HECTOR

D2.2

ASIC and FPGA Designs

Accompanying Report

Project number:	644052
Project acronym:	HECTOR
Project title:	HECTOR: Hardware enabled crypto and randomness
Start date of the project:	1st March, 2015
Duration:	36 months
Programme:	H2020-ICT-2014-1

Deliverable type:	Demonstrator
Deliverable reference number:	ICT-644052 / D2.2 / 1.0
Work package contributing to the deliverable:	WP2
Due date:	April 2017 – M26 (shifted from M24 to M26)
Actual submission date:	May 3rd 2017

Responsible organisation:	STR
Editor:	Bernard KASSER
Dissemination level:	PU
Revision:	1.0

Abstract:	This is the accompanying report to Deliverable D2.2 of the HECTOR project, which is of type demonstrator.
Keywords:	ASIC, FPGA, Design, TRNG, PUF, TERO, PLL, DC

The project HECTOR has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052.

Editor

Bernard KASSER (STR)

Contributors (ordered according to beneficiary numbers)

Dave SINGELEE, Vladimir ROZIC, Bohan YANG (KUL)

Viktor FISCHER, Oto PEŤURA, Ugo MUREDDU (UJM)

Jean NICOLAI (STR)

Marek LABAN (MIC)

Executive Summary

HECTOR deliverable D2.2 consists in the delivery of FPGA and ASIC designs for selected TRNG(s) and PUF(s). Considering that the main deliverable is of type “Demonstrator”, and that its dissemination level is “Confidential”, this accompanying report provides a short, publically-available document to summarize the D2.2 design deliveries.

After a brief document introduction in chapter 1, chapter 2 focuses on the FPGA design deliveries. It starts from the description of the data interfaces that have been defined for all the FPGA designs implemented in the HECTOR evaluation platform. The PLL and DC TRNG designs are then summarized, followed by the TERO-based PUF design.

Chapter 3 goes on to focus on the ASIC design deliveries. The HECTOR ASIC section lists the TRNG and PUF designs that have been produced for the HECTOR ASIC test chips that will be manufactured through the CMP multi-project scheme on ST 65nm technology. More specifically, it describes the PLL, ELO and STR TRNG designs, as well as the TERO and RO based PUF designs that have been produced. The ST ASIC designs lists additional TERO and PLL TRNG designs integrated into additional ST silicon opportunities.

Chapter 4 wraps-up this document.

Table of Content

List of Figures

List of Tables

Chapter 1 Introduction

HECTOR D2.2 consists in the delivery of FPGA and ASIC designs of selected TRNG(s) and PUF(s). The deliverable is of type “Demonstrator”, and its dissemination level is “Confidential”, i.e. restricted to members of the Consortium and the EU Commission services.

This accompanying report provides a short, publically-available document, illustrating and summarizing D2.2 design deliveries.

We acknowledge and agree with recommendation R3 from the expert report of the October 5th 2016 review, which stressed the importance of proper robustness assessments and comparisons of HECTOR WP2 designs. These important activities will continue until the end of the project and therefore were not the focus of D2.2 or this accompanying report.

Chapter 2 Design of TRNGs and PUFs in FPGA

2.1 Data Interfaces

The HECTOR hardware evaluation platform is composed of the motherboard and associated daughterboard. To make the remote testing of FPGA designs implemented in the daughterboard possible, the daughterboard is connected to the motherboard. Hereby we use a limited number of signals realizing a fast data interface based on the low voltage differential signalling (LVDS) and a simple synchronous serial interface using two data signals (input and output), and one clock signal.

Although both data interfaces are always available, they are not used for the TRNG and PUF implementation and testing in the same way. Therefore, we describe separately the two available modes of communication between motherboard and daughter board.

2.1.1 Data interface dedicated to TRNG implementation and testing

Data interface dedicated to TRNG implementation and testing uses three LVDS pairs (LVDS0, LVDS1, and LVDS2) and three general purpose input/output (GPIO) signals (GPIO0 - GPIO2) configured as presented in Figure 1. Note that the LVDS3 pair is separated into two independent inputs/outputs used in the serial control interface.

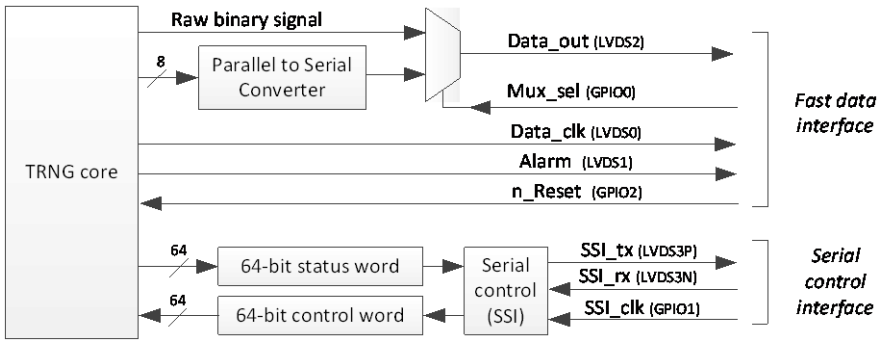

Figure 1: TRNG data and control interface

Two kinds of data can be output via the fast data interface depending on the value of the *Mux_sel* signal:

- A raw binary signal - *Mux_sel* = 0
- An 8-bit data signal (for jitter characterization) - *Mux_sel* = 1

The Parallel to Serial Converter converts 8-bit data to a bit stream and it adds start and stop bit to guarantee byte alignments. The interface is synchronized with the *Data_clk* signal featuring the maximum frequency of 250 MHz.

The Serial Control Interface includes 64-bit parallel to serial and serial to parallel converters, which are aimed to read the TRNG status and to write a control command to the TRNG core. The application software can read the TRNG status any time (usually before and after data acquisition or once the alarm signal (from the fast data interface) is asserted.

2.1.2 Data interface dedicated to PUF implementation and testing

The data interface dedicated to the PUF implementation and testing uses one LVDS pair (LVDS0), two GPIO signals (GPIO1 and GPIO2), and one LVDS3 pair separated to two independent inputs/outputs used in the serial control interface, as presented in Figure 2. Note that in this case, besides the clock signal sent from the motherboard, the fast data interface is not needed - only a limited amount of data is transferred.

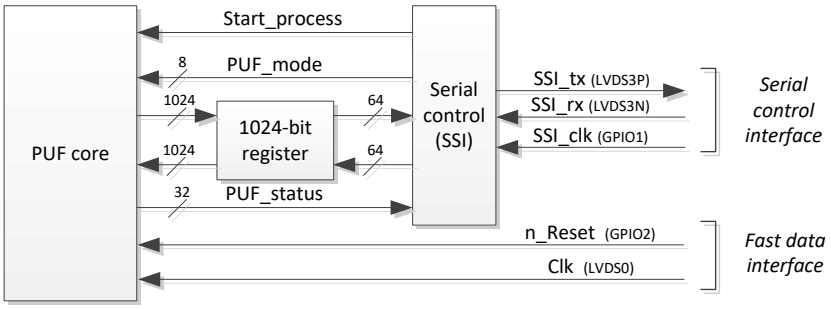

Figure 2: PUF data and control interface

A synchronous serial interface (SSI), similar to that used by the TRNG, is used to transfer challenges and responses. The data are transferred as 64-bit words, with the first word containing control information. The control word consists of the *PUF_mode* and the number of words which will be received and transferred by the daughter board. The data words are immediately received depending on this number. The daughter board responds by the sending 32-bit *PUF_status* and appropriate data words. The clock received from the motherboard is used as a main internal clock to avoid potential problems between two board clock domains.

2.2 Implementation of TRNGs in FPGA

Referring to deliverable D2.1, we implemented two types of TRNGs in HECTOR daughter boards:

- Phase locked loop (PLL) based TRNG,
- Delay chain (DC) based TRNG.

Their design is shortly described and discussed in the following subsections.

2.2.1 PLL TRNG

The PLL TRNG uses random jitter of the clock signal generated inside the phase locked loop as a source of randomness. The PLL TRNG is configured as presented in Figure 3.

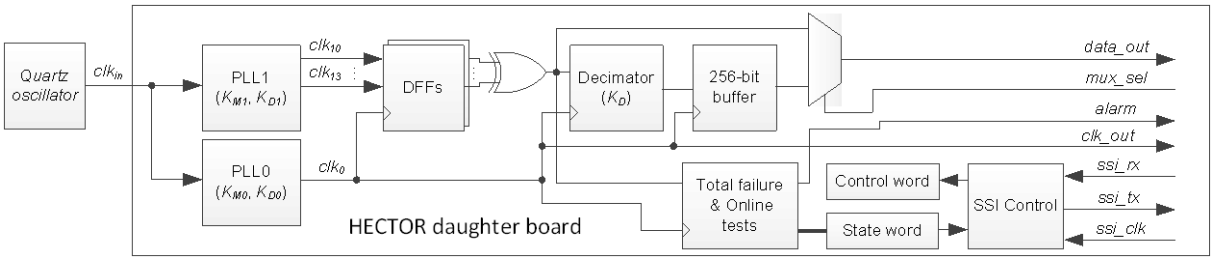

Figure 3: Architecture of the PLL TRNG

The TRNG core uses two PLLs connected in parallel. PLL1 generates one to four phases of the jittery clock signal and PLL0 generates the reference clock signal. A decimator adds modulo 2 the binary samples (the output of the XOR gate) obtained during K_D periods of the reference clock signal. The decimator input value (the output of the XOR gate) is used in embedded tests (the Total failure test and Online tests) and its output is used as a raw random bit. Since the Total failure test has a latency of 256 periods T_Q (each period T_Q lasts K_D periods of the reference clock signal), a 256-bit buffer is inserted before the output of the generator. This way, all output bits are verified by the Total failure test.

Depending on the value of the *mux_sel* signal, the TRNG core outputs the output of the XOR gate (this is particularly useful for jitter characterization) or the output of the decimator (which can be used to create a raw random bit stream).

The total failure alarm is output via a dedicated output included in the fast data interface and the corresponding bit is set up in the state register. Besides the Total failure test and Online test bit flags, the status register contains values of parameters needed for jitter characterization.

2.2.1.1 Implementation and implementation results

The PLL TRNG was implemented in VHDL and mapped to the HECTOR Cyclone V daughter board. The only part of the design which needed special attention is the routing between PLL1 outputs and sampling D flip-flops (see Figure 3). The routing needed to be made manually so that the signal delays between clock buffers (which route PLL outputs to the device) and flip-flops match as much as possible. The placement of the clock buffers and sampling flip-flops is clear from the floor plan, which is depicted in Figure 4.

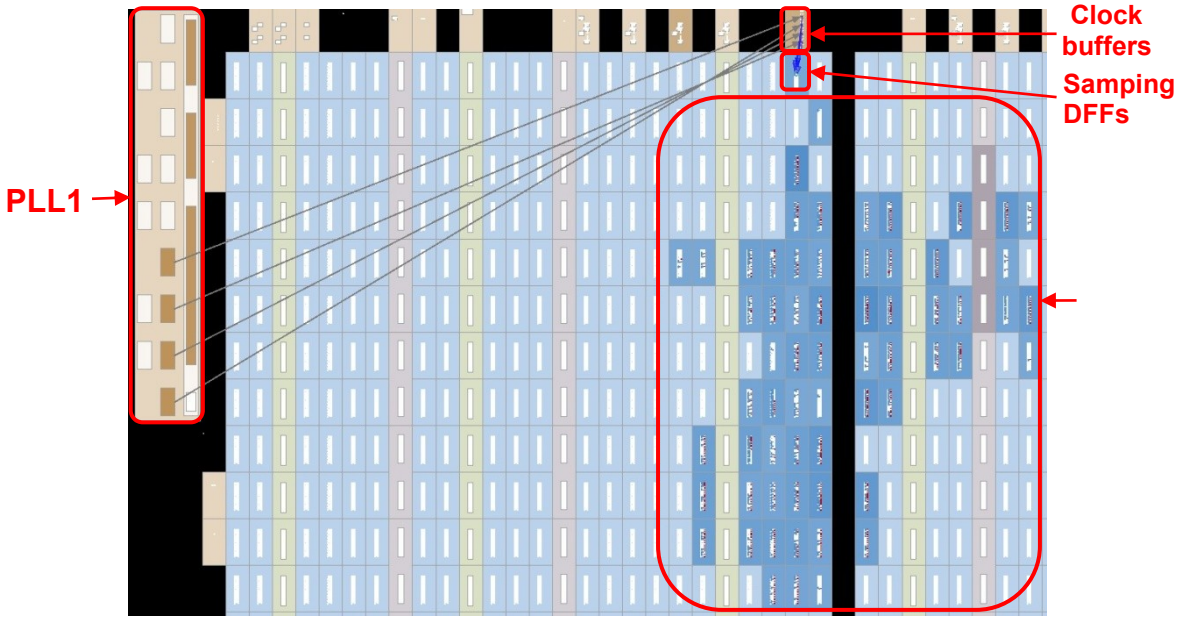

Figure 4: Floor plan of the HECTOR PLL TRNG

PLL TRNG implementation results in Altera Cyclone V FPGA are presented in Table 1. The following parameters characterize the area occupied by individual TRNG modules and by the whole TRNG:

ALMs: Altera adaptive logic modules,

Comb. ALUTs: Altera combinational adaptive look-up tables,

Registers: Dedicated registers.

Note that the total TRNG area is slightly bigger than the sum of the sizes of individual modules. The difference represents the area of the top level module containing status register and error vectors.

Module	ALMs	Comb. ALUTs	Registers
TRNG core	38	43	82
Jittter_eval	62	76	123
SSI	132	107	361
Out_sel	134	142	356
Total	386	413	960

Table 1: PLL TRNG implementation results

2.2.2 DC TRNG

The architecture of the DC TRNG is shown in Figure 5. The generator extracts the entropy from the timing jitter accumulated in the ring oscillator. The output of each stage in the oscillator is connected to a tapped delay-chain.

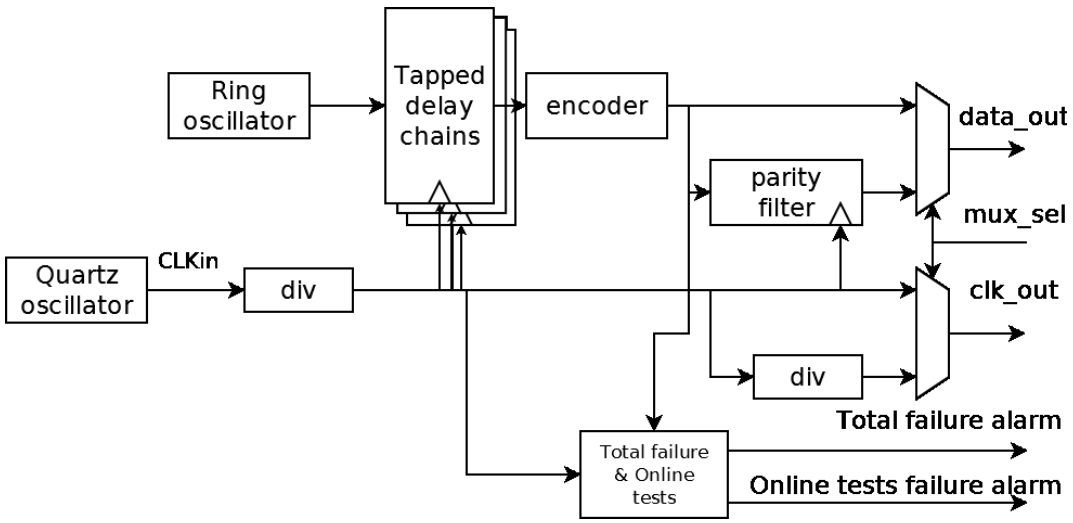

Figure 5: Architecture of the DC TRNG

A tapped delay chain consists of fast buffers with flip-flops connected at the output of each stage. The signal for sampling the tapped delay chains is derived from the quartz oscillator using the frequency divider. The encoder performs a simple operation on the data captured in the tapped delay chains in order to extract a single raw bit: it filters the “bubbles“ in the code that can be created due to violating the timing conditions of the flip-flops, determines the position of the signal edge using priority encoding and outputs the least significant bit of the position.

A parity filter is used for post-processing the raw data. Depending on the value of the signal *mux_sel,* either the raw data or the post-processed data is sent to the output. Two dedicated outputs are used to send the alarm signals in case of a total failure or a statistical weakness.

2.2.2.1 Implementation and implementation results

The DC TRNG was implemented in Verilog and mapped to the HECTOR Spartan-6 daughter board. Dedicated placement was required for the ring-oscillator and the tapped delay chains. The tapped delay chains were implemented using Xilinx CARRY4 primitives. Half of the slices on Xilinx Spartan-6 FPGA contain these primitives. Since these primitives are connected using dedicated routing paths on FPGA, it is necessary to place all elements of a single tapped delay chain in a single column, next to each other. The ring oscillator is implemented using three LUTs. Each LUT is placed next to the corresponding tapped delay chain. The placement of the tapped delay chains and the ring oscillator is clear from the floor plan, which is depicted in Figure 6.

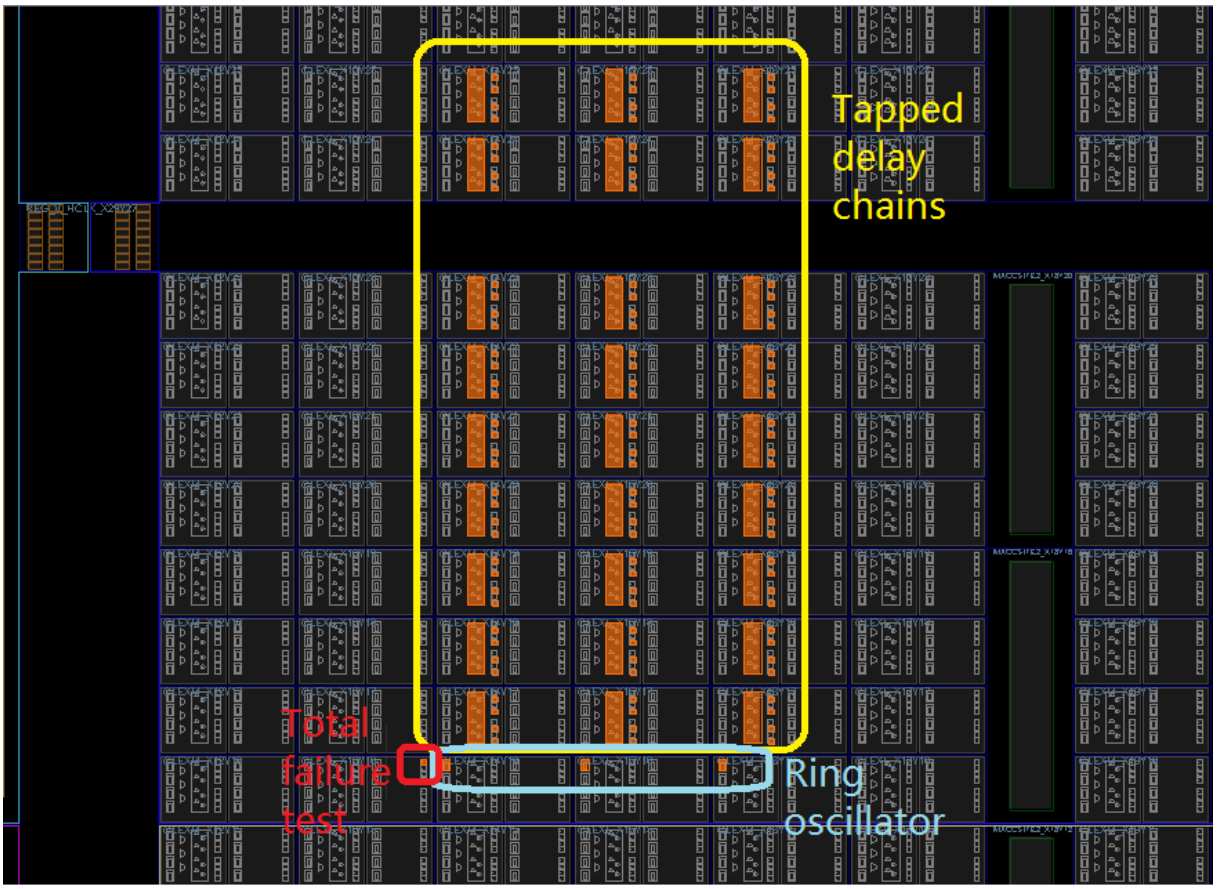

Figure 6: Floor plan of the HECTOR DC TRNG

The implementation of DC TRNG consumes 77 slices (183 LUTs and 125 registers) on Xilinx Spartan-6 FPGA.

2.3 Implementation of PUF in FPGA

Referring to the deliverable D2.1, we implemented the TERO PUF in FPGA. Its design is shortly described and discussed in the following subsections.

2.3.1 TERO PUF

A PUF extract process variation to generate a unique response as a *fingerprint* of the device. The TERO PUF generates this response by comparing the number of oscillations of TERO cells. The TERO PUF is configured as presented in Figure 7.

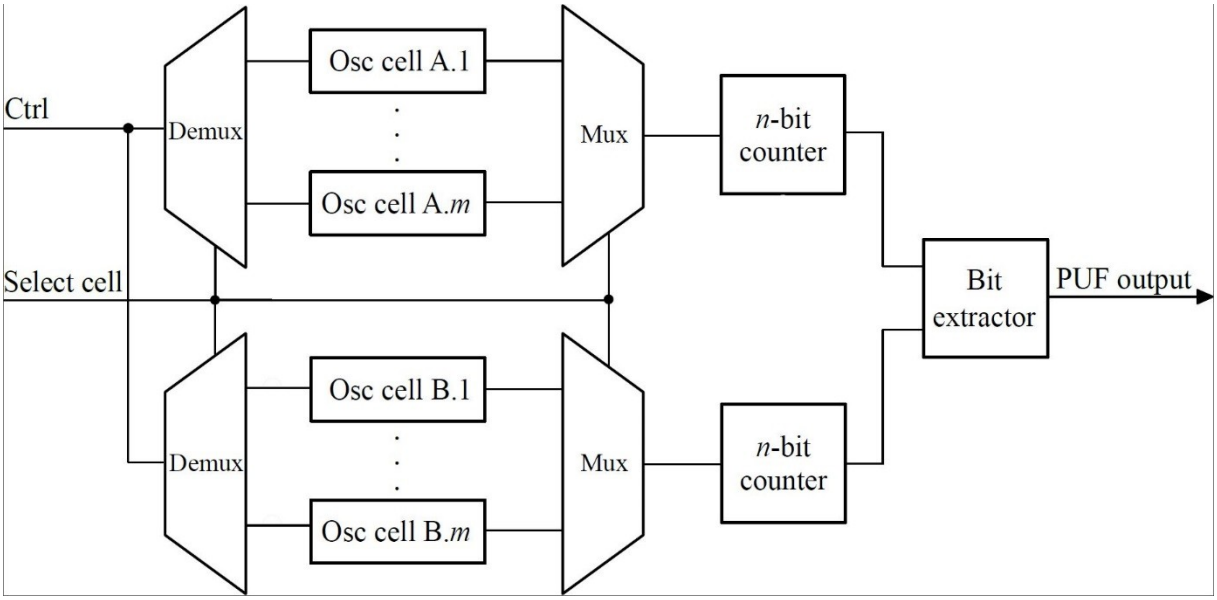

Figure 7: Architecture of the TERO PUF

The implemented PUF is composed of 128 oscillators (oscillating cells), two counters and a bit extractor as depicted in Figure 7. Cells are divided in two blocks, A and B. To avoid correlation, a cell of the block A is always compared to a cell of the block B and is used only once. One cell per block is selected using two de-multiplexers. Two multiplexers are placed right after the cell blocks in order to drive the correct cell outputs to the clock counters. The cell selection is usually called a challenge. Compared TERO cells are triggered at the same time.

We compare the number of oscillations at the output of two selected cells using a subtractor. Up to 2 bits per challenge can be extracted from the subtraction result. Counters and activation time of the control signal need to be sized according to the mean number of oscillations of the TERO cells. In our case, counters feature 11 bits and activation time is set to 1µs.

This is done for all implemented TEROs and the resulting bits are concatenated to form a 128-bits response. The response is stored in a shift register.

2.3.1.1 Implementation and implementation results

The TERO PUF was implemented in VHDL and mapped to the HECTOR SmartFusion2 daughter board. The implementation must be done carefully to avoid that the comparison of two cells depends on their position in the FPGA. Otherwise, some comparisons tend to be biased towards a certain value. Consequently, cells must have identical topology. For that, the first thing to do is to keep the cell structure unaltered during all the synthesis, place and route processes in the design workflow. This achieved by using a low level VHDL code.

Delays between gates in an FPGA have significant impact since many gates (or buffers) must be crossed to connect two elements. Consequently, elements of individual cells have to be rigorously placed side by side. This is achieved using the Chip Planner Constraint Manager or a Physical Design Constraint (pdc) file.

To ensure that no routing can cross oscillating cells, exclusive regions need to be created. They are physical regions on the FPGA where only assigned elements can be placed. To

avoid the routing to cross those regions, the “constrain routing” option must be selected. It is necessary to create one region per cell block.

The placement of the TERO cells is depicted in Figure 8.

Figure 8: Floor plan of the TERO PUF

TERO PUF implementation results in Microsemi SmarFusion2 FPGA are presented in Table 2. The following parameters characterize the area occupied by individual TERO PUF modules and by the whole PUF:

LUTs: Look-up tables

Registers: Dedicated registers

Module	LUTs	Registers
TERO cells	4460	30
Controller	38	35
Shift register	0	128
Total	4498	193

Table 2: TERO PUF implementation results

Chapter 3 Design of TRNGs and PUFs in ASIC

3.1 HECTOR ASICs

Two HECTOR-dedicated ASICs are designed in the framework of the HECTOR project. These chips will serve for TRNG and PUF characterization and testing. In the following sections, we describe each particular TRNG and PUF implementation, which will be included in these ASICs. Finally, we describe the contents of each ASIC as well as its interface.

3.1.1 Implementation of TRNGs in HECTOR ASIC

3.1.1.1 PLL TRNG

The ASIC implementation of the PLL TRNG is similar to the FPGA implementation. The differences come from different PLL parameters. While in FPGA there are usually multiple outputs available per each PLL, it is not the case in ASIC. The PLL, which was available for ASIC TRNG design has only one clock output. Therefore, the TRNG design was modified and sample counter (which replaces the decimator from the PLL TRNG design in FPGAs) was put directly after the sampling flip-flops.

In comparison to its FPGA counterpart the ASIC implementation of the PLL TRNG features only one data output. This output consists of 12 bits. The LSB of the output represents the random bit at the output of the TRNG. The whole 12-bit value can be used for jitter characterization.

The architecture of the ASIC implementation of the PLL TRNG is depicted in Figure 9.

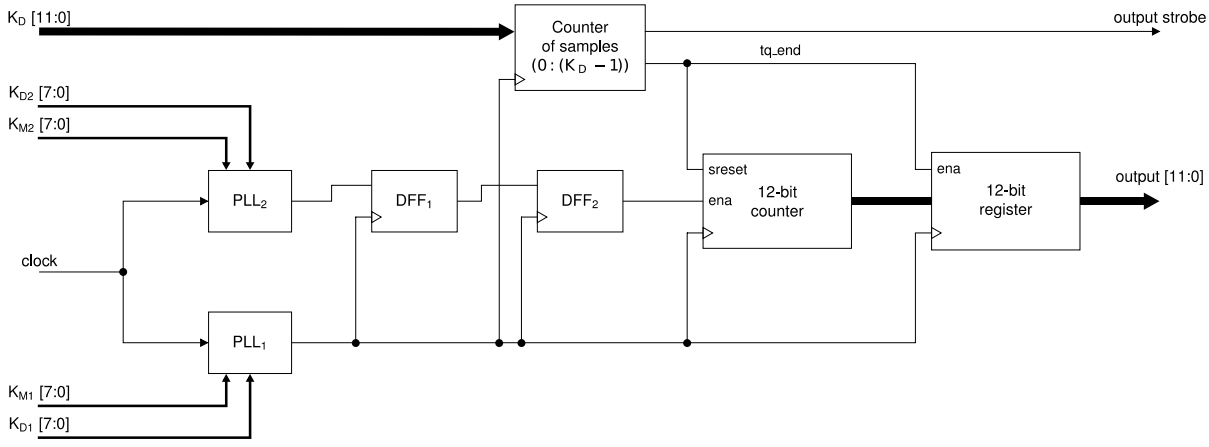

Figure 9: Architecture of the PLL TRNG in the HECTOR ASIC

Figure 10 depicts the layout of the PLL TRNG. It occupies the area of 325 µm x 750 µm.

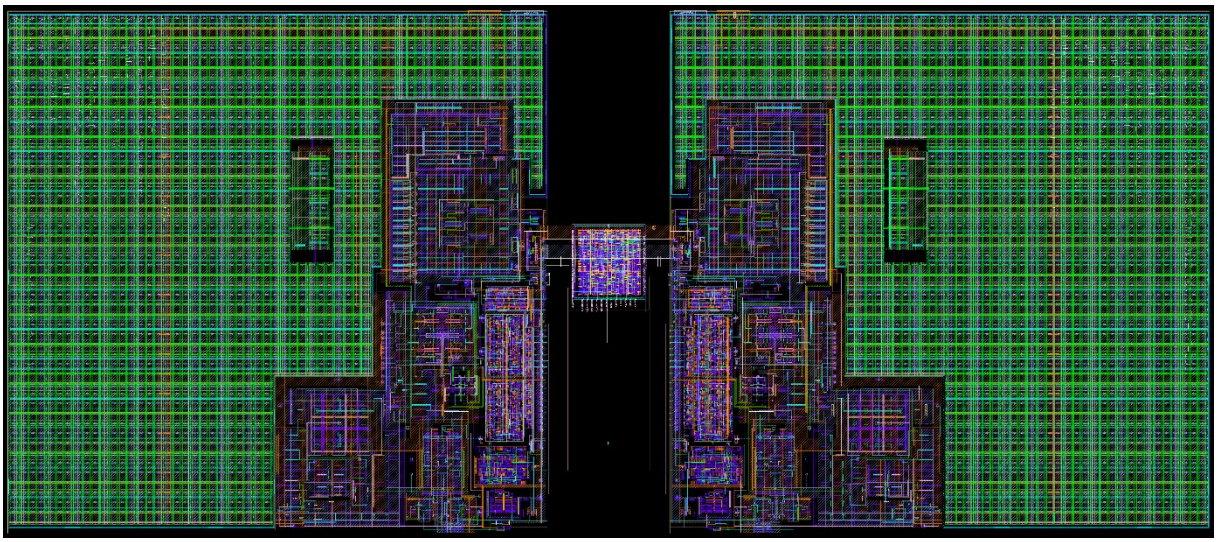

Figure 10: Layout of the PLL TRNG in the HECTOR ASIC

3.1.1.2 ELO TRNG

The elementary ring oscillator based TRNG (ELO TRNG) uses a jitter accumulated in ring oscillators to generate random bits. Figure 11 depicts the architecture of the ELO TRNG implemented in HECTOR ASIC.

The TRNG uses two ring oscillators (ROs), where one determines the sampling frequency and the second one is the sampled signal. One RO from Bank 0 (denoted RO 0.0 – RO 0.7 in Figure 11) serves as a noise source (it generates the sampled clock signal). The RO from Bank 1 (denoted RO 1.0 – RO 1.7 in Figure 11) is used, together with a 32-bit counter, to generate a sampling clock signal. The period of the sampling signal, and thus the jitter accumulation period, can be controlled by the preloaded value of *K*.

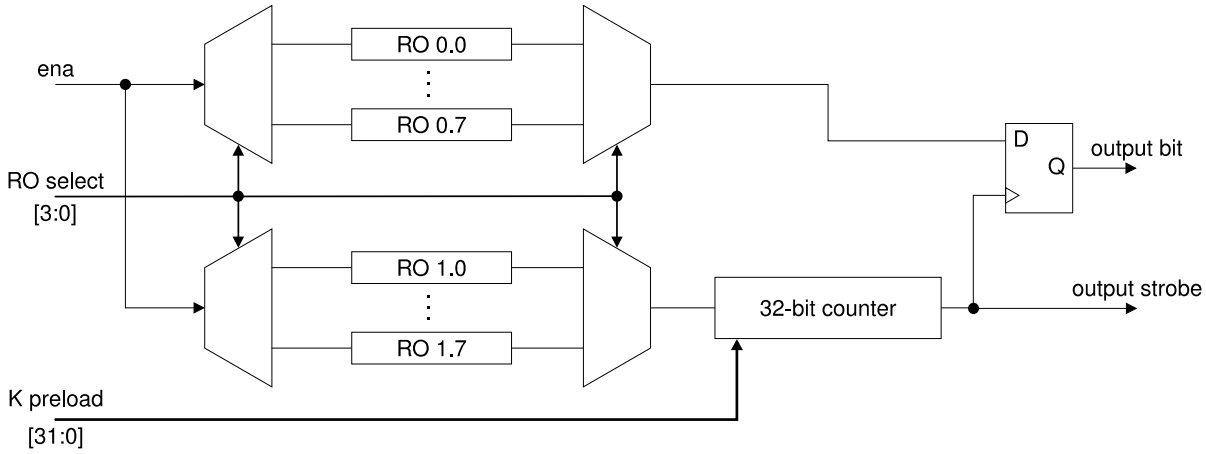

Figure 11: Architecture of the ELO TRNG in the HECTOR ASIC

We implemented the TRNG using 8 different ROs of different size to achieve 8 different frequencies. This approach allows for some flexibility in testing the TRNG since every RO has slightly different parameters. Before the delivery of ASIC we can only estimate the frequency of each RO (350 Mhz, 430 Mhz, 500 Mhz, 560 Mhz, 630 Mhz, 670 Mhzn 720 Mhz, 916 Mhz) using electrical simulations. Figure 12 shows the layout of the ELO TRNG ASIC implementation. It occupies the area of 42 µm x 90 µm.

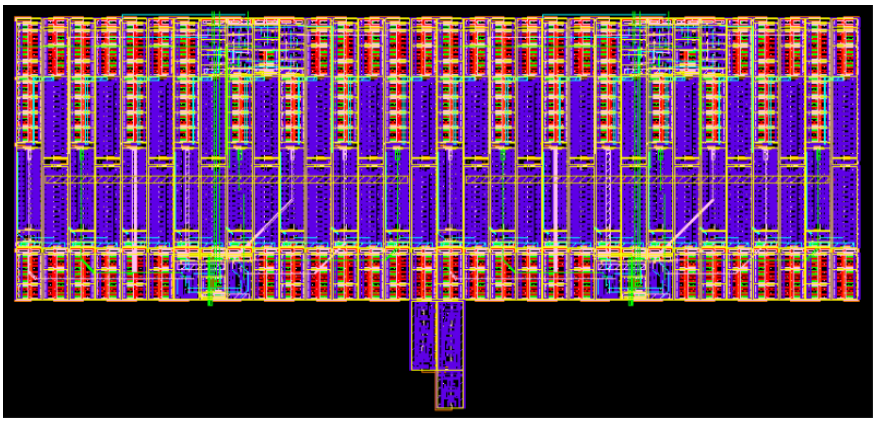

Figure 12: Layout of the ELO TRNG

3.1.2 Implementation of PUFs in HECTOR ASIC

3.1.2.1 TERO test modules

In order to characterize TERO cells and to evaluate the influence of the number of oscillations, we implemented 128 TERO cells with different configurations. Final objective is to establish a model to predict the mean number of oscillations of a TERO cell regarding its configuration. TERO cells are included by blocks of 8. By consequence we have 16 different configurations. To avoid correlations between cells, only one TERO cell is running at a time. We use a de-multiplexer to send the activation signal to the selected TERO cell and a multiplexer to send the correct TERO cell output. To maximize the signal quality, TERO cells are outputted on LVDS. Figure 13 depicts the layout of the TERO cells where we can clearly identify 16 implemented blocks.

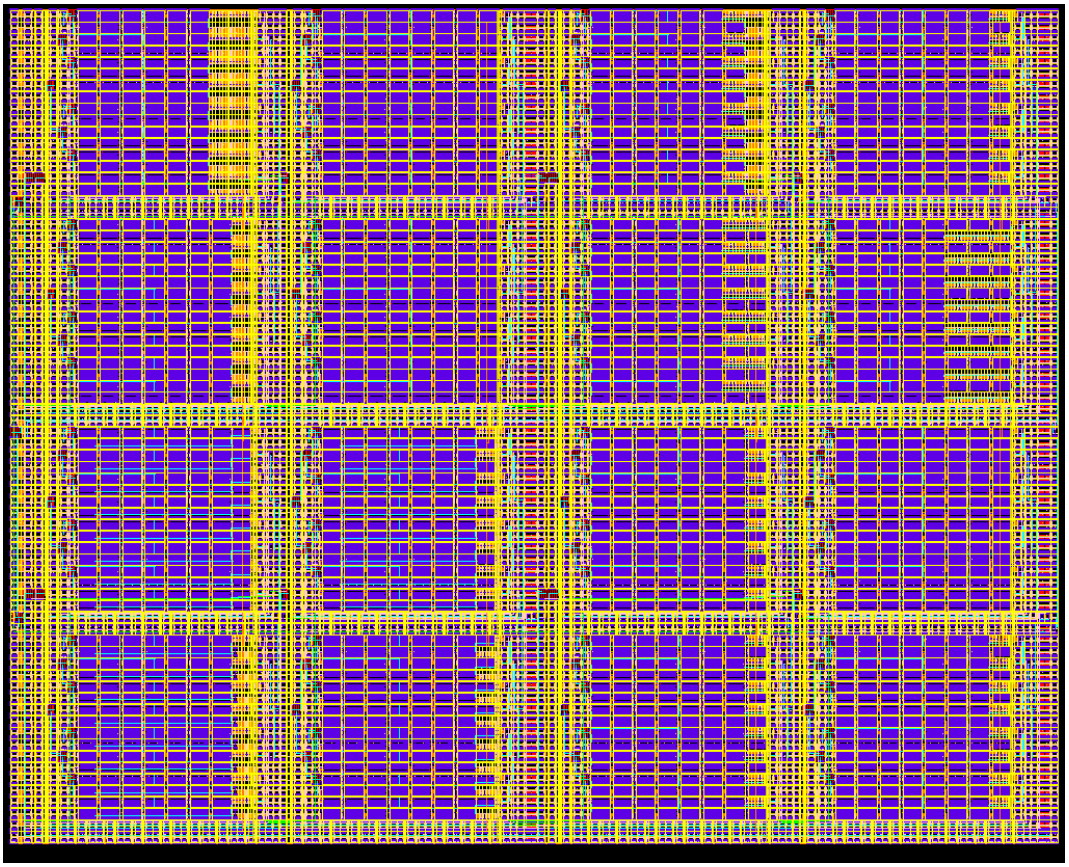

Figure 13: TERO test modules layout

3.1.2.2 TERO and RO based PUF

We implemented one TERO PUF and one RO PUF. Since those two structures are very similar and the only difference is that one structure is comparing the number of oscillations and the other the frequency of oscillation, we implemented them using the same counters.

Figure 14 depicts the structures of the PUFs. Both structures are composed of two blocks of 128 oscillating cells.

The TERO PUF principle is the same as that described in the FPGA implementation. The only difference is that the subtraction is done outside of the ASIC.

The RO PUF is comparing the oscillation frequency. Compared RO cells are triggered at the same time. As soon as one of the counters reaches a maximum value, an arbiter stops them. If it is the one from the block A, resp. block B, the arbiter generates a '1', resp. '0'. This is done for all implemented ROs and the resulting bits are concatenated to form the 128-bits response.

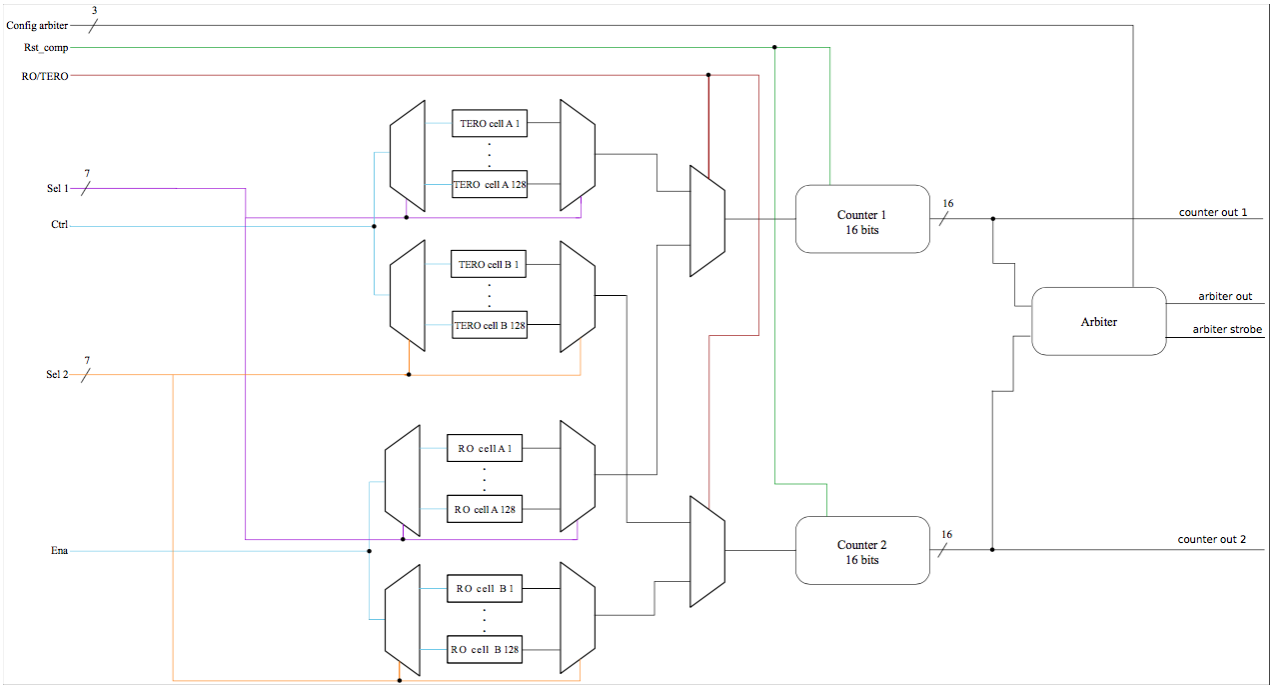

Figure 14: TERO and RO PUF structures

Counters are 16-bits and the arbiter is configurable. This means that we can choose which bit of the counters we compare. Figure 15 depicts the layout of the PUFs.

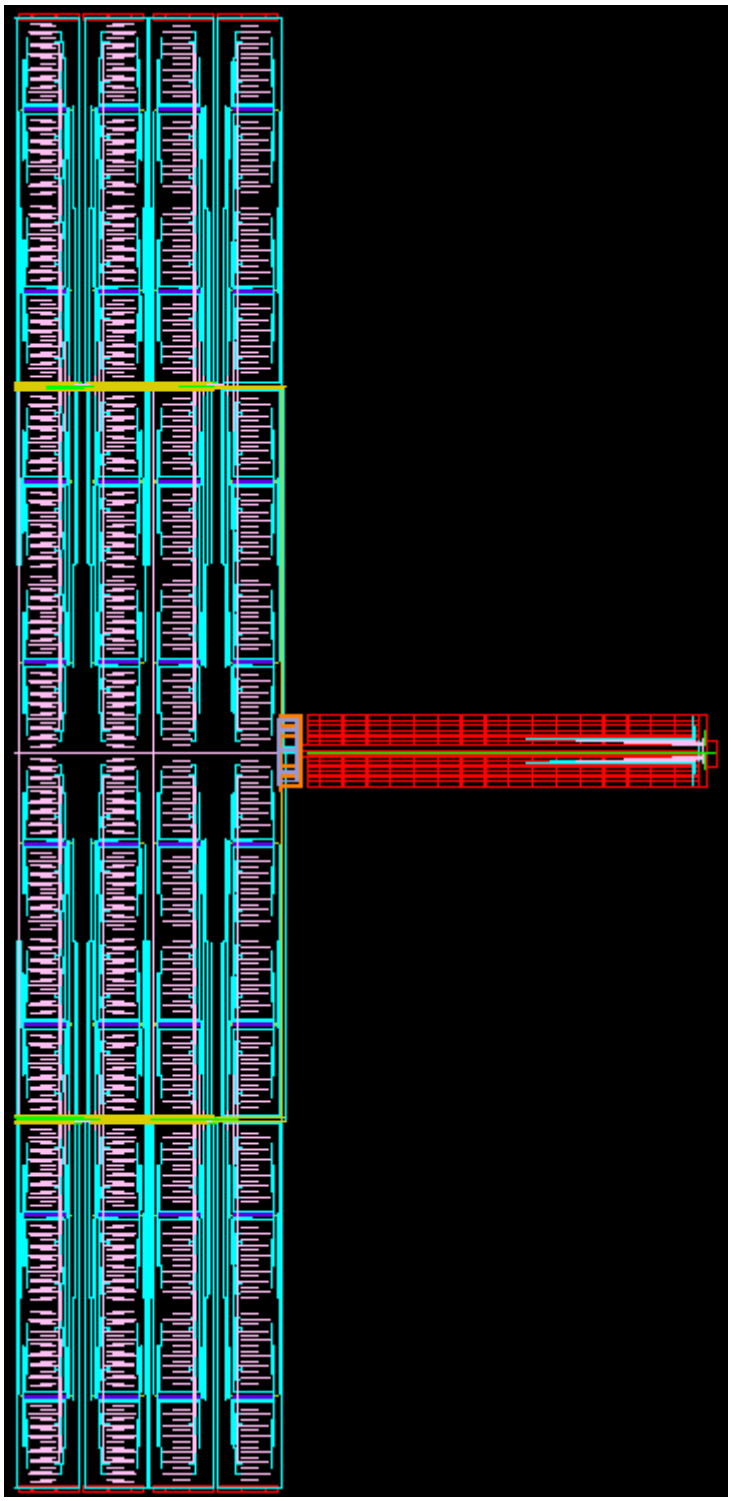

Figure 15: TERO and RO PUF layout

3.1.3 HECTOR ASIC #1 contents

A first HECTOR ASIC was submitted to manufacturing on February 20th 2017. This ASIC already contains the PLL TRNG design as well as test structures to start characterizing the TERO cell used in the HECTOR PUF design. Due to an earlier than expected (pulled-in) chip release schedule, the complete PUF design will be integrated later in the upcoming HECTOR ASIC #2 (next section). The expected delivery date for this first HECTOR ASIC is July 2017.

3.1.3.1 Interfaces

ASIC interfaces are designed with regard to the target application. Since the HECTOR ASIC is intended to be a research test chip focused on TRNGs and PUFs, we designed its interface to provide fast data outputs and a light-weight data input.

The inputs and outputs of the ASIC can be characterized as either dedicated to a particular block or shared within all the blocks. The dedicated inputs and outputs are:

- PLL clock input
- LVDS outputs for both PLLs
- LVDS output for TERO test modules

The shared inputs and outputs are:

- Global clock input
- Reset input
- 32-bit output bus
- Serial command input

Figure 16 shows the full pin-out of the HECTOR ASIC #1 with description of all pins.

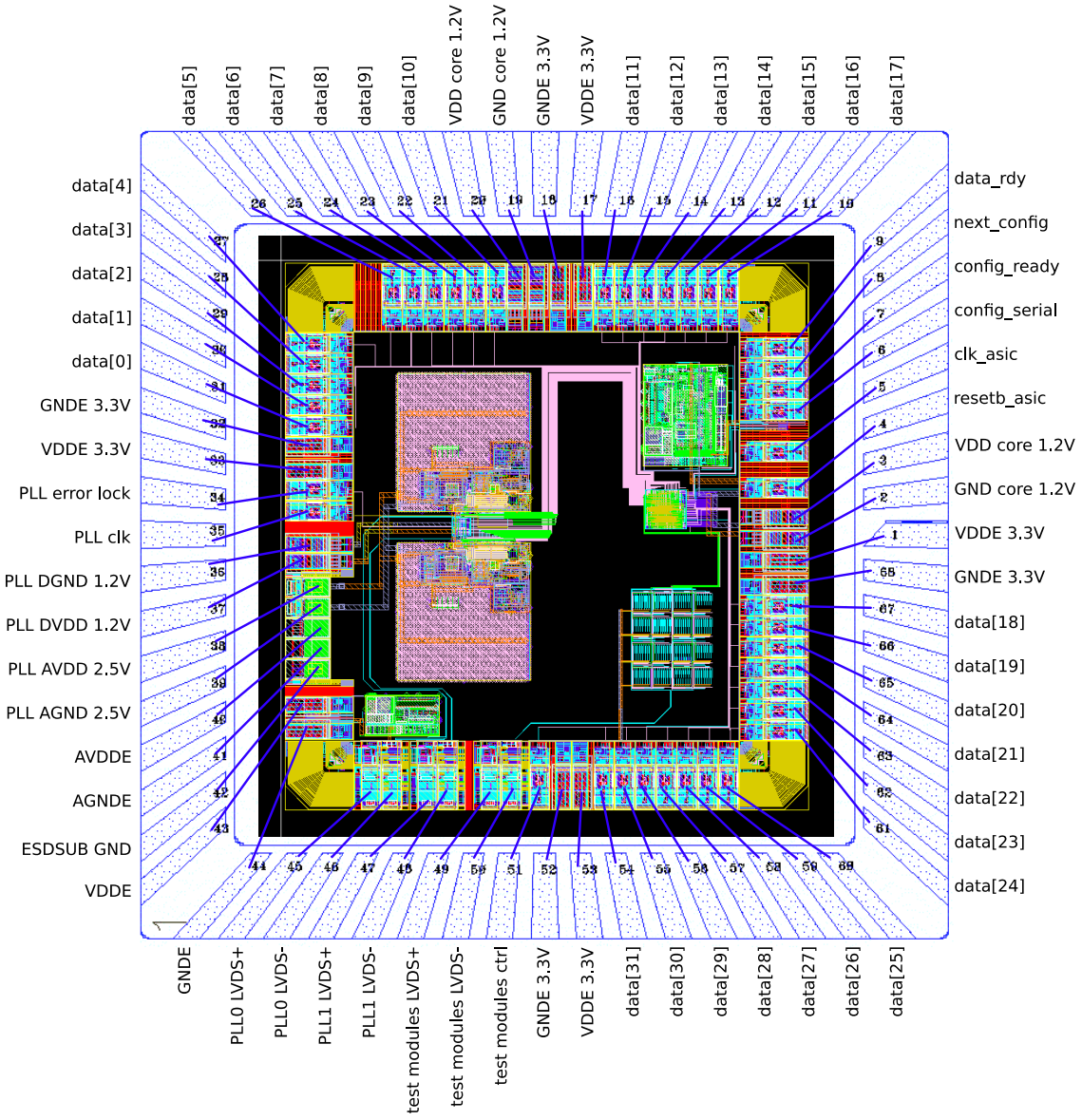

Figure 16: HECTOR ASIC pin-out

3.1.3.1.1 Data interface description

The HECTOR ASIC is controlled by commands sent using the *config_serial* and *config_ready* inputs. The command is an 88-bit long word, which is composed as follows:

- First 4 bits identify the block
- The rest of the word contains configuration data specific to each block

The output of the ASIC is 32-bit long data word which contains output data of the active block. Since only one block can be active at a time, the output can be easily shared between all of the core blocks.

Figure 17 shows the functional diagram of the data interface. Commands are read from the *config_serial* input by the shift register. Upon reception of the *config_rdy* signal, input is stored by the input latch. The command is then read and interpreted by the control logic, which then sends appropriate data to an appropriate core block. Output data is routed from the core block through the control logic. Control logic then sends output data to the ASIC output and generates a *data_rdy* and *next_config* signals. The *data_rdy* indicates that a new data word is available at the output. The *next_config* is used only for PUFs in order to indicate that a PUF is ready to process another challenge.

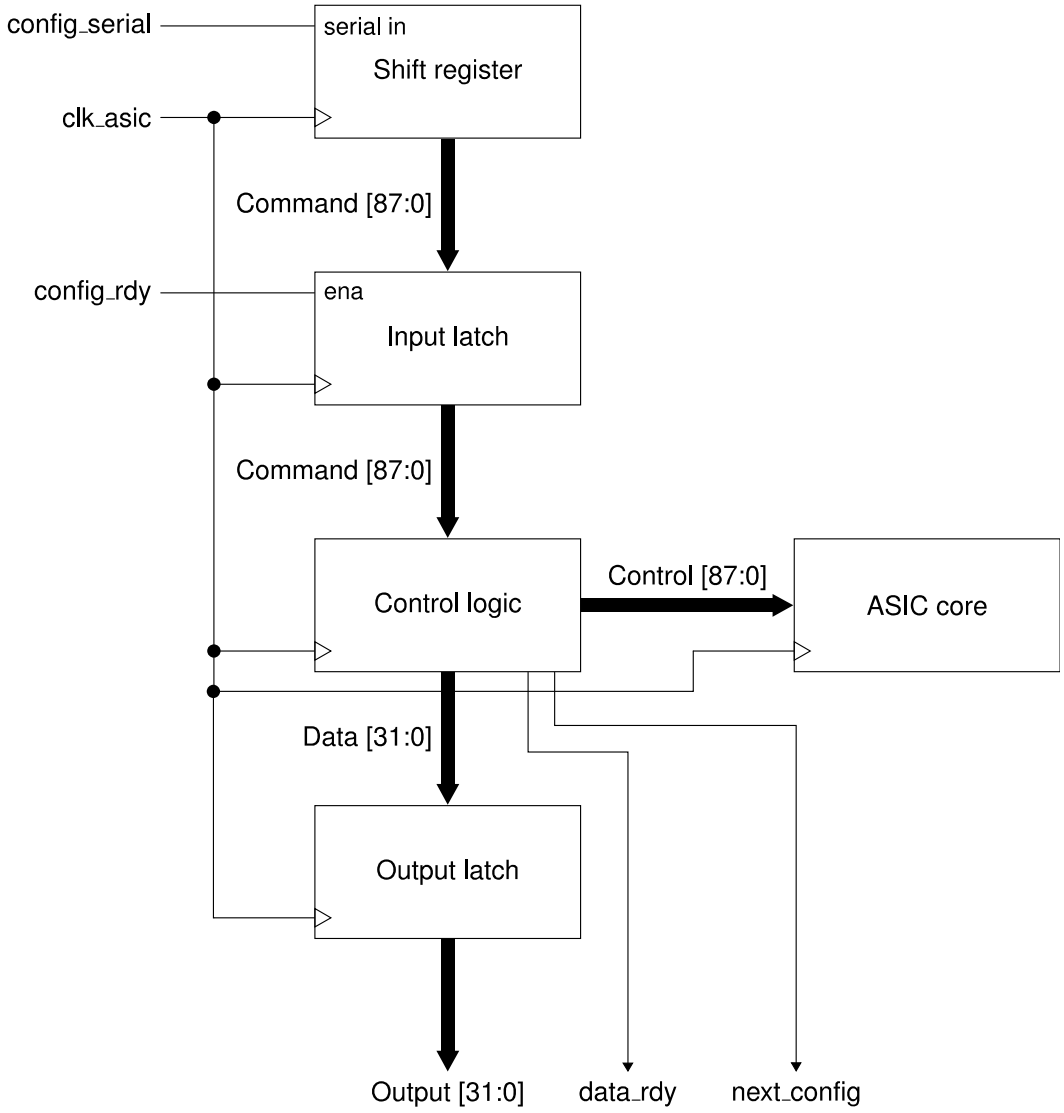

Figure 17: Functional diagram of HECTOR ASIC interface

3.1.4 HECTOR ASIC #2 contents

A second HECTOR ASIC with additional TRNG and PUF designs will be submitted for manufacturing in June 2017 with an expected delivery date in November 2017. This ASIC design features:

- PLL TRNG
- ELO TRNG
- RO PUF
- TERO PUF
- Updated TERO test modules

3.1.4.1 Interfaces

We reuse data interface of the HECTOR ASIC #1 in ASIC #2.

3.2 ST ASICs

Besides the HECTOR-dedicated and HECTOR-funded ASICs described in the previous section, we also had opportunities to get some HECTOR designs implemented into ST-internal ASICs. These are typically ST-internal test chips whose purpose is to validate new ST technology or product developments. These typically come with strong confidentiality and sample distribution restriction constraints. However, they do represent additional opportunities to see HECTOR primitives implemented on silicon and when successful could ease the path towards commercial exploitation in ST products.

3.2.1 ST ASIC #1

3.2.1.1 TERO TRNG in ST ASIC #1

The design is based around a TERO core (see deliverable D2.1) with an asynchronous counter counting the number of oscillations until the oscillator stops. A control register allows tuning adjustable TERO design parameters (delays). A status register allows detecting when the core stopped oscillating and a counter register allows reading the counter stop value.

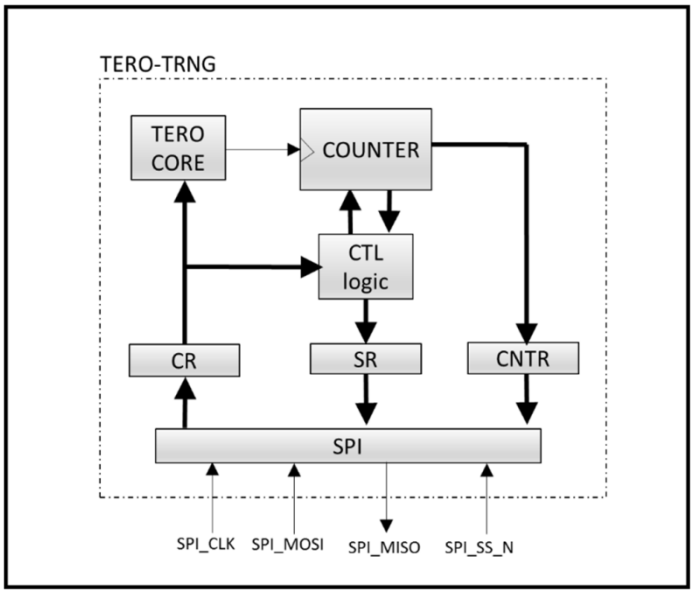

Figure 18: Architecture of the TERO-TRNG implementation in the ST ASIC #1

3.2.1.2 Interface

The interface was pre-defined (imposed) and is very simple. All communications are handled via a standard SPI interface, which uses the *SPI_CLK*, *SPI_POSI*, *SPI_MISO* and *SPI_SS_N* pins. Two additional pins are bonded but not used for TERO.

Several variants of the TERO core design have been implemented. Communication with these TRNGs is handled through registers accessed via the SPI:

- CRx control register (R/W)
  - tero_enable:
  - tero_start
  - tero_adj_sel
- SRx status register (RO)
  - stopl
  - stoph
- CNTRx counter stop value register (RO)

With "x" in CRx, SRx and CNTRx specifying which instance of TERO core is being addressed.

The generator is started by setting bits *tero_enable* and *tero_start* in the *CR* control register. TERO design parameters (delays) can be adjusted via the *tero_adj_sel configuration* bits.

Checking bits *stopl* or *stopr* from status register SR allows to check if the TERO has stopped and in which logical state.

The value of the counter at which the TERO stopped can be read from the *CNTR* registers.

The output bit (random) is the parity of the value at which the counter stopped.

The count value itself can be exploited to make statistics and run online tests. This is not done on-chip for this test chip but on a host PC computer. In a final instantiation, this could be done on-chip.

Pin #	Signal
4	CLK
6	NRST
7	VDD
10	VCC
13	Not used
14	Not used
16	GND
18	SPI_CLK
19	SPI_MOSI
20	SPI_MISO
21	SPI_SS_N

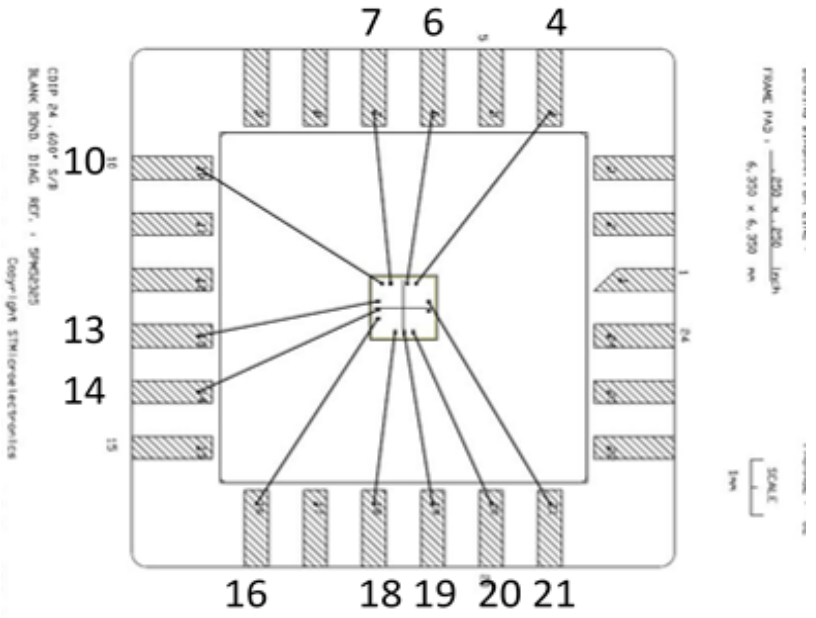

Figure 19: Pin-out of the STR ASIC #1.

3.2.1.3 Implementation

Several variants of the TERO core design allow comparing different ways to implement and control the delay elements from the TERO core.

The design has been implemented in Verilog and synthesized for the target 40nm ST silicon technology. Three iterations of this ASIC have been manufactured, which allowed fine-tuning some of the design parameters between iterations.

Below is a table summarizing the silicon area of the six TERO structures in the last test chip. It is worth noting that these structures are extremely small.

Tero 1	Tero 2	Tero 3	Tero 4	Tero 5	Tero 6
1100 µm²	2200 µm²	1750 µm²	1700 µm²	3100 µm²	1600 µm²

Table 3: Silicon cost for the different TEROs in ST ASIC #1

The “back-end” of the chip designs (layout, I/O-Ring, etc.) has been performed by an ST team not involved in HECTOR. Below is a picture of the ASIC assembled within a DIL test chip package.

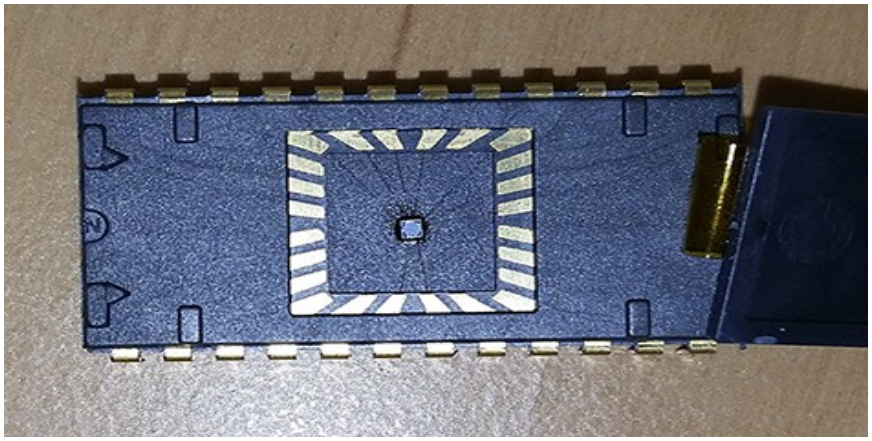

Figure 20: Picture of ST ASIC #1 in DIL test package

3.2.2 ST ASIC #2

We also seized an additional opportunity to design and integrate a PLL-based TRNG (see D2.1) in an ST-internal system-on-chip ASIC.

3.2.2.1 PLL TRNG in ST ASIC #2

The principle remains the same as for the PLL-based TRNG designs in FPGA and HECTOR ASICs. The differences are related to usage of a more aggressive silicon technology node, different PLLs with specific available division/multiplication factors and different jitter properties, and to the constraints related to PLL1 being imposed and not dedicated to the TRNG. PLL1 is a “system” PLL whose primary function is to generate the different clocks needed by the system-on-chip, from the various crystal oscillators’ configurations that the circuit needs to support. The sampling flip-flop is followed by a configurable decimator which allows accumulating entropy from the unstable, entropy-carrying samples, over as many periods as needed to reach the target level of entropy. The decimator is followed by memory which holds the output bits being tested by the dedicated online tests. They are released to the post-processing block once the dedicated online test results are successful. Diagnosis modes (not depicted in the above diagram) allow extracting bits at various stages of the

process for debug and characterisation purposes. The TRNG subsystem is addressed through control, status and result registers accessible from the processor of the system-on-chip.

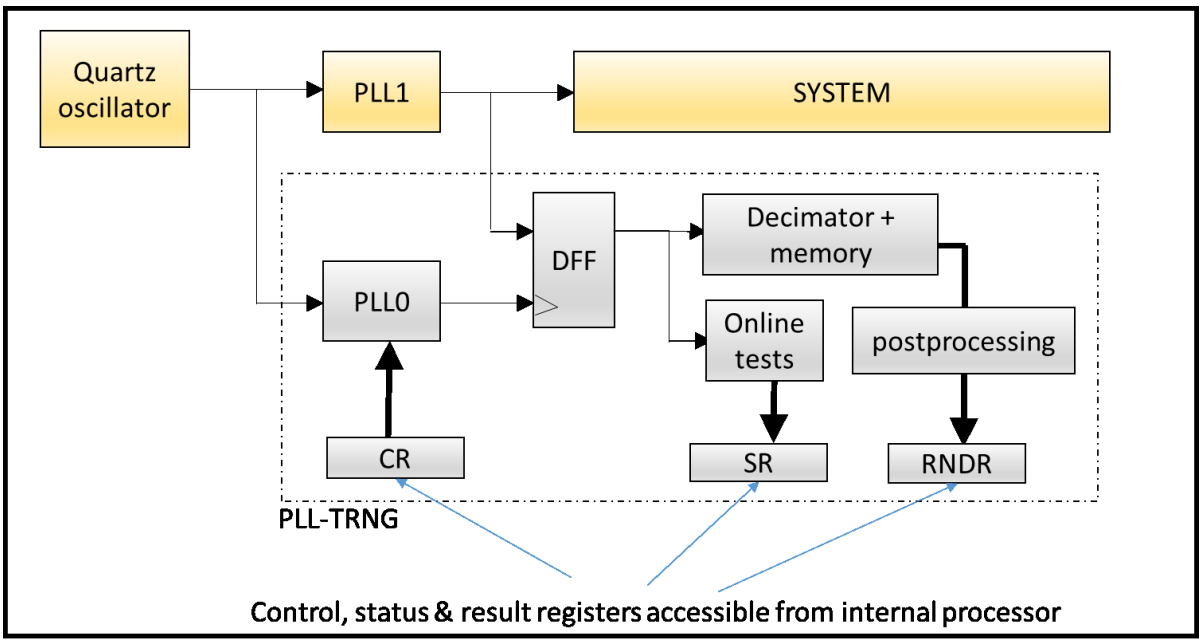

Figure 21: Architecture of the PLL-based TRNG in ST ASIC #2

3.2.2.2 Interface

In contrast to the HECTOR & ST ASIC #1, The TRNG isn't directly accessible from via the chips' input/output pins, but rather through one of the processors inside the chip. This is a complex, applicative system-on-chip circuit whose pin-out and interfaces aren't really relevant here and can't be detailed due to confidentiality constraints.

3.2.2.3 Implementation

The design has been implemented in Verilog and synthesized for the target 40 nm ST silicon technology. The table below summarizes the silicon area costs for the digital part of the design, as well as the surface for the memory and PLL0 (hard macros). The surface of the system PLL is not reported since it is not dedicated to the PRNG. It is worth noting that the silicon cost is dominated by the PLL and 50 to 100 times higher than TERO.

	Digital logic	RAM	PLL 0
Silicon cost	5'500 µm²	5'900 µm²	150'000 µm²

Table 4: Silicon cost for the different elements of the PLL-TRNG in ST ASIC #2

The design will be integrated in a large system-on-chip circuit managed by an ST team not involved in HECTOR. This chip should be released to manufacturing towards the end of the second quarter and we hope to receive samples around fall.

Chapter 4 Summary and Conclusion

This document provided a short, public summary of the D2.2 design deliveries from HECTOR WP2, which is the HECTOR work-package focusing on TRNGs and PUFs.

While FPGA designs can be produced and tested in a software-like “iterative mode”, the design and manufacturing of ASICs is a much longer and costlier process. We already received and evaluated TERO TRNG ASIC designs implemented in ST-internal test chips, and are now eagerly waiting for the HECTOR ASICs that we expect to receive from manufacturing during the second half of this year, as well as a new ST ASIC that we should receive in a similar time-frame. These will allow to measure performances and perform characterizations on real-silicon and to compare versus our modelling and simulation results.

FPGA instantiations of the PLL and DC TRNG designs, together with documentation have already been delivered to Brighsight for security evaluation. These allow starting “pen and paper” security analysis as well as lab-level security characterization efforts.

We acknowledge and agree with recommendation R3 from the expert report of the October 5th 2016 review, which stressed the importance of proper robustness assessments and comparisons of HECTOR WP2 designs. These important activities will continue until the end of the project (and in some cases only start once we receive ASICs back from manufacturing) and therefore were not the focus of D2.2 or this accompanying report.

Chapter 5 List of Abbreviations

ASIC	Application Specific Integrated Circuit
DC TRNG	Delay Chain TRNG
ELO TRNG	Elementary Oscillator TRNG
FPGA	Field Programmable Gate Array
PLL	Phased-Lock Loop
LVDS	Low Voltage Differential Signalling
PUF	Physically Unclonable Function
RO	Ring Oscillator
TERO	Transition Effect Ring Oscillator
TRNG	True Random Numbers Generator