Developer's Guide to REDEFINE Compiler Code

May 4, 2017

This document is a developer's guide to navigating around REDEFINE compiler. Details of most important code sections and where they belong maybe found here.

1 Clang Frontend

Clang frontend is used to generate LLVM IR from C code. State of the art clang was borrowed with minor changes as follows:

- Elimination of memcpy instruction insertion: memcpy is not supported as an instruction or a library call. It is supposed to expand to load-store operations instead. (Check this: This change maybe found in CGBuiltin.cpp in the older LLVM versions).
- Addition of REDEFINE target and ABI: REDEFINE target addition is found scattered in many clang files, but its easy to patch this change. ABI is in one file only and indicates that structs ought to be passed by value too instead of reference, no matter what the size of the struct.
- Addition of sin, cos, abs intrinsics in a few files and their expansion to the IR intrinsic.

2 HyperOp Creation Pass

HyperOp creation pass is implemented as an optimization pass and maybe found in lib/Transforms/HyperOpCreationPass.cpp. It generates parallel IR

from the serial IR such that each function represents a HyperOp and communication is encoded via custom metadata. Details of the metadata format maybe found in IR specification document. The pass identifies the entry point of the kernel i.e., redefine_start function and traverses every function call. Note that all calls to functions that are not a part of a call cycle are inlined. Loops with carried dependences with unit loop carried dependence distance get serialized. The code starts at *runonModule* method and creates new functions corresponding to the HyperOps and deletes the old ones from the module. This pass is slightly tricky to follow but one doesn't need to know the internals unless they are making fixes/enhancements. The biggest caveat in this pass is that it does not create an internal HIG data structure and maintains the structure indirectly, making it difficult to run other utilities that may have been written for HIG data structure.

3 LLC

Low level compilation encompasses the following steps in that order:

- Instruction Selection Pass i.e., REDEFINEISelDAGToDAG.cpp invokes HyperOp Interaction Graph utilities in the order in which the calls are made. The parser creates HIG data structure, sets hardware parameters of REDEFINE onto HIG,