BeatFrequency Detection

Ramya Viligilla
PG Student ECE.
Malla Reddy Engineering College
Jawaharlal Nehru Technological University
Hyderabad, India
ramyaviligilla3@gmail.com

Suman Sukhavasi,
Assistant Professor ECE
Malla Reddy Engineering College
Jawaharlal Nehru Technological University
Hyderabad, India
suman.sukhavasi@gmail.com

Dr. M. Jagadeesh Chandra Prasad
Professor & HoD ECE
Malla Reddy Engineering College
Jawaharlal Nehru Technological University
Hyderabad, India
jagadishmatta@gmail.com

***Abstract*—Built-In Self-Test (BIST) are the major building blocks in every integrated circuit, which corrects the memory faults, stuck-at faults automatically by applying the random patterns. The performance of BIST modules purely depends on randomization of patterns. However, conventional liner feedback shift registers (LFSR) are failed to provide the higher randomization with lower hardware resource utilization. Therefore, this work is focused on implementation of Random Test Pattern Generator using Beat Frequency Detection(RTPG-BFD) frameworks to solve this problem. Further, the ring oscillators in the conventional methods were replaced by Digital Clock Managers (DCM), which implements the tuneability of phase, frequency of random numbers. Further, the BFD operation is achieved by D-flip flop (D-FFs), post processing units, and counters. The simulations revealed that the proposed method resulted in better area, delay, power performance as compared to conventional approaches.**

Keywords*—Built-In Self-Test, True Random Number Generator, Digital Clock Managers, D-flip flop, post processing units, counters

I. INTRODUCTION

In our modern era almost, every digital equipment owns a BIST and for most people this piece of technology has become an essential part of their life. According to an estimation [1] there are currently around 1.9 billion BIST users around the world, which is over 25% of the world's population. This number is expected to keep increasing rapidly in the coming years. For a device with so many users worldwide it is important that it is properly secured [2]. Many BIST users will keep personal data on their memories, so a security leak across a large platform like IC would have a heavy impact on society [3]. However, a BIST does bring possible security vulnerabilities. A malicious party could intercept or alter BIST communication. This would have a serious impact on the security and privacy of the BIST user. For example, a malicious party could listen in on the communication between a BIST user and the bank. This would cause secure bank information to be compromised [4]. The hackers are actively targeting financial BIST modules and significant number of financial modules have been hacked and malicious versions of the modules have been uploaded [5]. Unsuspecting users who use these malicious apps risk their confidential credentials being captured, or their BISTs being exposed to adware.

In order to prevent these scenarios, we need to create a simple way for an IC device to encrypt data and engage in secure encrypted communication [6]. This would be a relatively simple system to build on a larger desktop system because it has practically unlimited resources to work with in terms of computational power and data storage space. However, for a similar system to work on an IC device it will need to take into account the limited number of resources the device has in terms of battery, computational power and data storage space compared to a desktop system [7]. If the encryption process takes up too much resources, it will interfere with the device user's normal RTPG-BFD activities. A crucial step for encrypted communication to take place is the generation of a strong random seed as input for the encryption scheme. Ideally the generation of this random seed would take place on the device using a limited number of computations, battery power and storage space [8]. Thus, RTPG-BFD present a strong lightweight randomness generator prototype in the IC user space. On boot-up of the IC-RTPG-BFD the prototype generates an entropy pool from noise in the RTPG-BFD data. Once the prototype has generated a sufficiently strong entropy pool, other processes on the RTPG-BFD are able to request the prototype for any size of random output. Before presenting the prototype [9], RTPG-BFD analyze the sensor data and estimate the required time to run the data generation process on device boot-up before the entropy pool can be considered sufficiently strong. For this purpose, RTPG-BFDs [10] have built an oscillator module, which generates a large amount of data from different device sensors. The goal of the randomness generation prototype is to provide a source of strong randomness which can be used in addition to randomness from existing sources like LFSR. This combination will create a stronger and more secure random seed for encryption processes on the device. The broad objective of the present study is to examine the degree and quality of randomness of various generators [15]. In tune with this, the following four specific objectives have been framed for the study.

- Implementation of RTPG-BFD with for generating the unpredictable random numbers.
- Design of DCMs to achieve the tuneability of phase, frequency of random numbers.
- The BFD operation is achieved by D-FFs, post processing units, and counters.

Rest of the article is organized as follows: section 2 delas with literature survey, section 3 delas with the proposed DCM based RTPG-BFD implementation, section 4 delas with analysis of results with performance comparison, section 5 concludes the article with possible future directions.

II. LITERATURE SURVEY

Several researchers use various entropy sources to generate true random numbers. For example, RAND Corporation [16] generate numbers using a random pulse generator. In [17] authors used the image data from a camera which is pointed at a couple of ring oscillators is used as excellent entropy source to generate true random numbers. RTPG-BFD uses radioactive decay as the entropy source to generate random number sequences. In a simple case, a variable environment of a fish tank is used as an entropy source of randomness. Up To date only few organizations offer true random numbers commercially using these kinds of techniques. In [18] authors used a method of generating random numbers using Celestial oscillator sources and the generated sequence is tested with NIST Statistical Test Suite for random data. They found that the resulting data sets pass all tests in the NIST with a mean of 98.9% of the 512 total bit streams as well as further testing in R. In [19] authors performed Entropy estimation is a vital part in building a RTPG-BFD because being able to give an accurate estimation of the amount of entropy contained in the entropy pool is required to reach a certain level of security. If the accuracy of the entropy estimation of a RTPG-BFD is high, it can give better security guarantees about the unpredictability of its entropy pool. This makes it less likely for an attacker to compromise the randomness of the RTPG-BFD system.

In [20] authors proved that the complexity of estimating the min-entropy of a distribution is SBP-complete, which stands for “small bounded-error probability” which is a custom class of complexity that is believed to be equal to NP-complete complexity. So, the problem of proving an entropy pool to be truly random is computationally hard, so instead estimation has to be made using indirect measures. In [21] authors stated a problem by the TRNG is an interesting issue, but it assumes a theoretical scenario that might not be realistic in a practical usage scenario. Especially for the usage scenario of this thesis (i.e., IC devices) the scenario described will be unlikely to cause serious problems. In [22] authors stated that, solutions should be found to provide additional entropy on the IC devices so that the internal state is much less likely to be compromised because of low-entropy events in the first place. The PRNG should continuously provide the user with strong random seeds for use in cryptographic protocols on the device, while still remaining as efficient as possible using the limited resources that are offered by the device. In [23] authors proposed, the PSPG generates an entropy pool from a number of sources on the hardware level such as inter-keyboard timings and inter-interrupt timings. These sources of entropy are assumed to be non-deterministic and hard for an outside observer to measure. In [24] authors proposed, the Linux kernel also provides a second PUF-TRNG. It is identical to LFSR in its functionality, the only difference is that LFSR is non-blocking and has no limit to the number of requests for bytes of randomness it can take. In [25] authors proposed, the RO-TRNG is widely used by most applications and generally considered secure. However, LFSR has been criticized by a number of papers which claim that it has vulnerabilities. Even the source code of LFSR states a weakness with regards to predictability on system start-up.

III. PROPOSED METHOD

This section gives the detailed analysis of proposed DCM-RTPG-BFD implementation. The detailed designs are shown in Figure 1. Here, the ring oscillators in the conventional methods were replaced by DCM, which implements the tuneability of phase, frequency of random numbers. Further, the BFD operation is achieved by D-FFs, post processing units, and counters.

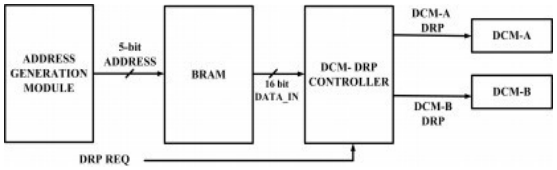

Fig. 1. Proposed DCM-RTPG-BFD block diagram.

The DCM-RTPG-BFD method's detailed process is as follows:

Step 1: Initially, address generation module generates the different address based on user requirements. The addresses are changed based on application introduced in BIST environment.

Step 2: The block random access memory (BRAM) stores the data in parallel manner, which gives lower synchronization problems as compared to SRAM.

Step 3: DCM-DRP controller: Here, the dynamic reconfigurable controller is introduced for controlling the frequencies of DCM modules. The DCM-DRP controllers functions using first in first out (FIFO) operation.

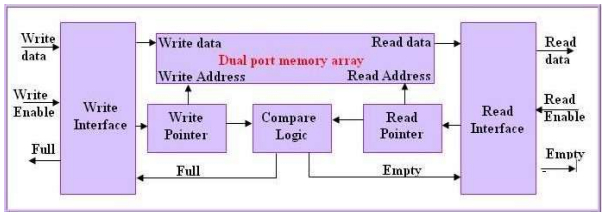

Fig. 2. operation diagram of FCM-DRP controller.

An array of memory called a FIFO or Queue is often used in electronics to transport data between two circuits with various clock rates. The dual port memory that FIFO employs has two pointers that point to the read and write locations. Figure 2 is a generalized block diagram of FIFO. Rotating pointers are often used to create FIFO modules. A FIFO's write and read pointers might be referred to as the head and tail of the data region. The FIFO's initial read and write pointers will both point to the same place. After writing data to the "n"th position in a FIFO with n locations, the write pointer refers to the 0th location. The write pointer and read pointer are located at the same positions in this 8-bit FIFO, respectively. The diagram's shaded region is filled with

information. Both synchronous and asynchronous FIFOs are possible. The asynchronous FIFO has no clock. Since some FIFOs have separate clocks for read and write, synchronous FIFO may have 1 or 2 clocks. Because synchronous FIFOs provide better interface time, asynchronous FIFOs are less often utilized nowadays. Empty or full flags are set if the FIFO counter reaches zero or BUF LENGTH. If a write occurs and the buffer is not full, the FIO counter is increased. If a read occurs and the buffer is not empty, the FIO counter is decremented. If both reading and writing occur, the counter will not change. Read and write pointers are rd ptr and wr ptr, respectively. The bits in these registers were chosen to match the address width of the buffer, so when the buffer overflows, the values will also overflow and become 0. Finally, DCM-DRP controller generates the control signals such as DCM-A-DRP and DCM-B-DRP.

Step 3: DCM-A performs the frequency division by 2 of clock upon enable of DCM-A-DRP.

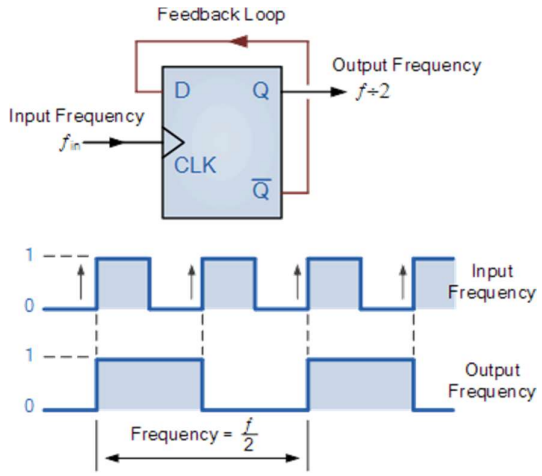

Fig. 3. DCM-A operation.

The frequency waveforms in Figure 3 demonstrate that when the output from Q is "fed back" to the input terminal D, the output pulses at Q have a frequency that is precisely one half (2) that of the input clock frequency. In other words, the circuit now divides the input frequency by a factor of two, producing frequency division (an octave). As a result, a "ripple counter" is created. In ripple counters, the clock pulse activates the first flip-flop, whose output triggers the second flip-flop, which in turn triggers the third flip-flop, and so on along the chain, creating a rippling effect that gives the timing signal its name.

Step 4: DCM-B performs the frequency division by 3 of clock upon enable of DCM-B-DRP. Figure 4 illustrates the 1/3 frequency divider circuit, which consists of two JK flip-flops, an OR gate, and a NOT gate. The previous output state of each flip-flop (QA and QB, respectively) must be used to determine the input of the first JK flip-flop and the inputs for the remaining flip-flops in the circuit. Each Flip-input Flop's should perform the same function in order to maintain the cycle of its combinational output. Karnaugh map is used to get the input equation for each JK Flip-Flop while designing a sequential circuit. To simplify the design process, the J terminal and K terminal of a Flip-Flop might be linked. The transition Table indicates that two Flip-Flops are required to get at the final outcomes. For JK flip Flop inputs, the standard count sequence is J=0, K=0, J= 0, K=1, J= 1, K=0. The rows of the columns Next JA, Next KA, Next JB, and Next KB are filled by the 1/3 frequency divider circuit. When analyzing a frequency cycle of QA and QB, QA's cycle is 0 1 0 0 1 0' and it includes the unit "X 0 0 X." As a result, the KA terminal may be directly linked to the HIGH signal. In the same circumstance, the KB terminal may be directly linked to signal logic "1" when the frequency cycle of QB does not include "X 1 1 X".

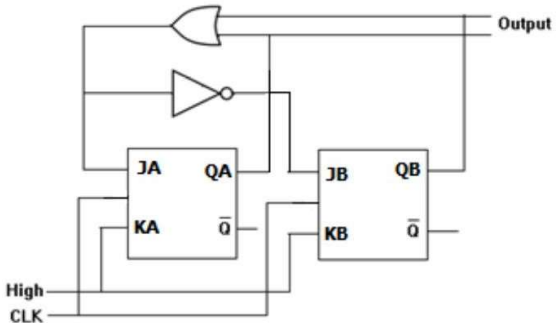

Fig. 4. DCM-B operation.

Step 5: DCM-A frequency is applied as data input to DFF and DCM-B frequency is applied as clock input to DFF, which estimates the difference in frequencies. The DFF output becomes high during high frequency change, and the DFF output becomes low during low frequency change.

Step 6: DFF output is applied as reset input to counter. If the reset=1, then counter initialized to zero, else reset=0, counter starts initializing the random sequences.

Step 7: Finally, post processing unit contains the zigzag random connections of XOR, which generates the non-repeated random numbers.

SIMULATION RESULTS

Xilinx ISE software was used to create all of the DCM-RTPG-BFD designs. This software programmed gives two types of outputs: simulation and synthesis. The simulation results provide a thorough examination of the DCM-RTPG-BFD architecture in terms of input and output byte level combinations. Decoding procedure approximated simply by applying numerous combinations of inputs and monitoring various outputs through simulated study of encoding correctness. The use of area in relation to the transistor count will be accomplished as a result of the synthesis findings. In addition, a time summary will be obtained with regard to various path delays, and a power summary will be prepared utilizing the static and dynamic power consumption.

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	16	35200	0%
Number of Slice LUTs	16	17600	0%
Number of fully used LUT-FF pairs	16	16	100%
Number of bonded IOBs	35	100	35%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig. 5. Design summary.

Figure 5 shows the design (area) summary of proposed method. Here, the proposed method utilizes the low area in terms of slice LUTs i.e., 16 out of available 17600. Further, the proposed method utilizes the slice registers as 16, out of available 35200. Further, the proposed method utilizes fully used LUT-FF as 16, out of available 16.

```
Data Path: load to r_LFSR_1
                           Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   IBUF:I->O             16   0.000   0.497  load_IBUF (load_IBUF)
   LUT4:I1->O             1   0.043   0.000  mux161 (r_LFSR[15]_seed[15]_mux_3_OUT<0>)
   FDE:D                     -0.001         r_LFSR_1
  ----------------------------------------
  Total                      0.540ns (0.043ns logic, 0.497ns route)
                                     (8.0% logic, 92.0% route)
```

Fig. 6. Time summary.

Figure 6 shows the time summary of proposed method. Here, the proposed method consumed total 0.316ns of time delay, which is entirely route delay.

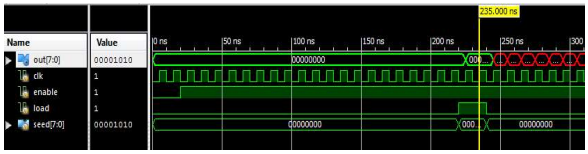

Fig. 7. Simulation outcome.

Figure 7 presents the simulation outcome of proposed system. Here, clock (clk), enable, load, and seed are the input data pins. Further, out is the output pin. Initially, a logic high enable causes the system will function and a logic high load resulted in loading of seed value. So, 8 bit seed value (example 1010) is loaded into TRNG and resulted out as 1010. Further, the random numbers are generated for each clock cycle.

```
2.  Summary
2.1.  On-Chip Power Summary
-------------------------------------------------------------------------------
|                          On-Chip Power Summary                              |
-------------------------------------------------------------------------------
|       On-Chip       | Power (mW) |  Used  | Available | Utilization (%) |
-------------------------------------------------------------------------------
| Clocks              |       1.30 |      3 |       --- |             --- |
| Logic               |       0.00 |     10 |     11776 |               0 |
| Signals             |       0.00 |     20 |       --- |             --- |
| IOs                 |       0.00 |     20 |       372 |               5 |
| Quiescent           |      31.52 |        |           |                 |
| Total               |      32.83 |        |           |                 |
-------------------------------------------------------------------------------
```

Fig. 8. Power summary.

Figure 8 shows the power consumption report of proposed DCM-RTPG-BFD. Here, the proposed DCM-RTPG-BFD consumed power as 32.83 milli watts. Table 1 compares the performance evaluation of various TRNG controllers. Here, the proposed DCM-RTPG-BFD resulted in superior (reduced) performance in terms of LUTs, slice registers, LUT-FFs, time-delay, and power consumption as compared to conventional approaches such as PRNG [22], PUF-TRNG [24], and RO-TRNG [25]. Further, the graphical representation of performance comparison is presented in Figure 9.

TABLE I. PERFORMANCE EVALUATION.

Metric	PRNG [22]	PUF TRNG [24]	RO-TRNG [25]	Proposed DCM-RTPG-BFD
Slice Registers	56	45	32	16
LUTs	67	55	42	16
LUT-FFs	74	52	39	16
Time delay (ns)	0.927	0.837	0.735	0.540
Power consumption (mw)	82.61	73.41	58.26	32.83

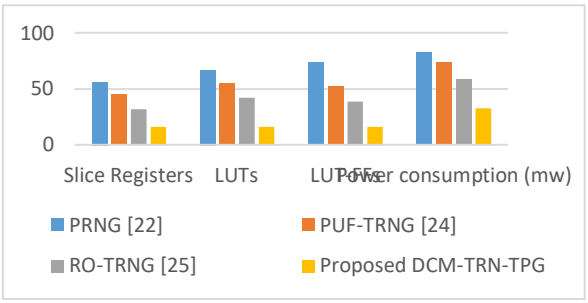

Figure 8. Graphical representation of performance evaluation.

CONCLUSION

The introduction of RTPG-BFD frameworks to address this issue is the main goal of this effort. Furthermore, DCM, which enables the tuneability of phase and frequency of random numbers, took the role of the ring oscillators in the traditional approaches. Furthermore, D-FFs, post processing units, and counters are used to carry out the beat frequency detecting operation. The simulations showed that the suggested technique outperformed existing methods in terms of area, latency, and power. The work may be expanded to build real-time safe protocols including public key cryptography, RSA, ECC, and HECC cryptography methods using the DCM-RTPG-BFD outputs as both public and private keys.

REFERENCES

[1] Tseng, Po-Hao, et al. "ReRAM-Based Pseudo-True Random Number Generator With High Throughput and Unpredictability Characteristics." IEEE Transactions on Electron Devices 68.4 (2021): 1593-1597.

[2] Talukdar, Jonti, et al. "A BIST-based Dynamic Obfuscation Scheme for Resilience against Removal and Oracle-guided Attacks." 2021 IEEE International Test Conference (ITC). IEEE, 2021.

[3] Saha, R., Geetha, G., Kumar, G., Buchanan, W. J., & Kim, T. H. (2021). A Secure Random Number Generator with Immunity and Propagation Characteristics for Cryptography Functions. Applied Sciences, 11(17), 8073.

[4] Singh, Vikram, and Kishor P. Sarawadekar. "FPGA Implementation of Chaos based Pseudo Random Number Generator." 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON). IEEE, 2021.

[5] Liu, B., Chang, Y. F., Li, J., Liu, X., Wang, L. A., Verma, D., ... & Lai, C. S. (2022). Bi2O2Se-Based True Random Number Generator for Security Applications. ACS nano, 16(4), 6847-6857.

[6] Bostancı, F. N., Olgun, A., Orosa, L., Yağlıkçı, A. G., Kim, J. S., Hassan, H., ... & Mutlu, O. (2022, April). DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 1141-1155). IEEE.

[7] Garipcan, Ali Murat, and Ebubekir Erdem. "DESSB-TRNG: A novel true random number generator using data encryption standard substitution box as post-processing." Digital Signal Processing 123 (2022): 103455.

[8] Liu, Jing, et al. "Min-entropy estimation for semiconductor superlattice true random number generators." Scientific Reports 12.1 (2022): 1-9.

[9] Lu, Yi - Fan, et al. "A High - Performance Ag/TiN/HfOx/HfOy/HfOx/Pt Diffusive Memristor for Calibration - Free True Random Number Generator." Advanced Electronic Materials (2022): 2200202.

[10] Zhou, Xue, et al. "Impact of relaxation on the performance of GeSe true random number generator based on Ovonic threshold switching." IEEE Electron Device Letters (2022).

[11] Liao, Teh-Lu, Pei-Yen Wan, and Jun-Juh Yan. "Design and synchronization of chaos-based true random number generators and its FPGA implementation." IEEE Access 10 (2022): 8279-8286.

[12] GALLI, DAVIDE. "On the effectiveness of FPGA implemented True Random Number Generators." (2022).

[13] Addabbo, Tommaso, et al. "Low-Level Advanced Design of True Random Number Generators Based on Truly Chaotic Digital Nonlinear Oscillators in FPGAs." International Conference on Applications in Electronics Pervading Industry, Environment and Society. Springer, Cham, 2022.

[14] Xu, Mingtao, et al. "Voltage and temperature dependence of Random Telegraph Noise and their impacts on random number generator." Microelectronics Journal 125 (2022): 105450.

[15] Gomez, Ana I., Markus Kiderlen, and Florian Pausinger. "Improved entropy bounds for parity filtered self-timed ring based random number generators." Information Processing Letters 174 (2022): 106212.

[16] Lv, Yang, Brandon R. Zink, and Jian-Ping Wang. "Bipolar Random Spike and Bipolar Random Number Generation by Two Magnetic Tunnel Junctions." IEEE Transactions on Electron Devices 69.3 (2022): 1582-1587.

[17] Shafi, K. M., Chawla, P., Hegde, A. S., Gayatri, R. S., Padhye, A., & Chandrashekar, C. M. (2022). Multi-bit quantum random number generator from path-entangled single photons. arXiv preprint arXiv:2202.10933.

[18] Cirauqui, David, et al. "Quantum Random Number Generators: Benchmarking and Challenges." arXiv preprint arXiv:2206.05328 (2022).

[19] Zia, Unsub, et al. "A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map." SN Applied Sciences 4.2 (2022): 1-17.

[20] Kong, Dexuan, et al. "Methodology for Random Number Generation Based on FPGA." Proceedings of Sixth International Congress on Information and Communication Technology. Springer, Singapore, 2022.

[21] Bezuidenhout, Riaan, Wynand Nel, and Jacques M. Maritz. "Embedding tamper-resistant, publicly verifiable random number seeds in permissionless blockchain systems." IEEE Access 10 (2022): 39912-39925.

[22] Gupta, Mangal Deep, and Rajeev K. Chauhan. "Hardware Efficient Pseudo-Random Number Generator Using Chen Chaotic System on FPGA." Journal of Circuits, Systems and Computers 31.03 (2022): 2250043.

[23] Ryan, C., Kshirsagar, M., Vaidya, G., Cunningham, A., & Sivaraman, R. (2022). Design of a cryptographically secure pseudo random number generator with grammatical evolution. Scientific reports, 12(1), 1-10.

[24] Gao, B., Lin, B., Li, X., Tang, J., Qian, H., & Wu, H. (2022). A unified PUF and TRNG design based on 40-nm RRAM with high entropy and robustness for IoT security. IEEE Transactions on Electron Devices, 69(2), 536-542.

[25] Chu, Wei-Yu, et al. "Security Study of RO-TRNG under Fault Disturbance Scenarios." 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC). Vol. 6. IEEE, 2022.