Altium Designer features powerful C-to-Hardware Compilation (CHC) technology. At the heart of this technology is the C-to-Hardware Compiler. The Compiler takes C source code as input and produces FPGA logic as output. Altium Designer’s integrated development environment and intuitive GUI essentially mean the CHC technology becomes a ‘black box’ as far as the designer is concerned – simply write C code on the one side and obtain the FPGA logic on the other, without ever needing to learn the intricacies of the C-to-Hardware Compiler itself!

In terms of FPGA design within Altium Designer, there are two ways in which the CHC technology can be used:

- **To design custom FPGA logic using C.** In this case the CHC technology is used to translate a single C source function – referenced using a C Code Symbol – into a hardware function. The end result is an independent module of FPGA logic which connects, through defined IO ports, to other areas of the circuit design. For more information, see the section Designing Custom FPGA Logic using C, later in this document.

---

**Summary**

This document provides an overview of C-to-Hardware Compilation (CHC) technology in Altium Designer. This technology facilitates the design of custom FPGA logic using C and processor acceleration.

---

**Figure 1.** High-level illustration of C-to-Hardware compilation in Altium Designer.

**Figure 2.** Using CHC technology to design custom FPGA logic using C. A single C function is converted into an FPGA logic module through use of a C Code Symbol.
To accelerate processor systems. In this case, the CHC technology is used to translate specified software functions in the C source code into hardware functions, which can be called by the embedded software running on a 'soft' processor in the design.

The C-to-Hardware Compiler is used in conjunction with Altium Designer's traditional embedded software compilers to build a system in which an embedded processor can offload critical functions to hardware. Access to, and control over, the generated hardware functions, is made possible through use of an appropriately-configured Application Specific Processor peripheral component (ASP). For more information, see the section Accelerating Processor Systems, later in this document.

Figure 3. Using CHC technology to accelerate a processor system. From the source code, the majority of functions will be compiled by an embedded software compiler to run on the CPU. One or more software functions are translated, using CHC, to hardware circuits and 'stored' within an ASP peripheral.

For detailed information on Altium Designer's C-to-Hardware Compilation technology, refer to the document GU0122 C-to-Hardware Compiler User Manual.
Designing Custom FPGA Logic using C

Altium Designer provides the ability to add custom logic to an FPGA design, where that logic is ‘captured’ in an underlying C source file, and referenced using a C Code Symbol primitive. The latter is a type of sheet symbol that is essentially used to ‘export’ a single top-level function from the referenced C source file – passing it to the C-to-Hardware Compiler. Simply write the functionality required in the comfort of Altium’s code-aware C Editor and then sit back as the CHC technology converts the code into VHDL or Verilog (depending on your defined netlisting preference). Armed with even a basic software background, there is now no reason why you can’t design custom FPGA logic!

The ability to incorporate underlying C code into an FPGA design is an extension to the concept of design hierarchy. Similar to the way in which a schematic sub-sheet or HDL file (VHDL/Verilog) is referenced by a Sheet Symbol on the parent design schematic, the C source file is referenced, as mentioned previously, by use of a C Code Symbol. Hierarchical net and bus connectivity between documents obeys the standard hierarchical project connection behavior, where parameters for the referenced function in the C source file connect to C Code Entries of the same name in the C Code Symbol that represents that document, as shown by example in Figure 4. The C Code Entries allow for connection to other logic in the design.

Figure 4. Adding custom logic through creation of hierarchical FPGA designs.

Combinatorial vs. Multi-Cycle Circuits

The C-to-Hardware Compiler can generate two types of circuit from the C source code for an exported function – Combinatorial and Multi-cycle. Referred to as the ‘interface mode’ for the C Code Symbol, the type of circuit is specified as part of the C Code Symbol’s properties or, more specifically, as part of the ‘function signature’ – the interface definition used to essentially ‘hook’ the C source function to the C Code Symbol.
It is worth looking more closely at the difference between the two:

- **Combinatorial** – this type of circuit consists of logic gates whose outputs at any time are determined only by the values of the inputs. In this interface mode, only the parameters of the exported function appear as C Code Entries on the C Code Symbol, commonly referred to as ‘Parameter’ entries.

  Combinatorial circuits can be generated for simple C functions which do not depend on stored state (memory, previous executions, etc). The C-to-Hardware Compiler will attempt to create a combinatorial circuit if requested but if this is not possible, an error will be issued during compilation and the interface mode must be changed to Multi-cycle. Note that if a function accesses global variables, this implies stored state and therefore a combinatorial circuit cannot be generated.

- **Multi-cycle** – this type of circuit requires more than one clock cycle to complete. When this interface mode is selected, additional START, DONE and CLOCK C Code Entries will automatically be added to the C Code Symbol, commonly referred to as ‘Control’ entries. These entries are not part of the exported C function.

  The circuit begins operating when the START signal is High. The inputs are assumed to be valid. When the circuit finishes operating, the DONE signal will be driven High and the outputs will be valid for one clock cycle. Due to the short availability of the outputs, external latching of the outputs is common practice.

  Note that if the Enable reset logic option is enabled, in the Variables region of the tab, reset logic will be built into the symbol, and two additional Control-type C Code Entries will be added to the symbol – RESET and RESET_DONE.

For a tutorial that explores the creation of custom FPGA logic using C, refer to the document **TU0133 Designing Custom FPGA Logic using C**.

**Accelerating Processor Systems**

In many processor systems, the embedded processor is charged with handling computationally-intensive algorithms. These algorithms place a burden on the processor, which can have a significant impact on intended performance. The performance of such designs can be greatly enhanced if these algorithms were moved to hardware, as circuits in the FPGA fabric itself, and that's where Altium Designer's CHC technology comes in...
Providing Hardware Acceleration

Hardware acceleration is the concept of enhancing the speed of a design by imparting software processes into hardware. Many computational algorithms that are straightforward to code and debug in software are inherently parallel in nature. Encryption algorithms, image manipulation and signal processing are just some examples. To remain as software entities, such functions place heavy demands on the processor.

FPGA devices are also parallel by nature, offering the ability to perform multiple operations simultaneously. To move computationally-intensive functions out of the software domain and into the hardware realm – through the use of C-to-Hardware compilation technology – not only eases the burden on the processor, but can also give your design a substantial boost in the speed department.

It is important to understand that while virtually all C programs can be converted to an electronic circuit by the C-to-Hardware Compiler, it is the characteristics of a source function that ultimately determines whether the Compiler can create an efficient hardware component, or whether it is better to execute that function on a processor core. The Compiler can only create a small and fast electronic circuit if the C source code is parallelizable.

The Role of the Application Specific Processor (ASP)

Altium Designer’s ASP peripheral component is essentially used as a ‘container’ for C source functions that are implemented in hardware through use of the CHC technology. The ASP peripheral enables a host processor to access and ‘communicate’ with the hardware-compiled functions within. The functions themselves populate the ASP once the design project has been compiled and synthesized.

When a hardware function is called – from within the embedded code running on the processor – the processor simply transfers values for that function's parameters to the ASP peripheral, starts the function and waits for it to return. If the hardware function delivers a return value, this will be read back by the host processor, to be used by the calling software routine.

Figure 7 illustrates the conceptual use of an ASP peripheral within a design. Two functions – *set_tabs* and *rotate* – in the C source code file will be translated to hardware circuits by the C-to-Hardware Compiler. Note that interconnect and arbitration components have been left out of the illustration for clarity.

Notice that the ASP has a memory interface which, in the example of Figure 8, is connected to the same memory as the soft processor. Variables that are common to both software and hardware functions are allocated in this shared memory. In terms of addressing, a software function can simply pass a pointer value – to a variable (data object) stored in the shared memory – as a parameter to a called hardware function.

![Figure 8. The ASP component acts as a container for source C functions that are subsequently implemented in the FPGA fabric through use of Altium Designer's CHC technology. The processor accesses these functions through communication with the ASP.](image-url)
Choosing Functions to Implement in Hardware

Although functions that are destined for implementation in hardware can be 'identified' directly at the source code level, Altium Designer provides a more friendly, straightforward and intuitive interface, as part of the configuration of its ASP peripheral.

The right-hand side of the configuration dialog (Figure 9) provides two distinct regions that deal with global variables and functions defined in the C source code of the embedded software project – linked to the processor which is wired to communicate with the ASP peripheral.

Figure 9. Set up your hardware acceleration requirements from within the ASP’s intuitive configuration dialog.

Use the upper list to determine which global variables should be allocated in hardware. Such variables will be allocated in ASP Block RAM by the C-to-Hardware Compiler. Access to this memory is much faster, in comparison to storage allocation in Block RAM outside of the ASP by the Embedded Compiler.

Use the lower list to specify which source functions should be implemented in hardware. To have the C-to-Hardware Compiler generate a function in hardware, as part of the ASP, simply enable the associated check box for that function, in the Implement in Hardware column. Should you wish to be able to call that hardware function from within the software running on the host processor, ensure that the corresponding check box in the Export to Software column is also enabled.

The following are some key points to consider when using these two function-related options:

- A function with enabled Implement in Hardware option will become a hardware function.
- A hardware function can call another hardware function.
- A hardware function cannot call a software function.
- A software function running on the host processor can call a hardware function, provided that hardware function has been exported to software (Exported to Software option enabled for function).
- A hardware-only function (not exported to software) can call a hardware function that has been exported to software. Conversely, a hardware function that has been exported to software can call a hardware-only function.

A global variable that is allocated in hardware can only be accessed by a function that has also been implemented in hardware. Such a variable can not be called from a software-based function running on the host processor.

Version (v1.0) Aug 12, 2008
Allocation of variables and implementation of functions in hardware can also be performed from within the C source code – either from the C To Hardware panel, or by right-clicking on a global variable/function directly in the code editor and using the relevant commands that appear on the context menu.

**Note:** When using the panel, only global variables and functions defined in the active C document will be listed.

---

**Generation and Use of Hardware Functions**

When configuring the ASP component, the following two options rank as the most important you will ever need to use:

- **Generate ASP** – this option provides the ability to enable or disable generation of hardware-compiled functions. With this option enabled, the C-to-Hardware Compiler will be invoked when you compile and synthesize your design project. All functions that have been enabled for implementation in hardware will be created as electronic circuits in the FPGA fabric.

- **Use ASP from Software** – this options enables you to control, on a global level, whether functions compiled into hardware can be called by software-based functions running on the processor. If this option is disabled, the relevant Embedded Compiler will generate the functions in software, and these will be used instead.

The typical (and most efficient) way in which these two options are used, is actually driven from a debugging perspective. In terms of code development, debugging a function is only really possible at the C source code level. It is near-impossible to debug the corresponding hardware implementation of that function.

By enabling the Generate ASP option, and disabling the Use ASP from Software option, you can effectively test and develop the software-compiled code only. The FPGA logic for any functions enabled for implementation in hardware will still be generated, but the processor will only use software-compiled versions of those functions.

Once the software is fully debugged and operates as required, simply enable the Use ASP from Software option, to switch over to using the hardware implementations of those functions. As the logic for the hardware functions already exists, full reprocessing of the entire FPGA design is not required – it is simply a case of recompiling and downloading the updated embedded software. In this way you can quickly switch between software-only and software-hardware implementations of the design, to observe the benefits obtained by using hardware acceleration.

For more detailed information on the ASP peripheral, refer to the document [CR0177 WB_ASP Configurable Application Specific Processor](#).

For a tutorial that explores the use of hardware acceleration in an FPGA design, refer to the document [TU0130 Getting Started with the C-to-Hardware Compiler](#).
Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version No.</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-Aug-2008</td>
<td>1.0</td>
<td>Initial release</td>
</tr>
<tr>
<td>18-Aug-2011</td>
<td>-</td>
<td>Updated template.</td>
</tr>
</tbody>
</table>

Software, hardware, documentation and related materials:

Copyright © 2011 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment.

Altium, Altium Designer, Board Insight, DXP, Innovation Station, LiveDesign, NanoBoard, NanoTalk, OpenBus, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.