Application Note: AN10078

How to stream data between two cores over a channel

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to stream data between two cores over a channel.

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 stream data between two cores over a channel

By default, channel I/O is synchronous. This means that for every byte/word sent over the channel there is some handshaking taking place and also that the task performing the output is blocked until the input task at the other end of the channel has received the data. The time taken performing the synchronization along with any time spent blocked can result in reduced performance. Streaming channels provide a solution to this issue. They establish a permanent route between two tasks over which data can be efficiently communicated without synchronization.

To stream data between cores you first need to declare a streaming channel:

```
streaming chan c;
```

You can then pass each end of the channel to each logical core, thus opening a permanent route between the two cores:

```
par {
    f1(c);
    f2(c);
}
```

This function outputs the value of 1 on the channel streaming c:

```
void f1(streaming chanend c) {
    c <: 1;
}
```

This function inputs the value of 1 from the streaming channel c:

```
void f2(streaming chanend c) {
    int i;
    c :> i;
    printintln(i);
}
```