Placement-and-routing-based Register Allocation for Coarse-grained Reconfigurable Arrays

Bjorn De Sutter
Ghent University - IMEC
brdsutte@elis.ugent.be

Paul Coene Tom Vander Aa
Interuniversity Micro-Electronics Center
{coene,vanderaa}@imec.be

Bingfeng Mei
meibf@yahoo.com

Abstract

DSP architectures often feature multiple register files with sparse connections to a large set of ALUs. For such DSPs, traditional register allocation algorithms suffer from a lot of problems, including a lack of retargetability and phase-ordering problems. This paper studies alternative register allocation techniques based on placement and routing. Different register file models are studied and evaluated on a state-of-the art coarse-grained reconfigurable array DSP, together with a new post-pass register allocator for rotating register files.

Categories and Subject Descriptors D.3.4 [*Programming Languages*]: Processors—code generation, retargetable compilers

General Terms Algorithms, Design, Performance

Keywords register allocation, placement and routing, coarse-grained, reconfigurable arrays

1. Introduction

Many applications contain loops that exhibit large amounts of parallelism. To exploit this, processors have to offer a high execution bandwidth. In embedded application domains, VLIW DSPs are popular because of their good performance/power ratio and compiler support. Unfortunately, VLIW architectures do not scale well, as the power consumption and delay of a register file (RF) scales super-linearly with its number of ports. While clustering overcomes this problem to some extent, clustered VLIWs still feature quite large, multi-port, and hence power-hungry, RFs. Alternatively, direct connections between arithmetic logic units (ALUs) can be added to the DSP data paths. For example, by making the forwarding paths on pipelined processor explicitly accessible through the instruction set architecture (ISA), many RF accesses can be omitted for short-lived values [25]. This creates opportunities for lowering the number of RF ports without paying a price in obtained IPC.

Another example of architectures with explicit connections between ALUs are coarse-grained reconfigurable arrays (CGRAs) [22, 24], of which Figure 1 depicts an example. CGRAs can be seen as coarse-grained FPGAs in which look-up tables have been replaced by word-wide ALUs and RFs, and in which a new CGRA configuration can be loaded every cycle. As such, a CGRA can also be seen as an extension of a clustered VLIW to a more generic, 2D type of VLIW in which all components, including the multiplexers of the interconnect network, are programmed explicitly every cycle with an ultra long instruction word. An important difference between CGRAs and clustered VLIWs is that a typical CGRA ALU is not connected to a single RF via multiple implicit connections. Instead each ALU can be connected to multiple RFs via a heterogeneous interconnect topology. As such, it can occur that the two or more inputs of an ALU are connected to different sets of RFs. Conversely, a single RF port may be shared by more than one ALU. This allows data to flow between multiple RFs and ALUs, which is required for executing complex loops at high instruction-level parallelism (ILP), while still using single-ported, power-efficient RFs. Also, to reach higher clock frequencies, additional pipelining latches can be inserted nearly anywhere in the interconnect network of the data path. Because CGRA instances optimized for different application domains can differ significantly in the number of RFs and ALUs and in interconnect topology, CGRA code generation techniques ideally should be retargetable. For example, they should support RFs that are tightly coupled to single ALUs, as well as RFs that are shared between a number of ALUs.

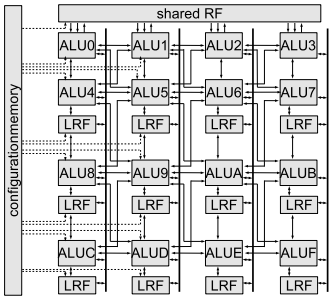

Figure 1. An example CGRA featuring one shared RF, 16 ALUs, 12 local RFs, and a sparse interconnect.

To the best of our knowledge, traditional code scheduling and register allocation strategies fail in targeting CGRA architectures or clustered VLIW architectures with explicit forwarding paths. For one thing, when there are a large number of small RFs, splitting the register allocation in the separate cluster assignment [19, 10] task with or without instruction replication [2] and intra-cluster register allocation does not work anymore. Furthermore, with sparsely connected RFs and ALUs, the assumption of most code schedulers that an ALU is connected to its corresponding RF through exclusive implicit connections does not hold anymore. So even code generation algorithms that integrate cluster assignment with instruction scheduling and register allocation in one phase [31] do not scale to CGRA-like architectures.

LCTES '08 June 12–13, 2008,Tucson, Arizona, USA.