DATAFLOW SOFTWARE PIPELINING: A CASE STUDY

Guang R. Gao Z. Paraskevopoulos

School of Computer Science
McGill University

Effectively exploiting parallelism embedded in loops with predominately array operations have long been a challenge to multiprocessor architectures. This paper reports a case study of dataflow software pipelining – an efficient code mapping strategy for array operations in loops on a highly pipelined static dataflow processor architecture based on argument-fetching data-driven principle [1, 4]. The new architecture has the potential of keeping the instruction processing pipeline fully busy as long as the structure of the program can keep enough enabled instructions for concurrent execution – one main objective of dataflow software pipelining proposed in this paper.

The argument-fetching dataflow processor consists of two processing modules: the pipelined instruction processing unit (PIPU) and the dataflow instruction scheduling unit (DISU) connected through a fire link and a done link. DISU holds the dataflow signal graph of the collection of dataflow instructions allocated to the processing element, and maintains a record of which instructions are enabled. PIPU is an instruction processor that uses conventional techniques to achieve fast pipelined operation. The PIPU executes enabled instructions and informs the DISU when each instruction finishes execution. The fire link is for transmitting the addresses of enabled instructions from the DISU to the PIPU. The done link is for transmitting to DISU the completion signals containing the address of each instruction which have completed their processing in PIPU, together with a condition code used by the DISU to control the sending of conditional signals.

Program mapping is performed on units of program text that define the major structured values involved in a computation. These program units are compiled into units of code called code blocks. Code blocks are the units of source program to be handled by our code mapping strategy and will be decomposed and assigned to the processing elements of a dataflow multiprocessor computer. In the dataflow graph based on the pipelined mapping of a code block, successive elements of the input arrays will be fetched and fed into the dataflow graph, so that the computation may proceed in a pipelined fashion. This is called dataflow software pipelining – the arcs drawn between actors correspond to addresses in stored dataflow machine code, and not to the wired or inlacements between logic elements. For more detailed discussion, concerning the software pipelining, the readers are referred to [2].

As a case study, three code mapping schemes of the one-dimensional Laplace equation solver are implemented and investigated in detail: (1) not software pipelined, (2) software pipelined, (3) software pipelined and balanced [2]. They are tested on an architecture testbed being developed at McGill University.

A considerable performance increase has been observed by using a pipelined code mapping scheme, and the balanced scheme outperformed the unbalanced scheme. For a 256 elements array, the speed-up are 1.15 for the non-pipelined versus 5.28 for the pipelined version. A much higher utilization factor is also observed for the pipelined version. The balanced version shows a performance increase over the unbalanced version. Similar encouraging results are also observed for a set of Livermore loops (in SISAL) and is the subject of another paper. The results provide strong experimental evidence that the degree of software pipelining has a direct impact on the overall performance, thus confirming the predictions of our earlier work. In the full paper, we include a comparison with other software pipelining techniques [5, 6].


