Application Note: AN10010

How to define and use a distributable function

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to define and use a distributable function.

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 define and use a distributable function

If a task is a never-ending loop containing a single select (like a combinable function) that *only has cases responding to interface messages*, the function can be marked as *distributable*. For example:

```
[[distributable]]
void port_wiggler(server interface wiggle_if c, port p)
{
  // This task waits for a message on the interface c and
  // wiggles the port p the required number of times.
  while (1) {
    select {
    case c.wiggle(int n):
      printstrln("Wiggling port.");
      for (int i=0;i<n;i++) {
        p <: 1;
        p <: 0;
      }
      break;
    case c.finish():
      return;
    }
  }
}
```

A distributable task can be distributed within a `par`. This means that the task will not run on any particular core but will be run on the core of the task that calls to it.

```
int main() {
  interface wiggle_if c;
  par {
    task1(c);
    [[distribute]] port_wiggler(c, p);
  }
  return 0;
}
```

A distributed task must be on the same tile as the tasks it is connected to.