An Abstract Model for Performance Estimation of the Embedded Multiprocessor CoreVA-MPSoC Technical Report (v1.0)

Johannes Ax*, Martin Flasskamp*, Gregor Sievers*, Christian Klarhorst*, Thorsten Jungeblut*, and Wayne Kelly†
*Cognitronics and Sensor Systems Group, CITEC, Bielefeld University, Bielefeld, Germany
†Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia
Email: jax@cit-ec.uni-bielefeld.de w.kelly@qut.edu.au

I. Introduction

This technical report presents an abstract model for the performance estimation of the multiprocessor CoreVA-MPSoC. The CoreVA-MPSoC targets streaming applications in embedded and energy-limited systems. The abstract model is used by our CoreVA-MPSoC compiler [1] to estimate the performance of a certain streaming application.

Our CoreVA-MPSoC compiler reads applications that are described in the programming language StreamIt [2]. A StreamIt program is represented by a structured data flow graph of its tasks (filter). The CoreVA-MPSoC compiler partitions all filter of a program onto particular cores of the MPSoC. An abstract model for such a partitioning is presented in Section II. Section III shows the abstract model of the hardware architecture of the CoreVA-MPSoC.

The configurable VLIW CPU CoreVA [3] is used as the basic building block for our MPSoC. The CPU features L1 scratchpad memories for instruction and data. Several CPU cores are tightly coupled within a cluster [4]. Several of those clusters are connected via a network on chip (NoC) [5] (cf. Fig. 1).

Within a cluster each CPU can access the L1 data memories of other CPUs via a bus based interconnect (shared, partial or full crossbar). The NoC interconnect is composed of three components: The (i) routers transport the data through the NoC in a packet-based manner. Routers are connected via (ii) network links. A (iii) network interface (NI) implements the interface between the routers and the CPUs.

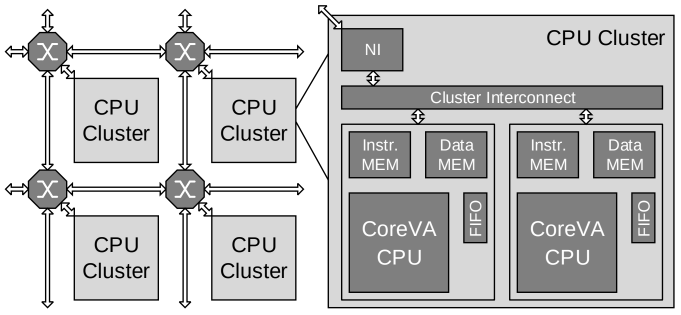

Fig. 1. Hierarchical CoreVA-MPSoC architecture

Section IV shows the abstract model for the total throughput of a certain partition of a StreamIt application. A goal for the CoreVA-MPSoC compiler is to maximize this throughput to achieve the best performance for an application.

II. Model of the StreamIt Program

A StreamIt program can be represented as a structured graph $\mathbb{G} = (\mathbb{F}, \mathbb{E})$, where $\mathbb{F}$ is a set of filters and $\mathbb{E}$ is a set of edges. Each $e \in \mathbb{E}$ is of form (a, b) which represents a communication channel between filter $a \in \mathbb{F}$ and $b \in \mathbb{F}$ in which a message with size $|e|$ (in $\frac{bytes}{work\ function\ of\ a}$) is send from filter a to filter b. Each filter $f \in \mathbb{F}$ has a work function with the estimated execution time $W(f)$ in cycles. The execution time $W(f)$ includes repeated executions of a work function, which may be required to consume or produce enough data for the filter at it's edges.

There exists a unique filter $\mathscr{L}$ without outgoing edges, which is the last filter of the application: $\nexists b \in \mathbb{F}\ s.t.\ (\mathscr{L}, b) \in \mathbb{E}$

There exists a unique filter $\mathscr{F}$ without ingoing edges, which is the first filter of the application: $\nexists b \in \mathbb{F}\ s.t.\ (b, \mathscr{F}) \in \mathbb{E}$

$\mathscr{M}$: Multiplicity how often the work functions of all filters are called during one steady state iteration $(\frac{work\ function\ calls}{steady\ state\ iteration})$.

III. Model of the CoreVA-MPSoC

The CoreVA-MPSoC consist of a set of processors $\mathbb{P}$.

The StreamIt compiler maps each filter to a processor: $M : \mathbb{F} \mapsto \mathbb{P}$

The MPSoC has a set of clusters $\mathbb{C}$ and each processor belongs to a cluster: $C : \mathbb{P} \mapsto \mathbb{C}$

Additionally the MPSoC consist of a set of network links $\mathbb{N}$. Each $n \in \mathbb{N}$ has a maximum bandwidth $B(n)\ \frac{bytes}{cycle}$ that it can handle. A network link could be a bus-link within a cluster, a network interface (NI) or a NoC-link.

$N(p_a, p_b)$ is a list of all network links involved when sending a message from processor $p_a \in \mathbb{P}$ to processor $p_b \in \mathbb{P}$ (depending on the routing algorithm): $N : (p_a, p_b) \rightarrow [\mathbb{N}]$

IV. Model of the Throughput

This section shows an abstract model for throughput estimation of a certain StreamIt program given by II and mapped to a configuration of CoreVA-MPSoC given by III.

A. Throughput of a Processor

A filter $f \in \mathbb{F}$ has input edges: $I(f) = \{(a,f)|(a,f) \in \mathbb{E}\}$

A filter $f \in \mathbb{F}$ has output edges: $O(f) = \{(f,b)|(f,b) \in \mathbb{E}\}$

For each filter $f \in F$ we generate code of the form:

```
foreach (Channel i in I(f))
          i.WaitInputReady

foreach (Channel o in O(f))
          o.WaitOutputReady

Work_f()

foreach (Channel i in I(f))
          i.DoneWithInput

foreach (Channel o in O(f))
          o.DoneWithOutput
```

Before executing the work function $Work_f$ of filter $f \in \mathbb{F}$ it is necessary to wait until all communication channels (input $I(f)$ and output $O(f)$ edges) are ready to use. After $Work_f$ all communication channels ($I(f)$ and $O(f)$) can be set to done. The execution time of these wait and done functions is given by the channel type of edge $(a,b) \in \mathbb{E}$, which depends on the location of the filter a and b (same processor, different processor but same cluster, or different cluster): $M(a) = M(b) \rightarrow$ *memory channel* $M(a) \neq M(b) \;\wedge C(M(a)) = C(M(b)) \rightarrow$ *cluster channel* $C(M(a)) \neq C(M(b)) \rightarrow$ *NoC channel* The execution time of the wait for input channels of edge $e \in E$ is represented by $I_w(e)$ and $O_w(e)$ for the output channels. The execution time of the done for input channels is represented by $I_d(e)$ and $O_d(e)$ for the output channels.

The execution time $E(f)$ (in cycles per steady state iteration) of filter $f \in \mathbb{F}$ is the sum of the execution time of the filters work function $W(f)$ multiplied by the Multiplicity $\mathscr{M}$ and a sum of all software overheads for the different communication channels of all it's input and output edges.

$$E(f) = \mathscr{M}\; W(f) + \sum_{e \in I(f)} (I_w(e) + I_d(e)) \\ \sum_{e \in O(f)} (O_w(e) + O_d(e)) \;\frac{cycles}{steady\;\;state\;\;iteration} \tag{1}$$

The maximum throughput $T(p)$ (in steady state iteration per cycle) of processor $p \in \mathbb{P}$ is the inverse of the sum of the execution time of all filters $f \in \mathbb{F}$ mapped to processor p.

$$T(p) = \frac{1}{\sum_{f \in M'(p)} E(f)} \;\frac{steady\;\;state\;\;iteration}{cycles} \tag{2}$$

Where $M'(p)$ are all filters mapped to processor $p \in \mathbb{P}$: $M'(p) = \{f \in \mathbb{F}|M(f) = p\}$

B. Throughput of a Network Link

An amount of data $D(n)$ (in bytes per steady state iteration) is crossing each network link $n \in \mathbb{N}$. This amount of data is based on the Multiplicity $\mathscr{M}$ and the message sizes of all edges going through this network link n

$$D(n) = \mathscr{M} \sum_{e \in \{(a,b) \in \mathbb{E} | n \in N(M(a),M(b))\}} |e| \;\frac{bytes}{steady\;\;state\;\;iteration} \tag{3}$$

The maximum throughput $T(n)$ of network link $n \in \mathbb{E}$ is the maximum bandwidth (in bytes per cycle) a network link n can handle divided by the time the network link needs to transmit all the data (in bytes) of one steady state iteration.

$$T(n) = \frac{B(n)}{D(n)} \;\frac{steady\;\;state\;\;iterations}{cycle} \tag{4}$$

C. Total throughput of the system

The throughput of all processors is given by set T_{comp}.

$$T_{comp} = \{T(p)|p \in \mathbb{P}\} \frac{steady\;\;state\;\;iterations}{cycle} \tag{5}$$

The throughput of all network links is given by set $T_{network}$.

$$T_{network} = \{T(n)|n \in \mathbb{N}\} \;\frac{steady\;\;state\;\;iteration}{cycles} \tag{6}$$

The amount of data $D(f)$ (in bytes per steady state iteration) consumed by a filter $f \in \mathbb{F}$ depends on Multiplicity $\mathscr{M}$ and the message sizes of all its input edges.

$$D(f) = \mathscr{M} \sum_{e \in I(f)} |e| \;\frac{bytes}{steady\;\;state\;\;iteration} \tag{7}$$

The total throughput of the StreamIt application T_{system} is given by the bottleneck of the system. The bottleneck of the system is the component (processor or network link) with the lowest throughput

$$T_{system} = min(\mathbb{T}_{comp} \cup \mathbb{T}_{network}) \;\frac{steady\;\;state\;\;iteration}{cycles} \tag{8}$$

Or if we also consider the amount of the produced data within one steady state:

$$T_{system} = D(\mathscr{L})\; min(\mathbb{T}_{comp} \cup \mathbb{T}_{network}) \;\frac{bytes}{cycle} \tag{9}$$

V. Conclusion

In this report, an abstract model for the performance estimation of the CoreVA-MPSoC has been presented. The abstract model is able to estimate the maximum throughput of a certain streaming application mapped to a particular configuration of the CoreVA-MPSoC.

References

[1] W. Kelly, M. Flasskamp, G. Sievers, J. Ax, J. Chen, C. Klarhorst, C. Ragg, T. Jungeblut, and A. Sorensen, "A Communication Model and Partitioning Algorithm for Streaming Applications for an Embedded MPSoC," in *Int. Symp. on System on Chip (SoC)*. IEEE, 2014.

[2] W. Thies, M. Karczmarek, and S. Amarasinghe, "StreamIt: A Language for Streaming Applications," in *Int. Conf. on Compiler Construction*. Springer, 2002, pp. 179–196.

[3] G. Sievers, P. Christ, J. Einhaus, T. Jungeblut, M. Porrmann, and U. Rückert, "Design-space Exploration of the Configurable 32 bit VLIW Processor CoreVA for Signal Processing Applications," in *NORCHIP*. IEEE, 2013.

[4] G. Sievers, J. Ax, N. Kucza, M. Flasskamp, T. Jungeblut, W. Kelly, M. Porrmann, and U. Rückert, "Evaluation of Interconnect Fabrics for an Embedded MPSoC in 28nm FD-SOI," in *ISCAS*. IEEE, 2015.

[5] J. Ax, G. Sievers, M. Flasskamp, W. Kelly, T. Jungeblut, and M. Porrmann, "System-level analysis of network interfaces for hierarchical mpsocs," in *NoCArc*. ACM, 2015. In press.