MUTARCH: Architectural Diversity for FPGA Device and IP Security

Robert Karam*[1], Tamzidul Hoque[1], Sandip Ray[2], Mark Tehranipoor[1], and Swarup Bhunia[1]

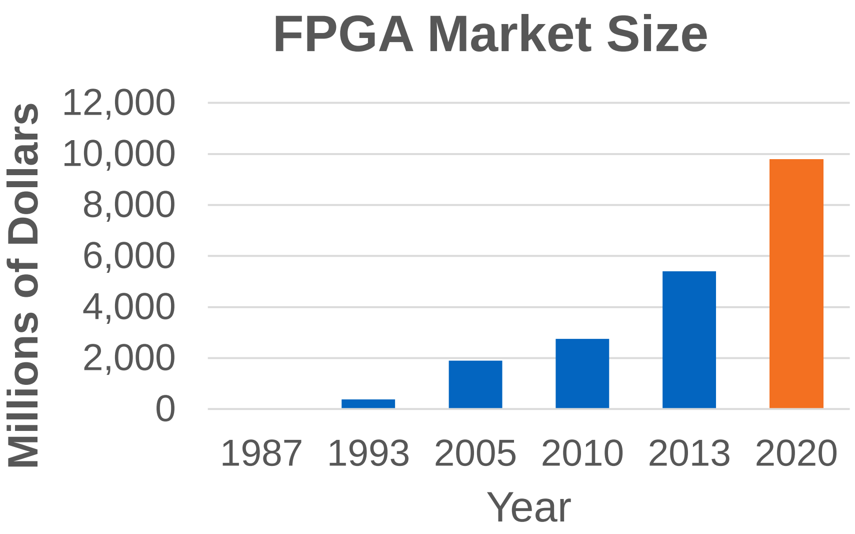
*Email: robkaram@ufl.edu
[1] University of Florida, Gainesville, FL, USA
[2] NXP Semiconductor, Austin, TX, USA

Outline

- **Introduction & Motivation**
  - Proliferation of FPGAs
  - Challenges for Encryption in IoT
  - Attacks on FPGA Bitstreams
- **Proposed Solution**
  - MUTARCH: Mutable FPGA Architecture
  - MUTARCH-enabled Obfuscation
  - Design & Upgrade Flow
- **Results**
  - Experimental Validation
  - Security Analysis / Performance Impact
- **Conclusion**

Introduction

- **FPGAs are increasingly used in numerous applications**
  - Automotive, Defense, Healthcare, Networking, *Internet of Things*
  - Reduce time to market & development costs (compared to ASIC) while providing better energy-efficiency (compared to processor)
- **~$9.8 billion market by 2020, >$14 billion by 2024[1]**



Why Intel will spend $16.7 billion on Altera

by Stacey Higginbotham @gigastacey AUGUST 27, 2015, 7:21 PM EST

[1] http://www.grandviewresearch.com/industry-analysis/fpga-market

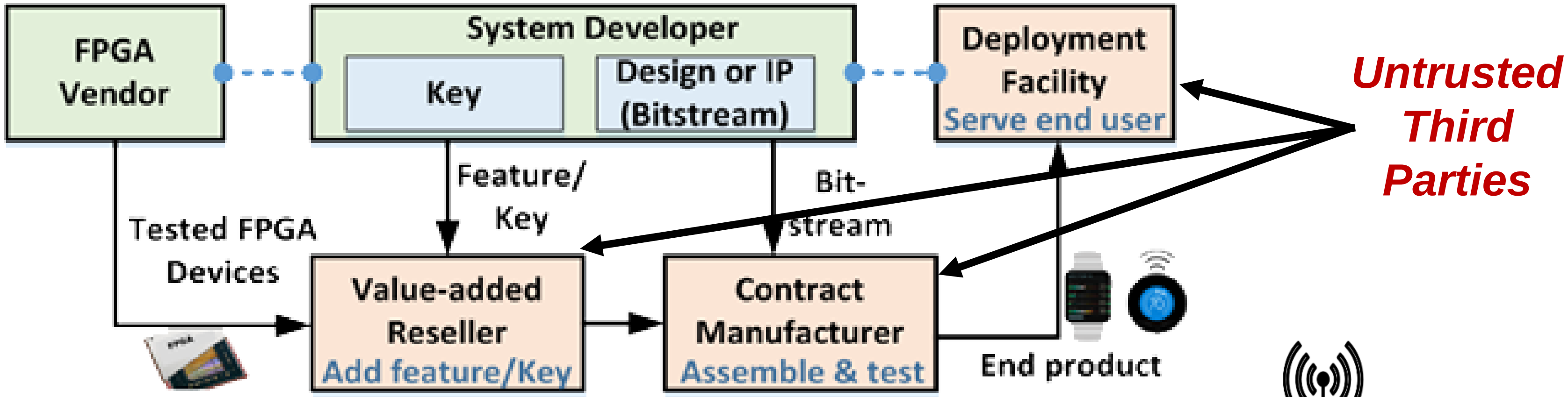
Motivation: Encryption in IoT Domain

- **Security is more challenging in IoT**
  - Remote/in-field reconfiguration
  - Long in-field lifetimes (physical attacks)
- **Encryption is strong, but not enough by itself for IoT**
  - Key extraction through DPA
  - eFUSE keys programmed at UTP facility
- **Symmetric key not always suitable for remote upgrade**
- **PKC not ideal for constrained environment**
  - Area/power intensive decryption blocks
  - May not be suitable for runtime reconfiguration applications

Need a novel approach to improve security while that can maintain interoperability, minimally affecting design flow, and without incurring significant overhead

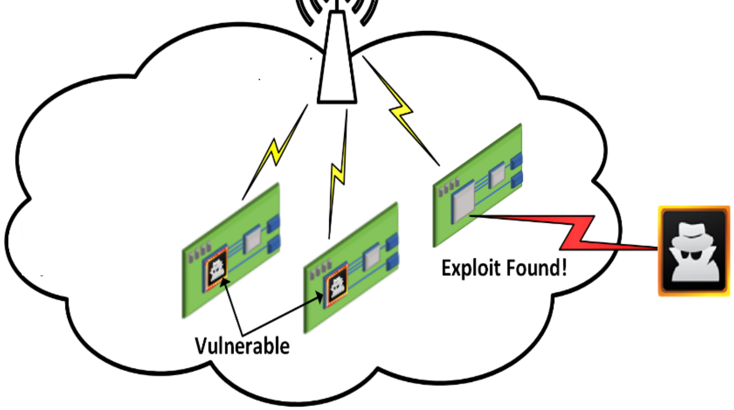
Motivation: Attacks on Bitstreams

- **FPGAs are vulnerable to various attacks**
  - Intellectual Property (IP) Piracy
  - (Targeted) Malicious Modification



Untrusted Third Parties

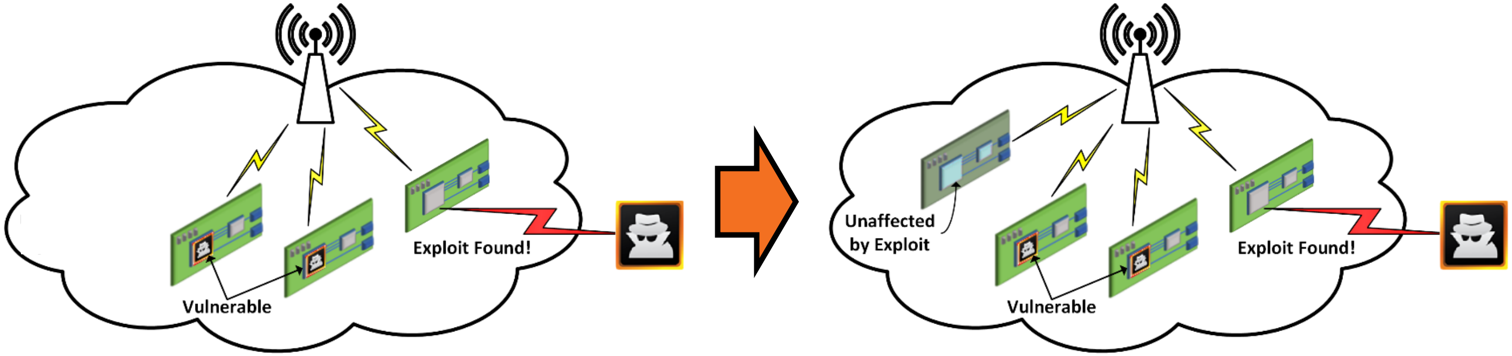
- **Device architectures are identical**
  - Easier to design, test, and manufacturer
  - *Break one, break all*



Proposed Solution: MUTARCH

- **MUTARCH: Mutable Architecture for FPGA**
  - Genetic diversity in nature helps ensure survival of species
  - Diverse FPGA architectures make all devices less vulnerable to attack

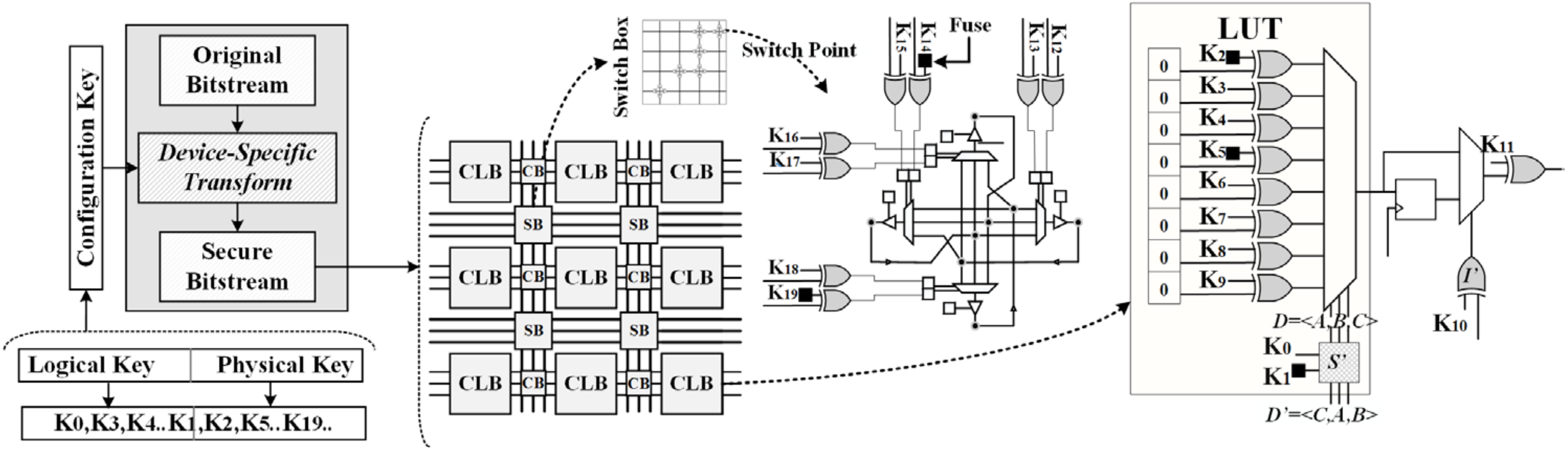
No longer "break one, break all"



- **Like software node-locking: *One Bitstream, One Device***
- **Provides strong protection against RE & TMM for HW IP**
- ***Can be used in conjunction with encryption*, or standalone in ultra-lightweight applications**

MUTARCH for FPGAs

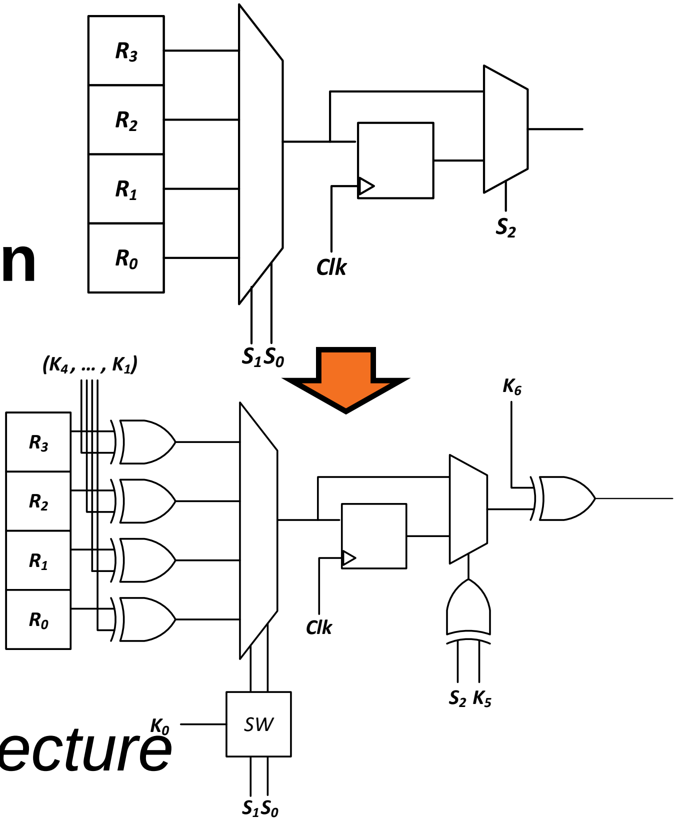
- **Static *physical* and time-varying *logical* architectures**
  - Make every device physically different
  - Logically change architecture over time
- Architectural configuration based on a key
- Design for *next-generation FPGAs, not existing ones*[1]



[1] *Karam, et al. "Robust Bitstream Protection in FPGA-based systems through Low Overhead Obfuscation," ReConFig, 2016.*

MUTARCH for FPGAs

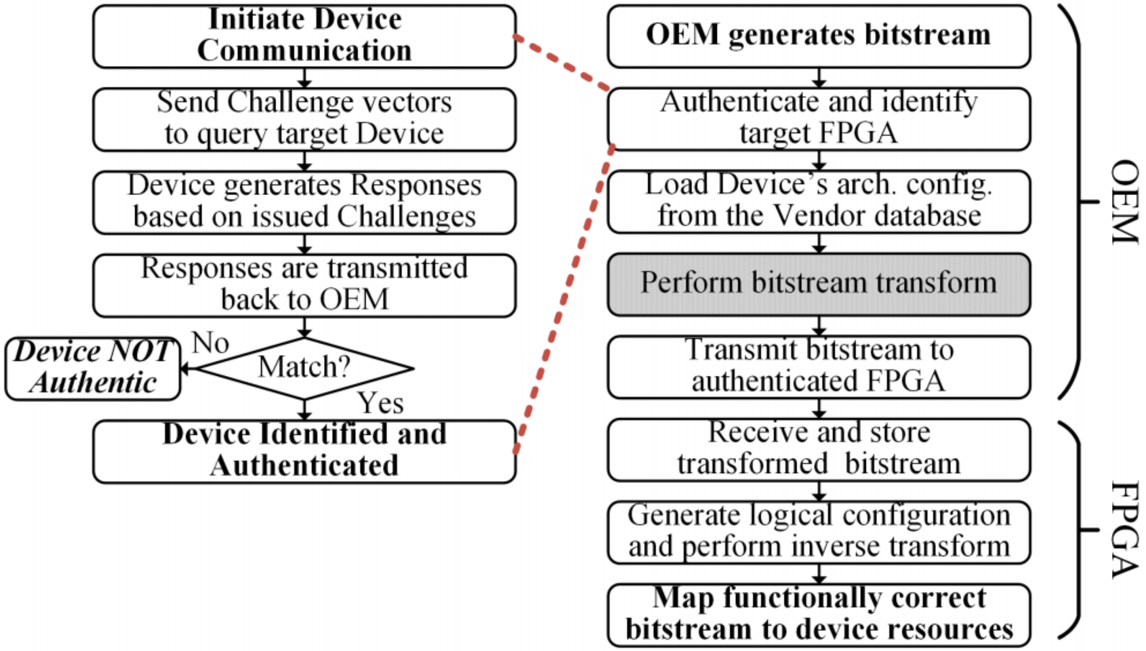
- **Architectural modifications perform inverse obfuscation**
- **Key-based obfuscation networks can be extended**
  - LUT Outputs
  - Programmable Interconnects
  - DSP Blocks, Multipliers, etc.
  - Blockram/Embedded Memories
- *Can modify any aspect of FPGA architecture*



Key Type	Time Var	Storage	Area Ovhd	In-Field Upg.	Known Design	Destructive RE
Physical	No	Fuses	Low	Not Secure	Weak	Strong
Logical	Yes	Runtime	Moderate	Secure	Strong	Weak
Combined	Yes	Mixed	High	Secure	Strong	Strong

Secure In-field Upgrades

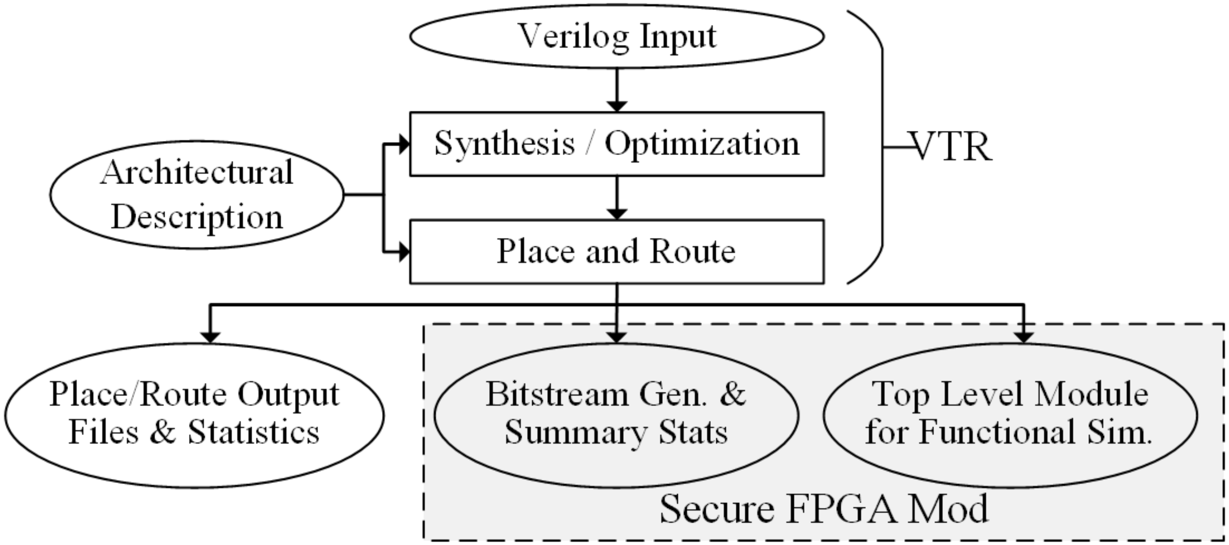
- **Firmware upgrade is essential for long-life devices**
  - Optimize design and add new features
  - Patch security vulnerabilities
- **MUTARCH update requires device identification**
  - Bitstream obfuscated using key specific to target architecture
  - Obfuscation natively inverted in only target device



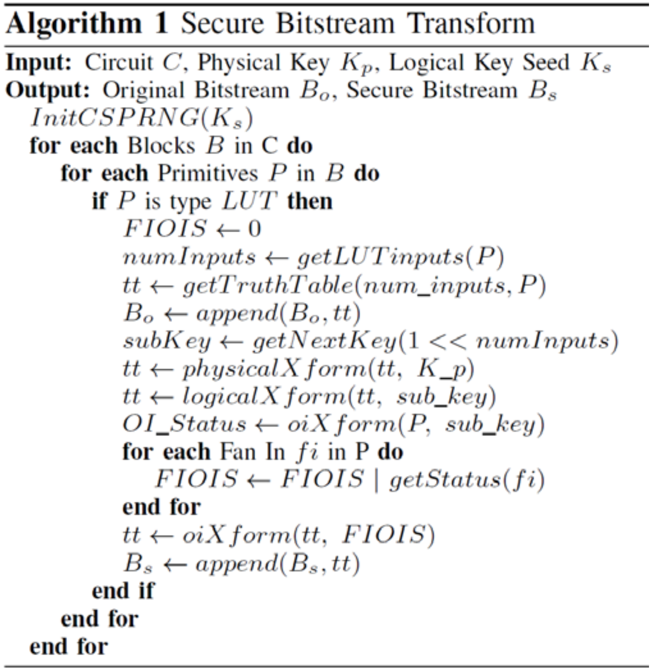
Initiate Device Communication
Send Challenge vectors to query target Device
Device generates Responses based on issued Challenges
Responses are transmitted back to OEM
Match?
No
Yes
Device NOT Authentic
Device Identified and Authenticated
OEM generates bitstream
Authenticate and identify target FPGA
Load Device's arch. config. from the Vendor database
Perform bitstream transform
Transmit bitstream to authenticated FPGA
Receive and store transformed bitstream
Generate logical configuration and perform inverse transform
Map functionally correct bitstream to device resources
OEM
FPGA

Evaluation

- **Prototype system based on VTR[1]**
  - Synthesis / Place & Route Verilog benchmarks
  - Compare original bitstream w/ bitstream for MUTARCH FPGA after Secure Bitstream Transform
  - Evaluate security (brute force, side channel, known design, reverse engineering)



Verilog Input
Synthesis / Optimization
Architectural Description
Place and Route
VTR
Place/Route Output Files & Statistics
Bitstream Gen. & Summary Stats
Top Level Module for Functional Sim.
Secure FPGA Mod

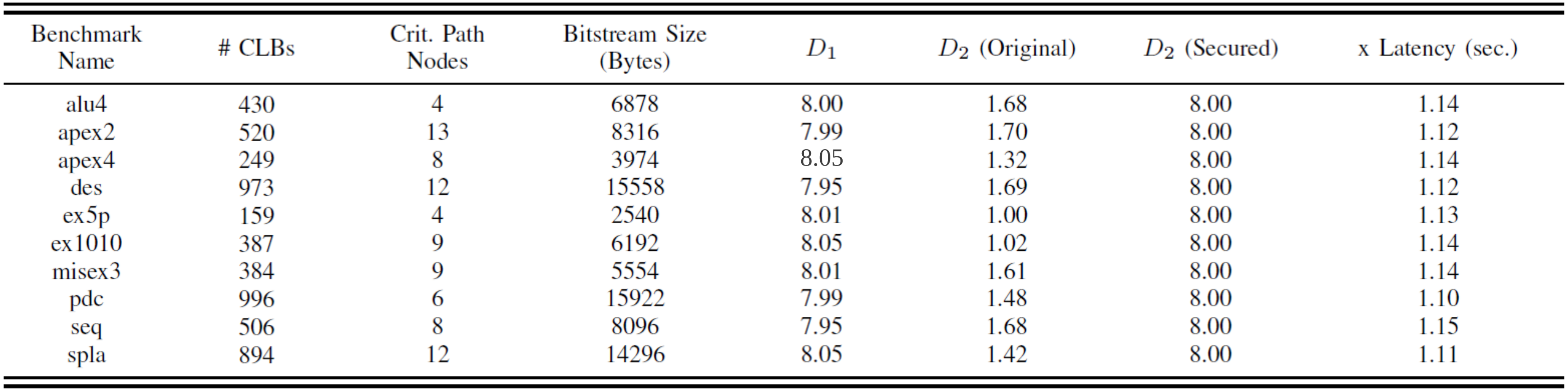


Algorithm 1 Secure Bitstream Transform

```
Input: Circuit $C$, Physical Key $K_p$, Logical Key Seed $K_s$
Output: Original Bitstream $B_o$, Secure Bitstream $B_s$
  $InitCSPRNG(K_s)$
  for each Blocks $B$ in C do
    for each Primitives $P$ in $B$ do
      if $P$ is type $LUT$ then
        $FIOIS \leftarrow 0$
        $numInputs \leftarrow getLUTinputs(P)$
        $tt \leftarrow getTruthTable(num\_inputs, P)$
        $B_o \leftarrow append(B_o, tt)$
        $subKey \leftarrow getNextKey(1 << numInputs)$
        $tt \leftarrow physicalXform(tt,\ K\_p)$
        $tt \leftarrow logicalXform(tt,\ sub\_key)$
        $OI\_Status \leftarrow oiXform(P,\ sub\_key)$
        for each Fan In $fi$ in P do
          $FIOIS \leftarrow FIOIS \mid getStatus(fi)$
        end for
        $tt \leftarrow oiXform(tt,\ FIOIS)$
        $B_s \leftarrow append(B_s, tt)$
      end if
    end for
  end for
```

[1] **J. Luu et al., "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," ACM TRETS,**

Results



Benchmark Name	# CLBs	Crit. Path Nodes	Bitstream Size (Bytes)	D_1	D_2 (Original)	D_2 (Secured)	x Latency (sec.)
alu4	430	4	6878	8.00	1.68	8.00	1.14
apex2	520	13	8316	7.99	1.70	8.00	1.12
apex4	249	8	3974	8.05	1.32	8.00	1.14
des	973	12	15558	7.95	1.69	8.00	1.12
ex5p	159	4	2540	8.01	1.00	8.00	1.13
ex1010	387	9	6192	8.05	1.02	8.00	1.14
misex3	384	9	5554	8.01	1.61	8.00	1.14
pdc	996	6	15922	7.99	1.48	8.00	1.10
seq	506	8	8096	7.95	1.68	8.00	1.15
spla	894	12	14296	8.05	1.42	8.00	1.11

$$D_1 = \frac{\sum_{i=0}^{N} HD(B_{O,i}, B_{S,j})}{N}$$

Inter-bitstream Hamming Distance

$$D_2 = \frac{\sum_{i=0}^{N} \sum_{j=i+1}^{N} HD(B_i, B_j)}{[N(N-1)]/2}$$

Intra-bitstream Hamming Distance

4-input LUT (16 content bits), HD = 8 = 50%

Security Analysis

- **Brute Force**
  - Determining configuration key from test patterns via simulation
  - Each trial requires significant computation
  - *Same procedure as brute forcing encryption*
- **Known Design**
  - Mapping multiple known designs to better understand obfuscation
  - Logical key changes resulting bitstream even in same design
  - *Provides moving target defense*
- **Side Channel**
  - Analyze power/time/etc. side channels when mapping application
  - Key generation at runtime may be susceptible
  - *No key used in physical network*

Conclusion

- **FPGA security using concept of diversity (esp. for IoT)**
- **Enables lightweight, secure remote reconfiguration**
- **Improves security for devices in field**
  - IP Piracy
  - Targeted tampering attacks
- **Requires minor changes to FPGA architecture**
  - Toolflow (almost) the same
  - Upgrades using existing infrastructure
- **Enables security/overhead tradeoff**
- ***Can still be used with existing encryption techniques***

Thank You!