Performance Comparison Between Polynomial Function and Lookup Table (LUT)

Carlos Ignacio Elias Contreras
Prof: Luis Emilio Tonix Gleason

September 16, 2025

1 Introduction

This report details the development and results of the second part of the microprocessor lab. The main objective was to optimize the calculation of a polynomial function on an ESP32 microcontroller. The task was divided into two phases: the first, already completed, involved using a Python script to perform a linear regression on ADC data from the ESP32 to obtain the coefficients of a quadratic function. The second phase, the subject of this report, consisted of implementing a lookup table (LUT) for the same calculation and comparing its runtime performance against the direct function.

2 Methodology

The following tools and components were used for this practice:

- **ESP32 Microcontroller**: The hardware platform for code execution.
- **Jupyter Notebook with Python**: The development environment used to generate the lookup table and the header file (`.h`).
- **C Programming Language**: For the code that runs on the ESP32.

The process was developed in the following stages:

2.1 Lookup Table (LUT) Generation

Using a script in Jupyter Notebook, a lookup table was generated from the polynomial function obtained in the previous task:

$$f(x) = -5.79212877 \times 10^{-6}x^2 + 4.93648274 \times 10^{-2}x$$

The script calculated the function's value for every possible ADC input, ranging from 0 to $2^{12} - 1 = 4095$. The results were stored in a list, which was then exported to a header file named `lookuptable.h`. This file contains a static integer array, optimized to be included in the ESP32's C code.

2.1.1 Python Code for Generation

```python
import sympy as s
import numpy as np
from IPython.display import display
x = s.Symbol('x')
poly = -5.79212877e-06*x**2 + 4.93648274e-02*x
display(poly)
sp = s.lambdify(x, poly, 'numpy')
N = 2**12
display(N)
lut_size = N
lookup = [ int ( poly . subs (x , i ) ) for i in range ( lut_size ) ]

with open("lookuptable.h", "w") as f:
    f.write("#ifndef LOOKUPTABLE_H\n")
    f.write("#define LOOKUPTABLE_H\n\n")
    f.write(f"#define LUT_SIZE {lut_size}\n")
    f.write("static const int lookup_table[LUT_SIZE] = {\n")
    for i, val in enumerate(lookup):
        f.write(f"    {val}, ")
        if (i + 1) % 8 == 0:
            f.write("\n")
    f.write("\n};\n\n")
    f.write("#endif // LOOKUPTABLE_H\n")

print("The 'lookuptable.h' file has been successfully generated.")
```

Listing 1: Python script to generate the LUT and the header file.

2.2 Implementation and Measurement on the ESP32

The `lookuptable.h` file was included in a C project for the ESP32. A program was developed to perform the following tasks for a test ADC value:

1. **Calculation by Direct Function**: The execution time for the direct polynomial function calculation was measured.
2. **Calculation by Lookup Table (LUT)**: The execution time for looking up the value in the lookup table, using the test value as the index, was measured.

The execution time was measured in microseconds (µs) using the `esp_timer_get_time()` function.

2.2.1 C Code for Measurement

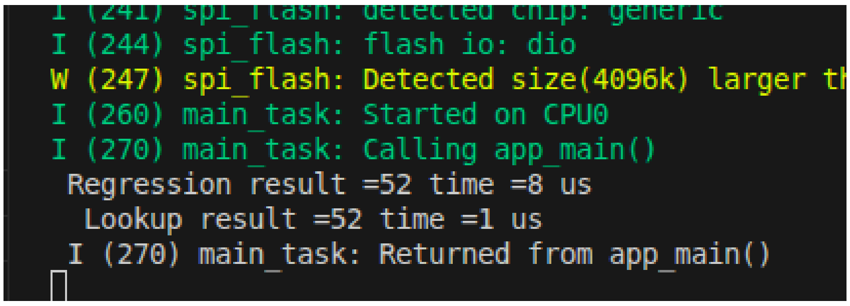
```c
# include "esp_timer.h"
# include "lookuptable.h"
# include <stdio.h>
#include <stdint.h>

int regression_func ( int x ) {
    // Polynomial approximation
    return ( int ) (-5.79212877e-06* x * x + 4.93648274e-02* x ) ;
}
```

```c

void app_main ( void ) {
    int test_val = 1234;
    int result ;

    // Regression timing
    int64_t start = esp_timer_get_time () ;
    result = regression_func ( test_val ) ;
    int64_t end = esp_timer_get_time () ;
    printf ( "Regression result = %d, time = %lld us\n", result , (end -
    start) ) ;

    // Lookup timing
    start = esp_timer_get_time () ;
    result = lookup_table [ test_val ];
    end = esp_timer_get_time () ;
    printf ( "Lookup result = %d, time = %lld us\n", result , (end -
   start) ) ;
}
```

Listing 2: C code to measure execution time.

3 Results and Analysis

Upon executing the code on the ESP32, the following results were obtained on the serial monitor:



```
I (241) spi_flash: detected chip: generic
I (244) spi_flash: flash io: dio
W (247) spi_flash: Detected size(4096k) larger th
I (260) main_task: Started on CPU0
I (270) main_task: Calling app_main()
 Regression result =52 time =8 us
  Lookup result =52 time =1 us
 I (270) main_task: Returned from app_main()
```

Figure 1: Results of the calculation

The analysis of the results reveals that the **lookup operation in the table is significantly faster** than the execution of the mathematical function. Although the direct function was extremely fast (8 µs), the LUT required (1 µs). In contrast, calculating the polynomial function involves multiple floating-point operations (multiplications, subtractions), which are more expensive in terms of CPU clock cycles.

4 Conclusion

The implementation of a lookup table (LUT) proved to be a highly effective optimization strategy for function calculation on the ESP32 microcontroller. Even though the direct polynomial function is relatively simple, access to the LUT offered superior performance. This technique is especially useful in applications that require real-time response and where results can be pre-calculated, such as in control systems or signal processing.