HEART 2015
Short Course
Unclassified Session

Introduction to FPGA Devices and The Challenges for Critical Application – A User's Perspective

Presenter:
Melanie Berg, AS&D in support of NASA/GSFC
Melanie.D.Berg@NASA.gov

Contributing Authors:
Kenneth LaBel NASA/GSFC
Kenneth.A.LaBel@NASA.gov

Acknowledgements

- ***Some of this work has been sponsored by the NASA Electronic Parts and Packaging (NEPP) Program and the Defense Threat Reduction Agency (DTRA).***
- ***Thanks is given to the NASA Goddard Radiation Effects and Analysis Group (REAG) for their technical assistance and support. REAG is led by Kenneth LaBel and Jonathan Pellish.***

Contact Information:

Melanie Berg: NASA Goddard REAG FPGA Principal Investigator:

Melanie.D.Berg@NASA.GOV

Acronyms

- Application specific integrated circuit (ASIC)
- Block random access memory (BRAM)
- Block Triple Modular Redundancy (BTMR)
- Clock (CLK or CLKB)
- Combinatorial logic (CL)
- Configurable Logic Block (CLB)
- Digital Signal Processing Block (DSP)
- Distributed triple modular redundancy (DTMR)
- Edge-triggered flip-flops (DFFs)
- Equivalence Checking (EC)
- Error detection and correction (EDAC)
- Field programmable gate array (FPGA)
- Gate Level Netlist (EDF, EDIF, GLN)
- Global triple modular redundancy (GTMR)
- Hardware Description Language (HDL)
- Input – output (I/O)
- Linear energy transfer (LET)
- Local triple modular redundancy (LTMR)
- Look up table (LUT)
- Operational frequency (*fs*)
- Power on reset (POR)
- Place and Route (PR)
- Radiation Effects and Analysis Group (REAG)
- Single event functional interrupt (SEFI)
- Single event effects (SEEs)
- Single event latch-up (SEL)
- Single event transient (SET)
- Single event upset (SEU)
- Single event upset cross-section (σ_{SEU})
- Static random access memory (SRAM)
- System on a chip (SOC)

Agenda

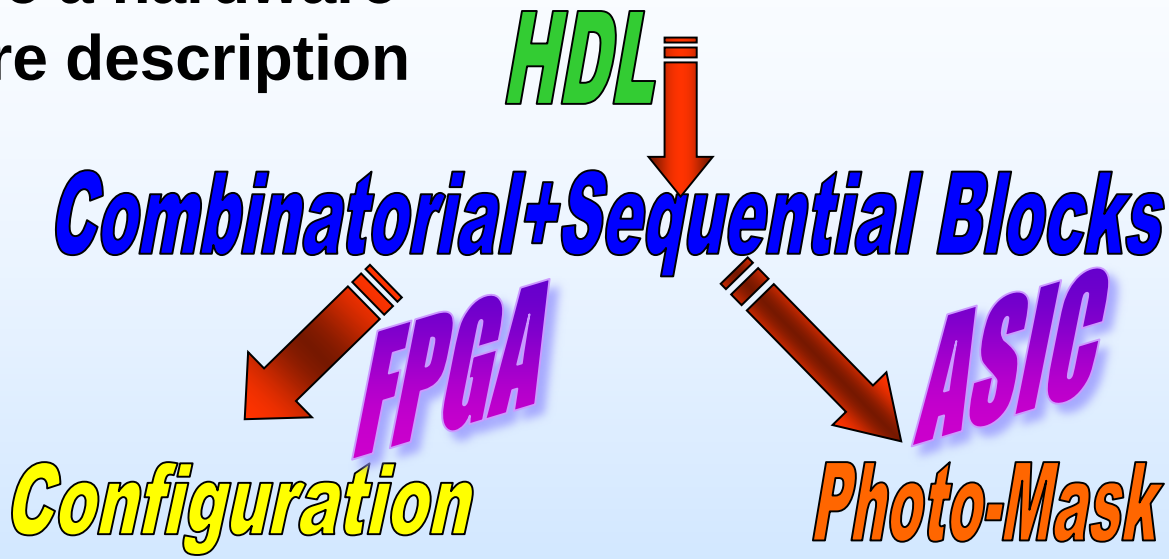
- **Field Programmable Gate Array (FPGA) versus Application Specific Integrated Circuit (ASIC) Devices.**
- **What's Inside An FPGA?**
- **FPGAs And Critical Applications.**
- **Single Event Upsets in FPGA Configuration.**
- **Single Event Upsets in an FPGA's Functional Data Path and Fail-Safe Strategies.**
- **Fail-Safe Strategies for FPGA Critical Applications.**

Definitions

- **A Field-Programmable Gate Array (FPGA) is a semiconductor device containing configurable logic components called "logic blocks", and configurable interconnects. Logic blocks can be configured to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions.**
- **An application-specific integrated circuit (ASIC) is an integrated circuit designed for a particular use, rather than intended for general-purpose use. Processors, RAM, ROM, etc are examples of ASICs.**
- **An FPGA is made out of an ASIC**

Creating A Design in An Integrated Circuit Device (FPGA or ASIC)

- The idea is to describe a hardware design using hardware description language (HDL):
  - Clocks,
  - Resets,
  - Sequential elements (e.g., flip-flops),
  - Combinatorial logic.



- The description gets synthesized into a hardware gate-level-netlist (GLN: file listing gates and connectivity).
- The synthesized hardware gates are mapped and placed into the cell library (or logic blocks) of the target FPGA or ASIC.

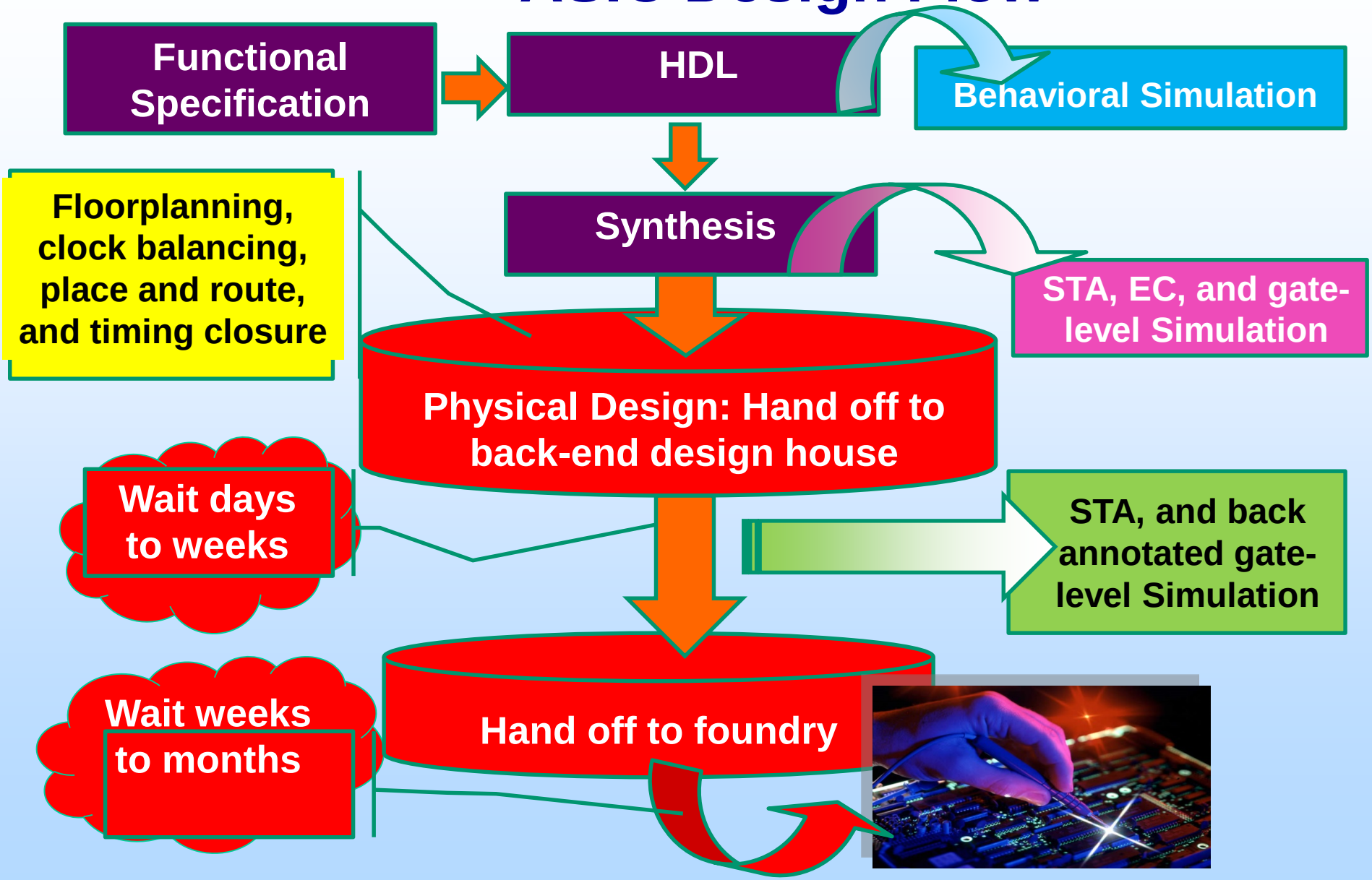
Design Tools

- **Design tools are used for each step of the design process.**
- **Synthesis: maps HDL into logic blocks (cells) … outputs gate-level net-lists.**
- **Place and route (PR): optimizes where the logic blocks and their interconnects should be.**
- **Synthesis along with place and route tools contain optimization algorithms within their tool sets.**
  - **These algorithms are used to optimize area, power, and logic function.**
  - **Tools are difficult and can produce incorrect functional logic.**
  - **Equivalence checking (EC) verifies tool output matches HDL.**
  - **Poorly designed tools can create designs that are too large to fit into the target device or output too much power. Hence, produce unusable designs.**

Best practice is to use a proven vendor's tool set – or product might be unreliable or unusable.

HDL: Hardware description language
STA: Static timing analysis
EC: Equivalence checking

ASIC Design Flow



FPGA Design Flow

FPGAs are created by manufacturers and are sold to users. The user maps a design into the FPGA fabric.

Manufacturer

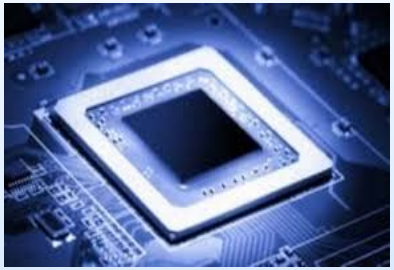
ASIC Design Flow

Manufacturer

Manufacturer creates FPGA design structure: logic block cells, routing structures, configuration

Manufacturer sends FPGA circuit to foundry

FPGA



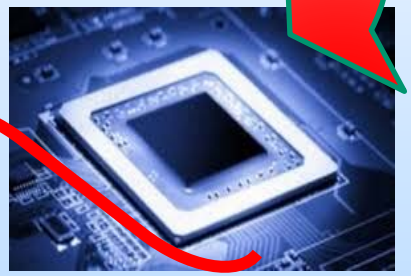
FPGAs are sold to users with configurable logic blocks and routes (they do not contain operable design)

FPGA

USER

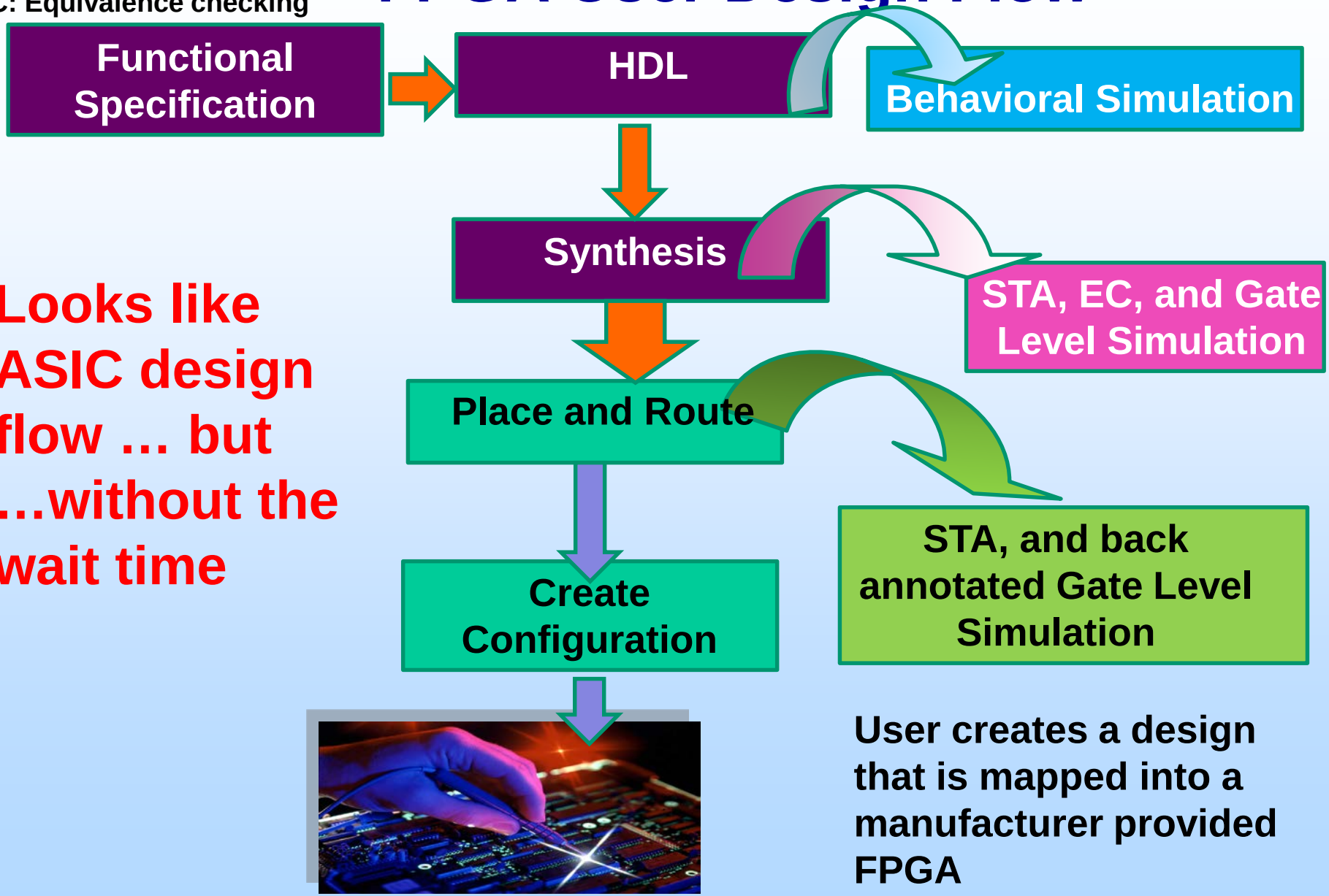
User Design Flow

User maps a design into FPGA circuits



HDL: Hardware description language
STA: Static timing analysis
EC: Equivalence checking

FPGA User Design Flow



Looks like ASIC design flow … but …without the wait time

User creates a design that is mapped into a manufacturer provided FPGA

FPGA or ASIC?

FPGA and ASIC Devices … System Usage

- **An FPGA (similarly to an ASIC) can be used to solve any problem which is computable:**
  - **User implements a digital (or mixed signal design).**
  - **Design can be trivial glue-logic (e.g., interface control) or**
  - **Design can be as complex as a system on a chip that may include processors, embedded memory, and high speed serial interfaces (Gigabit SERDES).** ***SERDES: serializer de-serializer***
- **The number of gates contained within the original FPGA devices were too small to compete with the ASIC devices of that time (1980s).**
  - **FPGAs were mostly used as interface glue logic.**
  - **Reduced system cost and added flexibility.**
- **Modern-day FPGAs contain millions of gates and have taken over a significant amount of the ASIC market.**

The ASIC Advantage

ASIC Advantage	Comment/Explanation
Full custom capability	The design is “tailored” and is manufactured to design specifications (no additional hidden logic)
Lower unit costs	Great for very high volume projects
Smaller form factor	Less logic is required because device is manufactured to design specs
No configuration	Overall reliability can decrease due to the addition of configuration technology/logic
Lower power	Less logic is required because device is manufactured to design specs

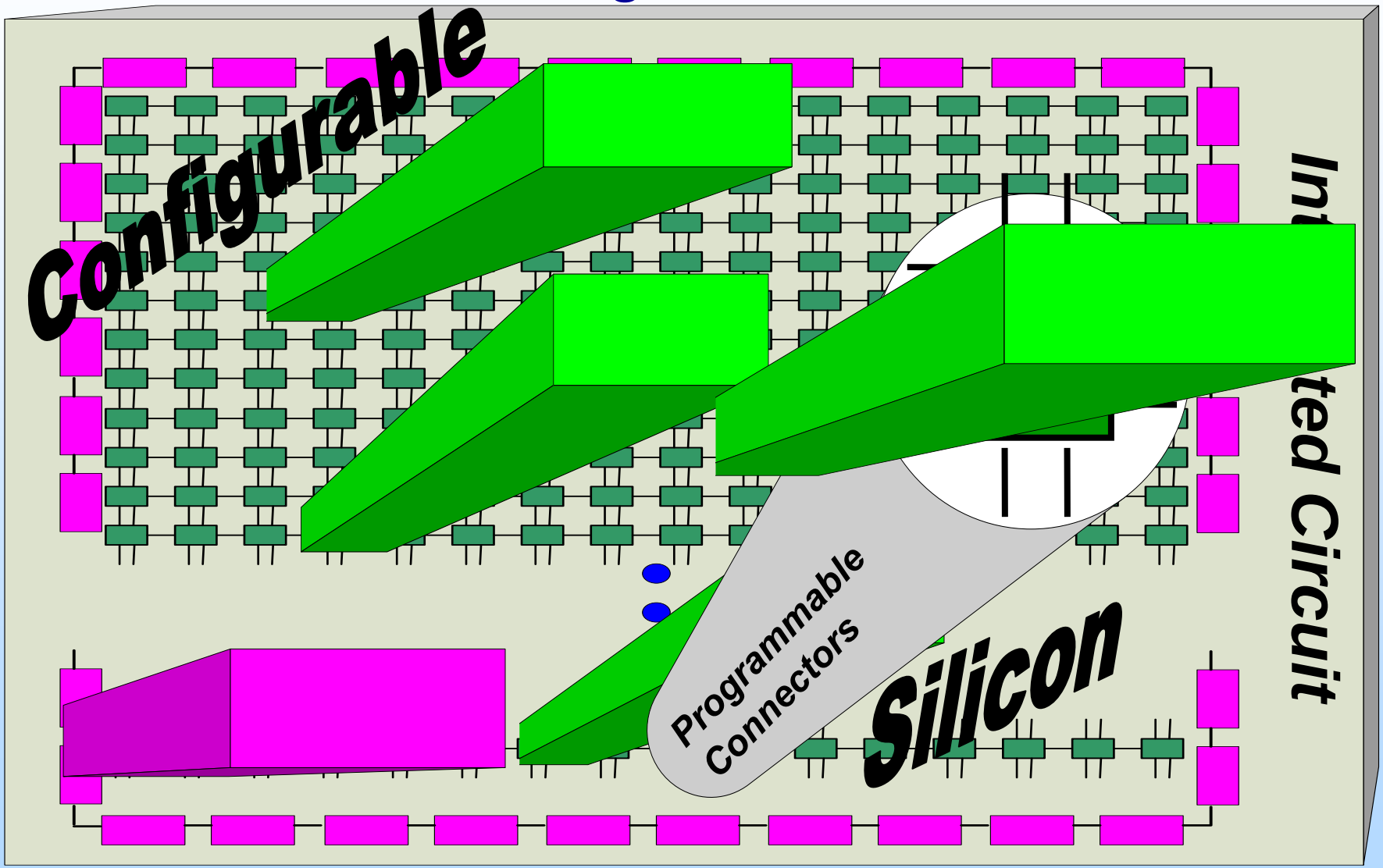
The FPGA Advantage

FPGA Advantage	Comment/Explanation
Faster time-to-market	No layout, masks or other manufacturing steps are needed
No upfront non-recurring expenses (NRE)	Costs typically associated with an ASIC design
Simpler design cycle	Due to the required tools that handle routing, placement, and timing
More predictable project cycle	Due to elimination of potential re-spins and lack of concern regarding wafer capacities as it would be in ASICs
Field reprogramability	It is easier to change a design in a system
Engineer availability	More students are taught FPGA design in school

FPGA: Faster design cycle and cheaper to implement

What is inside FPGA devices?

General FPGA Architecture: Fabric Containing Customizable Preexisting Logic…User Building Blocks



How Do FPGA's Differ?

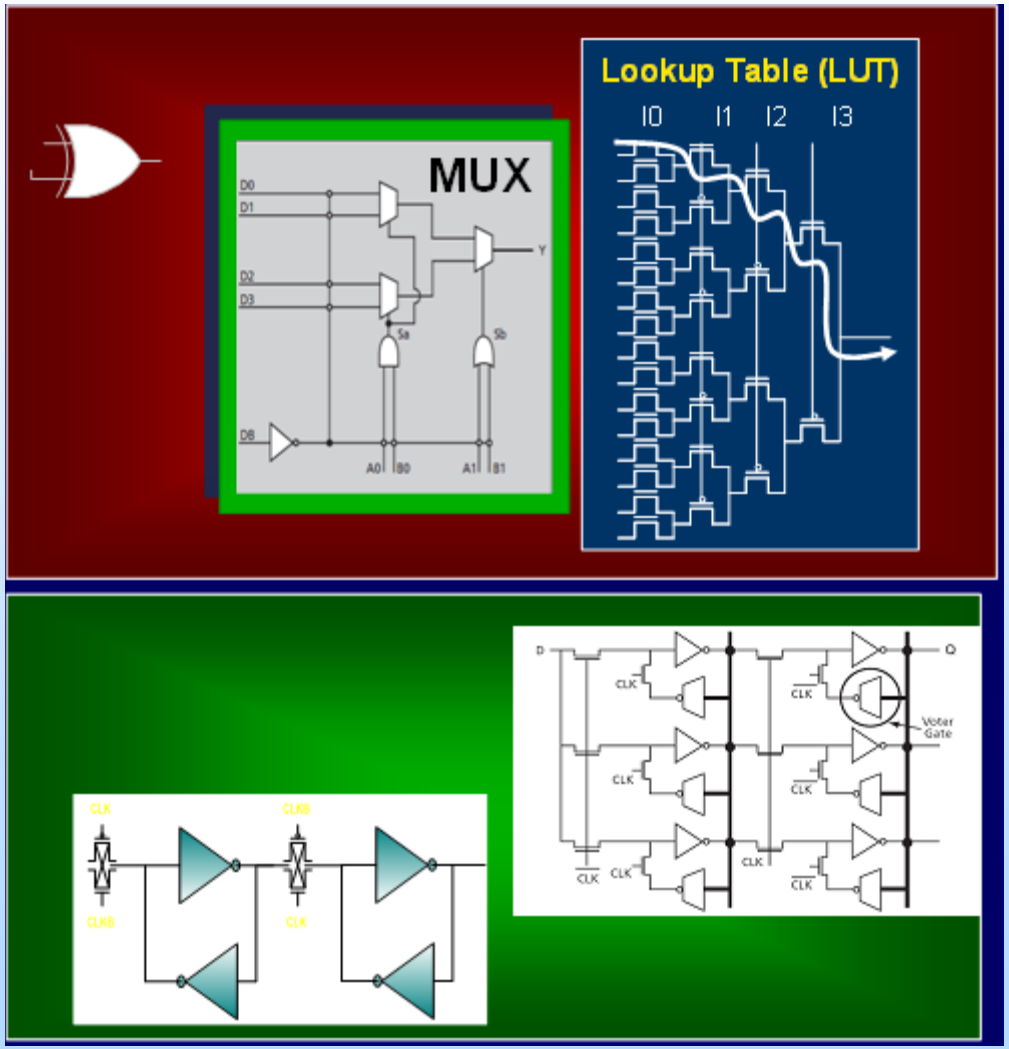
- **Manufacturer Architecture (not all are listed):**
  - **Configuration,**
  - **User building blocks (combinatorial logic cells, sequential logic cells),**
  - **Routing,**
  - **Clock structures,**
  - **Embedded mitigation, and**
  - **Embedded intellectual property (IP); e.g., memories and processors.**
- **Manufacturer design tool environment:**
  - **Synthesis,**
  - **Place and Route, and**
  - **Configuration management output.**

Difference in architectures and tools will affect the final design and design process – users be aware.

FPGA Component Libraries: Basic Designer Building Blocks (They Differ per FPGA Type)

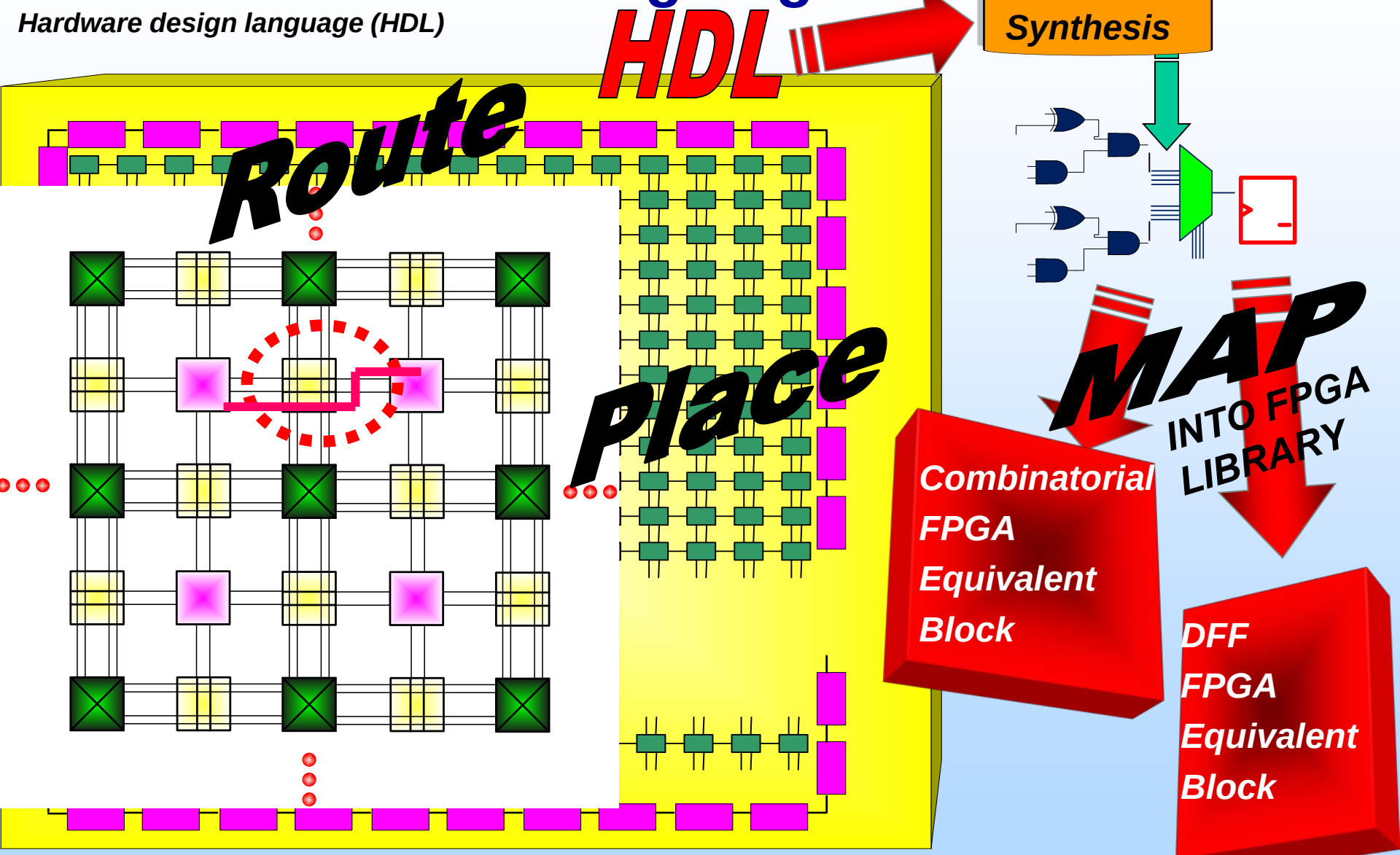
- **Combinatorial logic (CL) blocks**
  - **Vary in complexity.**
  - **Vary in I/O.**

- **Sequential logic blocks (DFF)**
  - **Uses global Clocks.**
  - **Uses global Resets.**
  - **May have mitigation.**



User Maps the Design Logic into FPGA Preexisting Logic

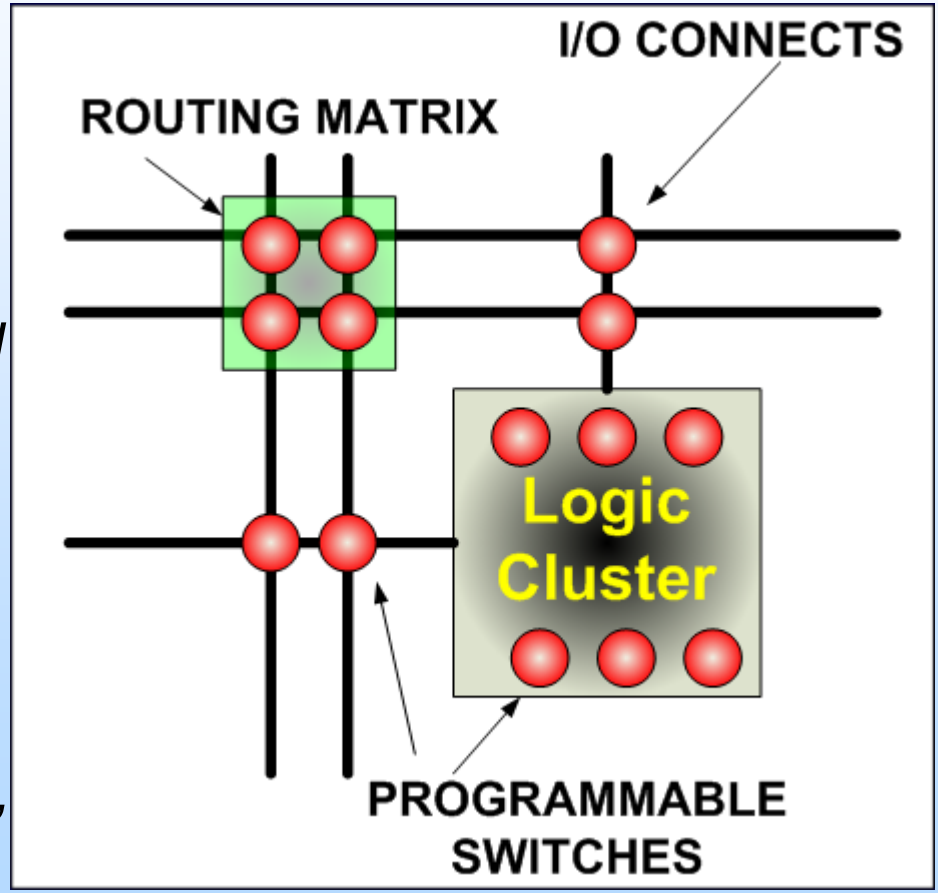
Hardware design language (HDL)



FPGA Configuration (Storage of User Design Mapping)

HDL → ***FPGA MAPPING*** → **Configuration**

- ***Configuration Defines:*** *Arrangement of pre-existing logic via programmable switches.*
  - *Functionality (logic cluster) and*
  - *Connectivity (routes)*
- ***Programmable Switch Types:***
  - ***Antifuse:*** *One time Programmable (OTP),*
  - ***SRAM:*** *Reprogrammable (RP), or*
  - ***Flash:*** *Reprogrammable (RP).*

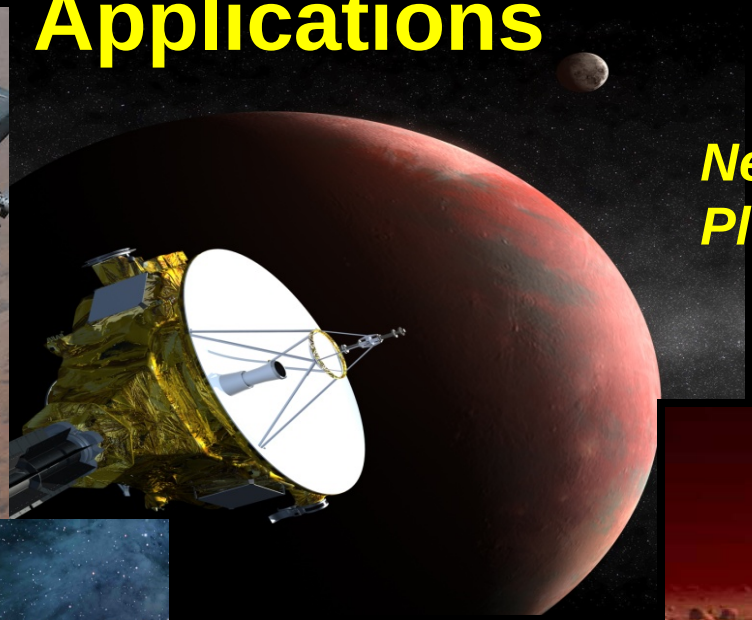


Common FPGA Applications

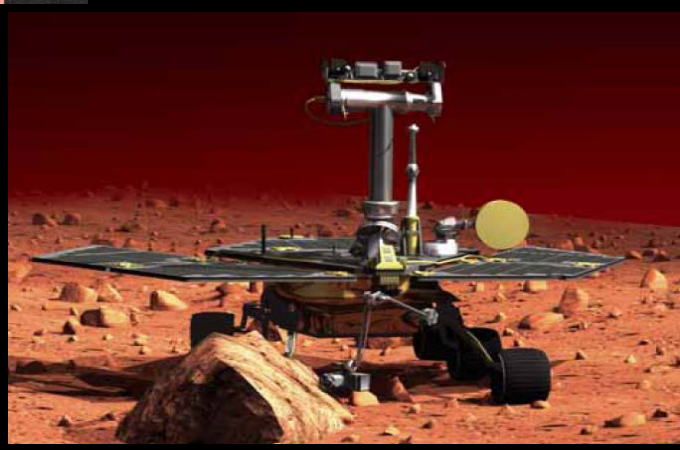
- **Controllers,**
- **Dataflow and interface adaptation,**
- **Digital signal processing (DSP),**
- **Software-defined radio,**
- **ASIC prototyping,**
- **Medical imaging,**
- **Robotic control (vision, movement, speech, etc.,…)**
- **Cryptology,**
- **Nuclear plant control,**
- **The list goes on…**

The following short course presentations will provide more details.

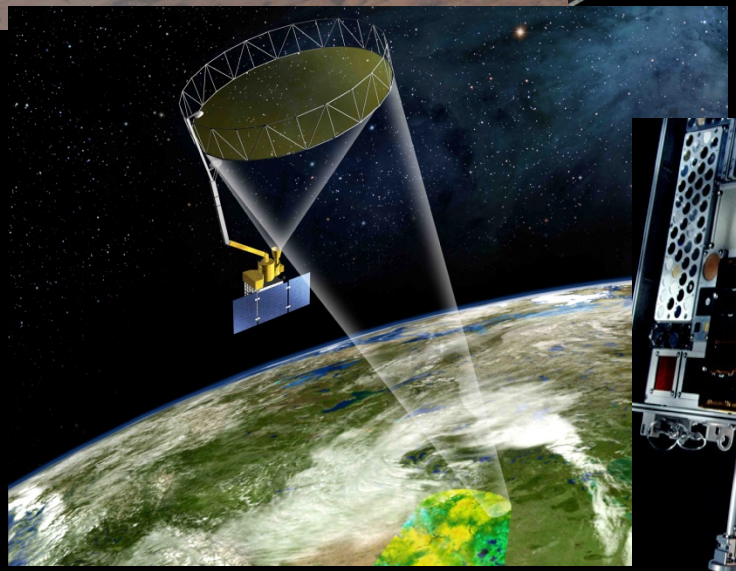
Example 1: FPGA Military and Space Applications



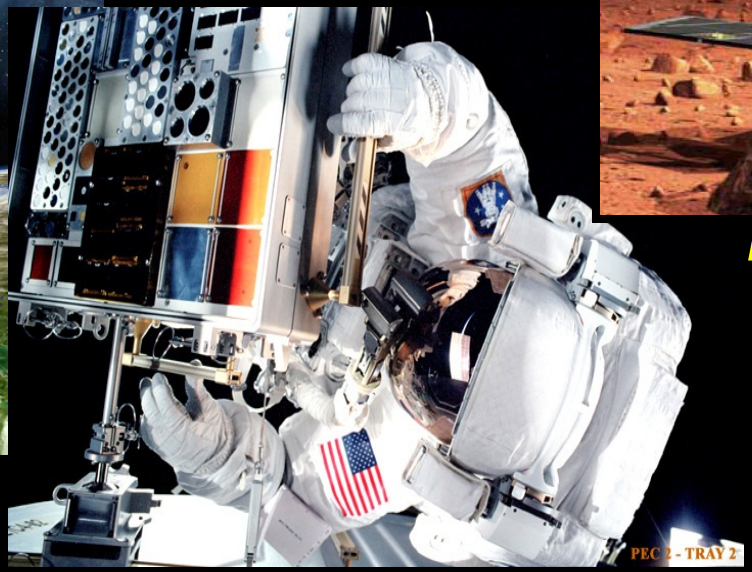
New Horizons
Pluto and Beyond



Mars Rover



Soil Moisture
Active Passive



Spacecube:
International
Space Station

Example 2: FPGA Terrestrial Application

Automotive applications that are opening up to FPGA-based solutions:

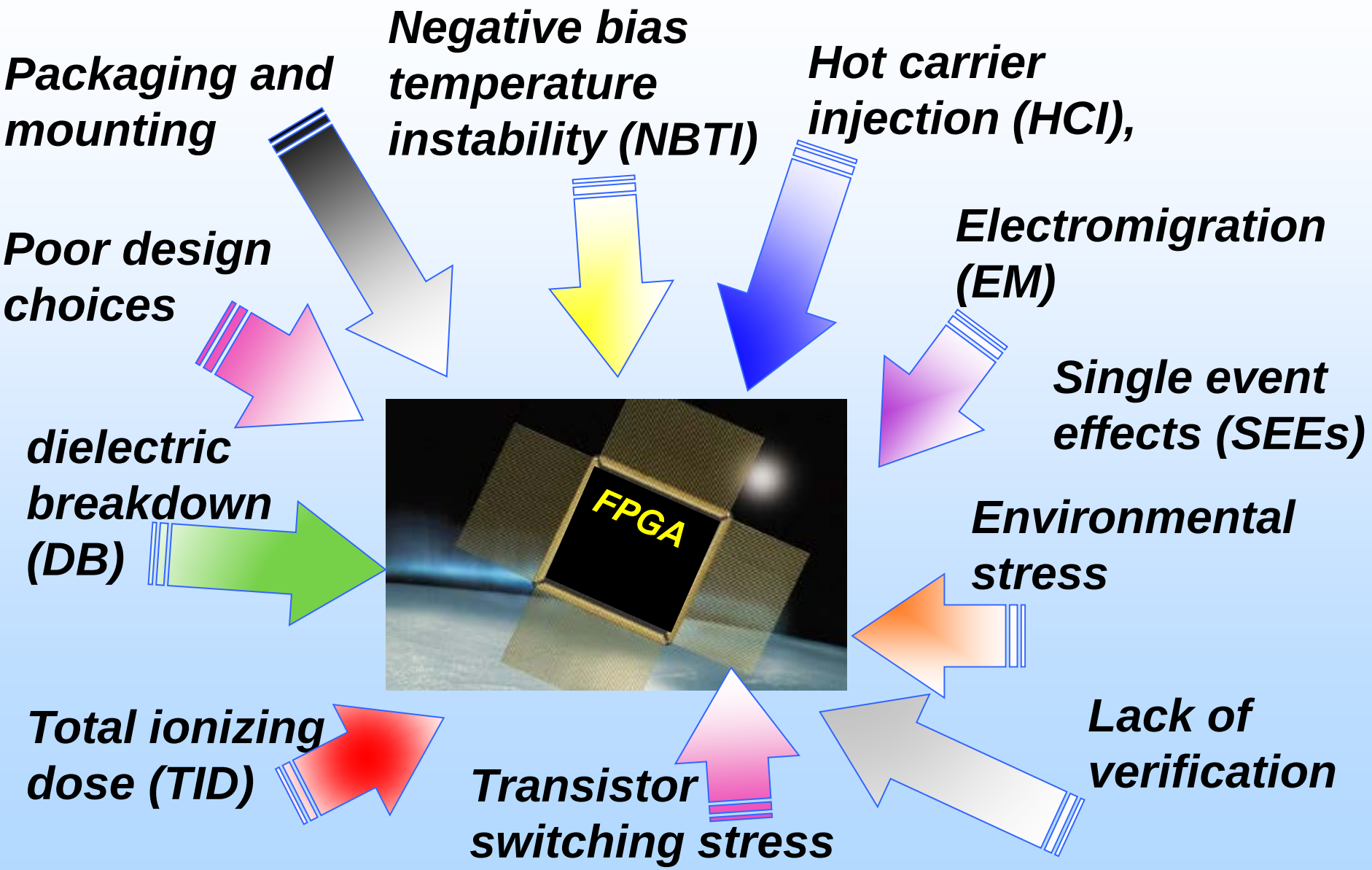
- Navigation and Telematics Displays
- Personnel Occupancy Detection Systems (PODS) for Next-Generation Airbags
- Blind-Spot Warning System
- Engine Control Module
- Lane Departure Warning System
- Adaptive Cruise Control
- Collision Avoidance System
- Injector Control (especially diesel engines)
- Power Steering Control
- Multi-Axis Power Seat Control
- Advanced Suspension and Traction Control
- Emissions Control
- Back-up Sensors
- Back-up Camera
- Rear-Seat Entertainment Source MUXing
- Digital Cluster

FPGAs and Critical Applications

- **Safety: can circuits or humans be damaged or hurt?**
- **Reliability : will the device operate as expected?**
- **Availability: how often will the system operate as expected?**
- **Recoverability: if the device malfunctions, can the system come back to a working state?**
- **Can the device and its design be trusted (security)**

Critical applications will want to avoid disaster.

Sources of FPGA Failure



How To Protect A System from Failure

- **Investigate failure modes – understand risk:**
  - **Reliability testing (temperature, voltage, mechanical, and logic switching stresses).**
  - **Radiation testing: Single event effects (SEE) and total ionizing dose (TID).**
- **Add redundancy:**
  - **Replication with correction.**
  - **Replication with detection. Requires recovery:**
    - **Switch to another device,**
    - **Try to recover state,**
    - **Start over,**
    - **Alert,**
    - **Do nothing… die.**
- **Add filtration: e.g., Finite impulse response (FIR) filters or Constant false alarm rate filter (CFAR).**
- **Add masking.**

Go no Go: Single Event Hard Faults and Common Terminology

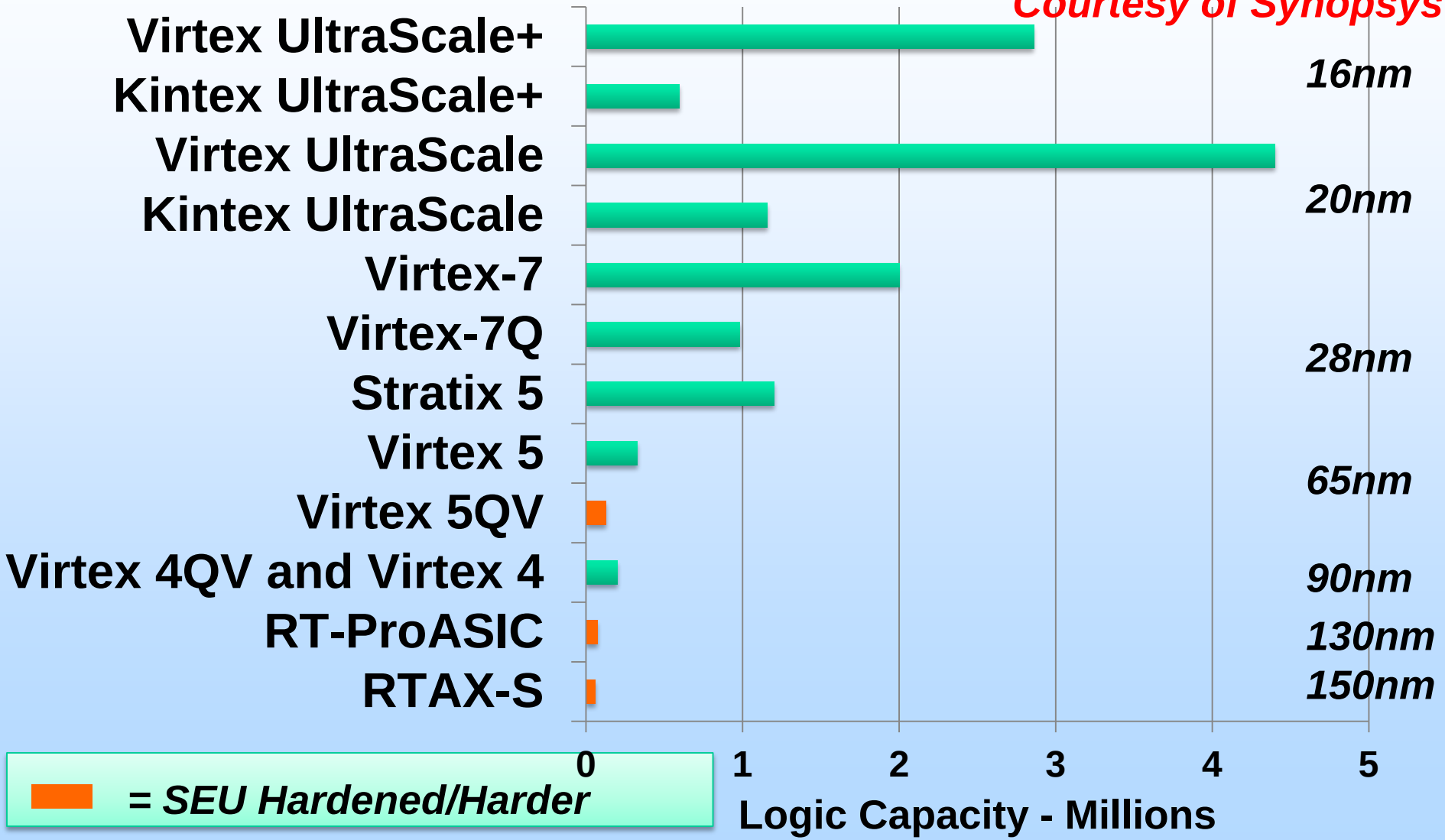
- **Single Event Latch Up (SEL): Device latches in high current state:**
  - **Has been observed in FPGA devices that are currently on the market.**
  - **Some missions choose to use the devices and design around the SEL.**
- **Single Event Burnout (SEB): Device draws high current and burns out.**
  - **Not observed in FPGA devices that are currently on the market.**
- **Single Event Gate Rupture: (SEGR): Gate destroyed typically in power MOSFETs.**
  - **Not observed in FPGA devices that are currently on the market.**

Radiation Hardened versus Commercial FPGA Devices

- **Radiation hardened FPGA devices are available to users. They make the design cycle much easier!**
- **They are considered hardened if:**
  - **Configuration susceptibility is reduced to an acceptable rate.**
  - **Generally, less than one node per $1x10^{-8}$ days.**
  - **Be careful: with millions of nodes, this can translate into 1 or two configuration failures per year.**
  - **However, if the node isn't being used, then your circuit may not be affected by the failure.**
- **The following presentation will discuss FPGAs with embedded mitigation.**
- **This presentation will focus on user inserted mitigation techniques.**

Small Device Geometries Enable High Capacity Applications but Non-Radiation Hardened Devices May Require SEU Mitigation

Courtesy of Synopsys



SEUs and FPGAs

- **Ionizing particles cause upsets (SEUs) in FPGAs.**
- **Each FPGA type has different SEU error signatures:**
  - **Temporary glitch (transient),**
  - **Change of state (incorrect state machine transitions),**
  - **Global upsets: Loss of clock or unexpected reset,**
  - **Configuration corruption. This includes route breakage (no signal can get through) – can be overwhelming.**
- **The question is how to avoid system failure and the answer depends on the following:**
  - **The system's requirements and the definition of failure,**
  - **The target FPGA and its surrounding circuitry susceptibility,**
  - **Implemented fail-safe strategies,**
  - **Reliable design practices,**
  - **Radiation environment.**

Fail-safe Strategies of Single Event Upsets (SEUs)

- Although there are many sources of FPGA malfunction, this presentation will focus on SEUs as a source of failure.
- The following slides will demonstrate commonly used mitigation strategies for FPGA devices.
- What you should learn:
  - The differences between FPGA mitigation strategies.
  - Strengths and weaknesses of various strategies.
  - Questions to ask or considerations to make when evaluating mitigation schemes.
  - Which mitigation schemes are best for various types of FPGA devices.

FPGA Structure Categorization as Defined by NASA Goddard REAG:

Single event functional interrupts (SEFI)
SEFI out of presentation scope

SEU cross section: σ_{SEU}

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

Design σ_{SEU}

Configuration σ_{SEU}

Functional logic σ_{SEU}

SEFI σ_{SEU}

Sequential and Combinatorial logic (CL) in data path

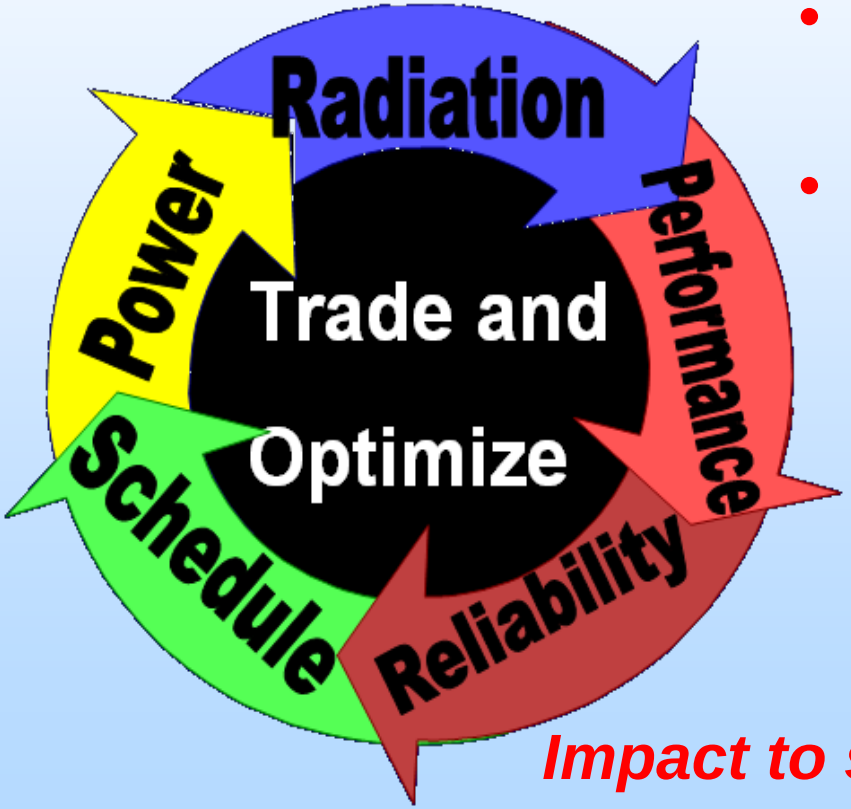
Global Routes and Hidden Logic

SEU Testing is required in order to characterize the σ_{SEU}s for each of FPGA categories.

Preliminary Design Considerations for Mitigation And Trade Space

Determine Most Susceptible Components:

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$



- **Does the designer need to add mitigation?**
- **Will there be compromises?**
  - **Performance and speed,**
  - **Power,**
  - **Schedule**
  - **Mitigating the susceptible components?**
  - **Reliability (working and mitigating as expected)?**

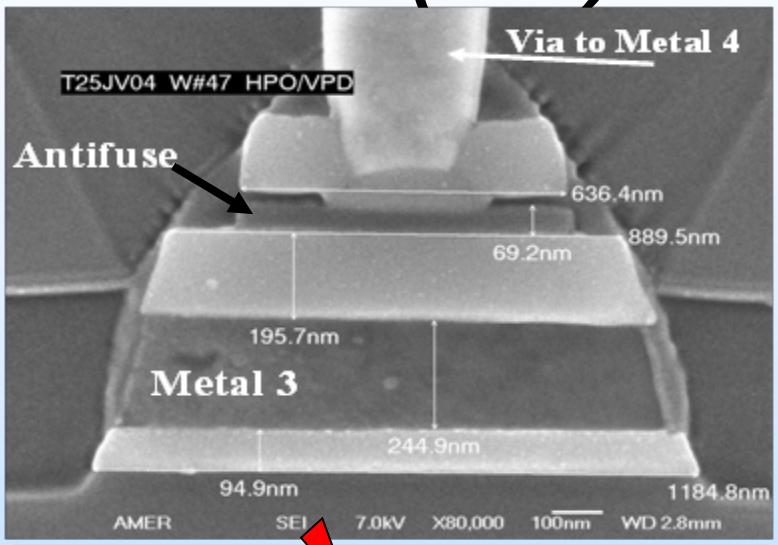
Impact to speed, power, area, reliability, and schedule are important questions to ask.

Single Event Upsets and FPGA Configuration

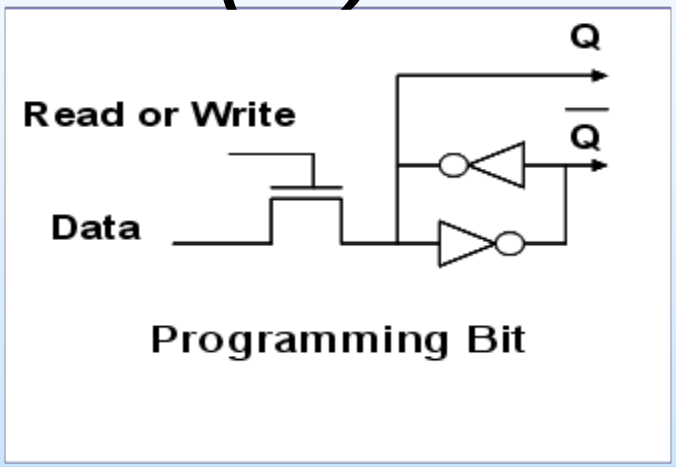
$$P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

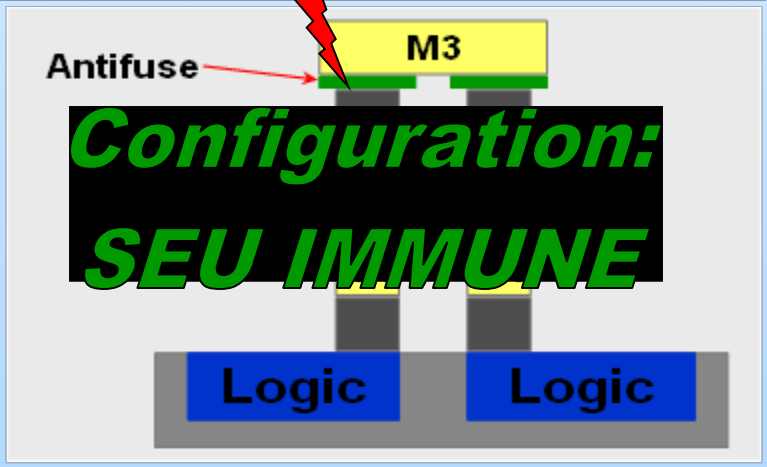
Programmable Switch Implementation and SEU Susceptibility

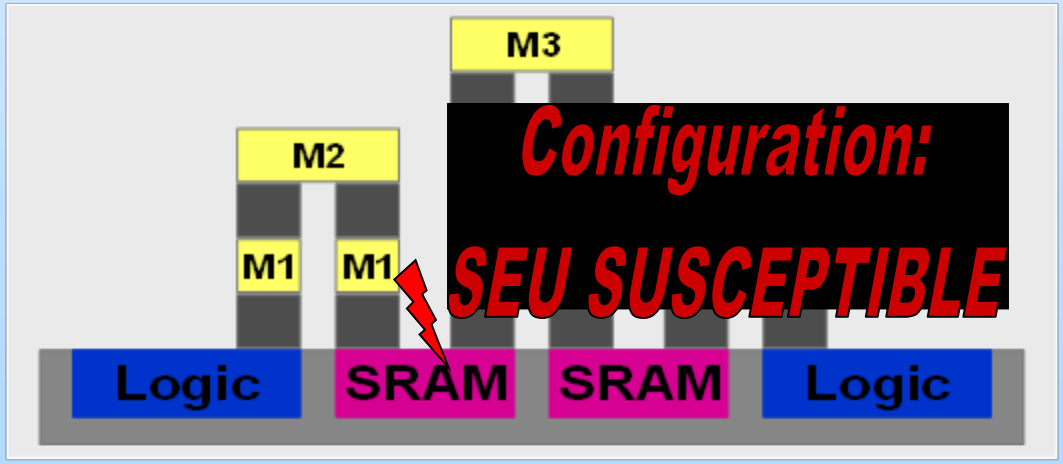
ANTIFUSE (OTP)



SRAM (RP)







Configuration SEU Test Results and the REAG FPGA SEU Model

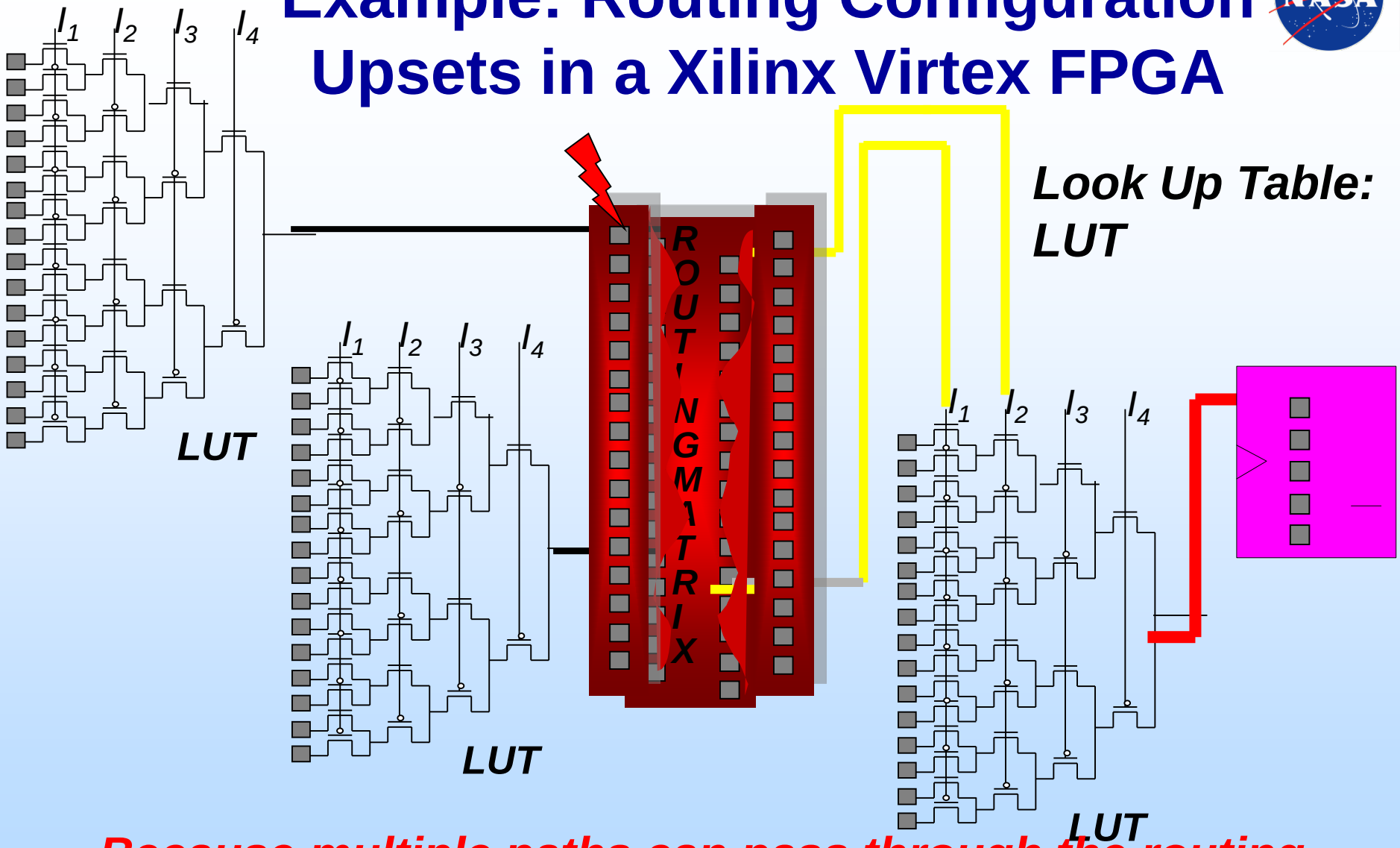
$$P(fs)_{error} \propto P_{Configurat\ ion} + P(fs)_{functional\ Logic} + P_{SEFI}$$

FPGA Configuration Type	REAG Model
Antifuse	$P(fs)_{error} \propto P(fs)_{functionalLogic} + P_{SEFI}$
SRAM **(non-mitigated)**	$P(fs)_{error} \propto P_{Configuration}$
Flash	$P(fs)_{error} \propto P(fs)_{functionalLogic} + P_{SEFI}$
Hardened SRAM	$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$

What Does The Last Slide Mean?

FPGA Configuration Type	Susceptibility Data-path: Combinatorial Logic (CL) and Flip-flops (DFFs); Global: Clocks and Resets; Configuration
Antifuse	Configuration has been designated as hard regarding SEEs. Susceptibilities only exist in the data paths and global routes. However, global routes are hardened and have a low SEU susceptibility.
SRAM **(non-mitigated)**	Configuration has been designated as the most susceptible portion of circuitry. All other upsets (except for global routes) are too statistically insignificant to take into account. E.g., it is a waste of time to study data path transients, however clock transient studies are significant.
Flash	Configuration has been designated as hard (but NOT immune) regarding SEEs. Susceptibilities also exist in the data paths and global routes (e.g., clocks and resets).
Hardened SRAM	Configuration has been designated as hardened (but NOT hard) regarding SEEs. Susceptibilities also exist in the data paths and global routes (e.g., clocks and resets).

Example: Routing Configuration Upsets in a Xilinx Virtex FPGA



Because multiple paths can pass through the routing matrix, this configuration can be catestrophic – i.e., break simple mitigation

Fixing SRAM-based Configuration…Scrubbing Definition

- **From SEU testing, it has been illustrated that the configuration memory of un-hardened SRAM-Based FPGAs is highly susceptible to SEUs.**
- **We address configuration susceptibility via scrubbing: Scrubbing is the act of simultaneously writing into FPGA configuration memory as the device's functional logic area is operating with the intent of correcting configuration memory bit errors.**

Configuration scrubbing only pertains to SRAM-based configuration devices.

Warning!

- **Fixing a configuration bit does not mean that you have fixed the state in the functional logic path.**
- **In order to guarantee that the functional logic is in the expected state after the configuration bit is fixed, either the state must be restored or a reset must be issued.**

Reliably getting to an expected state after a configuration-bit SEU (that affects the design's functionality) requires one of the following:

- ***Fix configuration bit + (reset or correct DFFs) or***
- ***Full reconfiguration.***

Single Event Upsets in an FPGA's Functional Data Path and Fail-Safe Strategies

$$P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

Data-path SEUs and Their Affect At The System Level

- **A system implemented in an FPGA is a cascade of sequential and combinatorial logic.**
- **Probability of a system error due to an SEU depends on many factors:**
  - **Probability of fault Generation in a gate (SET or SEU).**
  - **Probability of error propagation – will the SET or SEU force the system's next state to be incorrect?**

Probability of Error Propagation in A Data-Path

Upsets usually occur between clock cycles: Can cause a system-level malfunction if the SET or SEU will force the system's next state to be incorrect.

- **Capacitive filtration: data-path capacitance can stop transient upset propagation; e.g.:**
  - **Routing metal or heavy loading.**
  - **If a transient doesn't reach a sequential element, then it most likely will not cause a system upset.**
- **Logic masking: Redundancy and mitigation of paths can stop upset propagation.**
- **Logic masking: turned off paths from gated logic can stop upset propagation.**
- **Temporal delay: path delays can block temporary SEUs from disturbing next state calculation.**

Fail-Safe Strategies for FPGA Critical Applications

Goal for critical applications: Limit the probability of system error propagation and/or provide detection-recovery mechanisms via fail-safe strategies.

Differentiating Fail-Safe Strategies:

- **Detection:**
  - **Watchdog (state or logic monitoring).**
  - **Simplistic Checking … Complex Decoding.**
  - **Action (correction or recovery).**
- **Masking (does not mean correction):**
  - **Not letting an error propagate to other logic.**
  - **Redundancy + mitigation or detection.**
  - **Turn off faulty path.**
- **Correction (error may not be masked):**
  - **Error state (memory) is changed/fixed.**
  - **Need feedback or new data flush cycle.**
- **Recovery:**
  - **Bring system to a deterministic state.**
  - **Might include correction.**

Redundancy Is Not Enough

- **Just adding redundancy to a system is not enough to assume that the system is well protected.**
- **Questions/Concerns that must be addressed for a critical system expecting redundancy to cure all (or most):**
  - **How is the redundancy implemented?**
  - **What portions of your system are protected? Does the protection comply with the results from radiation testing?**
  - **Is detection of malfunction required to switch to a redundant system or to recover?**
  - **If detection is necessary, how quickly can the detection be performed and responded to?**
  - **Is detection enough?... Does the system require correction?**

Listed are crucial concerns that should be addressed at design reviews and prior to design implementation

Mitigation

- **Error Masking vs. Error Correction… there's a difference.**
- **Mitigation can be:**
  - **User inserted: part of the actual design process.**
    - **User must verify mitigation… Complexity is a RISK!!!!!!!!**
  - **Embedded: built into the device library cells.**
    - **User does not verify the mitigation – manufacturer does.**
- **Mitigation should reduce error…**
  - **Generally through redundancy.**
  - **Incorrect implementation can increase error.**
  - **Overly complex mitigation cannot be verified and incurs too high of a risk to implement.**

Availability versus Correct Operation

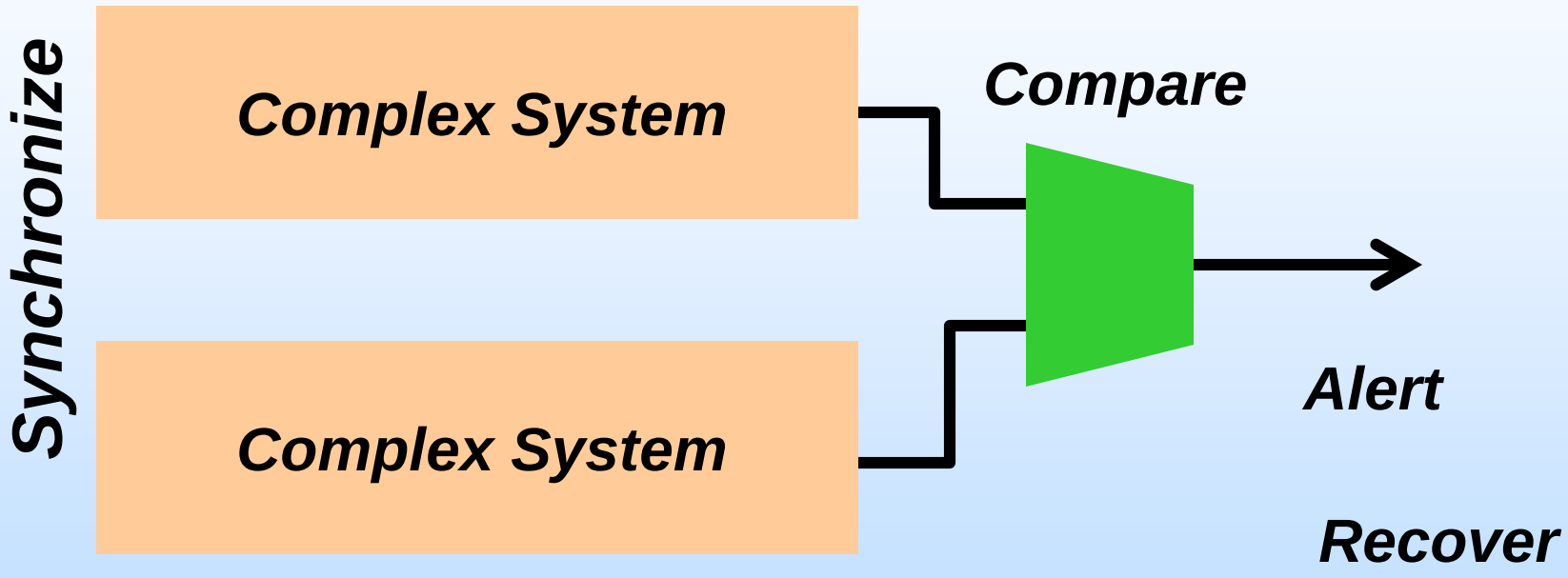
- **Requirements must be satisfied.**
- **What is your expected up-time versus down-time (availability)?**
- **Is correct operation well defined? Unambiguous!**
- **Is system failure well defined? Unambiguous!**
- **Can availability and correct operation be deterministic regardless of error signature?**
- **Availability:**
  - **Flushable designs: systems than can be reset or are self-correcting. Availability is affected during reset or correction time (down-time). However, downtime is tolerable as defined by system requirements.**
  - **Non-flushable designs: System requirements are strict and require minimal downtime. Usage of resets are required to be kept at a minimum.**

Detection and Recovery

- **Not all mitigation schemes require detection.**
- **Questions/Consideration:**
  - **If your scheme requires detection:**
    - **Can the system detect all error signatures?**
    - **Can the system detect all error signatures fast enough?**
    - **Do different errors require different recovery schemes… can the system accommodate.**
  - **How are you going to verify the detection and recovery?**
  - **How much downtime will there be during recovery (availability = detection time from error + recovery time – masked error time)**

Dual Redundant Systems (Detection Systems)

Dual Redundancy Example



Mitigation – Fail Safe Strategies That Do Not Require Fault Detection but Provide SEU Masking and/or Correction:
Triple Modular Redundancy (TMR)

TMR Schemes Use Majority Voting

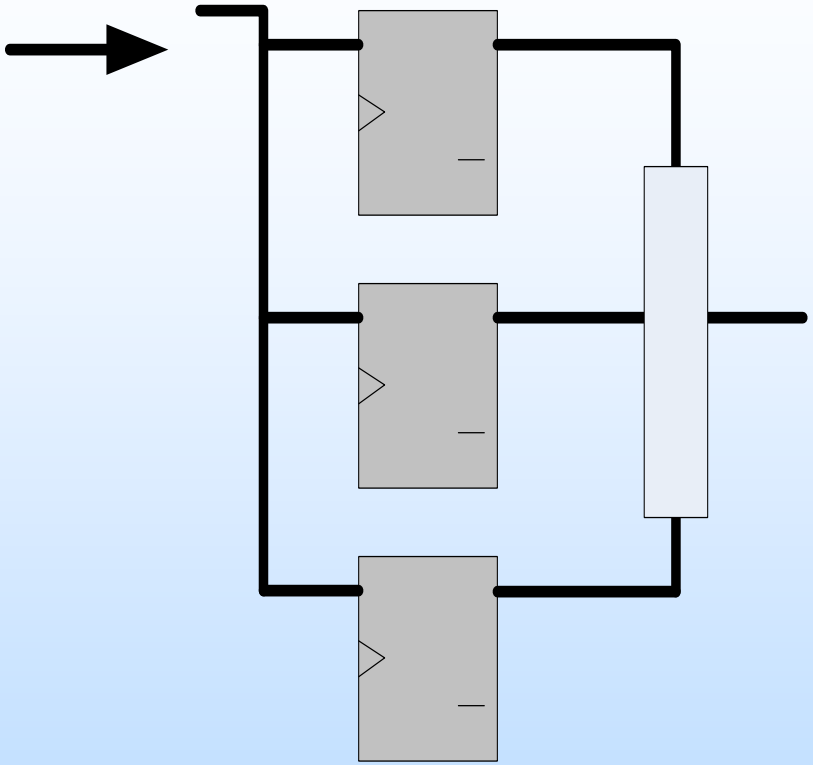
$$MajorityVoter = I1 \wedge I2 + I0 \wedge I2 + I0 \wedge I1$$

I0	I1	I2	Majority Voter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
			1

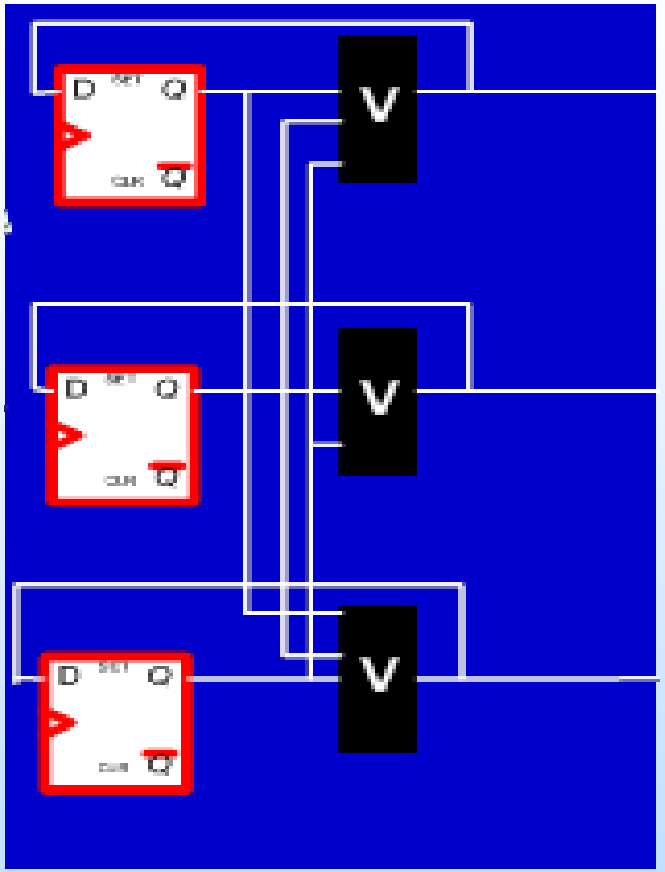
Best 2 out of 3

Triplicate and Vote

Triplicate and Vote



Singular Data Path



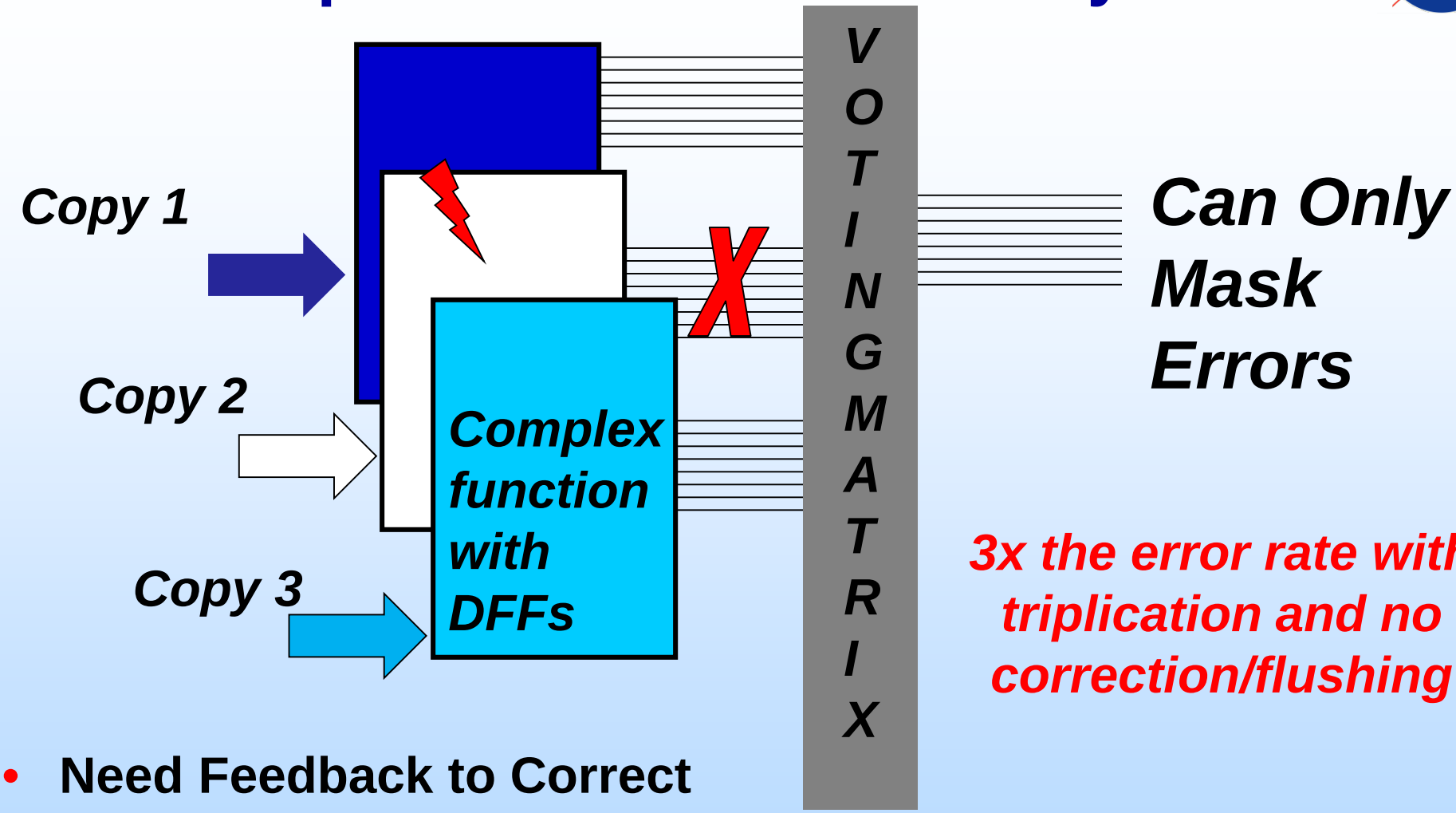
Redundant Data Path

But… it's not this easy!!!!!!!!!!!!!!!!!!!!

TMR Implementation

- As previously illustrated, TMR can be implemented in a variety of ways.
- The definition of TMR depends on what portion of the circuit is triplicated and where the voters are placed.
- The strongest TMR implementation will triplicate all data-paths and contain separate voters for each data-path.
  - However, this can be costly: area, power, and complexity.
  - Hence a trade is performed to determine the TMR scheme that requires the least amount of effort and circuitry that will meet project requirements.

Block Triple Modular Redundancy: BTMR

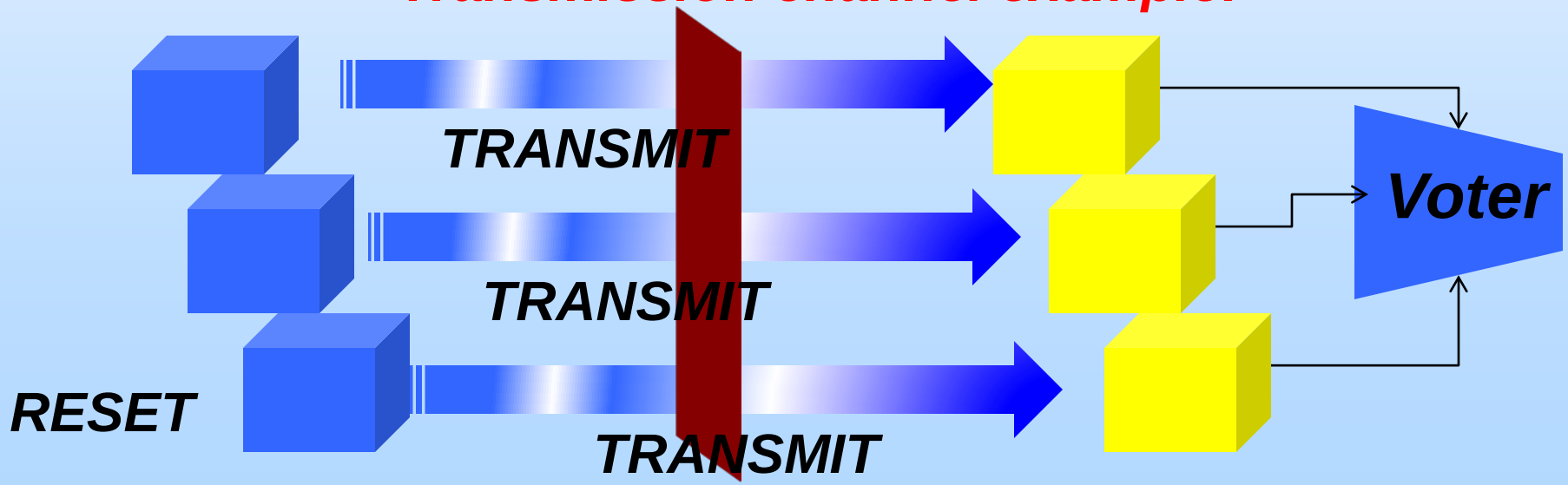


- **Need Feedback to Correct**
- **Cannot apply internal correction from voted outputs**
- **If blocks are not regularly flushed (e.g. reset), Errors can accumulate – may not be an effective technique**

Examples of a Flushable BTMR Designs

- **Shift Registers.**
- **Transmission channels: It is typical for transmission channels to send and reset after every sent packet.**
- **Lock-Step microprocessors that have relaxed requirements such that the microprocessors can be reset (or power-cycled) every so-often.**

Transmission channel example:

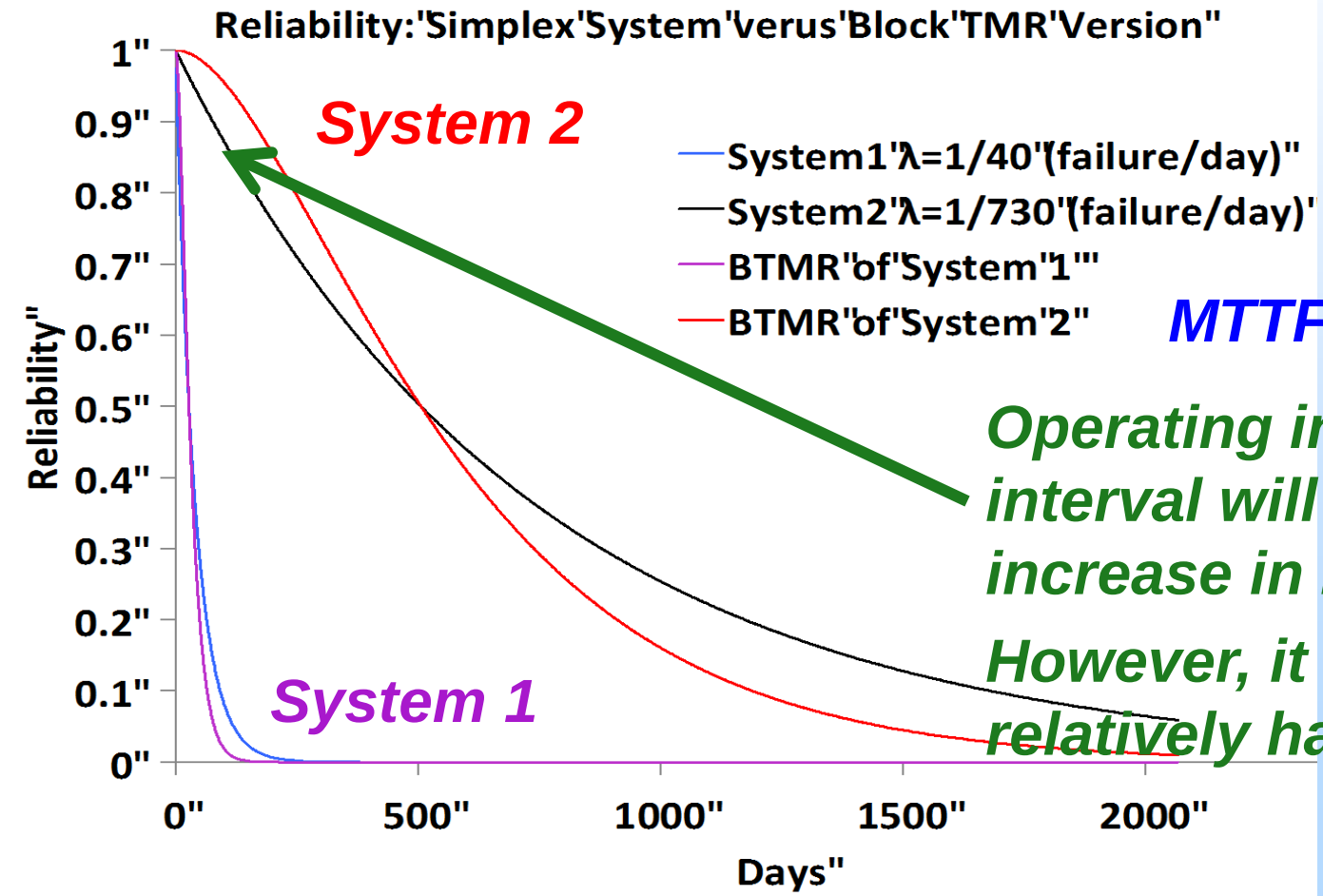


If The System Is Not Flushable, Then BTMR May Not Provide The Expected Level of Mitigation

- **BTMR can work well as a mitigation scheme if the expected MTTF >> expected window of correct operation.**
- **Clarification: If the expected time to failure for one block is less than the required full-availability window, then BTMR doesn't buy you anything.**
- **BTMR can actually be a detriment – complexity, power, and area, and false sense of performance.**

Combine SEU Data and Classical Reliability Models for Mitigation Analysis

Relibility for 1 block (R_{block})	Relibility for BTMR (R_{BTMR})	Mean Time to Failure for 1 block ($MTTF_{block}$)	Mean Time to Failure BTMR ($MTTF_{BTMR}$)
$e^{-\lambda t}$	$3\,e^{-2\lambda t}-2\,e^{-3\lambda t}$	$1/\lambda$	$(5/6\,\lambda)= 0.833/\lambda$



$$\lambda = \frac{Failures}{Time}$$

SEU Data

$MTTF_{BTMR} < MTTF_{Block}$

Operating in this time interval will provide a slight increase in reliability.

However, it will provide a relatively hard design.

What Should be Done If Availability Needs to be Increased?

- **If the blocks within the BTMR have a relatively high upset rate with respect to the availability window, then stronger mitigation must be implemented.**
- **Bring the voting/correcting inside of the modules… bring the voting to the module DFFs.**

DFF: Edge triggered flip-flop ***CL: Combinatorial Logic***

The following slides illustrate the various forms of TMR that include voter insertion in the data-path.

TMR Nomenclature	Description	TMR Acronym
Local TMR	DFFs are triplicated	LTMR
Distributed TMR	DFFs and CL-data-paths are triplicated	DTMR
Global TMR	DFFs, CL-data-paths and global routes are triplicated	GTMR or XTMR

Describing Mitigation Effectiveness Using A Model

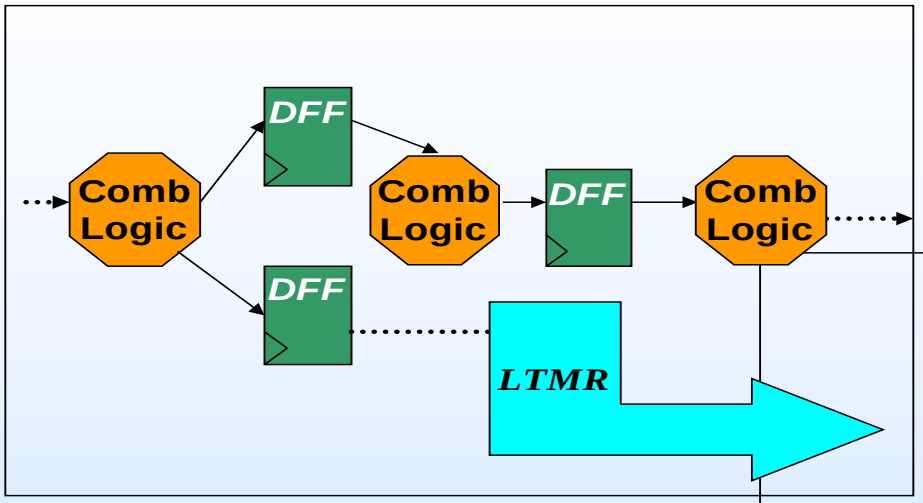
DFF: Edge triggered flip-flop ***CL: Combinatorial Logic***

$$P(fs)_{error} \propto P_{configuration} + \underbrace{P(fs)_{functionalLogic}}_{P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU}} + P_{SEFI}$$

Probability that an SEU in a DFF will manifest as an error in the next system clock cycle

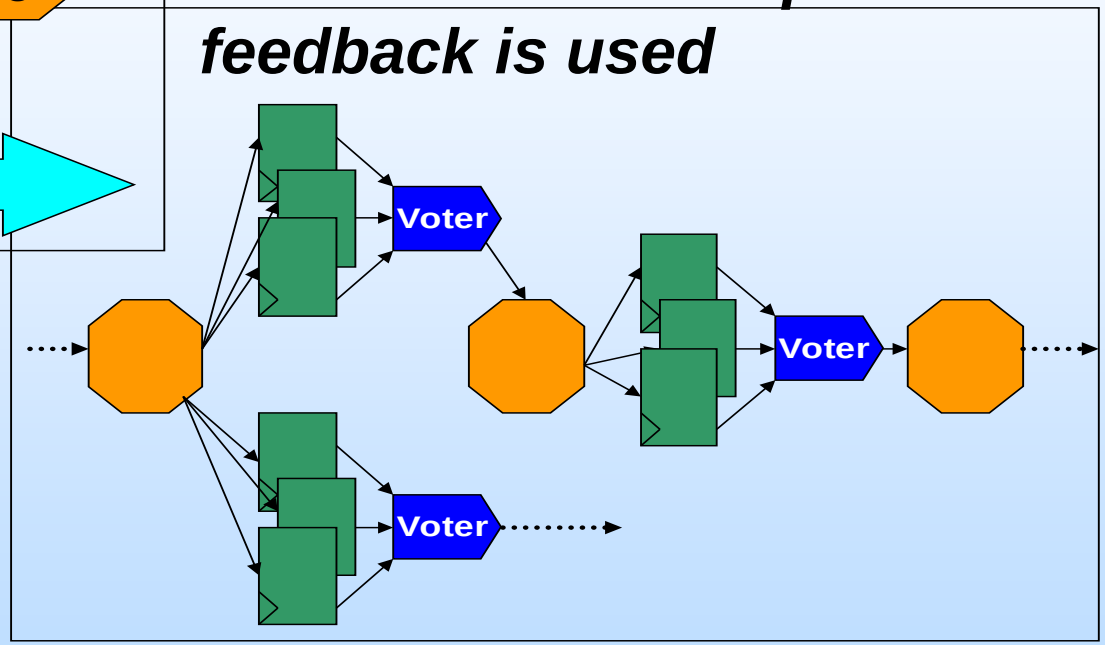
Probability that an SET in a CL gate will manifest as an error in the next system clock cycle

Local Triple Modular Redundancy (LTMR)



LTMR masks upsets from DFFs

and corrects DFF upsets if feedback is used

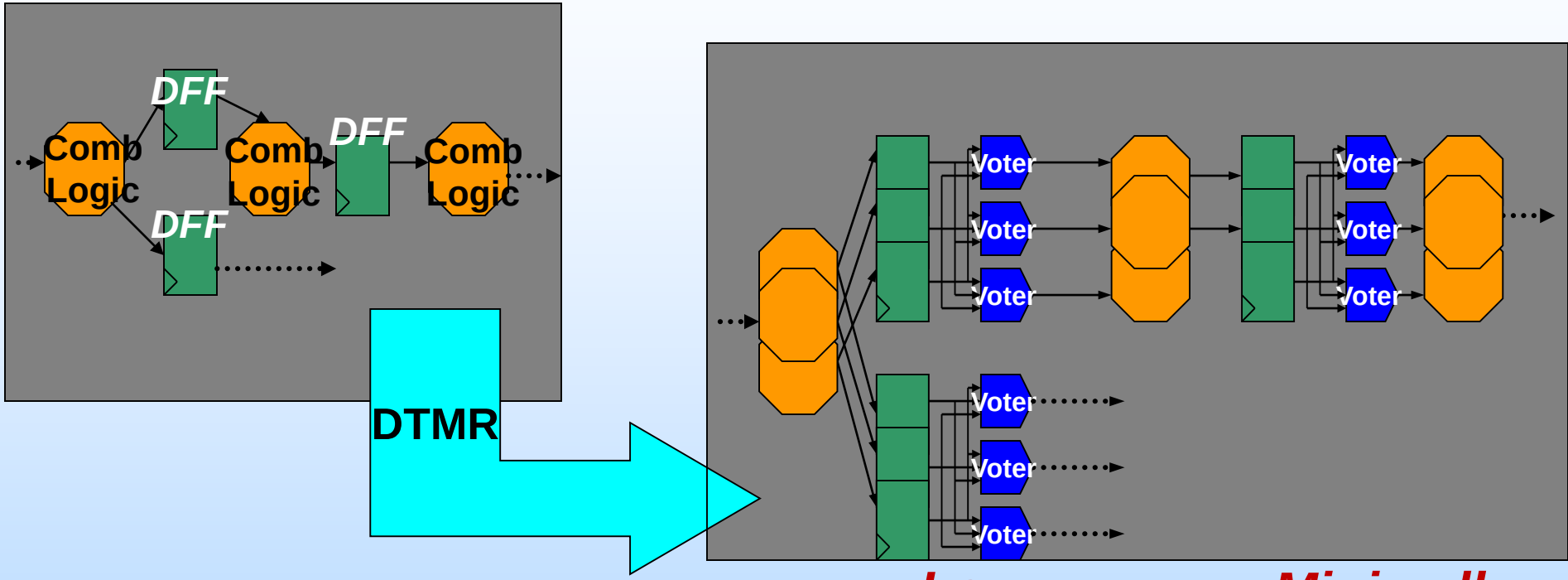


Only the DFFs are triplicated and mitigated

$$P(fs)_{error} \propto P_{configuration} + \underbrace{P(fs)_{functionalLogic}}_{0} + P_{SEFI}$$

$$P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU}$$

Distributed Triple Modular Redundancy (DTMR): DFFs + Data Paths
All DFFs with Feedback Have Voters



$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEE}$$

Low

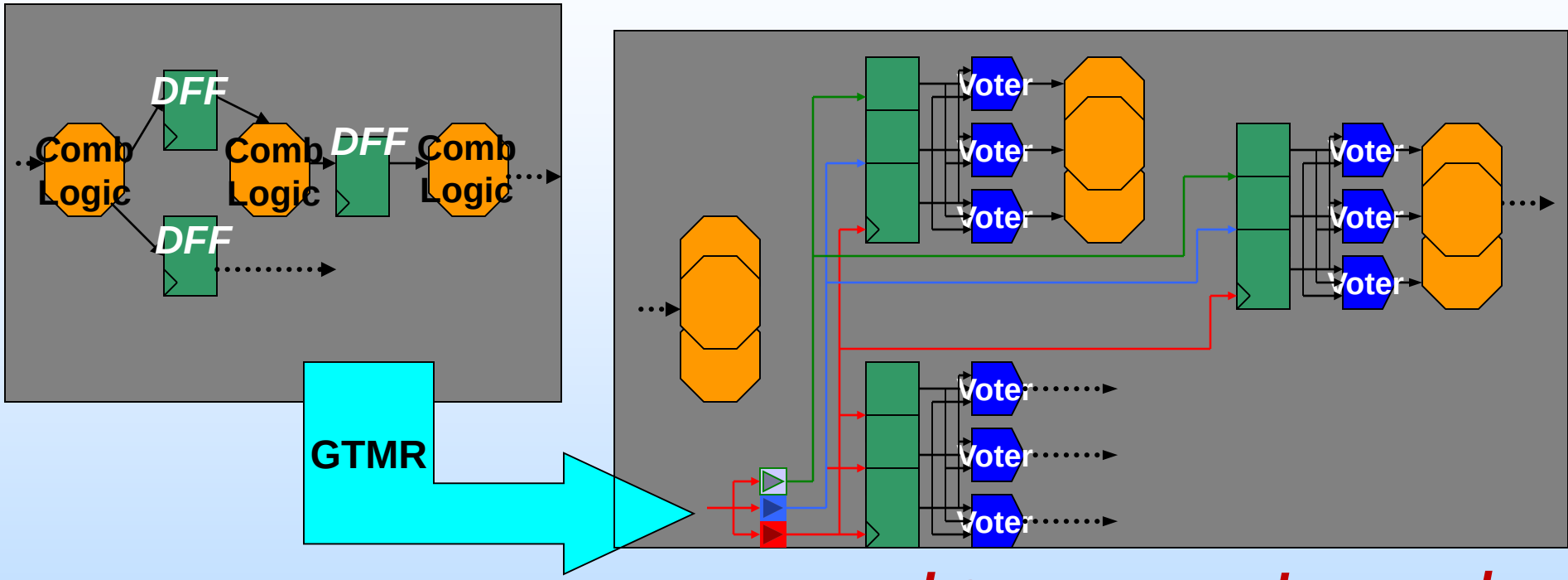
Minimally Lowered

$$P(fs)_{DFFSEU \rightarrow SEU} + P(f_s)_{NET \rightarrow SEU}$$

0

Low

Global Triple Modular Redundancy (GTMR):DFFs + Data Paths + Global Routes All DFFs with Feedback Have Voters



$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEFI}$$

Low Lowered

$$P(fs)_{DFFSEU \rightarrow SEU} + P(f_s)_{NET \rightarrow SEU}$$

Low Low

Theoretically, GTMR Is The Strongest Mitigation Strategy… BUT…

- **Triplicating a design and its global routes takes up a lot of power and area.**
- **Generally performed after synthesis by a tool– not part of RTL.**
- **Skew between clock domains must be minimized such that it is less than the feedback of a voter to its associated DFF:**
  - **Does the FPGA contain enough low skew clock trees? (each clock + its synchronized reset)x3.**
  - **Limit skew of clocks coming into the FPGA.**
  - **Limit skew of clocks from their input pin to their clock tree.**
- **Difficult to verify.**

Currently, What Are The Biggest Challenges Regarding Mitigation Insertion?

- **Tool availability.**
- **User's are not selecting the correct mitigation scheme for their target FPGA.**

FPGA Type	LTMR	DTMR	GTMR
Commercial Antifuse			
Antifuse+LTMR			
Commercial SRAM			
Commercial Flash			
Hardened SRAM			

General Recommendation

Not Recommended but may be a solution for some situations

Will not be a good solution

User versus Embedded Mitigation

- A subset of user inserted mitigation strategies have been presented.
- None of the strategies are 100% fail-safe.
- Depending on the project requirements, and the target device's SEU susceptibility, the most efficient mitigation strategy should be selected.
- The following short courses will provide information regarding FPGA devices that contain embedded mitigation.
- In most cases, devices with embedded mitigation do not require additional (user inserted) mitigation.

Concerns and Challenges for Mitigation Insertion

- User insertion of mitigation strategies in most FPGA devices has proven to be a challenging task because of reliability, performance, area, and power constraints.
  - Difficult to synchronize across triplicated systems,
  - Mitigation insertion slows down the system.
  - Can't fit a triplicated version of a design into one device.
  - Power and thermal hot-spots are increased.
- The newer devices have a significant increase in gate count and lower power. This helps to accommodate for area and power constraints while triplicating a design. However, this increases the challenge of module synchronization.
- Embedded mitigation has helped in the design process. However, it is proving to be an ever-increasing challenge for manufacturers.

Summary

- FPGA devices have become a lucrative alternative to ASICs.
- For critical applications, mitigation may be required.
- Determine the correct mitigation scheme for your mission while incorporating given requirements:
  - Understand the susceptibility of the target FPGA and how it responds to other devices.
  - Investigate if the selected mitigation strategy is compatible to the target FPGA.
  - Calculate the reliability of the mitigation strategy to determine if the final system will satisfy requirements.
- Although it is desirable from a user's perspective to have embedded mitigation, cost seems to be driving the market towards unmitigated commercial FPGA devices. Hence, it will be necessary for user's to familiarize themselves with optimal mitigation insertion and usage.