Semester Project in Digital circuit design: One-Dimensional Ping-Pong

Mykola Balyk
First-year Computer Science Student
Ukrainian Catholic University
Lviv, Ukraine
balyk.pn@ucu.edu.ua

***Abstract*—A hardware-only 1D "Ping-Pong" game is implemented using the GreenPAK SLG46620V. A shifting bit across six LEDs simulates the ball, with player input reflecting it at the edges. The design uses a clock divider, 6-bit shift register, direction control, and output logic—entirely without a microcontroller. This demonstrates the potential of configurable logic for simple, low-power interactive systems.**

***Index Terms*—GreenPAK, SLG46620V, digital logic, one-dimensional Ping-Pong, programmable logic, hardware game implementation, counters, logic elements.**

I. Introduction

This project implements a one-dimensional "Ping-Pong" game entirely in hardware using the GreenPAK SLG46620V programmable logic device. The game logic is built without any microcontroller or software, relying solely on configurable logic blocks.

A single active bit, representing the ball, moves across a 6-LED array. When the bit reaches either edge (Q0 or Q5), the player must press a button within a brief input window to reflect it. A successful hit triggers a buzzer and increases the game difficulty by doubling the ball's speed every two successful reflections. If the input is missed, the ball starts from another edge, giving you an opportunity to deflect it again.

The aim of this project is to explore the use of purely hardware-based logic to implement real-time interactive behavior, highlighting the capabilities of the GreenPAK platform for low-power embedded designs.

This paper describes the game architecture, implementation details, and core logic blocks involved in the system.

Summary:

This chapter introduced the project's goal: a hardware-only interactive game using GreenPAK. It outlined the gameplay concept, motivation, and key implementation constraints — no microcontroller, only configurable logic.

II. GreenPAK Configuration Overview

The SLG46620V is a dual-matrix programmable logic device, consisting of two interconnected logic arrays (Matrix0 and Matrix1). These matrices are connected via internal interconnects that allow routing of signals and sharing of functional blocks across both halves of the chip.

This project received no external funding.

In this design, Matrix0 handles the user input, clock division, bounce detection logic, and buzzer control. Matrix1 contains the core 6-bit shift register responsible for ball movement and directly drives the LED outputs without routing through Matrix0.

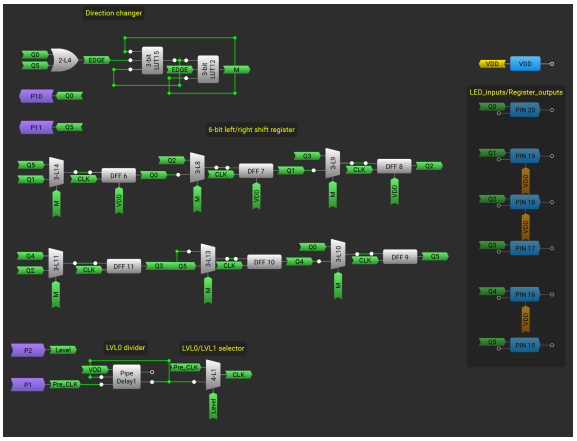

Fig. 1. Configuration of Matrix1 — main game logic. [1]

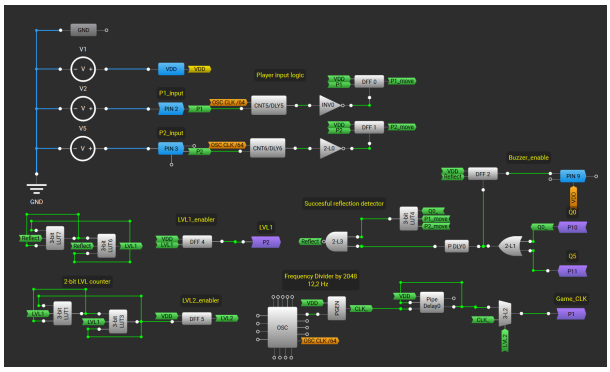

Fig. 2. Configuration of Matrix0 — clock and support logic. [2]

TABLE I
GAME I/O PIN ASSIGNMENTS

Signal	Function	Pin
Q0	Leftmost LED	20
Q1	2nd LED	19
Q2	3rd LED	18
Q3	4th LED	17
Q4	5th LED	16
Q5	Rightmost LED	15
Player1	Left button input	2
Player2	Right button input	3
Buzzer	activate buzzer on reflection	10

Summary:

This section described how the SLG46620V's dual-matrix structure was partitioned: Matrix0 manages inputs and auxiliary logic, while Matrix1 implements core game mechanics and LED driving. Key I/O pin assignments were also listed.

III. BALL MOVEMENT LOGIC

The ball movement is implemented using a 6-bit shift register built from DFF elements. One of the flip-flops is initialized with logic high ('1'), representing the ball. This bit shifts between registers in either direction based on the current direction signal.

The direction logic is implemented via 3-bit LUTs configured as multiplexers, selecting whether to shift left or right.

Each output Q_0-Q_5 is routed directly to LED output pins 20-15.

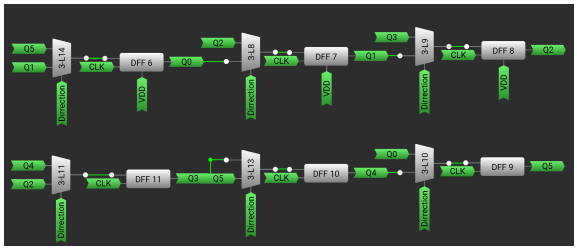

Fig. 3. Shift register implementation using DFFs and direction control via LUT-based multiplexers. [3]

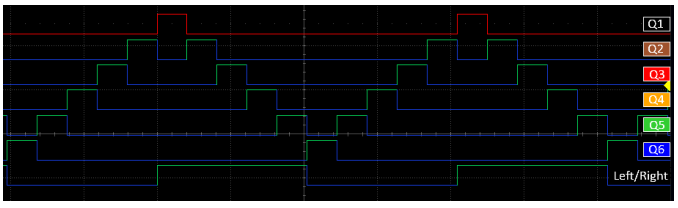

Fig. 4. Register outputs during series of reflections. [4]

Summary:

This section detailed the implementation of ball motion using a shift register with DFFs, guided by direction logic via LUT-based multiplexers. The ball is represented by a single high bit and is visible through direct LED output.

IV. BOUNDARY DETECTION AND DIRECTION CONTROL

Direction switching is triggered only upon a successful reflection. The reflection conditions activate a pair of 3-input LUTs configured to emulate a T flip-flop, which toggles the shift direction. If the ball is not reflected in time, it continues moving in the same direction from the opposite edge, effectively wrapping around.

Two LUTs — LUT15 and LUT12 — are used in this implementation:

- LUT15 receives as inputs: IN0 – output of LUT12 IN1 – OR(Q_0, Q_5) IN2 – its own previous output (for state holding)
- LUT12 receives as inputs: IN2 – output of LUT15 IN1 – OR(Q_0, Q_5) IN0 – its own previous output

These configurations enable the flip-flop to toggle only when the ball is reflected.

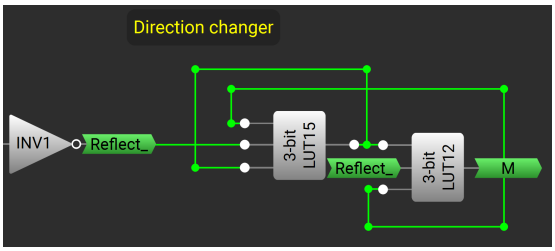

Fig. 5. LUT15 and LUT12 connection diagram implementing T flip-flop logic. [5]

LUT15 Configuration

Inputs: IN0 – $LUT12_{out}$, IN1 – OR, IN2 – $LUT15_{out}$

TABLE II
LUT15 TRUTH TABLE

IN2	IN1	IN0	OUT
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

LUT12 Configuration

Inputs: IN2 – $LUT15_{out}$, IN1 – OR, IN0 – 12_{out}

TABLE III
LUT12 TRUTH TABLE

IN2	IN1	IN0	OUT
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Summary:

The direction control is handled by two interconnected LUTs forming a T flip-flop. It toggles only on valid bounce signals, ensuring direction switches only on correct input. Truth tables define the LUT configuration.

V. Bounce Detection Logic

When a player presses their button, a DFF sets its output high, opening a short detection window (100 ms) during which a ball hit is considered valid. This window is defined by a CNT DLY block in Delay mode. Once the delay elapses, its Q output is inverted and sent to the DFF's nRST input, resetting the window.

The outermost ball positions (Q0 and Q5) are OR-combined and sent to a pulse edge detector, which generates a short pulse whenever the ball reaches either boundary. This pulse is further AND-ed with the player bounce window signal to determine whether a valid reflection occurred.

Direction switching is enabled only if the boundary pulse coincides with a valid player input. Otherwise, the ball wraps around and continues moving in the same direction.

A 3-input LUT is used to validate the bounce condition. Its configuration is shown in Table IV. The LUT receives as inputs:

- IN2: Q_0 (left boundary),
- IN1: $P1_{window}$ (player 1's valid interval),
- IN0: $P2_{window}$ (player 2's valid interval).

TABLE IV
Bounce Detection LUT Configuration

IN2 (Q0)	IN1 (P1_window)	IN0 (P2_window)	OUT (Reflected)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

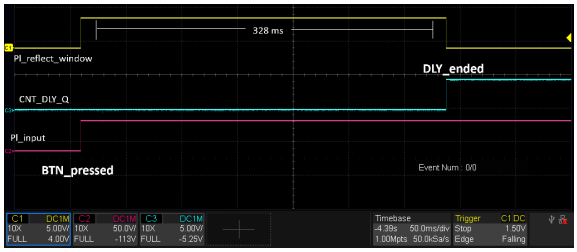

Fig. 6. Timing diagram of player input and reflection detection window(it was set to 328 ms for testing purposes). [6]

Summary:

Bounce detection relies on a timed input window and edge-triggered ball position detection. A 3-input LUT validates reflections by combining boundary presence with player input timing, ensuring direction changes occur only on correct interaction.

VI. Clock Generation and Level Control

This section describes two key subsystems: (A) clock generation and division used to control the ball speed, and (B) logic for tracking successful hits and switching difficulty levels.

A. Clock Generation and Frequency Division

The game uses the internal 25 kHz RC oscillator of the SLG46620V. The oscillator output passes through a chain of frequency dividers to reduce the frequency to a suitable range for gameplay. The complete division pipeline is:

$$f_1 = \frac{f_{osc}}{64} = \frac{25\,000}{64} = 390.625\,\text{Hz} \tag{1}$$

$$f_2 = \frac{f_1}{8} = \frac{390.625}{8} = 48.828\,\text{Hz} \tag{2}$$

$$f_3 = \frac{f_2}{4} = \frac{48.828}{4} = 12.207\,\text{Hz} \tag{3}$$

Optional Pipe Delay blocks halve the frequency successively:

$$f_4 = \frac{f_3}{2} = \frac{12.207}{2} = 6.103\,\text{Hz} \tag{4}$$

$$f_5 = \frac{f_4}{2} = \frac{6.103}{2} = 3.052\,\text{Hz} \tag{5}$$

A multiplexer selects between the raw or delayed clock path depending on the current game level. Pipe delays are enabled or bypassed accordingly.

B. Difficulty Level Logic

A 2-bit ripple counter tracks the number of successful ball reflections. After combinations DFFs, which control levels, are permanently set. After every two successful reflections, increment the counter and change the clock source by adjusting the MUX configuration.

TABLE V
Game Level Clock Mapping

Clock Path	**Frequency (Hz)**
Direct (no delay)	$f_3 = 12.207$
Pipe Delay 1	$f_4 = 6.103$
Pipe Delay 2	$f_5 = 3.052$

This progressively increases the game speed, increasing difficulty.

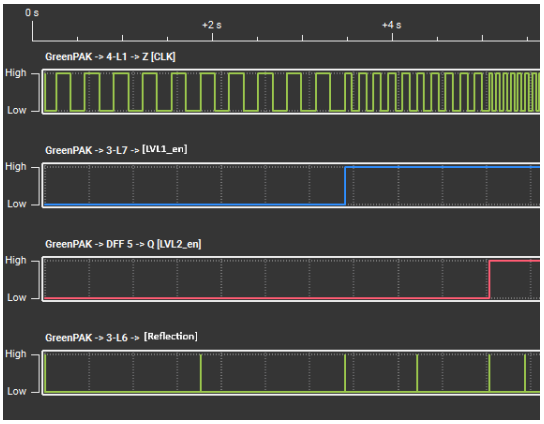

Fig. 7. Timing diagram of levels changing [7]

Summary: Through layered hardware division and level tracking, the game introduces progressively faster ball movement entirely in logic.

VII. Game Feedback System

The feedback system provides an audio signal via a buzzer to indicate a successful ball hit. The logic is structured as follows: when the hit detection logic asserts a signal (set by a DFF), the buzzer is enabled. This signal persists until the ball leaves the edge position (Q0 or Q5), at which point the DFF is asynchronously reset.

The buzzer draws less than 32 mA of current, which exceeds the capability of standard output pins, but is within safe limits for a pin configured as Open-Drain 4x, capable of sourcing up to 46 mA.

TABLE VI
Truth Table for Buzzer Activation

Q0	Q5	Hit Detected (DFF)	Buzzer Output
0	0	0	0
1	0	1	1
0	1	1	1
1	0	0	0
0	1	0	0
0	0	1	0 (reset)

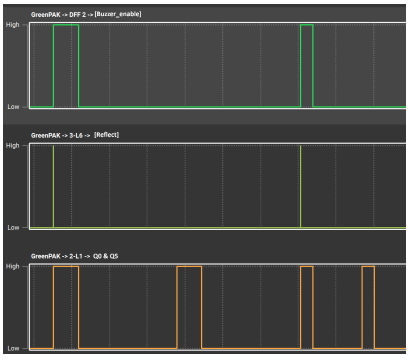

Fig. 8. Timing diagram showing DFF behavior and buzzer control with respect to edge states. [8]

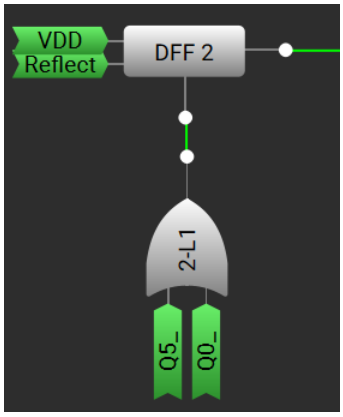

Fig. 9. Logical connection of hit detector, buzzer control and edge detection. [9]

Summary:

A dedicated DFF-based control path enables the buzzer on valid hits and deactivates it once the ball exits the boundary. The Open-Drain 4× output mode safely supports the required current for the audio indicator.

VIII. Resource Utilization

Due to the limited number of configurable elements in the GreenPAK SLG46620V, efficient usage of each resource is critical to implementing even moderately complex logic systems. The table below summarizes the allocation of key resources in this project.

TABLE VII
GreenPAK Resource Usage Summary

Resource	Used	Available	Notes
Invertors	2	2	Changing CNT DLY output
2-bit LUT	3	9	Invertors, OR-ing and AND-ing
3-bit LUT	14	16	Direction logic, MUX, T-flip-flop
4-bit LUT	1	1	MUX
4bit LUT/PGEN	1	1	Clock division
DFF	12	12	Register, counters, data storage
CNT/DLY	2	10	User input delaying
Oscillator (RC)	1	1	25 kHz system clock
Pipe Delay	2	2	Used as clock divider

One of the primary challenges during implementation was the shortage of DFFs needed for dividers, counters, and T-triggers. I overcame this using a combination of LUTs and pipe delay blocks — demonstrating GreenPAK's flexibility in resource reuse and logic optimization.

Summary:

The design exhausts all DFFs and nearly all 3-bit LUTs, highlighting the importance of creative resource reuse. Key functions such as timing, direction control, and level switching are implemented using minimal elements through clever multiplexing and delay structures.

IX. Connection Scheme

To ensure stable power delivery to the GreenPAK SLG46620V device, a 0.1 µF decoupling capacitor is placed

between the VDD and GND pins, located as close as possible to the chip. This capacitor filters out high-frequency noise and stabilizes the local voltage supply.

Each player input button is connected to a digital input pin of the GreenPAK device. To ensure defined logic levels when the button is not pressed, 10 kOhm pull-down resistors are used. These resistors connect the input lines to ground, preventing floating states and avoiding false triggers.

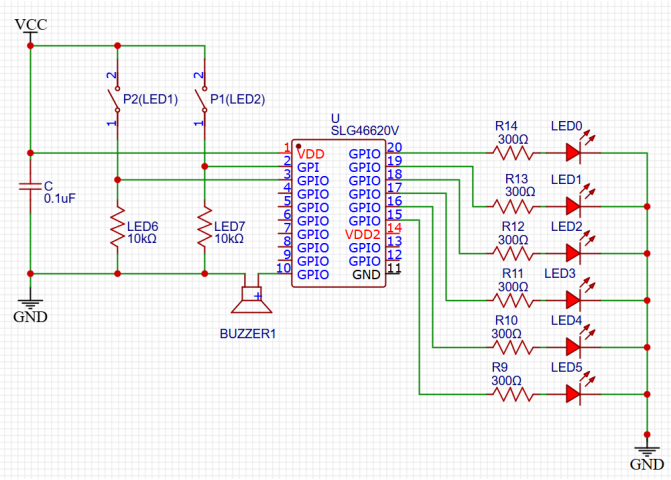

Fig. 10. Full circuit wiring diagram. [10]

Summary: Proper decoupling and input conditioning ensure reliable operation and minimize noise-induced errors.

Conclusion

The main objective of this project — to design and implement a working logic circuit for a Pong-style game — was fully achieved. I successfully created a functional one-dimensional Pong system, complete with player-controlled bounce detection, audio feedback, and a progressive difficulty mechanism. All features were implemented exclusively through GreenPAK logic resources.

However, the path to completion was not without challenges. Initially, I aimed to replicate the full two-dimensional Pong game, but quickly realized that such a goal was overly ambitious given the limited resources and strict deadlines. After several weeks of continuous work, I made the strategic decision to shift focus to a simpler, yet complete, 1D version of the game. This shift allowed me not only to achieve a functioning design but also to refine and polish the smaller details — an essential aspect of any successful engineering project.

The most important lesson I've learned is that setting clear, realistic goals — and completing them to a high standard — is far more effective than overcommitting to overly complex ideas. Through this process, I also discovered just how much logic can be packed into the GreenPAK device with careful planning and resource optimization. Even with limited elements like DFFs and LUTs, the flexibility of the platform — especially when working across two matrices — opens the door to surprisingly sophisticated designs.

This project has strengthened my understanding of digital logic, sharpened my problem-solving skills, and given me a deeper appreciation for iterative design.

Acknowledgment

I would like to express my heartfelt thanks to my lecturer, Sapiha Oleh, for his guidance and insightful feedback throughout this project. I am also grateful to assistant, Babenko Alina, for her constant support and motivation. I want to thank everyone who stood by me, offering encouragement and help during the development of this project.

References

[1] Configuration of Matrix1 — main game logic. Figure 1
[2] Configuration of Matrix0 — clock and support logic. Figure 2
[3] Shift register implementation using DFFs and direction control via LUT-based multiplexers. Figure 3
[4] Register outputs during series of reflections. Figure 4
[5] LUT15 and LUT12 connection diagram implementing T flip-flop logic. Figure 5
[6] Timing diagram of player input and reflection detection window(it was set to 328 ms for testing purposes). Figure 6
[7] Timing diagram of levels changing. Figure 7
[8] Timing diagram showing DFF behavior and buzzer control with respect to edge states. Figure 8
[9] Logical connection of hit detector, buzzer control and edge detection. Figure 9
[10] Full circuit wiring diagram. Figure 10
[11] Official SLG46620V Documentation. https://www.renesas.com/en/document/dst/slg46620-datasheet?r=1563691 s
[12] Go Configure™ Software Hub User Guide. https://www.renesas.com/en/document/mat/go-configure-software-hub-user-guide?r=1572736
[13] Information about Left/Right Shift Register. https://www.uoanbar.edu.iq/eStoreImages/Bank/5817.pdf