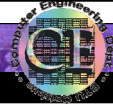


# EDA Technology Gaps and Related Work

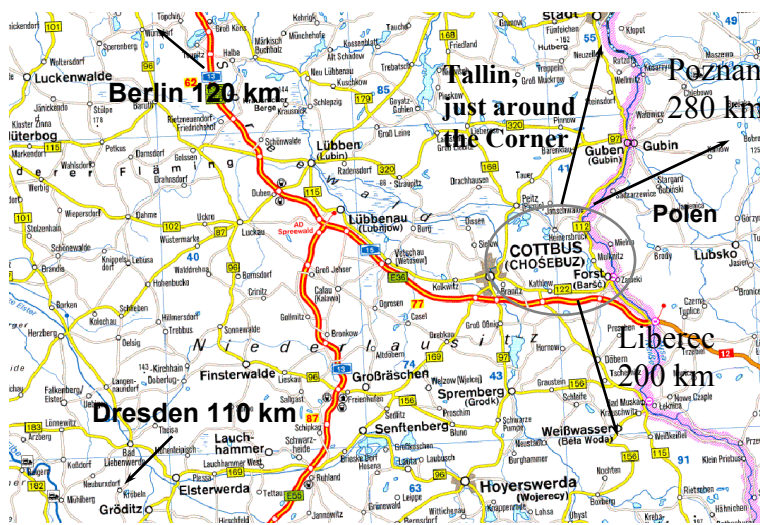
Heinrich Theodor Vierhaus

BTU Cottbus

Computer Engineering Group



## Cottbus on the Map





## Outline

1. Introduction: Changing EDA Perspectives and Gaps
2. The Hit List of Faults and Errors
3. High Level Design
4. Test Technology
5. Physical and Logic Design
6. Built-in Self Repair
7. What Needs to be Done



## 1. Changing EDA Perspectives

Early 1980: Hand-crafted ASIC (Mead-Conway)

Late 1980s: First IP use (Cells), Test Technology

Early 1990s: The Silicon Compiler Age

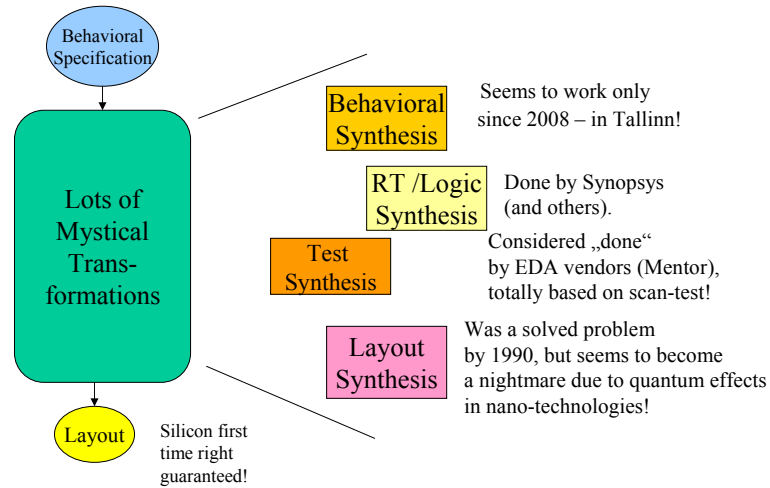
Late 1990s: Embedded Processors, HW / SW Co-Design

Early 2000s: FPGAs Come Off Age, SystemC

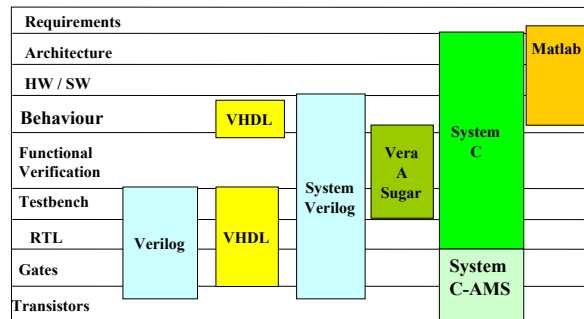
Late 2000s: Power-Aware Design, System Dependability

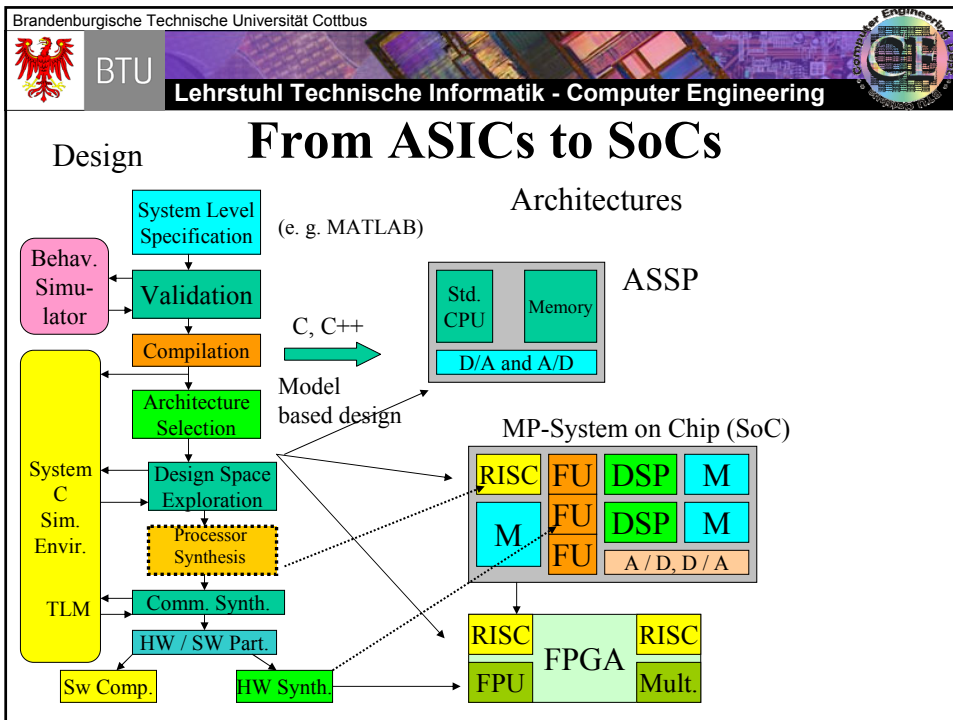


## The Silicon Compiler Perspective



## Levels and Languages





- Brandenburgische Technische Universität Cottbus  
 BTU  
 Lehrstuhl Technische Informatik - Computer Engineering
- ## Tendencies in EDA
- The basic functionality is defined by embedded processors with *embedded* software.
  - ASICs are becoming an endangered species except for high-performance / high-volume applications.
  - FPGAs are gaining an almost universal applicability as ASIC-or even SoC replacements due to embedded CPUs and / or functional units.
  - System design tools develop into standard chains of solutions (MatLab-SystemC-VHDL).
  - So everything is fine – except for the problem of *faults*.



## University Research Strategy

Alternative 1: Solve problems that have been solved already a bit or much better.

**Pre-condition:** You must have good friends in the EDA industry.

**Boundary conditions:** Nice life, lots of industrial acknowledgment.

**Advantage:** Lots of accepted papers, lots of money.

**Shortcomings:** No first solutions of real hard problems

Alternative 2: Try to solve unsolvable problems.

**Pre-condition:** You must be able to work hard.

**Boundary conditions:** Blood, sweat and tears.

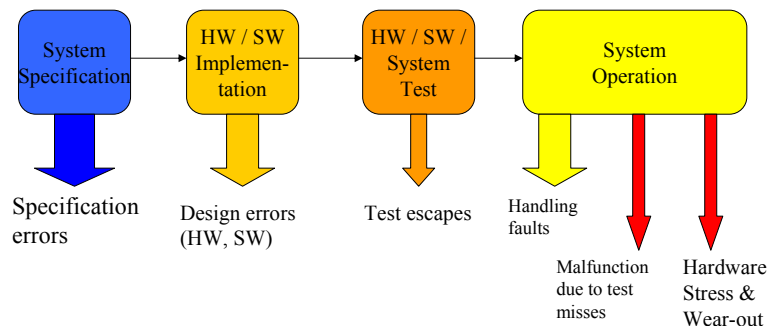
**Advantage:** Possibly real leap ahead in technology, real good people coming out!

**Shortcomings:** Lots of money are highly unlikely.



## 2. Origins of Faults and Errors

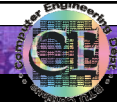
System Development and Evolution





## Bottlenecks

- Faults in the initial system specification.  
For example: Pilots are not able to override a control function which is faulty itself under side-conditions not thought of by the system designers.
- Timing verification: It seems to be difficult to prove that a system responds in-time under all side conditions.
- Physical Design: Models used for nano-devices in physical design are becoming questionable und unreliable.
- Integration of aspects such as faults, errors and exceptions into the system design flow.
- Validation of a system's behavior under fault conditions caused be specific elements & devices (bus, memory).

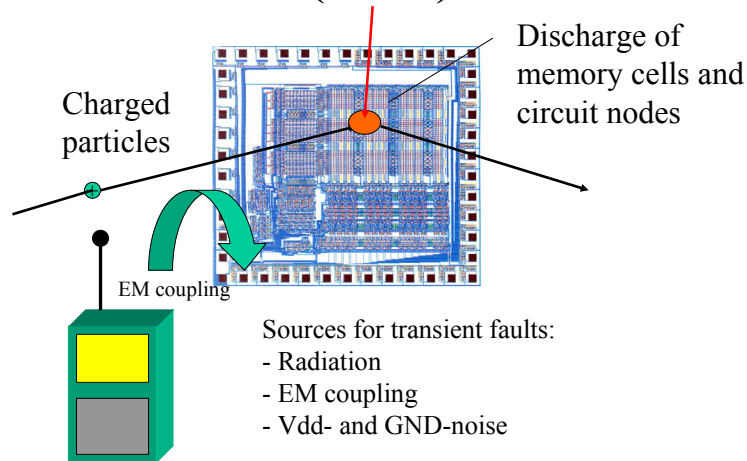


## Learnt from Real Life\*

- Automotive industry needs high-end microelectronics based on complex nano-electronic circuits.  
Example: Image recognition for driver assistance systems
- Faults and errors originating from embedded software are not the main source or errors any more.
- Fault effects on silicon are becoming a major issue, both with respect to transient and to permanent faults.



## Transient Faults and Single Event Upsets (SEUs)



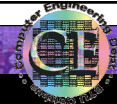
## Nanostructure Problems

- Individual device characteristics such as  $V_{th}$  are more dependent on statistical variations of underlying physical features such as doping profiles.
- A significant share of basic devices will be „out of specs“ and needs a replacement by backup elements for yield improvement after production.
- As smaller features mean higher stress (field strength, current density), also early failures „in the field“ are more likely and must be compensated.
- Transient error recognition and compensation „in time“ is becoming a must due to e. g. charged particles that can discharge circuit nodes.

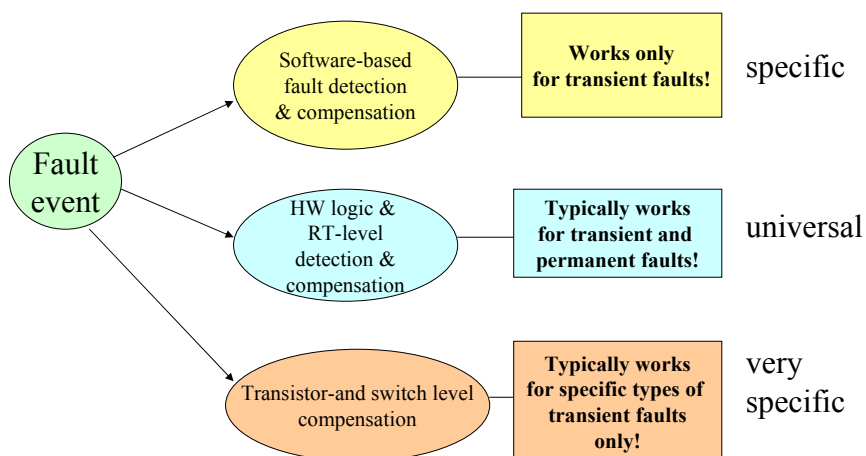


## Nanostructure Problems

- Individual device characteristics such as  $V_{th}$  are more dependent on statistical variations of underlying physical features such as doping profiles. **Primary Relevance: Yield**
- A significant share of basic devices will be „out of specs“ and needs a replacement by backup elements for yield improvement after production. **Primary Relevance: Yield**
- As smaller features mean higher stress (field strength, current density), also early failures „in the field“ are more likely and must be compensated. **Primary Relevance: Lifetime**
- Transient error recognition and compensation „in time“ is becoming a must due to e. g. charged particles that can discharge circuit nodes. **Primary Relevance: Reliability**



## Fault Tolerant Computing

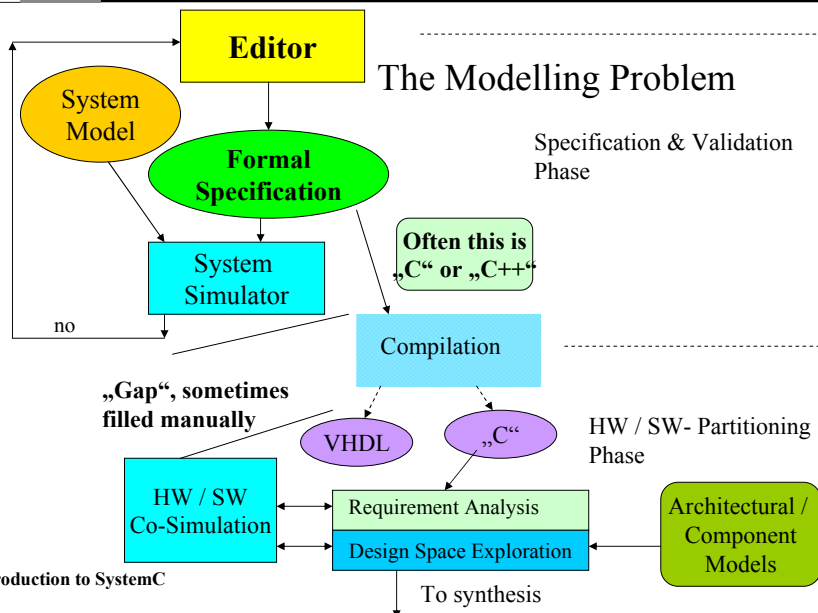
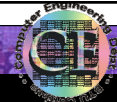






### 3. High-Level Design

Lots of models, languages, abstractions.



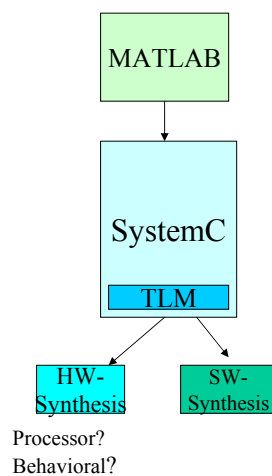


## Tendencies

- Many high-level specification tools have „C“ or „C++“ outputs.
- If the specification itself is in „C“ or „C++“, compiled-code simulation becomes an option for validation of the specification by field trials.
- „C“ specifications can immediately be executed on single Processor low-end target architectures.
- Recent SystemC extensions towards analog-mixed signal (SystemC-AMS) and transaction level modelling allow to model and simulate real MP-SoC systems.

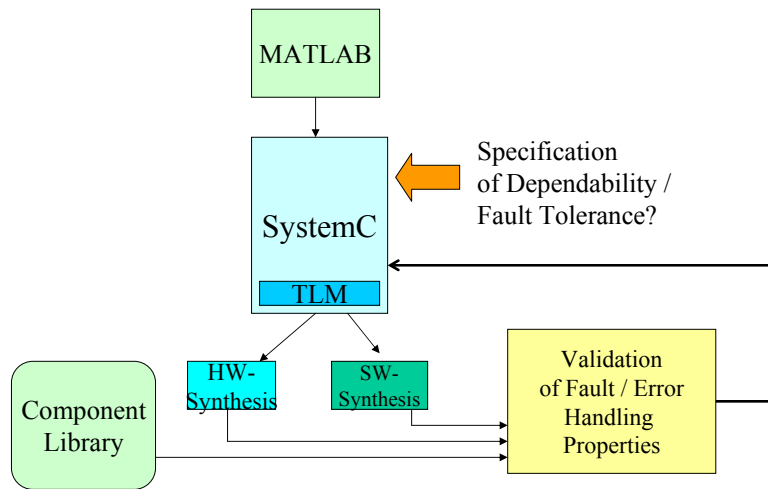


## Standard Design Environment ?





## What is Missing?



## 4. Test Technology

### Traditional:

- Semiconductor companies want sophisticated **scan test** for cost reduction in production test.
- Aspects of system dependability „in the field“ are not important.

### More recently:

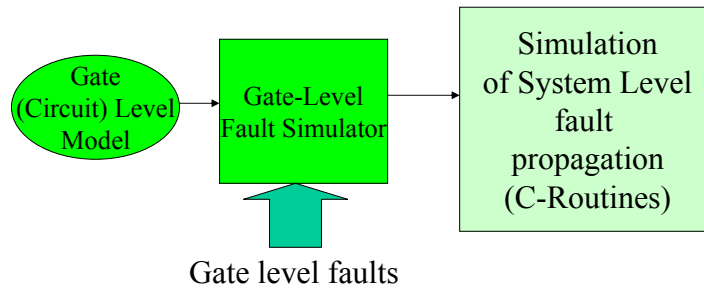
- Fault diagnosis in production testing becomes a must.
- Test escapes due to un-modeled faults or due to inherent shortcomings of (dynamic) scan test play a role.

### Really new:

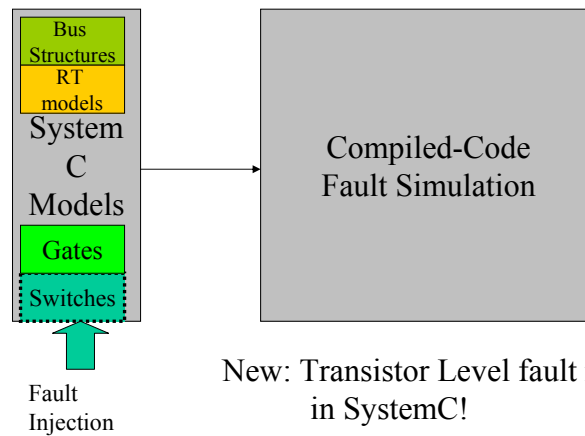
- In-system test using production test & (extra) circuitry.
- Fault tolerance, also to compensate remaining faults.
- Built-in self test with diagnosis and *self repair*!

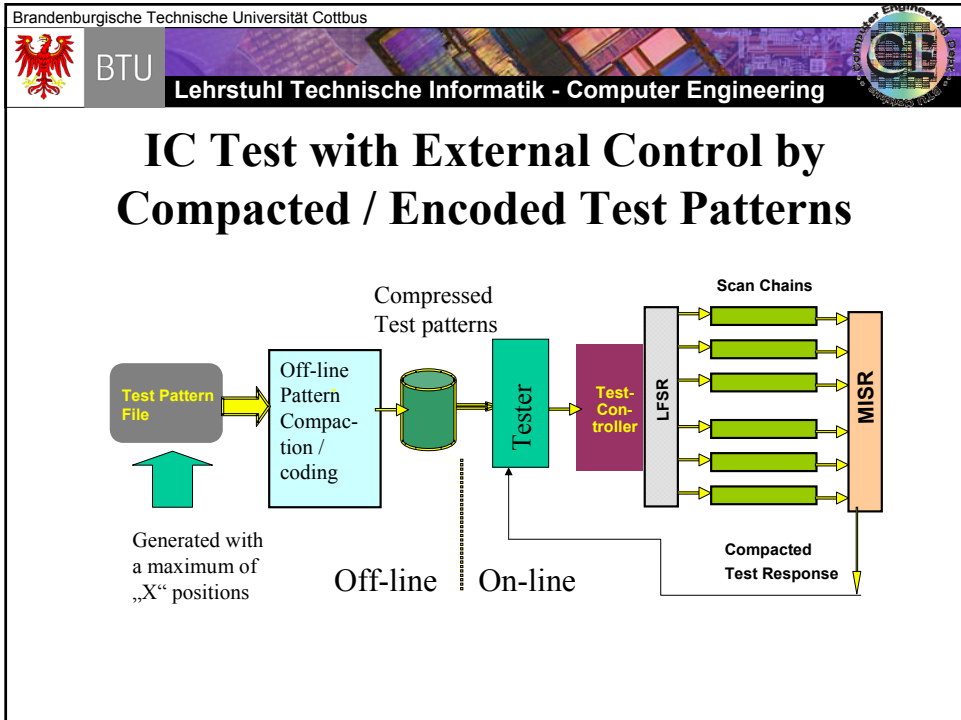
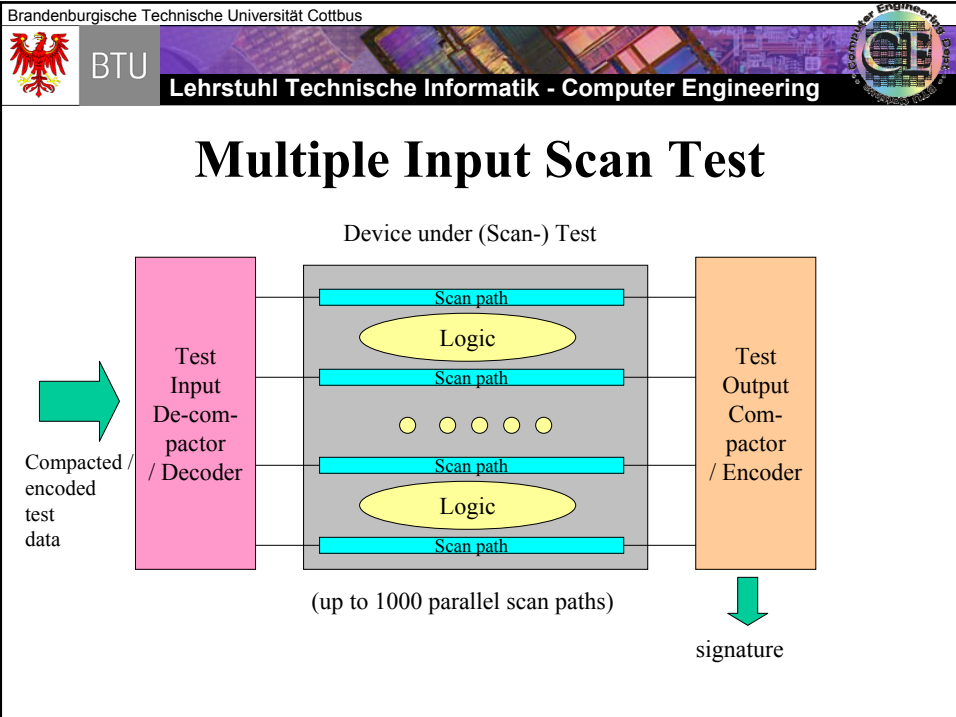


## Hierarchical Fault Simulator FSIM



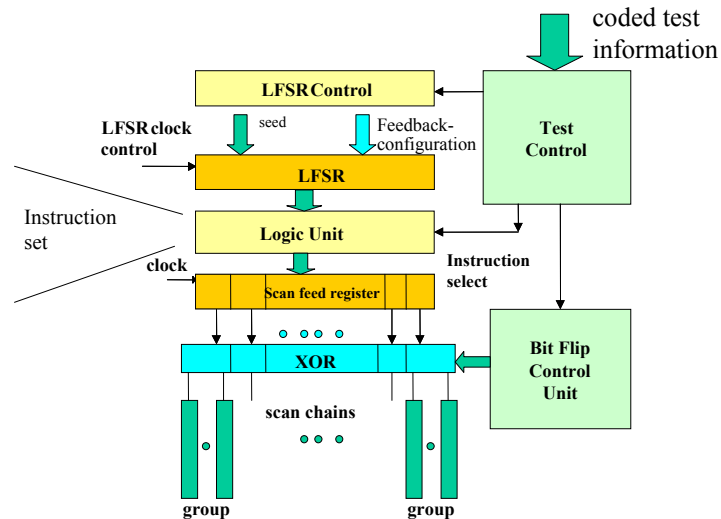
## SystemC-Based Fault Analysis







## Test-Controller for Scan-Test

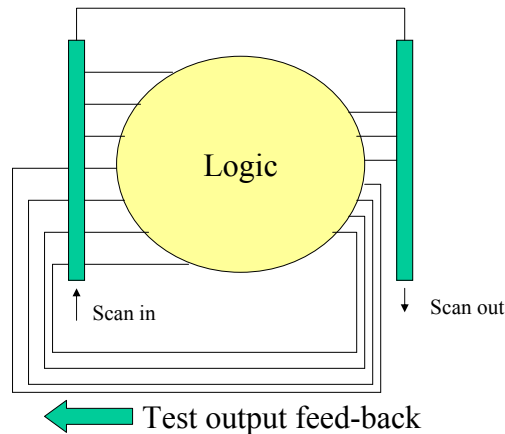


## Scan Test Problems

- „Normal“ scan test can apply only a single pattern in real-time.
- Coverage of dynamic faults is uncertain at best.
- The scan process itself causes a high level of circuit activity which can result on overheating problems.



## Dynamic Scan Test (Broadside Test)

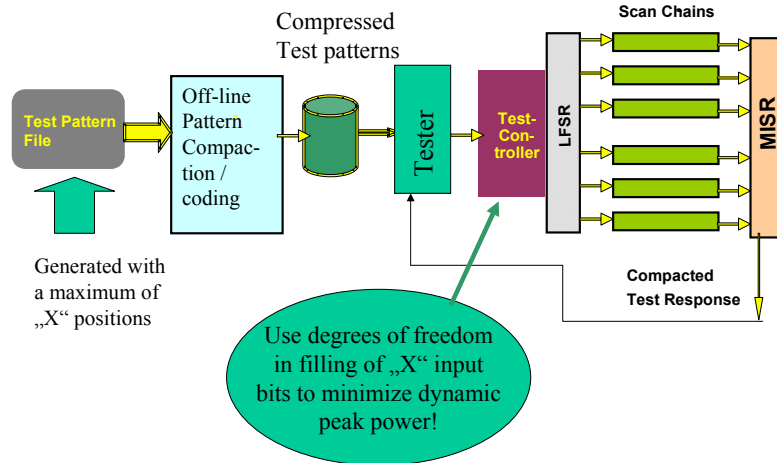


## Dynamic Scan Test Problems

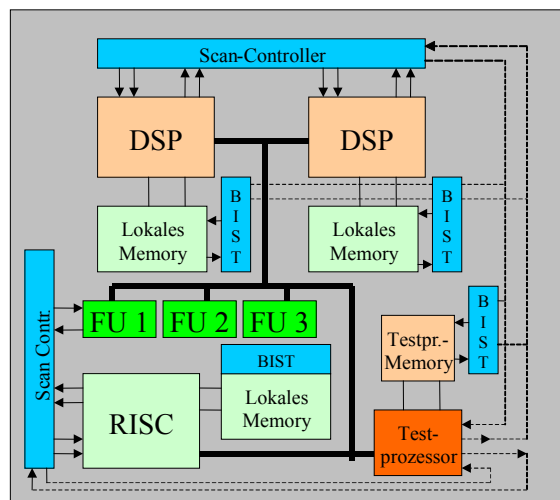
- Broadside test via feed-back of test output causes a high level of circuit activity.
- Switching activities may cause voltage drops on VDD and GND lines which are not experienced in functional operation.
- The circuit may exhibit (dynamic) faults which do not occur in normal operation.
  - ➔ Yield Loss!
- The circuit may also experience a high level of stress at test due to high peak currents and high power dissipation.



## Power-Optimised Scan Test



## Test Processor Concept for SoCs







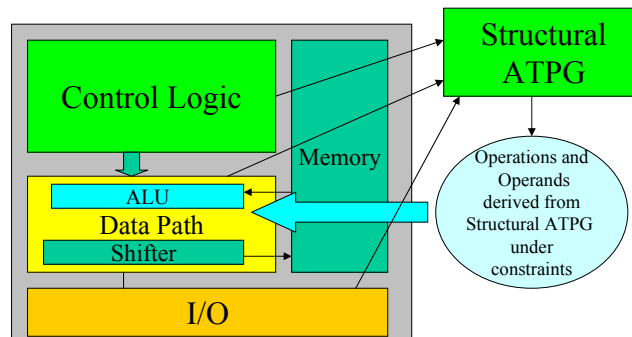
## Test Processor Concept

- Minimum sized RISC-processor, fully predictable operation (from about 5000 equivalent gates)
- Functional self test with high test coverage (static and dynamic)
- Special instructions to run internal registers as LFSR or MISR.
- Special I/O unit and special instructions to test static and dynamic faults on external bus structures.
- Optionally processor synthesis „upon demand“ by configurable VHDL files.



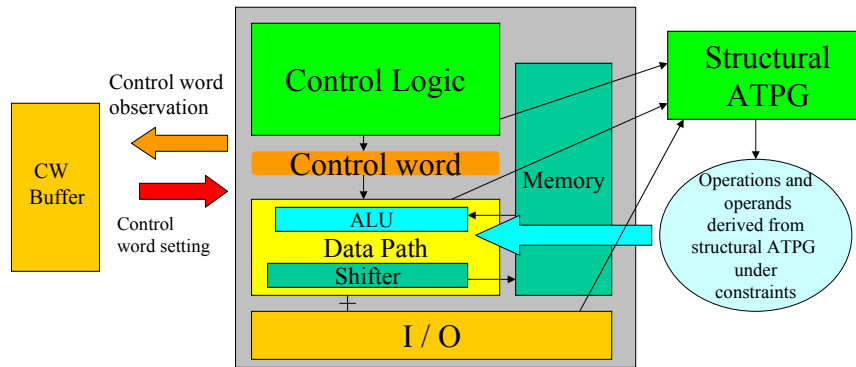
## Software-based BIST (SBIST)

Problem: Scan test triggers the wrong transitions!





## DFT for SBIST



Promises much enhanced fault coverage for control logic, requires modified architecture and instruction set!



## 5. Physical and Logic Design

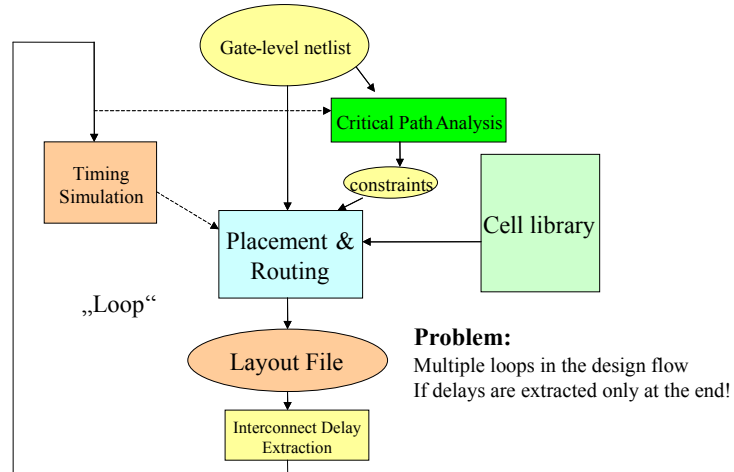
- Conventionally, logic design is followed by physical design. Logic design includes all necessary aspects of timing.
- From a structure size of about 200 nanometers, delays in (random) logic are mainly induced by signal delays on wires rather than transistor switching delays.
- With delays not yet known in logic design, timing validation needs to be done after physical design (place and route).



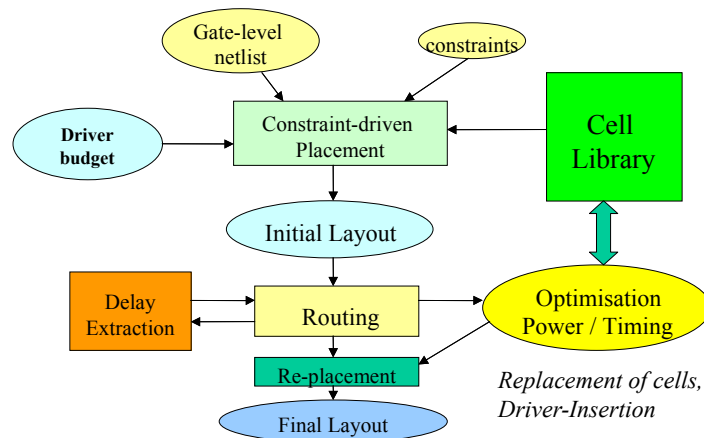
Cycles in the design flow, poorly converging design process with many loops!



## Design Flow „In the Loop“

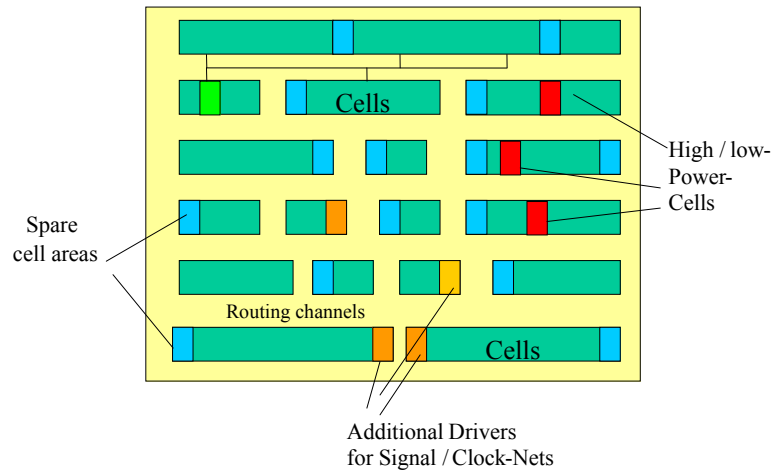


## Design Flow with „Local“ Timing and Power Optimization



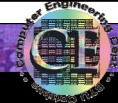


## Cell-based Design: Optimization

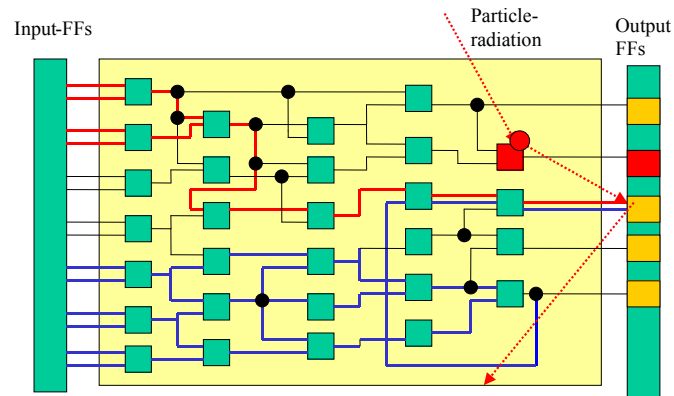


## New Tools for Physical Design

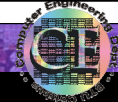
- Precise and efficient calculation of delays and slew-rates on interconnects.
- Extraction of truly sensitizable logic paths.
- Calculation of resulting path delays including internal gate delays and interconnects.



## Self-Testing and Fault Tolerant Flip-Flops



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



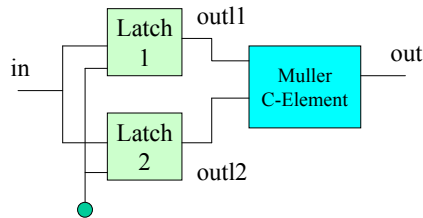
## Transient Faults

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



## Intel<sup>6</sup> Strategy



If both inputs are equal:  $out = out1, out2$

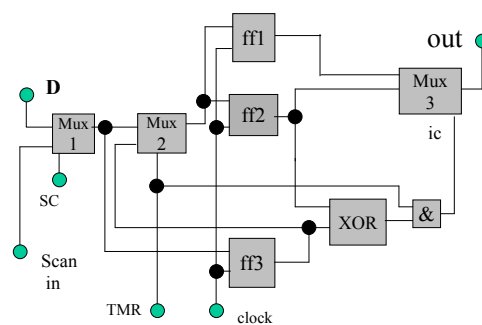
If both element are not equal:  $out = \text{previous} (out1, out2)$

Under local fault conditions on the latch outputs (one of 2 latches false), the C-element preserves the output condition from the „charge“ phase of the latch.

Effort: triple latches plus 2 transistors!



## Fault Tolerant Scan-Path FF Design



Mode 1: fault-tolerant functional / test mode

Mode 2: normal scan-mode (fault tolerant)

Mode 3: dynamic scan mode (not fault tolerant)

Mode 4: dynamic scan test mode (not fault tolerant)



## 6. Built-In Self Repair

- Device parameter fluctuations reduce IC production yield.
- Interconnects become unreliable due to effects such as metal migration inducing wear-out effects.

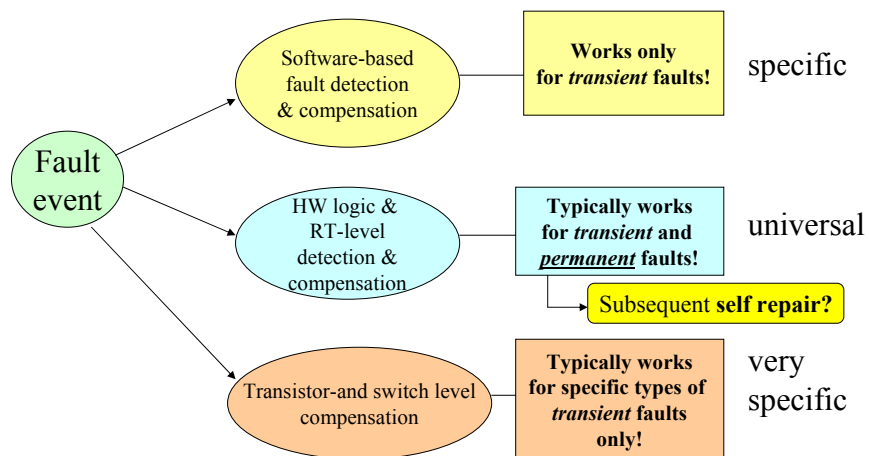
Technologies such as on-line error recognition or triplication (TMR) can compensate such faults, but then lose their ability to handle additional transient faults!



Off-line self repair!

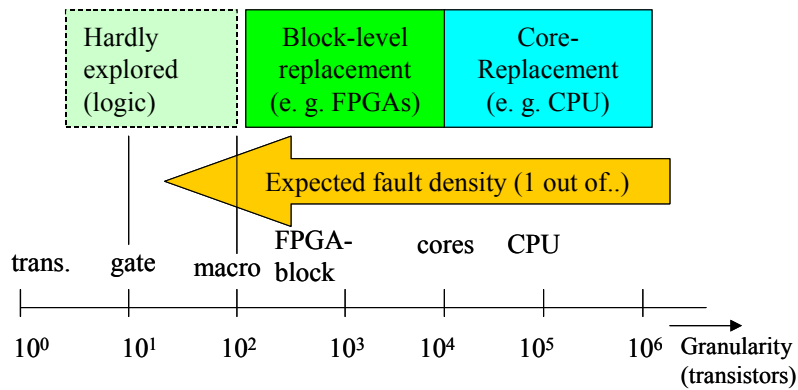


## Fault Tolerant Computing



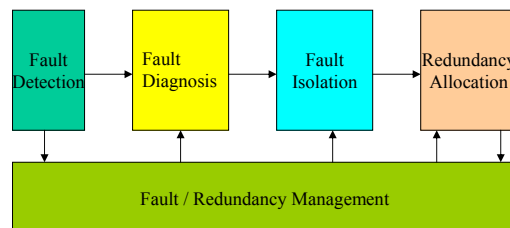


## Granularity of Repair



## Built-in Self Repair (BISR)

- BISR is well understood for highly regular structures such as embedded memory blocks.
- BISR is much more complex to implement on irregular logic.  
*Logic regularity needs to be "introduced" for repair actions.*

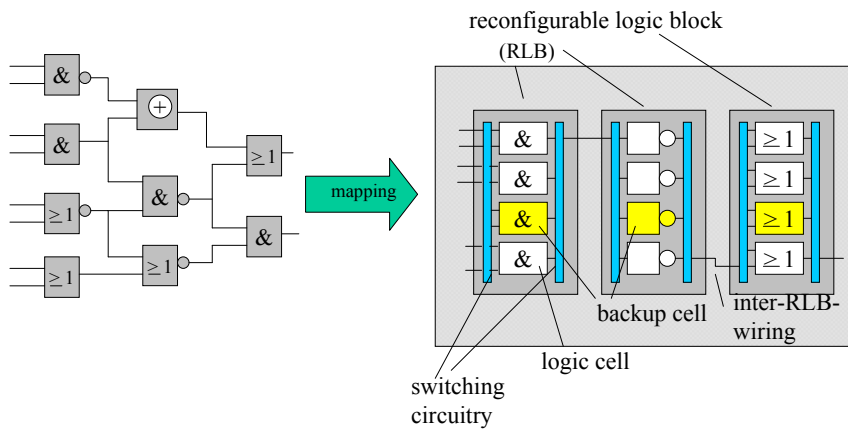


**Redundancy management must monitor faults, replacements, available redundancy and must also re-establish a „working“ system state after power-down states.**

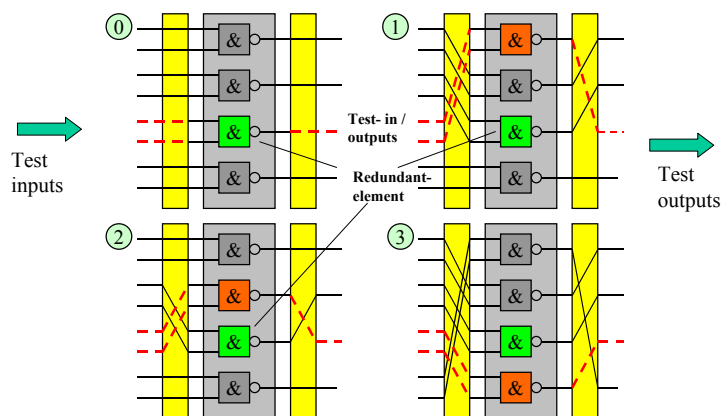




## Logic Repair Scheme



## Extended Switching Scheme

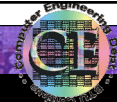


Scheme allows to test all basic elements via test- I/O access!

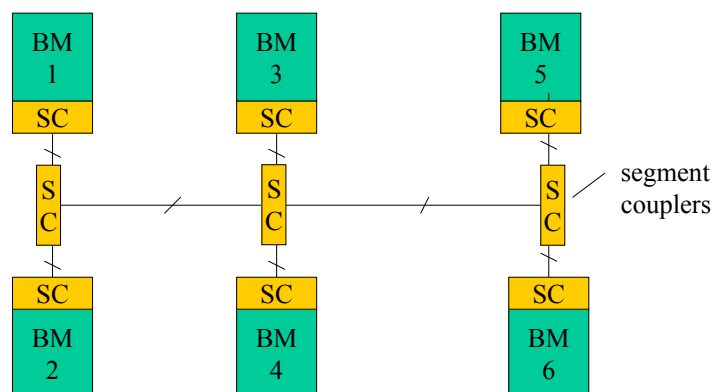


## Is Logic BISR feasible?

- Overhead is between about 40 % and 250 %, depending on the granularity of basic re-configurable logic blocks (RLBs).
- The overhead is clearly dominated by the number of signal switches and the additional control circuitry.
- A scheme that integrates local self test and control operations for re-configurations is possible.
- Extracting regular features from irregular logic for more efficient replacement strategies is feasible!



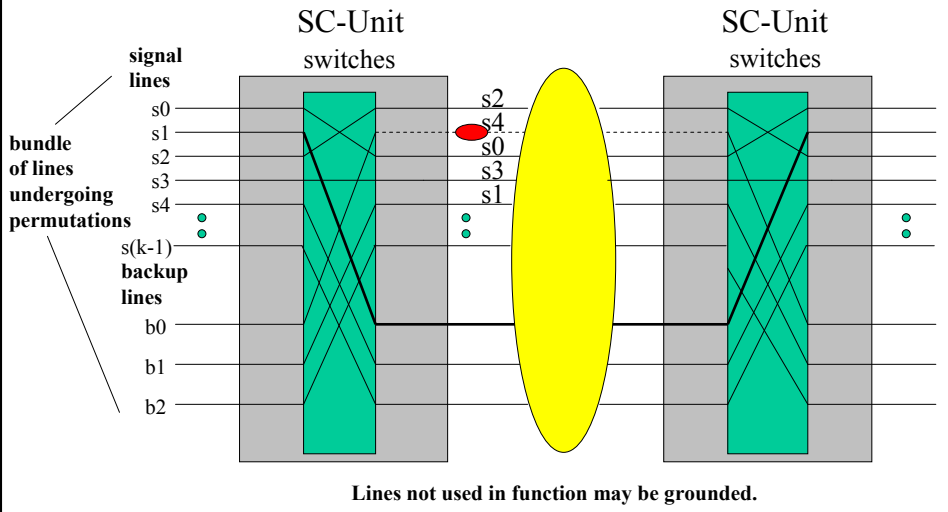
## Regular Interconnects: Bus Segmentation



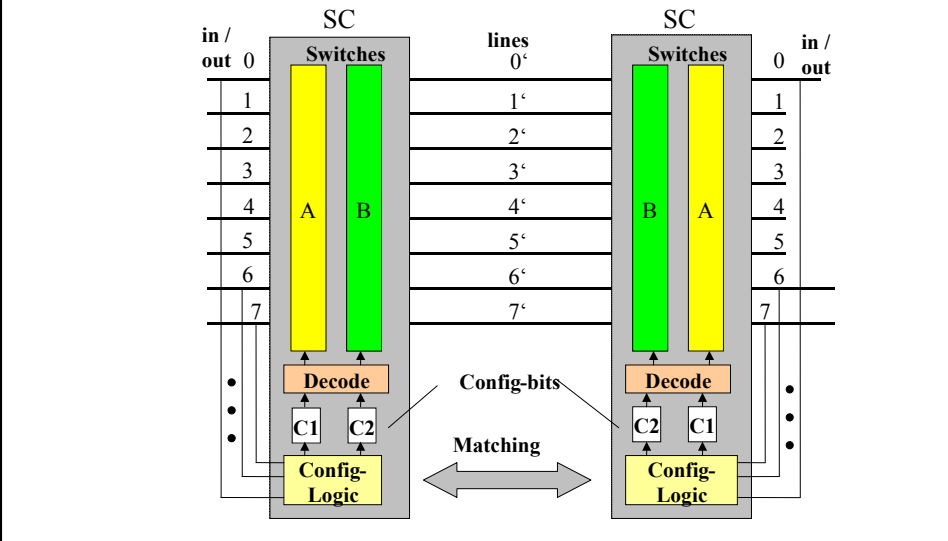
Structure the bus into segments that can be repaired individually!



## Switches and Backup Lines



## Repair Scheme for Regular Interconnects





## 7. What Needs to be Done

- Make reliability / dependability an integral part of design flows!
- Develop methods and tools that can predict fault behavior!
- Develop tools that can formally prove fault / error compensation!
- Integrate IC test and system test strategies
- Use multiple parallel functional units to implement „graceful degradation“ rather than functional breakdown.
- Find enough young bright people who are willing and able to do research in the field of  
*„Dependable HW / SW Systems“!!*