Application Note: AN10008

How to communicate between combined tasks

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to communicate between combined tasks.

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 communicate between combined tasks

When tasks are combined, they cannot communicate over channels due to deadlock issues but can communicate over interfaces. The method and results of communication are exactly the same as if the tasks were not combined. So the client can call interface functions:

```
[[combinable]]
void task1(interface my_interface client c)
{
  timer tmr;
  int time;
  int count = 0;
  tmr :> time;
  while (1) {
    select {
    case tmr when timerafter(time) :> int now:
      count++;
      if (count > 5) {
        c.finish();
        return;
      }
      // c is the client end of the connection,
      // let's send a message to the other end.
      c.msgA(count, 10);
      time += 1000;
      break;
    }
  }
}
```

and the server end can receive these messages:

```
[[combinable]]
void task2(interface my_interface server c)
{
  while (1) {
    select {
    case c.msgA(int x, int y):
      printf("Received msgA: %d, %d\n", x, y);
      break;
    case c.msgB(float x):
      // handle the message
      printf("Received msgB: %f\n", x);
      break;
    case c.finish():
      return;
    }
  }
}
```

The tasks can then be combined and communication will work as expected.

```
int main(void)
{
  interface my_interface c;
  [[combine]]
  par {
    task1(c);
    task2(c);
  }
  return 0;
}
```

In this case the compiler will change the code to not use xCONNECT channels to communicate, but will instead replace the communication with simple function calls that switch context between the tasks for the duration of the interface call.