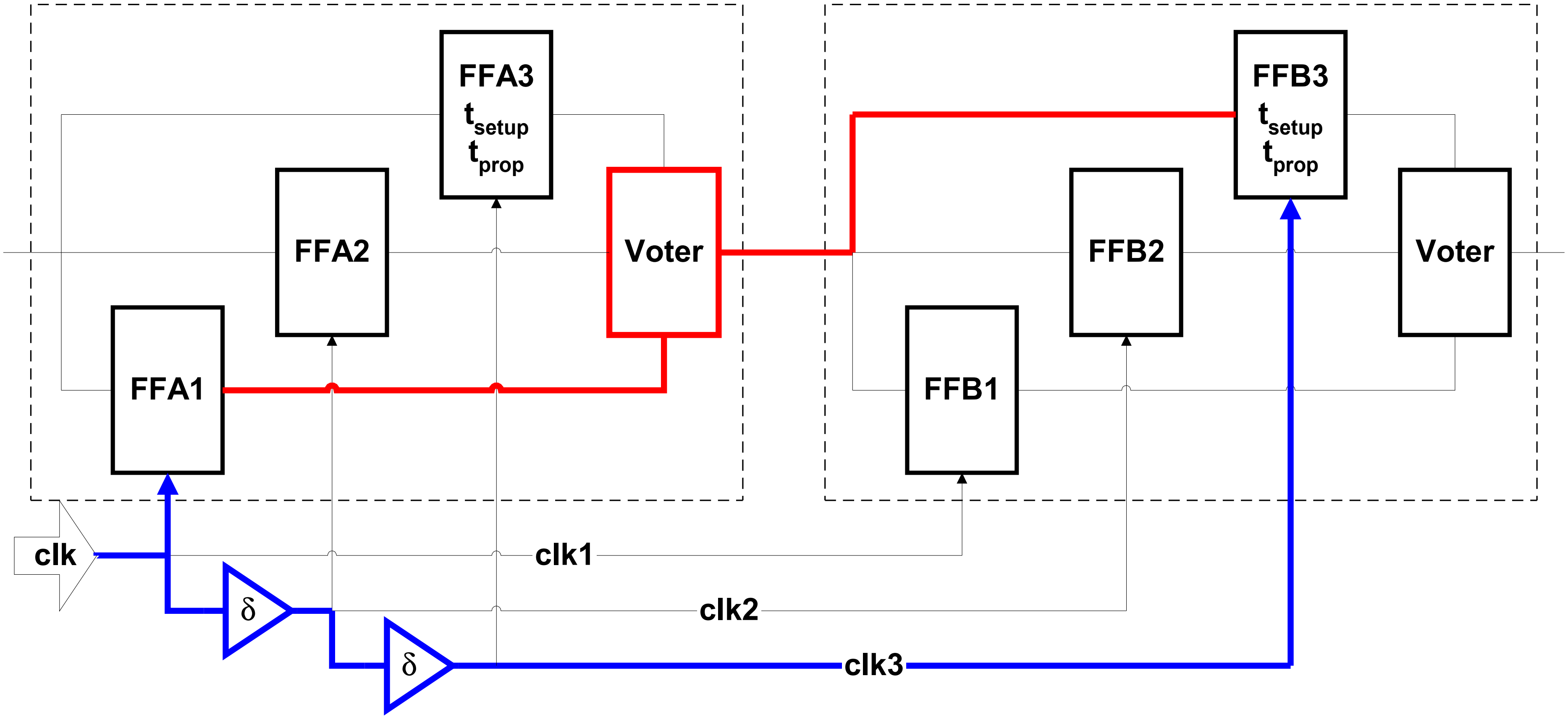
TMR Timing Issues



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

TMR voters and clock skewing reduce operating frequency

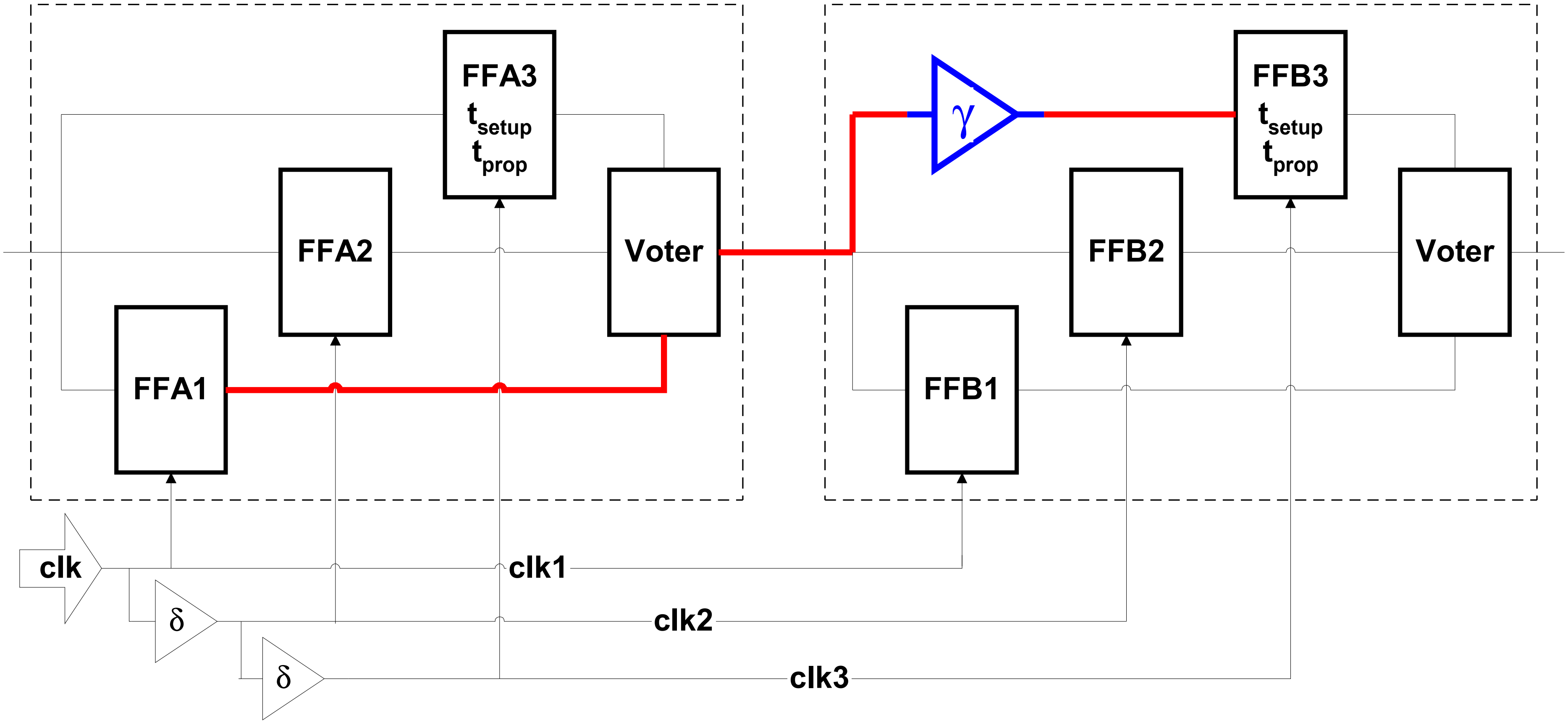
Hold violations with skewed clocks



When propagation delays (t$_{prop}$, voter) < (2 δ) clock skew

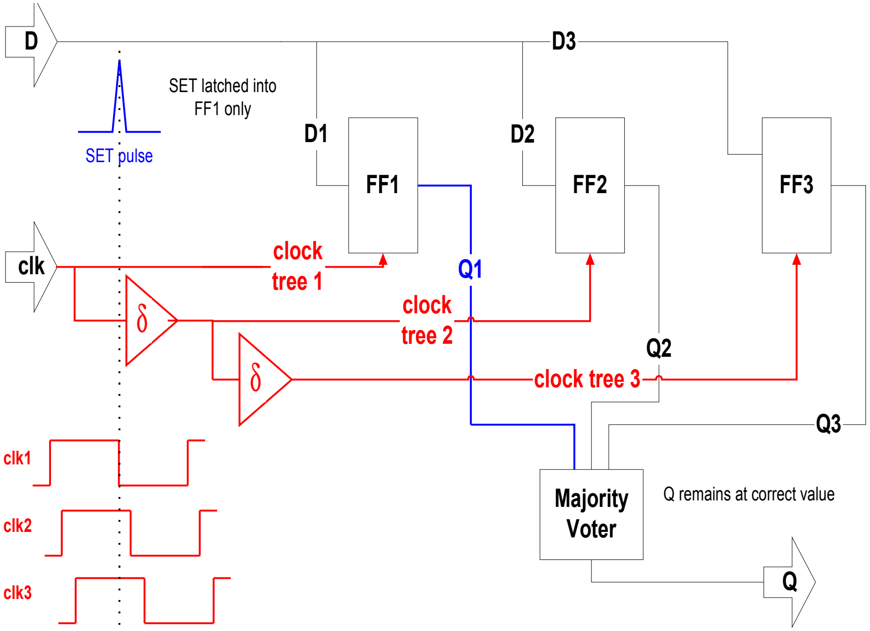
➔ hold violation FFA1 ➔ FFB3

Wrong hold fix by EDA tool



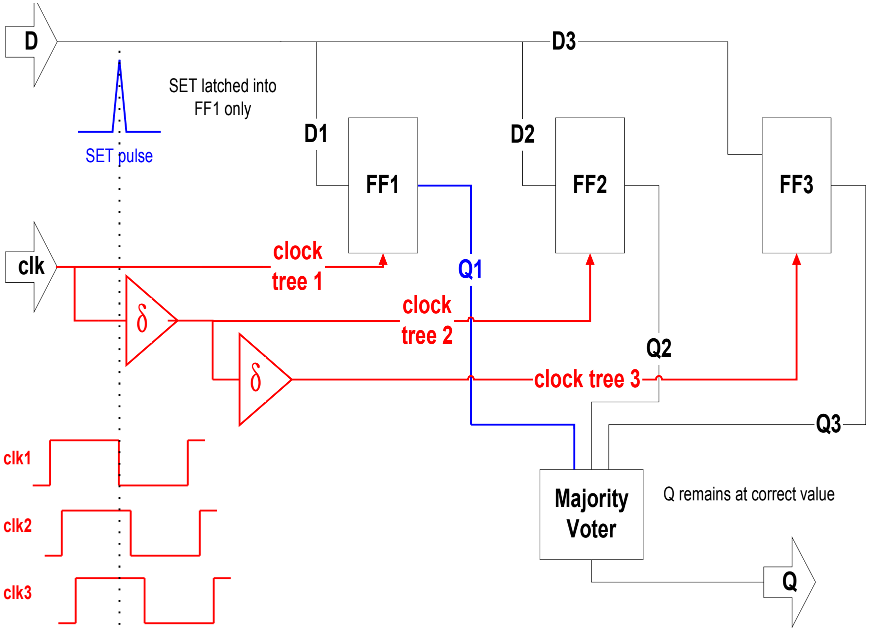
Automatic buffer insertion by fix-hold of backend tool compensates clock skew → and spoils SET protection

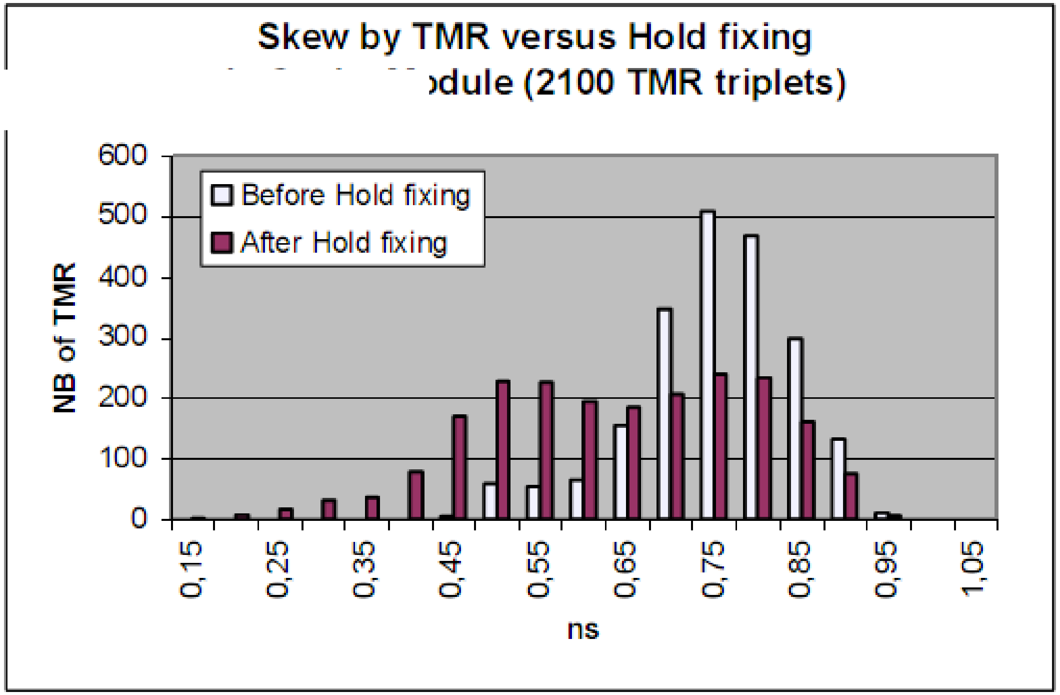
Clock spread dilution by wrong hold fix



- ***Difference between clock and data arrival in each TMR triplet***

Clock spread dilution by wrong hold fix

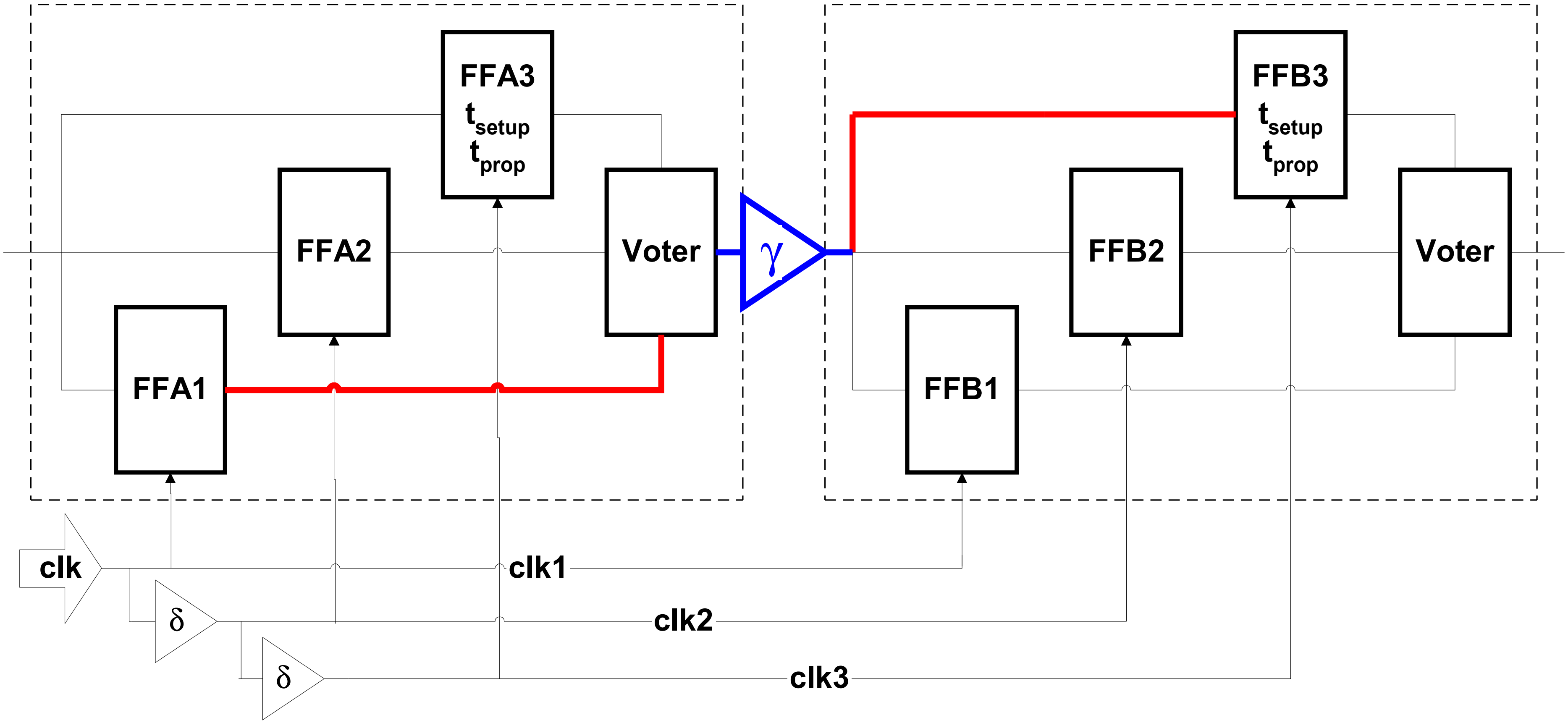




[T(clk2) – T(d2)] – [T(clk1) – T(d1)]

- ***Difference between clock and data arrival in each TMR triplet***
  - → **Before hold-fix: well pronounced peak $\delta_{eff} = \delta_{nominal}$**
  - → **Clock skew creates many hold violations**
  - → **After wrong hold-fix: two maxima (with and without delay insertion)**

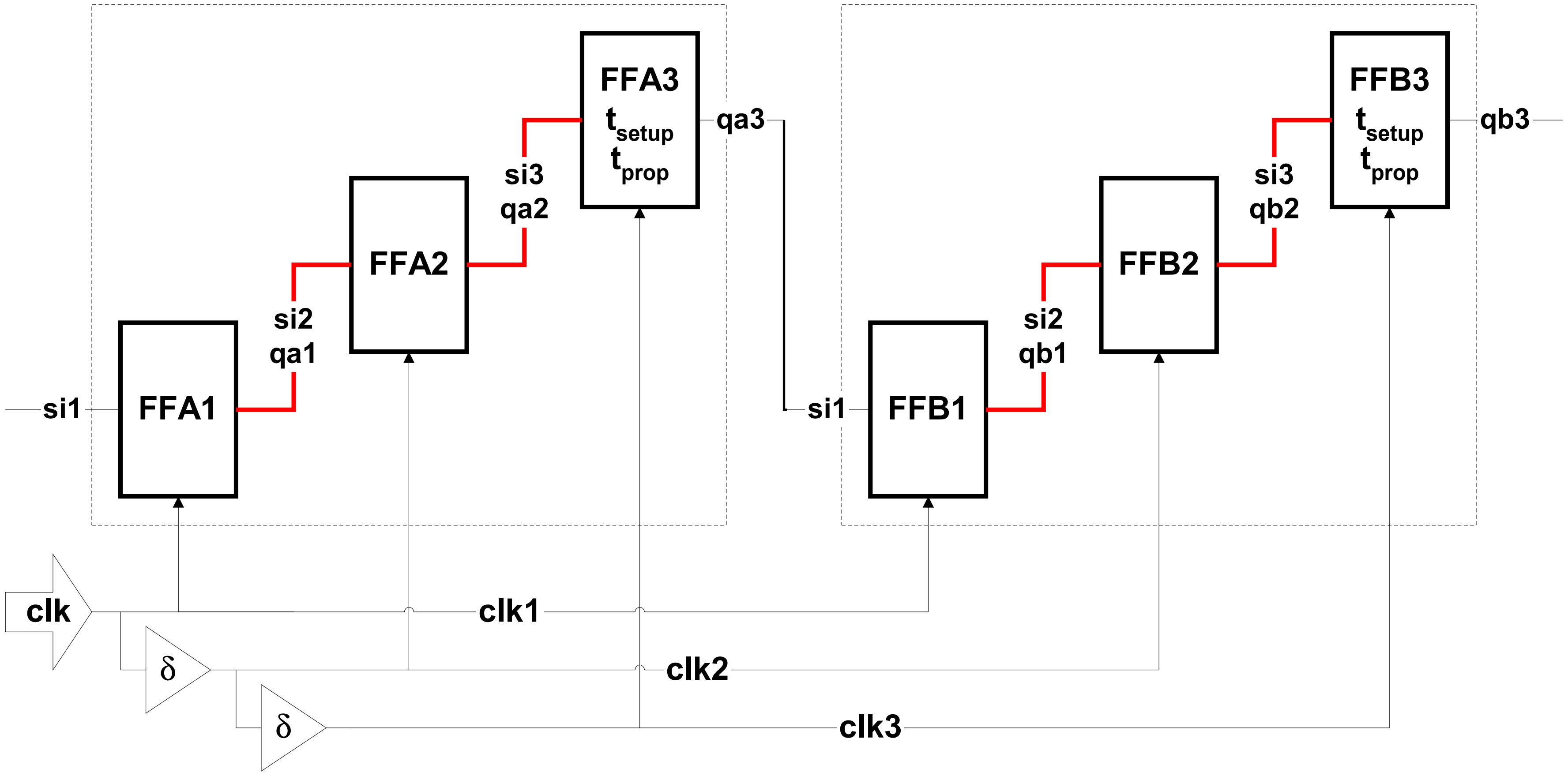
Correct hold fix



Group FF belonging to the same triplet and dont_touch

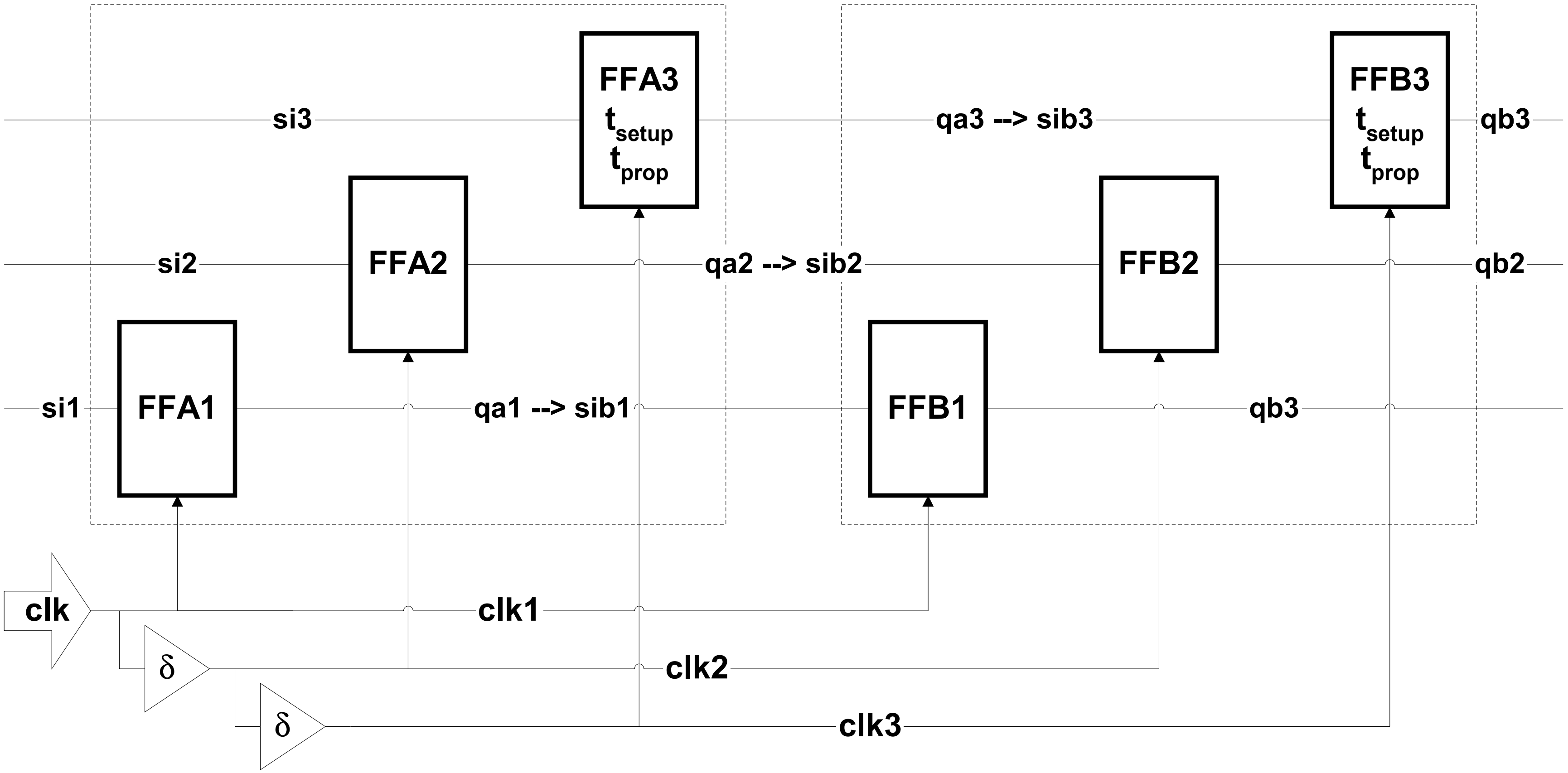
➔ *SET protection through clock skew conserved*

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

Overheads: area, power, performance

- ***TMR flip-flops***
  - Area overhead >~ factor 3
  - Power consumption ~ factor 3
- ***Performance impact***
  - $(\delta_{voter} + 2\delta)$ /period(10 ns) → ~ 10-20%
  - Secondary effect: larger design, higher interconnect

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

- ***Area overhead is mainly on flip-flops***
  - Cost for fault tolerance (FT) depends on sequential/combinatorial ratio
  - Design with 50% flip-flops (non-FT) gets twice as big (3*seq + 1*comb)
- ***Additional overhead for hold-fix buffer insertion***
  - δ >~ 600 ps → hold violation in feedback path of enable flip-flops
  - Many enable flip-flops implemented in the processor (pipeline freeze function)
  - Instance count in netlist increased from 160000 to 260000 after hold fix
  - Remedy: reduce the deliberate clock skew δ
    - trade-off SET protection (value of δ) and hold margin (guard band)

Protection of SRAM blocks

- *EDAC (Error Detection And Correction) for register files*
  - **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 for cache memories*
  - **Simple parity 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)**

- *EDAC for external RAM*

EDA tool issues

- *Increasing SEE awareness also in non-space designs*
  - **High availability products**
    - » Networking
    - » Medical
    - » Aircraft
- *ASIC EDA tools have little support for SEE-tolerant design*
  - **Hotlines may help to find workarounds**
  - **Built-in support in some FPGA tools, pushed by FPGA vendors**
  - **Dedicated tool development difficult to reach same performance**
- *Our wish list for tools*
  - **Controlling optimisation of redundancy**
  - **Generation (CTS) of triple coherent clock trees**
  - **Formal verification for TMR designs**
- *Alternative: SEU and SET protected flip-flop as library cells*
  - **DICE cells + glitch filtering at inputs**

Conclusion

- ***SEU and SET protection possible with standard flip-flops***
  - → **STMR: TMR flip-flops with three phase-skewed clock trees**
  - → **Protection of memories by parity and EDAC schemes**
  - → **About 100% area and power overhead (design dependent)**
  - → **About 10-20% speed performance reduction**
- ***STMR requires tricks and workarounds in the design flow***
  - → **Mastering clock skew and hold fix**
  - → **Prevent optimising away the desired redundancy**
- ***Thorough verification required***
  - → **Classical verification methods may fail or do not detect the errors**
  - → **Scripts or special tools required**
- ***Our activities***
  - → **http://www.esa.int/TEC/Microelectronics**

Questions?

References/Links (1)

[1] **AT697E, AT697F and ATC18RHA page at Atmel**

http://www.atmel.com/dyn/products/product_card.asp?part_id=3178

http://www.atmel.com/dyn/products/product_card.asp?part_id=4599

http://www.atmel.com/dyn/products/product_card.asp?part_id=2318

[2] **The LEON2-FT IP core**

http://www.esa.int/TEC/Microelectronics/SEMUD70CYTE_0.html

[3] **Melanie Berg: Design for Radiation Effects**

http://nepp.nasa.gov/mapld_2008/presentations/i/01%20-%20Berg_Melanie_mapld08_pres_1.pdf

[4] **Sandi Habinc: Functional Triple Modular Redundancy (FTMR)**

http://microelectronics.esa.int/techno/fpga_003_01-0-2.pdf

[5] **Mongkolkachit, P.; Bhuva, B.: Design technique for mitigation of alpha-particle-induced single-event transients in combinational logic**

IEEE Transactions on Device and Materials Reliability, Sept. 2003

[6] **Jiri Gaisler: A structured VHDL design method**

http://www.gaisler.com/doc/vhdl2proc.pdf

References/Links (2)

[7] **Simon Schulz, Giovanni Beltrame, David Merodio Codinachs: Smart Behavioural Netlist Simulation for SEU Protection Verification**

http://microelectronics.esa.int/papers/SimonSchulzInFault.pdf

[8] **SST: The SEU Simulation Tool**

http://microelectronics.esa.int/asic/SST-FunctionalDescription1-3.pdf

http://www.nebrija.es/~jmaestro/esa/sst.htm

[9] **FT-Unshades, a Xilinx-based SEU emulator**

http://microelectronics.esa.int/mpd2004/FT-UNSHADES_presentation_v2.pdf

[10] **Actel Core generator**

http://www.actel.com/documents/EDAC_AN.pdf

[11] **The Xilinx XTMR tool**

http://klabs.org/mapld05/presento/238_rezgui_p.ppt