Faster Nested loops by Overlapping Input/Output with Computation for Dynamically Reconfigurable Processors

Koichi Araki (610004)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 7, 2008

Keywords: Dynamically Reconfigurable Processor, Dynamic Partial Reconfiguraion, Data Input and Output, Context Reduction.

1 Introduction

Dynamic reconfigurable processors can generate different digital circuits more than once and reconfigure in a single clock cycle. Our target device is the Dynamically Reconfigurable Processor (DRP) developed by NEC electronics. DRP is more suitable than Field Programmable Gate Array (FPGA) for numerical processing such as stream processing. This is attributed to the DRP processing element (PE) property of including local ALU. DRP can store sixteen configuration data (called Contexts) locally. If processing requires more contexts, it must download the extra cotexts from host memory. The charactiristic of DRP is that it can run dynamic partial reconfiguration. DRP contains more than one core, each called Tile. DRP can run dynamic partial reconfiguration, since a Tile can be dynamically reconfigured independent of other Tiles.

The reconfiguration feature of DRP, allow DRP to generate highly parallel circuits to deal with the statements within iterative processing much faster than generic CPU. Since I/O bit rate of DRP is small, it is not possible to input a lot of data and output resulting data in a single clock cycle.

As a result, the typical processing flow on DRP consists of three phases. The first phase is to input the data required for computation possibly in many clock cycles. The data input to DRP is stored in internal memory modules distributed inside DRP (VMEM/HMEM). The second phase is to compute from data stored in VMEM/HMEM, the resulting data is stored back in VMEM/HMEM. Finally the last phase is to output resulting data in many clock cycles. If the processing repuires a lot of input data or output data, the execution time of the first phase or third phase occupies most of the execution time.

For DRP input and output time reduction is an important theme. Our work is concerned with this topic and also focuses on context reduction. Since downloading more contexts from host memory occurs during execution and becomes a bottleneck. Thus for DRP it is very important to reduce the number of required contexts.

2 Proposed Method

The goal of our work is to reduce input/output time and to reduce the number of contexts required for processing. To achieve this, our work proposes two contributions. First is to hide input/output time by overlapping input/output with computation. The second is to reduce the contexts by creating data-driven configurations.

To overlap input/output with computation, each Tile represents the circuits for dealing with the statements within the iterative instructions. These circuits provide low level parallelism on DRP. While a Tile is running the computation or being reconfigured, other Tiles are running data input or output. This can conceled the input clock cycles effectively which reduce the overall number of cycles. DRP contexts can be produced by partitioning Control Data Flow Graph (CDFG). If split into more than one smaller CDFG, the operating frequency is increased, but so does the number of data circuits (contexts). Unfortunately, this would have the side effect of decreasing the operating frequency back again, since more contexts downloading would occur.

In each cycle, according to the DRP input bit rate only limited input is possible. Contexts are generated by splitting CDFG, based on the size of

data input. So that each context doesn't exceed the permitted DRP input rate. To reduce the overall number of clocks, operating frequency should be decreased hence conceal the long data input/output durations. This can effectively speedup DRP typical processing.

3 Evaluation

We measured the context reduction ratio, total execution time and the number of reduced clock cycles. The enhancement was compared against typical DRP flow processing as mentioned in Section 1. Generic CPU (Pentium4 2.80GHz, Memory 1GB) was used for the evaluating execution time. The compiler used is GNU gcc3.2.2 with optimization option -O3 added.

The first contribution have reduced the number of the clock cycles by about 50% of typical DRP processing. Our second method have also reduced the number of contexts up to 50% of typical DRP processing. Typical DRP processing was already 150% faster than generic CPU. Our combined system was found 230% faster than generic CPU, in other words, 153% faster than typical DRP processing.

4 Conclusion

Our work foucsed on reducing input/output time and reducing the number of contexts required for processing. Our work included two contributions. First concel input/output time by overlapping input/output with computation. The second reduced the contexts by creating data-driven configurations. Experiments confirmed that our methods succeeded in achieving our goals. The number of total clock cycles and contexts of typical DRP processing was reduced by about 50% by our methods.