

使用 Catapult C 综合和 Altera 加速库设计高性能 DSP 硬件

引言

当今的高性能 FPGA，例如 Altera® Stratix® III 器件，为设计工程师提供了硬件平台，满足了他们对新一代无线和视频算法的计算需求。尽管这些器件提供了专用硬件来实现乘累加 (MAC) 等数字信号处理 (DSP) 算法基本构建模块，设计人员还需要在寄存器传送级 (RTL) 上迅速实现算法。

以前的设计流程含有 C++ 等高级语言编写的算法函数模型，将其手动编码为 RTL。手动建立 RTL 的方法不但耗时，而且容易出错，对后端布线延时问题非常敏感。Catapult 高级 C++ 综合被用于构建 ASIC 硬件子系统，例如无线、视频和图像处理领域非常复杂并且需要进行大量计算的应用。Catapult 的 ASIC 功能和 Altera 加速库相结合，使设计人员能够从 ANSI C++ 建模的算法迅速转换到运行在 FPGA 硬件中的 RTL。而且，这一设计流程还帮助设计人员直接从 C++ 中找到 FPGA DSP 模块，使用高级综合约束，很容易解决后端时序问题。

使用高级综合开发 DSP 算法

在典型的 DSP 算法设计流程中，软件设计人员或者系统规划人员采用 C++ 等高级语言对算法建模。在更高级的抽象层上，系统规划人员可以将精力集中在功能上，而不用担心实施细节，例如系统流水线，以及需要多少 FPGA DSP 模块才能满足性能要求等。算法功能可行后，建立完全符合面积和性能标准的规范，传送给 RTL 设计人员。

RTL 设计人员确定实现这些规范所需要的资源类型和数量，决定采用哪些硬件资源，例如 RAM、DSP 模块和移位寄存器的数量等。由于 RTL 综合工具的推断功能不能充分利用专用硬件资源的所有工作模式，因此，很难有效地进行 RTL 编码。为解决这一问题，RTL 设计人员一般在 RTL 代码中例化 DSP 宏，直接通过类 / 参数来配置 DSP 模块。但是，这要求硬件设计人员知道怎样实现这些模块和系统其他部分的接口。

为顺利实现这一非常耗时的过程，Catapult C 高级综合设计过程首先对算法进行描述，然后选择目标技术。算法描述是纯粹的 ANSI C++ 源代码，只对功能进行说明。并行和接口协议等硬件要求可通过约束在 Catapult 中实现，从而也指导了综合过程。

例如，下面的算法是一个基本有限冲击响应 (FIR) 滤波器，使用免费的 Mentor Graphics Algorithmic C™ 数据类型 (加链接) 来定义接口和内部位宽度。

```
void fir_filter (ac_int<16> *input, ac_int<16> coeffs[NUM_TAPS], ac_int<16> *output ) {
    static ac_int<16> regs[NUM_TAPS];
    ac_int<32> temp = 0;
    int i;
    SHIFT:for ( i = NUM_TAPS-1; i>=0; i--) {
        if ( i == 0 )
            regs[i] = *input;
        else
            regs[i] = regs[i-1];
    }
    MAC:for ( i = NUM_TAPS-1; i>=0; i--) {
        temp += coeffs[i]*regs[i];
    }
    *output = temp>>16;
}
```

C++ 算法并没有说明需要多少乘法器以及什么类型的乘法器来实现硬件。因此，系统规划人员不用在实施细节上耗费精力就能够有效的建立算法。

下一步是确定目标技术和关键规范。在 Catapult 中，目标技术可以是 ASIC 或者 FPGA，与源代码描述无关。Catapult C 综合使用专用技术库特征参数来建立高度优化的运算库，例如加法器和乘法器等。这一特性描述过程收集器件专用资源详细的面积和时序信息，使 Catapult 能够建立技术预知计划，不会浪费 HLS 探索过程中 RTL 综合的时间。其结果是快速的前端面积 / 性能估算，得到专用技术 RTL 输出。

指定好目标技术以及时钟频率后，设计人员使用自动高级综合研究设计空间。由于自动过程比手动 RTL 编码快得多，设计人员可以关注更多的选择，综合考虑面积和性能，所实现的硬件完全满足设计目标要求。高级综合工具对目标技术非常清楚，根据时钟频率要求来选择合适的运算，在需要的地方增加系统级流水线，确保不会违反时钟频率约束。设计人员可以使用开环和环流水线等高级综合约束，研究从最短串联到全并联实现的多种微体系结构（对比图 1 和图 2 中的具体实现）。

图 1. 串联 FIR 实现

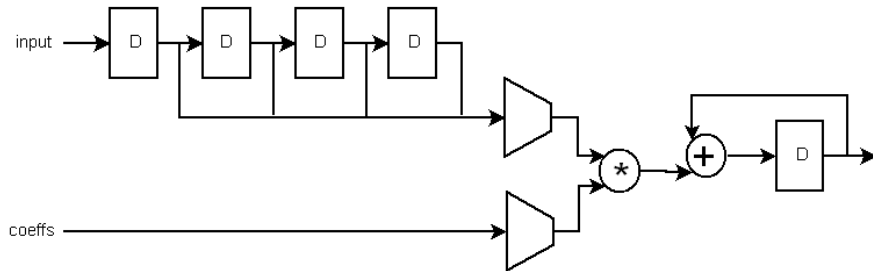
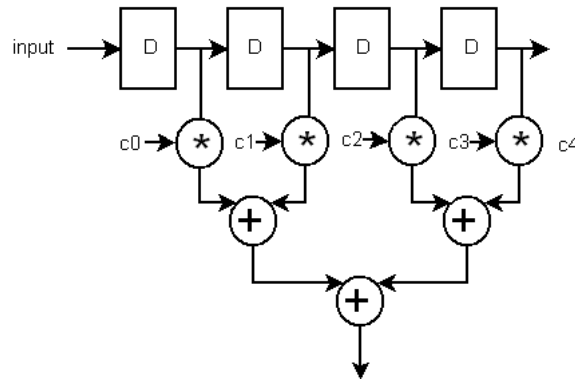


图 2. 并联 FIR 实现



选择调度所需要的运算以满足时钟频率约束。这样，对于频率更高的时钟，自动选择流水线元件。此外，可以采用高级综合资源约束来加入其他的流水线级，以减小后端走线延时。

尽管高级综合过程是技术预知的，但是依靠下游 RTL 综合来推断相应的 FPGA 资源，编写最终自动生成的 RTL 网表。例如，在高级综合过程中选择的三级流水线乘法器被网表化为乘法运算，其后是三个寄存器。然后，将寄存器和 DSP 模块内置寄存器重新定时，采用下游 RTL 综合来推断该网表相应的硬件资源。

这一流程的限制之一是 C++ 语言不含有实现 FPGA DSP 模块对应的“宏”内置运算操作，例如乘累加 (MAC) 和乘加法 (MULT-ADD) 等。而且，RTL 综合工具的推断功能目前只能推断出 DSP 模块的基本配置。Stratix III FPGA 等器件支持的高级模式可实现饱和以及取整等运算，只能在 RTL 中直接例化 DSP 宏才能使用这些模式。

Catapult C 综合硬件加速流程使用 Altera 加速库，避免了这些限制，使设计人员能够直接从一组纯粹的 ANSI C++ 函数中找到 FPGA DSP 模块。所有这些详细的 DSP 宏例化、配置和接口都非常清晰，很少的投入就可以实现最佳设计。

Stratix III 系列器件

库在相配合的硬件中运行时才能发挥作用，Altera 新的 Stratix III 系列高端 FPGA 提供可配置逻辑架构，是世界上性能最好、容量最大的 FPGA，而且功耗非常低。WiMAX、3GPP WCDMA、H.264 视频压缩、医疗成像和 HDTV 等很多复杂的系统都使用先进的 DSP 技术，需要进行大量的数学运算。Stratix III 器件非常适合这些系统应用，其 DSP 模块含有进行乘法、加法、减法、累加、求和以及动态移位操作等多种专用单元，设计人员可以配置这些单元来构建复杂的定点和浮点算术函数。然后这些函数很容易实现计算量更大的子系统，例如上述系统中常用的 FIR 滤波器、复数 FIR 滤波器、有限冲击响应 (IIR) 滤波器、快速傅立叶变换 (FFT) 以及离散余弦变换 (DCT) 等。

每一 Stratix III 器件都有 2 到 7 列 DSP 模块，非常适合实现流水线算法。当采用 Catapult C 等合适的工具时，性能可以达到 6.5 GMAC 至 270 GMAC，比高端 DSP 处理器高 60 倍。

加速库

Catapult 综合 Altera 加速库提供 C++ 函数库，直接映射到 Altera FPGA 专用硬件资源。表 1、表 2 和表 3 列出了 Altera 器件使用的所有库。Catapult 综合 Altera 加速库提供 C++ 函数库，直接映射到 Altera FPGA 专用硬件资源。表 1、表 2 和表 3 列出了 Altera 器件使用的所有库。

表 1. 所有 Stratix 和 Cyclone® 器件支持的函数

函数	说明
Alt_sqrt	计算平方根
Alt_sqrt_remainder	计算平方根余数
Alt_abs	计算绝对值
Alt_divide_quotient	计算除法运算的商
Alt_divide_remainder	计算除法运算的余数
Alt_shift_taps	基于 RAM 的移位寄存器，含多路输出。
Alt_barrel_shift	桶形移位寄存器

表 2. Stratix 和 Stratix II 器件支持的其他函数

函数	说明
Alt_4mult_add	4 个乘法器的和
Alt_4mult_sub	4 个乘法器的和，前两个乘法器输出、后两个乘法器输出互减，最后相加。
Alt_3mult_add	3 个乘法器的和
Alt_3mult_sub	3 个乘法器的和，前两个乘法器的输出互减，最后相加。
Alt_2mult_add	两个乘法器的和
Alt_2mult_sub	两个乘法器的输出相减
Alt_1mult_add_accum	乘法器，其后是加法累加器。
Alt_1mult_sub_accum	乘法器，其后是减法累加器。

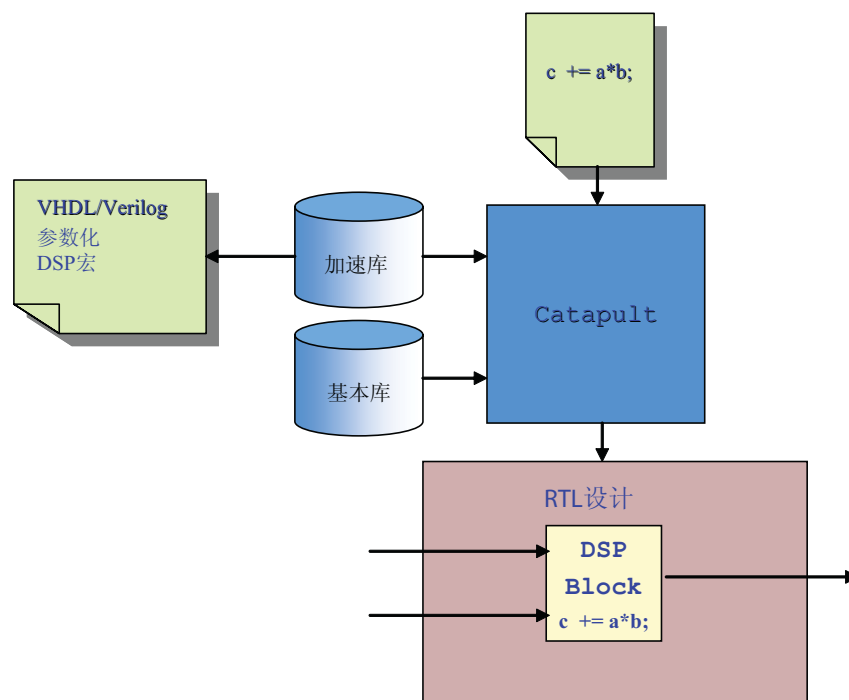
表 3. Stratix III 器件支持的其他函数

函数	说明
Alt_4mult_add_accum	4 个乘法器的和，其后是累加器。
Alt_4mult_sub_accum	4 个乘法器的和，前两个乘法器输出、后两个乘法器输出互减，最后相加，其后是累加器。
Alt_3mult_add_accum	3 个乘法器的和，其后是累加器。
Alt_3mult_sub_accum	3 个乘法器的和，前两个乘法器的输出互减，最后相加，其后是累加器。
Alt_2mult_add_accum	两个乘法器的和，其后是累加器。
Alt_2mult_sub_accum	两个乘法器的输出相减，其后是累加器。

当 Catapult 工程中含有加速库时，C++ 函数被绑定至 Catapult 库运算符，这些运算符相应地绑定至参数化 RTL 模块，例化并配置专用 Altera 器件 DSP 宏。如果工程中不包含库，直接对 C++ 代码综合，使用户能够将设计重新定位到其他没有内置 DSP 硬件的器件。

如图 3 所示，通过包含模板函数 C++ 头文件来访问 C++ 中的 Altera 加速库。用户利用模板函数来指定数据位宽度等参数，以及流水线寄存器等硬件参数。同时支持整数和定点 Algorithmic C 数据类型模板函数。

图 3. Altera 操作符流程图



一个实例函数是 Alt_4Mult_add，它将 Stratix III DSP 模块配置为 4 个 18×18 乘法器，以及随后的两个加法器级。

```
#pragma map_to_operator "Alt_4mult_add"
template <bool use_extra_input_reg, bool use_mult_reg, unsigned int output_register, int width_a,
         bool sign_a, int width_b, bool sign_b>
ac_int<width_a+width_b+2, sign_a || sign_b> Alt_4mult_add
(
    ac_int<width_a,sign_a> a0, ac_int<width_b,sign_b> b0,
```

```

    ac_int<width_a,sign_a> a1, ac_int<width_b,sign_b> b1,
    ac_int<width_a,sign_a> a2, ac_int<width_b,sign_b> b2,
    ac_int<width_a,sign_a> a3, ac_int<width_b,sign_b> b3
)
{
    return ((a1*b1 + a0*b0) + (a3*b3 + a2*b2));
}

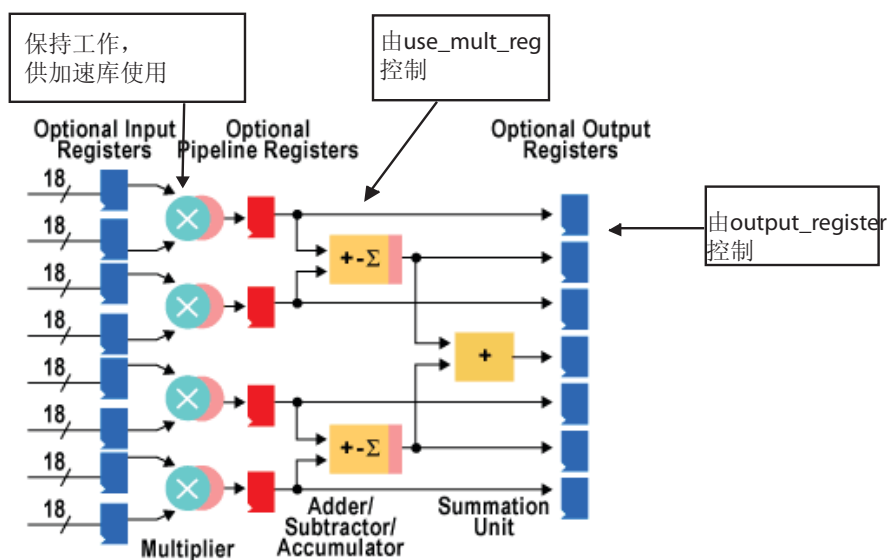
```

“map_to_operator” 编译语句说明如果有加速库，该函数直接映射至硬件。模板参数包括以下行为：

- use_extra_input_reg - 在 DSP 模块之外加入其他的输入寄存器，以解决走线延时时序问题。
- use_mult_reg - 寄存乘法器输出。有效值是 0（无寄存器）或者 1（一个寄存器）。
- output_register - 设置输出寄存器的数量。有效值是 0（无寄存器）或者 1（一个寄存器）或者 2（两个寄存器）。
- Width_a - A 输入位宽度。最大支持 18 位位宽。
- Sign_a - 输入的符号
- Width_b - B 输入位宽度。最大支持 18 位位宽。

图 4 所示为 Altera Stratix III DSP 模块，以及怎样根据 C++ 函数模板参数来配置它。

图 4. Altera DSP 模块 Alt_4mult_add 结构



使用 Altera 加速库函数很容易重新编写 FIR 源代码：

```

#include <ac_int.h>
#include "Altera_accel.h"
#include "stdio.h"
#define NUM_TAPS 32
#pragma design top
void fir_filter (ac_int<18> *input, ac_int<18> coeffs[NUM_TAPS], ac_int<18> *output ) {
    static ac_int<18> regs[NUM_TAPS];
    ac_int<38> temp = 0;
    int i;
    SHIFT:for ( i = NUM_TAPS-1; i>=0; i--) {

```

```
    if ( i == 0 )
        regs[i] = *input;
    else
        regs[i] = regs[i-1];
}
MAC:for ( i = 0; i< NUM_TAPS; i+=4) {
    temp += Alt_4mult_add<1,1,1>(
        coeffs[i],regs[i],
        coeffs[i+1],regs[i+1],
        coeffs[i+2],regs[i+2],
        coeffs[i+3],regs[i+3]);
}
*output = temp>>18;
}
```

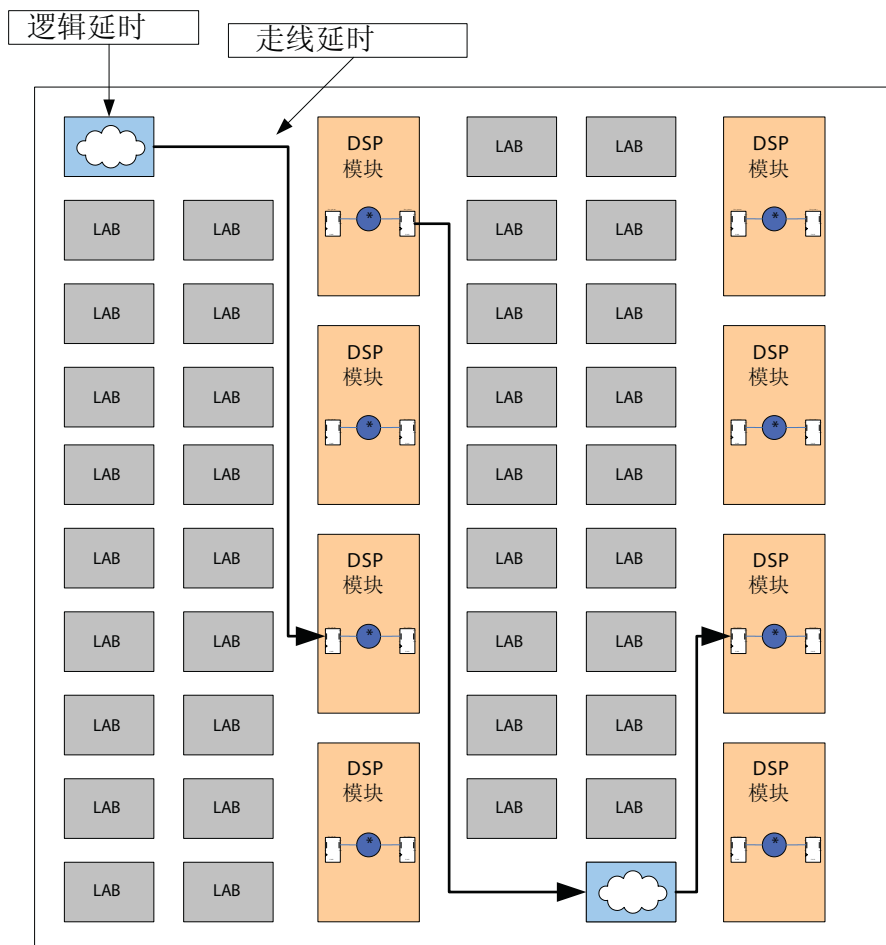
在上面的 FIR 实例中，只需要指定寄存器设置的模板参数。模板匹配自动置入宽度和符号模板参数。

- 库 C++ 模板函数
- 通过模板参数配置 DSP 模块
- 使用 `ac_fixed` 数据类型进行保护以及取整运算

固定走线延时时序问题

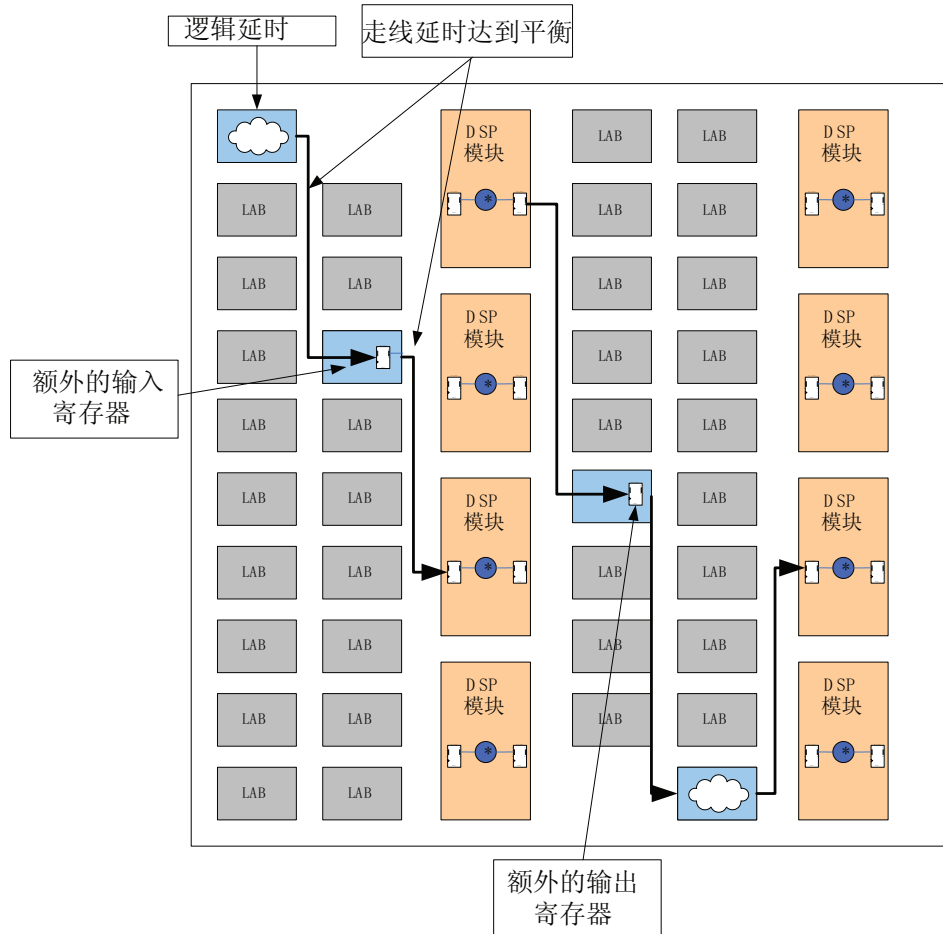
除了直接映射至 Altera DSP 模块以及其他的硬件特性之外，用户还可以利用加速库来解决比较难处理的后端时序问题，这些问题是由走线延时和拥塞造成的。如图 5 所示，时序失效的可能原因是 DSP 模块的进出走线。即使 DSP 模块输入和输出寄存器全部采用了流水线，也会发生这种情况。

图 5. Stratix III 高级体系结构



加速库函数使用模板参数，为硬件映射运算输入输出增加其他的寄存器。这些额外的寄存器提供平衡走线延时的布局布线功能，从而解决了走线延时问题（图 6）。这种方法相对于手动 RTL 编码的好处是它不需要重新进行设计。在 RTL 中实现这一功能可能需要对数据通路有限状态机 (FSM) 进行较大的改动。

图 6. 使用增加的输入和输出寄存器来减小走线延时



加速库的按键式验证

高级综合设计流程的一个关键要求是能够自动验证从非定时 C++ 代码以及后 RTL 综合网表中生成的 RTL。Catapult C 自动验证流程不但对 C++ 和 RTL 进行自动验证，而且还自动验证加速库 C++ 模板函数和例化 DSP 宏的 RTL。

从非定时纯 C++ 开始，结合一组接口综合约束，可以自动建立 SystemC™ 验证结构，使 C++ 源设计和测试台能够测试 RTL 和后 RTL 综合网表。这一自动生成的测试环境不但能够直接从 C++ 测试台中采集输入激励和设计输出，而且还生成了 SystemC 处理程序，在仿真平台上协同仿真 RTL 和 C++。还对 RTL 仿真输出进行采集，自动对比 C++ 源设计，以验证功能。

结论

具有专用 DSP 模块的高性能 FPGA 为实现大计算量的复数算法提供了硬件平台，但是，设计人员仍然面临将算法在硬件 RTL 中运行的挑战。Catapult C 和 Altera 加速库相结合后，设计人员不但能够迅速实施算法，而且还可以直接访问 Altera DSP 模块，调整性能和时序，获得的结果要优于其他的 C/C++ 综合技术。

致谢

- Thiagaraja Gopalsamy, 技术组督导, Altera 公司。
- Mike Fingeroff, 技术营销工程师, Mentor Graphics。

**ALTERA**®

101 Innovation Drive
San Jose, CA 95134
www.altera.com

版权 © 2007 Altera 公司。保留所有版权。Altera, 可编程解决方案公司、程式化 Altera 标识、专用器件名称和其他所有其他专有商标或服务标记, 除非特别声明, 均为 Altera 公司在美国和其他国家的商标和服务标记。所有其他产品或服务名称的所有权属于其各自持有人。Altera 产品受美国和其他国家多种专利、未决应用、掩模著作权和版权的保护。Altera 保证当前规范下的半导体产品性能与 Altera 标准质保一致, 但是保留对产品或服务在没有事先通知时的变更权利。除非与 Altera 公司的书面条款完全一致, 否则 Altera 不承担由使用或者应用此处所述信息、产品或服务导致的责任。Altera 建议客户在决定购买产品或服务, 以及确信任何公开信息之前, 阅读 Altera 最新版的器件规范说明。