Scheduling divisible loads on partially reconfigurable hardware

K. N. Vikram and V. Vasudevan
Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai, India - 600036
vikramkn@ieee.org, vinita@iitm.ac.in

Abstract

For a task mapped to the reconfigurable fabric (RF) of a partially reconfigurable hybrid processor architecture, significant speedup can be obtained if multiple processing units (PUs) are used to accelerate the task. In this paper, we present the results obtained from a quantitative analysis for a single data-parallel task mapped to the RF of a bus-based hybrid processor architecture. The architectural constraints in this case include run-time reconfiguration delay and a shared data bus to main memory.

1. Introduction

Reconfigurable hybrid processor architectures, consisting of general purpose processor (GPP) coupled to a reconfigurable fabric (RF), allow flexible implementation of any application. In order to obtain the maximum possible speedup, spatial parallelism and partial run-time reconfiguration (PRTR) are used. Typical applications mapped to reconfigurable hybrid processors include signal/image processing, multimedia and computer vision. Many of the tasks in these applications are *data-parallel*, i.e., their input data can be partitioned and processed independently by multiple, identical processing units (PUs) configured in the RF. This fact is used by task schedulers to increase speedup of various applications.

Use of partial run-time reconfiguration allows configuration of a PU to overlap with computation on other PUs. This can be used to minimize the overhead due to reconfiguration. Since PUs used for data-parallel tasks operate independently, each PU can start functioning as soon as the RF area allocated to it is configured.

In a bus-based hybrid processor architecture, all PUs use a shared data bus for accessing main memory. Reconfiguration delay and limited data bandwidth are the two architectural constraints present in such a system. In order to achieve minimum processing time under these constraints, a systematic technique is required for scheduling and allocating load to the PUs. We have developed a framework for this analysis based on divisible load theory (DLT) [2] in our earlier work [3]. The specific case considered in that work was a situation where the results of the computation could be retained within the local memory of the RF itself. The analysis in [3] is not applicable if the RF needs to be reconfigured to perform another task, in which case the computed results will have to be sent back to main memory. We have addressed this problem here. The problem of scheduling with consideration of result collection has received rigorous treatment in [1], for processors in an arbitrary tree network. The bus network considered in this paper is a specific case of an arbitrary tree network. We therefore use the results in [1] as the foundation for performing the required analysis. PRTR introduces an additional dimension to the problem and gives some interesting results.

Table 1. Notation used in the analysis of our system, based on DLT.

Symbol	Description
n	Number of PUs used
α_i	Fraction of total load assigned to PU p_i.
w	Ratio of computation time of a PU for a given load, to the computation time of a standard PU for the same load.
z	Ratio of time taken to transmit a given load on the bus, to the time taken to transmit the same load on a standard bus.
T_{cp}	Time taken to process entire load by the standard PU.
T_{cm}	Time taken to transmit entire load on a standard bus.
T_p	Total processing time, including result collection.
T_r	Time taken to configure / reconfigure a single PU.
σ	$wT_{cp}/(zT_{cm})$
β	$(\sigma+2)/(\sigma+1)$

2. Analysis and Results

The notation that we use for our computation and communication model is given in Table 1, based on DLT. Using the notation given in Table 1, the time taken to transfer load fraction α_i to PU p_i is $\alpha_i zT_{cm}$, whereas the time taken by p_i for processing it is $\alpha_i wT_{cp}$. If the result data size is the same as input data size, the time taken to transfer the result

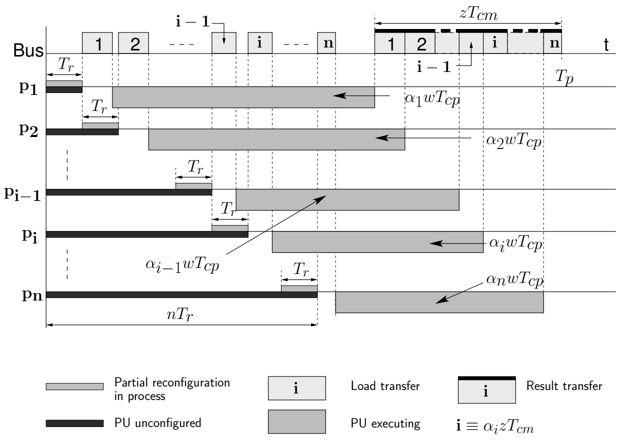

Figure 1. Timing diagram for $T_r > zT_{cm}/n$

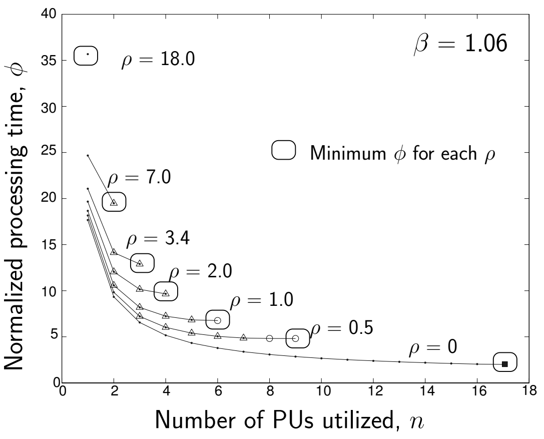

Figure 2. Results for 1-D DWT

from p_i back to memory is also $\alpha_i zT_{cm}$.

Reconfiguration of the PUs is assumed to occur continuously using a separate configuration bus, from p_1 to p_n, where n is the number of PUs used. Minimum processing time can be obtained if the following optimality criteria are satisfied: (1) Each PU should never be idle between its load transfer and result transfer phases, (2) The data bus and any PU should never simultaneously be idle and (3) Result transfer sequence is same as the load transfer sequence. We have rigorous proofs for each of them. A consequence of the optimality criteria is that during the result transfer phase there should be no gaps on the data bus.

Performing a quantitative analysis on our system while enforcing the above criteria gives us the following results. If n PUs are used for task acceleration and the reconfiguration time $T_r \leq zT_{cm}/n$, then the load fractions allocated to PUs are equal, and no gaps occur during load transfer. The total processing time is then $T_p = T_r + zT_{cm} + (zT_{cm} + wT_{cp})/n$. When $T_r > zT_{cm}/n$, there are gaps in load transfer, as shown in Fig. 2. Then the load fractions and optimum processing time are given by

$$\begin{aligned} \alpha_i &= \beta^{i-1}\alpha_1 - \frac{T_r}{zT_{cm}}(\beta^{i-1} - 1), \quad i = 1, \ldots, n \\ T_p &= T_r\left(1 + \frac{1}{\beta - 1} - \frac{n}{\beta^n - 1}\right) + zT_{cm}\left(1 + \frac{1}{\beta^n - 1}\right) \end{aligned}$$

where

$$\alpha_1 = \frac{T_r}{zT_{cm}} - \left(\frac{\beta - 1}{\beta^n - 1}\right)\left(\frac{nT_r}{zT_{cm}} - 1\right) \qquad (1)$$

However, if wT_{cp} is small, some PUs might finish computation even before the end of the load transfer phase. In such cases, an optimal schedule does not exist and we have developed heuristic strategies with the basic idea being that result transfer can occur in the gaps during load transfer. We have shown that for a limited range of T_r, there exists a heuristic load allocation strategy that results in an optimal processing time. For other reconfiguration times also, we have derived closed form expressions for the load fractions and processing time.

We have verified the developed theory for computation of 1-D DWT. Fig. 2 shows variation of $\phi = T_p/(zT_{cm})$ with the number of PUs, for different values of $\rho = T_r/(zT_{cm})$. The parameter β is based on the computation speed of a PU. The figure shows that an optimum number of PUs exists for a given ρ, beyond which more PUs do not contribute to speedup. It also gives the minimum possible processing time.

3. Conclusions

We have presented a theoretical framework for scheduling load for a data-parallel task mapped to the RF of a hybrid processor. The theory gives the maximum speedup that can be obtained, and is also a good approximation when the application load is not arbitrarily divisible. The theory is also useful for deriving the design considerations for optimal usage of the shared data bus.

References

[1] G. D. Barlas. Collection-aware optimum sequencing of operations and closed-form solutions for the distribution of a divisible load on arbitrary processor trees. *IEEE Trans. Parallel Distrib. Syst.*, 9(5):429–441, May 1998.

[2] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi. *Scheduling divisible loads in parallel and distributed systems*. IEEE CS Press, Sept. 1996.

[3] K. N. Vikram and V. Vasudevan. Mapping data-parallel tasks onto partially reconfigurable hybrid processor architectures. *Accepted for publication in IEEE Trans. VLSI Syst.*