Application Note: AN10081

How to transfer memory ownership between cores

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to transfer memory ownership between cores.

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 transfer memory ownership between cores

At any one point each task in the program will own its memory and other tasks cannot access it. However, movable pointers can be moved over interface calls to transfer ownership to another task.

You can do this by having an interface that passes movable pointers between tasks:

```
interface pointer_transfer_if {
  // These functions need to be guarded since they are not always selected on
  [[guarded]] void give_pointer(int * movable x);
  [[guarded]] int * movable retrieve_pointer(void);
};
```

The following task moves ownership of the pointer p (and therefore the array a) to another task.

```
void task1(client interface pointer_transfer_if c) {
  // This task starts of with ownership of a
  int x[5] = {4,5,6,7,8};
  int * movable p = x;

  printf("Task1 can look at the array: %d\n", p[1]);

  // Give the ownership away to another task
  c.give_pointer(move(p));

  // p is now null so this task cannot access the memory
  p = c.retrieve_pointer();

  printf("Task1 can look at the array again: %d\n", p[2]);
}
```

Note that movable pointer declarations always hide the object they are referencing so once p is set to point at x, task1 cannot access x either. This way there is no aliasing and once the pointer is transferred there is no way for task1 to access the memory.

The other end of the connection can use the pointer that has been transferred to it:

```
void task2(server interface pointer_transfer_if c) {
  int * movable p;
  // Get pointer from other task
  select {
  case c.give_pointer(int * movable q):
    p = move(q);
    break;
  }
  printf("Task2 can now look at the array: %d\n", p[3]);

  // Give the pointer back
  select {
  case c.retrieve_pointer() -> int * movable q:
    q = move(p);
    break;
  }
}
```

Pointers can only be passed over interface connections. They cannot be sent over raw channels.