RUB

TECHNISCHE UNIVERSITÄT DARMSTADT

MAX PLANCK INSTITUTE FOR SECURITY AND PRIVACY

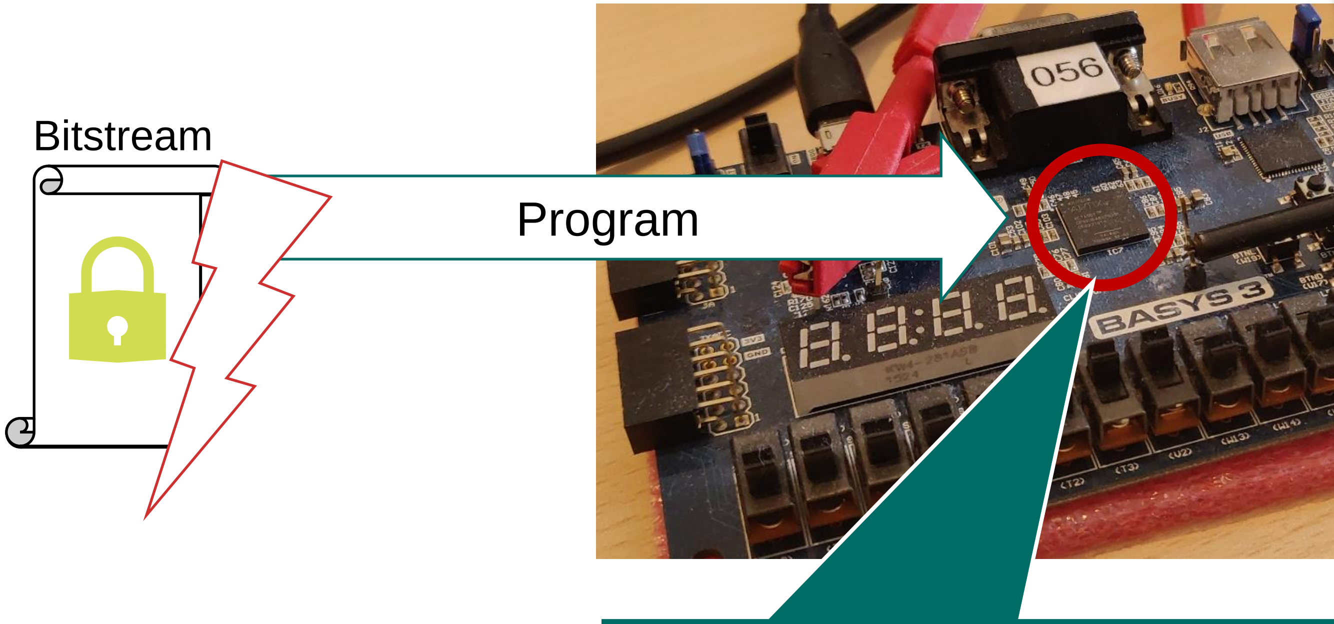
CONFUZZ: COMBINING HARDWARE REVERSE ENGINEERING AND SECURITY ANALYSIS THROUGH FUZZING

Maik Ender*, Felix Hahn*, Marc Fyrbiak*, Amir Moradi‡, and Christof Paar*

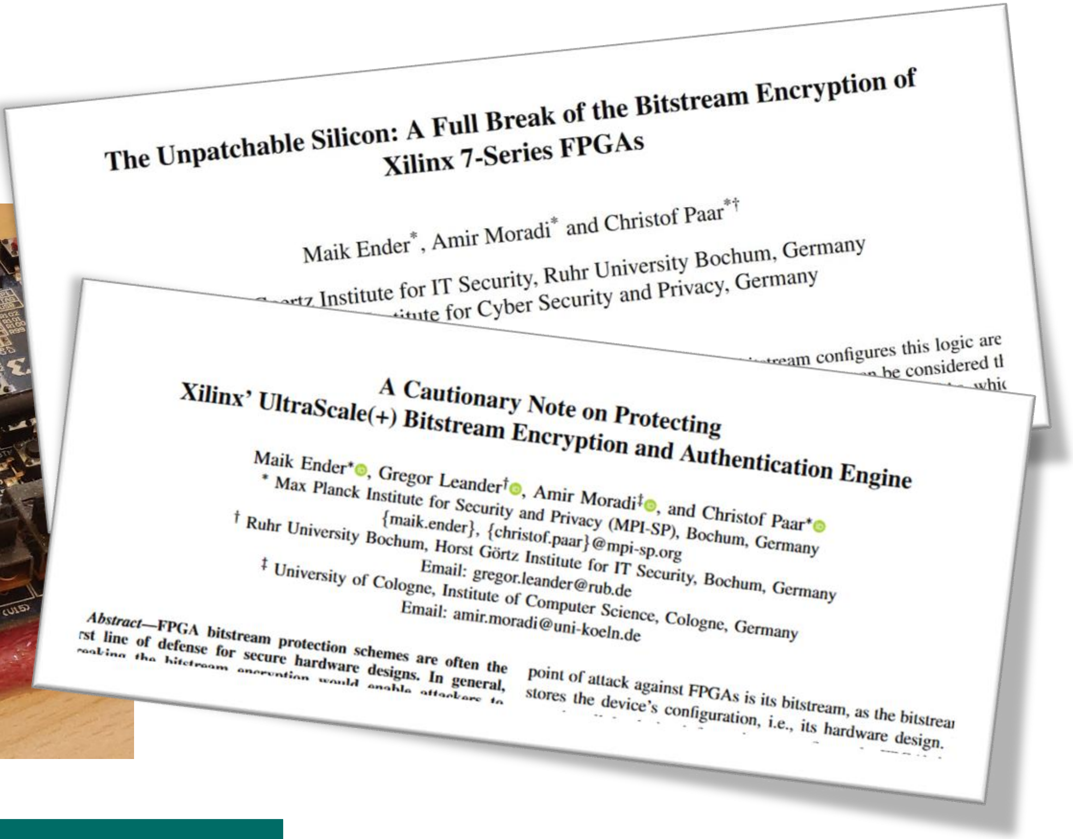
* MPI-SP, ‡ University of Darmstadt

Harris Workshop, March 19, 2024

BITSTREAM SECURITY



Field **Programmable** Gate Array (FPGA)



The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs

Maik Ender[*], Amir Moradi[*] and Christof Paar[*†]

…rtz Institute for IT Security, Ruhr University Bochum, Germany

…itute for Cyber Security and Privacy, Germany

…tream configures this logic are …n be considered tl … whic

A Cautionary Note on Protecting Xilinx' UltraScale(+) Bitstream Encryption and Authentication Engine

Maik Ender[*], Gregor Leander[†], Amir Moradi[‡], and Christof Paar[*]

[*] Max Planck Institute for Security and Privacy (MPI-SP), Bochum, Germany
{maik.ender}, {christof.paar}@mpi-sp.org

[†] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
Email: gregor.leander@rub.de

[‡] University of Cologne, Institute of Computer Science, Cologne, Germany
Email: amir.moradi@uni-koeln.de

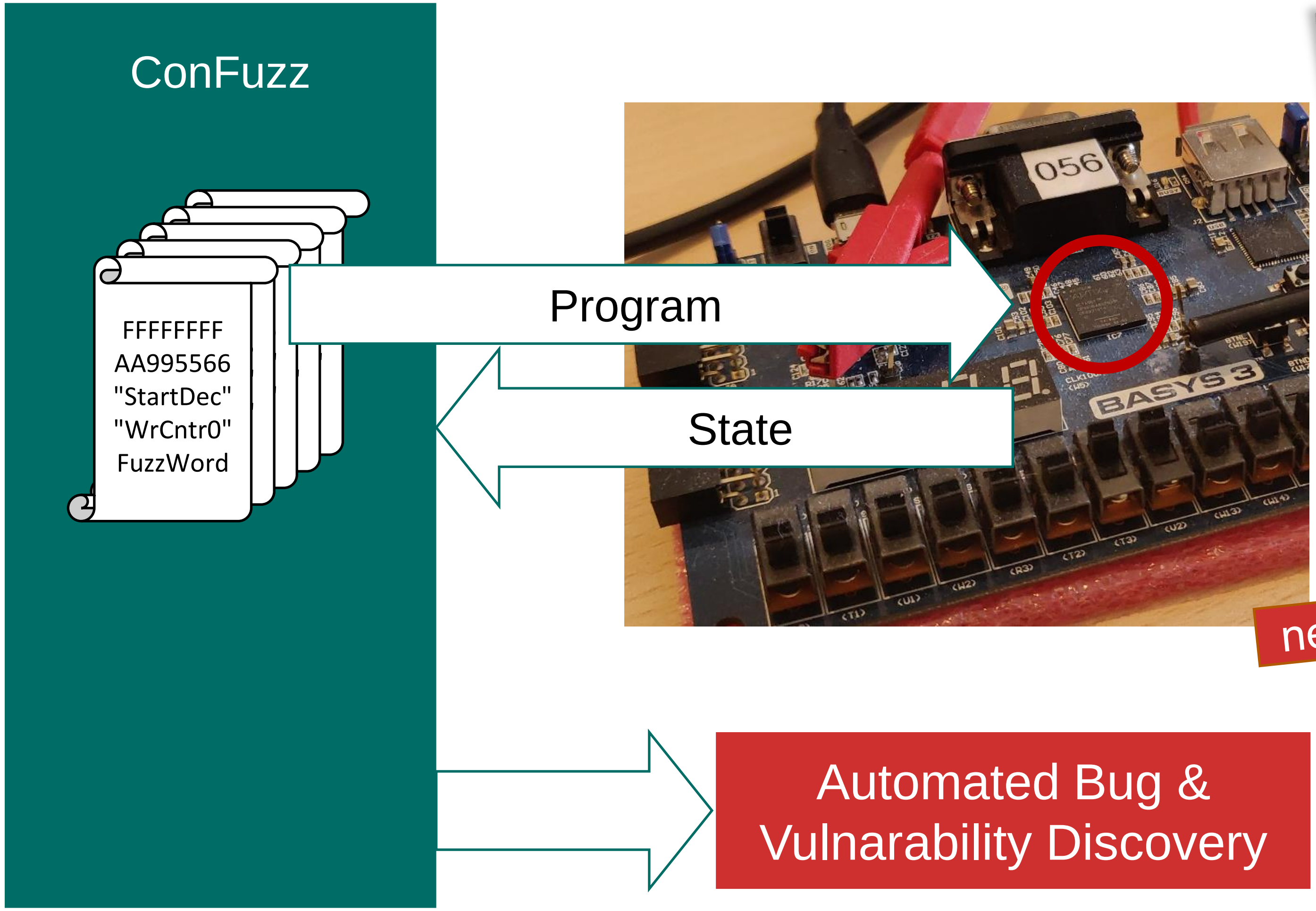
Abstract—**FPGA bitstream protection schemes are often the …rst line of defense for secure hardware designs. In general, …eaking the bitstream encryption would enable attackers to**

point of attack against FPGAs is its bitstream, as the bitstream stores the device's configuration, i.e., its hardware design.

FUZZING

1

BITSTREAM FUZZING



ConFuzz

FFFFFFFF
AA995566
"StartDec"
"WrCntr0"
FuzzWord

Program

State

Automated Bug & Vulnarability Discovery

new

The Unpatchable Silicon: A Full Break of the Bitstream Encryption of Xilinx 7-Series FPGAs

Maik Ender*, Amir Moradi* and Christof Paar*†

Institute for IT Security, Ruhr University Bochum, Germany
itute for Cyber Security and Privacy, Germany

tream configures this logic are
n be considered t

A Cautionary Note on Protecting Xilinx' UltraScale(+) Bitstream Encryption and Authentication Engine

Maik Ender*, Gregor Leander†, Amir Moradi‡, and Christof Paar*
* Max Planck Institute for Security and Privacy (MPI-SP), Bochum, Germany
{maik.ender}, {christof.paar}@mpi-sp.org
† Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochu
Email: gregor.leander@rub.de
‡ University of Cologne, Institute of Com

Abstract—FPGA bitstr

JustSTART: How to Find an RSA Authentication Bypass on Xilinx UltraScale(+) with Fuzzing

Maik Ender[1], Felix Hahn[1], Marc Fyrbiak[1], Amir Moradi[2] and Christof Paar[1]

[1] Max Planck Institute for Security and Privacy, Bochum, Germany
firstname.lastname@mpi-sp.org
[2] Technische Universität Darmstadt, Darmstadt, Germany
amir.moradi@tu-darmstadt.de

Abstract. Fuzzing is a well-established technique in the software domain to uncover

FUZZING GOALS

Automated bug & vulnerability discovery

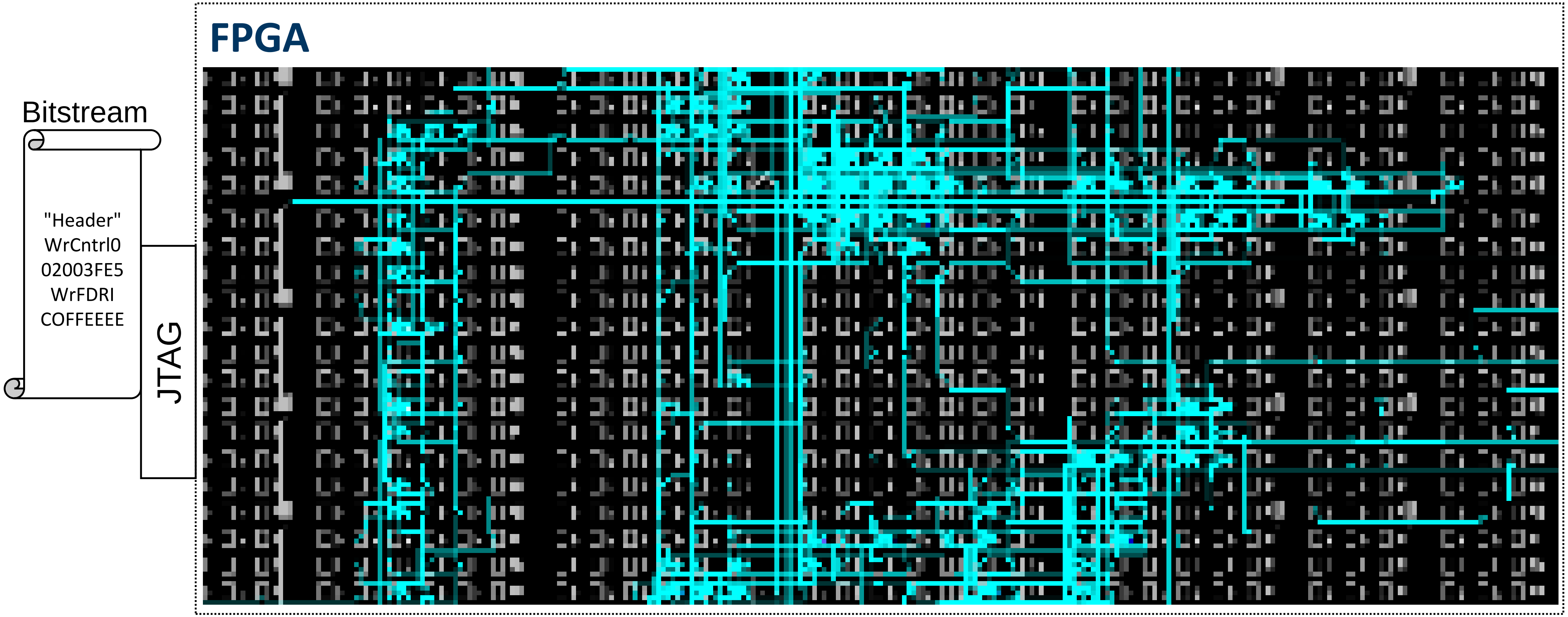
Reverse engineer Xilinx configuration engine

Explore fuzzing as a general defense technique for hardware

CONFIGURATION ENGINE FUNDAMENTALS

2

CONFIGURATION ENGINE



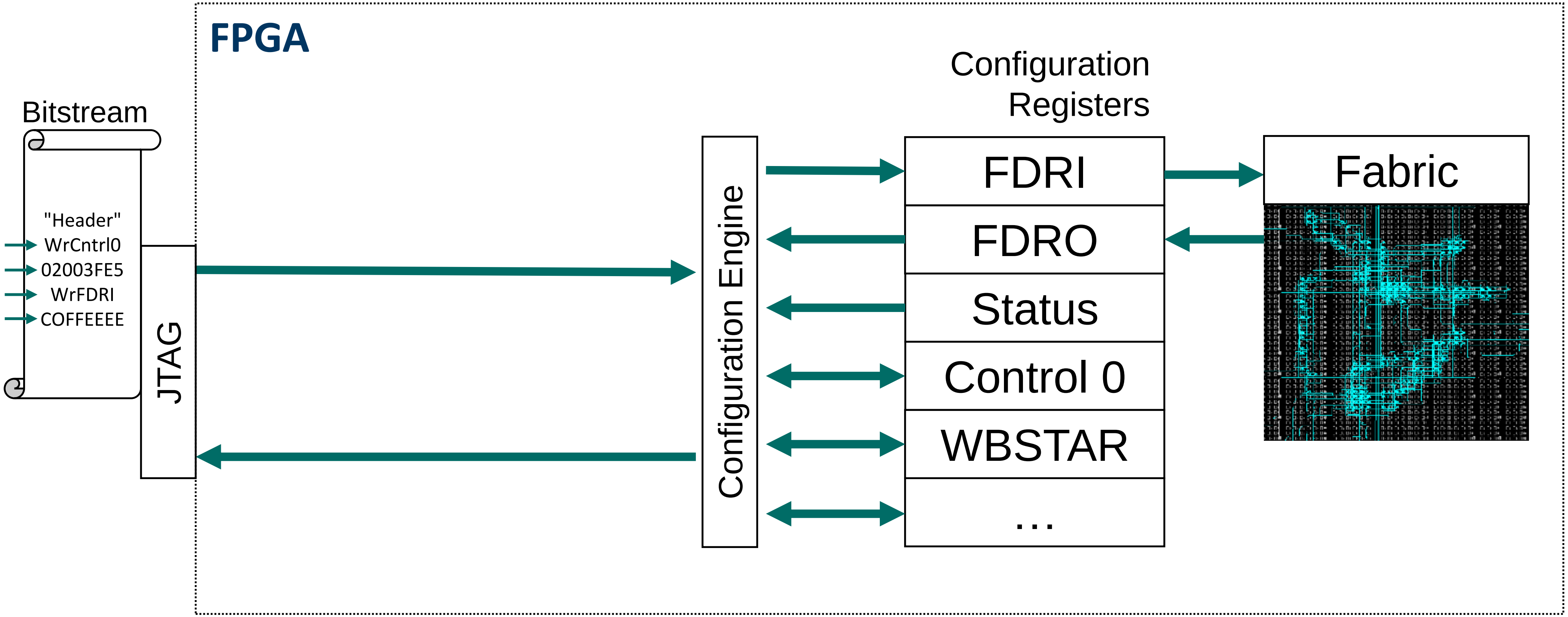
Bitstream

"Header"
WrCntrl0
02003FE5
WrFDRI
COFFEEEE

JTAG

FPGA

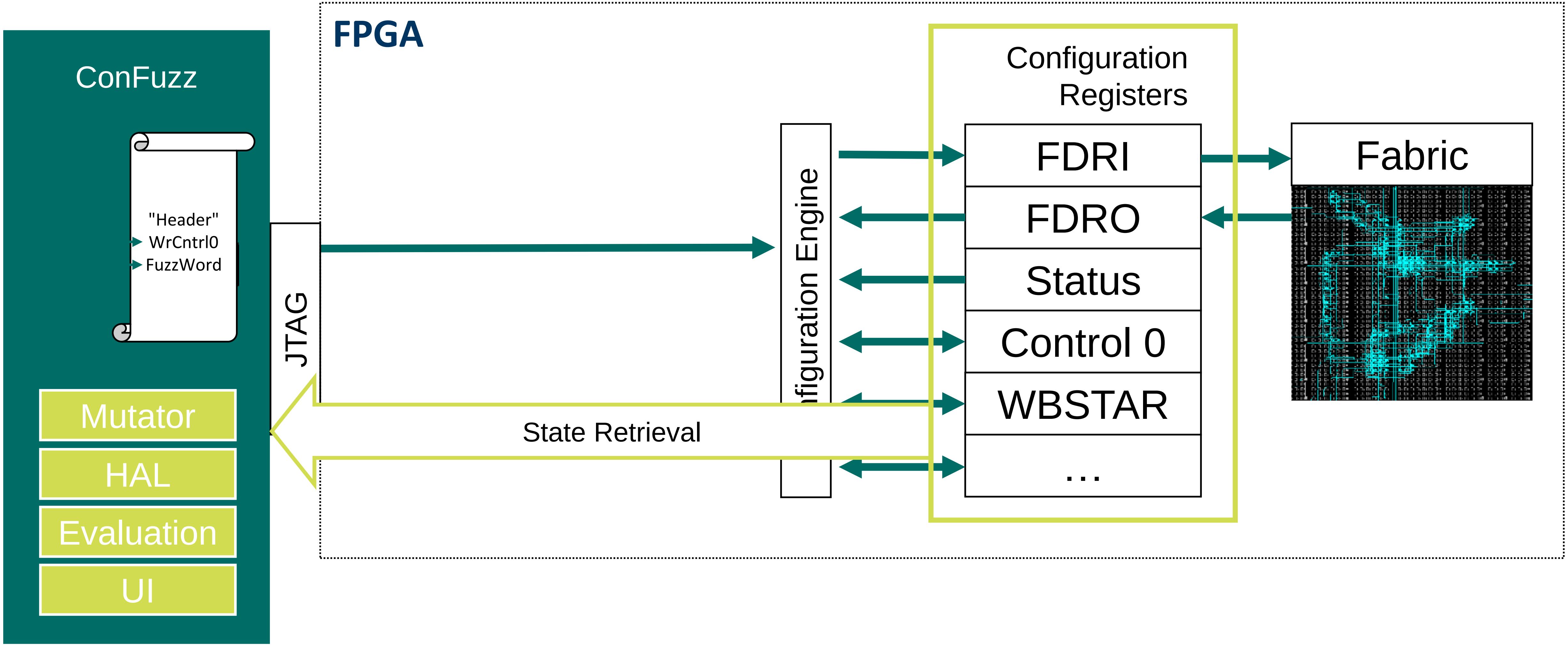
BITSTREAM PROGRAM



3

BITSTREAM FUZZING

BITSTREAM FUZZING



FUZZING STRATEGIES

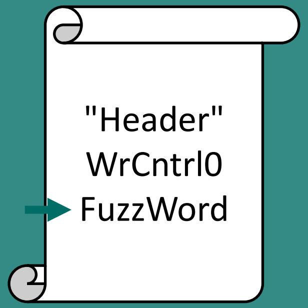
1. Bitstream Structure

Fuzz the general bitstream instruction set architecture

WrFDRI

2. Intra Command

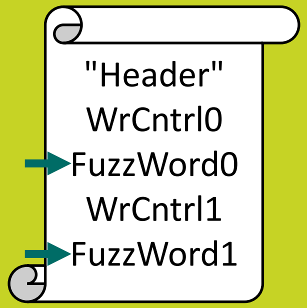
Fuzz single configuration registers (bit pattern)



"Header"
WrCntrl0
FuzzWord

3. Inter Command

Fuzz interaction between multiple registers and commands



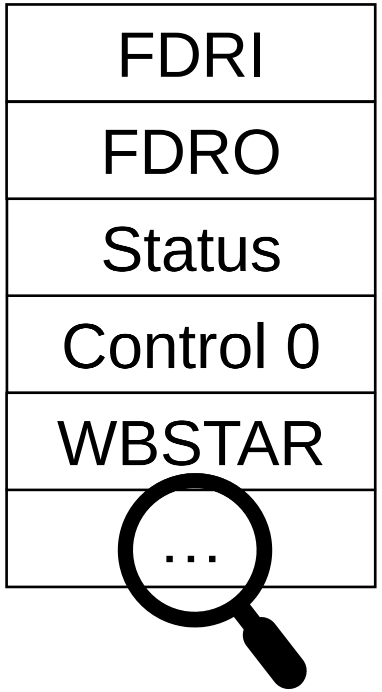
"Header"
WrCntrl0
FuzzWord0
WrCntrl1
FuzzWord1

FINDINGS

4

FINDINGS

- Reverse engineer towards better understanding
- Hard crash in register 23 (power cycle needed)
- RSA authentication test mode (rapid prototyping)
- Re-discover starbleed automatically
- Discover JustSTART



FDRI

FDRO

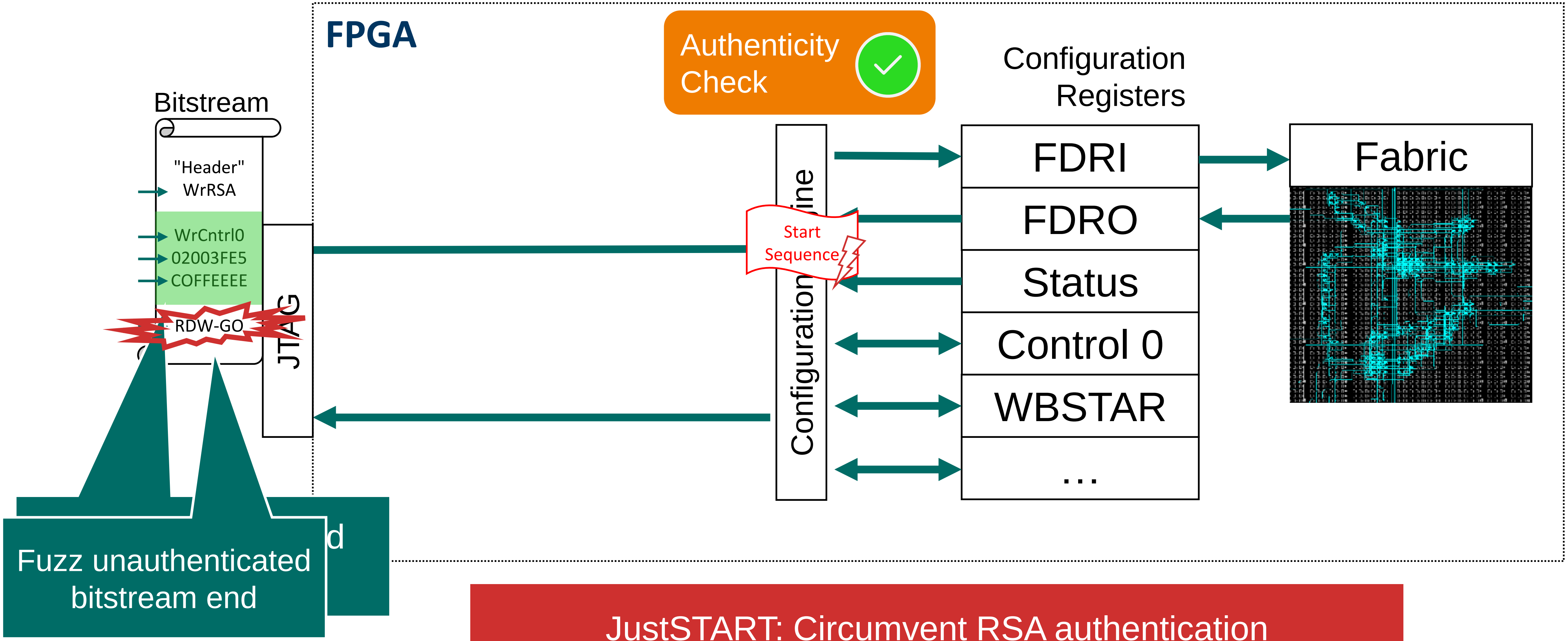
Status

Control 0

WBSTAR

…

JUSTSTART



Fuzz unauthenticated bitstream end

JustSTART: Circumvent RSA authentication
just start the FPGA

CONCLUSION

Fuzzing on Hardware

- Can be effective
  - Found new vulnerabilities
  - Better understanding
- Efficiency:
  - Strategies
  - Rapid prototyping

Limitations

- Scalability (Hardware for every Instance, slow interfaces)
- Internal state (in-)visibility
  Future work: Use side-channels
- Human assisted evaluation
  Future work: Automation

github.com
/emsec/ConFuzz