DOI: 30.0485/ijearst.05.02.3478 https://ijearst.co.in/

ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2386-7876
Vol.01, Issue.03
March-2026
Pages: 880-891

EXPLORING THE USAGE OF FAST CARRY CHAINS TO IMPLEMENT MULTISTAGE RING OSCILLATORS ON FPGAS:DESIGN AND CHARACTERIZATION

[1]M. HIMALATHA, [2]MATTAPARTHI SAHITHI, [3]YANAMADALA SANDHYA, [4]RONGALA RAMA DEVI, [5]POTHABATHULA MANOJ KUMAR, [6]UPPUGANTI MURALI MUKTHI LINGAM

[1]Assistant professor, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP

[2,3,4,5,6]B. Tech, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP

ABSTRACT: Ring oscillators (ROs) serve as basic building blocks in a lot of application scenarios, where they must ensure high reliability, flexibility, and low-area/energy footprint. With the recent advances of the Internet-of-Things (IoT) technology, in particular, the necessity to endow interconnected devices with security facilities has increased as well. In this context, the efficient implementation of ROs on field-programmable gate arrays (FPGAs) is crucial, even though it hides some pitfalls. This article presents a new design strategy for multistage ROs relying on the carry chains (CCs) available into modern FPGA devices. Several configurations of ROs designed as proposed here have been characterized in terms of hardware costs, jitter, and temperature/voltage sensitivity.

Keywords: Fast Carry Chains, Ring Oscillators, FPGA Design, Multistage Oscillator, Carry Chain Delay, High-Speed Oscillation, Timing Characterization, Frequency Analysis, Hardware Optimization, Low-Latency Circuits, Digital Oscillators, FPGA Architecture, Delay Elements, Process Variation, Power Consumption, Phase Noise, On-Chip Clock Generation.

INTRODUCTION: Computer systems and telecommunications play an important role in modern world technology. The communication and data transfer through computers touches almost every aspect of life, i.e. transferring data, tracking personal data, trading over the internet, online banking and sending emails. As more vital information is transferred through wire or wireless means, the need to safeguard all this data from hackers is growing. All these security concerns emphasize the importance of developing methods and technology for the transformation of data to hide its information content, prevent its modification, and prevent unauthorized use. Random number generation is a fundamental process for protecting the privacy of electronic communications. It is a key component of the encryption process that protects information from attackers by making it unreadable without the proper decryption process. Since the strength of an encryption mechanism is directly related to the randomness of the binary numbers used in it, there has been an enormous need to design and develop an efficient random number generator that can produce true random numbers to implement a safe and secure cryptographic system. In addition to cyber security, random number generators (RNGs) are a vital ingredient in many other areas such as computer simulations, statistical

sampling, and commercial applications like lottery games and slot machines. Random numbers are needed in some areas in computer science, such as authentication, secret key generation, game theory, and simulations. In these applications, particularly numbers should have good statistical properties and be unpredictable and nonreproducible. The number generation in the literature is performed in two different ways as deterministic and nondeterministic [1, 2]. PRNGs (Pseudo Random Number Generators), which are deterministic random number generators, generate numbers with fast, easy, inexpensive, and hardware independent solutions. The statistical qualities of these numbers produced are close to the ideal. PRNGs must meet the requirements specified in Table 1 to be used especially for authentication and key generation [3–5]. Therefore, nondeterministic functions are added to the output functions of PRNGs to guarantee these requirements. TRNGs (True Random Number Generators), which are nondeterministic random number generators, present slower, more expensive, and hardware-dependent solutions compared to PRNGs. Contrary to PRNGs, there is no need to include extra components in the TRNG system designs for R2, R3, and R4 requirements. Because of the unpredictability of random numbers generated by the use of high noise sources with high entropy in TRNGs, it is assumed that the R2 requirement is met. If the R2 requirement is satisfied, then it is assumed that the R3 and R4 requirements are also satisfied. To meet the R1 requirement in TRNGs, postprocessing techniques are applied on the random numbers obtained by sampling from noise sources.

LITERATURE REVIEW: In an irreversible circuit, if one bit information is lost then at least KTln2 joules of energy is dissipated. Where K is Boltzmann's constant and T is absolute temperature. This was stated by Landauer R in 1961[1]. In 1973, Bennett [2] proved that, KTln2 joules of energy dissipated due to information loss in irreversible circuit can be controlled by reversible logic where the reversible circuit allows to reproduce the inputs from output resulting in no information loss. He also showed that reversible systems can do the same computations as the classical or irreversible systems at same efficiency. This leads to the evolution of reversible logic based systems. Any reversible gate should have equal number of inputs and outputs such that, inputs can be recovered uniquely from outputs at any point of time. In paper [3], by Shibinu A.R , Rajkumar, a 4- bit LFSR design using Muller expression is proposed. This paper also gives realization of both edge triggered and level triggered D flip flop using reversible logic. At the end, comparative analysis has been given between conventional LFSR and Reversible LFSR. From this it is observed that, the proposed technique is efficient than conventional technique in implementing LFSR in terms of cost metrics like power, quantum cost, garbage output and gate count. D. Muthih and A. Arockia Bazil Raj [4] have presented a parallel architecture for designing high speed LFSR and explained that, BCH encoders and CRC operations are normally carried out by using LFSR. A novel approach for high speed BCH encoder is proposed. This paper presents two key points. First, it presents a linear transformation algorithm for converting a serial LFSR into parallel architecture, which can be used for generating polynomials in CRC and BCH encoders. Secondly, a new approach is proposed to amend parallel LFSR into pipelining and retiming algorithm. In paper [5], authors have presented two design approaches for designing reversible D FF with asynchronous set/reset which are optimized in terms of quantum cost, delay and garbage outputs. It also includes the design of 3-bit LFSR using two design approaches. The application of these FF's as LFSR is designed and discussed. The application of LFSR as pseudo random bit sequence generator is proposed. The paper is concluded with the comparative analysis of proposed approaches against

performance parameters like garbage output, delay and quantum cost. Research paper [6], presents three different automated techniques for implementing LFSR as well as D flip flop so that the layout area and power consumption will be minimized. It is illustrated that LFSR is key component to provide self-test of an Integrated Circuit (IC). This paper implements LFSR up to layout level which will be a key component for low power application. The research explores the LFSR as well as D flip flop using different architecture in a 0.18µm CMOS technology; so that the layout area will be minimized and consumes less power.

PROGRAMMABLE DELAY LINES (PDLS):

EXISTING TECHNIQUE:

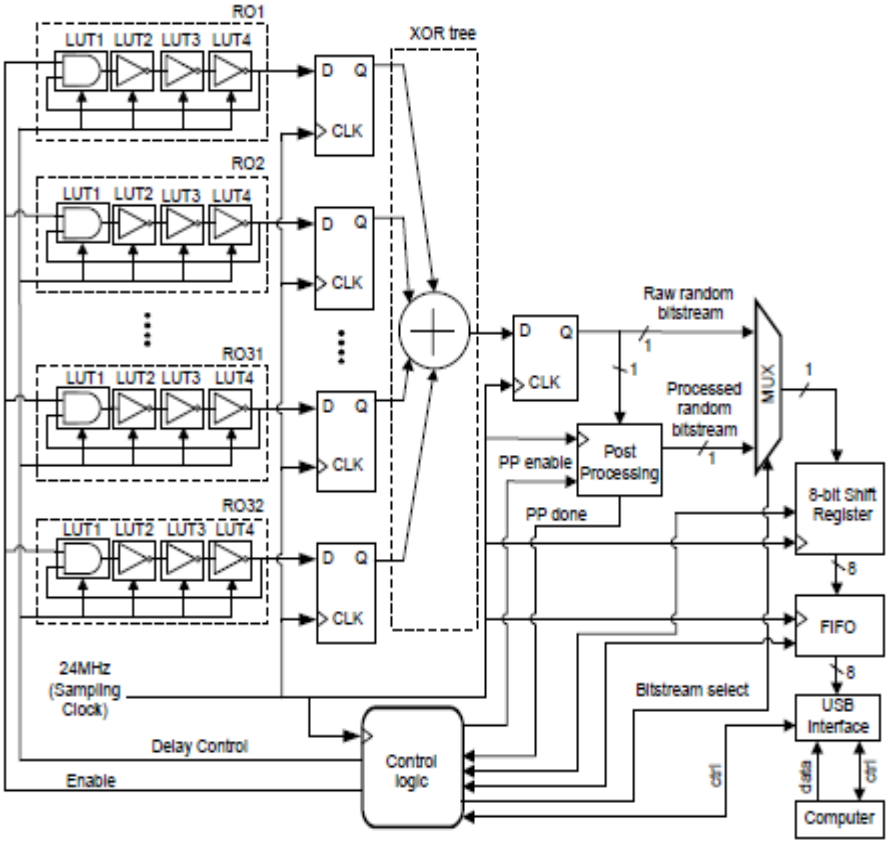

Fig:1 Architecture of the proposed TRNG.

It is imperative to note that the RO based TRNGs, although exciting, are extremely limited in terms of randomness when identical RO's are employed [14]. Equal length oscillator rings configured in FPGA are highly correlated with each other due to identical delays and therefore the XOR of the output from these rings returns mostly zeros. This leads to poor randomness from the design. We show in this work that the a TRNG incorporating PDL's can overcome this problem. Previously, the metastability of flip-flops was used for generating true random numbers [6]. They achieved metastability by using PDLs that accurately equalize the signal arrival times to flipflops. On the other hand, our work uses random jitter of free running oscillators for generating true random numbers. We employ the PDL's in oscillator rings to generate large variation of the oscillations and to introduce jitter in the generated RO's clocks. The main advantage of the proposed TRNG utilizing PDL's is to reduce correlation between several equal length

oscillator rings. For example, this can be achieved by variable RO outputs for each sampling clock by incorporating PDL as shown in Fig. Moreover, the variation in RO oscillations from cycle to cycle (CTC) is also introduced by each oscillator ring due to inverter-delay. As a result, the XOR operation significantly improve the randomness qualities.

PROGRAMMABLE DELAY LINES (PDLS): A programmable delay line allows a user to programmatically specify and modify delay in a high frequency/RF signal path. This results in a signal delay, as measured in the time domain, or a phase shift, as measured in the frequency domain. Precision delay is introduced into the high frequency signal path (or transmission line) by the expansion or compression of the patented electromechanical trombone. A digital delay line is a discrete element in digital filter theory, which allows a signal to be delayed by a number of samples. If the delay is an integer multiple of samples, digital delay lines are often implemented as circular buffers. This means that integer delays can be computed very efficiently. The delay by one sample is notated $z-1$ and delays of N samples is notated as $z-N$ motivated by the role the z-transform plays in describing digital filter structures. If a delay is not an integer of a sample additional filters are applied to account for the fraction of delay different from an integer. Hence delay lines with non-integer delay are called fractional delay lines.[1] Digital delay lines were first used to compensate for the speed of sound in air in 1973 to provide appropriate delay times for the distant speaker towers at Summer Jam at Watkins Glen in New York, with 600,000 people in the audience. New Jersey company Eventide provided digital delay devices each capable of 200 milliseconds of delay. Four speaker towers were placed 200 feet (60 m) from the stage, their signal delayed 175 ms to compensate for the speed of sound between the main stage speakers and the delay towers. Six more speaker towers were placed 400 feet from the stage, requiring 350 ms of delay, and a further six towers were placed 600 feet away from the stage, fed with 525 ms of delay. Each Eventide DDL 1745 module contained many 1000-bit shift register integrated chips, and cost the same as a new car.[2] Digital delay lines are widely used building blocks in methods to simulate room acoustics, musical instruments and effects units.[3] Digital waveguide synthesis shows how digital delay lines can be used as sound synthesis methods for various musical instruments such as string instruments and wind instruments. The internal variations of FPGA Look-Up Tables (LUT's) can be generated from changes in the LUT's propagation delays under different inputs [6]. For example, the LUT in An analog delay line is a network of electrical components connected in cascade, where each individual element creates a time difference between its input and output. It operates on analog signals whose amplitude varies continuously. In the case of a periodic signal, the time difference can be described in terms of a change in the phase of the signal. One example of an analog delay line is a bucket-brigade device.[1] Other types of delay line include acoustic (usually ultrasonic), magnetostrictive, and surface acoustic wave devices. A series of resistor–capacitor circuits (RC circuits) can be cascaded to form a delay. A long transmission line can also provide a delay element. The delay time of an analog delay line may be only a few nanoseconds or several milliseconds, limited by the practical size of the physical medium used to delay the signal and the propagation speed of impulses in the medium. Analog delay lines are applied in many types of signal processing circuits; for example the PAL television standard uses an analog delay line to store an entire video scanline. Acoustic and electromechanical delay lines are used to provide a "reverberation" effect in musical instrument amplifiers, or to simulate an echo. High-speed

oscilloscopes used an analog delay line to allow observation of waveforms just before some triggering event. With the growing use of digital signal processing techniques, digital forms of delay are practical and eliminate some of the problems with dissipation and noise in analog systems Before there was random access memory, there was delay line memory. It was random in a different sense; it involved turning electrical pulses into sound waves, sending them through long tubes of mercury, and re-electrifying them at the other end. The technology had its roots in the work of engineer J. Presper Eckert, who developed line delay systems to improve radar during World War II. Instead of storing data in individual bits, it was compressed down to sound waves and sent through a medium that slowed them down (initially mercury, then other substances, and finally wire). At the other end, they were re-electrified, processed, and then sent back through the tube. Because of slowing that occurred in the tube's substance, hundreds of pulses of data could be sent in a single tube—hence the name "delay line"—bouncing back and forth until they were needed.

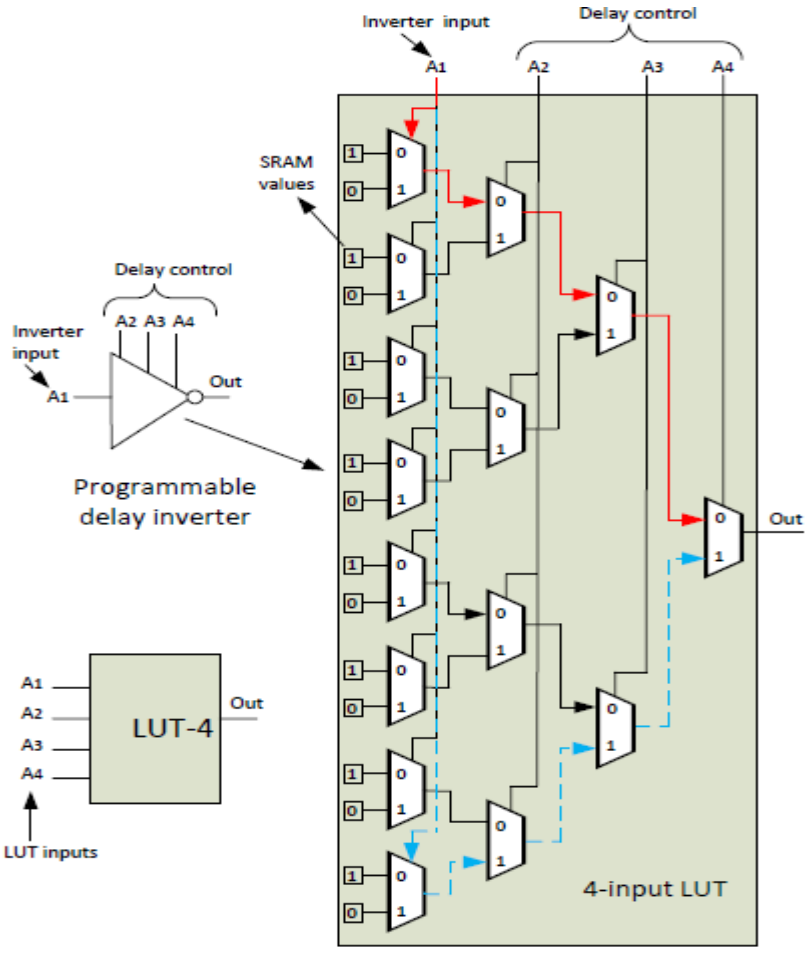

Fig. 2: PDL using a 4-input LUT.

Fig. 2 is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (A1). Other inputs A2, A3, and A4 act as “don’t-care” bits but their values affect the signal propagation path from A1 to the output (0). In this context, it has been shown, in Fig. 1, that the signal propagation path from A1 to the output (0) is shortest for A2A3A4 = 000 (marked with solid red line) and longest for A2A3A4 = 111 (marked with dashed blue lines) for 4-input LUT’s. Thus, a programmable delay inverter with three control inputs can be implemented by using

one LUT. For the PDL, the first LUT input A1 is the inverter input and the rest of the LUT inputs (three) are controlled by 23 = 8 discrete levels.

LOOK UP TABLE: In computer science, a lookup table is an array that replaces runtime computation with a simpler array indexing operation. The savings in processing time can be significant, because retrieving a value from memory is often faster than carrying out an "expensive" computation or input/output operation.[1] The tables may be precalculated and stored in static program storage, calculated (or "pre-fetched") as part of a program's initialization phase (memoization), or even stored in hardware in application-specific platforms. Lookup tables are also used extensively to validate input values by matching against a list of valid (or invalid) items in an array and, in some programming languages, may include pointer functions (or offsets to labels) to process the matching input. FPGAs also make extensive use of reconfigurable, hardware-implemented, lookup tables to provide programmable hardware functionality. Before the advent of computers, lookup tables of values were used to speed up hand calculations of complex functions, such as in trigonometry, logarithms, and statistical density functions.[2] In ancient (499 AD) India, Aryabhata created one of the first sine tables, which he encoded in a Sanskrit-letter-based number system. In 493 AD, Victorius of Aquitaine wrote a 98-column multiplication table which gave (in Roman numerals) the product of every number from 2 to 50 times and the rows were "a list of numbers starting with one thousand, descending by hundreds to one hundred, then descending by tens to ten, then by ones to one, and then the fractions down to 1/144"[3] Modern school children are often taught to memorize "times tables" to avoid calculations of the most commonly used numbers (up to 9 x 9 or 12 x 12). Early in the history of computers, input/output operations were particularly slow – even in comparison to processor speeds of the time. It made sense to reduce expensive read operations by a form of manual caching by creating either static lookup tables (embedded in the program) or dynamic prefetched arrays to contain only the most commonly occurring data items.

PROPOSED METHOD:

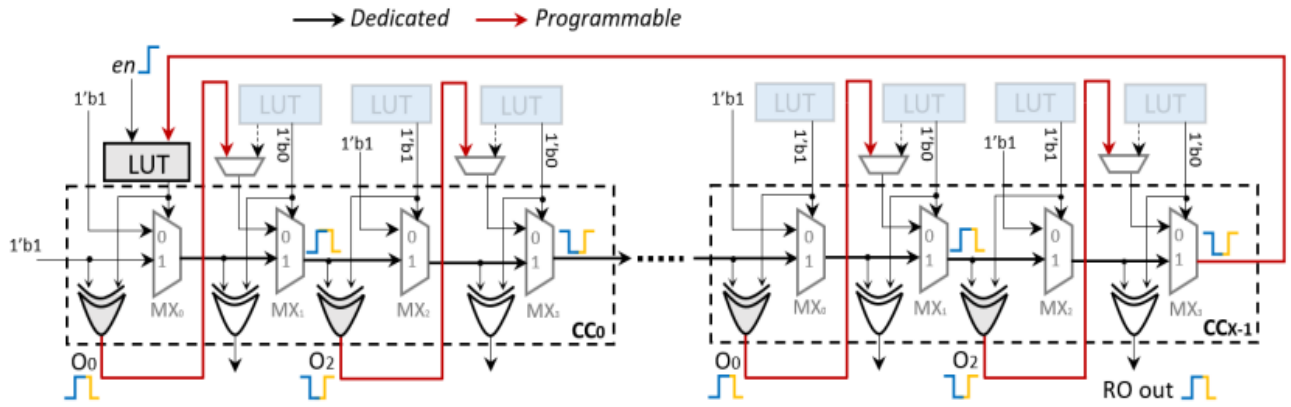

Fig. 3. Proposed CC-based multistage RO design (CC_ROv1 configuration). Inverting stages are realized through the XOR gates and the LUT highlighted in gray. Shadowed LUTs are exploited to permanently set the selectors of corresponding multiplexers. Forward propagation of the oscillation signal relies on the chain of MX elements

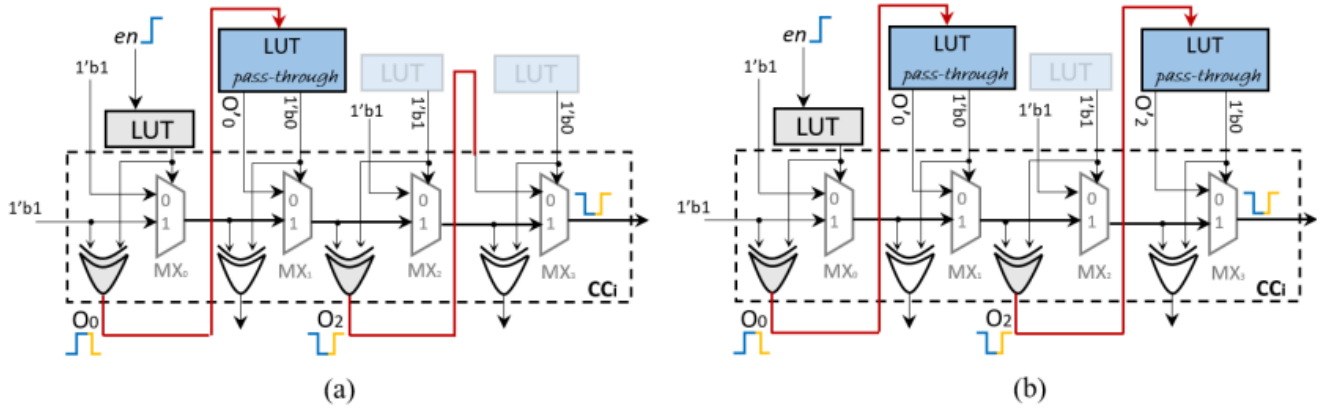

Fig. 4. Proposed CC-based multistage RO design. (a) CC_ROv2. (b) CC_ROv3. Additional LUTs are used as pass-through elements to enable fine-tuning of the RO characteristics. O ′ i signals are delayed copy of Oi .

Nowadays, CCs are available in various extents in most FPGA devices produced by many vendors [20], [21]. They consist of hard-wired resources specialized for efficient ON-chip implementation of arithmetic operations, such as additions and multiplications. A CC typically consists of k internal stages of cascaded multiplexers that, in combination with auxiliary XOR gates, implement as many full adders each exploiting the basic 1-bit carry look-ahead logic. Most importantly, CCs are placed neatly within the chip, so that longer chains can be implemented by cascading multiple instances through dedicated routing. Our proposal aims at exploiting this unique property for the implementation of ROs. Indeed, during the automatic P&R step, the position of the first CC can be used as the “anchor point” for the whole design; then, both cascaded CCs and nets are placed and routed through a delay-deterministic scheme, thus avoiding the need for manual P&R. According to the above consideration, the design illustrated in Fig. 3 (in the following, named CC_ROv1) is here proposed as the basic configuration to enable the CC unit to operate as an oscillator. Without loss of generality, the architecture includes x CCs, each composed by k = 4 internal stages. In order to enable the propagation of the oscillating signal along the circuit, all the internal stages must be used for each CC, except for the last one where the designer can choose the convenient number of stages to be exploited. Thereby, the selectors of multiplexers MXs, the first one excepted, are properly set to constant values following the scheme shown in Fig. 3. Thus, while the multiplexers MXs in the even positions propagate the signal coming from the previous multiplexer, those in the odd ones transfer on the carry line the signal coming from the XOR gate in the previous stage. In this way, a chain of XOR gates, each acting as an inverter stage, is formed. Finally, to make odd the number of inverting stages, the first LUT, highlighted in gray, maps a NAND function: the low-to-high transition of the en signal triggers the first inversion through the LUT, which is then propagated to the XOR gate in order to produce O0. It is important to note that the function mapped within the first LUT has to be chosen based on the configuration of other components (e.g., selectors and inputs of the multiplexers). We verified that such kind of changes in the first LUT content does not significantly affect the behavior of the RO architecture. In the subsequent chain positions, the proposed scheme allows the oscillating signal to be propagated to the next stage by alternating

the outputs from the XOR gate and the multiplexer, respectively. As a result, the CC_ROv1 configuration actually includes y = 2x + 1 inverting stages, highlighted in gray in Fig. 3. From the schematic of Fig. 3, it can be observed that the remaining LUTs are not necessary to produce the oscillation, but they could be exploited as pass-through elements to fine-tune the RO period. In such a case, the external multiplexers aligned to the CC even positions can be used to select the corresponding LUT outputs, as depicted in Fig. 4. This choice expands the design space, leading to the new configurations CC_ROv2 and CC_ROv3, where one or two LUTs per CC are enabled, respectively. As deeply analyzed in the following, the proposed configurations exhibit different characteristics in terms of oscillation frequency and sensitivity to voltage/temperature variations. Besides the interesting properties mentioned above, as shown in Fig. 3, the proposed scheme can provide the oscillating output RO out through one or more of the unused XOR gates, such as that at the position kx − 1, thus avoiding interference with the interconnect loop. This allows realizing efficient multiphase oscillators suitable for applications that need load-independent RO frequencies.

In this section, we introduce the mathematical model for the frequency estimation of the proposed RO scheme. Let us consider the path involved in the combinatorial loop of the CC_ROv1 configuration, as highlighted in blue in Fig. 5. Then, the RO frequency fRO can be computed as 1/2τp by applying (1) and (2) for modelling τp.

$$\tau p = \tau LUT + \tau S0 \rightarrow O0 + \tau d + (x - 1)(\tau CO3 \rightarrow CI + \tau CI \rightarrow O0 + \tau d) + \tau \text{ loop} \quad (1)$$

$$\tau d = \tau O0 \rightarrow B1 + \tau B1 \rightarrow O2 + \tau O2 \rightarrow B3 + \tau B3 \rightarrow CO3 \quad (2)$$

According to the adopted FPGA technology, the delay switching characteristics at the nominal conditions are provided by the vendor for most of the contributions highlighted in Fig, except τO0→B1 , τO2→B3 , and τ loop. Table I summarizes the values of each contribution τi→j , with reference to the FPGA chips belonging to the Xilinx Artix-7 (speed grade −1) family; a similar approach could be adopted with different devices. The four examined cases, in the following, named fast–fast (FF), fast–slow (FS), slow–fast (SF), and slow–slow (SS), take into account, respectively, the process corner (first letter) and the standard delay format (SDF) adopted for delay modelling (second letter). Table II, instead, reports the τO0→B1 and τO2→B3 delays related to the specific tracks illustrated in Fig. 5. Although related to programmable routing, such contributions can be considered as constants, regardless of the RO length, since the start point and the endpoint of the interconnection are fixed by the adopted architecture, as it will be detailed later. It is worth noting that the τ loop delay depends not only on how many stages are involved in the oscillator, but also on

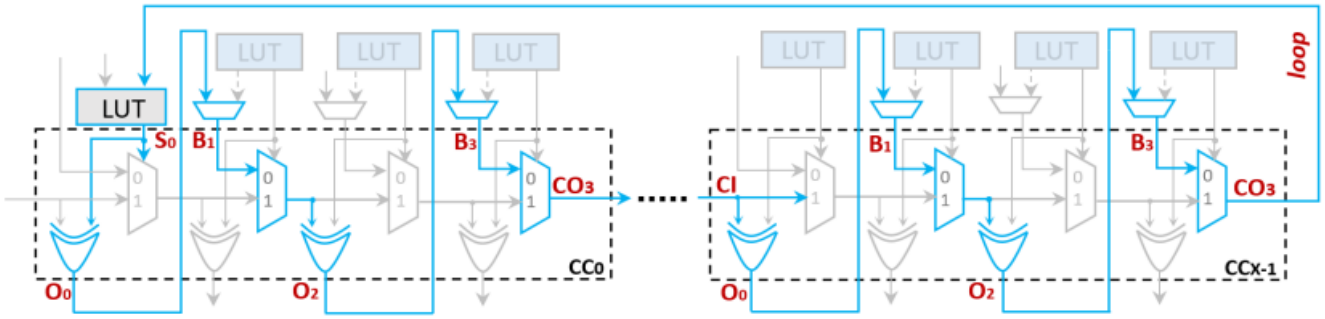

Fig. 5. Propagation delay path for the CC_ROv1 design. Red labels identify the interconnection segments as reported in the technical documentation for delay estimation purpose

$\tau_{i \to j}$	FF (ns)	FS (ns)	SF (ns)	SS (ns)
$\tau_{S_0 \to O_0}$	0.060	0.079	0.170	0.223
$\tau_{B_1 \to O_2}$	0.110	0.186	0.358	0.554
$\tau_{B_3 \to CO_3}$	0.088	0.140	0.248	0.385
$\tau_{CO_3 \to CI}$	0.009	0.009	0.009	0.009
$\tau_{CI \to O_0}$	0.054	0.085	0.150	0.235
τ_{LUT}	0.045	0.056	0.100	0.124

TABLE I NOMINAL DELAY SWITCHING CHARACTERISTICS

$\tau_{i \to j}$	FF (ns)	FS (ns)	SF (ns)	SS (ns)
$\tau_{O_0 \to B_1}$	0.181	0.216	0.461	0.557
$\tau_{O_2 \to B_3}$	0.179	0.215	0.426	0.515

TABLE II INTRASLICE ROUTING DELAY

RESULTS:

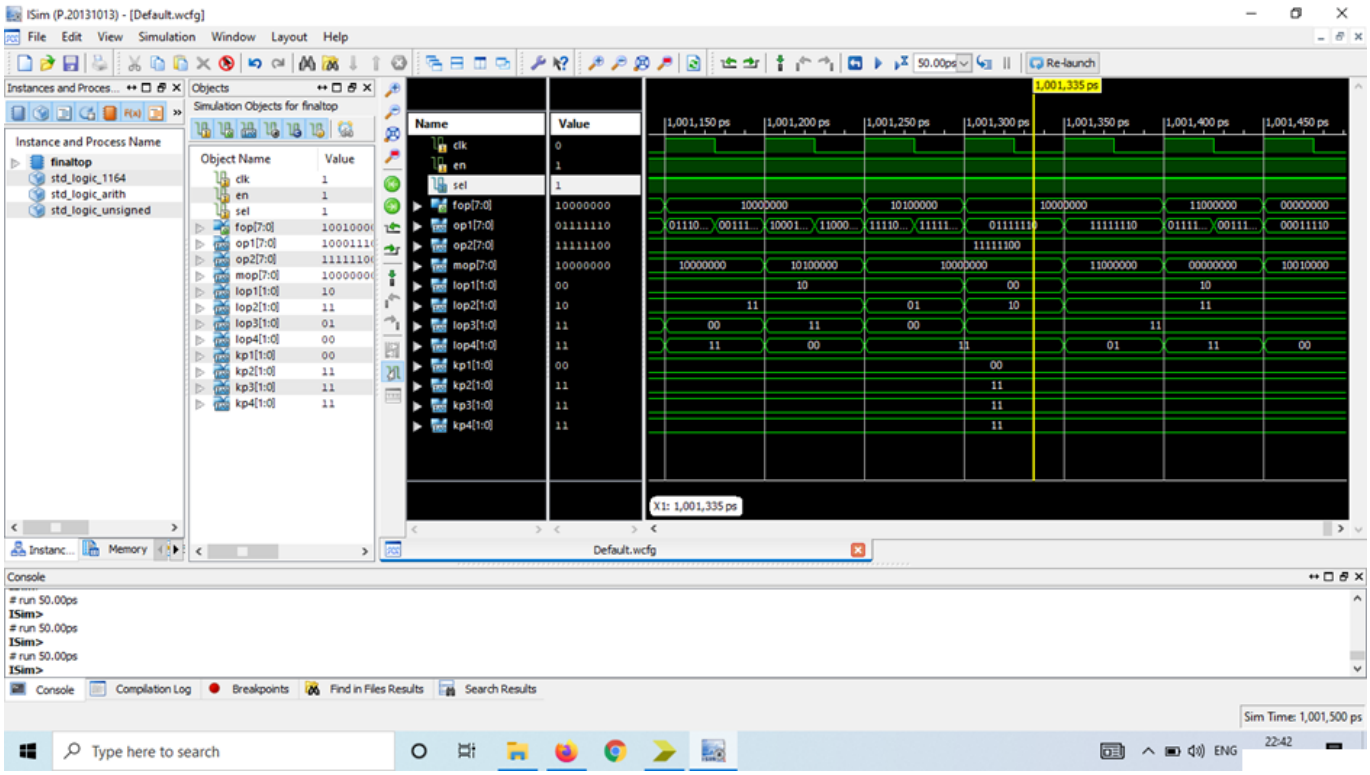

Fig a: Proposed simulation results

Advantages

1. Very High Oscillation Frequency

- Fast carry chains are dedicated, hard-wired paths in FPGAs.
- They provide minimal propagation delay compared to LUT-based logic.
- Enables GHz-range oscillation, which is difficult to achieve with standard LUT ring oscillators.

2. Low Jitter and Improved Signal Stability

- Carry chains exhibit deterministic and symmetric delay characteristics.
- Reduced routing uncertainty leads to lower phase noise and jitter.
- Suitable for high-precision timing applications.

3. Fine-Grained Delay Control

- Each carry element introduces a small and predictable delay.
- Multistage configurations allow precise control of oscillation period.
- Useful for delay tuning and characterization

Applications

1. Physical Unclonable Functions (PUFs)

- Exploits manufacturing variations in carry-chain delays.
- Used in:
    - Device authentication
    - Hardware fingerprinting
    - Anti-counterfeiting systems

2. True Random Number Generators (TRNGs)

- Jitter and metastability from high-speed oscillations provide true entropy.
- Suitable for:
    - Cryptographic key generation
    - Secure communication systems

3. On-Chip Delay and Timing Characterization

- Measures intra-chip and inter-chip delay variations.
- Helps in:
    - FPGA process monitoring

4. Clock Generation and Calibration

- Used as local high-frequency clocks.
- Enables self-calibrating clock systems without PLL/DLL blocks.

Conclusion: This work explored the implementation of multistage ring oscillators (ROs) using fast carry chains on FPGAs, focusing on their design methodology and performance characterization. By exploiting the dedicated carry-chain logic available in modern FPGA architectures, the proposed approach achieves significantly higher oscillation frequencies, reduced jitter, and more predictable delay behavior compared to conventional LUT-based ring oscillators. The experimental characterization demonstrates that carry-chain-based ROs provide fine-grained delay resolution and improved repeatability, making them highly suitable for applications requiring precise timing and

sensitivity to process, voltage, and temperature (PVT) variations. Furthermore, the efficient utilization of FPGA resources ensures minimal overhead while preserving logic elements for other system functionalities. These advantages validate the effectiveness of fast carry chains as a robust and scalable building block for high-speed oscillatory circuits in reconfigurable platforms.

Future Scope: Despite the promising results, several directions remain open for further research and enhancement: Advanced PVT Compensation Techniques: Future work can focus on integrating adaptive calibration and compensation mechanisms to mitigate the effects of voltage and temperature variations, thereby improving long-term stability and accuracy. Integration with Security Primitives: The proposed design can be extended to develop more robust PUF architectures and true random number generators (TRNGs) with enhanced entropy, resistance to modeling attacks, and improved reliability.

REFERENCES

[1] F. Kodýtek, R. Lórencz, and J. Bucek, “Three counter value based ˇ ROPUFs on FPGA and their properties,” Microprocessors Microsystems, vol. 88, Feb. 2022, Art. no. 104375. [2] B. Halak, M. Zwolinski, and M. S. Mispan, “Overview of PUF-based hardware security solutions for the Internet of Things,” in Proc. IEEE 59th Int. Midwest Symp. Circuits Syst. (MWSCAS), Abu Dhabi, UAE, Oct. 2016, pp. 1–4. [3] M. Barbareschi, G. Di Natale, L. Torres, and A. Mazzeo, “A ring oscillator-based identification mechanism immune to aging and external working conditions,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 65, no. 2, pp. 700–711, Feb. 2018. [4] B. Yang, V. Rožic, M. Grujic, N. Mentens, and I. Verbauwhede, “ESTRNG: A high-throughput, low-area true random number generator based on edge sampling,” IACR Trans. Cryptograph. Hardw. Embedded Syst., vol. 10, pp. 267–292, Aug. 2018. [5] M. A. Prada-Delgado, C. Martínez-Gómez, and I. Baturone, “Autocalibrated ring oscillator TRNG based on jitter accumulation,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Seville, Spain, Oct. 2020, pp. 1–4. [6] M. Grujic and I. Verbauwhede, “TROT: A three-edge ring oscillator based true random number generator with time-to-digital conversion,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 69, no. 6, pp. 2435–2448, Jun. 2022. [7] S. Moini et al., “Voltage sensor implementations for remote power attacks on FPGAs,” ACM Trans. Reconfigurable Technol. Syst., vol. 16, no. 1, pp. 1–21, Dec. 2022. [8] I. Giechaskiel, K. B. Rasmussen, and J. Szefer, “Measuring long wire leakage with ring oscillators in cloud FPGAs,” in Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL), Barcelona, Spain, Sep. 2019, pp. 45–50. [9] M. Ebrahimi and Z. Navabi, “Selecting representative critical paths for sensor placement provides early FPGA aging information,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 39, no. 10, pp. 2976–2989, Oct. 2020. [10] M. M. Alam, M. Tehranipoor, and D. Forte, “Recycled FPGA detection using exhaustive LUT path delay characterization and voltage scaling,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 12, pp. 2897–2910, Dec. 2019. [11] T. Kilian, D. Tille, M. Huch, M. Hanel, and U. Schlichtmann, “Performance screening using functional path ring oscillators,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no. 6, pp. 711–724, Jun. 2023. [12] S. Bae, M. Lee, S.-M. Yoo, and J.-Y. Sim, “A temperature compensated ring oscillator with LC-based period error detection,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 31, no.

12, pp. 2152–2156, Dec. 2023, doi: 10.1109/TVLSI.2023.3322709. [13] B. Razavi, “The ring oscillator [a circuit for all seasons],” IEEE Solid StateCircuits Mag., vol. 11, no. 4, pp. 10–81, Jun. 2019. [14] M. Elnawawy, A. Farhan, A. A. Nabulsi, A. R. Al-Ali, and A. Sagahyroon, “Role of FPGA in Internet of Things applications,” in Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT), Ajman, United Arab Emirates, Dec. 2019, pp. 1–6. [15] N. N. Anandakumar, M. S. Hashmi, and M. Tehranipoor, “FPGAbased physical unclonable functions: A comprehensive overview of theory and architectures,” Integration, vol. 81, pp. 175–194, Nov. 2021. [16] C. Gu, C. H. Chang, W. Liu, N. Hanley, J. Miskelly, and M. O’Neill, “A large scale comprehensive evaluation of single-slice ring oscillator and PicoPUF bit cells on 28 nm Xilinx FPGAs,” in Proc. 3rd ACM Workshop Attacks Solutions Hardw. Secur. Workshop, London, U.K., Nov. 2019, pp. 101–106. [17] A. Wild, G. T. Becker, and T. Güneysu, “On the problems of realizing reliable and efficient ring oscillator PUFs on FPGAs,” in Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST), McLean, VA, USA, 2016, pp. 103–108. [18] H. Kareem and D. Dunaev, “Towards performance optimization of ring oscillator PUF using Xilinx FPGA,” in Proc. 17th Iberian Conf. Inf. Syst. Technol. (CISTI), Madrid, Spain, Jun. 2022, pp. 1–6. [19] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, “A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices,” in Proc. 26th Int. Conf. Field Program. Log. Appl. (FPL), Lausanne, Switzerland, Aug. 2016, pp. 1–10. [20] AMD. Xilinx 7 Series FPGAs Data Sheet: Overview DS180 (v2.6.1). Accessed: May 4, 2024. [Online]. Available: https://docs.amd. com/v/u/en-US/ds180_7Series_Overview.