Hierarchical Agent Monitoring Services on Reconfigurable NoC Platform : A Formal Approach

Liang Guang, Juha Plosila, Jouni Isoaho, Hannu Tenhunen
Department of Information Technology, University of Turku, Finland
{liagua, juha.plosila, jisoaho, hatenhu}@utu.fi

I. Introduction

The constant integration of on-chip resources calls for dynamic run-time monitoring services. Many-core architecures have become the mainstream platform with the releases of 80-core TeraFLOPS processor [5] and 64-core TILE64 processor [1] in the industry. The large number of on-chip processing units with accessory components including memories, caches, I/O interface etc.. have formed a parallel and distributed system. Conventional control methods, for instance with a centralized monitor, are no longer efficient nor scalable. The underlying transistor scaling only excerbates the design complexity by introducing a more profound influence from transistor and circuit level variations [2]. These runtime design issues are beyond the analysis and management capability of any static methods, thus systematic and scalable monitoring approach is needed on such parallel systems.

Previous work [6] presents a scalable design approach, *hierarchical agent monitored system*, which provides hierarchical monitors on distributed components. The analysis and application of the approach focus on many-core on-chip systems with NoC (network-on-chip) communication infrastructure. Monitoring services are organized by and partitioned into several levels of agents, and the interactions among these agents provide dynamic tracing and reconfiguring operations from the circuit to the system level.

This paper addresses the initial but novel research work proposing a formal model of such hierarchical agent monitored system. The model abstracts on-chip resources and agents at each architectural level and specifies how the components can be monitored by each level of agents. It provides a proper system abstraction that enables designers to specify various reconfiguration alternatives while hiding the implementation details. Compared to exisiting system modeling languages, this formal model is tailored for high-level specification and straightforward for theoretical verification. And the seperation between monitoring components and functional components as well as the hierarchization of system structure are essential for monitoring-centric parallel system design. Compared to exisiting formal methods, the formal model is targeted at defining the monitoring services on reconfigurable platform, which has not yet been captured by formal languages to the best of our knowledge.

The rest of the paper is outlined as follows: Section II overviews the hierarchical agent monitoring design approach. Section III specifies the initial formal model. Section IV examplies the application of the formal model on a study case of hierarchical power monitoring. Section V explains the ongoing work and concludes the paper.

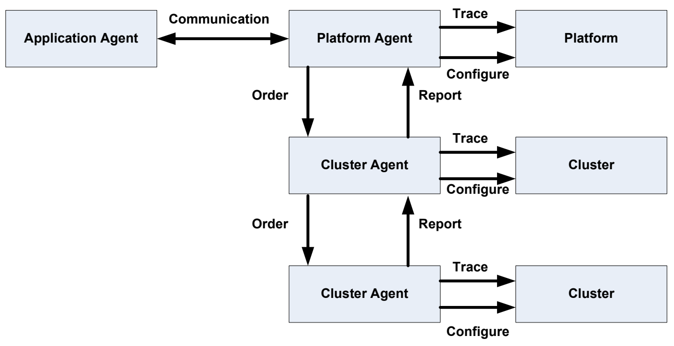

Figure 1. Generic Hierarchical Agent Monitoring Architecture

II. Hierarchical Agent Monitoring Services

In hierarchical agent monitoring architecture, all monitoring operations are handled by agents, which are capable of monitoring the system at certain architectural level. Fig. 1 presents our abstraction of agent hierarchy. The platform is equipped with three levels of monitors: cell agents, cluster agents and platform agents. The monitoring of the application is simplified to be handled by one application agent as it is beyond our current work to address program parallelization and hierarchization.

The platform agent (unique for each platform), the highest level monitor of all on-chip resources, provides coarse-grained resource management. Typical services include resource acquirement and clusterization, network configuration, as well as high-level fault management. These operations have considerable and universal influence on all components, and are usually required infrequently. Since these chip-wide operations tend to require large processing capacity and diversity to provide various optimizations with the information of all resources, the platform agent is to be implemented as a software component. Each cluster agent monitors a cluster of components (a dynamically configured group of atomic units) with finer-grainular operations which mostly influence the performance within the cluster. Dependent upon the complexity of monitoring algorithms required and the size of the cluster, the cluster agent is either a software component or

Symbol	Explanation
Res	generally referring to any resource in the system
CEAG	cell agent
CLAG	cluster agent
PAG	platform agent
AAG	application agent
$\oplus$	union of components of different types
$:\Leftrightarrow$	represented as
$\rightarrow$	connector of phrases in a monitoring operation
$Res : Property$	the property of a resource
$Res : Property \vdash NewValue$	a new value is given to the property of the resource
$\{ \}$	set, a list of elements with no order
$( )$	tuple, an ordered list of elements
/	or

Table I
SELECTED LIST OF NOTATIONS USED IN THE FORMAL SPECIFICATION

a SW/HW co-design. Each cell agent monitors a cell, which is the basic reconfiguration unit with a few atomic functional units. The monitoring operations at this level include timing-critical actions such as error detection and power gating, which are commonly implemented as dedicated hardware circuits. Software may still be used for some monitoring operations even at this low level of hierarchy if required and feasible, for instance a software monitor embedded in a processor core. The operations labeled in Fig. 1 are formally defined and explained in Section III.

It is important to note that the concept of clusters is scalable. In case of a large number of on-chip components, clusters can be grouped into sub-clusters. The hierarchy presented in Fig. 1 should be correspondingly expanded. Moreover, the hierarchy can be applied to multi-application-multiple-platform systems, in which case additional interactions are needed to handle the arbitration between different applications and platforms. The description in this paper is based on single-application-single-platform system with one level of clusters.

III. FORMAL MODELS

A. Resources and Their Properties

1) Resources: Any resource (functional unit, cell, cluster or the whole system) is represented by a tuple including its resource ID and the set of its properties. The notations used throughput the paper are summarized in Table I.

$Res :\Leftrightarrow (ID, Properties)$

ID: unique identifier of the resource

$Properties$: the set of observable and controllable properties of the resource

2) Properties: A property of a resource is a dynamic parameter of the resource, indicating its working state. A property can be a direct parameter, for instance, the working frequency of a processor, or the temperature of certain region (measured by a thermal sensor, for instance). It can also be an abstract parameter, for instance, the structure of a resource group. Generically, the following properties are commonly required:

- *Affiliation*: to which resource group the resource directly belongs to. For instance, the affiliation of a cell is the identifier of the cluster it belongs to. Any resource not assigned to any resource group is uniqued labelled and the top resource group (the platform) is uniquely annotated as well.
- $Status$: the current status of the resource. It can be an enumeration type. For instance, {proper, temporally unavailable, broken}. More elaborated status types are to be defined based on the platform and monitoring services.
- $Structure$: the dynamic formation of a resource group, for instance:

$Cell_i : Structure = \{FU | FU \triangleright Cell_i\} \oplus CEAG_i$

The structure of a cell is the dynamic union of the functional units assigned to the cell and the cell agent.

3) Categorization of Properties: As an important classification required for monitoring services, properties of a resource are categorized into *controllable* and *observable* properties (non-exclusive).

- OP: the set of observable properties. An observable property is a property that can be observed by agents (directly measured or indirectly induced). For instance, if the temperature of a processor can be sensed by the temperature sensor and reported to the agent, it is an observable property.
- CP: the set of controllable properties. A controllable property is a property than be configured by agents. For instance, in a DVFS-enabled system, the V_{DD} can be configured dynamically by the agent.

For instance, *Affiliation* is a controllable property and $Status$ is an observable property.

The properties can also be categorized into electrical properties or logic properties to facilitate the system engineering.

- EP: Electrical properties of any resource. For instance, $V_{DD}$and V_T(threshold voltage), $Freq$(working frequency)
- LP: logic properites of any resource (in the form of digital circuitry). For instance, a certain output/input (likely of special purposes, for instance, for testing) of a digital device.

4) Example: To give a simple example, if a cluster is composed of processor1-processor6 and a cluster agent at location XY; the working frequency of the cluster can be dynamically adjusted from the set (100MHz, 500MHz, 1GHz); the temperature of the cluster can be measured by a thermal sensor. The cluster can be formally specified as:

$Cluster : ID = XY$
$Cluster : Structure = \{Processor1 - 6\}$
$\oplus CLAG | CLAG : ID = XY\}$
$Cluster : OP = Cluster : Temperature$
$Cluster : CP = Cluster : Frequency |$
$FrequencyValue \in \{100MHz, 500MHz, 1GHz\}$

B. Monitoring Operations

As labelled in Fig. 1, five types of monitoring operations are defined to realize the monitoring interactions between the agents. The five types of operations are not designed as being atomic. Dependent upon actual implementation, each operation may be fulfilled by a sequence of suboperations.

1) Communicate:

- Format

$Agent \rightarrow Communicate \rightarrow (message)$
$\rightarrow DestinationAgent$

- Explanation

One agent sends message to the destination agent. There is no supervising relation between the two, which is different from $Order$ operation. The receiver agent decides if it will act upon the message received.

- Note

At present, this operation only takes place between application agent and platform agent. Between different agent levels, there is either $Order$ operation from higher level agent to lower level ones; or $Report$ operation from lower level agent to its monitoring agent. Currently, agents at the same level can not send message to each other (this limitation is to be explored in the future).

2) Configure:

- Format

$Agent \rightarrow |TriggeringEvent|_{(OPT)} \rightarrow$
$Configure \rightarrow [Res_i : Property|$
$Property \in Res_i : CP \vdash NewValue]$

- Explanation

One agent attempts to configure certain property of a resource (under its supervision) to a new value. The property must be a controllable property of the resource. This operation may be triggered by another event $TriggeringEvent$, for instance a $Report$ operation. $|TriggeringEvent|_{(OPT)}$ means the triggering event is optional.

3) Trace:

- Format

$Agent \rightarrow Trace \rightarrow$
$[Res_i : Property|Property \in Res_i : OP]$

- Explanation

One agent attempts to trace (observe dynamically) certain property of a resource (under its supervision). The property must be an observable property of the resource.

4) Order:

- Format

$Agent \rightarrow |Report(Agent_i)|_{OPT} \rightarrow Order$
$\rightarrow [Res_j : Property|Property \in Res_j : CP$
$\vdash NewValue]$

- Explanation

One agent orders a lower level resource to be configured (its property/properties to have a new value). This operation may be triggered by the report from a lower level agent (indicating the lower level resource is working improperly, for instance). The $Order$ operation is always sent firstly to the lower level agent which monitors the resource to be configured. $Order$ operation always triggers a $Configure$ operation, by which the lower level agent configures the resource according to the $Order$ operation issued by the higher level agent.

5) Report:

- Format

$Agent \rightarrow Report \rightarrow (\{Res_i : Property|$
$Property \in Res_i : OP_{performance}\})$
$\rightarrow Agent_{SUP}$

- Explanation

One agent reports the performance (as special type of observable properties; denoted by $OP_{performance}$) of its monitored resource (group) to its supervising agent ($Agent_{SUP}$).

C. Formal Representation of Hierarchically Monitored System

A hierarchically monitored system can be formally specified on each level with the expressions defined above. At the platform level:

$\mathit{Platform} :\Leftrightarrow (\mathit{PlatformID}, Structure, Status, OP, CP)$
$\mathit{Platform}: Structure = PAG \oplus$
$\{Cluster \mid Cluster : \mathit{Affiliation} = \mathit{Platform}\}$
$\oplus \{Res \mid Res : \mathit{Affiliation} = NonAssigned\}$

The Structure of the platform is dynamically assigned as the union of the platform agent, all configured clusters and all non-assigned resources.

$PAG :\Leftrightarrow (ID, Status, Operations)$
$PAG : Operations :=$
$\{Communication, Trace, Order, Configure\}$ The operations of the platform agent are such a set.

Other observable (OP) and controllable (CP) properties of the platform are design specific based on the underlying system. Each cluster and cell can be similarly defined and omitted here for brevity.

IV. Hierarchical NoC Power Monitoring in Formal Expression

A short-version example is illustrated here to demonstrate applying the formal model to specify hierarchical power monitoring service on NoC platforms. We assume that there are $n * k$ processors (each connected to the network via a switch) partitioned into n clusters (each with k processors and one cluster agent), and each cluster is assigned with a voltage regulator and a PLL. The power of each cluster can be measured from the voltage regulator [3]. The controllable properties of a cluster include its $Structure$, $V_{DD-cluster}$ (supply voltage), $F_{cluster}$(working frequency), formally expressed as $Cluster_i^{i=1..n} : CP = \{Structure, V_{DD-Cluster}, F_{Cluster}\}$. And the observable properties include its $P_{cluster}$(power consumption). Each processor with its switch are grouped into a cell with a cell agent, which can trace the buffer load in the switch dynamically (observable property). The monitoring service is to minimize the total interconnection power of all clusters with the average buffer load in each cluster lower than a threshold value L_{th}[1].

At initiation stage, default supply voltages and frequencies are assigned to each cluster by the platform agent.

$PAG \rightarrow Order \rightarrow [Cluster_i^{i=1..n} : V_{DD}, F_{cluster}$

[1]Buffer load, as a major indicator of network congestion, determines the average packet latency in a best-effort transmission [4].

$\vdash V_{default}, F_{default}]$

After application starts running, at the lowest level, cell agents trace the buffer loads of their own cells, and report the figures to their supervising cluster agents.

$CEAG_j^{j=1..n*k} \rightarrow Trace \rightarrow [Cell_j$:*BufferLoad*$]$

$CEAG_j^{j=1..n*k} \rightarrow Report \rightarrow$

$(Cell_j$:*BufferLoad*$) \rightarrow CLAG_{sup}$

Each cluster agent examines the average buffer load of all affiliated cells, and increases the voltage and frequency if it exceeds the threshold or decreases them otherwise. $|Avg(\{Cell|Cell \triangleright Cluster_i\}$: *BufferLoad*$) > L_{th}|$ is the formal notation expressing a triggering event that the average of all cell buffer loads is higher than the threshold. The cluster agent also reports the cluster power consumption to the platform agent.

$CLAG_i^{i=1..n} \rightarrow$

$|Avg(\{Cell|Cell \triangleright Cluster_i\}$:*BufferLoad*$) > L_{th}|$

$\rightarrow$***Configure***$\rightarrow [Cluster_i : V_{DD}, F_{cluster}$

$\vdash HigherV_{DD}, HigherF_{cluster}]^2$

$CLAG_i^{i=1..n} \rightarrow$ *Report* $\rightarrow [Cluster_i : P_{cluster}] \rightarrow PAG$

At the highest level, the platform agent reports the total power to the application agent. More complicated interactions between them can be defined in the similar manner.

$PAG \rightarrow$ *Report* $\rightarrow [$*Platform*: $Power] \rightarrow AAG$

V. Ongoing Work & Conclusion

The presented formal model for hierarchically monitored system demonstrates the feasibility of a highly abstracted framework which exposes the necessary parameters for defining and specifying monitoring operations on a reconfigurable platform. Such a formal framework not only enables convenient design space exploration in the design process, but also theoretical verification of system properties.

Currently we are elaborating the specification of on-chip components to support more detailed and flexible definition of reconfigurable operations. The types of monitoring operations are being slightly diversified to address better the hierarchical manner of system monitoring. The state space of such hierarchically monitored parallel systems is to be analyzed so as to theoretically verify the expected system properties.

References

[1] Shane Bell, Bruce Edwards, John Amann, Rich Conlin, Kevin Joyce, Vince Leung, John MacKay, Mike Reif, Liewei Bao, John Brown, Matthew Mattina, Chyi-Chang Miao, Carl Ramey, David Wentzlaff, Walker Anderson, Ethan Berger, Nat Fairbanks, Durlov Khan, Froilan Montenegro, Jay Stickney, and John Zook. Tile64tm processor: A 64-core soc with mesh interconnect. In *Proc. Digest of Technical Papers. IEEE International Solid-State Circuits Conference ISSCC 2008*, pages 88–598, 2008.

[2] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. High-performance cmos variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4/5):433–449, 2006.

[3] Chuen-Song Chen. *On-chip current and power measurement techniques for integrated circuits with regulated power.* PhD thesis, University of Rhode Island, 2008.

[2]The expression for decreasing the voltage and frequency is similar and omitted here.

[4] G. Liang and A. Jantsch. Adaptive power management for the on-chip communication network. In *Proc. 9th EUROMICRO DSD Conference*, pages 649–656, 2006.

[5] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43(1):29–41, 2008.

[6] Alexander Wei Yin, Liang Guang, Pasi Liljeberg, Pekka Rantala, Ethiopia Nigussie, Jouni Isoaho, and Hannu Tenhunen. Hierarchical agent architecture for scalable noc design with online monitoring services. 1st International Workshop on Network on Chip Architectures (in conjunction with MICRO41), 2008.