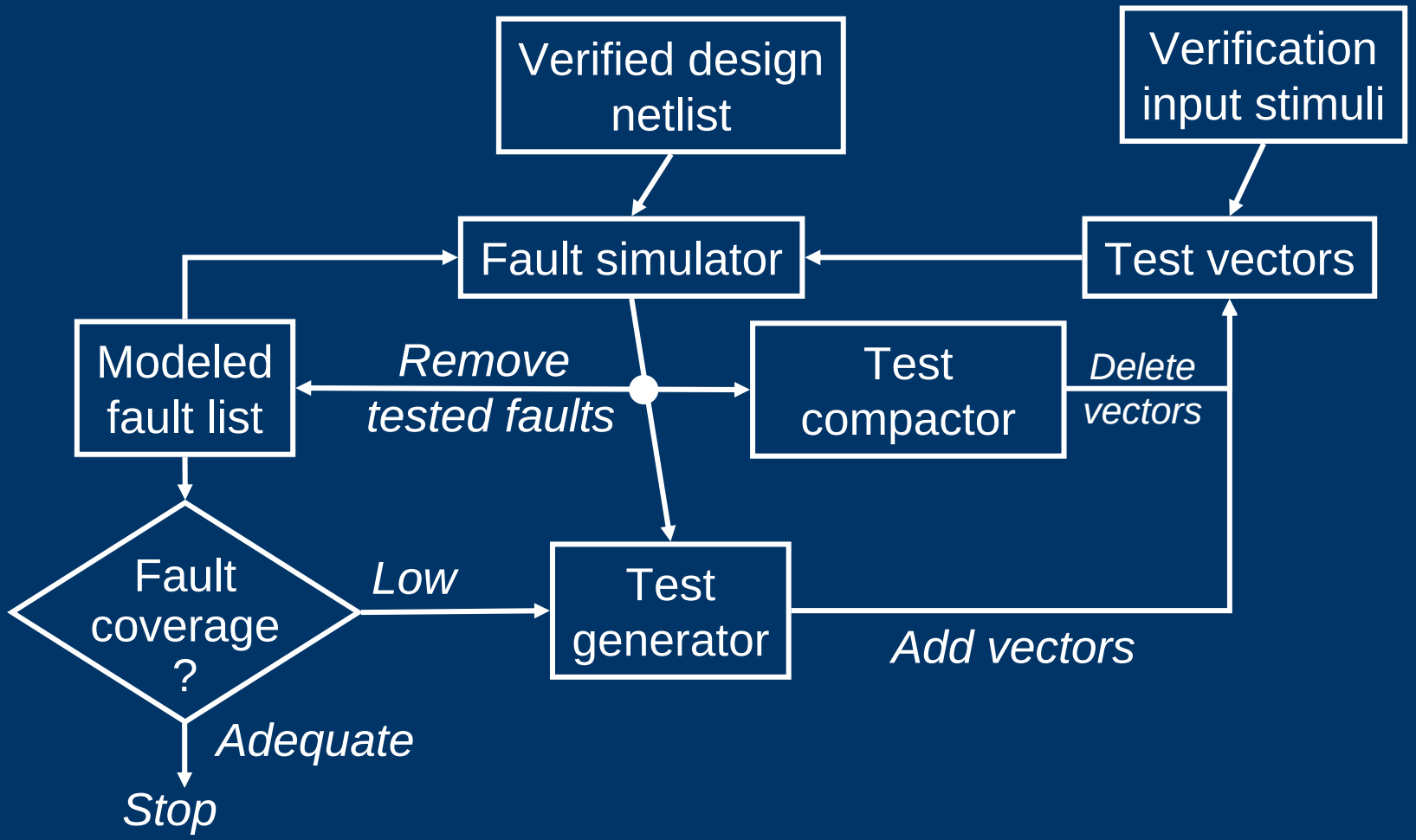
Lecture 3 - Fault Simulation and Fault Injection

- Fault simulation
- Fault simulation algorithms
  - Serial
  - Parallel
  - Deductive
  - Random Fault Sampling
- Fault Injection
  - ASIC
  - FPGAs

Problem and Motivation

- Fault simulation Problem: Given
  - A circuit
  - A sequence of test vectors
  - A fault model
  - Determine
    - Fault coverage - fraction (or percentage) of modeled faults detected by test vectors
    - Set of undetected faults
- Motivation
  - Determine test quality and in turn product quality
  - Find undetected fault targets to improve tests

Fault simulator in a VLSI Design Process



Fault Simulation Scenario

- Mostly single stuck-at faults
- Sometimes stuck-open, transition, and path-delay faults; analog circuit fault simulators are not yet in common use
- Equivalence fault collapsing of single stuck-at faults
- Fault-dropping -- a fault once detected is dropped from consideration as more vectors are simulated; fault-dropping may be suppressed for diagnosis
- Fault sampling -- a random sample of faults is simulated when the circuit is large

Fault Simulation Algorithms

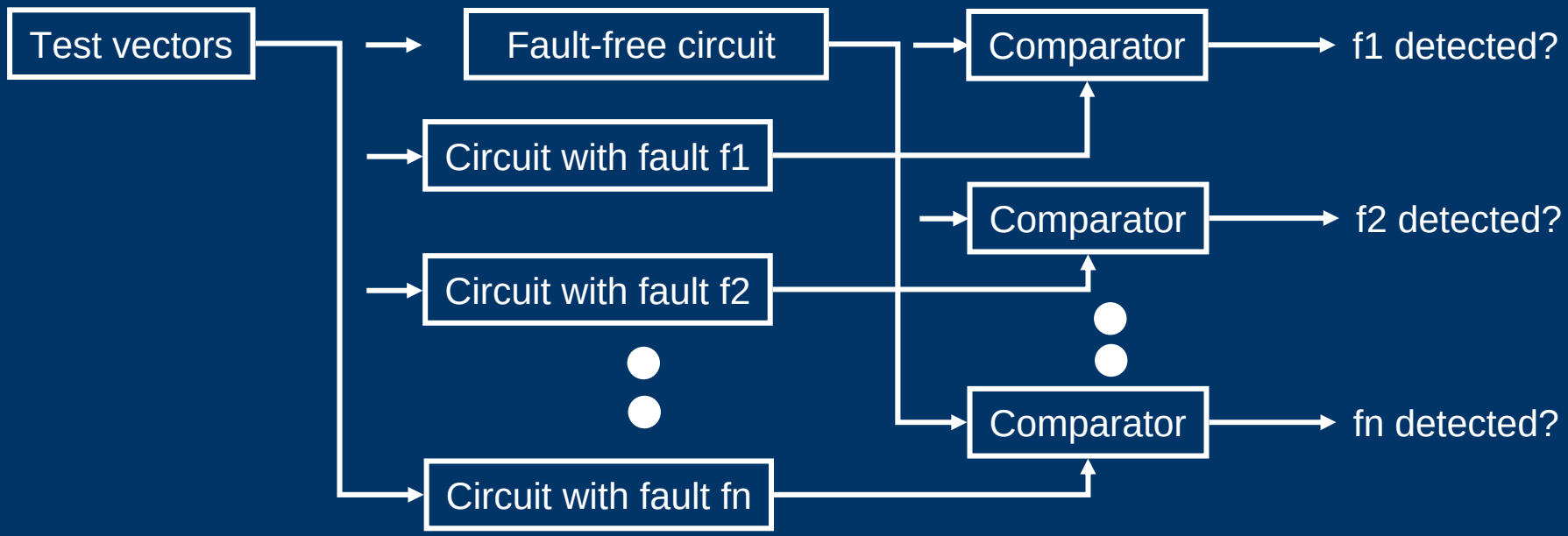
- Serial
- Parallel
- Deductive
- Random Fault Sampling

Serial Algorithm

- Algorithm: Simulate fault-free circuit and save responses. Repeat following steps for each fault in the fault list:
  - Modify netlist by injecting one fault
  - Simulate modified netlist, vector by vector, comparing responses with saved responses
  - If response differs, report fault detection and suspend simulation of remaining vectors
- Advantages:
  - Easy to implement; needs only a simulator, less memory
  - Most faults, including analog faults, can be simulated

Serial Algorithm (Cont.)

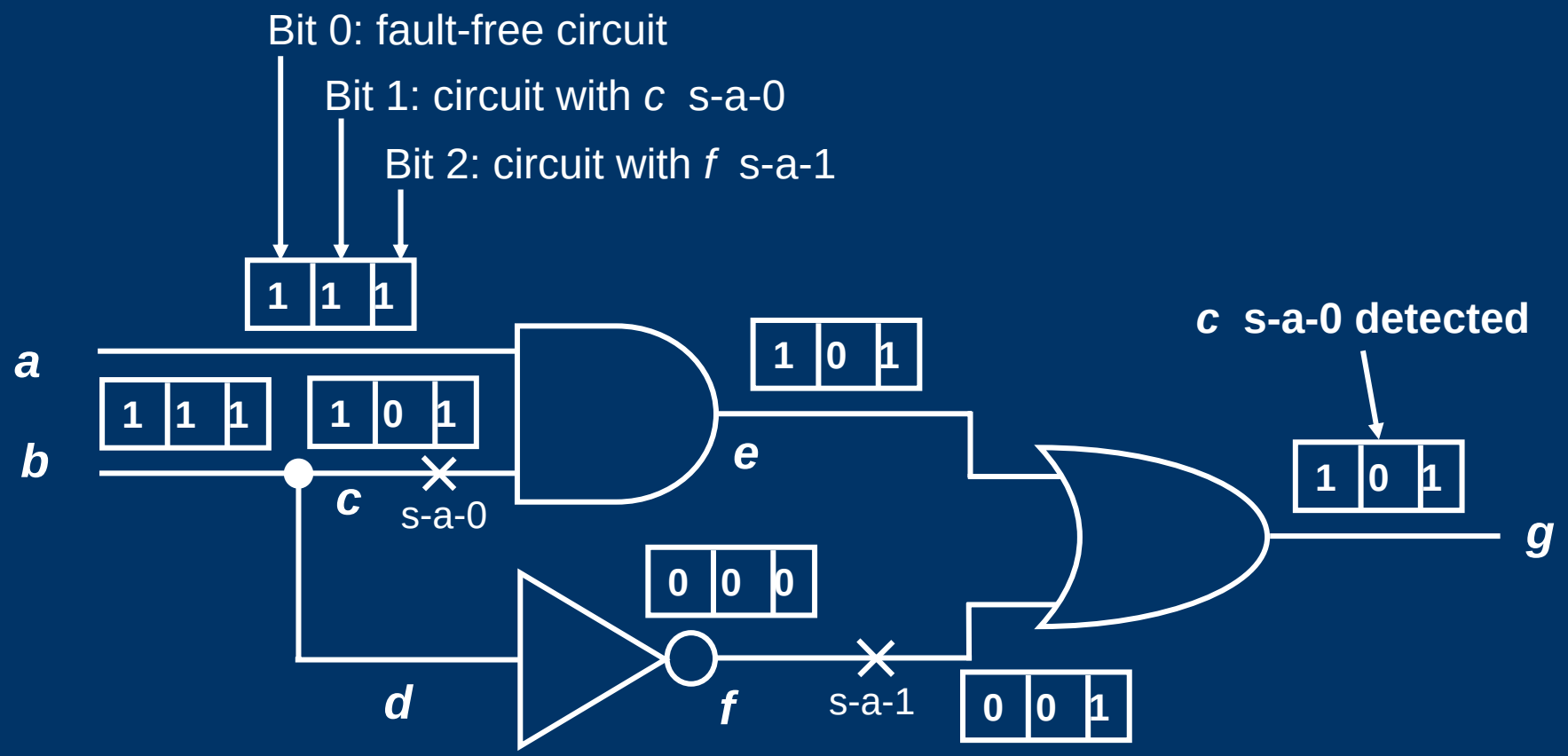
- Disadvantage: Much repeated computation; CPU time prohibitive for VLSI circuits
- Alternative: Simulate many faults together



Parallel Fault Simulation

- Exploits inherent bit-parallelism of logic operations on computer words
- Storage: one word per line for two-state simulation
- Multi-pass simulation: Each pass simulates *w*-1 new faults, where *w* is the machine word length
- Speed up over serial method ~ *w*-1

Parallel Fault Sim. Example

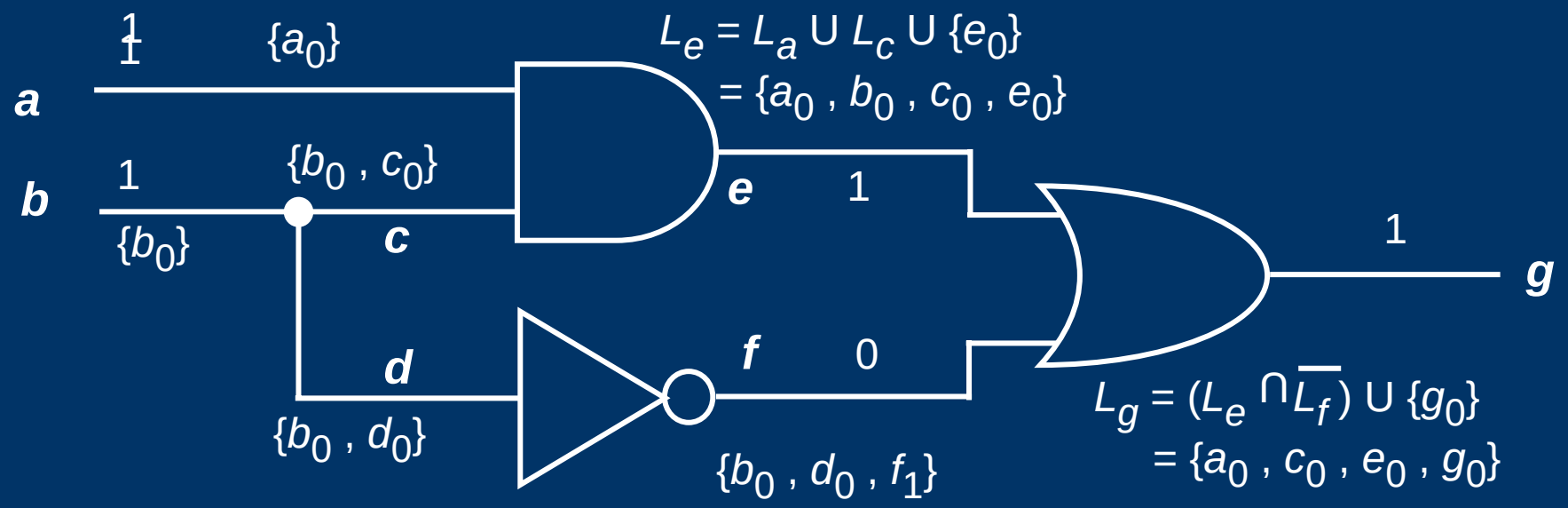


Deductive Fault Simulation

- One-pass simulation
- Each line k contains a list L_k of faults detectable on k
- Following simulation of each vector, fault lists of all gate output lines are updated using set-theoretic rules, signal values, and gate input fault lists
- PO fault lists provide detection data

Deductive Fault Simulation Example

Notation: L_k is fault list for line k
k_n is s-a-n fault on line k



Fault Sampling

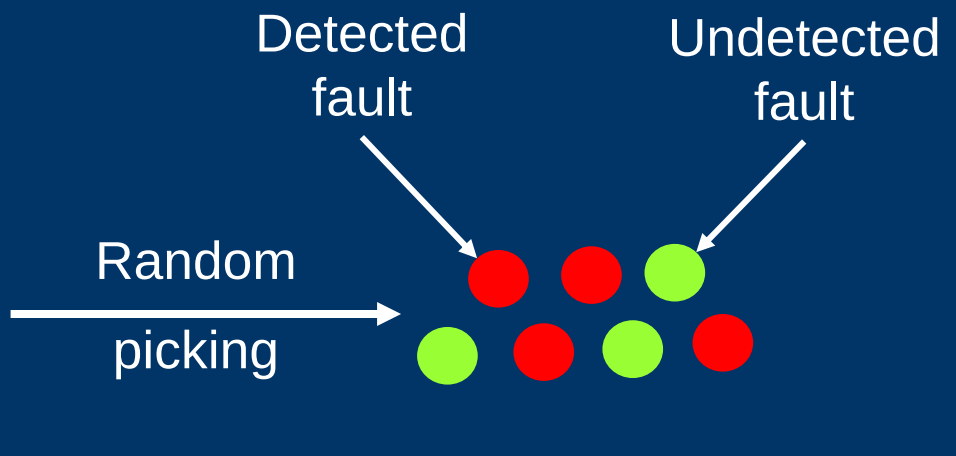
- A randomly selected subset (sample) of faults is simulated.
- Measured coverage in the sample is used to estimate fault coverage in the entire circuit.
- Advantage: Saving in computing resources (CPU time and memory.)
- Disadvantage: Limited data on undetected faults.

Motivation for Sampling

- Complexity of fault simulation depends on:
  - Number of gates
  - Number of faults
  - Number of vectors
- Complexity of fault simulation with fault sampling depends on:
  - Number of gates
  - Number of vectors

Random Sampling Model

All faults with a fixed but unknown coverage



N_p = total number of faults

(population size)

C = fault coverage (unknown)

N_s = sample size

$N_s << N_p$

c = sample coverage
(a random variable)

Example

A circuit with 39,096 faults has an actual fault coverage of 87.1%.
The measured coverage in a random sample of 1,000 faults is 88.7%.
CPU time for sample simulation was about 10% of that for all faults.

Fault Injection

- Problem and Motivation
- ASIC
  - performed by simulation or
  - emulation to speed up the process
- FPGAs
  - directly into the bitstream

Problem and Motivation

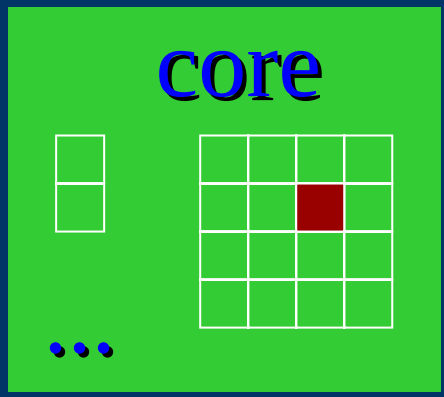
- Fault injection Problem: Given
  - A circuit
  - An application
  - A fault model
  - Determine
    - Behavior of faulty circuit
- Motivation
  - Determine product safety
  - Find unsafe states to improve product safety
  - Redesign product for fault tolerance

Fault Injection

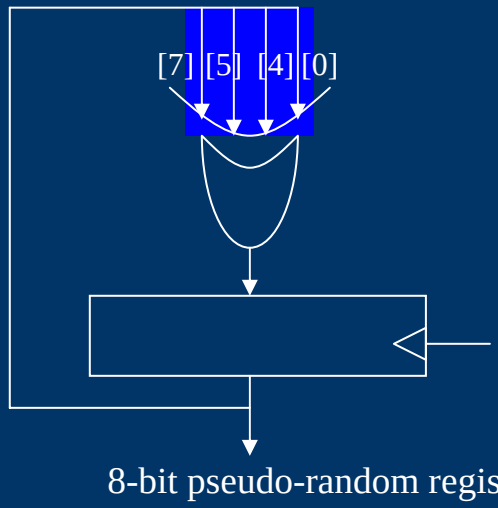
General: ASIC performed by simulation or emulation to speed up the process

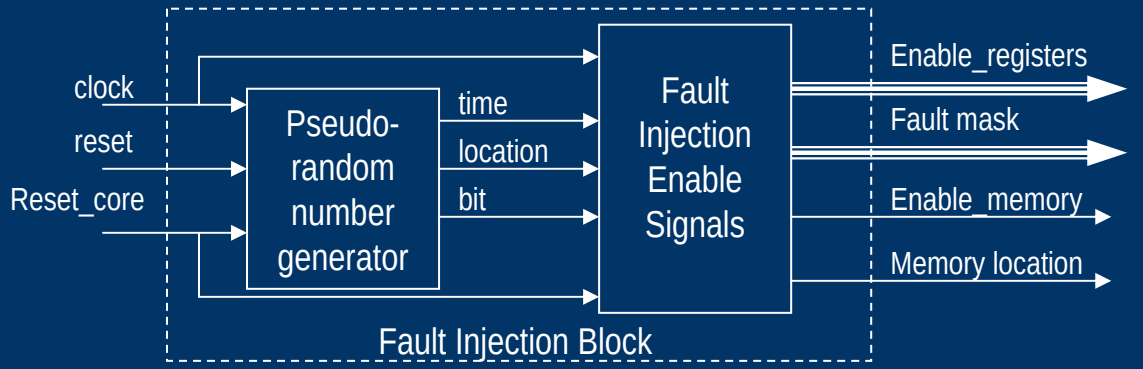
Fault Injection Control Block

- When should a fault be inserted?



- Where should a fault be inserted?

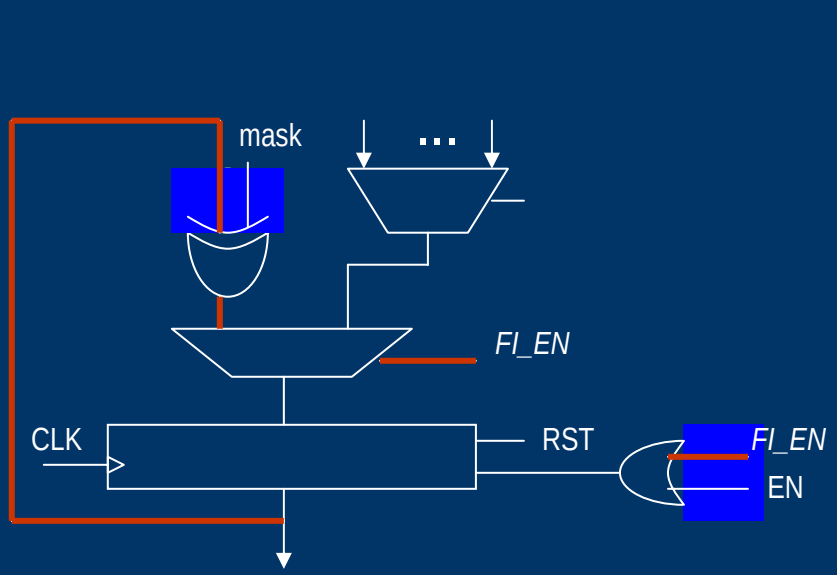


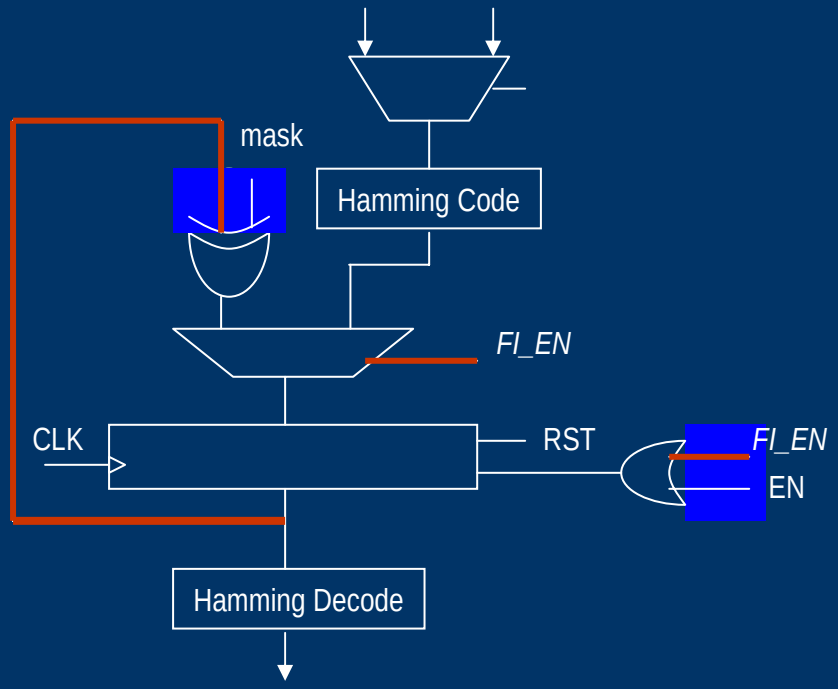


Device Under Test Core

Registers

- Special paths are required in the high-level description to ensure the full controllability of a fault insertion.

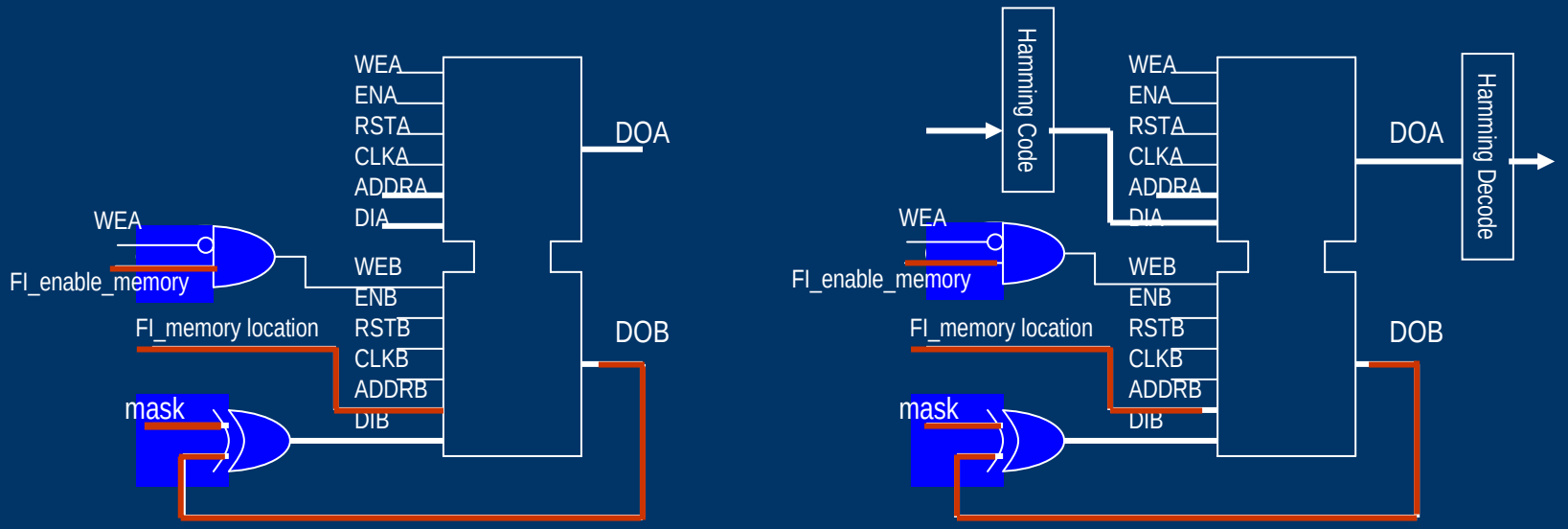




Device Under Test Core

Memory

- Dual-port memories allows high flexibility with low extra logic.
- Good fault model approximation: the faults can be injected in the memory at any time (all micro-controller states) without perturbing the normal operation.

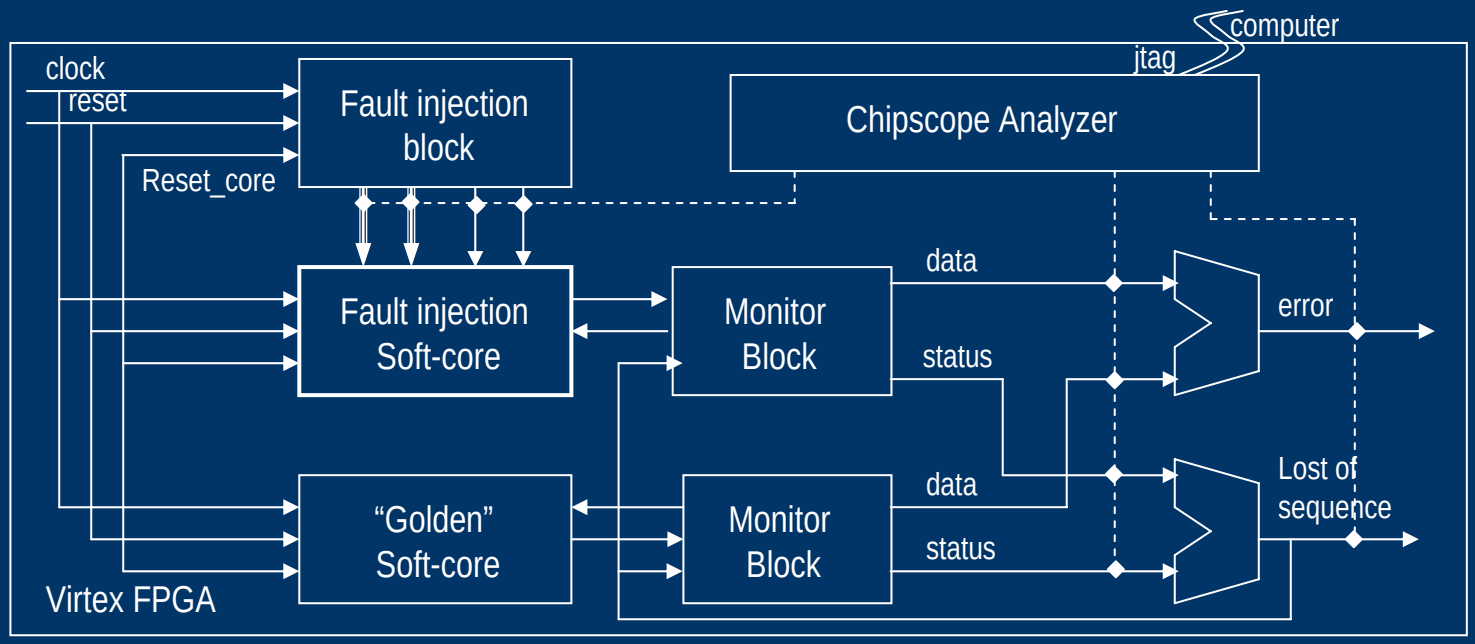


Monitor Block

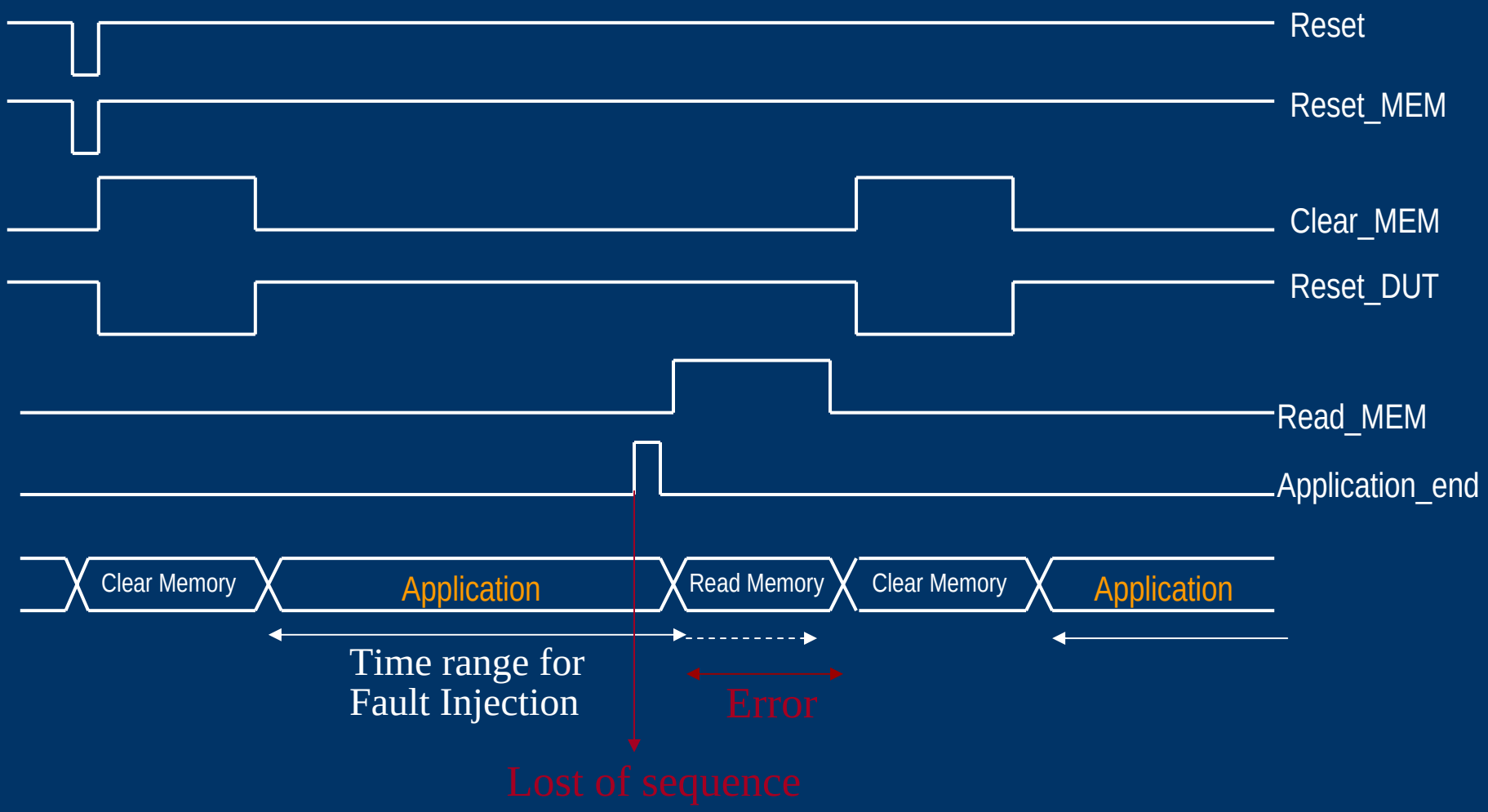
- Observability of an error in response of a fault injected in the DUT.
- Assumption: application results will be stored in the internal memory.
- Two main tasks are performed by the monitor block:
  - Clear all the internal memory in order to avoid accumulation of faults;
  - Read all the internal memory to analyse if there is an error or not.

Fault Injection Platform (Emulation)

- "Golden" chip approach was used to observe the faults in the system.
- Comparing:
  - Data memory → error
  - Application_end signal in the end → lost of sequence.



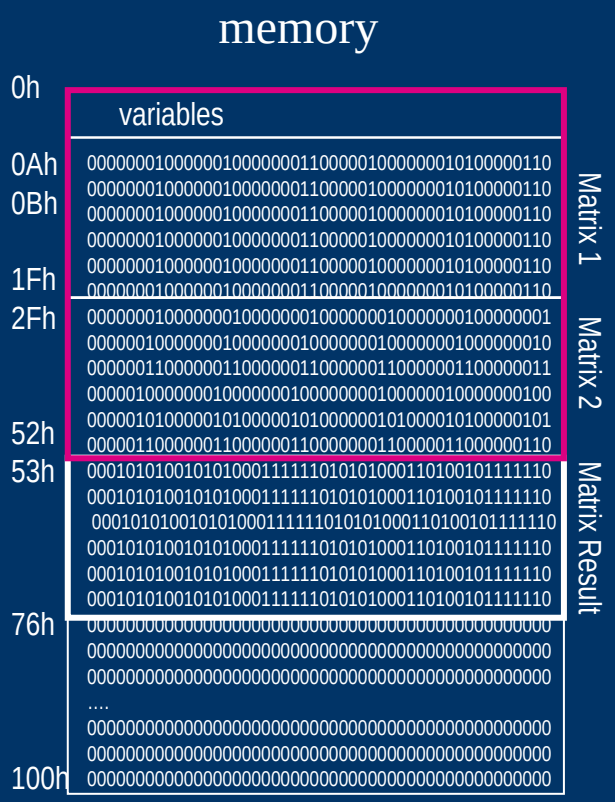
Fault Injection Signals



Case Sudy: μ8051 Fault Injection Experiment

- Registers: Datapath, Control block, state machine (16 registers)
- Memory: 256 bytes application memory (RAM)

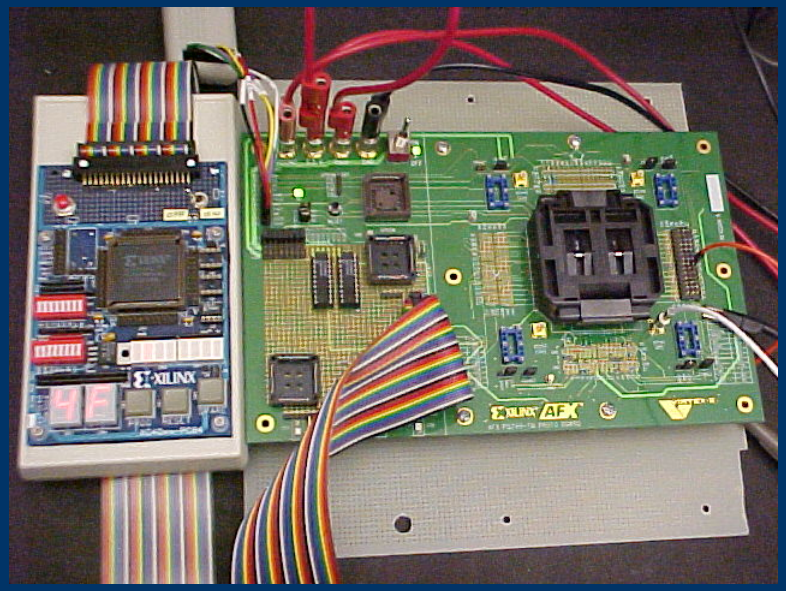
- 8051 Application: 6x6 Matrix Multiplication
  - Multiplication is performed by subsequent addition and left shift register.
- According to the TIME:
  - fault in the memory location < 52h can generate an error or a lost of sequence.



Case Study: µ8051 Fault Injection Experiment

Virtex Platform

- The fault injection system and the (DUT) were implemented in Virtex FPGA platform.
- Thousands of faults were injected in some seconds
- The emulation in FPGA showed a ~120,000 times acceleration



- **Component:** Virtex XCV300 - 240p
- 8051-like micro-controller runs at 10 MHz
- Fault injection Standard 8051
  - 28 % LUTs
- Fault injection Hardened 8051
  - 30 % LUTs

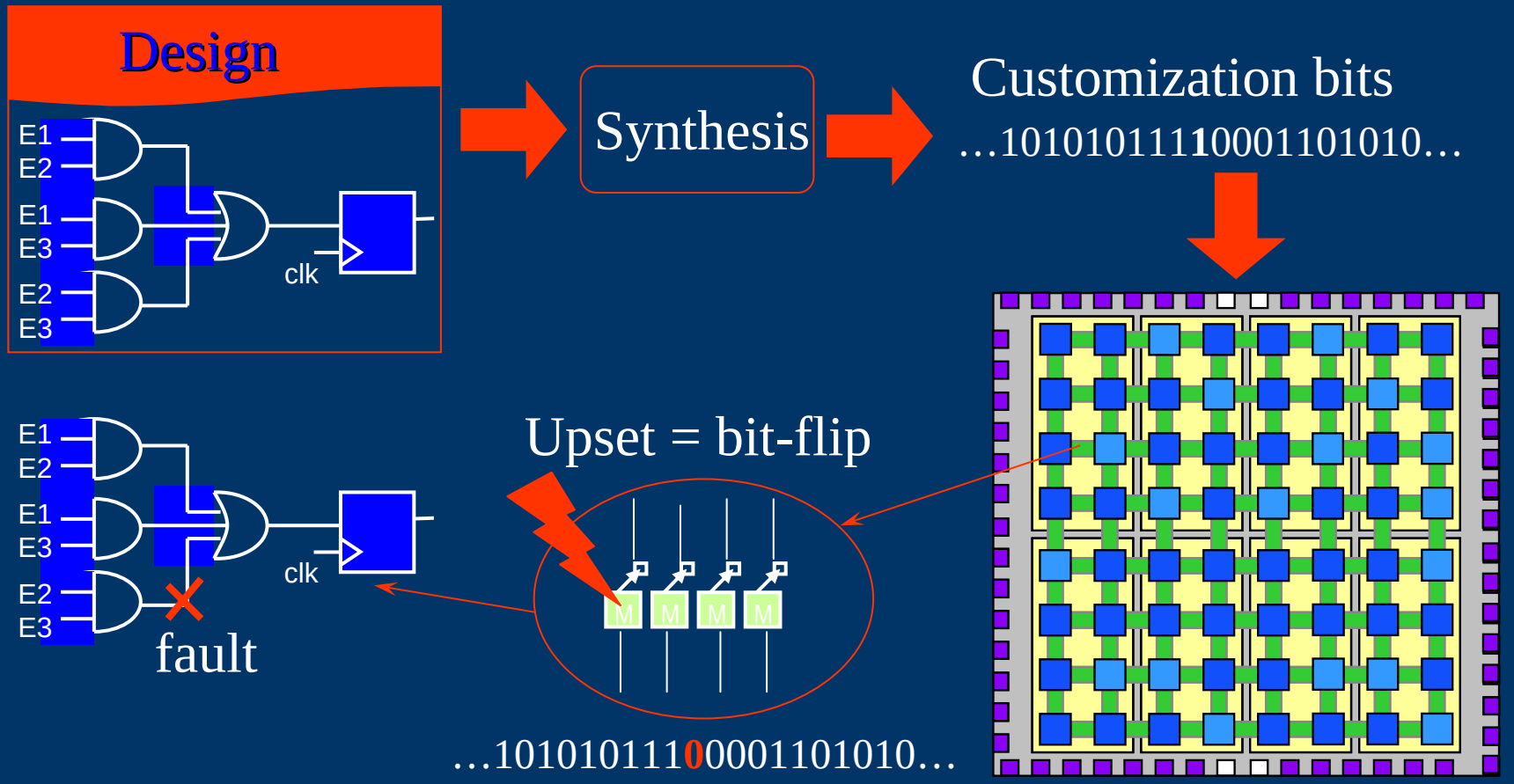
Fault Injection

FPGAs

(Directly into the bitstream)

Context

- Many digital designs are synthesized in SRAM based FPGAs
- Transient faults?



Motivation

- A way to test the reliability of a (fault-tolerant) design is to perform fault injection, simulating the effects of upsets (bit-flips), and see how the design responds
- The estimated time for injecting one bit fault in a FPGA bitstream is around 1 to 3 seconds
- The main steps of the fault injection are:
  - read the bit file
  - flip one bit
  - write the new bit file
  - download it to the board by the configuration mode interface
  - check the output signal

Fault Injection Cost Analysis

XCV300 Analysis

CLB Rows	x	CLB Cols
32	x	48

Total of 1.536 CLBs

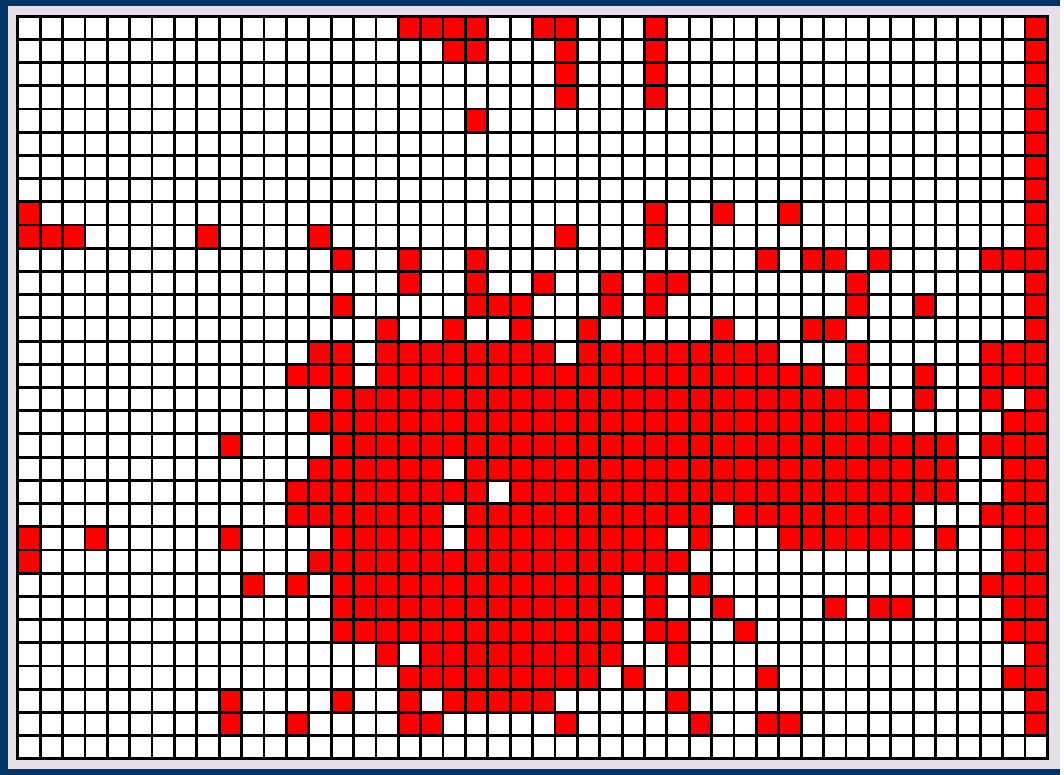
1.327.104 CLB Bits

31 days!

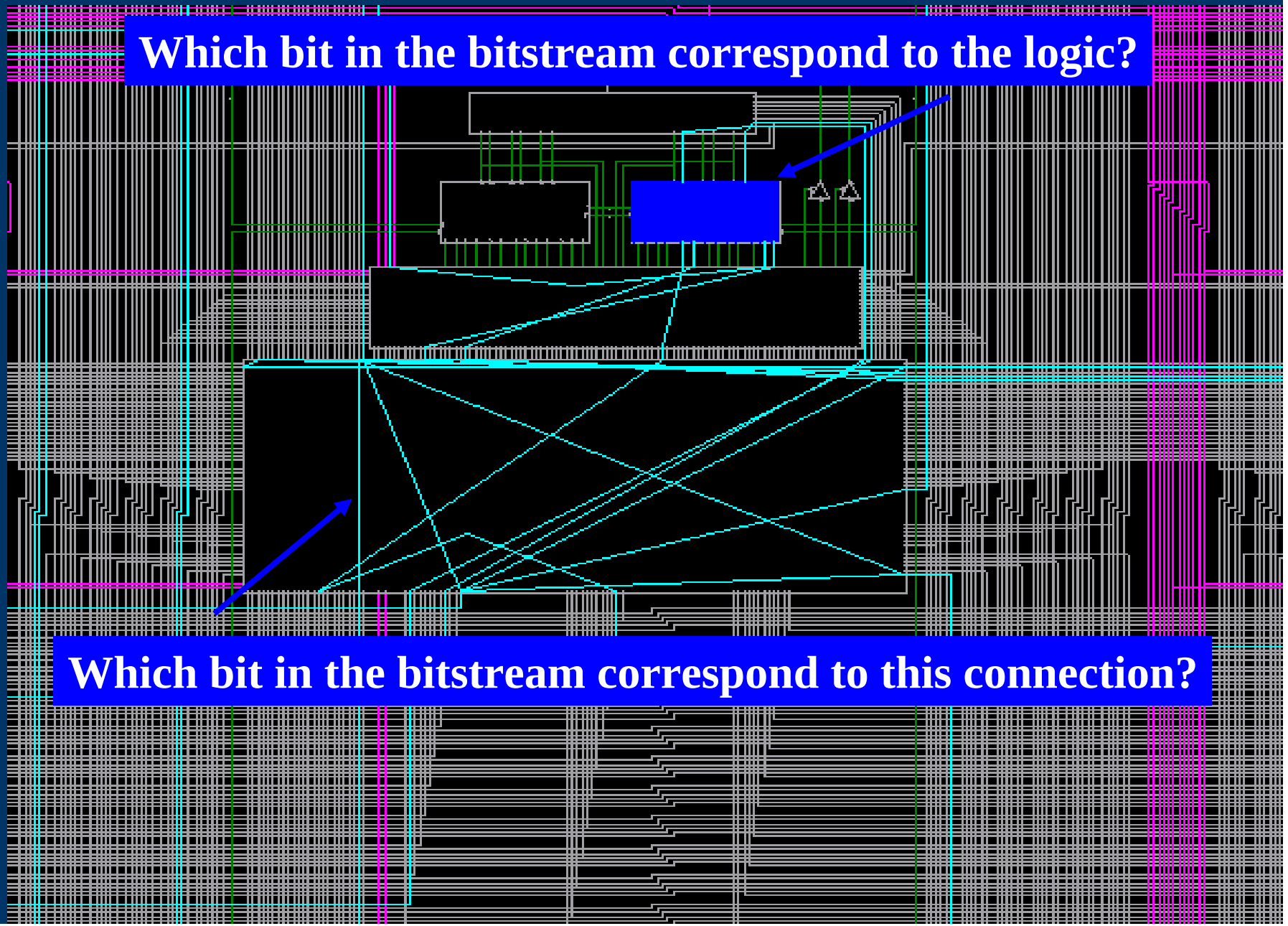
Total of 470 CLBs Used

406.080 CLB Bits

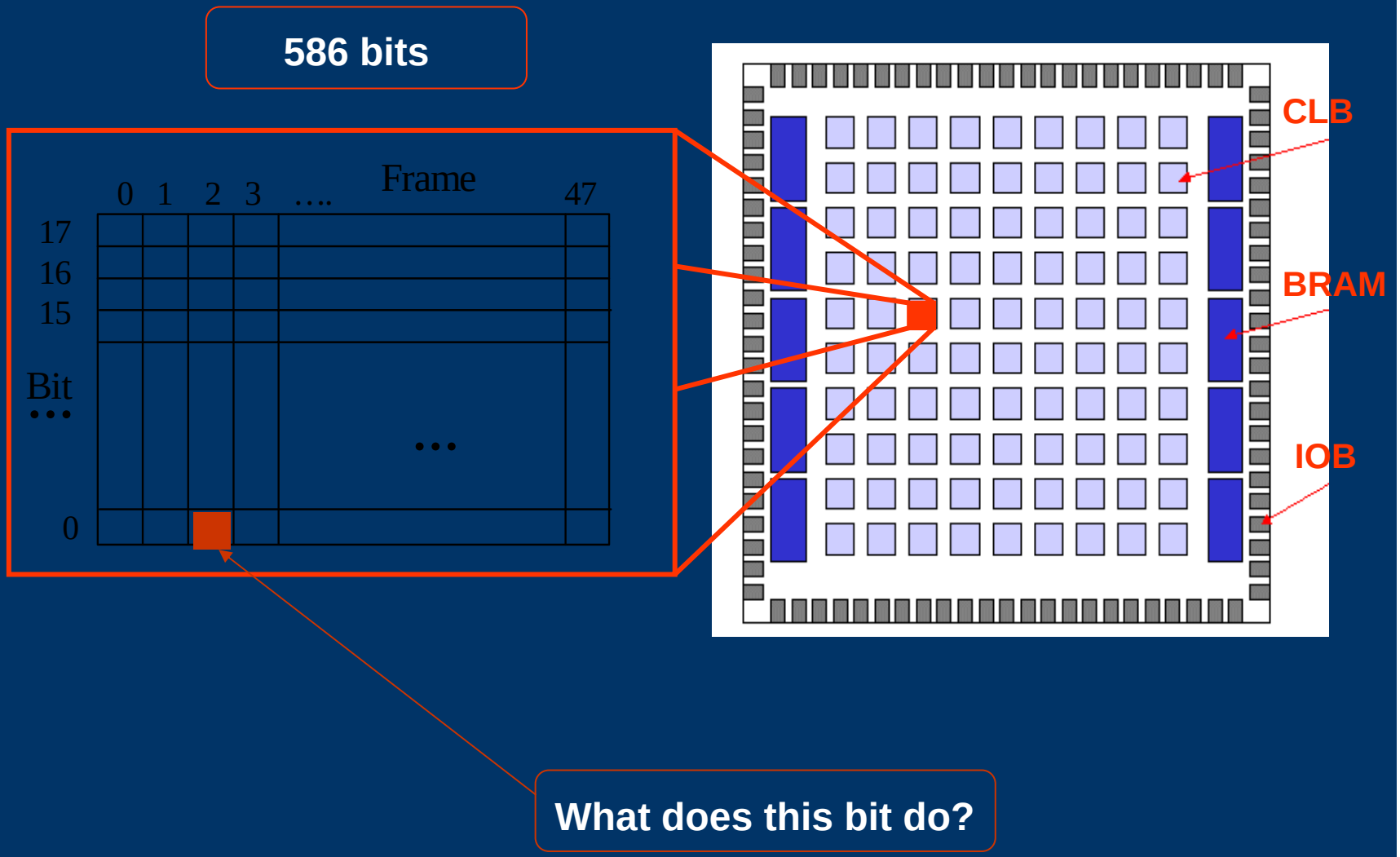
9.5 days!



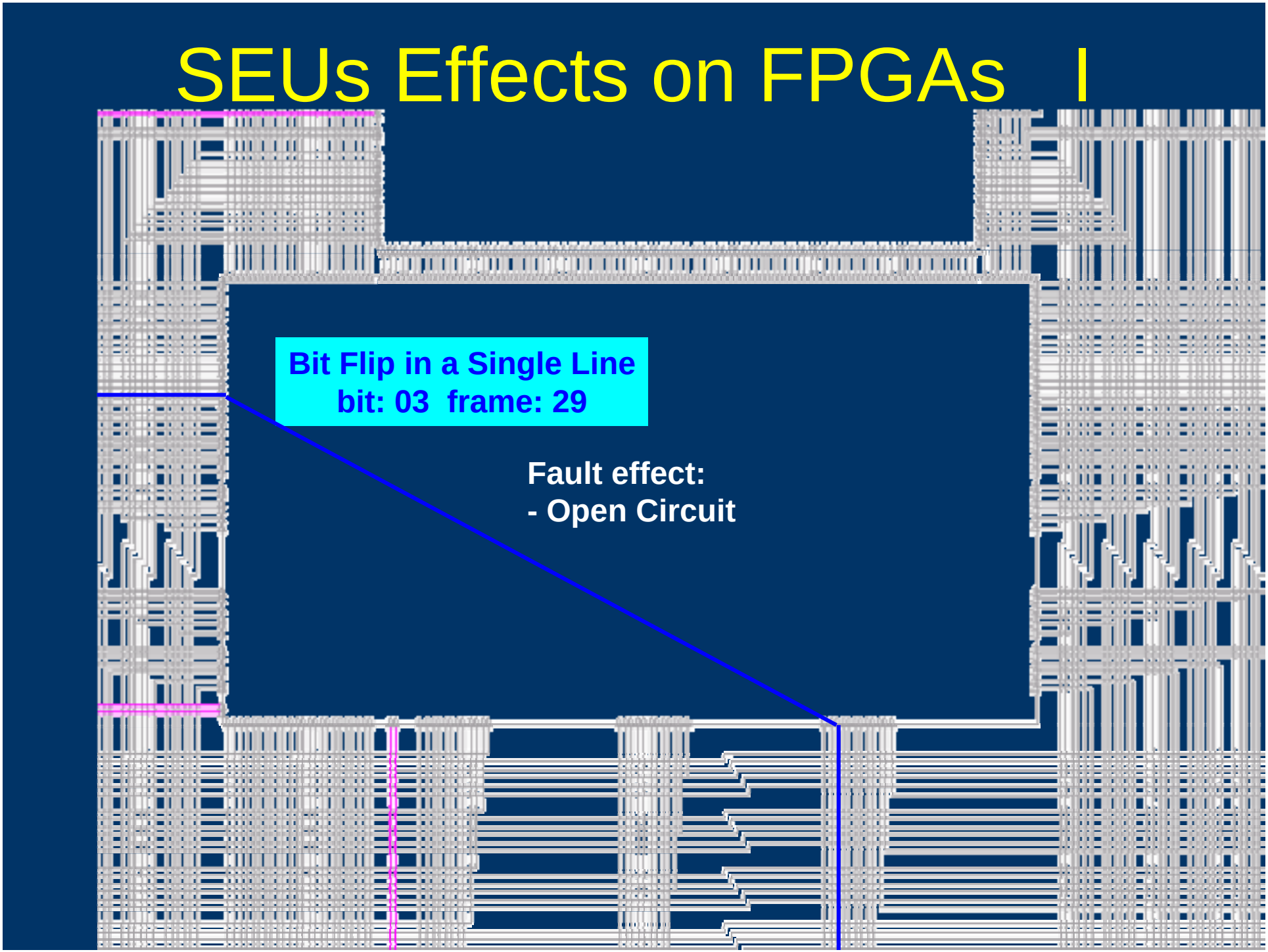
CLB View in the FPGA Editor



Virtex Bitstream



SEUs Effects on FPGAs I



SEUs Effects on FPGAs II