Throughput of Streaming Applications Running on a Multiprocessor Architecture

Nikolay Kavaldjiev, Gerard J. M. Smit, Pierre G. Jansen
Department of EEMCS,
University of Twente, the Netherlands
{nikolay, smit, jansen}@cs.utwente.nl

Abstract

In this paper we study the timing behaviour of streaming applications running on a multiprocessor architecture. Dependencies are derived between the application throughput and the timing characteristics of the processors and communication. Four different processor organizations that strongly influenced the results are considered and compared.

1. Introduction

The advances of silicon technology make it possible to build multi-processor system-on-chip (MPSoC) devices. A potential application domain for these systems is the domain of mobile multimedia devices where high performance and energy-efficiency is required. The applications in that domain typically deal with processing of data streams, for example: base-band processing for wireless communication channels, audio/video (de)coding, image processing etc. In these applications a data source delivers portions of data to the application at regular intervals (input stream). The application handles and processes the data producing portions of data regularly (output stream). The portions of data are called data items (or stream items).

To run an application on a multi-processor architecture it has first to be represented in a parallel form using some parallel computation model, like Kahn process networks [1]. When a stream-processing application is represented as a Kahn process network the result is usually similar to the process graph shown in *Figure 1*. In this graph two parts can be distinguished: a processing part and a control part. The processing part, shown bold in the figure, is a pipe of processes through which the data stream passes in the order of processing. This is the part of the application where the actual processing of the data is done and so it generates most of the computation load for the application. The processes there (denoted as P1, P2, .., PN in the figure) implement some DSP kernel ,e.g. FFT, DCT, FIR.

The control part of the application comprises all the tasks dealing with application organisation and control, run-time adaptation and reconfiguration. Because of the reactive nature of these tasks we expect them to run not so often and to require lighter computation.

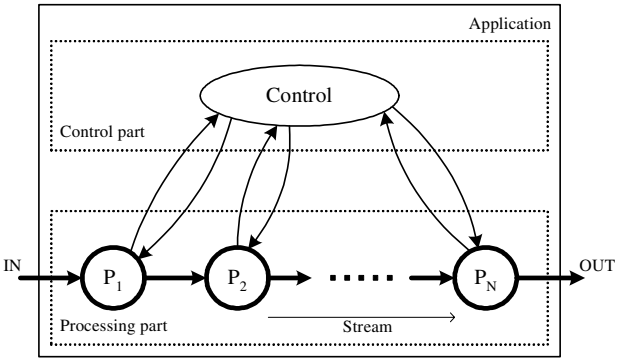

Figure 1 General process graph of a stream-processing application

In *Figure 1* all the nodes in the processing part are connected to the control part by communication edges. These communications are optional as their presence depends on the application - some of them might be missing or there might be applications without a control part.

The observations about the parallel representation of streaming applications stated here are based on our experience with base-band processing applications for wireless communications, like HiperLAN/2, UMTS, Bluetooth [2] [3]. A similar observation for media processing applications is made in [4]. A similar structure can be observed in ASIC implementations of stream-processing architectures which are typically organized as a pipe of dedicated hardware blocks.

This paper focuses on the processing part of streaming applications and studies the timing aspects of its operation. The processing part usually operates under real-time constrains and knowledge about its timing behaviour is needed in order to provide real-time guarantees.

The timing aspects are studied assuming that each process runs on separate processor. This scenario is realistic for architectures where most of the processors are not complex general purpose processors, but simplified

Proceedings of the 2005 8th Euromicro conference on Digital System Design (DSD'05)
0-7695-2433-8/05 $20.00 © 2005 IEEE

domain-specific processors that do not support multi tasking.

The paper is organized as follows. Section 2 discusses the time constraints on the processing part operation. Section 3 introduces our notation. In Section IV results are derived for the pipeline throughput.

2. Time constraints on the pipeline operation

Considering the stream-processing part of an application, three main cases of time constrained operation can be distinguished: time-constrained input, time-constrained output and constrained processing rate.

Time-constrained input - the input data arrive at a fixed period of time and the pipeline must always be ready to accept it; when it is not ready, the data is lost. The pipeline output data are consumed immediately. This is the case for operation of the receiving part of base-band processing applications where a data item arrives periodically from the antenna through the analogue front-end.

Time-constrained output - pipeline output data are read at a fixed period of time and at that time the pipeline output must always be ready to send; if it is not ready, false data are read and the processed data item is lost. The pipeline input is fed with new data immediately. This is the type of operation for the transmitting part of base-band processing applications where a data item is sent to the antenna periodically at exact time instances.

Constrained processing rate - the pipeline is expected to process data items at rate higher than a certain minimum. The input is fed immediately and its output is immediately consumed. This states that the pipeline throughput must be above certain minimum. The first two cases can also be reduced to this one when the pipeline input or output, respectively, is decoupled from the data stream by a data-buffer.

A process in the pipeline fires repetitively and regularly - every time a new data item from the stream arrives. When a process fires it takes time to process the data item – *processing time (P)*. Since the processes run on separate processing units, communication is introduced between them. Moving a data item from one processor to the next one also takes time – *communication time (C)*. For the processor that sends the data this is *sending time (S)* and for the processor that receives it this is *receiving time (R)*. We assume that a message passing mechanism is used for communication and the sending and receiving times of consecutive processes are equal. Communication time includes the time needed for all procedures involved in the communication – requesting, arbitration, data transportation, etc. The processing times and the communication times determine the timing behavior of the whole pipeline.

3. Notation

In this paper the following notation is used:

N – number of stages in the pipeline

$C_{i,n}$ – time for transmitting the *n*-th data item out of stage *i*,

$P_{i,n}$ – time for processing the *n*-th data item in the *i*-th pipeline stage

$W_{i,n}$ – period of time stage *i* has to wait after it has processed the *n*-th data item

n=1,2,…∞; i = 1, 2,..,N

$C_{0,n}$ – time for receiving the n-th pipeline input data item

$C_{N,n}$ – time for transmitting the n-th data item out of the pipeline

↑T – start of a time period T

↓T – end of a time period T

@(e) – time of occurrence of an event e

Thus:

$@(\uparrow C_{i,n})$ – time at which the transmission of the n-th data item out of stage i starts

$@(\downarrow C_{i,n})$ – time at which the transmission of the n-th data item out of stage i ends

Since the output communication channel of stage *i-1* is the input channel for stage *i*, the *n*-th data item enters the *i*-th stage during $C_{i-1,n}$, it is being processed during $P_{i,n}$ and is sent to the next stage during $C_{i,n}$.

Assumption (A): the processing and communication times of a pipeline stage are constant for all data items.

$$C_{i,j} = C_i, P_{i,j} = P_i, W_{i,j} = W_i, \forall j \in \{0,1,2,..,\infty\}$$

$$0 < C_i, 0 < P_i, 0 \le W_i, \forall i \in \{1,2,..,N\}$$

That is the case for the base-band processing applications we have studied. For applications with varying times the upper bounds for these times should be used.

4. Throughput of a stream processing pipeline

This section derives dependencies between communication times, processing times and pipeline throughput. These dependencies are strongly influenced by the parallelisms between processing and communication allowed by the processors organization. Considering this aspect, we distinguish four models of processor operation. Below, the symbols *R*, *S* and *P* refers to receive, send and process operation, symbol "||" is used to denote parallelism and the symbol "=" is used to denote sequential operation. The four models are as follows:

- *Sequential communications and sequential processing, (R=S)=P* - the three operations are performed strictly sequentially.
- *Parallel communication and sequential processing, (R||S)=P* - the send and receive operations can be performed at the same time, but the processing can be performed only if the processor does not communicate.
- *Sequential communications and parallel processing, (R=S)||P* – the tile can process while

communicating, but still sending and receiving cannot be performed in parallel.

- *Parallel communications and parallel processing (R||S)||P* – all the operations (send, receive and process) can be performed simultaneously.

This paper considers only pipelines consisting of processors with the same organization – homogeneous pipelines.

4.1. Sequential communication and sequential processing, (R=S)=P

This model allows the processor to perform the operations only sequentially following the strict order: receive, process, send. It corresponds to a processor organization where the same memory is used to store the received, processed and sent data. Since the memory is shared between the three operations, these steps are mutually exclusive.

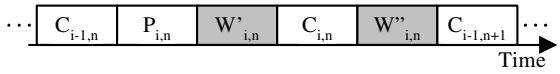

Figure 2 Time diagram of a pipeline stage operation for (R=S)=P

Figure 2 presents a time diagram of a stage operation. When data has been received the processing can start. After the processing is done the data can be sent to the next stage. If in that moment the next stage is not ready to accept the data, a waiting time $W'_{i,n}$ is introduced. When the data is sent the stage finishes its cycle and is ready to receive again. If the previous stage is not ready with the data, a waiting time $W''_{i,n}$ is introduced after $C_{i,n}$.

For stage *i* the period between two successive input data items is:

$$@(\uparrow C_{i,n+1}) - @(\uparrow C_{i,n}) =$$

$$= C_{i,n} + P_{i+1,n} + W'_{i+1,n} + C_{i+1,n} + W''_{i+1,n}$$

For stage i the period between two successive output data is:

$$@(\uparrow C_{i,n+1}) - @(\uparrow C_{i,n}) =$$

$$= C_{i,n} + W''_{i,n} + C_{i-1,n+1} + P_{i,n+1} + W'_{i,n+1}$$

From the equivalence of both equations:

$$C_{i,n} + W''_{i,n} + C_{i-1,n+1} + P_{i,n+1} + W'_{i,n+1} =$$

$$= C_{i,n} + P_{i+1,n} + W'_{i+1,n} + C_{i+1,n} + W''_{i+1,n}$$

Using the assumption (A) from *Section 3* and substituting $W'_i + W''_i = W_i$ we derive the following recurrent equation:

$$C_{i-1} + P_i + W_i + C_i = C_i + P_{i+1} + W_{i+1} + C_{i+1},$$

$$\forall i \in \{1,2..N-1\}$$

Thus the performance of the pipeline is determined by the performance of the slowest pipeline stage – if its period $(C_{i-1}+P_i+W_i+C_i)$ of operation is *T*, then:

$$C_{i-1} + P_i + W_i + C_i = T, \forall i \in \{1,2,..,N\}$$

The pipeline throughput is *1/T*. Since *0≤Wi*:

$$C_{i-1} + P_i + C_i \le T, \forall i \in \{1,2,..,N\}$$

This is the general constraint for communication and processing times in the *(R=S)=P* pipeline. In order to guarantee pipeline throughput higher or equal 1/T, the last equation must hold.

4.2. Parallel communication and sequential processing, (R||S)=P

This model allows sending and receiving data in parallel but the processing still has to be done when the processor does not communicate. The model corresponds to a processor organization where two separate memories are used for sent and received data, but both of them are used during the processing.

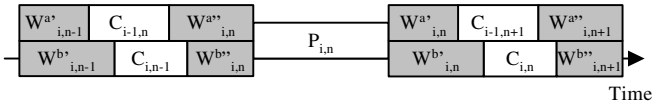

Figure 3 Time diagram of a pipeline stage operation for (R||S)=P

A time diagram of a pipeline stage operation is presented in *Figure 3*. When the input data has been received and the previously processed data has been sent the processing can start. If one of these two conditions does not hold, either waiting time $W^{a}{}''_{i,n}$ or $W^{b}{}''_{i,n}$ is introduced (but not both together). After the processing has finished, communication can start. If the previous stage is not ready to transmit or the next stage is not ready to receive, waiting times $W^{a}{}'_{i,n}$ and $W^{b}{}'_{i,n}$ are introduced here.

Using the same reasoning as in *Section 4.1* the constraint for the communication and processing times in a *(R||S)=P* pipeline can be derived:

$$C_{i-1} + P_i \le T,$$

$$C_i + P_i \le T, \forall i \in \{1,2,..,N\}$$

If these inequalities hold, then pipeline throughput higher or equal *1/T* is guaranteed.

4.3. Sequential communication and parallel processing, (R=S)||P

This model allows communication and processing to be performed simultaneously, but still sending and receiving must be done sequentially. It is valid for a processor organization where two separate memories are used for communication and processing which are swapped in the beginning of each operation cycle. Since the same memory is used for sent and received data these operation are mutually exclusive. *Figure 4* presents a time diagram of a stage operation. At the beginning of the cycle, immediately after memory swap, the processor's memory contains the last received data and the communication memory contains the last processed data. The processing starts immediately.

The sending can start too, but if the next stage is not ready to receive, waiting time $W'_{i,n-1}$ is introduced before $C_{i,n-1}$. After the data has been sent the communication memory is free and receiving of the next data can start. If the previous stage is not ready to send, waiting time $W''_{i,n+1}$ is introduced before $C_{i-1,n+1}$. Note that new data cannot be received until the old data is sent. The processing result is stored in the processor's memory. After the finishing the processor waits until the new data is received. At that moment the memories are swapped again and a new cycle starts.

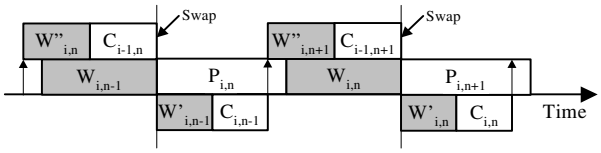

Figure 4 Time diagram of a pipeline stage operation for (R=S)||P

Using the same reasoning as in *Section 4.1* the constraint for the communication and processing times in a *(R=S)||P* pipeline can be derived:

$$P_i \le T,$$

$$C_{i-1} + C_i \le T, \forall i \in \{1,2,..,N\}$$

If these inequalities hold, then pipeline throughput higher or equal *1/T* is guaranteed.

4.4. Parallel communication and parallel processing, (R||S)||P

This model allows the three operations - receive, process and send - to be performed simultaneously. It is valid for a processor organisation where two memory banks each containing two separate memory blocks are used for processing and communication. A swap between the banks is performed at the beginning of each stage cycle. The presence of two separate memory blocks in the communication bank allows simultaneous sending and receiving of data.

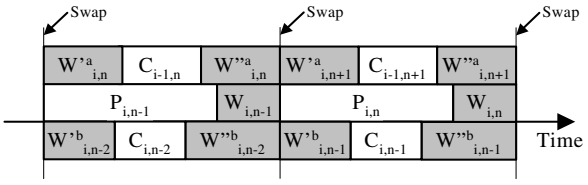

Figure 5 Time diagram of a pipeline stage operation for (R||S)||P

A time diagram of a stage operation is shown in *Figure 5*. At the beginning of the cycle the memory banks are swapped. After the swapping one of the memory blocks in the processor's bank contains the new data to be processed and the other block is empty. In the communication bank one block contains the data to be sent and the other block is empty. After swap the processing can start. Sending and receiving can start too. If either the next stage is not ready to receive or the previous stage is not ready to send, waiting times $W''^{a}_{i,n+1}$ and $W''^{b}_{i,n-1}$ are introduced before $C_{i-1,n+1}$ and $C_{i,n-1}$ respectively. The cycle finishes when all three operations finish. If an operation finishes before the end of the cycle, waiting time is introduced after it: $W_{i,n}$, $W''^{a}_{i,n+1}$ or $W''^{b}_{i,n-1}$ respectively.

Using the same reasoning as in *Section 4.1* constraints for the communication and processing times in the *(R||S)||P* pipeline can be derived:

$$C_{i-1} \le T,$$

$$P_i \le T,$$

$$C_i \le T, \forall i \in \{1,2,..,N\}$$

If these inequalities hold, then pipeline throughput higher or equal *1/T* is guaranteed.

4.5. Summary

Table I summarizes the memory requirements and the constraints for the four models of operation. The memory requirements are presented normalized to the (R=S)=P model.

TABLE I

Model	SCSP	PCSP	SCPP	PCPP
Mem*	1	2	2	4
Constraints $\forall i \in \{1..N\}$	$C_{i-1}+P_i+C_i \le T$	$C_{i-1}+P_i \le T$ $P_i+C_i \le T$	$P_i \le T$ $C_{i-1}+C_i \le T$	$P_i \le T$ $C_i \le T$

* normalized to SCSP

5. Conclusion

This work studied the timing aspects of streaming applications running on a multiprocessor architecture. Relations were derived between the application throughput and the time operation of each processor, which allows giving real-time guarantees. Although the used method was good enough to derive the results it is quite inflexible, works only on very simple application graphs and for processors with the same organization. This makes it inapplicable for applications with more complex process graphs and architectures with heterogeneous processors.

6. References

[1] G. Kahn, "The semantics of a simple language for parallel programming." *In Information Processing*, pp. 471-475, Stockholm, August 1974.

[2] Gerard K. Rauwerda, Paul M. Heysters, Gerard J.M. Smit, "Mapping Wireless Communication Algorithms onto a Reconfigurable Architecture", Journal of Supercomputing, Kluwer Academic Publishers, December 2004.

[3] Pascal T. Wolkotte, Gerard J.M. Smit, L.T. Smit, "Partitioning of a DRM receiver", Proceedings of the 9th International OFDM-Workshop, pp. 299-304, Dresden, September 2004.

[4] William Dally et al. "Stream Processors: Programmability with Efficiency" ACM Queue, pp. 52-62, March 2004.