Application Note: AN10035

How to target the the xCORE multiply-accumulate instruction

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to target the the xCORE multiply-accumulate instruction.

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 target the the xCORE multiply-accumulate instruction

Multiply-accumulate is an operation that computes the product of two numbers and adds the result to an accumulator. This is a particularly important operation in many digital signal processing algorithms.

The xCORE provides multiply accumulate instructions that multiply two 32 bit integers to form a 64 bit result which is added to a 64 bit accumulator. The maccu instruction is used when the source operands are unsigned and the maccs instruction is used when the source operands are signed. These instructions execute in a single logical core cycle. For example on a 500MHz XMOS part with 8 tasks running a multiple-accumulate operation takes 16 ns to complete.

The compiler emits multiply-accumulate instructions where possible. For example:

```
long long f(int a[], int b[], unsigned size)
{
  long long accumulator = 0;
  for (unsigned i = 0; i < size; i++) {
    accumulator += (long long)a[i] * b[i];
  }
  return accumulator;
}
```

In the above loop the compiler will use a single maccs instruction to perform the multiplication and addition.

You can also target the maccs and maccu instructions in your code by calling compiler intrinsics which map directly to these instructions. To make use of these intrinsics you must include *xs1.h*:

```
#include <xs1.h>
```

The intrinsics are called as follows:

```
// Computes ((long long)a * b) + c.
{result_high, result_low} = macs(a, b, c_high, c_low);
// Computes ((unsigned long long)a * b) + c.
{result_high, result_low} = mac(a, b, c_high, c_low);
```