How to use transactions over channels

version	1.0.1
scope	Example. This code is provided as example code for a user to base their code on.
description	How to use transactions over channels
boards	Unless otherwise specified, this example runs on the SliceKIT Core Board, but can easily be run on any XMOS device by using a different XN file.

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 doing 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. Transactions provide a possible solution to this issue. They allow 2 cores to engage in a *transaction*, in which a sequence of matching outputs and inputs are communicated over a channel ansynchronously, with the entire transaction being synchronised.

To use a transaction to send data between cores you first need to declare a channel e.g.

```
chan c;
```

You can then pass each end of the channel to each logical core.

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

This function outputs the values of 1, 2 and 3 on the channel ‘c’ using a master transaction.

```
void f1(chanend c) {
  master {
    c <: 1;
    c <: 2;
    c <: 3;
  }
}
```

This function inputs the values of 1, 2 and 3 from the channel ‘c’ using a slave transaction.

```
void f2(chanend c) {
  int x;
  slave {
    c :> x;
    printintln(x);
    c :> x;
    printintln(x);
    c :> x;
    printintln(x);
  }
}
```