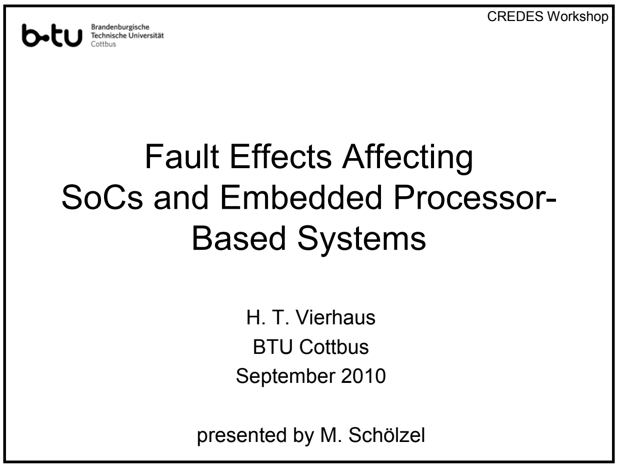
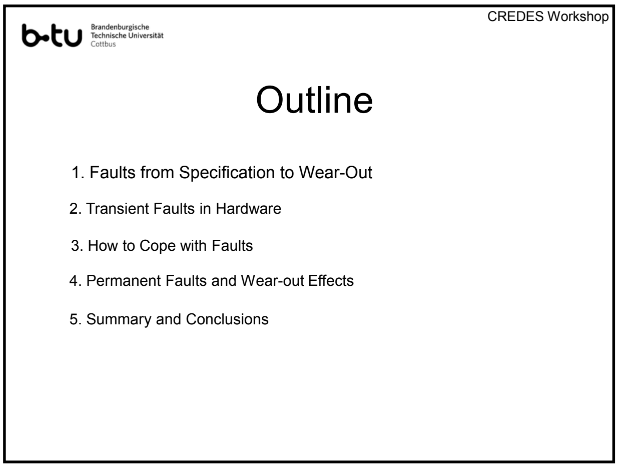


CREDES Workshop

Fault Effects Affecting SoCs and Embedded Processor-Based Systems

H. T. Vierhaus
BTU Cottbus
September 2010

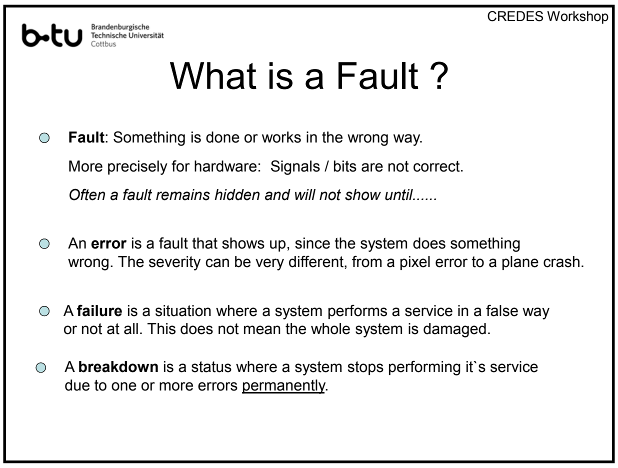
presented by M. Schölzel



CREDES Workshop

Outline

1. Faults from Specification to Wear-Out
2. Transient Faults in Hardware
3. How to Cope with Faults
4. Permanent Faults and Wear-out Effects
5. Summary and Conclusions



CREDES Workshop

What is a Fault ?

- **Fault**: Something is done or works in the wrong way.

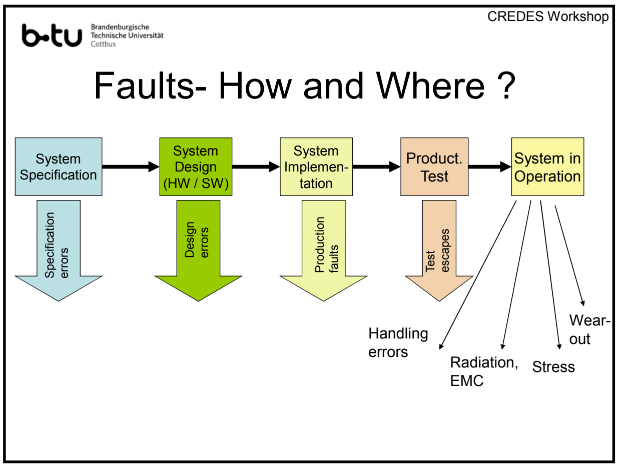
  More precisely for hardware: Signals / bits are not correct.

  Often a fault remains hidden and will not show until......

- An **error** is a fault that shows up, since the system does something wrong. The severity can be very different, from a pixel error to a plane crash.

- A **failure** is a situation where a system performs a service in a false way or not at all. This does not mean the whole system is damaged.

- A **breakdown** is a status where a system stops performing it`s service due to one or more errors permanently.



CREDES Workshop

Faults- How and Where ?

System Specification → System Design (HW / SW) → System Implemen-tation → Product. Test → System in Operation

Specification errors

Design errors

Production faults

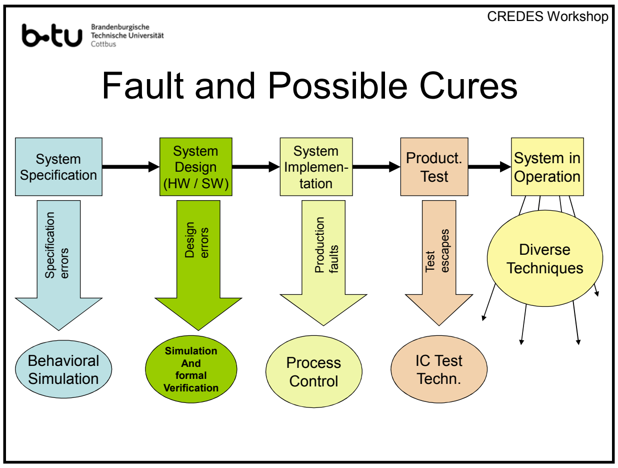
Test escapes

Handling errors

Radiation, EMC

Stress

Wear-out



CREDES Workshop

Fault and Possible Cures

System Specification → System Design (HW / SW) → System Implemen-tation → Product. Test → System in Operation

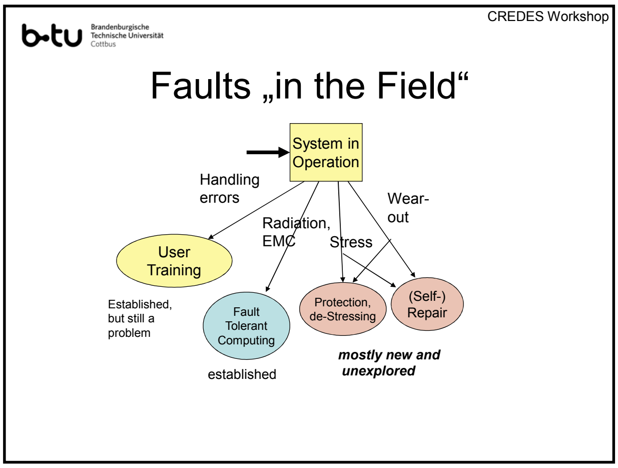
Specification errors → Behavioral Simulation

Design errors → Simulation And formal Verification

Production faults → Process Control

Test escapes → IC Test Techn.

System in Operation → Diverse Techniques



CREDES Workshop

Faults „in the Field“

System in Operation

Handling errors → User Training

Established, but still a problem

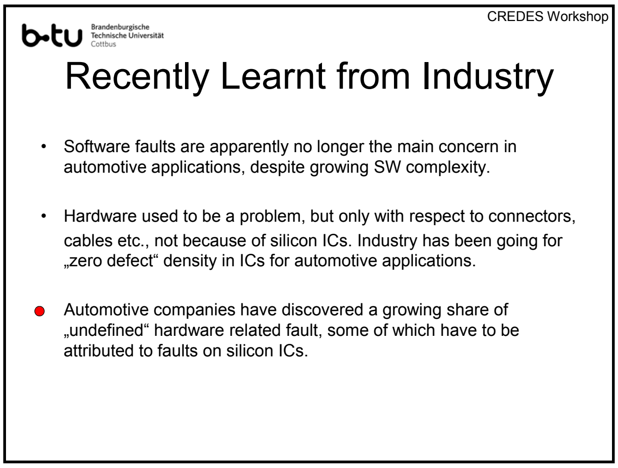
Radiation, EMC → Fault Tolerant Computing

established

Stress → Protection, de-Stressing

Wear-out → (Self-) Repair

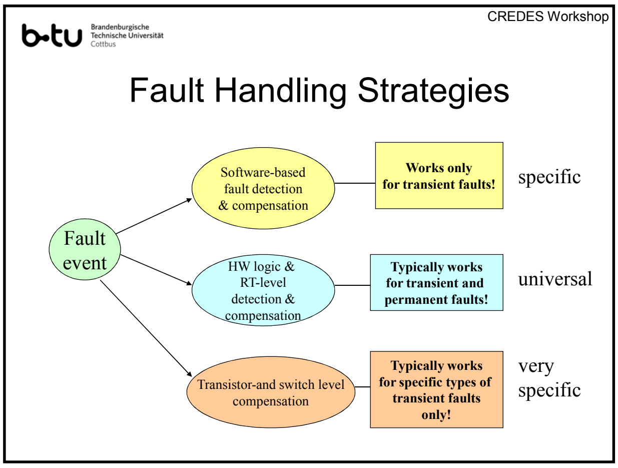
mostly new and unexplored



CREDES Workshop

Recently Learnt from Industry

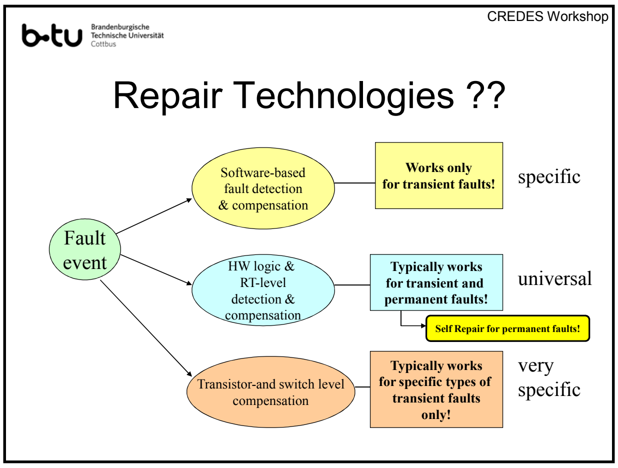
- Software faults are apparently no longer the main concern in automotive applications, despite growing SW complexity.
- Hardware used to be a problem, but only with respect to connectors, cables etc., not because of silicon ICs. Industry has been going for „zero defect“ density in ICs for automotive applications.
- Automotive companies have discovered a growing share of „undefined“ hardware related fault, some of which have to be attributed to faults on silicon ICs.



CREDES Workshop

Fault Handling Strategies

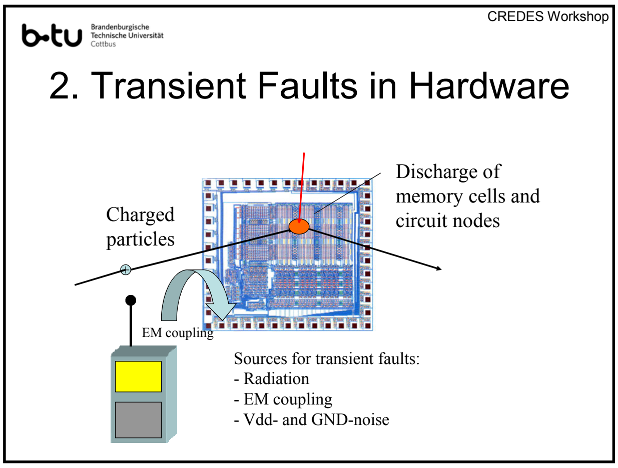
Fault event
Software-based fault detection & compensation
Works only for transient faults!
specific
HW logic & RT-level detection & compensation
Typically works for transient and permanent faults!
universal
Transistor-and switch level compensation
Typically works for specific types of transient faults only!
very specific



CREDES Workshop

Repair Technologies ??

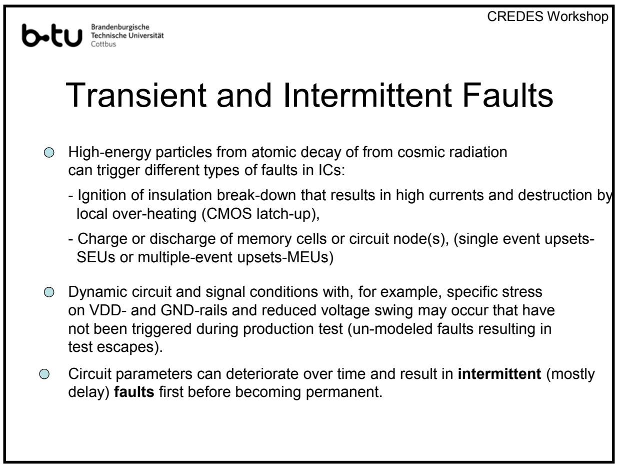
Fault event
Software-based fault detection & compensation
Works only for transient faults!
specific
HW logic & RT-level detection & compensation
Typically works for transient and permanent faults!
universal
Self Repair for permanent faults!
Transistor-and switch level compensation
Typically works for specific types of transient faults only!
very specific



CREDES Workshop

2. Transient Faults in Hardware

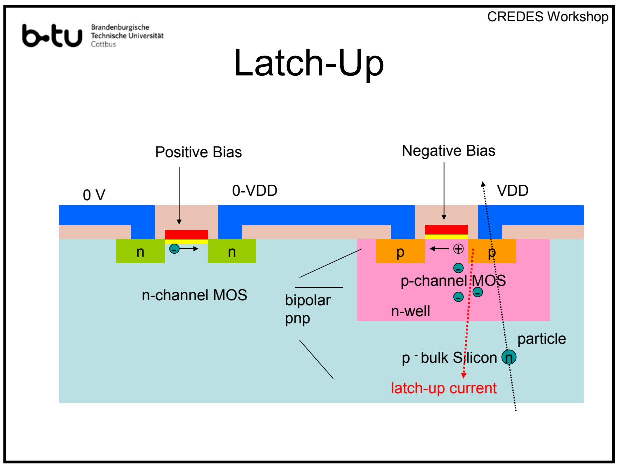
Sources for transient faults:
- Radiation
- EM coupling
- Vdd- and GND-noise



CREDES Workshop

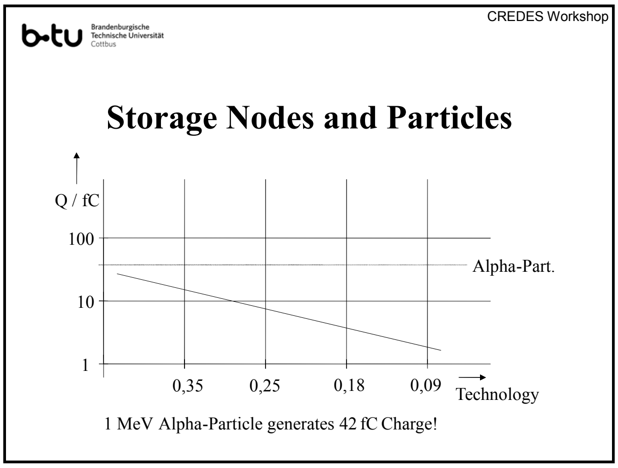
Transient and Intermittent Faults

- High-energy particles from atomic decay of from cosmic radiation can trigger different types of faults in ICs:
  - Ignition of insulation break-down that results in high currents and destruction by local over-heating (CMOS latch-up),
  - Charge or discharge of memory cells or circuit node(s), (single event upsets-SEUs or multiple-event upsets-MEUs)
- Dynamic circuit and signal conditions with, for example, specific stress on VDD- and GND-rails and reduced voltage swing may occur that have not been triggered during production test (un-modeled faults resulting in test escapes).
- Circuit parameters can deteriorate over time and result in **intermittent** (mostly delay) **faults** first before becoming permanent.



CREDES Workshop

Latch-Up



CREDES Workshop

Storage Nodes and Particles

1 MeV Alpha-Particle generates 42 fC Charge!

CREDES Workshop

Contribution to Soft-Error Rates

Static combinational logic:	11%
Sequential elements (FFs, Latches):	49%
Unprotected SRAM:	40%

Source: S. Mitra, N. Seifert, M. Zhang, Q. Shi, K. S. Kim,
„Robust System Design with Built-In Soft Error Resilience"
IEEE Computer, Vol. 38, No.2, Febr. 2005, pp. 43-52

CREDES Workshop

3. How to Cope with Faults

- Using extra redundant **Hardware** (e. g. by triplication):

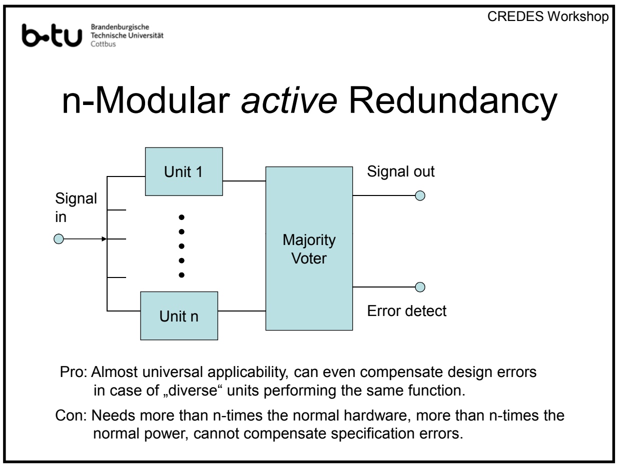
  May handle all types of faults, but at high extra cost in terms of devices, chip area and power consumption.

- Using extra **Information** by coding (information redundancy):

  May also handle almost any type of fault, as long as the faults are not too massive (single- or double bit faults).

- Using extra **Time** by re-calculating the same job several times (timing redundancy):

  Works only for „transient" faults, not for permanent faults.

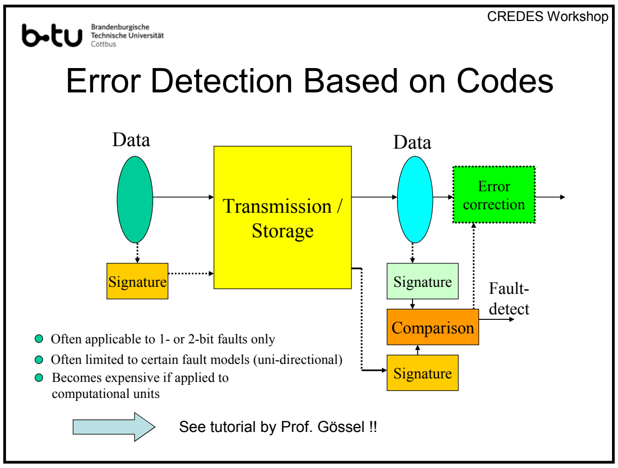


CREDES Workshop

n-Modular *active* Redundancy

Pro: Almost universal applicability, can even compensate design errors in case of „diverse" units performing the same function.

Con: Needs more than n-times the normal hardware, more than n-times the normal power, cannot compensate specification errors.

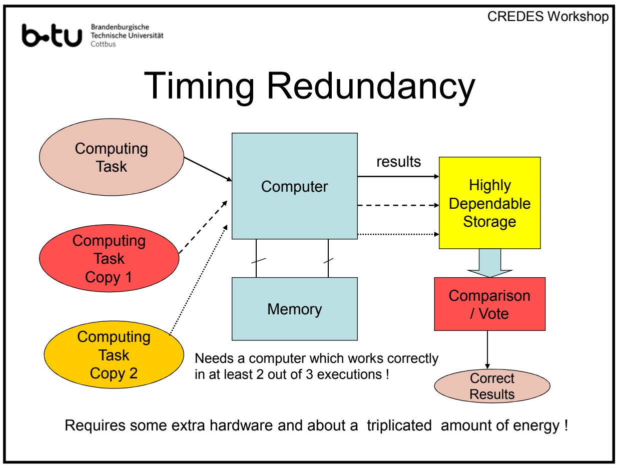


CREDES Workshop

Error Detection Based on Codes

- Often applicable to 1- or 2-bit faults only
- Often limited to certain fault models (uni-directional)
- Becomes expensive if applied to computational units

See tutorial by Prof. Gössel !!



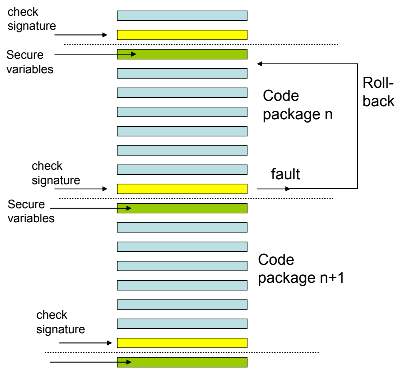
CREDES Workshop

Timing Redundancy

Needs a computer which works correctly in at least 2 out of 3 executions !

Requires some extra hardware and about a triplicated amount of energy !

Roll-Back Technology



Can TMR and Codes Compensate Permanent Faults?

- Fault / error detection circuitry typically works on a clock-cycle base. It does not „know“ if a fault is transient or permanent.
- A permanent fault is a fault event that occurs in several to many successive clock cycles repeatedly.
- Error correction technology can detect and compensate such permanent faults as well as transient faults.
- A critical condition occurs if transient faults occur *on top of* permanent faults. Then the superposition of fault effects is likely to exceed the system‘s fault handling capacity.

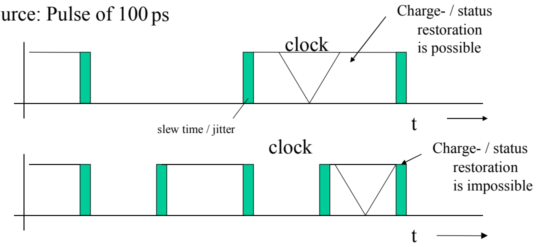
→ Self repair!

Short-Time Transient Fault Detection

- Fault effects induced by sub-atomic particles are likely to be very short (shorter than one clock cycle).
- Processor manufacturers (Intel, IBM) have developed specific low-cost technologies to cope with such faults. Mainly those working with „bulk silicon“.
- Processor manufacturers working with „silicon on insulator“ (SoI) (AMD) seem to have fewer problems with such effects!

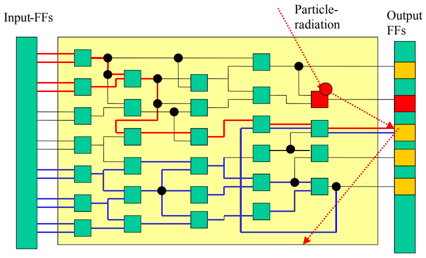
Spikes and Clock Rates in Logic

Source: Pulse of 100 ps



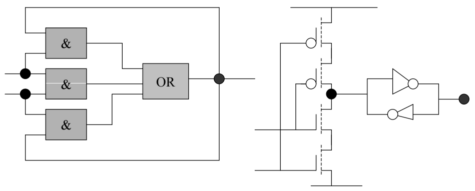
Fault probability in digital logic is about proportional to clock frequency!

Logic Structures and Fault Events



Flip-flops need fault tolerance / fault hardening in the first place, logic close-to outputs comes next.

Muller-C-Element

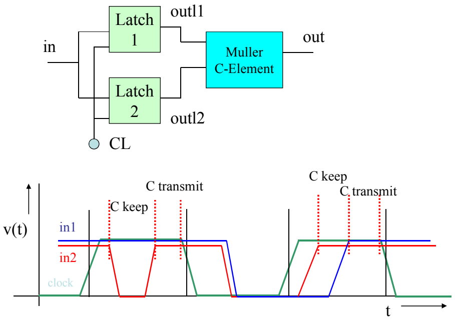


If both inputs are equal, the value is stored.

If the inputs are different, the previous value is kept.

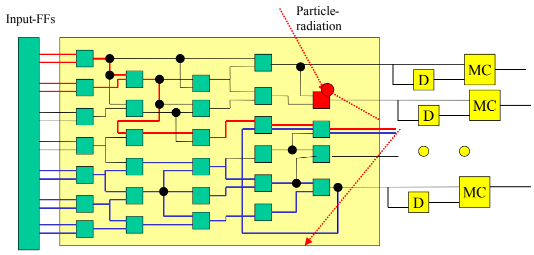
CREDES Workshop

Fault Compensation



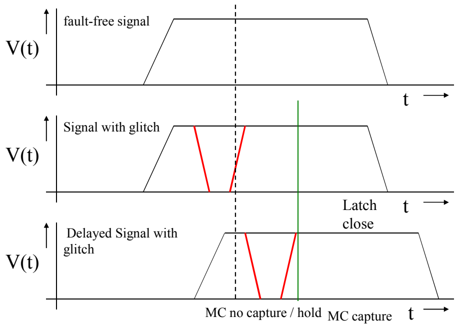
CREDES Workshop

Fault Compensation in Combinational Logic



CREDES Workshop

Fault Compensation in Combinational Logic



CREDES Workshop

Limits of „Low Level“ Fault Compensation

- The chips must have specific extensions in their flip-flops and scan path elements, which are not cheap. Most probably, therefore the schemes are economical only for high-end processor manufacturers (Intel, AMD).
- The method works only under specific timing conditions and- assumptions, e. g. the SEU-effect must be shorter than a clock cycle.
- No cure against „internal“ fault effects resulting from wear-out and / or parameter degradation.
- **Companies like IHP have developed „radiation hardened“ IC technologies, using e. g. shielding, multiple contacts, extra insulation etc.**

CREDES Workshop

4. Permanent Faults

Permanents faults are „steady“ due to real physical defects.

They may be of static (never works) or of dynamic (works, but not fast enough) nature.

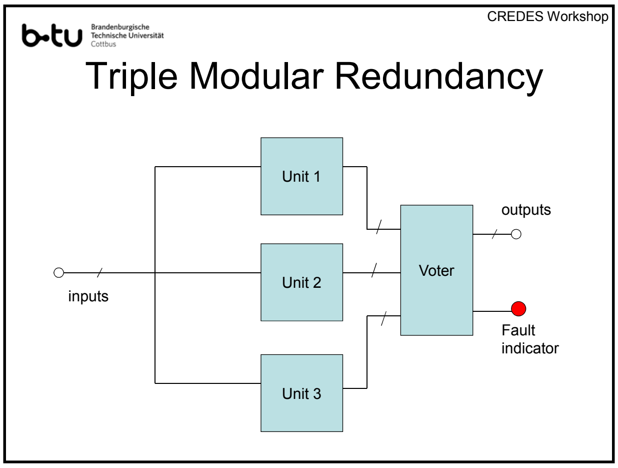
Permanent faults are not „steady and stable“, but may occur after production as *production faults*, but also later during the life-time of a system as *wear-out based faults*, or due to system over- stressing as *stress-induced faults* (by voltage, temperature).

Permanent faults cannot be compensated by redundancy in time, since they will occur also in repeated operations on the same hardware.

CREDES Workshop

Sources of Intermittent and Permanent Faults

- Test escapes: IC test technology for production testing often uses operating modes for ICs which stress VDD / GND rails quite differently from normal operation. The results may be faults which are not „real“ (*overtesting*), but also *test escapes*.
- Unmodeled faults: IC test technology for digital circuits and systems often uses relatively „simple“ fault assumptions, such as gate inputs / outputs stuck to 0 / 1 and then tests for such faults. But real physics can be **much** more complex.
- Hidden defects: ICs may have defects, which do not harm the function much after production, such as kOhm-bridges between lines. During system operation, such defects may vanish (fuse!) or become worse (short).



CREDES Workshop

Triple Modular Redundancy

Unit 1

Unit 2

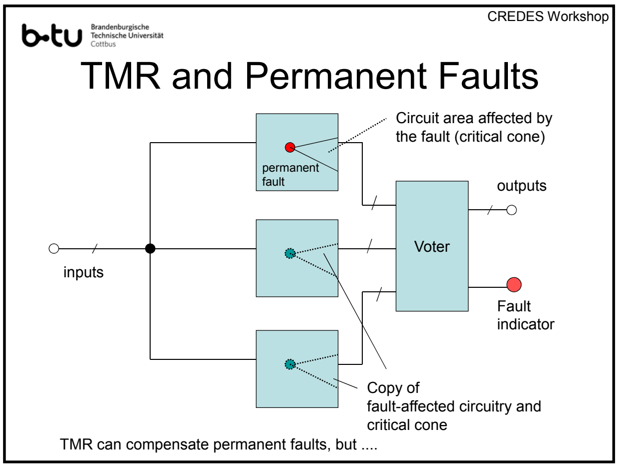
Unit 3

Voter

inputs

outputs

Fault indicator



CREDES Workshop

TMR and Permanent Faults

permanent fault

Circuit area affected by the fault (critical cone)

Voter

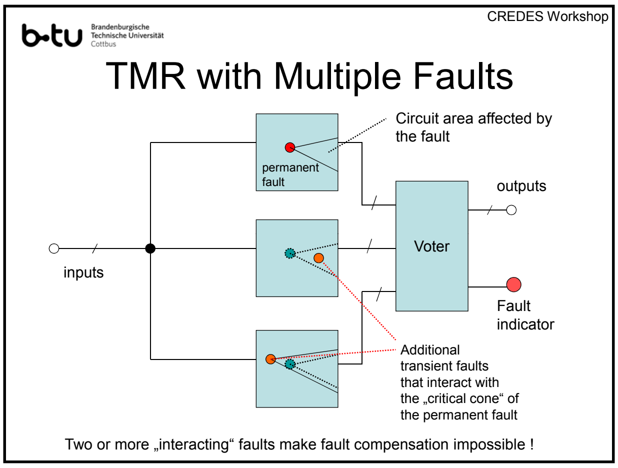
outputs

inputs

Fault indicator

Copy of fault-affected circuitry and critical cone

TMR can compensate permanent faults, but ....



CREDES Workshop

TMR with Multiple Faults

permanent fault

Circuit area affected by the fault

Voter

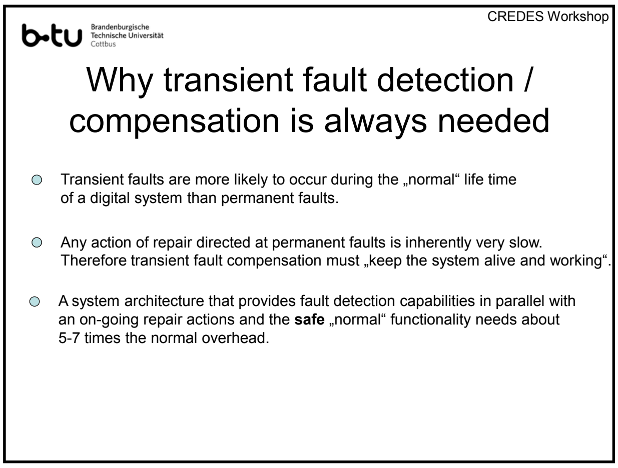
outputs

inputs

Fault indicator

Additional transient faults that interact with the „critical cone" of the permanent fault

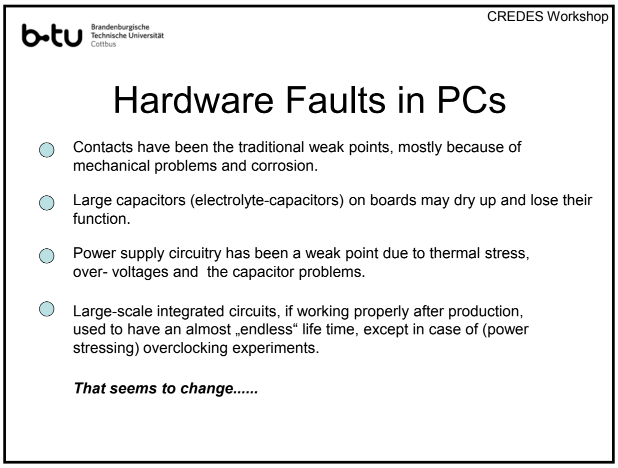
Two or more „interacting" faults make fault compensation impossible !



CREDES Workshop

Why transient fault detection / compensation is always needed

- Transient faults are more likely to occur during the „normal" life time of a digital system than permanent faults.
- Any action of repair directed at permanent faults is inherently very slow. Therefore transient fault compensation must „keep the system alive and working".
- A system architecture that provides fault detection capabilities in parallel with an on-going repair actions and the **safe** „normal" functionality needs about 5-7 times the normal overhead.

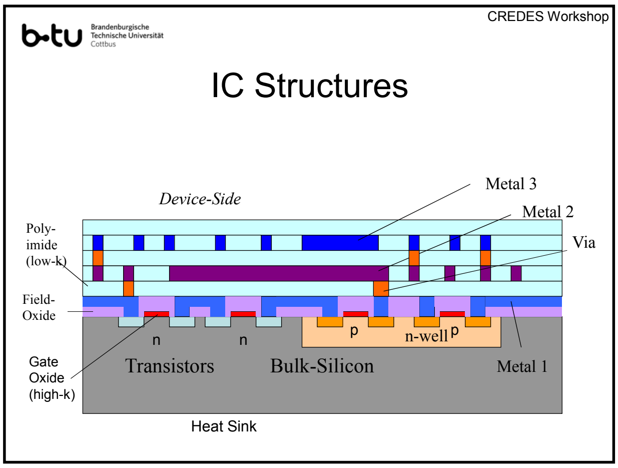


CREDES Workshop

Hardware Faults in PCs

- Contacts have been the traditional weak points, mostly because of mechanical problems and corrosion.
- Large capacitors (electrolyte-capacitors) on boards may dry up and lose their function.
- Power supply circuitry has been a weak point due to thermal stress, over- voltages and the capacitor problems.
- Large-scale integrated circuits, if working properly after production, used to have an almost „endless" life time, except in case of (power stressing) overclocking experiments.

That seems to change......



CREDES Workshop

IC Structures

CREDES Workshop

Where Do Permanent Faults Come From?

- Interconnects
- Insulating layers
- MOS transistors

Reasons are:

- Defects induced during IC production

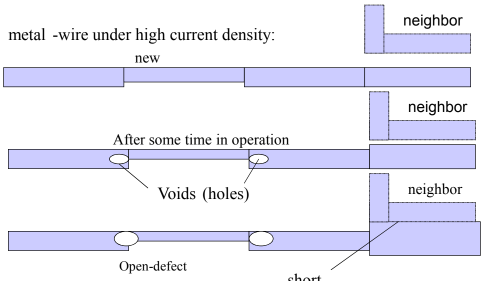
⇨ Tend to be more and more well controlled by IC companies.

- Defects emerging „in the field“, for example due to stress and wear-out

⇨ May even be welcome, since it may promote the sales of new Systems (mobile phones) , but may become critical for „high quality“ products (cars, trains, planes, satellites).

CREDES Workshop

Metal Migration


metal -wire under high current density:
new
neighbor
neighbor
After some time in operation
Voids (holes)
neighbor
Open-defect
short

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

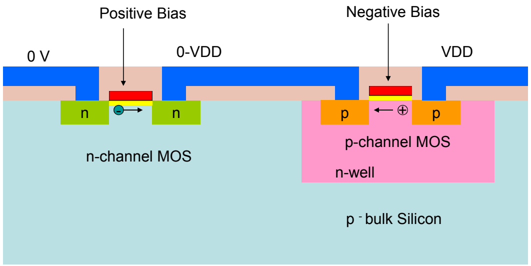
CREDES Workshop

Metal Migration

- Metal migration is „stress migration“, mainly on interconnects with a high current density and single current direction.
- The effect grows exponentially with temperature.
- Wire-to-via-connections are specifically affected.
- De-stressing and „healing“ is partly possibly, if current directions can be reversed.
- Systems need to be designed for a minimum guaranteed life time, considering metal migration under realistic conditions.

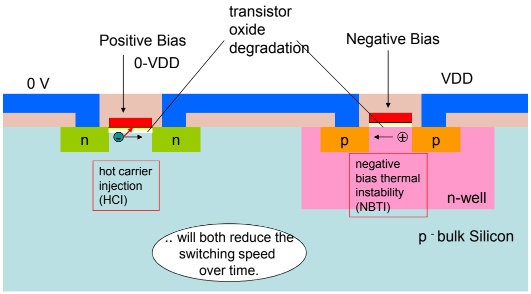
CREDES Workshop

CMOS Inverter



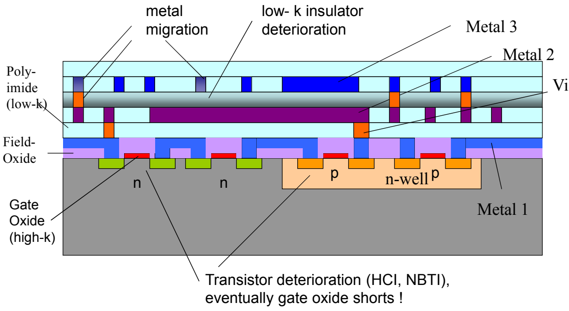
CREDES Workshop

Transistor Degradation


transistor oxide degradation
Positive Bias
Negative Bias
0-VDD
0 V
VDD
n
n
p
p
hot carrier injection (HCI)
negative bias thermal instability (NBTI)
n-well
p - bulk Silicon
.. will both reduce the switching speed over time.

CREDES Workshop

Degradation Effects


metal migration
low- k insulator deterioration
Metal 3
Metal 2
Via
Poly-imide (low-k)
Field-Oxide
Gate Oxide (high-k)
n
n
p
n-well
p
Metal 1
Transistor deterioration (HCI, NBTI), eventually gate oxide shorts !

Degradation Effects

- Metal migration is predictable and can partly be reversed by reversing the direction of currents.
  - ⇨ De-stressing is *partly* possible !
- HCI and NBTI seem to be predictable and can partly be reversed or prevented by changing the bias condition.
- Gate oxide deterioration and low-k-insulator breakdown seem to be difficult to predict by lifetime-simulation.

Cures for Wear-out Induced Faults

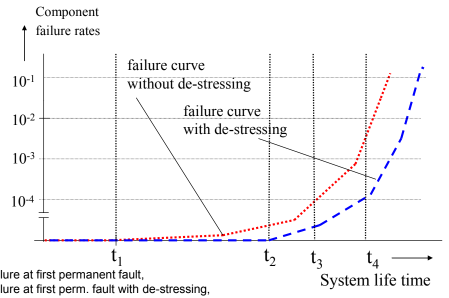
- Repair by redundant elements that have been kept „on store“ , suffering little or not at all from degradation.
  - ⇨ A state-of-the-art self-repair technology for memory devices (specifically embedded SRAM-blocks).
- De-stressing by rotation of tasks between execution units, may also involve extra (redundant) units.
  - ⇨ Can easily be done on multi-core-CPUs, but not in any type of hardware.

5. System Life-Time Design

Systems will have to be designed for a specific „built-in“ life time, depending on :

- Features of the basic technology,
- Design style with / without using devices up to their limits,
- Environmental conditions (voltage levels, temperature),
- With / without error detection and – compensation,
- The amount and the administration of redundant resources.

Combining De-Stressing and Repair

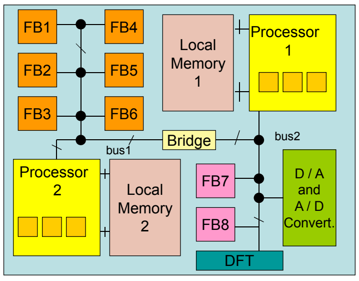


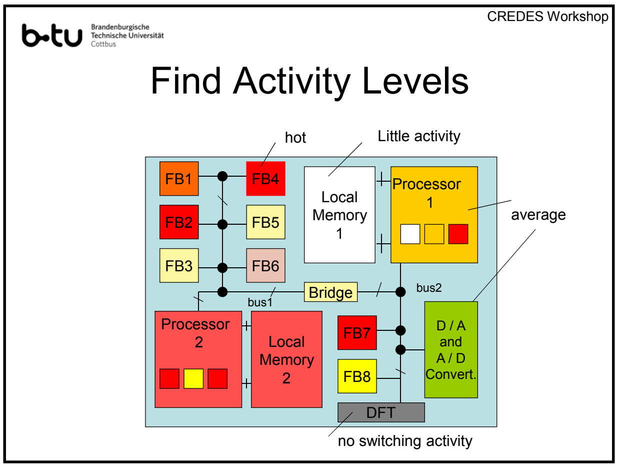
t1: failure at first permanent fault,
t2: failure at first perm. fault with de-stressing,
t3: failure when redundancy is exhausted for repair,
t4: failure after de-stressing and exhausted repair supplies.

Design for Lifetime

- ● Highly active areas and building blocks will suffer from metal migration
  - ⇨ De - stressing, re-distribution or activities, allocation of redundancy
- ● Areas which are virtually un-used except for special cases (test, interrupt) but are under constant bias need some level of activity for de-stressing !
- ● Software functions may be designed to „migrate“ between different processor units for load- balancing and de- stressing !

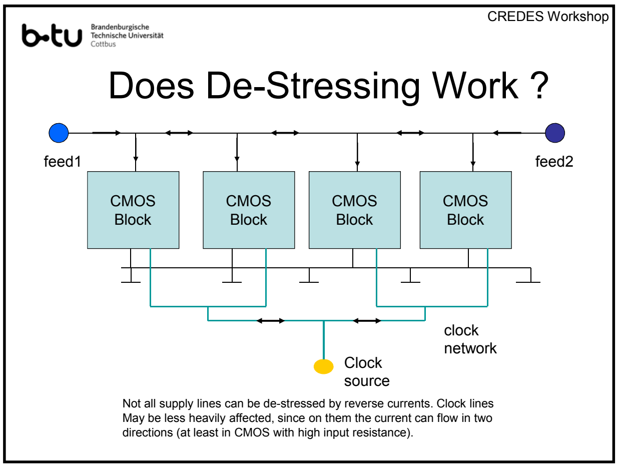
System-on-Chip (SoC)





CREDES Workshop

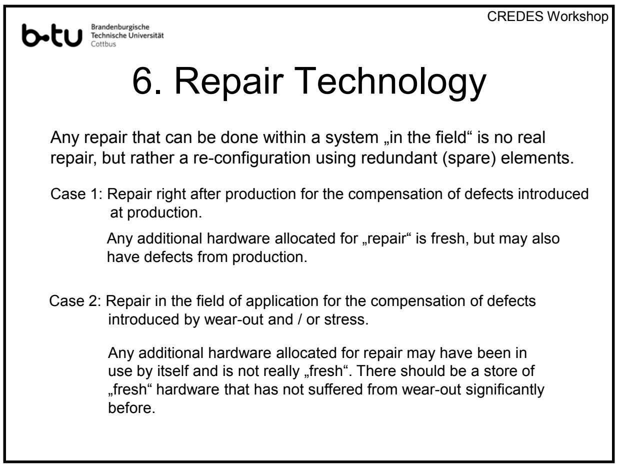
Find Activity Levels



CREDES Workshop

Does De-Stressing Work ?

Not all supply lines can be de-stressed by reverse currents. Clock lines May be less heavily affected, since on them the current can flow in two directions (at least in CMOS with high input resistance).



CREDES Workshop

6. Repair Technology

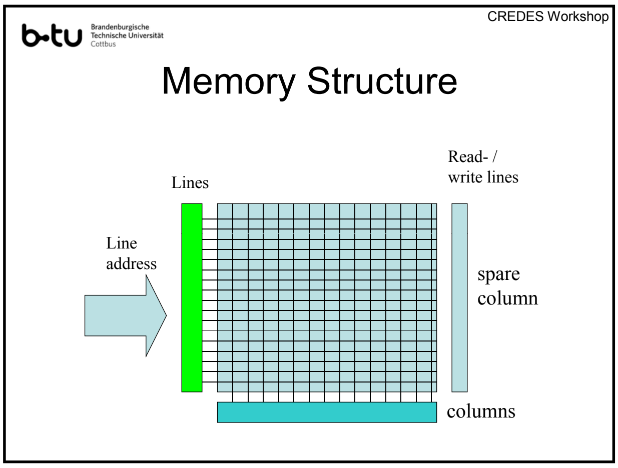
Any repair that can be done within a system „in the field“ is no real repair, but rather a re-configuration using redundant (spare) elements.

Case 1: Repair right after production for the compensation of defects introduced at production.

Any additional hardware allocated for „repair“ is fresh, but may also have defects from production.

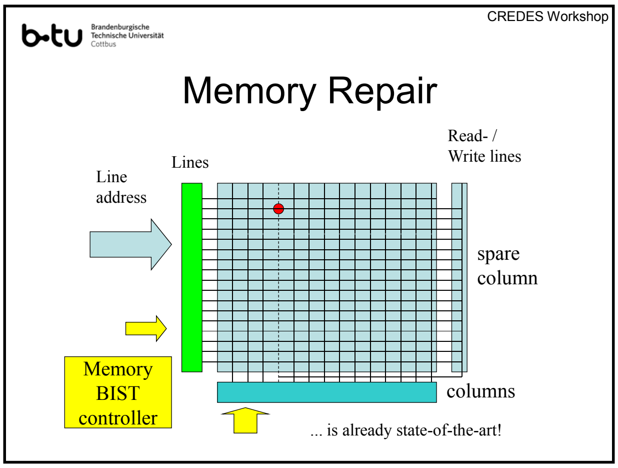
Case 2: Repair in the field of application for the compensation of defects introduced by wear-out and / or stress.

Any additional hardware allocated for repair may have been in use by itself and is not really „fresh“. There should be a store of „fresh“ hardware that has not suffered from wear-out significantly before.



CREDES Workshop

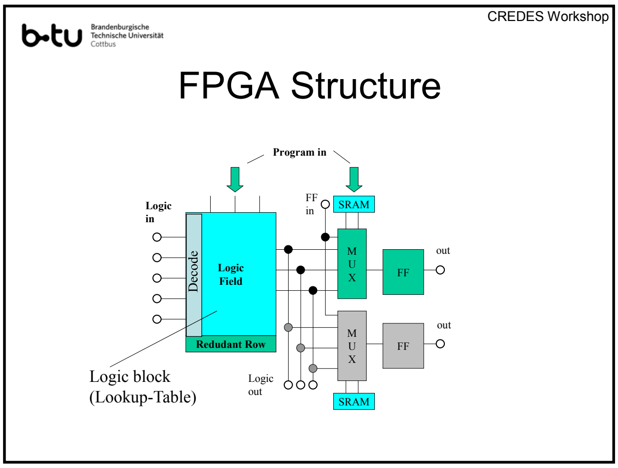
Memory Structure



CREDES Workshop

Memory Repair

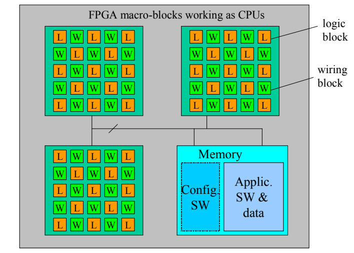
... is already state-of-the-art!



CREDES Workshop

FPGA Structure

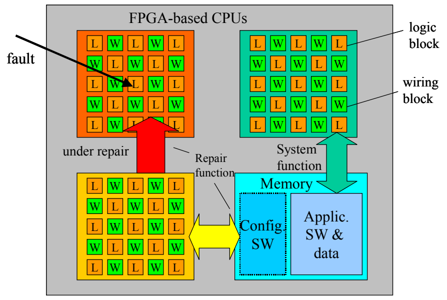
FPGA with TMR



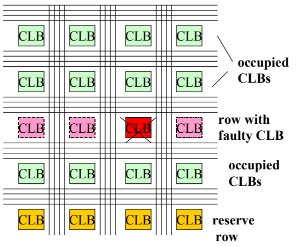
* e. g. proposed by McCluskey et al. IEEE Design and Test 2004

FPGA-based embedded controller: 8051

FPGA Self Repair

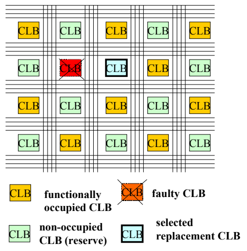


Regular Repair Scheme



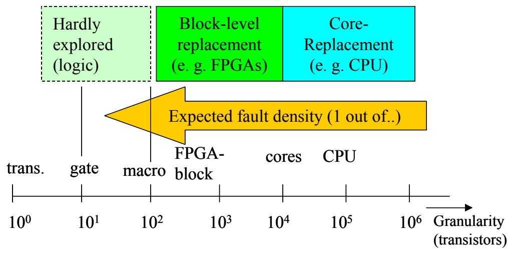
- ○ Little overhead for the re-configuration process
- ● Loss of many “good” CLBs for every fault

Irregular Repair Scheme

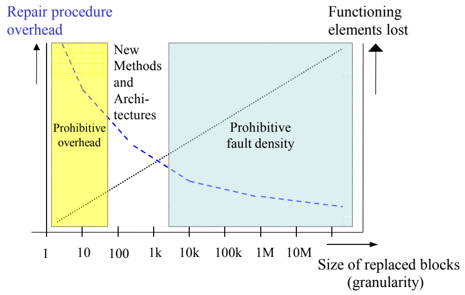


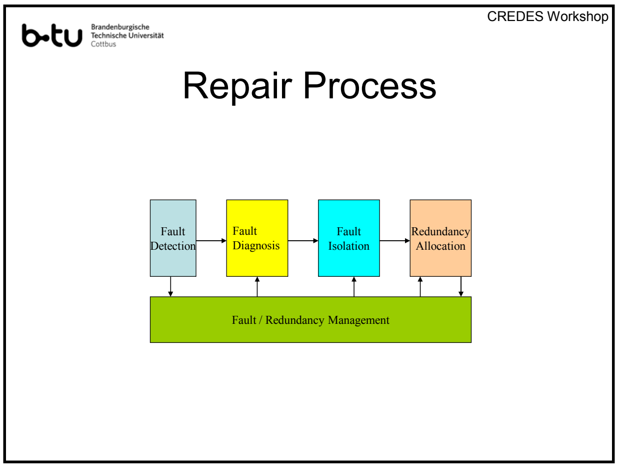
- ○ Minimum loss of functional CLBs
- ● High effort for re-wiring requires massive „embedded“ computing power (32-bit CPU, 500 MHz)

Granularity of Repair


Hardly explored (logic)
Block-level replacement (e. g. FPGAs)
Core-Replacement (e. g. CPU)
Expected fault density (1 out of..)
trans.
gate
macro
FPGA-block
cores
CPU
10⁰
10¹
10²
10³
10⁴
10⁵
10⁶
Granularity (transistors)

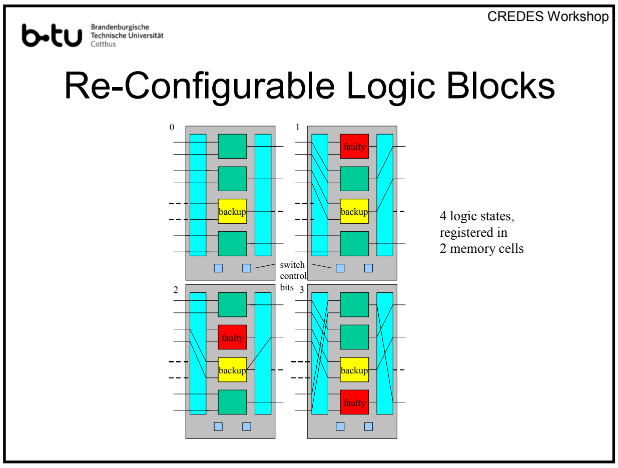
The Complexity Trap





CREDES Workshop

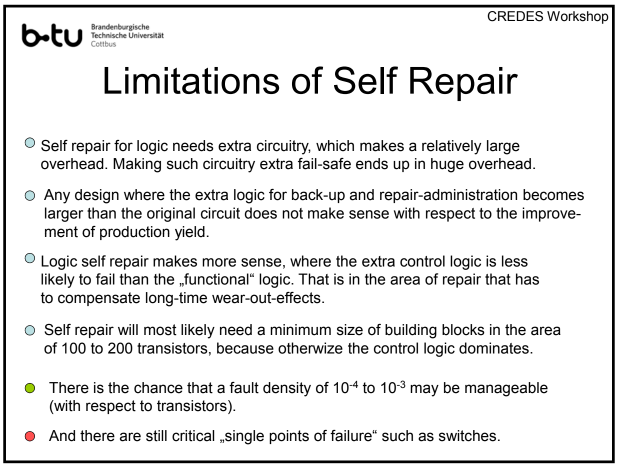
Repair Process



CREDES Workshop

Re-Configurable Logic Blocks

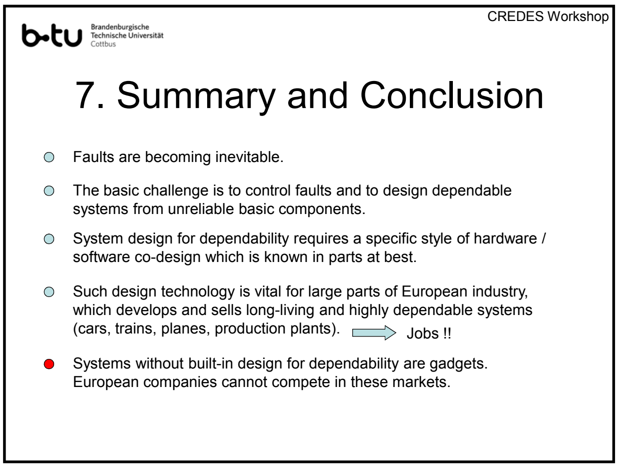
4 logic states,
registered in
2 memory cells



CREDES Workshop

Limitations of Self Repair

- ○ Self repair for logic needs extra circuitry, which makes a relatively large overhead. Making such circuitry extra fail-safe ends up in huge overhead.
- ○ Any design where the extra logic for back-up and repair-administration becomes larger than the original circuit does not make sense with respect to the improvement of production yield.
- ○ Logic self repair makes more sense, where the extra control logic is less likely to fail than the „functional“ logic. That is in the area of repair that has to compensate long-time wear-out-effects.
- ○ Self repair will most likely need a minimum size of building blocks in the area of 100 to 200 transistors, because otherwize the control logic dominates.
- ● There is the chance that a fault density of 10^{-4} to 10^{-3} may be manageable (with respect to transistors).
- ● And there are still critical „single points of failure“ such as switches.



CREDES Workshop

7. Summary and Conclusion

- ○ Faults are becoming inevitable.
- ○ The basic challenge is to control faults and to design dependable systems from unreliable basic components.
- ○ System design for dependability requires a specific style of hardware / software co-design which is known in parts at best.
- ○ Such design technology is vital for large parts of European industry, which develops and sells long-living and highly dependable systems (cars, trains, planes, production plants). ⇨ Jobs !!
- ● Systems without built-in design for dependability are gadgets. European companies cannot compete in these markets.