Practical 2

TSHDAV008[†] and SMBTHA002[‡]
EEE3096S
University of Cape Town
South Africa

***Abstract*—This experiment aims to gain a better understanding on various optimization methods in embedded systems**

I. Introduction

The objective of this experiment is to gain a better understanding on various optimization methods in embedded systems. These various optimization methodologies will be implemented in this experiment to improve the testing programs launch speed. The different methods include using various compiler flags, bit widths and threading. Through running tests using each of these methods it will provide an insight into how these methods achieve optimization and their efficacy. Using the results of from the experimentation the most efficient combination of optimization methods will be determined to achieve the fastest program launch speed.

II. Methodology

A. Hardware

Raspberry Pi Zero W

B. Implementation

The first optimization method is multi-threading. Compile the threaded version of testing program called CHeterodyning threaded.c and

modifying the number of threads in the associated library file CHeterodyning threaded.h.

```
#define Thread_Count 1 //Change number of Threads here
```

Fig. 1

The second optimization method is achieved by using different compiler flags to improve program launch speed. Different compiler flags were added to the make file of the unthreaded testing program CHeterodyning.c .

```
3  CFLAGS = -lm -lrt //Add compiler Flags here
```

Fig. 2

The final optimization method is achived by using different bit widths. Change the bit size of the carrier and data variable in the file global.h

and the data,carrier and result variables in the unthreaded version of the program.

```
float carrier[SAMPLE_COUNT]
float data[SAMPLE_COUNT]
```

Fig. 3

```
extern float data [SAMPLE_COUNT];
extern float carrier[SAMPLE_COUNT];

float result [SAMPLE_COUNT];
```

Fig. 4

C. Experiment Procedure

Run the program using various bit widths,compiler flags and number of threads. After each change is made recompile source files after editing using makefile. Start by running test using different number of threads then using different compiler flags and lastly using various it widths.

III. Results

A. Figures

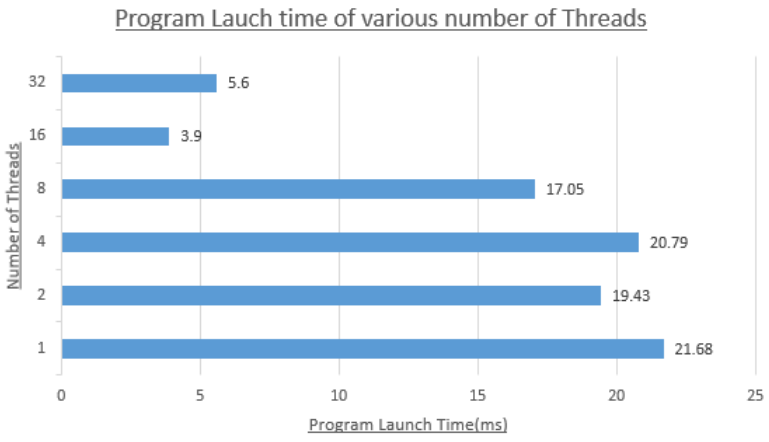

Fig. 5: Graph showing results of Program launch times using various amount of threads

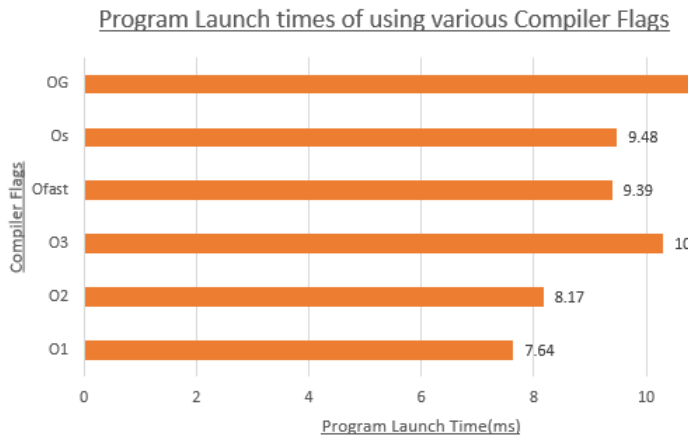

Fig. 6: Graph showing results of Program launch times using various Compiler Flags

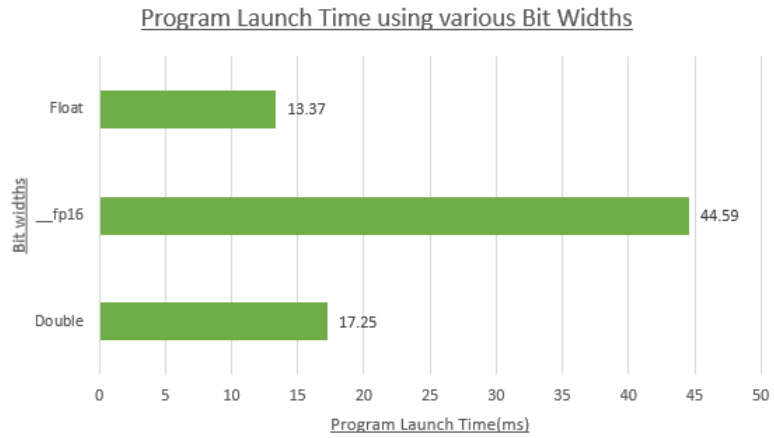

Fig. 7: Graph showing results of Program launch times at various Bit Widths

IV. Conclusion

This experiment was used to improve the understanding of optimizing embedded systems and determining which combination of optimization methods yield the fastest launch speed. The program ran fastest using multi-threading with 16 cores.

References

Overleaf.com. 2021. *Bibliography management in LaTeX - Overleaf, Online LaTeX Editor*. [online] Available at: <https://www.overleaf.com/learn/latex/Bibliography_management_in_LaTeX> [Accessed 29 August 2021].

Wiki.ee.uct.ac.za. 2021. *Category:RaspberryPi - UCT EE Wiki*. [online] Available at: <http://wiki.ee.uct.ac.za/Category:RaspberryPi> [Accessed 29 August 2021].

Wiki.ee.uct.ac.za. 2021. *LaTeX - UCT EE Wiki*. [online] Available at: <http://wiki.ee.uct.ac.za/LaTeX> [Accessed 29 August 2021].

Wiki.ee.uct.ac.za. 2021. *UCT EE Wiki*. [online] Available at: <http://wiki.ee.uct.ac.za/Main_Page> [Accessed 28 August 2021].