HLS 与 Aladdin 对比

January 29, 2018

1 System Overview

HLS 的典型流程如下图所示，在这一流程中 HLS 可以自动帮用户做的：1. 根据控制流生成合适的 FSM 结构; 2. 确定硬件资源; 3. schedule、资源分配; 4. 生成合适的中间表示。需要用户来设置的：1. 并行度设置; 2. 模块之间如何交互; 3. 数据存储结构选择; 4. TIming Budget 设置。

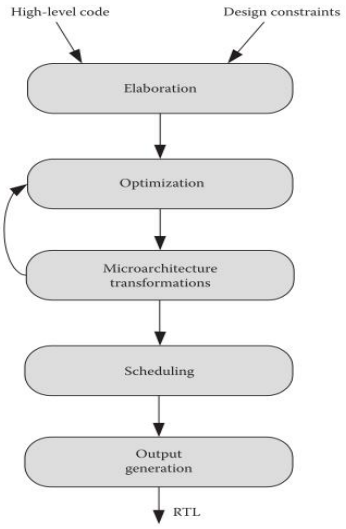

FIGURE 11.3 Typical HLS flow.

Figure 1: HLS 典型流程

2 Input

Aladdin 和 HLS 的输入都是高层语言的描述的算法和需要用户决定的硬件约束或选择。不同地方是 HLS 输入是高层语言的代码，而 Aladdin 输入的是代码运行的指令 trace。原始代码包含更完整的逻辑结构而 trace 只包含执行的分支，但 trace 会包含执行过程中更多细节如实际的访存地址、寄存器分配等。

Table 1: Input 对比

Input	HLS	Aladdin
算法描述	High-Level code	C-code trace
硬件约束	Hardware Constrain	Hardware Config

3 Elaboration

第二步是要将输入的高层代码和硬件约束转化为可以做优化和处理的中间表示，一般基于图。HLS 使用 CDFG 即 control flow graph 和 data flow graph，前者表示控制逻辑后者表示计算，两者的联系是 DFG 中的节点对应 CFG 的一条边。而 Aladdin 使用 data dependency graph，类似 DFG，但因为输入是 trace 所以本身也体现了实际执行的控制流，没有再使用 CFG，可以理解为为 DFG 映射到 CFG 的边中并根据实际执行结果展开后获得的 graph。

Table 2: Elaboration 对比

Elaboration	HLS	Aladdin
中间表示	CDFG	DDDG

4 Optimization

第三步是要对中间表示的 graph 做优化。在 HLS 中，DFG 有两大部分的优化方法，其一来自于编译器如 CSE 等见图一，另一部分为硬件相关：

* 位宽调整：根据数值的位宽或数值推断所需要的位宽。一般适用于 Affine arithmetic，即与常数的乘加计算。在 Aladdin 中分析 CNN 一类应用时，位宽的调整来自于量化算法的分析。

* Chain-to-Tree：时序相关，将链式转化为树型，例如加法链转化为加法器树，可以降低 latency。

* Mux-Retiming：调整 MUX 与计算操作的位置，方便优化。

此外，还有针对 CFG 的优化方式，主要是准对条件判断 (if,case) 和循环 (Loop)。Aladdin 中，由于 trace 中包含了访存等信息，优化主要体现在对 Load/Store 节点的处理。

Table 3: Optimization 对比

Optimization	HLS	Aladdin
Compiler-related	Dead Code Elimination Constant Folding and Propagation Common Subexpression Extraction Strength Reduction ...	
Hardware-related	Bit trimming Chain-to-Tree Mux-retiming ...	Remove Repeated Load/Store Store Buffer

5 Hardware Transformation

第四步则是将用户定义的硬件约束或设置在中间表达中实现，主要是对 loop、function、array、pipeline 等问题的处理。在这一阶段两者考虑的问题比较相似，但 HLS 具体的处理手段细节还不清楚。

5.1 Loop

Loop Unrolling，即通过将循环体复制多份从而展开了循环结构。完全展开和部分展开。Aladdin 根据用户定义的 unroll factor 修改不同 loop 间的依赖关系，实现循环的展开。

Loop Break：对于未展开的循环体中要执行的计算，其可能有多个 cycle，如何插入合适的 state 将计算分配到不同的 cycle。一般是自动的，Aladdin 中根据定义的 cycle time，当 op 的 latency 超过 cycle time 时即 break。

Loop Inversion：将 while 调整成 do-whilt。

Loop-invariant Code Motion：把循环中不变的操作移动到循环外。

Loop Fusion：当两个循环迭代次数相同且没有数据依赖时，可以合并为同一个。

5.2 Functions

函数可以用来体现不同的硬件模块、相应的层次结构以及硬件资源的复用，但函数的使用不全是为了以上原因，使用 inline 函数则可以避免相应的优化。对于 latency 固定的函数，也可以做流水线的处理。在 Aladdin 中每个节点的属性有函数这一项，可以用来实现复用等功能。

5.3 Array

Array 在高层语言中用来表示数据，HLS 和 Aladdin 对 array 的处理相似，主要有以下三种方式：使用 Memory with read/write port 实现，N latency。使用 Register Bank 实现，zero latency。Flatten array：将 array 展开为相互独立的变量。当 array 没有被 flatten 时，其访问会被看作单独的操作 (graph 中的节点)，同时还要考虑访问同一

地址时的冲突和数据依赖。Aladdin 区分计算节点和访存节点，并根据不同存储的访存机制建立模型，包括仿真方式、latency 和功耗等。

5.4 Pipeline

Loop pipeline 只能应用于没有 nested loops 的循环，即需要将内层循环与外层 merge 才可以。Pipeline 的变量包括 stage(分为几级)、initial interval(每个 stage 耗用的周期数) 和 latency(执行完所有 stage 耗用的周期数)。scheduling pipeling 即选择合适的 stage nums 并将 loop 内的操作分配至各个 stage 中，这个过程需要满足数据依赖的约束以及额外的 pipeline 约束。在 HLS 中，需要注意 pipeline 仍要保证原有的依赖关系。此外，还有不同 loop 之间的 “loop-carried dependency”。而在 Aladdin 中，由 trace 可获得循环间数据的依赖关系，因此 pipeline 的实现主要通过修改不同的 loop 间的依赖关系

6 Scheduling

在 HLS 中，schedule 可以看作两个问题，1. 将 dfg 中的操作 map 到 cfg 的边上 (schedule);2.。将 dfg 的操作 map 到对应的硬件资源中 (binding)。优化的目标是 latency 或 resources，由于有许多的可能情况，需要在设计空间中不断迭代 (NP-complete)。在 Aladdin 中，schedule 的问题根据指令 trace 和数据依赖决定，不需要设置约束，而是由 default 的 schedule 规则，即依次将操作放入 cycle 中，当操作的总 latency 超过一个 cycle 时，则放入下一个 cycle; binding 的规则是同一个 function 内乘法器复用加法器不复用，最终估算出需要的 resource 和 latency，可以看做是 HLS 做探索时的一个情况。

在上述流程结束后，HLS 可以生成对应的 verilog 代码 (RTL)，Aladdin 则给出硬件执行的性能、功耗、面积等信息。