

Application Note: AN10111 How to specialize defines using backtrails

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to specialize defines using backtrails.

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 specialize defines using backtrails

A code reference (as defined by an *xta label* pragma) may occur multiple times within a program. For example, a function containing a code reference can be called from multiple places. Consider the following code:

```
int g(int j) {
  int x = 0;
  for (unsigned int i = 0; i < j; ++i) {
    #pragma xta label "loop_label"
    ++x;
  }
  return x;
}
int f() {
  #pragma xta call "g1"
  g(10);
  #pragma xta call "g2"
  g(20);
  return 0;
}
int main() {
  f();
  return 0;
}
```

Assume that you want to time the function *f*. You could try to use the *set loop* command to assign the loop iterations to the loop within *g*. However, the number of iterations taken by this loop depends on a passed parameter, the actual value being specified at the call point. To accurately capture this behavior in the XTA you need to use a *backtrail* when referencing the loop. This is achieved by using *xta call* pragmas. Consider the following XTA commands:

set loop g1,loop_label 10
set loop g2,loop_label 20

This tells the tool to set the loop containing loop_label to 10 iterations when called from g1, and to 20 iterations when called from g2.

XMOS®

Copyright © 2016, All Rights Reserved.

Xmos Ltd. is the owner or licensee of this design, code, or Information (collectively, the "Information") and is providing it to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. Xmos Ltd. makes no representation that the Information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.