FPGA高层综合中的内存子系统研究综述

张展鹏 张治国

(中山大学信息科学与技术学院 广州 510006)

摘 要 高层综合从高级编程语言对系统的行为描述出发,把系统中的计算转移到可重构的硬件中,以加速系统运行。高层综合中生成有效的内存子系统尤为重要,特别是对于数据密集型的计算。分析了现阶段FPGA高层综合技术及其内存子系统,把生成的内存子系统从体系上分为三类:DSP型体系、以CPU为核心的体系以及基于可重构内存功能单元的体系。结合实例介绍了各体系的特点,然后按照高层综合过程中的前端和后端,分类讨论了内存子系统的优化技术。经过分析评价,指出片外与片上内存间的映射、程序的有效建模等问题仍有待解决,自动化生成内存组织体系和多模块综合是可能的研究方向。

关键词 高层综合,FPGA,内存子系统,可重构体系结构

中图法分类号 TP368 **文献标识码** A

Review on Memory Subsystems in High Level Synthesis for FPGA

ZHANG Zhan-peng ZHANG Zhi-guo

(School of Information Science and Technology, Sun Yat-Sen University, Guangzhou 510006, China)

Abstract In order to accelerate the systems, High Level Synthesis(HLS) aims to map the computation of the system to the reconfigurable hardware, based on the behavioral description of the system in high level programming language. In HLS, the generation of efficient memory subsystem is critically important, especially for data-intensive computation. In this paper, the existing HLS technologies for FPGA and their memory subsystems were analyzed. The generated memory subsystems' architectures were divided into three categories: DSP-like architecture, CPU-control architecture and architecture based on the reconfigurable memory functional units. These architectures were discussed with examples. After that, the front end and back end optimization techniques for memory subsystems in HLS were discussed respectively. In addition, the aforementioned architectures and techniques were analyzed and evaluated. Finally, the mapping between the off-chip and on-chip memories and the efficient modeling for the programs were listed as the remained problems. In HLS, syntheses for multi-module and automatic generation of the memory organization can be future research directions.

Keywords High level synthesis, FPGA, Memory subsystems, Reconfigurable architecture

1 引言

凭借高可并行性以及日益增加的规模和速度,FPGA的应用越来越广泛。同时,在FPGA上增加的片上内存、乘法器等常用粗粒度元件[1]也提升了FPGA在实际开发和应用中的效率。日益完善的FPGA片上系统(System on Chip, SoC)方案、第三方提供的丰富IP核资源[1],为FPGA增加了生命力和实际应用价值。此外,FPGA的可重构特性适合于"演化硬件"的实现,使系统能够适应运行环境[2]。

在设计综合方面,以VHDL和Verilog HDL等硬件描述语言为设计输入是FPGA开发的主要手段。然而,这种设计输入涉及到硬件编程细节且编写过程繁琐。面对日益复杂的系统,这种方式不容易被开发人员使用并及时转化为硬件推出市场[3]。另一方面,在通用计算平台中使用高级编程语言积累了很多算法和应用,这种方式也不能利用这些现成的资源。因此,在2000年以后,系统级的设计模式逐步引入各种设计综合工具[4]。设计者以C、SystemC等语言,在更高的抽象层次上对系统进行行为描述,并使用工具将输入直接综合到可重构硬件。基于高层综合的研究成果,目前已有多种针对FPGA的高层综合编译器[5-7]。

1.1 FPGA高层综合

高层综合(high level synthesis)是指从系统的高层抽象描述出发,直接综合出对应的硬件描述[8],其主要流程如图1所示。

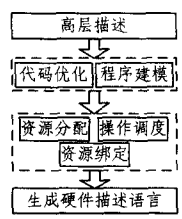


图1 高层综合的编译流程

张展鹏(1987—),男,硕士,主要研究方向为可重构计算,E-mail:zhzhanp@gmail.com;张治国(1962—),男,副教授,主要研究方向为并行与分布计算、系统形式描述与验证、程序与计算理论。

首先，通过人工或辅助工具，选取系统的全部或计算耗时的部分，根据高层综合工具的输入规定，对该部分进行修改或重新编写，即完成高层描述。高层描述是对系统的行为描述。

然后可以对高层描述中的程序代码进行优化调整，例如循环转换或死码删除。更多的代码优化需要基于程序的建模，或者直接对模型进行修改。建立的模型体现了程序的数据或执行特性，同时，它也为后端的硬件资源分配和操作调度提供了依据。不同的综合技术，会采用不同的代码优化策略，代码优化以及程序建模这两者的执行没有绝对的先后顺序，本文将此步骤统一划分为编译的前端。

接着，根据前端输入的模型，进行资源分配、操作调度以及资源绑定。与编译前端类似，这几个步骤因不同的技术，执行顺序有所不同，本文将此划分为编译的后端。一般地，后端编译中，先进行资源分配，为输入模型中各个变量和操作分配资源，例如存储和计算功能单元。然后，对各个操作进行调度。由于之前的高层描述是行为描述，不带时序信息[39]，为了映射到硬件电路，需要把各个操作安排在特定的时钟周期执行。根据具体的调度策略，操作会安排成执行序列顺序执行或并发执行。调度完成后，各个操作有了先后时序，变量的生存期也得到了确定，可以对变量和操作进行资源绑定。生存期不交叉的变量可以共享存储单元，一些计算单元也可以复用。

一般地，综合过程中对目标硬件模块预设特定的体系结构，每个高层描述会综合生成一个硬件模块。图 2 是高层综合工具 ROCCC[23] 生成的硬件模块，主要包含控制器、数据通路(data path)逻辑以及内存子系统。根据操作调度和资源绑定的结果以及预设的体系结构，最终编译成可综合的硬件描述语言。

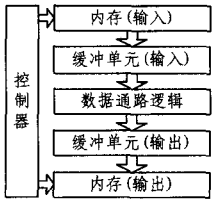


图 2　ROCCC 生成的硬件模块

1.2　高层综合的内存子系统

内存子系统是嵌入式系统的关键部件之一。首先，数据密集型的应用需要频繁地访问内存。其次，内存访问消耗的能量占据了总能量消耗的很大一部分[13]。第三，内存子系统占用了嵌入式系统中较多的硬件资源。嵌入式片上系统中，内存子系统在芯片面积和功耗方面的消耗是数据通路的 10 倍[14]。无论是体系结构[14-16]，还是编译时的优化技术[17,18]，其都是嵌入式应用中内存子系统的重要研究方向。

对于 FPGA 高层综合，内存子系统的优化很有必要。而从 FPGA 的特性来看，也有优化的可能性。首先，FPGA 的可重构特性使得编译甚至运行过程中，可以根据不同的应用类型，生成对应的内存子系统。其次，FPGA 芯片上有一定容量的片上内存，这些内存块的访问周期远远小于对片外内存的访问。充分利用片上内存，减少片外内存的访问能有效地减少系统功耗[13]。凭借这些特性，高层综合技术可以为应用生成合适的内存子系统，增加数据通路的并行性，减少数据访问延时，也可以通过合适的缓存技术减少对片外内存的访问，提高执行效率，降低系统功耗。

2　高层综合中内存子系统的体系分类

高层综合中可以对其目标系统中的内存子系统预设不同的体系结构。将此划分为三种类型进行讨论。

2.1　DSP 型体系

这种综合技术把整个程序或函数编译成一个独立的硬件模块，如 GAUT[35]，ROCCC[23]，SPARK[30]。综合生成的模块内部集成了运算处理单元、接口单元以及内存单元。模块以独占的方式使用内存，不与外界共享，内存子系统完全服务于该模块。

GAUT 针对 DSP 应用，是这种体系的典型代表，编译器接收 C 语言的高层描述，生成 VHDL 的 RTL 级描述。图 3 描述了 GAUT 的目标体系结构。高层描述被综合成一个独立的硬件模块，包含通信单元、处理单元以及内存单元。内存块不仅分布于内存单元，而且在通信单元、处理单元还设置了少量内存。通信模块中包含一个同步处理器以及输入/输出存储单元。处理单元中包含了目标应用的数据通路逻辑以及控制器。同时也有存储单元，包括寄存器以及少量片上内存实现的 FIFO、LIFO，用于存储使用中的数据。内存单元中包括主要的内存块以及内存访问控制逻辑。多个内存块被组织成层次性的结构。内存分配可以是静态或动态的。对于静态分配，应用执行过程中，数据保持在同一内存块。动态情况下，应用运行过程中根据编译时生成的内存映射文件，在不同层次的内存块中传送数据。

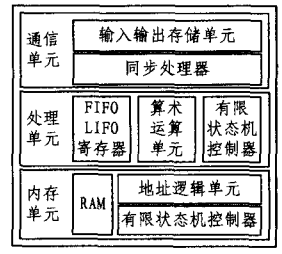


图 3　GAUT 的目标体系结构

可以看出，这种类型的内存子系统可以根据具体的应用进行高度定制。基于对内存的独占使用，高层综合时可以根据应用的特点安排内存的分布，并根据数据访问的特点，生成特定的逻辑，以提高数据访问的并行性。同时，也可以有效地实现内存体系中不同层次的内存块间的映射，因为不同层次的内存访问延时往往也不同，这种映射可以尽可能地利用高效的内存块。但同时，这种 DSP 型的模块包含了应用运行中所有需要的逻辑，如内存访问、同步和通信等，使得这种体系占用较多的逻辑资源。

2.2　以 CPU 为核心的体系

这种体系以通用计算平台为基础，运用于软硬件协同工作[27]的系统。C2H[24]，CHiMPS[36]，CCAP[9]，LegUP[10] 的目标内存子系统等都属于这种体系。程序中适合 FPGA 硬件实现的部分被综合成硬件模块，以外设或协处理器的形式与片上或片外的通用平台 CPU 协同工作。硬件模块和通用平台的 CPU 共享主存，主存一般使用片外内存，用于存放全局变量或静态变量。主存的分配由 CPU(软件)管理。在系统运行过程中，软件部分仍然在 CPU 执行，当执行到达被综合的部分时，调用硬件执行。CPU 向硬件模块传递参数，可以

是数据或内存地址。若硬件模块接收到内存地址，可以通过DMA访问主存数据。此外，可以为硬件模块分配局部内存(一般为片上内存)，用于存储局部变量。但这种体系由于缺乏协调单元，无法自动避免局部内存导致的数据一致性问题。

C2H是Altera公司提供的高层综合工具，工作于Altera的FPGA片上系统方案中，实现C语言到硬件描述语言的映射[24]。系统设计者在CPU执行的C主程序中，指定待综合的函数，每个函数将分别被综合成一个硬件模块，以外设的形式连接到Nios处理器，其目标内存子系统是典型的以CPU为核心的体系。C2H生成的硬件模块在系统中的位置和连接方式如图4所示。硬件模块主要包含CPU交互逻辑、硬件运算逻辑以及DMA。调用硬件模块时，CPU向硬件模块发出启动信号，传递输入参数，然后开始执行。硬件模块使用的变量被分配在片外内存、片上内存或芯片的寄存器。对于分配在片外或片上内存的变量，硬件模块通过DMA的方式进行读写。

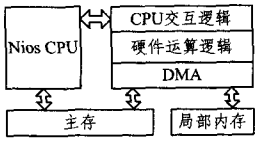


图4 C2H硬件模块在系统中的位置和连接方式

这种以CPU为核心的体系沿用通用计算平台的思想，易于实现，生成的模块只需遵循原有系统的总线连接方式即可集成到系统中，不影响原有系统的内存结构。这种方式也可以实现软硬件协同工作，因为不是所有的计算都适合于使用FPGA硬件实现(例如控制密集型等难于并行化的程序)，软硬件协同的方法能充分利用FPGA硬件的优势。但由于共享主存，多个模块并行工作时会降低主存的访问效率。体现在两个方面：多个模块共享片外内存会引起访问冲突从而导致等待；访问片外内存的端口数量非常有限，不利于数据通路中的并行访问。体系中引入的局部内存一定程度上缓解了上述问题。但由于局部内存一般使用片上内存来实现，片上内存在FPGA上数量少，硬件模块的独占使用方式不能有效利用这种资源。此外，这种体系不能保证局部内存与主存的数据一致性。

2.3 基于可重构内存功能单元的体系

为克服以CPU为核心的体系造成的内存利用率以及访问效率问题，一些学者基于FPGA可重构的特性，提出了在系统中引入可重构内存功能单元的想法，例如COMRADE[21]、Molen[42]以及文献[25,26]。这些方案中的内存子系统可总结为基于可重构内存功能单元的体系。

这种体系建立在以CPU为核心的体系基础上，同样以软硬件协同工作为背景，不同的是，它增加了一个或多个可重构的内存功能单元。这些内存功能单元在高层综合预设的体系中，可根据综合中对具体应用的分析，进行参数化配置。内存功能单元的作用大致包括以下方面：1)实现片外和片上内存的映射，保持数据一致性；2)为内存访问提供多端口、流水化处理或数据预读取；3)为应用提供定制化的缓存结构。

COMRADE是Hagen G和Andreas Koch等人的高层综合研究项目。其目标体系结构如图5所示。内存子系统的核心是其中的内存管理单元MARC(Memory Architecture for Reconfigurable Computers)[20]。CPU是FPGA上内嵌的Power PC。硬件模块通过MARC访问主存。为增加数据访问的并行性，引入了片上内存作为局部内存，它被CPU和硬件模块共享。源代码中被标识的变量可映射到片上内存。LPU(Local Paging Unit)与MARC连接，用于协调主存和片上内存的映射。具体来说，硬件模块开始工作之前，LPU初始化片上内存的数据，硬件模块完成计算后，LPU再把被修改过的数据写回主存。MARC的具体结构如图6所示。

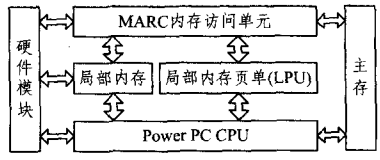


图5 COMRADE目标体系

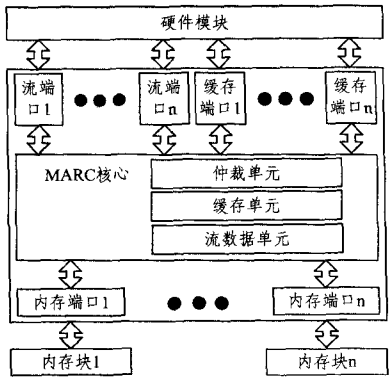


图6 MARC结构

为了充分利用主存的带宽，在MARC的前端，硬件模块可以通过多个并行端口与之连接。在后端，MARC可以通过多个并行端口与多个内存块连接。前端的端口分为缓存端口与流端口。使用缓存端口访问内存时，MARC先进行缓存查找，成功则返回数据，否则返回暂停信号，直到从主存获取数据为止。而流端口则专门用于处理连续多个数据的读写请求。另外，考虑到FPGA的可重构特性，MARC中缓存的行数以及每行的长度可以根据应用的特点在综合时确定。

可以看出，这种体系具有以下优势。首先，这种体系继承了通用计算平台的优点，综合生成的模块易于集成。第二，可重构的内存功能单元根据应用的特性定制，用以提高数据的访问效率。第三，可实现片上内存和主存的映射，有效利用片上内存。然而，增加内存功能单元也带来了一定的访问时间消耗以及逻辑资源消耗。

3 高层综合中内存子系统的优化技术

对于给定的内存子系统体系，高层综合的编译过程中会采取多种优化技术。本节根据编译的前端和后端，总结目前编译中的各种优化技术。

3.1 前端优化

在编译的前端，从代码本身以及建立的模型出发，对代码或对应的模型进行转换，把这种优化划分为前端优化。从目前的研究情况来看，前端优化的主要目标是通过代码或模型转换增加系统的可并行性以及数据访问效率。主要的策略可分为以下方面。

1)数据分割。不同于通用计算平台，可重构体系中可以在编译或运行时为数据分配独立的内存块。多个独立的内存块可以增加数据访问的效率。很多研究中，致力于把一个数

组变量分配到多个独立的内存块，或者在每个内存块中存储多个不同的数组或子数组[29,31,38]。如图 7(a)到图 7(b)的转换中，把 img 分为 imgOdd，imgEven 两个子数组，再把两个子数组映射到不同的内存块，因此代码中可以并行地访问 imgOdd，imgEven。循环内的语句可并行执行，但不增加每次迭代所需的内存访问时间。图 7(b)中进行的循环展开会变得十分有效。然而，这种技术面临两个挑战：1)自动化地进行数组的划分，并且保持程序的正确性；2)选取合适的内存块分配方案，实现数据访问并行化，同时减少片上内存使用。目前的研究中通过预设一些限制条件来解决以上问题，例如已知片上内存的大小、数量。然而，对于代码中存在多个条件判断语句，或者内存访问模式较为复杂的情况，数据分割仍然难以在编译时确定[29]。

```
int img[N][N];
for(i = 0;i<N;i++){
  for(j = 0;j<N;j++){
  … = img[i][j];
  }
}
```

(a)

```
int imgOdd[N][N/2],imgEven[N][N/2];
for(i = 0;i<N;i++){
  for(j = 0;j<N;j = j+2){
  … = imgOdd[i][j/2];
  … = imgEven[i][j/2];
  }
}
```

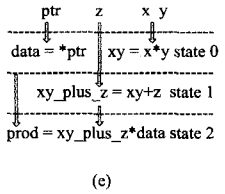
(b)

```
int imgA[N][N],imgB[N][N];
for(i = 0;i<N;i++){
  for(j = 0;j+1<N;j = j+2){
  … = imgA[i][j];
  … = imgB[i][j+1];
  }
}
```

(c)

```
int foo(int *ptr, int x, int y, int z){
        int xy = x * y;
        int xy_plus_z = xy + z;
        int data = *ptr;
        int prod = data * xy_plus_z;
        return prod;
}
```

(d)



(e)

```
unsigned char narrow_array[N];
char a, b, c, d;
for(i = 0; i < N; i+=4){
  a = narrow_array[i];
  b = narrow_array[i+1];
  c = narrow_array[i+2];
  d = narrow_array[i+3];
}
```

(f)

```
unsigned int *wide_array = (unsigned int *) narrow_array;
unsigned int temp;
for(i = 0; i < N/4; i++){
   temp = wide_array[i];
   a = (char)( temp and 0x000000FF);
   b = (char)( (temp and 0x0000FF00) >> 8);
   c = (char)( (temp and 0x00FF0000) >> 16);
   d = (char)( (temp and 0xFF000000) >> 24);
}
```

(g)

图 7　前端编译的代码转换示例

2)数据冗余。数据冗余的基本思路是使数据重复存储于多个存储单元，增加数据的可访问性。如图 7(a)到图 7(c)的转换。这种思路与数据分割相似，可以在数据无法进行分割的情况下进行，也可以作为一种辅助策略和数据分割同时使用。但这种策略存在两个问题，一是冗余数据的一致性问题。这需要系统设计者自行协调，或者由专门的内存功能单元保证。二是存储资源的占用问题。数据冗余需要加倍地占用存储资源，在数据量大的时候不能实现。此时可以考虑先对代码进行划分，例如循环分块(loop tiling)。

3)数据复用。相对于片外内存，片上内存或寄存器具有较短的访问周期，通过数据复用减少对片外内存的访问可以有效地减少延时。例如文献[13,28]中详细地讨论了如何在多层嵌套的循环中把片外内存访问放在较外层的循环中进行，即在外层循环中对一部分数据预读取，存储在片上内存或寄存器，在内层循环中直接访问预读取的数据。当内层循环中需要多次访问同一片外内存地址时，这种方法显得有效。另外，一些针对传统嵌入式平台的数据复用研究也值得借鉴。如文献[17]提出使用循环分割和循环分块重构循环代码，增加数据访问的时间局部性，从而提高每个循环执行过程中的缓存命中率。

4)基于内存访问接口的优化技术。高层综合直接面向可重构的硬件，能获得内存访问接口的信息，根据这些信息，从更底层的方面进行分析，从而优化代码，例如基于访问延时的代码调度。综合中根据不同语句中的数据访问延时，可以对指令进行重新排列，以增加代码的可并行性。例如图 7(d)的程序，循环中第三行读取 * ptr 需要多个时钟周期。根据程序的依赖性分析，可以把第三行代码调整到循环中的第一行，然后在读取 * ptr 的同时计算 xy 及 xy_plus_z，调整后执行状态见图 7(e)。通过这种方式，增加了代码的并行性，减少了每次循环所需的时钟周期。数据封包(data packing)也是基于访问接口的技术。如图 7(f)，需要访问多个 8 位的 char 类型变量，意味着多次的内存请求。然而，对于 32 位的内存通道，可以转换为图 7(g)所示的代码，一次读取中包含了 4 个 char 型数据。这样可以充分利用每次内存访问的带宽，减少访问次数。另外，DRAM 访问的页模式(page mode)可以减少邻近地址的访问时间，也有利于提高内存访问效率。文献[37,40]是通过调整内存访问顺序，增加访问在地址上的连续性来进行优化。

前端编译中构造的模型主要基于程序的数据流图。研究中常常在此基础上，结合系统的内存体系信息以及特定的优化目标，构造新的模型。例如，GAUT 首先通过系统设计者或者前端的编译器(如 SUIF[11])确定内存分配方案，包括变量的实现形式以及变量在内存体系中的存储位置，这些信息将存储在一个内存映射文件中。然后 GAUT 根据内存映射文件以及 DFG，构造出内存限制图(Memory Constraint Graph，MCG)[19]，MCG 不仅可以描述数据依赖关系，也能体现内存块的访问冲突。后端的操作调度将基于 MCG 进行。而文献[38]中，先采集循环结构中对内存的访问地址列表，得出一系列的线性内存模式(liner memory pattern，LPM)，每个 LPM 包含内存访问的首地址、步长以及访问次数。每个 LPM 将映射到特定的内存块。算法再结合程序的读写模式，例如同一操作中的 LPM 不能分配给同一内存块，把 LPM 的分配转化为一个图着色问题，通过求解这个问题一方面为 LPM 分配内存，另一方面增加操作指令的并行性。

上述模型是在 DFG 基础上，根据具体的优化策略而定制的。为方便讨论，本文把这类模型称为类 DFG 模型。类 DFG 模型是直观的有向图结构，一般只体现了系统的某一部分特性。在这些模型的基础上设计的算法往往有一定的缺陷，表现为能有效分析的代码类型受限、与其他优化策略不兼容，或者不能容易地实现自动化执行。而且，面对多重循环嵌套等复杂的代码时，这种直观模型的复杂度会急剧上升。因此，相对于类 DFG 模型，这方面的研究还需要一种更全面和形式化的模型来表示出多种类型的程序，使得多种优化技术可以统一到这种模型上，并实现自动化。多面体模型(Polyhedral Model)[34]可能是一种有效的解决办法。它使用线性代数描述代码中的循环结构，循环的转换可以表示为对应的

代数操作。由于循环执行的时间占了系统的一大部分，很多研究中的优化都是针对循环方面的[19,31,38]。一直以来，有不少基于多面体模型的代码优化研究[12,41]。高层综合领域中，MMAlpha[32]就基于多面体模型进行代码优化，类似的CLooG[33]也被用于高层综合的研究，这种通过代数模型对代码及其转换进行统一化描述的方式，或许可以解决类DFG模型中存在的问题。

3.2 后端优化

前端优化中进行的代码转换或模型修改，需要在后端实现。对于内存子系统，高层综合的后端优化体现在资源分配、配置内存功能单元这两个步骤中。

1)资源分配。一方面是变量的硬件实现。有些变量没有在前端被明确地分配到具体的内存块，则需要在后端进行存储资源的分配，为不同类型的变量分配适当的存储资源。另一方面是内存访问通道的优化。

以C2H为例，会自动地为被综合的函数中涉及的变量分配存储空间或提供数据访问方式，其实现方法如表1所列。

表1 C2H中变量的硬件实现方式

C语言数据类型	硬件实现方式
本地标量变量	基于逻辑单元的寄存器
无初始化的本地数组	片上内存
被初始化的本地数组	主存
指针和非本地数组引用	Avalon-MM主端口
其他类型(如全局变量、静态变量)	主存

其中，主存指CPU所使用的内存，可以是片上或片外内存。Avalon-MM是Altera定义的片上系统中各部件通用的接口体系。C2H通过给指针和外部引用生成Avalon-MM主端口，使得综合出的硬件模块可以通过DMA访问其他内存块的数据，对片外内存的多次访问可流水化执行。需要指出的是，生成Avalon-MM接口可以根据应用的需要，为数据访问提供一个“半独占”的内存访问通道，即接口能独占地进行若干次数据传输而不被系统中其他模块的内存访问请求所干扰。

2)配置内存功能单元。有些综合技术中预设了内存相关的功能单元，例如可定制化的缓存结构、内存控制器等。在后端的处理中，需要根据前端模型分析后获得的数据，对这些功能单元进行参数化配置。例如ROCCC中，针对信号处理中常见的“窗口操作(windows operations)[22]”实施缓存。编译器预先定义一个可参数化的缓存结构，在编译过程中，根据代码获取数据起止地址、窗口大小等信息，然后生成对应的内存访问接口、缓存存储单元以及地址生成单元。

这种可配置的单元充分利用了FPGA可重构的特点，在综合阶段为系统生成定制化的方案。这种方案是基于特定应用的，因此能有效地减少不必要的逻辑资源占用，获得更小更高效的硬件实现。

表2 内存子系统优化技术在通用计算和可重构计算中的适用性比较

技术/策略	通用计算体系	FPGA可重构体系
数据分割		√
数据冗余		√
数据复用	√	√
基于访问延时的代码调度		√
基于DRAM页模式的优化	√	√
数据封包	√	√
内存访问流水化	√	√
配置内存功能单元		√

3.3 小结

本节对高层综合中内存子系统的优化技术，按编译前端和后端进行了归纳。其中，有些技术在通用计算体系的编译中已经得到深入的研究，例如数据复用。为了分析在高层综合中这些技术的不同特点，把这些技术在通用计算平台和FPGA可重构平台中的适用性做了对比，见表2。

对于通用计算体系，由于内存体系是确定的，不可重构的，而且内存是完全共享的，由CPU统一分配的，因此其中的数据分割、冗余策略以及重配置内存功能模块并不适用。编译过程中编译器无法获知硬件细节，因此基于访问延时的代码调度不能实现。可以看出，对于高层综合，针对内存子系统的优化技术有以下特点：1)可定制性，体现为功能单元以及内存体系的可定制性；2)硬件细节感知，不同于通用计算平台，作为可重构的体系，综合过程中不仅是软件代码的编译，还包括了硬件电路的生成。芯片的硬件细节会暴露给综合工具。综合过程中可根据这些知识生成合适的方案。这两个特性为高层综合生成符合优化目标的结果提供了优势，但同时，设计空间(design space)的扩大也为综合算法的设计带来了困难。综合过程中可控制的参数过多会导致难以对方案进行快速评估，甚至无法生成方案。因此，目前的研究中，很多采用了：预设内存子系统的体系、优化技术以及优化目标，然后根据模型分析的结果，获取目标系统的特征参数，最后根据参数实例化内存子系统。

4 分析与评价

本节将从内存的分配、访问以及缓存方面，对FPGA高层综合中内存子系统的体系进行分析与评价。表3中对3种体系的特性做了比较。表3中所列的“高-中-低”以及“多-少”是对3种体系进行互相比较得出的相对性描述。

表3 3种体系的特性对比

特性	DSP型	以CPU为核心	基于可重构内存功能单元
内存共享性	低	中	高
内存分配的自由度	大	小	大
输入输出的类型	数据	数据/内存指针	数据/内存指针
内存带宽利用	高	低	高
数据访问的可并行性	高	低	中
缓存有效性	高	低	中
逻辑资源占用	多	少	中
可定制性	高	低	中

内存分配的方面。DSP型体系中，综合的目标是一个独立运行的硬件模块，没有全局变量和局部变量之分，综合中假定模块可以独占内存。因此，相对于其他两种体系，这种体系下内存分配的自由度最大。而且分配后内存块在系统内的分布也可以更灵活。对于以CPU为核心的体系，内存分配就是主存与局部内存之间的选择。一般地，主存用于全局和静态变量，而局部内存则用于局部变量。为了提高内存读取效率，可以在硬件模块执行之前，把数据先读入局部内存(片上内存)。然而，由于片上内存在FPGA上不多，往往不足以完整地存储待处理数据。可以对代码进行循环分块，在运行的时候分块读取，但一方面，目前自动化的分块算法还未被很完善地实现，另一方面读取本身也会带来一定的延时。因此，以CPU为核心的体系，内存分配简单，但不能充分利用片上内存，也不容易达到很好的性能。基于可重构的内存功能单元的体系，有专门的内存功能单元，可克服这一困难。通过协调

局部内存(片上内存)和主存,两者可进行动态的映射,从而实现了内存的动态分配。同时,多个模块可以共享数量有限的片上内存。在减少资源占用的同时保证了内存的高效访问。

内存访问方面。以上3种体系均可通过DMA方式进行内存访问。其区别是,DSP型和基于可重构内存功能单元的体系可以针对特定的应用重配置内存访问单元。这种定制化的结构可以以更少的资源为代价,生成更高效的结构,充分利用内存带宽。

对于内存访问的并行性,由于DSP型为单个应用定制,不存在多模块访问冲突的情形,而且片上内存可根据数据通路的需要,分布于模块内部,因此数据访问的可并行最高。而基于可重构内存功能单元的体系,通过定制的模块,可以多端口地进行访问,主内存和局部内存的映射使得局部内存的可使用率大大增加,这些都增强了数据访问的可并行性。对于以CPU为核心的体系,数据访问的可并行性体现在独占的局部内存。

缓存方面,3种体系中都有缓存结构。从缓存的有效性,即缓存的可命中率来看,DSP型最高,基于可重构内存功能单元的体系次之,以CPU为核心的最低。这是因为DSP型的系统内,只有单个应用,缓存存储结构中不存在其他模块的干扰(缓存替换),基于可重构内存功能单元可通过组织多个缓存存储结构缓解这个情况。再者,这两者可以根据系统中数据访问的特点,生成定制化的缓存结构,因此,这两个体系中的缓存表现较好。而以CPU为核心的体系,可以在内存控制器添加缓存单元,但这种缓存单元是通用的,没有根据应用而定制。

逻辑资源占用方面。由于DSP型的体系中,包含了整个系统内存访问所需的逻辑,新增模块时,这些逻辑不能被共享,因此这种类型的内存子系统逻辑资源占用最多。而以CPU为核心的体系,共享统一的内存控制器,在新增模块时,只需生成内存访问逻辑,占用逻辑资源最少。对于基于可重构内存功能单元的体系,共享内存控制器,但新增模块时,视不同综合技术而定,需要对共享的内存功能单元进行修改或增加模块专用的功能单元。

结束语 FPGA高层综合中的内存子系统体系可分为3类:DSP型体系、以CPU为核心的体系、基于可重构内存功能单元的体系。其中,DSP型体系占用内存和逻辑资源较多,容易取得较好的性能,但不与外界共享内存,适合于模块独立工作的情形,或者以一个独立于外部内存子系统的计算核的形式集成到嵌入式系统中。以CPU为核心的体系,新增模块时所需内存和逻辑资源最少,综合生成的模块能很好地集成到系统中,可用于软硬件协同工作情形,但不容易达到较好的性能,适合于对性能要求不高,但需要多模块并行工作的系统。基于可重构内存功能单元的体系,在资源占用方面处于其他两种体系之间,适合于软硬件协同工作的情形,同时也能达到相对较好的性能。

对于内存子系统的优化技术,一方面,可以参考通用计算平台中的编译优化技术。另一方面,高层综合具有两个优势:1)底层硬件的可定制性;2)硬件细节感知,可针对具体系统和硬件信息进行优化。

针对FPGA高层综合中的内存子系统,仍然存在很多问题及值得研究的方向:

1)片外/片上内存映射的问题。对于FPGA,少量的高性能片上内存以及大量需要处理的数据始终是高层综合中必须面临的问题。基于可重构内存功能单元的体系提供了片上与片外内存动态映射的可能性,这种映射可以使片上内存得到尽可能的使用。但目前映射的算法仍然需要进一步的探讨。

2)建模问题。高层综合直接面对可重构的硬件,编译时面对的设计空间比通用计算平台的大,即多种系统属性需要考虑,如内存分块、访问冲突、芯片资源等。常见的类DFG模型,为每个策略单独定制,不能很好地被集成和实现自动化,因此,需要统一的数学化模型对代码进行建模。多面体模型是一种可能的方法。

3)内存组织体系方面。目前很多算法和研究都基于系统设计者指定的内存组织体系进行,即内存块的大小、数量、层次等都是确定的。然而,这些确定的信息是硬件上固有的或者设计者手工生成的。而针对可重构的FPGA,可以由编译器针对目标系统,自行生成片上内存的组织体系。需要指出,这种方法虽然从理论上可以实现更优的内存体系,但综合过程中的设计空间会进一步增大,为编译算法带来更大的挑战。

4)多模块协同工作。目前的高层综合技术主要针对单个模块的编译。然而,多模块下各个模块的协同工作也值得研究,因为多模块协同意味着缓存一致性、共享数据等方面的进一步考虑。高层综合中的内存子系统,需要针对多模块的情形,配置相应的数据同步单元和存储单元。例如,现阶段高层综合实现的流水线是指令级的,可以考虑在多个模块中实现模块级的流水化。

参 考 文 献

[1] Corporation A. Altera Product Catalog[R/OL]. http://www.altera.com,2011

[2] 朱继祥,李元香,夏学文,等. 基于演化硬件的在线自适应系统[J]. 计算机科学,2009,36(7):267-269

[3] 栾静,顾君忠. 基于SystemC的嵌入式系统设计的描述模型[J]. 计算机科学,2005,32(8):209-212

[4] Coussy P, Gajski D D, Meredith M, et al. An Introduction to High-Level Synthesis[J]. Design & Test of Computers, 2009, 26(4):8-17

[5] Cardoso J M P, Diniz P C. Compilation Techniques for Reconfigurable Architectures [M/OL]. http://www.springerlink.com,2008

[6] Graphics M. Catapult C Synthesis[EB/OL]. http://www.mentor.com/esl/catapult,2011

[7] Coussy P, Morawiec A. High-Level Synthesis-From Algorith to Digital Circuit[M/OL]. http://www.springerlink.com,2008

[8] Martin G, Smith G. High-Level Synthesis: past, present, and future[J]. Design & Test of Computers, 2009, 26(4):18-25

[9] Nishimura M, Nishiguchi K, Ishiura N, et al. High-Level Synthesis of Variable Accesses and Function Calls in Software Compatible Hardware Synthesizer CCAP[C]// The 13th Workshop on Synthesis and System Integration of Mixed Information Technologies. 2006

[10] Canis A, Choi J, Aldham M, et al. LegUp: high-level synthesis for FPGA-based processor/accelerator systems [C] // Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays(FPGA '11). 2011:33-36

[11] The Stanford SUIF Compiler Group [EB/OL]. http://suif.stanford.edu,2011

[12] The LLVM Compiler Infrastructure Project[EB/OL]. http://www.llvm.org,2010

[13] Liu Q, Constantinides G A, Masselos K, et al. Data-reuse exploration under an on-chip memory constraint for low-power FPGA-based systems[J]. IET Computers & Digital Techniques, 2009,3(4):235-246

[14] Kumar T S R. On-Chip Memory Architecture Exploration of Embedded System on Chip[D]. Bangalore: Indian Institute of Science,2008

[15] 张钦,韩承德.基于存储技术的高速嵌入式处理器的设计与实现[J].计算机学报,2007,30(5):831-837

[16] Ozturk O, Kandemir M, Irwin M J, et al. Multi-Level On-Chip Memory Hierarchy Design for Embedded Chip Multiprocessors [C] // Proceedings of the 12th International Conference on Parallel and Distributed Systems. 2006:383-390

[17] Tembe W, Pande S. Loop restructuring for data I/O minimization on limited on-chip memory embedded processors[J]. IEEE Transactions on Computers,2002,51(10):1269-1280

[18] Udayakumaran S, Barua R. Compiler-decided dynamic memory allocation for scratch-pad based embedded systems [C] // Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems. 2003:276-286

[19] Corre G E, Senn E, Bomel P, et al. Memory accesses management during high level synthesis[C] // International Conference on Hardware/Software Codesign and System Synthesis. 2004:42-47

[20] Lange H, Koch A. Memory Access Schemes for Configurable Processors[C] // Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications. 2000:615-625

[21] Lange H, Thielmann B, Koch A. A Flexible Compute and Memory Infrastructure for High-level Language to Hardware Compilation[C] // 2010 International Conference on Field Programmable Logic and Applications. 2010:475-482

[22] Guo Z, Buyukkurt B, Najjar W. Input data reuse in compiling window operations onto reconfigurable hardware[C] // Proceeding of the 2004 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems. 2004:249-256

[23] Villarreal J, Park A, Najjar W, et al. Designing Modular Hardware Accelerators in C with ROCCC 2.0[C] // 18th IEEE Annual International Symposium on Field-Programmable Custom Comp-uting Machines. 2010:127-134

[24] Corporation A. Automated Generation of Hardware Accelerators With Direct Memory Access From ANSI/ISO Standard C Functions[R/OL]. http://www.altera.com,2006

[25] Sima V, Bertels K. Runtime Memory Allocation in a Heterogeneous Reconfigurable Platform [C] // International Conference on Reconfigurable Computing and FPGAs. 2009:71-76

[26] Séméria L, Sato K, De Micheli G. Synthesis of hardware models in C with pointers and complex data structures[J]. IEEE Transactions on Very Large Scale Integration Systems, 2001, 9(6): 743-756

[27] 于苏东,刘雷波,尹首一,等.嵌入式粗颗粒度可重构处理器的软硬件协同设计流程[J].电子学报,2009,37(5):1136-1140

[28] Liu Q, Constantinides G A, Masselos K, et al. Compiling C-like Languages to FPGA Hardware: Some Novel Approaches Targeting Data Memory Organization [J]. The Computer Journal, 2011,54(1):1-10

[29] Huang C, Ravi S, Raghunathan A. Generation of distributed logic-memory architectures through high-level synthesis[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2005,4(11):1694-1711

[30] Gupta S, Dutt N, Gupta R, et al. SPARK: A High-Level Synthesis Framework For Applying Parallelizing Compiler Transformations[C] // 16th International Conference on VLSI Design. 2003:461-466

[31] Seo J, Kim T, Panda P R. An integrated algorithm for memory allocation and assignment in high-level synthesis[C] // Proceedings of 39th Design Automation Conference. 2002:608-611

[32] IRISA. MMAlpha project[EB/OL]. http://www.irisa.fr,2010

[33] The CLooG Code Generator in the Polyhedral Model[EB/OL]. http://www.cloog.org,2011

[34] 陆平静,车永刚,束尧,等.多面体表示技术及在程序性能优化中的应用[J].计算机工程与科学,2008,30(9):137-140

[35] GAUT[EB/OL]. http://Web.univ-ubs.fr/gaut,2011

[36] Putnam A, Bennett D, Dellinger E, et al. CHiMPS: A C-level compilation flow for hybrid CPU-FPGA architectures [C] // 2008 International Conference on Field Programmable Logic and Applications(FPL 2008). 2008:173-178

[37] Panda P R, Dutt N D, Nicolau A. Exploiting off-chip memory access modes in high-level synthesis[C] // Proceedings of the 1997 IEEE/ACM International Conference on Computer-aided Design. 1997:333-340

[38] Asher Y B, Rotem N. Automatic memory partitioning: increasing memory parallelism via data structure partitioning[C] // Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. 2010:156-162

[39] Edwards S A. The Challenges of Synthesizing Hardware from C-Like Languages[J]. Design & Test of Computers, 2006, 23(5): 375-386

[40] Christophe A, Alain D, Alexandra P. Optimizing DDR-SDRAM communications at C-level for automatically-generated hardware accelerators an experience with the Altera C2H HLS tool[C] // 21st IEEE International Conference on Application-specific Systems Architectures and Processors(ASAP). 2010:329-332

[41] Wikipedia. Frameworks supporting the polyhedral model [EB/OL]. http://en.wikipedia.org/wiki/Frameworks_supporting_the_polyhedral_model,2011

[42] Breijer S D. Memory organization of the Molen prototype[D]. Delft University of Technology,2007