Application Note: AN10109

How to run XTA scripts on compilation

This application note is a short how-to on programming/using the xTIMEcomposer tools. It shows how to run XTA scripts on compilation.

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 run XTA scripts on compilation

XTA scripts can be run on automatically on compilation. This will ensure that any timing issues introduced as a result of modifications to the source will be reported.

For example, compile the following code:

```
#include <stdlib.h>
#include <xs1.h>

port p1 = XS1_PORT_1A;
port p2 = XS1_PORT_1B;

int main() {
  int x;

  #pragma xta endpoint "input"
  p1 :> x;

  // Checks for errors..
  if (x == 1) {
    #pragma xta label "error_case"
    exit(1);
  }

  // do some computation here..

  #pragma xta endpoint "output"
  p2 <: 0;
  return 0;
}
```

Assume that there is a timing requirement between the input and the output of 100.0 ns. Assume also that you are not interested in the timing of the error case.

Add the following lines to a file *test.xta*:

```
analyze endpoints input output
set exclusion - error_case
set required - 100.0 ns
```

Passing this on the command line to XCC will cause these commands to be executed on compilation. For example:

```
xcc -target=XK-1A test.xc test.xta -o test.xe
```

The above will compile the test.xc source for the XK-1A target then execute the XTA commands in the script file: test.xta.

Note: In the xTIMEcomposer Studio, any .xta files in the project will automatically be passed to XCC on compilation.

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.