

FPGAs and Cryptography

NTNU – 12. November 2024

Håkon Jacobsen
Thales Norway


Thales Group

Over **77000***
employees 

* Excluding ground transportation

€1 bn 
Self-funded R&D**

** Does not include externally financed R&D

68 
Countries
Global presence

Sales in 2022 
€17,6bn



Thales Norway AS

A nighttime panoramic view of Oslo, Norway, showing the city's lights reflecting on the water. The city is built on a hillside, and the lights are a mix of warm yellow and cool blue. The water in the foreground is dark, with reflections of the city lights. The sky is dark with some light clouds.

High-tech company implementing high assurance cryptography and communication solutions

Independent unit within the Thales Group, subject to the Norwegian Security Act

200+ employees (Oslo & Trondheim)
NOK 1B yearly revenue

About me

- **Cryptographer at Thales Norway**

- Supervise cryptography use and implementations in our projects (and security more broadly)
- Implement both software and hardware (VHDL)
 - › Mostly cryptography

- **Teach TEK4500 – Introduction to Cryptography at UiO**

- <https://www.uio.no/studier/emner/matnat/its/TEK4500/h24/>

Why FPGAs?

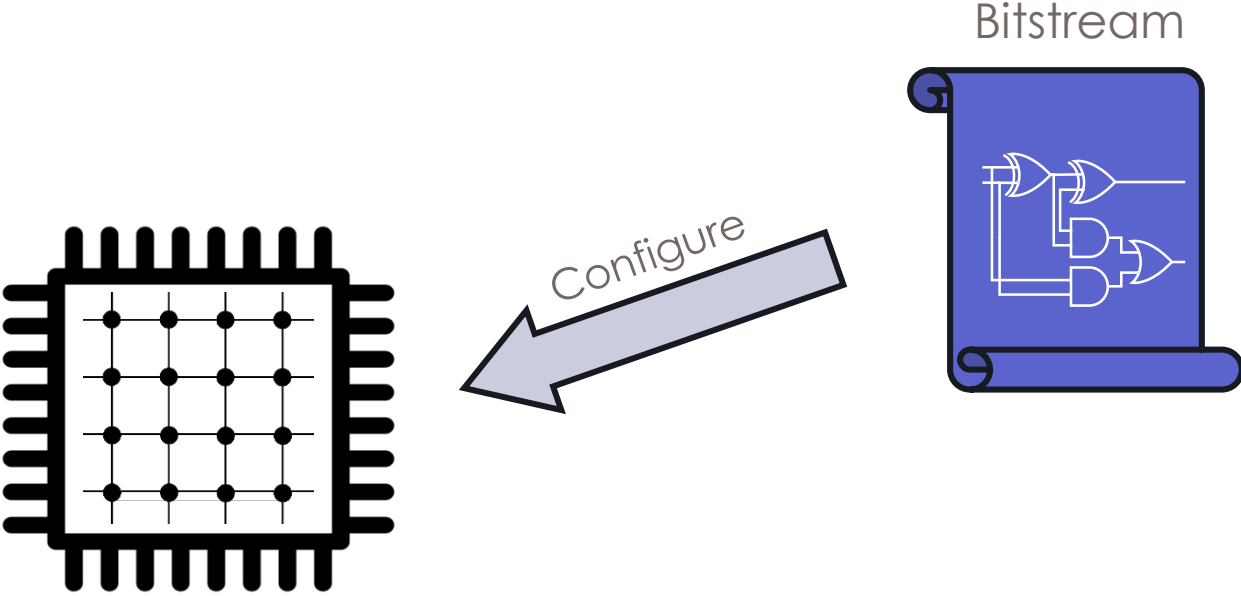
- High performance
- Low latency
- Flexibility
- Precise control over:
 - running time
 - resource usage
- ASIC prototyping
- Regulatory requirements



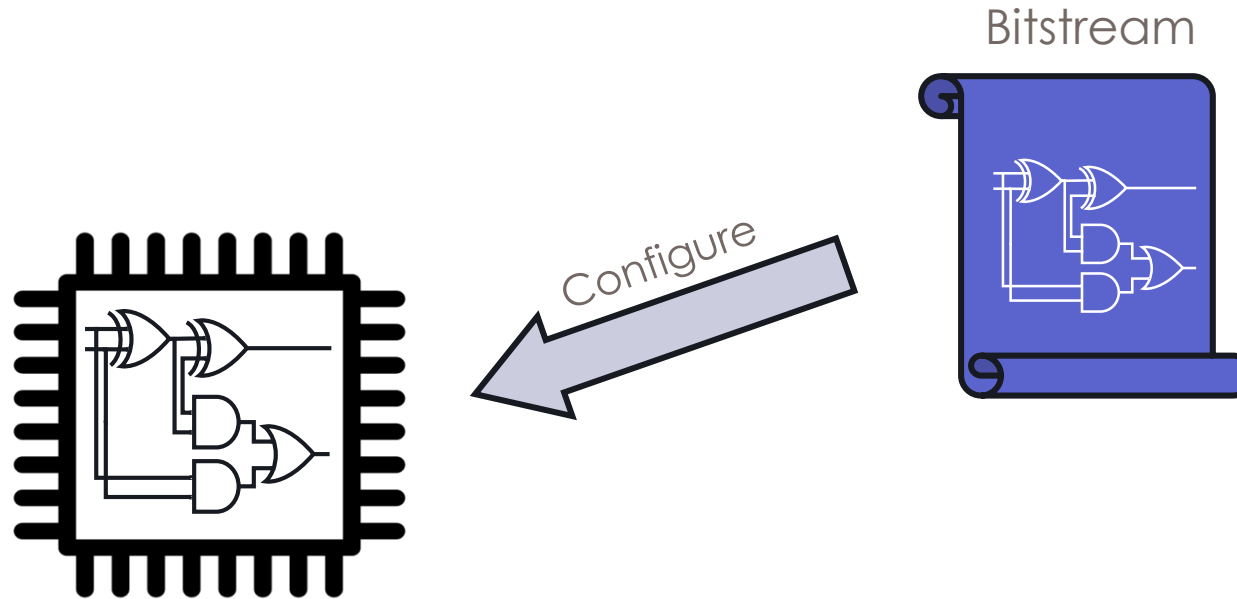
FPGA applications:

- Aerospace and avionics
- Digital signal processors
- Defense and military
- Medical devices
- General hardware accelerators (e.g. cryptography)

FPGA – Field Programmable Gate Array



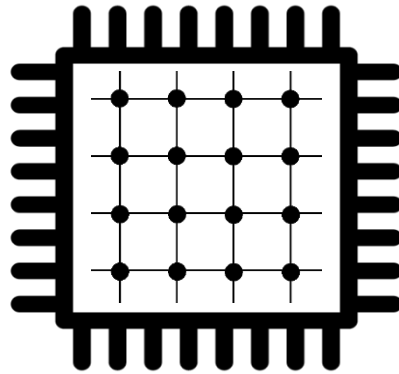
FPGA – Field Programmable Gate Array



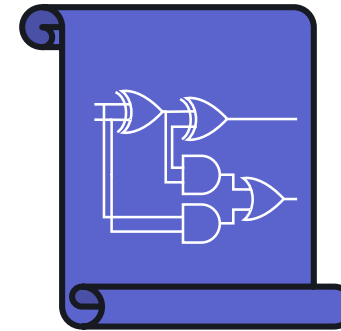
Implementations

- SRAM (volatile, re-programmable)
- Flash (non-volatile, re-programmable)
- Antifuse (non-volatile, one-time programmable)

FPGA – Field Programmable Gate Array



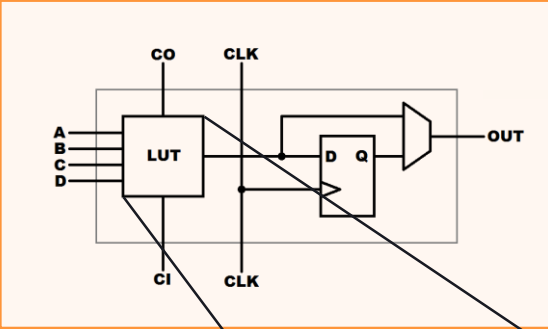
Bitstream



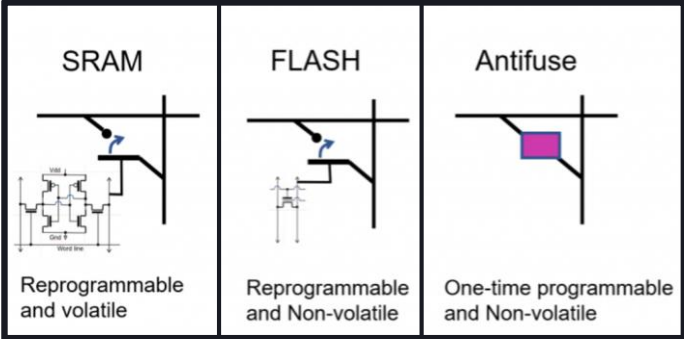
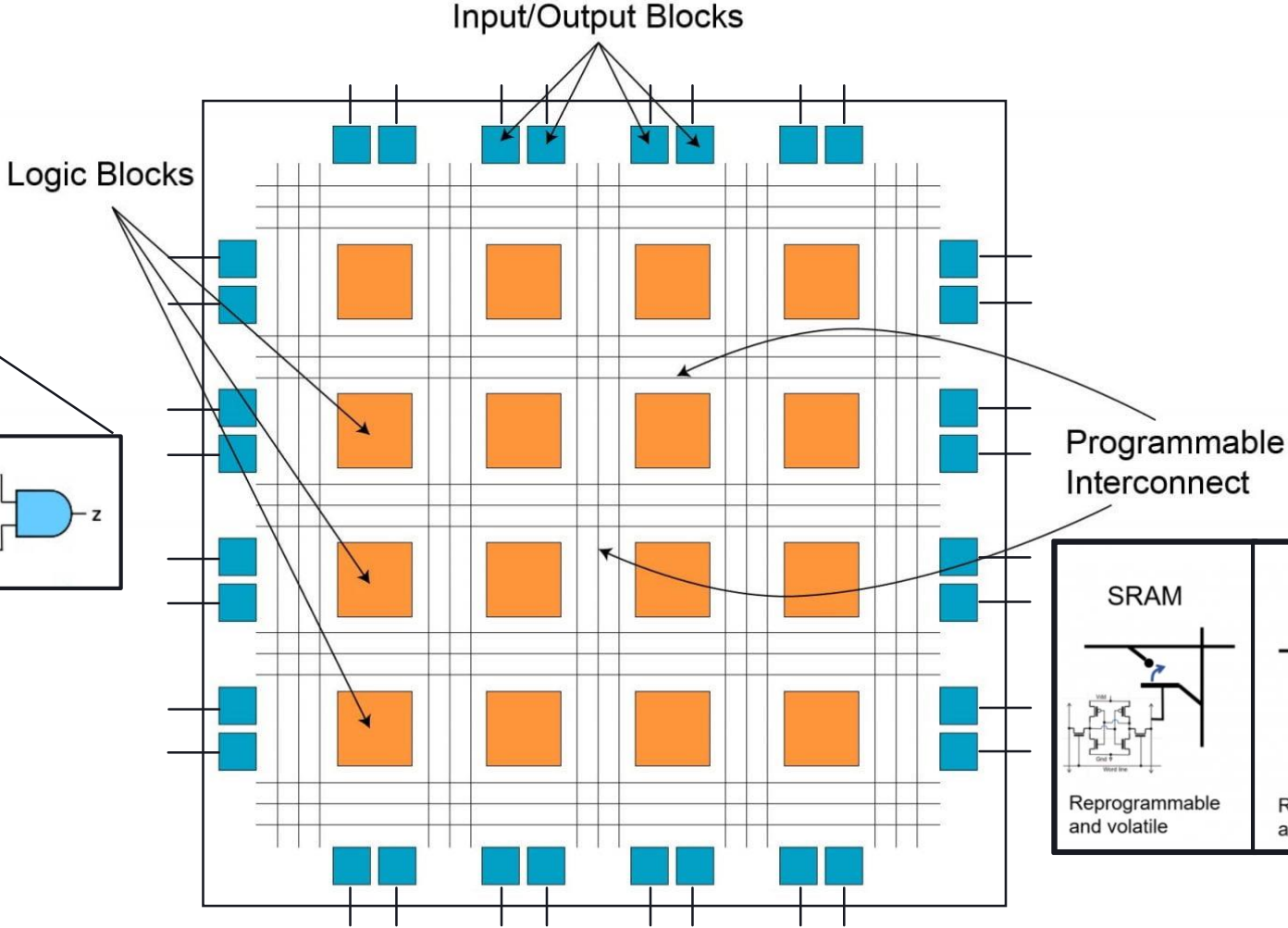
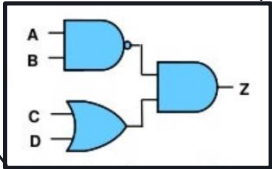
Implementations

- SRAM (volatile, re-programmable)
- Flash (non-volatile, re-programmable)
- Antifuse (non-volatile, one-time programmable)

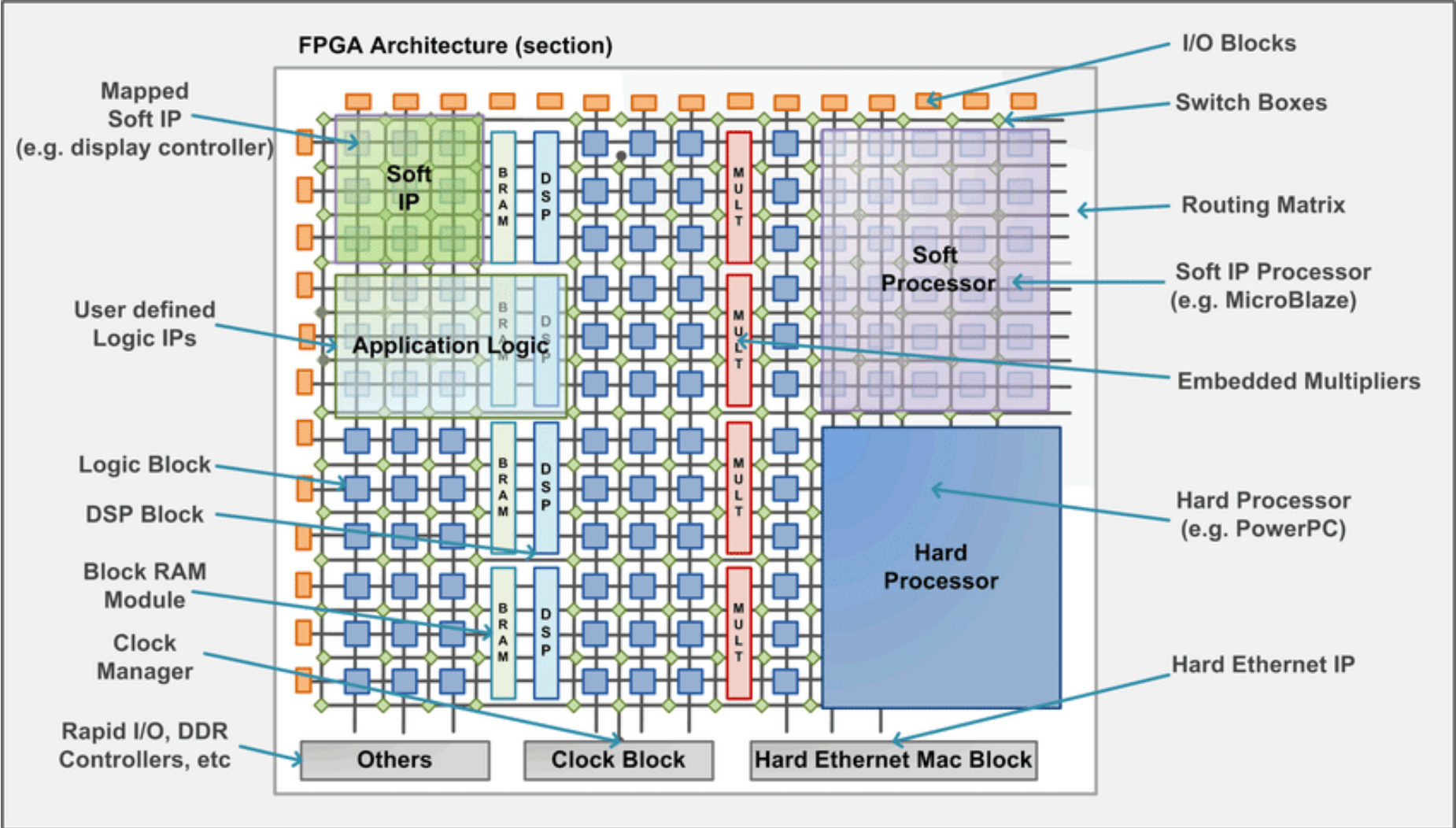
FPGA components



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



Modern FPGAs



Spartan-7 FPGA Feature Summary

Table 2: Spartan-7 FPGA Feature Summary by Device

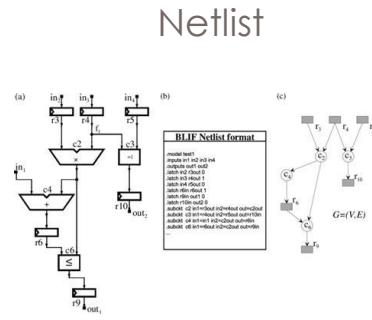
Device	Logic Cells	CLB		DSP Slices ⁽²⁾	Block RAM Blocks ⁽³⁾			CMTs ⁽⁴⁾	PCIe	GT	XADC Blocks	Total I/O Banks ⁽⁵⁾	Max User I/O
		Slices ⁽¹⁾	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)						
XC7S6	6,000	938	70	10	10	5	180	2	0	0	0	2	100
XC7S15	12,800	2,000	150	20	20	10	360	2	0	0	0	2	100
XC7S25	23,360	3,650	313	80	90	45	1,620	3	0	0	1	3	150
XC7S50	52,160	8,150	600	120	150	75	2,700	5	0	0	1	5	250
XC7S75	76,800	12,000	832	140	180	90	3,240	8	0	0	1	8	400
XC7S100	102,400	16,000	1,100	160	240	120	4,320	8	0	0	1	8	400

Notes:

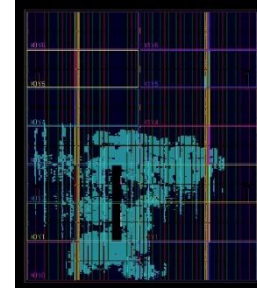
- Each 7 series FPGA slice contains four LUTs and eight flip-flops; only some slices can use their LUTs as distributed RAM or SRLs.
- Each DSP slice contains a pre-adder, a 25 x 18 multiplier, an adder, and an accumulator.
- Block RAMs are fundamentally 36 Kb in size; each block can also be used as two independent 18 Kb blocks.
- Each CMT contains one MMCM and one PLL.
- Does not include configuration Bank 0.

FPGA design flow

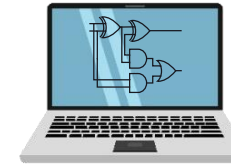
VHDL or Verilog



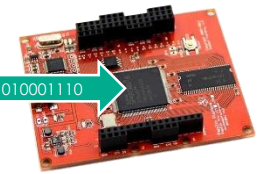
Place & Route



Bitstream



100010110111010001110

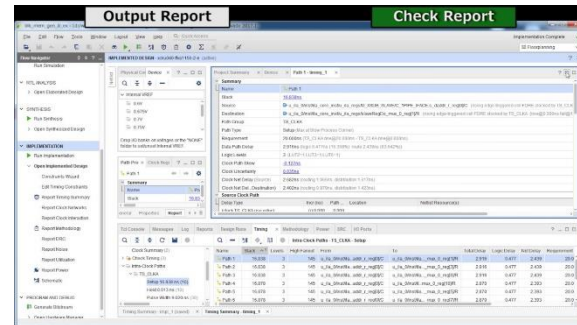
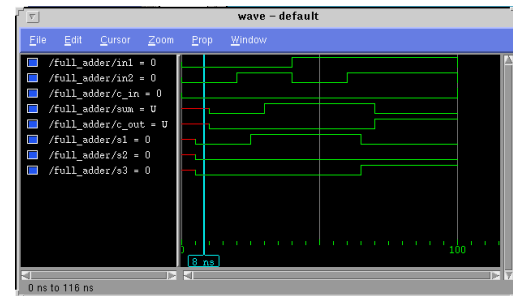


HDL coding

Synthesis

Implementation

Device programming

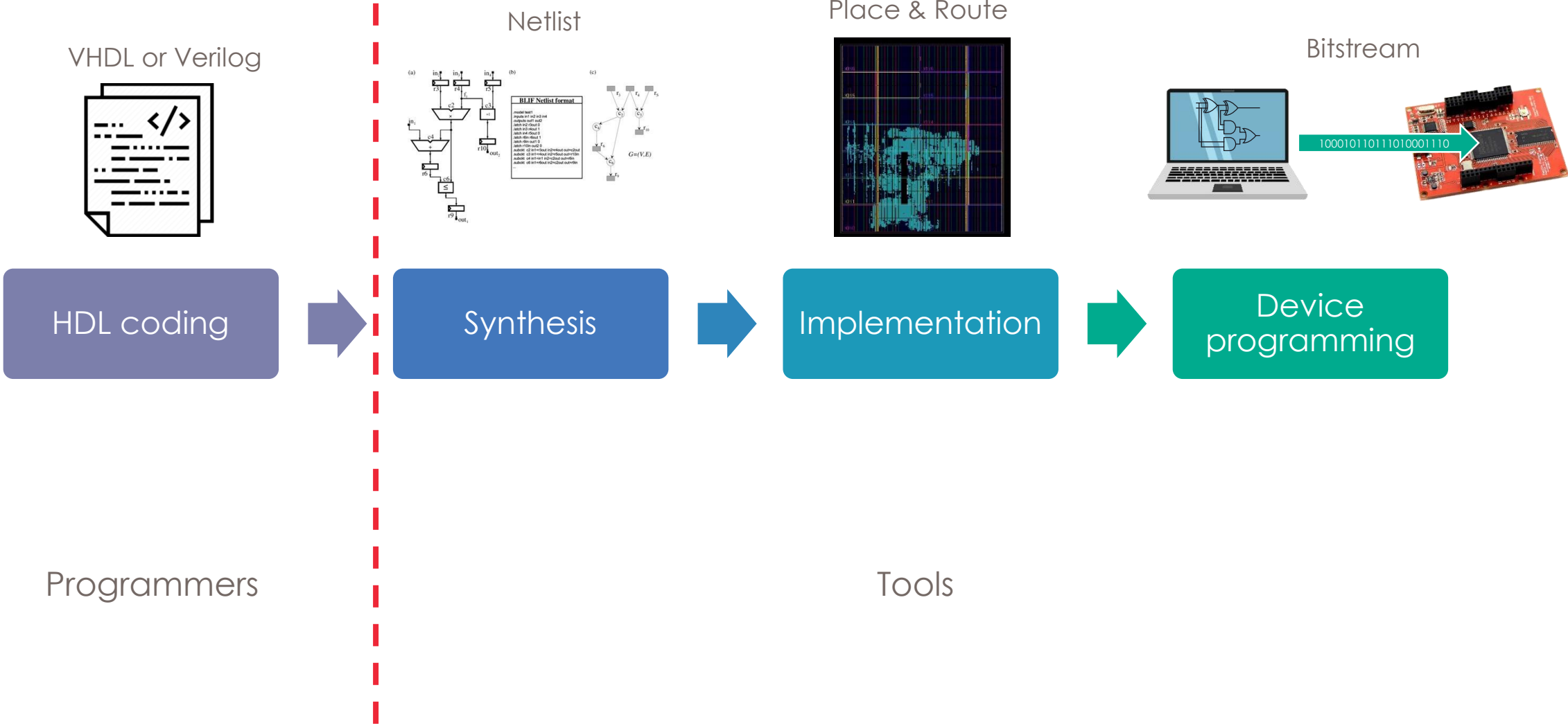


Simulation

Timing analysis

OPEN

FPGA design flow



Synthesis

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

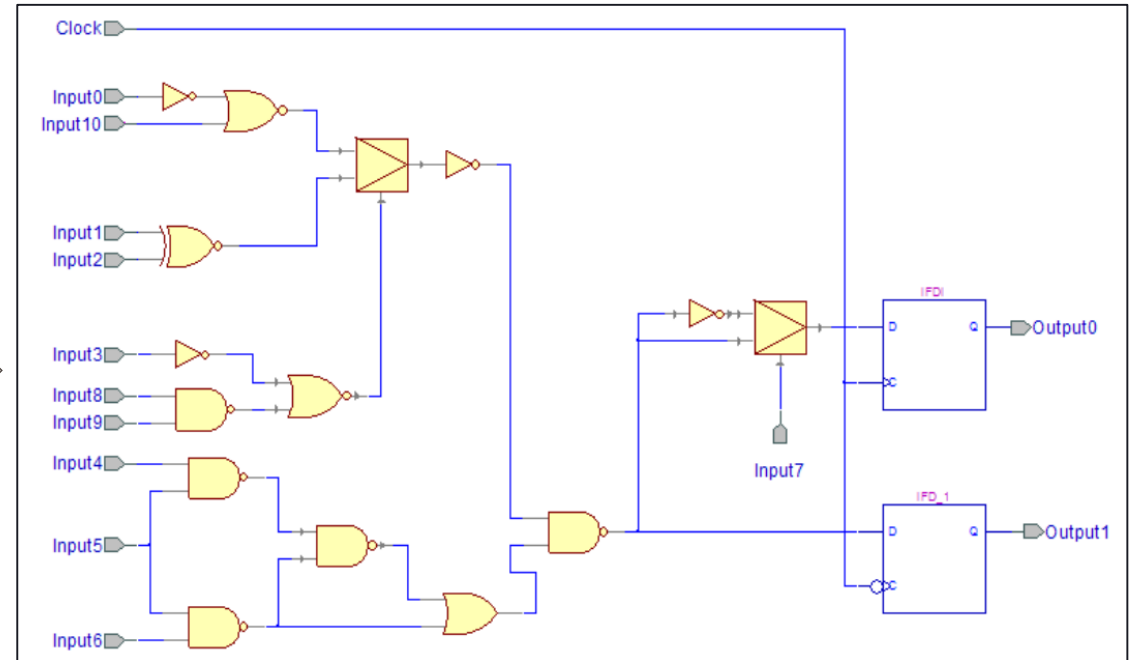
entity Test_Counter_VHDL is
  Port ( Clk_xxxHz : in std_logic;
        Step_Clk : in std_logic;
        Select_Clk : in std_logic;
        Clr, Count_Enable : in std_logic;
        Bcd0, Bcd1, Bcd2, Bcd3 : out std_logic_vector(3 downto 0));
end Test_Counter_VHDL;

architecture Behavioral of Test_Counter_VHDL is
  Signal Q: std_logic_vector( 15 downto 0 );
  Signal Clk: std_logic;
begin
  -- 2x1bit multiplexer: Clk_xxx or Step_Clk = [Btn0]
  Clk <= Clk_xxxHz when Select_Clk='1' else
    Step_Clk;

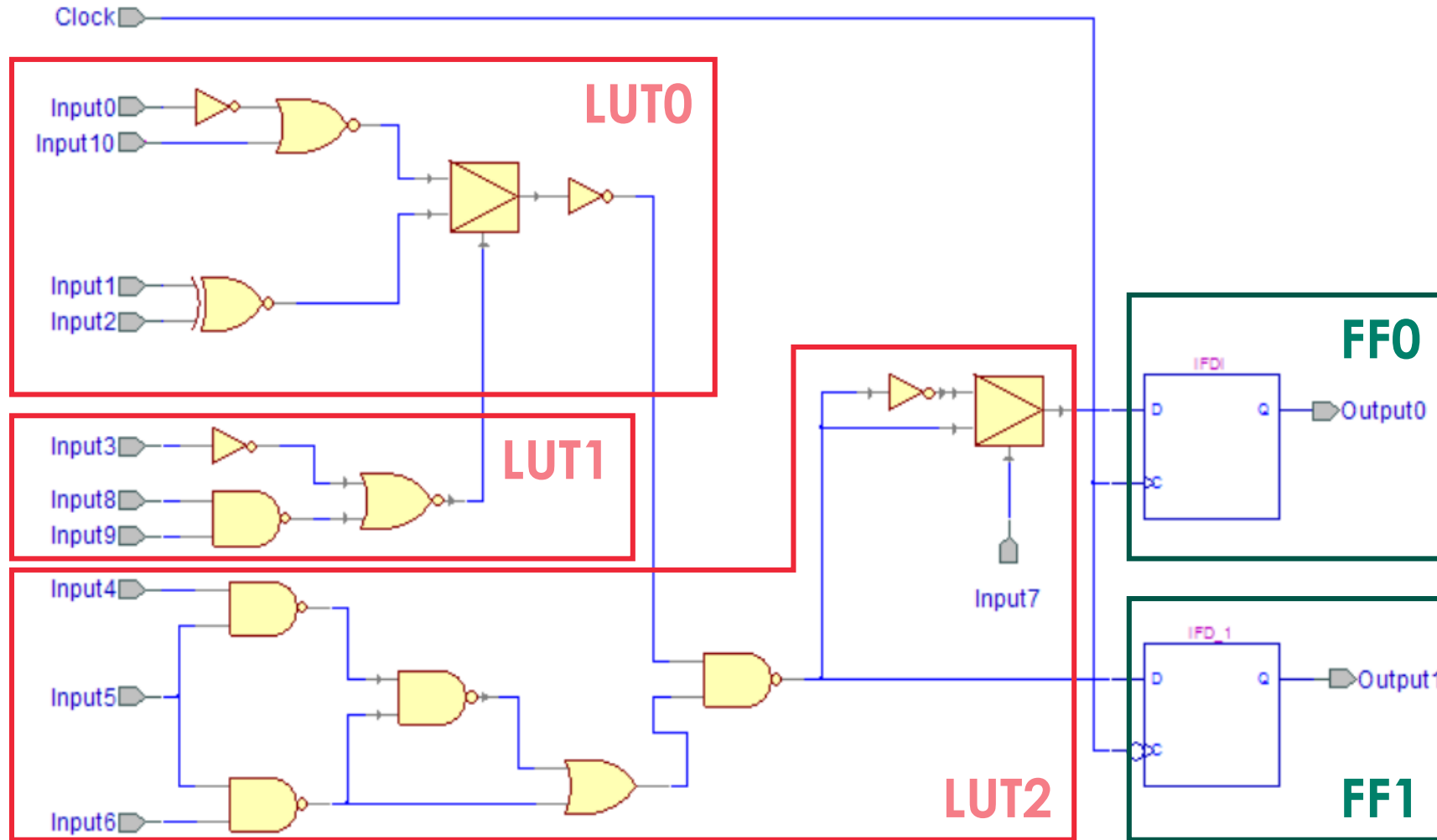
  process( Clk, Clr)
  begin
    if Clr='1' then
      Q <= (others => '0'); -- "0000000000000000"
    elsif rising_edge( Clk) then
      if Count_Enable='1' then
        Q <= Q+1;
      end if;
    end if;
  end process;

  Bcd3 <= Q(15 downto 12);
  Bcd2 <= Q(11 downto 8);
  Bcd1 <= Q( 7 downto 4);
  Bcd0 <= Q( 3 downto 0);

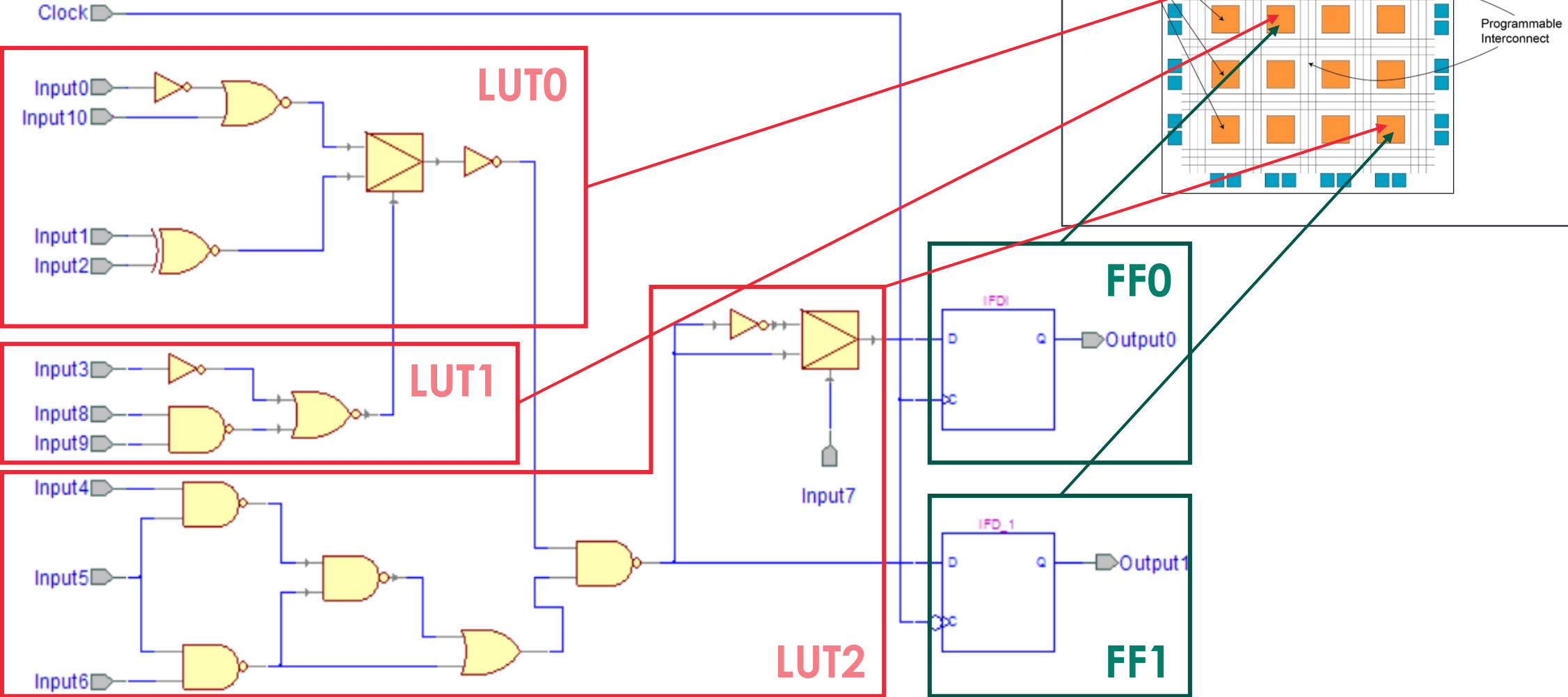
end Behavioral;
```



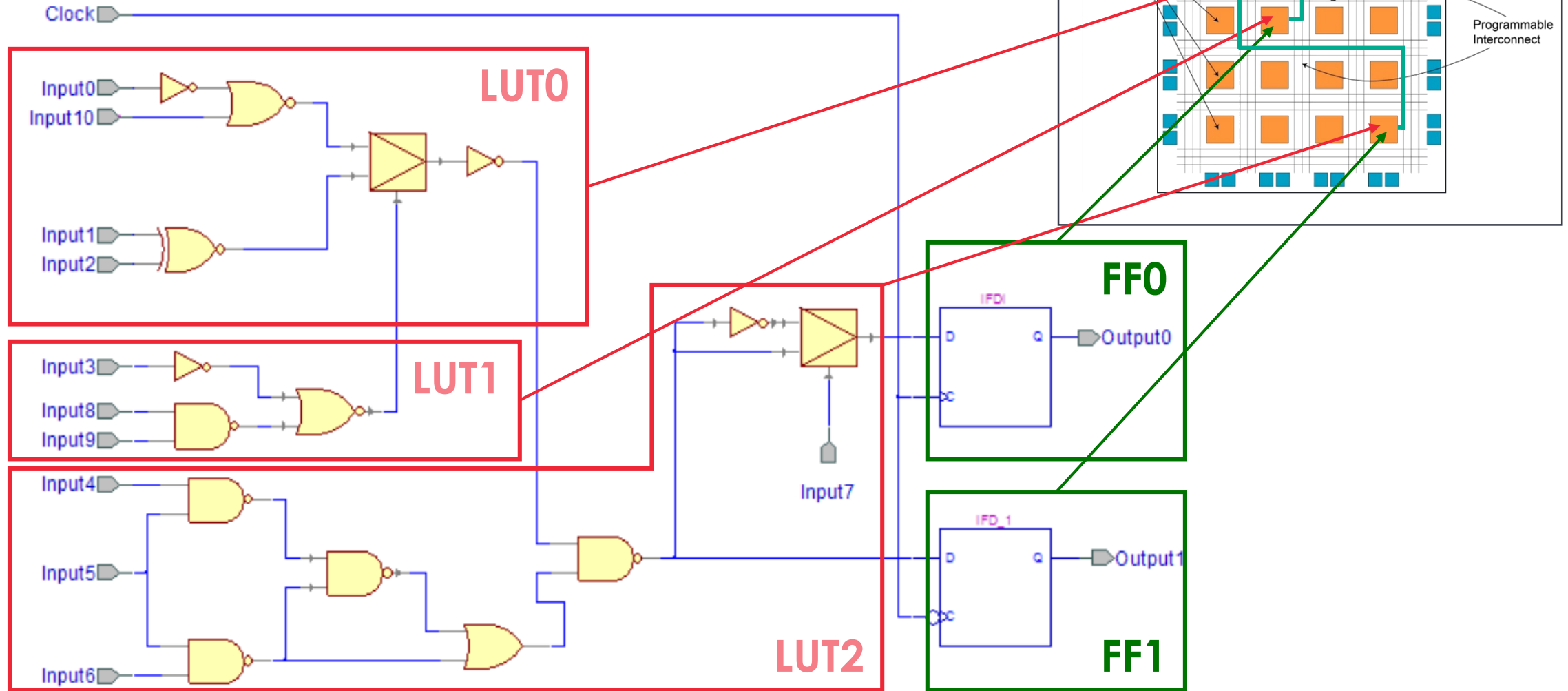
Technology mapping (synthesis)



Placement



Routing



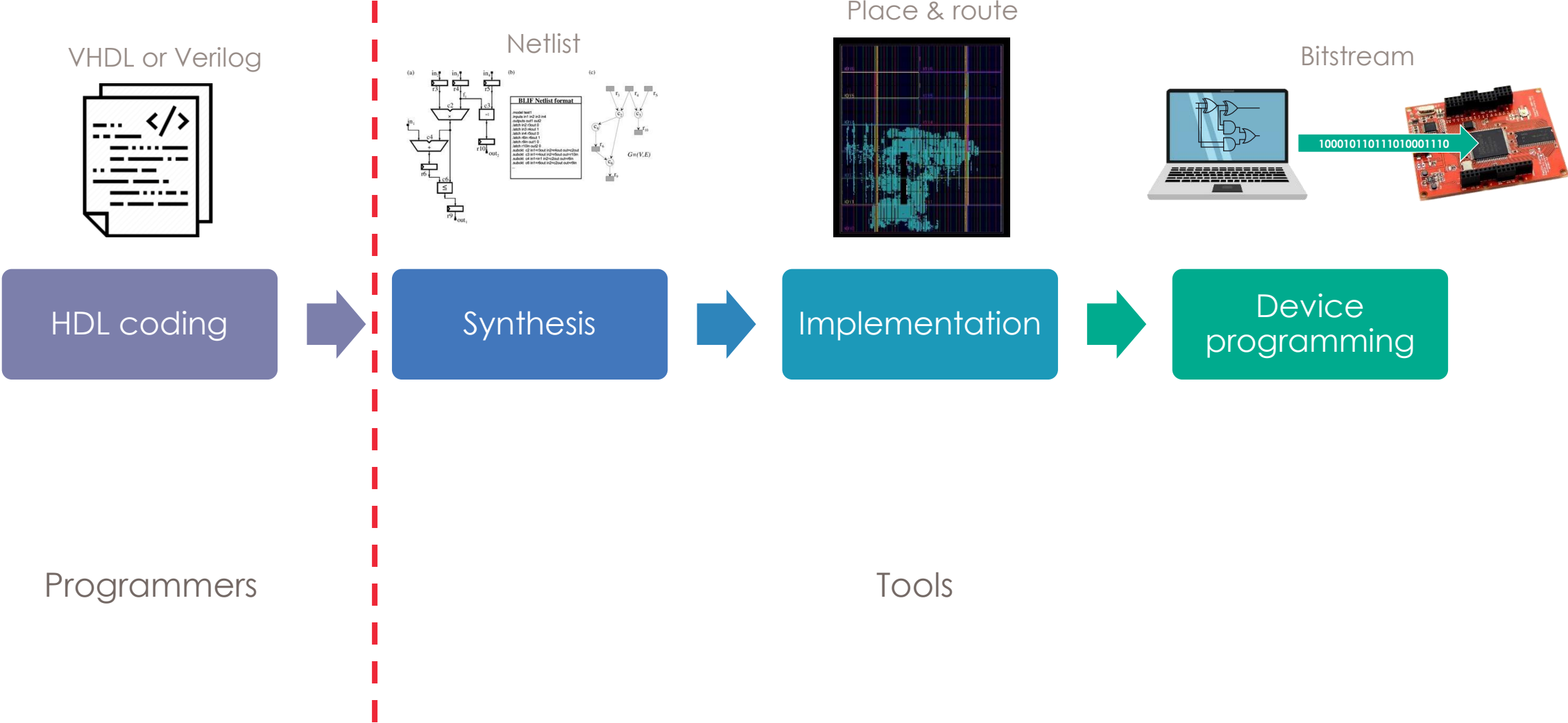
Place & Route

The screenshot displays the Vivado 2017.4 Place & Route interface. The main window shows a timing summary report for a design named 'constr_2'. The report is titled 'Design Timing Summary' and includes the following data:

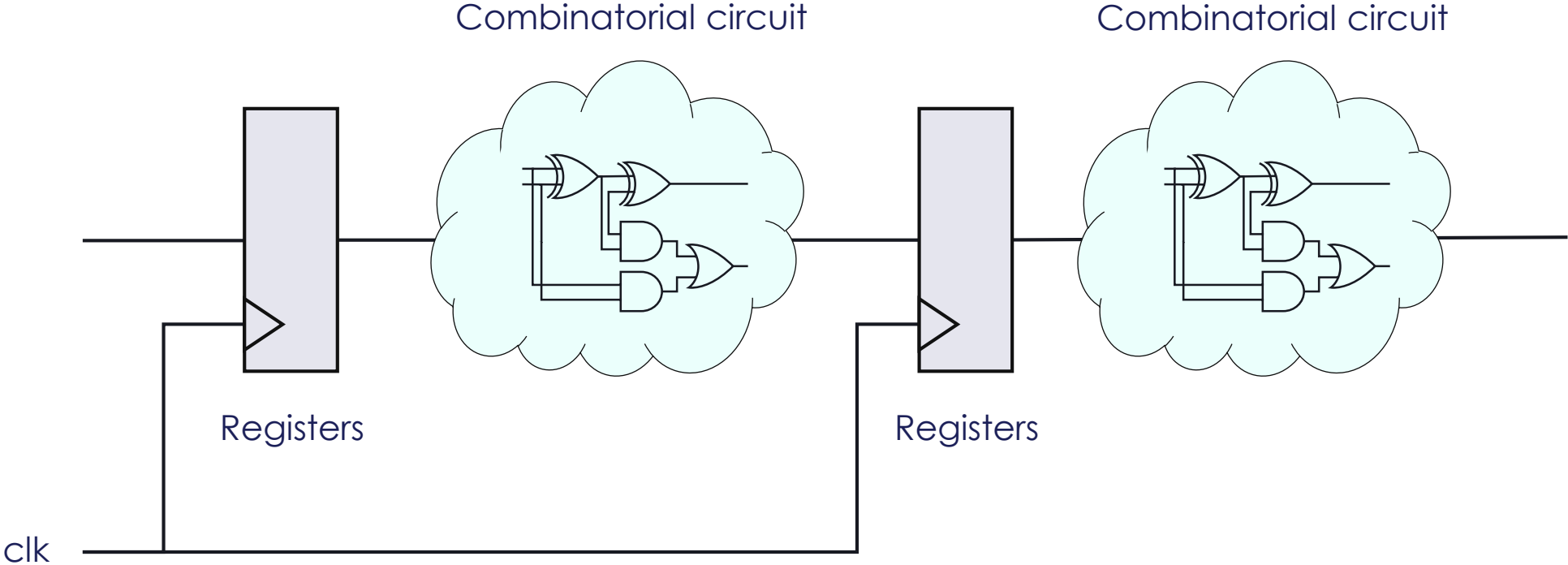
Setup			Hold			Pulse Width		
Worst Negative Slack (WNS):	1.316 ns		Worst Hold Slack (WHS):	0.011 ns		Worst Pulse Width Slack (VFPWS):	3.000 ns	
Total Negative Slack (TNS):	0.000 ns		Total Hold Slack (THS):	0.000 ns		Total Pulse Width Negative Slack (TPWS):	0.000 ns	
Number of Failing Endpoints:	0		Number of Failing Endpoints:	0		Number of Failing Endpoints:	0	
Total Number of Endpoints:	47911		Total Number of Endpoints:	47611		Total Number of Endpoints:	15935	

At the bottom of the report, it states: "All user specified timing constraints are met." The interface also shows a 'Sources' window with a list of LUTs, a 'BEL Properties' window for 'SLICE_X36Y102F8MUX', and a 'Timing' window with a 'Design Timing Summary' tab.

FPGA design flow



FGPA circuits – high level view



VHDL

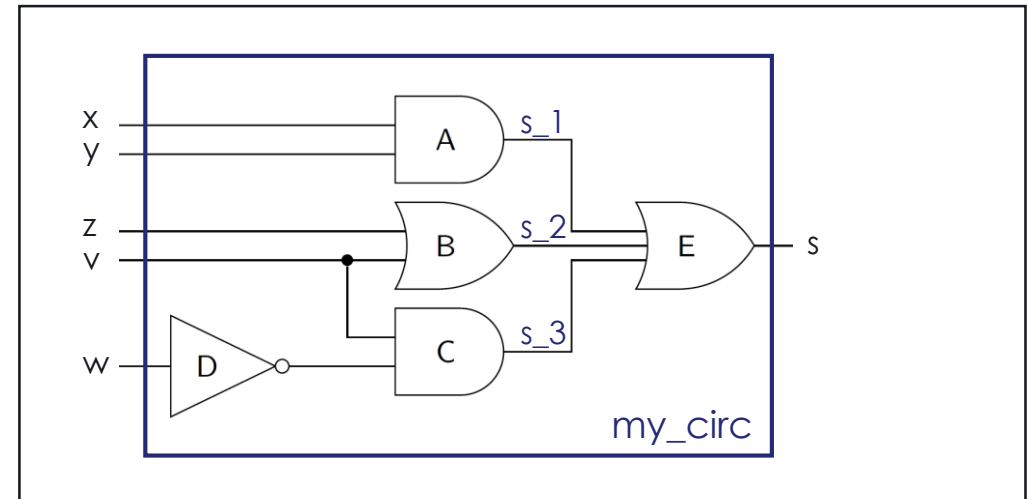
```
library IEEE;
use IEEE.std_logic_1164.all;

entity my_circ is
  port (
    x : in  std_logic;
    y : in  std_logic;
    z : in  std_logic;
    v : in  std_logic;
    w : in  std_logic;
    s : out std_logic
  );
end my_circ;

architecture impl of my_circ is

  signal s_1 : std_logic;
  signal s_2 : std_logic;
  signal s_3 : std_logic;

begin
  s_1 <= x and y;           -- A
  s_2 <= z or v;           -- B
  s_3 <= v and (not w);    -- C + D
  s   <= s_1 or s_2 or s_3; -- E
end impl;
```



VHDL

```
entity my_larger_circ is
  port (
    x_1,x_2,x_3,x_4,x_5 : in std_logic;
    y_1,y_2             : in std_logic;
    z                   : out std_logic
  );
end my_larger_circ;

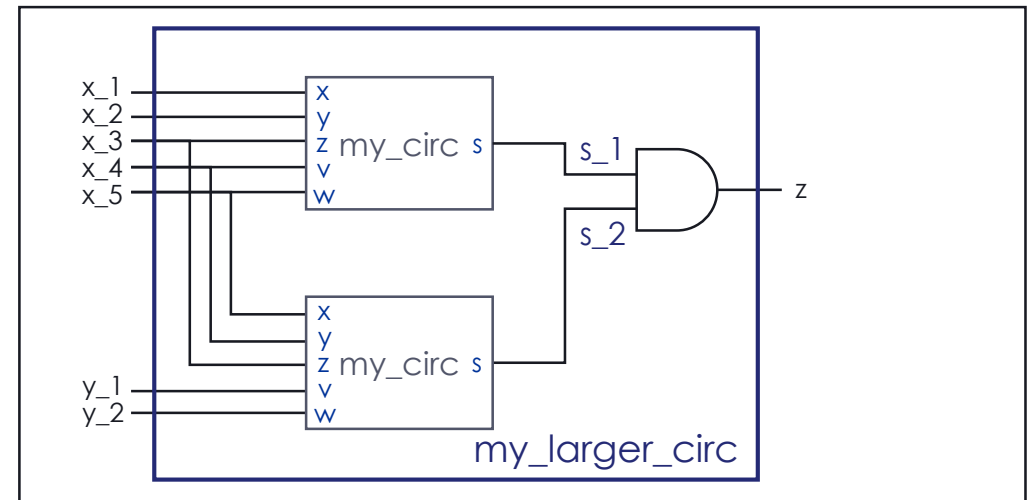
architecture impl of my_larger_circ is
  signal s_1, s_2 : std_logic;
begin

  i_my_circ_1 : my_circ port map (
    x => x_1,
    y => x_2,
    z => x_3,
    w => x_4,
    v => x_5,
    s => s_1);

  i_my_circ_2 : my_circ port map (
    x => x_5,
    y => x_4,
    z => x_3,
    w => y_2,
    v => y_1,
    s => s_2);

  z <= s_1 and s_2;

end impl;
```



VHDL

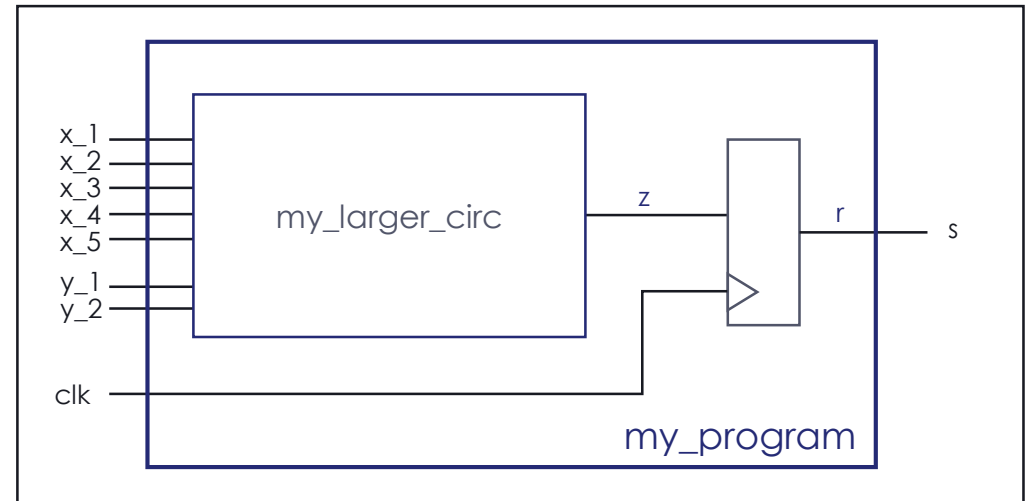
```
entity my_program is
  port (
    clk           : in  std_logic;
    x_1,x_2,x_3,x_4,x_5 : in  std_logic;
    y_1,y_2      : in  std_logic;
    s             : out std_logic
  );
end my_program;

architecture rtl of my_program is
  signal z : std_logic;
  signal r : std_logic;
begin

  i_combinatorial : my_larger_circ
    port map (x_1, x_2, x_3, x_4, x_5, y_1, y_2, z);

  p_store_output : process(clk)
  begin
    if rising_edge(clk) then
      r <= z;
    end if;
  end process;

  s <= r;
end rtl;
```



THALES Key material Equipment Organization 10:36:43 GMT

Key Material From: 2020/09/01 To: 2020/12/01

Field: Search...

Short title	Edition	Reg. n°	Seg.	Transaction type
NMSD 9000 PPTKEK	AAA	11	0	Key Allocate to OU
NMSD 9000 PPTKEK	AAA	11	0	Key Allocate to OU
NMSD 9000 PPTKEK	AAA	11	0	Key Allocate to OU
NMSD 9000 TKEK	AAA	11	0	Key Allocate to OU
NMSD 9000 PPTKEK	AAA	10	0	Key Allocate to OU
NMSD 9000 TKEK	AAA	10	0	Key Allocate to OU
NMSD 9000 PPTKEK	AAA	10	0	Key Allocate to OU
NMSD 9000 PPTKEK	AAA	10	0	Key Allocate to OU

Key Allocate to OU

Date: 2020/12/01

Classification: **SECRET**

Short Title: NMSD 9000 PPTKEK Edition: AAA

Register No: 10 (0xa) Segment: 0

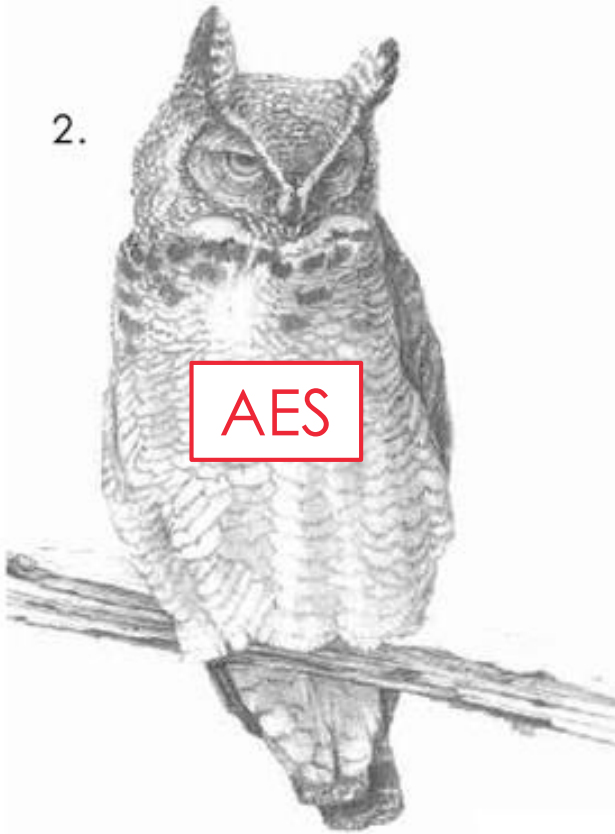
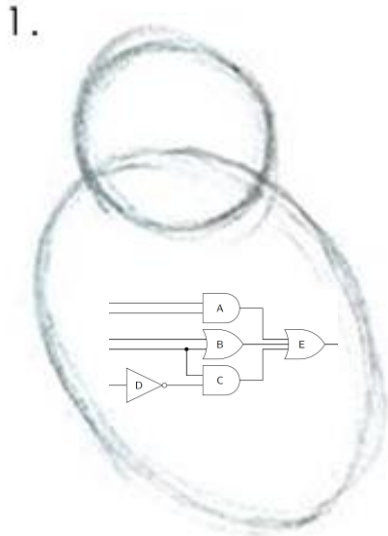
From: Production To: Navy

Cryptography in VHDL



AES in FPGA

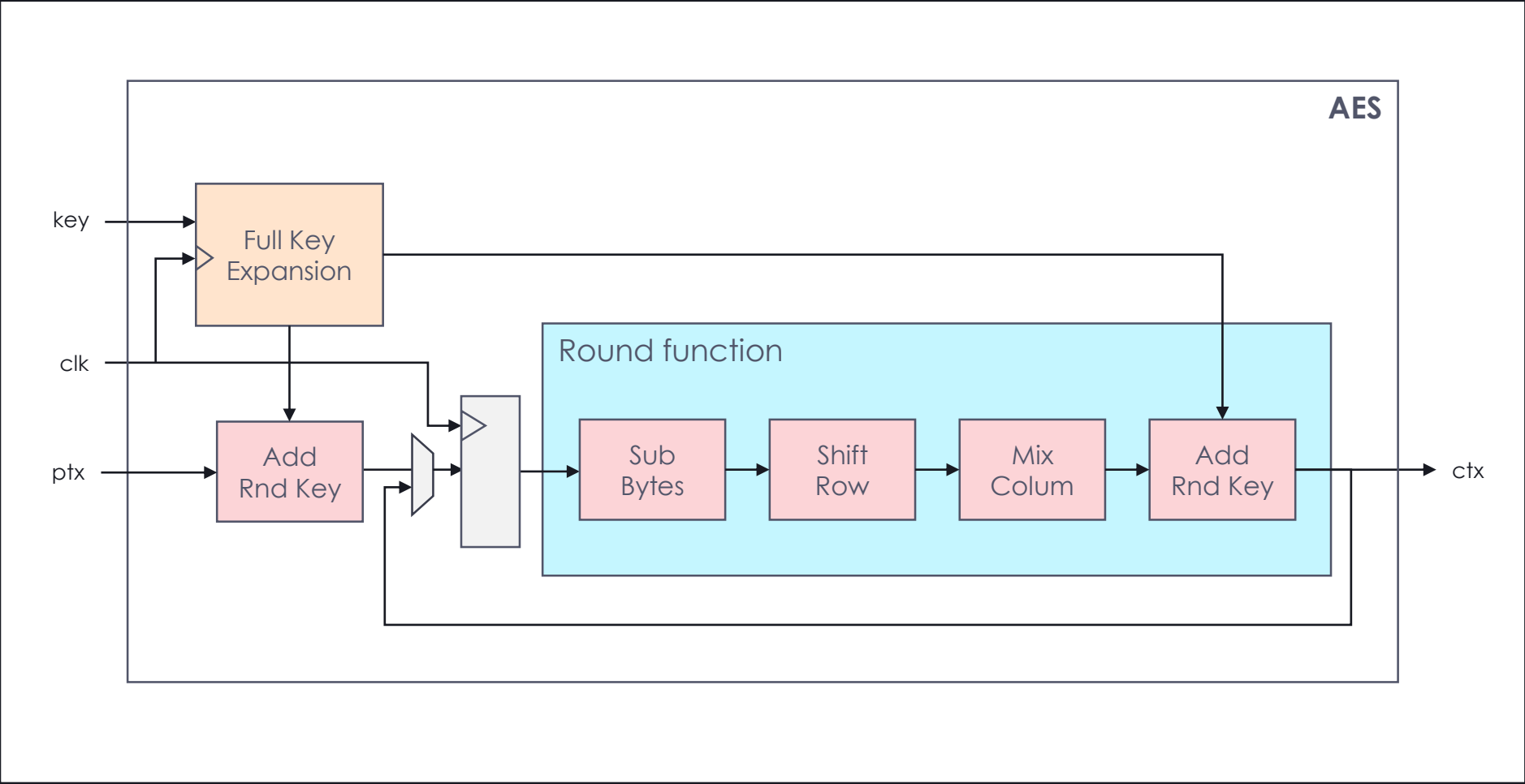
How to draw an owl



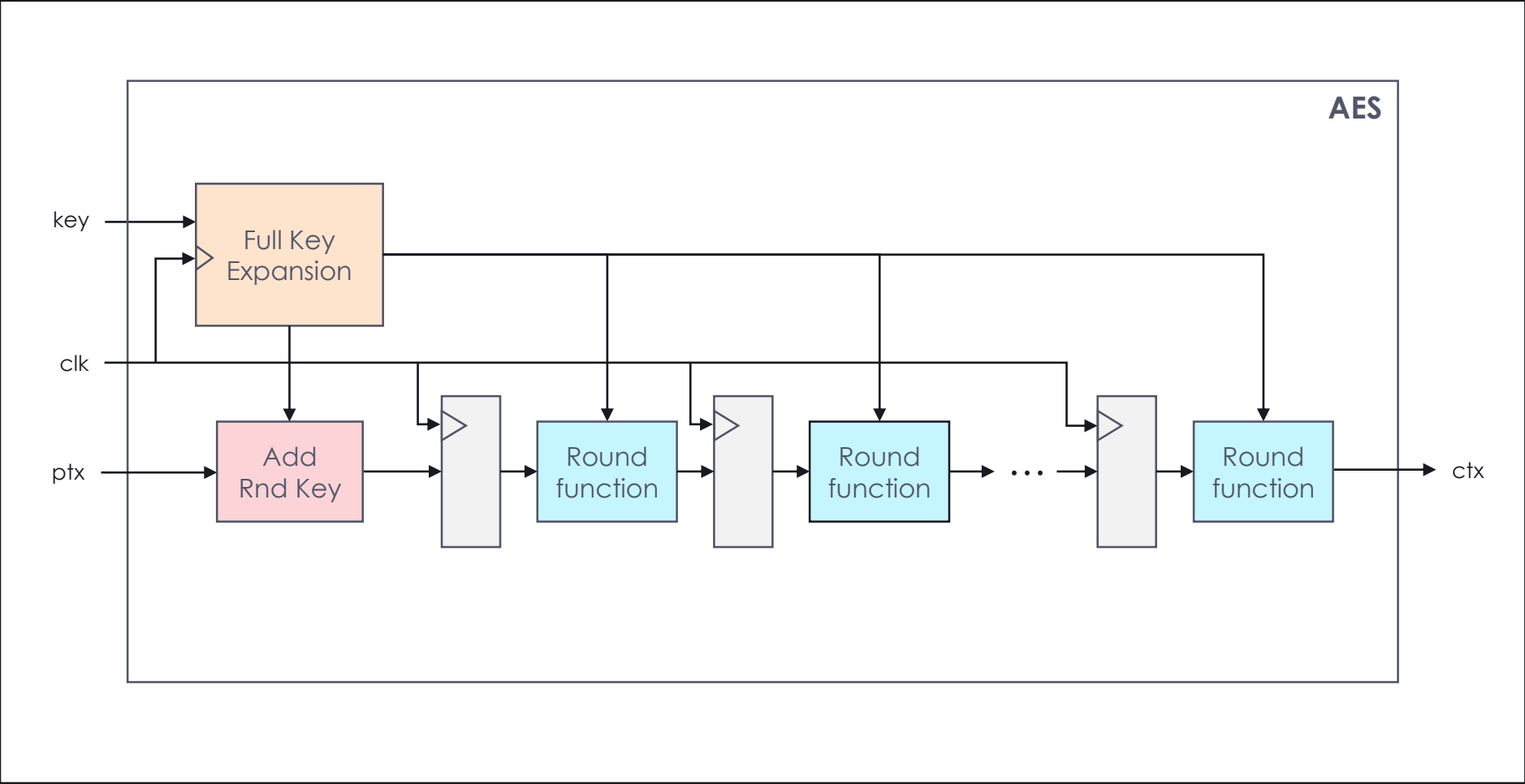
1. Draw some circles

2. Draw the rest of the fucking owl

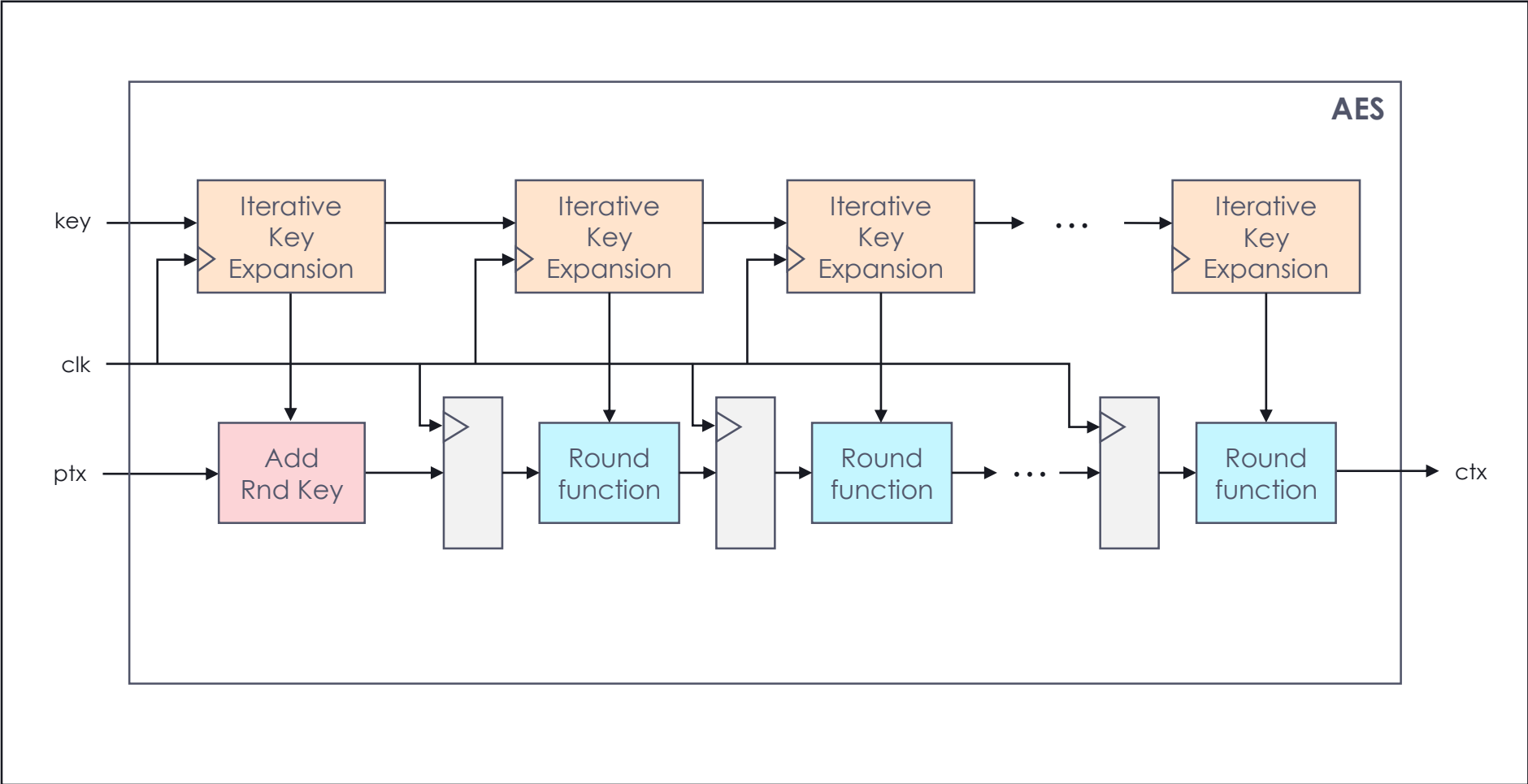
Compact design



Pipelined design



Fully pipelined design



High throughput implementations

Table 2 Implementation results and comparison

References	Devices	Frequency, MHz	Slices register	Slices LUT	Occupied Slices	BRAM	Throughput, Gbps	FPGA-Eff, Mbps/slice LUT	Occupied
this work	Virtex-5 XC5VLX85	622.4	19,123	14,966	5974	0	79.7	5.3	13.3
[5] ^a	Virtex-5 XC5VLX85	348.8	—	30,806	—	0	178.62	—	5.8
[23]	Virtex-5 XC5VLX85	576.0	—	22,994	—	0	73.7	3.2	—
[24]	Virtex-5 xc5vfx70t	460.0	—	—	9756	0	60.0	—	6.1 ^b
[25]	Virtex-5 XC5VLX85	528.4	—	3557	—	0	67.6	19.0	—
[13]	XC2V6000-6	194.7	—	—	3720	—	24.9	—	6.7
[26]	Virtex-5 XC5VFX70T	91.6	—	2030	—	28	0.9	0.5	—
[27]	Virtex-5 XC5VLX50	425.0	922	564	303	10	1.3	2.3 ^b	4.4
[28]	Virtex-5 XC5VLX50	242.2	—	5256	1745	0	3.1	0.6 ^b	1.8 ^b
[29]	Virtex-5 XC5VLX50	339.1	—	1338	399	0	4.3	3.2 ^b	10.8 ^b
[30]	Virtex-5 xc5vlx110t	250.0	—	—	—	0	31.2	—	—

^aIt has four cores; to make a fair comparison and according to [5], the throughput should be divided by four. The throughput for a single core is $178.62/4 = 44.7$ Gbps.

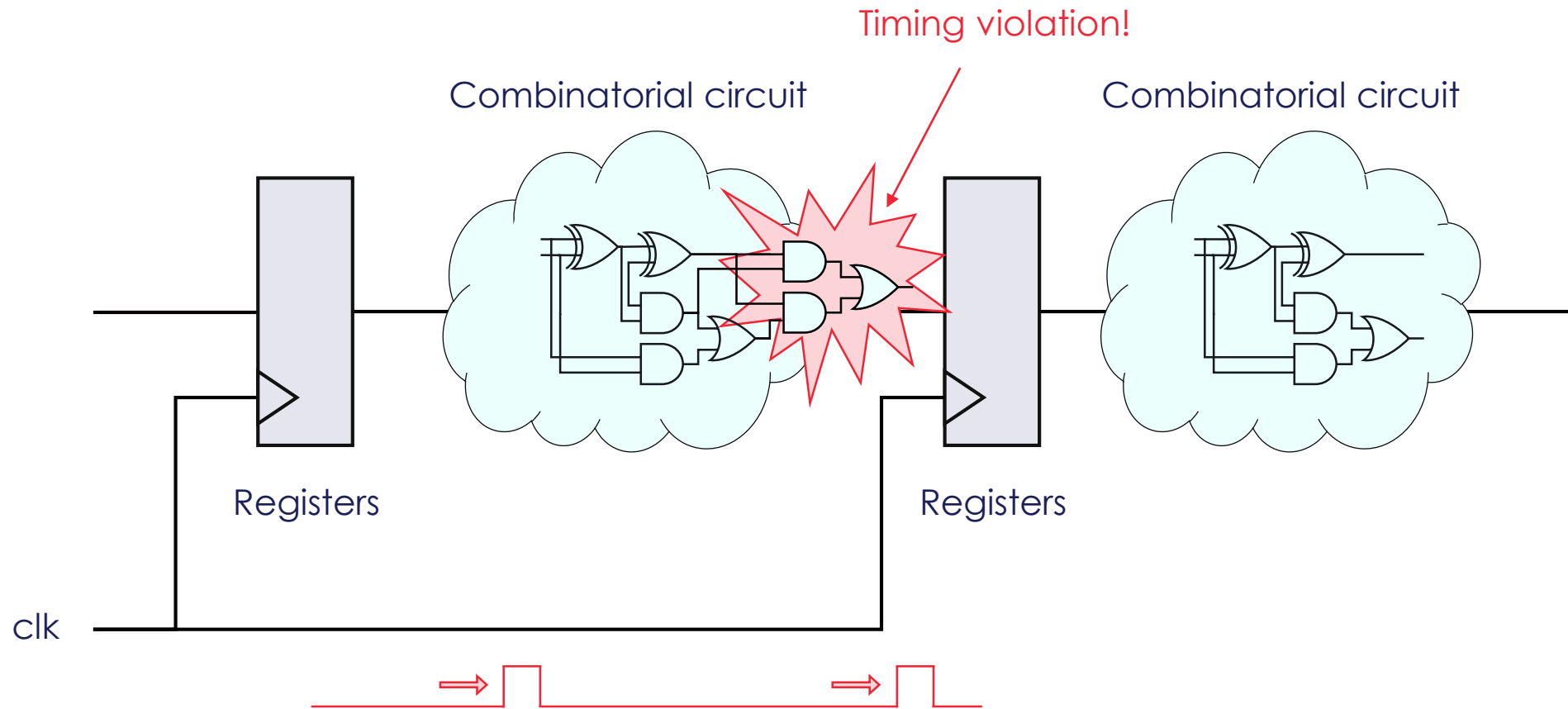
^bManually calculated.

Low area implementations

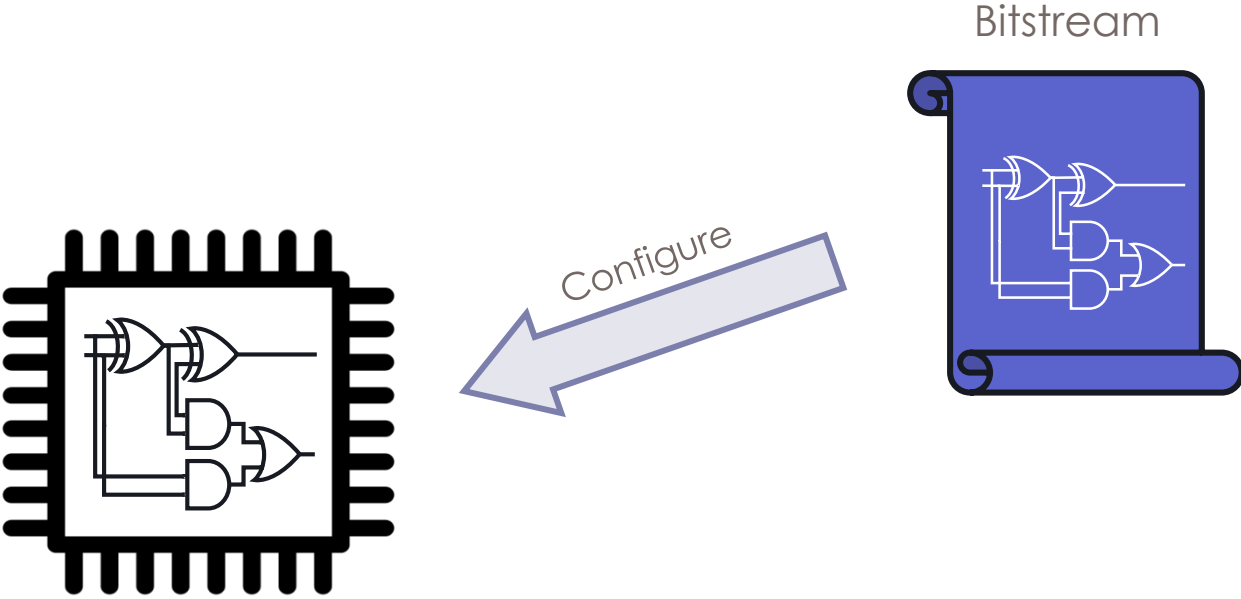
Design	Dec ^a	Key sch.	Device	Resources ^b						f (MHz)	Throughput ^c (Gbit/s)
				slices	LUT	FF	d.RAM	BRAM	DSPs		
<i>Basic</i>	○	○	Virtex-5	93	245	274	7838	2×36K	4	550	1.76
<i>Round</i>	○	○	Virtex-5	277	204	601	1432	8×36K	16	485	6.21
<i>Unrolled</i>	●	○	Virtex-5	428	672	992	1696	80×36K	160	430	55

Saar Drimer, Tim Guneysü, and Christof Paar. DSPs, BRAMs and a Pinch of Logic: New Recipes for AES on FPGAs.
https://saardrimer.com/sd410/papers/aes_dsp.pdf

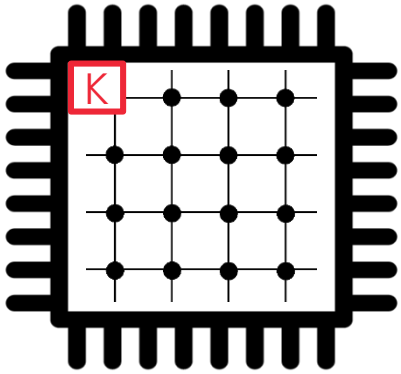
Timing analysis – critical path vs. clock frequency



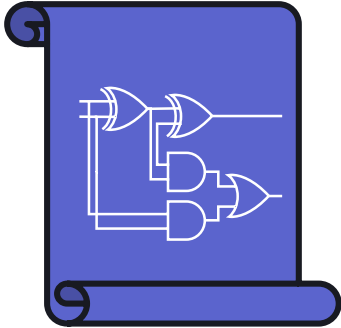
Protecting FPGA bitstreams



Protecting FPGA bitstreams



Bitstream

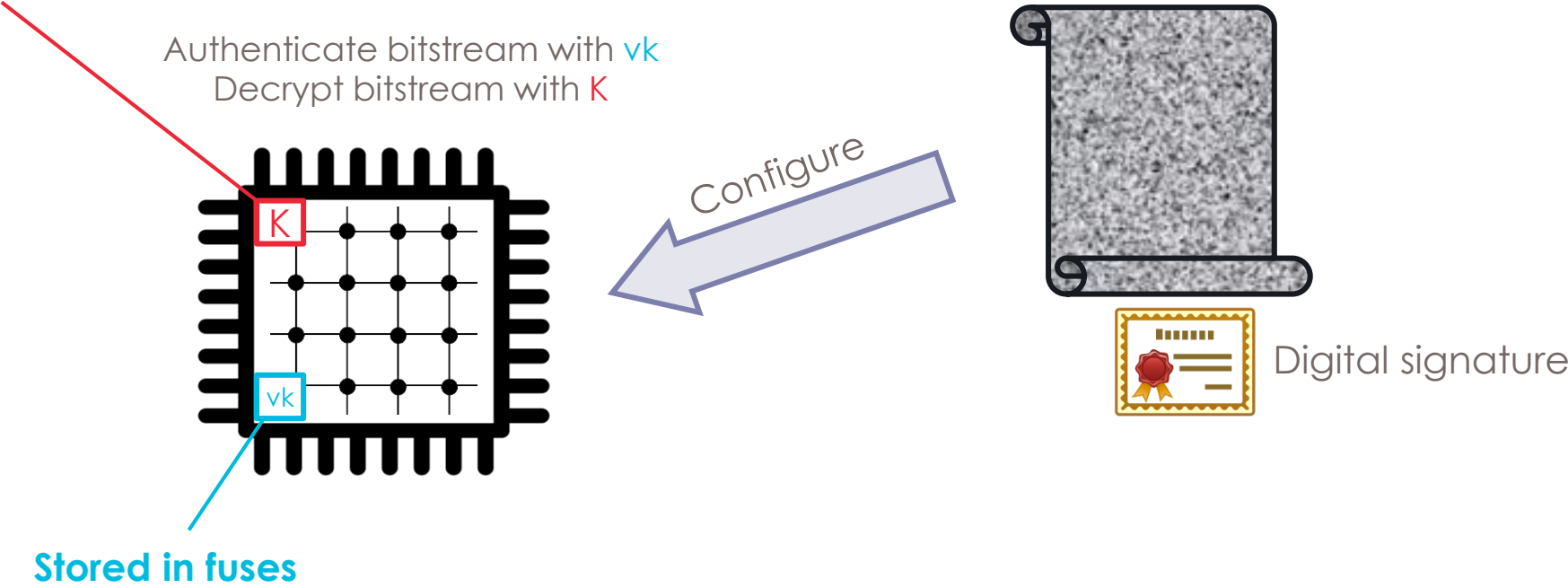


Stored in non-volatile memory outside FPGA

Bitstream design a business secret (or even a national/military secret)

Protecting FPGA bitstreams

Stored in battery-backed RAM (BBRAM)



Xilinx FPGA – Starbleed attack

ROOTDAEMON.COM

SECURITY NEWS

Unpatchable 'Starbleed' Exposes Critical Design Flaw

BY ROOTDAEMON © APRIL 21, 2020

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

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

*Horst Goertz Institute for IT Security, Ruhr University Bochum, Germany
**Max Planck Institute for Cyber Security and Privacy, Germany

Abstract

The security of FPGAs is a crucial topic, as any vulnerability within the hardware can have severe consequences, if they are used in a secure design. Since FPGA designs are encoded in a bitstream, securing the bitstream is of the utmost importance. Adversaries have many motivations to recover and manipulate the bitstream, including design cloning, IP theft, manipulation of the design, or design subversions e.g., through hardware Trojans. Given that FPGAs are often part of cyber-physical systems e.g., in aviation, medical, or industrial devices, this can even lead to physical harm. Consequently, vendors have introduced bitstream encryption, offering authenticity and confidentiality. Even though attacks against bitstream encryption have been proposed in the past, e.g., side-channel analysis and probing, these attacks require sophisticated equipment and considerable technical expertise.

In this paper, we introduce novel low-cost attacks against the Xilinx 7-Series (and Virtex-6) bitstream encryption, resulting in the total loss of authenticity and confidentiality. We exploit a design flaw which piecewise leaks the decrypted bitstream. In the attack, the FPGA is used as a decryption oracle, while only access to a configuration interface is needed. The attack does not require any sophisticated tools and, depending on the target system, can potentially be launched remotely. In addition to the attacks, we discuss several countermeasures.

1 Introduction

Nowadays, Field Programmable Gate Arrays (FPGAs) are common in consumer electronic devices, aerospace, financial computing, and military applications. Additionally, given the trend towards a connected world, data-driven practices, and artificial intelligence, FPGAs play a significant role as hardware platforms deployed in the cloud and in end devices. Hence, trust in the underlying platform for all these applications is vital. Altera, who are (together with Xilinx) the FPGA market leader, was acquired by Intel in 2015.

FPGAs are reprogrammable ICs, containing a repetitive logic area with a few hundred up to millions of reprogrammable gates. The bitstream configures this logic area; in analogy to software, the bitstream can be considered the 'binary code' of the FPGA. On SRAM-based FPGAs, which are the dominant type of FPGA in use today, the bitstream is stored on an external non-volatile memory and loaded into the FPGA during power-up.

In order to protect the bitstream against malicious actors, its confidentiality and authenticity must be assured. If an attacker has access to the bitstream and breaks its confidentiality, he can reverse-engineer the design, clone intellectual property, or gather information for subsequent attacks e.g., by finding cryptographic keys or other design aspects of a system. If the adversary succeeds in violating the bitstream authenticity, he can then change the functionality, implant hardware Trojans, or even physically destroy the system in which the FPGA is embedded by using configuration outside the specifications. These problems are particularly relevant since access to bitstream is often effortlessly possible due to the fact that, for the vast majority of devices, it resides in the external non-volatile memory, e.g., flash chips. This memory can often either be read out directly, or the adversary wiretaps the FPGA's configuration bus during power-up. Alternatively, a microcontroller can be used to configure the FPGA, and consequently, the microcontroller's firmware includes the bitstream. When the adversary gains access to the microcontroller, he also gains access to the configuration interface and the bitstream. Thus, if the microcontroller is connected to a network, remotely attacking the FPGA becomes possible.

In order to protect the design, the major FPGA vendors introduced bitstream encryption around the turn of the millennium, a technique which nowadays is available in most mainstream devices [1,56]. In this paper, we investigate the security of the Xilinx 7-Series and Virtex-6 bitstream encryption. On these devices, the bitstream encryption provides authenticity by using an SHA-256 based HMAC and also provides confidentiality by using CBC-AES-256 for encryption. By our attack, we can circumvent the bitstream encryption and decrypt an assumedly secure bitstream on all Xilinx 7-Series devices completely and on the Virtex-6 devices partially. Adversaries can then access the bitstream and break its confidentiality and authenticity.

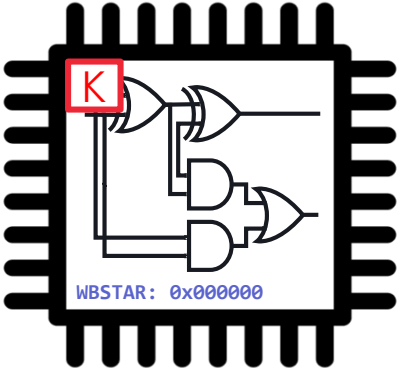
A newly discovered unpatchable hardware flaw could allow an attacker to break bitstream encryption.

USenix Association 29th USENIX Security Symposium 1803

bitstream, bitstream encryption, bug, Chips, decryption, decryption key, Field Programmable Gate Arrays, flaw, FPGA chips, Hardware, hardware encryption, hardware vulnerability, Starbleed, Starbleed vulnerability, vulnerability

Xilinx Starbleed attack

Bitstream

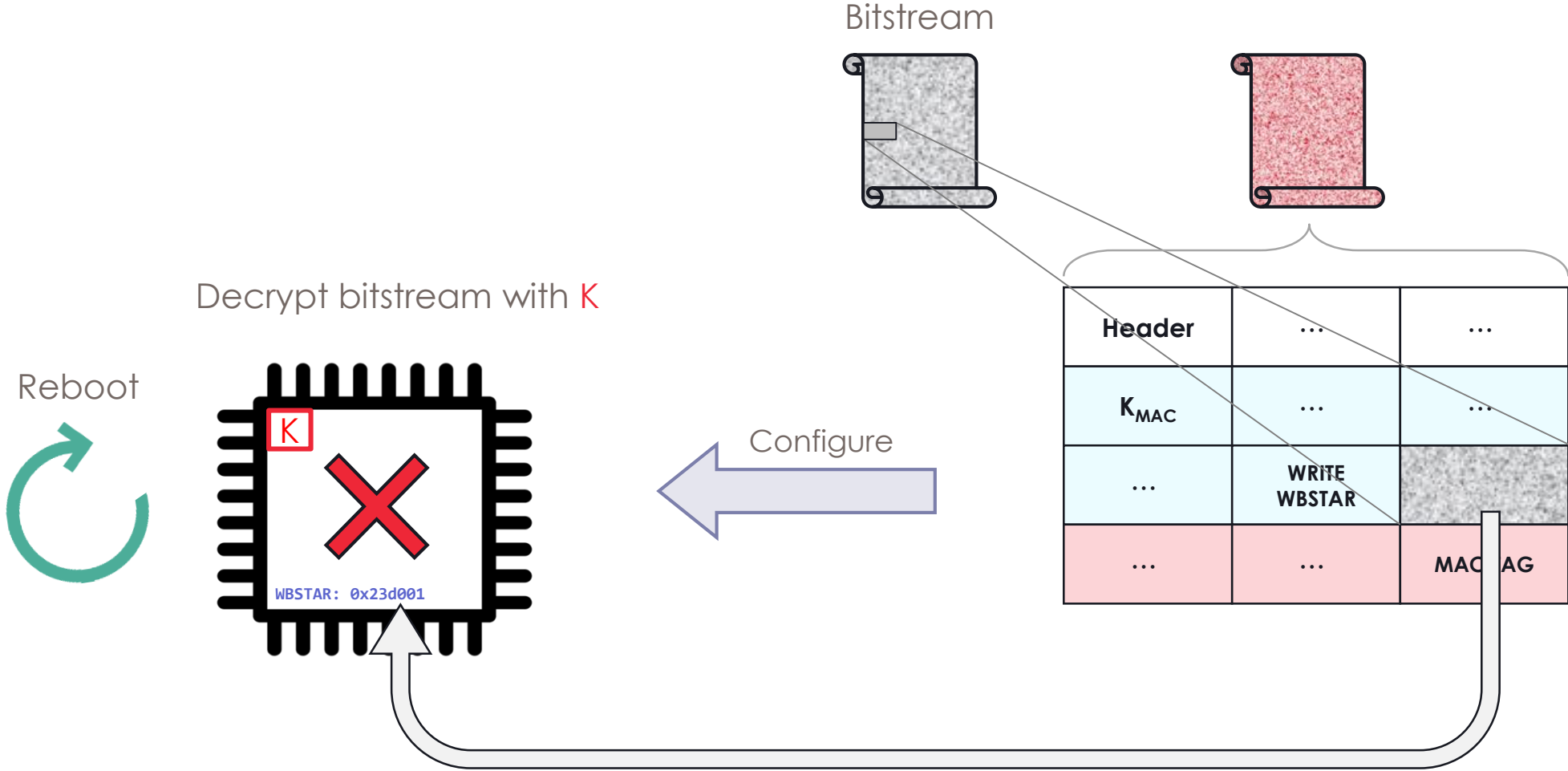


WBSTAR = Warm-Boot Start-address

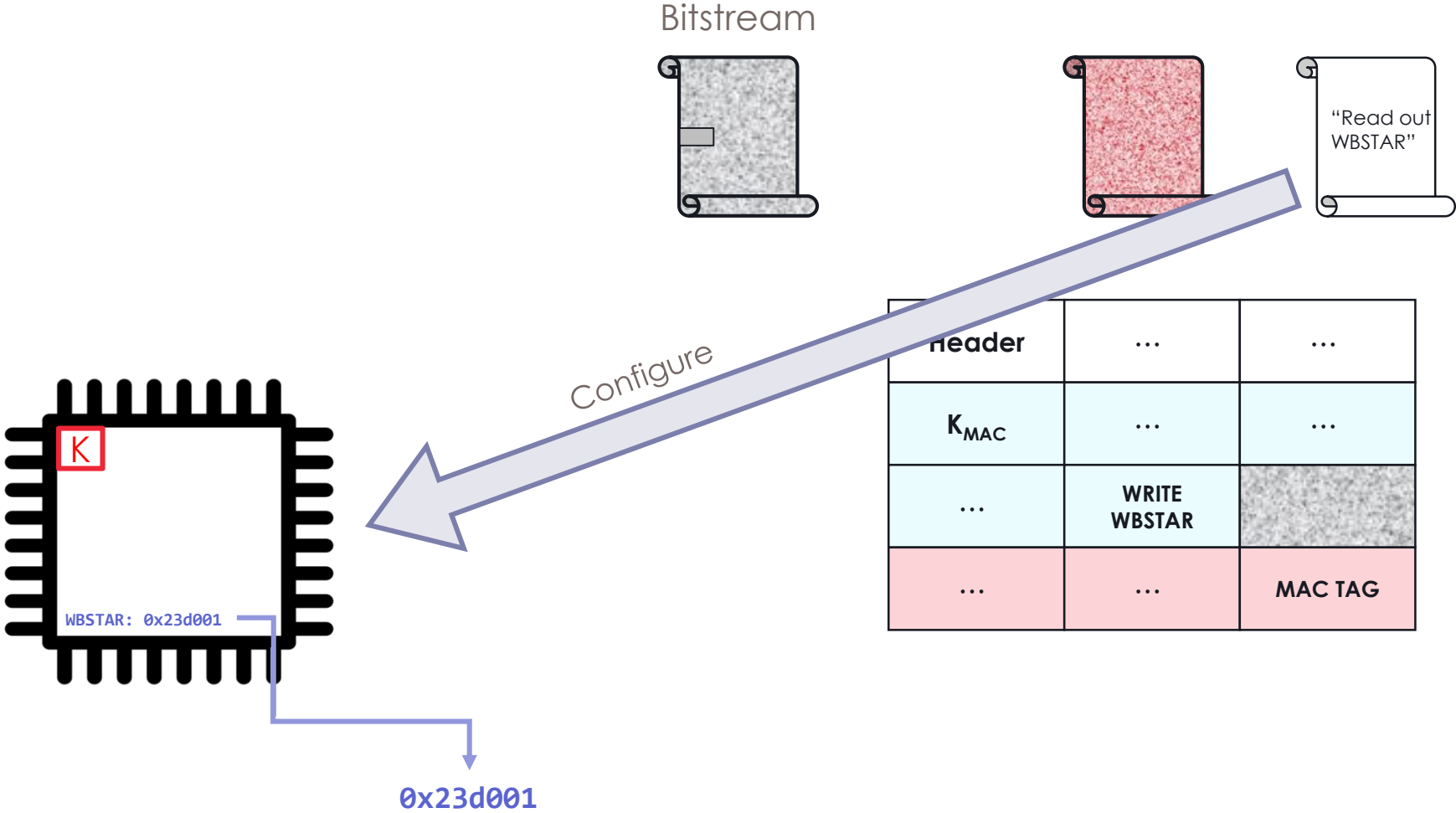
AES-CBC encrypted

Header
K_{MAC}
...	WRITE WBSTAR	0x00000000
...	...	MAC TAG

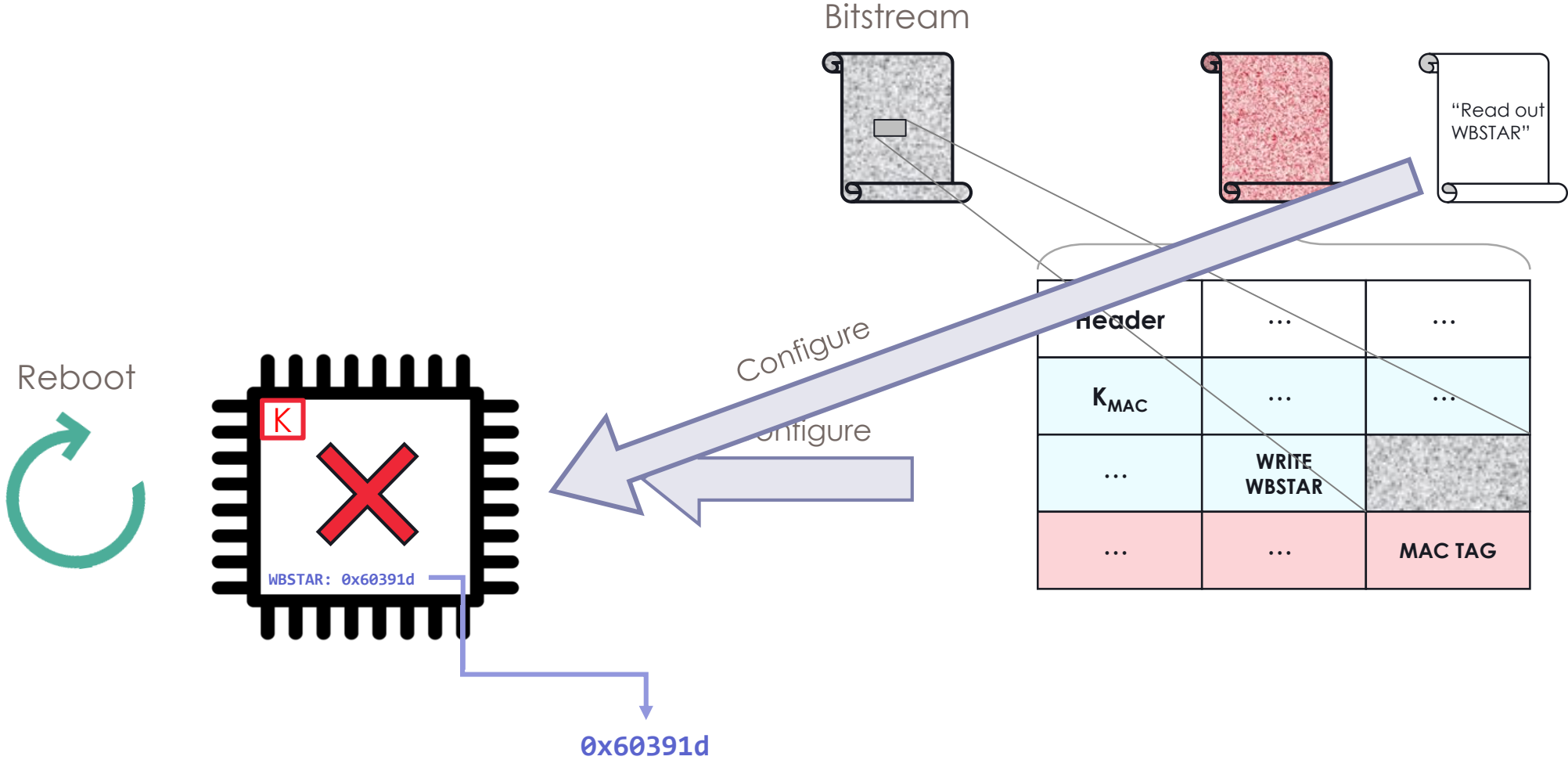
Xilinx Starbleed attack



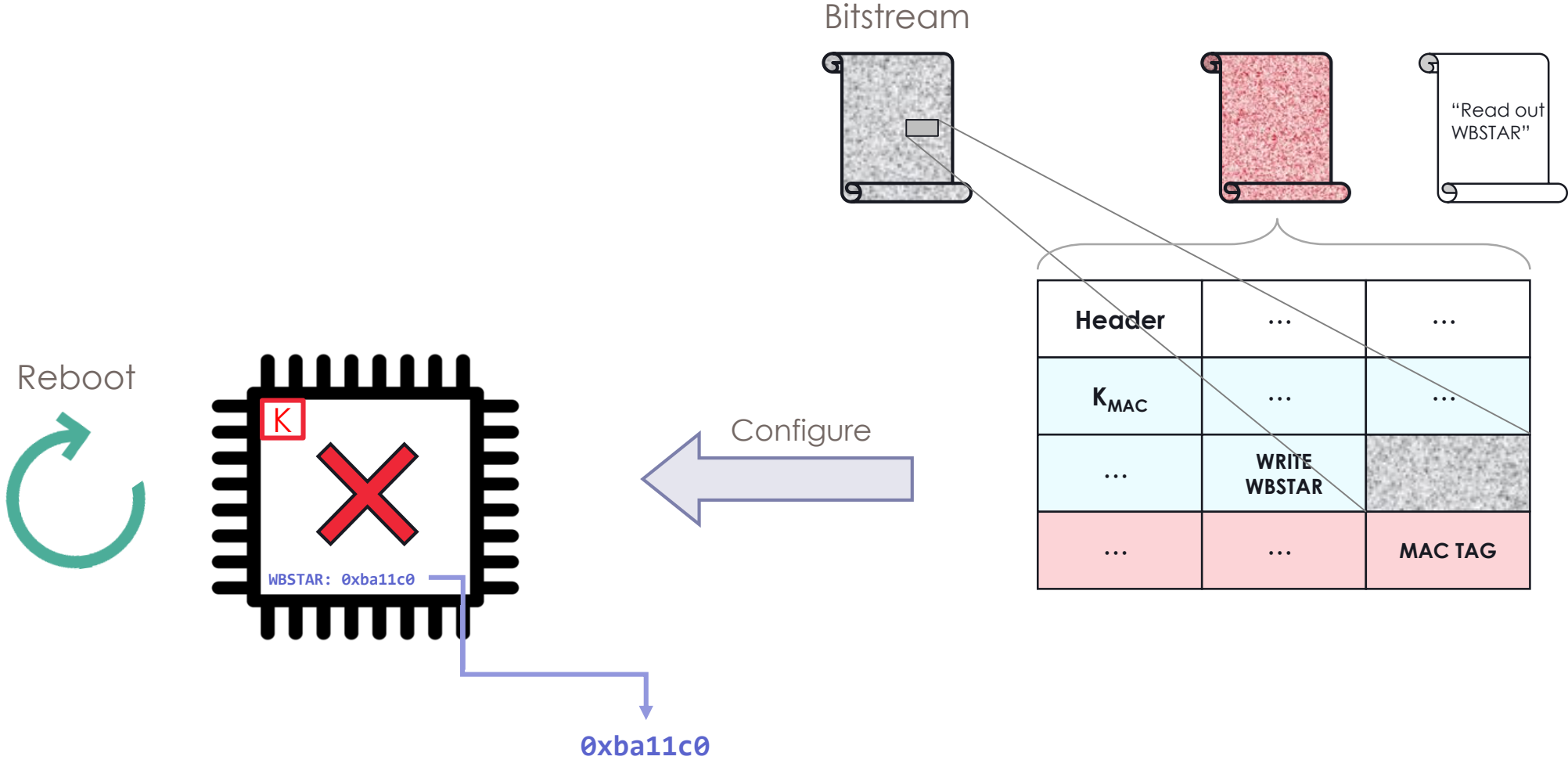
Xilinx Starbleed attack



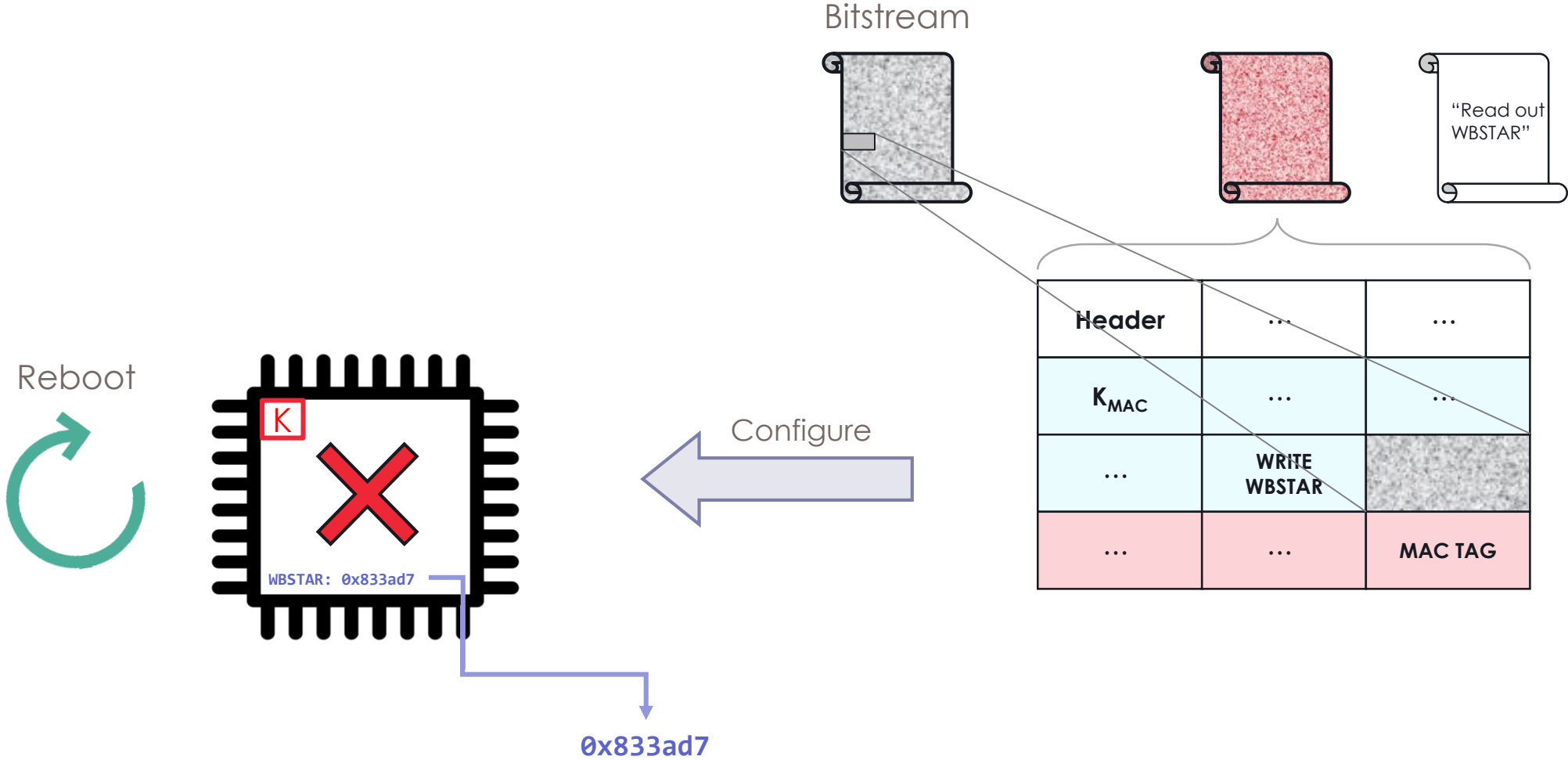
Xilinx Starbleed attack



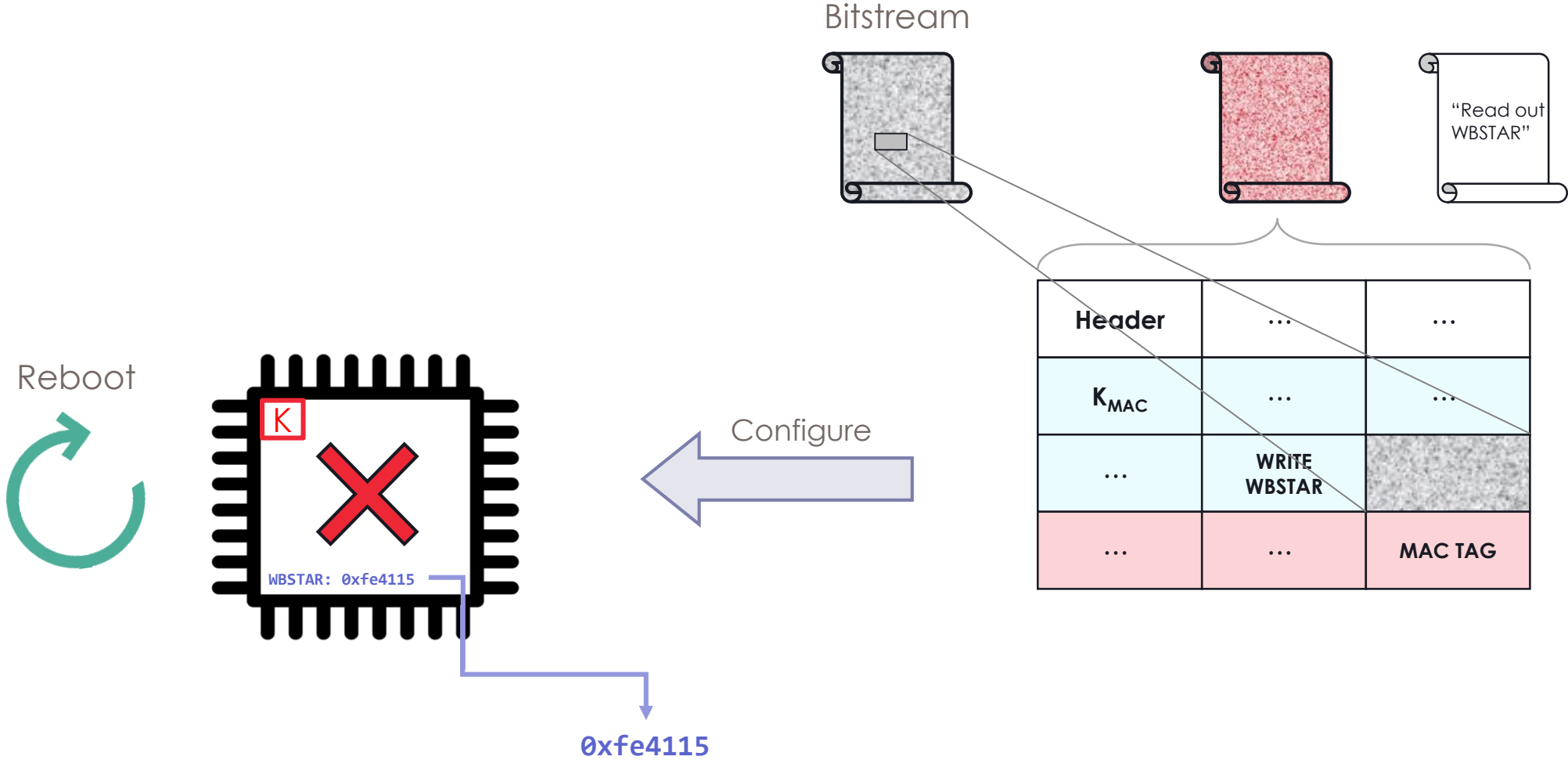
Xilinx Starbleed attack



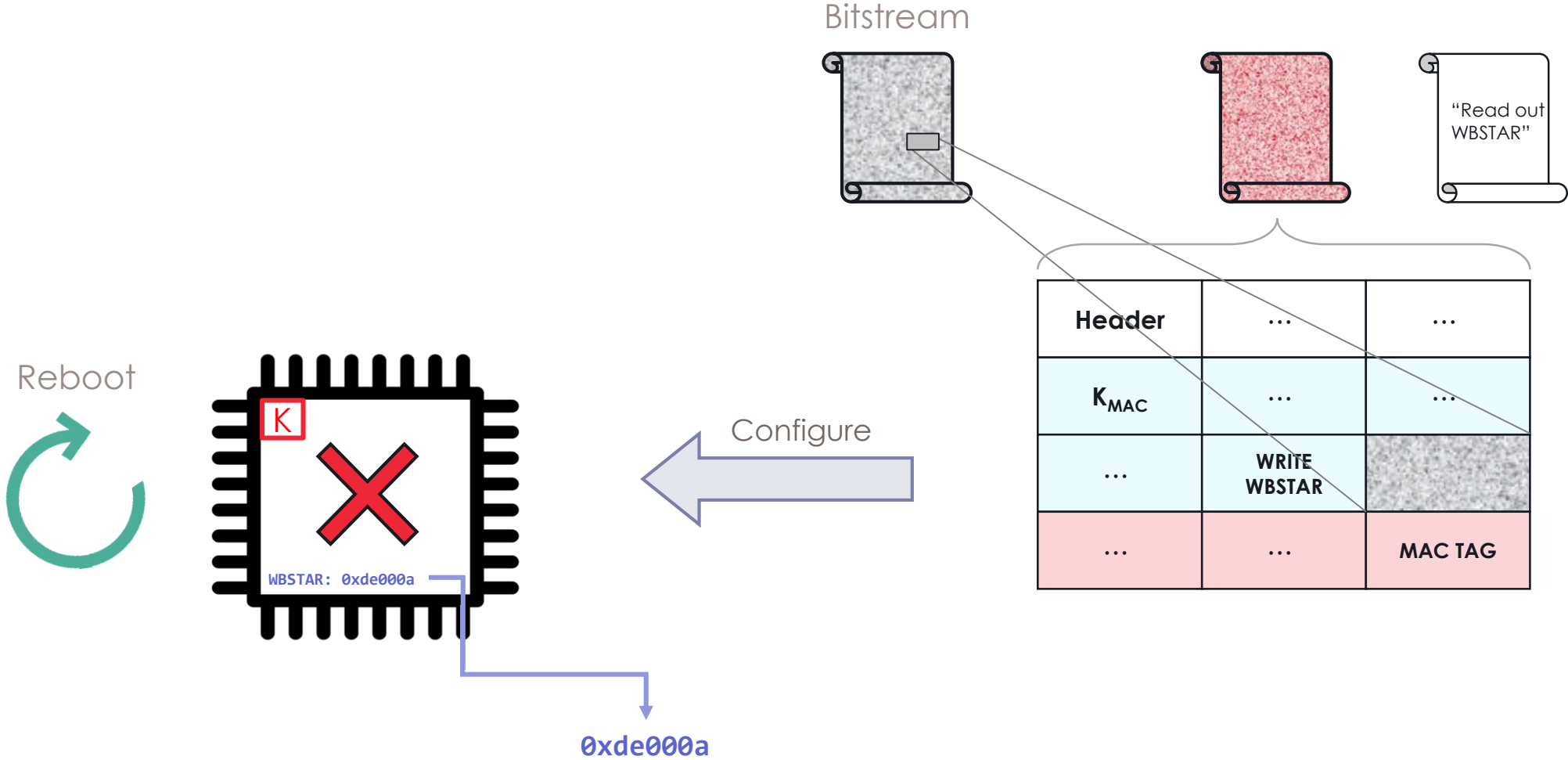
Xilinx Starbleed attack



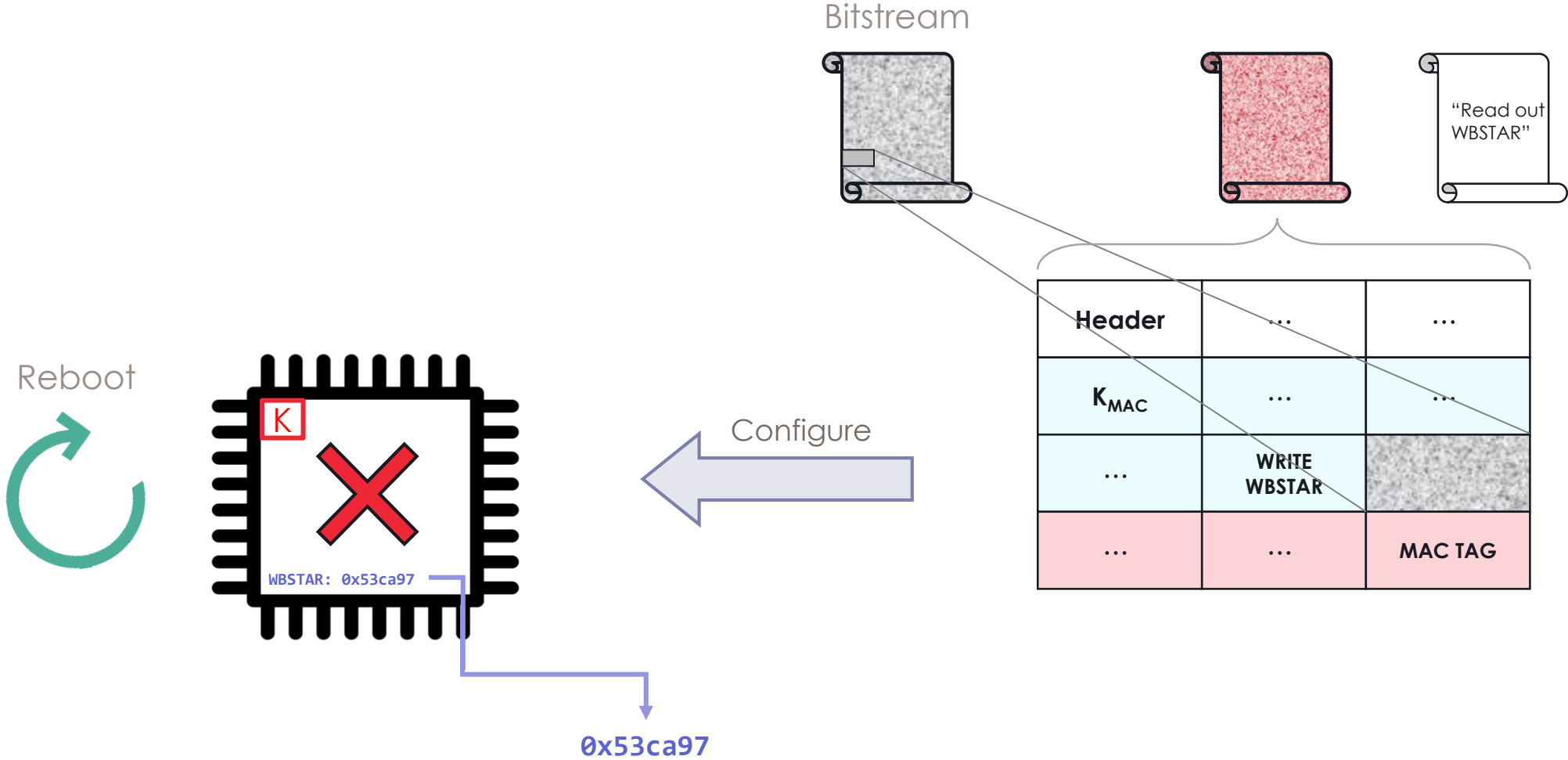
Xilinx Starbleed attack



Xilinx Starbleed attack

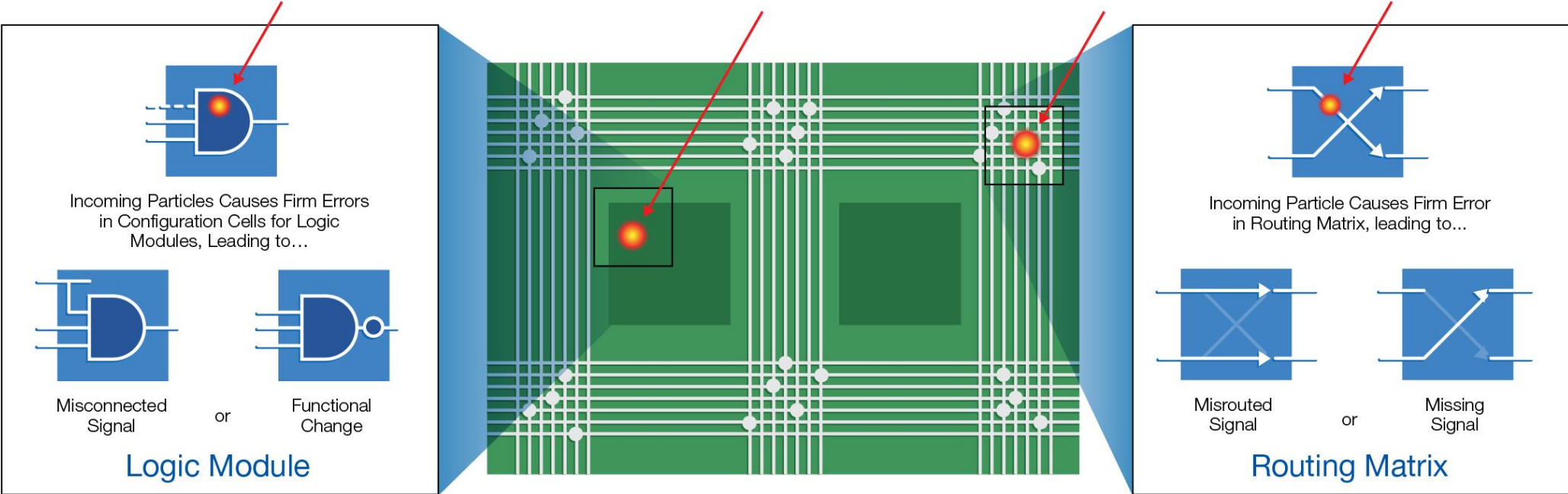


Xilinx Starbleed attack



Time to fully decrypt bitstream: 26 hours

Single Event Upsets (SEU)



SEU mitigation

Silicon

- Built-in Error Detection and Correction
- Optimized Integrated SRAM Designs

Packaging & Process

- Ultra-Low Alpha (ULA) Materials
- Material Quality Actively Monitored

Soft Error Mitigation Solutions

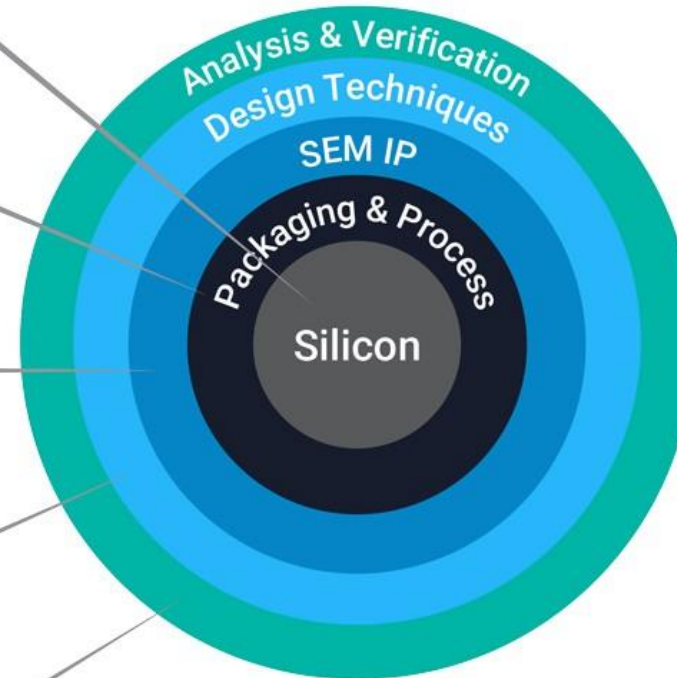
- Detection, Correction, and Classification
- Verification and Debug Management

Integrated Design Flow

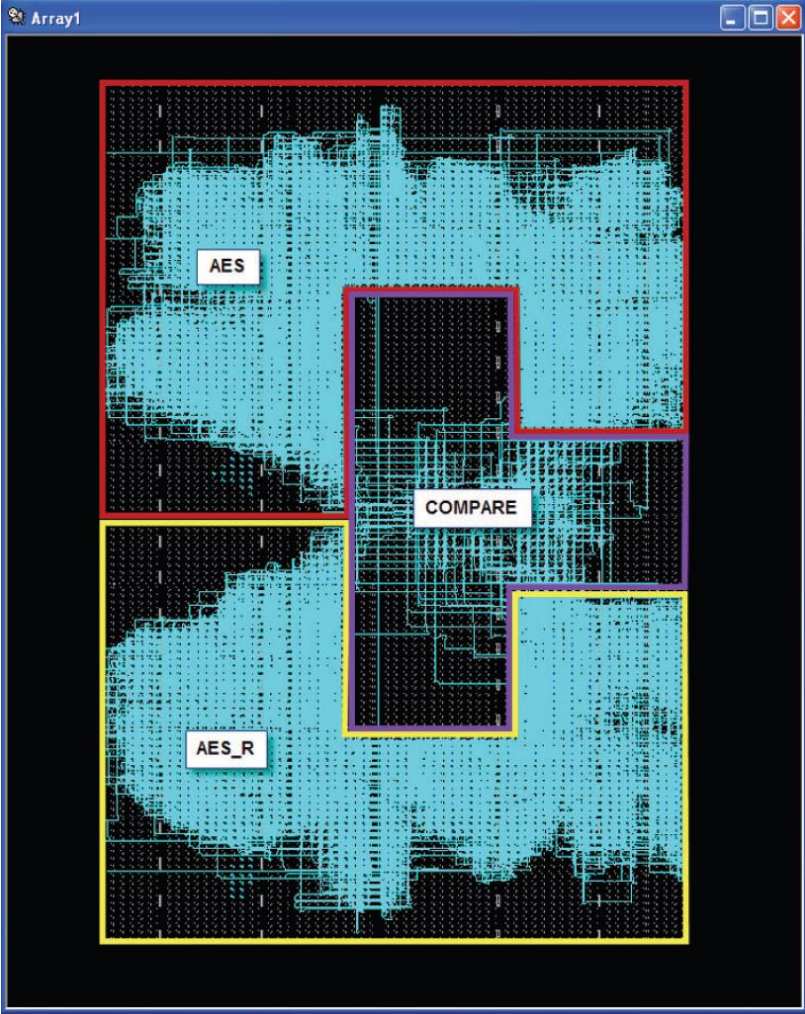
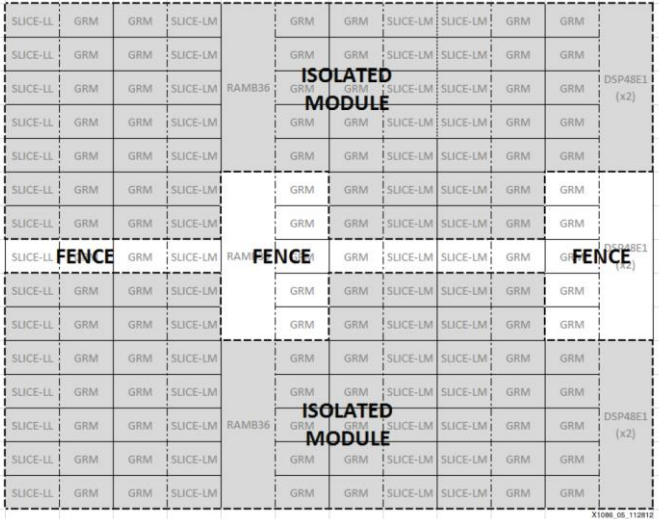
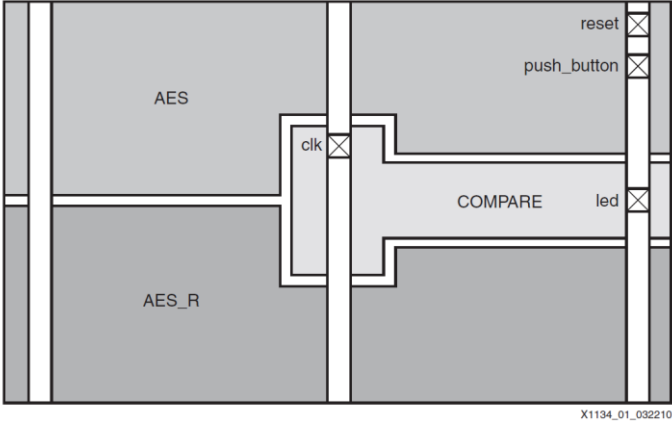
- Essential Bits Classification
- ECC-Protected Memory Solutions

Analysis & Verification

- SEU FIT and Vulnerability Analysis
- Fault Injection for System Validation



Hardware isolation



Protecting FPGA bitstreams

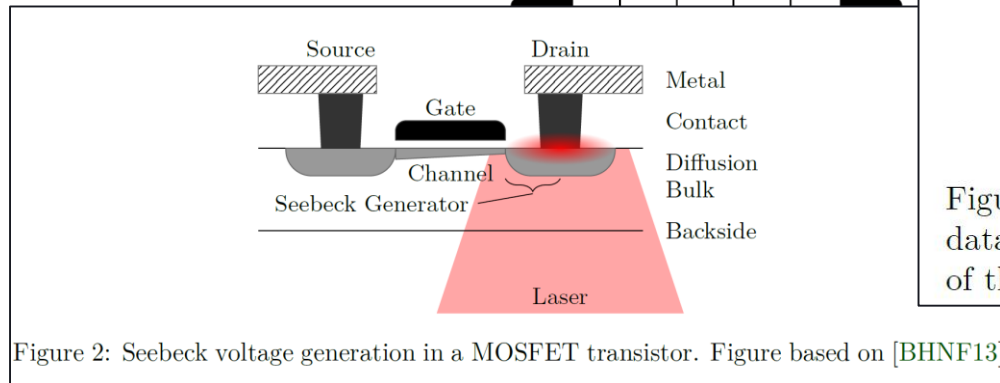
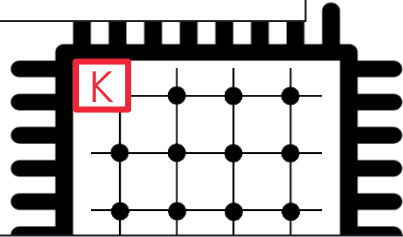
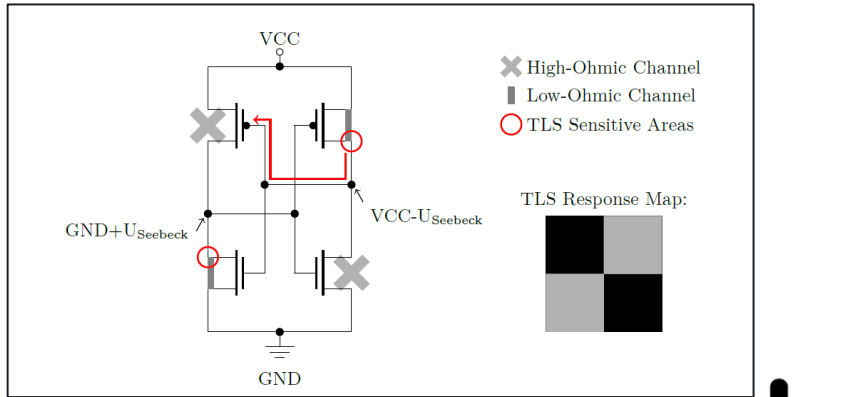


Figure 2: Seebeck voltage generation in a MOSFET transistor. Figure based on [BHN13]

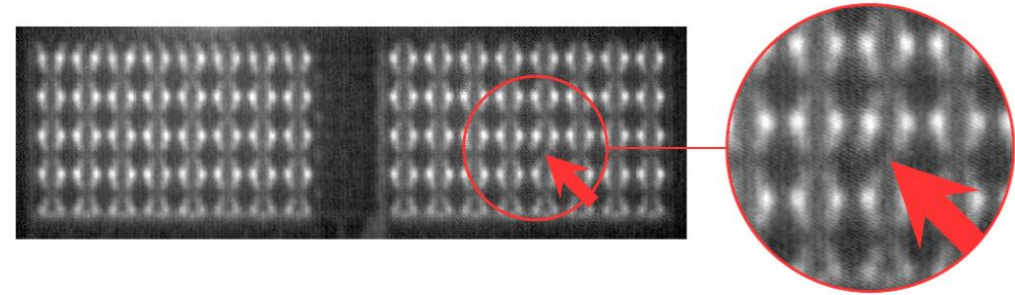


Figure 11: Data dependency of the TLS response. A single bit (bit 120) has been set in the BBRAM key data which manifests as an irregularity in the measurement result.

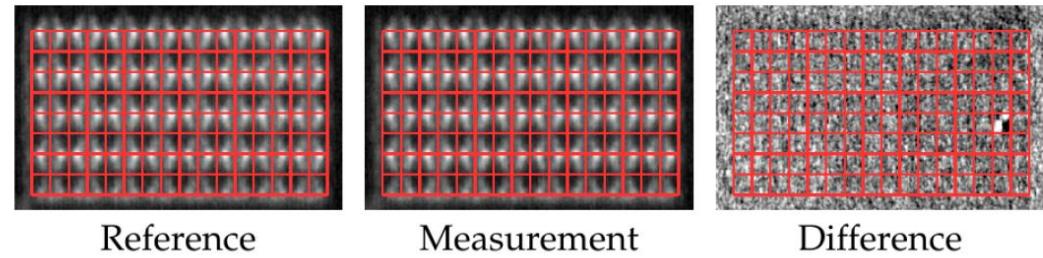
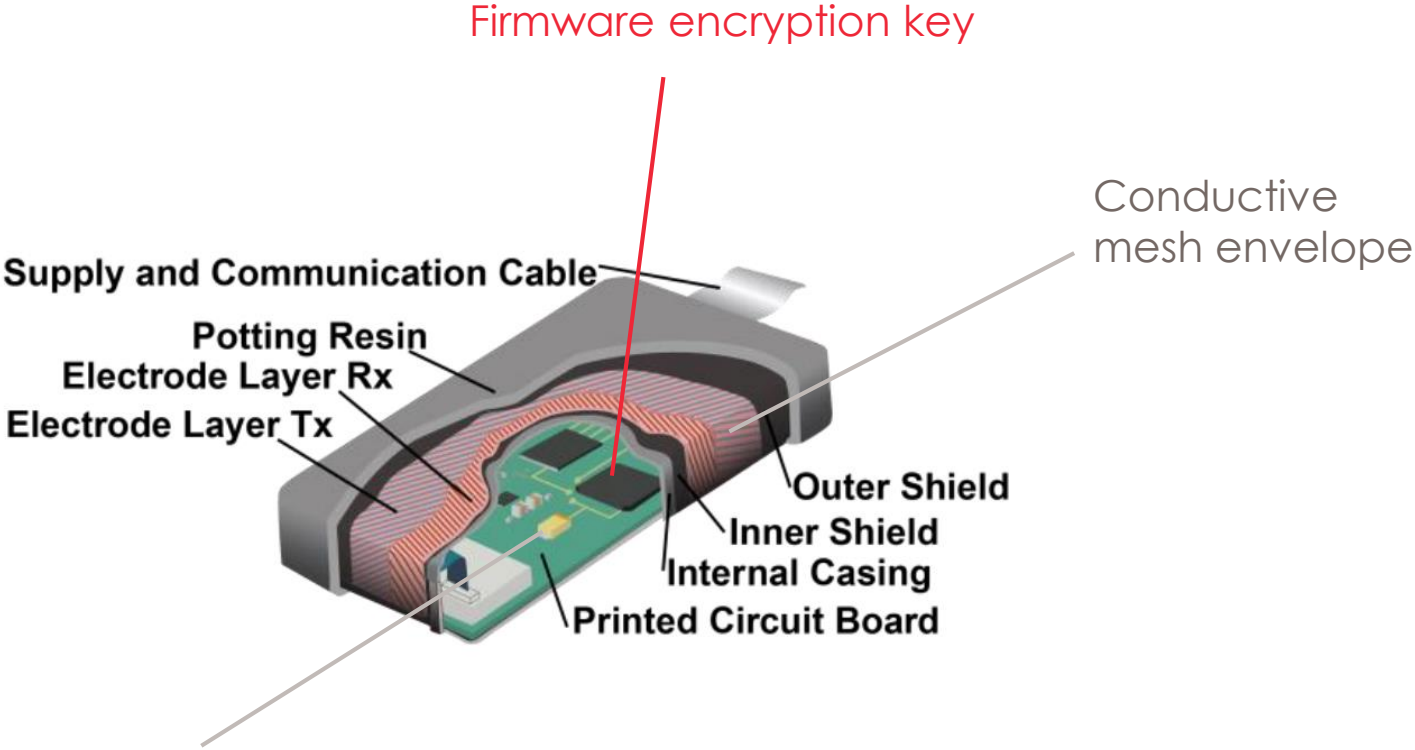


Figure 12: Difference calculation between an “all bits zero” TLS reference and measurement data quickly reveals which bits are set in the AES key. As an example the right-hand half of the BBRAM with a single bit set (bit 126) is shown here.

Anti-tamper



- Tamper detection circuit
- Alarm
 - Zeroization (erase key + configured design)

Summary

- **FPGAs are powerful and flexible**
- **Well suited for implementing cryptography**
- **Comes with unique possibilities and challenges**

THALES

www.thalesgroup.com