An Integer Linear Programming Approach to Classify the Communication in Process Networks

Alexandru Turjan, Bart Kienhuis, and Ed Deprettere
Leiden Institute of Advanced Computer Science (LIACS)
Leiden, The Netherlands
aturjan@liacs.nl

Abstract. New embedded signal processing architectures are emerging that are composed of loosely coupled heterogeneous components like CPUs or DSPs, specialized IP cores, reconfigurable units, or memories. We believe that these architectures should be programmed using the Process Network model of computation. To ease the mapping of applications, we are developing the Compaan compiler that automatically derives a Process Network (PN) description from an application written in input Matlab. In this paper, we investigate a particular problem in nested loop programs, which is about classifying the interprocess communication in the PN representation of the nested loop program. The global memory arrays present in the Matlab code have to be replaced by a distributed communication structure used for sending data to the network processes. We will show that four types of communication exists, each exhibiting different requirements when realizing them in hardware or software. We present two compile time tests that decide the type of the communication corresponding to a particular static array. These tests are based on Integer Linear Programming and have become an important part of our Compaan compiler.

1 Introduction

Applications that are envisioned for the next decade in the area of multi-media, imaging, bioinformatics, and classical signal processing have a ferocious appetite for compute power. To satisfy this appetite, new embedded signal processing architectures are emerging. These are typically composed of loosely coupled heterogeneous components that exchange data using programmable interconnections such as a switch matrix or a network on chip (NoC). The components can be CPUs or DSPs, specialized IP cores, reconfigurable units, or memories. Also, a central control microprocessor is present for the configuration of the components at run-time using a low-bandwidth bus. An impression of such architecture is shown in Figure 1. Aside from the use of specialized heterogeneous components and instruction level parallelism on the CPUs, these architectures will employ more and more task level parallelism to deliver the required performance.

From a technology standpoint, companies and research institutions are already able to build instances of the presented architecture. Three examples are, for example, the Picochip from PicoChip, the Virtex Pro from Xilinx, and although still in research, the SpaceCAKE architecture from Philips. The PicoChip combines 430 simple RISC architectures on a single die [10]. Xilinx combines FPGA technology with four embedded
Embedded Signal Processing Architecture consisting of loosely coupled heterogeneous components like CPUs, DSPs, Specialized IP cores, Reconfigurable Components, and Memories.

PowerPCs on their Virtex-Pro chips [22]. Philips is researching the SpaceCAKE architecture which consists of a heterogeneous mix of memories, CPUs like the MIPS or the ARM and DSPs [15]. We observe that the problem with these heterogeneous architectures is not building them, but programming them, i.e., writing programs that take advantage of the offered heterogeneity and task-level parallelism. Writing a program for such an architecture means partitioning the application over the various components of the architecture and generating embedded software for each component. In case of a CPU or DSP, this means writing a piece of C code and in case of an IP core or reconfigurable unit, it means writing a VHDL or Verilog program. This partitioning and compilations are very time consuming and error prone, which makes the deployment of the heterogeneous architectures difficult.

We believe that the PN model of computation is suitable to cope with the heterogeneity of the new embedded architectures. Still, writing application in a PN format is a time consuming process. Therefore, we are developing the Compaan compiler [7, 14], which automatically derives a Process Network (PN) description from an application written in Matlab. Using Compaan, we can quickly derive PNs from existing applications and then map them on heterogeneous embedded architectures.

In this paper, we investigate a particular problem in the Compaan compiler regarding the ability to classify the interprocess communication in a PN. Within Compaan, we distribute the memory arrays present in the input program over a number of communication structures used for transmitting data between network’s processes. We will show that four types of communication exist, each exhibiting different requirements when a hardware or software implementation is generated. Each communication structure in a process networks is classified to one of the four types, using two tests that are described in this paper. These tests can be performed at compile time and have become an important part of our Compaan compiler.

This paper is organized as follows. In Section 2, we explain our Compaan compiler project. The four kinds of communication in a process network are presented in Section 3. In Section 4 we presented related work. In Section 5 and Section 6, we present the two tests used for a compile time classification of the communication structures in a process network. In Section 7 we present some results and we conclude this paper in Section 8.