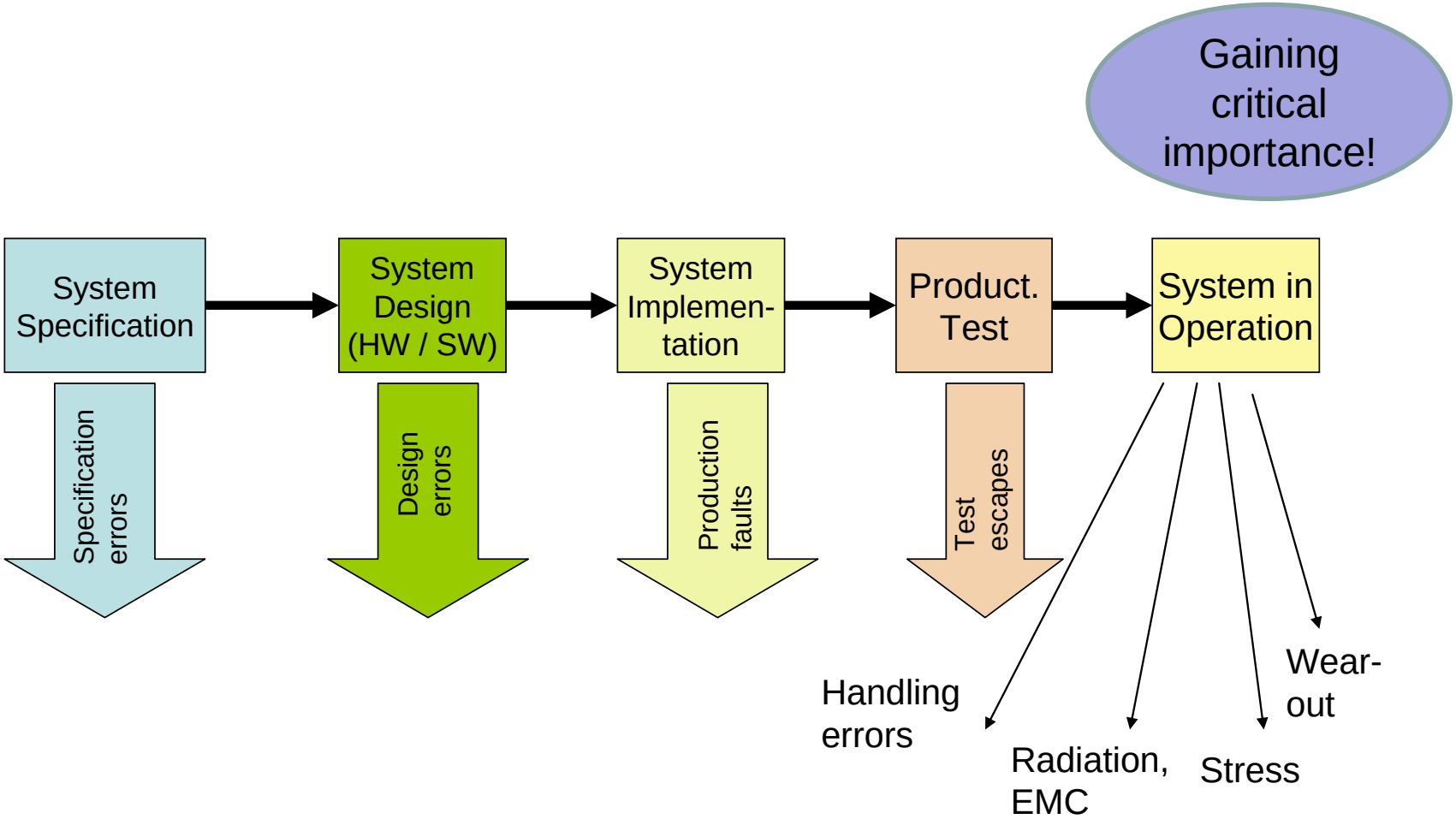
Dependability and Fault Tolerance
Zuverlässigkeit und Fehlertoleranz

Chapter 5: Fault behavior and fault / error correction for digital circuits

H. T. Vierhaus

Winter Semester 2018 / 2019

Faults / Errors from where ?



Hardware Faults and Sources

Particle Radiation: Mainly affects memory cells, flip-flips, registers
Apparently less harmful than predicted 10-15 years ago at „ground level“. Becomes a strong issue at high elevations and in space flight, also in radiating environments.

Electromagnetic (EM) Coupling :

Is becoming a strong issue for IC- operating voltages and signal high / low voltage margins of 1 V and less. Mechanisms are internal coupling of lines on one hand and disturbing EM-radiation from outside the system.

Wear-out Effects:

Negative / positive bias thermal instability (NBTI / PBTI), effects shifts of MOS transistor threshold voltages, mostly causing extra delays.

Metal migration, may cause interrupts on interconnects (Voids)

Dielectric gate rupture, may cause shorts through transistor gate oxides

..all enhanced by high field strength and high temperatures !!

Transient Hardware Faults

Sources:

- Particle-radiation from atomic decay
- Cosmic / solar radiation (airplanes !!)
- EM-coupling

Effects:

- Discharge / change of memory cells
- Switching of Flip-Flops
- Switching (mostly glitches) of logic nodes
- CMOS Latch-Up: breakdown of p-n-junctions, strong reverse current, distruction by overheating

Permanent Faults

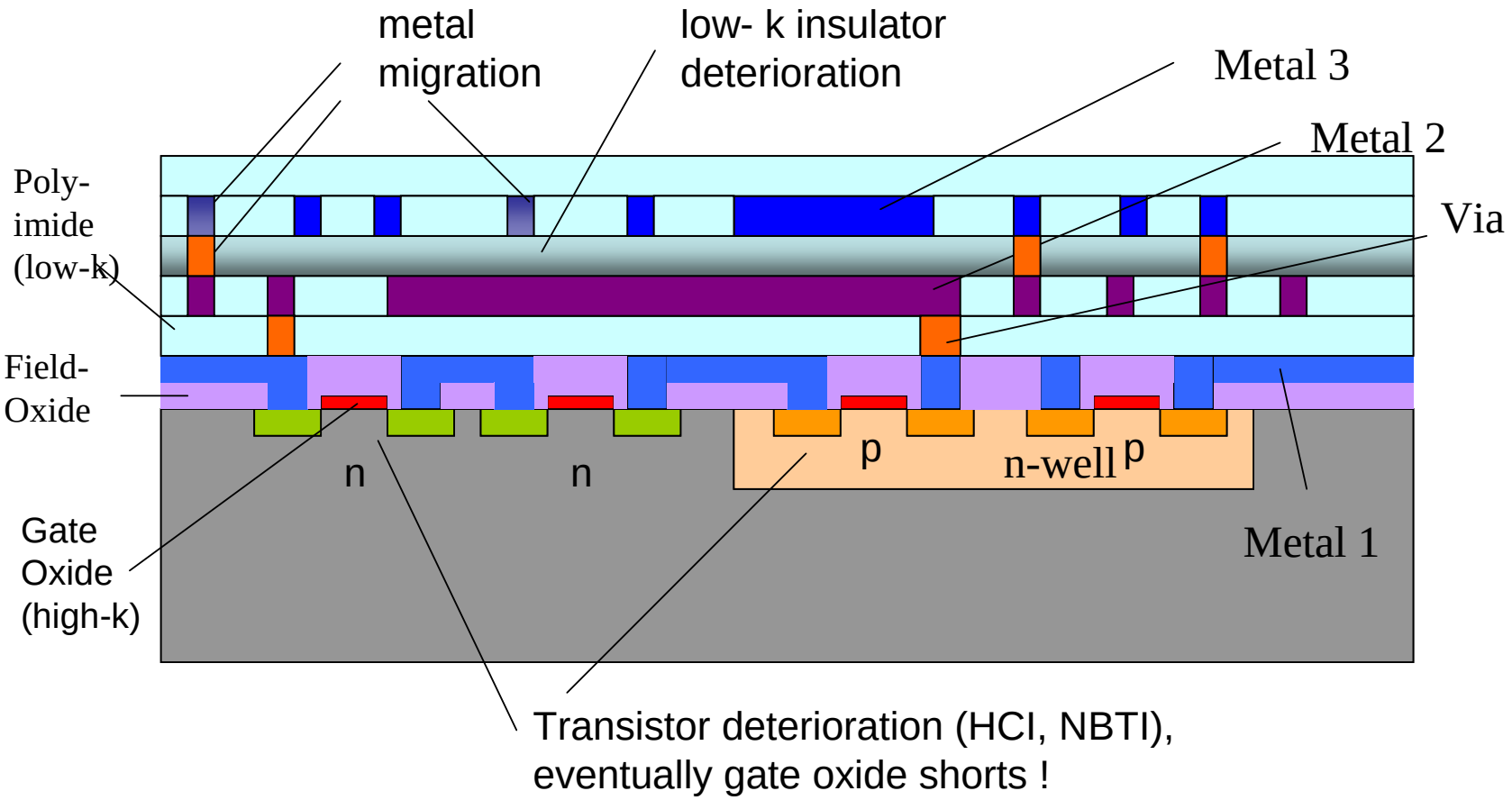
Reasons: Defects from production , Wear-Out

Types of Faults: - Bridges between lines
- Leakage / short trough insulation layers (gates, field-oxide)
- Line interrupts (e. g. from electromigration)

IC production test is supposed to find any such faults / defects before an IC goes into real-life operations !

But IC production test technology is impossible to apply to ICs already „in the field“.

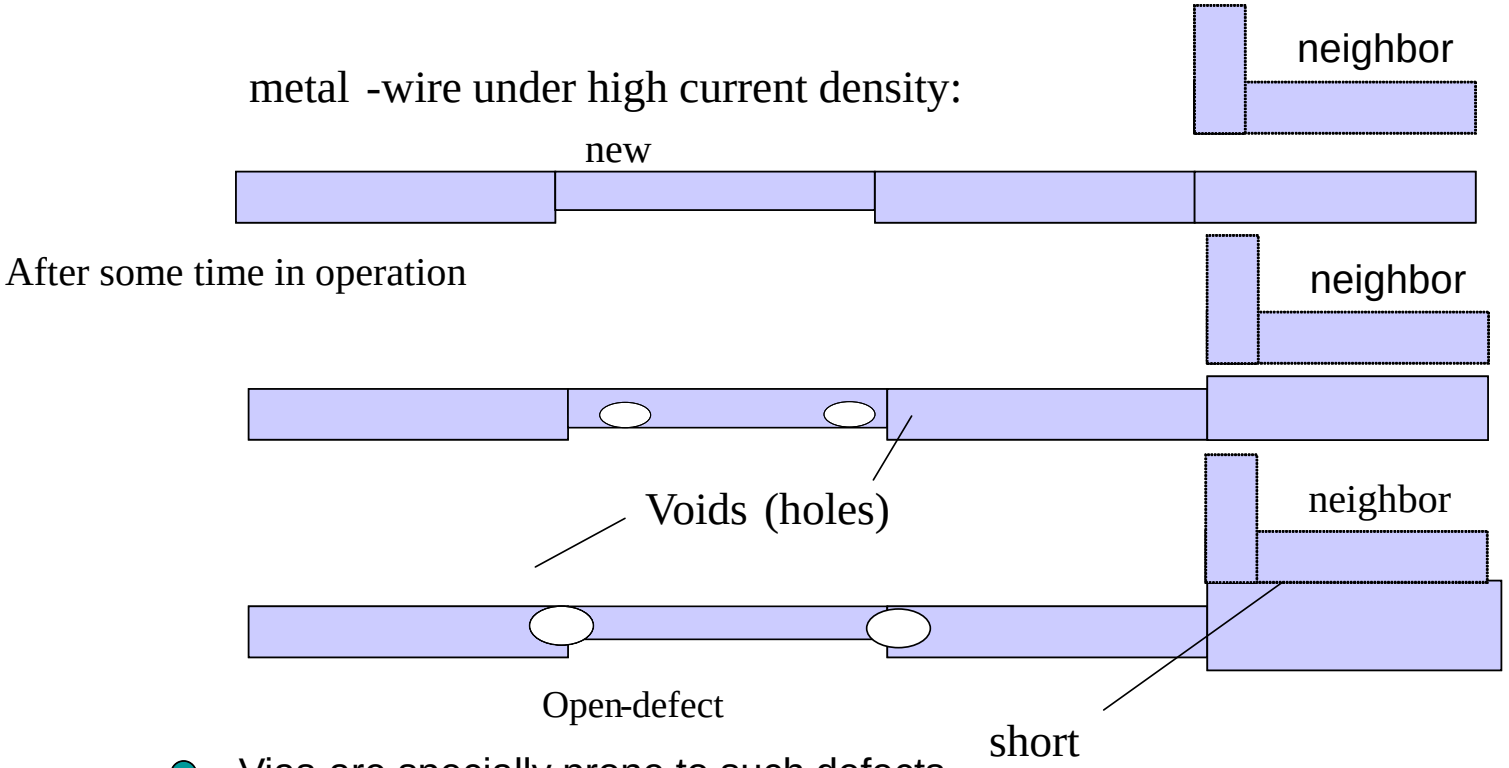
CMOS-IC-Structure



Transistor-Fault-Effects

- **Negative / Positive Bias Thermal Instability (NBTI / PBTI):** Migration of Hydrogen ions in the gate oxide results in higher (absolute) threshold voltages. The effect is strongest in p-channel transistors (NBTI). Results in reduced switching speed and longer delays. Occurs mainly after a gate has been biased constantly for a longer time. Partly reversible.
- **Hot Carrier Injection (HCI):** Electrons in transistor channel achieve high speed and high energy, which is enough to “tunnel” through the oxide. Occurs only on n-channel MOS.
- **Gate Oxide Deterioration / Rupture**: The thin gate oxide is damaged or ruptured, resulting in a conducting path between gate and channel. Permanent damage to the trans.
- **Dielectric Breakdown**: Also insulators between metal layers of ICs may occurs. Typically this implies resistive shorts between lines, resulting in signal lines connected to VDD or GND or bridges between signal nodes.

Metal Migration



- Vias are specially prone to such defects
- The effect is reversible by reversing the direction of current f low !

Metal Migration (2)

Metal Migration:

Metal atoms (Al, Cu) tend to migrate under high current density and high temperature.

Stress migration:

Migration effects may be enhanced under mechanical stress conditons.

Effect:

Metal lines and vias may actually cause line interrupts. The effect is partly reversible by changing current directions.

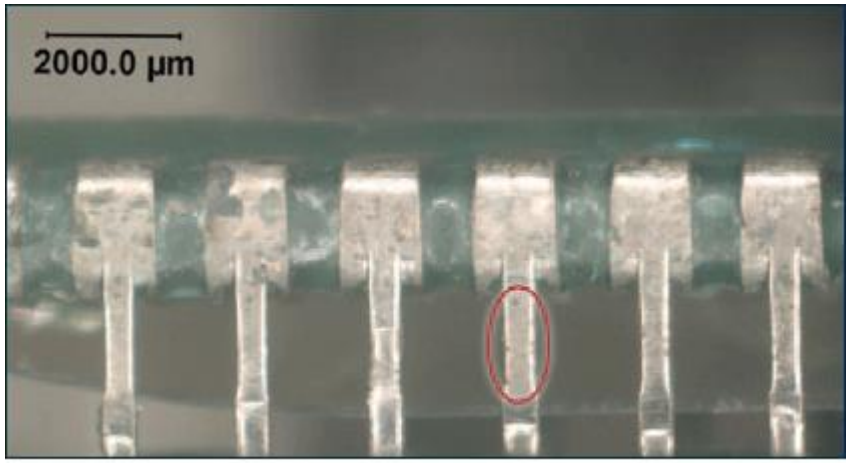
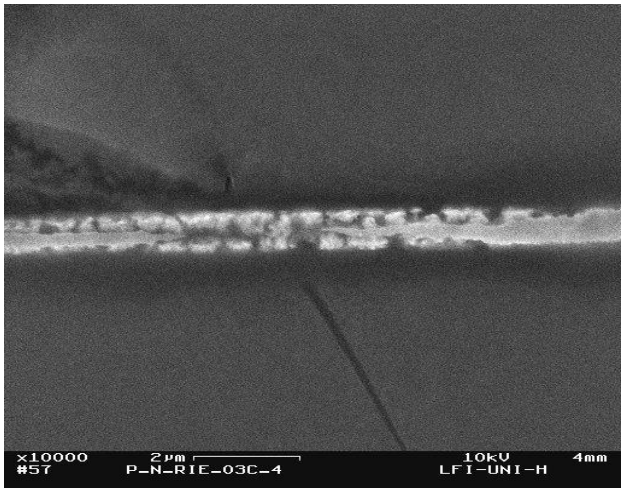
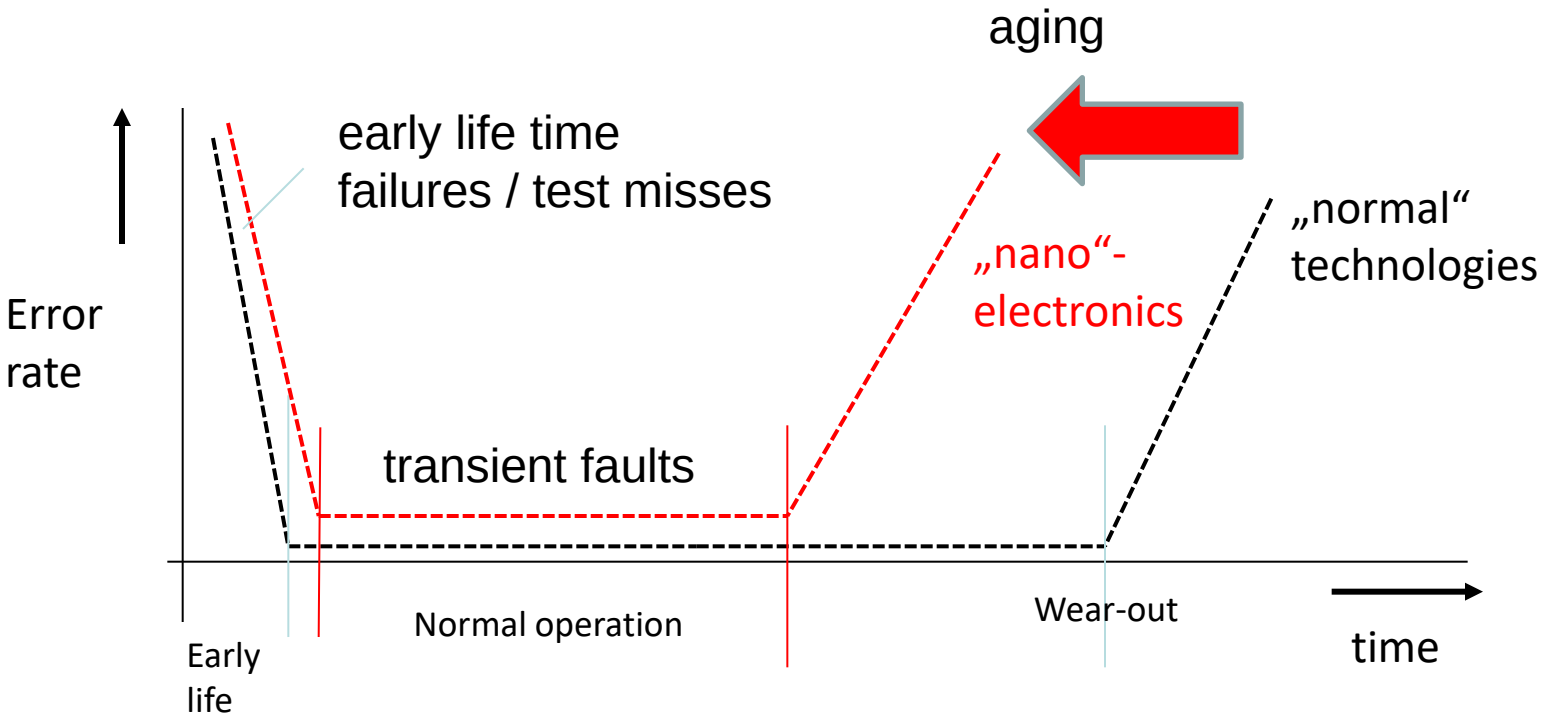


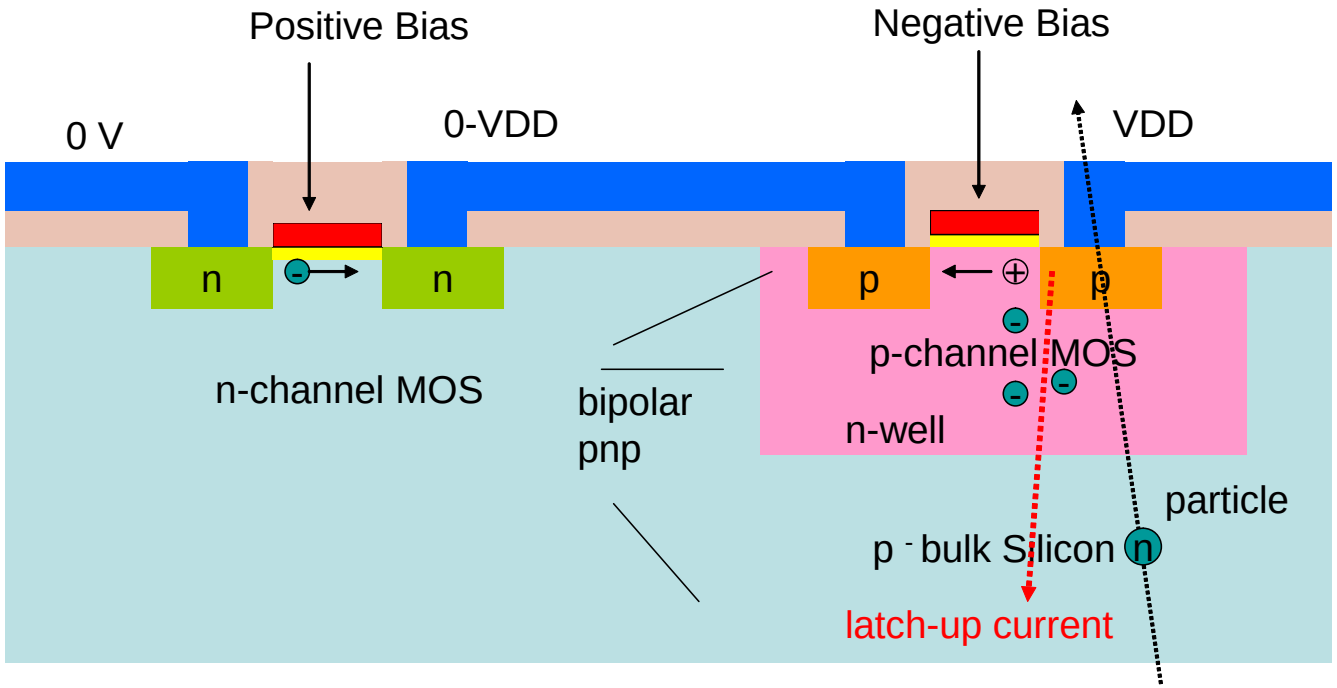
Figure 3-3: Pitting corrosion observed along the pins of the component.



Bathtub-Curve

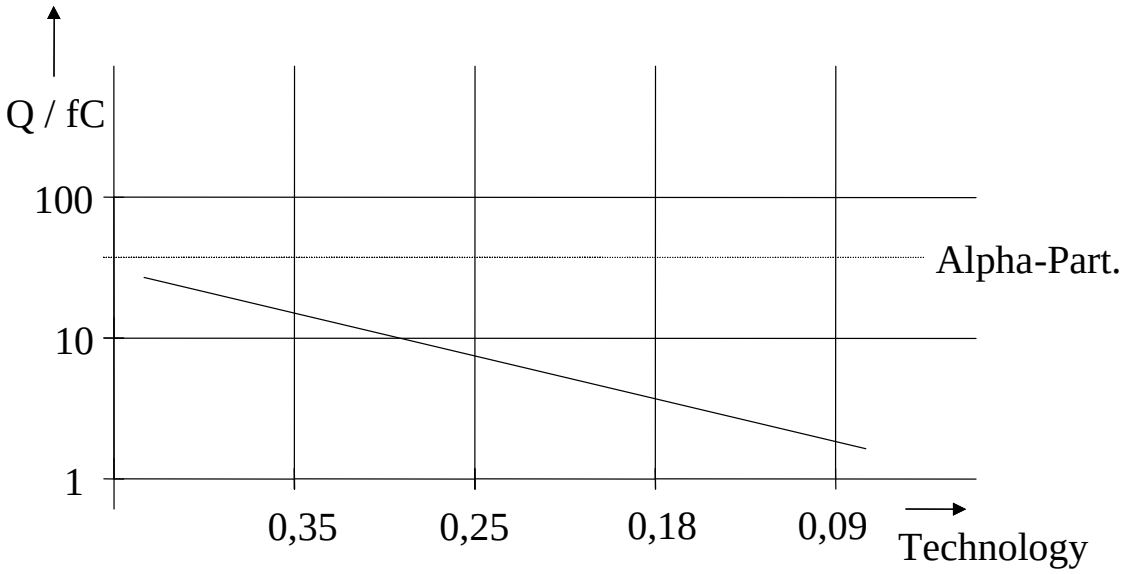


CMOS-Struture with „Latch-Up“



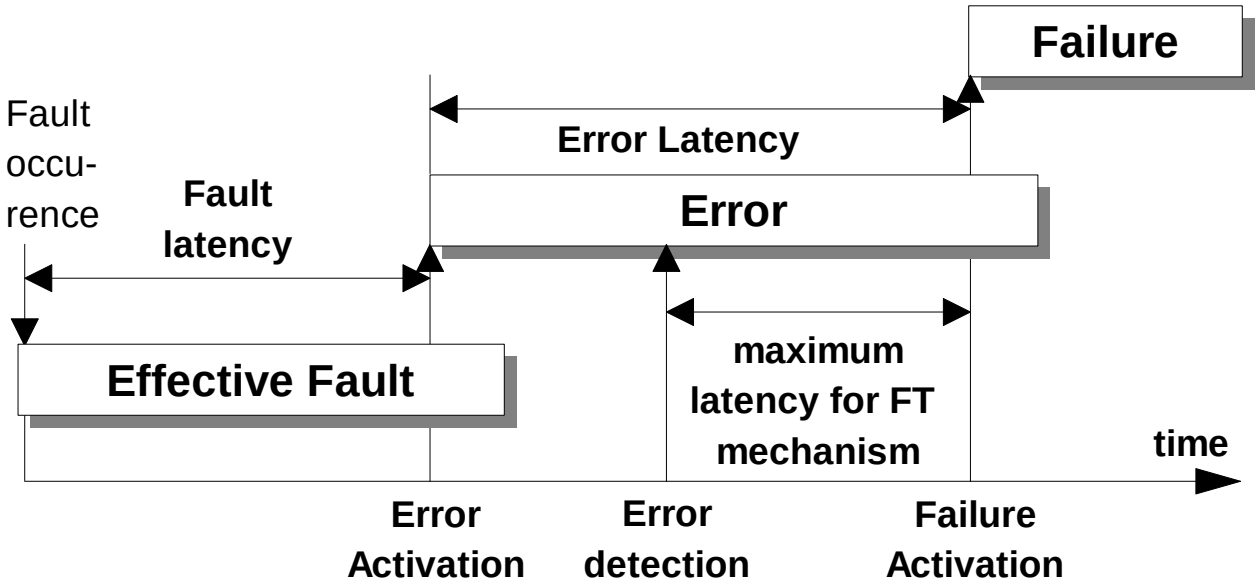
High-energy particle radiation may generate electrons in CMOS ICs
at the wrong place in the p-n-p-n structure, making reverse-biased p-n-junctions
„fire“ just like a silicon controlled rectifier (SCR) in forward direction.

Ionisation Potential



1 MeV Alpha-Particle generates 42 fC Charge!

Fault Tolerance Window



Error Correction

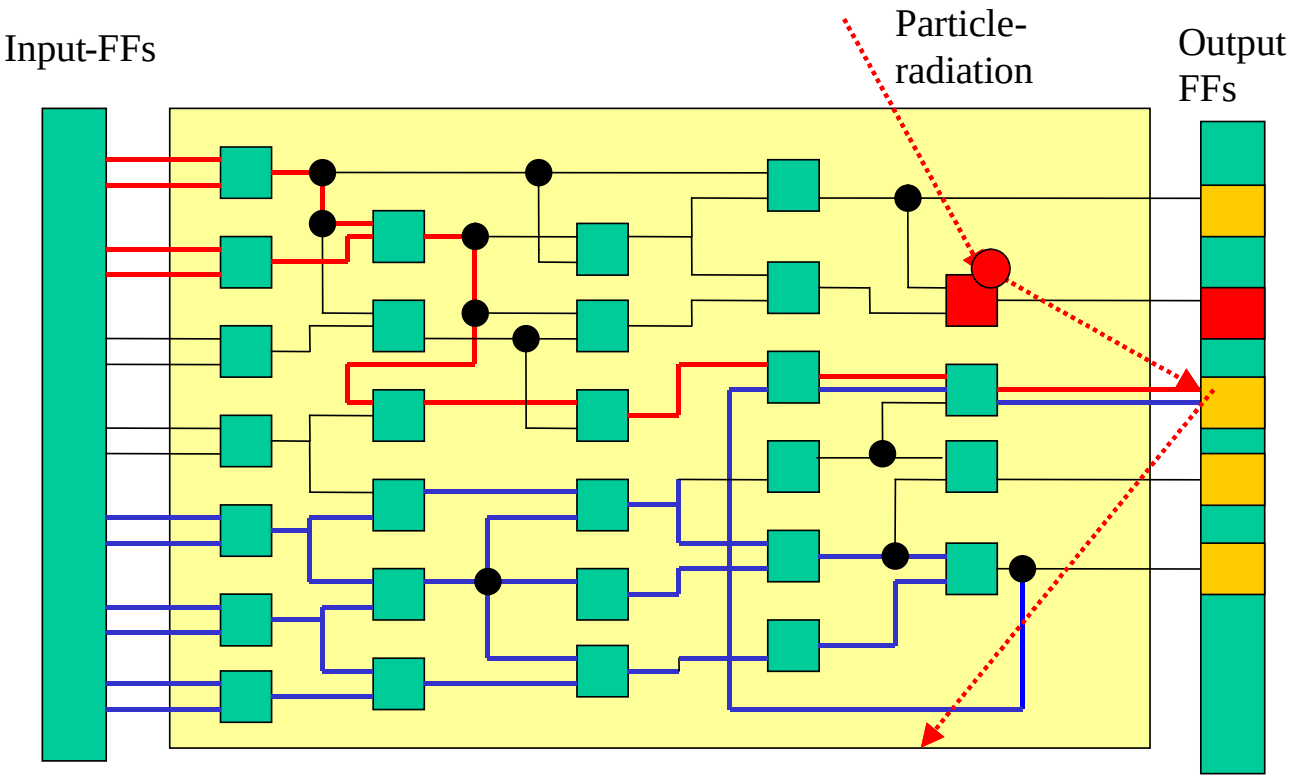
On-Line / in running operation:

- Detection / correction of short „glitch“-type faults (below half a clock cycle)
- Detection / correction for delay faults
- Detection / correction of delay faults of arbitrary length and permanent faults

Off-Line /in off-line phases:

- Correction of permanent faults by self repair / self healing

Combinational Circuit and Flip-Flops

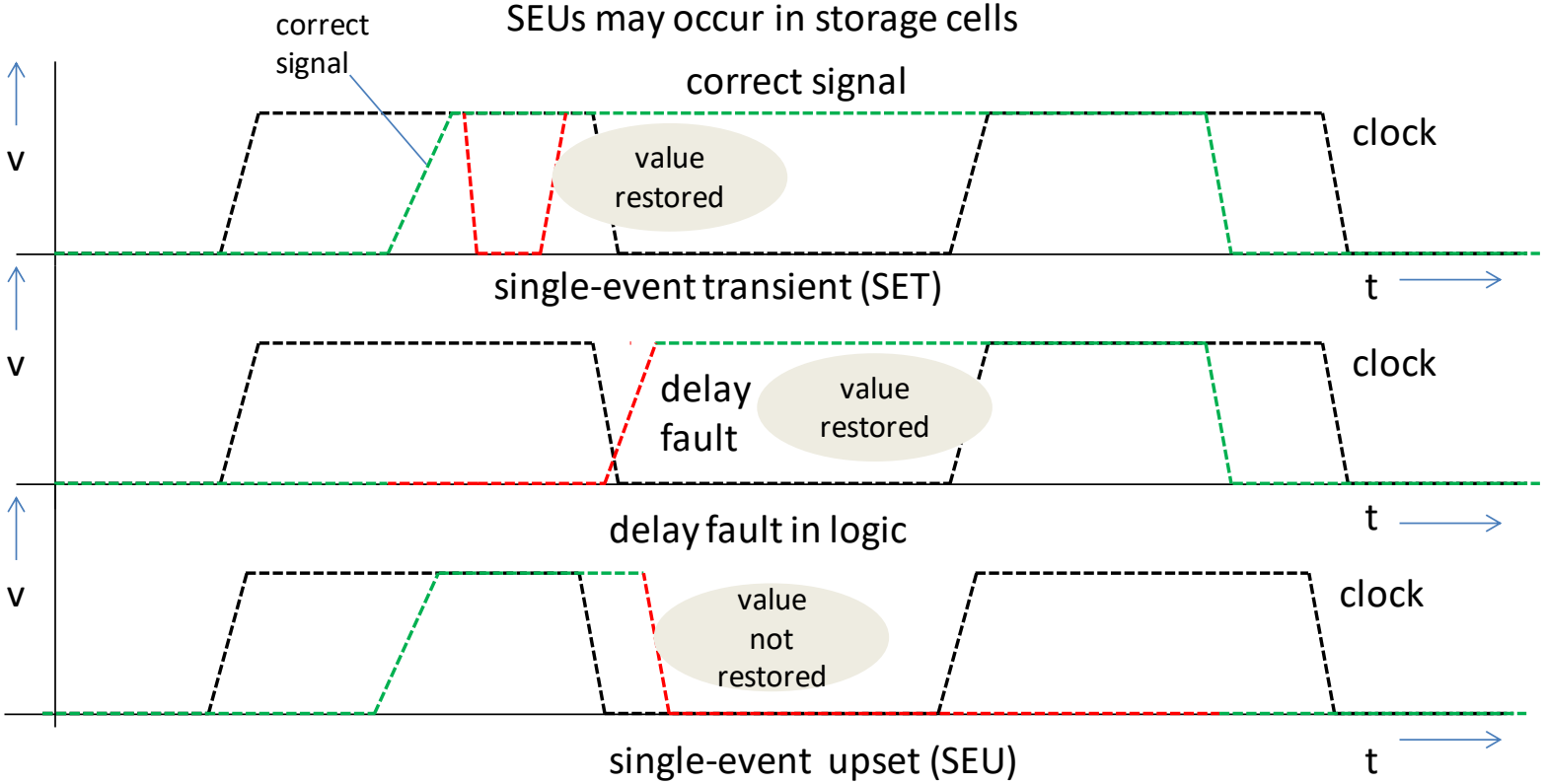


Memory cells and flip-flops are much more sensitive to radiation effects than combinational logic. In FFs in storage mode, a „node discharge“ triggers a single event upset (SEU), while logic nodes will mainly have a „single event transient“ and return to the correct value after a short glitch.

Types of Transient Faults (1)

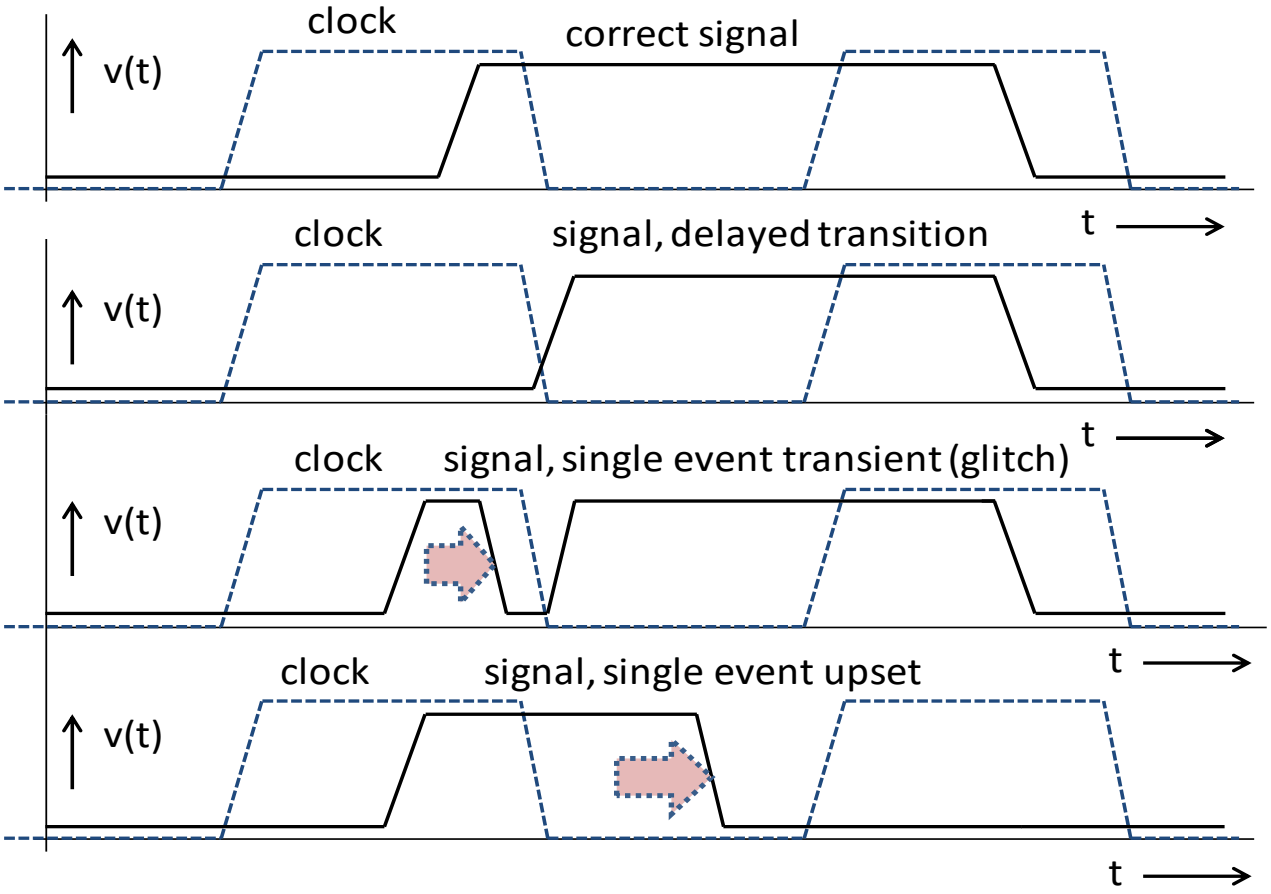
SETs typically occur in logic

SEUs may occur in storage cells



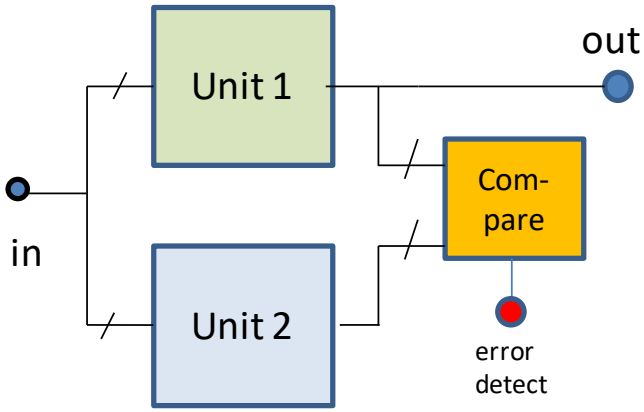
Transient faults or SEUs that last longer than a clock cycle T are not detectable by most methods targeting SETs or delay faults !

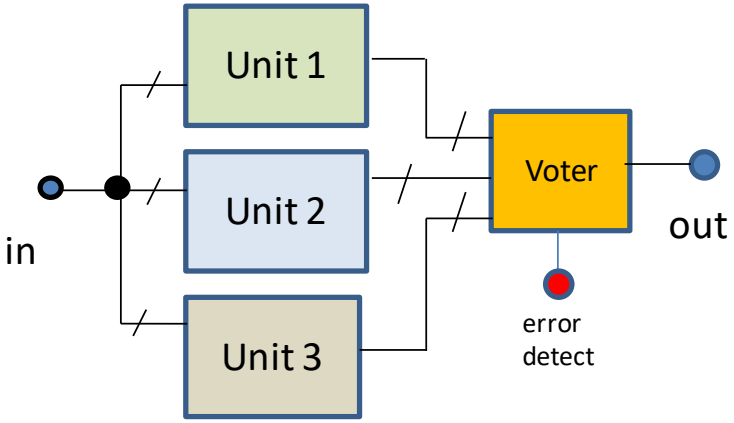
Types of Transient Faults (2)



Classical Methods

Basic Error Detection / Correction Schemes





Duplication and Compare (DMR):

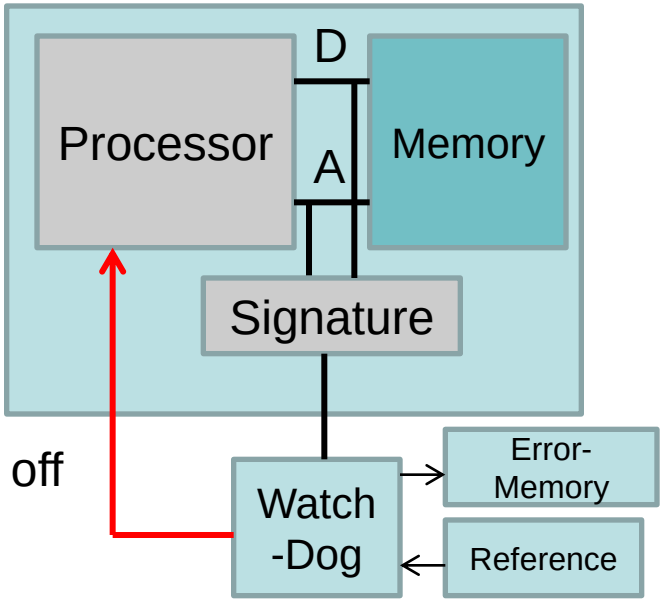
- small extra delay
- only error dection
- 100% overhead in power and energy
- provides full backup
- no extra clock control
- full coverage of static / dynamic faults

Triplication and Majority Vote (TMR):

- significant extra delay
- full error correction
- 200% overhead in power and energy
- provides full backup
- no extra clock control
- full coverage of static / dynamic faults

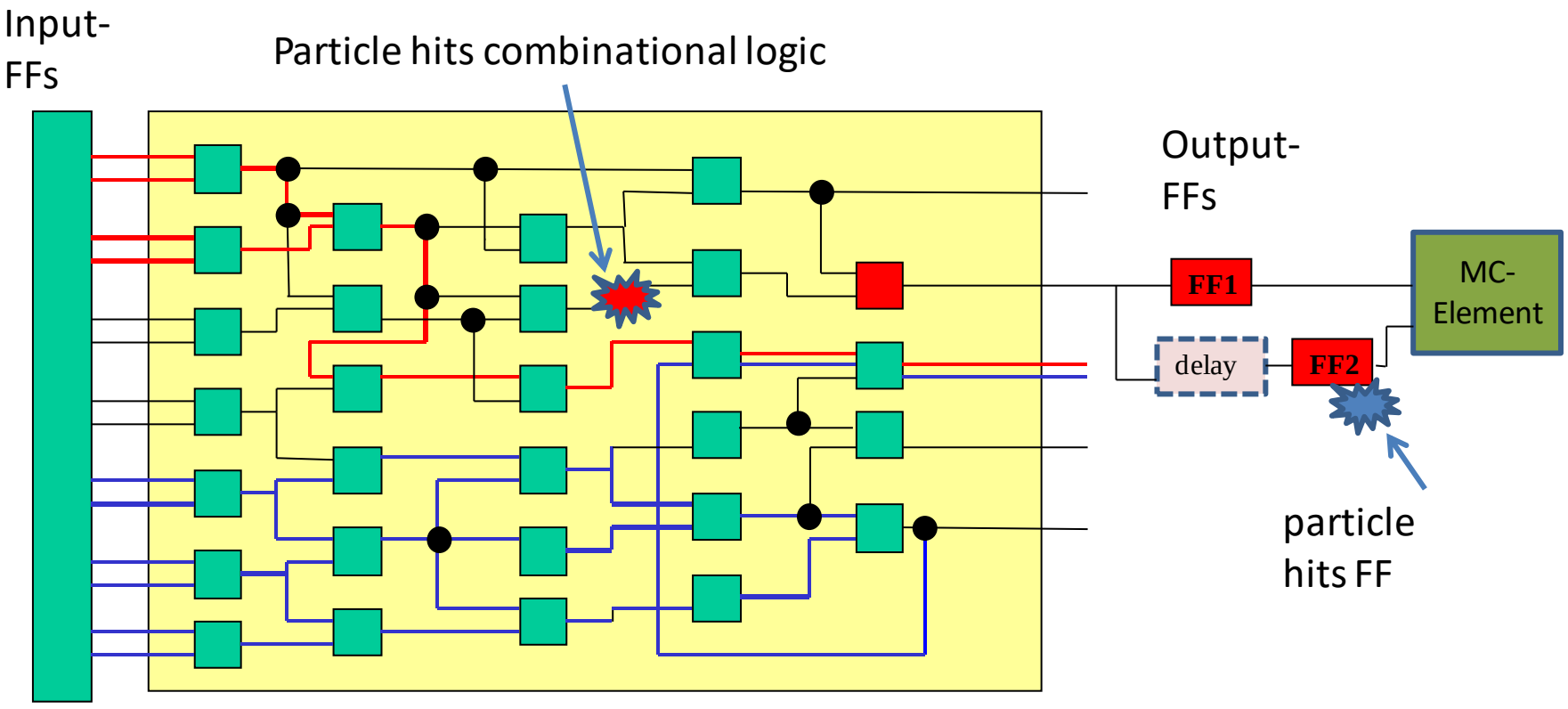
The extra stress by double / triple power may reduce life time !

Watchdog Scheme



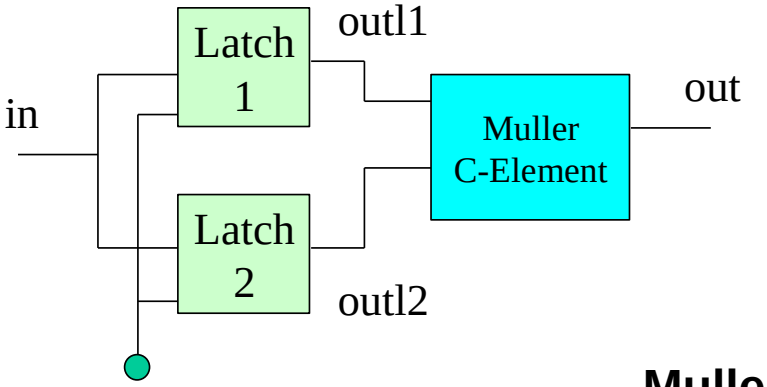
Embedded applications often run in specific program loops. Then specific signatures can be created, for example from memory addresses. Such signatures are then checked by an (independent) watchdog circuit against references about every 1000 clock cycles. If an error occurs ince, the processor gets a „re-start“ signal. If the error then occurs again, the processor is switched off.

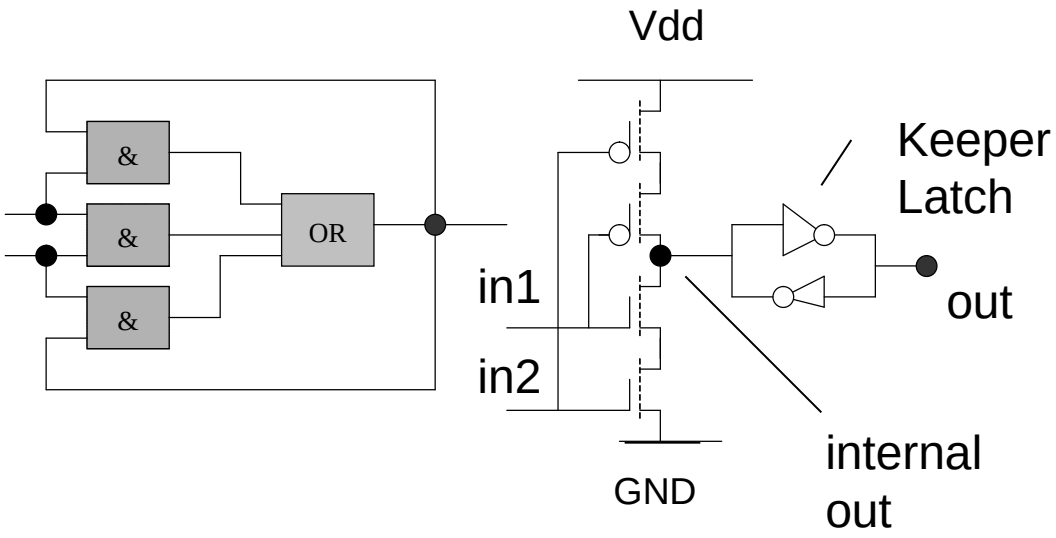
Glitch-Filter



Intel`s flip-flops in processors seem to contain 4 latches for dynamic scan testing in production test anyhow, then the extra cost is negligible !

Fault Detection / Compensation by Muller-C-Element



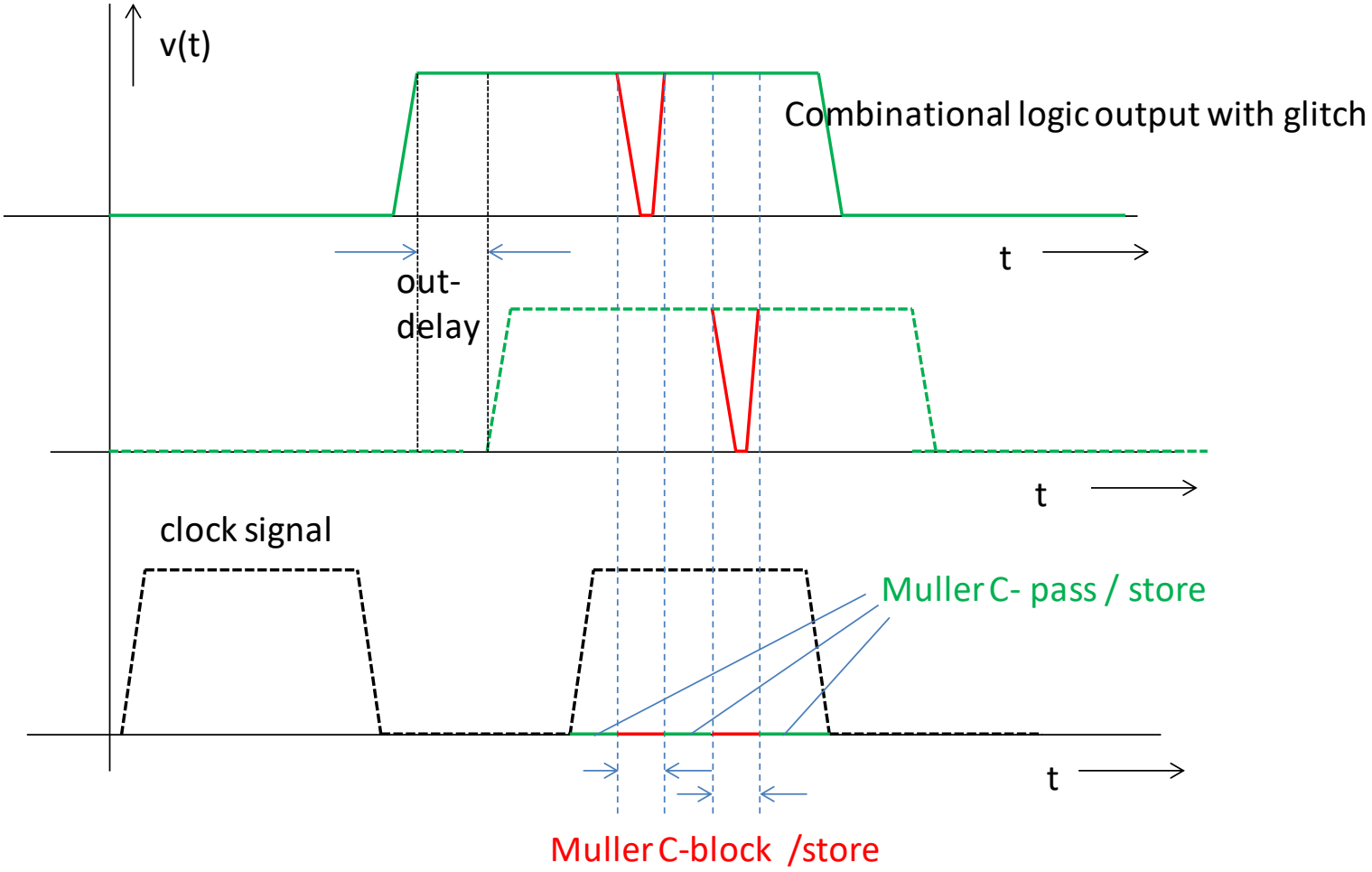


Muller-C-Element:

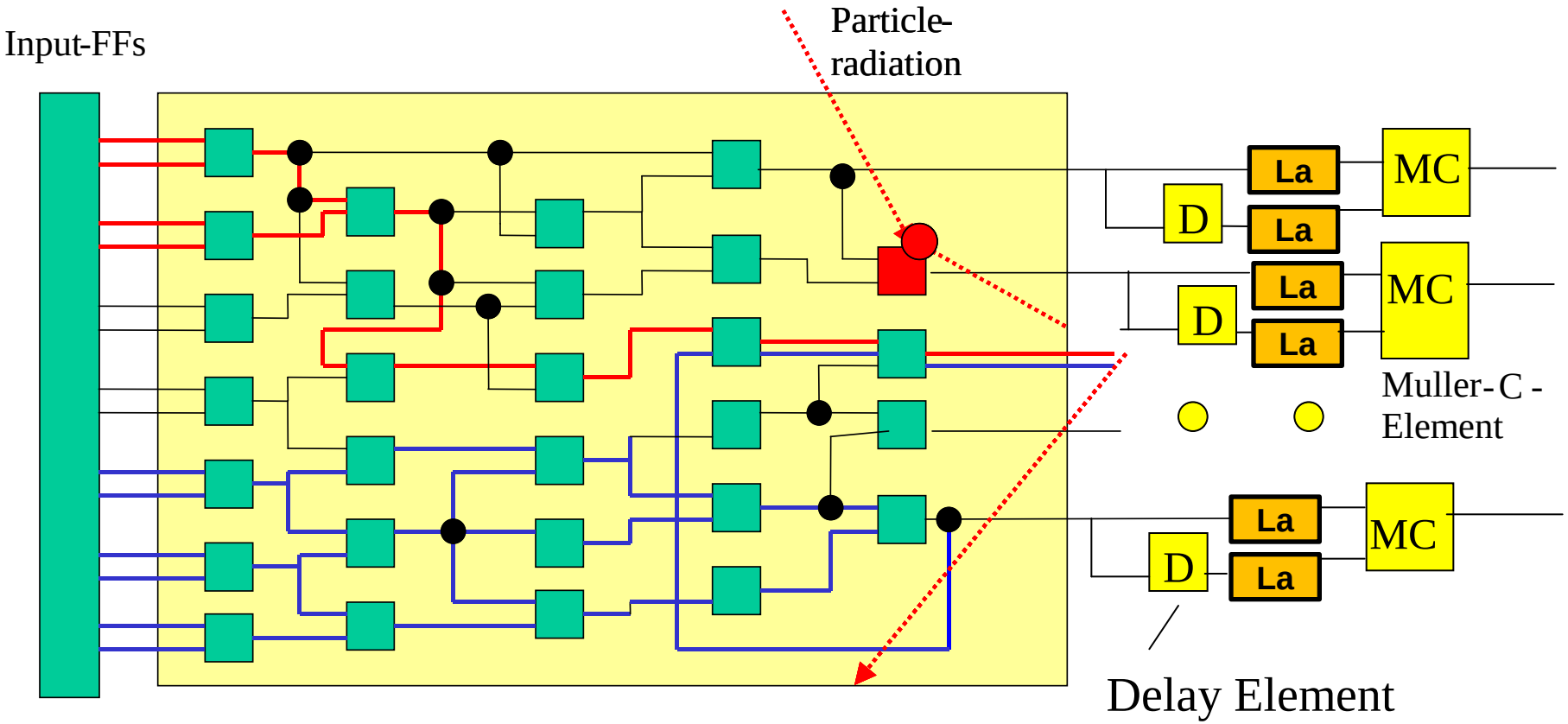
- Both in1, in2 inputs equal:
Input is inverted, switches keeper latch, inverted

-Inputs in1, in2 differ:
Internal out is „isolated“, keeper latch keeps the previous value at out.

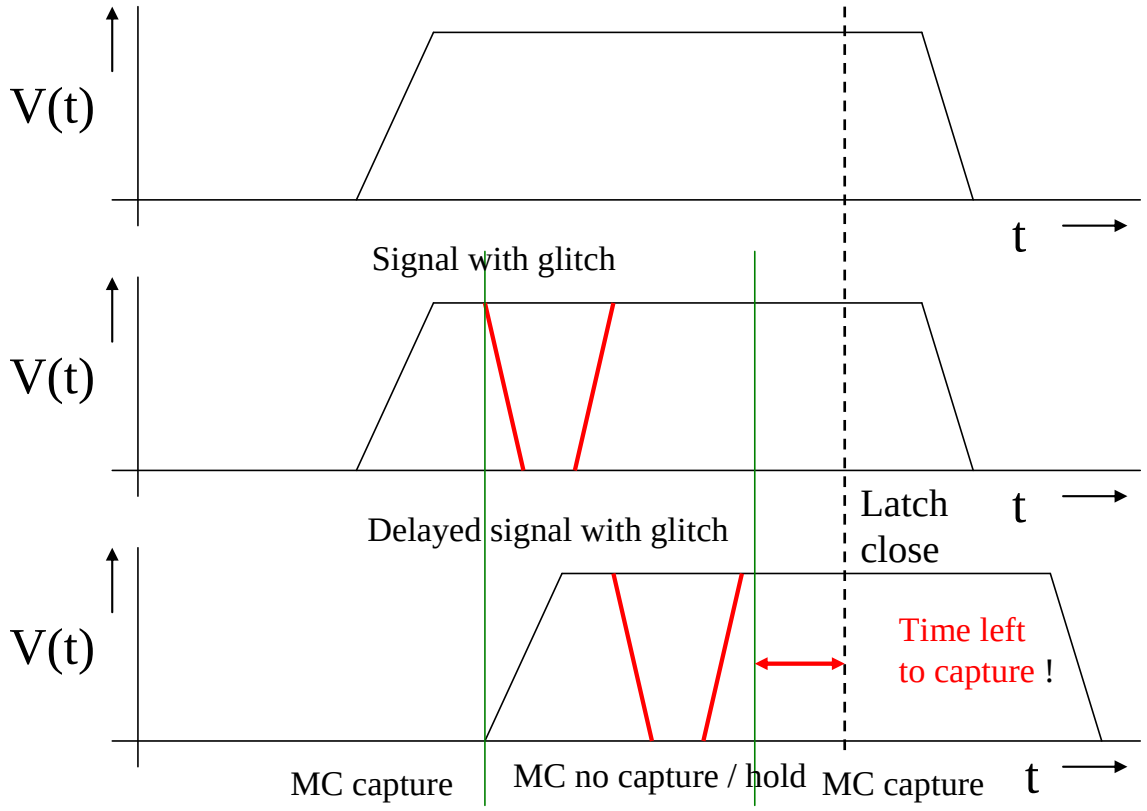
Glitch Suppression by MC-Element



Combinational Glitch Compensation



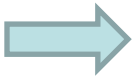
Timing-Diagramm for Glitch-Compensation



Delay Faults

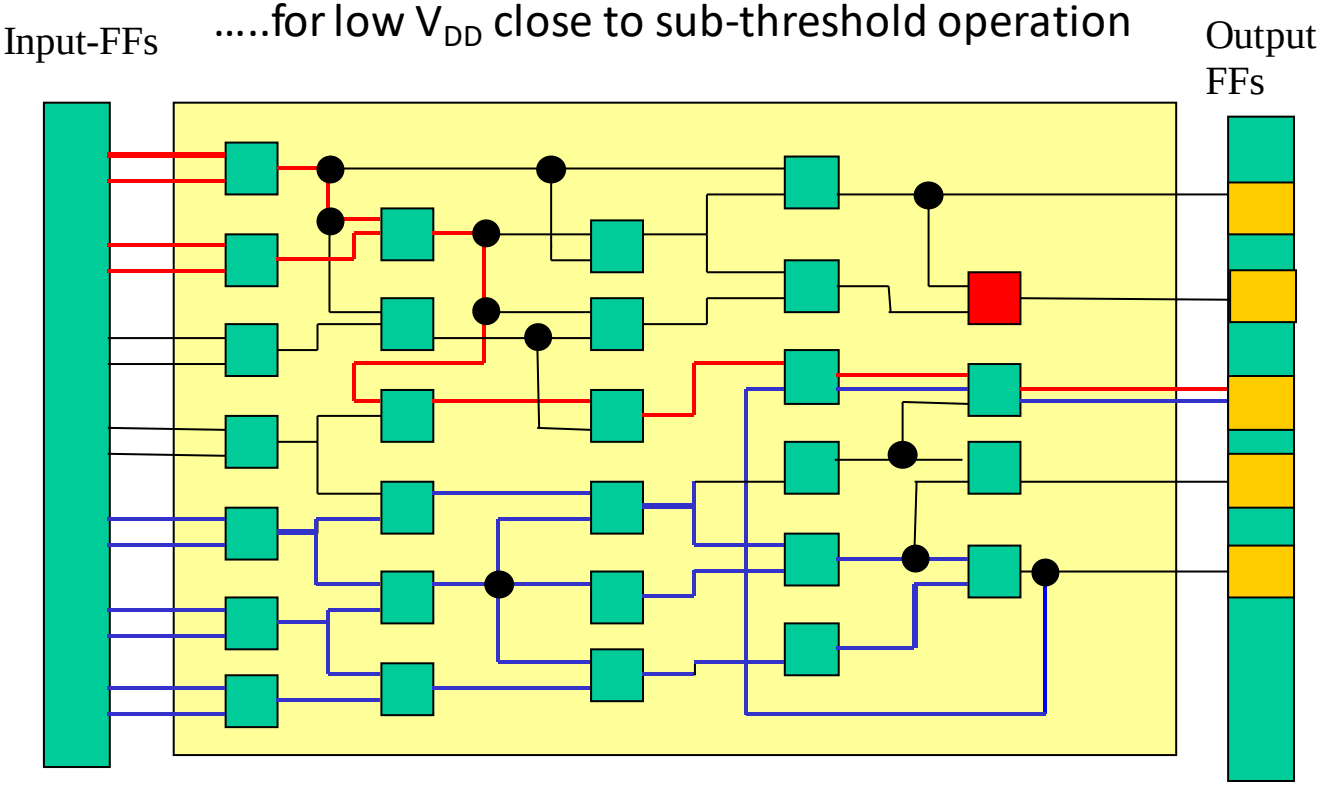
Delay faults due to extra delays on logic paths:
- aging effects on MOS-transistors (higher V_{th}, slower switching),
- instabilities / noise on VDD / GND rails,
- coupling between lines.

Inevitable variations of signal delays even on the same logic paths become much worse at supply voltages below about 0.8 V. Either circuits are then designed with a very high „safety margin“ and low clock rates, or delays on specific „long“ paths are detected and corrected.



RAZOR: Compensation of delay faults for low-VDD circuits,
University of Michigan since about 2004.

Delay Faults

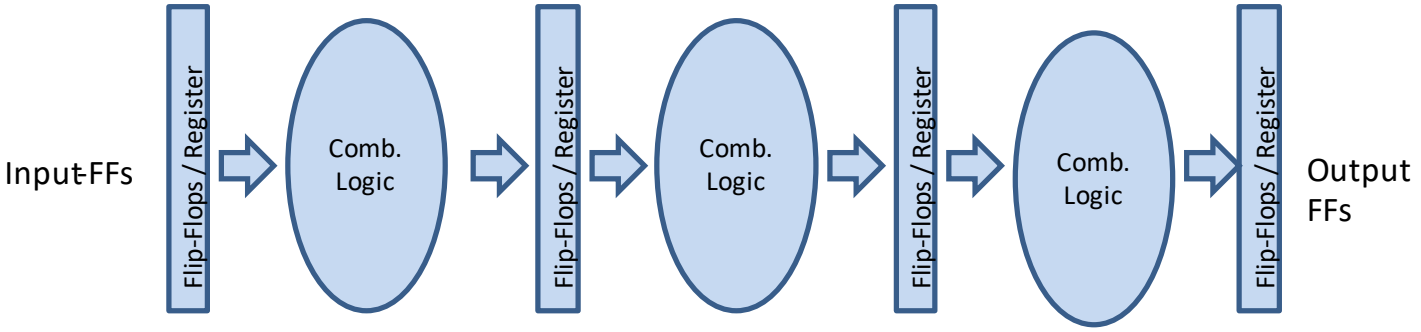


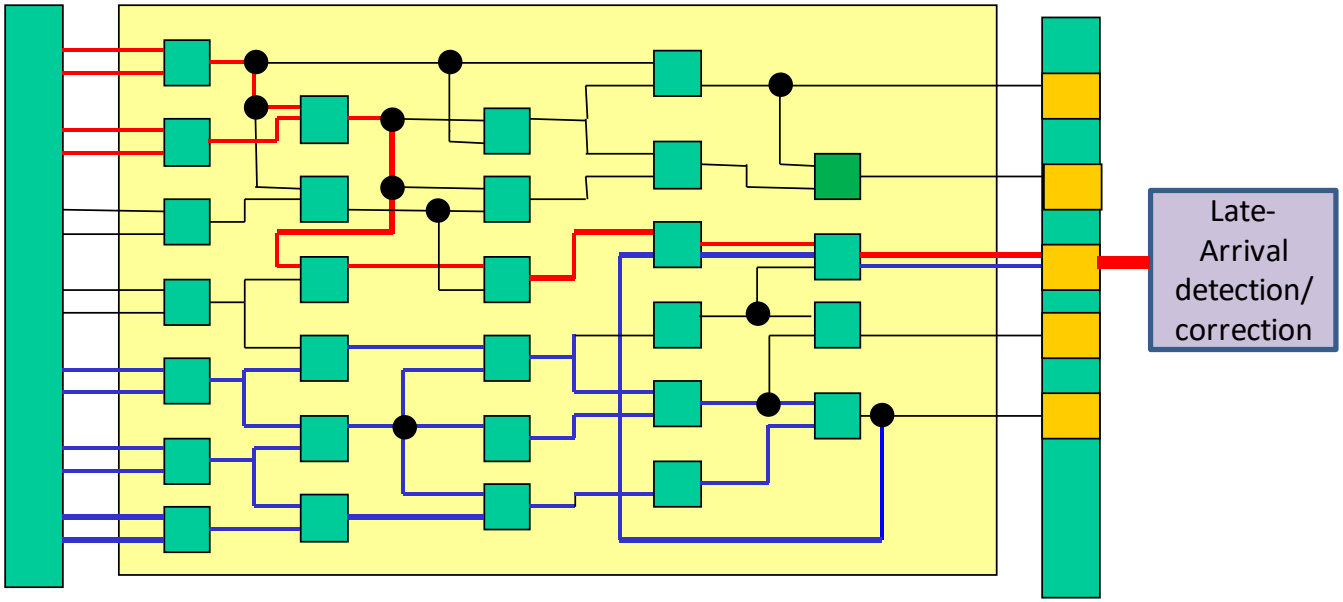
Rock-solid design: The circuit is designed to work even assuming worst-case deterioration and parameter shifts late in its expected time of use. **Costly, inefficient !**

Error correction : The system is designed with much less „reserves“ in parameters , but is additionally equipped with means of detecting and correction errors.

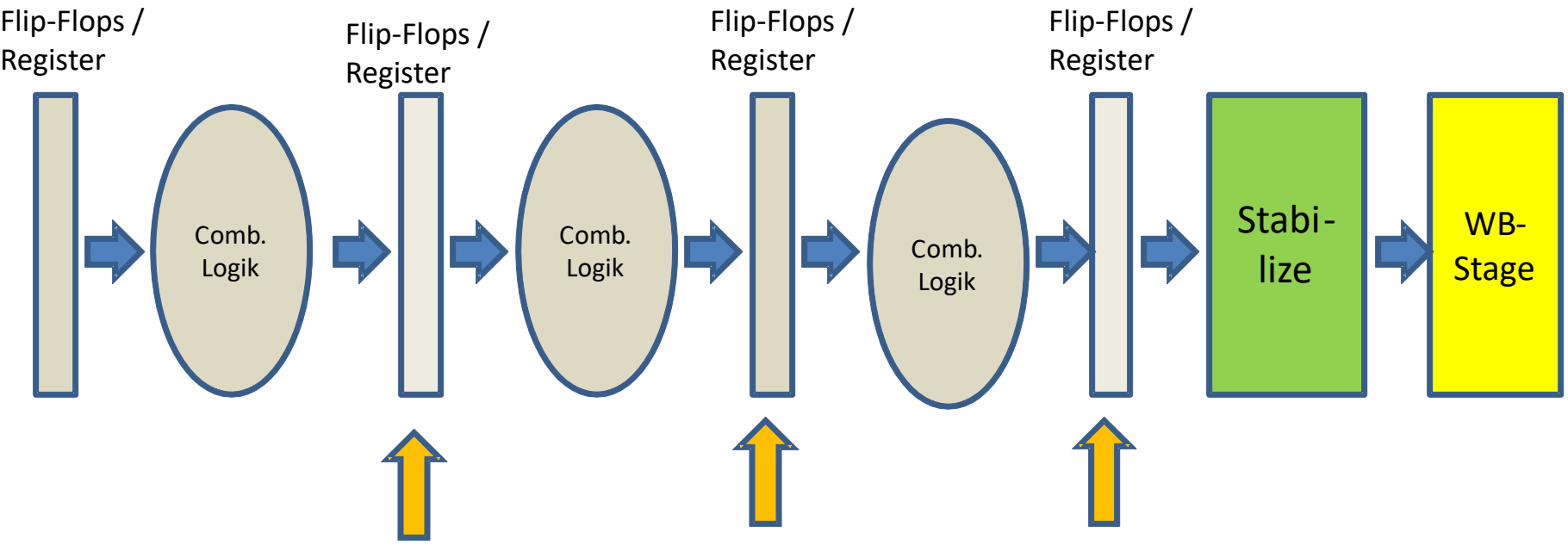
Adaptive system : The (sub-) systems gets capabilities of analyzing its own status and can adapt critical parameters (e. e. VDD, clock) to stay alive and functional .

Pipelined Execution Unit





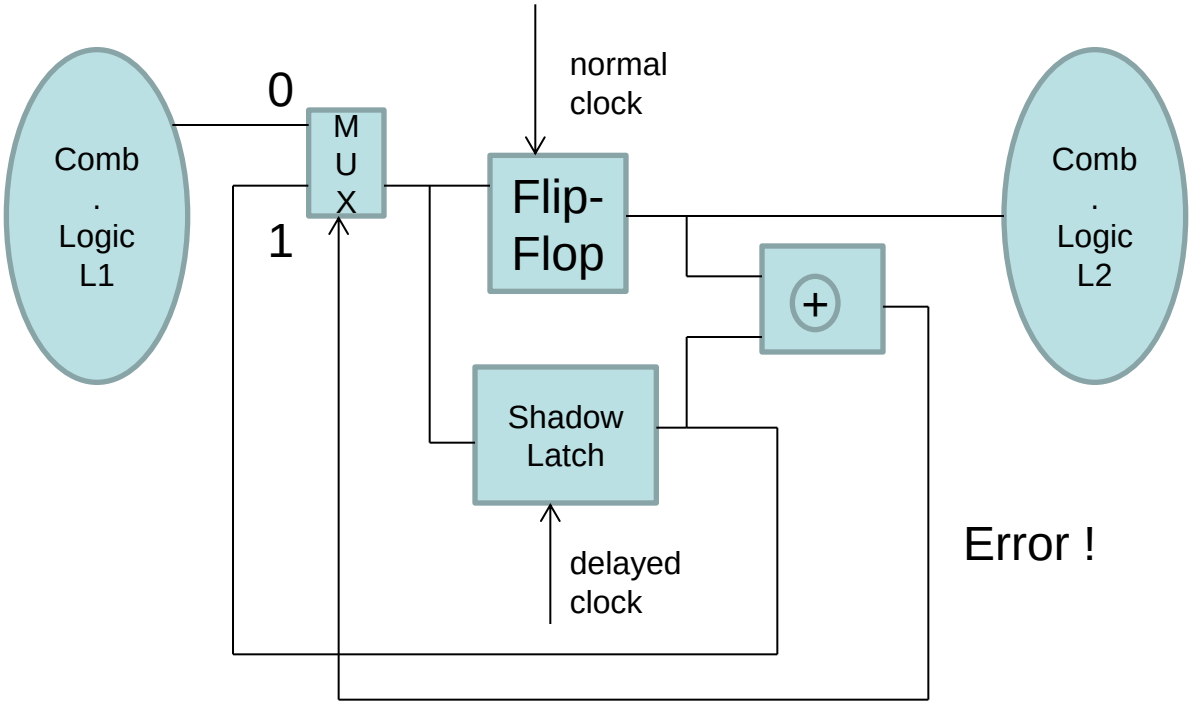
Delay Faults in Pipelines



RAZOR latches /flip-flops

„RAZOR“ technology: Detect and repair late-arrivals of transitions, induced by statistical variations of path delays for low V_{DD} (Univ. of Michigan, 2004).

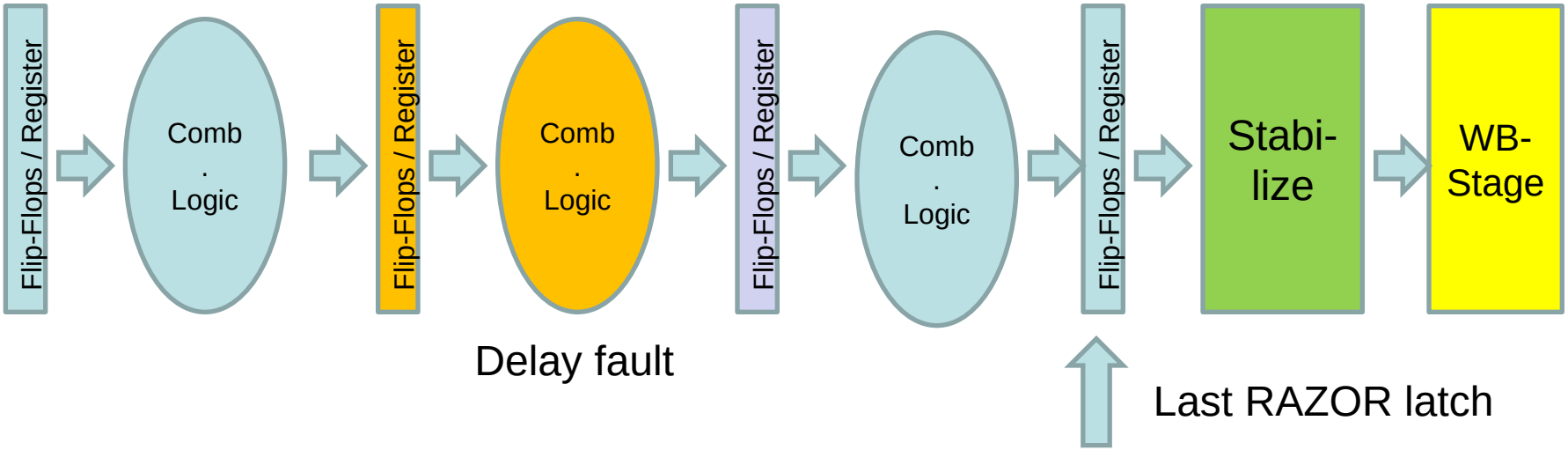
RAZOR-Original Circuit



RAZOR-Idea

- In normal operation (about 99%) the output flip flops receive the logic transition in time and passes it to the next stage.
- If one of the longest combinational paths is activated and VDD and coupling conditions are bad, the transition may arrive too late to be caught by the output flip-flop. The „shadow latch“ is controlled by a delayed clock signal and can catch the delayed transition within a certain „speculation window“. The pipeline is stalled, and the corrected value is generated and applied to the next stage one cycle later.
- **Problems**: The next stage has already received the false value. And, during the „speculation window“, a fast response from the next input may also arrive (short path problem).
- The circuit may show a false „correction“ in case of transient faults !

Pipeline-Organisation



Problems:

- Implementing the „pipeline stall“ function upon „fault detect“ is very time-critical, it requires gated clock trees, which are know to cause test problems,
- Inevitably, the next input pattern is already applied to the affected combinational logic stage during the „speculation window“. Next output change may arrive. That means, an early arrival via short path is impossible to distinguish from a late arrival on a very long path !

Pipeline Error Recovery

- **Gating the global clock**
  After „error detect“ the whole pipeline is stalled for one clock cycle. Meanwhile the affected pipeline stage repairs the false output. Requires extremely difficult timing on gated clock nets !

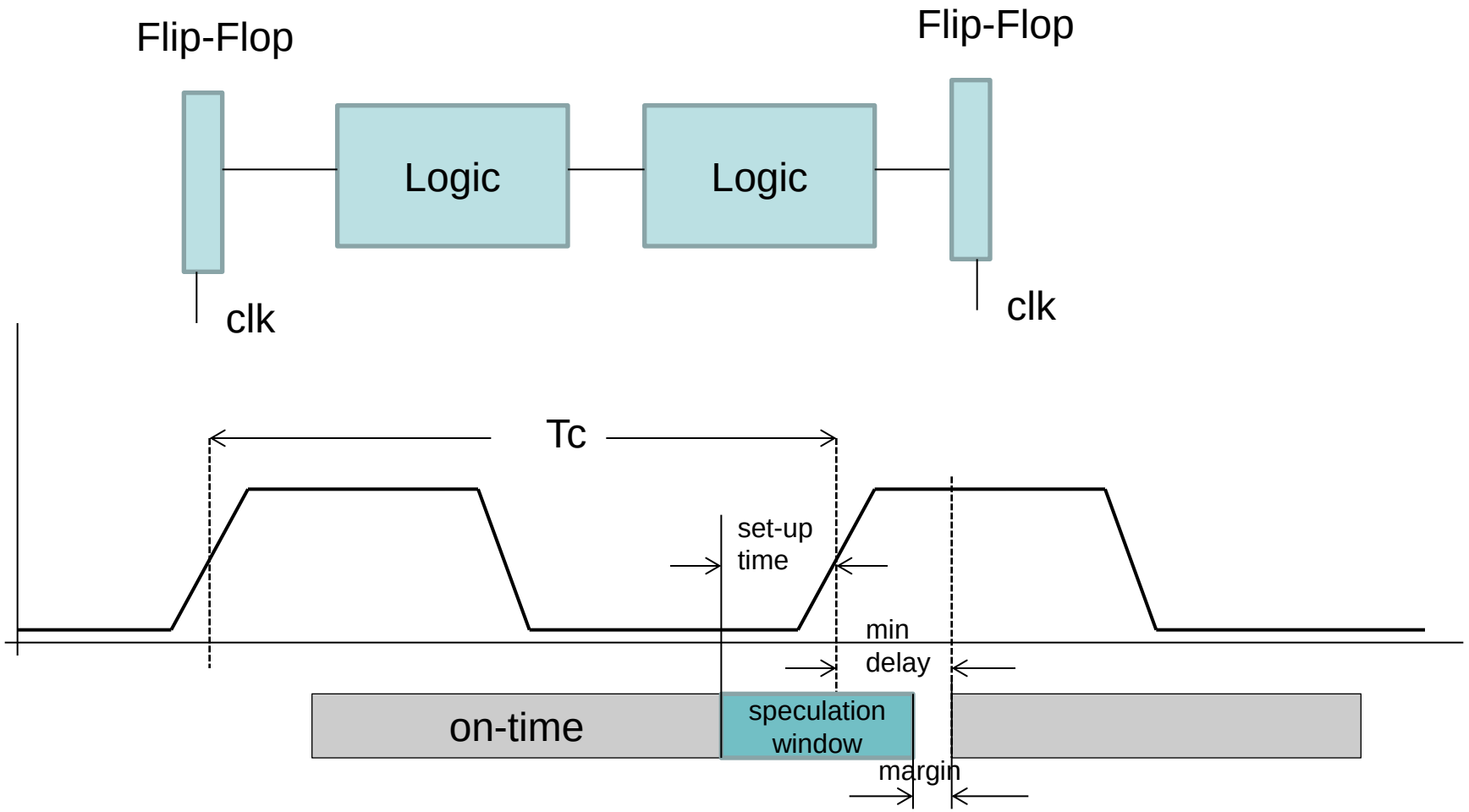
- **Counterflow Pipelining**

  The error signal (ID) generates a „Bubble“ signal for subsequent stages that the actual content /output is void / invalid. The signal is propagated in forward and backward direction from the affected pipeline slot in order to invalidate just the „right“ value. After arrival at the pipeline input, new-start with the next input value.

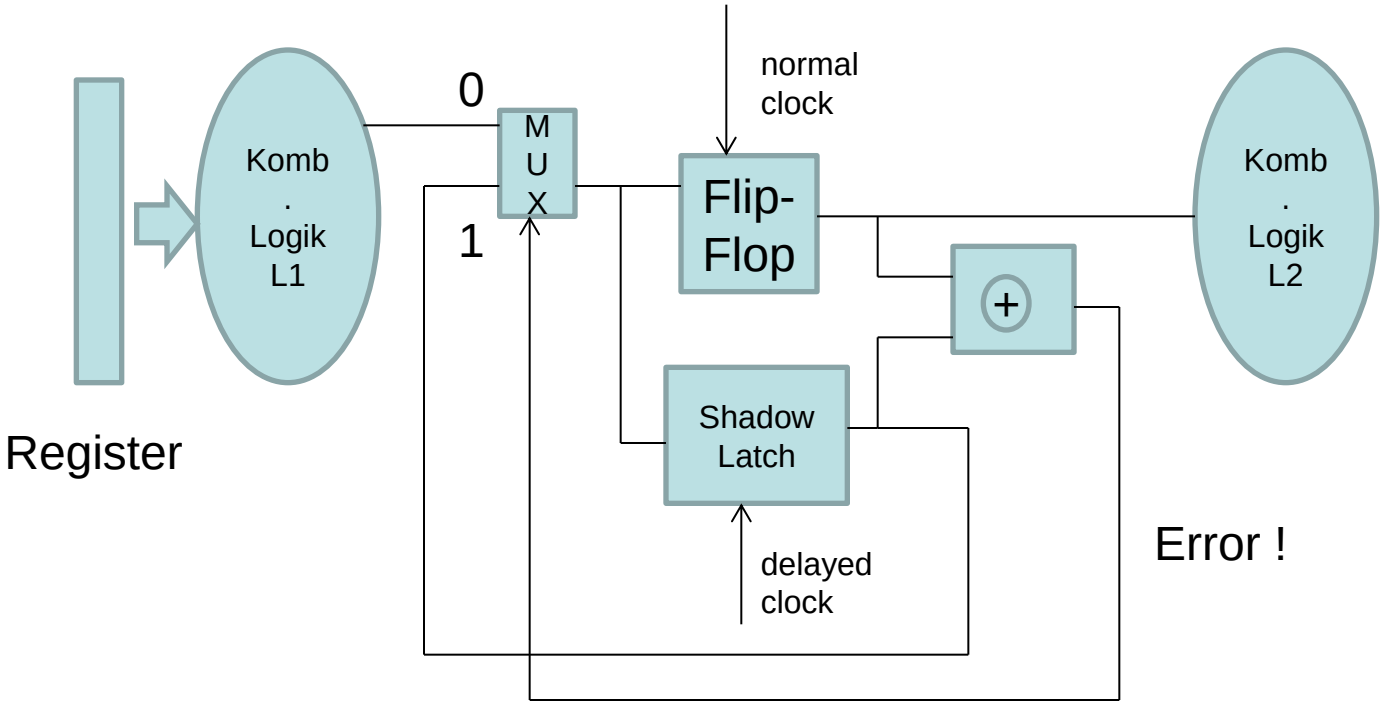
- **Micro-Rollback**

Every register between pipeline stages gets a duplicate register that contains values from one clock cycle earlier. Upon „error detect“, all pipeline stages that feed the affected stage are re-stored by their previous value.

Razor Clocking Scheme

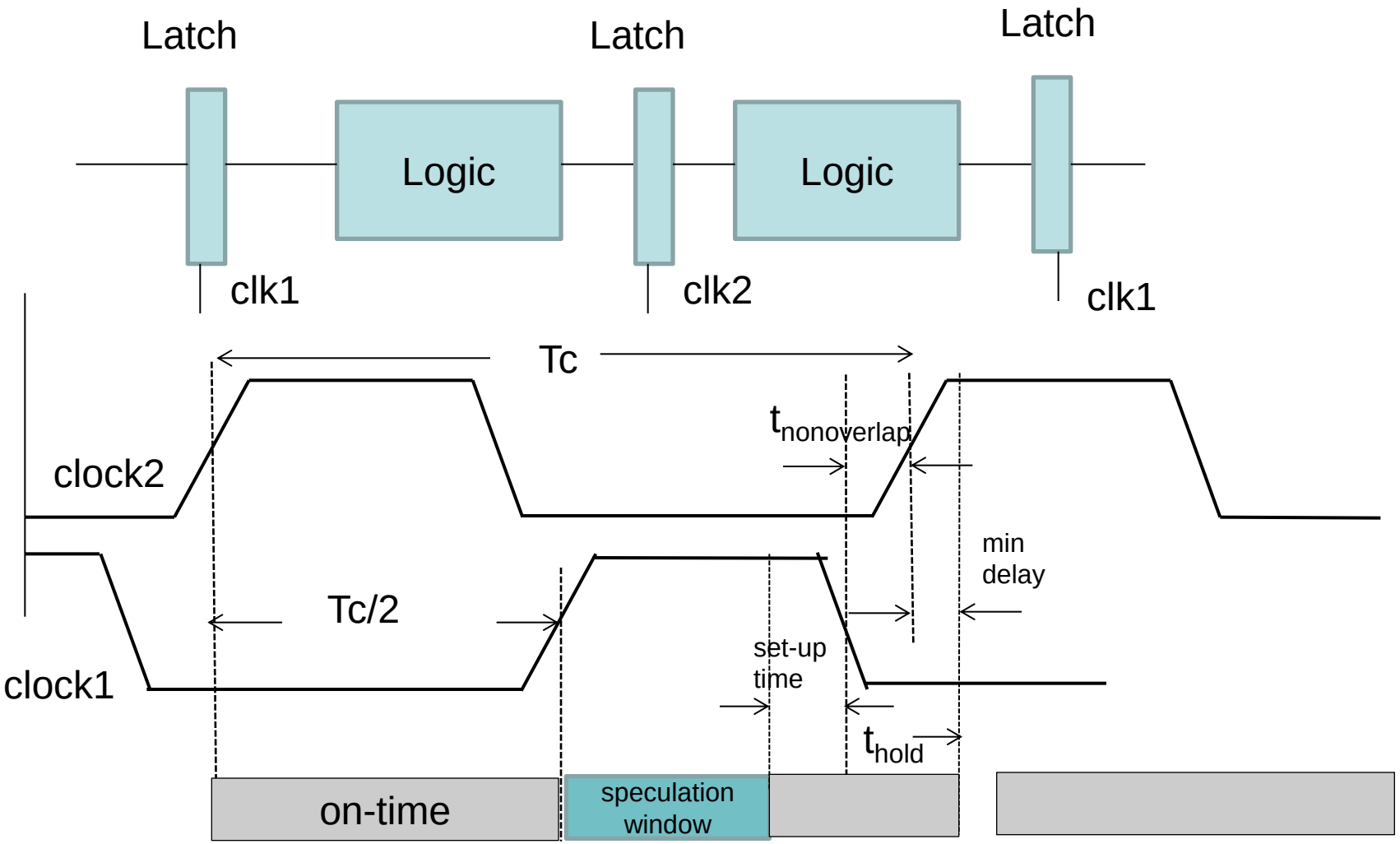


RAZOR-Problem Short Paths

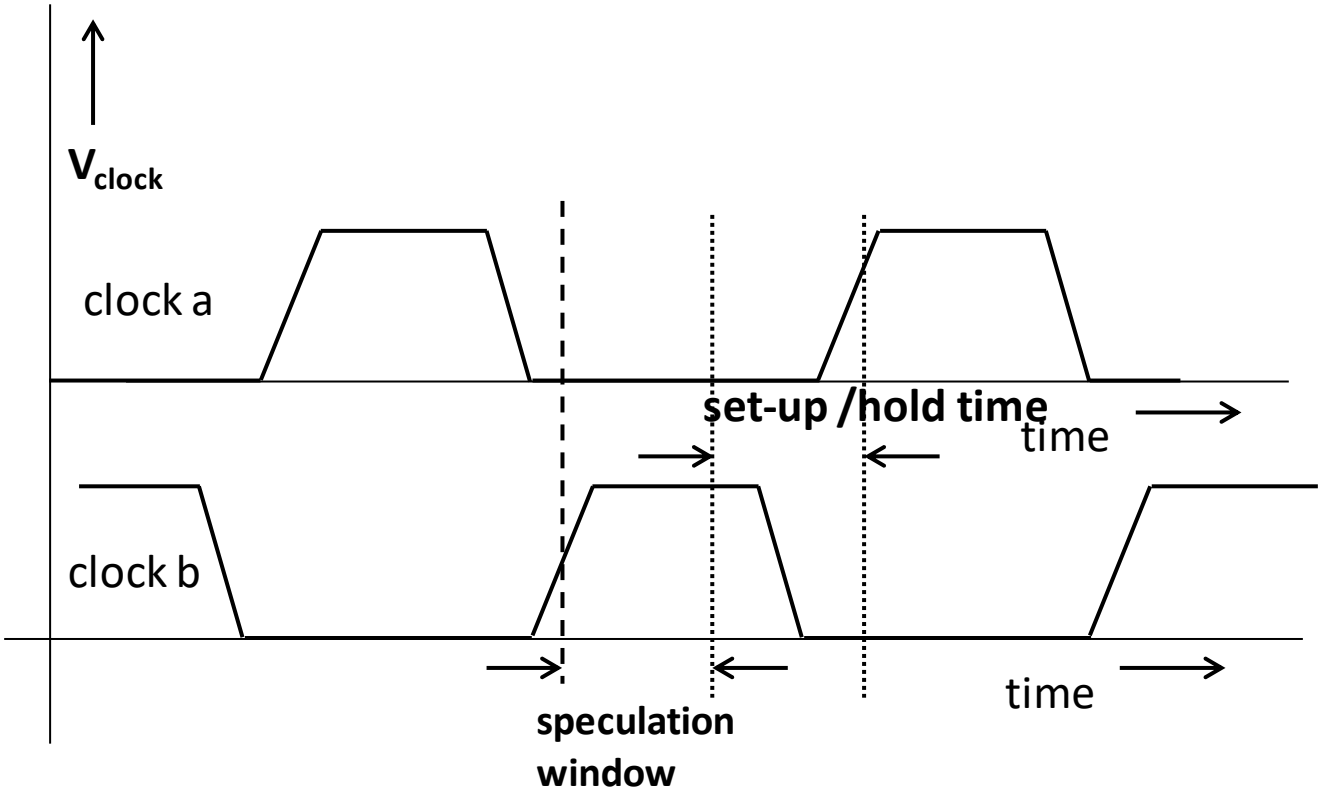


During the waiting time (speculation window) for a possible delayed transition on the shadow latch, the fast response for the nex input signal at L1 may arrive on a short path. This may trigger a false repair !

Bubble Razor

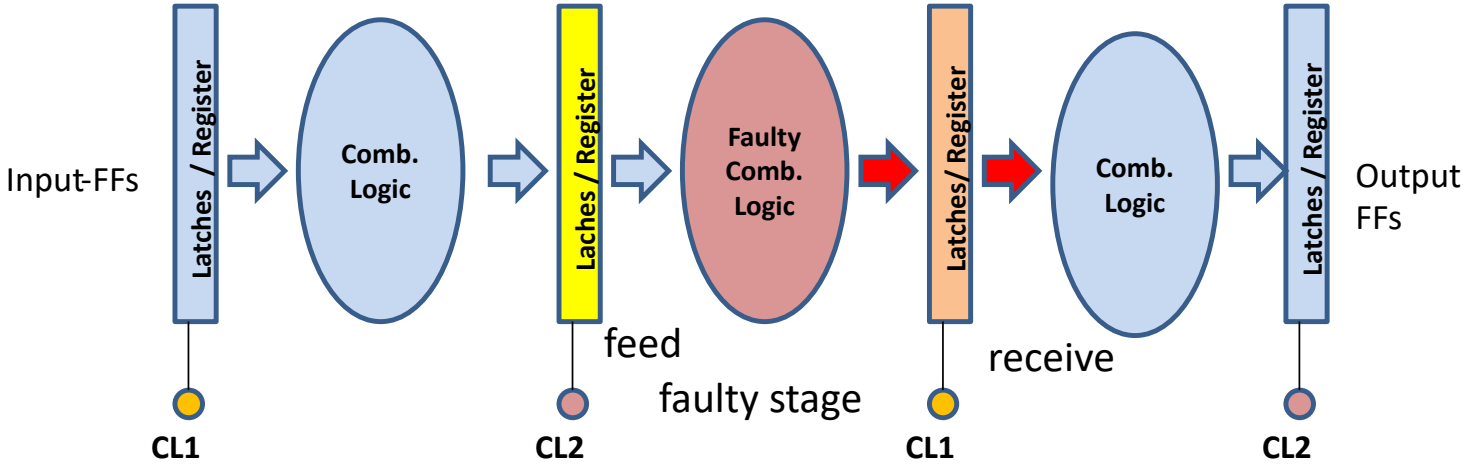


Bubble-Razor: non-overlapping double clocks

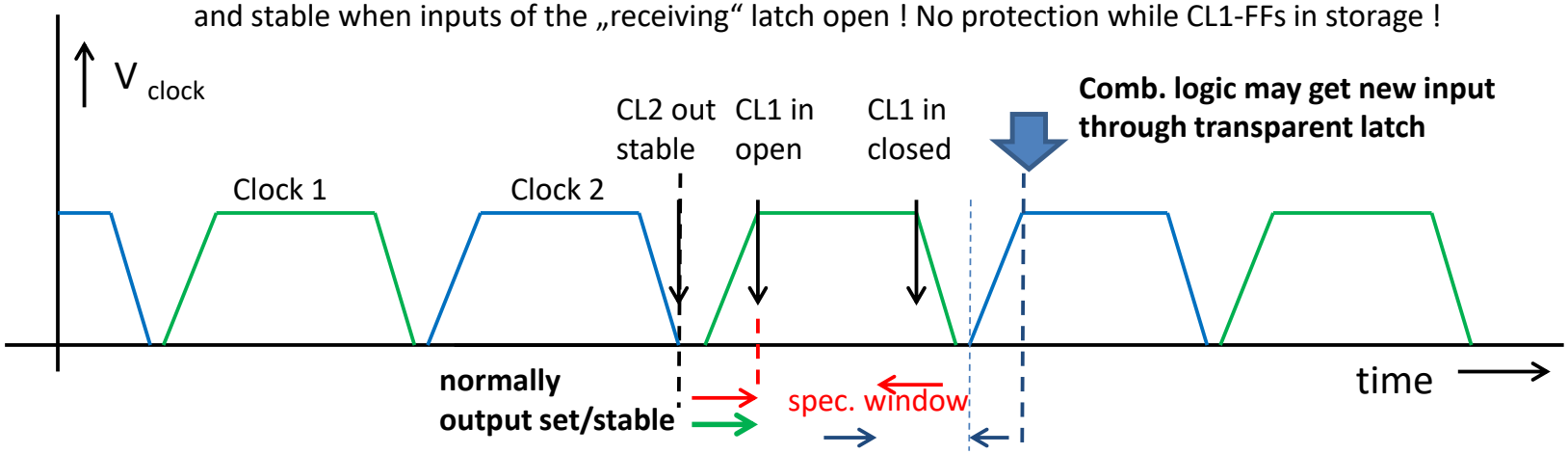


The 2-clock latch-based clocking scheme provides an adequate time space (speculation window) for fault detection. By stalling the pipeline the same input can be kept applied for another clock period.
This is enough time to cover delay faults and (most) transient faults by re-execution.

Error Correction by Bubble Razor



With 2 non-overlapping clocks, the scheme now shows a larger time window for „late detection“ before the inputs of the faulty stage get new values. In normal operation, the functional inputsto the register behind the „faulty“ stage are are set when the latches open. System timing is adjusted such that outputs from combinational logic are set and stable when inputs of the „receiving“ latch open ! No protection while CL1-FFs in storage !



Error Correction by Bubble Razor

- Delays are recognized in time before input changes.
- After a „late arrival“-detect, the pipeline is „stalled“ in a special process, and the application of the next input pattern to the „late stage“ is postponed.
- The output takes the „right“ value (with the original input pattern) and can be propagated to successive pipeline stages one clock cycle later.
- If a „transient“ fault occurs in a flip-flop during the „speculation window“, it is recognized as a late arrival. The circuit is then given extra time to restore the right output value by just keeping the previous input applied.

⇨ Bubble Razor may correct delay faults and transient faults (in the flip-flops) in combination !

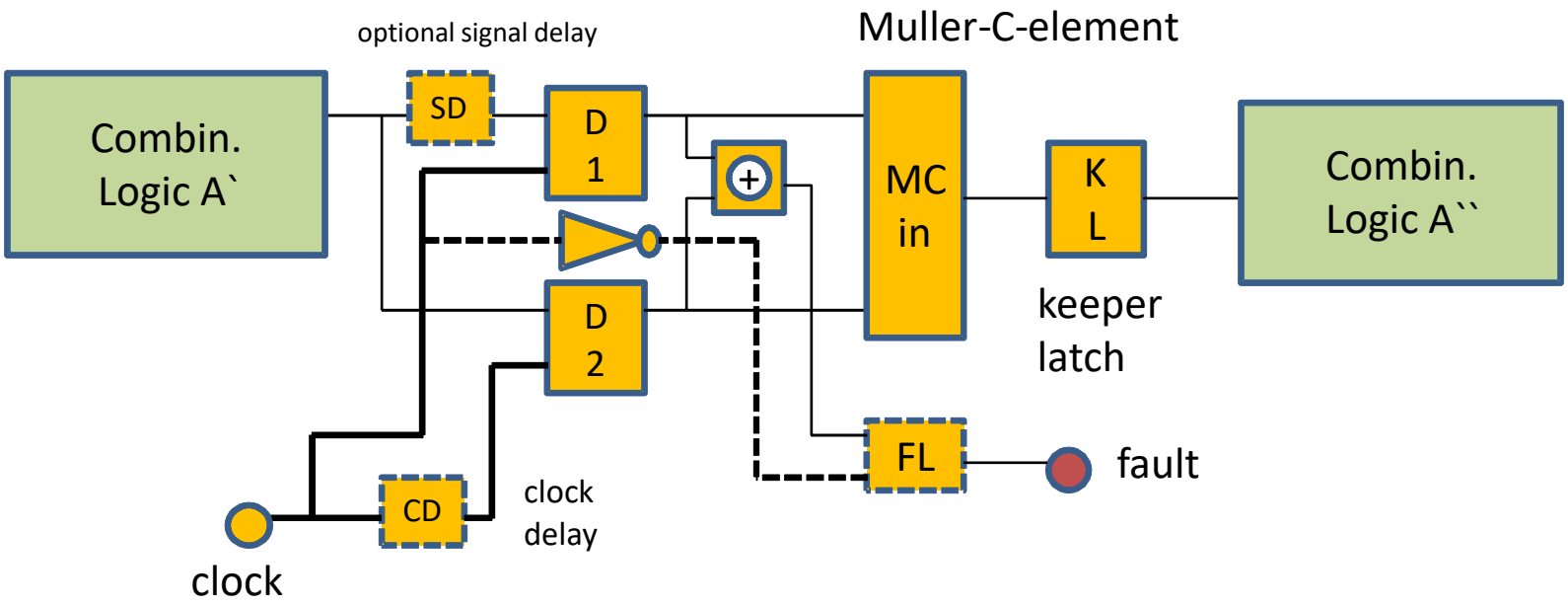
Bubble Razor-Effects

ARM Cortex-M3
processor in 45 nm technology

- Area overhead 10-40 %
- Clock frequency may be about doubled
- Supply voltage can be reduced from 1V to about 0.75 V
- No energy savings if Vdd goes below 0.75 V, since the high number of corrections costs extra energy

Critical parameters: - the time window set for recovery
- the share of latches equipped with error detection

Error Correction by Re-Execution



By using a clock and a delayed clock, a time window for „speculation“ is defined.
Any change of inputs that occurs during this time is taken as a „fault“.
Correction by re-execution, while the same input keeps applied.

Problem: *Works only for delays / transitions during the „speculation window“ !*
Large overhead if applied to all outputs, timing control including clock-gating is costly and may even cause test problems !

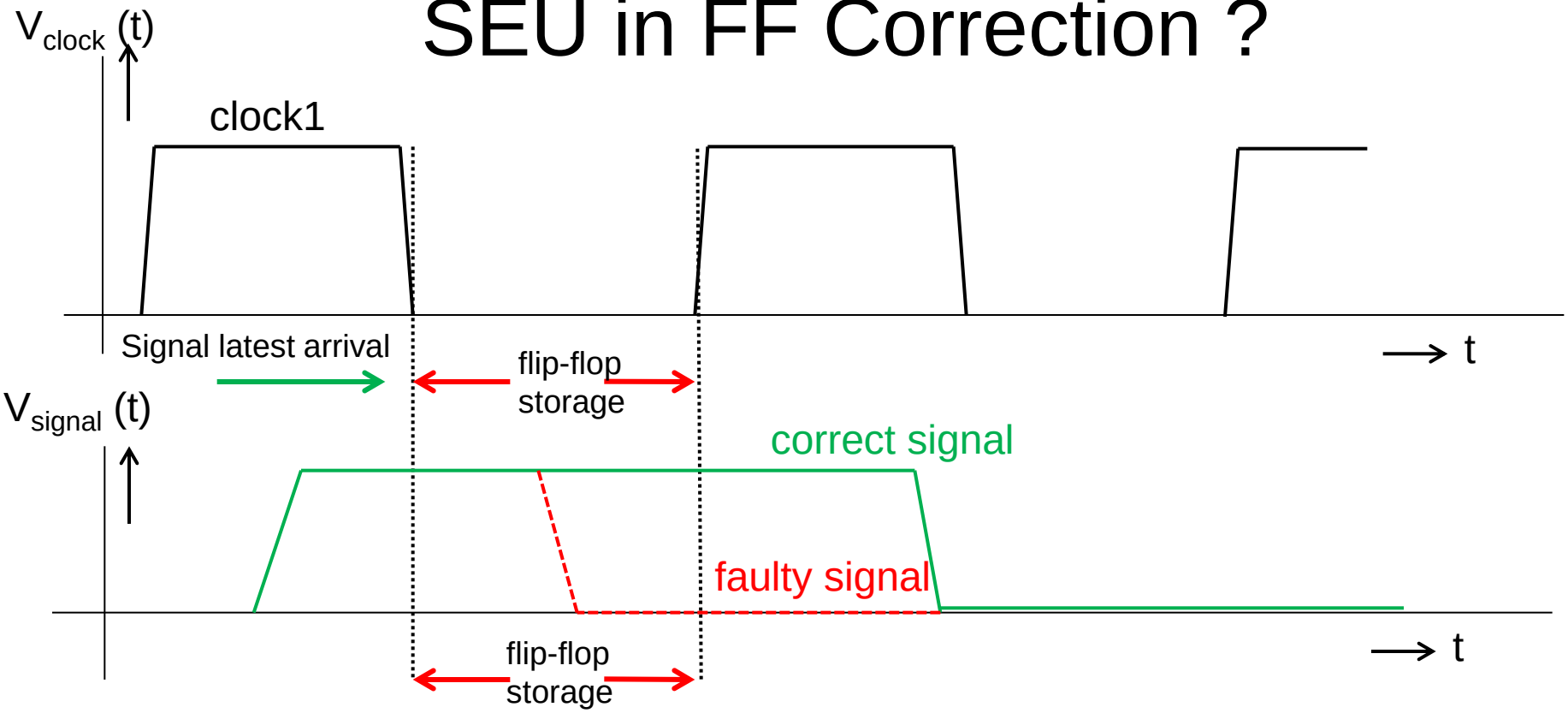
Overhead: 30 transistors versus 8 for every output, plus pipeline stall management !

Delays and Transient Faults

- Units designed for delay fault correction will not automatically detect or even correct transient faults.
- Delay fault detection / correction requires pipeline stall- and extra clock control.
- So far, delay fault detection / correction has been applied to known „longest“ paths only because of relatively high cost.
- Advanced „Bubble Razor“ concepts are known which can detect / correct permanent and transient faults by different mechanisms.
- … but if applied to all outputs for complete coverage, the overhead in hardware may be above 100 % !!

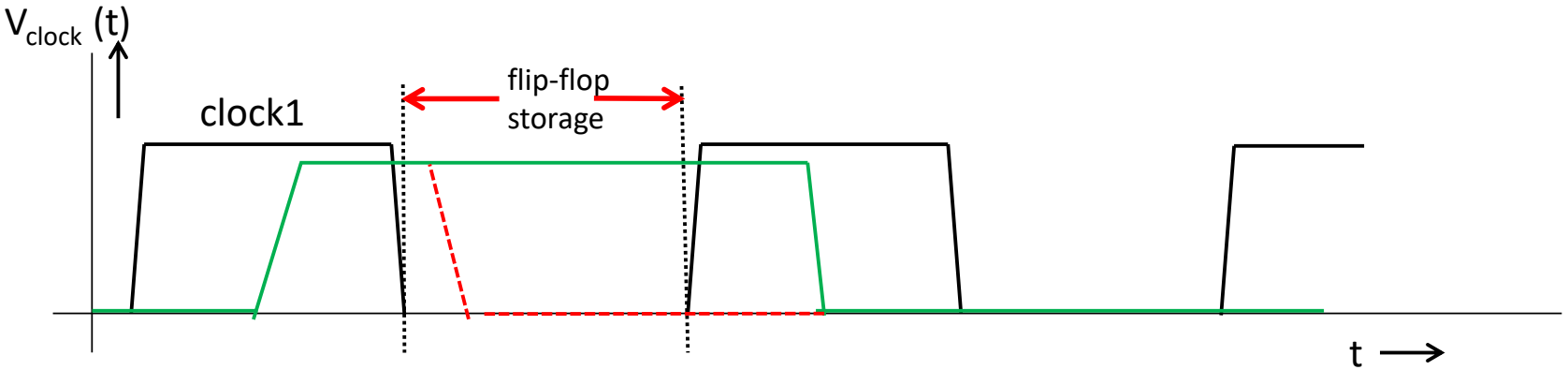
Essential question: Can we combine the detection / correction of delay faults and transient faults without high extra cost in power / energy ?

SEU in FF Correction ?

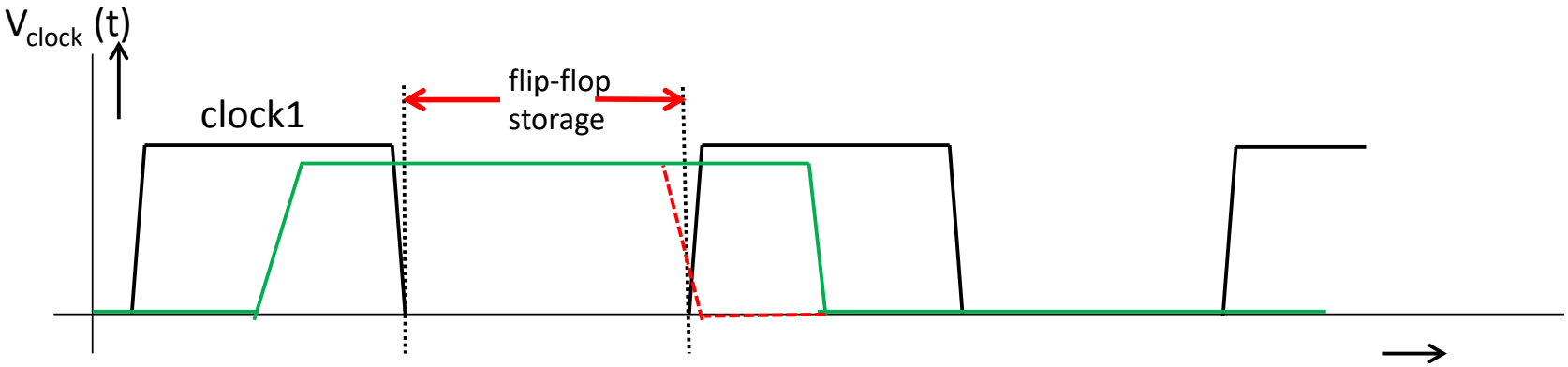


The scheme for correcting SEUs and SETs also uses double flip-flops and a Muller-C-element at the output. The Muller C-element takes the value that both FFs produce during the storage phase as the correct one. If outputs differ, a previously stored value is taken from an extra „keeper latch“. No explicit „fault detect“ signal produced.

SEU Detection Limits

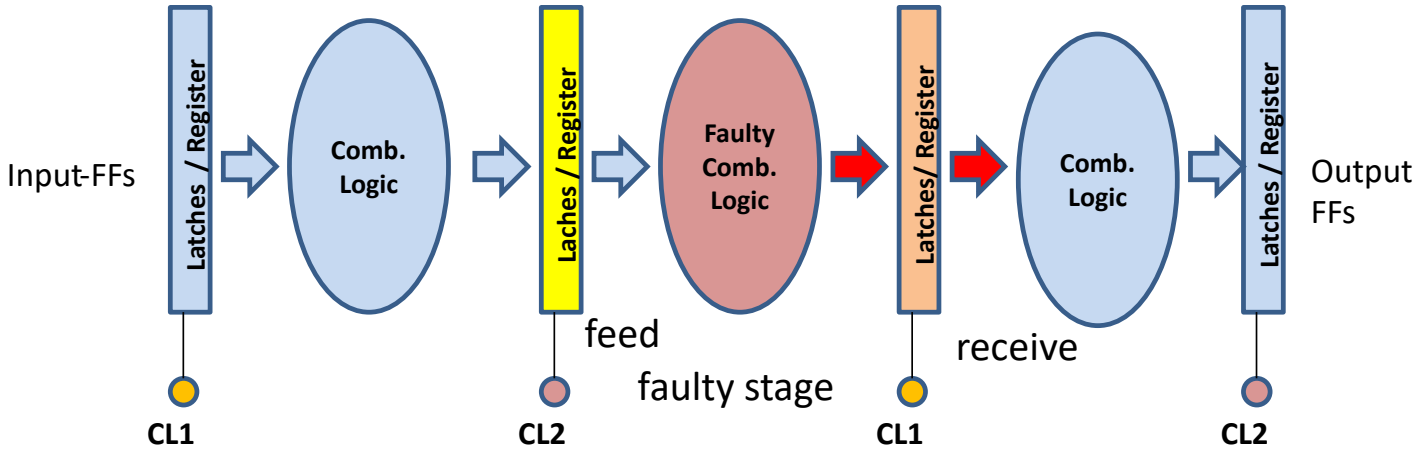


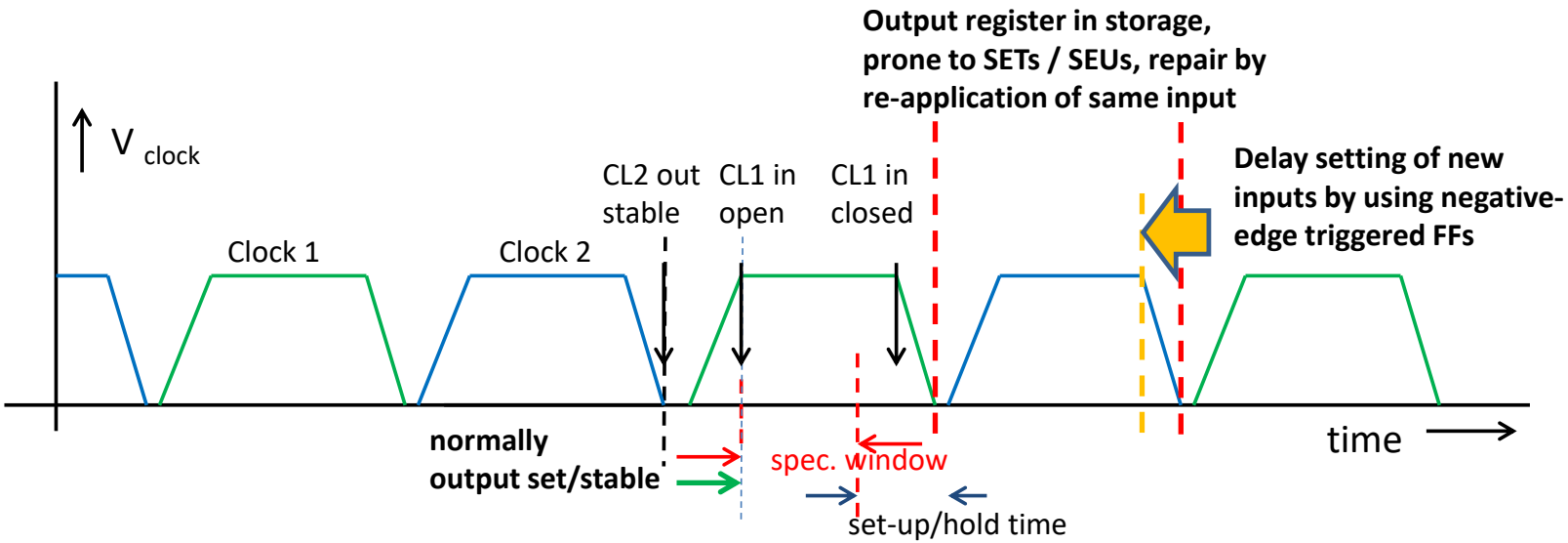
Signal change early in the FF storage phase, such as in case of delay faults. Possibly too little time to capture the correct value by a Muller C- element. But enough time to trigger „fault detect“ by comparison, stall and re-execution.



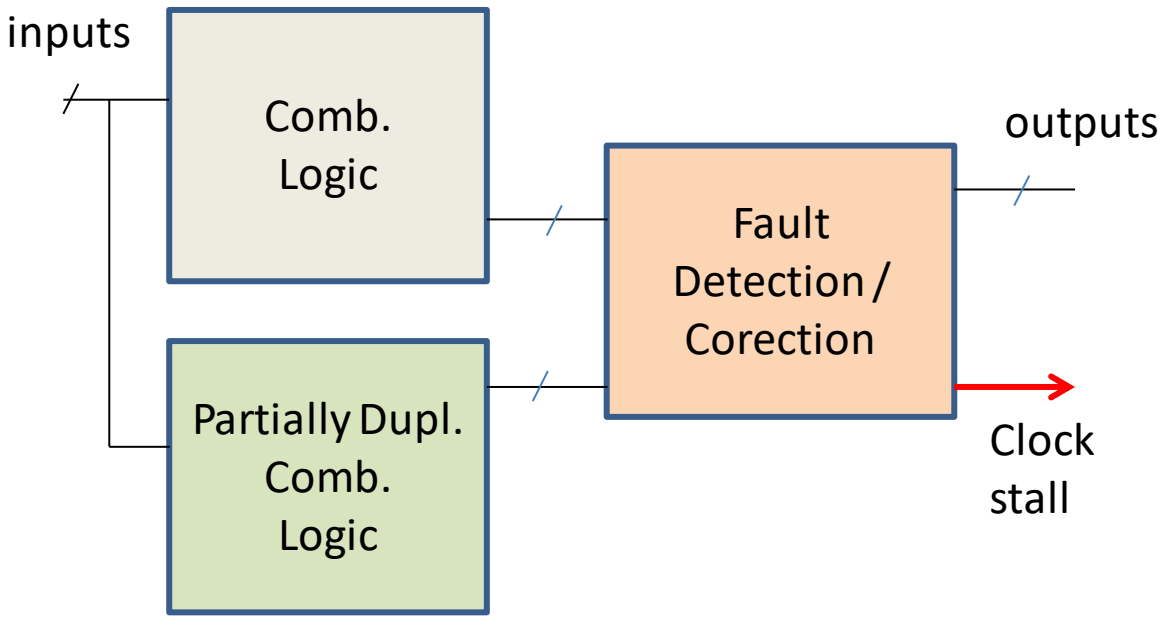
Signal change late in the storage phase. Possibly too little time to trigger „fault detect“ by comparison, stall and re-execution, but enough time for the MC-element to store the correct value.

Bubble Razor for SEU Correction?





Long Delay Faults in Comb. Logic



The scheme can detect and correct delay faults in the combinational logic of arbitrary length. Clock-stall and time-management required. Overhead below but close to triple modular redundancy !

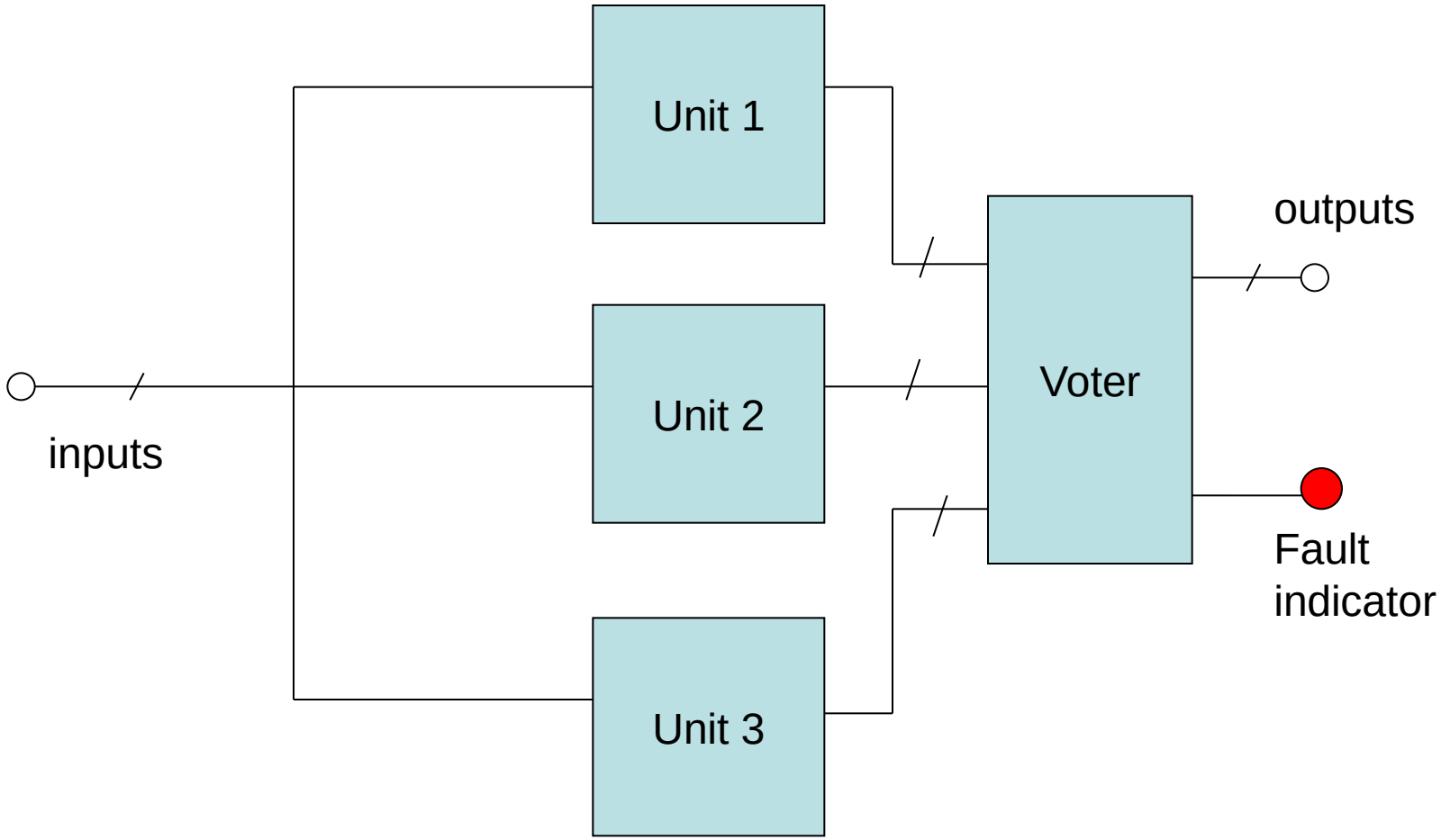
Transient Fault Summary

	Correctable	extra effort	extra clock control	extra power
SETs in logic	most *	moderate*	no	little
SEUs in logic	hardly**	high**	yes	little
SETs in FFs	most*	moderate*	no	little
SEUs in FFs	most *	moderate *	no	little
Delay fault < < T	yes	high **	yes	little
Delay fault > T /2	yes	very high***	yes	about double

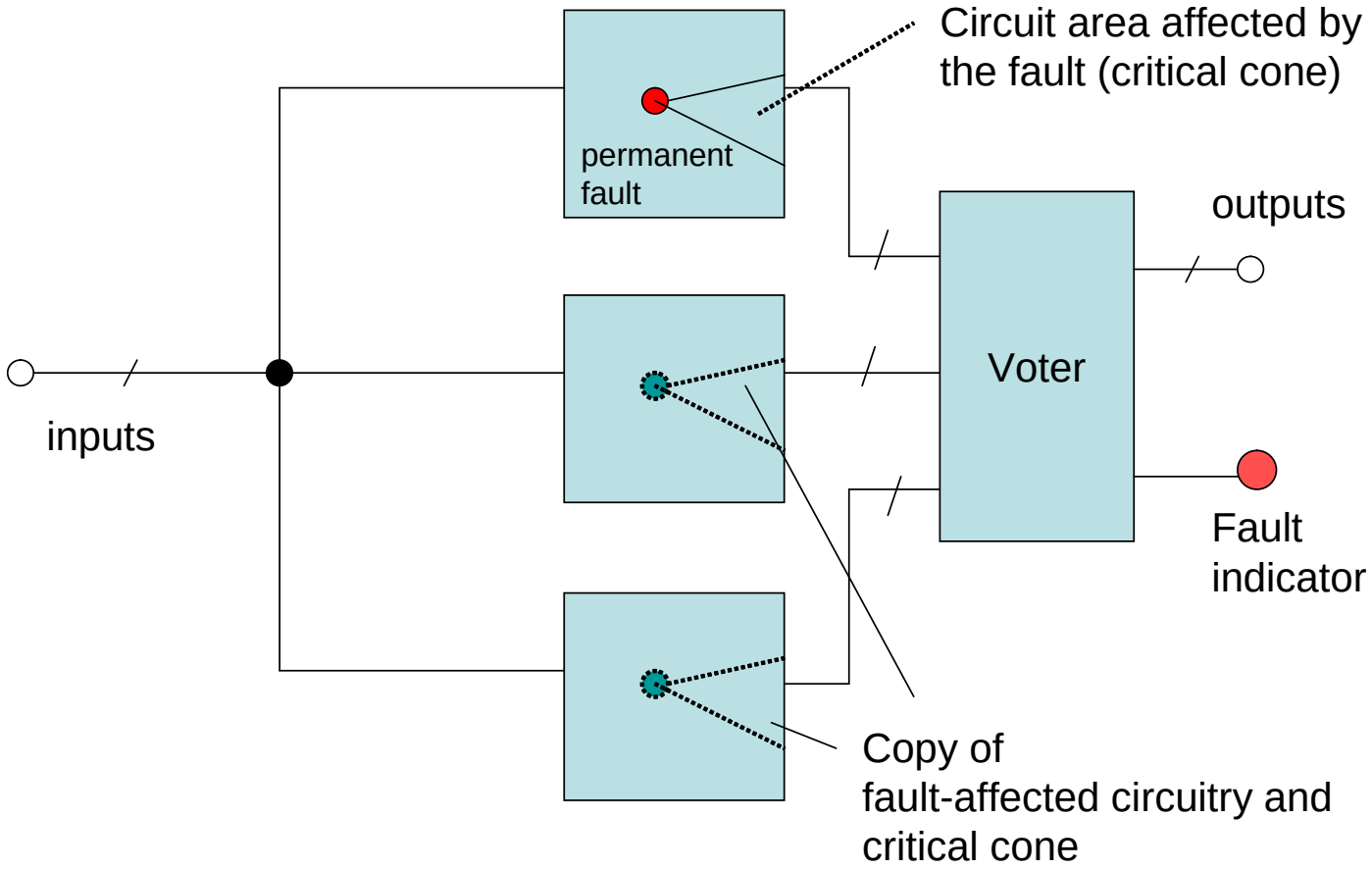
* Intel-Mitra scheme
** Bubble Razor
*** Gössel / Krstic

„Short“ delay faults are cheap to detect and repair by simple repetition of the act. Methods become costly where correction by repetition does not work !

Triple Modular Redundancy (TMR)

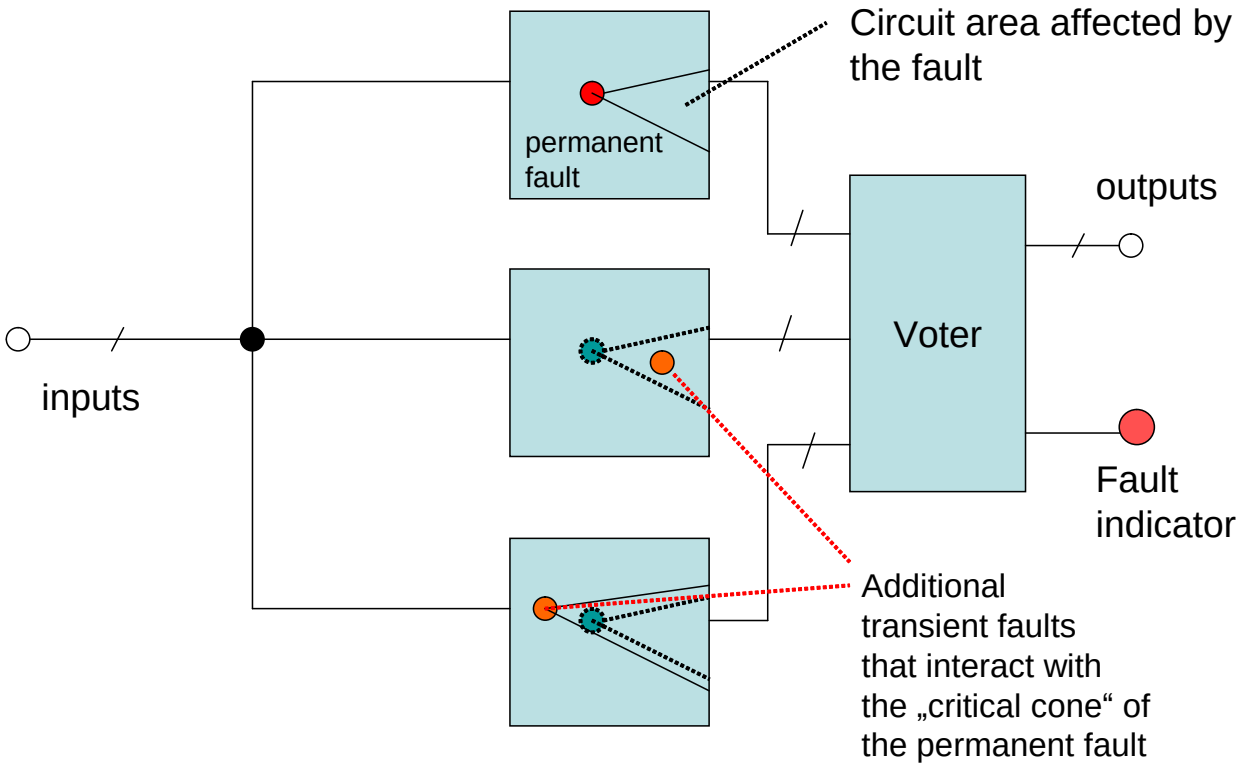


TMR with Faults



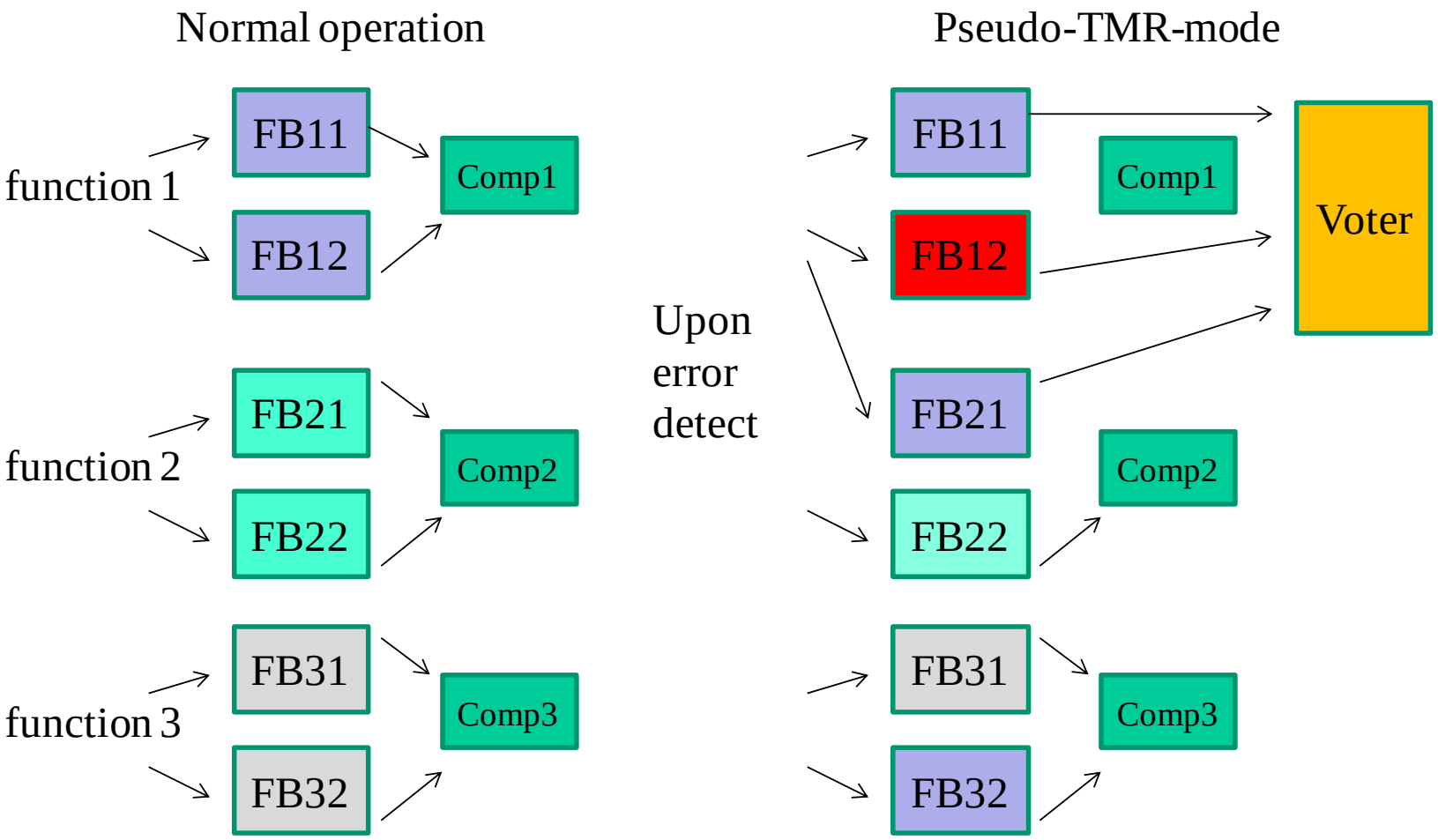
TMR may only go wrong, if two or more faults occur that are „related“ by influencing the same outputs ! In case of an existing permanent fault, an additional transient fault in the „affected“ zone of the parallel units may trigger a false correction !

TMR and Multiple Faults



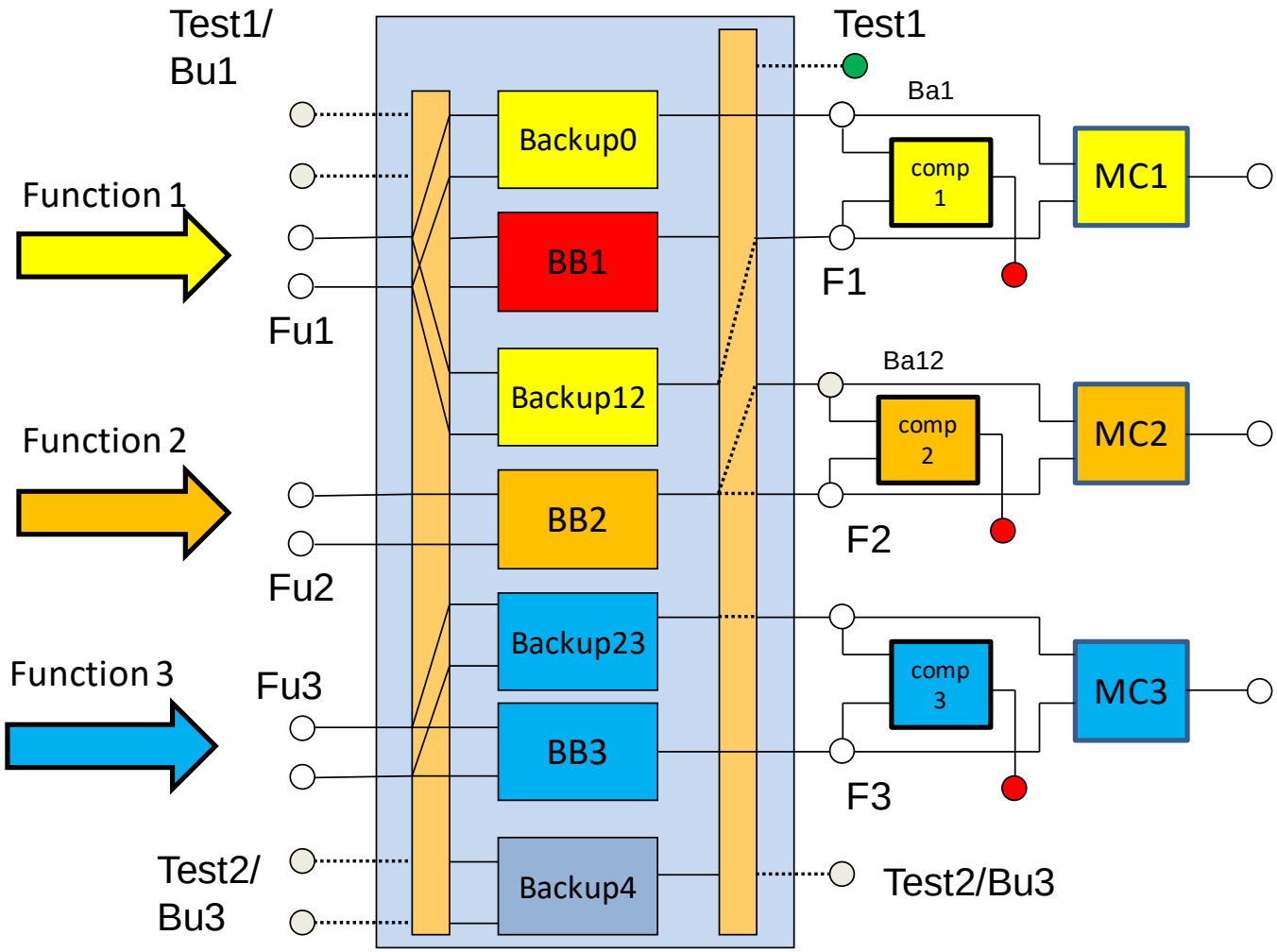
TMR will often work even under multiple fault conditions, if the faults will not affect each other directly or indirectly !

Pseudo-TMR



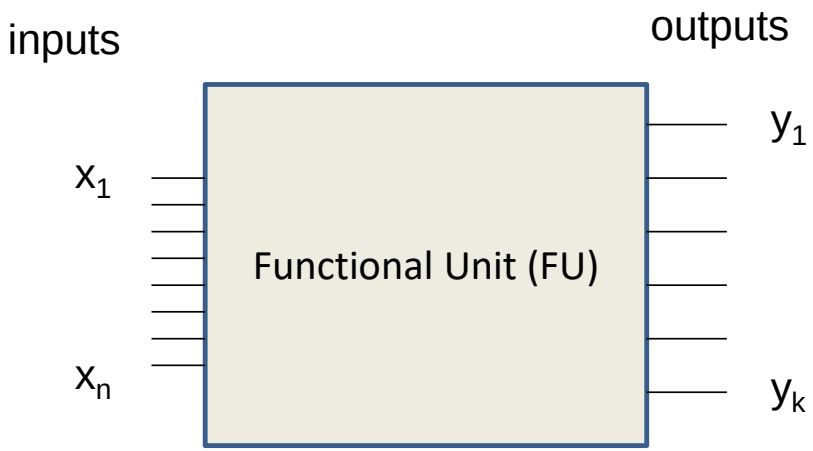
Functional blocks (FBs) have identical structure !

Selective TMR



Normally units are duplicated and run in parallel for each task for fast fault detection. Upon detection of a fault, a unit is „borrowed“ from the neighbout for fault correction. The MC element passes only „equal“ inputs to outputs.

Self-Dualism

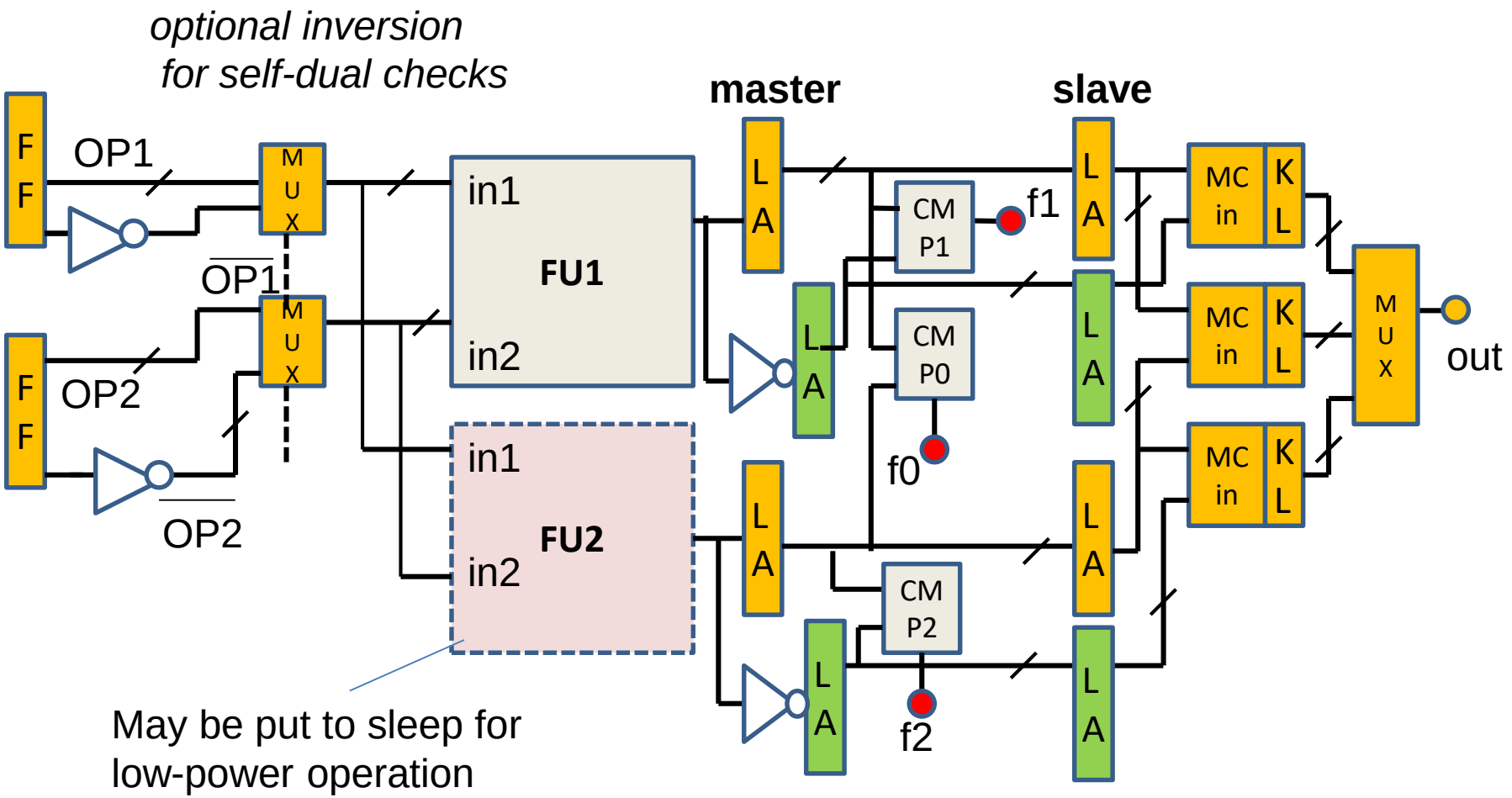


A circuit is self-dual, if upon the inversion of all input bits from any input vector $x_1,\ldots x_n$, also all bits of the output vector $y_1 \ldots y_k$ become inverted. If the circuit has a permanent (stuck-at) fault, this property is not preserved.

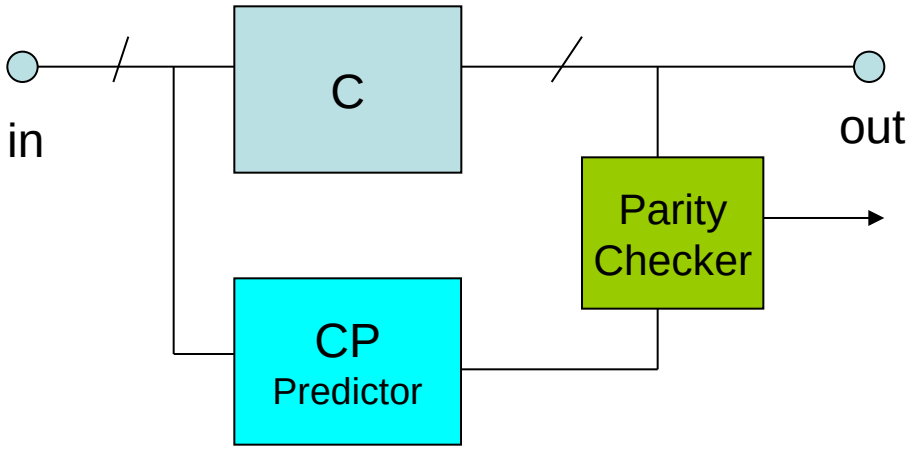
This is a way of fault detection without full duplication of hardware !

Adders that use the 1-complement for representation of negative values are self dual. ALUs are not, but they like many other circuits can be designed to get „auxiliary“ self-dual outputs at little exta cost.

Fault Detection by Self-Dualism

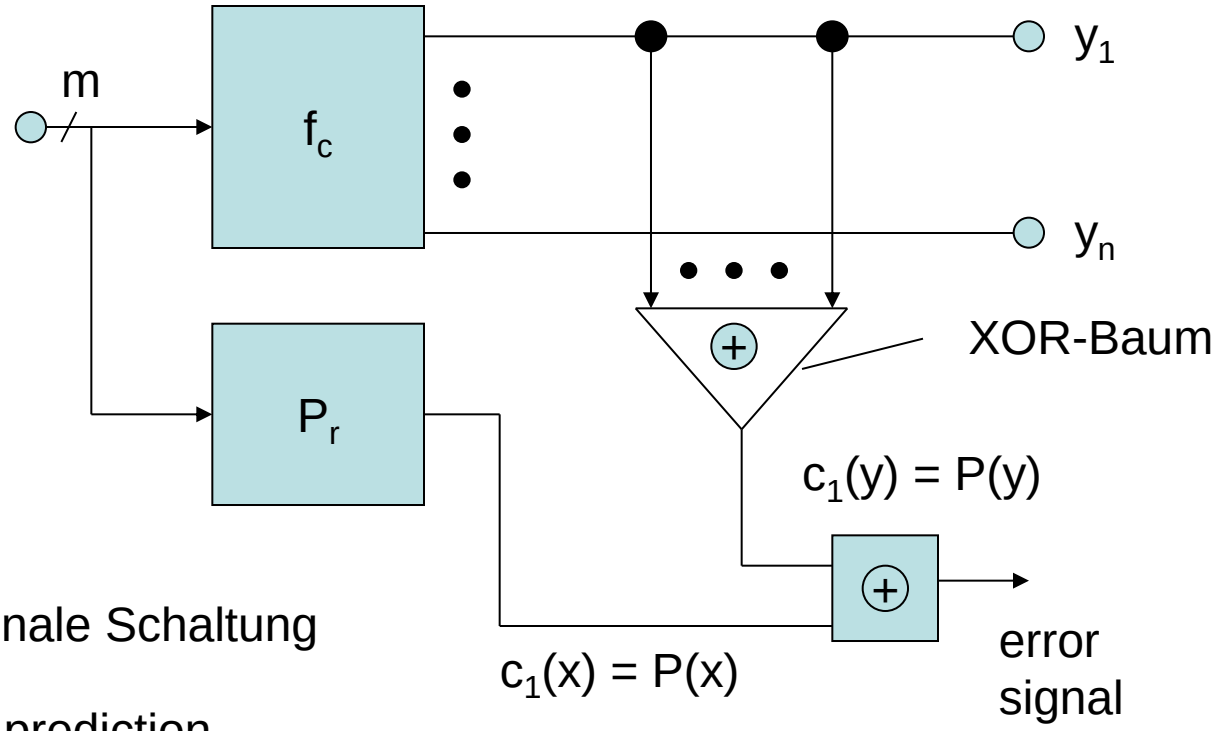


Parity-Bit - Prediction



Code-based methods are also available where circuit outputs are not identical to circuit inputs. But then syndroms have to be computed rather than just transmitted. In the simplest case, a „parity predictor“ is needed.

Parity Prediction

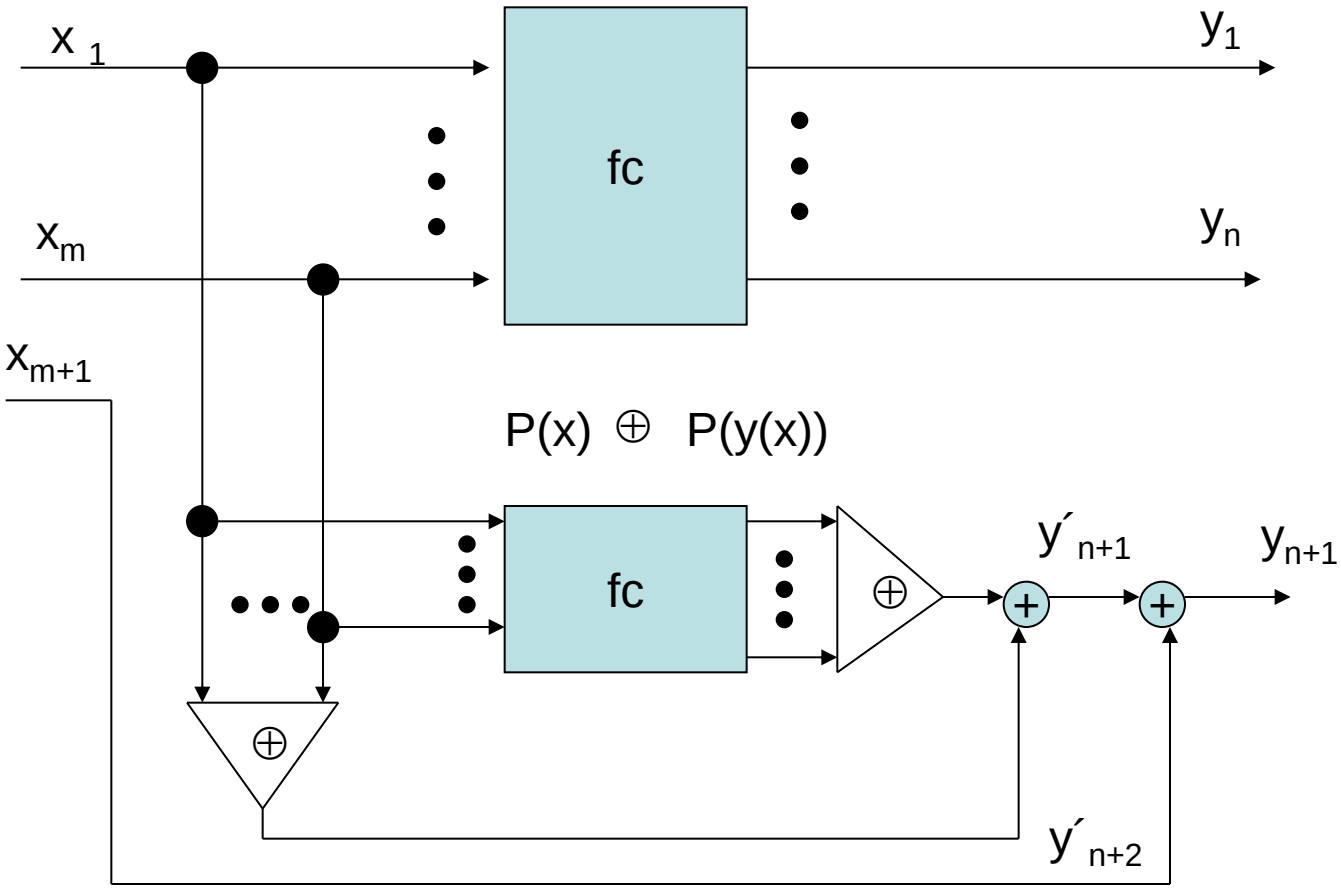


f_c: funktionale Schaltung

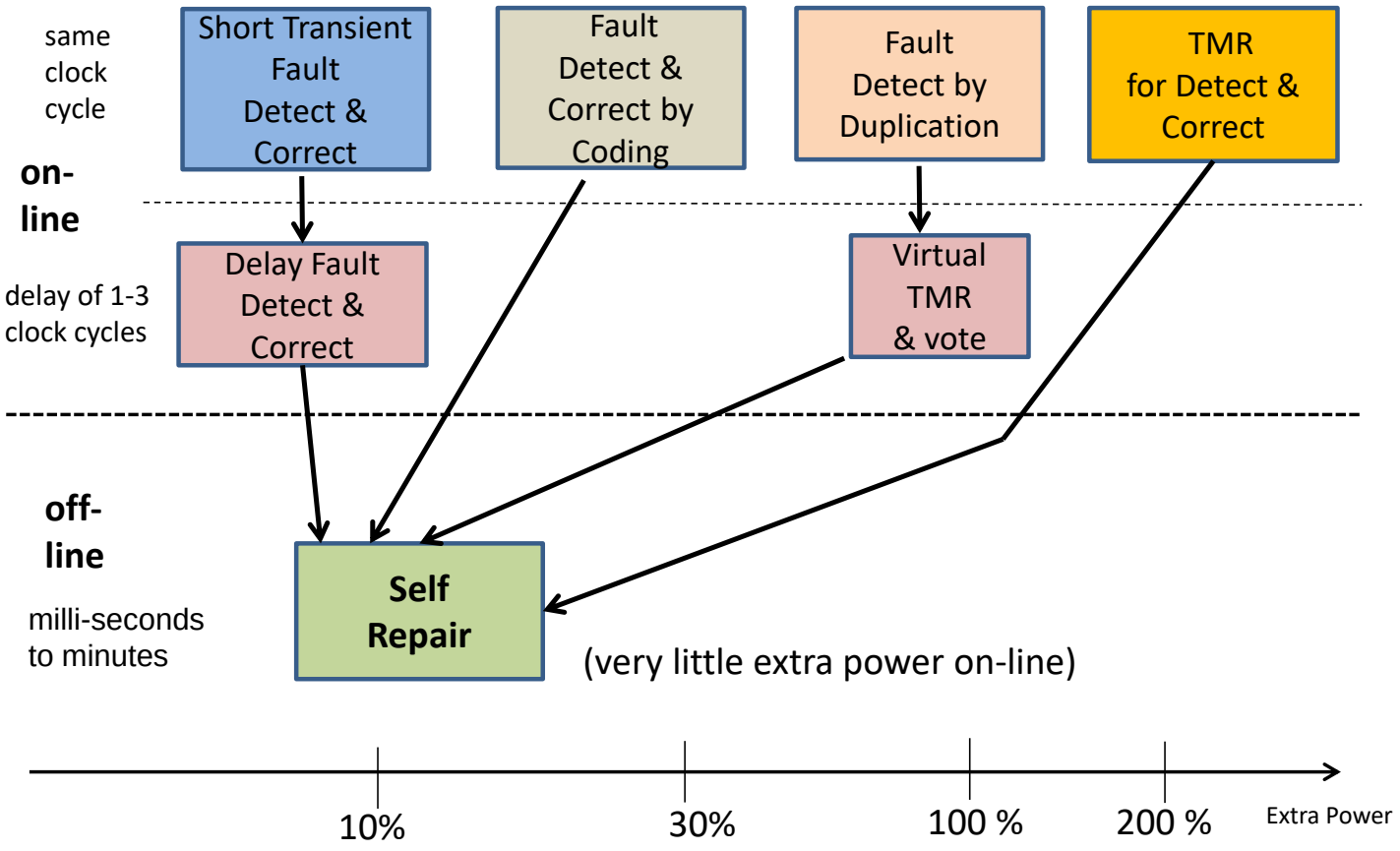
P_r: parity prediction

Parity-Predictor-Circuit

(extension also used to supervise inputs)

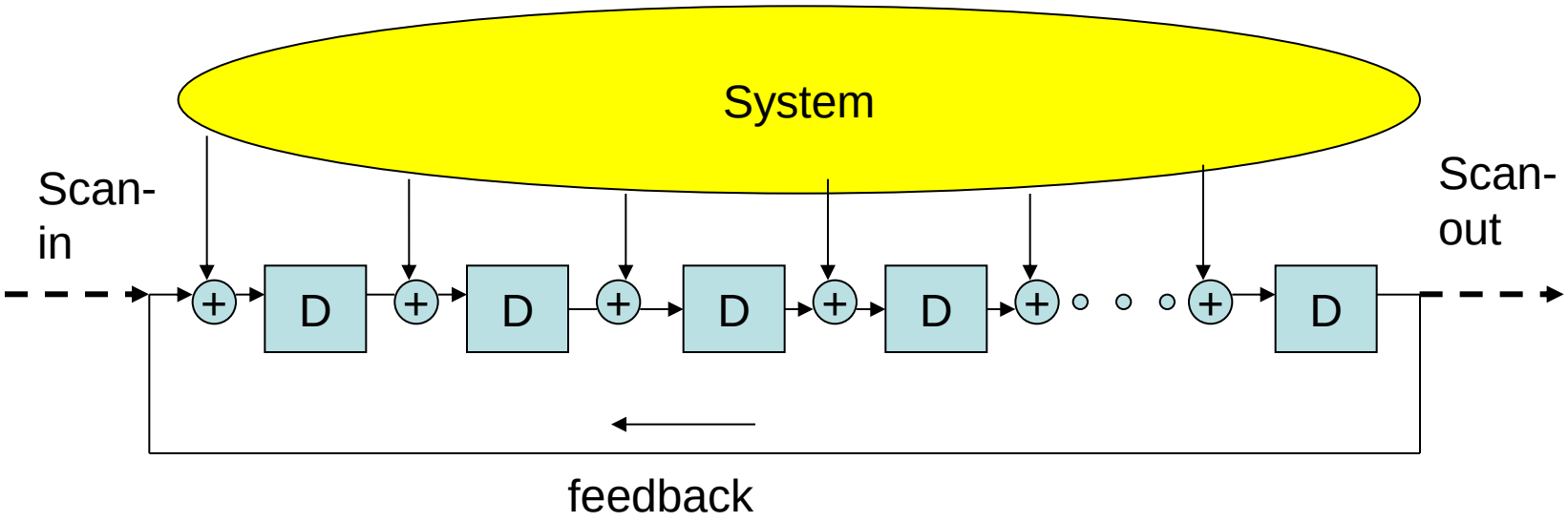


Timing versus Power



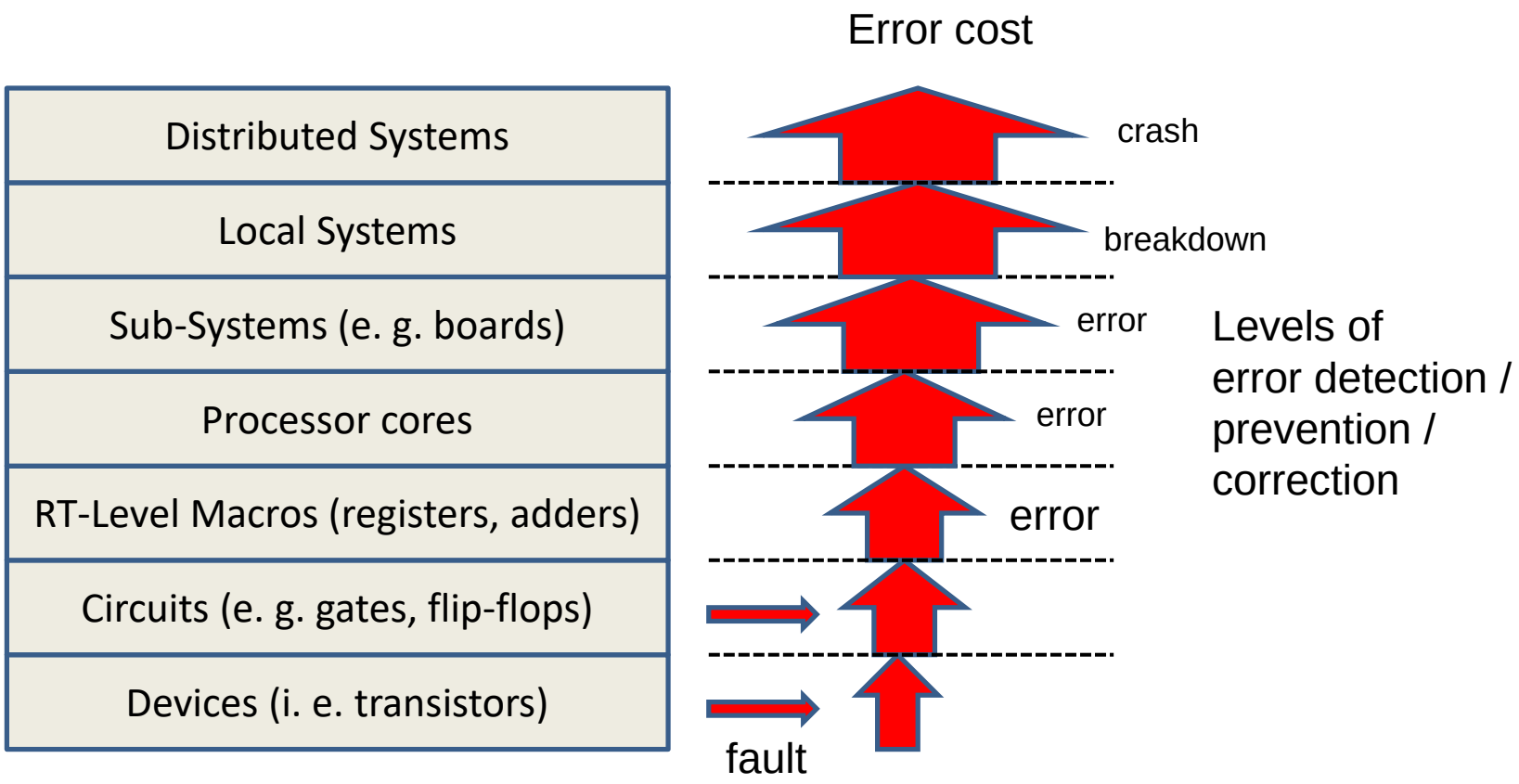
Multiple-Input Signatur-Register

(MISR)



The multi-input signature register is a feedback shift register with additional side- inputs fed in via XORs. This structure is used to collect and compress test information. With the initial content, the feedback and the „good“ side inputs known, the „good“ content after k clock cycles can be predicted easily. Thereby errors „show“.

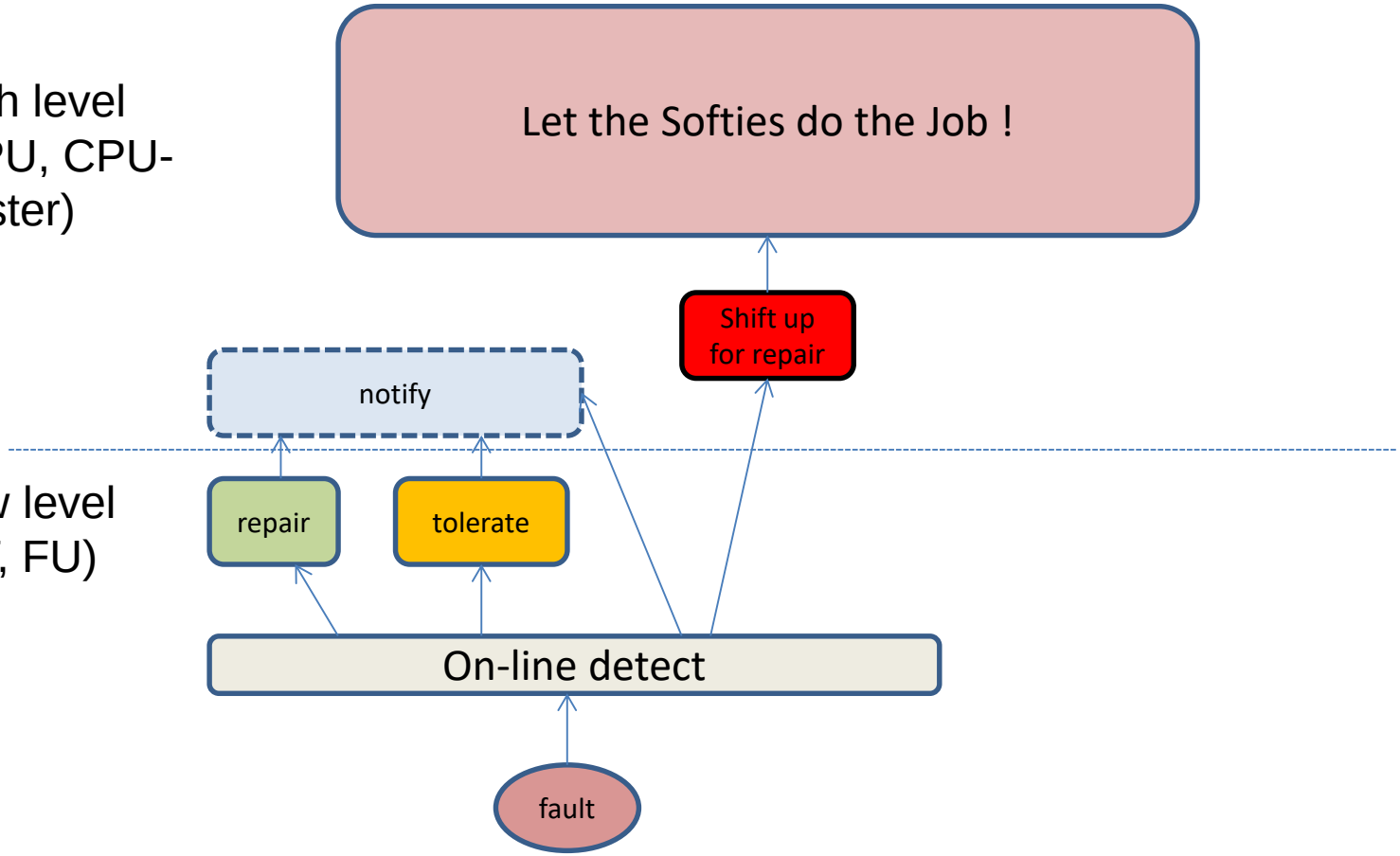
Going to High Levels ?



The real objective is to achieve „**error resilience**“ by keeping low- level / local errors from propagating to higher levels. Systems should be „fail-safe“ even under local error conditions ! *This does not necessary mean immediate error correction* !

Going to High Levels ?

High level (CPU, CPU-cluster)



Low level (RT, FU)

..looks attractive if faults / errors are very rare events (such as once a week).

High-Level Error Handling

Time

- Cyclic redundancy check (CRC)
  Works on large sets of data (transmission, memory) using polynomial divison and comparison of remainder, repair by re-execution
  > 100 clock cycles
  extra power

- Checkpointing
  Partitioning software into packets, double / triple execution by „roll back“, storage, repeate, majority vote
  > 100 clock cycles
  double power

- Signature building (check sum) and comparison
  Works on compaction of data using multi-input feedback shift registers (MISRs), e.g. checking for correct addressing
  > 100 clock cycles
  little extra power

- **Checking numerical data for probable values**

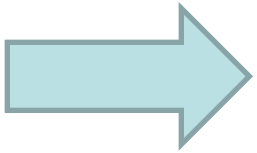
> 10 clock cycles
power

Supervise for maximum allowed difference, replace „unlikely“ result.

Typically high cost in extra time and extra power !

Faults that Can be Tolerated

- Faults that do not show due to redundancy (e. g. in control logic)
- Faults in registers that are overwritten before read.
- Faults in data that have minor overall effects (e. g. single pixels in images), small errors in numerical data.



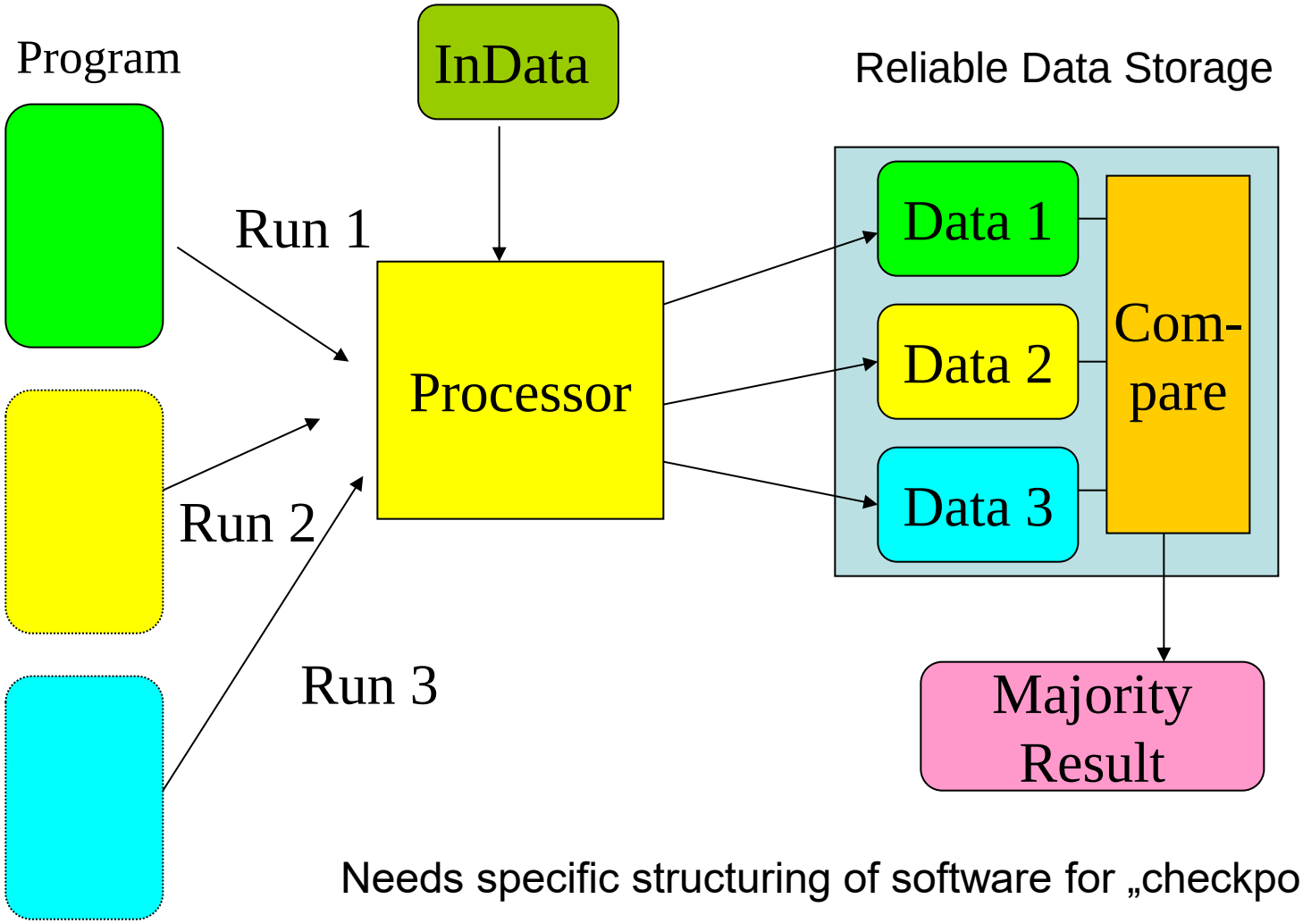
Typically faults that have a direct impact on control flows are most critical !

Non-Tolerable Faults in Processors

- Program counter (instruction pointer) register
- Instruction register
- Address generation units (address ALU)
- Processor status register
- Registers that control interrupts
- Registers that contain information for conditional jumps

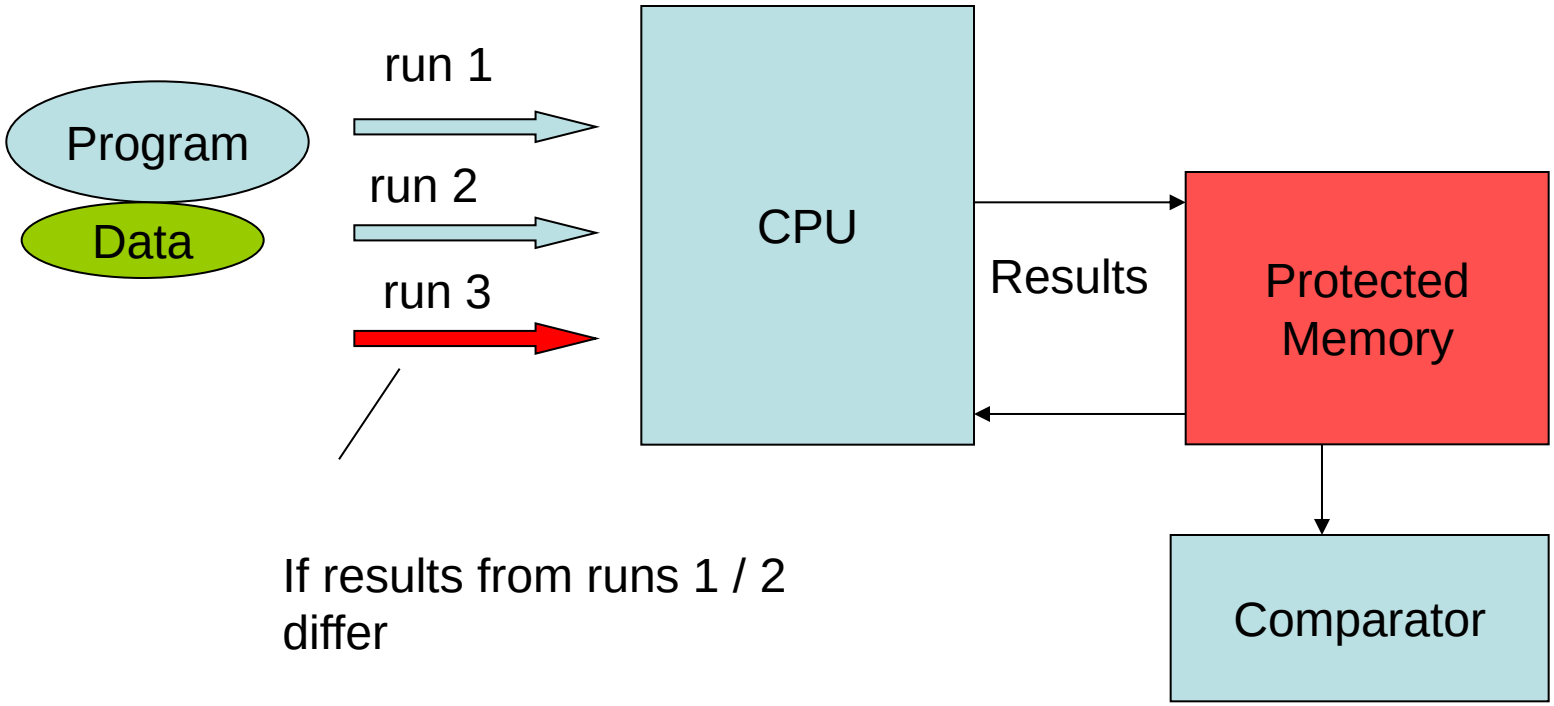
Duplication of registers and compare, pipeline stall ??

Virtual TMR by Double / Triple Execution

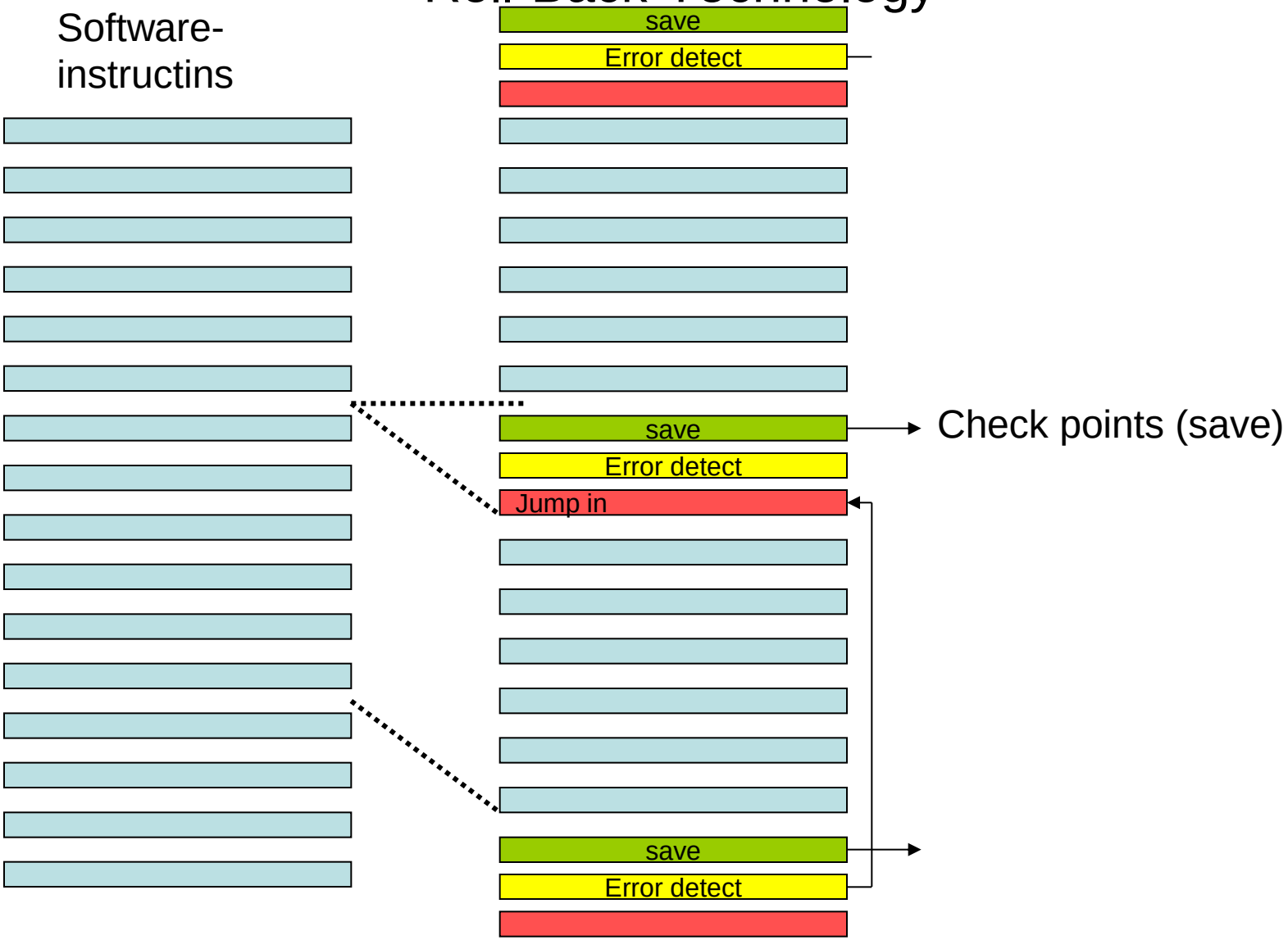


Needs specific structuring of software for „checkpointing“ !

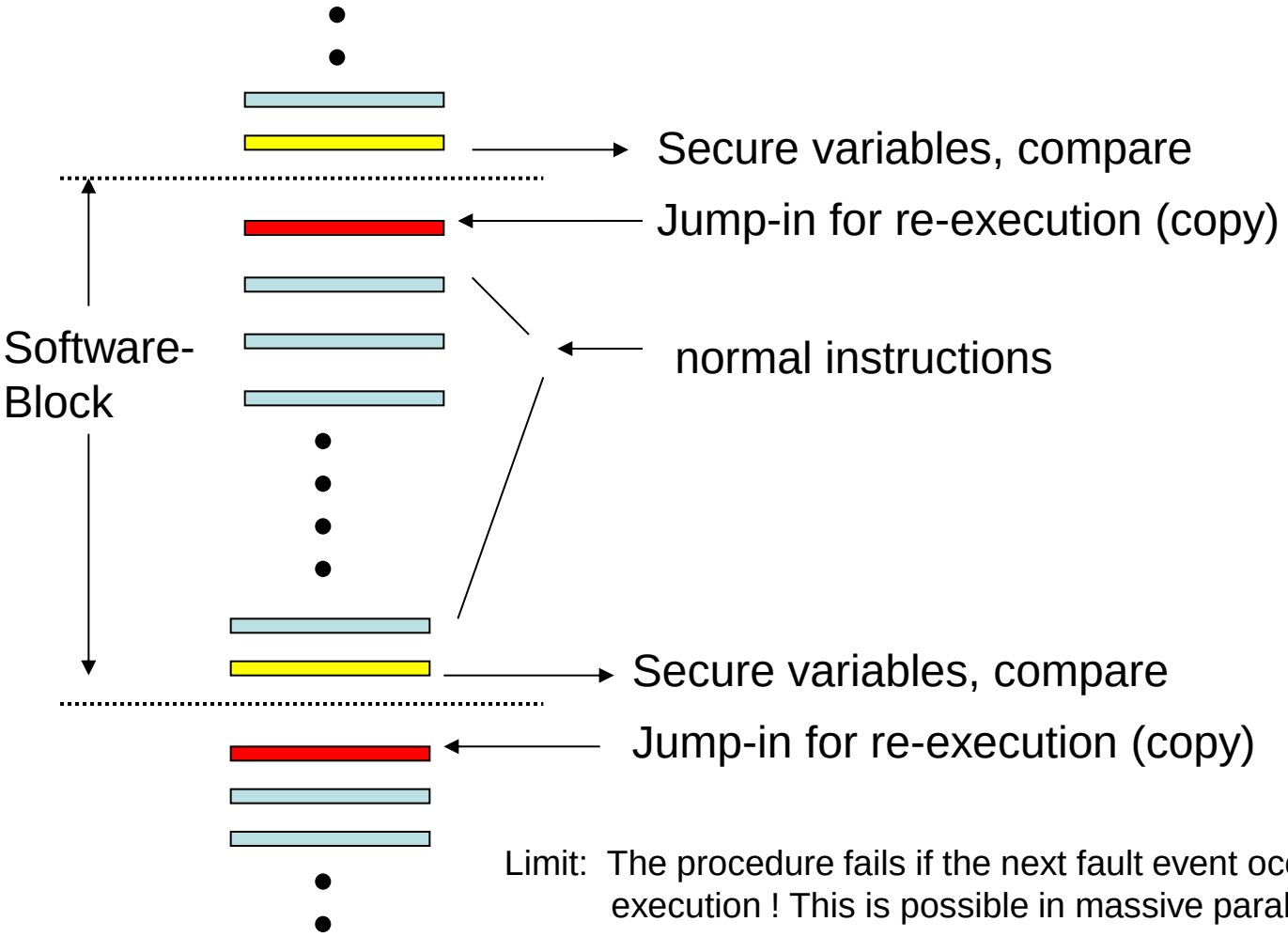
Double / Triple Execution



Roll-Back-Technology

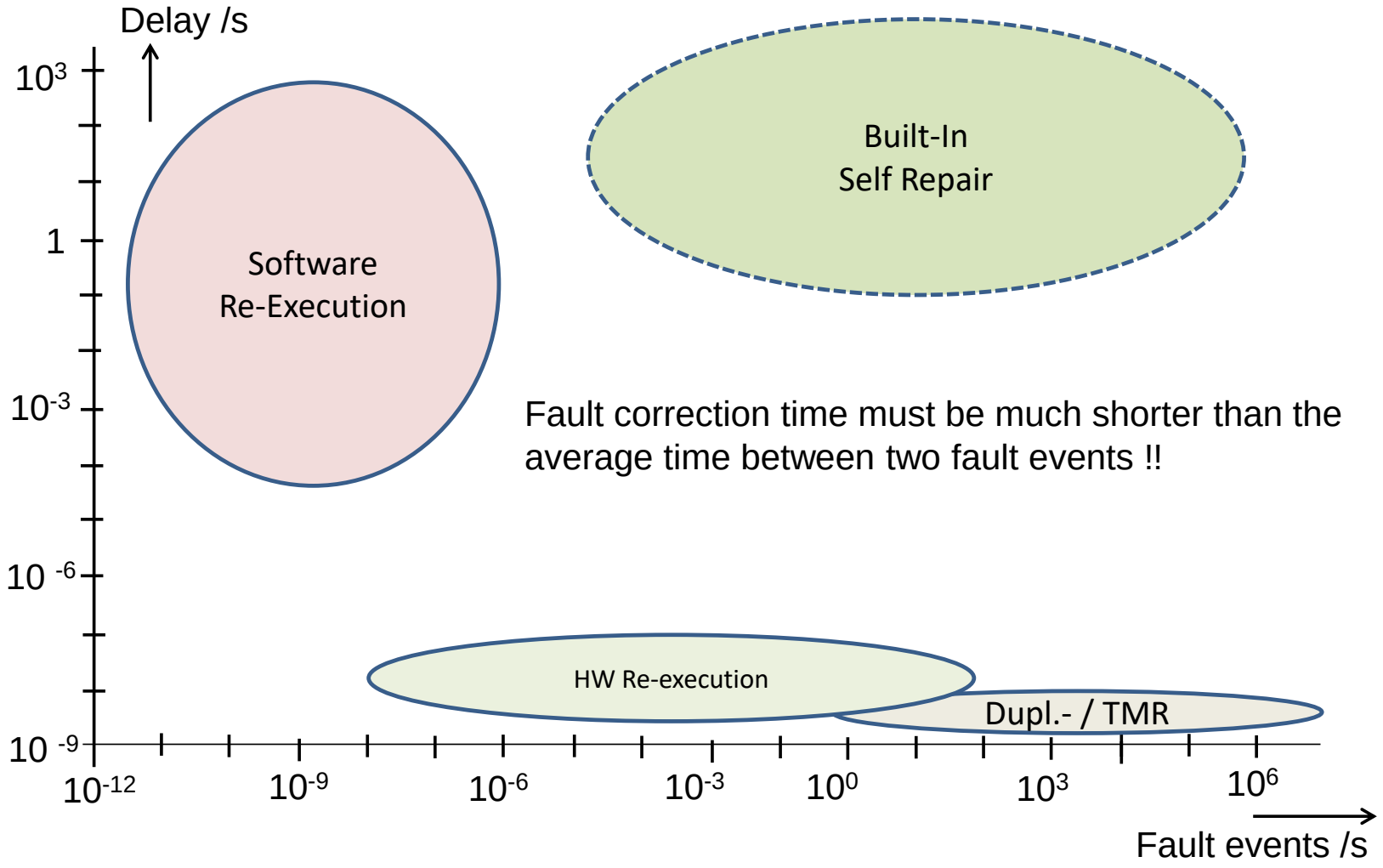


Software Checkpointing

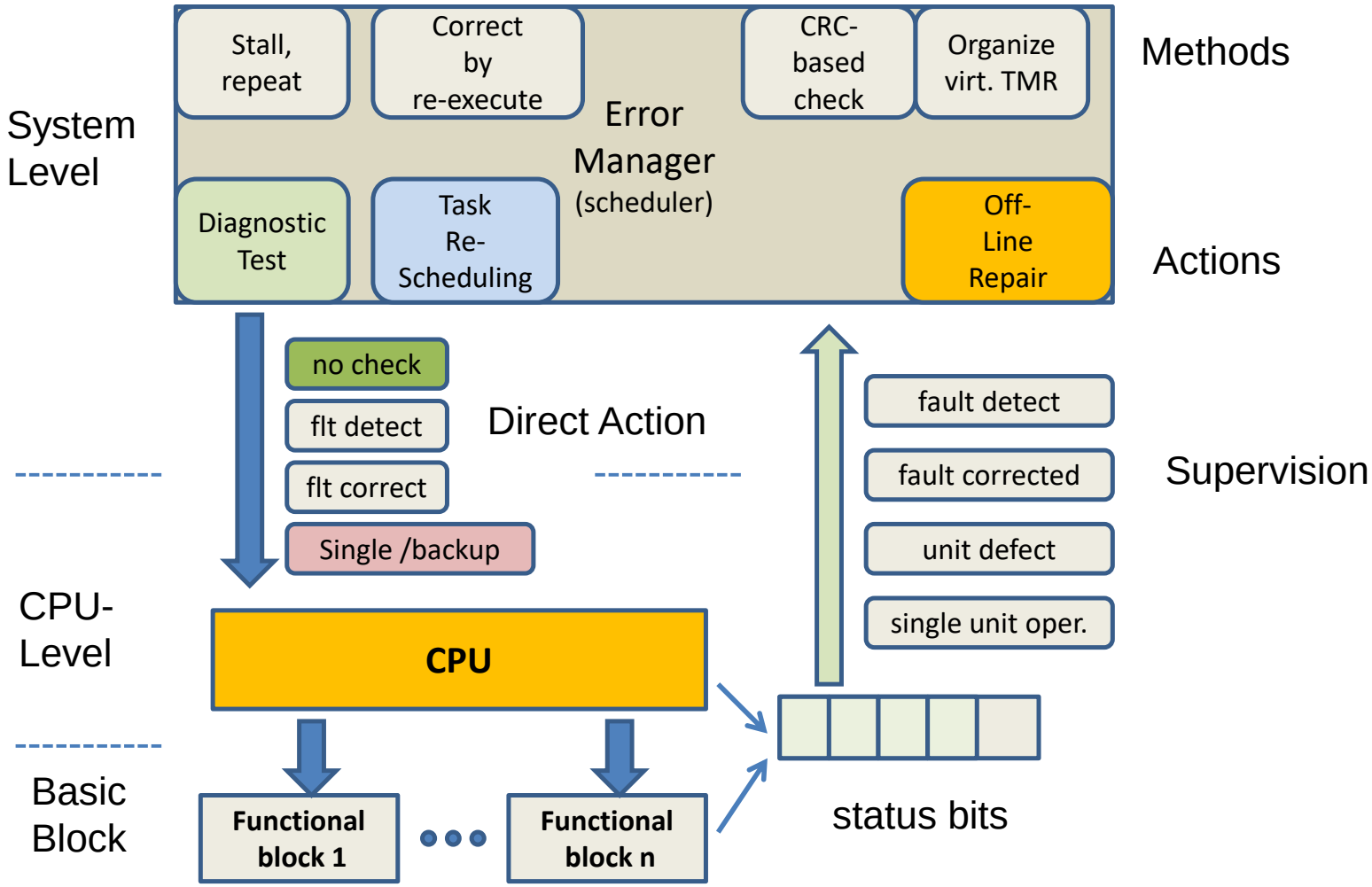


Limit: The procedure fails if the next fault event occurs during re-execution ! This is possible in massive parallel systems !

Timing for Error Correction



Error Management



Summary

There are several methods for the detection and eventual correction in hardware.

The most „universal“ methods like duplication and triplication are also fast, but they need a lot of extra power.

Only under very specific faults assumptions (short SETs) fault detection and compensation is possible almost without extra cost in power and time.

Schemes based on re-execution upon error detection cost extra time, but not much in extra power. But they cause irregular timing and work in transient faults only.

„High-Level“ methods such as „checkpointing“ are relatively expensive in terms of software (re-) structuring and double power consumption.