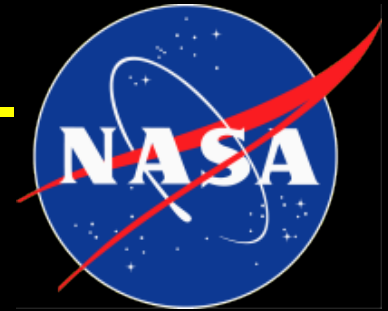


Engineering Trade-off Considerations Regarding Design-for-Security, Design- for-Verification, and Design-for-Test



Melanie Berg
AS&D in Support of NASA/GSFC
Melanie.D.Berg@NASA.gov

Kenneth LaBel
NASA/GSFC
Kenneth.A.LaBel@NASA.gov



Acronyms

- Application specific integrated circuit (ASIC)
- Advanced Encryption Standard (AES)
- Agile Mixed Signal (AMS)
- ARM Holdings Public Limited Company (ARM)
- Asynchronous assert synchronous de-assert (AASD)
- Automotive Electronics Council (AEC)
- Block random access memory (BRAM)
- Built-in-self-test (BIST)
- Bus functional Model (BFM)
- Clock domain crossing (CDC)
- Combinatorial logic (CL)
- Commercial off the shelf (COTS)
- Complementary metal-oxide semiconductor (CMOS)
- Configurable Logic Block (CLB)
- Configuration Management (CM)
- Controller Area Network (CAN)
- Correct Coding Initiative (CCI)
- Design for Reliability (DFR)
- Design for Security (DFS)
- Design for Test(DFT)
- Design for Verification (DFV)
- Digital Signal Processing (DSP)
- Direct Memory Access (DMA)
- Double Data Rate (DDR3 = Generation 3; DDR4 = Generation 4)
- Edge-triggered flip-flops (DFFs)
- Electronic Design Automation (EDA)
- Electronic Design Interchange Format (EDIF)
- Equipment Monitor And Control (EMAC)
- Equivalence Checking (EC)
- Error-Correcting Code (ECC)
- Evolutionary Digital Filter (EDF)
- Field programmable gate array (FPGA)
- Floating Point Unit (FPU)
- Global Industry Classification (GIC)
- Gate Level Netlist GLN)
- Global Route (GR)
- Hardware Design Language (HDL)
- High Performance Input/Output (HPIO)
- High Pressure Sodium (HPS)
- High Speed Bus Interface (PS-GTR)
- Input – output (I/O)
- Intellectual Property (IP)
- Inter-Integrated Circuit (I2C)
- Internal configuration access port (ICAP)
- Joint test action group (JTAG)
- Kilobyte (KB)
- Logic equivalence checking (LEC)
- Look up table (LUT)
- Low Power (LP)
- Low-Voltage Differential Signaling (LVDS)
- Megabit (MB)
- Memory Management Unit (MMU)
- Microprocessor (MP)
- Multi-die Interconnect Bridge (EMIB)
- MultiMediaCard (MMC)
- Multiport Front-End (MPFE)
- Negated AND or NOT AND (NAND)
- Not OR logic gate (NOR)
- On-chip RAM (OCM)
- On-The-Go (OTG)
- Operational frequency (fs)
- Peripheral Component Interconnect Express (PCIe)
- Phase locked loop (PLL)
- Physical unclonable function (PUF)
- Place and Route (PR)
- Power on reset (POR)
- Processor (PC)
- Random Access Memory (RAM)
- Register transfer language (RTL)
- Reliability (R)
- Reliability of BRAM (RBRAM)
- Reliability of configuration ($R_{\text{Configuration}}$)
- Reliability of configurable logic block (R_{CLB})
- Reliability of global routes (R_{GL})
- Reliability of hidden logic ($R_{\text{HiddenLogic}}$)
- Reliability of operation ($R_{\text{operation}}$)
- Reliability of parametrics ($R_{\text{parametrics}}$)
- Serial Peripheral Interface (SPI)
- Serial Quad Input/Output (QSPI)
- Static random access memory (SRAM)
- System Memory Management Unit (SMMU)
- System on a chip (SOC)
- Temperature (Temp)
- Transceiver Type (GTH/GTY)
- Transient width (twidth)
- Ultra Random Access Memory (UltraRAM)
- Universal Asynchronous Receiver/Transmitter (UART)
- Universal Serial Bus (USB)
- Very High Speed Integrated Circuits (VHSIC)
- VHSIC Hardware Design Language (VHDL)
- Watchdog Timer (WDT)

Motivation



ASIC: Application specific integrated circuit

FPGA: field programmable gate array

- **The United States government has identified that ASIC/FPGA hardware circuits are at risk from a variety of adversary attacks.**
- **As an effect, system security and trust can be compromised.**
- **The scope of this tutorial pertains to potential vulnerabilities and countermeasures within the ASIC/FPGA design cycle.**
- **The presentation demonstrates how design practices can affect risk for an adversary to:**
 - Change circuitry,
 - Steal intellectual property, or
 - Listen to data operations.
- **An important portion of the design cycle is assuring the hardware is working as specified or as expected. This is accomplished by extensively testing the target design.**
- **It has been shown that well established schemes for test coverage enhancement (design-for-verification (DFV) and design-for-test (DFT)) can create conduits for adversary accessibility.**
- **As a result, it is essential to perform a trade between robust test coverage versus reliable design implementation.**

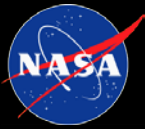


Goals

V&V: Verification and validation

- **Explain conventional design practices and how they affect risk : design-for-reliability (DFR), design-for-verification (DFV), design-for-test (DFT), and design-for-security (DFS).**
- **Review adversary accessibility points due to DFV and DFT circuitry insertion (back door circuitry).**
- **Describe common engineering trade-off considerations for V&V versus adversary threats.**
- **Discuss risk analysis.**

Field Programmable Gate Array (FPGA) Basics





The FPGA Design Process

SRAM: static random access memory

- **Goal: A final product requires an end-user to acquire an FPGA base-array from a manufacturer.**
- **After acquisition, the end-user will customize the FPGA base-array with a specified design.**
- **Process:**
 - Manufacturers create base-arrays that contain existing configurable logic cells plus other complex intellectual property (IP).
 - End-Users acquire FPGA base-arrays with the intent to map designs into the devices' existing logic cells.
 - The output of the end-user's mapping process is used to configure (program) the FPGA's existing logic cells.
 - The FPGA is configured by:
 - Downloading a bitstream to the FPGA's configuration memory (SRAM or Flash), or
 - Blowing configuration fuses (anti-fuse).

Vulnerabilities and The FPGA Design Process



- **Vulnerabilities can be created during the manufacturer design cycle and the end-user design cycle that persist in their final products.**
 - These vulnerabilities create avenues for adversary infiltration.
 - It is important to note that potential adversary access does not definitely lead to system malfunction or information leakage.
 - Subsequently, a combination of threat, implemented mitigation, and outcome must be studied.
- **There are design choices that cause systems to be less vulnerable in some areas, while increasing vulnerabilities in others.**
- **Trade-offs are made to determine if the design choices should be implemented; and if mitigation is required.**

FPGA Manufacturer Design Cycle versus End-User Design Cycle



- Design of the FPGA base-array (ASIC design flow) maps logic onto a blank slate... **flexible design choices.**
- An end-user's FPGA design maps into the target base-array's existing logic cells... **limited design choices.**
- **ASICs require device fabrication – additional challenges:**
 - Reliability of fabrication (fab) process:
 - Stuck-at-faults
 - Transistor lifetime
 - Routing (net) lifetime
 - Process variations
 - Device timing and other electrical parametrics
 - Requires high levels of V&V post fabrication for product assurance.
- **Benefit of using existing logic: once users buy the device, they do not have to go through a costly fabrication process with its additional reliability challenges. Manufacturer is expected to perform post-fab assurance.**
- **Con of using existing logic... area, power, and general performance are lessened.**

Vulnerabilities within The FPGA End-User Design Cycle

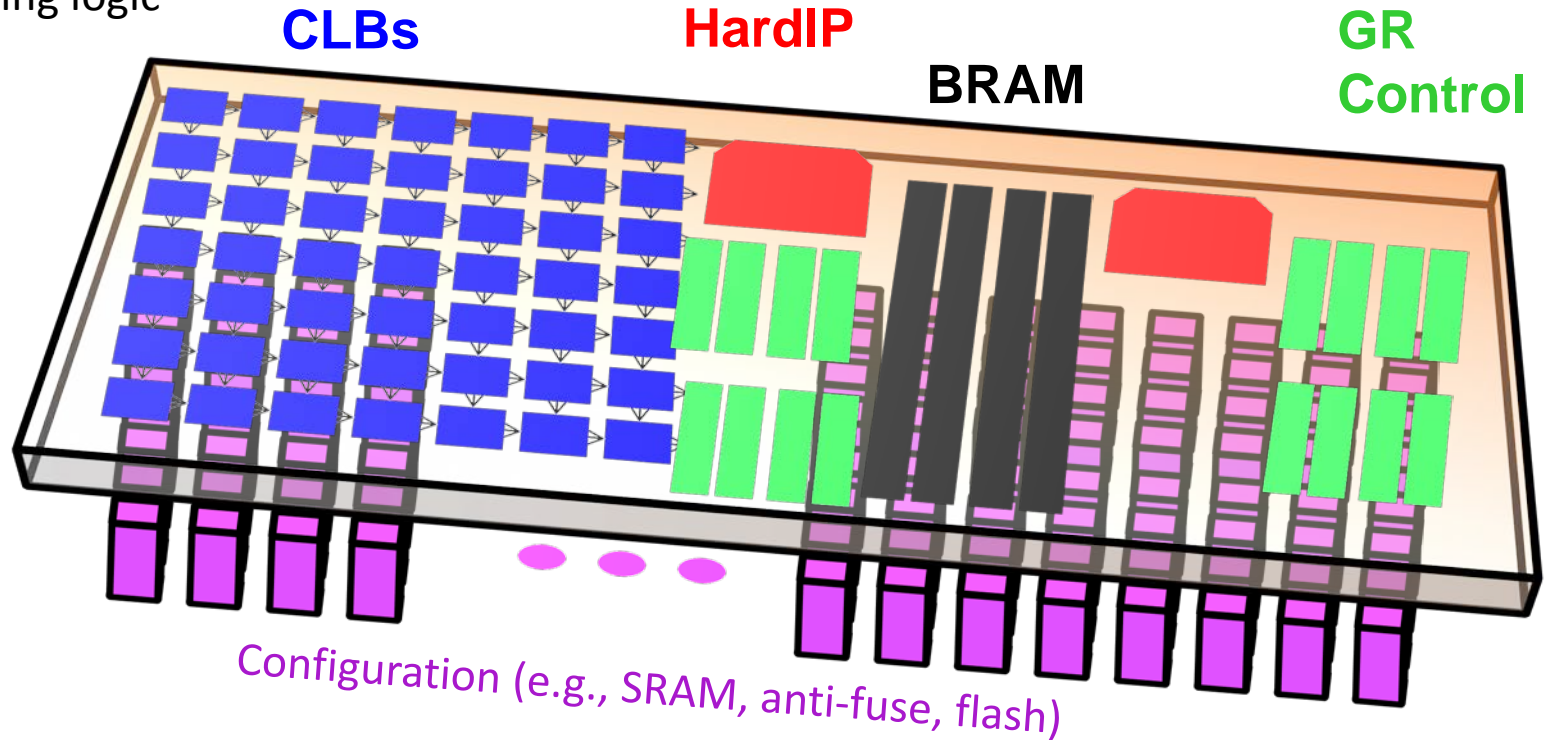


- **End-users buy FPGA devices (base-arrays):**
 - Many of the manufacturers' vulnerabilities can propagate to the end-users.
 - It is important to understand these vulnerabilities so that the end-user can add the appropriate mitigation if necessary.
- **When evaluating vulnerabilities to adversary infiltration, it is essential to assess the full ecosystem of the design cycle (personnel, equipment, storage schemes, data transfer, etc.)**
- **However, the scope of this presentation is design. Only design specific vulnerabilities, threats, and countermeasures (mitigations) will be discussed.**

Not every susceptibility is a vulnerability!

Understanding What Is Inside of An FPGA

Complex routing logic everywhere.



Configurable logic block: (CLB)

Block random access memory: (BRAM)

Intellectual property: (IP); e.g., micro processors, digital signal processor blocks (DSP), PUF, Key control, etc.,...

Global Routes: (GR)

Reliability: R

$$R_{operation} \propto R_{Configuration} + R_{CLB} + R_{BRAM} + R_{GL} + R_{IP} + R_{HiddenLogic} + R_{parametrics}$$

Reliable operation depends on a variety of parameters.



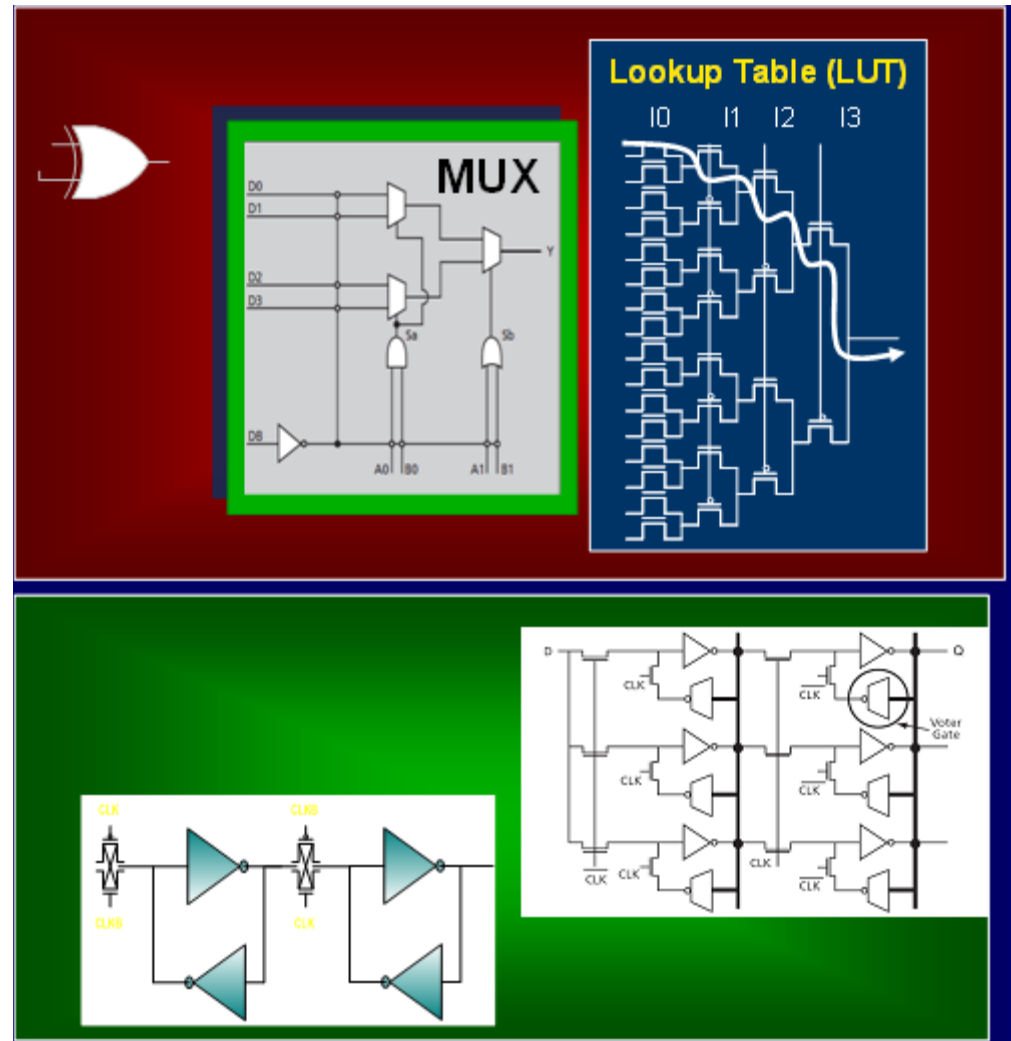
Cannot Evaluate Susceptibilities/Vulnerabilities without Understanding What Is Inside An FPGA

- Data-path glitching **Configuration**
- Change of state **End-user data-path logic (CLB)**
- Global route glitching **Global routes (GR)**
- Configuration corruption **Embedded (hidden) logic**
- Insertion or deletion of expected circuitry
- Current jumps or increases (contention)
- Single event upsets

Each FPGA has different susceptibilities. Important to understand mission requirements to determine vulnerabilities, differentiate per FPGA device, and mitigate appropriately.

Example: FPGA Component Libraries - Basic Designer Building Blocks

- **Combinatorial logic blocks**
 - Vary in complexity
 - Vary in block I/O
- **Sequential Memory blocks (DFF)**
 - Uses global Clocks
 - Uses global Resets
 - May have mitigation
- **Device I/O**
 - Direction
 - Standard

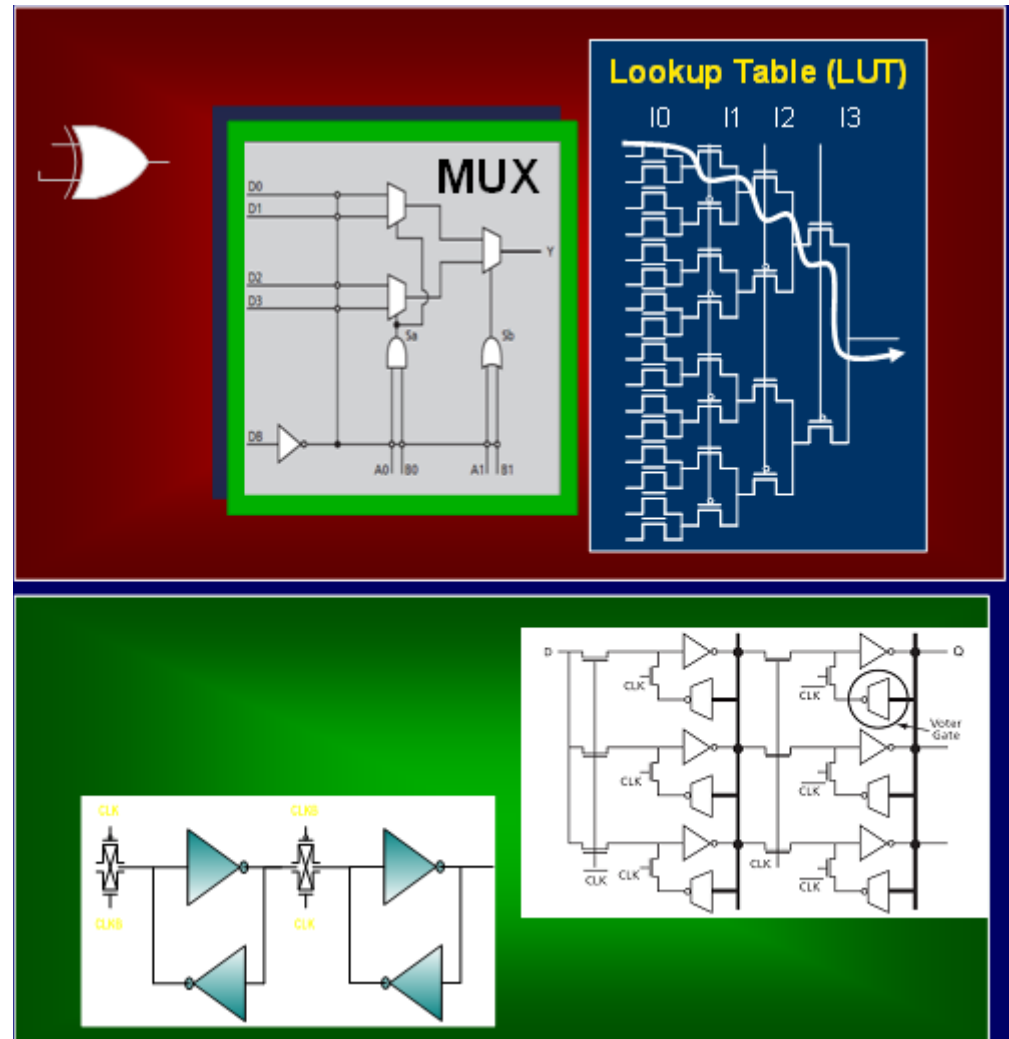


Building Blocks: Susceptibilities and End-User Mitigation



- Designer building blocks are replicated thousands of times within an FPGA device.
- Although it is possible for an adversary to change a cell, **due to the V&V performed by the manufacturer and the widespread usage, it is an unlikely point of attack.**
- Countermeasures: End-user V&V with parametric analysis (current, hotspots, signal leakage, etc.)

Verification and validation (V&V)



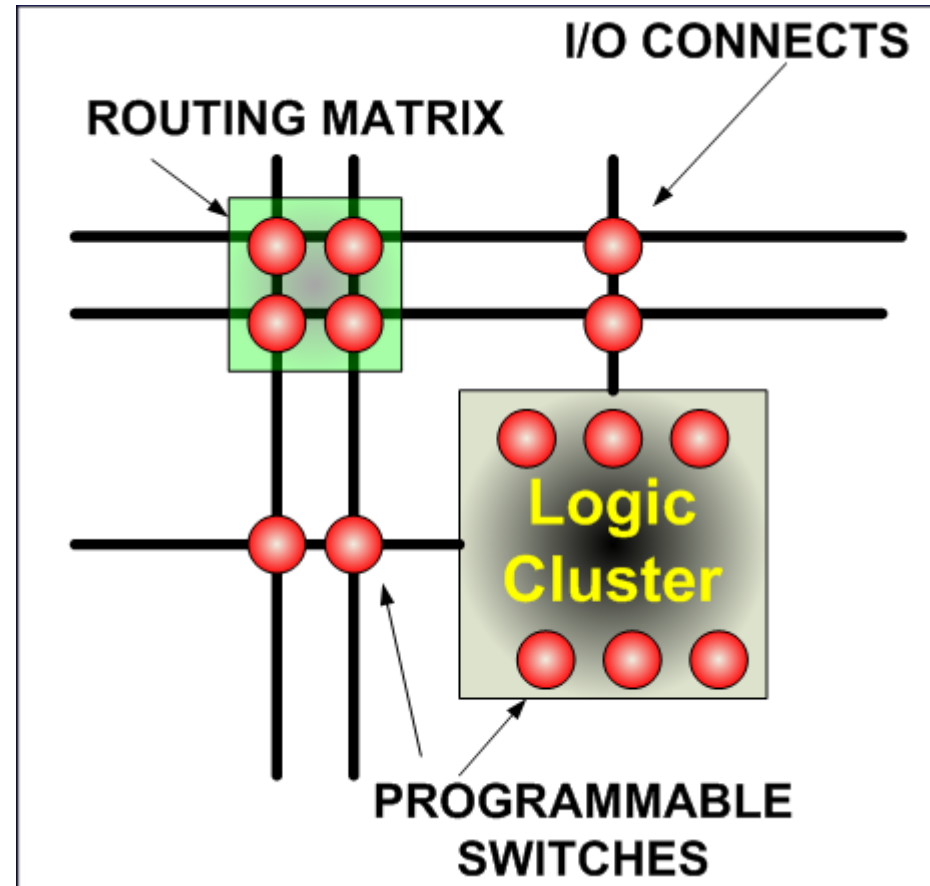
HDL

FPGA MAPPING

Configuration

- Configuration defines arrangement of pre-existing logic via programmable switches:
 - Functionality (logic cluster)
 - Connectivity (routes)
- Programming Switch Types:
 - **anti-fuse:** One time Programmable (OTP)
 - **SRAM:** Reprogrammable (RP)
 - **Flash:** Reprogrammable (RP)

Configuration technologies vary and are managed differently.



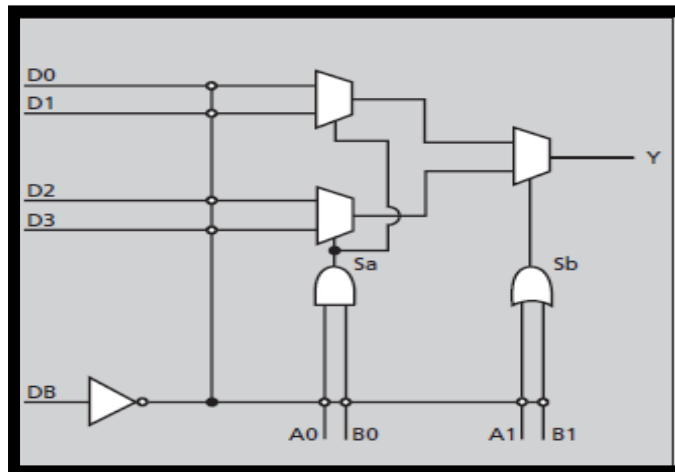
Example: Mapping Combinatorial Logic into Configuration



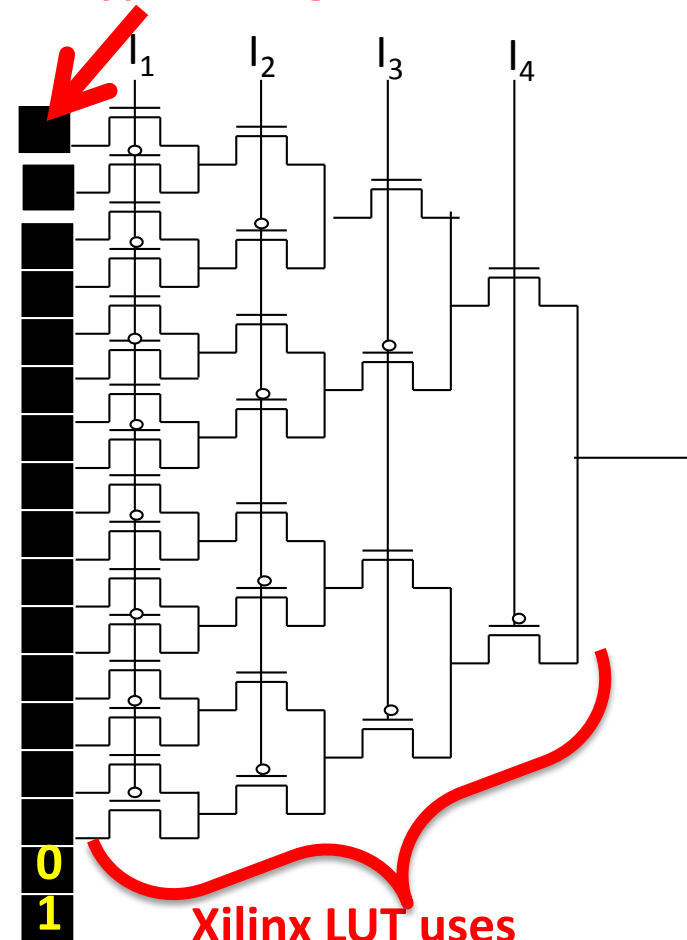
Lookup Table LUT

- Output is affected by inputs after gate delay (t_{dly}).
- Used for computing or routing.
- FPGAs provide blocks of combinatorial logic (library components)... blocks vary per manufacturer.

Actel RTAXs C-CELL requires anti-fuse to select gate mapping.



Xilinx LUT uses SRAM type Configuration.

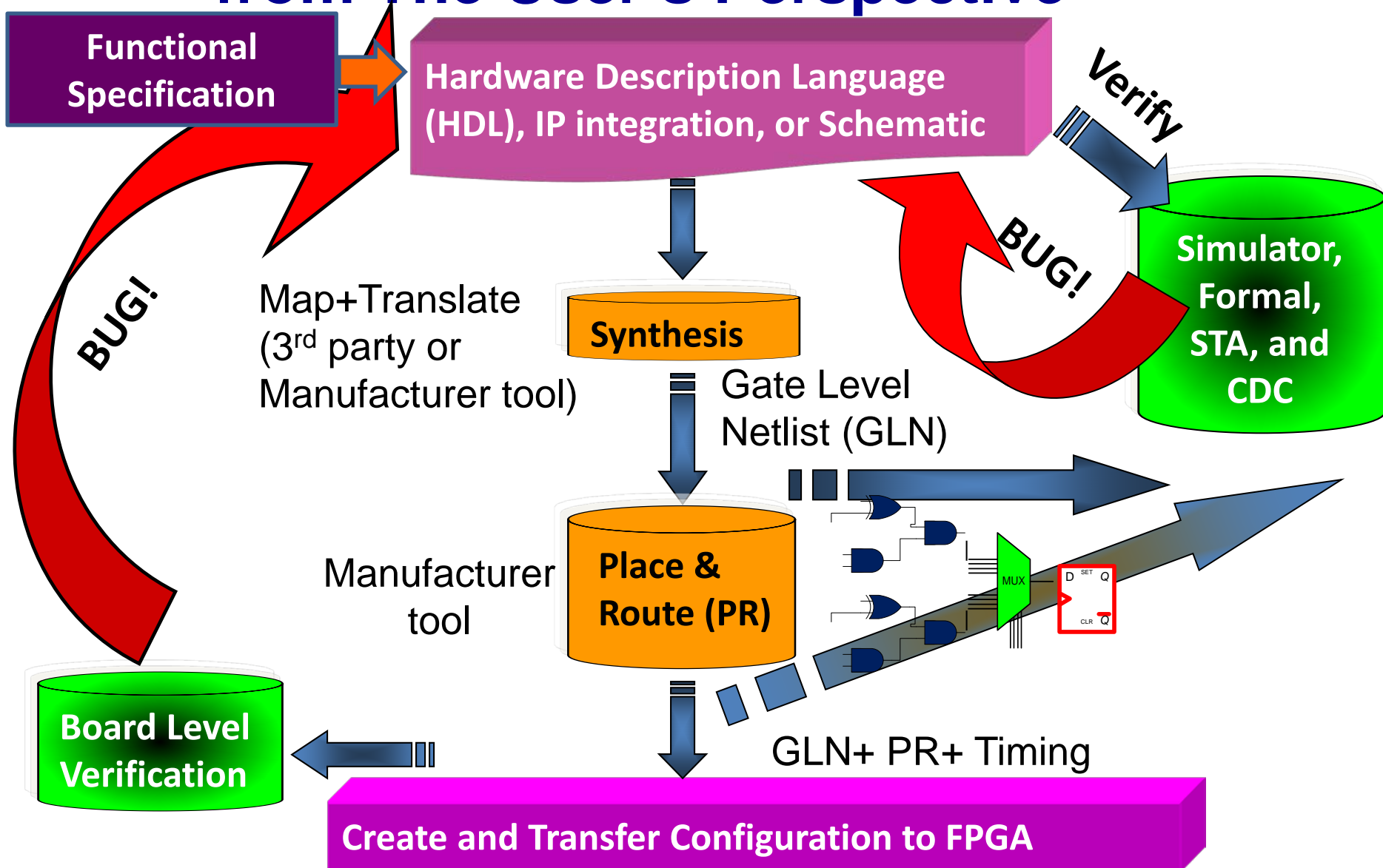


Xilinx LUT uses Pass transistors. THIS IS NOT CONFIGURATION SRAM.

Configuration Vulnerabilities

- **anti-fuse:**
 - Configuration is a hard process.
 - It cannot be changed once programmed.
 - Susceptibilities/vulnerabilities: imaging (reverse engineering), complex process bugs, or lifetime deficiencies.
- **Flash:**
 - Configuration is stored in non-volatile memory (persists after the removal of power).
 - Can be changed.
 - Susceptibilities/vulnerabilities: imaging (reverse engineering) and bitstream manipulation.
- **SRAM**
 - Configuration is stored in volatile memory (does not persist after the removal of power).
 - Requires another component for volatile storage or for remote reconfiguration.
 - Can be changed.
 - Susceptibilities/vulnerabilities: imaging (reverse engineering) , bitstream manipulation, additional component for configuration data storage, potential configuration data transmission, **Single Event Upsets (SEUs).**

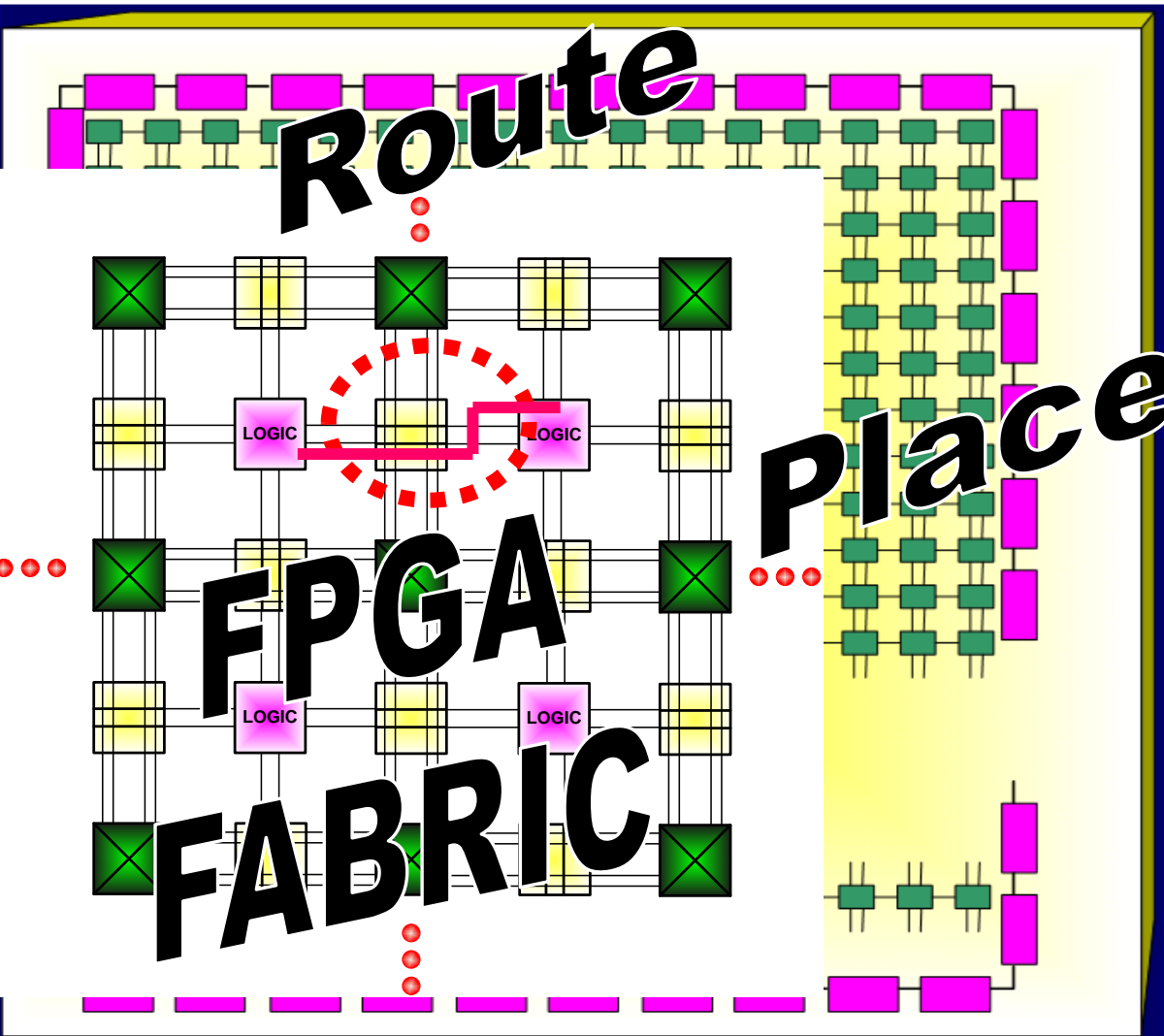
The FPGA Design and Verification Process from The User's Perspective



FPGA End-User Mapping into Existing Logic with Place and Route



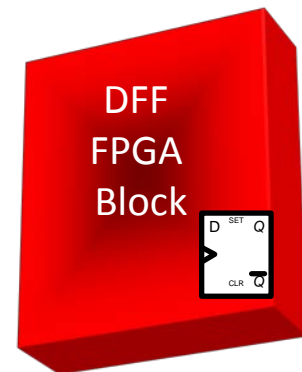
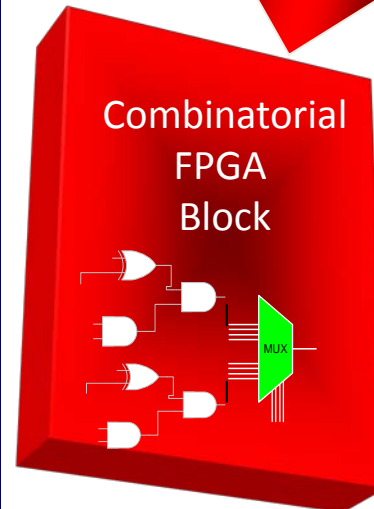
Hardware design language (HDL)



HDL

MAP

INTO FPGA
LIBRARY





FPGA Design is Hardware

- **Reminder: HDL stands for Hardware Description Language.**
- **Misperception that HDL is similar to writing software**
 - The electrical characteristics of the circuit are generally overlooked and designs are improperly implemented.
 - Verification (state-space coverage and transition) is not performed correctly.
 - Identification of vulnerabilities are in accurate.
- **Bottom line: in order for the end-user to create a reliable product, hardware concepts must be incorporated into the design process.**



Design Methodology and Reliable Operation Considerations

HDL: hardware description language

Number of Clock Domains

Clock Balancing

Area

Metastability

Reset Structure

**Long Traces
(charge sharing)**

**I/O Standard
Selection**

Power (Hot-spots)

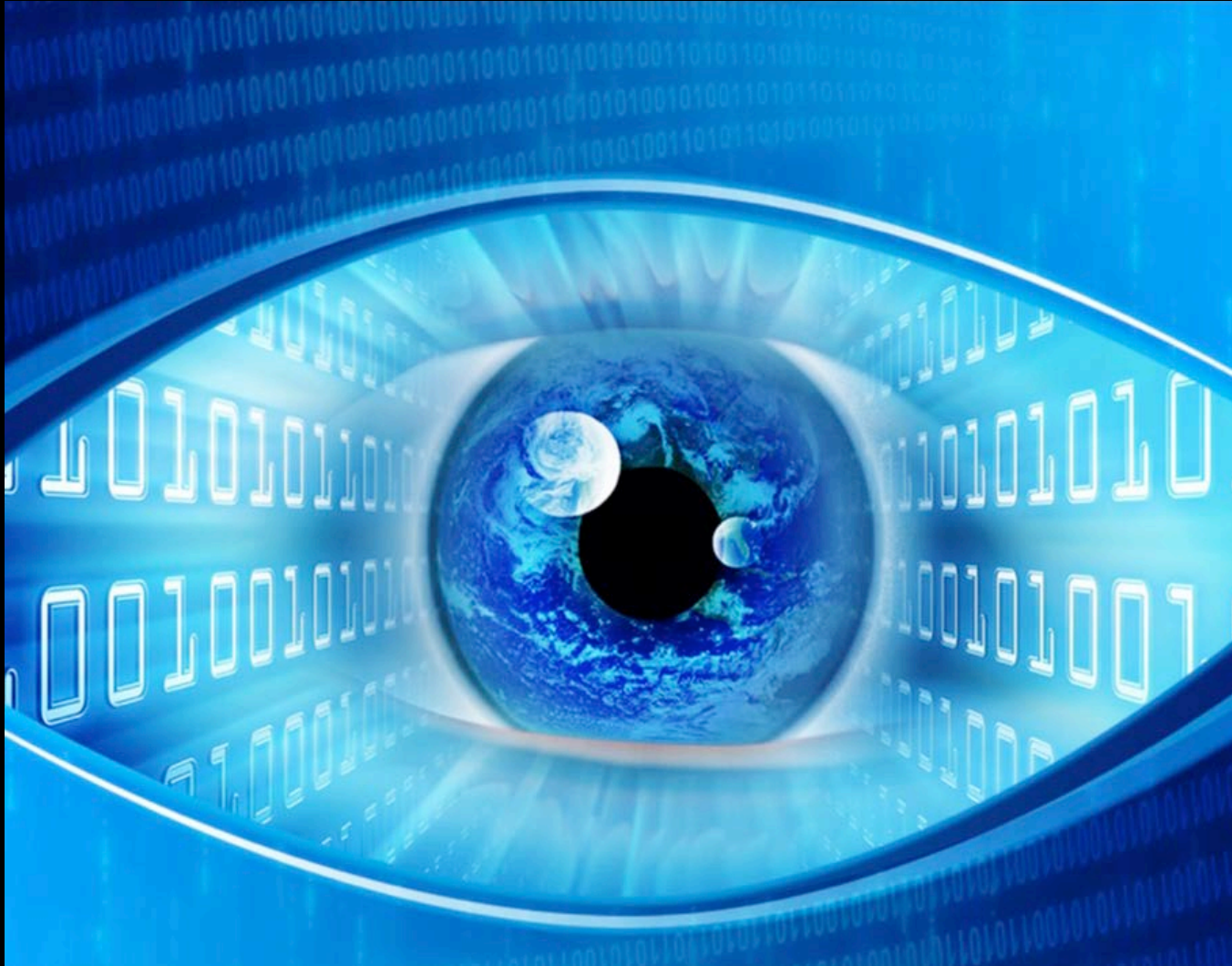
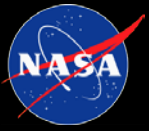
**I/O Rings and
Pin Switching
(ground-bounce)**

**Static Timing
Analysis ...
Setup/hold time
violations (race
conditions)**

**Creation of
Latches versus
Edge-triggered
flip-flops**

**Synthesis tool
interpretation of HDL**

Design-for-Reliability (DFR): Synchronous Design



Introduction to Reliable Design (Synchronous Design)



- **This section establishes requirements and best-practice guidelines for creating reliable digital designs.**
- **Why go through the trouble?**
 - Due to advancements in technology and the resulting increase in device resources, the complexity of digital designs has grown exponentially.
 - In order to bound and manage the complexities of design, engineers must follow practices that yield deterministic system behavior.
- **The design-for-reliability methodology described in this presentation is used at NASA and other critical-application design houses across the world.**

Synchronous Design and Deterministic Behavior



- **Deterministic behavior = controllability and observability.**
- **Deterministic behavior is essential for functional and physical testability:**
 - **Can cause conduits to vulnerabilities if not strictly followed:**
 - **Bad design can create untestable logic (blind spots).**
 - **Bad design can cause the system to easily become unstable.**
 - **Bad design can leave inputs and outputs unprotected.**
 - **Bad design can cause parametric vulnerabilities.**
 - **Can cause conduits to vulnerabilities if deterministic mechanisms are not mitigated.**
 - **Deterministic behavior is easier for an adversary to reverse engineer.**
 - **Design solutions for determinism can cause massive disruption (e.g.: clocks and resets).**
 - **Design solutions for testability can cause access points for adversaries.**



There are many rules a designer must follow for reliable system behavior. Some are contradictory to the concept of security.

Solution: mitigate those components.

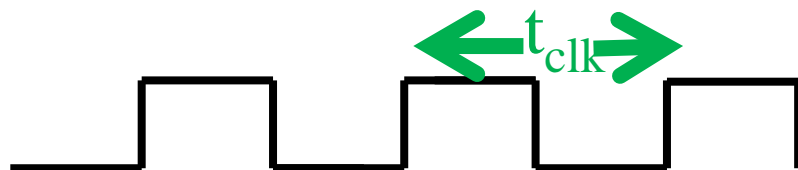


Synchronous Design Building Blocks: Flip-Flops (DFFs) and Combinatorial Logic (CL)

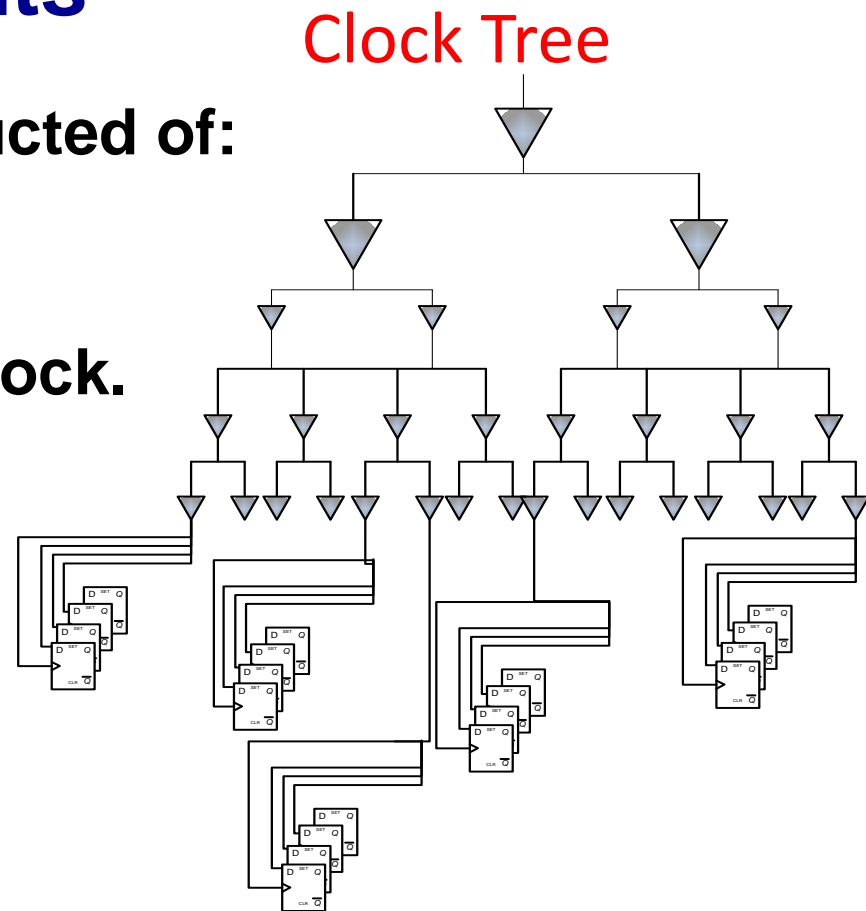
Synchronous Design Data Path Components



- **Design data-paths are constructed of:**
 - Combinatorial Logic (CL)
 - Edge Triggered Flip-Flops (DFFs)
- **All DFFs are connected to a clock.**
- **Clock period:** t_{clk}
- **Clock frequency:** f_s



$$t_{clk} = 1/f_s$$



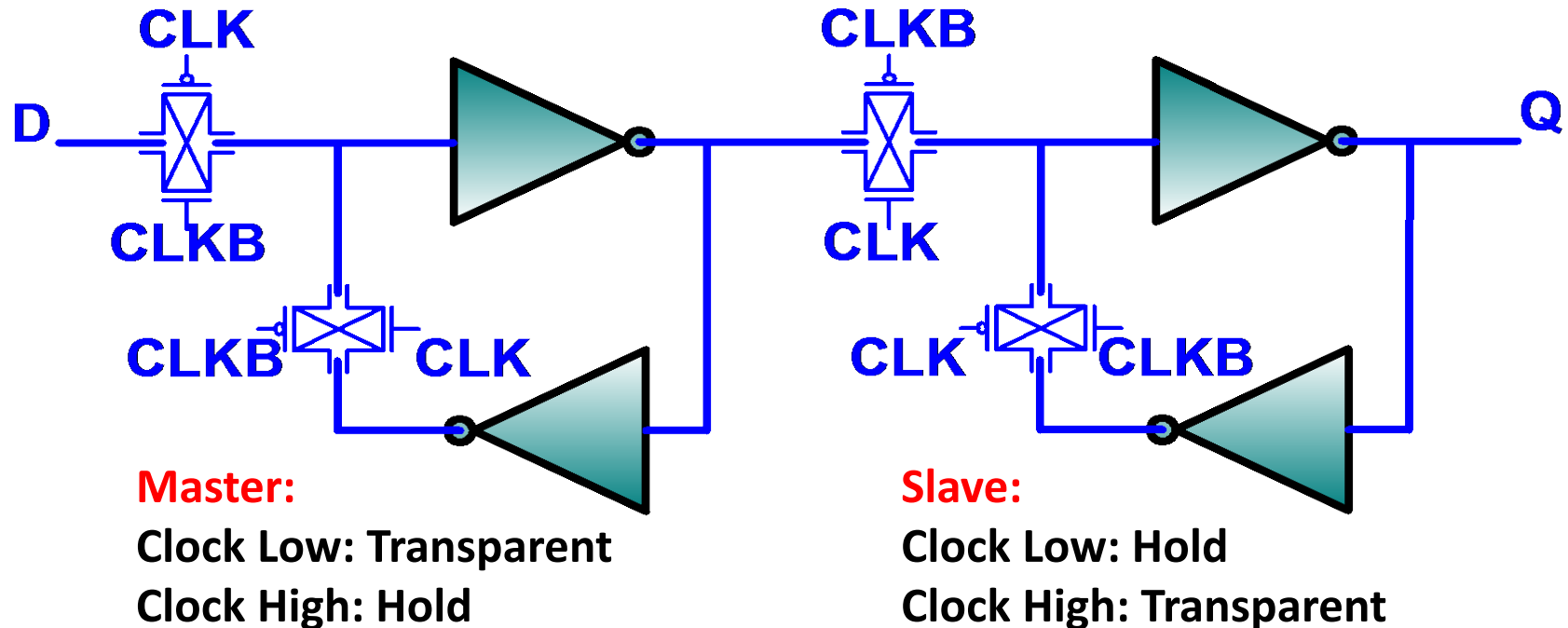
The premise of synchronous design is to compute and hold in a deterministic manor.

Edge Triggered Flip Flops... Creating Deterministic Boundary Points



D input must be settled by rising edge of clock.

Output will only change at rising edge of clock.

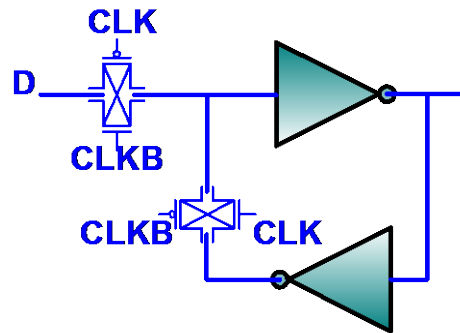


In order to create precise boundary points of state capture, **Latches** are **NOT allowed** in Synchronous designs.

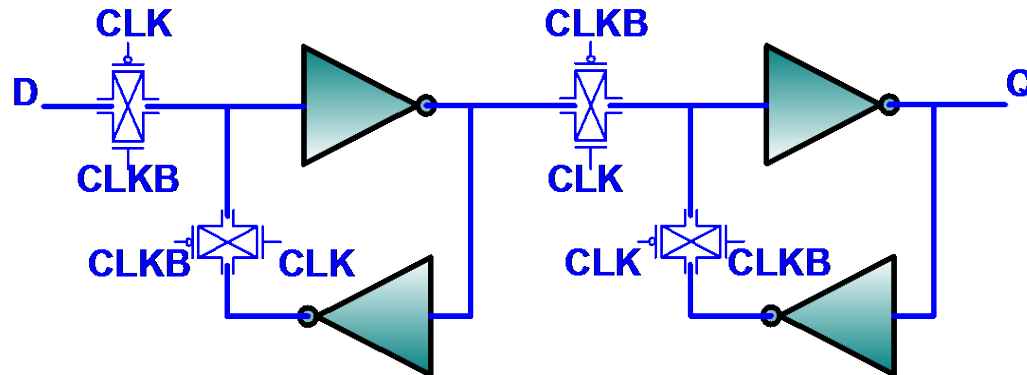
Why are Edge Triggered DFFs Considered Boundary Points and Are Considered Deterministic?



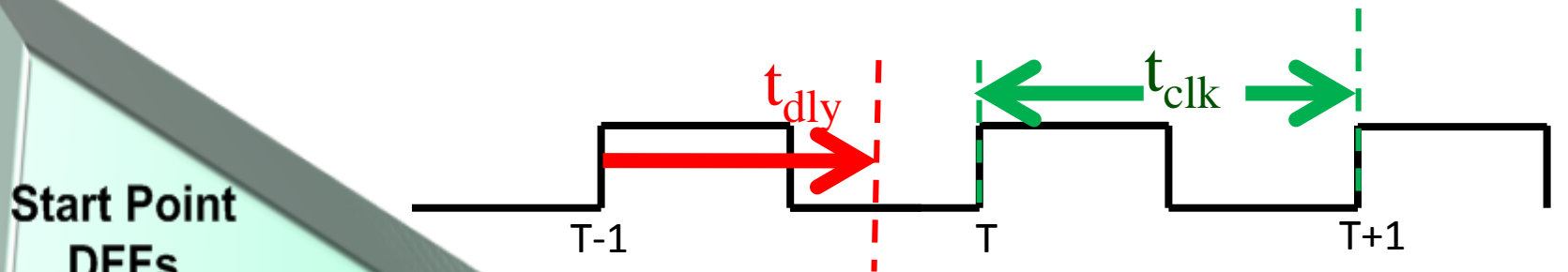
- Latch is checking its input the entire time the clock is low.



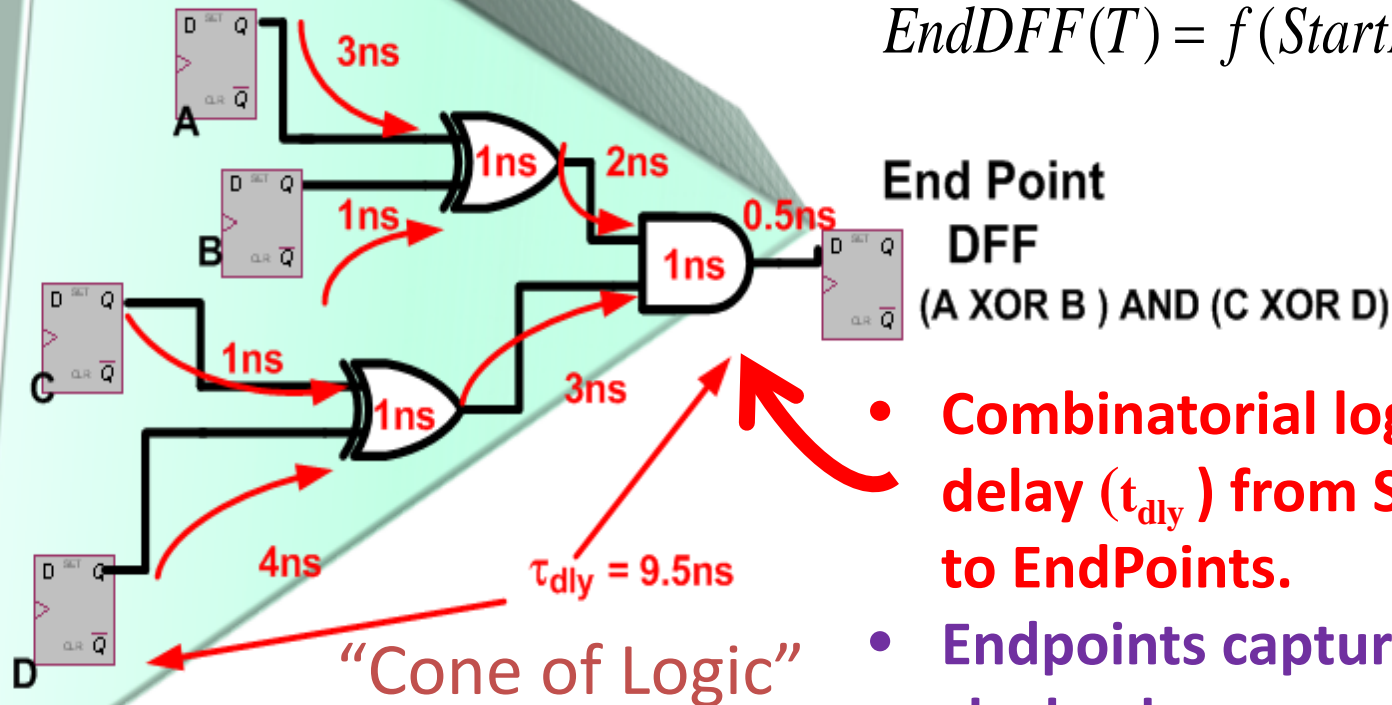
- Edge triggered DFF only samples data exactly at clock edge.



Synchronous System Data Paths: StartPoint DFFs → EndPoint DFFs



$$EndDFF(T) = f(StartDFFs(T-1))$$



- Combinatorial logic create delay (t_{dly}) from StartPoints to EndPoints.
- Endpoints capture only at clock edge.

Every DFF has a cone of logic.



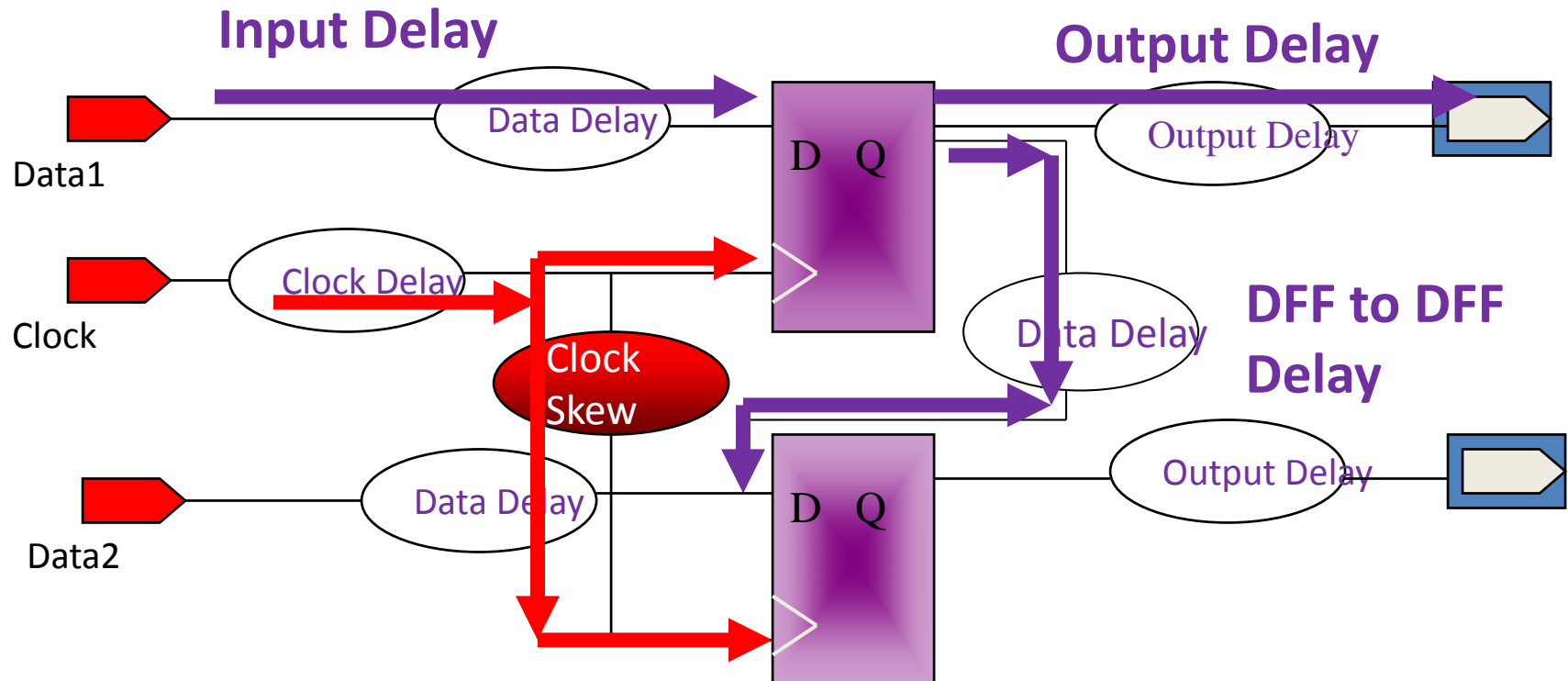
Synchronous Design...Timing and Data Capture with Static Timing Analysis Basics

Static Timing Analysis (STA) Basics

$$t_{dly} < t_{clk} - \text{overhead}$$

Various delays within a Synchronous design:

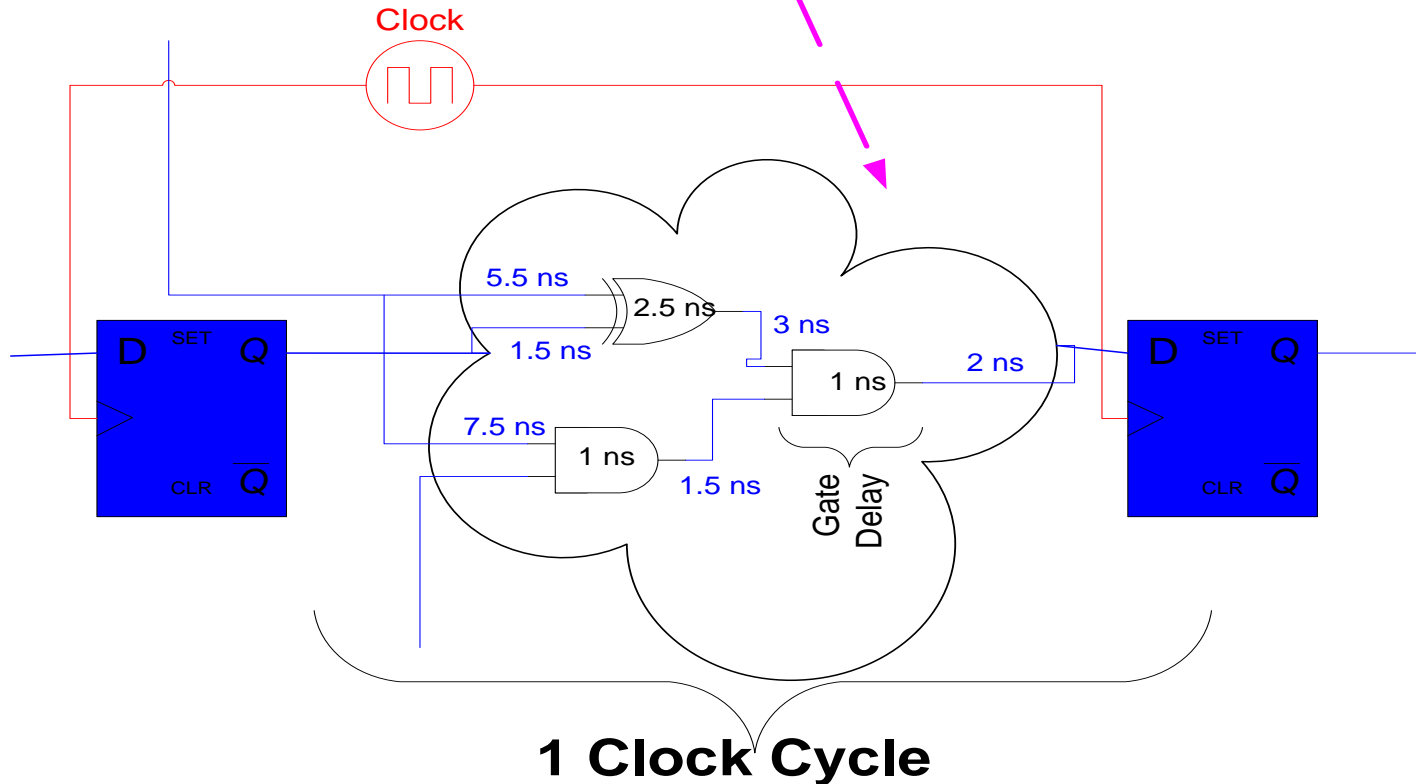
Concept... when will data arrive at a DFF or an Output?



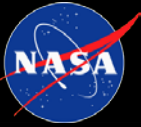
Static Timing Analysis (DFF to DFF)



DFF to DFF Boundary with Combinatorial Logic



Longest Path: 14ns... Clock must have a period longer than 14ns + overhead (temperature, voltage, process variation, and clock jitter).

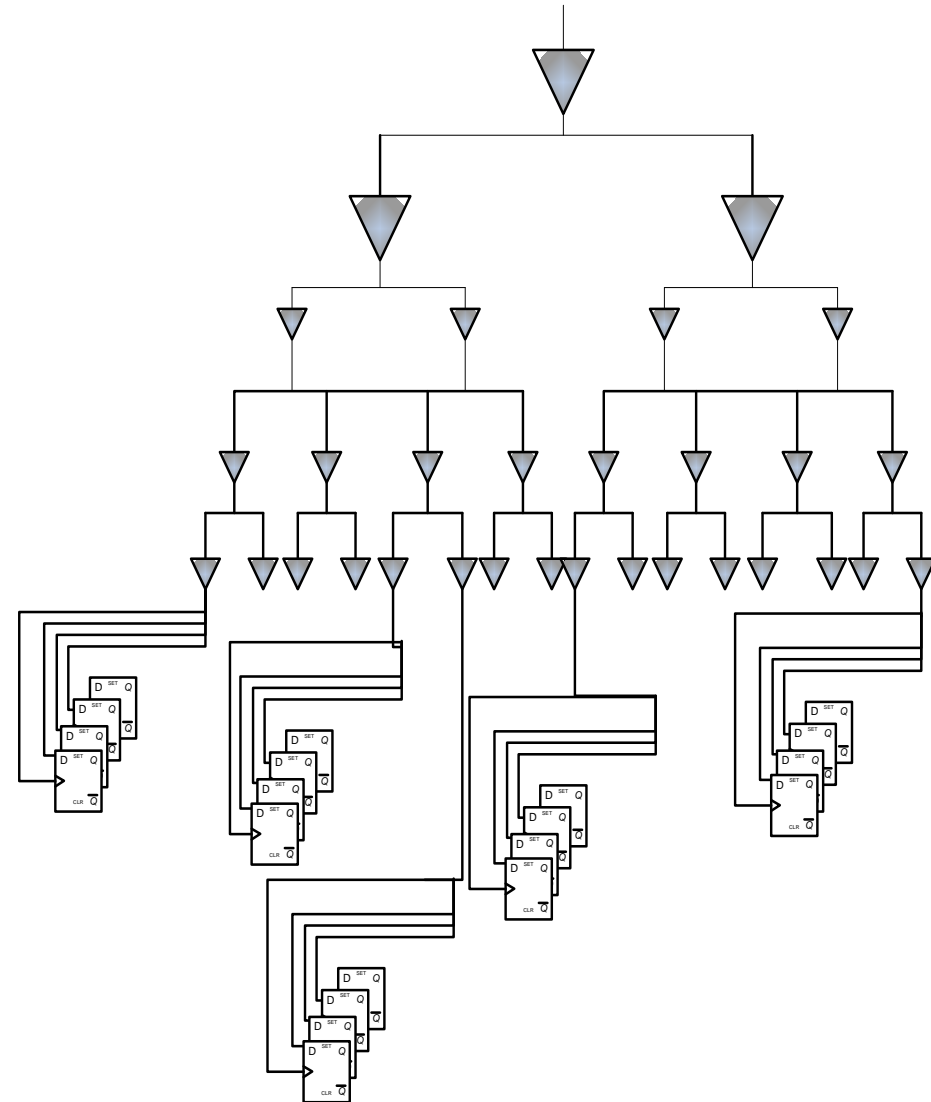


Clocks (Skew, Jitter, and Clock Domain Crossings)

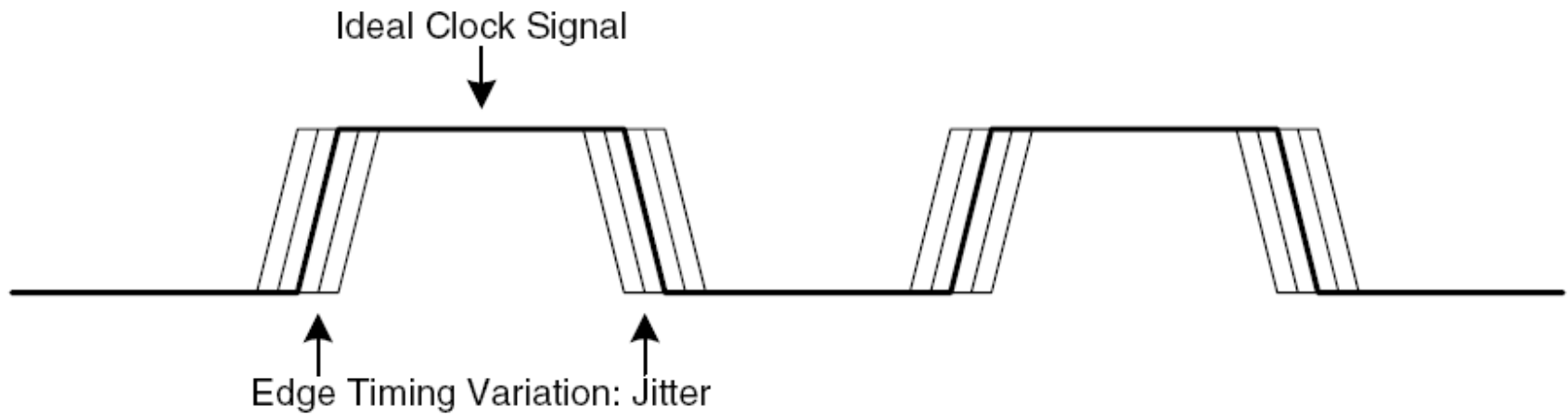
Clock Tree – Clock Connected to every DFF



- **Synchronous Design rule:**
 - All Clocks are on a balanced clock tree.
 - FPGA – use the provided clock tree buffers (global routes)
- **This minimizes skew from DFF to DFF.**
- **However, clock tree buffers are not perfect.**
 - They are very good for closely placed DFFs.
 - However, there is significant skew from DFFs that are placed far apart.
- **Race conditions (or hold time violations will occur if skew is not controlled.**



Clock Jitter



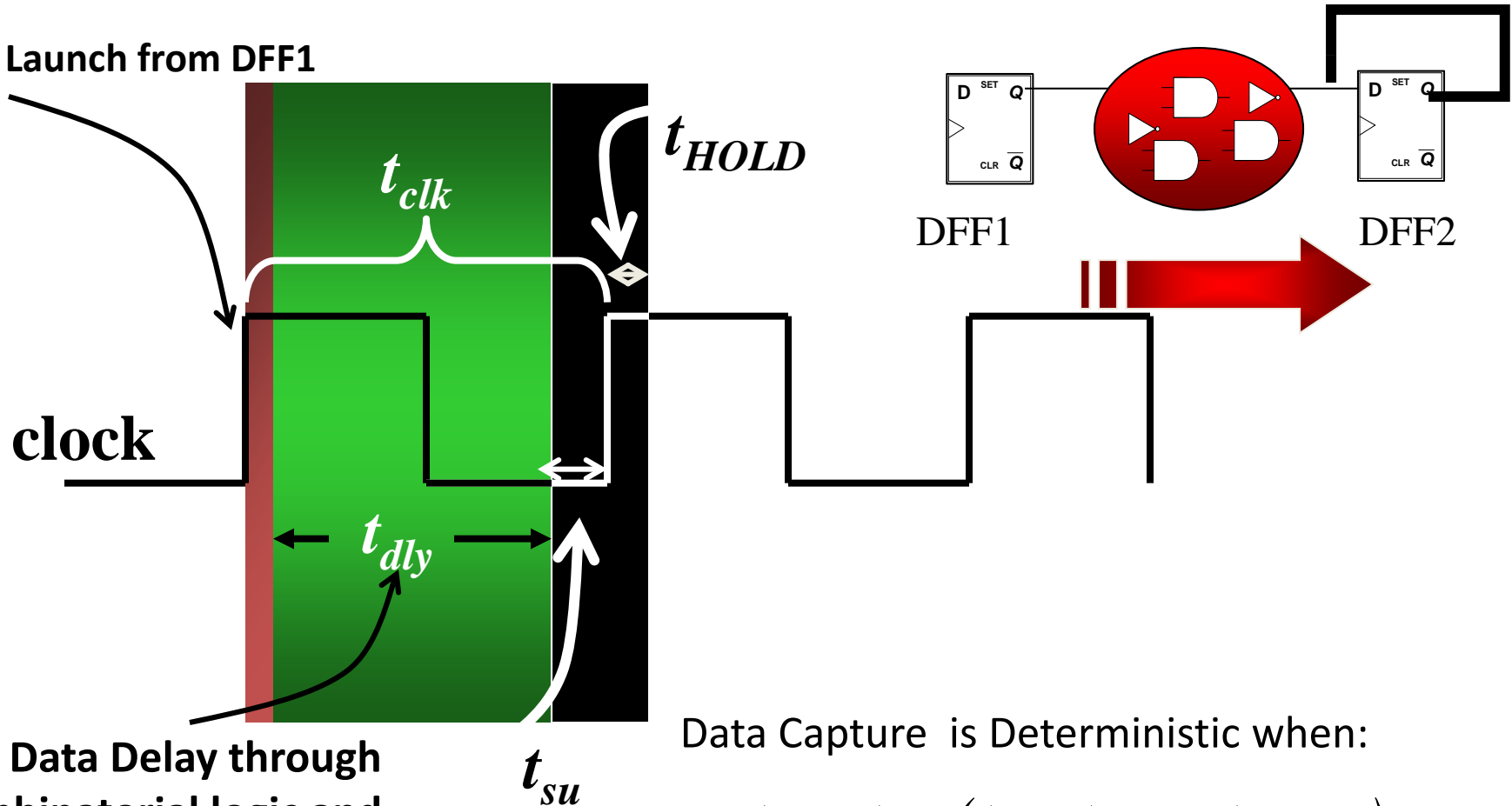
Clock Skew

- **Skew: it is the measurement of the difference in clock arrival time seen at one DFF compared to another DFF**
- **Can cause a synchronous design to become asynchronous due to set-up and hold violations**
- **Clock tree must be balanced to avoid skew – beware of tree connections – should only be to a DFF clock pin (i.e. can not feed combinatorial logic).**
- **Designs that don't use balanced clock trees will most likely contain unpredictable behavior.**

STA: Deterministic Data Capture... Incorporating Skew and Jitter

Data arrival at all DFFs must be stable between setup time (t_{su}) and hold time (t_h)
... or there is potentially metastability in the capturing DFF.

Data Launch from DFF1

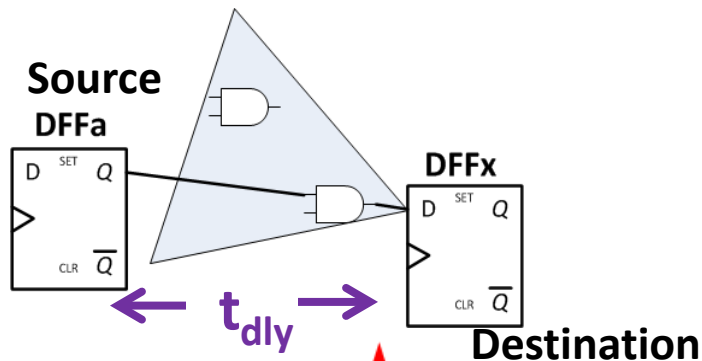


t_{dly} : Data Delay through
combinatorial logic and
routes.

Data Capture is Deterministic when:

$$t_{dly} < t_{clk} - (t_{su} + t_{skew} + t_{margin})$$

Synchronous Data Capture... No Clock Skew

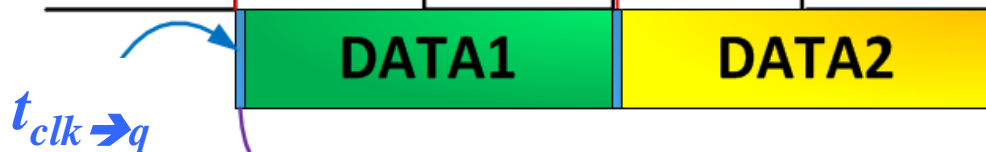


$$t_{clk} > t_{dly} + t_{su} + t_{margin} - t_{skew} \quad \text{Max}$$

$$t_{skew} < t_{dly} + t_{HOLD} + t_{margin} - t_{skew} \quad \text{Min}$$

Both Equations must be satisfied at all times.

Clock at DFFa



Data launched from DFFa

t_{dly}



New Data (after t_{dly})
as seen by DFFx

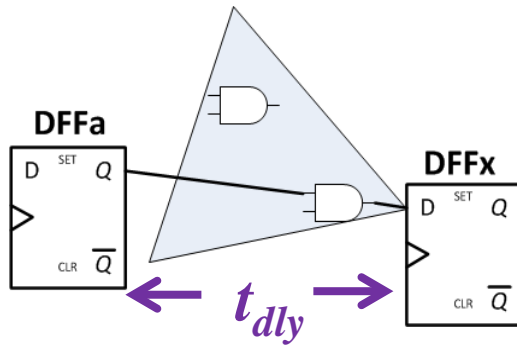
Clock at DFFx

DATA1 must be stable during
 $DFF_X t_{su}$

DATA1 must be stable during
 $DFF_X t_{HOLD}$

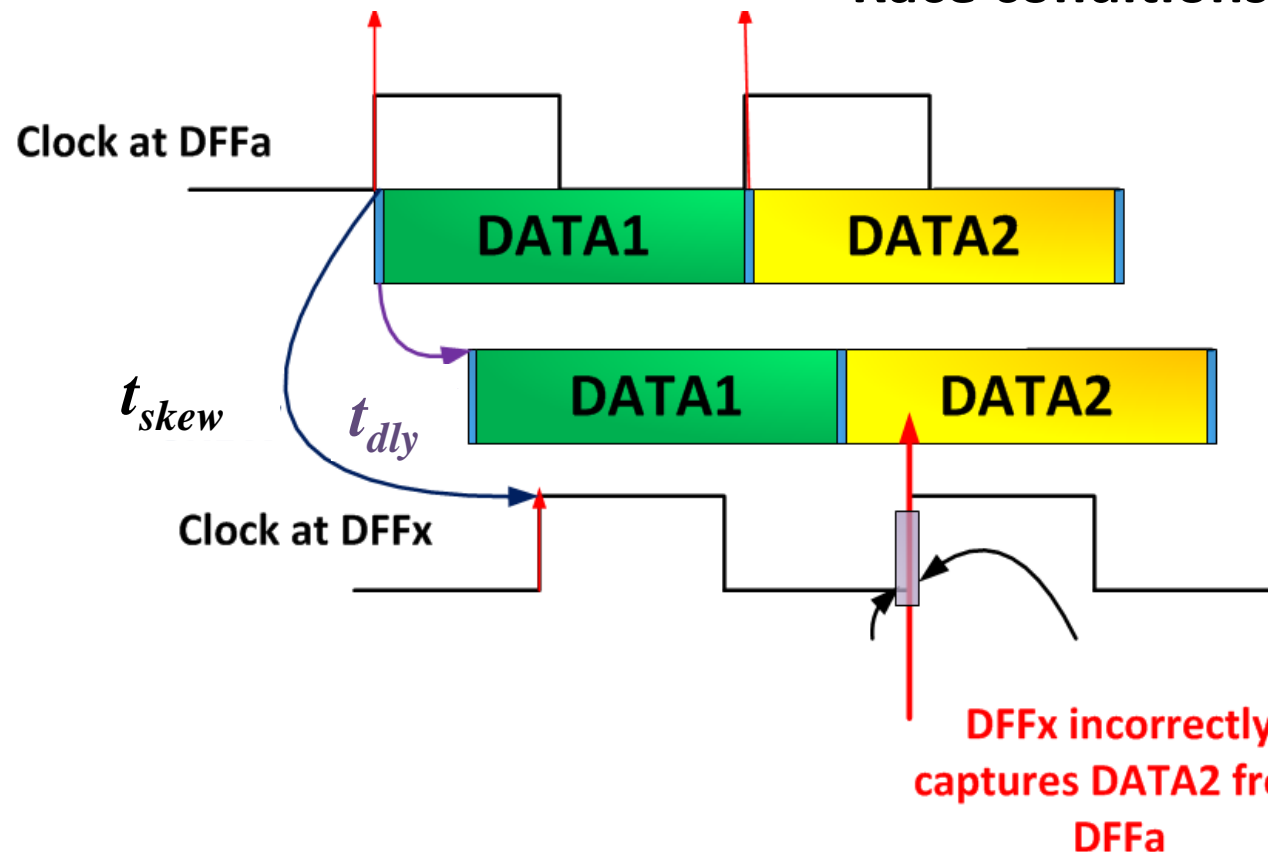
**DFFx captures DATA1
from DFFa**

Synchronous Data Capture... $T_{skew} > 0$



$$t_{skew} > t_{dly} + t_{HOLD} + t_{margin} - t_{skew}$$

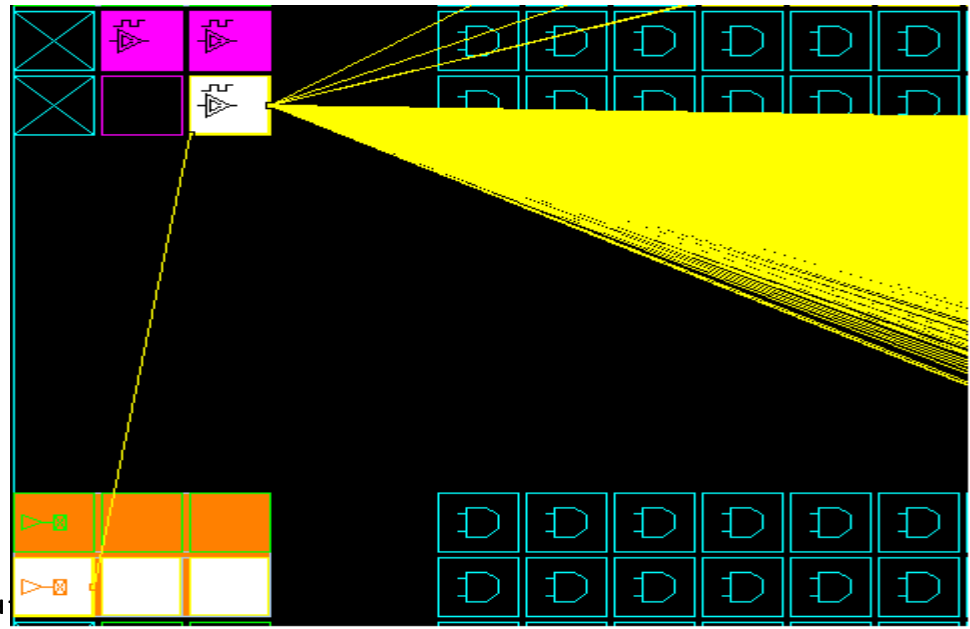
Min delay equation is violated.
Race conditions will occur.



Solution to Help Control Clock Skew: Global Clock Trees



- **Balanced clock trees are available to the end-user in all modern day FPGA devices.**
- **It is the designer's responsibility to avoid corrupting tree (global route) balance.**
- **Maintaining balance adheres to the synchronous requirement of using minimally skewed clocks.**



Designer Guidelines for Clocks in Synchronous Designs... Maintain Balance



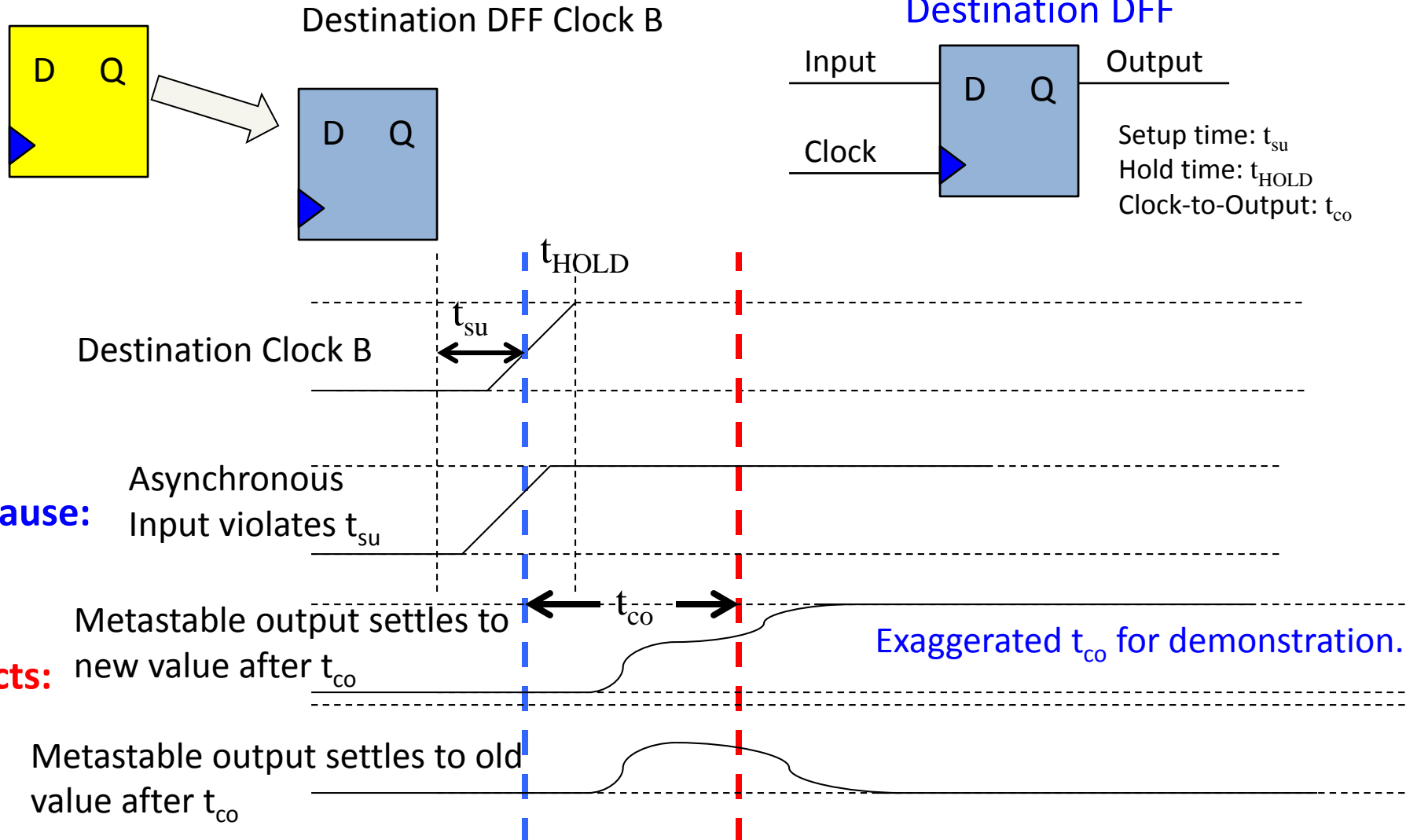
- **Avoid introducing unacceptable noise levels by forcing the clock input pin (or other clock source) is in close proximity to the clock buffer.**
 - If the pins are too far apart, the net will be too long. Long nets can cause issues with capacitance, crosstalk, and transmission line effects.
 - Designers should consult the manufacturer's data sheet.
- **If a clock tree buffer is connected to the clock pin of FFs, then it cannot connect to any other type of logic or pin.**
- **Clock gating must be done prior to the clock tree buffer and in a glitch free implementation:**
 - Clock gating is not recommended. However, if necessary, build a glitch-free circuit that switches clocks such that clocks end/start on the same edge. If implemented, the best practice is to switch clocks while circuitry is in reset.
 - A favorable alternative to clock gating is to use FF enables when possible, though it depends on the circuit and required fan-out.

Metastability

- **Cause:**
 - Introducing an asynchronous signal into a synchronous (edge triggered) system... Or
 - creating a combinatorial logic path that does not meet timing constraints.
- **Effect:**
 - Flip-flop (DFF) clock captures signal during window of vulnerability.
 - DFF output Hovers at a voltage level between high and low, causing the output transition to be delayed beyond the specified clock to output (t_{CO}) delay.
- **Probability that the DFF enters a metastable state and the time required to return to a stable state varies on the process technology and on ambient conditions.**
- **Generally the DFF quickly returns to a stable state.**
However, the resultant stable state is not deterministic.

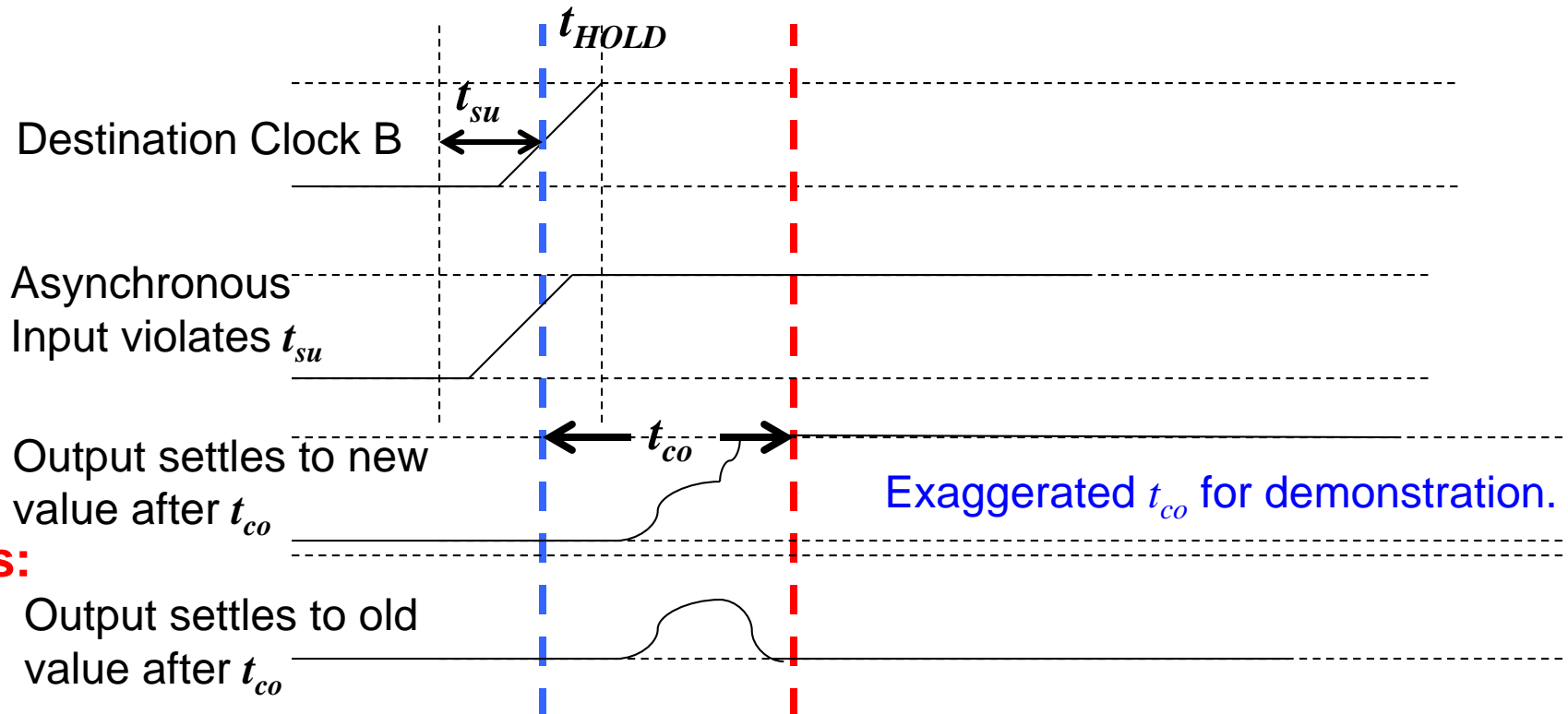
Metastability Timing Diagram (Clock Domain A to Clock Domain B)

Source DFF Clock A



No Metastability Timing Diagram (Clock Domain A to Clock Domain B)

Clarification, If a signal is unstable within the setup and hold window, the resultant may or may not go metastable. However, the resultant will be nondeterministic.



Cause:

Asynchronous
Input violates t_{su}

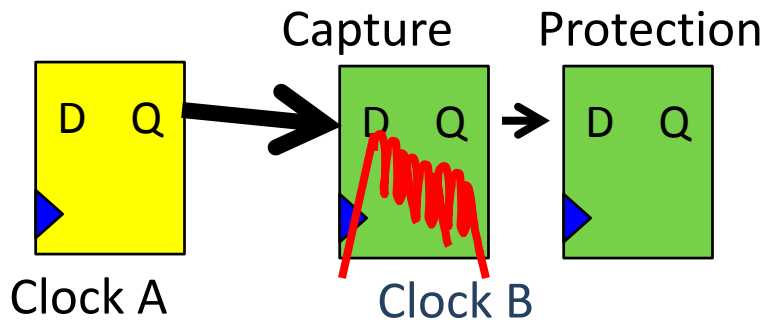
Effects:

Output settles to new
value after t_{co}

Output settles to old
value after t_{co}

Solution: Metastability Filter

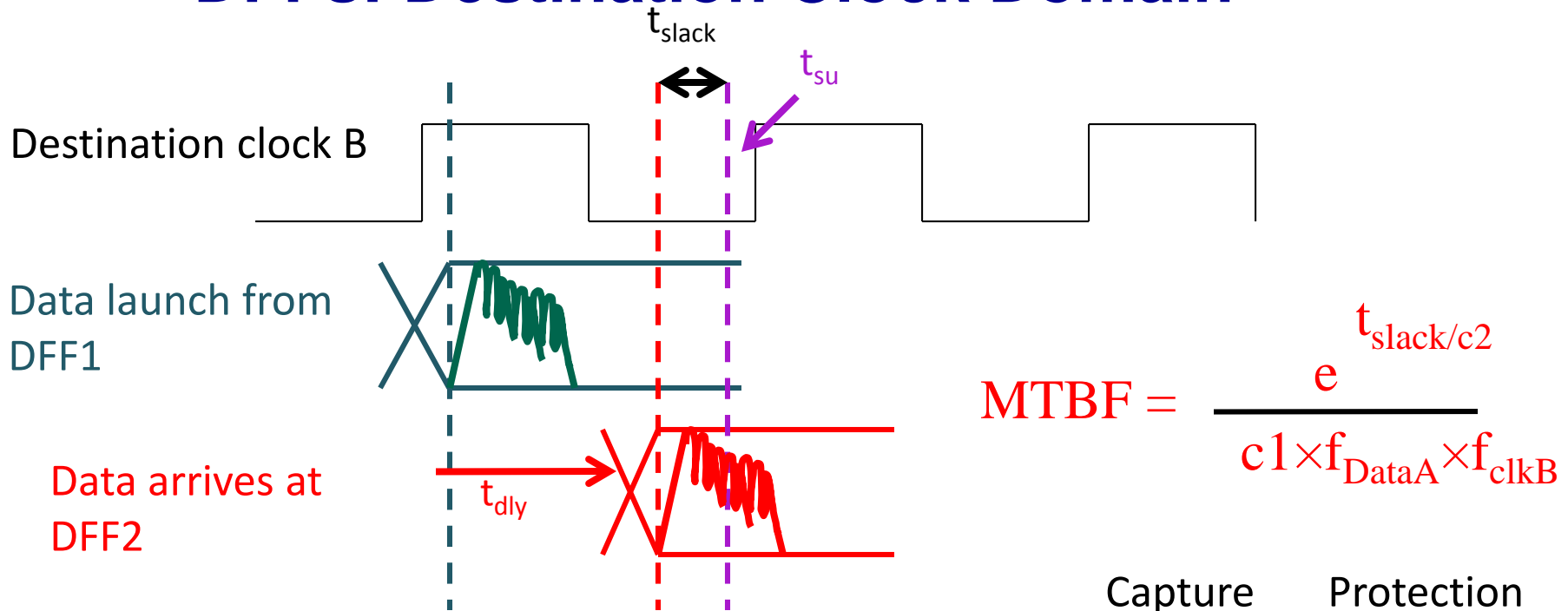
- System requires protection from metastability.
- Incoming signal is clocked in Domain A.
- Destination signals are clocked in Domain B.
- **Filter: Use a capture DFF and at least one protection DFF.**
 - Both filter DFFs are clocked in the capture domain.
 - The first DFF is expected to go metastable.
 - The second DFF is used to protect the rest of the system from potential metastable output.
- **However, there is no guarantee that the second DFF will be immune to metastability. Metastability filters have a mean time between failure (MTBF).**



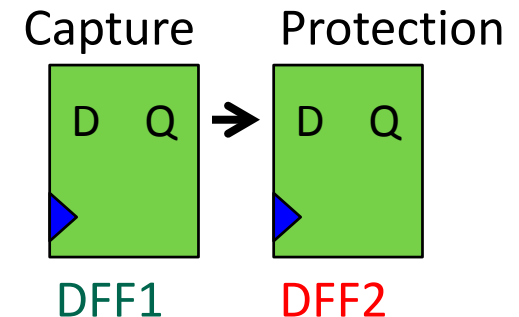
- Mean time between failure (MTBF)
- C2 and C1 are process dependent constants.
- f_{clkB} is the capture clock domain frequency.
- f_{DataA} is the maximum data switching frequency.

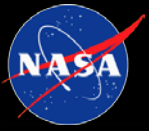
$$MTBF = \frac{e^{t_{slack}/c2}}{c1 \times f_{DataA} \times f_{clkB}}$$

Slack Time (t_{slack}) between Metastability DFFs: Destination Clock Domain



- Nets and combinatorial logic add delay.
- Delay reduces slack time.
- More slack = more time for metastability to settle.
- Metastability filter rule: **no combinatorial logic between** metastability filter DFFs; and **connection net length must be minimized.**

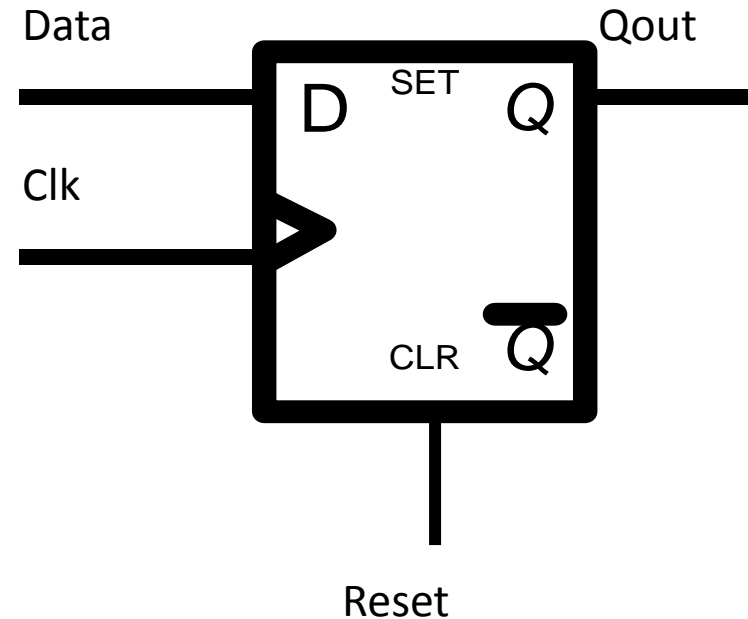




Synchronous Design Resets

Reset Circuitry

- Just like the clock – a reset will go to every DFF.
- Within a reliable synchronous design, carefully thought-out reset circuitry is crucial.
- However, very often reset circuits are over-looked and the appropriate planning does not occur.
- Improper use of asynchronous resets has led to metastable (or unpredictable) states.
- Resets must be kept in a reset-active-state for a significant amount of time.





Asynchronous Resets

- **No clock is necessary – DFFs respond to an active reset immediately.**
- **No problems exist as the system goes into reset because all DFFs will eventually enter their reset state (i.e. a deterministic state space is reachable).**
- **The predicament occurs when the system comes out of the reset state.**
- **If an asynchronous reset signal is released near a clock edge, it is possible for the flip flops to become metastable, or come out of reset relative to different clock edges.**

Example: Problem Coming Out of Asynchronous Resets

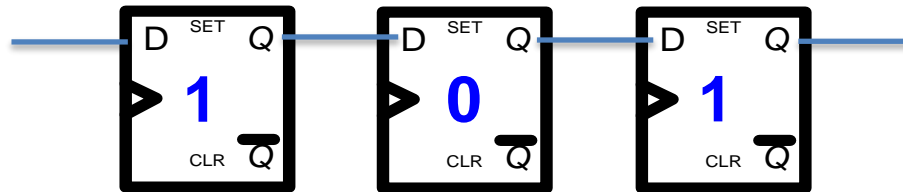


RESET

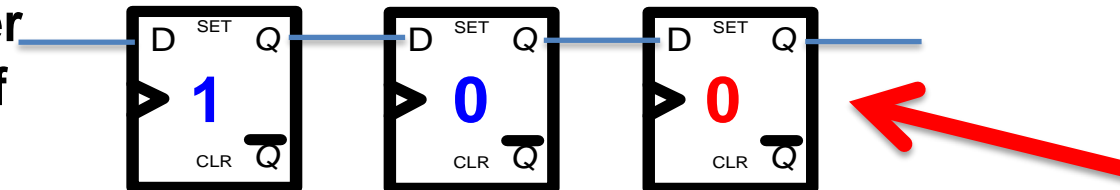


Non deterministic RESET recognition at DFF because switch is too close to clock edge.

DFFs during RESET



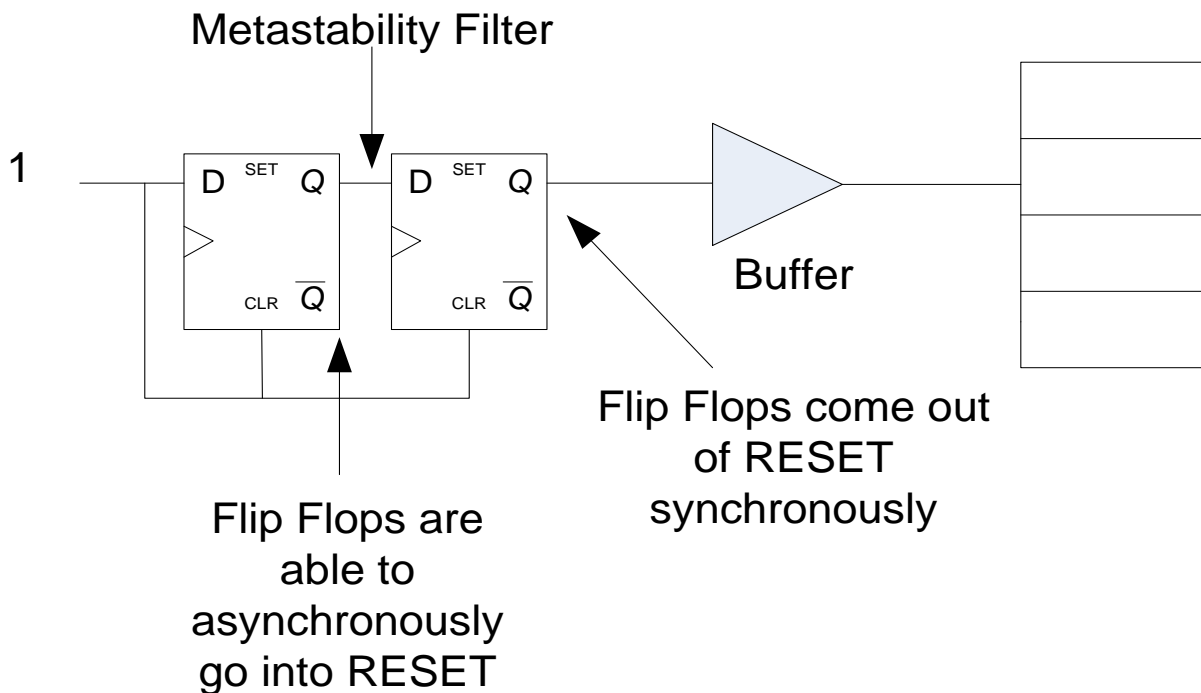
DFFs after release of RESET



DFF comes out of RESET early compared to the first two DFFs.

Asynchronous/Synchronous Resets

- **Solution: Use Asynchronous Assert Synchronous De-assert (ASSD) Reset circuit**
- **Such a design uses typical metastability filter theory. Diagram is Active Low.**

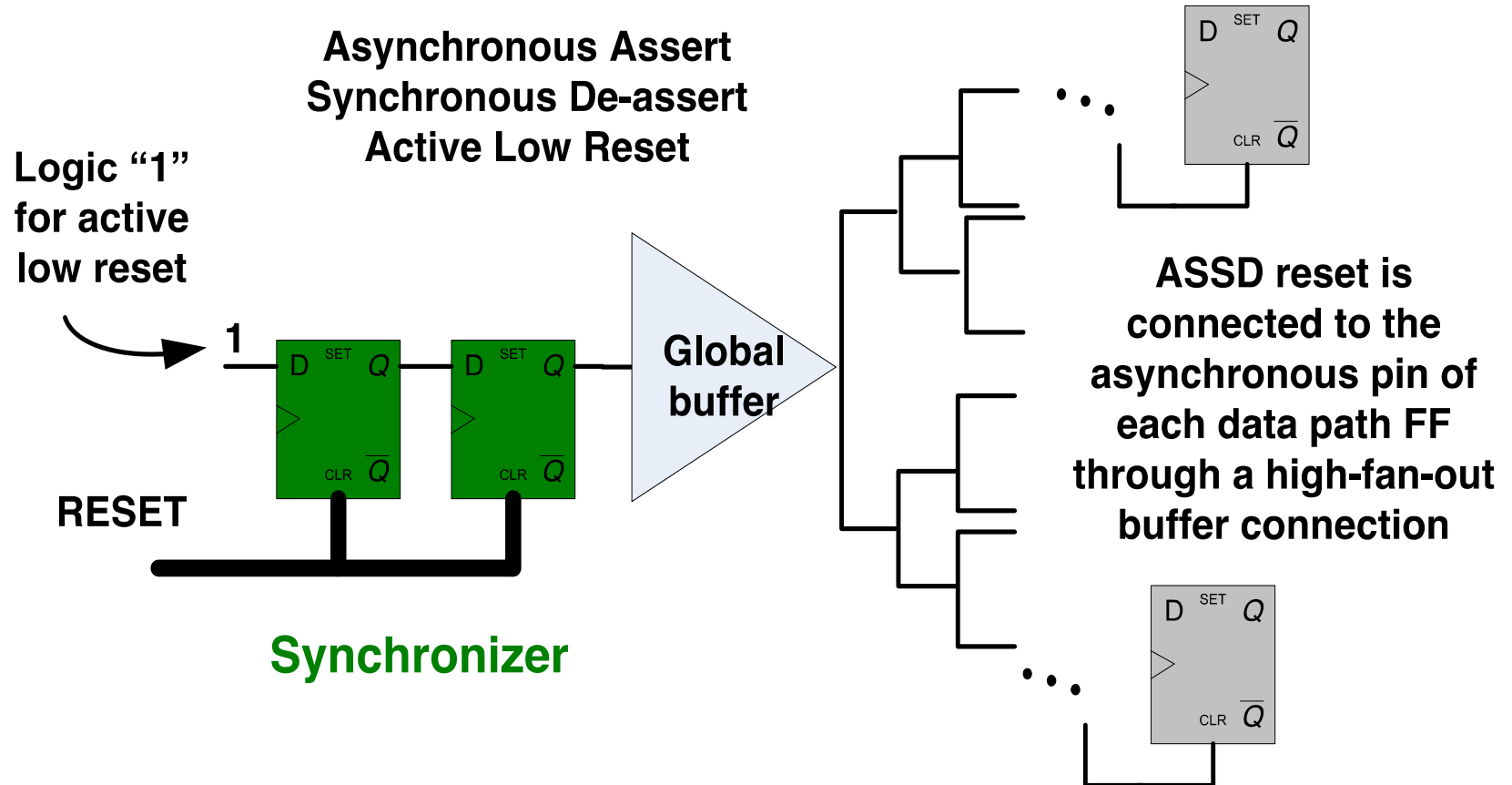


ASSD Resets



- **Upon the release of the reset signal, the first Flip Flop is not guaranteed to correctly catch the release of the reset pulse upon the nearest clock edge .**
- **At most the next clock edge.**
- **It is also probable that the first Flip Flop will go metastable.**
- **The second Flip Flop is used to isolate the rest of the circuitry from any metastable oscillations that can occur when the reset is released near a clock edge (setup/hold time violation).**

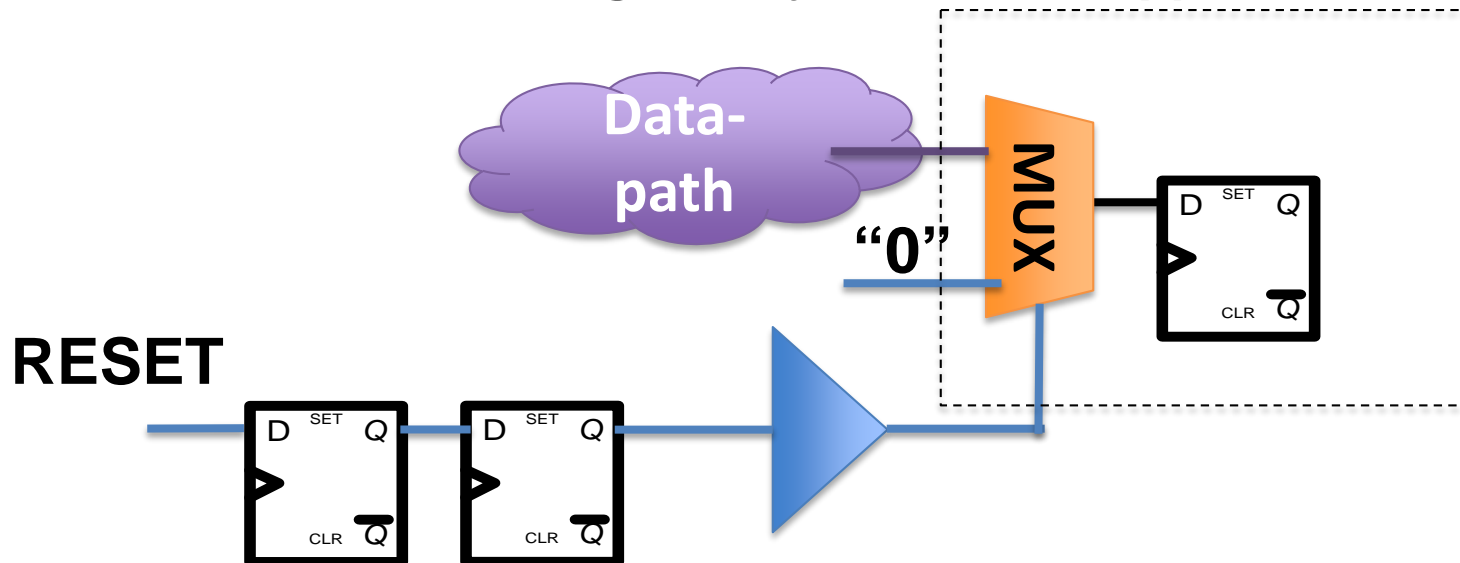
ASSD Diagram

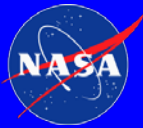


-
- The diagram illustrates a sequential circuit for a 4-bit counter. It consists of two D flip-flops, a 2-to-1 MUX, and a 4-to-1 MUX. The first flip-flop is reset by a 'RESET' signal. Its output goes to the second flip-flop and also to a 2-to-1 MUX. The 2-to-1 MUX selects between the first flip-flop's output and a '0' signal. The output of the 2-to-1 MUX goes to the second flip-flop and also to a 4-to-1 MUX. The 4-to-1 MUX selects between the output of the second flip-flop and a 'Data-path' signal. The output of the 4-to-1 MUX goes to the second flip-flop. The output of the second flip-flop is the final output of the circuit.

Synchronous Resets Disadvantages

- Adds latency to data-path because of required multiplexer (MUX).
- Can potentially damage parts on the board during power up/down because of required clock.
- It is highly recommended to implement ASSD reset circuitry for critical applications.
- However, if there are no sensitive components that the FPGA/ASIC is feeding, the synchronous approach is sufficient.





Presented Aspects of DFR (synchronous design) reflect how to create deterministic behavior in complex circuitry.

**No design is complete until it goes through a rigorous verification and validation process.
Challenge: complex designs are difficult to test.**

Design-for-verification (DFV) and Design-for-testability (DFT)

Design-for-Verification (DFV)



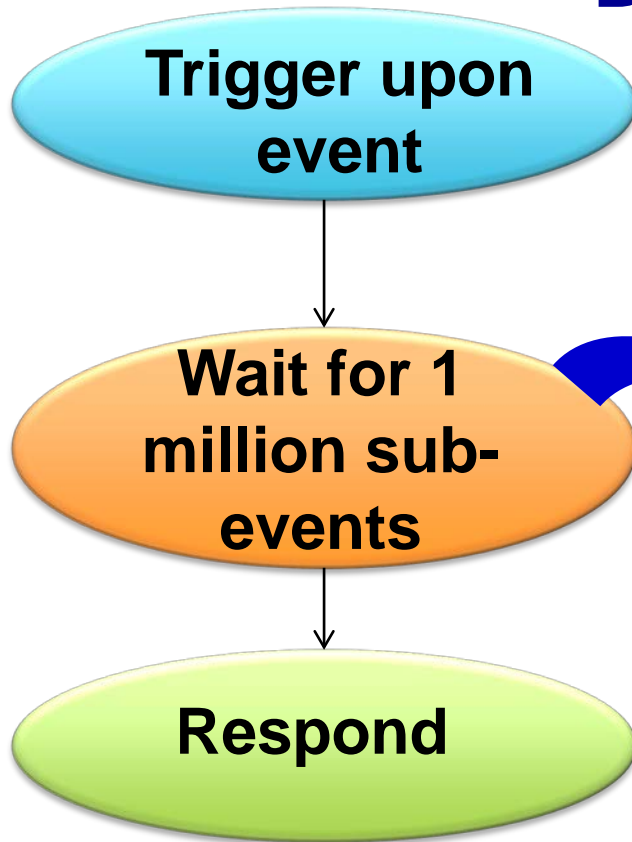


What Is DFV?

RTL: register transfer language

- **The intention of DFV is to enhance V&V coverage.**
- **DFV is limited to V&V tests during the design phase:**
 - Simulation
 - Emulation
- **Conventional DFV has three major categories:**
 - Additional logic insertion that is used to force states during testing.
 - Assertion placement in VHDL/Verilog/RTL to enhance internal visibility and real time reporting during simulation.
 - Modular design strategies:
 - Divide and conquer – design is broken into smaller more manageable pieces.
 - Plug and play – V&V testing doesn't rely on big pieces of design to be finished. Modules can be tested with models of surrounding environment (bus functional models or system level C models).

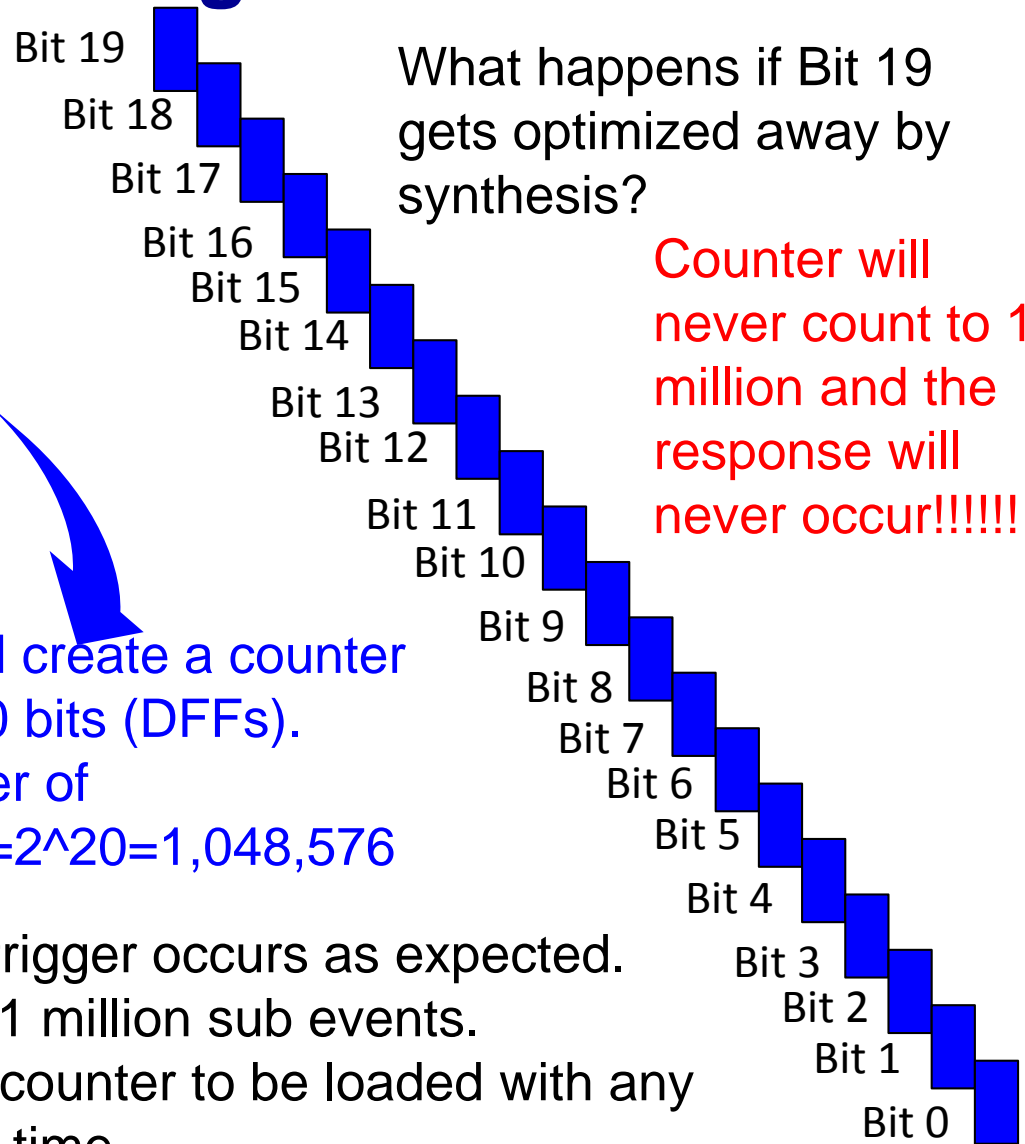
Example: DFV Used for A Common Design Bug



Should create a counter with 20 bits (DFFs).
Number of states= $2^{20}=1,048,576$

What happens if Bit 19 gets optimized away by synthesis?

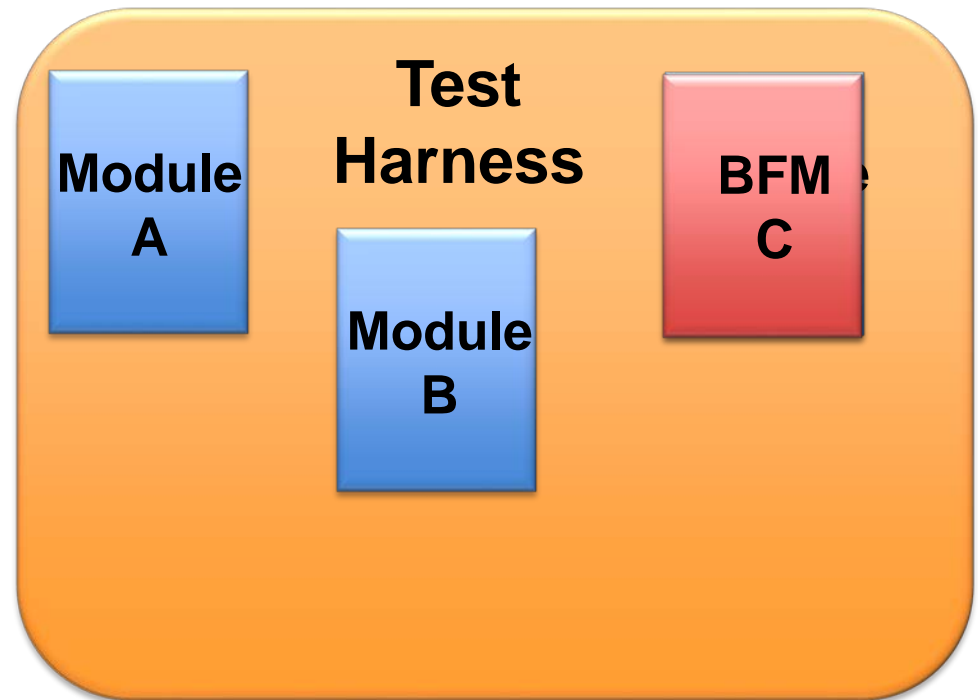
Counter will never count to 1 million and the response will never occur!!!!!!



- Verification goal: guarantee trigger occurs as expected.
- Might be difficult to simulate 1 million sub events.
- DFV: test mode enables the counter to be loaded with any number to reduce simulation time.

DFV: Modular Design Strategies

- Test harnesses are created to mimic a design; and to perform simulations.
- Eventually final versions of models are expected to be simulated in an interactive (real time) environment.
- DFV takes advantage of the modular concept.
 - Use of bus functional models (BFMs).
 - Interchange modules and their BFMs in the simulation test environment.



BFM is a high level model of a module.

Design-for-Test (DFT)



What Is DFT?

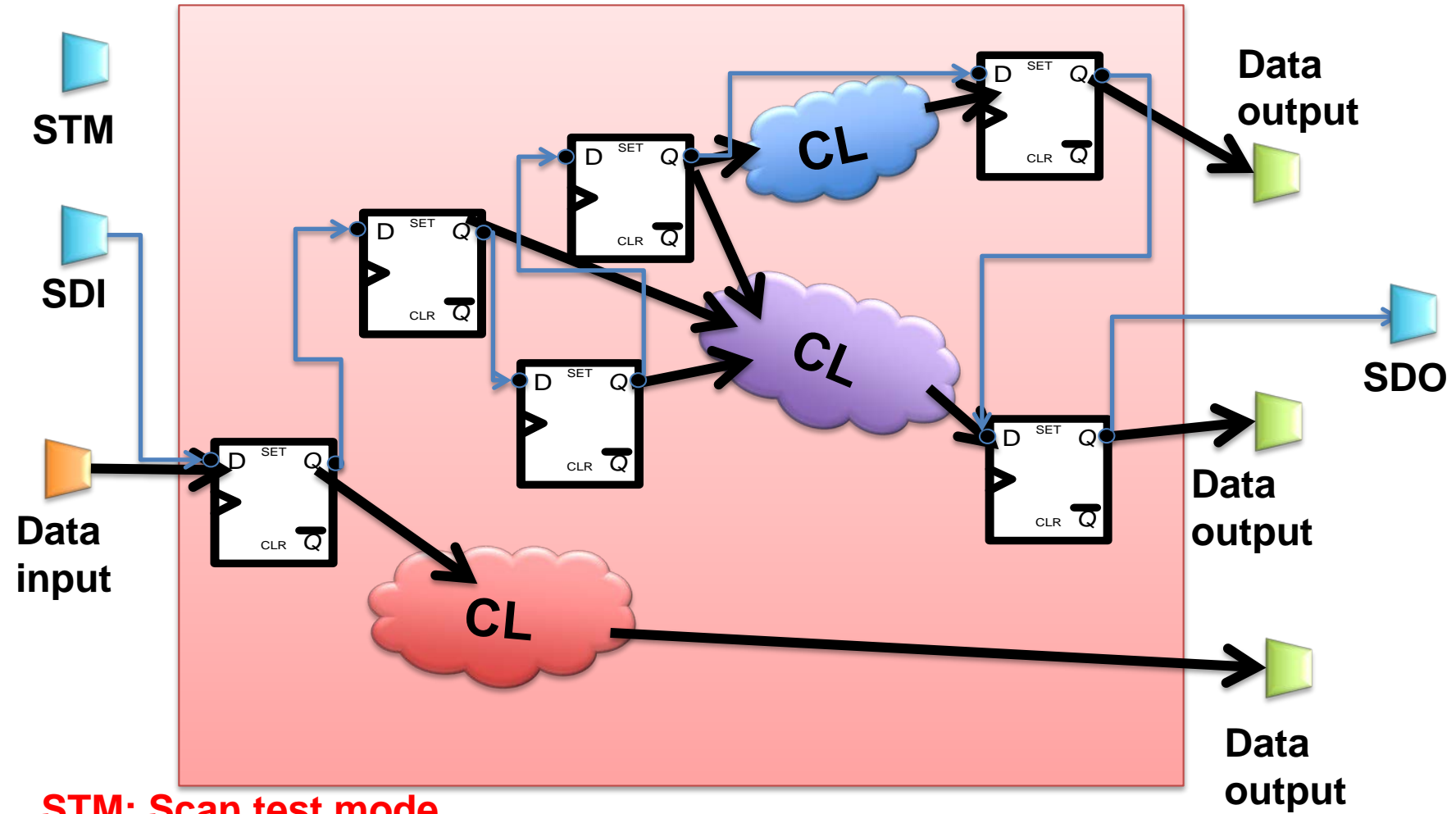
- DFT is used for post-manufactured devices.
- Generally implemented in an ASIC design and is inserted prior to place and route.
- It can be used to test manufacturing defects and can be used to perform functional testing.
- DFT is similar to DFV: controllability and observability.
- FPGA base-arrays contain DFT logic:
 - Some DFT circuits can be implemented by the end-user.
 - Some DFT circuits is hidden logic and is disabled prior to end-user base-array acquisition.
- Conventional DFT methodology:
- Insert logic to change between normal operational mode and test mode. Requires a test mode pin and a mux added to the DFFs.



DFT Process

- **Place into test mode:**
 - Test mode pin is enabled.
 - Connections are changed such that DFFs are placed into a shift register.
 - System is clocked. Test data are serially shifted into the test shift register (controllability).
- **Place into normal operation mode:**
 - Test mode pin is disabled.
 - Connections are changed such that DFFs are placed into normal operation mode.
 - System is clocked.
- **Place into test mode:**
 - Test mode pin is enabled.
 - Connections are changed such that DFFs are placed into a shift register.
 - System is clocked. Test data are serially shifted out of the test shift register (observability).

DFT Connectivity: Normal Operation to Test Mode



STM: Scan test mode

SDI: Scan data in

SDO: Scan data output

Design-for-Security (DFS)



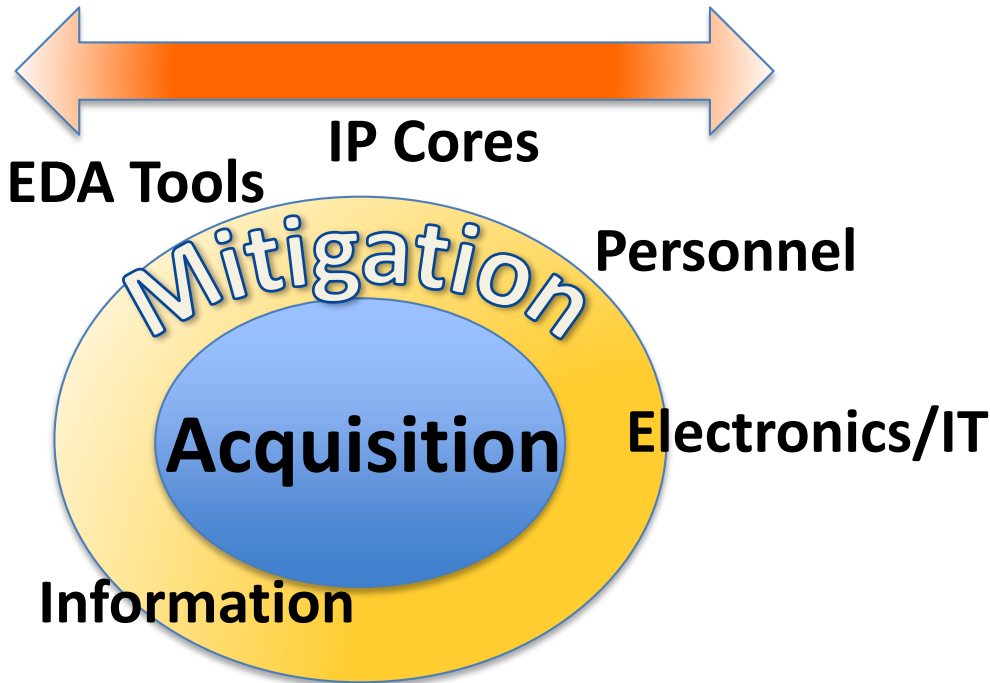


What is DFS?

- **Hardware DFS pertains to design strategies that reduce the risk of adversary infiltration throughout the full design ecosystem.**
- **The major concerns for risk and countermeasure application pertain to the potential for adversaries to:**
 - Steal intellectual property:
 - Counterfeiting
 - Obtaining knowledge of system
 - Add or delete Malicious circuit (trojan)
 - Perform side channel attacks:
 - Stealing hardware key information
 - Listening for specific operation

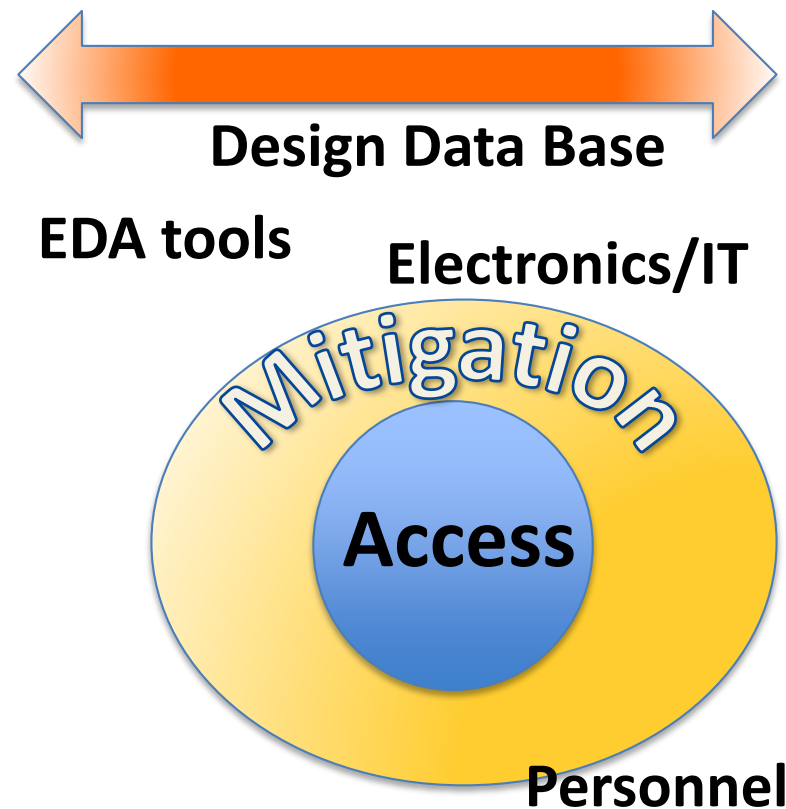
Primary Design Cycle Vulnerabilities

Design Cycle Preparation



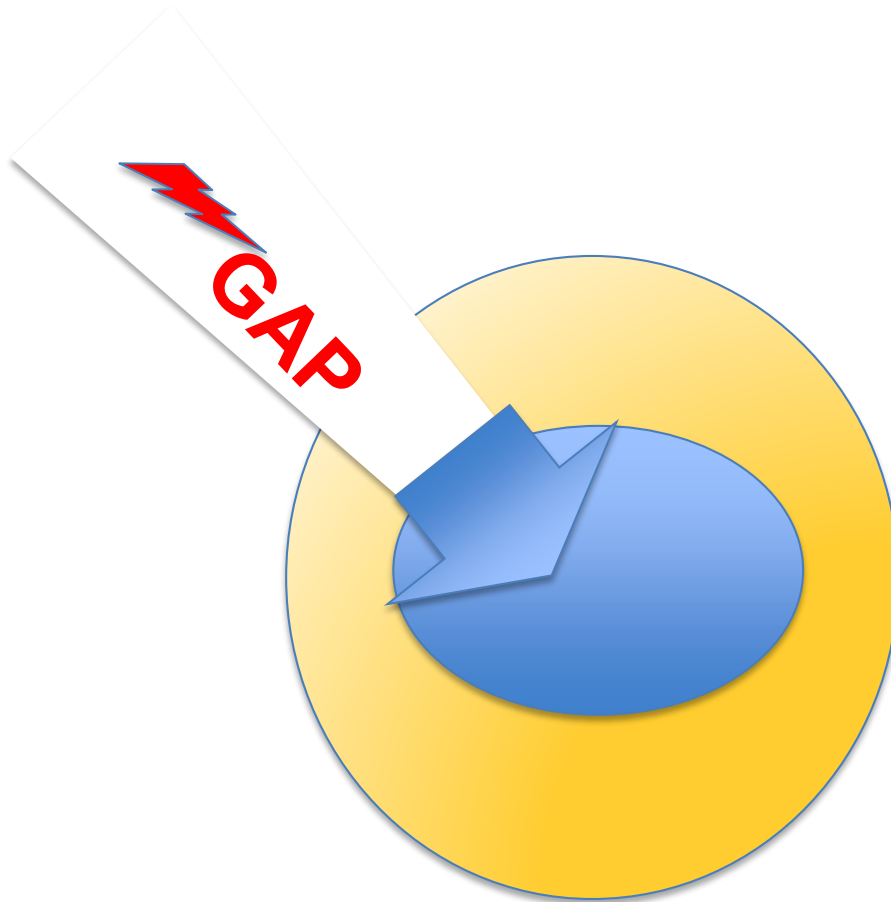
Mostly External threats except for Personnel making acquisitions.

Design Cycle and Deployment



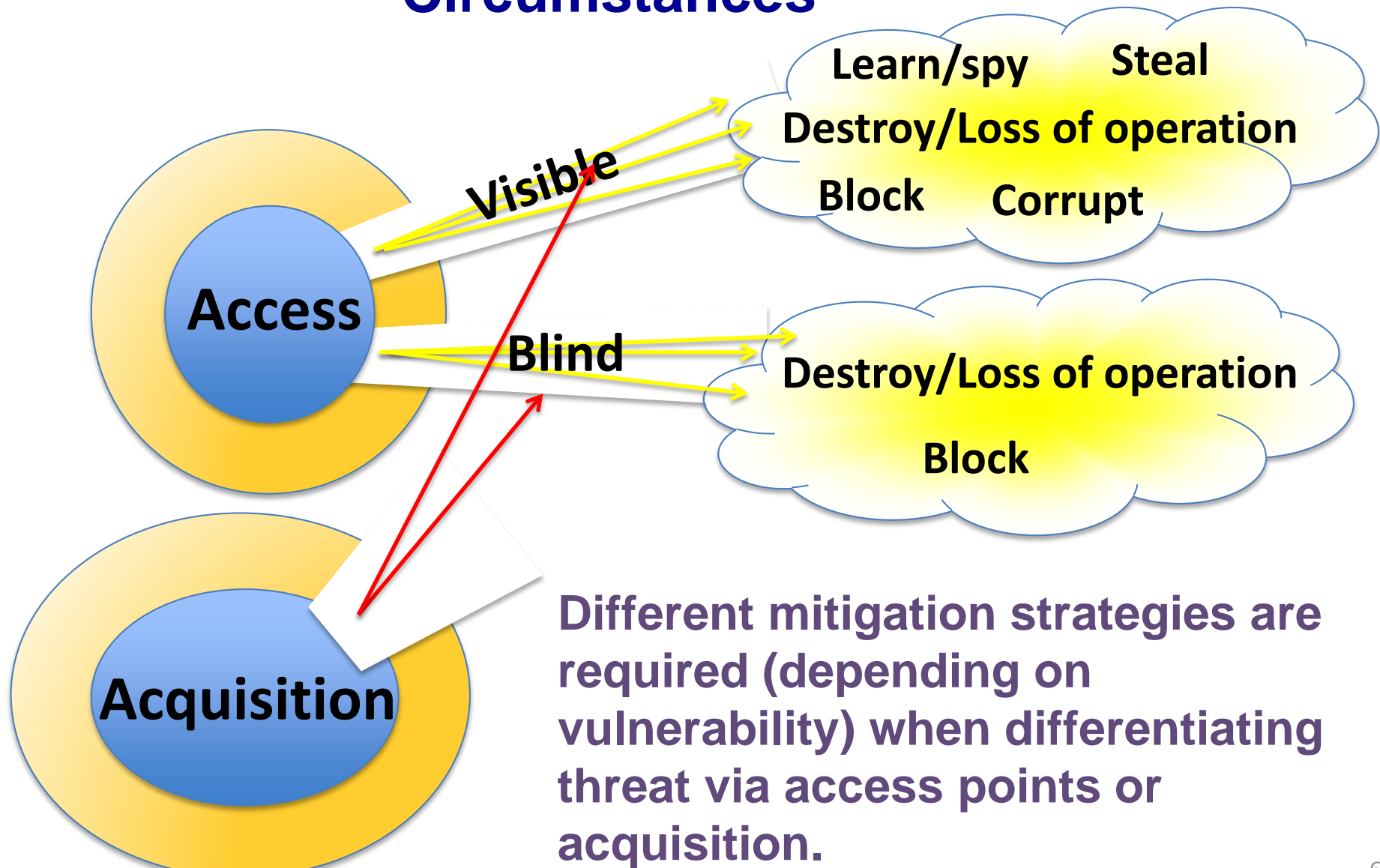
External or internal threats.

Learned Accessibility ... Actor Finds Gaps in Mitigation



- **Adversary learns the system under analysis including mitigation.**
- **Adversary tries to detect or create gaps in mitigation.**
- **Adversary attacks system via gap.**
- **Must be taken into account in risk analysis.**
- **Will additional layers or dynamic layers of mitigation reduce risk?**
- **This action can be modeled in traditional game theory.**

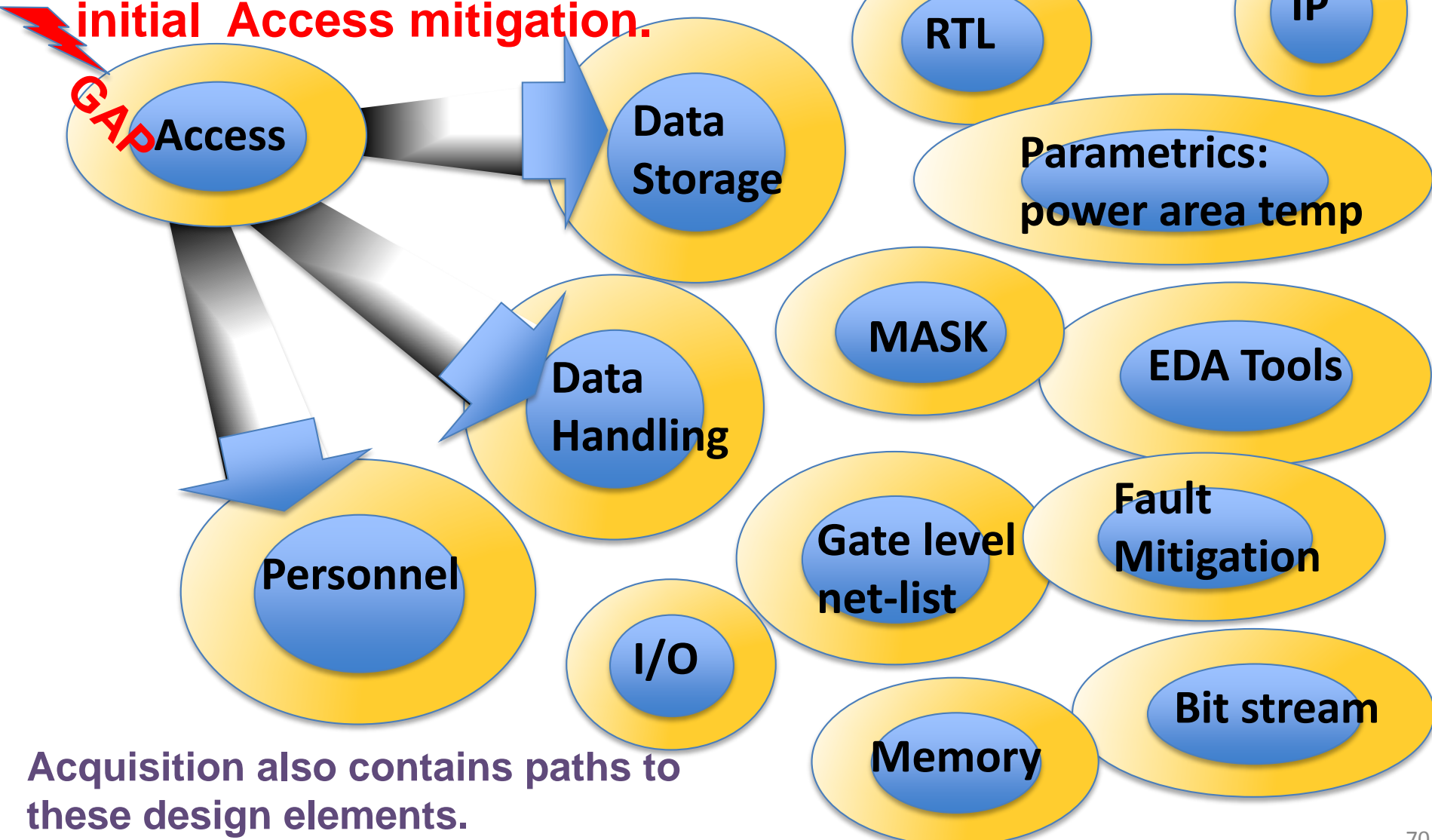
Gaps in Mitigation: Channels of Vulnerability and Circumstances



Accessibility into Internal Design Elements: Multiple Layers of Mitigation



**Actor has broken through
initial Access mitigation.**



**Acquisition also contains paths to
these design elements.**

Determining System Risk

- Each step within the design flow can be depicted using **acquire/mitigate** or **access/mitigate** game theory models.
- In order to assess system vulnerability, the design must be evaluated:
 - Information (at each step of the design flow) is gathered regarding design implementation.
 - Design implementation is evaluated according to mission requirements, threat, and best practices.
 - Risk is determined from gathered information and assessments.
Search for gaps in mitigation.

Cannot perform risk analysis without proper gathering of design information

Note Mitigation Application and Strength Must Be Carefully Assessed

- Risk assessments are complex, but they are a necessity.
- Piling on mitigation can add risk.
- Mitigation complexity might have hidden modes that are blind to the review team or unreachable by the EDA tools:
 - System lock out,
 - Unwarranted self-destruct,
 - Flags that ease adversary's learning phase.



**Mitigation eats
access to all!**

When Mitigation becomes a threat!



DFS and DFR

- **One aspect of trust and security is to assure that operations are at all times as expected... nothing more... nothing less.**
- **System complexity has increased such that the required assurance process is infeasible.**
- **Lack of V&V coverage increases the risk of being unable to identify malicious circuitry insertion.**
- **However, there are techniques that can enhance assurance and hence reduce risk.**
 - DFR is the process of creating deterministic designs.
 - The deterministic operation is a product of the discrete nature of synchronous design.
 - Accordingly, following strict DFR rules enhances system V&V.

DFS versus DFV and DFT

- **The insertion of test modes requires external control and provides external visibility.**
- **This has been termed backdoor accessibility.**
- **As a result an adversary can gain access to the system and do the following:**
 - Change or disrupt the operational state.
 - Run test vectors to gain knowledge of the device.
- **FPGA base-arrays provide backdoor access. In order to avoid adversary infiltration:**
 - All test-pins (backdoor inputs and outputs) should be either tied down on the board or strongly controlled by reliable circuitry.
 - If pins are tied down, the end-user loses access to device internal visibility and control.
 - If pins are not tied down and are accessible by other circuitry:
 - Protection keys should be used to obtain accessibility.
 - Keys should be dynamic in nature.
 - Data encryption should be applied (also is a side channel attack countermeasure).
 - Protocols of accessibility should be established.



Summary

- **The United States government has identified that ASIC/FPGA hardware circuits are at risk from a variety of adversary attacks.**
- **As an affect, system security and trust can be compromised.**
- **The tutorial covered how design practices can affect the risk for the adversary to:**
 - Change circuitry
 - Steal intellectual property
 - Listen to data operations
- **A description of design practices and how they affect risk was presented: design-for-reliability (DFR), design-for-verification (DFV), design-for-test (DFT), and design-for-security (DFS).**
- **Information pertaining to common countermeasures and risk analysis was provided.**