

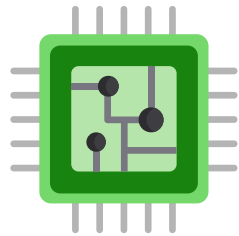
Insights into the Mind of a Trojan Designer

The Challenge to Integrate a Trojan into the Bitstream

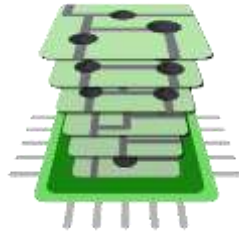
Maik Ender, Pawel Swierczynski, Sebastian Wallat, Matthias Wilhelm, Paul M. Knopp, Christof Paar

Hardware Reverse Engineering

ASICs



Decapsulation



Delayering



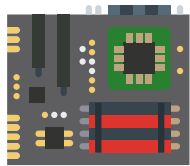
Imaging



Image Processing



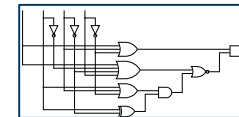
FPGAs



Bitstream Extraction



Bitstream Conversion

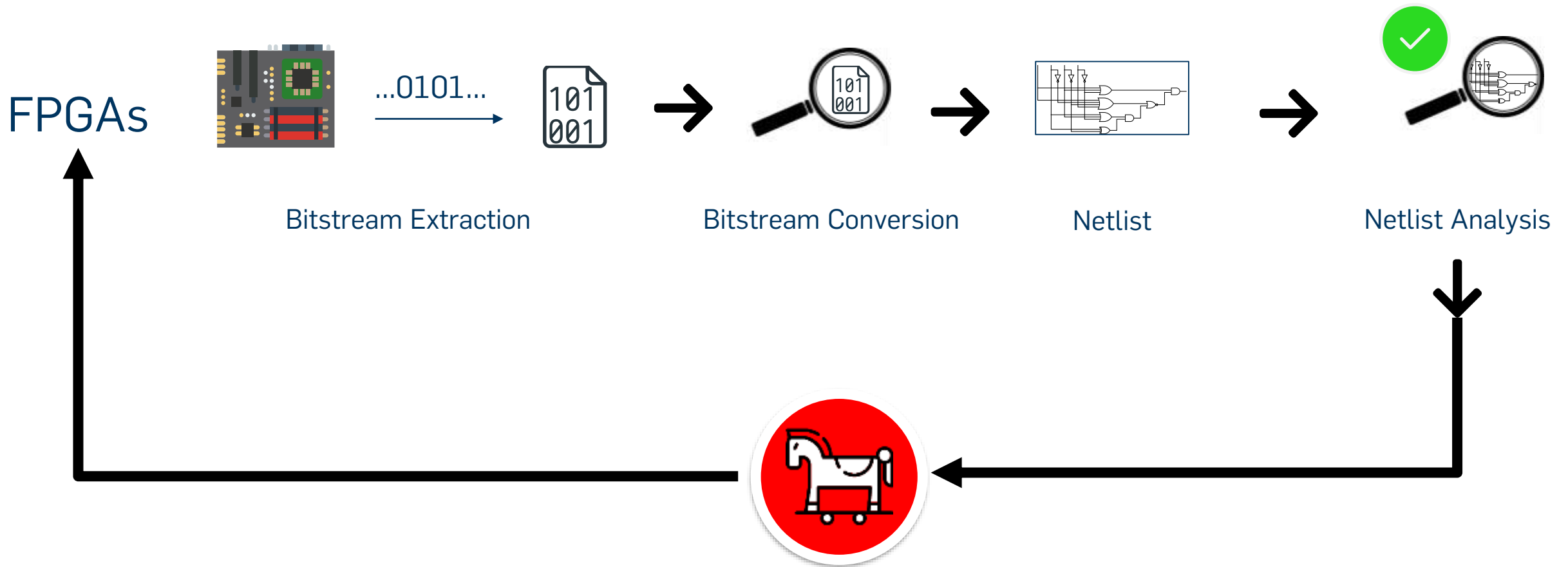


Netlist



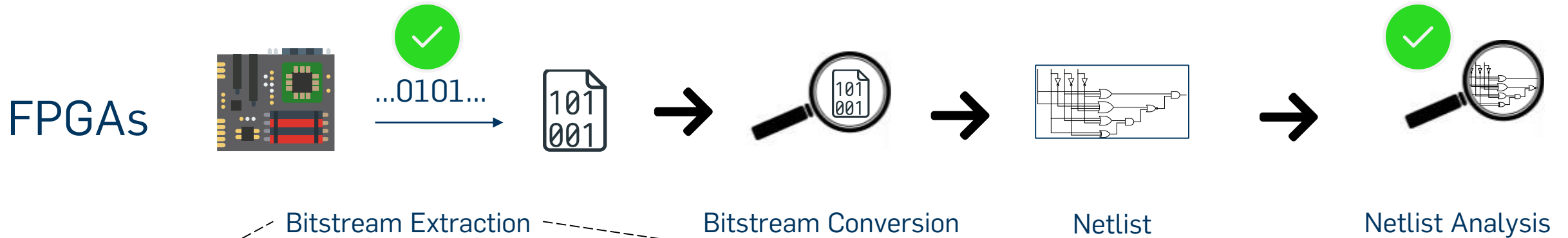
Netlist Analysis

Hardware Reverse Engineering

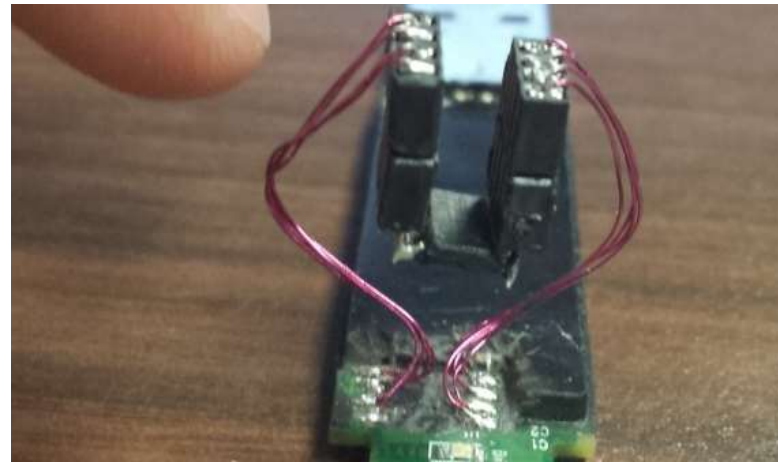


Hardware Trojan Insertion

Hardware Reverse Engineering



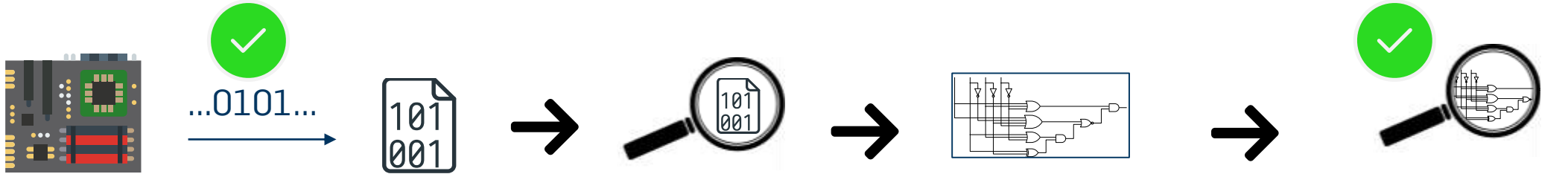
Databus Wiretapping



Flash Tampering

Hardware Reverse Engineering

FPGAs

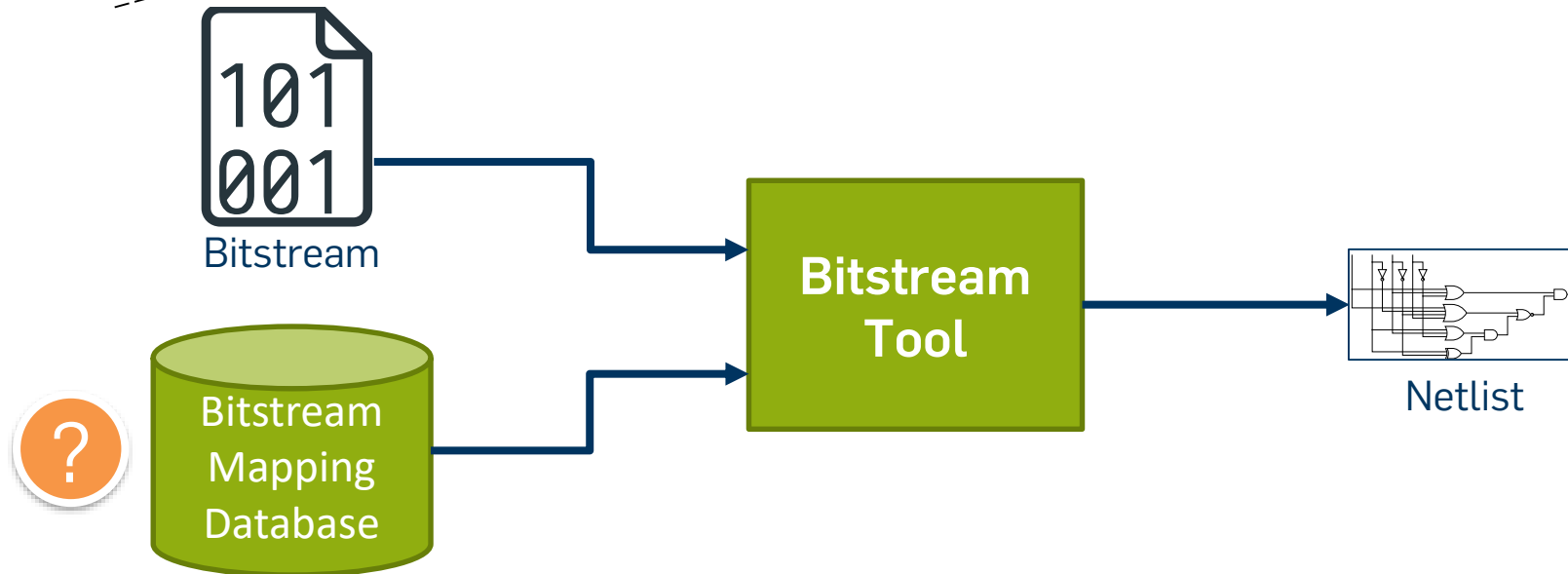


Bitstream Extraction

Bitstream Conversion

Netlist

Netlist Analysis



Database Generation

Bitstream File Format Reverse Engineering

Xilinx Spartan-6

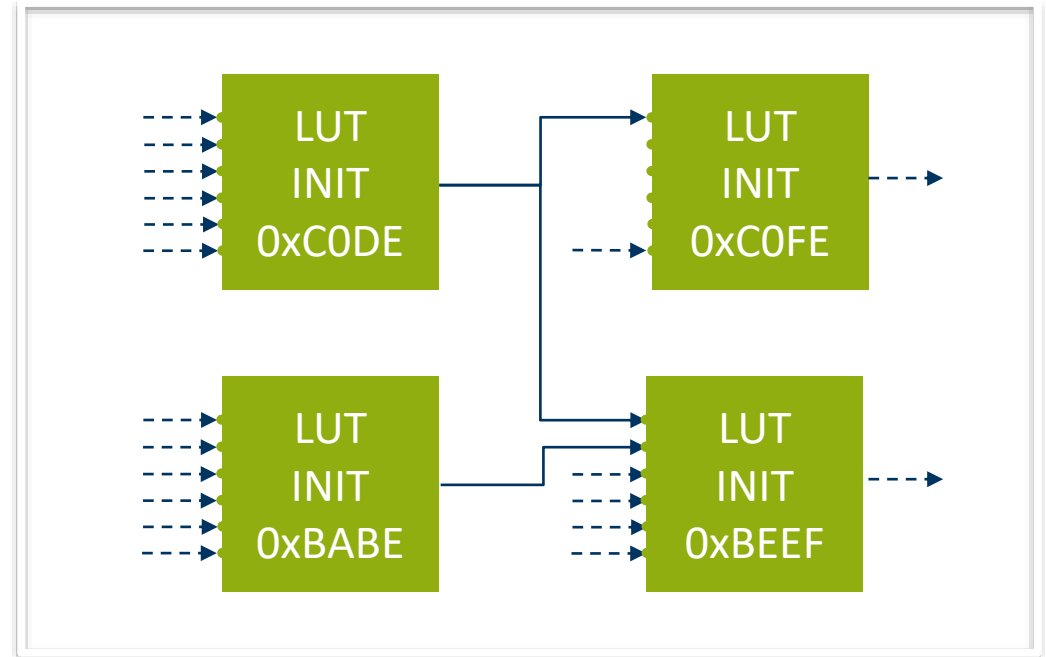
Bitstream

Encoding after Synthesis

Placed&Routed Netlist

00000000	00	09	0F	F0	0F	F0	0F	F0	0F	F0	00	00	01	61	00	27
00000010	33	2D	74	6F	70	5F	41	64	64	53	69	67	6E	61	6C	73
00000020	2E	6E	63	64	3B	55	73	65	72	49	44	3D	30	78	46	46
00000030	46	46	46	46	46	46	00	62	00	0D	36	73	6C	78	31	36
00000040	63	73	67	33	32	34	00	63	00	0B	32	30	31	38	2F	30
00000050	33	2F	30	39	00	64	00	09	31	39	3A	35	38	3A	32	37
00000060	00	65	00	07	15	44	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000070	FF	FF	FF	FF	FF	FF	AA	99	55	66	30	A1	00	07	20	00
00000080	31	A1	04	30	31	41	3D	00	31	61	09	EE	31	C2	04	00
00000090	20	93	30	E1	00	CF	30	C1	00	81	20	00	20	00	20	00
000000a0	20	00	20	00	20	00	20	00	20	00	20	00	20	00	20	00
000000b0	20	00	20	00	20	00	20	00	20	00	20	00	33	81	3C	C8
000000c0	31	81	08	81	34	21	00	00	32	01	00	1F	31	E1	FF	FF
000000d0	33	21	00	05	33	41	00	04	33	01	01	00	32	61	00	00
000000e0	32	81	00	00	32	A1	00	00	32	C1	00	00	32	E1	00	00
000000f0	33	A1	1B	E2	33	C2	00	00	00	00	20	00	20	00	30	22
00000100	00	00	00	00	30	A1	00	01	50	60	00	03	8A	11	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

=

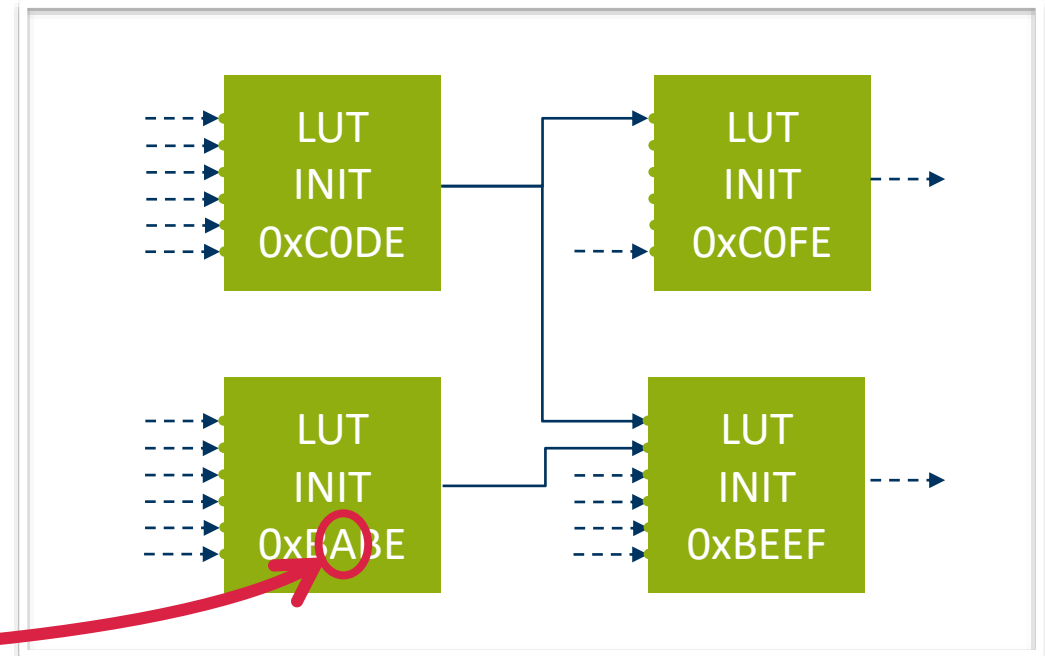


Conversion/Decoding

Bitstream

00000000	00	09	0F	F0	0F	F0	0F	F0	0F	F0	00	00	01	61	00	27
00000010	33	2D	74	6F	70	5F	41	64	64	53	69	67	6E	61	6C	73
00000020	2E	6E	63	64	3B	55	73	65	72	49	44	3D	30	78	46	46
00000030	46	46	46	46	46	46	00	62	00	0D	36	73	6C	78	31	36
00000040	63	73	67	33	32	34	00	63	00	0B	32	30	31	38	2F	30
00000050	33	2F	30	39	00	64	00	09	31	39	3A	35	38	3A	32	37
00000060	00	65	00	07	15	44	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000070	FF	FF	FF	FF	FF	FF	AA	99	55	66	30	A1	00	07	20	00
00000080	31	A1	04	30	31	41	3D	00	31	61	09	EE	31	C2	04	00
00000090	20	93	30	E1	00	CF	30	C1	00	81	20	00	20	00	20	00
000000a0	20	00	20	00	20	00	20	00	20	00	20	00	20	00	20	00
000000b0	20	00	20	00	20	00	20	00	20	00	20	00	33	81	3C	C8
000000c0	31	81	08	81	34	21	00	00	32	01	00	1F	31	E1	FF	FF
000000d0	33	21	00	05	33	41	00	04	33	01	01	00	32	61	00	00
000000e0	32	81	00	00	32	A1	00	00	32	C1	00	00	32	E1	00	00
000000f0	33	A1	1B	E2	33	C2	00	00	00	00	20	00	20	00	30	22
00000100	00	00	00	00	30	A1	00	01	50	60	00	03	8A	11	00	00
00000110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

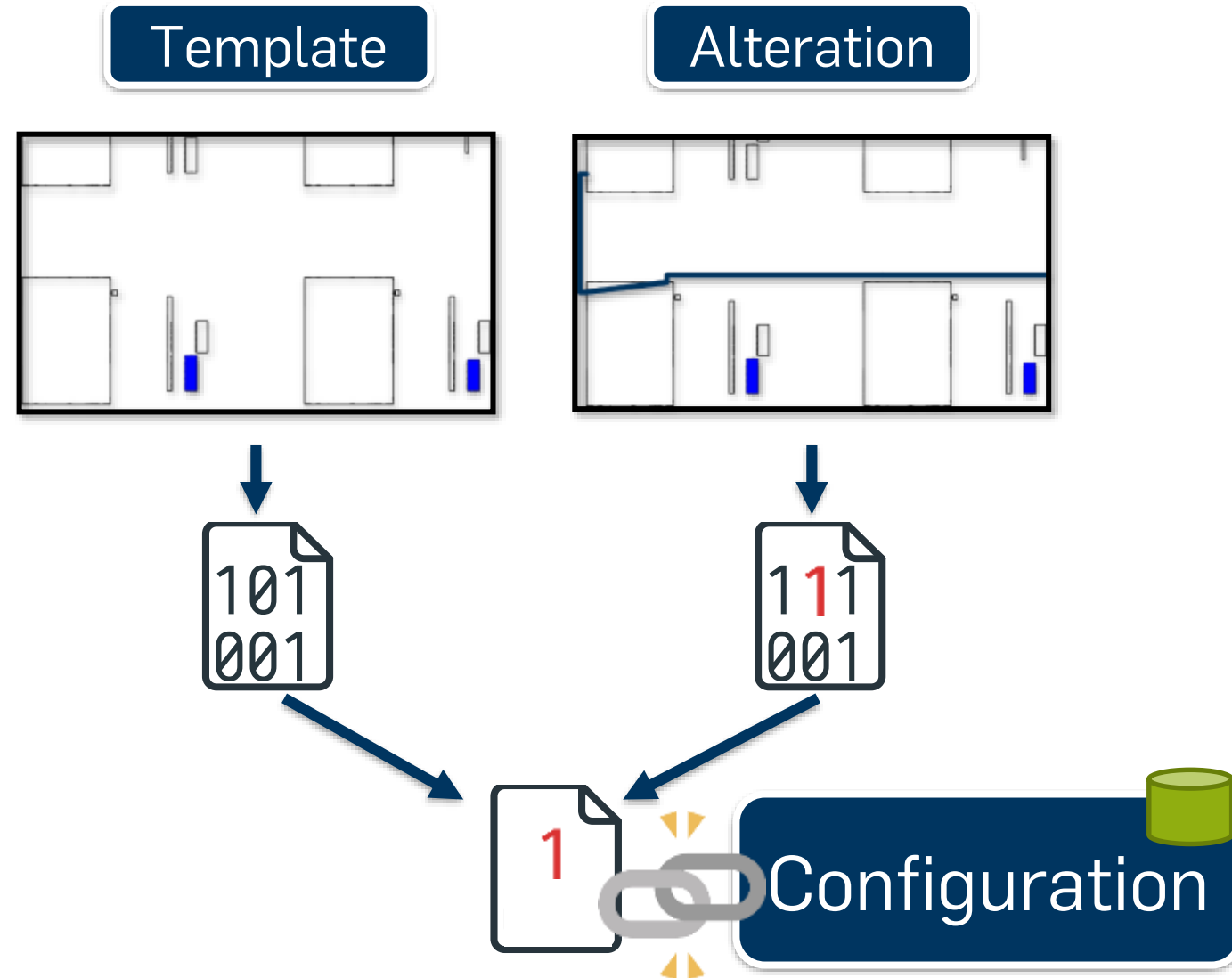
Placed&Routed Netlist



Task: Find the **mapping** between bitstream bits & element configuration

1. Create template netlist
2. Create altered netlist
3. Create bitstreams
4. Correlate the bitstream differences with the design change
5. Retrieve mapping bitstream bit and configuration

REPEAT

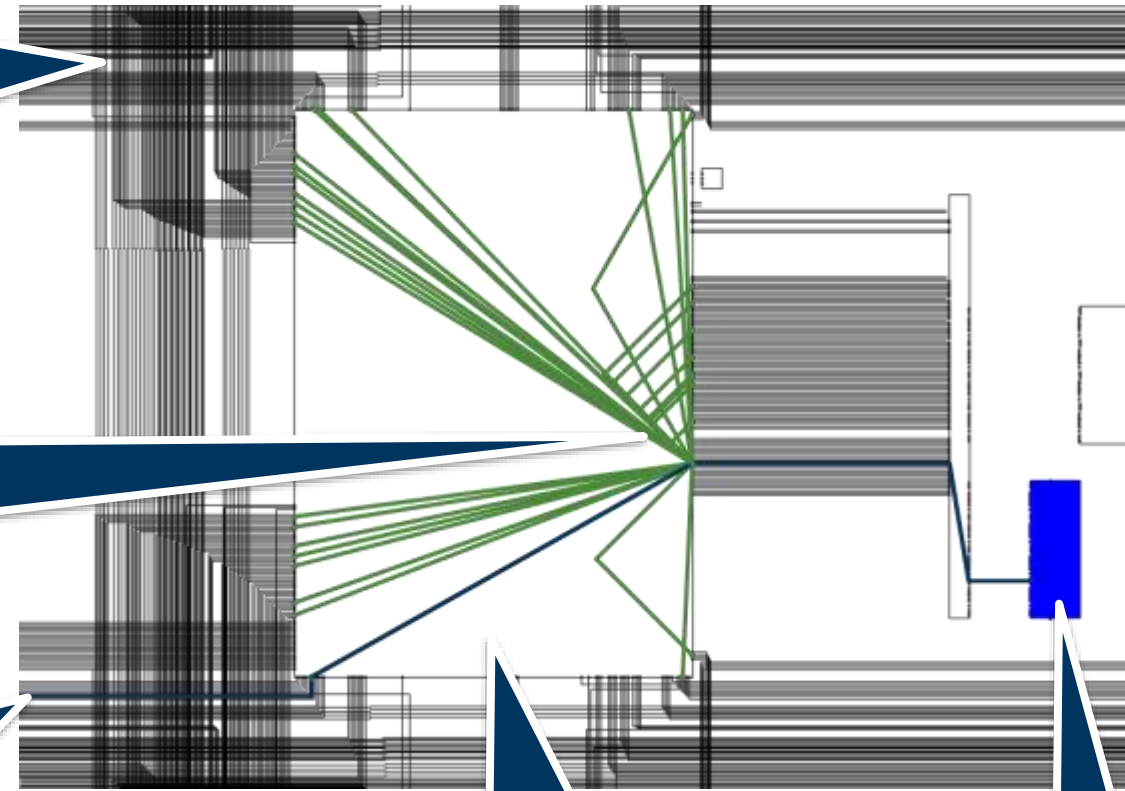


Background: PIPs and Nets

Wire: Physical static connection between two switch matrixes

Programmable Interconnect Point (PIP): Configures which wires get connected

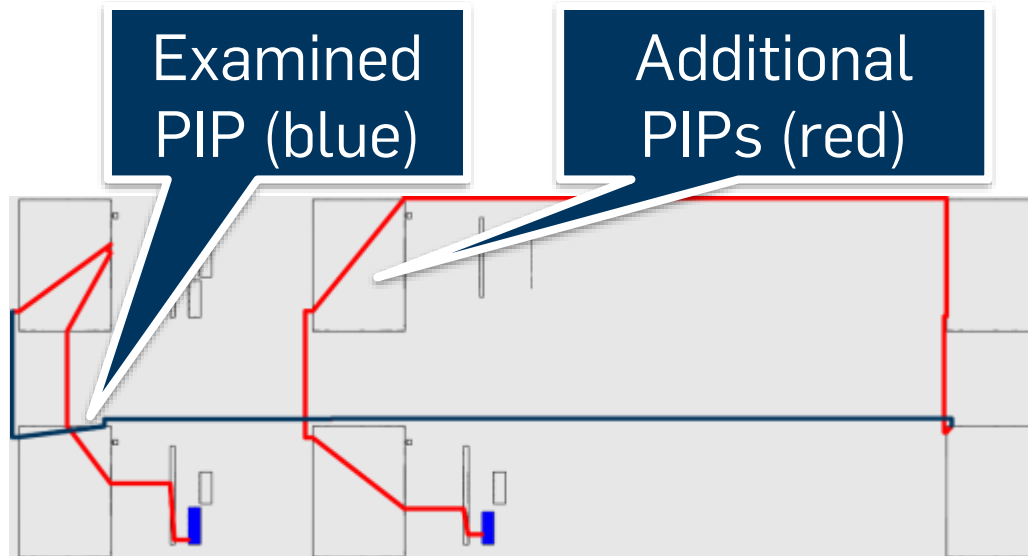
Net: Bunch of PIPs form a net, i.e., multiple pips connect basic elements



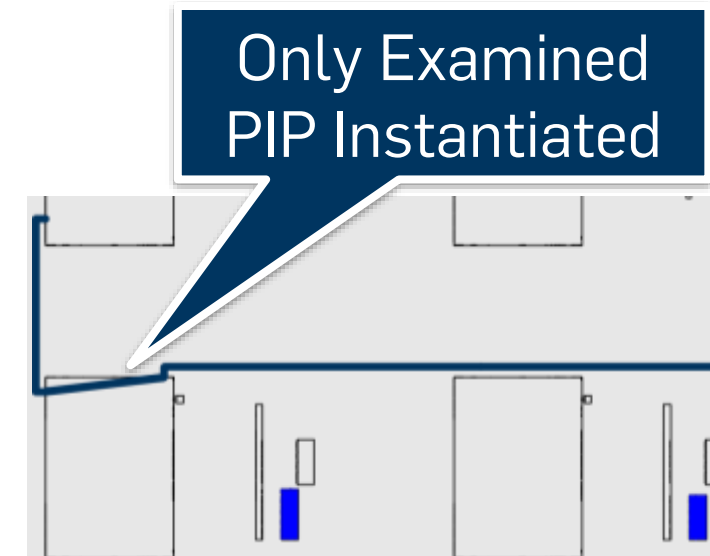
Switch Matrix

SLICE

- Former methods (*Ding et al.*) need fully routed nets
 - Requires multiple helper PIPs
 - Causes additional unwanted changes in bitstream
 - Requires pre/post-processing

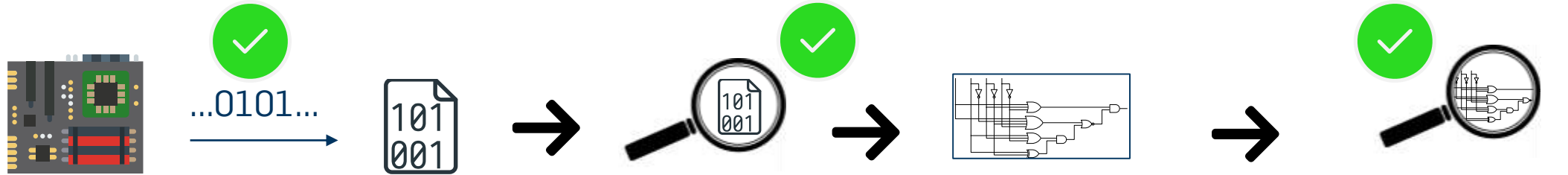


- Our approach: one bitstream with a single PIP
 - **Using force options** of *bitgen* and *xdl*
 - No pre/post-processing
 - Less complex



Hardware Reverse Engineering

FPGAs

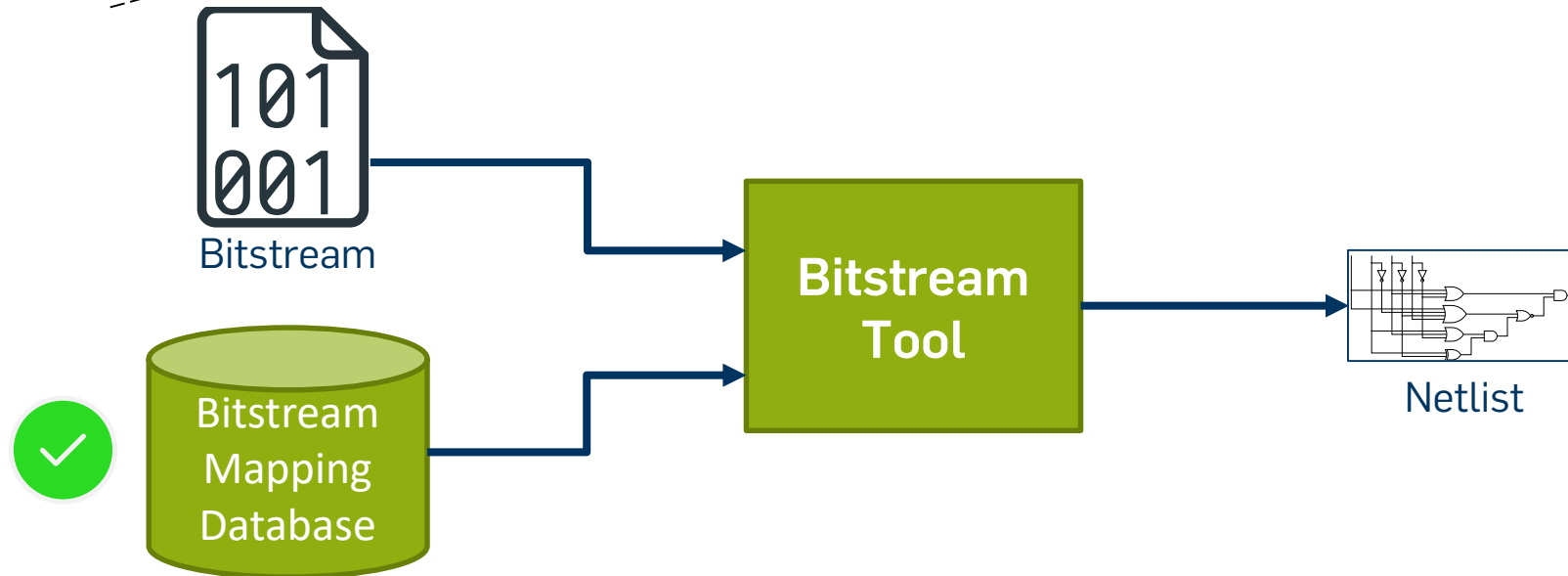


Bitstream Extraction

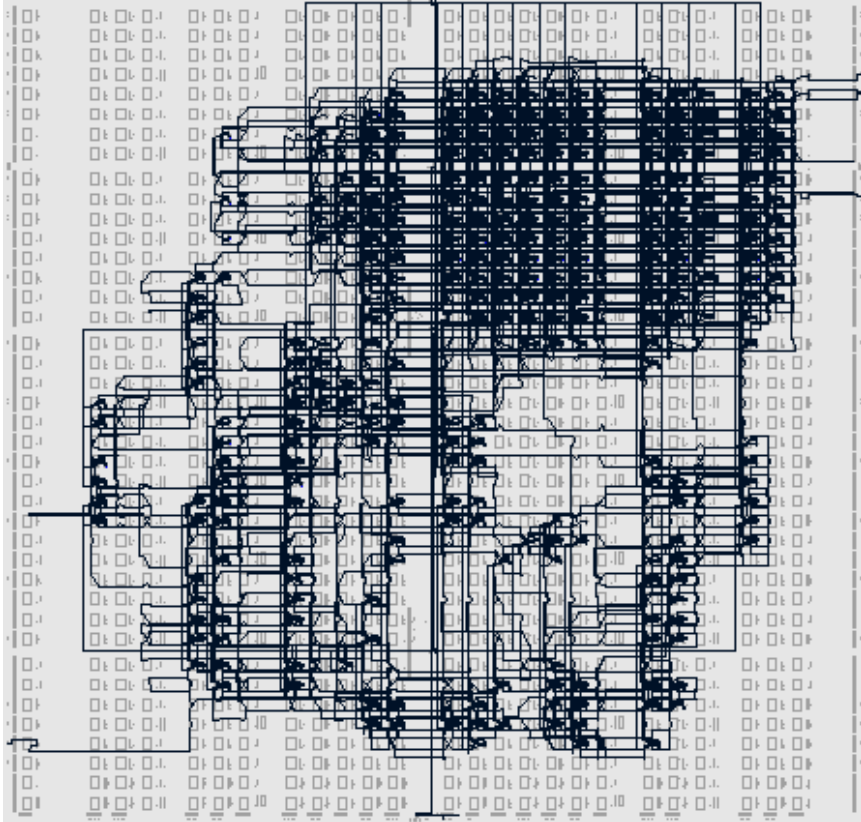
Bitstream Conversion

Netlist

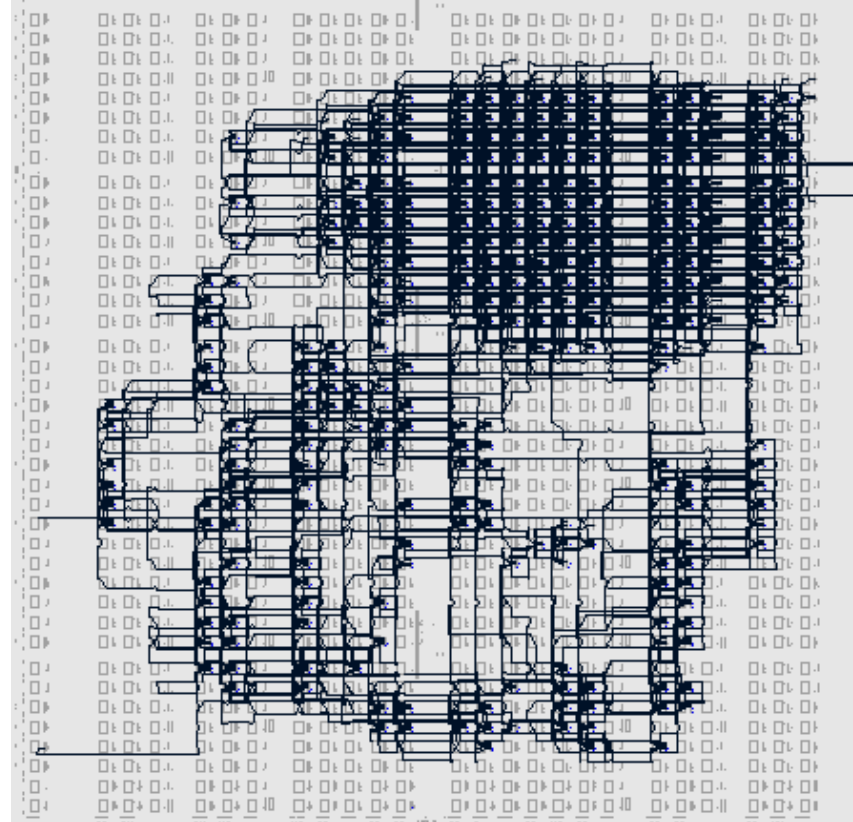
Netlist Analysis



Bitstream Reverse Engineering – Exemplary Design



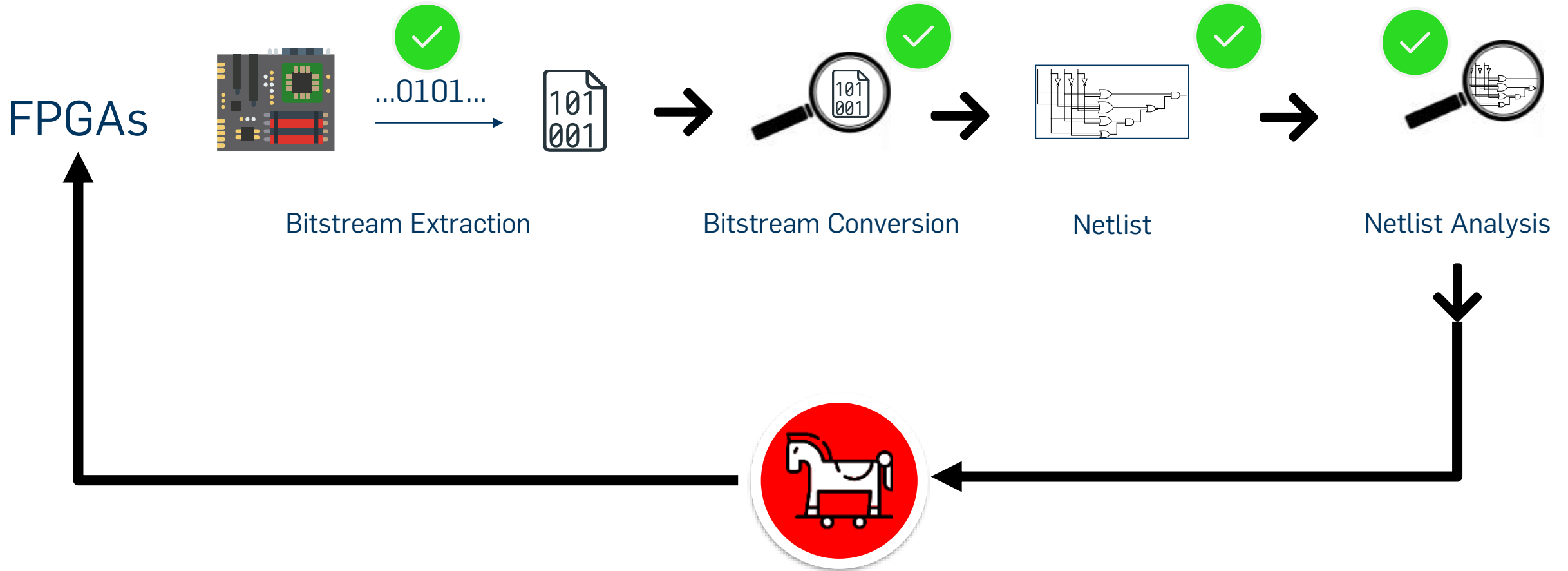
Original Design



Converted Design

- Design almost fully converted
- Still sufficient for HW Trojan insertion
- Fully automated

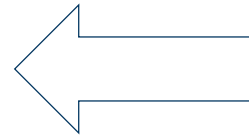
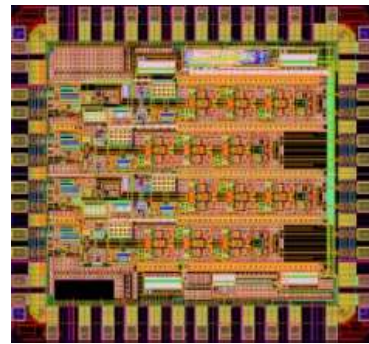
Hardware Reverse Engineering



Hardware Trojan Insertion

Hardware Trojans

Malicious changes or additions to a hardware design that adds or remove functionality, or reduces reliability



Many rather unpleasant “applications” & hard to mitigate in field

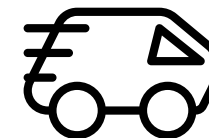


Trojanized USB Drive – Scenario (1)

- Flash drive with AES-encryption on FPGA
- Scenario similar to prior work [1]
 - Trojan insertion successfully verified by means of simulation
- Attacker scenario: Supply chain attack




Factory



Customer



Trojanized USB Drive – System & Insertion

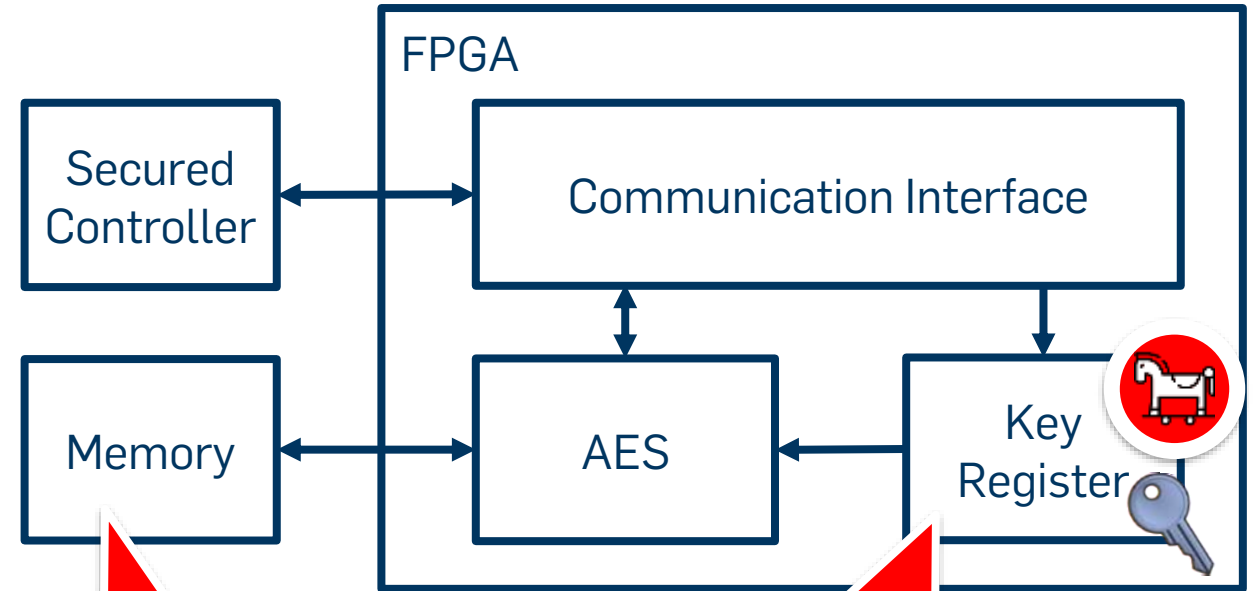
1. AES integrity check
 1. Set the (known) test key 
 2. Test (known) plaintext/ciphertext pairs

2. Derive/provide user key to key register 

3. Process data encryption

Result:

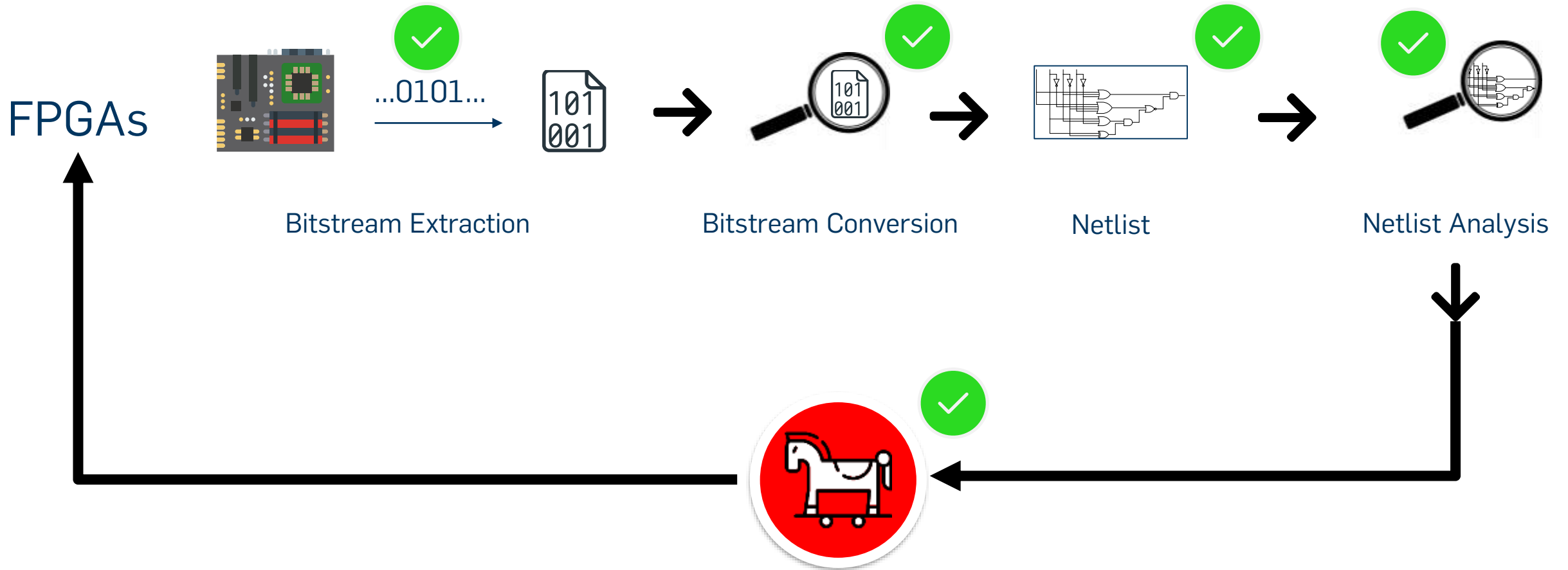
- Attacker can read all encrypted content
- Still passes self-test



Result: Data encrypted w/ known test key

Trojan Idea: Hardwire logic to always output self-test key

Hardware Reverse Engineering



Hardware Trojan Insertion

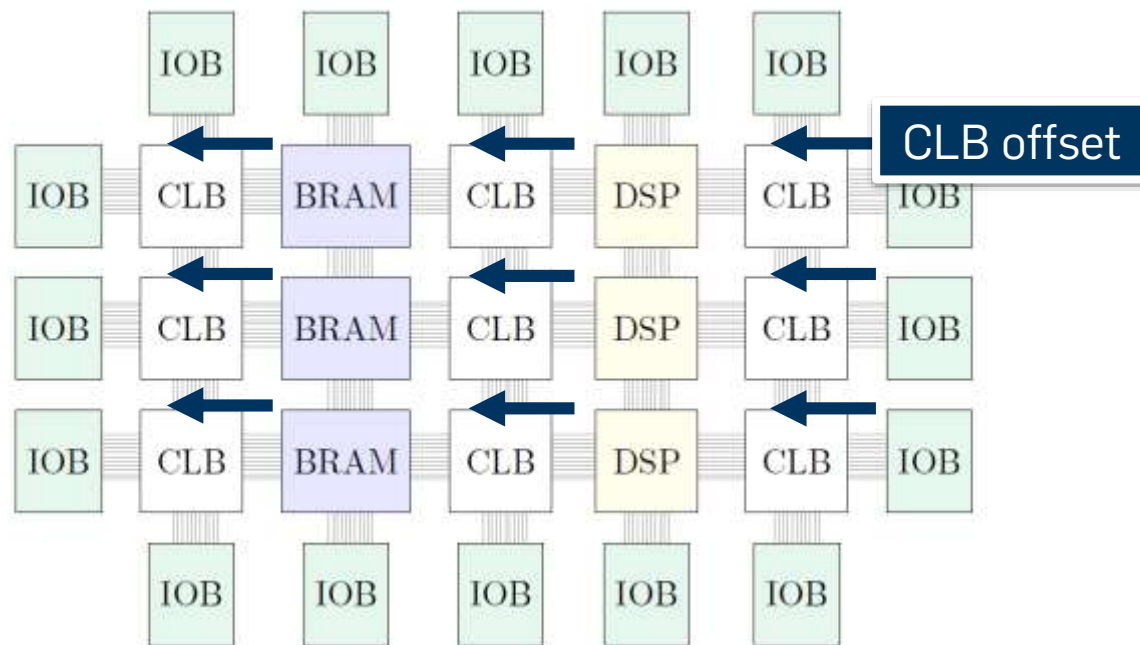
- Bitstream Conversion
 - Accelerated mapping database generation
 - Routing reverse engineering can be simplified
- Hardware Trojan Insertion
 - Partial netlist is sufficient
 - Even more complex attacks are possible
- Tamper-proof external self-test is not enough



**Thank You For Your Attention!
Any Questions?**

Reversing Examples – Compression/Tiling

- If we would save for each bit it's meaning, it would be huge file.
- Compressing by using the FPGA's architecture
- FPGA uses an array of logic blocks
=> Working with offsets



- Reverse the whole CLB
- Reverse only the first bit of each CLB in the Grid
- Position in bitstream is then: $\langle \text{offset CLB} \rangle + \langle \text{pos in CLB} \rangle$
- Saves: space and time (~99%)

Device Family	Introduced	Bitstream Enc./Auth.	Known Vulnerability	No integrity/authenticity
Xilinx Spartan 3	2005	not supported	–	✓
Xilinx Spartan 6	2009	AES-256/HMAC	[MS16]	✓
Xilinx Virtex II	2001	3-DES/no	[MBKP11]	✓
Xilinx Virtex 4	2005	AES-256/no	[MKP12]	✓
Xilinx Virtex 5	2006	AES-256/no	[MKP12])	✓
Xilinx Virtex 6	2009	AES-256/HMAC	[MS16])	✓
Xilinx 7 series	2010	AES-256/HMAC	[MS16])	✓
Xilinx UltraSCALE	2014	AES-256 GCM/RSA-2048	no research reports so far	unclear
Xilinx UltraSCALE ⁺	2015	AES-256 GCM/RSA-4096	no research reports so far	unclear
Altera Stratix II	2004	AES-128	[MOPS13]	✓
Altera Stratix III	2006	AES-256	[SMOP14]	✓
Altera Stratix IV	2008	AES-256	no research reports so far	unclear
Altera Stratix V	2010	AES-256	no research reports so far	unclear
Altera Stratix 10	2013	AES-256, SHA-256, PUF	no research reports so far	unclear
Microsemi FPGAs	–	AES-256, SHA-256	no research reports so far	unclear

No integrity

Table 1.1: List of Xilinx FPGA families and Altera FPGA devices, which are vulnerable to side-channel attacks. No side-channel attacks for the UltraSCALE and UltraSCALE⁺ family have been reported so far. Note that the Xilinx 7 series includes the Kintex, Artix, and Virtex families