Uneven Seven-Sided Dice Roller

Muhammed Ahmet Polat
Electrical and Electronics Engineering
Abdullah Gül University
Kayseri/TURKIYE

muhammedahmet.polat @agu.edu.tr

Osman Samet Şenol
Electrical and Electronics Engineering
Abdullah Gül University
Kayseri/TURKIYE

osmansamet.şenol @agu.edu.tr

***Abstract*— The objective of this project is to build and implement an uneven seven-sided dice roller using digital system components. The system design consists of a 1-bit data generator, a 2-bit full-adder, and registers for data storage and delivery. Key challenges include creating a pulse generator that generates a random 3-bit integer (corresponding to a decimal range of 1 to 7) at a frequency of 100 Hz while ensuring that the generated data is not synchronized with system clocks for unpredictability. In addition, a bespoke full-adder circuit will be chosen and simulated in CMOS design, using a fixed carry (C0 = 1) for addition operations.**

Signals A and B will be generated using a microcontroller-produced clock (CLK1) and distributed to registers at the given period. The project will the culmination with the integration of all components to demonstrate the system's functionality, with the outcomes compared to theoretical probability. This abstract describes a complete strategy to constructing the uneven seven-sided dice roller within specified limits and timelines.

Keywords: digital system design, pulse generator, complete adder circuit, microcontroller interface, random number generation.

I. INTRODUCTION

The goal of this project is to create and execute an uneven seven-sided dice roller with digital system components. This project includes the creation of a pulse generator (D0) and the design of a one-bit complete adder circuit built in CMOS technology. The resulting system will be able to generate random 3-bit values (corresponding to a decimal range of 1 to 7) at a rate of 100 Hz, which will be saved and updated in an output register (RO) for later retrieval by a microcontroller.

Instead than depending on a single microcontroller or integrated circuit (IC) device, the project's initial phase focuses on building the pulse generator (D0) using discrete components. To find out if two circuit models—the ring oscillator and the NE555 timer circuit—could produce square wave outputs in the required frequency range, they were put through a simulation. The project then uses CMOS technology to develop and analyze a one-bit complete adder circuit after the pulse generator has been implemented successfully. Because it conducts binary addition on two input bits (A and B) and a fixed carry bit (C0 = 1), this circuit is essential to the dice-rolling system because it produces the necessary random number outputs in the specified range. It is ensured that every dice roll yields random and statistically fair outcomes by using the whole adder circuit.

The integrity and proper operation of the dice rolling mechanism depend on the adder circuit as a whole, which was constructed using binary arithmetic concepts. The one-bit full adder circuit and pulse generator will be thoroughly studied and put into practice after this introduction, which will ultimately result in the successful completion of the chaotic seven-sided dice rolling system. Four challenges make up the project tasks: creating a circuit for a pulse generator that generates random data; utilizing CMOS technology to create a complete adder circuit; producing clock signals; and controlling the distribution of data across registers. Every task necessitates the physical and virtual implementation of designated circuits. Every job necessitates the physical implementation of the planned circuits in addition to simulation.

The development of low-power microcontrollers and low-power models, however, is one technical area we may exploit in our endeavor. Because Internet of Things (IoT) devices often need extended battery life and low energy consumption, energy efficiency is particularly critical for these applications.

This report will include our methodical approach to every assignment, as well as the circuit designs, simulation outcomes, real-world implementations, and system performance overall. Furthermore, in order to assess the efficacy of our design, we will examine the produced numbers' distribution and compare it with theoretical probabilities.

II. DESCRIPTION OF ENTIRE SYSTEM

A. *Data (D0) Generator*

For the purpose of producing random numbers, the data generator in the seven-sided dice roller system creates a pulse signal (D0) with a certain frequency range (10fck1 < fdata < 100fck1). This part includes the actual implementation of the data generator, manual calculations, simulation results, and a comprehensive design.

A ring oscillator is a circuit design that generates oscillations using feedback from a series of inverters. In contrast, the NE555 timer can be used as both a timer and a pulse generator. When it receives a trigger, it emits a loud signal for the chosen amount of time. This is single-shot mode[1]. When employed in continuous mode, the NE555 time model was used for the circuit since it simplifies our design by producing square waves as requested.

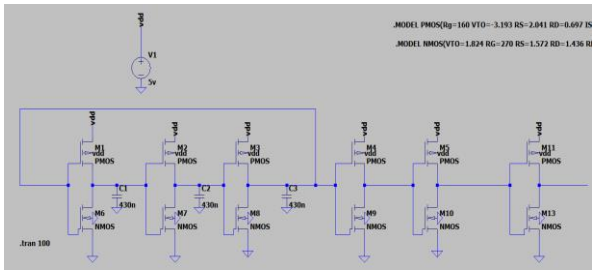

Figure 1. Ring Oscillator Circuit Design

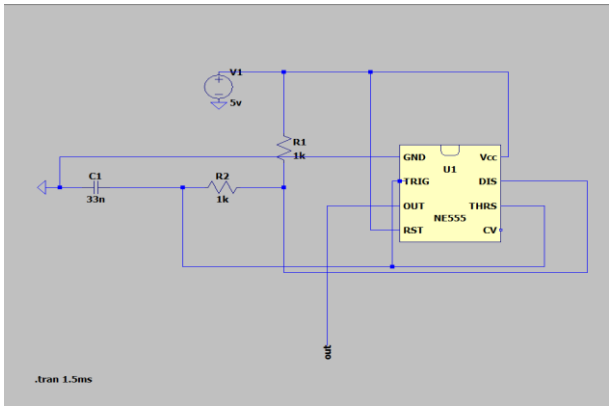

Figure 2. Ne555 Timer Circuit Design

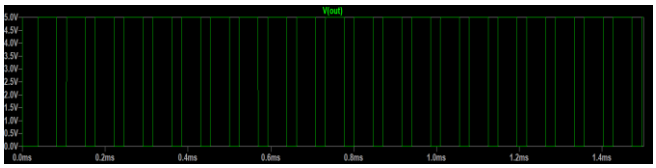

Figure 3. Simulation of Ne555

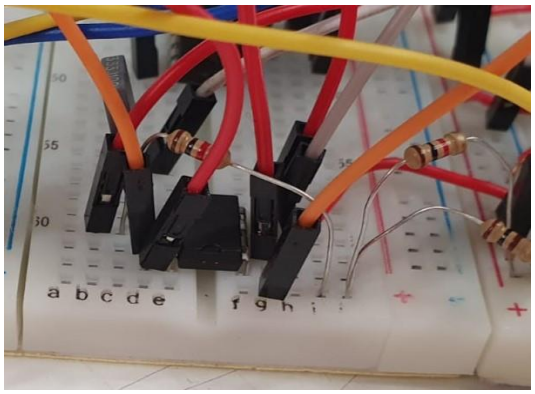

Figure 4. Ne555 Timer Circuit.

Frequency of astable 555 timer:

It is generally determined as the sum of the high and low pulse durations divided by one. To determine high pulse duration T_H) and low pulse duration T_L, the following formula is used:

$$T_H = 0.693 * (R1 + R2) * C1$$

$$T_L = 0.693 * (R2) * C1$$

$$f = \frac{1}{T_H + T_L}$$

So, when frequency calculation was made according to desired circuit, the results are below:

$$T_H = 0.693 * (1000 + 1000) * 33 * 10^{-9} = 45.738\mu s$$

$$T_L = 0.693 * (1000) * 33 * 10^{-9} = 22.869\mu s$$

$$f = \frac{1}{(45.738 + 22.869) * 10^{-6}} = 14.576\ kHz$$

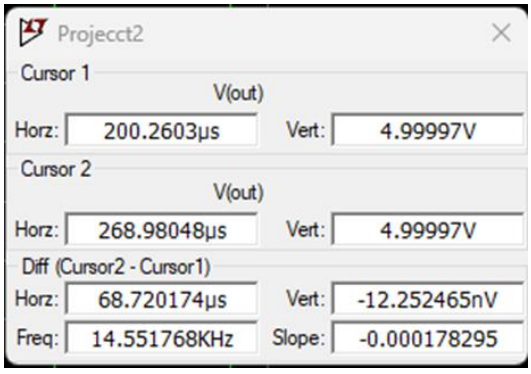

Figure 5. Calculation of Freq. in Simulation

As demonstrated above, there is a slight variation between calculated and simulated frequencies. The main reason for this is that there is no tool in LTspice that automatically calculates frequency, and cursors must be put manually.

B. Full Adder

To generate random numbers within your desired range, the dice roller project requires complete adders with an uneven seven-sided design. Every complete adder adds three bits to its input: the two main inputs (A and B), and the fixed carry-in bit (C0), which is set to '1). The carry-out bit shows any carry-over to the next higher bit position, whereas the total output is the outcome of this addition operation.

The result of this adding operation is known each time the dice are rolled, ensuring that the numbers produced are random and unexpected. In addition to boosting the system's unpredictability and fairness, the total output corresponds to the possible results of the dice roll by reflecting various numerical values in binary. Because any flaws may affect how randomly the numbers are generated, the dice roller's integrity and functionality are dependent on the entire adder system functioning properly. As a result, complete adders are critical components that maintain the unpredictability, integrity, and randomness of the created outputs, thus improving the project's overall performance.

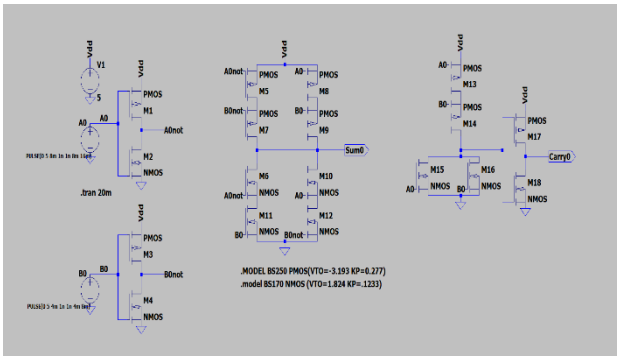

Figure 6. Schematic of the 1-bit full adder.

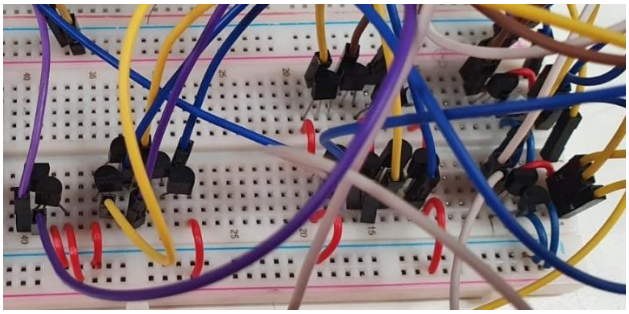

Figure 7. 1-bit full adder circuit as Cmos level.

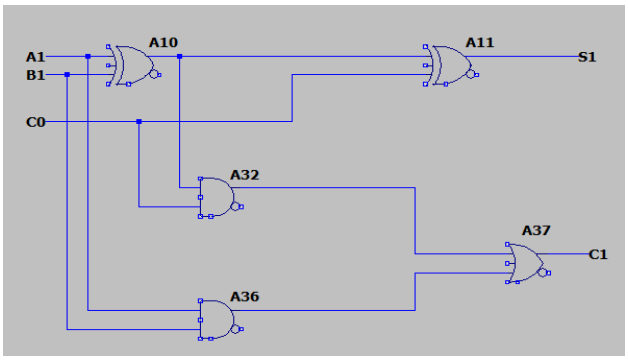

Figure 8. Schematic of the 2-bit full adder.

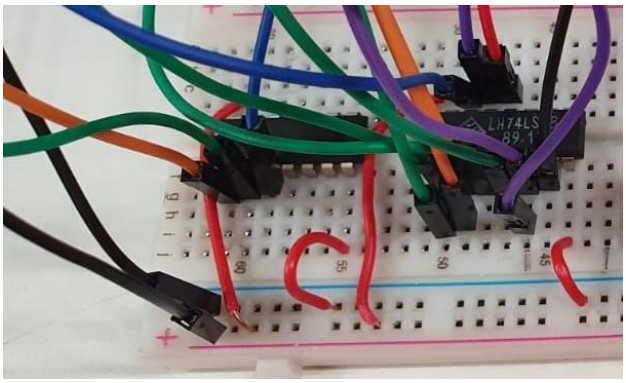

Figure 9. 2-bit full adder circuit as Gate level.

To obtain the desired performance, it is used that BS170 for NMOS and BS250 Mosfet devices for PMOS. The threshold and conductance voltage values of these MOSFETs were submitted to LTSpice, as illustrated in Figures 5-6 and 3.BS170 and BS250 are chosen based on their specifications and suitability for the required application.PMOS (BS250) and NMOS (BS170) work together to create an efficient CMOS (Complementary Metal-Oxide-Semiconductor) arrangement, which is widely used in digital logic design. The design strives to provide optimal switching behavior and overall circuit performance by using each type's distinct features, such as threshold voltage and transconductance.

Operation of the Full Adder Circuit

Sum Generation: Combining inputs A, B, and Cin yields the sum output (S). We can simplify the logic for the specific case in which Cin is always '1'. We observe that the XOR of A, B, and Cin yields the total output (S). This XOR function can be achieved in CMOS circuits by using NMOS transistors for the pull-down network and PMOS transistors for the pull-up network.

Carry-out generation is based on the majority function of A, B, and Cin[2]. When at least two inputs are '1', the carry-out is proclaimed. Cout will be '1' if at least two of A, B, or Cin are '1'; Cin is always '1'. A sequence of NAND gates can be used to detect situations in which two or more inputs are '1'.

Feedback system play a significant role in multi-bit addition operations. The carry-in for the next phase is fed back into the carry-out created in the current stage. However, because Cin is already fixed in our specific scenario—where Cin is always '1'—this feedback is not explicitly necessary.

Theoretical Truth Table:

This is a summary of the truth table for the one-bit full adder, assuming that the carry-in (C0) is set to '1':

Table 1 Truth Table of 1 Bit Adder

A0	B0	Cin0	Sum0	Carry0
0	0	1	1	0
0	0	1	1	0
0	1	1	0	1
0	1	1	0	1
1	0	1	0	1
1	0	1	0	1
1	1	1	1	1
1	1	1	1	1

Table 2. Truth Table of 2 Bit Full Adder

A1	B1	Cin1(Carry0)	Sum1	Cout1
0	0	0	0	0
0	0	0	0	0
0	1	1	0	1
0	1	1	0	1
1	0	1	0	1
1	0	1	0	1
1	1	1	1	1
1	1	1	1	1

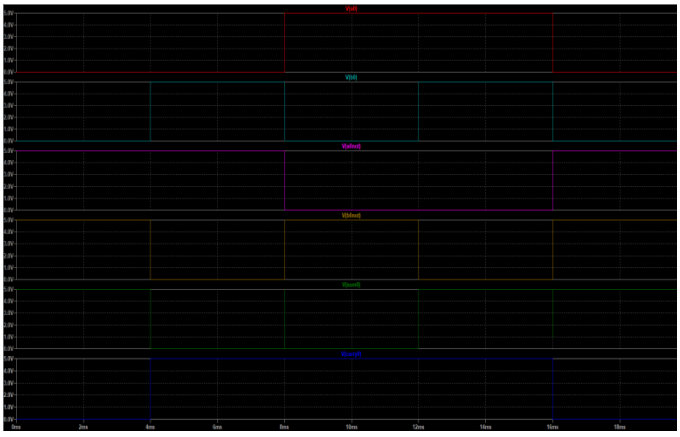

Figure 10. Graphs of Inputs (A0, B0) and Outputs(Sum0,Carry0) for 1-bit full adder.

Figures 7 and 8 exhibit graphs of inputs (A0, B0) and outputs (Sum0, Carry0) and inputs (A1, B1) and outputs (Sum1, Carry1), respectively, demonstrating the behavior of a 1-bit full adder. For a 1-bit full adder circuit, each graph shows the binary combinations of inputs (A, B) and outputs (Sum, Carry).

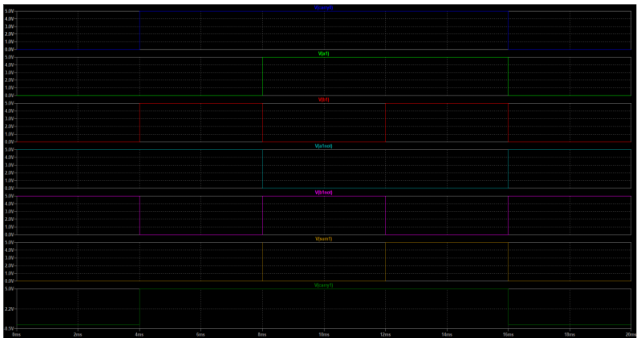

Figure 11. Graphs of Inputs (A1, B1) and Outputs(Sum1,Carry1) for 1-bit full adder.

Similarly, Figure 8 expands this representation to include the Carry1 and Sum1 outputs for the next two bit positions, A1 and B1. With Carry1 signaling any additional carry propagation in the addition process and Sum1 adding to the second bit of the sum, these outputs reflect the results of the addition operation for the next significant bit.

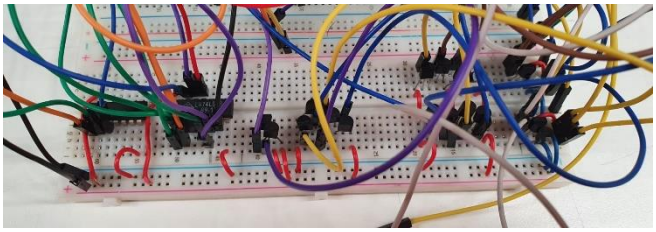

Figure 12. 2-bit Full Adder Circuit. +

C. Intermediate Register

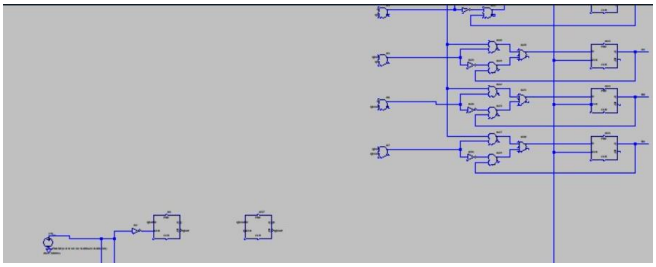

Figure 13. Intermediate Register Simülation of Entire System

In terms of the uneven seven-sided dice roller project:

The pulse generator generates random data (D0), which is then received by the intermediate register. This data is distributed to the proper inputs (A1, A0, B1, B0) of the full-adder circuit via timing signals (CLK1). The intermediate register (RI) guarantees that data is transferred precisely and on time, which contributes to the dice roller system's general functionality and reliability.

Function of the Intermediate Register (RI)

This project relies heavily on the Intermediate Register (RI), which provides temporary data storage. In particular, until more processing is needed, it accepts and stores the incoming data (D0) produced by the data generating circuit. Apart from its storing purpose, the RI is essential to the

managed transfer of information across different parts of the digital system.

It is in perfect time with timing signals (CLK1) that control the 2-bit full-adder circuit, coordinating the distribution of D0 to designated inputs (A1, A0, B1, B0). The Intermediate Register (RI) plays a critical role in this project by storing temporary data.

It accepts and stores the incoming data (D0) produced by the data generator circuit until further processing is necessary. In addition to its storing function, the RI is responsible for the regulated distribution of data throughout various sectors of the digital system.

It carefully synchronizes the distribution of D0 to specified inputs (A1, A0, B1, B0) of the 2-bit full-adder circuit with timing signals (CLK1) that regulate system operations.

D. Data Distribution

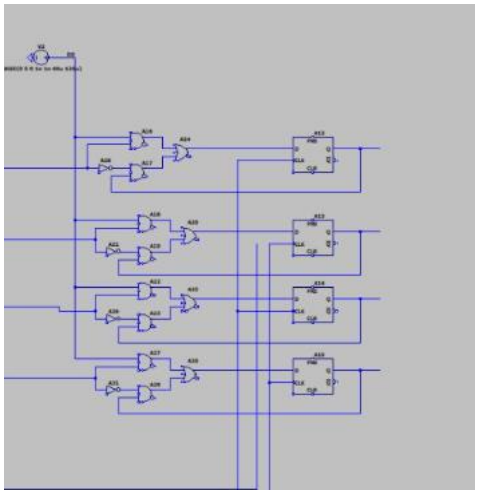

Figure 14. Simulation of Data Distribution.

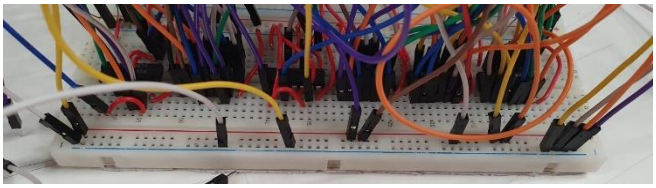

Figure 15. Data Distribution Circuit.

With the uneven seven-sided dice roller system, data distribution is accomplished by employing a clock signal (CLK1) produced by a microcontroller to direct the transfer of the generated pulse signal (D0) to particular registers (A1, A0, B1, B0) within the intermediate register (RI). This section describes the data distribution process's design, implementation, and scheduling issues.

Data Distribution Circuit Design

Based on the time supplied by CLK1, the data distribution circuit stores and transfers the D0 signal to specific bits of the intermediate register (RI) using flip-flops or other comparable sequential logic devices. Because the circuit design guarantees correct data transmission synchronization and sequencing, binary data may be accurately stored and managed inside the system.

Timing Diagram for Clock 1

Create a timing diagram that shows how CLK1, D0, and the data distribution circuit's outputs (A1, A0, B1, B0) are related to one another. Based on the rising edges of CLK1, the timing diagram should clearly show when each flip-flop samples the D0 signal, resulting in the proper distribution of data across the register bits.

ARDUINO IDE code for CLK

```
void setup() {
  pinMode(9, OUTPUT);        // Set Pin 9 as an output
  Serial.begin(115200);      // Start serial communication at 115200 baud rate
    // Configure Timer1
  TCCR1A = _BV(COM1A0);      // Toggle OC1A (Pin 9) on compare match
  TCCR1B = _BV(WGM12) | _BV(CS10);  // Set Timer to CTC mode and no prescaler
  OCR1A = 19999;             // Set the compare value to generate a 400 Hz frequency
}

void loop() {
  // Empty loop, no code to run continuously
}
```

This sketch sets up Timer1 to generate a 400 Hz square wave on Pin 9 via hardware timer interrupts, freeing up the main loop for other duties. The resulting waveform can be viewed on an oscilloscope or measured using a frequency counter. Adjust the OCR1A value to produce different frequencies by recalculating the compare value based on the desired frequency and system clock speed.

Figure 16. Oscilloscope image of Clk1 and Clk4 which are connected Ch1 and Ch2 in order.

E. Output Register(R0)

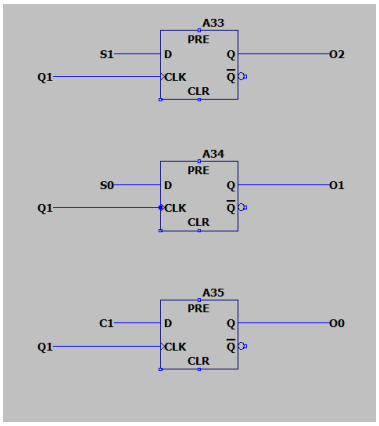

Figure 17. Simulation of Output Register.

As seen in figure 15, it was obtained from the 100 Hz clock counter circuit and connected to the input of the d flipflop to produce C0, S1, S0 O2, O1, O0 respectively. In this way, the outputs of the flip flop held O2, O1, O0 data.

F. Full System Of Circuit Design

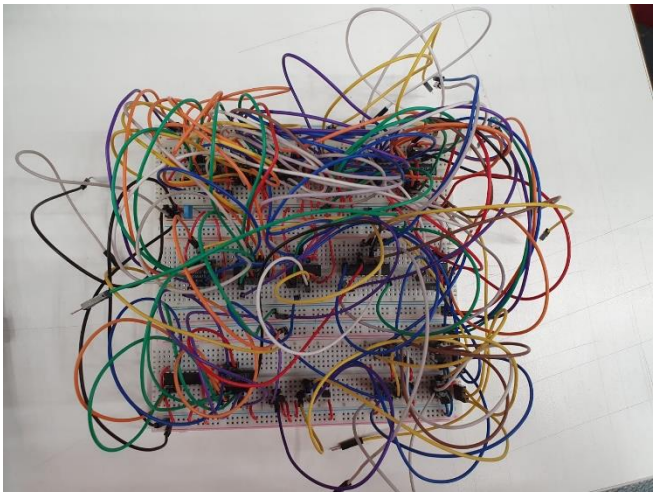

Figure 17. Combined Circuit.

To finalize the circuit, it was created by combining the Data (Do) generator circuit, Signals A and B Generator circuit (Intermadiate register R1), Full adder circuit and Output Register (R0) circuits respectively. The produced Do signal was obtained with the established Intermadiate Register R1 circuit, and two 2-bit signals were obtained, and after these signals were sent to the full adder circuit, the actual outputs were recorded with the Output Register circuits.

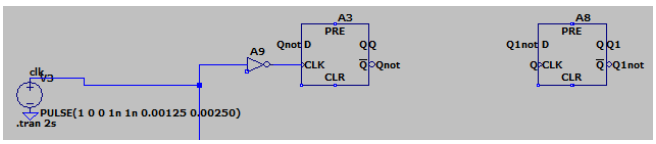

Figure 18. Counter Circuit for Clk4 on LTSpice.

The counter circuit was obtained by serially connecting 2 D flip flops according to the working principles of frequency dividers. Since the final output of the counter circuit will be one fourth of the given first clock frequency[3], Clk4 was obtained as 100 Hz on Q2 output in this way.

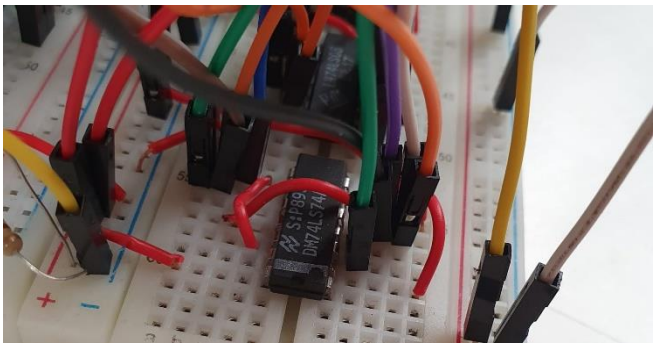

Figure 19. Counter Circuit for Clk4 on Breadboard.

Since 74ls74 contains 2 independent d flip flops, the output from pin 9 gives Q2 output.

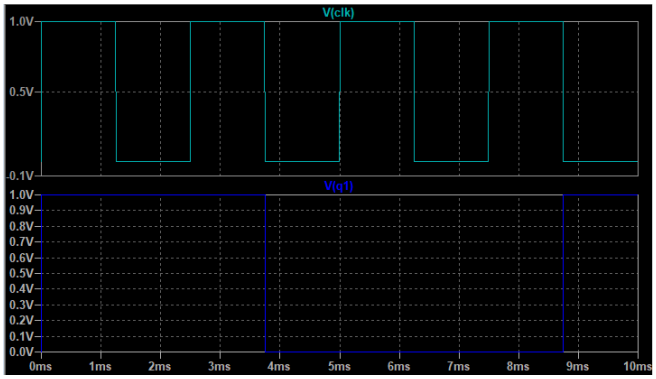

Figure 20. Simulation result of Clk1 and Clk4.

Figure 21. Timing diagram of D0, Clk1, Clk4, A1, A0, B1, B0, X2, X1, X0, O2, O1, O0.

While reading the circuit outputs via Matlab with Arduino, we experienced problems related to sampling because we were stuck at the limits of Arduino[4]. This caused the timing diagram to look different than it actually is.

```
-> Probability of 1: 15.02%
-> Probability of 2: 23.77%
-> Probability of 3: 37.67%
-> Probability of 4: 6.20%
-> Probability of 5: 2.20%
-> Probability of 6: 5.97%
-> Probability of 7: 9.17%
```

Figure 22. Probability of numbers

As seen above, the total probability is equal to 1. This is the biggest proof that the circuit is working correctly. because the circuit should never produce the value "000", that is, 0, in line with the given task.

CONCLUSION

The report's conclusion focuses on the methodical approach used to build a circuit for generating random integers with uneven probabilities. The project's objective is to develop effective dice rolling mechanism by concentrating on developing a pulse generator, making use of a custom full-adder circuit, and including a microcontroller interface. The focus on simulation, circuit design, and practical application highlights the dedication to attaining statistically equitable results for every dice roll.

REFERENCES

[1] Atwell, C. (2023, December 18). The origin, explanation, and applications of Triple-Five Timers. Electronic Design. https://www.electronicdesign.com/technologies/analog/article/21252714/electronic-design-the-origin-explanation-and-applications-of-triple-five-timers

[2] GeeksforGeeks. (2023, August 7). Full Adder in digital logic. GeeksforGeeks. https://www.geeksforgeeks.org/full-adder-in-digital-logic/

[3] Libretexts. (2021, February 1). 6.8: Frequency divider. Engineering LibreTexts. https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electronics/Microwave_and_RF_Design_IV%3A_Modules_(Steer)/06%3A_Mixer_and_Source_Modules/6.08%3A_Frequency_Divider

[4] How high of a baud rate can I go (without errors)? (n.d.). Arduino Stack Exchange. https://arduino.stackexchange.com/questions/296/how-high-of-a-baud-rate-can-i-go-without-errors#:~:text=The%20Arduino%20Serial%20Monitor%20window,the%20highest%20baud%20rate%20capable