Application Note: AN10000

How to perform 64 bit arithmetic operations

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to perform 64 bit arithmetic operations.

Required tools and libraries

This application note is based on the following components:

- xTIMEcomposer Tools - Version 14.0.0

Required hardware

Programming how-tos are generally not specific to any particular hardware and can usually run on all XMOS devices. See the contents of the note for full details.

1 How to perform 64 bit arithmetic operations

The natural word length on the xCORE is 32 bits and it is usually fastest to operate with 32 bit integer types (e.g. int or long). For some algorithms wider integer types may be required. To address this the C and XC compilers provide the long long integer type which is 64 bits wide. You can declare variables of type long long as follows:

```
long long x; // Signed 64 bit integer.
unsigned long long y; // Unsigned 64 bit integer.
```

A decimal integer constant that doesn't fit in a long will have type long long. You can force a constant to have long long type using the LL and ULL suffixes:

```
f(123LL); // Constant has type long long.
g(42ULL); // Constant has type unsigned long long.
```

The xCORE provides a number of instructions intended to improve the performance of multi-word arithmetic. The compiler targets these instructions when 64 bit arithmetic operations are used. For example consider 64 bit addition:

```
long long add64(long long b, long long c) {
  long long a = b + c;
  return a;
}
```

Without any special support for multi-word arithmetic this operation would require 5 instructions. By using making use of ladd instruction (long add) which has carry-in and carry-out operands the compiler can reduce the number of xCORE instructions required to 2.

The xCORE also provides instructions to improve the performance of multi-word addition, subtraction, multiplication and division.