



POLITECHNIKA POZNAŃSKA

**WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI
Katedra Systemów Telekomunikacyjnych i Optoelektroniki**

**WYKORZYSTANIE ZASOBÓW UKŁADÓW
REPROGRAMOWALNYCH DO WYTWARZANIA CIĄGÓW
LOSOWYCH**

ROZPRAWA DOKTORSKA

Łukasz Matuszewski

Promotor:
dr hab. inż. Mieczysław Jessa, prof. PP

Poznań 2019

SPIS TREŚCI

Wykaz ważniejszych skrótów i oznaczeń	5
Streszczenie.....	7
Abstract	9
1. Wstęp.....	11
2. Generatory ciągów liczbowych w kryptografii	13
2.1. Klasyfikacja generatorów ciągów liczbowych	13
2.2. Generatory liczb pseudolosowych.....	14
2.3. Generatory liczb rzeczywiście losowych.....	15
2.3.1. Generatory TRNG wykorzystujące zjawisko szumu.....	16
2.3.2. Generatory TRNG wykorzystujące stany metastabilne.....	22
2.3.3. Generatory TRNG oparte na zjawisku szybkozmiennych fluktuacji fazowych.....	26
2.3.4. Kwantowe generatory ciągów liczb rzeczywiście losowych.....	33
2.3.5. Hybrydowe generatory TRNG	34
2.3.6. Generatory TRNG wykorzystujące chaos	36
2.3.7. Rzut monetą, kością	37
2.3.8. Inne generatory postrzegane jako losowe.....	37
2.3.9. Podsumowanie rozwiązań generatorów liczb rzeczywiście losowych.....	38
3. Układy FPGA	40
3.1. Architektura układów FPGA	40
3.2. Proces projektowania układów cyfrowych w technologii FPGA	41
3.2.1. Poziomy abstrakcji opisu projektu	43
3.2.2. Języki opisu sprzętu	43
3.3. Obszary zastosowania.....	44
3.4. FPGA a generatory ciągów losowych.....	44
4. Testy generatorów ciągów rzeczywiście losowych	45
4.1. Testowanie źródła entropii.....	45
4.2. Testy statystyczne ciągów	51
4.3. Metoda restartów	55
5. Łaczony generator TRNG z detektorem fazy	57
5.1. Charakterystyka jittera w oscylatorach pierścieniowych.....	57
5.2. Koncepcja łączonego generatora ciągów losowych.....	65
5.3. Odporność na atak typu wstrzykiwanie częstotliwości.....	67
5.4. Wykorzystanie detektora fazy do ekstrakcji losowości	70
5.5. Ocena właściwości ciągów wytwarzanych przez proponowany generator TRNG	78
5.6. Wytwarzanie ciągów losowych z dużą szybkością.....	96
6. Podsumowanie.....	101
Literatura.....	103
Spis rysunków i tabel	111

WYKAZ WAŻNIEJSZYCH SKRÓTÓW I OZNACZEŃ

A/C	–	Przetwornik analogowo-cyfrowy
AND	–	Bramka iloczynu logicznego
ASIC	–	Specjalizowany układ scalony (ang. <i>Application Specific Integrated Circuit</i>)
BIS	–	Federalne Biuro Bezpieczeństwa Informacyjnego (niem. <i>Bundesamt für Sicherheit in der Informationstechnik</i>)
CASR	–	Automat komórkowy z rejestrem przesuwnym (ang. <i>Cellular Automata Shift Register</i>)
CLB	–	Konfigurowalna komórka logiczna (ang. <i>Configurable Logic Block</i>)
CLK	–	Zegar – przebieg taktujący układ cyfrowy (ang. <i>Clock</i>)
CMOS	–	Technologia wytwarzania układów scalonych (ang. <i>Complementary Metal-Oxide Semiconductor</i>)
DCM	–	Kontroler sygnałów zegarowych (ang. <i>Digital Clock Manager</i>)
DFT	–	Dyskretna transformata Fouriera (ang. <i>Discrete Fourier Transform</i>)
FIPS	–	Federalna Norma Przetwarzania Informacji (ang. <i>Federal Information Processing Standard</i>)
FPGA	–	Programowalna macierz bramek (ang. <i>Field Programmable Gate Array</i>)
GARO	–	Oscylator pierścieniowy typu Galois (ang. <i>Galois Ring Oscillator</i>)
HDL	–	Język opisu sprzętu (ang. <i>Hardware Description Language</i>)
IID	–	Niezależne i o identycznym rozkładzie (ang. <i>Independent and Identically Distributed</i>)
IOB	–	Bloki wejści/wyjści (ang. <i>Input Output Block</i>)
LBA	–	Macierz komórek logicznych (ang. <i>Logic Block Array</i>)
LFSR	–	Rejestr przesuwny z liniowym sprzężeniem zwrotnym (ang. <i>Linear Feedback Shift Register</i>)
LUT	–	Tablica odwzorowań logicznych (ang. <i>Lookup Table</i>)
NIST	–	Narodowy Instytut Standaryzacji i Technologii (ang. <i>National Institute of Standards and Technology</i>)
NOT	–	Inwerter
PLL	–	Pętla synchronizacji fazy (ang. <i>Phase Locked Loop</i>)
PRNG	–	Generator liczb pseudolosowych (ang. <i>Pseudo-Random Number Generator</i>)
QRNG	–	Kwantowy generator liczb rzeczywiście losowych (ang. <i>Quantum Random Number Generator</i>)
RAM	–	Pamięć o dostępie swobodnym (ang. <i>Random Access Memory</i>)
RO	–	Oscylator pierścieniowy (ang. <i>Ring Oscillator</i>)
SC	–	Karta mikroprocesorowa (ang. <i>Smart Card</i>)
SHA	–	Bezpieczny algorytm skrótu (ang. <i>Secure Hash Algorithm</i>)
SRAM	–	Statyczna pamięć o dostępie swobodnym (ang. <i>Static Random Access Memory</i>)
STR	–	Samotakujące pierścienie (ang. <i>Self-Timed Rings</i>)
TE	–	Błąd czasu (ang. <i>Time Error</i>)

WYKAZ WAŻNIEJSZYCH SKRÓTÓW I OZNACZEŃ

TRNG	–	Generator ciągów liczb rzeczywiście losowych (ang. <i>True Random Number Generator</i>)
TTL	–	Technologia wytwarzania cyfrowych układów scalonych oparta na tranzystorach bipolarnych (ang. <i>Transistor-Transistor Logic</i>)
XOR	–	Bramka sumy modulo 2 (ang. <i>Exclusive OR</i>)
E	–	Wartość oczekiwana
f	–	Częstotliwość sygnału
H	–	Entropia
J_{cc}	–	Szybkozmienne fluktuacje fazy cykl do cyklu — jitter cykl do cyklu (ang. <i>cycle to cycle jitter</i>)
K	–	Liczba generatorów źródłowych
M	–	Długość ciągu
m_{min}	–	Wartość oznaczająca pierwszy rzeczywiście losowy bit w ciągu
N	–	Liczba restartów
P_T	–	P-wartość
T	–	Okres sygnału
α, β	–	Poziom istotności
δ	–	Impuls delty Diraca
τ	–	Element opóźniający
φ	–	Faza sygnału
χ^2	–	Test zgodności chi-kwadrat
ω	–	Pulsacja

STRESZCZENIE

W rozprawie przedstawiono sposób wykorzystania zasobów układów FPGA (ang. *Field Programmable Gate Array*) do budowy szybkich i skalowalnych generatorów ciągów liczb rzeczywiście losowych. Dokonano przeglądu i oceny rozwiązań istniejących generatorów losowych z punktu widzenia cech przydatnych w implementacji w układach reprogramowalnych. Do oceny losowości posłużyły metody testowania zaproponowane przez NIST (ang. *National Institute of Standards and Technology*) i opisane w dokumentach firmowanych przez tę organizację. Jako najbardziej obiecującą metodę wytwarzania losowych bitów wybrano metodę bazującą na szybkozmiennych fluktuacjach fazy oscylatorów pierścieniowych. Przedstawiono model matematyczny jittera deterministycznego i niedeterministycznego sygnału wyjściowego oscylatorów pierścieniowych. Zbadano oscylatory pierścieniowe zaimplementowane w układzie FPGA pod kątem obecności w sygnale wyjściowym jittera niedeterministycznego. W efekcie powstał generator liczb rzeczywiście losowych wykorzystujący wiele niezależnych źródeł losowości – oscylatorów pierścieniowych. Nowością jest wykorzystanie detektora fazy jako elementu pozyskującego losowość z sygnału oscylatorów pierścieniowych. Wyjścia detektorów fazy poddano operacji próbkowania, a sygnały wyjściowe połączono za pomocą sumy modulo 2 w jeden strumień bitów. Losowość tak otrzymanego ciągu zbadano za pomocą testów statystycznych oraz z wykorzystaniem metody restartów, po uprzednim zaimplementowaniu proponowanego generatora w układach FPGA wykonanych w różnych technologiach i pochodzących od różnych producentów. Badania przeprowadzono dla czternastu różnych cyfrowych układów reprogramowalnych wytwarzanych przez trzech producentów posiadających łącznie ponad 90% rynku układów FPGA. Rezultaty badań potwierdziły uniwersalność proponowanego rozwiązania, a wytwarzane przez generator ciągi spełniły wszystkie testy statystyczne rekomendowane przez NIST. Na podstawie otrzymanych danych skonstruowano generator skalowalny, który oferuje wymienność zasobów i szybkość wytwarzania bitów. Proponowany generator wyróżnia się dużą wydajnością na tle innych znanych rozwiązań, gdyż może wytwarzać losowe bity z szybkością dochodzącą do 38,4 Gb/s.

ABSTRACT

The dissertation concerns the use of FPGA (*Field Programmable Gate Array*) circuits to build high-speed and scalable true random number generators. The review and evaluation of existing solutions was performed considering systems useful in implementation in reprogrammable systems. The main requirement for generated bitstreams is to pass all tests proposed by NIST (*National Institute of Standards and Technology*). A thorough review of testing methods for true random sequences proposed by NIST has been made. The most promising method of generating random bits was deemed the randomness acquisition from phase jitter of ring oscillators. A mathematical model of deterministic and non-deterministic jitter in ring oscillators was presented. Ring oscillators implemented in FPGA were tested for the presence of non-deterministic jitter in their signal. As a result, a generator was designed based on many independent sources of randomness – ring oscillators. The mathematical model was also presented. The novelty is the use of a phase detector as a randomness extractor from the signal of ring oscillators. The phase detector outputs were sampled and the signals were combined XOR forming one random bitstream. This made it possible to obtain strings that consist only of random bits, which was confirmed by running a series of tests using the restart method. The presented generator has been tested using NIST statistical tests and restarts mechanism to assess the possibility of implementation in FPGAs of various manufacturers and in various manufacturing technologies. The research was carried out on fourteen different FPGAs from three main manufacturers, which hold about 90% of the FPGA market. The research confirmed the universality of the proposed solution, and the sequences produced by the generator passed all statistical tests recommended by NIST. Based on the obtained results, a scalable generator was designed in which the used resources of the FPGA system were exchanged for the speed of generating sequences. Such a generator stands out due to its high efficiency in comparison to other known solutions and can generate random bits at a rate of 38.4 Gb/s.

1. WSTĘP

Generatory ciągów liczb rzeczywiście losowych są używane w wielu dziedzinach nauki i techniki. Jednym z najważniejszych obszarów zastosowań liczb losowych jest kryptografia i ochrona informacji. Potrzeba zabezpieczenia informacji znana jest już od starożytności. Wśród najstarszych metod ochrony informacji wymienia się szyfr Cezara. Nazwa pochodzi od Juliusza Cezara (100 r. p.n.e. – 44 r. p.n.e.), który wykorzystywał szyfrowanie do ochrony informacji przed niepożądany dostałem. Czasy wojen i niepokoju wzmagają zawsze potrzebę bezpiecznego przesyłania informacji związanych z działalnością szpiegowską i ruchami wojsk. Przez wieki potrzeba ta nie ulegała zmianie, zmieniały się natomiast metody.

W latach 70-tych XX wieku popularność zaczęły zdobywać komputery domowe. To w dużej mierze dzięki nim rozwinęła się globalna sieć, którą obecnie znamy jako Internet. Wraz z rozwojem Internetu pojawiła się tzw. kultura hackerska. Pasjonaci komputerowi zaczęli rozszerzać swoją wiedzę poprzez włamywanie się do różnych systemów. Spowodowało to odpowiedź administratorów, którzy zaczęli zabezpieczać systemy informatyczne coraz bardziej skomplikowanymi metodami, stosując coraz bardziej zaawansowane algorytmy oraz urządzenia zapewniające szyfrowanie i integralność danych. Hakerzy starają się złamać zabezpieczenia aby zdobyć dane, które umożliwiają np. przejęcie naszych środków finansowych, czy podszycie się pod inną osobę. Z kolei administratorzy systemów informatycznych starają się zapobiegać nieautoryzowanemu dostępowi do danych lub przynajmniej zminimalizować szkody spowodowane takim dostępem. Jest to coraz trudniejsze ponieważ systemy informatyczne obecne są w każdej dziedzinie życia. Są integralną częścią samochodów, systemów sterowania ruchem drogowym, przesyłem gazu, energii elektrycznej, systemów finansowych, instytucji rządowych itd. Uzyskanie dostępu do tych systemów przez osoby nieupoważnione mogłoby spowodować wiele katastrofalnych następstw.

Współczesna ochrona informacji i uwierzytelnianie komunikatów wykorzystują systemy kryptograficzne, które do poprawnego działania potrzebują liczb losowych. Z kolei ciągły wzrost zapotrzebowania na różnego rodzaju dane, połączony z większymi szybkościami transmisji i przetwarzania, wymaga nieustanego rozwijania dostępnych metod wytwarzania liczb losowych. Niezbędne są generatory, które wytwarzają liczby losowe z dużą szybkością, najlepiej na żądanie, zintegrowane w tym samym układzie

cyfrowym z systemem kryptograficznym. Wytwarzane liczby muszą być nieprzewidywalne, a sam generator odporny na ataki.

Jako tezę rozprawy przyjęto sformułowanie: **układy FPGA** (ang. *Field Programmable Gate Array*) **można wykorzystać do budowy szybkich generatorów liczb rzeczywiście losowych z możliwością wymiany zasobów układów reprogramowalnych na szybkość wytwarzanego ciągu bitów, ocenianego za pomocą testów statystycznych NIST** (ang. *National Institute of Standards and Technology*).

W rozdziale drugim przedstawiono klasyfikację generatorów liczb losowych. Dokonano szczegółowego przeglądu literatury przedmiotu z podziałem na zjawiska fizyczne odpowiadające za wytwarzanie liczb losowych.

Rozdział trzeci poświęcono układom FPGA. Przedstawiono architekturę cyfrowych układów reprogramowalnych oraz proces projektowania układów wewnętrz struktur FPGA. Opisano obszary zastosowań, ze szczególnym uwzględnieniem generatorów losowych implementowanych w tych układach.

Rozdział czwarty stanowi opis metod testowania generatorów i wytwarzanych przez nie ciągów. Szczegółowo opisano testy statystyczne NIST wykorzystywane do testowania źródła entropii jak i ciągów liczb. Opisano metodę restartów jako propozycję oceny losowości wytwarzanych bitów.

Rozdział piąty przedstawia model matematyczny i właściwości szybkozmiennych fluktuacji fazy oscylatorów pierścieniowych wykorzystywanych w generatorach ciągów bitów losowych. Przedstawiono sposoby pomiaru jittera dla oscylatorów pierścieniowych zaimplementowanych z elementem opóźniającym o różnej długości i w różnej technologii. Opisano model i właściwości łączonego generatora ciągów liczb rzeczywiście losowych wykorzystującego detektor fazy jako element pozyskujący losowość. Wyjaśniono koncepcję, szczegóły implementacyjne i proces testowania. Zaproponowano metodę wytwarzania ciągów liczb losowych z bardzo dużą szybkością, zależną tylko od dostępnych zasobów układu FPGA. Rozdział kończy ocena właściwości statystycznych ciągów wytwarzanych przez proponowany generator.

Podsumowanie przeprowadzonych prac i uzyskanych wyników znajduje się w rozdziale szóstym, kończącym rozprawę.

2. GENERATORY CIĄGÓW LICZBOWYCH W KRYPTOGRAFII

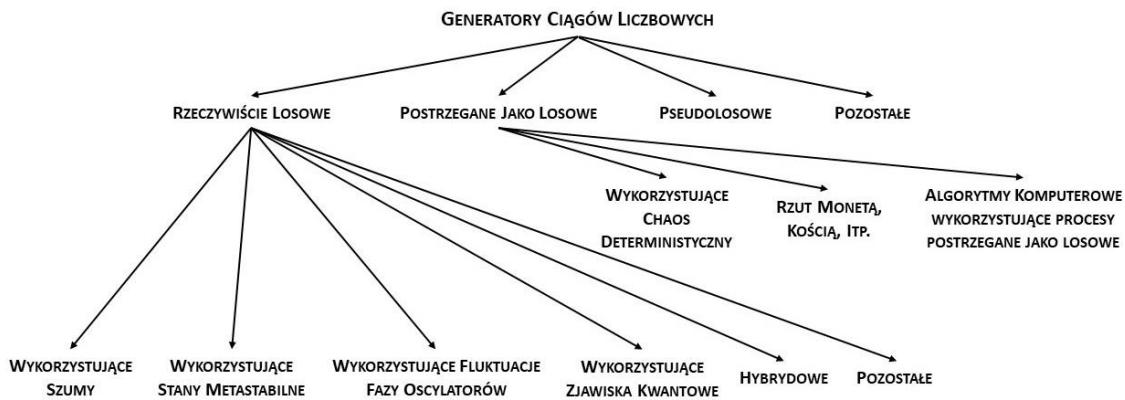
Generatory ciągów liczbowych stanowią jeden z podstawowych elementów używanych w protokołach kryptograficznych. Znajdują zastosowanie w wytwarzaniu kluczy kryptograficznych, wektorów inicjacji, wartości dopełnień itp. [1], a także zyskują na ważności w takich dziedzinach jak symulacje numeryczne, metoda Monte Carlo, badania statystyczne, algorytmy randomizacji czy gry losowe. Można tu wspomnieć takie algorytmy jak: testowanie pierwszości Millera-Rabina, symulowane wyżarzanie czy testowanie równoważności wielomianów [2]. Dzięki generatorom ciągów liczbowych uzyskujemy bezpieczny dostęp do wielu usług jak na przykład: komunikacja bezprzewodowa, dostęp do poczty elektronicznej, bankowości elektronicznej, bankomatów, zakupów internetowych i innych aplikacji ogólnie związanych z cyber-bezpieczeństwem.

Generatory ciągów liczbowych do zastosowań kryptograficznych powinny spełniać kilka wymogów bezpieczeństwa. Przede wszystkim wytwarzane przez nie liczby powinny być nieprzewidywalne, a więc muszą być wytwarzane przez procesy niedeterministyczne. Ważne jest, aby ciągi liczb wytwarzanych przez generatory miały dobre właściwości statystyczne, kolejne liczby były niezależne od siebie, a wszystkie elementy ciągu były równoprawdopodobne. Generatory losowe powinny również być odporne na różnego rodzaju ataki.

2.1. KLASYFIKACJA GENERATORÓW CIĄGÓW LICZBOWYCH

Często generatory ciągów liczbowych klasyfikuje się jako deterministyczne, czyli takie, których działanie jest zdeterminowane przez warunki początkowe i znany algorytm generacji, oraz niedeterministyczne czyli takie, których nie można opisać znanymi równaniami matematycznymi. W przypadku generatorów deterministycznych oznacza to, że uruchamiany kilkakrotnie generator doprowadzi za każdym razem do takiego samego wyniku. Generatorami deterministycznymi są między innymi generatory ciągów pseudolosowych PRNG (ang. *Pseudo-Random Number Generator*). Generatory niedeterministyczne natomiast nie są zdeterminowane, a każde uruchomienie takiego generatora doprowadza do innego wyniku. Generatorami niedeterministycznymi są generatory liczb losowych TRNG (ang. *True Random Number Generator*), które do wytwarzania ciągów losowych wykorzystują zjawiska powszechnie uważane za losowe. Dodatkowo generatory liczb losowych można podzielić ze względu na charakter

wykorzystanego zjawiska [3]. Wśród generatorów ciągów nieprzewidywalnych można wyodrębnić grupę generatorów, które są postrzegane jako losowe. Takie generatory to np. generatory chaosu. Są one opisane całkowicie deterministycznymi równaniami, mogą jednak wytwarzać ciągi nieprzewidywalne ze względu na zmienne warunki początkowe, które znacznie zaburzają trajektorię systemu. Innym przykładem zjawiska postrzeganego za losowe może być rzut monetą, czy też rzut monetą. Podstawowy podział generatorów ciągów liczbowych przedstawia rysunek 2.1.



Rysunek 2.1 Klasifikacja generatorów ciągów losowych

Generatory losowe i postrzegane jako losowe są źródłem ciągów, które łącznie określamy jako ciągi losowe. Generatory pseudolosowe dostarczają ciągów liczb pseudolosowych, których właściwości statystyczne w skończonym przedziale czasu, wystarczającym dla danego zastosowania, mają być takie same jak właściwości ciągów losowych. Generatory określane jako pozostałe dostarczają ciągów liczbowych, które nie mogą być zakwalifikowane jako losowe lub pseudolosowe.

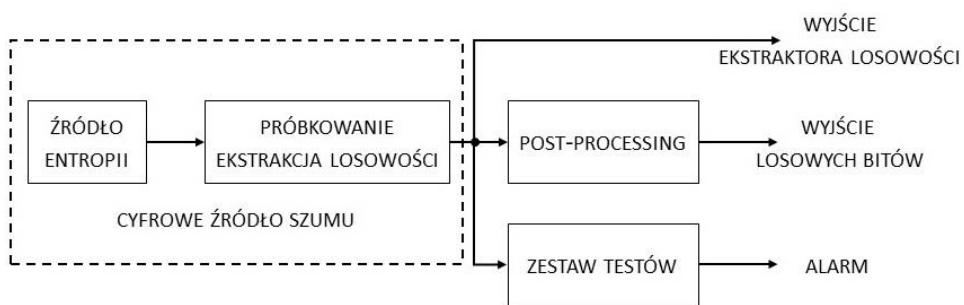
2.2. GENERATORY LICZB PSEUDOLOSOWYCH

Generator liczb pseudolosowych to w istocie algorytm, który na podstawie niewielkiej ilości informacji zwanej ziarnem, generuje deterministyczny ciąg liczb. Każdy wytworzony ciąg charakteryzuje się okresowością, co oznacza, że po pewnej liczbie iteracji elementy ciągu zaczynają się powtarzać. Współczesne konstrukcje posiadają na tyle długie okresy, że niedogodność ta nie jest uciążliwa. Powstały ciąg w czasie mniejszym od okresu nie jest odróżnialny od ciągu rzeczywiście losowego [4]. Niewątpliwą zaletą generatorów pseudolosowych jest szybkość wytwarzania ciągów wynosząca dzisiaj dziesiątki Gb/s. Podstawowe rozwiązania takich generatorów można znaleźć w [5].

2.3. GENERATORY LICZB RZECZYWIŚCIE LOSOWYCH

Generatory liczb rzeczywiście losowych są to aplikacje sprzętowe wykorzystujące zjawiska fizyczne powszechnie uznawane za losowe. Takimi zjawiskami są między innymi: szумy w układach elektronicznych, szybkozmienne fluktuacje fazy oscylatorów, rozpad promieniotwórczy czy zjawiska kwantowe [6]. Generatory te są na ogół wolne, a uzyskiwane szybkości wytwarzania ciągów wynoszą od pojedynczych b/s do kilku Mb/s. Wykorzystywane są głównie do wytwarzania kluczy kryptograficznych, wektorów inicjujących i ziarna generatorów pseudolosowych.

Podstawowy schemat blokowy generatora TRNG przedstawiony jest na rysunku 2.2.



Rysunek 2.2 Schemat blokowy generatora ciągów losowych

Generator TRNG składa się ze źródła losowości o możliwie dużej entropii, układu wyodrębniającego składniki losowe (ekstraktora losowości), mechanizmu poprawiającego właściwości statystyczne wytwarzanych ciągów (ang. *post-processing*) i ewentualnego układu wbudowanych testów [7]. Generator zazwyczaj posiada trzy wyjścia: jedno będące właściwym wyjściem generatora, drugie dostarczające danych bezpośrednio z wyjścia ekstraktora losowości i trzecie wyjście alarmu. Przepustowość i właściwości statystyczne tworzonych bitów ściśle zależą od jakości źródła losowości i zastosowanej metody ekstrakcji losowości. Fizyczne źródła losowości zazwyczaj mają niską entropię, która nie spełnia wymagań dotyczących aplikacji kryptograficznych. Z tego powodu stosuje się algorytmy post-processingu, które poprawiają właściwości statystyczne wyjściowego ciągu bitów. Przetwarzanie sygnału wyjściowego niesie jednak za sobą zagrożenia, mogące maskować zarówno wady wynikające ze słabego źródła losowości, błędów konstrukcyjnych generatora, jak również zewnętrzne ataki. Zamaskowana w ten sposób wada może zostać niezauważona przez standardowe testy statystyczne. Bezpieczeństwo generatora można zwiększyć przez implementację wbudowanych testów losowości, które działają równocześnie z generatorem. Testy te wykrywają odchylenia i błędy w sygnale bezpośrednio ze źródła szumu.

Głównym i najważniejszym elementem każdego generatora TRNG jest źródło entropii, ponieważ determinuje jego właściwości losowe [8]. Niektóre źródła wykazują się niezrównoważeniem zwanym też obciążeniem (ang. *bias*), które powinno być wyeliminowane w etapach ekstrakcji i post-processingu. Niezrównoważenie źródła jest przyczyną nierówno prawdopodobnych stanów na jego wyjściu, co w konsekwencji może przełożyć się na wystąpienie nierówno prawdopodobnych bądź nawet nielosowych podciągów w wytwarzanej sekwencji.

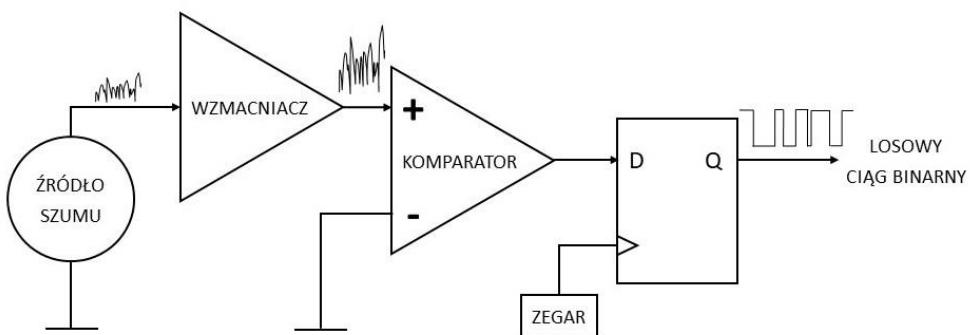
Ekstrakcja losowości to proces, który przekształca entropię źródła do postaci losowych wartości. Ważne jest, aby proces ekstrakcji uwzględniał charakter źródła entropii i był do niego odpowiednio dobrany. Nie może również zaburzać właściwości źródła entropii [8]. W urządzeniach cyfrowych najczęstszą metodą pozyskiwania losowości jest próbkowanie źródła. Mechanizm ten pobiera próbki ze źródła wtedy, gdy spodziewamy się możliwie wysokiej entropii. Metoda ekstrakcji i źródło entropii muszą być rozpatrywane łącznie ponieważ są ze sobą bezpośrednio związane. Dla każdego TRNG metoda ekstrakcji może być inna i może mieć swoje zalety i wady.

Post-processing jest stosowany, aby poprawić właściwości statystyczne sygnału. Po przetwarzaniu wyjściowy sygnał powinien mieć rozkład równomierny. Jest często stosowany do ukrywania niedoskonałości źródła entropii i procesu ekstrakcji [9]. Nie jest elementem wymaganym, jeżeli źródło entropii ma odpowiednie parametry. W zależności jednak od zastosowanego algorytmu, prawidłowe zastosowanie post-processingu zwiększa bezpieczeństwo generatora [10]. Istnieje wiele różnych algorytmów post-processingu: od najprostszych jak korektor Von Neumanna [11], przez korektory złożone z bramek XOR, rejestrzy przesuwne z liniowym sprzężeniem zwrotnym LFSR (ang. *Linear Feedback Shift Register*), czy funkcje skrótu [12], [13].

2.3.1. GENERATORY TRNG WYKORZYSTUJĄCE ZJAWISKO SZUMU

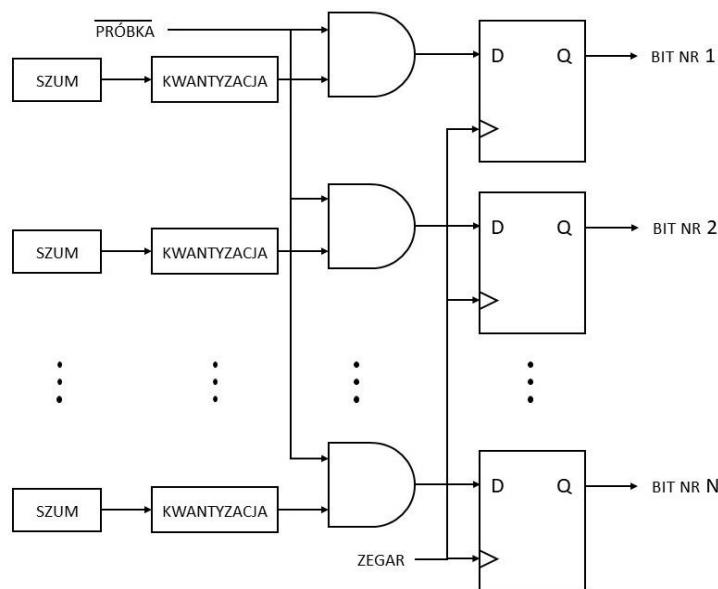
Każdy generator losowy potrzebuje źródła entropii. Jedne z podstawowych realizacji takich źródeł wykorzystują szum obecny w układach elektronicznych [14], [15], [16], [17], [18]. Jest to zarówno szum śrutowy inaczej zwany szumem Schottkiego [19], jak i szum termiczny. Szum śrutowy związany jest z nierównomiernym przepływem niezależnych od siebie elektronów przez barierę potencjału w półprzewodniku. Szum termiczny natomiast związany jest ze zmianami ruchu nośników prądu spowodowanych zmianami temperatury. Szum termiczny obecny jest w każdym rezystorze. Zasadę działania generatora liczb

losowych wykorzystującego źródło szumu przedstawia rysunek 2.3. Szum jest wzmacniany za pomocą szerokopasmowego wzmacniacza o dużym wzmacnieniu. Następnie wzmacniony sygnał szumu jest porównywany w komparatorze z napięciem odniesienia. Jeżeli wartość napięcia szumu jest większa od napięcia odniesienia na wyjściu komparatora otrzymujemy stan wysoki (logiczną „1”) w przeciwnym wypadku na wyjściu komparatora otrzymujemy stan niski (logiczne „0”). Próg napięcia odniesienia komparatora powinien być tak dobrany, aby zniwelować wpływ niezrównoważenia. Wyjście komparatora często jest próbkowane, aby uzyskać strumień bitów o stałej przepływności.



Rysunek 2.3 Generator oparty o zjawisko szumu w układach elektronicznych

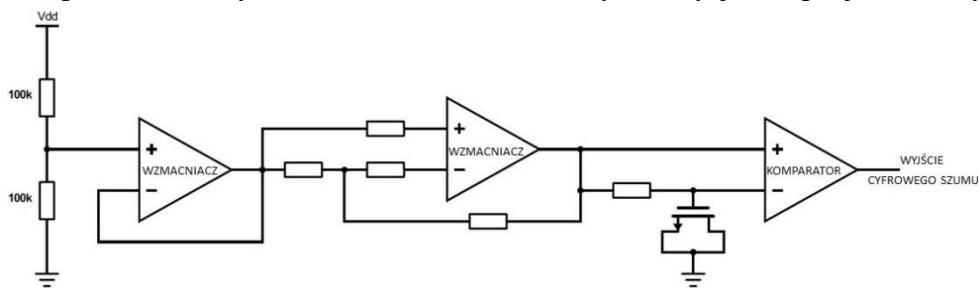
W pracy [16] przedstawiony został sposób wykorzystania szumu do wytwarzania losowych bitów. Realizacja układu przedstawiona jest na rysunku 2.4. Generator składa się z wielu niezależnych źródeł szumu. Sygnał szumu jest kwantowany za pomocą komparatora z histerezą. Następnie próbkowany jest w takt zewnętrznego sygnału zegarowego przy użyciu bramek AND. Wykorzystanie bramek AND zapobiega zmianie



Rysunek 2.4 Schemat generatora liczb losowych z bramkami AND i przełącznikami

bitów w trakcie próbkowania a przerzutnik wyjściowy jest niezbędny do zapamiętania ostatniego bitu. Pomaga również wyeliminować niezrównoważenie bitów, wyrównuje prawdopodobieństwo wystąpienia bitu „0” i „1”, zmniejsza to jednak przepływność bitową o połowę. W pracy [16] wyprowadzono również zależność na maksymalną częstotliwość próbkowania skwantowanego szumu, która wynosi 1,155 razy szerokość pasma sygnału szumu.

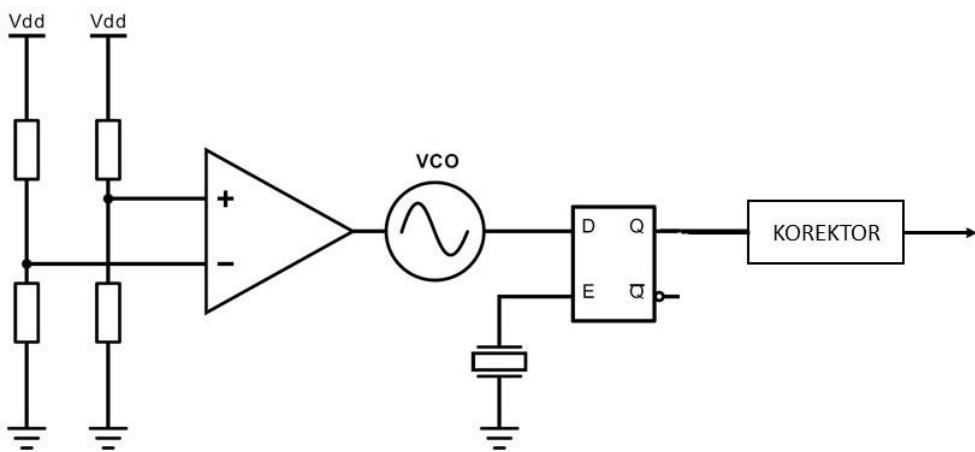
Autorzy pracy [18] zaprojektowali scalony generator liczb losowych oparty o szum termiczny powstały w dwóch polikrzemowych rezystorach o wartości $100\text{ k}\Omega$. Schemat generatora przedstawia rysunek 2.5. Układ został wykonany jako specjalizowany układ



Rysunek 2.5 Schemat scalonego źródła szumu

scalony ASIC (ang. *Application-Specific Integrated Circuit*). Testy wykazały, że szerokość pasma szumu analogowego wynosi 3,2 MHz, a pasma cyfrowego 1 MHz. Prawdopodobieństwo wystąpienia bitu „0” wynosi 49,87% a bitu „1” 50,13%.

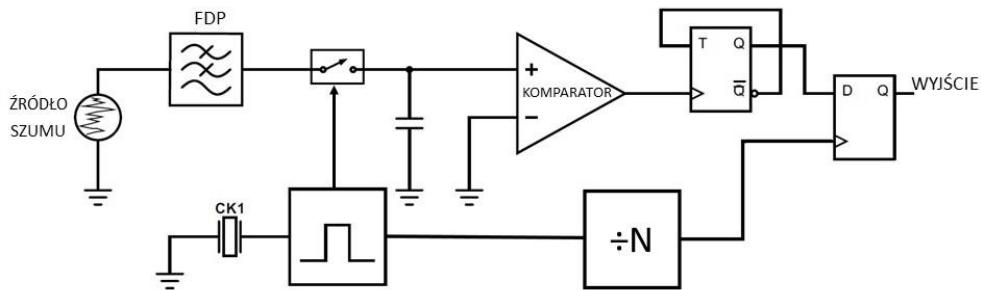
Kolejnym przykładem generatora TRNG wykorzystującego szum jest układ zaprojektowany przez firmę Intel [12]. Przedstawiony na rysunku 2.6 generator wykorzystuje źródło szumu i dwa oscylatory. Jeden o małej częstotliwości i drugi o częstotliwości znacznie większej. Wzmocniony szum termiczny jest wykorzystywany do modulacji częstotliwości oscylatora o małej częstotliwości. Tak powstały zaburzony sygnał



Rysunek 2.6 Generator TRNG firmy Intel

jest następnie próbkowany z częstotliwością drugiego oscylatora. Powstały sygnał losowy jest jednak silnie obciążony i wymaga korekcji. W projekcie zastosowano korektor von Neumanna [11]. Po korekcji generator jest w stanie dostarczyć losowe bity z szybkością 75 kb/s. Dalszy proces przetwarzania odbywa się nie sprzętowo a programowo. Dla poprawy właściwości statystycznych ciągu losowego wykorzystano funkcję skrótu SHA-1 (ang. *Secure Hash Algorithm*) [20]. Wariantem tego typu generatora jest konstrukcja zaproponowana w [17]. Wykorzystanie dodatkowego ogranicznika i przetwornika analogowo cyfrowego, którego wyjście jest dodawane w sprzężeniu zwrotnym do wzmacnionego sygnału szumu, pozwoliło uzyskać szybkość wytwarzania ciągów losowych rzędu 1,4 MHz. Podobne rozwiązanie zaproponowano w [21] z taką różnicą, że wolny oscylator wytwarza zaszumiony przebieg trójkątny. Generator dostarcza ciągów losowych z przepływnością 10 Mb/s.

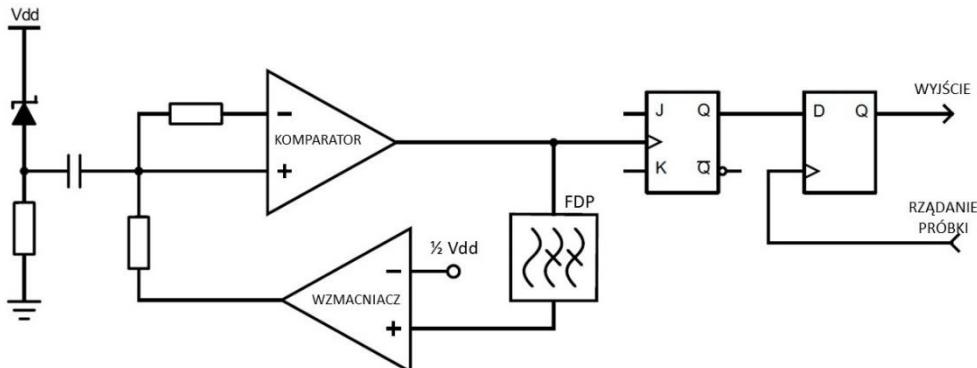
Generator z rysunku 2.7 zaproponowany w publikacji [22] jest kolejnym przykładem wykorzystania źródła szumu do wytwarzania losowych bitów. Analogowy szum jest



Rysunek 2.7 Schemat blokowy generatora TRNG z układem próbującym pamiętającym

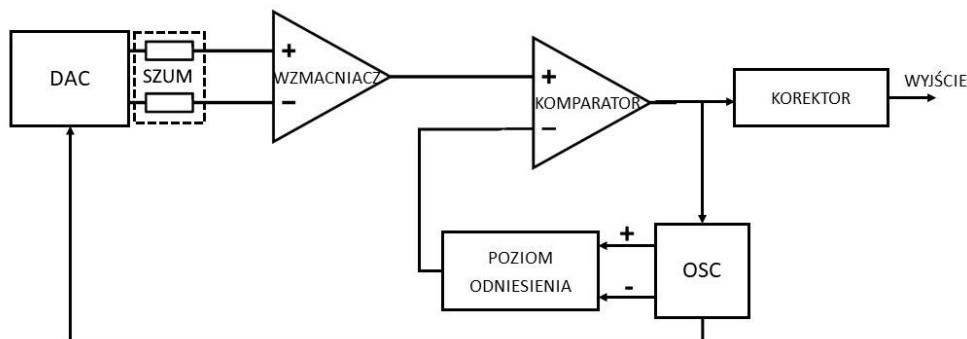
okresowo próbkowany z częstotliwością f_{CK1} i porównywany z wartością progową komparatora. Jak tylko na wyjściu komparatora pojawia się sygnał logicznej „1” przerzutnik typu T zmienia swój stan na przeciwny. Wyjście przerzutnika typu T jest próbkowane sygnałem o mniejszej częstotliwości niż był próbkowany szum. Jest to spowodowane tym, że przetwarzanie analogowo-cyfrowe za pomocą komparatora zajmuje jakiś czas, a przerzutnik typu T dzieli jeszcze przepływność sygnału przez 2. Więc, aby zapobiec sytuacji, że na wyjściu otrzyma się ciąg tych samych wartości, należy sygnał z wyjścia przerzutnika T próbować z N razy mniejszą częstotliwością. Odpowiedni dobór parametru N zmniejsza korelację między bitami. Podobnym przykładem jest układ zaproponowany w [23]. Jako źródło szumu wykorzystuje szum śrutowy powstały w diodzie Zenera, a komparator wyposażony jest w układ do automatycznej regulacji poziomu odniesienia. Układ ten zapewnia automatyczną regulację niezrównoważenia między bitami

„0” i „1”. Zrezygnowano tutaj z układu próbkującego pamiętającego a losowe bity są pobierane na żądanie użytkownika. Układ przedstawiony jest na rysunku 2.8. Generator może dostarczać rzeczywiście losowych bitów z szybkością 1,2 Mb/s. Innym przykładem



Rysunek 2.8 Generator TRNG z automatyczną regulacją poziomu odniesienia komparatora

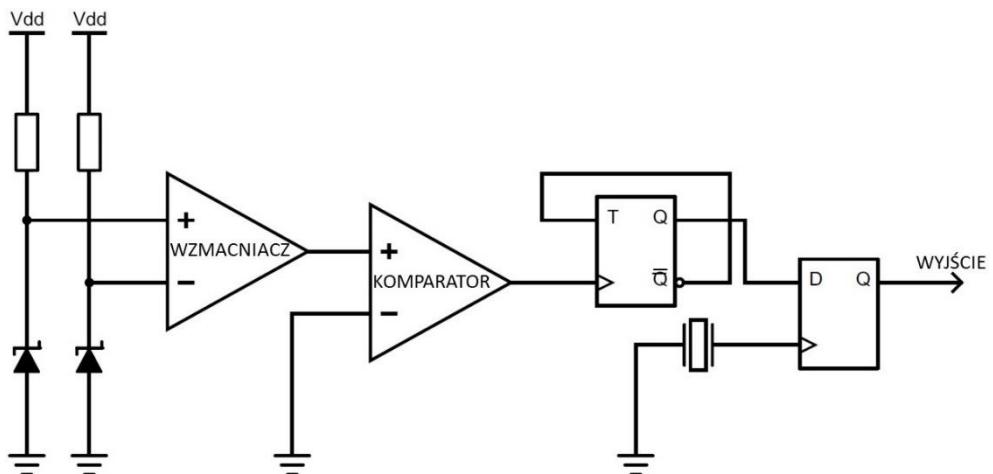
generatora, w którym wykorzystano automatyczną regulację poziomu odniesienia komparatora jest układ z rysunku 2.9 [24]. Generator został zaprojektowany i wykonany jako ASIC do wykorzystania w intelligentnych kartach mikroprocesorowych SC (ang. *Smart Card*). Szum termiczny powstały w zintegrowanych rezystorach jest wzmacniany za pomocą wzmacniacza różnicowego i porównywany z napięciem odniesienia komparatora. Układ regulacji niezrównoważenia działa dwustopniowo.



Rysunek 2.9 Generator z automatycznym równoważeniem bitów

Najpierw przetwornik analogowo cyfrowy A/C dobiera odpowiednie napięcie między rezystorami, aby wyeliminować niezrównoważenie wzmacniacza, następnie dobierane jest takie napięcie odniesienia komparatora, aby bity na jego wyjściu pojawiały się z jednakowym prawdopodobieństwem. W generatorze zastosowano post-processing do poprawy korelacji między bitami. Generator może wytwarzać ciągi losowe o przepływności 40 Mb/s.

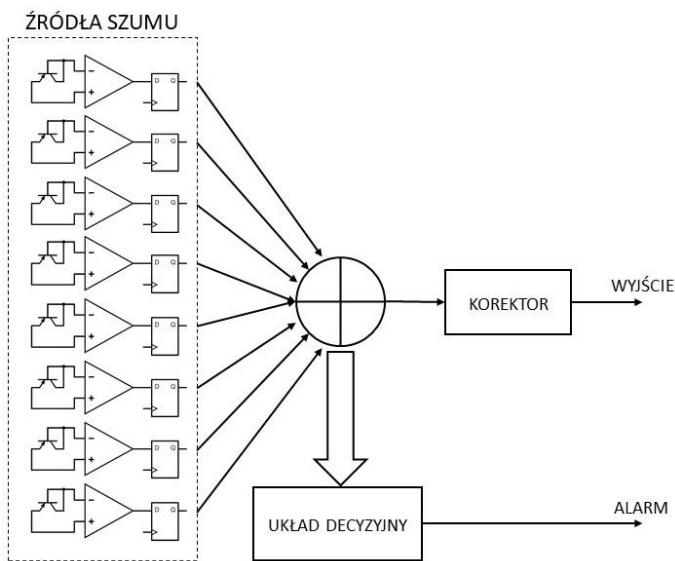
Na rysunku 2.10 przedstawiono generator TRNG z dwoma diodami Zenera [25]. Wzmacniacz operacyjny wyznacza i wzmacnia różnicę napięć między diodami. Następnie



Rysunek 2.10 Generator TRNG z dwoma diodami Zenera

wzmocniony szum wchodzi na przerzutnik z wejściem Schmitta gdzie następuje dyskretyzacja sygnału i wychwycone zostaje przejście między bitami „0” i „1”. Wyjściowy przerzutnik zapewnia synchronizację z zegarem. Generator dostarcza losowych bitów z przepływnością 500 kb/s.

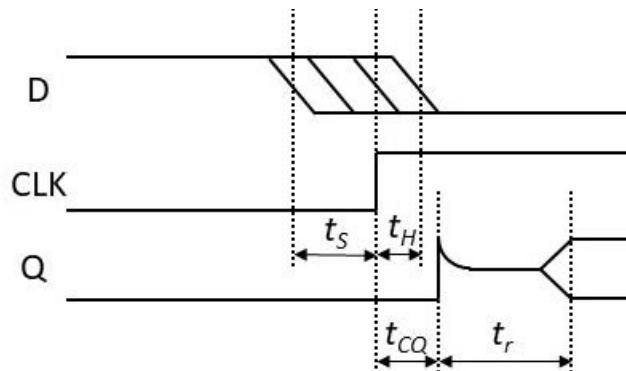
W pracy [26] przedstawiono generator złożony z ośmiu źródeł szumu o rozkładzie Poissona. Schemat układu pokazano na rysunku 2.11. Każde źródło szumu zostało zbudowane w ten sam sposób, wykorzystując złącze półprzewodnikowe tranzystora. Źródła te są próbkowane za pomocą przerzutnika typu D, a strumienie wyjściowe łączone sumą modulo 2 w jeden ciąg binarny. Nad prawidłową pracą generatora czuwa układ kontroli losowości. Generowany ciąg spełnia wszystkie testy statystyczne z pakietu NIST, FIPS 140 jak również pakiet testów Marsaglii – DIEHARD. Przepływność wytwarzanych ciągów wynosi 8 Mb/s.



Rysunek 2.11 Generator TRNG z 8 niezależnymi źródłami szumu

2.3.2. GENERATORY TRNG WYKORZYSTUJĄCE STANY METASTABILNE

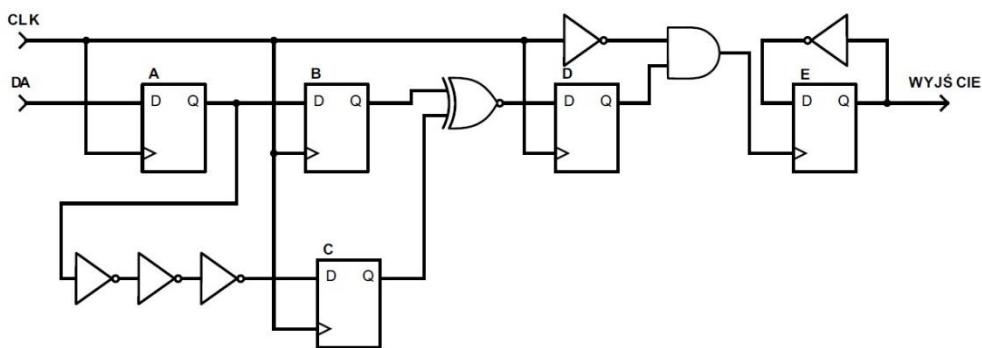
Zjawisko metastabilności ma miejsce w elektronicznych układach przełączających takich jak różnego rodzaju przerzutniki czy bramki logiczne. Warunkiem prawidłowego funkcjonowania przerzutników jest zachowanie określonych relacji czasowych między zmianami sygnałów na wejściu informacyjnym D a zmianami sygnału zegarowego CLK. Zostało to zilustrowanie na rysunku 2.12. Istotnym parametrem jest czas t_s podczas którego



Rysunek 2.12 Zjawisko metastabilności

stan na wejściu D przerzutnika musi być ustalony przed wystąpieniem zbocza na wejściu CLK. Przez czas t_H nie powinno dochodzić do zmian sygnału na wejściu D po wystąpieniu zbocza sygnału CLK. Niespełnienie tych wymogów może skutkować zwiększeniem czasu propagacji t_{CQ} o wartość t_r lub zatrzaśnięciem losowej wartości na wyjściu Q przerzutnika.

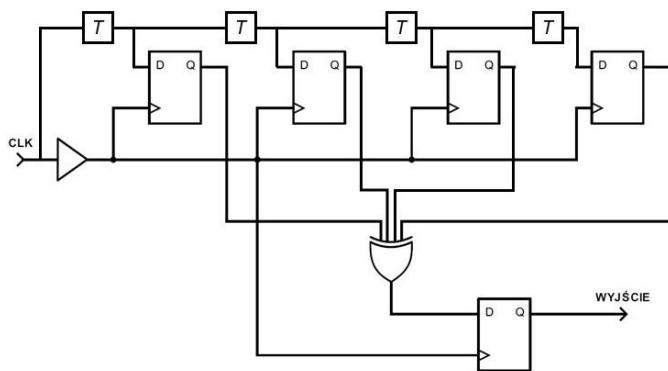
W oparciu o zjawisko metastabilności powstało wiele rozwiązań generatorów liczb rzeczywiście losowych. Jednym z najprostszych rozwiązań opartych na jednym przerzutniku jest układ z rysunku 2.13 [27]. Przerzutnik, który jest wprowadzany w stan



Rysunek 2.13 Schemat generatora wykorzystującego stany metastabilne przerzutnika typu D

metastabilny jest oznaczony jako A. Jego wyjście jest prowadzone dwoma równoległymi ścieżkami; jedna droga to bezpośrednie połączenie z przerzutnikiem B, a druga przez linie opóźniającą złożoną z trzech inwerterów do przerzutnika C. Jeżeli wystąpił stan metastabilny, stany wyjść przerzutników B i C będą takie same, więc na bramce

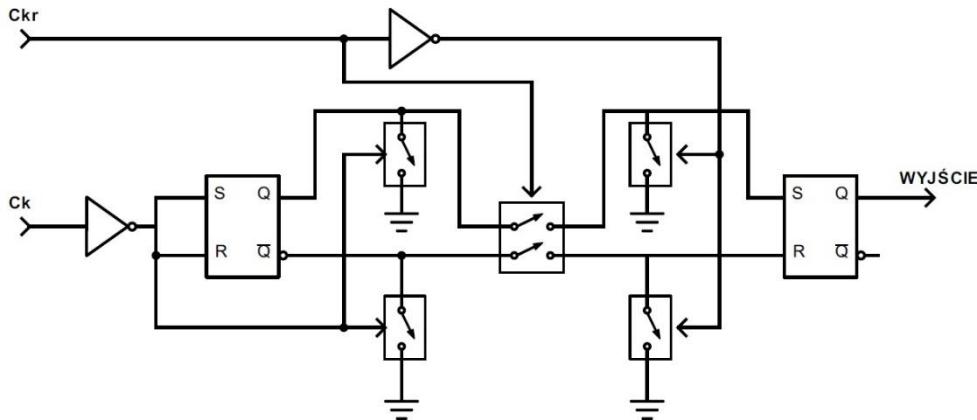
XOR-NOT, będzie stan wysoki. Wystąpienie takiego stanu jest rejestrowane przez przerzutnik D, a następnie sygnał wędruje do przerzutnika E, którego wyjście jest negowane, kiedy pojawia się stan metastabilny. Układ dostarcza ciągi losowe z niewielką szybkością wynoszącą 25 b/s. Generator z [28] wykorzystuje podobną właściwość z tą różnicą, że elementem wprowadzanym w stan metastabilny jest zatrzask (ang. *latch*). W projekcie uwzględniono automatyczną redukcję niezrównoważenia generatora. Całość zaimplementowano jako układ ASIC. Generator dostarcza losowych ciągów z szybkością 500 b/s, a po wykorzystaniu dodatkowego przetwarzania 50 kb/s. Inne bardzo zbliżone rozwiązanie to tak zwane generatorы ciągów losowych z otwartą pętlą [29], [30], [31], [32], [33], [34], przedstawione na rysunku 2.14 . Generator wykorzystuje wiele przerzutników



Rysunek 2.14 Wieloelementowy generator TRNG z otwartą pętlą

wprowadzanych w stan metastabilny. Na linię danych przerzutnika i linię zegarową podaje się ten sam sygnał. Przez odpowiedni dobór opóźnienia, linia danych może być próbkowana w monecie, gdy poziom napięcia wynosi połowę maksymalnego napięcia zasilającego. Wtedy szумy termiczne i środowiskowe powodują losową zmianę na wyjściu przerzutnika. Następnie wyjścia przerzutników są łączone za pomocą sumy modulo 2, tworząc strumień bitów losowych lub każde wyjście przerzutnika jest oddzielnym bitem losowego słowa [35]. Ponieważ bardzo trudno jest dobrać odpowiednie opóźnienie między linią danych a linią zegarową, w pracach [34] i [33] zastosowano rozwiązanie automatycznej regulacji opóźnienia. Generatorы z otwartą pętlą mogą być implementowane w układach FPGA lub jako specyfikowane układy cyfrowe. Różne technologie wykonania pozwalają uzyskać różną przepływność bitową generowanych ciągów, która waha się od kilku Mb/s do 20 Mb/s.

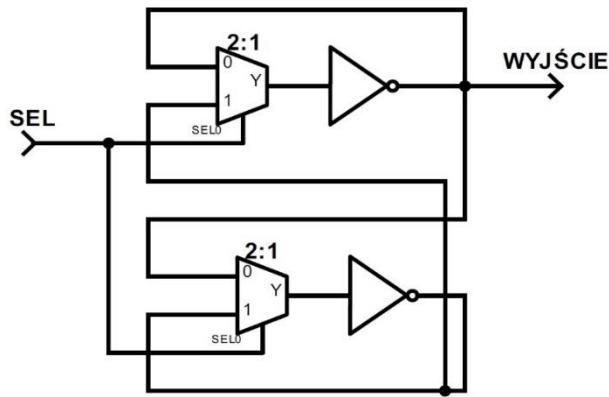
W pracy [36] autorzy proponują wykorzystanie przerzutnika RS na którego wejścia danych R oraz S podawany jest naprzemiennie sygnał „0” i „1”. Schemat układu przedstawia rysunek 2.15. W chwili jednoczesnego podania na wejścia R oraz S tego samego poziomu sygnału, przerzutnik znajduje się w tak zwanym stanie zabronionym. W stanie tym nie można przewidzieć jak zachowa się wyjście przerzutnika. Losowy stan na wyjściu jest następnie zatrzaskiwany w drugim przerzutniku. Warunkiem koniecznym



Rysunek 2.15 Generator TRNG z przerzutnikiem RS

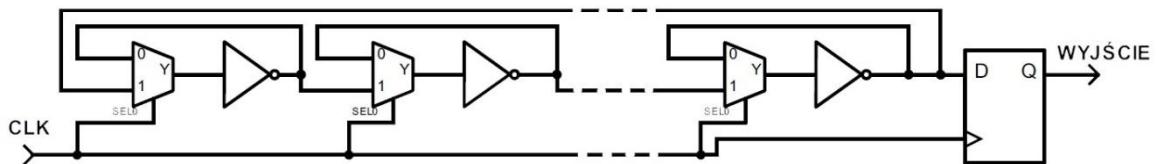
do wystąpienia stanu metastabilnego w przerzutniku RS jest jego budowa zapewniająca całkowitą symetrię dla sygnałów R oraz S. Metodę tę rozwinięto w pracy [37] zastępując przerzutnik RS parą sprzężonych inwerterów, gdzie wejście jednego zostało połączone z wyjściem drugiego, a wejście drugiego połączono z wejściem pierwszego. Na wejścia tak połączonych negatorów podawano naprzemiennie sygnał „0” i „1” z częstotliwością 10 MHz wprowadzając układ w zakres pracy metastabilnej. Rozwiążanie z przerzutnikami RS zostało również wykorzystane w [38]. Autorzy zaimplementowali przerzutniki RS w blokach logicznych układu FPGA, następnie wyjścia 256 przerzutników połączono sumą modulo 2 uzyskując losowy ciąg bitów o przepływności 12,5 Mb/s.

Innym rozwiązaniem jest zastosowanie układu bistabilnego [39]. Generator losowy z takim układem przedstawia rysunek 2.16. Składa się z dwóch multiplekserów i dwóch inwerterów połączonych tak, aby tworzyły układ metastabilny. Kiedy wejście SEL jest w logicznym stanie niskim „0”, wówczas układ zachowuje się jak dwa oscylatory pierścieniowe. Kiedy na wejściu SEL jest stan wysoki „1”, wtedy układ jest kaskadowym połączeniem dwóch inwerterów. Stany metastabilne powstają w momencie przełączania wejścia SEL ze stanu „0” na „1” i nie można przewidzieć w jakim stanie znajdzie się wyjście układu. Podobną zasadę wykorzystującą stany metastabilne oscylatora pierścieniowego zaproponowano w [40]. Generator jest szeregowym połączeniem wielu



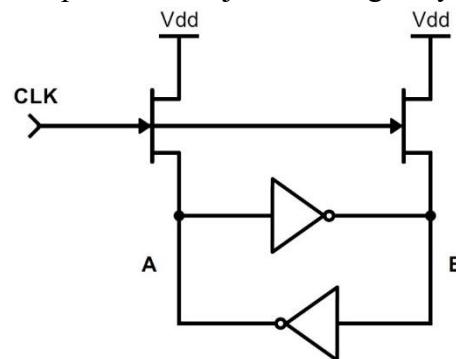
Rysunek 2.16 Generator losowy z układem bistabilnym

oscylatorów pierścieniowych złożonych z inwertera i multipleksera. Do prawidłowej pracy niezbędna jest nieparzysta liczba inwerterów (rys. 2.17). Stany nieustalone powstają w chwili przełączania multiplekserów między dwoma stanami. Pierwszy stan pracy to szybkie oscylacje pojedynczego oscylatora pierścieniowego, a drugi stan pracy to wolne oscylacje oscylatora złożonego z szeregu inwerterów. Przerzutnik na wyjściu układu zatrzaszcza losową wartość. Generator zaimplementowano w układzie FPGA może dostarczać losowych bitów z szybkością 35 Mb/s. Kombinacją dwóch wyżej wymienianych metod jest generator opisany w pracy [41].



Rysunek 2.17 Generator TRNG z metastabilnymi oscylatorami pierścieniowymi

Połączenie w pierścień dwóch kluczowanych inwerterów jest kolejną metodą pozwalającą wykorzystać stany metastabilne do wytwarzania losowych bitów [42], [43]. Na rysunku 2.18 przedstawiono układ takiego generatora. Generator może być w prosty sposób zaimplementowany w każdym urządzeniu cyfrowym. Kiedy sygnał zegarowy zmienia stan z „0” na „1” tranzystory zostają wyłączone, punkty A i B znajdująca się wcześniej na tym samym potencjale przechodzą w stan metastabilny. Dzięki występowaniu szumu termicznego nie można przewidzieć jaki stan logiczny ustabilizuje się w punktach

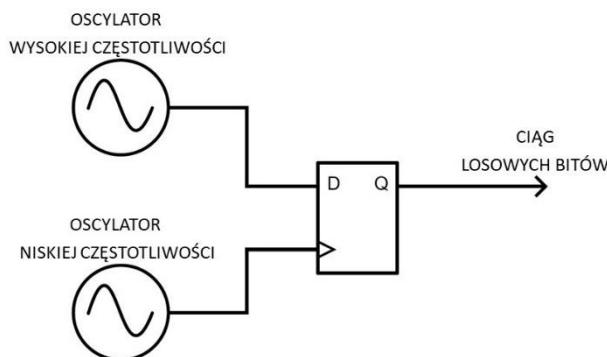


Rysunek 2.18 Źródło entropii złożone z dwóch inwerterów

A i B. Losowe bity są wytwarzane z każdym cyklem zegara, a generator wykonany w technologii 45nm CMOS (ang. *complementary metal–oxide–semiconductor*) dostarcza losowych bitów z szybkością 4 Gb/s [42]. Jednakże wytwarzane ciągi mają bardzo słabe właściwości statystyczne. Podobne rozwiązanie uzupełnione o drugi stopień w postaci przerzutnika RS i post-processing zaproponowano w pracy [44]. Generator rozwinięto w pracy [45] dodając kilkustopniowe przetwarzanie, zegar wielofazowy i układ regulacji opóźnienia przełączania. Całość zaimplementowano w układzie FPGA. Generator spełnia testy statystyczne, przy maksymalnej przepływności 5 Mb/s.

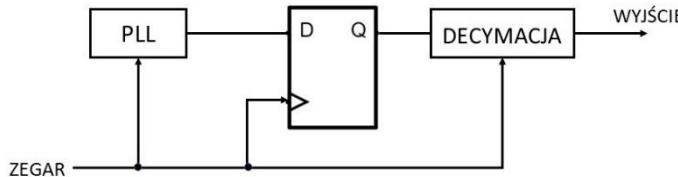
2.3.3. GENERATORY TRNG OPARTE NA ZJAWISKU SZYBKOZMIENNYCH FLUKTUACJI FAZOWYCH.

Najpopularniejszą grupę generatorów TRNG stanowią obecnie konstrukcje wykorzystujące szybkozmienne fluktuacje fazy (ang. *jitter*) oscylatorów jako źródło entropii. Popularność swoją zawdzięczają przede wszystkim prostej konstrukcji, możliwości integracji w systemach cyfrowych, łatwości implementacji w układach reprogramowalnych FPGA jak również dobrym właściwościom statystycznym i wysokiej szybkości wytwarzania ciągów bitów. Koncepcję generatora przedstawiono na rysunku 2.19. Wykorzystuje relacje między dwoma swobodnie oscylującymi generatorami, jednym o niskiej częstotliwości i drugim o wysokiej częstotliwości. Zasada działania polega na próbkowaniu sygnału obarczonego szybkozmiennymi fluktuacjami fazy z szybkiego generatora przez sygnał generatora wolnego. Jitter i wzajemny dryf częstotliwości oscylatorów powodują, że powstały w procesie próbkowania sygnał wyjściowy jest losowy [46]. Generator wykorzystujący opisywaną koncepcję zaproponowano w pracy [47]. W celu wyodrębnienia losowości z fluktuacji fazy sygnału zegarowego wykorzystano pętlę synchronizacji fazy. Sygnał wyjściowy z pętli fazowej jest próbkowany za pomocą przerzutnika typu D. Wykorzystanie pętli PLL umożliwia



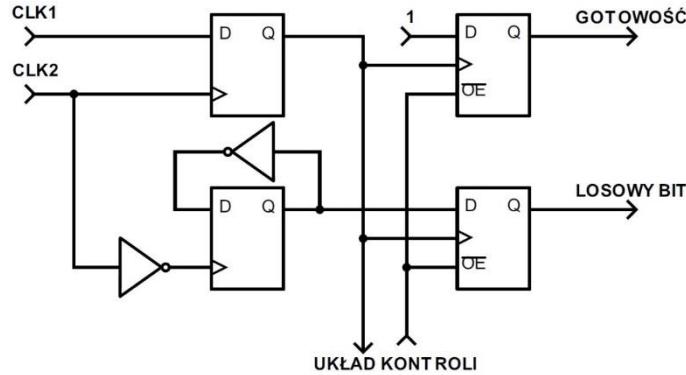
Rysunek 2.19 Generator TRNG z próbkowaniem oscylatora

zbieranie próbek sygnału w pobliżu narastającego zbocza sygnału z wyjścia pętli. Schemat układu przedstawiono na rysunku 2.20. Generator ten rozwinięto przez dodanie drugiej pętli fazowej [48] co pozwoliło zwiększyć jitter obecny w sygnale. Zaimplementowano go w układzie FPGA. Przedstawiono również rozwiązanie wykorzystujące serię elementów opóźniających, aby zwiększyć prawdopodobieństwo nałożenia się zboczy sygnału zegarowego i wyjściowego z pętli PLL. Generator dostarcza losowych bitów



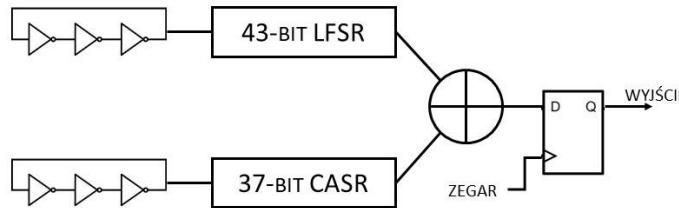
Rysunek 2.20 Generator TRNG z pętlą synchronizacji fazy

z przepływnością 1 Mb/s. Podobne rozwiązanie wykorzystujące oscylatory pierścieniowe i cyfrową pętlę fazową zawarto w [49]. Generator jest konstrukcją cyfrową jednakże nie jest implementowany w strukturach reprogramowalnych. Generuje ciągi z szybkością 100 kb/s. W pracy [50] zaproponowano model stochastyczny generatora z pętlami PLL. Proponowany model może być wykorzystany do testowania generatora i ustawienia odpowiednich relacji częstotliwościowych między sygnałem wyjściowym z pętli fazowej a sygnałem próbującym, co zwiększa poziom losowości i bezpieczeństwa generatora. Niektóre układy FPGA nie są wyposażone w pętle PLL lub dostępne pętle przeznaczono do innych zastosowań. Dlatego autorzy pracy [51] zaproponowali rozwiązanie bazujące na dwóch oscylatorach pierścieniowych i układzie sterującym. Wszystko zaimplementowane z wykorzystaniem tylko bloków logicznych układu FPGA. Proponowane rozwiązanie przedstawia rysunek 2.21. Sygnał z jednego generatora pierścieniowego jest próbowany przez sygnał z drugiego oscylatora w przerzutniku typu D. Oba generatory mają bardzo zbliżoną częstotliwość. Na wyjściu przerzutnika pojawi się sygnał o losowym okresie, zależnym od różnicy faz między sygnałami wejściowymi. Kolejny przerzutnik, który jest



realizacją przerzutnika typu T odmierza czas równy okresowi sygnału. Wyjście tego przerzutnika jest próbkowane przez sygnał wyjściowy z pierwszego przerzutnika typu D. Na wyjściu systemu otrzymuje się losowe bity o przepływności 600 kb/s. Ciąg wymaga jednak korekcji zrównoważenia bitów. Układ ten może być teoretycznie implementowany w dowolnej strukturze FPGA, jednak do prawidłowej pracy wymaga, aby częstotliwości oscylatorów pierścieniowych różniły się co najwyżej o 8%. Tak mała różnica częstotliwości sprawia, że układ wymaga ręcznego doboru opóźnienia, a elementy układu należy rozmieszczać wewnątrz struktury FPGA manualnie. Czyni to układ mało praktycznym i problematycznym do powielania nawet w układach reprogramowalnych tej samej serii. Modyfikację układu pozbawioną wymienionych wad i osiągającą przepływność 2 Mb/s zaproponowano w pracy [52].

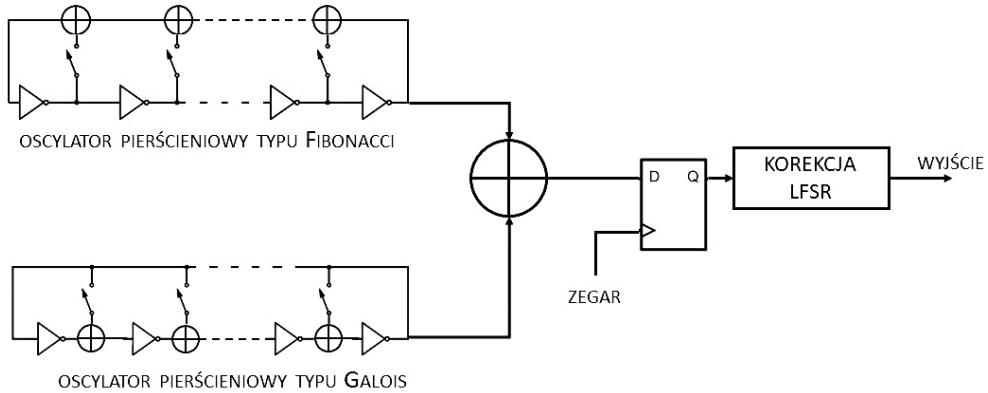
Inne rozwiązanie generatora losowego z oscylatorami pierścieniowymi zaproponowano w [53] i przedstawiono na rysunku 2.22. Proponowany generator składa



Rysunek 2.22 Generator z LFSR i CASR

się z dwóch oscylatorów pierścieniowych, które taktują dwa rejesty przesuwne. Jeden w konfiguracji liniowego rejestru ze sprzężeniem zwrotnym LFSR (ang. *Linear Feedback Shift Register*) i drugi w konfiguracji automatu komórkowego CASR (ang. *Cellular Automata Shift Register*). Rejestr LFSR bazuje na wielomianie pierwotnym i ma długość 43 bitów. Automat komórkowy ma długość 37 bitów. Wyjściowe bity z obu rejestrów są wybierane i mieszane w ścisłe określony sposób tworząc dwa ciągi o długości 32 bitów, które następnie są poddawane operacji modulo 2. Operacja mieszania bitów ma za zadanie zmniejszyć korelację między nimi [54]. Oscylatory pierścieniowe, które taktują rejesty nigdy nie są zatrzymywane nawet jeśli generator nie jest używany. Przez to LFSR i CASR, po odpowiednio długim okresie czasu, wydają się być w nieokreślonym stanie. Generator został skrytykowany w pracy [55], w której pokazano teoretyczne, że entropia oscylatorów pierścieniowych jest zbyt mała, aby zagwarantować bezpieczeństwo wytwarzanych ciągów, co w konsekwencji umożliwia predykcję kolejnych bitów.

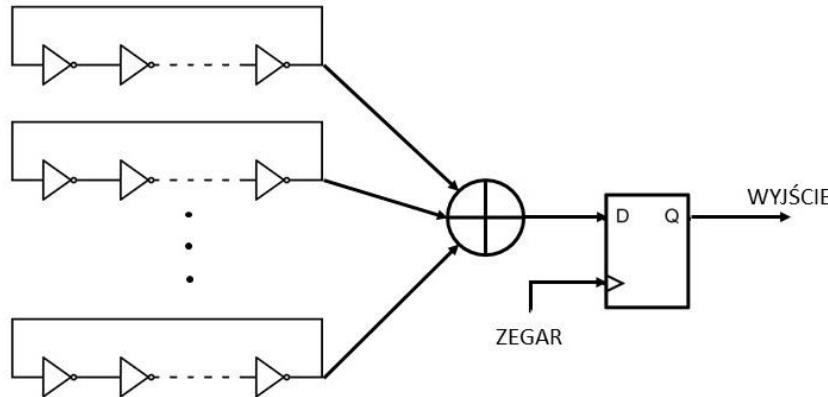
Generator TRNG wykorzystujący oscylatory pierścieniowe typu Galois i Fibonacci opisano w [56] i rozwinięto w [57]. Schemat proponowanego generatora przedstawia rysunek 2.23. Generator bazuje na dwóch oscylatorach z liniowym sprzężeniem zwrotnym. Tak samo jak w rejestrach LFSR oscylator opisuje się równaniem wielomianu pierwotnego z tą różnicą, że elementy pamiętające (przerzutniki) zastępuje się inwerterami. Ze względu na zmienność opóźnienia czasowego sprzężenia zwrotnego oscylatora, częstotliwość pracy zmienia się nieregularnie bardzo szybko, co pociąga za sobą zmiany stanów wyjściowych,



Rysunek 2.23 Generator z oscylatorami GARO i FIRO

których nie można przewidzieć. Aby zwiększyć odporność na ataki, wyjścia dwóch niezależnych oscylatorów są łączone za pomocą sumy modulo 2, a następnie próbkowane za pomocą przerzutnika typu D. Ostateczna przepustowość jest wprost proporcjonalna do częstotliwości próbkowania, którą należy wybrać w zależności od widma sygnału występującego na bramce XOR, aby uniknąć korelacji między bitami wyjściowymi. Generator wykorzystuje przetwarzanie końcowe oparte na rejestrze LFSR w celu zmniejszenia niezrównoważenia bitów. W [57] zaproponowano również metodę restartów jako ocenę losowości generowanych ciągów. Generator startuje się wiele razy z tymi samymi warunkami początkowymi, po czym porównuje się otrzymane ciągi. Kiedy otrzymane ciągi różnią się znacząco to można sądzić, że w generowaniu sekwencji losowej znaczący udział miały czynniki niedeterministyczne. Autorzy obserwowali analogowy przebieg sygnałów na wyjściu generatora TRNG za pomocą oscyloskopu. Generator ten można w pełni zaimplementować w układach FPGA.

Generator TRNG złożony z wielu oscylatorów pierścieniowych został przedstawiony w [8], [58] i [59]. Schemat rozwiązania pokazano na rysunku 2.24. Generator wykorzystuje dużą liczbę swobodnie oscylujących generatorów pierścieniowych. Każdy oscylator składa się z 13 inwerterów. Wyjścia generatorów są połączone za pomocą sumy modulo 2.



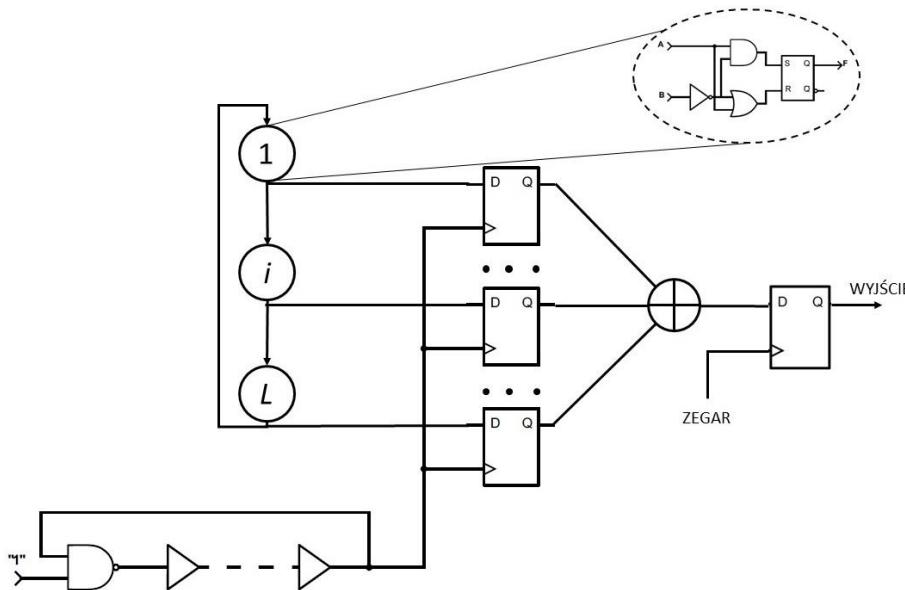
Rysunek 2.24 Łaczony generator TRNG

Wyjście sumy modulo 2 jest próbkowane za pomocą przerzutnika typu D, aby uzyskać sygnał losowy. Bazując na przeprowadzonej estymacji entropii wyznaczono funkcję post-processingu bazującą na kodzie cyklicznym. Pierwszą praktyczną realizację sposobu generowania losowych bitów przedstawiono w [59]. W pracy [8] zaproponowano generator TRNG złożony ze 110 oscylatorów pierścieniowych po 13 inwerterów każdy. Natomiast w [59] zaproponowano użycie 110 oscylatorów pierścieniowych po 3 inwertery każdy, co jest korzystniejsze ze względu na zasoby zajmowane przez układ. Zaproponowany generator jest w pełni cyfrową konstrukcją, niezależną od technologii [60], która może być z powodzeniem implementowana w różnych układach FPGA. Ponadto nie ma żadnych ograniczeń projektowych i nie wymaga manualnej interwencji podczas rozmieszczania elementów wewnętrz struktury FPGA. Generator wykorzystuje również nieznaczną ilość zasobów, a generowane ciągi mogą uzyskać wysoką przepływność rzędu kilku Mb/s [8]. W [61] zaproponowano modyfikację generatora [8] polegającą na dodaniu próbkującego przerzutnika typu D na wyjście każdego oscylatora pierścieniowego. Przerzutniki te są taktowane zewnętrznym oscylatorem np. kwarcowym. Wyjścia przerzutników tak jak w poprzednim rozwiązaniu łączy się za pomocą sumy modulo 2 w jeden strumień binarny, który dodatkowo jest próbkowany za pomocą przerzutnika typu D i tego samego generatora co wyjścia oscylatorów pierścieniowych. Otrzymany w ten sposób ciąg losowych bitów spełnia wszystkie testy statystyczne z pakietu NIST 800-22 bez potrzeby stosowania układów dodatkowego przetwarzania poprawiającego właściwości statystyczne, przy znacznie mniejszej liczbie oscylatorów pierścieniowych. Do wytwarzania ciągów losowych użyto ich 25. Zaimplementowany w układzie FPGA generator osiągnął przepływność 100 Mb/s. Dalsza optymalizacja generatora pozwoliła uzyskać przepływność 300 Mb/s [62]. Jednocześnie autorzy pracy [63] wykazali za pomocą symulacji, że modyfikacja polegająca na dodaniu przerzutników na wyjściu oscylatorów

pierścieniowych nie wpływa na pracę układu złożonego z idealnych komponentów. Wpływa jednak znacząco na poprawę właściwości statystycznych ciągów generowanych w rzeczywistym urządzeniu. Wykazano również w eksperymencie symulacyjnym, że ciągi wytwarzane przez generator z pracy [61] mogą spełnić wszystkie testy statystyczne NIST 800-22 nawet wówczas, gdy oscylatory pierścieniowe są pozbawione szybkozmiennych fluktuacji fazy, a więc nie posiadają źródła losowości. Do spełnienia testów statystycznych wystarczy zaledwie 18 idealnych oscylatorów pierścieniowych różniących się nieznacznie generowaną częstotliwością. Konkluzją pracy [63] jest stwierdzenie, że bardzo dobre właściwości statystyczne generatora [61] mogą być wynikiem mechanizmu deterministycznego, co wyklucza zastosowanie takiego generatora w kriptografii. Autorzy prac [64] i [65] przeanalizowali ten problem dokładniej. Wykorzystując mechanizm restartów i test chi-kwadrat pokazali, że nawet mała ilość losowości obecna w pojedynczym oscylatorze pierścieniowym ulega akumulacji wraz ze wzrostem liczby oscylatorów źródłowych. W rezultacie otrzymanie ciągu losowego dla zastosowań w kriptografii jest jak najbardziej możliwe z tym, że ciąg wyjściowy powinien być utworzony z co j -tego elementu ciągu wytwarzanego przez generator, co znacząco zmniejsza przepływność ciągu losowego z 300 Mb/s do 7,14 Mb/s [64]. Kontynuacja badań pokazała, że efektywną szybkość wytwarzania losowych ciągów można zwiększyć obniżając szybkość próbkowania [66]. Ten pozorny paradoks bierze się stąd, że dla danej liczby oscylatorów pierścieniowych losowość ciągu wyjściowego zależy także od wielkości różnicy pomiędzy częstotliwością próbkującą, która ustala przepływność ciągu wyjściowego, a częstotliwościami poszczególnych generatorów pierścieniowych. W pracy [66] uzyskano efektywną przepływność 20 Mb/s przy częstotliwości próbkowania równej 100 MHz. Ponieważ wszystkie oscylatory pierścieniowe mają identyczną strukturę i są umieszczone wewnątrz jednego układu scalonego mogą ulegać wzajemnym wpływom takim jak wzajemna synchronizacja przez iniekcję czy przeniki między poszczególnymi drogami sygnałów. Generator taki jest również narażony na ataki typu wstrzykiwanie częstotliwości [67]. Podczas tego ataku pewna liczba oscylatorów pierścieniowych blokuje się na wstrzykiwanej częstotliwości, eliminując niezależność generatorów i znacznie zmniejszając losowość wyjściowego strumienia bitów. Aby uniknąć tego typu ataków zaproponowano ręczną korektę długości opóźnienia w oscylatorach pierścieniowych [58], aby poszczególne oscylatory różniły się między sobą częstotliwością. To rozwiązanie na ogół daje satysfakcyjujące rezultaty, ale jest mało praktyczne. W pracy [68] zaproponowano sposób na zwiększenie odporności łączonego generatora

TRNG złożonego z oscylatorów pierścieniowych. Kluczowym punktem tej metody jest zapewnienie różnych nominalnych częstotliwości generatorów źródłowych, co utrudnia sprzążenie między sąsiednimi oscylatorami. Sprawia to, że generator TRNG jest mniej wrażliwy na lokalizacje projektu wewnątrz struktury FPGA i utrudnia przeprowadzenie ataku wstrzykiwania częstotliwości. Inną modyfikacją generatora łączonego przedstawiono w [69]. Metoda polega na połączeniu dwóch różnych generatorów TRNG. Podstawą konstrukcji jest generator łączony złożony z oscylatorów pierścieniowych [61] i generator GARO [56]. Pozwoliło to zmniejszyć wykorzystywane zasoby oraz osiągnąć efektywną szybkość wytwarzania ciągów losowych na poziomie 20 Mb/s przy częstotliwości próbkowania 100 MHz. Dalsze zwiększenie przepływności losowych bitów było możliwe przez wykorzystanie funkcji skrótu z rodziny SHA-2 jako post-processingu [70]. Zastosowanie funkcji skrótu poprawia właściwości statystyczne i zwiększa szybkość generowanych ciągów do około 36 Mb/s.

Inne konstrukcje korzystające z szybkozmiennych fluktuacji fazy jako źródła entropii to między innymi generatory wykorzystujące samotaktujące pierścienie STR (ang. *Self-Timed Rings*) [71], [72], [73], [74]. Schemat generatora TRNG z takimi pierścieniami pokazano na rysunku 2.25. Zasada działania opiera się o kontrolowaną ścieżkę linii



Rysunek 2.25 Samotaktyczny generator TRNG

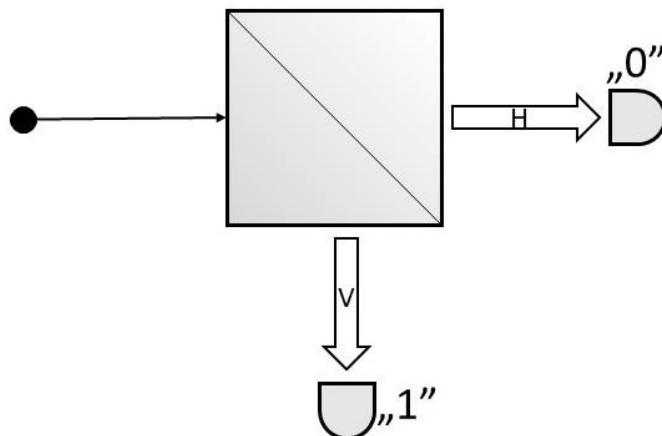
mikropotokowej zaproponowaną w [75], która została zamknięta w pierścień o L stopniach. Każdy stopień to bramka Müllera i inwerter. Powstały jitter propaguje się przez każdy stopień oscylatora pierścieniowego. Wyjście każdego stopnia jest próbkowane za pomocą przerzutnika typu D i łączone za pomocą sumy modulo 2 w strumień losowych bitów.

Taka konstrukcja zwiększa prawdopodobieństwo wychwycenia rzeczywistej losowości z sygnału oscylatora pierścieniowego. Generator jest całkowicie cyfrową konstrukcją i może być implementowany w układach FPGA. Generator może uzyskiwać wysokie przepływności losowych ciągów rzędu kilkuset Mb/s [73].

Zaproponowany w [76] niezależny technologicznie generator liczb rzeczywiście losowych to modyfikacja generatora [8] w którym zamiast oscylatorów pierścieniowych o stałej długości zaproponowano wykorzystanie oscylatorów o zmiennym opóźnieniu przez co zmienna jest też generowana częstotliwość. Generator został zaimplementowany w kilku układach FPGA o różnej technologii wykonania. Podobną koncepcję zaproponowano w [77] i [78], z tą różnicą, że zmienne opóźnienie oscylatora pierścieniowego zapewnia użycie programowalnego rezystora z pamięcią tzw. memristora. Taki generator może być wykonany z elementów dyskretnych lub jako specyfikowany układ scalony ASIC. Trzystopniowe oscylatory pierścieniowe i układ przechwytywania losowych bitów zaproponowano w [79]. Układ wyposażono w post-processing zautomatem komórkowym. Generator może być implementowany wewnątrz struktur FPGA, osiąga przepływność 50 Mb/s. Wykorzystanie logiki trójstanowej zaproponowano w pracy [80]. Trzy elementy kombinacyjne logiki trójstanowej tworzą oscylator pierścieniowy, każdy element wprowadza losową wartość opóźnienia przez co wytwarzane przez niego oscylacje mają losowy okres. Generator nie jest implementowany wewnątrz układu FPGA.

2.3.4. KWANTOWE GENERATORY CIĄGÓW LICZB RZECZYWIŚCIE LOSOWYCH

Kwantowe generatory ciągów liczb rzeczywiście losowych QRNG (ang. *Quantum Random Number Generator*) wykorzystują do wytwarzania losowych bitów zjawiska powstałe w wyniku oddziaływania elementów świata mikroskopowego takich jak atomy, elektrony, fotony itp. Bardzo popularnym rozwiązaniem jest wykorzystanie fotonów do wytwarzania losowych bitów, ponieważ są łatwe do tworzenia, manipulowania i wykrywania. Przykładowy generator pokazano na rysunku 2.26. Spolaryzowany kołowo foton pada na polaryzacyjny rozdzielacz wiązki. Rozdzielacz ten na jedno wyjście przepuszcza foton o polaryzacji poziomej a na drugie o polaryzacji pionowej. Spolaryzowany kołowo foton ma 50% szans przejścia na wyjście „0” oraz 50% szans przejścia na wyjście „1”. Generatory wykorzystujące opisaną zasadę są komercyjnie dostępne, a produkuje je między innymi firma ID Quantique [81]. Podstawowym

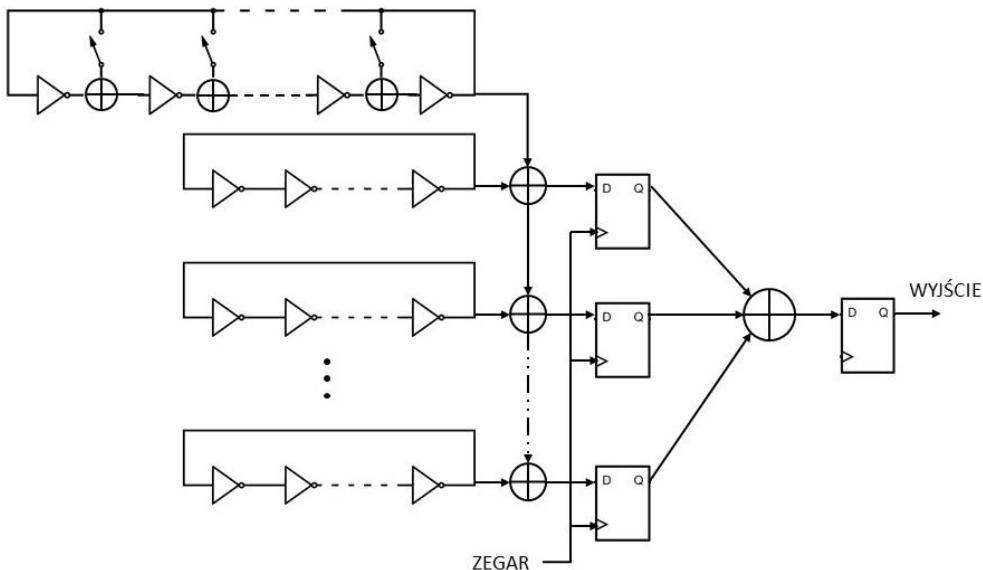


Rysunek 2.26 Kwantowy generator TRNG wykorzystujący polaryzację fotonów

problemem w projektowaniu kwantowych generatorów losowych jest to, że wymagają precyzyjnie dobranych elementów. Rozdzielacz wiązki musi być precyzyjnie ustawiony i dzielić wiązkę w stosunku 50/50, a detektory nie mogą mieć różnic między sobą. Niewielkie różnice czy efekty starzeniowe oraz temperaturowe mają niekorzystny wpływ na jakość wytwarzanych ciągów [82]. Efekty niekorzystne można zredukować stosując rozwiązańe z tylko jednym detektorem fotonów [83], jednak nadal rozdzielacz wiązki musi być precyzyjnie ustawiony. Generatorem wolnym od tych wad jest konstrukcja opisana w [84], która wykorzystuje tylko emiter i detektor fotonów. Wartość losową uzyskuje się przez pomiar odstępu przedziału czasu między dwoma fotonami. Jeżeli odstęp czasu między fotonami jest większy od wartości granicznej wówczas na wyjściu generatora otrzymujemy logiczną „1”, jeżeli odstęp czasu jest mniejszy od wartości granicznej wówczas na wyjściu jest logiczne „0”. Generatorów kwantowych nie można zaimplementować wewnątrz struktury FPGA. Są one dedykowanymi konstrukcjami a szybkości wytwarzania losowych bitów mogą osiągać wiele Gb/s.

2.3.5. HYBRYDOWE GENERATORY TRNG

Losowe generatory hybrydowe to konstrukcje, które do wytwarzania losowych ciągów wykorzystują więcej niż jedno z opisywanych wcześniej zjawisk. Takim generatorem może być wspomniane wcześniej połączenie generatora złożonego z wielu oscylatorów pierścieniowych z generatorem pierścieniowym typu Galois (rys. 2.27) [69]. Podobną metodę wykorzystuje projekt [85], tylko zamiast generatora GARO użyto drugi, inny generator łączony z oscylatorów pierścieniowych. Może być implementowany wewnątrz struktur reprogramowalnych, uzyskuje przepływność 95,37 Mb/s. Generator [17] wykorzystuje natomiast aż trzy różne zjawiska. Został wykonany jako układ scalony ASIC.

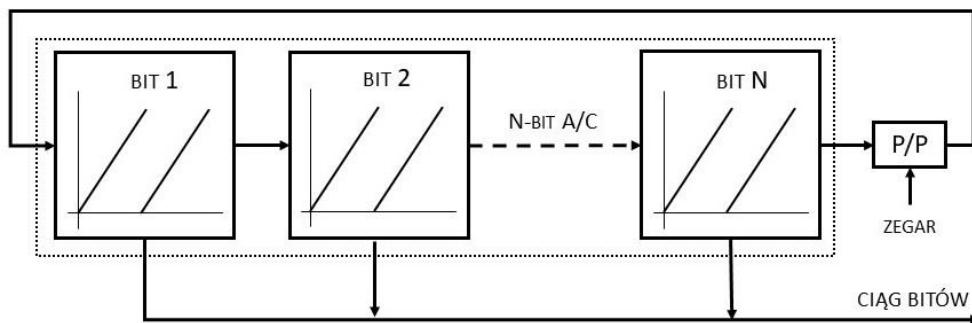


Rysunek 2.27 Hybrydowy generator TRNG

Jest połączeniem metody wzmacniania szumu, układu chaotycznego i próbkowania szybkozmiennych fluktuacji fazy generatora sterowanego prądem. Jego przepływność wynosi 1,4 Mb/s. Praca [86] wykorzystuje stany metastabilne przerzutnika i układ chaotyczny. Generator TRNG osiąga przepływność około 5 Mb/s i może być integrowany wewnątrz układu FPGA. Metastabilność przerzutnika RS w połączeniu z kluczowanym oscylatorem pierścieniowym tworzącym układ chaotyczny wykorzystuje generator [87]. Może być on implementowany w układach reprogramowalnych jednak wymaga kilku elementów zewnętrznych. Osiąga przepływność 1 Mb/s. Wykorzystanie oscylatorów pierścieniowych wytwarzających chaotyczne oscylacje opisano w pracach [88] i [89]. Generator takie mogą być implementowane wewnątrz struktur reprogramowalnych a uzyskiwane szybkości po przejściu przez układ post-processingu osiągają 2 Gb/s. Innym przykładem połączenia jittera i chaosu są generatorzy wykorzystujące sieci boolowskie [90], [91]. Konstrukcja generatora to połączone w pierścień bramki sumy modulo 2, przy czym jedna z nich posiada negowane wyjście. Bramki są połączone ze sobą w taki sposób, aby utworzyć sieć boolowską. Takie połączenie wywołuje chaotyczne oscylacje na wyjściu generatora, które jest próbkowane przy pomocy przerzutnika typu D. Generator taki jest w pełni cyfrową konstrukcją, która może być implementowana w strukturach FPGA. Generator z pojedynczym pierścieniem [90] i [91] osiąga przepływność 100 Mb/s i spełnia testy statystyczne NIST 800-22. Natomiast implementacja generatora wykorzystująca 128 pierścieni uzyskała przepływność 12,8 Gb/s [90].

2.3.6. GENERATORY TRNG WYKORZYSTUJĄCE CHAOS

Generatory chaosu mogą być uznawane za generatory ciągów losowych ponieważ równania opisujące układ chaotyczny posiadają znaczną wrażliwość na parametry wejściowe. Nawet nieznaczająca zmiana na wejściu układu powoduje znaczną zmianę na jego wyjściu. Elektroniczne układy chaotyczne to konstrukcje analogowe, więc z powodu szumów nie jesteśmy w stanie zagwarantować identycznych warunków pracy układu chaotycznego, przez co generowany ciąg bitów może być uznawany za losowy. Chaos w układach elektronicznych może być wytworzony za pomocą dyskretyzacji i przetwarzania sygnałów analogowych. Popularnym rozwiązaniem jest zastosowanie układu z przełączanymi pojemnościami [92], [93] lub przełączanym prądem [94], [95]. Inny przykład systemu chaotycznego wykorzystuje potokowy przetwornik analogowo-cyfrowy [96], gdzie szum kwantyzacji jest próbkiowany, pamiętany i przesyłany na wejście przetwornika (rys. 2.28). Przedstawiony na schemacie 2.28 układ może być opisany za



Rysunek 2.28 Przykład układu chaotycznego

pomocą przesunięć Bernoulliego, spełniających zależność [17]:

$$X_n = [2(X_{n-1} + e(n))] \bmod 1, \quad (2.1)$$

gdzie $e(n)$ reprezentuje szum gaussowski. Zostało udowodnione, że kolejne iteracje systemu opisanego równaniem (2.1) wytwarzają ciągi, które posiadają płaskie widmo i rozkład równomierny [97]. Dodatkowo rozbieżność trajektorii związana z chaosem i szumem powoduje, że wytwarzane ciągi są nieprzewidywalne, co pozwala wykorzystać ten generator jako źródło rzeczywiście losowych bitów. Słabością tego rozwiązania może być niedostateczna rozdzielcość przetwornika analogowo-cyfrowego przez co wytwarzane ciągi mogą być w znacznym stopniu obciążone. Dlatego to rozwiązanie wykorzystuje się w połączeniu z układami post-processingu. Niewątpliwą zaletą tego typu układów jest możliwość integracji w układach cyfrowych i w cyfrowych układach reprogramowalnych wyposażonych w zintegrowany przetwornik analogowo-cyfrowy.

Efektywnym źródłem ciągów losowych integrowanym w układach cyfrowych może być również pętla synchronizacji fazy PLL (ang. *Phase Locked Loop*) wprowadzona w stan pracy chaotycznej [98], [99], [100], [101]. Stan ten można uzyskać przez podanie na wejście pętli synchronizacji fazy sygnału, którego kątowa dewiacja częstotliwości przekroczy pewną wartość krytyczną. Również stosowanie pętli z niesymetrycznym wzmacnieniem lub zbyt dużym wzmacnieniem wprowadza pętlę fazową w stan pracy chaotycznej [99].

2.3.7. RZUT MONETĄ, KOŚCIĄ

Uważamy, że rzucając monetą nie jesteśmy w stanie przewidzieć wyniku, a prawdopodobieństwo otrzymania „orła” lub „reszki” jest identyczne i wynosi 50%. Dzieje się tak gdyż rzut monetą jest operacją bardzo skomplikowaną, na którą wpływa bardzo dużo czynników tak, że nie jesteśmy w stanie opisać go w zadowalający sposób. Gdyby jednak rozłożyć taki rzut na mniejsze fragmenty i poznać wszystkie czynniki wpływające na ruch monety, takie jak: rozkład masy monety, jej kształt, początkowe przyspieszenie, opory i ruchy powietrza, a także inne wartości i zjawiska mające wpływ na proces rzutu, można by przewidzieć jego wynik. Tak samo można postąpić analizując rzut kościami czy pękanie żółwiej skorupy znane już w 11 wieku p.n.e. Zjawiska te, ze względu na nadmierną złożoność nie mogą być opisane za pomocą modelu deterministycznego. Dlatego uznajemy je za losowe.

2.3.8. INNE GENERATORY POSTRZEGANE JAKO LOSOWE

W pracy [102] jako źródło entropii generatora losowego wykorzystano kliknięcia myszy komputerowej. Specjalne oprogramowanie przechwytuje współrzędne ekranu i na podstawie ukrytego obrazu odczytuje współczynniki koloru danego punktu obrazu. Przeliczone współczynniki tworzą liczbę. Za każdym kliknięciem generowana jest nowa liczba. W artykułach [103] i [104] zaproponowano generator liczb losowych, który wytwarza 256-bitową liczbę losową wykorzystując ruch myszy komputerowej. Aby wyeliminować efekt podobnych wzorców ruchu generowanych przez tego samego użytkownika, zastosowano przetwarzanie wytwarzanych liczb oparte o algorytmy chaosu. Autorzy [105] przedstawili sposób wytwarzania liczb losowych wykorzystując szum otoczenia zbierany przez mikrofon wbudowany w przenośny komputer. W [106] przedstawiono sposób pozyskania losowych bitów z pamięci SRAM (ang. *Static Random Access Memory*). Pokazano, że za każdym włączeniem pamięci stan 512 bitów

jest losowy. Po zastosowaniu funkcji skrótu ciąg 128 bitowych słów spełnia testy statystyczne NIST 800-22. Praca [107] pokazuje sposób pozyskiwania losowych liczb z węzłów sieci sensorowej. Każdy taki węzeł wyposażony jest w zestaw sensorów np. czujnik temperatury, wilgotności, ciśnienia, nasłonecznienia itp., który może być wykorzystany jako źródło entropii. W danej chwili wartość parametrów środowiskowych pozyskanych z czujników jest wartością losową. Po zastosowaniu dodatkowego przetwarzania ciąg liczb spełnia testy statystyczne.

2.3.9. PODSUMOWANIE ROZWIĄZAŃ GENERATORÓW LICZB RZECZYWIŚCIE LOSOWYCH

Projektowanie generatorów liczb rzeczywiście losowych nie jest łatwe. Generator taki nie dość, że musi sprostać rygorystycznym testom statystycznym to wymaga się, aby wytwarzane ciągi były nieprzewidywalne mimo, że znana jest struktura układu. Powinien też posiadać zrozumiały model statystyczny, gdyż może się okazać, że spełnienie testów statystycznych zawdzięczamy zjawiskom deterministycznym [63]. Dlatego niezbędne jest testowanie generatora ze źródłem entropii za pomocą dodatkowych mechanizmów, na przykład restartów [108]. Inną trudnością jest dobór odpowiedniego algorytmu post-processingu. Źródła entropii często nie mają rozkładu równomiernego i wytwarzane za ich pomocą ciągi są obciążone. Algorytm post-processingu powinien niwelować obciążenie, a jego koncepcja powinna być zrozumiała i możliwa do poparcia odpowiednimi równaniami matematycznymi. Oczekuje się, że generatory będą odporne na różnego rodzaju ataki. Niestety źródła entropii są często wrażliwe na parametry zewnętrzne takie jak napięcie zasilania czy temperatura otoczenia. Należy je konstruować tak, aby zminimalizować wpływ czynników zewnętrznych na pracę źródła entropii. Generatory oparte o źródło szumu są szczególnie narażone na te procesy. Dlatego zamiast jednego źródła szumu często wykorzystuje się dwa takie same źródła i wzmacniacz różnicowy, który wzmacnia różnicę sygnałów występujących na obu źródłach. Ten zabieg eliminuje wpływ tętnień napięcia zasilania na losowość wytwarzanych ciągów. Wpływ temperatury na źródło entropii nie jest łatwy do wyeliminowania [109]. Inną poważną wadą generatorów ze źródłem szumu jest niewielka szybkość wytwarzanych ciągów. Konstrukcje cyfrowe wydają się być pozbawione efektu oddziaływanego tętnień napięcia zasilania na źródło losowości. Jednak i w tym przypadku problemem jest oddziaływanie temperatury. Precyzyjnie dobierane opóźnienia w generatorach wykorzystujących efekty metastabilności na skutek rozszerzalności cieplnej i różnej szybkości propagacji sygnału w przewodniku mogą spowodować zanik efektów metastabilności. Dlatego takie

generatory należy projektować z wykorzystaniem mechanizmu kontroli losowości i sprzężeniem zwrotnym umożliwiającym korekcję opóźnienia [34]. Jedną z najlepiej przebadanych grup generatorów są generatory wykorzystujące fluktuacje fazy oscylatora pierścieniowego. Chociaż i tutaj można zauważać wpływ temperatury na szybkość oscylacji [110], jednak zmiana częstotliwości jest na tyle mała, że nie wpływa znacząco na jitter obecny w generowanym sygnale. W pracy [67] pokazano, że taki generator może być podatny na atak typu wstrzykiwanie częstotliwości. Jednak autorzy wykorzystali do budowy generatora tylko dwa oscylatory pierścieniowe wykonane z elementów dyskretnych w technologii TTL (ang. *Transistor Transistor Logic*). Zaimplementowany w technologii FPGA generator z wieloma oscylatorami nie wykazał podatności na tego typu atak [111]. Również przeniki i wzajemna synchronizacja nie wpływają na bezpieczeństwo wytwarzanych ciągów [112]. Istotną wadą takiego generatora jest duże zużycie energii z powodu wykorzystania wielu oscylatorów pierścieniowych pracujących z wysokimi częstotliwościami. W celu wyeliminowania wad generatorów losowych proponuje się często konstrukcje hybrydowe. Z punktu widzenia podatności na ataki i podsłuchy, wydaje się zasadne, aby generator liczb rzeczywiście losowych był konstrukcją, która może być integrowana razem z całym systemem kryptograficznym wewnątrz jednego układu scalonego. Takiej możliwości nie mają generatory wykorzystujące efekty kwantowe.

3. UKŁADY FPGA

Układy FPGA to układy cyfrowe, które posiadają możliwość konfiguracji przez użytkownika po ich wyprodukowaniu. Można je skonfigurować w taki sposób, aby wykonywały dowolną funkcję logiczną. Zostały jednak zaprojektowane w szczególności do pracy synchronicznej. Posiadają one bardzo zbliżone właściwości do specyfikowanych układów ASIC z tą różnicą, że układy FPGA mogą być wielokrotnie konfigurowane w docelowym systemie. Dzięki temu możliwe jest szybkie i stosunkowo proste prototypowanie i testowanie prototypowej wersji układu, który następnie może być wyprodukowany jako ASIC. Wadą układów FPGA jest ich wolniejsze działanie, które wynika z ich uniwersalności i regularnej budowy. Umożliwiają natomiast zmianę konfiguracji w trakcie pracy całego systemu co pozwala na elastyczne dostosowanie systemu do wymaganych warunków i zadań.

Największymi producentami układów FPGA są firmy Xilinx i Intel (Altera), które posiadają ponad 90% rynku. Inni producenci to Lattice Semiconductor, Microsemi czy Atmel.

3.1. ARCHITEKTURA UKŁADÓW FPGA

Cechą charakterystyczną architektury układów FPGA jest duża liczba regularnie rozmieszczonych komórek logicznych, zwanych w zależności od producenta CLB (ang. *Configurable Logic Block*) lub LBA (ang. *Logic Array Block*). Budowa komórki logicznej oparta jest na tablicy odwzorowań logicznych LUT (ang. *Look-Up Table*). Tablice LUT można wykorzystać również jako rozproszoną pamięć RAM (ang. *Distributed Random Access Memory*). W komórkach CLB oprócz tablic LUT znajdują się element przełączający, często przerzutnik typu D i linie szybkich przeniesień arytmetycznych (ang. *Carry Chain*) umożliwiające projektowanie szybkich układów arytmetycznych i rejestrów przesuwnych.

Wymiana danych między komórkami logicznymi odbywa się poprzez linie połączeniowe, których w układzie jest kilka rodzajów. Globalne linie zegarowe rozprowadzają sygnały taktujące wewnątrz całego układu FPGA z jak najmniejszym opóźnieniem. Linie długie, nie są dołączone do wszystkich komórek logicznych, są natomiast najszybszymi traktami komunikacyjnymi stosowanymi zamiennie z globalnymi liniami zegarowymi. Linie krótkie rozprowadzają sygnały na mniejsze

odległości i są bardziej elastyczne niż linie długie. Pozostałe linie połączeniowe zapewniają komunikację pomiędzy sąsiadującymi blokami CLB.

Komunikacja układu FPGA z urządzeniami peryferyjnymi odbywa się dzięki blokom wejść/wyjść IOB (ang. *Input Output Block*). Bloki wejść/wyjść pozwalają na zdefiniowanie typu pracy wyprowadzenia układu jako wejściowy, wyjściowy lub jako stan wysokiej impedancji. Możliwa jest również konfiguracja w różnych standardach logicznych, włączając standardy pracy różnicowej. Można także wybrać wydajność prądową wyjścia. Każde wyprowadzenie może być niezależnie dołączane do masy lub do linii zasilającej dzięki wewnętrznym rezystorom pull-down i pull-up. Można teżłączyć przerzutnik podtrzymujący ostatni stan logiczny na wyjściu tzw. pin-keeper.

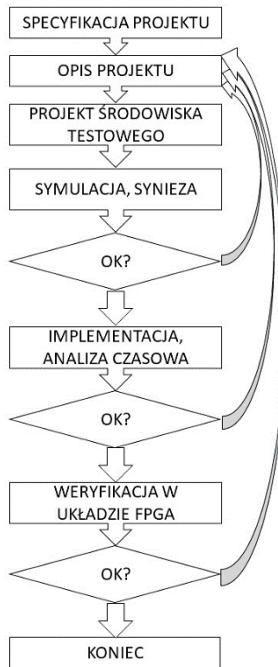
Odpowiednie taktowanie układów FPGA zapewniają generatory wewnętrznych sygnałów zegarowych DCM (ang. *Digital Clock Manager*) lub PLL (ang. *Phase Locked Loop*). Zadaniem generatora jest mnożenie i dzielenie sygnałów zegarowych oraz wytwarzanie sygnałów przesuniętych w fazie.

Nowoczesne układy FPGA są standardowo wyposażone w bloki umożliwiające szybkie wykonywanie najpopularniejszych operacji cyfrowego przetwarzania sygnałów DSP (ang. *Digital Signal Processing*). Podstawą bloku DSP jest układ mnożący. Bloki te znajdują zastosowanie w projektowaniu filtrów cyfrowych czy obliczaniu szybkiej transformaty Fouriera.

Kolejnym blokiem znajdującym się w układach FPGA jest pamięć statyczna SRAM (ang. *Static RAM*). Bloki tej pamięci można łączyć między sobą aby uzyskać wymaganą przez użytkownika pojemność. Mogą być konfigurowane jako pamięci jedno lub dwuportowe, kolejki FIFO (ang. *First In First Out*) lub pamięci ROM.

3.2. PROCES PROJEKTOWANIA UKŁADÓW CYFROWYCH W TECHNOLOGII FPGA

Proces projektowania układów cyfrowych w technologii FPGA rozpoczyna się od określenia specyfikacji projektu. Ten etap obejmuje analizę wymagań projektu. Efektem określenia specyfikacji projektu jest dokument opisującym przyszłą architekturę urządzenia, bloki strukturalne, ich funkcje i interfejsy. Kolejne etapy projektu przedstawiono na rysunku 3.1.



Rysunek 3.1 Proces projektowania układów cyfrowych w technologii FPGA

Opis projektu polega na zdefiniowaniu działania projektu przez jego opis, najczęściej w jednym z języków opisu sprzętu HDL (ang. *Hardware Description Language*). Opis może też być wykonany za pomocą schematu lub listy połączeń.

Projekt środowiska testowego obejmuje pisanie środowisk i modeli testowych. Są one później używane w celu weryfikacji poprawności opisu układu.

Symulacja behawioralna to ważny etap, który sprawdza poprawność opisu projektu porównując jego wyniki z wynikami modelu.

Synteza to konwersja opisu projektu na tak zwaną listę połączeń (ang. *netlist*), która jest odzwierciedleniem połączeń pomiędzy blokami docelowego układu cyfrowego. Syntezę przeprowadza się za pomocą specjalnego oprogramowania. Synteza może ujawnić pewne problemy i potencjalne błędy, których nie można znaleźć za pomocą symulacji, jeśli niektóre elementy projektu nie mogą być odzwierciedlone wewnętrz układowego FPGA.

W fazie implementacji lista sieci generowana przez syntezer jest rzutowana na wewnętrzną strukturę konkretnego układu FPGA. Rozmieszczane są elementy i połączenia między nimi tak, aby zapewnić odpowiednie relacje czasowe, zgodnie z ograniczeniami zdefiniowanymi w projekcie.

Podeczas analizy czasowej sprawdza się, czy zaimplementowany projekt spełnia ograniczenia czasowe (takie jak częstotliwość zegara) określone przez projektanta.

Pomimo wykonania symulacji behawioralnej i analizy czasowej niezbędne jest zweryfikowanie poprawności pracy projektu zaimplementowanego wewnątrz struktury FPGA. Nowoczesne metody projektowe pozwalają na testowanie dowolnego punktu wewnątrz układu FPGA przez umieszczenie dodatkowej logiki komunikującej się z zewnętrznym oprogramowaniem.

3.2.1. POZIOMY ABSTRAKCJI OPISU PROJEKTU

Podczas projektowania projektant może wybrać jeden z kilku dostępnych poziomów abstrakcji.

- Poziom przełączników (ang. *switch level*). Jest to najniższy poziom abstrakcji dostępny dla projektanta. Projekt na tym poziomie opisywany jest za pomocą tranzystorów i połączeń między nimi. Wymaga dużej znajomości szczegółów technicznych docelowej technologii.
- Poziom bramek (ang. *gate level*). Projekt opisywany jest za pomocą bramek i połączeń między nimi. Projektowanie na tym poziomie przypomina tworzenie schematu.
- Poziom przepływu danych (ang. *data flow level*). Na tym poziomie projektant zwraca uwagę na powiązania między poszczególnymi elementami takimi jak rejstry, sumatory itp., oraz na sposób przetwarzania danych w całym układzie.
- Poziom behawioralny (ang. *behavioral level*). Jest najwyższym poziomem abstrakcji. Opis układu nie jest zależny od szczegółów technicznych. Nie opisuje się struktury układu cyfrowego, a jego działanie. Projektant nie musi znać szczegółów konstrukcyjnych układu, wystarczy tylko, że wie w jaki sposób układ ma działać.

3.2.2. JĘZYKI OPISU SPRZĘTU

Najpopularniejsze języki opisu sprzętu to VHDL (ang. *Very High Speed Integrated Circuits Hardware Description Language*), AHDL (ang. *Altera Hardware Description Language*) i Verilog. Języki te różnią się między sobą składnią, poziomami abstrakcji i możliwościami opisu wysokiego poziomu. Największe możliwości w programowaniu wysokopoziomowym posiada język VHDL. Jego wadą jest skomplikowana składnia.

Język AHDL jest językiem posiadającym prostą składnię umożliwiającą szybką realizację układów cyfrowych. Jednak dużą wadą tego języka jest brak możliwości projektowania na wysokim poziomie abstrakcji. Język Verilog posiada składnię zbliżoną do języka C co czyni ten język prostym do opanowania. Dodatkową zaletą języka Verilog jest znacznie mniejszy rygor kontroli typów sygnałów.

3.3. OBSZARY ZASTOSOWANIA

Reprogramowalne układy cyfrowe FPGA stanowią obecnie nie tylko platformę prototypową dla projektantów układów cyfrowych, ale są również coraz częściej wykorzystywane jako docelowe układy w gotowych systemach. Szczególną popularność układy FPGA zyskały w aplikach cyfrowego przetwarzania sygnałów. Zasadniczym powodem tego jest olbrzymia moc przetwarzania jaką daje możliwość równoległej realizacji algorytmów. Nowoczesne układy FPGA wyposaża się często w rdzenie procesorów co znacząco zwiększa ich obszar zastosowania. Umożliwia to tworzenie całych systemów w jednym układzie scalonym SoC (ang. *System on Chip*).

3.4. FPGA A GENERATORY CIĄGÓW LOSOWYCH

Ponieważ systemy kryptograficzne to prawie wyłącznie konstrukcje cyfrowe oczekuje się, że generator TRNG będzie zbudowany wyłącznie z układów cyfrowych. Wiele takich systemów projektuje się wewnątrz struktur FPGA dlatego poszukuje się rozwiązań generatorów ciągów liczb losowych, które można w nich zaimplementować. W przypadku układów FPGA możliwe jest wytworzenie ciągu liczb z generatora TRNG bazując na języku opisu sprzętu i zjawiskach, które mogą wystąpić wewnątrz zaprojektowanego układu logicznego. Przykładami takich zjawisk mogą być szумy fazowe generatorów pierścieniowych lub stany metastabilne.

4. TESTY GENERATORÓW CIĄGÓW RZECZYWIŚCIE LOSOWYCH

Generatory ciągów rzeczywiście losowych są podstawowym elementem wielu systemów kryptograficznych. Takie systemy pełnią bardzo ważną rolę w różnych aplikacjach zapewniając bezpieczną transmisję danych. Niezbędne więc jest aby dostarczane ciągi liczb były odpowiedniej jakości - były rzeczywiście losowe.

Ciągi losowe sprawdza się za pomocą testów statystycznych. Jednakże testy statystyczne nie rozróżniają ciągów powstałych w sposób deterministyczny od ciągów rzeczywiście losowych. Testowanie należy zacząć już od samego źródła entropii, z którego pochodzi rzeczywista losowość ciągu.

4.1. TESTOWANIE ŹRÓDŁA ENTROPII

Entropia jest miarą nieuporządkowania układu [113]. W teorii informacji entropia jest miarą średniej ilości informacji niesionej przez pojedynczą wiadomość, opisana jest równaniem [114]:

$$H(X) = \sum_{i=1}^n p(x_i) \log_2 \frac{1}{p(x_i)}, \quad (4.1)$$

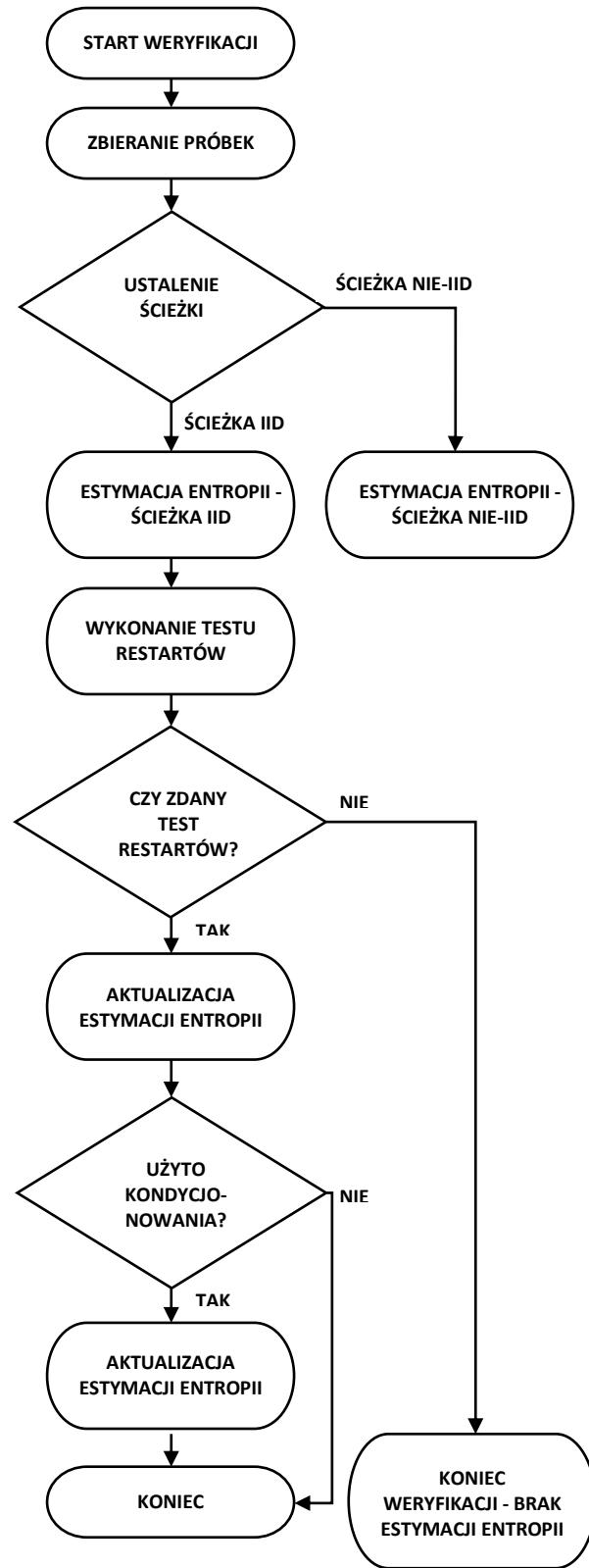
gdzie $p(x_i)$ to prawdopodobieństwo zajścia zdarzenia x_i . W przypadku ciągów binarnych prawdopodobieństwo $p(x_i)$ to prawdopodobieństwo wystąpienia symbolu „0” lub „1”. Entropia może być interpretowana jako miara losowości ciągu. Dla ciągu doskonale losowego wartość entropii wynosi 1 bit informacji na 1 bit wytwarzany.

Testowanie źródła entropii odbywa się według schematów zaproponowanych przez organizacje standaryzujące. Jedną z takich organizacji jest północnoamerykański NIST (ang. *National Institute of Standards and Technology*). Organizacja ta wydała zalecenie NIST SP 800-90B [108], które przedstawia wymagania i sposoby testowania źródła entropii. Schemat testowania źródła entropii przedstawiono na rysunku 4.1. W pierwszej kolejności należy zebrać próbki losowych ciągów wytwarzanych przez źródło entropii. Zalecana minimalna długość ciągu to 1000000 próbek. Kolejnym etapem jest wykonanie testów, które wyznaczają czy wartości próbek są niezależne i o identycznym rozkładzie prawdopodobieństwa IID (ang. *Independent and Identically Distributed*). Następnie należy oszacować entropię źródła szumu. Rekomendacja NIST zaleca wykorzystywanie w obliczeniach entropii minimalnej opisanej wzorem [108]:

$$H(X) = -\log_2(\max_{1 \leq i \leq k}(p(x_i))). \quad (4.2)$$

Entropia minimalna jest bardziej restrykcyjna od entropii Shannona (4.1). Jest często używana jako najgorszy przypadek nieprzewidywalności zmiennej losowej. Potem wykonuje się test restartów. Weryfikuje on czy źródło szumu wytwarza nieskorelowane ciągi liczbowe. Sprawdza zależności między próbками źródła szumu wygenerowanymi po ponownym uruchomieniu źródła entropii i porównuje wyniki z początkową szacunkową entropią. Należy wykonać 1000 restartów po 1000 próbek każdy. Jeżeli test nie zostanie spełniony to takie źródło szumu nie może być użyte jako źródło entropii w generatorze liczb rzeczywiście losowych.

W zbadaniu czy źródło szumu wytwarza próbki niezależne i o identycznym rozkładzie pomaga zestaw testów zaproponowanych przez NIST [108]. Niektóre testy (np. test kompresji) są skuteczne w wykrywaniu powtarzających się wzorców określonych wartości (np. ciągi wartości, które występują częściej niż można by oczekiwać od ciągu wartości IID). Inne testy (np. test liczby ciągów kierunkowych i test liczby ciągów na podstawie mediany) koncentrują się na powiązaniu wartości liczbowych następujących po sobie próbek, w celu znalezienia trendu lub innej zależności, takiej jak wysokie wartości



Rysunek 4.1 Algorytm testowania źródła entropii

próbek następujące po niskich wartościach. Poniżej przedstawiono testy, które są częścią zalecenia NIST 800-90B:

- Test błędzenia (ang. *Excursion Test Statistic*). Określa w jakim stopniu suma wyników próbek odbiega od wartości średniej w każdym punkcie zbioru danych. Statystyka testu jest największym odchyleniem od średniej.
- Test liczby ciągów kierunkowych (ang. *Number of Directional Runs*). Określa liczbę ciągów zbudowanych przy użyciu relacji między kolejnymi próbками.
- Test długości ciągów kierunkowych (ang. *Length of Directional Runs*). Wyznacza długość najdłuższego przebiegu zbudowanego przy użyciu relacji między kolejnymi próbками.
- Test liczby wzrostów i spadków (ang. *Number of Increases and Decreases*). Wyznacza maksymalną liczbę wzrostów lub spadków między kolejnymi wartościami próbek.
- Test liczby ciągów na podstawie mediany (ang *Number of Runs Based on the Median*). Określa liczbę ciągów, które są konstruowane w odniesieniu do mediany danych wejściowych.
- Test długości ciągu na podstawie mediany (ang. *Length of Runs Based on the Median*). Określa długość najdłuższego przebiegu, który jest konstruowany w odniesieniu do mediany danych wejściowych.
- Test średniej kolizji (ang. *Average Collision Test Statistic*). Wyznacza średnią liczbę kolejnych wartości próbek do momentu znalezienia duplikatu.
- Test maksymalnej wartości kolizji (ang. *Maximum Collision Test Statistic*). Wyznacza maksymalną liczbę kolejnych wartości próbek do momentu znalezienia duplikatu.
- Test okresowości (ang. *Periodicity Test Statistic*). Ma na celu określenie liczby okresowych struktur w testowanym ciągu.
- Test kowariancji (ang. *Covariance Test Statistic*). Mierzy wartości kowariancji.
- Test kompresji (ang. *Compression Test Statistic*). Wyznacza długość ciągu danych po kompresji.
- Test dobroci dopasowania (ang. *Goodness-of-fit*) Test wykorzystuje wartość statystyki χ^2 do sprawdzenia czy rozkład próbek jest identyczny dla różnych podciągów z ciągu wejściowego.

- Test niezależności (ang. *Independence Test*). Sprawdza niezależność między sąsiednimi próbками.
- Test najdłuższego powtórzonego podciagu (ang. *Length of the Longest Repeated Substring Test*). Wyznacza długość najdłuższego powtórzonego podciagu w danych wejściowych.

Wymienione testy wykonywane są wielokrotnie dla różnych permutacji danych wejściowych. Jeżeli wszystkie testy zostaną spełnione to nie ma podstaw do odrzucenia hipotezy, że źródło entropii wytwarza ciągi niezależne o identycznym rozkładzie.

Źródło entropii powinno zawierać kilka elementów: źródło szumu analogowego, ekstraktor losowości, przetwarzanie analogowo-cyfrowe, opcjonalny układ przetwarzania i wbudowany zestaw testów. Na zestaw testów składają się: test rozruchowy (ang. *start-up*), testy działające w tle i testy na żądanie. Testy te powinny alarmować kiedy źródło entropii nie działa według założeń i żadne losowe bity nie powinny się wtedy pojawiać na wyjściu generatora.

Test rozruchowy powinien być zaprojektowany w taki sposób, aby został przeprowadzony po każdorazowym włączeniu generatora przed rozpoczęciem wysyłania losowych bitów na wyjście generatora. Ma on na celu sprawdzenie czy generator pracuje prawidłowo, i że nic się z nim nie stało po poprzednim wykorzystaniu. Próbki sygnału ze źródła entropii w trakcie testu rozruchowego nie powinny być wykorzystane jako sygnał wyjściowy generatora. Jako test rozruchowy może być wykorzystany test restartów.

Testy działające w tle są wykonywane podczas normalnej pracy źródła entropii. Są zaprojektowane w taki sposób aby śledzić zmiany sygnału szumu. Testy te wykonuje się w sposób ciągły na każdej cyfrowej próbce otrzymanej ze źródła szumu. Testy powinny zapewnić bardzo małe prawdopodobieństwo wzbudzenia fałszywego alarmu. Rekomendacja przedstawia dwa tego rodzaju testy, które mogą być implementowane w strukturach generatora. Jeżeli oba testy są zaimplementowane i równocześnie wykonywane, to żadne inne testy nie są wymagane. Oba testy zostały zaprojektowane w taki sposób, aby sprostały obliczeniom „w locie”, w trakcie pobierania próbek szumu. Testy wykonane na określonej liczbie danych mogą być wykorzystane również jako testy rozruchowe.

Jednym z rekomendowanych testów jest test liczby powtórzeń (ang. *repetition count test*). Celem testu jest wykrycie poważnych awarii, które skutkują tym, że przez dłuższy

czas źródło szumu wytwarza próbki o tej samej wartości. Biorąc pod uwagę wyznaczoną entropię minimalną H źródła szumu, prawdopodobieństwo tego, że źródło generuje kolejno identyczne próbki wynosi co najwyżej $2^{-H(n-1)}$. Test sygnalizuje błąd jeżeli próbki powtarzają się C lub większą liczbę razy. Wartość C wyznacza się dla danego poziomu istotności α i entropii H wykorzystując wzór [108]:

$$C = 1 + \left\lceil \frac{-\log_2 \alpha}{H} \right\rceil. \quad (4.3)$$

Wartość C jest najmniejszą całkowitą wartością spełniającą nierówność $\alpha \geq 2^{-H(C-1)}$, co zapewnia, że prawdopodobieństwo wytworzenia ciągu z co najwyżej C próbek tej samej wartości wynosi α . Pseudokod testu liczby powtórzeń przedstawiono na rysunku 4.2.

```

1. A = następna próbka
2. B = 1
3. X = następna próbka
4. if(X = A),
       B = B + 1
       if(B ≥ C), BŁĄD!
    else:
        A = X
        B = 1
5. powtóż krok 3.

```

Rysunek 4.2 Algorytm testu liczby powtórzeń

Drugi rekomendowany test to adaptacyjny test proporcji (ang. *adaptive proportion test*). Ma na celu wykrycie dużych zmian entropii, które mogą być wynikiem awarii lub zmian środowiskowych wpływających na źródło szumu. Test w sposób ciągły mierzy lokalną częstotliwość występowania wartości próbki sygnału w ciągu próbek, aby określić czy dana próbka nie występuje zbyt często. Test pobiera próbkę ze źródła szumu a następnie zlicza ile razy ta sama próbka pojawiła się w przedziale $W - 1$ próbek. Jeśli licznik osiągnie wartość graniczną C , wtedy zgłoszany jest błąd. Szerokość okna W dla ciągów binarnych powinna być ustawiona na 1024, w innym wypadku na 512. Wartość graniczna C spełnia nierówność $W - B \geq C$. Pseudokod testu przedstawiono na rysunku 4.3.

```

1. A = następna próbka
2. B = 1
3. for i = 1 to W - 1
       if(A = następna próbka) B = B + 1
       if(B ≥ C), BŁĄD!
4. powtóż krok 1.

```

Rysunek 4.3 Algorytm adaptacyjnego testu proporcji

Testy na żądanie mogą być wykonane w każdej chwili. Nie muszą być wykonywane podczas normalnej pracy źródła entropii. Próbki pobrane ze źródła szumu podczas testu nie powinny być używane do wytwarzania losowych bitów do czasu zakończenia testów. Mogą natomiast zostać odrzucone lub wykorzystane po zakończeniu testów, pod warunkiem, że nie występują żadne błędy. Rekomendacja nie preczyje jakie testy mają być wykonywane.

Inną organizacją standaryzacyjną jest niemieckie Federalne Biuro Bezpieczeństwa Informacyjnego BSI (niem. *Bundesamt für Sicherheit in der Informationstechnik*), które wydało zalecenie AIS-31 (niem. *Anwendungshinweise und Interpretationen*) [115]. Definiuje ono trzy klasy fizycznych generatorów ciągów losowych:

- PTG.1 Jest najniższym poziomem bezpieczeństwa, który może być użyty do wytwarzania losowych liczb dla systemów kryptograficznych, które nie wymagają nieprzewidywalności kolejnych wartości. Kolejne generowane liczby mogą być odgadnięte.
- PTG.2 Generatorы tej klasy wytwarzają ciągi losowe o wysokiej entropii i rozkładzie równomiernym. Kolejne elementy ciągu powinny być nieprzewidywalne i niezależne.
- PTG.3 Jest najwyższą klasą bezpieczeństwa definiującą zarówno generator liczb rzeczywiście losowych jak i kryptograficzne algorytmy post-processingu.

W rekomendacji AIS-31 do oceny parametrów źródła losowości wykorzystuje się entropię Shannona (4.1). Do testowania przewidziano testy rozruchowe i testy działające w tle, które powinny wykryć nieprawidłowości w funkcjonowaniu źródła entropii.

Procedura testowania rozruchowego jest następująca. Należy wygenerować 512 bitów i podzielić je na cztero-bitowe wiadomości. Następnie wykonuje się test zgodności χ^2 opisany wzorem:

$$\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}, \quad (4.4)$$

gdzie O_i to wartość mierzona zdarzenia i , E_i wartość oczekiwana zdarzenia i . Test z piętnastoma stopniami swobody jest spełniony na poziomie istotności równym $3,8 \times 10^{-7}$, jeżeli wartość statystyki χ^2 jest mniejsza od 65. W przeciwnym wypadku źródło entropii działa nieprawidłowo co sygnalizowane jest przez alarm.

Do testów działających w tle można zaliczyć również przeprowadzany na bieżąco test zgodności χ^2 . Różnica polega na tym, że wykonuje się 512 testów zgodności na sześciobitowych słowach. Wstępny alarm jest sygnalizowany jeżeli trzy kolejne wartości statystyki są większe niż 26,75, pojedyncza wartość statystyki jest większa niż 269,5 lub wartość statystyki leży poza przedziałem (13,0: 17,0). Jeżeli w trzech kolejnych zestawach testów zostanie zasygnalizowany alarm wstępny to oznacza, że źródło entropii nie pracuje poprawnie. Wtedy sygnalizowany jest alarm główny a generator jest wyłączany.

4.2. TESTY STATYSTYCZNE CIĄGÓW

Testowanie źródła entropii ma na celu wyeliminowanie błędów w jego działaniu. Nie zapewnia to jednak tego, że generowane ciągi będą miały dobre właściwości statystyczne. Wytwarzane przez generator ciągi liczb rzeczywiście losowych muszą posiadać szereg określonych cech. Przede wszystkim nie może być możliwe odgadnięcie kolejnego elementu ciągu. Następujące po sobie elementy ciągu muszą być nieprzewidywalne. Poszczególne elementy ciągu muszą być niezależne i posiadać rozkład normalny. Między innymi te parametry są sprawdzane za pomocą testów statystycznych. Testy mają za zadanie wykrycie nielosowych właściwości wyjściowych ciągów, a w konsekwencji odrzucenie generatora o nieakceptowalnych parametrach. Nie potrafią natomiast odpowiedzieć na pytanie czy ciągi zostały wytworzone przez źródło niedeterministyczne. Sprawdzają tylko hipotezę czy dany ciąg wygląda na losowy (posiada zdefiniowany zestaw cech weryfikowany testami statystycznymi). Według publikacji NIST SP 800-22 [116] zakłada się, że ciągi losowe charakteryzuje:

- Jednorodność: w dowolnej chwili generowania ciągu losowego prawdopodobieństwo wystąpienia jedynki lub zera jest jednakowe i wynosi $\frac{1}{2}$. Oczekiwana liczba jedynek lub zer w ciągu o długości n wynosi $\frac{n}{2}$.
- Skalowalność: każdy podciąg wyodrębniony z ciągu liczb losowych powinien przejść dowolny test losowości z pozytywnym wynikiem.
- Spójność: wytwarzane przez generator ciągi powinny mieć takie same właściwości w każdej chwili jego pracy i za każdym jego uruchomieniem.

Istnieje wiele różnych testów statystycznych oceniających generowane ciągi. W publikacji [117] znajduje się obszerny opis wielu testów statystycznych. Do testowania generatorów najczęściej stosowane są zestawy kilkunastu testów. Zbiory testów są dobrane w taki sposób, aby sprawdzaniu podlegały parametry charakterystyczne dla ciągu

losowego. Najbardziej znane zestawy testów to: NIST SP 800-22 [116], AIS-31 [118], FIPS 140-2 (ang. *Federal Information Processing Standards*) [119], DIEHARD [120], Dieharder [121] i TestU01 [122]. Pakiety te są przeznaczone do zastosowań kryptograficznych jednak tylko dwa pierwsze z nich są opracowane przez rządowe organizacje zajmujące się bezpieczeństwem informatycznym i standaryzacją. Poniżej przedstawiono najważniejsze testy wraz z ich opisem, które są częścią wyżej wymienionych zestawów:

- Test częstości (ang. *Frequency Monobit Test*). Celem testu jest ustalenie czy liczba zer i jedynek w ciągu, jest w przybliżeniu taka sama jak wartość oczekiwana dla ciągu rzeczywiście losowego. Liczba zer i jedynek w ciągu powinna być mniej więcej taka sama, tj. w przybliżeniu równa połowie długości ciągu.
- Blokowy test częstości (ang. *Frequency Test within a Block*). Test wyznacza proporcję liczby jedynek do liczby wszystkich bitów w M -bitowych blokach. Celem testu jest sprawdzenie czy liczba jedynek w poszczególnych blokach wynosi w przybliżeniu $\frac{M}{2}$.
- Test ciągów (ang. *Runs Test*). Test sprawdza liczbę ciągów o długości k w ciągu testowanym. Ciągiem o długości k jest nieprzerwany ciąg tych samych bitów. Celem testu jest ustalenie, czy liczba ciągów zer i jedynek o różnych długościach jest zgodna z oczekiwaniami dla ciągu losowego.
- Test na najdłuższy ciąg jedynek w bloku (ang. *Test for the Longest Run of Ones in a Block*). Celem testu jest znalezienie najdłuższego ciągu jedynek w M -bitowych blokach. Otrzymany wynik jest porównywany z wynikiem oczekiwany dla ciągu rzeczywiście losowego.
- Test stopnia macierzy binarnej (ang. *Binary Matrix Rank Test*) W teście sprawdzane są stopnie rozłącznych podmacierzy utworzonych z testowanego ciągu. Celem tego testu jest sprawdzenie liniowej zależności między podciągami o stałej długości w testowanym ciągu.
- Test widmowy DFT (ang. *Discrete Fourier Transform (Spectral) Test*). Test sprawdza wartości szczytowe prążków dyskretnej transformaty Fouriera ciągu wejściowego. Celem testu jest wykrycie okresowych cech ciągu, tj. powtarzających się krótko po sobie pewnych wzorców, które wskazywałyby na odchylenia ciągu testowanego od ciągu prawdziwie losowego.

- Test dopasowania nienakładających się wzorców (ang. *Non-overlapping Template Matching Test*). Sprawdza się liczbę wystąpień zdefiniowanych wcześniej nieokresowych ciągów wzorcowych w ciągu testowanym. Celem testu jest wykrycie generatorów dostarczających ciągi, w których zdefiniowany wzorzec pojawia się zbyt często. Do detekcji wzorca o długości m służy okno o tej samej szerokości. Jeżeli wzorzec nie zostanie znaleziony w oknie to okno przesuwa się o jeden bit. Jeżeli wzorzec zostanie znaleziony w oknie to okno przesuwa się o m bitów.
- Test dopasowania nakładających się wzorców (ang. *Overlapping Template Matching Test*). Cel tego testu jest taki sam jak poprzedniego testu. Różnica polega na tym, że okno zawsze jest przesuwane o jeden bit niezależnie czy wzorzec został znaleziony czy nie.
- Test uniwersalny Maurera (ang. *Maurer's "Universal Statistical" Test*). Test sprawdza czy testowany ciąg podlega kompresji bezstratnej. Ciągi dające się mocno skompresować bez utraty informacji nie są ciągami losowymi.
- Test złożoności liniowej (ang. *Linear Complexity Test*). Celem testu jest znalezienie długości liniowego rejestru przesuwnego ze sprzężeniem zwrotnym LFSR. Ustala się, czy ciąg jest wystarczająco złożony, aby mógł być uważana za losowy. Ciągi uważane za losowe ciągi wytwarza się wykorzystując długie LFSR. Ciąg wytworzony przez krótki LFSR, nie jest uważany za losowy.
- Test serii (ang. *Serial Test*). Sprawdza częstotliwość występowania nakładających się m -bitowych wzorców w testowanym ciągu. Bada się, czy liczba wystąpień 2^m różnych wzorców jest zbliżona do wartości oczekiwanej dla ciągu losowego. W ciągu rzeczywiście losowym prawdopodobieństwo wystąpienia każdego z 2^m wzorca jest takie samo. Dlatego testowany ciąg powinien zawierać w przybliżeniu taką samą liczbę różnych wzorców o długości m .
- Test przybliżonej entropii (ang. *Approximate Entropy Test*). Podobnie jak test serii, sprawdza on częstotliwość występowania nakładających się m -bitowych wzorców w testowanym ciągu. Celem testu jest porównanie częstotliwości nakładających się m -bitowych bloków dla dwóch kolejnych długości bloków

tj. dla m i $m + 1$, z częstotliwością oczekiwana dla ciągu rzeczywiście losowego.

- Test skumulowanych sum (ang. *Cumulative Sums (Cusum) Test*). Test wyznacza skumulowane sumy dla ciągu wejściowego $(0, 1)$, który zamieniono na ciąg $(-1, +1)$. Następnie sprawdza się, czy lokalne sumy, będące najdłuższymi błędzeniami przypadkowego od punktu zerowego, odpowiadają lokalnym sumom dla błędnienia rzeczywiście losowego. Dla ciągów rzeczywiście losowych skumulowane sumy są niewielkie, błądzenie losowe często przecina punkt zerowy.
- Test błędnienia losowego (ang. *Random Excursions Test*). Błądzenie losowe zdefiniowane jest w postaci skumulowanych sum częściowych ciągu $(0, 1)$ zamienionego na ciąg $(-1, +1)$. Cykl w błędzeniu losowym zaczyna się w chwili opuszczenia stanu początkowego i kończy się w chwili powrotu do tego stanu. Sprawdza się czy liczba przejść przez dany stan w każdym cyklu jest zbliżona do liczby przejść dla ciągu losowego. Test składa się z ośmiu podtestów przeprowadzonych dla ośmiu stanów: $-4, -3, -2, -1, +1, +2, +3, +4$.
- Test wariancji błędnienia losowego (ang. *Random Excursions Variant Test*). Błądzenie losowe zdefiniowane jest w postaci skumulowanych sum częściowych ciągu $(0, 1)$ zamienionego na ciąg $(-1, +1)$. Cykl w błędzeniu losowym zaczyna się w chwili opuszczenia stanu początkowego i kończy się w chwili powrotu do tego stanu. Celem tego testu jest wykrycie odchyleń od spodziewanej liczby przejść przez różne stany podczas błędnienia losowego. Test składa się z osiemnastu podtestów przeprowadzanych dla osiemnastu stanów: $-9, -8, \dots, -1$ i $+1, +2, \dots, +9$.
- Test pokerowy (ang. *Poker Test*). Sprawdza częstotliwość występowania odpowiadających 4-bitowych wzorców w testowanym ciągu. Następnie sprawdza się czy prawdopodobieństwo wystąpienia określonego wzorca jest zbliżone do prawdopodobieństwa w ciągu rzeczywiście losowym.
- Test autokorelacji (ang. *Autocorrelation Test*). Test sprawdza czy istnieją zależności w testowanym ciągu. Wyznacza się autokorelację podciągów o różnej długości. W ciągu rzeczywiście losowym wszystkie jego elementy są niezależne.

- Test rozkładu normalnego (ang. *Uniform Distribution Test*). Sprawdza się, czy testowany ciąg posiada rozkład równomierny. Prawdopodobieństwo wystąpienia każdego podciągu o określonej długości, utworzonego z testowanego ciągu powinno być takie samo.
- Test porównawczy dla rozkładów wielomianowych (ang. *Comparative Test for Multinomial Distributions*). Test wyznacza rozkłady wielomianowe podciągów testowanego ciągu. Następnie suma otrzymanych rozkładów wielomianowych porównywana jest z wartością statystyki χ^2 tego ciągu. Dla ciągu rzeczywiście losowego wartości te powinny być mniej więcej takie same.
- Test entropii (ang. *Entropy Test*). Wyznaczana jest entropia ciągu. Dla ciągu doskonale losowego wartość entropii wynosi 1.

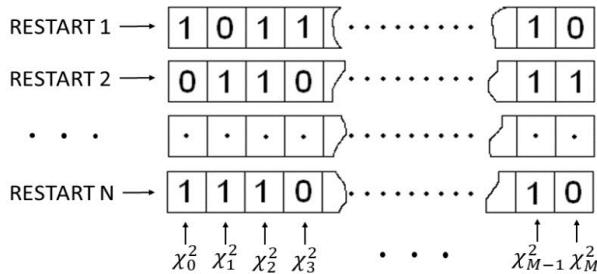
Pierwsze piętnaście z wyżej wymienionych testów składa się na pakiet testów statystycznych NIST SP 800-22 [116]. Pakiet ten jest przeznaczony do testowania ciągów o dużych wymaganiach jakościowych na potrzeby systemów kryptograficznych.

4.3. METODA RESTARTÓW

Generatory ciągów liczb rzeczywiście losowych wytwarzają ciągi, które powstają z wykorzystaniem nieidealnego źródła entropii. Otrzymane ciągi oprócz składników, które powstały w rzeczywiście niedeterministyczny sposób posiadają składniki powstałe w sposób deterministyczny. W szczególności problem ten dotyczy generatorów opartych o zjawisko szybkozmiennych fluktuacji fazy [123], [124], [125]. Zarówno składnik deterministyczny jak i niedeterministyczny wpływa na właściwości statystyczne wytwarzanego ciągu, ale tylko składniki szumowe są źródłem rzeczywistej losowości w ciągu wyjściowym. Składniki deterministyczne są źródłem pseudolosowości. Dlatego, aby zdecydować, czy generator wytwarza ciągi nadające się do zastosowań kryptograficznych należy ocenić ilość rzeczywistej losowości i pseudolosowości w wytwarzanym ciągu.

Metoda restartów polega na wielokrotnym włączaniu generatora z tymi samymi warunkami początkowymi. Jeżeli otrzymany ciąg zawdzięczamy pseudolosowości (determinizmowi), to przy każdym powtórzeniu eksperymentu otrzymamy taki sam ciąg. W przypadku, kiedy ciąg powstał w rzeczywiście losowy sposób (niedeterministyczny), ciągi będą się różnić. Metodę restartów zastosowano w pracy [57] do oceny losowości oscylatora pierścieniowego, oscylatora pierścieniowego typu Galois i typu Fibonacci.

Zarejestrowano 1000 ciągów za pomocą oscyloskopu, dla każdego przypadku. Ilość rzeczywistej losowości w otrzymanych przebiegach zmierzono przez obliczenie odchylenia standardowego napięcia wyjściowego w funkcji czasu. Jeżeli odchylenie standardowe było odpowiednio duże dla danego bitu, wówczas nie ma podstaw do odrzucenia hipotezy, że ten bit powstał w sposób losowy. Taką samą metodę zastosowano w pracy [61]. Inne rozwiązanie zaproponowano w pracy [126] i rozwinięto w publikacji [69]. Podstawowa różnica polega na ocenie losowości na wyjściu całego generatora liczb rzeczywiście losowych a nie z jego pojedynczego źródła losowości. Dlatego ta metoda jest lepsza do oceny właściwości losowych ciągów wytwarzanych przez generatory złożone z wielu niezależnych źródeł entropii. Podczas jednego restartu wytwarza się ciąg M bitów i N razy ponawia się uruchamianie generatora. Pod koniec eksperymentu uzyskuje się N , M -bitowych ciągów. Następnie wyznacza się wartość m_{min} , która jest numerem bitu, który jako pierwszy w M -bitowym ciągu spełnia test χ^2 (4.4) dla N -bitowego ciągu. Jeżeli wartość tej statystyki jest mniejsza od wartości granicznej to z pewnym poziomem istotności możemy stwierdzić, że dany bit powstał w wyniku oddziaływania czynników niedeterministycznych. Schemat ilustrujący sposób wyznaczania M wartości χ^2 przedstawia rysunek 4.4. Idealne źródło losowe wytwarza niezależne bity z takim



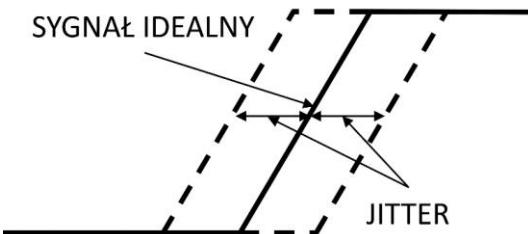
Rysunek 4.4 Sposób wyznaczania statystyki χ^2

samym prawdopodobieństwem od pierwszego bitu. W konsekwencji wszystkie ciągi są równoprawdopodobne. Pojedyncze wystąpienie sekwencji, która nie spełnia testu chi-kwadrat nie może jednak zdyskwalifikować danego bitu jako losowego. Możemy odrzucić hipotezę o losowości, jeśli test χ^2 nie jest spełniony dla odpowiednio dużej liczby bitów. Dla $M = 20000$ i $N = 2048$ testu może nie spełnić co najwyżej 195 pojedynczych bitów, przy czym niespełnienie testu dla dwóch kolejnych bitów jest możliwe tylko 2 razy w całym ciągu [69]. Podobną metodę zastosowano w pracy [127]. Różnica polega na wyznaczeniu odchylenia standardowego dla każdego bitu z serii restartów. Jeżeli powstanie danego bitu zawdzięczamy rzeczywistej losowości to odchylenie standardowe wynosi około 0,5. Jeżeli bit powstanie w sposób deterministyczny to odchylenie standardowe wynosi 0 [127].

5. ŁĄCZONY GENERATOR TRNG Z DETEKTOREM FAZY

5.1. CHARAKTERYSTYKA JITTERA W OSCYLATORACH PIERŚCIENIOWYCH

Jitterem nazywamy szybkozmienne fluktuacje fazy sygnału cyfrowego. W czasowym ujęciu stanowi on różnicę między rzeczywistą chwilą nadania narastającego lub opadającego zbocza sygnału a chwilą idealną, kiedy zbocze to powinno się pojawić (rys. 5.1). Jitter można podzielić na dwa podstawowe rodzaje: deterministyczny



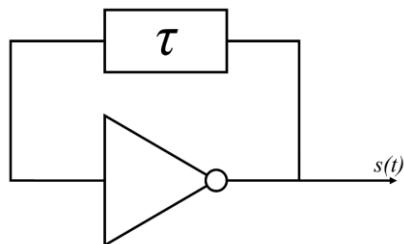
Rysunek 5.1 Definicja jittera

i niedeterministyczny. Jitter deterministyczny wywołują takie zjawiska jak przeniki między sygnałowe, niedopasowanie impedancji, interferencja międzysymbolowa czy też zmiana współczynnika wypełnienia impulsu. Jitter niedeterministyczny (losowy) powstaje wskutek oddziaływań szumów termicznego i śrutowego. Jitter losowy posiada charakterystykę gaussowską [62], [123]. Sygnał okresowy obarczony szybkozmiennymi fluktuacjami fazy możemy zapisać jako:

$$s(t) = P[2\pi ft + \Delta\varphi(t)], \quad (5.1)$$

gdzie P oznacza ciąg okresowych impulsów, f częstotliwość sygnału okresowego, t chwile obserwacji, a $\Delta\varphi$ dewiację fazy.

Oscylator pierścieniowy przedstawiono na rysunku 5.2. Składa się z inwertera i elementu opóźniającego τ . Jako element opóźniający może być wykorzystana parzysta



Rysunek 5.2 Oscylator pierścieniowy

liczba inwerterów, ciąg lub pojedynczy bufor, otwarty zatrzask (ang. *latch*), odpowiednio dłuża ścieżka połączeń lub linia opóźniająca. Oscylator pierścieniowy wytwarza okresowy sygnał o okresie opisanym równaniem:

$$T = 2 \sum_{i=1}^k \tau_i, \quad (5.2)$$

gdzie k jest liczbą elementów opóźniających, τ_i jest opóźnieniem i -tego elementu opóźniającego. Zakłada się, że opóźnienie jest stałe w czasie i pomija się opóźnienie połączeń między elementami opóźniającymi. W rzeczywistym układzie ten czynnik jest bardzo ważny i nie można go ignorować [128].

W przypadku oscylatorów pierścieniowych, zmiana czasu propagacji sygnału przez elementy opóźniające objawia się zmianą okresu sygnału. Zmiany te są spowodowane występowaniem szumów śrutowych i termicznych, a także przeników z innych fragmentów układu elektronicznego. Opóźnienie i -tego elementu opóźniającego można zapisać jako [123], [129]:

$$\tau_i = \Theta_i + dn_i + \Delta dt_i + \Delta dv_i + \Delta dr_i, \quad (5.3)$$

gdzie: Θ_i jest statycznym opóźnieniem i -tego elementu opóźniającego, dn_i oznacza opóźnienie i -tego połączenia w oscylatorze pierścieniowym, Δdt_i jest zmianą opóźnienia w i -tym elemencie i połączeniu spowodowaną przez zmiany temperatury, Δdv_i reprezentuje zmiany opóźnienia w i -tym elemencie i połączeniu spowodowane niestabilnością napięcia zasilania a Δdr_i definiuje inne losowe opóźnienia w i -tym elemencie i ścieżce w oscylatorze pierścieniowym, np. przeniki. Korzystając ze wzoru (5.3) okres oscylatora pierścieniowego można zapisać w postaci:

$$T = 2 \sum_{i=1}^k (\Theta_i + dn_i + \Delta dt_i + \Delta dv_i + \Delta dr_i). \quad (5.4)$$

Okres sygnału oscylatora pierścieniowego może zmieniać się w czasie dzięki zmianom Δdt , Δdv i Δdr . Zmiany te odpowiadają za szybkozmiennne fluktuacje fazy - jitter. Ponieważ centralne twierdzenie graniczne stwierdza, że suma wielu niezależnych zmiennych losowych ma rozkład normalny, jitter w oscylatorze pierścieniowym posiada rozkład Gaussa. Komponenty zmieniające się w czasie można zapisać w postaci [123]:

$$\begin{aligned} \Delta dt &= \Delta dt_n + p \cdot \Delta dt_d, \\ \Delta dv &= \Delta dv_n + p \cdot \Delta dv_d, \\ \Delta dr &= \Delta dr_n + p \cdot \Delta dr_d, \end{aligned} \quad (5.5)$$

gdzie Δdt_n , Δdv_n , Δdr_n to niedeterministyczny składnik a Δdt_d , Δdv_d , Δdr_d to składnik deterministyczny, wartość p odpowiada za proporcję deterministycznego składnika.

Niedeterministyczne składniki odpowiadają za niedeterministyczne fluktuacje, które są idealnym źródłem entropii.

Jednym z ważnych zjawisk w oscylatorach pierścieniowych jest akumulacja szybkozmiennych fluktuacji fazy w czasie. Zjawisko to można zaobserwować wykorzystując np. metodę restartów [57], [69]. Zmiana okresu sygnału oscylatora pierścieniowego z zakumulowanym jitterem w l okresach przedstawia się jako [123]:

$$\Delta T = \sum_{j=1}^{2l} \sum_{i=1}^k (\Theta_i + dn_i + \Delta J_{i,j}), \quad (5.6)$$

gdzie

$$\Delta J = \Delta dt + \Delta dv + \Delta dr, \quad (5.7)$$

to zakumulowany jitter w półokresie sygnału oscylatora pierścieniowego złożonego z k elementów opóźniających [123]. Może być podzielony na czynnik deterministyczny i niedeterministyczny:

$$\Delta J = \Delta J_n + \Delta J_d, \quad (5.8)$$

gdzie

$$\Delta J_n = \Delta dt_n + \Delta dv_n + \Delta dr_n, \quad (5.9)$$

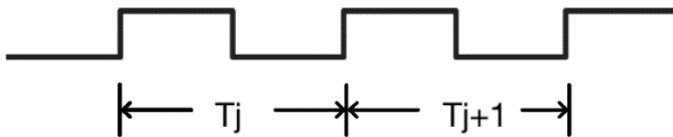
oznacza jitter niedeterministyczny a

$$\Delta J_d = p \cdot (\Delta dt_d + \Delta dv_d + \Delta dr_d), \quad (5.10)$$

oznacza jitter deterministyczny ze współczynnikiem proporcji równym p . Jak wspomniano w [130], odchylenie standardowe niedeterministycznego jittera nagromadzonego w oscylatorze pierścieniowym jest proporcjonalne do pierwiastka kwadratowego czasu. Deterministyczny jitter ma odchylenie standardowe proporcjonalne do czasu. Wynika z tego, że jitter deterministyczny kumuluje się szybciej niż niedeterministyczny.

Istnieje kilka podstawowych metod pomiaru jittera. Najbardziej rozpowszechniona metoda to pomiar błędu czasu TE (ang. *Time Error*). Ten rodzaj pomiaru wymaga idealnego sygnału referencyjnego do porównania. W wyniku pomiarów otrzymuje się minimalną, typową i maksymalną różnicę między idealnym sygnałem zegarowym a sygnałem mierzącym. Inny rodzaj pomiaru to pomiar jittera okresu (ang. *period jitter*). Mierzy się odchylenie czasu okresu w dużej liczbie własnych okresów (zwykle 10000 cykli) zegara. Wyznacza minimalne, średnie i maksymalne odchylenie w odniesieniu do średniego okresu z tych pomiarów. Pomiarem, który najbardziej uwydatnia niedeterministyczny jitter jest pomiar jittera cykl do cyklu (ang. *cycle to cycle*) [131]. Mierzy się jeden okres zegara i porównuje z sąsiednim okresem tego sygnału. Typowo

wykonuje się między 1000 a 10000 pomiarów okresu. Sposób pomiaru przedstawia rysunek 5.3 i wzór (5.11).



Rysunek 5.3 Pomiar jittera cykl do cyklu

$$J_{cc} = \max_j |T_j - T_{j+1}|. \quad (5.11)$$

W celu dokonania pomiaru jittera w oscylatorach pierścieniowych zaimplementowano w układzie FPGA Virtex 5 firmy Xilinx oscylatory pierścieniowe z różnymi elementami opóźniającymi o różnej długości. Wyniki pomiarów przedstawiono w tabeli 5.1. Dla każdego przypadku zebrano 5000 pomiarów okresu.

Tabela 5.1 Wyniki pomiarów parametrów oscylatorów pierścieniowych

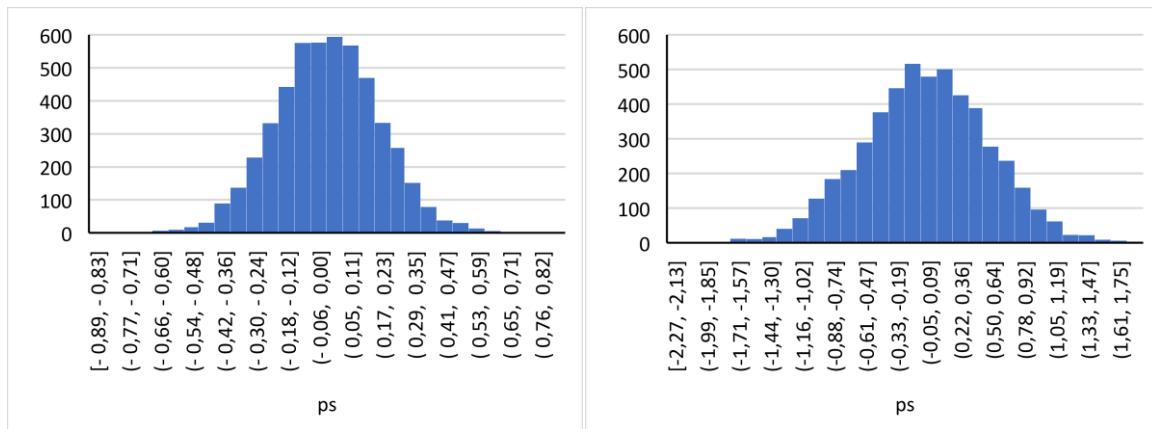
RODZAJ ELEMENTU OPÓŹNIAJĄCEGO	ŚREDNI OKRES [ns]	ODCHYLENIE STANDARDOWE [ps]	ŚREDNIA CZĘSTOTLIWOŚĆ [MHz]	JITTER J_{cc} [ps]
DWA INWERTERY	1,611975	0,356686	620,284643	0,891673
CZTERY INWERTERY	3,823743	0,856881	261,523815	1,263503
SZEŚĆ INWERTERÓW	5,425286	0,834084	184,287125	1,851567
OSIEM INWERTERÓW	6,138584	1,167592	162,871296	2,903784
DZIESIĘĆ INWERTERÓW	6,894132	0,979526	145,022057	3,653787
JEDEN ZATRZASK (LATCH)	1,401718	0,219868	713,410238	0,671906
DWA ZATRZASKI (LATCH)	2,694253	0,468249	371,090216	1,361389
TRZY ZATRZASKI (LATCH)	4,585984	0,644425	218,013382	1,433087
CZTERY ZATRZASKI (LATCH)	5,363355	0,605102	186,417631	1,694469
PIĘĆ ZATRZASKÓW (LATCH)	5,888442	0,779957	169,788632	1,651550
LINIA OPÓŹNIAJĄCA (CARRY4) 1 ODCZEP	1,422321	0,252037	702,954188	0,814570
LINIA OPÓŹNIAJĄCA (CARRY4) 2 ODCZEP	1,879508	0,320720	531,986773	0,964429
LINIA OPÓŹNIAJĄCA (CARRY4) 3 ODCZEP	2,212560	0,416462	451,923787	1,203147
LINIA OPÓŹNIAJĄCA (CARRY4) 4 ODCZEP	2,516784	0,520282	397,241307	1,522886
LINIA OPÓŹNIAJĄCA (CARRY4) 5 ODCZEP	2,686517	0,641456	372,174577	1,555622

Z danych przedstawionych w tabeli 5.1 wynika, że im dłuższe jest opóźnienie τ tym maksymalny jitter zawarty w sygnale jest większy. Należy jednak zwrócić uwagę na to, że procentowy udział jittera w okresie sygnału maleje wraz ze wzrostem okresu. Z punktu widzenia wytwarzania losowych bitów korzystne jest więc, aby okres sygnału wytwarzanego w oscylatorze pierścieniowym był jak najkrótszy. Rysunki 5.4 - 5.18 pokazują rozkład jittera. Z kolei w tabeli 5.2 przedstawiono wyniki testów zgodności

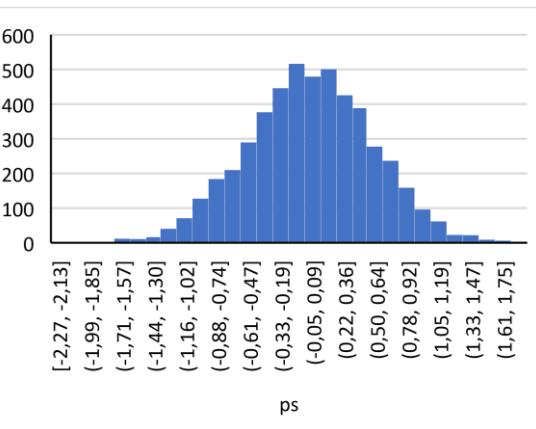
z rozkładem normalnym. Do oceny normalności rozkładu wykorzystano test Andersona-Darlinga i test zgodności χ^2 z poziomem istotności $\alpha = 0,05$. Jeżeli $P_{\text{wartość}}$ jest większa od poziomu istotności α to nie ma podstaw do odrzucenia hipotezy, że rozkład jest normalny. W większości przypadków mierzony jitter posiada charakterystykę rozkładu normalnego a więc przeważa w nim pożądany czynnik niedeterministyczny [62], [123]. Dla opóźnień realizowanych z wykorzystaniem linii opóźniającej testy nie zostały zdane, mimo że charakterystyka swoim kształtem przypomina rozkład normalny. W tym przypadku należy wnioskować, że we fluktuacjach fazy dominuje czynnik deterministyczny [68].

Tabela 5.2 Testy normalności rozkładu

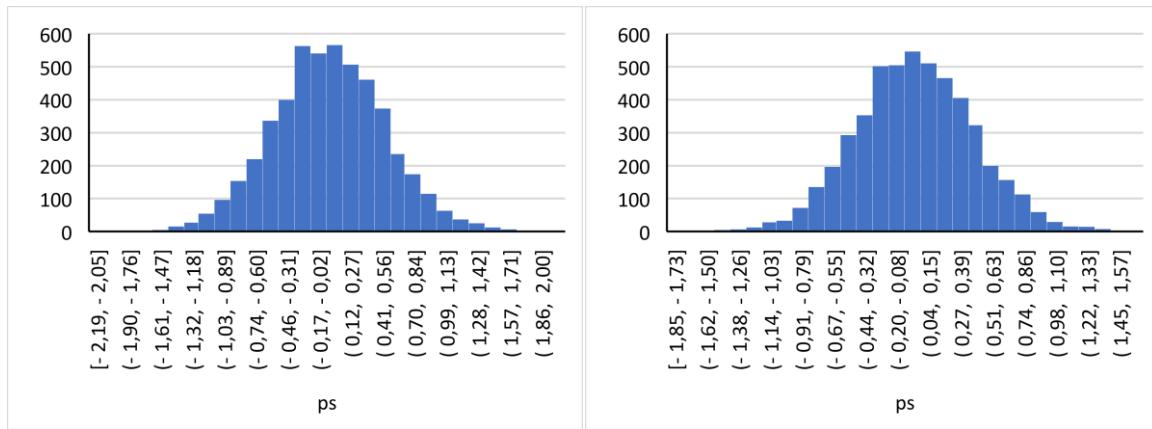
RODZAJ ELEMENTU OPÓŹNIAJĄCEGO	TEST ANDERSONA-DARLINGA	$P_{\text{WARTOŚĆ}}$	TEST ZGODNOŚCI χ^2	$P_{\text{WARTOŚĆ}}$
DWA INWERTERY	ZDANY	0,3687	ZDANY	0,1259
CZTERY INWERTERY	ZDANY	0,2037	ZDANY	0,4217
SZEŚĆ INWERTERÓW	NIEZDANY	0,0435	ZDANY	0,2027
OSIEM INWERTERÓW	ZDANY	0,0583	ZDANY	0,0884
DZIESIEC INWERTERÓW	ZDANY	0,1280	ZDANY	0,4992
JEDEN ZATRZASK (LATCH)	ZDANY	0,0524	NIEZDANY	0,0163
DWA ZATRZASKI (LATCH)	ZDANY	0,5940	ZDANY	0,6312
TRZY ZATRZASKI (LATCH)	ZDANY	0,0978	ZDANY	0,1536
CZTERY ZATRZASKI (LATCH)	ZDANY	0,6045	ZDANY	0,8706
PIĘĆ ZATRZASKÓW (LATCH)	ZDANY	0,8622	ZDANY	0,9447
LINIA OPÓŹNIAJĄCA (CARRY4) 1 ODCZEP	NIEZDANY	0,0466	NIEZDANY	0,0458
LINIA OPÓŹNIAJĄCA (CARRY4) 2 ODCZEP	NIEZDANY	0,0005	NIEZDANY	0,0000
LINIA OPÓŹNIAJĄCA (CARRY4) 3 ODCZEP	NIEZDANY	0,0283	ZDANY	0,1094
LINIA OPÓŹNIAJĄCA (CARRY4) 4 ODCZEP	NIEZDANY	0,0011	NIEZDANY	0,0063
LINIA OPÓŹNIAJĄCA (CARRY4) 5 ODCZEP	ZDANY	0,1167	NIEZDANY	0,0459



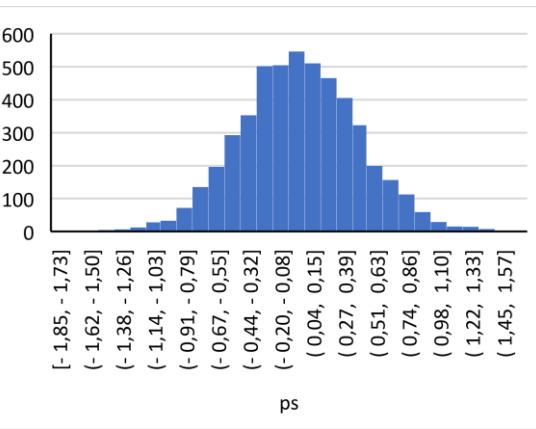
Rysunek 5.4 Histogram jitterera dla opóźnienia złożonego z dwóch inwerterów



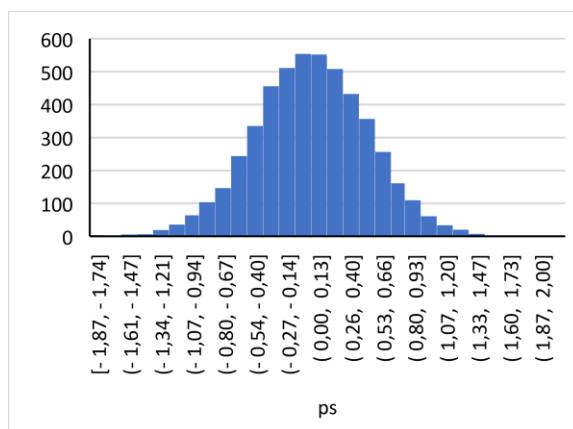
Rysunek 5.5 Histogram jitterera dla opóźnienia złożonego z czterech inwerterów



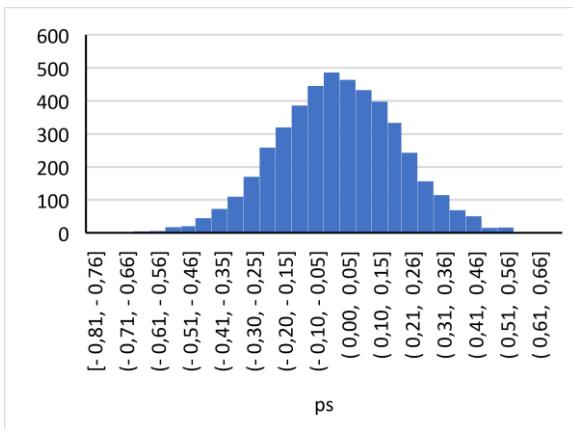
Rysunek 5.6 Histogram jitterera dla opóźnienia złożonego z sześciu inwerterów



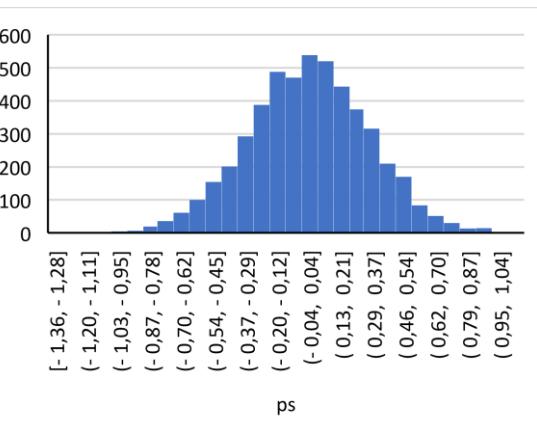
Rysunek 5.7 Histogram jitterera dla opóźnienia złożonego z ośmiu inwerterów



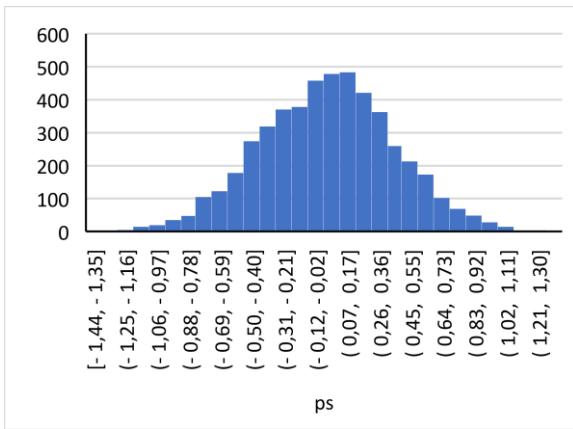
Rysunek 5.8 Histogram jitterera dla opóźnienia złożonego z dziesięciu inwerterów



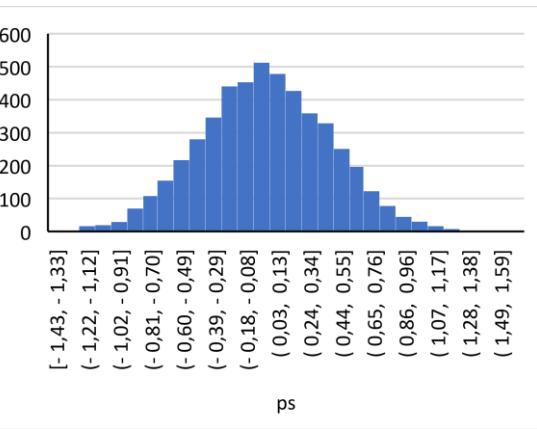
Rysunek 5.9 Histogram jittera dla opóźnienia złożonego z jednego zatrzasku



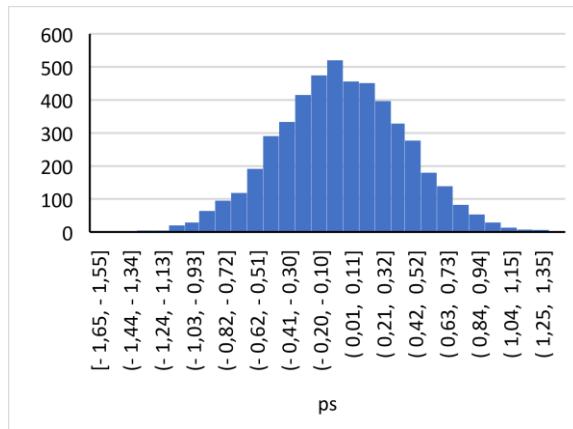
Rysunek 5.10 Histogram jittera dla opóźnienia złożonego z dwóch zatrzasków



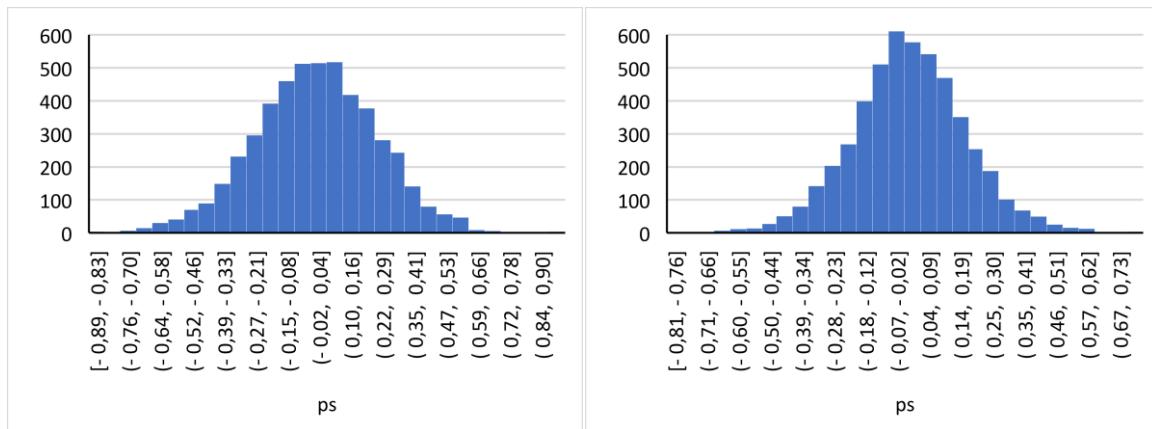
Rysunek 5.11 Histogram jittera dla opóźnienia złożonego z trzech zatrzasków



Rysunek 5.12 Histogram jittera dla opóźnienia złożonego z czterech zatrzasków

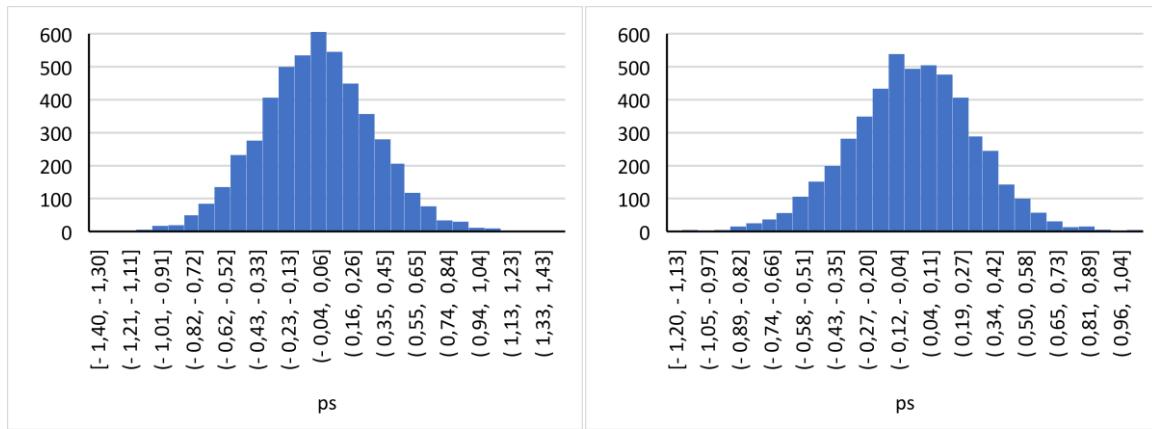


Rysunek 5.13 Histogram jittera dla opóźnienia złożonego z pięciu zatrzasków



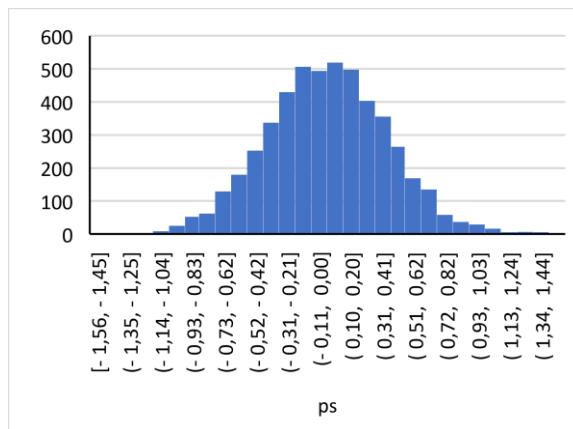
Rysunek 5.14 Histogram jittera dla opóźnienia złożonego z jednego odczepu linii opóźniającej (CARRY 4)

Rysunek 5.15 Histogram jittera dla opóźnienia złożonego z dwóch odczepów linii opóźniającej (CARRY 4)



Rysunek 5.16 Histogram jittera dla opóźnienia złożonego z trzech odczepów linii opóźniającej (CARRY 4)

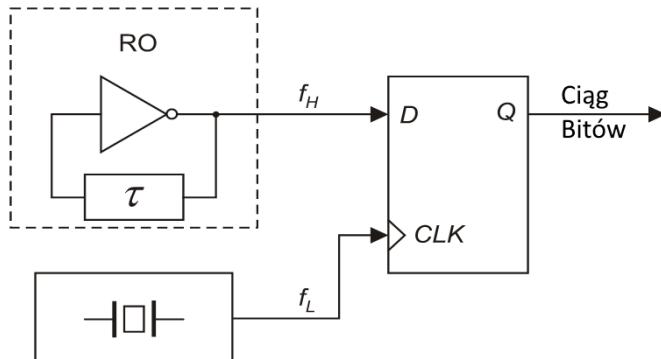
Rysunek 5.17 Histogram jittera dla opóźnienia złożonego z czterech odczepów linii opóźniającej (CARRY 4)



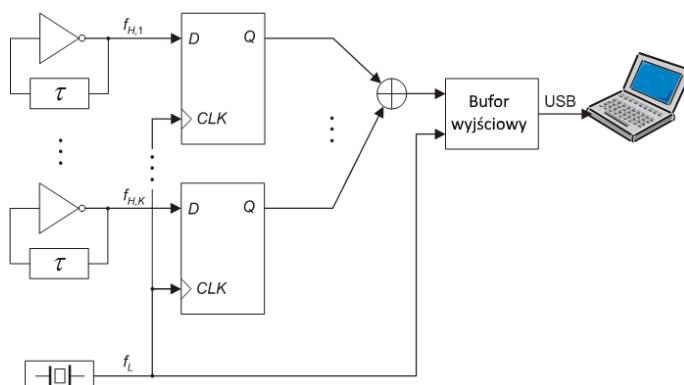
Rysunek 5.18 Histogram jittera dla opóźnienia złożonego z pięciu odczepów linii opóźniającej (CARRY 4)

5.2. KONCEPCJA ŁĄCZONEGO GENERATORA CIĄGÓW LOSOWYCH

Idea łączzonego generatora ciągów losowych została zaproponowana w pracy [8]. Polega na łączeniu za pomocą sumy modulo 2 wielu niezależnych źródeł losowości w postaci oscylatorów pierścieniowych. Następnie sygnał wyjściowy jest próbkowany za pomocą przerzutnika typu D (zobacz rys. 2.24). Taka konstrukcja ma jednak wady. Przede wszystkim drzewo bramek sumy modulo 2 nie jest zdolne propagować na wyjście sumowanych częstotliwości z wejścia. Potrzeba by do tego bardzo szybkich układów cyfrowych. Sygnał wyjściowy jest więc przefiltrowany dolnopasmowo, a częstotliwość graniczna związana jest z technologią wykonania układu cyfrowego. Jak pokazują autorzy pracy [26], taki schemat postępowania powoduje, że ciąg wynikowy uzyskany z próbkowania sygnału powstającego przez łączenie modulo 2 wielu źródeł losowości ma znacząco większe błędy próbkowania niż sygnał z pojedynczego źródła. Z tego powodu należy najpierw próbować sygnał z każdego źródła (rys. 5.19) a następnie łączyć modulo 2 poszczególne elementy ciągów binarnych [26], [61] tak jak pokazano na rysunku 5.20. Rezultatem takiej zmiany jest możliwość projektowania szybszych



Rysunek 5.19 Próbkowanie jittera oscylatora pierścieniowego jako metoda wytworzania losowych bitów



Rysunek 5.20 Łączenie modulo 2 wielu źródeł losowości

oscylatorów pierścieniowych wewnątrz struktur cyfrowych, co przekłada się na większą ilość losowości w wytwarzanym sygnale. Łączenie modulo 2 poszczególnych elementów ciągów binarnych przebiega ze stałą szybkością w chwilach czasu wyznaczanych przez oscylator kwarcowy o niskiej częstotliwości. Obliczanie sumy modulo 2 staje się deterministyczne, podczas gdy losowość jest zbierana przez próbkowanie swobodnie oscylujących generatorów pierścieniowych.

Wykorzystanie sumy modulo 2 nie tylko zwiększa udział losowości w ciągu wynikowym, ale również zmniejsza niezrównoważenie wynikowego ciągu. Zakładając n niezależnych ciągów losowych, każdy o wartości oczekiwanej μ to wartość oczekiwana ciągu po łączaniu modulo 2 wyraża się wzorem [132]:

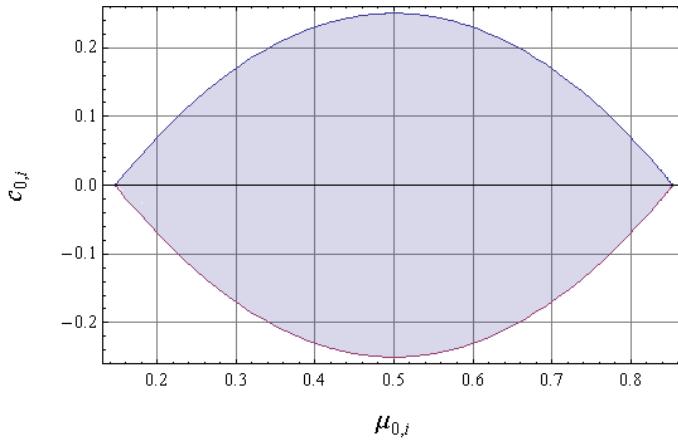
$$E(X_1 \oplus X_2, \dots, X_n) = \frac{1}{2} + (-2)^{n-1}(\mu - \frac{1}{2})^n. \quad (5.12)$$

Z równania (5.12) wynika, że wartość oczekiwana ciągu dąży do $\frac{1}{2}$ wraz ze wzrostem liczby n , łączonych modulo 2 niezależnych ciągów binarnych.

Wytwarzanie wielu niezależnych ciągów losowych w układach cyfrowych nie jest proste. W praktyce ciągi często są skorelowane między sobą [128], [133]. Skorelowane ciągi w rezultacie łączenia modulo 2 dają ciągi, które zawierają prawie same zera. W efekcie nie dostarczają losowości, i obciążają wynikowy ciąg a wartość oczekiwana takiego ciągu zależna jest od korelacji [132]. Należałyby tak projektować łączone generatory losowych ciągów binarnych, aby uniknąć korelacji między ciągami. W rzeczywistych układach jest to jednak bardzo trudne zadanie. Jednak tak silnie skorelowanych ciągów w praktyce jest mało [128]. W pracy [133] udowodniono, że łącząc modulo 2 odpowiednio dużą liczbę niezależnych ciągów o skorelowanych bitach można osiągnąć nieobciążony ciąg wyjściowy z nieskorelowanymi sąsiednimi bitami. Jednak poszczególne łączone ciągi muszą spełniać następujące założenia:

- wartość oczekiwana każdego ciągu należy do przedziału: $\mu_i \in \left(\frac{2-\sqrt{2}}{4}, \frac{2+\sqrt{2}}{4}\right)$
- kowariancja ciągów należy do przedziału: $c_i \in (2\mu_i^2 - 2\mu_i + 0,25, -2\mu_i^2 + 2\mu_i - 0,25)$

Graficzne przedstawienie zależności podobne jest do wykresu oka (rys. 5.21) z wartościami oczekiwanyymi na osi poziomej i wartościami dopuszczalnej kowariancji między sąsiednimi bitami na osi pionowej [133]. Gdy obciążenie wynosi zero, łączanie modulo 2 może zredukować kowariancję (korelację) między sąsiednimi bitami do zera dla dowolnej



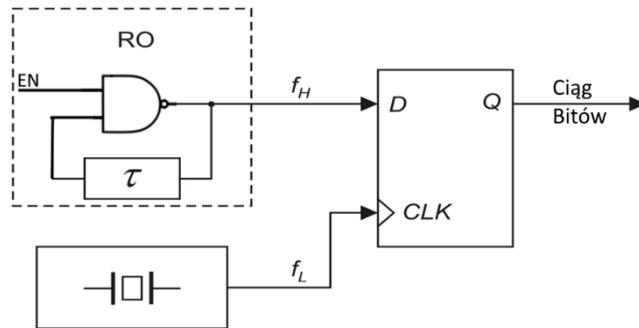
Rysunek 5.21 Wykres oka, wartości μ i c , dla których wytwarzanie są nieskorelowane bity [133]

wartości początkowej kowariancji z przedziału (-0,25: 0,25) lub dla dowolnej wartości początkowej korelacji z przedziału (-1: 1) [133]. To stwierdzenie jest prawdziwe, gdy łączone ciągi bitów są tworzone przez niezależne generatory. Niestety nie zawsze udaje się osiągnąć niezależność generatorów w jednym układzie scalonym. Niektóre generatory są zależne, a niektóre w pełni niezależne. Teoretycznie możliwe jest pogrupowanie źródłowych generatorów TRNG na dwa niezależne zestawy. Jeden zestaw zawiera generatorы zależne a drugi niezależne. Zestaw niezależnych generatorów wytwarza ciągi o małej korelacji między bitami, a zestaw zależnych generatorów wytwarza ciągi silnie skorelowane. Z przeprowadzonych obliczeń i eksperymentów wynika, że łącząc ciągi modulo 2 uzyskujemy znaczną i szybką redukcję obciążenia i korelacji pomiędzy sąsiednimi bitami, co wydaje się być niemożliwe bez wystarczająco dużej liczby niezależnych źródłowych generatorów TRNG. Wyznaczona eksperymentalnie minimalna liczba źródłowych generatorów pierścieniowych dla łączonego generatora losowych ciągów binarnych wynosi $n = 16$. Taki generator dostarcza losowych bitów z równym prawdopodobieństwem wystąpienia wartości zero i jeden i dostatecznie małą korelacją między bitami na wyjściu. Pozostaje jednak pytanie, czy zerowa korelacja między sąsiednimi bitami wytwarzanymi przez niezależne generatory przekłada się na niezależność tych bitów. To stwierdzenie może nie być prawdziwe ponieważ nieskorelowane zmienne losowe mogą być zależne.

5.3. ODPORNOŚĆ NA ATAK TYPU WSTRZYKIWANIE CZĘSTOTLIWOŚCI

Bardzo ważne jest aby generator ciągów losowych był odporny na różnego rodzaju ataki. Dlatego pożądane jest, żeby generator ciągów losowych był konstrukcją zintegrowaną z systemem kryptograficznym wewnątrz jednego układu scalonego. Taka budowa zmniejsza prawdopodobieństwo zewnętrznych manipulacji i podmiany generatora

na inny, taki który wytwarza ciągi znane atakującemu. Ważne jest również ciągłe monitorowanie jakości wytwarzanych ciągów i alarmowanie w przypadku, gdy ich właściwości losowe ulegną pogorszeniu. Pożądane jest też konstruowanie generatorów w taki sposób aby umożliwiały pracę start-stopową. Zmniejsza to ryzyko podsłuchu generatora. W generatorze ciągów losowych zbudowanym z oscylatorów pierścieniowych pracę start-stopową można zapewnić przez zamianę inwertera na bramkę NAND otwieraną przez zewnętrzny sygnał EN (rys. 5.22). Dla wysokiego stanu linii EN oscylator

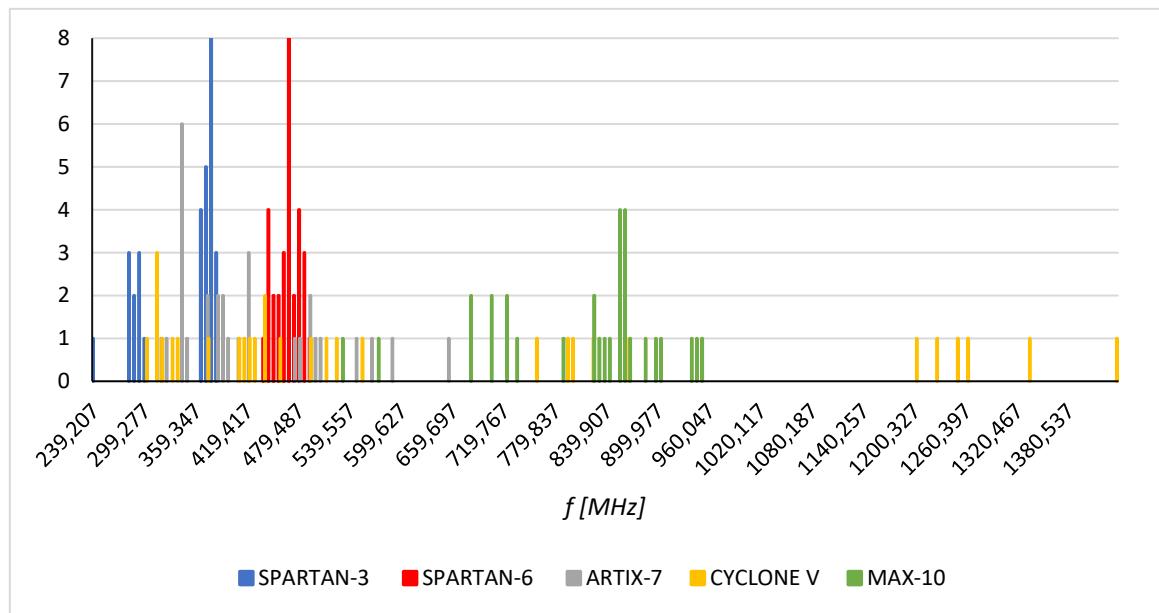


Rysunek 5.22 Start-stopowy generator TRNG

pierścieniowy wytwarza przebieg prostokątny, który jest próbkowany w przerzutniku typu D z częstotliwością generatora kwarcowego. Dla niskiego stanu linii EN oscylator nie działa. Taki generator TRNG może dostarczyć wymaganą liczbę bitów na żądanie, którą łatwo wyznaczyć za pomocą zliczania impulsów oscylatora kwarcowego.

Najpopularniejszym rodzajem ataku na generator ciągów losowych zbudowany z oscylatorów pierścieniowych jest tak zwany atak wstrzykiwania częstotliwości [67], [128], [134], [135]. Atak ten wykorzystuje znaną metodę synchronizacji dwóch oscylujących systemów przez iniekcję [136]. Jeżeli sygnał o częstotliwości zbliżonej do częstotliwości drgań swobodnych oscylatora lub jego harmonicznych zostanie w jakiś sposób wprowadzony do oscylatora, taki oscylator przestaje działać swobodnie i jest zmuszony do pracy z inną częstotliwością, to jest z częstotliwością sygnału wstrzykiwanego. Wstrzykiwanie częstotliwości w układach elektronicznych najczęściej odbywa się przez działanie na oscylator silnym polem elektromagnetycznym, zaburzając napięcie zasilające lub płaszczyznę masy oraz przez przeniki. Podczas tego ataku na generator TRNG złożony z oscylatorów pierścieniowych, pewna ich liczba jest zsynchronizowana do częstotliwości wstrzykiwanej eliminując niezależność generatorów źródłowych i zmniejszając losowość wyjściowego ciągu bitów. Jeżeli generator ciągów liczb losowych zbudowany jest z identycznych oscylatorów pierścieniowych, drgających ze zbliżoną częstotliwością, wówczas atak zmniejszy liczbę niezależnych generatorów

teoretycznie do zera. Aby zwiększyć odporność na taki atak możemy zastosować dwa rozwiązania. Po pierwsze, możemy zaprojektować oscylatory pierścieniowe o znacząco różnych częstotliwościach nominalnych zmieniając opóźnienie τ przez wykorzystanie innej liczby inwerterów, zatrzasków lub odczepów linii opóźniających dla różnych oscylatorów [68]. Po drugie, możemy zwiększyć liczbę generatorów źródłowych, implementowanych w FPGA. To podejście zwiększa zasoby, które mogą być ważne dla niektórych aplikacji, ale mają marginalne znaczenie w tradycyjnej kryptografii, gdzie zasoby niezbędne do monitorowania jakości pracy generatora TRNG znacznie przekraczają zasoby zajmowane przez sam generator ciągów losowych. Implementacja takiego generatora TRNG w układzie FPGA i wykorzystanie w procesie projektowania narzędzi automatycznego rozmieszczania i łączenia elementów (ang. *place and routing*) ma niewątpliwą zaletę. Narzędzia te automatycznie dobierają długości ścieżek łączących poszczególne elementy oscylatorów pierścieniowych wprowadzając opóźnienie. Długości te dla poszczególnych generatorów źródłowych są różne, dlatego oscylatory pracują z różną częstotliwością. Jak wspomniano w [62] oscylatory złożone z trzech inwerterów zapewniają największy rozrzut częstotliwości (rys. 5.23 i tabela 5.3). Stosując dodatkowo pracę start-stop i wysoką przepływność można uzyskać bezpieczną konstrukcję.



Rysunek 5.23 Rozrzut częstotliwości oscylatorów pierścieniowych

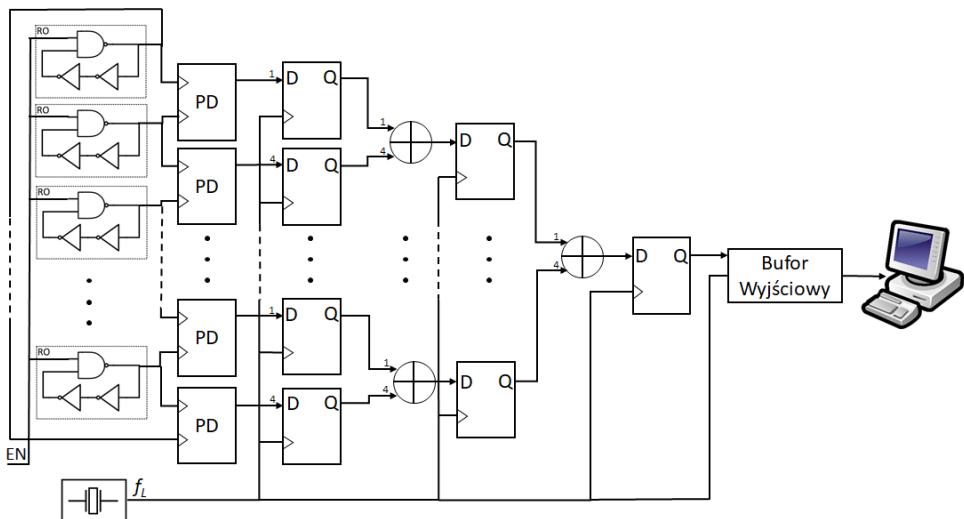
Tabela 5.3 Częstotliwości oscylatorów pierścieniowych w różnych układach FPGA

K	SPARTAN-3 [MHz]	SPARTAN-6 [MHz]	ARTIX-7 [MHz]	CYCLONE V [MHz]	MAX-10 [MHz]
1	366.4	444.8	583.2	295.6	839.2
2	361.2	466.2	393.8	454.6	675.2
3	361.8	443.4	336.4	1220.6	850.0
4	377.4	473.8	473.4	786.8	563.6
5	281.4	456.4	339.8	419.0	898.2
6	371.4	455.0	339.4	325.8	933.8
7	372.2	442.0	489.6	516.8	676.0
8	370.6	436.0	653.4	819.0	938.2
9	376.8	467.4	313.4	795.0	946.6
10	374.2	458.0	415.6	436.6	816.8
11	287.6	437.6	493.6	409.6	854.8
12	281.2	469.4	338.8	366.4	852.4
13	376.0	462.4	338.8	311.2	857.8
14	294.0	465.8	489.6	402.6	826.6
15	282.4	480.0	386.2	863.8	701.6
16	381.0	479.8	558.6	420.6	880.2
17	360.2	466.0	368.2	309.8	849.0
18	369.4	485.2	474.4	753.2	699.0
19	372.6	468.6	540.4	433.2	850.4
20	288.4	459.8	378.0	1255.4	717.2
21	361.6	479.2	340.8	1247.4	860.0
22	376.8	452.6	501.6	311.2	847.8
23	233.2	465.2	406.0	1196.4	718.2
24	367.8	463.0	378.6	488.8	729.8
25	281.0	465.6	346.6	1326.8	782.2
26	380.0	440.8	371.2	315.0	817.4
27	368.0	478.0	417.8	508.6	852.2
28	275.8	444.2	383.8	551.4	525.2
29	291.0	479.2	317.6	1434.6	833.0
30	375.0	487.6	415.4	333.6	892.0
ŚREDNIA	340.5	462.4	419.5	643.6	802.8
OD. STAN.	45.2	14.7	87.3	364.0	103.9
F _{MAX} -F _{MIN}	147.8	51.6	340.0	1139.0	421.4

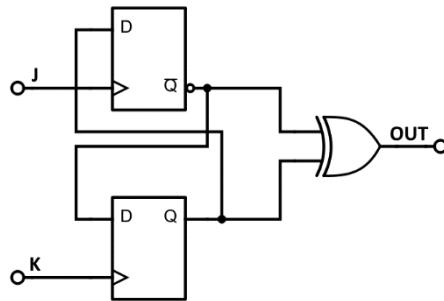
5.4. WYKORZYSTANIE DETEKTORA FAZY DO EKSTRAKCJI LOSOWOŚCI

Próbkując sygnał wyjściowy z oscylatora pierścieniowego za pomocą przerzutnika typu D taktowanego przez sygnał generatora kwarcowego, jesteśmy w stanie wychwycić szybkozmienne fluktuacje fazy oscylatora pierścieniowego. Wzbogacając łączony generator ciągów losowych o detektor fazy, który porównuje fazę między dwoma oscylatorami pierścieniowymi otrzymujemy układ wyznaczający różnicę faz między tymi oscylatorami. Różnica ta jest zależna od szumów fazy i dryfu częstotliwości między oscylatorami. Wartości te dla dwóch niezależnych oscylatorów pierścieniowych mają losowe wartości, a więc i różnica faz jest wartością losową. Schemat łączonego start-stopowego generatora TRNG wykorzystującego K oscylatorów pierścieniowych i K detektorów fazy przedstawiono na rysunku 5.24. Detektor fazy składa się z dwóch

przerzutników typu D i bramki XOR (rysunek 5.25). Jest to detektor, którego wejścia



Rysunek 5.24 Łaczony start-stopowy generator TRNG z detektorem fazy



Rysunek 5.25 Detektor fazy użyty w start-stopowym generatorze TRNG

reagują na narastające zbocze sygnałów wejściowych i nie jest on wrażliwy na wypełnienie i częstotliwość impulsów wejściowych. Tabelę prawdy detektora fazy przedstawia tabela 5.4.

Tabela 5.4 Tabela prawdy detektora fazy

J	K	OUT _{N+1}
0	0	OUT _n
0	1	0
1	0	1
1	1	OUT _n

Detektor fazy wyznacza różnicę faz między sąsiednimi oscylatorami pierścieniowymi. Wszystkie oscylatory pierścieniowe są zbudowane w ten sam sposób i wykorzystują opóźnienie w postaci dwóch inwerterów. Następnie różnica faz jest próbkiwana przy użyciu przerzutnika typu D taktowanego przez oscylator kwarcowy, który ustala wyjściową przepływność bitową. Bit y wytwarzane przez K źródłowych generatorów TRNG są łączone modulo 2 w kaskadzie, synchronicznie do próbkującego sygnału z generatora kwarcowego. Ponieważ TRNG został zaprojektowany do pracy w dowolnym

typie układu FPGA, podczas procesu implementacji wykorzystano narzędzia automatycznego rozmieszczenia i łączenia elementów dostarczane przez producentów układów FPGA. Generator został zaprojektowany do pracy z częstotliwością zegara większą niż 100 MHz. Pojedyncza bramka sumy modulo 2 lub raczej połączenie tablic LUT, które realizują bramkę modulo 2, nie może efektywnie przetwarzać wielu sygnałów z wysoką częstotliwością [127]. Aby uniknąć problemu błędów na wyjściu, sygnały są pogrupowane i dodane są dodatkowe przerzutniki. W obecnie produkowanych układach FPGA minimalna liczba wejść tablicy LUT wynosi cztery. Próbkowane sygnały z czterech detektorów fazy są łączone w jednej tablicy LUT. Bitы wyjściowe są następnie próbkowane przez ten sam zegar z wykorzystaniem przerzutnika typu D, aby zapewnić synchronizację i utworzyć nowy zestaw ciągów źródłowych. Nowe ciągi są następnie grupowane i łączone za pomocą bramki sumy modulo 2, a otrzymane bitы są próbkowane przez ten sam zegar, który wytwarza następny串 źródłowy, i tak dalej. Do oceny jakości wytwarzanych ciągów, bitы są przesyłane do komputera za pomocą bufora i interfejsu USB. Na komputerze nie jest wykonywane żadne przetwarzanie poprawiające właściwości statystyczne ciągów.

Wykorzystując detektor fazy wyznaczający różnicę faz między sąsiednimi oscylatorami pierścieniowymi (rys. 5.24) próbkowaniu podlega sygnał z większymi fluktuacjami fazy. Sygnał oscylatora pierścieniowego z jitterem można zapisać jako sygnał zmodulowany fazowo [137]:

$$s(t) = P[\omega t + \varphi(t)], \quad (5.13)$$

gdzie P oznacza串 okresowych impulsów, t czas, $\omega = 2\pi f$ pulsację sygnału okresowego o częstotliwości f , a φ szybkozmienne fluktuacje fazy. Wykorzystując rozwinięcie sygnału $s(t)$ w szereg Fouriera można wykazać, że sygnał prostokątny ma takie same właściwości fazowe co jego podstawowa harmoniczna [137]. Właściwość ta znacznie ułatwia analizę fluktuacji fazowych sygnału. Sinusoidalny sygnał z jitterem można zapisać jako:

$$s(t) = \sin(\omega t + \varphi(t)). \quad (5.14)$$

Rozpisując równanie (5.14) otrzymujemy:

$$s(t) = \sin(\omega t) \cos(\varphi(t)) + \sin(\varphi(t)) \cos(\omega t). \quad (5.15)$$

Ponieważ fluktuacje fazowe są małe, kosinus małych kątów jest bliski jedności, a sinus bliski wartości kąta, równanie (5.15) może być przybliżone w następujący sposób:

$$s(t) = \sin(\omega t) + \varphi(t) \cos(\omega t). \quad (5.16)$$

Sygnal z m -tego oscylatora pierścieniowego, gdzie $m = 0, 1, 2, \dots, K-1$, ma postać:

$$s_m(t) = \sin(\omega_m t) + \varphi_m(t) \cos(\omega_m t). \quad (5.17)$$

Z kolei na wyjściu m -tego detektora fazy otrzymujemy:

$$sd_m(t) = \sin(\omega_m t) + (\varphi_m(t) + \varphi_{(m+1) \bmod K}(t)) \cos(\omega_m t). \quad (5.18)$$

Następnie sygnał ten podlega próbkowaniu. Dla uproszczenia rozważono próbkowanie idealne ciągiem impulsów Diraca z odstępem T_s . Wzór na spróbkowany sygnał jest następujący:

$$sd_{\delta_m}(t) = \sum_{n=-\infty}^{\infty} [\sin(\omega_m n T_s) + (\varphi_m(n T_s) + \varphi_{(m+1) \bmod K}(n T_s)) \cos(\omega_m n T_s)] \delta(t - n T_s). \quad (5.19)$$

Po próbkowaniu sygnał jest łączony modulo 2. Ponieważ model jittera jest liniowy [138] to fluktuacje fazy z każdego oscylatora pierścieniowego propagują się na wyjście układu w postaci sumy. Sumę modulo dwa dwóch liczb a i b można zapisać jako:

$$a \oplus_2 b = \begin{cases} a + b, & \text{gdy } a + b < 2 \\ a + b - 2, & \text{gdy } a + b \geq 2 \end{cases} \quad (5.20)$$

Zatem sygnał po łączeniu modulo 2 K wyjść detektorów fazy wyraża się wzorem:

$$s_{\oplus}(t) = \sum_{n=-\infty}^{\infty} \sum_{m=1}^K [\sin(\omega_m n T_s) + (\varphi_m(n T_s) + \varphi_{(m+1) \bmod K}(n T_s)) \cos(\omega_m n T_s)] \delta(t - n T_s) - C, \quad (5.21)$$

gdzie $C = 0$, gdy wszystkie składniki sumy są mniejsze od 2 i $C = 2$ gdy składniki sumy są większe, równe 2.

Transformata Fouriera funkcji sinus jest następująca:

$$\sin(\omega_0 t) \xrightarrow{F} \frac{1}{2j} (\delta(\omega - \omega_0) - \delta(\omega + \omega_0)). \quad (5.22)$$

Funkcja kosinus ma transformatę Fouriera równą:

$$\cos(\omega_0 t) \xrightarrow{F} \frac{1}{2} (\delta(\omega - \omega_0) + \delta(\omega + \omega_0)), \quad (5.23)$$

zakładając, że widmo szybkozmiennych fluktuacji fazy jest:

$$\varphi_m(t) \xrightarrow{F} \Phi_m(\omega), \quad (5.24)$$

korzystając z właściwości delty Diraca:

$$x(t)\delta(t - t_0) = x(t_0)\delta(t - t_0), \quad (5.25)$$

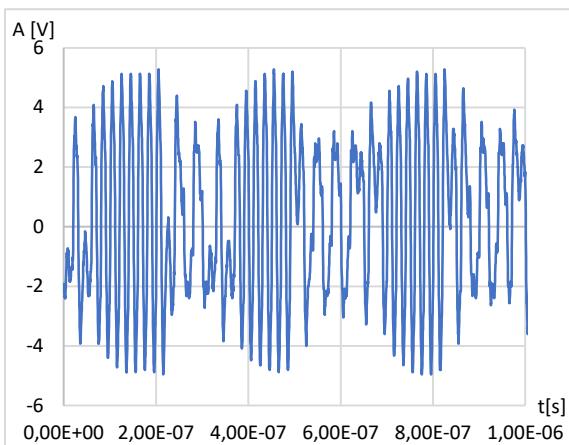
widmo sygnału (5.21) można zapisać jako:

$$\begin{aligned} S_{\oplus}(\omega) = & \frac{1}{2T_s} \sum_{n=-\infty}^{\infty} \sum_{m=1}^K \frac{1}{j} (\delta(\omega - \omega_m - n\omega_s) - \delta(\omega + \omega_m - n\omega_s)) + \Phi_m(\omega - \omega_m - n\omega_s) \\ & + \Phi_m(\omega + \omega_m - n\omega_s) + \Phi_{(m+1)\oplus K}(\omega - \omega_m - n\omega_s) \\ & + \Phi_{(m+1)\oplus K}(\omega + \omega_m - n\omega_s) - 2\pi C\delta(0). \end{aligned} \quad (5.26)$$

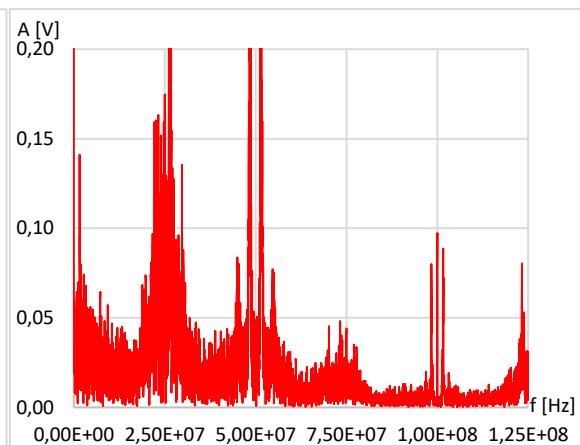
Dla porównania widmo sygnału z generatora pokazanego na rysunku 5.20 [61] ma postać:

$$\begin{aligned} Sw_{\oplus}(\omega) = & \frac{1}{2T_s} \sum_{n=-\infty}^{\infty} \sum_{m=1}^K \frac{1}{j} (\delta(\omega - \omega_m - n\omega_s) - \delta(\omega + \omega_m - n\omega_s)) \\ & + \Phi_m(\omega - \omega_m - n\omega_s) + \Phi_m(\omega + \omega_m - n\omega_s) - 2\pi C\delta(0). \end{aligned} \quad (5.27)$$

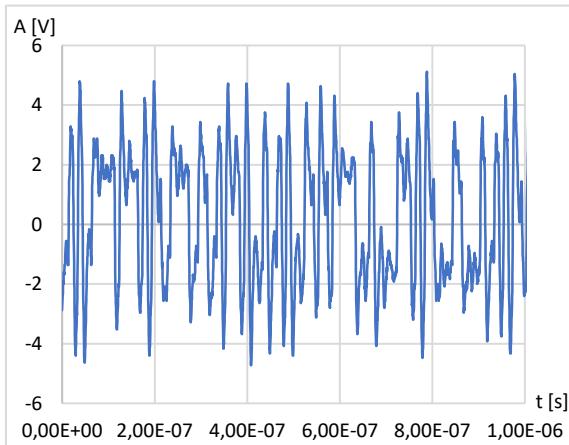
Analizując równania (5.26) i (5.27) można zauważyć, że w proponowanym generatorze składniki widma odpowiadające fluktuacjom fazy z m -tego oscylatora pierścieniowego propagują się na wyjście podwójnie na dwóch różnych podnośnych. Jest to zjawisko pożądane, gdyż pozwala dokładniej pokryć pasmo i zbliżyć sygnał do szumu białego. Na rysunkach 5.26 - 5.37 pokazano rzeczywiste obrazy ekranu oscyloskopu ukazujące przebiegi wyjściowe i widmo generatora według pracy [61] oraz generatora z detektorem fazy jako elementem pozyskującym losowość, proponowanego w rozprawie. Badane generatory składały się z 2, 4, i 8 źródłowych oscylatorów pierścieniowych. Częstotliwość próbkowania wynosiła 100 MHz, a szybkość bitowa była równa 100 Mb/s.



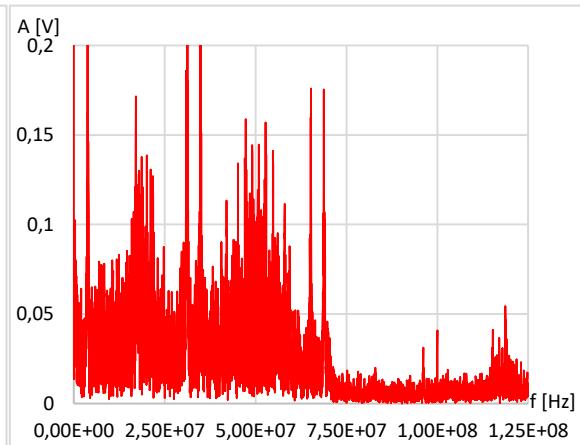
Rysunek 5.26 Przebieg czasowy generatora TRNG z pracy [61] złożonego z dwóch RO



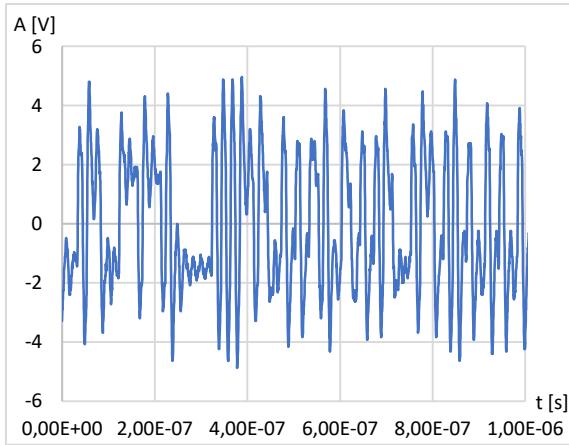
Rysunek 5.27 Widmo przebiegu generatora TRNG z pracy [61] złożonego z dwóch RO



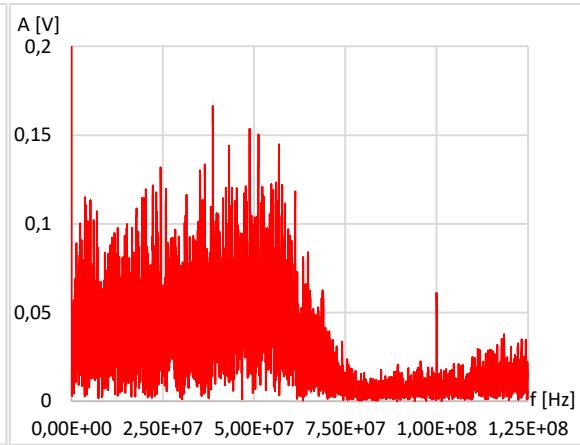
Rysunek 5.28 Przebieg czasowy generatora TRNG z pracy [61] złożonego z czterech RO



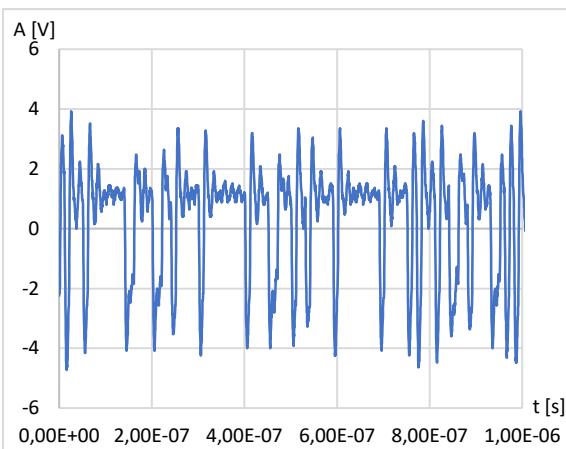
Rysunek 5.29 Widmo przebiegu generatora TRNG z pracy [61] złożonego z czterech RO



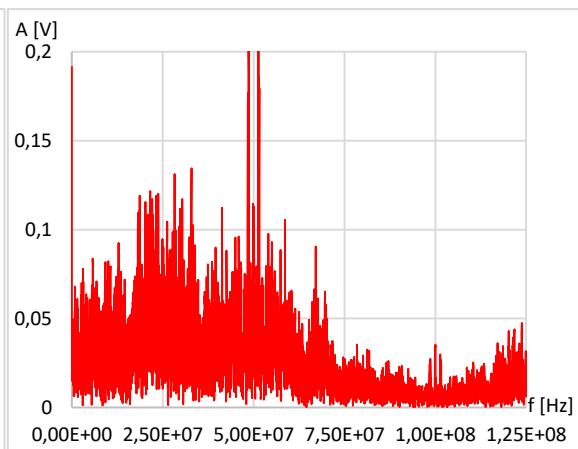
Rysunek 5.30 Przebieg czasowy generatora TRNG z pracy [61] złożonego z ośmiju RO



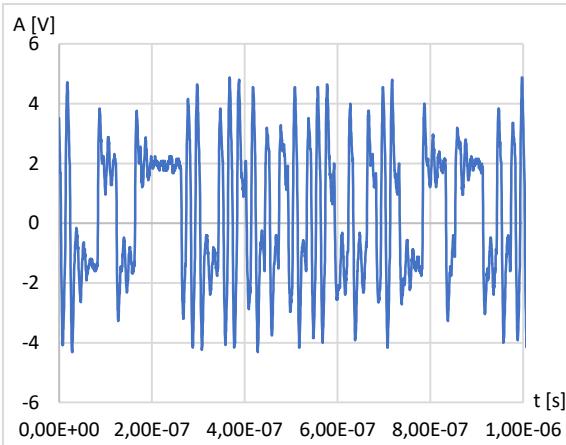
Rysunek 5.31 Widmo przebiegu generatora TRNG z pracy [61] złożonego z ośmiju RO



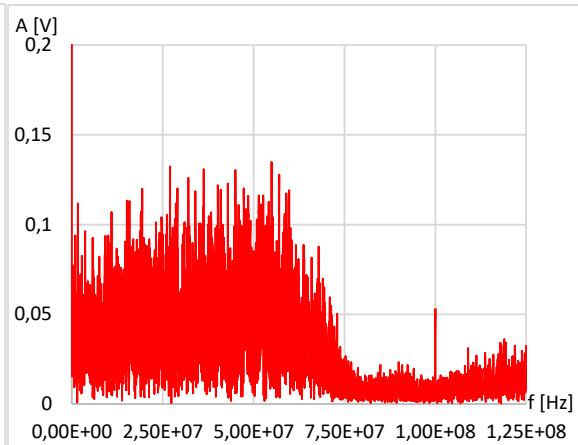
Rysunek 5.32 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z dwóch RO



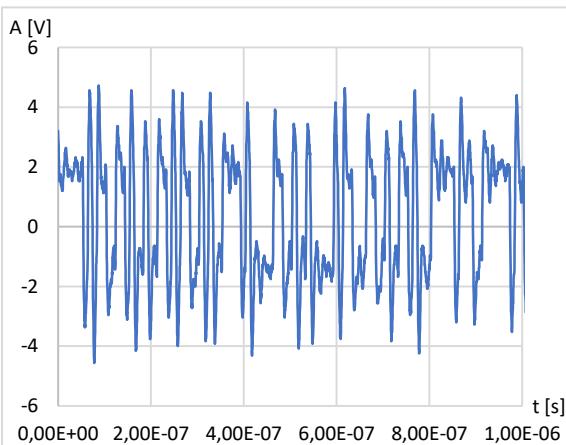
Rysunek 5.33 Widmo generatora TRNG z detektorem fazy złożonego z dwóch RO



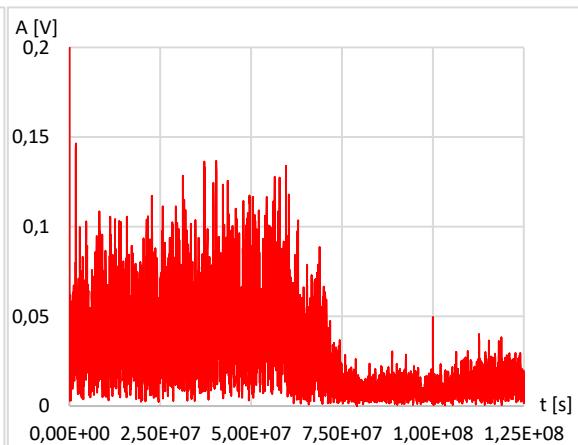
Rysunek 5.34 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z czterech RO



Rysunek 5.35 Widmo generatora TRNG z detektorem fazy złożonego z czterech RO



Rysunek 5.36 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z ośmiu RO



Rysunek 5.37 Widmo generatora TRNG z detektorem fazy złożonego z ośmiu RO

Dodając kolejne oscylatory pierścieniowe przebieg wyjściowy generatora TRNG staje się coraz bardziej zbliżony do ciągu losowego. Dzieje się tak zarówno dla generatora [61] jak i generatora z detektorem fazy. Dla generatora z detektorem fazy ten proces jest lepiej widoczny i już dla dwóch oscylatorów pierścieniowych przebieg przypomina ciąg losowych bitów. Analizując widma zauważać należy, że dla generatora z pracy [61] widmo jest skupione wokół kilku częstotliwości, podczas gdy widmo generatora z detektorem fazy jest bardziej rozproszone i lepiej pokrywa całe pasmo sygnału. Potwierdzają to wartości płaskości widmowej (tabela 5.5) wyznaczonej za pomocą entropii Wienera opisanej wzorem [139]:

$$F_l = \frac{\sqrt{\prod_{n=0}^{N-1} x(n)}}{\frac{1}{N} \sum_{n=0}^{N-1} x(n)} = \frac{e^{\frac{1}{N} \sum_{n=0}^{N-1} \ln x(n)}}{\frac{1}{N} \sum_{n=0}^{N-1} x(n)}, \quad (5.28)$$

gdzie $x(n)$ są kolejnymi próbками widma w zbiorze N próbek. Jest to średnia geometryczna próbek widma podzielona przez średnią arytmetyczną próbek widma. Płaskość widma równa jeden oznacza, że w mierzonym pasmie sygnału mamy do czynienia z szumem białym. Natomiast wartość F_l równa zero oznacza czysty ton.

Tabela 5.5 Porównanie wartości płaskości widma generatorów TRNG

	GENERATOR TRNG Z PRACY [61]	GENERATOR TRNG Z DETEKTOREM FAZY
2 OSCYLATORY PIERŚCIENIOWE	0,553671	0,673340
4 OSCYLATORY PIERŚCIENIOWE	0,599396	0,700494
8 OSCYULATORÓW PIERŚCIENIOWYCH	0,651399	0,707224

Generator z rysunku 5.24 został zaimplementowany w różnych układach FPGA pochodzących od głównych producentów tj. Xilinx, Altera (Intel) i Lattice mających około 90% rynku układów FPGA. Użyte w testach układy to:

Xilinx:

- Spartan-3 XC3S500E
- Spartan-6 XC6LX16
- Virtex-4 XC4VLX25
- Virtex-5 XC5VLX50T
- Virtex-6 XC6VLX240T
- Artix-7 XC7A35T

Altera (Intel):

- Cyclone-II EP2C35
- Cyclone-IV EP4CE22
- Cyclone-V 5CSEMA5

- Cyclone-10 10CL025
- Stratix-IV EP4SGX530
- MAX-10 10M50DA

Lattice:

- ECP3 LFE3-35EA-8FN484C
- ECP5 LFE5UM-45F-8BG381C.

Warto wspomnieć, że wyżej wymienione układy wytwarzane są z wykorzystaniem różnych technologii.

5.5. OCENA WŁAŚCIWOŚCI CIĄGÓW WYTWARZANYCH PRZEZ PROPONOWANY GENERATOR TRNG

Testy statystyczne z rekomendacji NIST 800-90B [108] zostały użyte do oceny, czy generator wytwarza niezależne bity o identycznym rozkładzie. NIST w zaleceniu 800-90B proponuje strategię testowania opartą na permutacjach. Takie testowanie jest sposobem na sprawdzenie hipotezy statystycznej, w której rzeczywistą wartość statystyki porównuje się z rozkładem referencyjnym, wywnioskowanym z danych wejściowych. Ogólne podejście do testowania opartego na permutacjach polega na wygenerowaniu 10000 permutacji ze zbioru danych wejściowych, obliczeniu statystyk testowych dla każdej permutacji i porównaniu wyniku ze statystyką testową wyliczoną na oryginalnym zbiorze danych. Czynność powtarza się dla każdego testu statystycznego. Jeżeli elementy ciągu są niezależne i o identycznym rozkładzie to permutacja zbioru danych nie powinna znacząco zmienić wartości statystyki testu. W szczególności oczekuje się, że pierwotny zestaw danych i jego permutacje pochodzą z tego samego rozkładu. Dlatego ich statystyki powinny być podobne. Niezwykle wysokie lub niskie wyniki testu powinny występować rzadko. Jednakże, jeśli próbki nie są IID, to oryginalne i permutowane statystyki testu mogą się znacznie różnić. Do wyznaczenia wyniku testu opartego na permutacjach wykorzystuje się liczniki oznaczane jako $C_{i,0}$ i $C_{i,1}$ [108]. Ekstremalne wartości liczników sugerują, że próbki danych nie są IID. Jeżeli suma $C_{i,0}$ i $C_{i,1}$ jest mniejsza od 5, oznacza to, że oryginalna statystyka testowa ma bardzo wysoką rangę. Odwrotnie, jeśli $C_{i,0}$ jest większe niż 9995, oznacza to, że oryginalne statystyki testowe mają bardzo niską rangę. Wartości graniczne dla $C_{i,0}$ i $C_{i,1}$ są wyznaczone dla poziomu istotności $\beta = 0,001$.

Testy statystyczne z pakietu NIST 800-22 [116] zostały użyte do oceny właściwości statystycznych ciągów wytwarzanych przez łączony generator TRNG z detektorami fazy.

NIST zaproponował tutaj dwa podejścia do testowania. Najpierw bada się proporcję R_β ciągów, które przechodzą test statystyczny, a następnie bada się czy rozkład P -wartości otrzymanych dla różnych ciągów jest rozkładem równomiernym w przedziale (0: 1). Dla testu jest obliczana P -wartość, oznaczana jako P_T [116]. Podczas testów przyjęto standardowy zestaw parametrów zaproponowany przez NIST. Dla poziomu istotności $\beta = 0,01$ i dla 1000 testowanych ciągów, każdy długości 10^6 bitów, minimalna wartość proporcji ciągów, które przechodzą testy wynosi około $R_\beta = 0,9805$. Jeżeli $P_T \geq 0.0001$ to można uznać, że badany ciąg ma rozkład równomierny.

Testowaniu poddano ciągi losowe wytworzone przez generatory dla $K = 10, 15, 20, 25, 30$ generatorów źródłowych i częstotliwości próbkowania $f_L = 100$ MHz, 200 MHz, 300 MHz. Dla testów z zalecenia NIST 800-90B, dla pojedynczego przypadku, wytworzono ciągi złożone z 1000000 bitów. Do wykonania testu restartów, według NIST, wykonano 1000 restartów. Podczas każdego重启u wytworzono 1000 bitów. Wyniki testów NIST 800-90B przedstawiono w tabelach 5.6 - 5.19. Testy statystyczne z pakietu NIST 800-22 wykonano na 1000 ciągach po 1000000 bitów każdy. Wyniki testów statystycznych NIST 800-22 przedstawiono w tabelach 5.20 - 5.33. Wszystkie wyniki dotyczą najgorszych warunków pracy łączonego generatora ciągów losowych, to jest najwyższej częstotliwości próbkowania $f_L = 300$ MHz oraz parametrów $K = 10$ i $K = 15$. We wszystkich przypadkach testy statystyczne są zdane dla $K \geq 15$ źródłowych oscylatorów pierścieniowych. W niektórych układach FPGA wszystkie testy są spełnione już dla $K = 10$ oscylatorów pierścieniowych, co można uznać za wartość graniczną dla tego typu generatora TRNG. Ocenić należy, że odpowiednia minimalna liczba generatorów źródłowych niezbędnych do zdania wszystkich testów z pakietu NIST 800-22 i NIST 800-90B dla $f_L = 300$ MHz i dowolnego producenta układów FPGA wynosi 15.

Tabela 5.6 Wyniki testów statystycznych NIST 800-90B dla układu Spartan 3

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	22	0	TAK	10	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	7	6	TAK	21	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	12	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	9	1	TAK	298	1	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	11	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	11	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	20	2	TAK	2	4	TAK
TEST OKRESOWOŚCI*	6	0	TAK	17	0	TAK
TEST KOWARIANCJI*	6	0	TAK	6	0	TAK
TEST KOMPRESJI	15	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994804			0,994856		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.7 Wyniki testów statystycznych NIST 800-90B dla układu Spartan 6

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	1213	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	11	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	4	3	TAK	2	4	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	27	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	14	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	72	0	TAK	33	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	11	1	TAK	7	2	TAK
TEST OKRESOWOŚCI*	7	0	TAK	9	0	TAK
TEST KOWARIANCJI*	7	0	TAK	11	0	TAK
TEST KOMPRESJI	13	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995955			0,995034		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.8 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 4

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	2	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	3	3	TAK	2	4	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	22	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	8	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	17	3	TAK	36	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	3	0	TAK	4	2	TAK
TEST OKRESOWOŚCI*	6	0	TAK	8	0	TAK
TEST KOWARIANCJI*	6	0	TAK	7	0	TAK
TEST KOMPRESJI	6	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUJSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995039			0,992709		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.9 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 5

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	31	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	37	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	2	1	NIE	1	5	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	30	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	15	0	TAK	50	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	11	0	TAK
TEST ŚREDNIEJ KOLIZJI	14	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	27	4	TAK	5	5	TAK
TEST OKRESOWOŚCI*	6	0	TAK	671	0	TAK
TEST KOWARIANCJI*	6	0	TAK	138	0	TAK
TEST KOMPRESJI	10000	0	NIE	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUJSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,993840			0,995115		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.10 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 6

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	14	5	TAK	12	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	56	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	5	1	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	11	0	TAK	44	0	TAK
TEST ŚREDNIEJ KOLIZJI	20	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	40	5	TAK	3	3	TAK
TEST OKRESOWOŚCI*	6	0	TAK	6	0	TAK
TEST KOWARIANCJI*	6	0	TAK	6	0	TAK
TEST KOMPRESJI	6	0	TAK	25	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994914			0,994704		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.11 Wyniki testów statystycznych NIST 800-90B dla układu Artix 7

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	117	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	43	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	7	5	TAK	7	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	10	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	47	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	12	1	TAK
TEST ŚREDNIEJ KOLIZJI	10	0	TAK	796	1	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	3	3	TAK	4	2	TAK
TEST OKRESOWOŚCI*	6	0	TAK	5	0	TAK
TEST KOWARIANCJI*	6	0	TAK	6	0	TAK
TEST KOMPRESJI	6	0	TAK	75	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995869			0,995103		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.12 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone II

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	28	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	2	4	TAK	20	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	8	0	TAK	7	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	12	0	TAK	10	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	225	0	TAK
TEST ŚREDNIEJ KOLIZJI	25	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	6	0	TAK	4	2	TAK
TEST OKRESOWOŚCI*	6	0	TAK	705	0	TAK
TEST KOWARIANCJI*	6	0	TAK	21	0	TAK
TEST KOMPRESJI	31	0	TAK	40	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,993597			0,993704		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.13 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone IV

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	11	0	TAK	225	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	36	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	15	6	TAK	3	3	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	12	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	22	5	TAK	7	2	TAK
TEST OKRESOWOŚCI*	42	0	TAK	15	0	TAK
TEST KOWARIANCJI*	18	0	TAK	27	0	TAK
TEST KOMPRESJI	6	0	TAK	15	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994916			0,996047		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.14 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone V

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	18	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	15	5	TAK	6	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	23	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	16	0	TAK	8	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	22	0	TAK	29	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	41	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	5	1	TAK	4	2	TAK
TEST OKRESOWOŚCI*	38	0	TAK	55	1	TAK
TEST KOWARIANCJI*	87	0	TAK	109	0	TAK
TEST KOMPRESJI	108	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994000			0,994741		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.15 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone 10

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	38	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	1	5	TAK	1	5	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	23	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	24	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	21	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	2	4	TAK	6	0	TAK
TEST OKRESOWOŚCI*	107	0	TAK	15	1	TAK
TEST KOWARIANCJI*	110	0	TAK	110	0	TAK
TEST KOMPRESJI	10	0	TAK	8	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994641			0,995034		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.16 Wyniki testów statystycznych NIST 800-90B dla układu Stratix IV

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	11	6	TAK	3	4	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	40	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	0	0	NIE	11	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	84	0	TAK
TEST ŚREDNIEJ KOLIZJI	10000	0	NIE	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	4	2	TAK	4	2	TAK
TEST OKRESOWOŚCI*	573	0	TAK	120	1	TAK
TEST KOWARIANCJI*	10000	0	NIE	32	0	TAK
TEST KOMPRESJI	10000	0	NIE	40	0	TAK
TEST DOBROCI DOPASOWANIA	NIEZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	NIEZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,800764			0,995190		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.17 Wyniki testów statystycznych NIST 800-90B dla układu MAX 10

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	6	0	TAK	14	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	8	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	3	3	TAK	2	4	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	14	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	7	1	TAK	2	4	TAK
TEST OKRESOWOŚCI*	6	1	TAK	6	0	TAK
TEST KOWARIANCJI*	49	0	TAK	20	0	TAK
TEST KOMPRESJI	32	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995323			0,994846		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.18 Wyniki testów statystycznych NIST 800-90B dla układu ECP3

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	31	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	6	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	3	3	TAK	8	6	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	97	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	11	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	9	4	TAK	7	1	TAK
TEST OKRESOWOŚCI*	61	0	TAK	6	0	TAK
TEST KOWARIANCJI*	21	0	TAK	22	0	TAK
TEST KOMPRESJI	10000	0	NIE	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995996			0,995996		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.19 Wyniki testów statystycznych NIST 800-90B dla układu ECP5

RODZAJ TESTU	K = 10			K = 15		
	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>	<i>C_{i,0}</i>	<i>C_{i,1}</i>	<i>IID</i>
TEST BŁĄDZENIA	72	0	TAK	8	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	19	0	TAK	44	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	2	4	TAK	4	2	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	12	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	6	0	TAK	5	1	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	297	0	TAK	47	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	4	2	TAK	3	3	TAK
TEST OKRESOWOŚCI*	140	0	TAK	64	0	TAK
TEST KOWARIANCJI*	3470	0	TAK	56	0	TAK
TEST KOMPRESJI	6	0	TAK	118	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	NIEZDANY			ZDANY		
MINIMALNA ENTROPIA	0,994086			0,995287		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.20 Wyniki testów statystycznych NIST 800-22 dla układu Spartan 3

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,982	0,000006	0,994	0,009467
BLOKOWY TEST CZĘSTOŚCI	0,994	0,628790	0,987	0,546283
TEST SKUMULOWANYCH SUM*	0,980	0,001475	0,991	0,135720
TEST CIĄGÓW	0,979	0,000830	0,992	0,628790
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,989	0,897763	0,988	0,404728
TEST STOPNIA MACIERZY BINARNEJ	0,993	0,624627	0,992	0,358641
TEST SPEKTRALNY DFT	0,990	0,006758	0,987	0,024028
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,982	0,406499	0,982	0,987079
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,990	0,104993	0,988	0,977480
TEST UNIERSALNY MAURERA	0,988	0,657933	0,991	0,285427
TEST PRZYBLIŻONEJ ENTROPII	0,988	0,142872	0,988	0,610070
TEST BŁĄDZENIA LOSOWEGO**	0,979	0,013520	0,973	0,655009
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,986	0,206591	0,984	0,554739
TEST SERII*	0,989	0,794391	0,988	0,224821
TEST ZŁOŻONOŚCI LINIOWEJ	0,986	0,138069	0,991	0,295391

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.21 Wyniki testów statystycznych NIST 800-22 dla układu Spartan 6

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,988	0,260930	0,988	0,703417
BLOKOWY TEST CZĘSTOŚCI	0,991	0,019587	0,992	0,873987
TEST SKUMULOWANYCH SUM*	0,985	0,337688	0,984	0,504219
TEST CIĄGÓW	0,990	0,562591	0,989	0,926027
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,983	0,771469	0,991	0,164425
TEST STOPNIA MACIERZY BINARNEJ	0,987	0,132640	0,991	0,415422
TEST SPEKTRALNY DFT	0,993	0,786830	0,986	0,032061
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,982	0,197981	0,987	0,518106
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,986	0,165340	0,991	0,670396
TEST UNIERSALNY MAURERA	0,990	0,066882	0,988	0,961039
TEST PRZYBLIŻONEJ ENTROPII	0,992	0,593478	0,992	0,079538
TEST BŁĄDZENIA LOSOWEGO**	0,986	0,980192	0,977	0,314042
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,984	0,765069	0,977	0,807412
TEST SERII*	0,983	0,956729	0,990	0,368587
TEST ZŁOŻONOŚCI LINIOWEJ	0,987	0,313041	0,992	0,263572

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.22 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 4

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,991	0,881662	0,989	0,463512
BLOKOWY TEST CZĘSTOŚCI	0,989	0,455937	0,990	0,474986
TEST SKUMULOWANYCH SUM*	0,991	0,994488	0,987	0,035174
TEST CIĄGÓW	0,991	0,953089	0,988	0,939005
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,990	0,461612	0,990	0,707513
TEST STOPNIA MACIERZY BINARNEJ	0,993	0,154629	0,990	0,263572
TEST SPEKTRALNY DFT	0,990	0,743915	0,985	0,890582
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,979	0,603841	0,982	0,723804
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,992	0,877083	0,992	0,311542
TEST UNIERSALNY MAURERA	0,992	0,532132	0,984	0,637119
TEST PRZYBLIŻONEJ ENTROPII	0,990	0,211064	0,993	0,686955
TEST BŁĄDZENIA LOSOWEGO**	0,983	0,376284	0,982	0,081639
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,985	0,130111	0,979	0,312750
TEST SERII*	0,985	0,130369	0,991	0,755819
TEST ZŁOŻONOŚCI LINIOWEJ	0,988	0,241741	0,992	0,375313

GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.23 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 5

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,993	0,478839	0,987	0,439122
BLOKOWY TEST CZĘSTOŚCI	0,987	0,674543	0,993	0,761719
TEST SKUMULOWANYCH SUM*	0,988	0,239266	0,988	0,282626
TEST CIĄGÓW	0,988	0,649612	0,990	0,542228
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,979	0,864494	0,987	0,180568
TEST STOPNIA MACIERZY BINARNEJ	0,987	0,630872	0,990	0,803720
TEST SPEKTRALNY DFT	0,988	0,331408	0,991	0,492436
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,976	0,329850	0,982	0,158133
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,992	0,786830	0,988	0,993493
TEST UNIERSALNY MAURERA	0,985	0,136499	0,991	0,755819
TEST PRZYBLIŻONEJ ENTROPII	0,993	0,100109	0,989	0,299736
TEST BŁĄDZENIA LOSOWEGO**	0,978	0,107428	0,987	0,028539
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,987	0,401777	0,984	0,705598
TEST SERII*	0,987	0,183547	0,991	0,180568
TEST ZŁOŻONOŚCI LINIOWEJ	0,988	0,620465	0,986	0,894918

GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.24 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 6

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,993	0,944274	0,990	0,036833
BLOKOWY TEST CZĘSTOŚCI	0,988	0,419021	0,988	0,779188
TEST SKUMULOWANYCH SUM*	0,991	0,534146	0,989	0,145326
TEST CIĄGÓW	0,987	0,114040	0,992	0,003967
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,992	0,767582	0,986	0,422638
TEST STOPNIA MACIERZY BINARNEJ	0,994	0,830808	0,990	0,015171
TEST SPEKTRALNY DFT	0,990	0,118812	0,983	0,014754
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,980	0,263572	0,983	0,940080
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,992	0,401199	0,989	0,004177
TEST UNIwersalny MAURERA	0,986	0,801865	0,986	0,361938
TEST PRZYBLIŻONEJ ENTROPII	0,991	0,004837	0,995	0,202268
TEST BŁĄDZENIA LOSOWEGO**	0,990	0,596228	0,984	0,622481
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,982	0,231444	0,986	0,667997
TEST SERII*	0,983	0,727851	0,988	0,798139
TEST ZŁOŻONOŚCI LINIOWEJ	0,987	0,275709	0,990	0,492436

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.25 Wyniki testów statystycznych NIST 800-22 dla układu Artix 7

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,988	0,769527	0,988	0,411840
BLOKOWY TEST CZĘSTOŚCI	0,990	0,006661	0,992	0,773405
TEST SKUMULOWANYCH SUM*	0,987	0,786830	0,987	0,114712
TEST CIĄGÓW	0,985	0,118120	0,996	0,406499
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,992	0,433590	0,993	0,162606
TEST STOPNIA MACIERZY BINARNEJ	0,992	0,812905	0,993	0,803720
TEST SPEKTRALNY DFT	0,989	0,925287	0,992	0,230755
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,984	0,092041	0,983	0,705466
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,984	0,498313	0,990	0,101311
TEST UNIwersalny MAURERA	0,986	0,641284	0,989	0,032705
TEST PRZYBLIŻONEJ ENTROPII	0,986	0,169981	0,987	0,823725
TEST BŁĄDZENIA LOSOWEGO**	0,987	0,098273	0,985	0,539193
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,983	0,337586	0,983	0,218048
TEST SERII*	0,991	0,272977	0,990	0,530120
TEST ZŁOŻONOŚCI LINIOWEJ	0,985	0,080027	0,988	0,593478

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.26 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone II

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,993	0,440975	0,993	0,518106
BLOKOWY TEST CZĘSTOŚCI	0,987	0,251837	0,987	0,172816
TEST SKUMULOWANYCH SUM*	0,992	0,461612	0,992	0,223648
TEST CIĄGÓW	0,988	0,463512	0,994	0,903338
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,993	0,528111	0,990	0,240501
TEST STOPNIA MACIERZY BINARNEJ	0,993	0,809249	0,989	0,488534
TEST SPEKTRALNY DFT	0,985	0,820143	0,987	0,576961
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,980	0,917870	0,981	0,147815
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,990	0,713641	0,989	0,307077
TEST UNIwersalny MAURERA	0,991	0,811080	0,993	0,637119
TEST PRZYBLIŻONEJ ENTROPII	0,994	0,703417	0,990	0,206629
TEST BŁĄDZENIA LOSOWEGO**	0,984	0,432890	0,981	0,292383
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,985	0,118978	0,984	0,228008
TEST SERII*	0,990	0,021554	0,987	0,009467
TEST ZŁOŻONOŚCI LINIOWEJ	0,991	0,771469	0,988	0,622546

GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.27 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone IV

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,992	0,595549	0,994	0,078567
BLOKOWY TEST CZĘSTOŚCI	0,992	0,444691	0,991	0,761719
TEST SKUMULOWANYCH SUM*	0,989	0,116065	0,991	0,348869
TEST CIĄGÓW	0,994	0,801865	0,993	0,171867
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,991	0,406499	0,991	0,262249
TEST STOPNIA MACIERZY BINARNEJ	0,987	0,278461	0,989	0,796268
TEST SPEKTRALNY DFT	0,988	0,821937	0,988	0,779188
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,980	0,000026	0,983	0,009007
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,991	0,921624	0,988	0,699313
TEST UNIwersalny MAURERA	0,986	0,087692	0,984	0,031428
TEST PRZYBLIŻONEJ ENTROPII	0,991	0,003371	0,989	0,192724
TEST BŁĄDZENIA LOSOWEGO**	0,990	0,333231	0,983	0,674611
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,986	0,208652	0,982	0,965581
TEST SERII*	0,984	0,114712	0,989	0,052947
TEST ZŁOŻONOŚCI LINIOWEJ	0,987	0,597620	0,993	0,593478

GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.28 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone V

RODZAJ TESTU	K = 10		K = 15	
	R_B	P_T	R_B	P_T
TEST CZĘSTOŚCI	0,987	0,031637	0,992	0,796268
BLOKOWY TEST CZĘSTOŚCI	0,991	0,649612	0,991	0,794391
TEST SKUMULOWANYCH SUM*	0,985	0,459717	0,992	0,595549
TEST CIĄGÓW	0,984	0,365253	0,988	0,695200
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,994	0,225998	0,990	0,751866
TEST STOPNIA MACIERZY BINARNEJ	0,988	0,001316	0,987	0,889118
TEST SPEKTRALNY DFT	0,991	0,965083	0,989	0,725829
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,979	0,670396	0,985	0,540204
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,990	0,693142	0,991	0,825505
TEST UNIwersalny MAURERA	0,993	0,193767	0,987	0,701366
TEST PRZYBLIŻONEJ ENTROPII	0,989	0,130369	0,989	0,163513
TEST BŁĄDZENIA LOSOWEGO**	0,987	0,016958	0,979	0,329085
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,982	0,550218	0,985	0,485416
TEST SERII*	0,990	0,266235	0,992	0,043087
TEST ZŁOŻONOŚCI LINIOWEJ	0,991	0,415422	0,989	0,607993

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.29 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone 10

RODZAJ TESTU	K = 10		K = 15	
	R_B	P_T	R_B	P_T
TEST CZĘSTOŚCI	0,992	0,193767	0,989	0,085587
BLOKOWY TEST CZĘSTOŚCI	0,984	0,081013	0,991	0,934599
TEST SKUMULOWANYCH SUM*	0,988	0,703417	0,986	0,155499
TEST CIĄGÓW	0,983	0,570792	0,989	0,932333
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,991	0,552383	0,988	0,340858
TEST STOPNIA MACIERZY BINARNEJ	0,992	0,961869	0,987	0,943242
TEST SPEKTRALNY DFT	0,988	0,027313	0,988	0,751866
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,983	0,496351	0,983	0,224821
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,993	0,344048	0,989	0,713641
TEST UNIwersalny MAURERA	0,986	0,041169	0,986	0,419021
TEST PRZYBLIŻONEJ ENTROPII	0,993	0,446556	0,990	0,921624
TEST BŁĄDZENIA LOSOWEGO**	0,982	0,600812	0,985	0,872450
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,987	0,312136	0,989	0,037449
TEST SERII*	0,991	0,108150	0,988	0,128132
TEST ZŁOŻONOŚCI LINIOWEJ	0,990	0,428095	0,988	0,859637

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.30 Wyniki testów statystycznych NIST 800-22 dla układu Stratix IV

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,989	0,411840	0,992	0,998474
BLOKOWY TEST CZĘSTOŚCI	0,992	0,906069	0,992	0,949278
TEST SKUMULOWANYCH SUM*	0,987	0,158133	0,992	0,490483
TEST CIĄGÓW	0,970	0,000000	0,993	0,781106
TEST NA NAJDŁUŻSY CIĄG JEDYNEK W BLOKU	0,992	0,069863	0,988	0,444691
TEST STOPNIA MACIERZY BINARNEJ	0,990	0,773405	0,991	0,883171
TEST SPEKTRALNY DFT	0,986	0,187581	0,991	0,000336
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,978	0,616305	0,983	0,897763
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,988	0,293952	0,994	0,589341
TEST UNIwersalny MAURERA	0,986	0,630872	0,992	0,532132
TEST PRZYBLIŻONEJ ENTROPII	0,985	0,045380	0,991	0,484646
TEST BŁĄDZENIA LOSOWEGO**	0,983	0,930692	0,983	0,596885
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,986	0,410494	0,985	0,790255
TEST SERII*	0,990	0,508172	0,989	0,916599
TEST ZŁOŻONOŚCI LINIOWEJ	0,994	0,618385	0,991	0,801865

GWIĄZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.31 Wyniki testów statystycznych NIST 800-22 dla układu MAX 10

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,988	0,934599	0,989	0,455937
BLOKOWY TEST CZĘSTOŚCI	0,993	0,205531	0,994	0,699313
TEST SKUMULOWANYCH SUM*	0,991	0,336111	0,988	0,727851
TEST CIĄGÓW	0,989	0,697860	0,991	0,756974
TEST NA NAJDŁUŻSY CIĄG JEDYNEK W BLOKU	0,986	0,794391	0,992	0,777265
TEST STOPNIA MACIERZY BINARNEJ	0,992	0,471146	0,990	0,610070
TEST SPEKTRALNY DFT	0,986	0,672470	0,981	0,231956
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,992	0,000024	0,983	0,173770
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,992	0,200115	0,994	0,653773
TEST UNIwersalny MAURERA	0,990	0,901959	0,988	0,910091
TEST PRZYBLIŻONEJ ENTROPII	0,993	0,282626	0,992	0,944274
TEST BŁĄDZENIA LOSOWEGO**	0,982	0,787975	0,988	0,921704
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,987	0,075645	0,982	0,167034
TEST SERII*	0,992	0,036352	0,990	0,292519
TEST ZŁOŻONOŚCI LINIOWEJ	0,992	0,735908	0,988	0,478839

GWIĄZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.32 Wyniki testów statystycznych NIST 800-22 dla układu ECP3

RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,985	0,014348	0,992	0,123755
BLOKOWY TEST CZĘSTOŚCI	0,990	0,870856	0,991	0,759756
TEST SKUMULOWANYCH SUM*	0,984	0,005319	0,991	0,463512
TEST CIĄGÓW	0,654	0,000000	0,993	0,104993
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,990	0,113372	0,984	0,457825
TEST STOPNIA MACIERZY BINARNEJ	0,993	0,266235	0,989	0,380407
TEST SPEKTRALNY DFT	0,995	0,094285	0,990	0,954015
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,979	0,000241	0,982	0,867692
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,986	0,550347	0,991	0,444691
TEST UNIwersalny MAURERA	0,986	0,823725	0,990	0,336111
TEST PRZYBLIŻONEJ ENTROPII	0,978	0,032489	0,995	0,153763
TEST BŁĄDZENIA LOSOWEGO**	0,966	0,039244	0,987	0,092723
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,980	0,950449	0,985	0,437274
TEST SERII*	0,984	0,630872	0,992	0,378705
TEST ZŁOŻONOŚCI LINIOWEJ	0,991	0,612147	0,997	0,674543

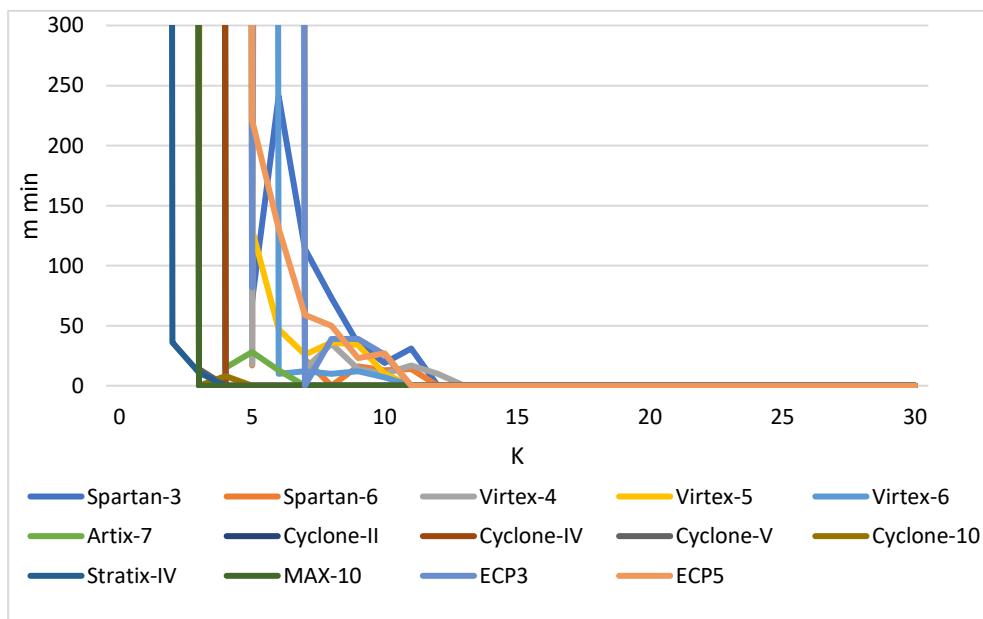
GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.33 Wyniki testów statystycznych NIST 800-22 dla układu ECP5

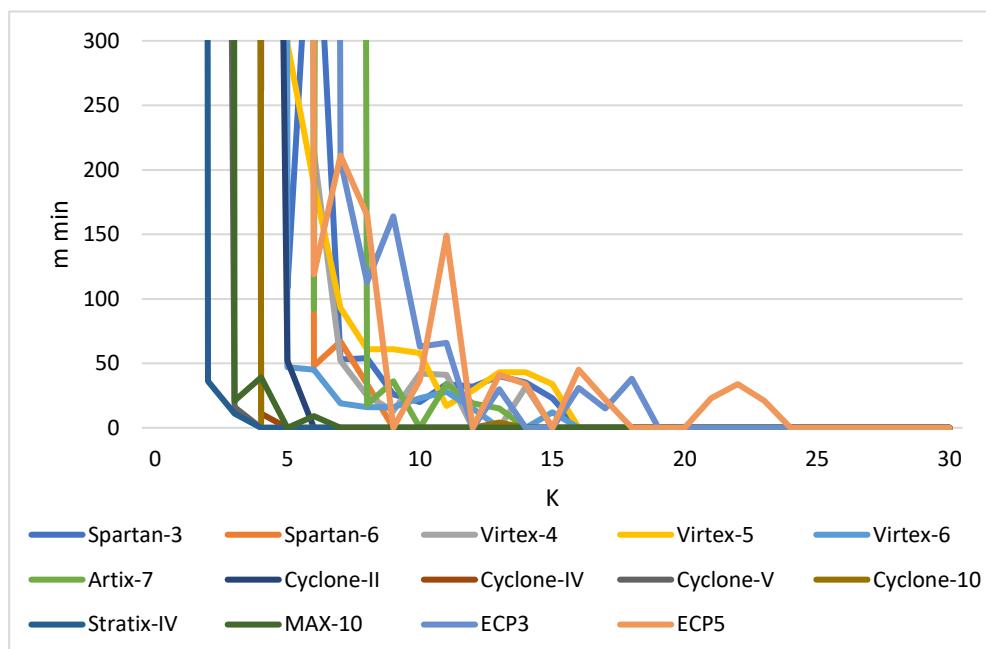
RODZAJ TESTU	K = 10		K = 15	
	<i>R_B</i>	<i>P_T</i>	<i>R_B</i>	<i>P_T</i>
TEST CZĘSTOŚCI	0,966	0,000000	0,988	0,108150
BLOKOWY TEST CZĘSTOŚCI	0,988	0,185555	0,989	0,328297
TEST SKUMULOWANYCH SUM*	0,965	0,000000	0,987	0,375313
TEST CIĄGÓW	0,772	0,000000	0,989	0,000477
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,989	0,098330	0,992	0,174728
TEST STOPNIA MACIERZY BINARNEJ	0,990	0,053627	0,983	0,719747
TEST SPEKTRALNY DFT	0,988	0,999958	0,986	0,856359
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,976	0,002203	0,981	0,224821
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,987	0,542228	0,991	0,253122
TEST UNIwersalny MAURERA	0,985	0,921624	0,983	0,358641
TEST PRZYBLIŻONEJ ENTROPII	0,990	0,004207	0,989	0,373625
TEST BŁĄDZENIA LOSOWEGO**	0,983	0,365684	0,985	0,266680
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,987	0,263229	0,983	0,164120
TEST SERII*	0,992	0,415422	0,983	0,140453
TEST ZŁOŻONOŚCI LINIOWEJ	0,993	0,450297	0,993	0,763677

GWIADKA * WSKAŻUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO $R_B = 0,9777$. W TABELI POKAZANO WYNIK NAJGORSZY. POGRUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

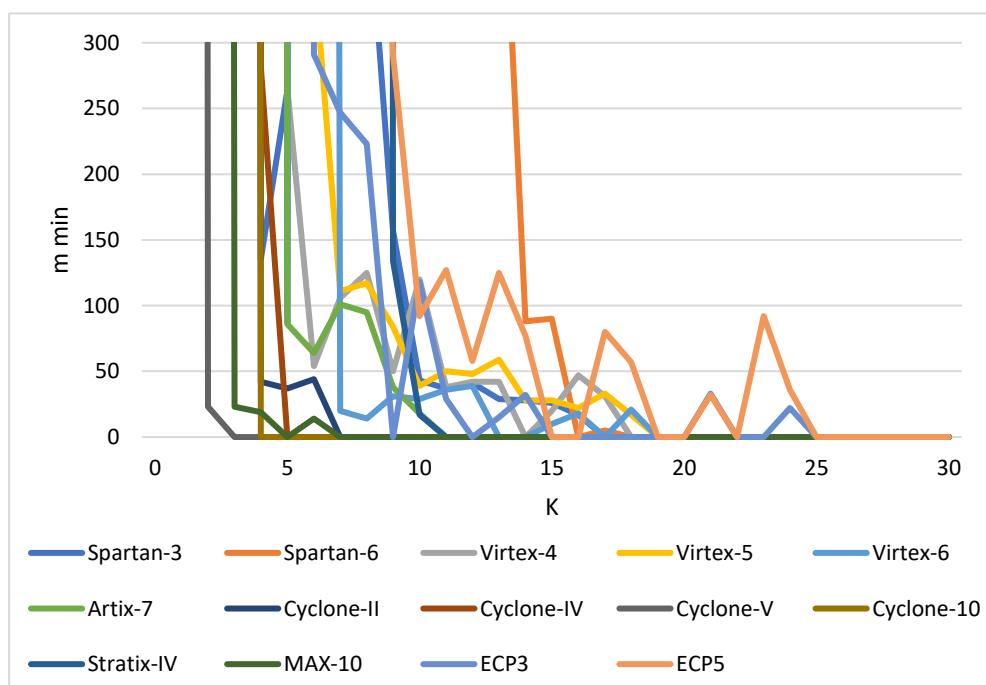
Ponieważ testy statystyczne mogą być spełnione zarówno przez ciągi pseudolosowe, jak i przez ciągi powstałe w wyniku zjawisk niedeterministycznych, konieczne jest zbadanie czy wytwarzany przez generator ciąg powstaje w wyniku zjawisk niedeterministycznych, czy też w wyniku procesów deterministycznych. Aby ocenić ilość losowości w generowanym ciągu zastosowano metodę restartów. Metoda restartów uzupełnia test restartów zaproponowany przez NIST w rekomendacji 800-90B. Test restartów według NIST wyznacza najczęściej powtarzającą się wartość w kolumnach i wierszach. Na jej podstawie wyznacza minimalną entropię dla rzędów i kolumn, która jest porównywana z minimalną entropią wyznaczoną dla pojedynczego ciągu o długości minimum 1000000 bitów. Jeżeli minimalne entropie znacząco się różnią, wtedy test restartów nie jest zdany. W metodzie restartów podczas pojedynczego restartu wytwarzano ciąg o długości $M = 20000$ bitów. Wykonano $N = 2048$ ponownych uruchomień generatora z tymi samymi warunkami początkowymi. Następnie wykonano testy zgodności χ^2 . Jeżeli dany bit w ciągu został wytworzony na skutek działania procesów niedeterministycznych to test χ^2 został zdany. Potem wyznaczono minimalny numer bitu m dla którego test χ^2 został zdany i oznaczono jako m_{min} . Numer ten oznacza, że każdy co m_{min} bit jest losowy. Porównanie wyników dla różnych układów FPGA wytwarzających losowe bity przy użyciu łączonego generatora TRNG z detektorem fazy pokazano na rysunkach 5.38 - 5.40.



Rysunek 5.38 Wartość m_{min} w funkcji K dla częstotliwości próbkowania $f_L = 100 \text{ MHz}$



Rysunek 5.39 Wartość m_{min} w funkcji K dla częstotliwości próbkowania $f_L = 200 \text{ MHz}$

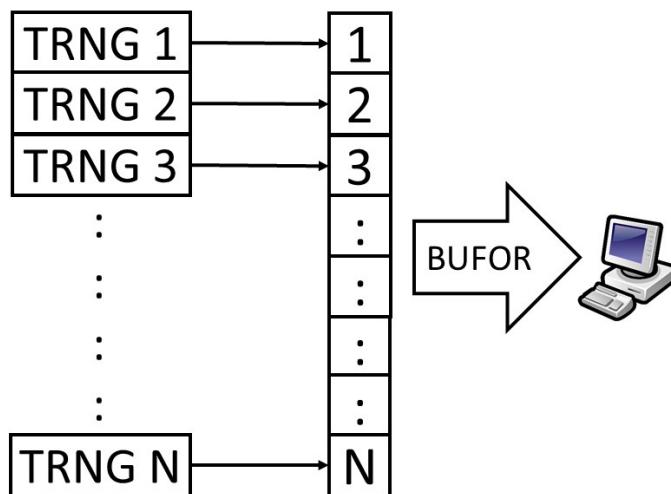


Rysunek 5.40 Wartość m_{min} w funkcji K dla częstotliwości próbkowania $f_L = 300 \text{ MHz}$

Dla wszystkich częstotliwości próbkowania wartość m_{min} zmniejsza się nieregularnie wraz ze wzrostem K liczby oscylatorów pierścieniowych. Dla dużych K wartości m_{min} stają się równe jeden. Oznacza to, że każdy bit ciągu jest wytwarzany w wyniku procesów niedeterministycznych. Wartość K dla której m_{min} osiąga jeden, rośnie wraz ze wzrostem częstotliwości próbkowania f_L i wynosi około 13 dla $f_L = 100$ MHz, 20 dla $f_L = 200$ MHz i 25 dla $f_L = 300$ MHz.

5.6. WYTWARZANIE CIĄGÓW LOSOWYCH Z DUŻĄ SZYBKOŚCIĄ

Szybki rozwój usług teleinformatycznych wymaga aby coraz więcej informacji było przesyłanych ze wzrastającą szybkością. Jednocześnie oczekuje się transmisji bezpiecznej. Dlatego generatory TRNG muszą być szybkie i muszą wytwarzać bezpieczne ciągi liczb losowych. W oparciu ołączony generator TRNG z detektorem fazy (rys. 5.24) stworzono szybki generator losowych ciągów przedstawiony na rysunku 5.41. Nowy generator to



Rysunek 5.41 Równoległe połączenie N generatorów TRNG

równolegle połączenie generatora z rysunku 5.24 w taki sposób, że podczas jednego cyklu zegarowego wytwarzanych jest N bitów ciągu. Każdy bit wytwarza niezależny generator łączony. W rezultacie otrzymuje się ciąg losowych bitów o N razy większej przepływności. Ważne aby każdy z użytych do zwielokrotnienia generatorów TRNG z rysunku 5.24 wytwarzał losowe bity już od pierwszego cyklu zegarowego, tj. aby m_{min} było równe jeden. W przeciwnym wypadku zwielokrotnieniu ulegnie również liczba nielosowych bitów, czyli m_{min} zwiększy się N razy.

Na podstawie przeprowadzonych badań powstał skalowalny moduł równoległego generatora TRNG gotowy do użycia w dowolnym projekcie. Możliwe jest określenie liczby K źródłowych oscylatorów pierścieniowych w łączonym generatorze (domyślnie 16), jak

również liczby N (domyślnie 1), która określa ile bitów jest wytwarzanych podczas jednego cyklu zegarowego. Przepływność strumienia losowych bitów wyznacza się w następujący sposób:

$$R_b = N \cdot f_L. \quad (5.29)$$

Liczba N jest praktycznie dowolna. Ograniczają ją tylko dostępne zasoby układu FPGA. Kod opisujący moduł generatora przedstawia rysunek 5.42.

```
module pa_trng #(parameter K = 16, parameter N = 1)
    (input wire F1,
     input wire start,
     output wire [N-1:0] out);

    wire [N-1:0] rout;

    genvar i;
    generate
        for(i=0; i<N; i=i+1)
        begin
            df_trng #(K=30) trng (.f1(F1), .start(start), .Out(rout[i]));
        end
    endgenerate
    assign out = rout;
endmodule
```

Rysunek 5.42 Kod modułu równoległego generatora TRNG

Generator został zaimplementowany w układzie Virtex-5 XC5VLX50T dla $N = 8$ i $N = 128$. Biorąc pod uwagę wyniki metody restartów i częstotliwość próbkowania $f_L = 300$ MHz ustalono liczbę oscylatorów pierścieniowych w pojedynczym generatorze TRNG na $K = 30$. Dla częstotliwości próbkowania $f_L = 300$ MHz i $N = 8$ szybkość wytwarzania losowych ciągów wynosi 2,6 Gb/s, gdy N zwiększymy do 128 szybkość generowanych bitów wynosi 38,4 Gb/s. Wykorzystanie zasobów i szybkość wytwarzania losowych ciągów przedstawia tabela 5.34. Generator przetestowano testami statystycznymi według rekomendacji NIST 800-90B, pakietem testów NIST 800-22 i metodą restartów. Dla testów z rekomendacji NIST 800-90B wytworzono ciąg 1000000 bitów i 1000 ciągów po 1000 bitów dla testu restartów. Dla pakietu testów statystycznych NIST 800-22 przygotowano dane składające się z 1000 ciągów po 1000000 bitów każdy. Dla metody restartów wytworzono 2048 ciągów po 20000 bitów. Wyniki testów statystycznych przedstawiono w tabelach 5.35 - 5.36. Wartość m_{min} dla każdego przypadku wynosiła jeden. We wszystkich przypadkach wszystkie testy statystyczne są spełnione. Potwierdza to, że generator wytwarza ciągi liczb rzeczywiście losowych.

ŁĄCZONY GENERATOR TRNG Z DETEKTOREM FAZY

Tabela 5.34 - Wykorzystanie zasobów FPGA i maksymalna szybkość wytwarzania losowych bitów

RODZAJ ELEMENTU	N = 1	N = 8	N = 32	N = 64	N = 128
SLICES	84	747	2189	3706	6275
SLICE LUTs	157	1207	4807	9607	19207
SLICE REGISTERS	96	726	2886	5766	11526
SZYBKOŚĆ BITOWA	300 MB/S	2,4 GB/S	9,6 GB/S	19,2 GB/S	38,4 GB/S

Tabela 5.35 Wyniki testów statystycznych NIST 800-90B dla szybkiego generatora TRNG

RODZAJ TESTU	N = 8			N = 128		
	C _{i,0}	C _{i,1}	IID	C _{i,0}	C _{i,1}	IID
TEST BŁĄDZENIA	6	0	TAK	6	0	TAK
TEST LICZBY CIĄGÓW KIERUNKOWYCH	12	0	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGÓW KIERUNKOWYCH	4	4	TAK	2	4	TAK
TEST LICZBY WZROSTÓW I SPADKÓW	6	0	TAK	37	0	TAK
TEST LICZBY CIĄGÓW NA PODSTAWIE MEDIANY	43	1	TAK	6	0	TAK
TEST DŁUGOŚCI CIĄGU NA PODSTAWIE MEDIANY	6	0	TAK	6	0	TAK
TEST ŚREDNIEJ KOLIZJI	6	0	TAK	6	0	TAK
TEST MAKSYMALNEJ WARTOŚCI KOLIZJI	11	3	TAK	18	2	TAK
TEST OKRESOWOŚCI*	7	0	TAK	8	0	TAK
TEST KOWARIANCJI*	83	0	TAK	68	0	TAK
TEST KOMPRESJI	6	0	TAK	6	0	TAK
TEST DOBROCI DOPASOWANIA	ZDANY			ZDANY		
TEST NIEZALEŻNOŚCI	ZDANY			ZDANY		
TEST NAJDŁUŻSZEGO POWTÓRZONEGO PODCIĄGU	ZDANY			ZDANY		
TEST RESTARTÓW	ZDANY			ZDANY		
MINIMALNA ENTROPIA	0,995040			0,994900		

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. POGΡUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Tabela 5.36 Wyniki testów statystycznych NIST 800-22 dla szybkiego generatora TRNG

RODZAJ TESTU	N = 8		N = 128	
	R _B	P _T	R _B	P _T
TEST CZĘSTOŚCI	0,991	0,059734	0,992	0,365253
BLOKOWY TEST CZĘSTOŚCI	0,994	0,977480	0,994	0,741918
TEST SKUMULOWANYCH SUM*	0,991	0,024855	0,991	0,000361
TEST CIĄGÓW	0,997	0,328297	0,990	0,854708
TEST NA NAJDŁUŻSZY CIĄG JEDYNEK W BLOKU	0,994	0,183547	0,987	0,348869
TEST STOPNIA MACIERZY BINARNEJ	0,985	0,496351	0,991	0,821937
TEST SPEKTRALNY DFT	0,989	0,657933	0,983	0,825505
TEST DOPASOWANIA NIENAKŁADAJĄCYCH SIĘ WZORCÓW*	0,984	0,788728	0,983	0,084037
TEST DOPASOWANIA NAKŁADAJĄCYCH SIĘ WZORCÓW	0,994	0,422638	0,992	0,910091
TEST UNIWERSALNY MAURERA	0,989	0,607993	0,989	0,448424
TEST PRZYBLIŻONEJ ENTROPII	0,987	0,996839	0,983	0,420827
TEST BŁĄDZENIA LOSOWEGO**	0,983	0,746463	0,979	0,404784
TEST WARIANCJI BŁĄDZENIA LOSOWEGO**	0,988	0,347837	0,982	0,396540
TEST SERII*	0,989	0,382115	0,989	0,467322
TEST ZŁOŻONOŚCI LINIOWEJ	0,988	0,147815	0,987	0,915317

GWIAZDKA * WSKAZUJE, ŻE DANY TEST SKŁADA SIĘ Z KILKU PODTESTÓW, A NAJGORSZY WYNIK JEST POKAZANY W TABELI. DLA TESTÓW OZNACZONYCH **, TEST SKŁADA SIĘ Z KILKU PODTESTÓW, MINIMALNA WARTOŚĆ PROPORCJI WYNOSI OKOŁO R_B = 0,9777. W TABELI POKAZANO WYNIK NAJGORSZY. POGΡUBIONA CZCIONKA POKAZUJE NIEZDANE TESTY.

Ostatnim etapem badań było porównanie zaproponowanego rozwiązania z propozycjami znymi z literatury. Rezultaty porównań zawiera tabela 5.37.

Tabela 5.37 Porównanie generatorów TRNG

POZYCJA	ŹRÓDŁO ENTROPII	IMPLEMENTACJA	POST-PROCESING	TESTY STATYSTYCZNE	PRZEPŁYWNOŚĆ
[8]	JITTER	FPGA	NIE	BRAK DANYCH	2,5 MB/S
[12]	SZUM	ASIC	TAK	FIPS	75 KB/S
[16]	SZUM	BRAK DANYCH	NIE	BRAK DANYCH	$1,155 \times$ PASMO SZUMU
[17]	SZUM, JITTER, CHAOS	ASIC	NIE	DIEHARD	1,4 MB/S
[18]	SZUM	ASIC	NIE	FFT	1 MB/S
[21]	SZUM, JITTER	ASIC	TAK	NIST	10 MB/S
[22]	SZUM	BRAK DANYCH	NIE	AUTOKORELACJA	BRAK DANYCH
[23]	SZUM	ELEMENTY DYSKRETNE	NIE	DIEHARD, NIST	1,2 MB/S
[24]	SZUM	ASIC	TAK	FIPS	40 MB/S
[26]	SZUM	ELEMENTY DYSKRETNE	TAK	FIPS, DIEHARD, NIST	8 MB/S
[27]	METASTABILNOŚĆ	FPGA	NIE	NIST	25 B/S
[28]	METASTABILNOŚĆ	ASIC	NIE	NIST	500 B/S
[29]	METASTABILNOŚĆ	FPGA	TAK	NIST	20 MB/S
[30]	METASTABILNOŚĆ	FPGA	TAK	NIST	BRAK DANYCH
[31]	METASTABILNOŚĆ	FPGA	TAK	AIS-31, NIST	20 MB/S
[32]	METASTABILNOŚĆ	ASIC	NIE	AIS-31	BRAK DANYCH
[33]	METASTABILNOŚĆ	FPGA	TAK	NIST	BRAK DANYCH
[34]	METASTABILNOŚĆ	FPGA	TAK	NIST	16 MB/S
[35]	METASTABILNOŚĆ	ASIC	TAK	BRAK DANYCH	BRAK DANYCH
[36]	METASTABILNOŚĆ	ASIC	NIE	KNUTH TEST [117]	BRAK DANYCH
[37]	METASTABILNOŚĆ	ASIC	NIE	KNUTH TEST [117]	10 MB/S
[38]	METASTABILNOŚĆ	FPGA	TAK	NIST	12,5 MB/S
[39]	METASTABILNOŚĆ	ASIC	NIE	DIEHARD	BRAK DANYCH
[40]	METASTABILNOŚĆ	FPGA	TAK	AIS-31, FIPS	35 MB/S
[41]	METASTABILNOŚĆ	FPGA	NIE	FIPS, NIST	250 KB/S
[42]	METASTABILNOŚĆ	ASIC	NIE	NIST	4 GB/S
[43]	METASTABILNOŚĆ	ASIC	TAK	BRAK DANYCH	OKOŁO 3 GB/S
[45]	METASTABILNOŚĆ	FPGA	NIE	DIEHARD, NIST	5 MB/S
[46]	JITTER	ELEMENTY DYSKRETNE	NIE	BRAK DANYCH	26,8 B/S
[47]	JITTER	FPGA	NIE	NIST	69 KB/S
[48]	JITTER	FPGA	NIE	NIST	1 MB/S
[49]	JITTER	ASIC	TAK	NIST	100 KB/S
[51]	JITTER	FPGA	NIE	NIST	600 KB/S
[52]	JITTER	FPGA	NIE	FIPS, NIST	2 MB/S
[53]	JITTER	ASIC	TAK	DIEGARD, FIPS, CRYPT-X	BRAK DANYCH
[56]	JITTER	FPGA	TAK	DIEHARD	12,5 MB/S
[57]	JITTER	FPGA	TAK	DIEHARD	13,8 MB/S
[58]	JITTER	FPGA	TAK	NIST, DIEHARD	80 MB/S
[59]	JITTER	FPGA	TAK	NIST, DIEHARD	40 MB/S
[61]	JITTER	FPGA	NIE	NIST, DIEHARD	100 MB/S
[62]	JITTER	FPGA	NIE	NIST, DIEHARD	300 MB/S
[66]	JITTER	FPGA	NIE	NIST	250 MB/S
[68]	JITTER	FPGA	NIE	NIST	250 MB/S
[69]	JITTER	FPGA	NIE	NIST	250 MB/S
[70]	JITTER	FPGA	TAK	NIST	36 MB/S

[73]	JITTER	FPGA	TAK	FIPS, NIST	16 MB/S
[74]	JITTER	ASIC	NIE	FIPS	BRAK DANYCH
[76]	JITTER	FPGA	NIE	NIST	BRAK DANYCH
[78]	JITTER	ASIC	NIE	NIST	BRAK DANYCH
[79]	JITTER	FPGA/ASIC	TAK	NIST	50 MB/S / 1 GB/S
[81]	EFEKT KWANTOWY	ELEMENTY DYSKRETNE	BRAK DANYCH	NIST	5 MB/S
[84]	EFEKT KWANTOWY	ELEMENTY DYSKRETNE	TAK	DIEHARD	280 GB/S
[85]	JITTER	FPGA	TAK	NIST	95,37 MB/S
[86]	METASTABILNOŚĆ I CHAOS	FPGA	NIE	NIST	5 MB/S
[87]	METASTABILNOŚĆ I CHAOS	FPGA	NIE	NIST	1 MB/S
[88]	JITTER I CHAOS	FPGA	TAK	NIST	125 MB/S
[89]	JITTER I CHAOS	ELEMENTY DYSKRETNE	NIE	NIST	BRAK DANYCH
[90]	JITTER I CHAOS	FPGA	TAK	NIST	12,8 GB/S
[92]	CHAOS	ASIC	NIE	BRAK DANYCH	100 KB/S
[93]	CHAOS	PSoC	TAK	FIPS	60 KB/S
[94]	CHAOS	ASIC	NIE	BRAK DANYCH	500 KB/S
[95]	CHAOS	ASIC	NIE	BRAK DANYCH	500 KB/S
[96]	CHAOS	BRAK DANYCH	TAK	FIPS, NIST	10 MB/S
[98]	CHAOS	ELEMENTY DYSKRETNE	NIE	BRAK DANYCH	BRAK DANYCH
[100]	CHAOS	ELEMENTY DYSKRETNE	NIE	BRAK DANYCH	BRAK DANYCH
PROPOZYCJA	JITTER	FPGA	NIE	NIST	38,4 GB/S

Generatory oparte o zjawisko szumu nie są implementowane wewnątrz struktur reprogramowalnych FPGA. Posiadają również niską szybkość wytwarzania losowych bitów. Również generatory wykorzystujące zjawiska kwantowe nie mogą być implementowane wewnątrz układów FPGA. W układach FPGA można implementować układy chaotyczne i metastabilne jednak nie jest to zadanie proste, a szybkości wytwarzanych ciągów zazwyczaj nie są zadowalające. Także jakość produkowanych bitów przez te generatory nie zawsze jest satysfakcyjną. Najbardziej obiecującą grupą generatorów TRNG są te, które jako źródło entropii wykorzystują jitter obecny w sygnale oscylatorów pierścieniowych. Implementacja takiego generatora w strukturach FPGA jest łatwa, a uzyskiwane przepływności zadowalające. Ciągi wytwarzane przez takie generatory spełniają również wszystkie znane testy statystyczne. Generator proponowany w rozprawie również wykorzystuje zjawisko szybkozmennych fluktuacji fazowych. Dzięki wykorzystaniu detektora fazy zwiększo ilość pozyskiwanej losowości z oscylatorów pierścieniowych. Umożliwiło to budowę szybkiego generatora TRNG, którego szybkość bitowa jest 3 razy większa niż najszybszego generatora implementowanego w FPGA znanego z literatury i może być nadal zwiększana poprzez zastosowanie układów reprogramowalnych o większych zasobach.

6. PODSUMOWANIE

Przedstawione w rozprawie wyniki badań dowodzą, że układy FPGA można wykorzystać do budowy szybkich generatorów ciągów liczb rzeczywiście losowych, z możliwością wymiany zasobów układów reprogramowalnych na szybkość wytwarzania ciągu bitowego z jednoczesnym spełnieniem wszystkich testów statystycznych zaproponowanych przez NIST. Potwierdzona zatem została teza rozprawy.

W ramach rozprawy powstał szczegółowy przegląd metod i układów sprzętowych służących do wytwarzania ciągów liczb rzeczywiście losowych. W pracy zwrócono szczególną uwagę na te metody, które można zaimplementować w cyfrowych układach reprogramowalnych razem z systemem kryptograficznym. Spośród nich wybrano metodę wykorzystującą szybkozmienne fluktuacje fazy sygnałów oscylatorów pierścieniowych jako najbardziej obiecujący sposób wytwarzania ciągów losowych o bardzo dobrej jakości i dużej szybkości wytwarzania bitów.

Szczegółowo opisano testy statystyczne i proces testowania generatorów losowych. Skupiono się na testach statystycznych zaproponowanych przez NIST. Uzupełniono je o metodę restartów sprawdzającą, czy dany bit powstał w wyniku procesów niedeterministycznych. Dzięki tej metodzie można wyznaczyć liczbę bitów w ciągu, które są rzeczywiście losowe.

Przeprowadzono pomiary i analizę jittera w oscylatorach pierścieniowych implementowanych wewnątrz układu FPGA z różnymi elementami opóźniającymi. Wykazano, że rozkład szybkozmennych fluktuacji fazowych w oscylatorach pierścieniowych w układach FPGA posiada rozkład normalny w związku z czym do powstania jittera przyczyniają się niedeterministyczne czynniki obecne w układach elektronicznych.

Przedstawiono koncepcję łączonego generatora TRNG, gdzie ciągi losowe z poszczególnych generatorów źródłowych są łączone za pomocą sumy modulo 2. Opisano w jaki sposób należy projektować łączony generator ciągów losowych, aby zminimalizować skutki ataku typu wstrzykiwanie częstotliwości.

Zaproponowano wykorzystanie detektora fazy jako układu pozyskującego losowość z oscylatorów pierścieniowych. Przeprowadzono analizę tego rozwiązania i porównano z metodami z bezpośrednim próbkowaniem sygnału wyjściowego oscylatorów

pierścieniowych. Badania proponowanej metody wykonano po jej implementacji w układach FPGA różnych typów, pochodzących od różnych producentów. Wyznaczono minimalną liczbę oscylatorów pierścieniowych niezbędną do spełnienia wszystkich testów statystycznych dla danej przepływności i zapewniającą wartość m_{min} równą jeden, czyli taką która pozwala na potraktowanie każdego wytworzzonego bitu jako bitu rzeczywiście losowego. Następnie zaproponowano i zaprojektowano skalowalny moduł generatora ciągów bitów rzeczywiście losowych, w którym szybkość wytwarzania ciągu może być wymieniana na zasoby układu FPGA. W wyniku badań przeprowadzonych dla rzeczywistego układu otrzymano maksymalną przepływność równą 38,4 Gb/s. Proponowany generator TRNG spełnia wszystkie testy statystyczne zaproponowane przez NIST, a wyniki metody restartów potwierdzają, że wszystkie bity wytwarzanego ciągu są rzeczywiście losowe. Przepływność 38,4 Gbit/s można jeszcze zwiększyć korzystając z idei zaproponowanej w pracy ale wymaga to użycia układów FPGA o zasobach większych od rozważanych w rozprawie. Obecnie wydaje się to bezcelowe, ze względu na brak komercyjnych systemów kryptograficznych pracującymi z szybkościami większymi niż 40 Gbit/s.

Podstawowym kierunkiem dalszych badań powinno być sprawdzenie jakości pracy proponowanego generatora w warunkach symulowanych ataków, co wymaga dostępu do specjalistycznego laboratorium. W rozprawie nie analizowano metod monitorowania jakości pracy generatora w czasie rzeczywistym, które w dużej mierze są takie same jak dla systemów kryptograficznych korzystających z analogowych źródeł ciągów losowych, opisanych na przykład w pracy [26].

LITERATURA

- [1] A. Vassilev, T. A. Hall, „The Importance of Entropy to Information Security,” *Computer*, vol. 47, no. 2, pp. 78 - 81, February 2014.
- [2] M. Dietzfelbinger, Primality Testing in Polynomial Time, Springer-Verlag Berlin Heidelberg, 2004.
- [3] Ç. K. Koç, „About Cryptographic Engineering,” w *Cryptographic Engineering*, Springer, 2009, pp. 1-4.
- [4] O. Goldreich, Modern Cryptography, Probabilistic Proofs and Pseudorandomness, Springer-Verlag Berlin Heidelberg, 1999.
- [5] J. M. Alfred, C. O. Paul, A. V. Scott, Kryptografia stosowana, Wydawnictwo WNT, 2009.
- [6] D. Eastlake, S. Crocker, J. Schiller, „Randomness Recommendations for Security,” Internet Request for Comments 1750, RFC1750 , 1994.
- [7] V. Fischer, A. Aubert, F. Bernard, B. Valtchanov, J.-L. Danger, N. Bochard, „True random number generators in configurable logic devices,” Project ANR–ICTeR, 2009.
- [8] B. Sunar, W. J. Martin, D. R. Stinson, „A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks,” *IEEE Transactions on Computers*, pp. 109 - 119, January 2007.
- [9] B. Badrignans, J. L. Danger, V. Fischer, G. Gogniat, L. Torres, Security Trends for FPGAs, Dordrecht: Springer, 2011.
- [10] S.-H. Kwok, Y.-L. Ee, G. Chew, K. Zheng, K. Khoo, C.-H. Tan, „A Comparison of Post-Processing Techniques for Biased Random Number Generators,” *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, 2011.
- [11] J. V. Neumann, „Various techniques used in connection with random digits,” *Monte Carlo Method*, Washington DC., US Government Printing Office, 1951, pp. 36-38.
- [12] B. Jun, P. Kocher, „The Intel random number generator,” Cryptography Research Inc. White Paper, San Francisco, 1999.
- [13] S. Łoza, Ł. Matuszewski, M. Jessa, „A random number generator using ring oscillators and SHA-256 as post-processing,” *International Journal of Electronics and Telecommunications*, vol. 61, no. 2, pp. 199-204, 2015.
- [14] C. S. Petrie, J. A. Connelly, „The sampling of noise for random number generation,” *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, Orlando, FL, USA , 1999.
- [15] M.-S. Gupta, „Applications of Electrical Noise,” *Proceedings of IEEE*, vol. 63, no. 7, pp. 996-1010, 1975.
- [16] H. F. Murry, „A General Approach for Generating Natural Random Variables,” *IEEE Transactions on Computers*, pp. 1210-1213, December 1970.
- [17] C. S. Petrie, J. Connelly, „A Noise-Based IC Random Number Generator for Applications in Cryptography,” *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 615-621, May 2000.
- [18] T. W. Holman, A. J. Connelly, A. B. Dowlatabadi, „An Integrated Analog/Digital Random Noise Source,” *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 44, no. 6, p. 521528, June 1997.

- [19] W. Schottky, „Spontaneous current fluctuations in various conductors,” *Ann. Physik*, no. 57, pp. 541-567, 1918.
- [20] „Secure Hash Standard (SHS),” Federal Inf. Process. Stds. (NIST FIPS) - 180-4, 2015.
- [21] M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, M. Varanouovo, „A high-speed oscillator-based truly random number source for cryptographic applications on a smart card IC,” *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 403 - 409, April 2003.
- [22] V. Bagini, M. Bucci, „A Design of Reliable True Random Number Generator for Cryptographic Applications,” *Cryptographic Hardware and Embedded Systems*, C. K. Koc i C. Paar, Springer-Verlag Berlin Heidelberg, 1999, pp. 204-218.
- [23] M. Stipčević, „Fast nondeterministic random bit generator based on weakly correlated physical events,” *Review of Scientific Instruments*, vol. 75, no. 11, 2004.
- [24] M. Bucci, L. Germani, R. Luzzi, P. Tommasino, A. Trifiletti, M. Varanouovo, „A High-Speed IC Random-Number Source for SmartCard Microcontrollers,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 11, pp. 1373 - 1380, November 2003.
- [25] W. Killmann, W. Schindler, „A Design for a Physical RNG with Robust Entropy Estimators,” *Cryptographic Hardware and Embedded Systems – CHES 2008*, Springer, Berlin, Heidelberg, 2008, pp. 146-163.
- [26] M. Leśniewicz, Sprzętowa generacja losowych ciągów binarnych, Zegrze: Wojskowa Akademia Techniczna, 2009.
- [27] P. Kubczak, Ł. Matuszewski, M. Jessa, „Generator ciągów losowych wykorzystujący stany metastabilne zaimplementowany w układzie FPGA firmy Xilinx,” *Pomiary Automatyka Kontrola*, pp. 450-452, 7 2014.
- [28] J. Holleman, S. Bridges, B. P. Otis, C. Diorio, „A 3 μ W CMOS True Random Number Generator With Adaptive Floating-Gate Offset Cancellation,” *IEEE Journal of Solid-State Circuits*, vol. 43, no. 5, pp. 1324 - 1336, May 2008.
- [29] J.-L. Danger, S. Guilley, P. Hoogvorst, „Fast True Random Generator in FPGAs,” *IEEE Northeast Workshop on Circuits and Systems*, Montreal, Que, Canada, 2007.
- [30] J.L.Danger, S.Guilley, P.Hoogvorst, „High speed true random number generator based on open loop structures in FPGAs,” *Microelectronics Journal*, vol. 40, no. 11, pp. 1650-1656, November 2009.
- [31] F. Lozach, M. Ben-Romdhane, T. Graba, J.-L. Danger, „FPGA Design of an Open-Loop True Random Number Generator,” *Euromicro Conference on Digital System Design*, Santander, Spain, 2013.
- [32] M. Ben-Romdhane, T. Graba, J.-L. Danger, Y. Mathieu, „Design methodology of an ASIC TRNG based on an open-loop delay chain,” w *IEEE 11th International New Circuits and Systems Conference (NEWCAS)*, Paris, France, 2013.
- [33] D. Lee, H. Seo, „Metastability-based Feedback Method for Enhancing FPGA-based TRNG,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 3, pp. 235-248, 2014.
- [34] M. Majzoobi, F. Koushanfar, S. Devadas, „FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control,” *Cryptographic Hardware and Embedded Systems – CHES 2011*, vol. 6917, P. B., T. T., Red., Berlin, Heidelberg, Springer, 2011.
- [35] M. Bucci, R. Luzzi, „A Testable Random Bit Generator based on a High Resolution Phase Noise Detection,” *IEEE Design and Diagnostics of Electronic Circuits and Systems*, Kraków, Poland, 2007.
- [36] M. J. Bellido, A. Acosta i M. Valencia, „Simple binary random number generator,” *Electronics Letters*, vol. 28, no. 7, pp. 617 - 618, 26 March 1992.

- [37] M. Bellido, A. Acosta, M. Valencia, A. Barriga, J. Huertas, „A simple binary random number generator: new approaches for CMOS VLSI,” *35th Midwest Symposium on Circuits and Systems*, Washington, DC, USA, 1992.
- [38] H. Hata, S. Ichikawa, „FPGA Implementation of Metastability-Based True Random Number Generator,” *IEICE Transactions on Information and Systems*, pp. 426-436, February 2012.
- [39] M. Epstein, L. Hars, R. Krasinski, M. Rosner i H. Zheng, „Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts,” *Cryptographic Hardware and Embedded Systems - CHES 2003*, vol. 2779, W. C.D., K. Ç.K. i P. C., Red., Springer, Berlin, Heidelberg, 2003, pp. 152-165.
- [40] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, B. Karpinskyy, „Fast Digital TRNG Based on Metastable Ring Oscillator,” *Cryptographic Hardware and Embedded Systems – CHES 2008. Lecture Notes in Computer Science*, O. E. i R. P., Red., Berlin, Heidelberg, Springer, 2008, pp. 164-180.
- [41] M. Varchola, M. Drutarovsky, „New High Entropy Element for FPGA Based True Random Number Generators,” *Cryptographic Hardware and Embedded Systems, CHES 2010. CHES 2010. Lecture Notes in Computer Science*, Berlin, Heidelberg, Springer, 2010, pp. 351-365.
- [42] S. Srinivasan, S. Mathew, V. Erraguntla, R. Krishnamurthy, „A 4Gbps 0.57pJ/bit Process-Voltage-Temperature Variation Tolerant All-Digital True Random Number Generator in 45nm CMOS,” *22nd International Conference on VLSI Design*, New Delhi, India, 2009.
- [43] G. Taylor, G. Cox, „Behind Intel’s New Random-Number Generator,” *IEEE Spectrum*, 2011.
- [44] P. Wieczorek, „Dual-metastability fpga-based true random number generator,” *Electronics Letters*, vol. 49, no. 12, pp. 744 - 745, 6 June 2013.
- [45] P. Z. Wieczorek, K. Gołofit, „Generator, Dual-Metastability Time-Competitive True Random Number,” *IEEE Transactions on Circuits and Systems I*, vol. 61, no. 1, pp. 134 - 145, January 2014.
- [46] R. C. Fairfield, R. L. Mortenson, K. B. Coulthart, „An LSI Random Number Generator (RNG),” *Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science*, vol. 196, 1984.
- [47] V. Fischer, M. Drutarovský, „True Random Number Generator Embedded in Reconfigurable Hardware,” *Cryptographic Hardware and Embedded Systems - CHES 2002. Lecture Notes in Computer Science*, vol. 2523, 2002.
- [48] V. Fischer, M. Drutarovský, M. Šimka, N. Bochard, „High Performance True Random Number Generator in Altera Stratix FPLDs,” *Field Programmable Logic and Application. FPL 2004. Lecture Notes in Computer Science*, vol. 3203, 2004.
- [49] C. Liu, J. McNeill, „A digital-PLL-based true random number generator,” *Research in Microelectronics and Electronics, 2005 PhD*, Lausanne, Switzerland, 2005.
- [50] M. Simka, M. Drutarovsky, V. Fischer, J. Fayolle, „Model of a true random number generator aimed at cryptographic applications,” *International Symposium on Circuits and Systems*, Island of Kos, Greece, 2006.
- [51] P. Kohlbrenner, K. Gaj, „An embedded true random number generator for FPGAs,” *12th international symposium on Field programmable gate arrays, FPGA’04*, Monterey, California, USA, 2004.
- [52] B. Valtchanov, V. Fischer, A. Aubert, „Enhanced TRNG based on the coherent sampling,” *3rd International Conference on Signals, Circuits and Systems (SCS)*, Medenine, Tunisia, 2009.
- [53] T. E. Tkacik, „A Hardware Random Number Generator,” *Cryptographic Hardware and Embedded Systems - CHES 2002. Lecture Notes in Computer Science*, vol. 2523, pp. 450 - 453, 2002.
- [54] P. Hortensius, R. McLeod, H. Card, „Parallel random number generation for VLSI systems using cellular automata,” *IEEE Transactions on Computers*, vol. 38, no. 10, pp. 1466 - 1473, Oct. 1989.

- [55] M. Dichtl, „How to Predict the Output of a Hardware Random Number Generator,” *Cryptographic Hardware and Embedded Systems - CHES 2003. Lecture Notes in Computer Science*, pp. 181-188, 2003.
- [56] J. Golic, „New Methods for Digital Generation and Postprocessing of Random Data,” *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1217 - 1229, Oct. 2006.
- [57] M. Dichtl, J. D. Golić, „High-Speed True Random Number Generation with Logic Gates Only,” *Cryptographic Hardware and Embedded Systems - CHES 2007. Lecture Notes in Computer Science*, vol. 4727, pp. 45 - 62, 2007.
- [58] S.-K. Yoo, B. Sunar, D. Karakoyunlu, B. Birand, „Improving the Robustness of Ring Oscillator TRNGs,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 2, May 2010.
- [59] D. Schellekens, B. Preneel, I. Verbauwhede, „FPGA Vendor Agnostic True Random Number Generator,” *International Conference on Field Programmable Logic and Applications*, Madrid, Spain, 2006.
- [60] P. Kubczak, M. J. S. Łukasz Matuszewski, „Digital random bit generators implemented in FPGAs offered by various manufacturers,” *Measurement Automation Monitoring*, vol. 61, no. 7, pp. 293 - 295, Jul. 2015.
- [61] K. Wold, C. H. Tan, „Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings,” *International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, 2008.
- [62] K. Wold, S. Petrovic, „Optimizing Speed of a True Random Number Generator in FPGA by Spectral Analysis,” *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, South Korea, 2009.
- [63] N. Bocharad, F. Bernard, V. Fischer, „Observing the Randomness in RO-Based TRNG,” *International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, 2009.
- [64] M. Jessa, M. Jaworski, „Randomness of a combined TRNG based on the ring oscillator sampling method,” *ICSES 2010 International Conference on Signals and Electronic Circuits*, Gliwice, Poland, 2010.
- [65] M. Jessa, M. Jaworski, „Generacja binarnych ciągów losowych w układzie Virtex-5,” *Pomiary Automatyka Kontrola*, vol. 56, no. 7, pp. 681 - 684, 2010.
- [66] M. Jessa, Ł. Matuszewski, M. Jaworski, „Losowość generatora TRNG zaimplementowanego w FPGA,” *Pomiary Automatyka Kontrola*, vol. 57, no. 8, pp. 880-882, 2011.
- [67] A. T. Markettos, S. M. Moore, „The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators,” *11th International Workshop on Cryptographic Hardware and Embedded Systems*, Lausanne, Switzerland, 2009.
- [68] M. Jessa, Ł. Matuszewski, „The use of delay lines in a ring-oscillator-based combined true random number generator,” *International Conference on Signals and Electronic Systems (ICSES)*, Wroclaw, Poland, 2012.
- [69] M. Jessa, Ł. Matuszewski, „Enhancing the Randomness of a Combined True Random Number Generator Based on the Ring Oscillator Sampling Method,” *International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, 2011.
- [70] S. Łoza, Ł. Matuszewski, „A true random number generator using ring oscillators and SHA-256 as post-processing,” *International Conference on Signals and Electronic Systems (ICSES)*, Poznan, Poland, 2014.
- [71] A. Winstanley, M. Greenstreet, „Temporal Properties of Self-Timed Rings,” *Correct Hardware Design and Verification Methods. CHARME 2001. Lecture Notes in Computer Science*, Berlin, Heidelberg, Springer, 2001, pp. 140 - 154.

- [72] A. Cherkaoui, V. Fischer, A. Aubert, L. Fesquet, „Comparison of Self-Timed Ring and Inverter Ring Oscillators as entropy sources in FPGAs,” *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2012.
- [73] A. Cherkaoui, V. Fischer, A. Aubert, L. Fesquet, „A Self-Timed Ring Based True Random Number Generator,” *IEEE 19th International Symposium on Asynchronous Circuits and Systems*, Santa Monica, CA, USA, 2013.
- [74] G. Gimenez, A. Cherkaoui, R. Frisch, L. Fesquet, „Self-timed Ring based True Random Number Generator: Threat model and countermeasures,” *IEEE 2nd International Verification and Security Workshop (IVSW)*, Thessaloniki, Greece, 2017.
- [75] I. E. Sutherland, „Micropipelines,” *Communications of the ACM*, vol. 32, no. 6, pp. 720 - 738, June 1989.
- [76] M. T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, M. Tehranipoor, „TI-TRNG: Technology independent true random number generator,” *51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2015.
- [77] N. A. B. N. Hashim, F. A. B. Hamid, J. Teo, M. S. A. Hamid, „Analysis of memristor based ring oscillators for hardware security,” *IEEE International Conference on Semiconductor Electronics (ICSE)*, Kuala Lumpur, Malaysia, 2016.
- [78] V. K. Rai, S. Tripathy, J. Mathew, „Memristor based Random Number Generator: Architectures and Evaluation,” *International Conference on Smart Computing and Communications, ICSCC*, Kurukshetra, India, 2017.
- [79] J. Cartagena, H. Gomez, E. Roa, „A fully-synthesized TRNG with lightweight cellular-automata based post-processing stage in 130nm CMOS,” *IEEE Nordic Circuits and Systems Conference (NORCAS)*, Copenhagen, Denmark, 2016.
- [80] R. Latypov, E. Stolov, „Ternary jitter-based true random number generator,” *Journal of Physics: Conference Series*, vol. 783, no. 012046, 2017.
- [81] „ID Quantique, the home of Quantum-Safe Crypto,” ID Quantique , [Online]. Available: <https://www.idquantique.com/>. [Data uzyskania dostępu: 10 07 2018].
- [82] Ł. Matuszewski, P. Kubczak, „Tests of a quantum random number generator in different temperatures,” *Problems and Progress in Metrology: PPM'15*, Kościelisko, Poland, 2015.
- [83] A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, H. Zbinden, „Optical quantum random number generator,” *Journal of Modern Optics*, vol. 47, no. 4, pp. 595 - 598, 2000.
- [84] M. Stipčević, B. M. Rogina, „Quantum random number generator based on photonic emission in semiconductors,” *Review of Scientific Instruments*, vol. 78, no. 4, 2007.
- [85] T. Tuncer, „Implementation of Duplicate TRNG on FPGA,” *Elektronika ir Elektrotechnika*, vol. 21, no. 4, 2015.
- [86] P. Z. Wieczorek, „True random number generator resistant to frequency injection attacks,” *Electronics Letters*, vol. 51, no. 5, pp. 384 - 386, 2015.
- [87] P. Z. Wieczorek, K. Gołofit, „True Random Number Generator Based on Flip-Flop Resolve Time Instability Boosted by Random Chaotic Source,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, April 2018.
- [88] Y. Yang, S. Jia, Y. Wang, S. Zhang, C. Liu, „A reliable true random number generator based on novel chaotic ring oscillator,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, USA, 2017.

- [89] S. Rajagopalan, S. Rethinam, A. N. Deepika, A. Priyadarshini, M. Jyothirmai, A. Rengarajan, „Design of boolean chaotic oscillator using CMOS technology for true random number generation,” *International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, Vellore, India, 2017.
- [90] D. P. Rosin, „Ultra-Fast Physical Generation of Random Numbers Using Hybrid Boolean Networks,” *Dynamics of Complex Autonomous Boolean Networks. Springer Theses (Recognizing Outstanding Ph.D. Research)*, Cham, Springer, 2015, pp. 57 - 79.
- [91] L. Dong, H. Yang, Y. Zeng, „Analysis and Improvement of True Random Number Generator Based on Autonomous Boolean Network,” *13th International Conference on Computational Intelligence and Security (CIS)*, Hong Kong, China, 2017.
- [92] S. Espejo-Meana, A. Rodriguez-Vazquez, J. Huertas, J. Quintana, „Application of chaotic switched-capacitor circuits for random number generation,” *1989 European Conference on Circuit Theory and Design*, Brighton, UK., 1989.
- [93] M. Drutarovský, P. Galajda, „Chaos-based True Random Number Generator embedded in a mixed-signal reconfigurable hardware,” *Journal of ELECTRICAL ENGINEERING*, vol. 57, no. 4, pp. 218-225, 2006.
- [94] M. Degaldo-Restituto, F. Medeiro, A. Rodriguez-Vazquez, „Nonlinear switched-current CMOS IC for random signal generation,” *Electronics Letters*, vol. 29, no. 25, pp. 2190 - 2191, 1993.
- [95] M. Delgado-Restituto, A. Rodriguez-Vazquez, „Integrated chaos generators,” *Proceedings of the IEEE*, pp. 747 - 767, May 2002.
- [96] S. Callegari, R. Rovatti, G. Setti, „Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos,” *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 793 - 805, Feb. 2005.
- [97] L. O. Chua, Y. Yao, Q. Yang, „Generating randomness from chaos and constructing chaos with desired randomness,” *International Journal of Circuit Theory and Applications*, vol. 18, no. 3, pp. 215-240, 1990.
- [98] G. Bernstein, M. Lieberman, „Secure random number generation using chaotic circuits,” *IEEE Transactions on Circuits and Systems*, vol. 37, no. 9, pp. 1157 - 1164, Sep 1990.
- [99] P.-y. Wang, „Chaos In Phase Locked Loop,” *International Symposium on VLSI Design, Automation and Test*, Hsinchu, Taiwan, 2006.
- [100] T. Endo, L. Chua, „Chaos from phase-locked loops,” *IEEE International Symposium on Circuits and Systems*, Espoo, Finland, 1988.
- [101] B. Vizvari, G. Kolumban, „Quality evaluation of random numbers generated by chaotic sampling phase-locked loops,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, no. 3, pp. 216 - 224, Mar 1998.
- [102] J. Sobotka, V. Zeman, „Design of the true random numbers generator,” *Elektrorevue*, pp. 47-52, 2011.
- [103] Y. Hu, X. Liao, K.-w. Wong, Q. Zhou, „A true random number generator based on mouse movement and chaotic cryptography,” *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2286 - 2293, 15 June 2009.
- [104] G. Liu, F. Yang, Y. Zhang, C. Du, „A new true random number generator based on feedback system,” *Journal of Computational Information Systems*, vol. 10, no. 19, pp. 8469 - 8476, 2014.
- [105] N. Bardis, P. Markovskyi, N. Doukas, N. V. Karadimas, „True Random Number Generation Based on Environmental Noise Measurements for Military Applications,” *8th WSEAS, International Conference on Signal Processing, Robotics and Automation*, Cambridge, United Kingdom, 2009.

- [106] D. E. Holcomb, W. P. Burleson, K. Fu, „Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers,” *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198 - 1210, Sep. 2009.
- [107] V. Gaglio, A. D. Paola, M. Ortolani, G. L. Re, „A TRNG Exploiting Multi-Source Physical Data,” *6th ACM workshop on QoS and security for wireless and mobile networks*, Bodrum, Turkey, 2010.
- [108] M. S. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, M. Boyle, „NIST Special Publication 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation,” National Institute of Standards and Technology, 2018.
- [109] M. Soucarros, C. Canovas-Dumas, J. Clédière, P. Elbaz-Vincent, D. Réal, „Influence of the temperature on true random number generators,” *IEEE International Symposium on Hardware-Oriented Security and Trust*, San Diego CA, USA, 2011.
- [110] S. Michalak, „Raspberry Pi as a measurement system control unit,” *International Conference on Signals and Electronic Systems (ICSES)*, Poznan, Poland, 2014.
- [111] K. Wold, S. Petrović, „Robustness of TRNG against Attacks that Employ Superimposing Signal on FPGA Supply Voltage,” *Norwegian Information Security Conference*, Gjøvik, Norway, 2010.
- [112] G. O. Dyrkolbotn, W. Knut, E. Snekkenes, „Security Implications of Crosstalk in Switching CMOS Gates,” *Information Security. ISC 2010. Lecture Notes in Computer Science*, pp. 269 - 275, 2011.
- [113] F. Reif, *Fizyka statystyczna*, Warszawa: PWN, 1973.
- [114] M. Borda, *Fundamentals in Information Theory and Coding*, Springer-Verlag Berlin Heidelberg, 2011.
- [115] W. Killmann, W. Schindler, „A proposal for: Functionality classes for random number generators. Version 2.0,” Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, 2011.
- [116] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, N. Heckert, J. Dray, „A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” National Institute of Standards and Technology Special Publication 800-22 revision 1a, 2010.
- [117] D. E. Knuth, *The Art of Computer Programming*, vol. 2 Seminumerical Algorithms, Addison-Wesley, 1981.
- [118] W. Killmann, W. Schindler, „A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators, Version 3.1,” Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, 2001.
- [119] National Institute of Standards and Technology, „Security Requirements for Cryptographic Modules,” Information Technology LaboratoryNational Institute of Standards and Technology, Gaithersburg, MD, 2002.
- [120] G. Marsaglia, „The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness,” Department of Statistic, Florida State University, Tallahassee, Florida, 1995.
- [121] R. G. Brown, „Dieharder: A Random Number Test Suite,” [Online]. Available: <http://webhome.phy.duke.edu/~rgb/General/dieharder.php>. [Data uzyskania dostępu: 07 09 2018].
- [122] Optimization, Canada Research Chair in Stochastic Simulation and, „TestU01,” [Online]. Available: <http://simul.iro.umontreal.ca/testu01/tu01.html>. [Data uzyskania dostępu: 09 09 2018].
- [123] B. Valtchanov, A. Aubert, F. Bernard, V. Fischer, „Modeling and observing the jitter in ring oscillators implemented in FPGAs,” *11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, Bratislava, Slovakia, 2008.

- [124] V. Fischer, F. Bernard, N. Bochard, M. Varchola, „Enhancing security of ring oscillator-based trng implemented in FPGA,” *International Conference on Field Programmable Logic and Applications*, Heidelberg, Germany, 2008.
- [125] B. Valtchanov, V. Fischer, A. Aubert, F. Bernard, „Characterization of randomness sources in ring oscillator-based true random number generators in FPGAs,” *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Vienna, Austria, 2010.
- [126] M. Jessa, M. Jaworski, „Randomness of a combined TRNG based on the ring oscillator sampling method,” *ICSES 2010 International Conference on Signals and Electronic Circuits*, Gliwice, Poland, 2010.
- [127] X. Xu, Y. Wang, „High Speed True Random Number Generator Based on FPGA,” *International Conference on Information Systems Engineering (ICISE)*, Los Angeles, CA, USA, 2016.
- [128] K. Wold, S. Petrović, „Security properties of oscillator rings in true random number generators,” *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, Tallinn, Estonia, 2012.
- [129] Ü. Güler, S. Ergün, G. Dündar, „A digital IC Random Number Generator with logic gates only,” *17th IEEE International Conference on Electronics, Circuits and Systems*, Athens, Greece , 2010.
- [130] A. Hajimiri, S. Limotyrakis, T. Lee, „Jitter and phase noise in ring oscillators,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 790 - 804, 1999.
- [131] J. Wilson, *Timing Jitter Tutorial & Measurement Guide*, Silicon Labs Timing.
- [132] R. B. Davies, „Exclusive OR (XOR) and hardware randomnumber generators,” 28 February 2002. [Online]. Available: <http://www.robertnz.net/pdf/xor2.pdf>.
- [133] M. Jessa, „On the Quality of Random Sequences Produced with a Combined Random Bit Generator,” *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 791 - 804, 2015.
- [134] P. Bayon, F. Poucheret, L. Bossuet, B. Robisson, A. Aubert, P. Maurine i V. Fischer, „Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator,” *Third International Workshop on Constructive Side-Channel Analysis and Secure Design, COSADE 2012*, Darmstadt, Germany, 2012.
- [135] C. Shao, H. Li, J. Zhou, „Fast and automatic security test on cryptographic ICs against fault injection attacks based on design for security test,” *IET Information Security*, vol. 11, no. 6, pp. 312 - 318, 2017.
- [136] H. M. Oliveira i L. V. Melo, „Huygens synchronization of two clocks,” *Scientific Reports*, no. 5, 2015.
- [137] Maxim Integrated, *Clock (CLK) Jitter and Phase Noise Conversion*, APPLICATION NOTE 3359, 2004.
- [138] P. R. Trischitta, E. L. Varma, *Jitter in Digital Transmission Systems*, Artech House, 1989.
- [139] N. Madhu, „Note on measures for spectral flatness,” *Electronics Letters*, vol. 45, no. 23, pp. 1195-1196, November 2009.

SPIS RYSUNKÓW I TABEL

Rysunek 2.1 Klasyfikacja generatorów ciągów losowych	14
Rysunek 2.2 Schemat blokowy generatora ciągów losowych	15
Rysunek 2.3 Generator oparty o zjawisko szumu w układach elektronicznych	17
Rysunek 2.4 Schemat generatora liczb losowych z bramkami AND i przerzutnikami	17
Rysunek 2.5 Schemat scalonego źródła szumu	18
Rysunek 2.6 Generator TRNG firmy Intel	18
Rysunek 2.7 Schemat blokowy generatora TRNG z układem próbującym pamiętającym	19
Rysunek 2.8 Generator TRNG z automatyczną regulacją poziomu odniesienia komparatora	20
Rysunek 2.9 Generator z automatycznym równoważeniem bitów.....	20
Rysunek 2.10 Generator TRNG z dwoma diodami Zenera.....	21
Rysunek 2.11 Generator TRNG z 8 niezależnymi źródłami szumu	21
Rysunek 2.12 Zjawisko metastabilności.....	22
Rysunek 2.13 Schemat generatora wykorzystującego stany metastabilne przerzutnika typu D.....	22
Rysunek 2.14 Wieloelementowy generator TRNG z otwartą pętlą.....	23
Rysunek 2.15 Generator TRNG z przerzutnikiem RS	24
Rysunek 2.16 Generator losowy z układem bistabilnym.....	25
Rysunek 2.17 Generator TRNG z metastabilnymi oscylatorami pierścieniowymi	25
Rysunek 2.18 Źródło entropii złożone z dwóch inwerterów	25
Rysunek 2.19 Generator TRNG z próbkowaniem oscylatora	26
Rysunek 2.20 Generator TRNG z pętlą synchronizacji fazy	27
Rysunek 2.21 Układ próbujący generatora TRNG	27
Rysunek 2.22 Generator z LFSR i CASR	28
Rysunek 2.23 Generator z oscylatorami GARO i FIRO.....	29
Rysunek 2.24 Łaczony generator TRNG.....	30
Rysunek 2.25 Samotaktyczny generator TRNG.....	32
Rysunek 2.26 Kwantowy generator TRNG wykorzystujący polaryzację fotonów	34
Rysunek 2.27 Hybrydowy generator TRNG	35
Rysunek 2.28 Przykład układu chaotycznego	36
Rysunek 3.1 Proces projektowania układów cyfrowych w technologii FPGA	42
Rysunek 4.1 Algorytm testowania źródła entropii	46
Rysunek 4.2 Algorytm testu liczby powtórzeń.....	49
Rysunek 4.3 Algorytm adaptacyjnego testu proporcji.....	49
Rysunek 4.4 Sposób wyznaczania statystyki χ^2	56
Rysunek 5.1 Definicja jittera	57
Rysunek 5.2 Oscylator pierścieniowy	57
Rysunek 5.3 Pomiar jittera cykl do cyklu.....	60
Rysunek 5.4 Histogram jittera dla opóźnienia złożonego z dwóch inwerterów	62
Rysunek 5.5 Histogram jittera dla opóźnienia złożonego z czterech inwerterów	62
Rysunek 5.6 Histogram jittera dla opóźnienia złożonego z sześciu inwerterów	62

SPIS RYSUNKÓW I TABEL

Rysunek 5.7 Histogram jittera dla opóźnienia złożonego z ośmiu inwerterów	62
Rysunek 5.8 Histogram jittera dla opóźnienia złożonego z dziesięciu inwerterów	62
Rysunek 5.9 Histogram jittera dla opóźnienia złożonego z jednego zatrzasku	63
Rysunek 5.10 Histogram jittera dla opóźnienia złożonego z dwóch zatrzasków	63
Rysunek 5.11 Histogram jittera dla opóźnienia złożonego z trzech zatrzasków	63
Rysunek 5.12 Histogram jittera dla opóźnienia złożonego z czterech zatrzasków	63
Rysunek 5.13 Histogram jittera dla opóźnienia złożonego z pięciu zatrzasków	63
Rysunek 5.14 Histogram jittera dla opóźnienia złożonego z jednego odczepu linii opóźniającej.....	64
Rysunek 5.15 Histogram jittera dla opóźnienia złożonego z dwóch odczepów linii opóźniającej.....	64
Rysunek 5.16 Histogram jittera dla opóźnienia złożonego z trzech odczepów linii opóźniającej.....	64
Rysunek 5.17 Histogram jittera dla opóźnienia złożonego z czterech odczepów linii opóźniającej	64
Rysunek 5.18 Histogram jittera dla opóźnienia złożonego z pięciu odczepów linii opóźniającej.....	64
Rysunek 5.19 Próbkowanie jittera oscylatora pierścieniowego jako metoda wytwarzania losowych bitów.	65
Rysunek 5.20 Łączenie modulo 2 wielu źródeł losowości	65
Rysunek 5.21 Wykres oka, wartości μ i c , dla których wytwarzanie są nieskorelowane bity [131]	67
Rysunek 5.22 Start-stopowy generator TRNG	68
Rysunek 5.23 Rozrzut częstotliwości oscylatorów pierścieniowych.....	69
Rysunek 5.24 Łaczony start-stopowy generator TRNG z detektorami fazy.....	71
Rysunek 5.25 Detektor fazy użyty w start-stopowym generatorze TRNG.....	71
Rysunek 5.26 Przebieg czasowy generatora TRNG z pracy [61] złożonego z dwóch RO	75
Rysunek 5.27 Widmo przebiegu generatora TRNG z pracy [61] złożonego z dwóch RO	75
Rysunek 5.28 Przebieg czasowy generatora TRNG z pracy [61] złożonego z czterech RO	75
Rysunek 5.29 Widmo przebiegu generatora TRNG z pracy [61] złożonego z czterech RO	75
Rysunek 5.30 Przebieg czasowy generatora TRNG z pracy [61] złożonego z ośmiu RO	75
Rysunek 5.31 Widmo przebiegu generatora TRNG z pracy [61] złożonego z ośmiu RO	75
Rysunek 5.32 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z dwóch RO.....	76
Rysunek 5.33 Widmo generatora TRNG z detektorem fazy złożonego z dwóch RO	76
Rysunek 5.34 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z czterech RO	76
Rysunek 5.35 Widmo generatora TRNG z detektorem fazy złożonego z czterech RO.....	76
Rysunek 5.36 Przebieg czasowy generatora TRNG z detektorem fazy złożonego z ośmiu RO	76
Rysunek 5.37 Widmo generatora TRNG z detektorem fazy złożonego z ośmiu RO	76
Rysunek 5.38 Wartość m_{\min} w funkcji K dla częstotliwości próbkowania $f_L = 100 \text{ MHz}$	94
Rysunek 5.39 Wartość m_{\min} w funkcji K dla częstotliwości próbkowania $f_L = 200 \text{ MHz}$	95
Rysunek 5.40 Wartość m_{\min} w funkcji K dla częstotliwości próbkowania $f_L = 300 \text{ MHz}$	95
Rysunek 5.41 Równolegle połączenie N generatorów TRNG.....	96
Rysunek 5.42 Kod modułu równoległego generatora TRNG	97

Tabela 5.1 Wyniki pomiarów parametrów oscylatorów pierścieniowych	60
Tabela 5.2 Testy normalności rozkładu.....	61
Tabela 5.3 Częstotliwości oscylatorów pierścieniowych w różnych układach FPGA	70
Tabela 5.4 Tabela prawdy detektora fazy	71
Tabela 5.5 Porównanie wartości płaskości widma generatorów TRNG	77
Tabela 5.6 Wyniki testów statystycznych NIST 800-90B dla układu Spartan 3	80
Tabela 5.7 Wyniki testów statystycznych NIST 800-90B dla układu Spartan 6	80
Tabela 5.8 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 4	81
Tabela 5.9 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 5	81
Tabela 5.10 Wyniki testów statystycznych NIST 800-90B dla układu Virtex 6	82
Tabela 5.11 Wyniki testów statystycznych NIST 800-90B dla układu Artix 7	82
Tabela 5.12 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone II	83
Tabela 5.13 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone IV	83
Tabela 5.14 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone V	84
Tabela 5.15 Wyniki testów statystycznych NIST 800-90B dla układu Cyclone 10	84
Tabela 5.16 Wyniki testów statystycznych NIST 800-90B dla układu Stratix IV	85
Tabela 5.17 Wyniki testów statystycznych NIST 800-90B dla układu MAX 10	85
Tabela 5.18 Wyniki testów statystycznych NIST 800-90B dla układu ECP3	86
Tabela 5.19 Wyniki testów statystycznych NIST 800-90B dla układu ECP5	86
Tabela 5.20 Wyniki testów statystycznych NIST 800-22 dla układu Spartan 3	87
Tabela 5.21 Wyniki testów statystycznych NIST 800-22 dla układu Spartan 6	87
Tabela 5.22 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 4	88
Tabela 5.23 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 5	88
Tabela 5.24 Wyniki testów statystycznych NIST 800-22 dla układu Virtex 6	89
Tabela 5.25 Wyniki testów statystycznych NIST 800-22 dla układu Artix 7	89
Tabela 5.26 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone II	90
Tabela 5.27 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone IV	90
Tabela 5.28 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone V	91
Tabela 5.29 Wyniki testów statystycznych NIST 800-22 dla układu Cyclone 10	91
Tabela 5.30 Wyniki testów statystycznych NIST 800-22 dla układu Stratix IV	92
Tabela 5.31 Wyniki testów statystycznych NIST 800-22 dla układu MAX 10	92
Tabela 5.32 Wyniki testów statystycznych NIST 800-22 dla układu ECP3	93
Tabela 5.33 Wyniki testów statystycznych NIST 800-22 dla układu ECP5	93
Tabela 5.34 - Wykorzystanie zasobów FPGA i maksymalna szybkość wytwarzania losowych bitów	98
Tabela 5.35 Wyniki testów statystycznych NIST 800-90B dla szybkiego generatora TRNG	98
Tabela 5.36 Wyniki testów statystycznych NIST 800-22 dla szybkiego generatora TRNG	98
Tabela 5.37 Porównanie generatorów TRNG.....	99
