DK2

Using the logic estimator
Using the logic estimator

Celoxica, the Celoxica logo and Handel-C are trademarks of Celoxica Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous development and improvement. All particulars of the product and its use contained in this document are given by Celoxica Limited in good faith. However, all warranties implied or express, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Celoxica Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any incorrect use of the product.

The information contained herein is subject to change without notice and is for general guidance only.

Copyright © 2003 Celoxica Limited. All rights reserved.

Authors: SB

**Document number: 1**

Customer Support at http://www.celoxica.com/support/

<table>
<thead>
<tr>
<th>Celoxica in Europe</th>
<th>Celoxica in Japan</th>
<th>Celoxica in Asia Pacific</th>
<th>Celoxica in the Americas</th>
</tr>
</thead>
<tbody>
<tr>
<td>T: +44 (0) 1235 863 656</td>
<td>T: +81 (0) 45 331 0218</td>
<td>T: +65 6896 4838</td>
<td>T: +1 800 570 7004</td>
</tr>
<tr>
<td>E: <a href="mailto:sales.emea@celoxica.com">sales.emea@celoxica.com</a></td>
<td>E: <a href="mailto:sales.japan@celoxica.com">sales.japan@celoxica.com</a></td>
<td>E: <a href="mailto:sales.apac@celoxica.com">sales.apac@celoxica.com</a></td>
<td>E: <a href="mailto:sales.america@celoxica.com">sales.america@celoxica.com</a></td>
</tr>
</tbody>
</table>

www.celoxica.com
Contents

1 TUTORIAL: USING THE LOGIC ESTIMATOR ......................................................... 3
  1.1 ENABLING THE LOGIC ESTIMATOR .................................................. 3
  1.2 USING THE LOGIC ESTIMATOR RESULTS ......................................... 4
  1.3 REDUCING THE LOGIC DELAY .......................................................... 7
  1.4 REDUCING THE LOGIC AREA .............................................................. 12

2 INDEX ........................................................................................................... 17
Conventions

A number of conventions are used in this document. These conventions are detailed below.

**Warning Message.** These messages warn you that actions may damage your hardware.

**Handy Note.** These messages draw your attention to crucial pieces of information.

Hexadecimal numbers will appear throughout this document. The convention used is that of prefixing the number with '0x' in common with standard C syntax.

Sections of code or commands that you must type are given in typewriter font like this:

```c
void main();
```

Information about a type of object you must specify is given in italics like this:

```c
Copy SourceFileName DestinationFileName
```

Optional elements are enclosed in square brackets like this:

```c
struct [type_Name]
```

Curly brackets around an element show that it is optional but it may be repeated any number of times.

```c
string ::= "{character}""
```
1 Tutorial: Using the logic estimator

The following examples illustrate the use of the DK Logic Estimator to produce smaller and faster designs. A basic knowledge of Handel-C is assumed, and some knowledge of digital electronics and design techniques will also be helpful.

New users are recommended to work through the following topics in order:

1.1 Enabling the logic estimator

The logic estimator is a tool included in DK which generates an HTML based report on the expected logic area and delay of the Handel-C code in the current project. This information can be very useful to increase the speed and reduce the size of a Handel-C design. Further information on the detailed operation of the logic estimator can be found in the DK online help, the information below is to instruct you in enabling it correctly, and the link at the end will take you to information on using the results it generates.
To enable the logic estimator for a given project, select the Project->Settings menu, and select the Linker tab. Make sure that the Settings for drop-down list is set to EDIF. Check the boxes for Generate estimation info and Use technology mapper, as shown below:

**ENABLING THE LOGIC ESTIMATOR**

Enabling the technology mapper allows the logic estimator to produce more accurate results. If the mapper is not enabled, logic estimation can still be used, but the timing and resource usage information will be expressed in general terms, rather specific components and delays.

Next: **Using the logic estimator results** (see page 4)

### 1.2 Using the logic estimator results

The results from the logic estimator can help you to improve the speed and reduce the size of a Handel-C design. The following examples show you how to do this, the code is contained in the TutorialEstimator workspace, and the screenshots are from this workspace.

The version1 project in the TutorialEstimator workspace contains the following simple piece of code:
Using the logic estimator

```c
set clock = external;
void main(void)
{
    interface bus_in(unsigned 16) InBusA();
    interface bus_in(unsigned 16) InBusB();
    unsigned 16 A, B, C, D, Output;
    unsigned 32 Index;
    interface bus_out() OutBus(Output);

    while(1)
    {
        par
        {
            A = InBusA.in;
            B = InBusB.in;
            Index = 0;
        }

        do
        {
            par
            {
                C = A * B;
                D = A + B;
            }
            par
            {
                Output = C + D;
                Index++;
            }
        } while (Index < 10000);
    }
}
```

Build the above code in the version1 project in the TutorialEstimator workspace. The logic estimator will save its results in the TutorialEstimator\version1\EDIF directory. Open the file named Summary.html by double-clicking on it (this should load your computer's default web browser).

www.celoxica.com
It should appear as below:

> Area and delay estimation summary

**Area estimation by file**

<table>
<thead>
<tr>
<th>File name</th>
<th>LUT</th>
<th>FF</th>
<th>Mem</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATutorialEstimator/version1/version1.hcc</td>
<td>222</td>
<td>83</td>
<td>0</td>
<td>398</td>
</tr>
<tr>
<td>TOTAL</td>
<td>222</td>
<td>83</td>
<td>0</td>
<td>398</td>
</tr>
</tbody>
</table>

**Longest paths summary**

<table>
<thead>
<tr>
<th>Path</th>
<th>Grade 6</th>
<th>Grade 5</th>
<th>Grade 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum logic delay from Flip flop to Flip flop</td>
<td>11.25ns</td>
<td>12.38ns</td>
<td>14.29ns</td>
</tr>
<tr>
<td>Maximum logic delay from Flip flop to Pin</td>
<td>0.40ns</td>
<td>0.50ns</td>
<td>0.58ns</td>
</tr>
<tr>
<td>Maximum logic delay from Pin to Flip flop</td>
<td>0.30ns</td>
<td>0.33ns</td>
<td>0.38ns</td>
</tr>
</tbody>
</table>

Note: All area and delay estimates given here are approximate. For full information about the size and speed of a design, use the appropriate vendor's place and route and timing analysis software.

**Estimation summary from version1 project**

The first section of the summary provides an estimation of the logic area, described in terms of LUTs, FFs, memory bits and miscellaneous other components. The numbers of these components are listed per source file in the project, with a total at the end. Clicking on the link to the source file will take you to a page providing more detail on how the logic area is distributed within the source file.

The second section of the summary provides an estimate of the logic delay for the project, giving times for the available speed grades for the selected target device. Clicking on the link to Detailed path information takes you to a page showing which lines of Handel-C source code contributed to the longest path in the design. It is important to note that the delays shown here are purely for logic elements, and do not make any allowances for routing when the design is implemented in an FPGA. The exact delay can only be found by implementing the design using the FPGA vendors Place and Route tools. An approximate value for the total delay can be obtained by doubling the logic delay, as in many designs the logic and routing delays are roughly equal.
Note that if the Technology Mapper is not turned on (as described in *Enabling the logic estimator* (see page 3)), the information provided will not be as detailed or accurate as that shown here.

The following sections include instructions for reducing the logic delay and area of the design in the `version1` project in the `TutorialEstimator` workspace.

### 1.3 Reducing the logic delay

Build the above code in the `version1` project in the `TutorialEstimator` workspace. The logic estimator will save its results in the `TutorialEstimator\version1\EDIF` directory. Open the file named `Summary.html` by double-clicking on it (this should load your computer's default web browser). Click on the link to Detailed path information, which will take you to a page showing which lines of Handel-C source code contributed to the longest path in the design, as shown below:

```
> : Longest paths

Flip flop to Flip flop:
11.25ns

```

![Image showing longest paths](image_url)

```c
version1_ccc. Line: 29
0. Flip flop: 0.45ns

version1_ccc. Line: 49
1. XilinxMux 0.39ns
2. XilinxMux 0.28ns
3. XilinxMux 0.04ns
4. LUT 0.35ns
5. XilinxMux 0.34ns
6. XilinxMux 0.41ns
7. XilinxMux 0.10ns
8. LUT 0.35ns
9. XilinxMux 0.34ns
10. XilinxMux 0.04ns
11. XilinxMux 0.04ns
12. XilinxMux 0.04ns
13. XilinxMux 0.10ns
14. LUT 0.35ns
15. XilinxMux 0.34ns
16. XilinxMux 0.10ns
17. LUT 0.35ns
18. XilinxMux 0.34ns
19. XilinxMux 0.04ns
20. LUT 0.35ns
21. XilinxMux 0.34ns
22. XilinxMux 0.04ns
23. XilinxMux 0.04ns
```

www.celoxica.com
This information shows that most of the logic in the longest path is on line number 49 in the Handel-C source, and also that this line is associated with a large number of LUTs and other logic elements. The high logic delay is due to line 49 including a multiply and two adds in a single cycle, and it can be reduced by creating two extra variables C and D, and performing the calculation over two cycles, as shown below:

```c
do
{
    par
    {
        C = A * B;
        D = A + B;
    }
    par
    {
        Output = C + D;
        Index++;
    }
} while (Index < 10000);
```
The while() loop now takes two cycles to execute, but the logic delay has been reduced from 14.29ns to 10.1ns (for a grade 4 part), as shown in the new estimation summary below (compare to results in Using the logic estimator results (see page 4)). You can generate this summary yourself by building the version2 project in the TutorialEstimator workspace. The logic estimator will save its results in the TutorialEstimator\version2\EDIF directory. Open the file named Summary.html by double-clicking on it (this should load your computer's default web browser).

---

### Area and delay estimation summary

**Area estimation by file**

<table>
<thead>
<tr>
<th>File name</th>
<th>LUT</th>
<th>FF</th>
<th>Mem</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:\TutorialEstimator\version2\version2.hcc</td>
<td>223</td>
<td>116</td>
<td>0</td>
<td>398</td>
</tr>
<tr>
<td>TOTAL</td>
<td>223</td>
<td>116</td>
<td>0</td>
<td>398</td>
</tr>
</tbody>
</table>

**Longest paths summary**

<table>
<thead>
<tr>
<th>Path</th>
<th>Grade 6</th>
<th>Grade 6</th>
<th>Grade 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum logic delay from Flip flop to Flip flop</td>
<td>7.95ns</td>
<td>8.75ns</td>
<td>10.10ns</td>
</tr>
<tr>
<td>Maximum logic delay from Flip flop to Pin</td>
<td>0.48ns</td>
<td>0.50ns</td>
<td>0.58ns</td>
</tr>
<tr>
<td>Maximum logic delay from Pin to Flip flop</td>
<td>0.30ns</td>
<td>0.33ns</td>
<td>0.38ns</td>
</tr>
</tbody>
</table>

[Detailed path information](#)

---

Note: All area and delay estimates given here are approximate. For full information about the size and speed of a design, use the appropriate vendor's place and route and timing analysis software.

---

**Estimation summary from version2 project**

The code can be altered to allow the loop to execute in one cycle again by implementing a two-stage pipeline, where the first stage calculates the values of C and D, and the second stage adds them together. The pipeline must be primed before the while() loop begins executing, as shown below:

---

*www.celoxica.com*
/ * prime pipeline */
par
{
    C = A * B;
    D = A + B;
}
do
do
{  
    par
    {
        /* pipeline stage 1 */
        C = A * B;
        D = A + B;

        /* pipeline stage 2 */
        Output = C + D;
        Index++;
    }
} while (Index < 10000);
Try modifying the code in the version2 project in the TutorialEstimator workspace to use this pipeline, rebuild it, and open the estimation summary again. You will see that the logic delay is unchanged, and there has been no significant change in the number of LUTs or other logic elements used, despite calculating the values for C and D in two separate places. This is because the optimizations in DK include identifying common expressions which do not execute at the same time, and sharing hardware between them. The detail on the new longest paths when using the pipeline are shown below:

> Longest paths

**Flip flop to Flip flop:** 
7.95ns

```
version2.hc: Line: 27
1: Flip flop: 0.45ns
version2.hc: Line: 51
1: XilinxCARRYMux: 0.39ns
2: XilinxCARRYMux: 0.29ns
3: XilinxCARRYXor: 1.01ns
4: LUT: 0.35ns
5: XilinxCARRYMux: 0.34ns
6: XilinxCARRYMux: 0.04ns
7: XilinxCARRYXor: 1.01ns
8: LUT: 0.35ns
9: XilinxCARRYMux: 0.34ns
10: XilinxCARRYXor: 0.04ns
11: XilinxCARRYMux: 0.04ns
12: XilinxCARRYXor: 1.01ns
13: LUT: 0.35ns
14: XilinxCARRYMux: 0.34ns
15: XilinxCARRYXor: 0.04ns
16: XilinxCARRYMux: 0.04ns
17: XilinxCARRYMux: 0.04ns
18: XilinxCARRYMux: 0.04ns
19: XilinxCARRYMux: 0.04ns
20: XilinxCARRYMux: 0.04ns
21: XilinxCARRYXor: 0.04ns
22: LUT: 0.35ns
23: LUT: 0.35ns
24: LUT: 0.35ns
version2.hc: Line: 27
25: Flip flop: 0.29ns
```

**Flip flop to Pin:** 0.46ns
1.4 Reducing the logic area

The section on *Reducing the logic delay* (see page 7) looked at using the Logic Estimator to help increase the maximum clock rate at which a design could run. This section looks at how you can use the Estimator to reduce the logic area of a design.
Build the above code in the version2 project in the TutorialEstimator workspace. The logic estimator will save its results in the TutorialEstimator\version2\EDIF directory. Open the file named Summary.html by double-clicking on it (this should load your computer's default web browser). Click on the link to version3.hcc, which will take you to a page showing the logic used to implement each line of Handel-C source code, as shown below:

From this it is clear which lines of Handel-C source contribute most to the logic area of the design. Some of the logic is associated with the calculation of the values of C, D and
Output, and there is no opportunity to eliminate this, unless the widths of the variables was reduced. However, the loop control code is not as efficient as it could be, so we will now look at how to improve it.

First, the while condition on line number 56 uses a "less than" < comparison, when in fact a "not equal" != will perform the same function, as index is only incremented by 1 each time through the loop. Try changing this line of code from < to != in the in the version2 project in the TutorialEstimator workspace, rebuild it, and look at the Estimator output again. You will notice that the logic associated with line number has now reduced, as shown below:

```
LUT FF Mem Other
61    if 
62    {
16     29  Output = C + D;
11     61   Index++;
65     } 
66   } while (Index != 10000);
67   }
68 }
```

A further reduction in logic area is possible because the Index variable is 32 bits wide, but is never incremented above 10,000, which only requires a width of 14 bits. Try changing the width of index in the version2 project in the TutorialEstimator workspace, rebuild it, and look at the Estimator output again. You will notice that the logic associated with line numbers 30 and 56 has now reduced, as shown below:

```
LUT FF Mem Other
16  14    unsigned 14 Index;
56    }
56    } while (Index != 10000);
57    }
58 }
```
The version3 project in the TutorialEstimator workspace contains these changes (but not the pipelining described in *Reducing the logic delay* (see page 7)), and the estimation summary from building it is show below. Comparing with the summary in *Reducing the logic delay* (see page 7), it can be seen that a logic area reduction of over 15% has been achieved by changing only two lines of code.

---

### Area and delay estimation summary

#### Area estimation by file

<table>
<thead>
<tr>
<th>File name</th>
<th>LUT</th>
<th>FF</th>
<th>Mem</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:\TutorialEstimator\version3\version3.hcc</td>
<td>186</td>
<td>98</td>
<td>0</td>
<td>335</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>186</td>
<td>98</td>
<td>0</td>
<td>335</td>
</tr>
</tbody>
</table>

#### Longest paths summary

<table>
<thead>
<tr>
<th>Path</th>
<th>Grade 6</th>
<th>Grade 5</th>
<th>Grade 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum logic delay from Flip flop to Flip flop</td>
<td>7.95ns</td>
<td>8.75ns</td>
<td>10.10ns</td>
</tr>
<tr>
<td>Maximum logic delay from Flip flop to Pin</td>
<td>0.46ns</td>
<td>0.50ns</td>
<td>0.58ns</td>
</tr>
<tr>
<td>Maximum logic delay from Pin to Flip flop</td>
<td>0.30ns</td>
<td>0.33ns</td>
<td>0.38ns</td>
</tr>
</tbody>
</table>

[Detailed path information](#)

---

*Note: All area and delay estimates given here are approximate. For full information about the size and speed of a design, use the appropriate vendor’s place and route and timing analysis software.*

---

**Estimation summary from version3 project**
2 Index

E
estimator........................................... 3

L
logic area ........................................... 12
logic delay ......................................... 7
logic depth ......................................... 7
logic estimator .................................... 3

R
reducing logic delay ............................. 7