Loop Unrolling Standard Deviation

Software Modules

ABSTRACT

This routine implements an unrolled 16-bit 'mean of difference' standard deviation calculation. It allows the developer to calculate the standard deviation for an input of 16-bit values. This calculation uses loop unrolling in the average calculation and subsequent mean of difference calculation to optimize performance.

Contents

1 Introduction

This routine is designed to help the developer save time on calculating standard deviation. It uses the method of unrolling, instead of using a method of iterative loops. In other words, it buffers eight numbers at the same time.

1.1 About Standard Deviation

Standard deviation is a measurement of variability used in statistics. It helps to show how much variation there is from an average value.

1.2 About Loop Unrolling

Loop unrolling is a technique used to optimize program executions when summing a function. It reduces the instructions to be run by running certain instructions in parallel. Loop unrolling has several advantages, including significant reduction in execution time and the potential to be implemented dynamically.

2 Software Interface

All software was written in C. This routine uses loop unrolling to provide optimized average value and standard deviation calculations.

2.1 Source Code

The archive for this software contains all the necessary header files to enable the code and run the functions: stddev.c and stddev.h

2.2 Software Flow

The program accepts 16-bit values, and then uses loop unrolling and buffers to average eight values at a time. It then uses the sum to calculate the mean of difference.

2.3 Header File – 'stddev.h'

```
extern uint16_t
```

```
avg16 ( uint16_t *buf, uint16_t size );

extern uint16_t
stdev16 ( uint16_t *buf, uint16_t size, uint16_t avg );
```

2.4 Source Code – 'stddev.c'

```
/*
** Implements an unrolled 16-bit 'mean of difference' standard deviation.
*/
#include <stdint.h>
#include "stddev.h"



/*
** Calculates a 16-bit average for a 16-bit data set.
*/
uint16_t
avg16 ( uint16_t *buf, uint16_t size )
{
  uint16_t cnt = 0;
  uint32_t accum = 0;

  cnt = size;

  /* calc the average (unrolled loop for speed) */
  while( cnt >= 8 ) {
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
    accum += buf[--cnt];
  }

  /* calc the average */
  while( cnt-- )
    accum += buf[cnt];

  return (accum/size);
}



/*
** Calculates a 16-bit 'mean of difference' standard deviation.
*/
uint16_t
stddev16 ( uint16_t *buf, uint16_t size, uint16_t avg )
{
```

```
  uint16_t cnt = 0;
  uint32_t accum = 0;

  cnt = size;

  /* calc the average of the differences (unrolled loop for speed) */
  while( cnt >= 8 ) {
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
    cnt--; accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);
  }

  /* calc the average of the differences */
  while( cnt-- )
    accum += ((buf[cnt] > avg) ? buf[cnt]-avg : avg-buf[cnt]);

  /* return the average difference */
  return (accum/size);
}
```

3 About Simma Software, the SAE J1939 and UDS Experts

Simma Software, Inc. specializes in real-time embedded software for the automotive industry. Products and services include protocol stacks, bootloaders, device drivers, training, and consultation on the following technologies: J1939, CAN, CAN FD, J1587, J1708, J2497, J1922, J1979, ISO 15765, OBD-II, CANopen, UDS, XCP, NMEA2000, and Secure Boot.