CSE/CEN 598
Hardware Security & Trust

Secure Hardware Primitives:
Physical Unclonable Functions

Prof. Michel A. Kinsy

Physical Unclonable Functions (PUF)

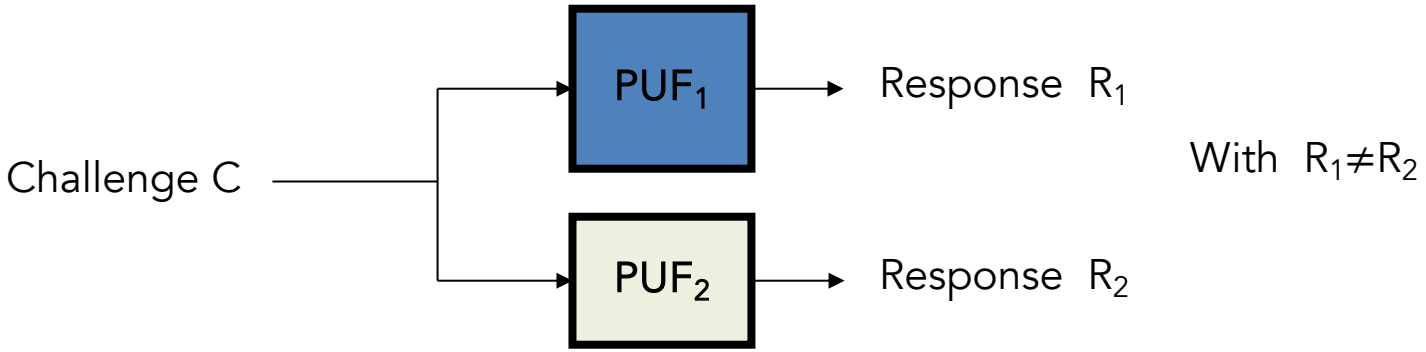
- Physical Unclonable Functions (PUFs) have been introduced as the hardware equivalent of a one-way function
  - Due to random process variations, no two Integrated Circuits even with the same layouts are identical
  - Variation is inherent in fabrication process
    - Even circuits produced by the same design and technology will have slight difference/variations
  - Hard to remove or predict
    - Unpredictable
      - To users
      - To manufacturers (even the manufacturer cannot produce two identical PUFs)
    - Unclonable
      - For the most part
- A PUF can be used as an unclonable key

Physical Unclonable Functions (PUF)

- Applications:
  - Secret Key Generation / Storage
  - Random Number Generator
  - Identification
  - Authentication
  - Hardware Obfuscation
  - Key exchange
  - …

Physical Unclonable Functions (PUF)

- Computable
  - Given PUF and x, it is easy to evaluate y = PUF(x)
- Unique
  - PUF(x) contains some information about the identity of the physical entity embedding the PUF



Physical Unclonable Functions (PUF)

- Computable
  - Given PUF and x, it is easy to evaluate y = PUF(x)
- Unique
  - PUF(x) contains some information about the identity of the physical entity embedding the PUF
- Reproducible
  - y ≈ PUF(x) is reproducible - up to a small error

Physical Unclonable Functions (PUF)

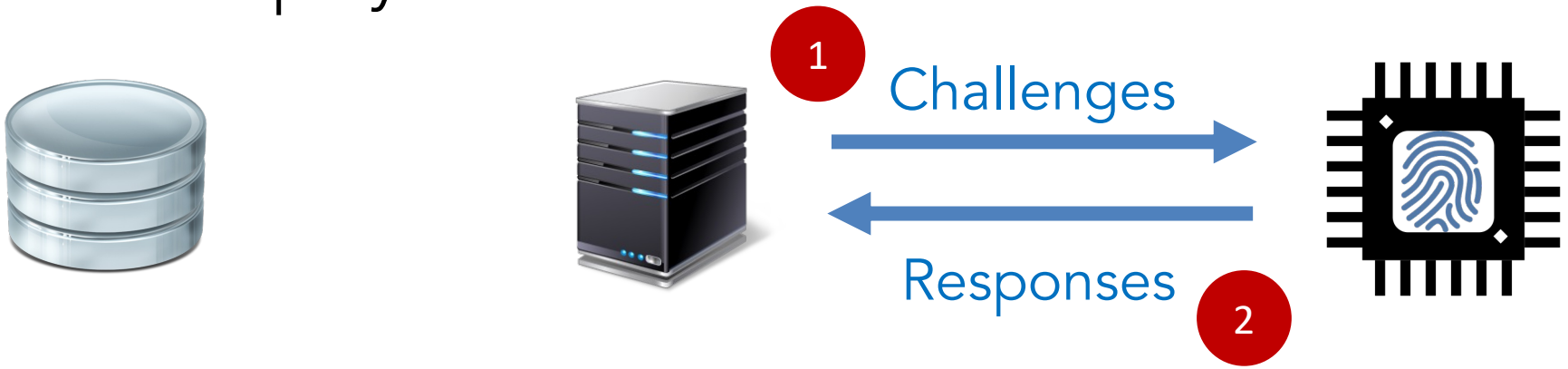
- Unclonable
  - Given PUF, it is hard to construct a procedure *PUF' where* PUF(x) ≈ PUF'(x)
- Unpredictable
  - Given a set of CRPs, it is hard to predict $y \approx PUF(x)$
  - Meaning learning is hard
- One-way
  - Given only y and the corresponding PUF, it is hard to find x such that $y \approx PUF(x)$

Challenge-Response Pairs (CRPs)

- Before PUF deployment

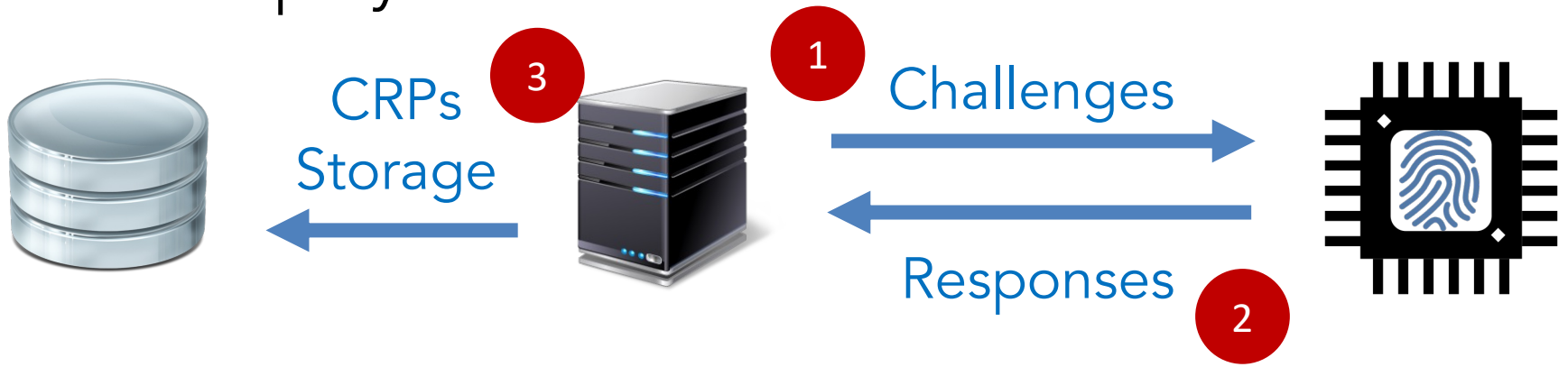
Challenge-Response Pairs (CRPs)

- Before PUF deployment



Challenge-Response Pairs (CRPs)

- Before PUF deployment



Challenge-Response Pairs (CRPs)

- Before PUF deployment

- At authentication

Challenge-Response Pairs (CRPs)

- Before PUF deployment

- At authentication

Challenge-Response Pairs (CRPs)

- Before PUF deployment

- At authentication

PUF Challenges and Limitations

- CRPs used in authentication must be stored for validation
- Where do you store them?
  - Must keep them secure for the lifetime of the device

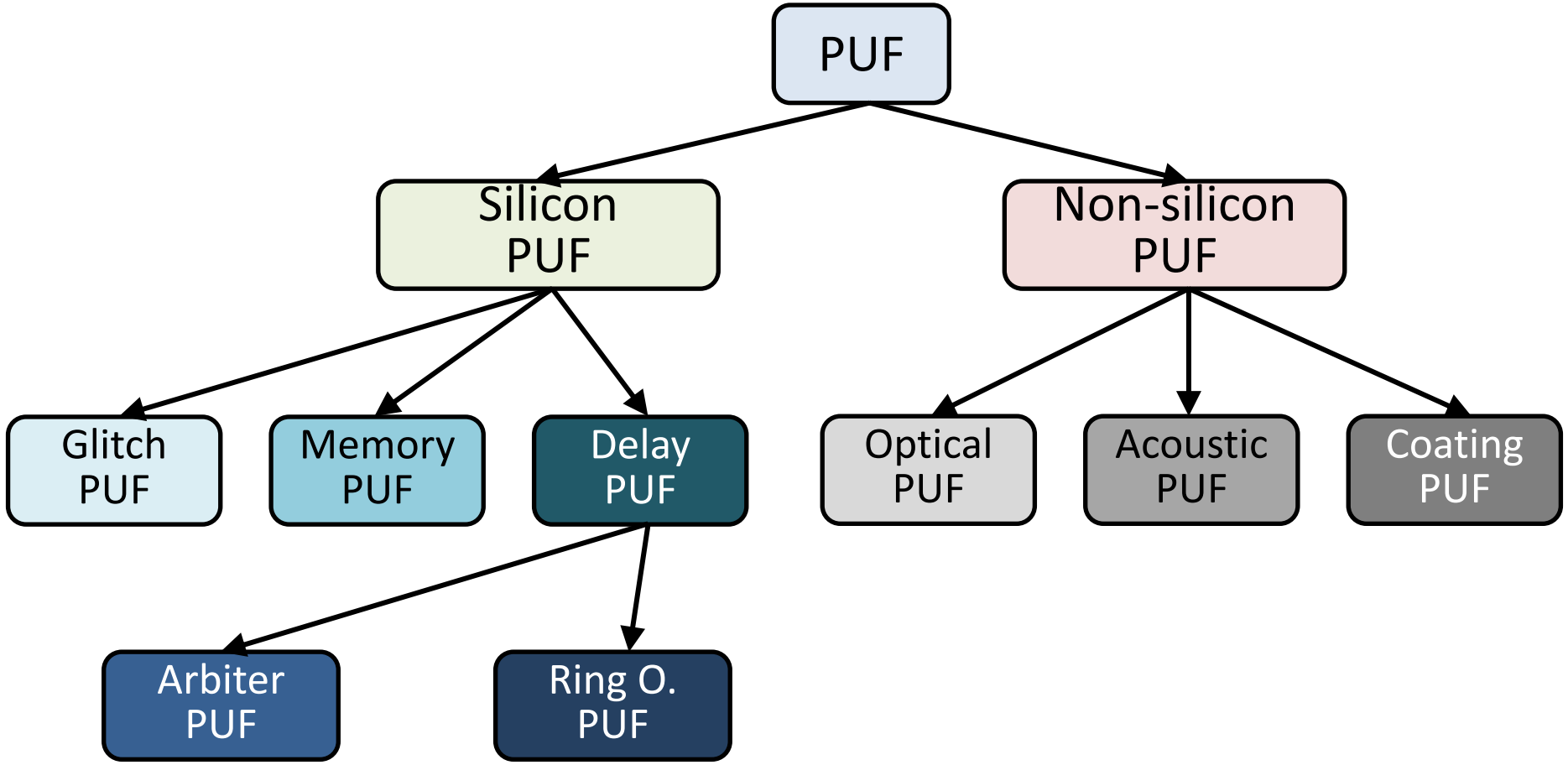
PUF Challenges and Limitations

- CRPs used in authentication must be stored for validation
- Where do you store them?
  - Must keep them secure for the lifetime of the device
- What if they are stolen?
  - Can someone impersonate your device now?

PUF Challenges and Limitations

- CRPs used in authentication must be stored for validation
- Where do you store them?
  - Must keep them secure for the lifetime of the device
- What if they are stolen?
  - Can someone impersonate your device now?
- Who generates the CRPs?
  - Just the end user?
  - What if the manufacturer reads the CRPs?
    - Are they trusted?

Physical Unclonable Functions (PUF)



- There are more types of PUF implementations

Source of Randomness

- PUFs Using Explicitly-introduced Randomness
- *Easier to control PUF uniqueness*
  - Optical PUF
  - Coating PUF
- PUFs Using Intrinsic Randomness
- *More popular, no modification to the original design*
  - Delay PUF – ring oscillator, arbiter PUFs etc.
  - Memory PUF – SRAM, DRAM, FF PUFs etc.
  - Mixed signal PUF – analog PUFs
  - Other types – Bi-stable Ring, magnetic stripe card, quantum confinement PUF etc.

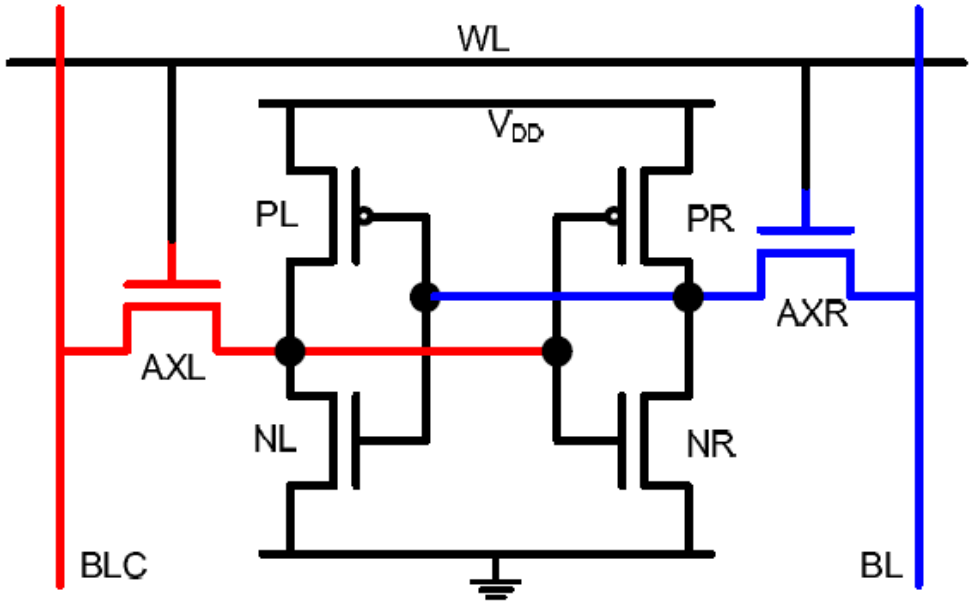
Weak vs Strong PUFs

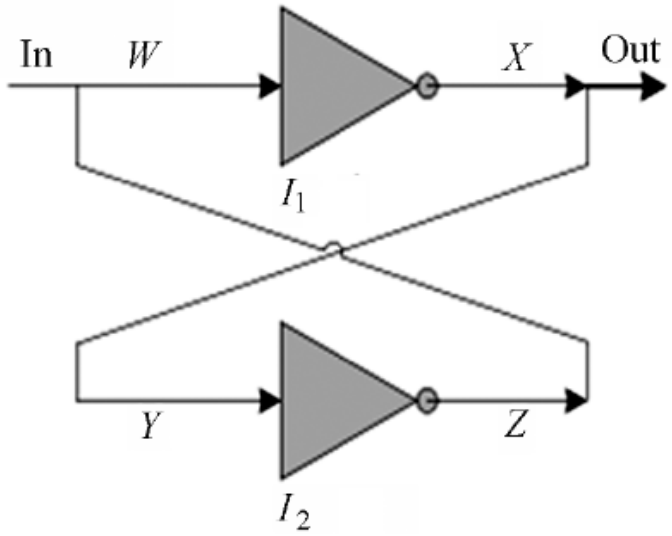
- Based size of Challenge-Response Pairs
- Weak PUFs
  - Small size of CRP set (usually 1)
  - Mostly used for key storage
  - The CRP access must be restricted from attackers
- Strong PUFs
  - Large size of CRP set
  - Mostly used for authentication
  - A portion of CRP set can be public
  - Impossible to predict the unknown CRPs

Popular PUF Designs

Weak Memory PUF - SRAM PUF

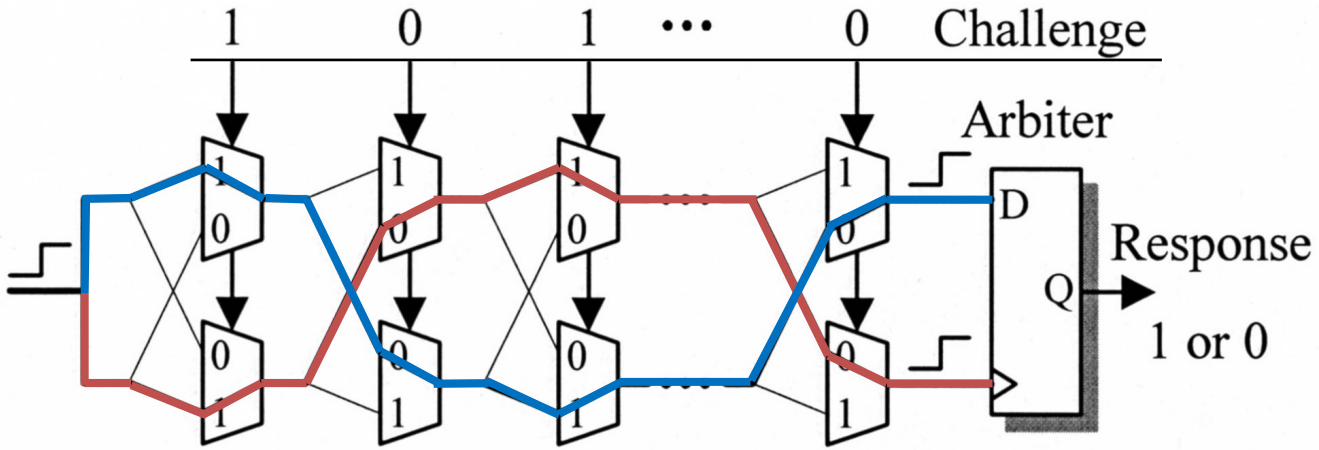
- Memory cell (a cross-coupled inverter) based
- Uses intrinsic randomness in each cell's initial state at power up
- Easy to implement, but not applicable to all FPGAs
  - Some modern FPGAs assign fixed value to the cells' initial state





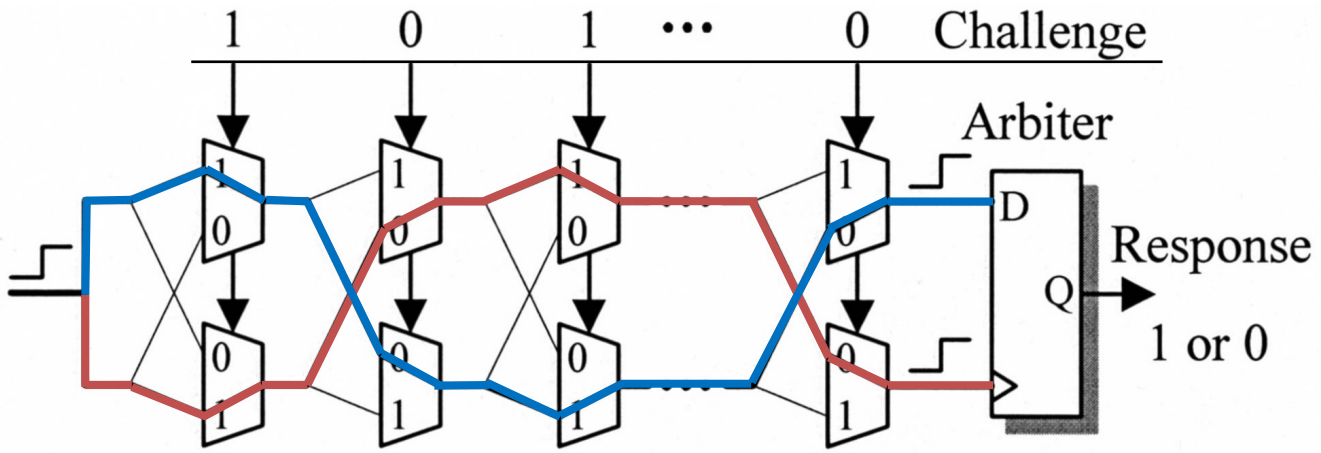
Popular PUF Designs

- Arbiter PUF– A strong delay PUF
  - MUX based
  - Using the intrinsic delay differences in each MUX
  - Stronger PUF
    - n challenges produce 2^n possible routes (responses)
  - Hard to implement on FPGAs (explained later)



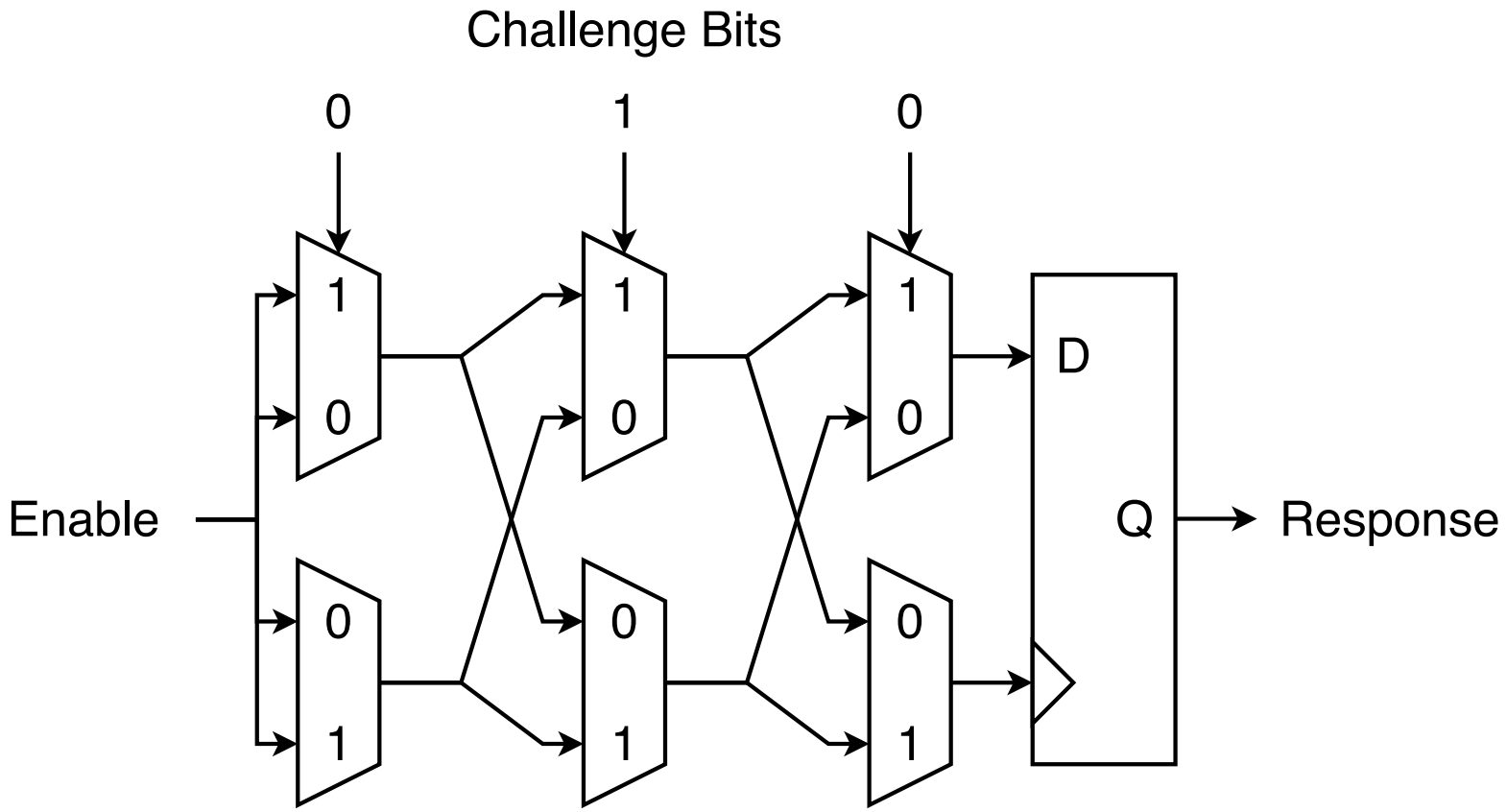
Strong Delay PUF – Arbiter PUF

- MUX based
- Using the intrinsic delay differences in each MUX
- Stronger PUF
  - N challenges produce 2^N possible routes (responses)
- Hard to implement on FPGAs (explained later)



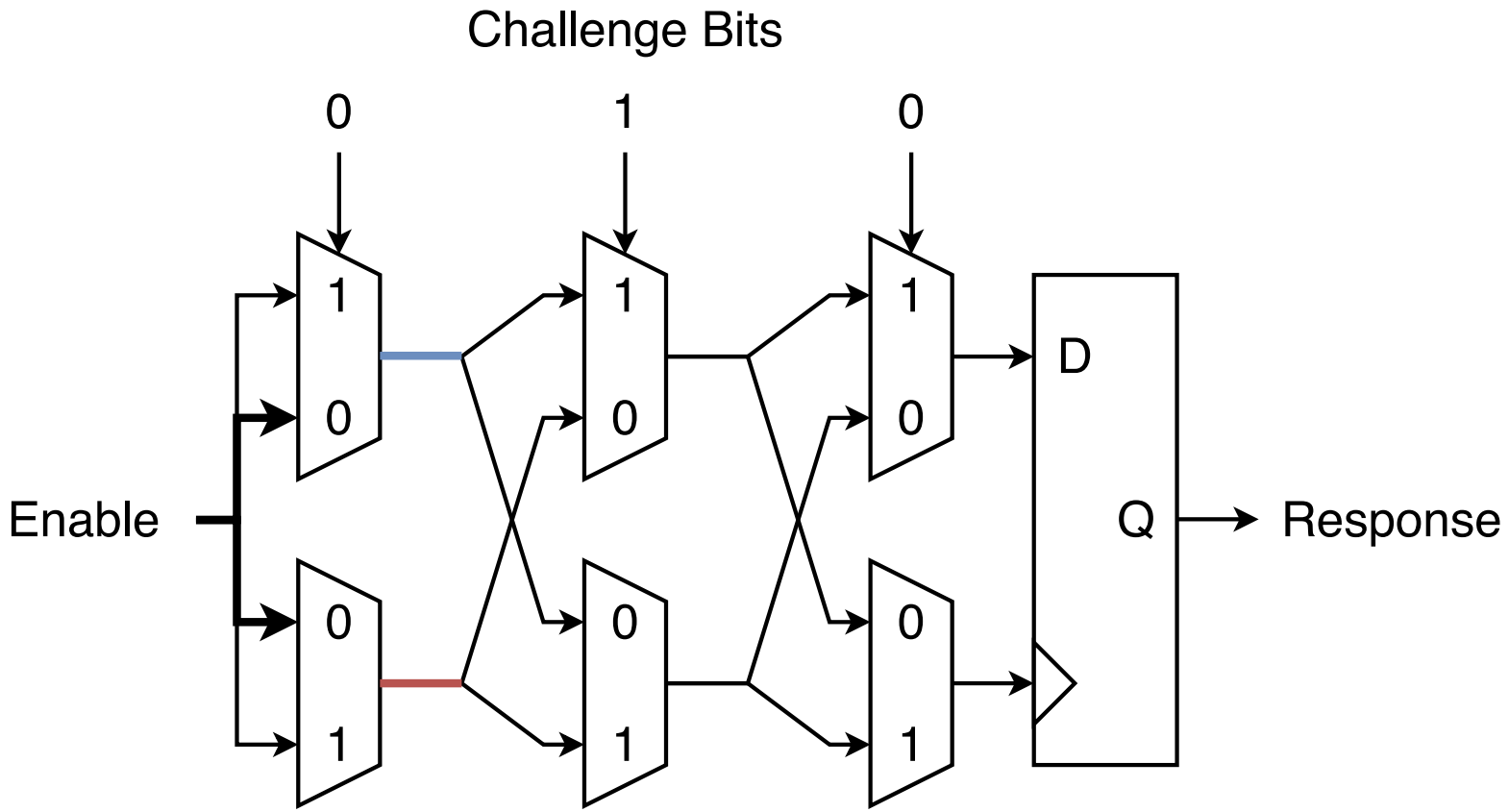
Arbiter PUF Example

- Challenge bits are set



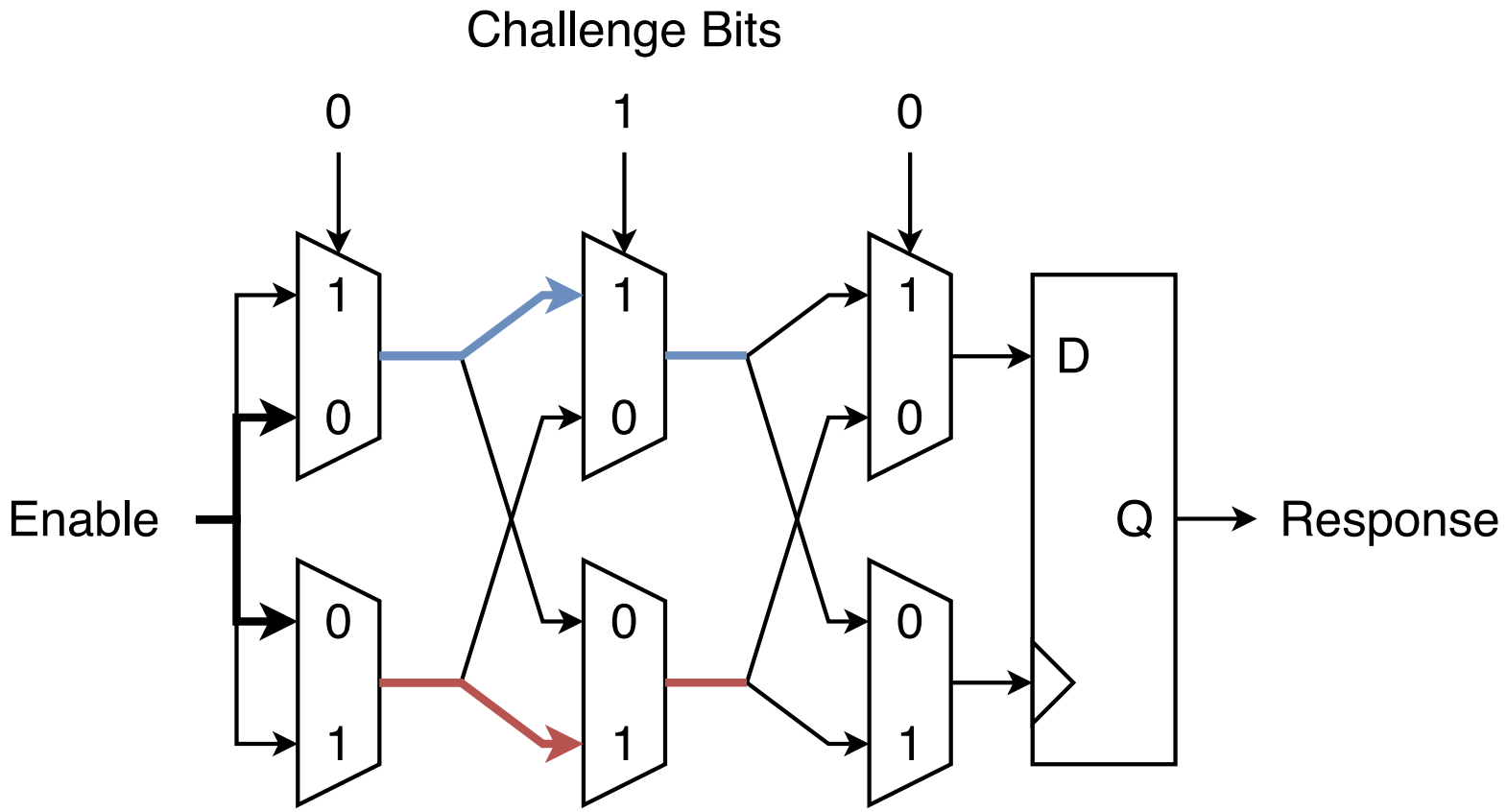
Arbiter PUF Example

- Enable signal is raised, race condition starts
- Signals propagate through first multiplexor and towards second



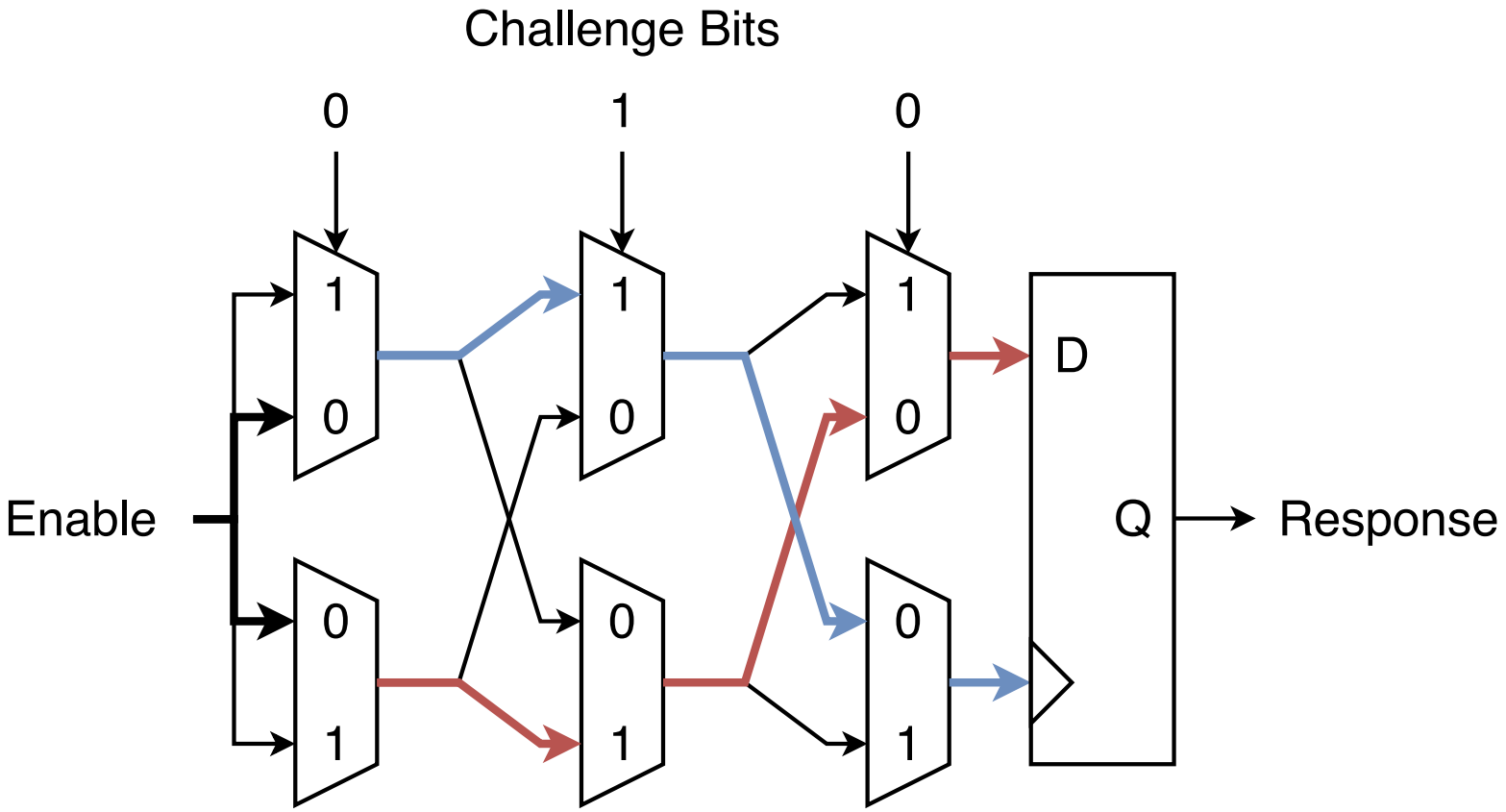
Arbiter PUF Example

- Signals pass through second multi-plexor towards third



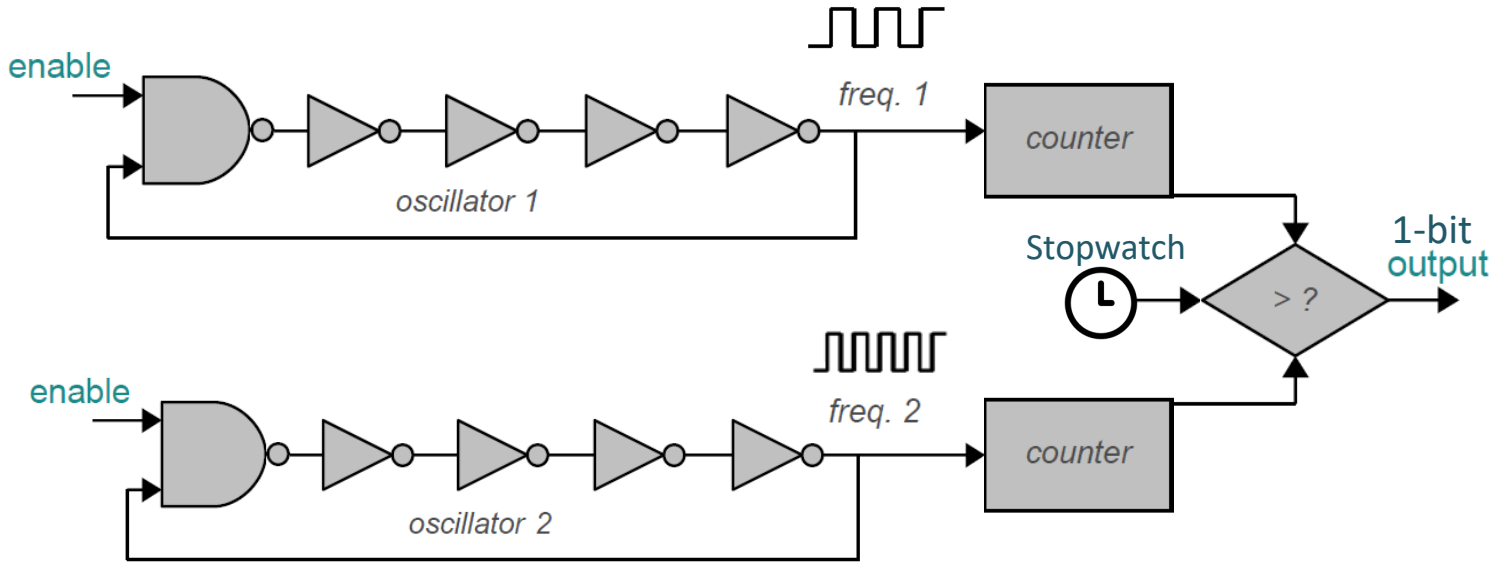
Arbiter PUF Example

- Signals pass through third multi-plexor towards register
- Only one signal can win the race condition



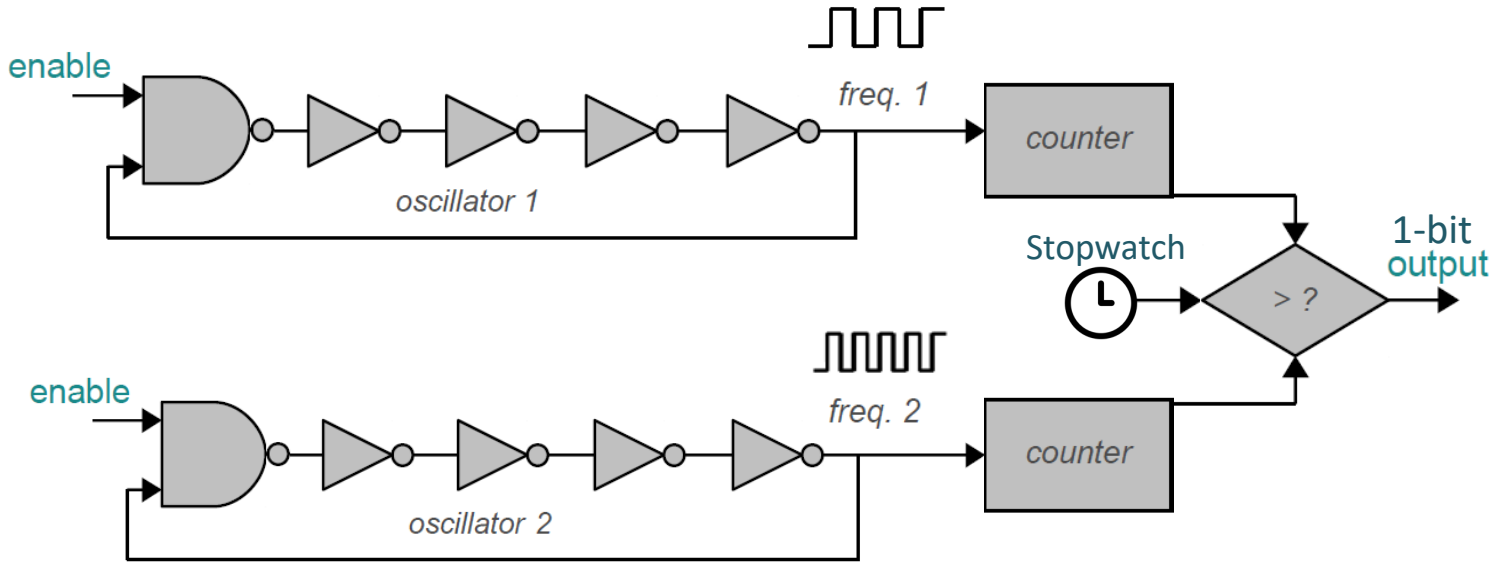
Delay PUF – Ring Oscillator PUF

- Using the intrinsic delay differences in each inverter (LUT);
- Weak PUF (*not that weak*)
  - $\frac{n(n-1)}{2}$, where n = # of ROs per RO group



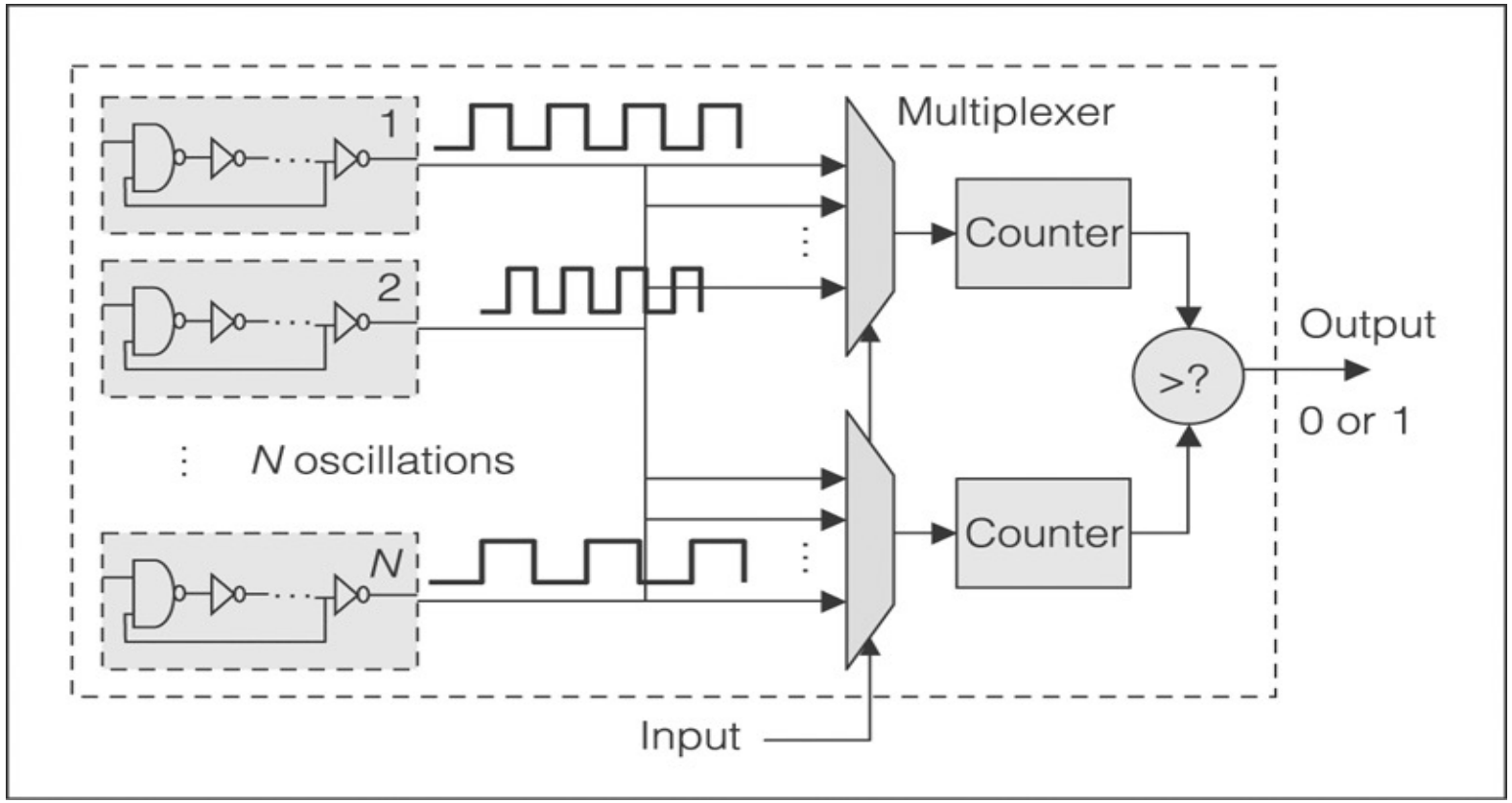
Delay PUF – Ring Oscillator PUF

- Easier to implement on FPGA
- Costs more area than Arbiter PUF



Delay PUF – Ring Oscillator PUF

- Multiple Oscillators improve strength of PUF
  - Number of CRPs grows quadratically

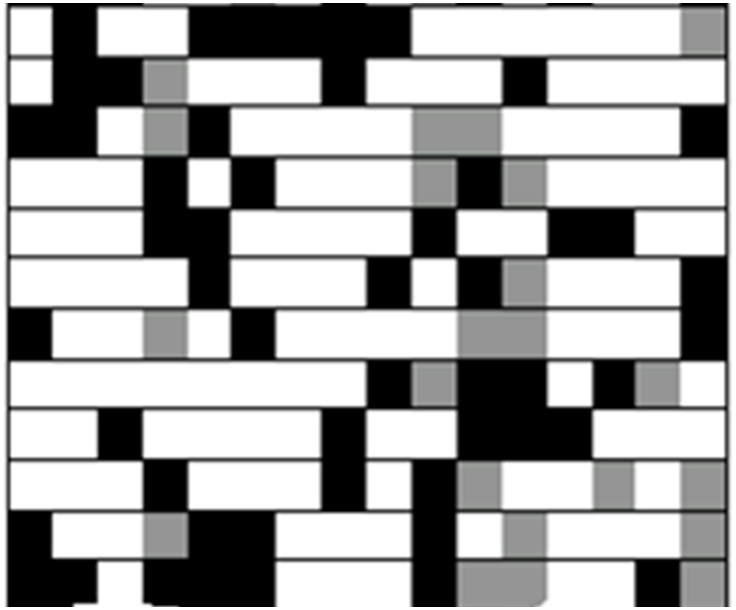


Types of (Silicon) PUFs

Memory PUFs

- Although the initial value of all cells at start up is unpredictable
- The stable ones should be selected for the PUF response
- A stable cell: it is read as 1 or 0 at most boot-ups

SRAM Cells

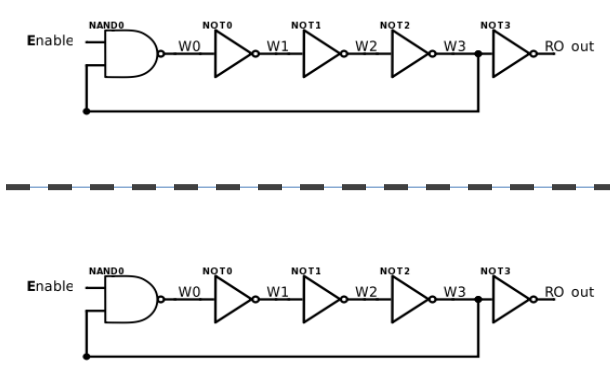


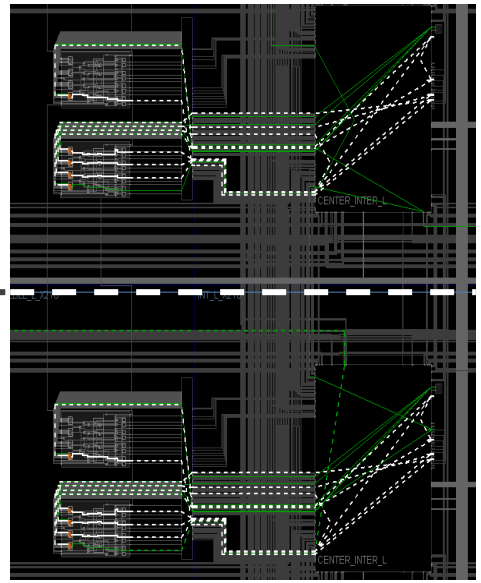
Black	Stable 1
White	Stable 0
Gray	Unstable Bits (Should not be Selected)

Delay PUFs

- Symmetric place & route
  - The two racing routes need to be identical / symmetric
  - Only factor determining delay difference is each cell's variation
    - Not dependent on routing difference.
- Difficult to achieve on FPGA for some designs
  - Simpler for ASICs

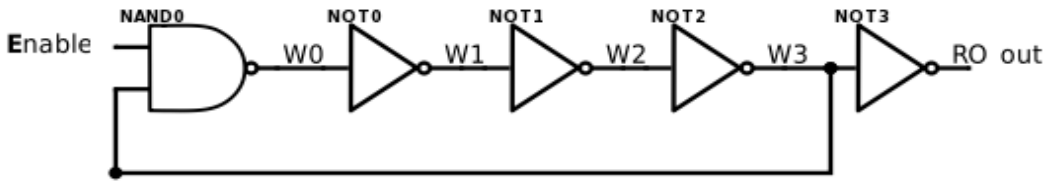
Two symmetrically routed ROs

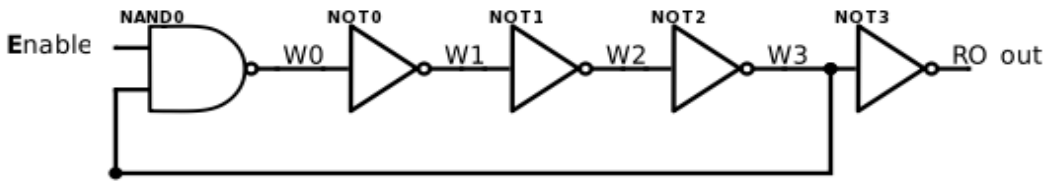


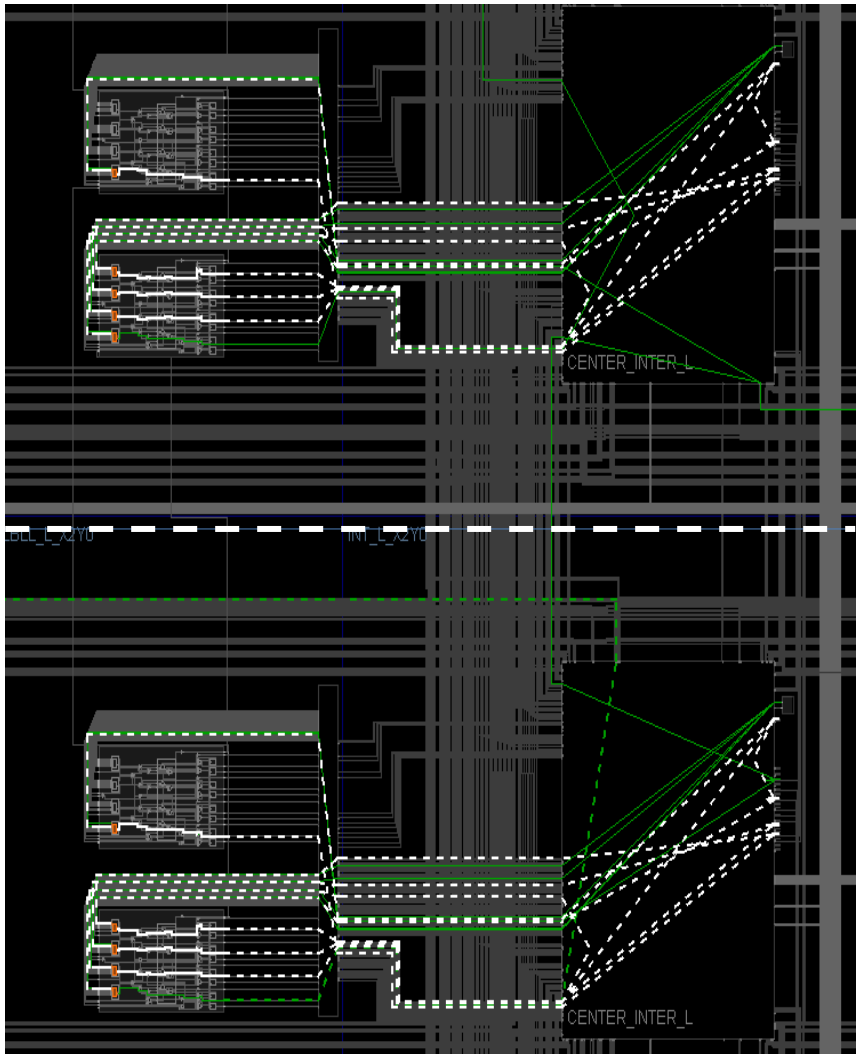


Delay PUFs

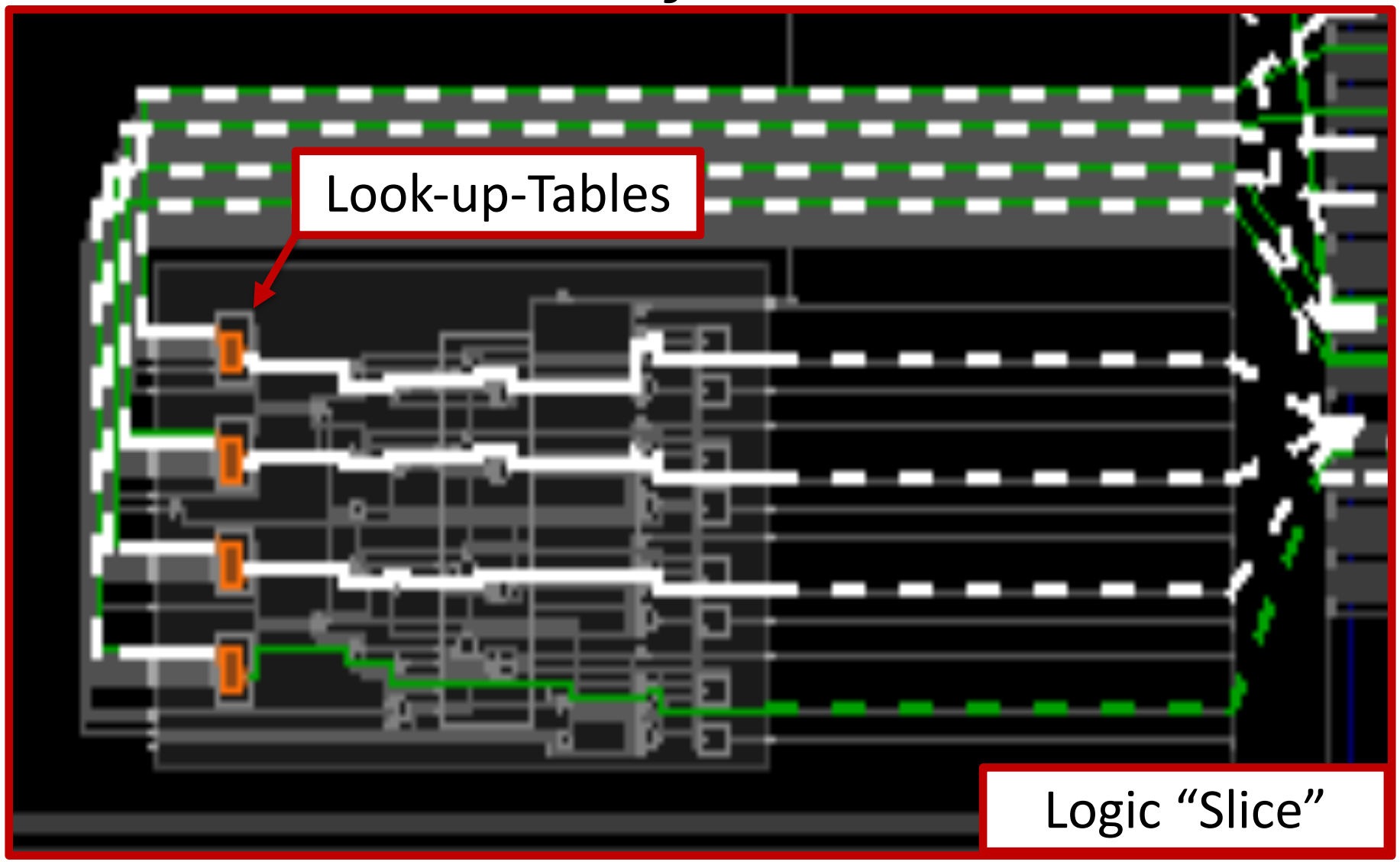
Two symmetrically routed ROs







Delay PUFs



Delay PUFs on FPGAs

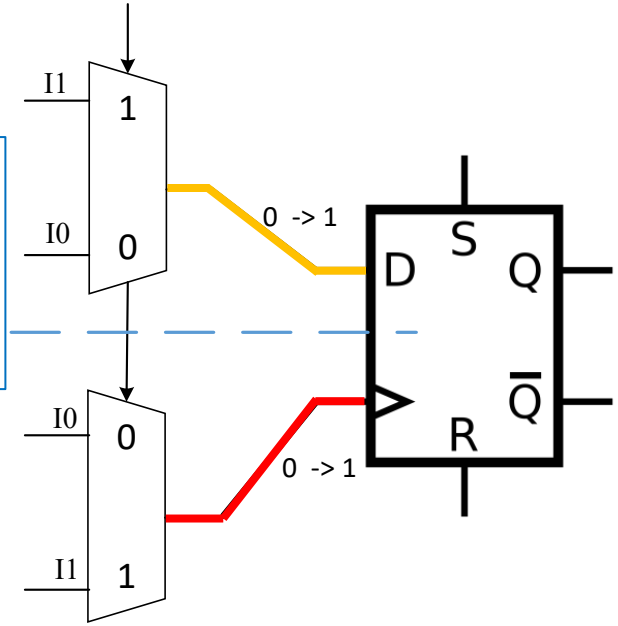
- Manually place and route delay sensitive modules with constraints
- Vivado uses ".xdc" format
- Placement is strait forward
- Routing for >2 "Slices" is a challenge

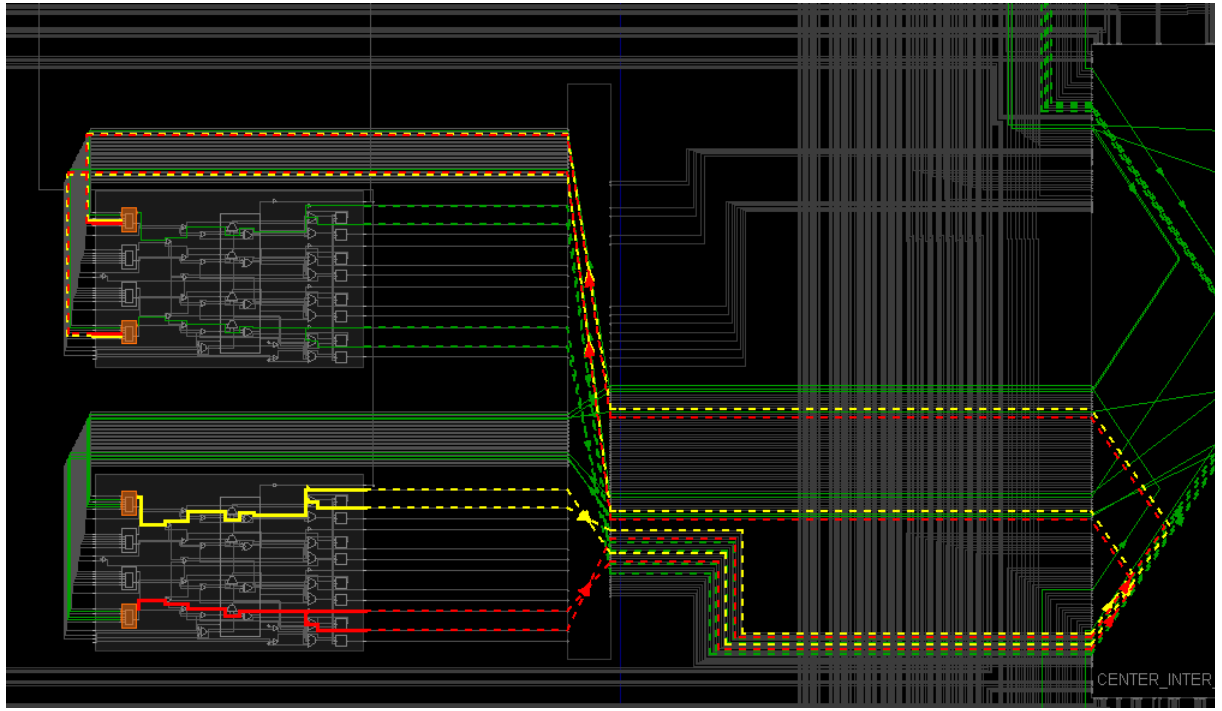
```
22
23 ##############################################################################
24 #  RO_G0/RO_0 Direct Routing
25 ##############################################################################
26 # The NAND gate
27 set_property BEL A6LUT [get_cells RO_G0/RO_0/nand_out0_inferred_i_1]
28 set_property LOC SLICE_X0Y0 [get_cells RO_G0/RO_0/nand_out0_inferred_i_1]
29 set_property LOCK_PINS {I0:A1 I1:A3} [get_cells RO_G0/RO_0/nand_out0_inferred_i_1]
30 set_property FIXED_ROUTE { CLBLL_LL_A CLBLL_LOGIC_OUTS12 IMUX_L24 CLBLL_LL_B5 }  [get_
31
```

Delay PUFs - Arbiters

- Delay sensitive routes must be identical
- Difficult to achieve with arbiters on FPGA
  - Routing between several slices

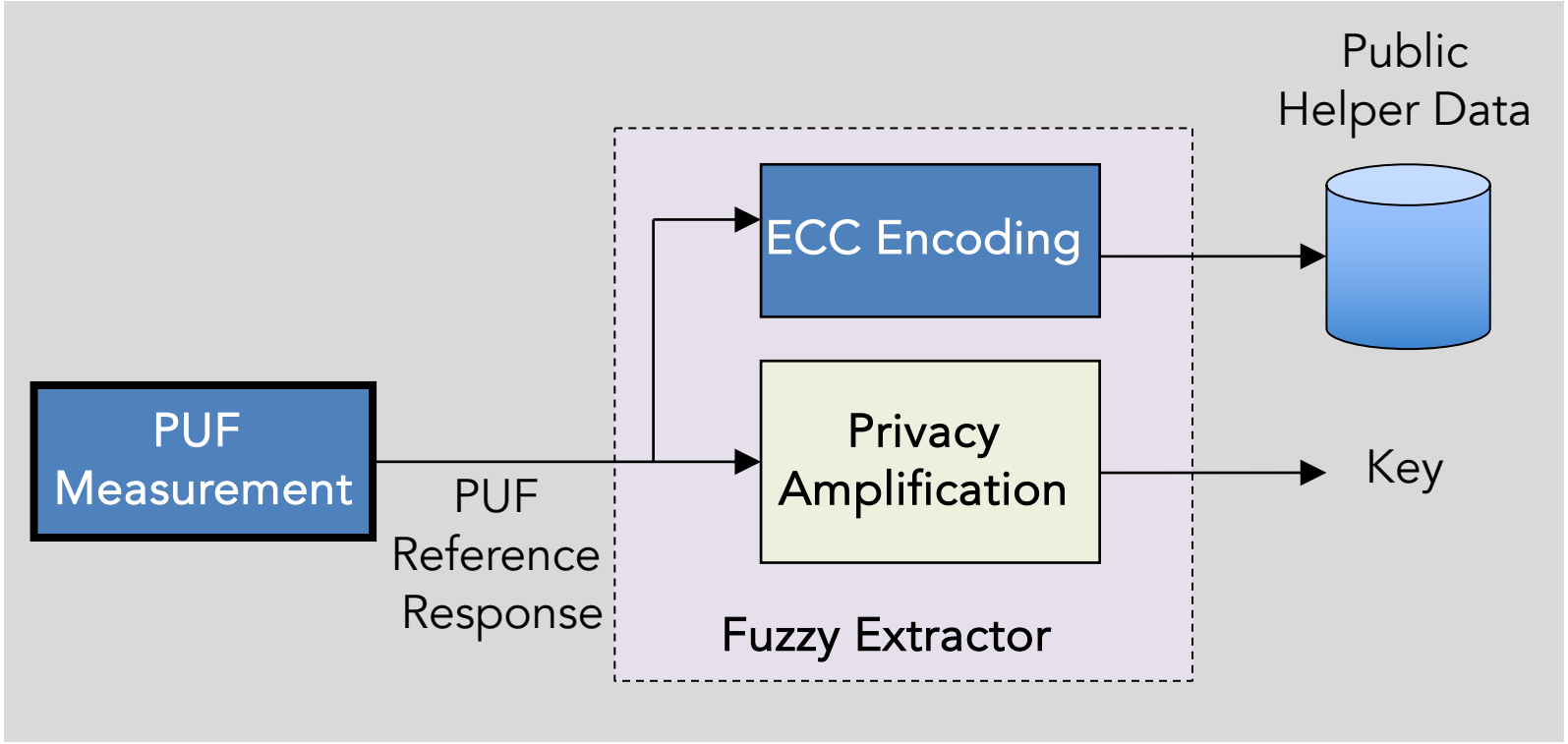
Two almost symmetrically routed MUXes





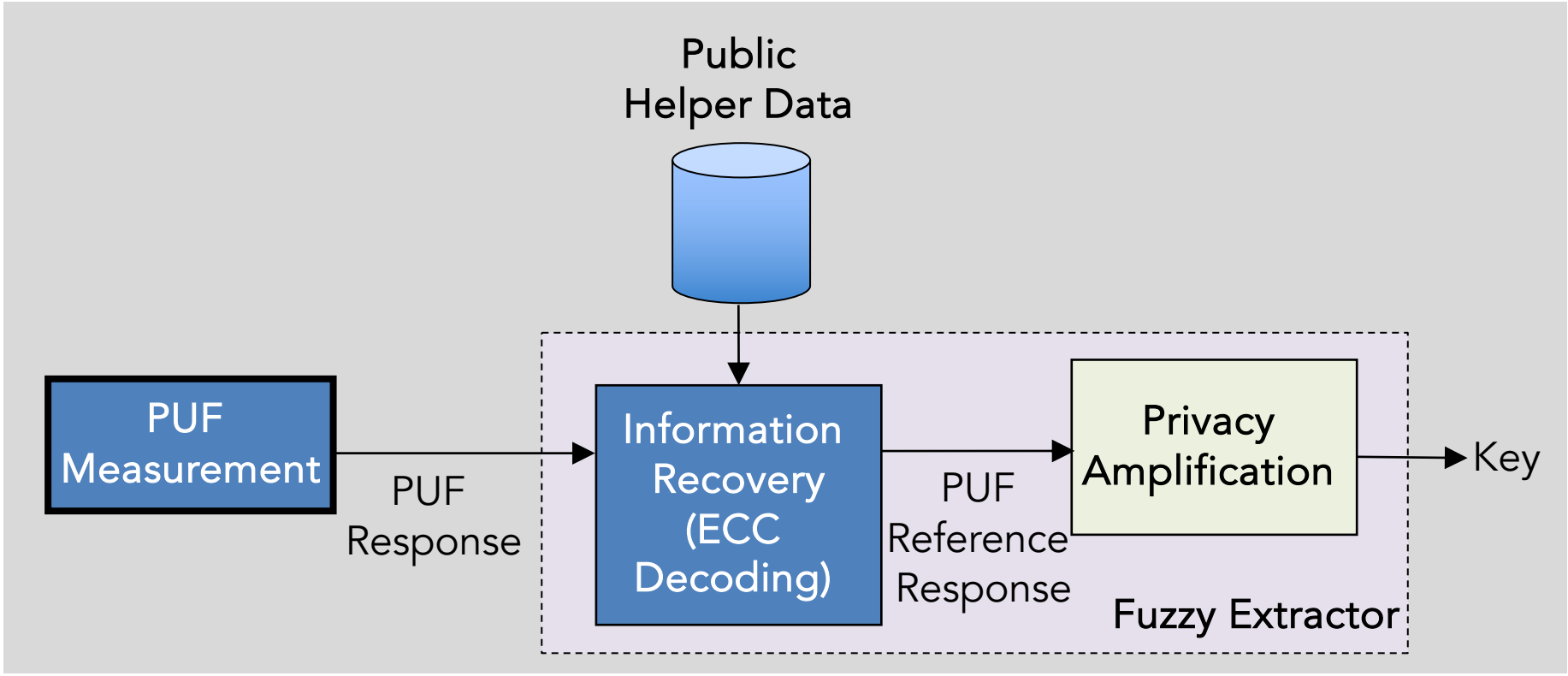
Fuzzy Extractor and Helper Data

- Enrollment process



Fuzzy Extractor and Helper Data

- Reconstruction process



PUF Security Evaluation Properties

- PUF designs are generally analyzed and evaluated with respect to hamming distance (HD), reliability, confidence interval, uniformity, and aliasing properties
  - For the hamming distance (HD) which measures the distance or bitwise difference between two responses R_i and R_j, both same-chip HD and multi-chip HD can be evaluated
  - Theoretically, for the same chip u, the HD for a 1-delay difference in challenges C_i and C_j is estimated with

$$HD_{same-chip} = \frac{1}{U}\sum_{n=1}^{U}\frac{HD(R_i, R_j)}{N} \times 100\%$$

PUF Security Evaluation Properties

- PUF designs are generally analyzed and evaluated with respect to hamming distance (HD), reliability, confidence interval, uniformity, and aliasing properties
  - For the hamming distance (HD) which measures the distance or bitwise difference between two responses R_i and R_j, both same-chip HD and multi-chip HD can be evaluated
  - Where U is the universe of chips and N the number of delays in the responses. When the same challenge C_i is applied to chips u and v

$$HD_{multi-chip} = \frac{2}{U(U-1)} \sum_{n=1}^{U-1} \sum_{v=2}^{U} \frac{HD(R_u, R_v)}{N} \times 100\%$$

PUF Security Evaluation Properties

- PUF designs are generally analyzed and evaluated with respect to hamming distance (HD), reliability, confidence interval, uniformity, and aliasing properties
  - Aliasing happens when different chips will produce similar responses
  - Aliasing avoidance is critical to protect the technique against controlled guesses Similarly, uniformity, which defines how uniform the delays are in the delay sequence, is a byproduct of the aliasing effect and also increases the vulnerability of the technique

$$Reliability = 100 - \frac{1}{K}\sum_{t=T_2}^{T_k}\frac{HD(R_{T1}, R_t)}{N} \times 100\%$$

  - Where T_1, T_2, ..., T_k are different time instances

Attacks on PUF: Clone the Unclonable

- Exhaustive Reading of the Weak PUFs
  - Reading out the only 1 CRP on memory PUFs On chip channel
- Modeling the Strong PUFs
  - With the large public subset of the CRPs of Arbiter, RO PUFs.
  - Machine Learning
  - Prediction of the unknown CRPs – 90% and up
- Side-Channel Analysis
  - Information leakage from the public helper data
  - Information leakage from power analysis

Upcoming Lectures

- Secure Hardware Primitives
  - ORAM
  - Hardware Trojans