

The communication mechanism in a generic platform

Sylvain Alliot
ASTRON

Oude Hoogeveensedijk 4
7991 PD Dwingeloo, The Netherlands
Email: alliot@astron.nl

Ed Deprettere
LIACS

Leiden University, The Netherlands
Email: edd@liacs.nl

Abstract— This paper describes the communication mechanism of a generic platform (i.e. a metamodel) for the specification of large scale embedded systems (e.g. radars, radio-telescopes, sensor networks). The mechanism is such that the performance of the communication network can be evaluated and validated given different topologies and different computation models. The performance is evaluated stochastically without simulation. This approach is useful when different system architecture alternatives need to be explored rapidly in the early design phases going from requirements to specification.

I. INTRODUCTION

Exploring architectures for distributed signal processing systems such as the next generation radio-telescopes and sensor network as become mandatory. With the rapid evolution of embedded systems technology, these systems integrate more and more functionalities, grow in size and complexity, and become a large fraction of the system costs. The design space for these systems is therefore huge and exploring alternatives in an early stage is indispensable.

The approach we have taken in [1] is to specify a system at an abstract level in terms of an application, an architecture, and a mapping relation between these two. The application is modelled as a network whose nodes are themselves specified as a Kahn Process Network [2]. The architecture is modelled as a network whose processing nodes are themselves modelled as platforms. We consider the whole as a hierarchy of platforms. Crucial to this is how the performance/cost analysis is performed. We take advantage of an application domain¹ to specify platform versions such that the performance of a system can be composed of the individual performances of the subsystems. The design space exploration is supported by a tool set that includes a performance model [3], and the semi-automation of the requirements and constraints derivation [4].

Given the same approach, the objective is to allow for different but also more complex platform topologies. In order to keep the model consistent, we describe formally a unique platform (generic) from which the platform versions are derived. This generic platform is a metamodel for composing the performance model of a system architecture. The consequence of having defined a generic platform is that the communication

mechanism should be unique. The communication between platforms is crucial for analysing the performance in the approach (i.e. without simulation) and should be such that the compositional properties are preserved.

This paper is organised as follows, after giving first an overview of the platforms and their semantics in Section III we present in Section IV the communication mechanism. The communication mechanism includes the arbitration of concurrent communication channels given a system and the evaluation of the communication performances. They are illustrated for an example of an application mapped onto multiple platforms in Section V.

II. RELATED WORK

The models for evaluating the performance of processing network architectures are of different nature depending on the level of accuracy required, the application domain, the size of the system and the ease of use. They can be classified into simulation based (co-simulation of mixed models [5], [6], trace based [7] or analytical models (dynamic [8] or static [3]).

When simulation is used, in the Metropolis design framework [6], the architecture is modelled at various levels of abstraction and with multiple models of computation. Central to the approach is the Metropolis Meta Model (MMM) representation and semantics [9]. The processes in a network and the connections (mediums) are coordinated with a scheduler. This however makes the model more complex since specific interfaces and scheduling mechanisms are needed.

With the LACCES [5] design language (Language of Components and Connectors for Embedded Systems) the software components are separated from the hardware components. The system is described using LACCES to capture the requirements, structure, behavior in the form of hierarchies of components. It enables the evaluation of systems for non functional as well as functional requirements in an integrated way. However, this approach cannot be employed easily at an early stage when large systems need to be composed and alternatives explored as a lot of code is required.

At a higher level of abstraction than the methods previously mentioned, the architecture simulation is avoided in [10] and the communication performance is estimated analytically for a

¹(1) the topologies are fixed and simple; 2) the interfaces can be made simple and uniform

control oriented application on a platform. It applies the real-Time calculus method [11], [8] that is suited to model sporadic and event streams behaviours.

We are also estimating the performance analytically but for an application that is modelled as a process network in which the processes have a predictable behaviour and repeat periodically. In order to limit the modelling effort, a static approach is preferred. The performance model used in [3] is stochastic and hierarchical. The Communication mechanism we present is an improvement of the model as it allows to account for shared resources on arbitrary topologies in the network hierarchy.

III. THE PLATFORM PRELIMINARIES

The platform concept originates from the need to *re-use designs* with more flexible and programmable solutions and to reduce complexity. The initial benefits are primarily economy, a gain in the design time and a wider class of application. A platform can be defined [12] as an abstraction layer in the design flow that hides the details of the underlying layers of abstraction, facilitating refinements in this flow.

In our context, platforms are used to specify into several layers of abstraction. The higher level architecture is made uniform being composed with platforms that obey the same rules. The platforms for large scale array signal processing systems are presented in [13]. Platforms consist of rules to obey when composing with components, methods for construction and validation and communication. Roughly speaking, a platform comprises a number of components that are designed for processing information, and a communication, synchronization, and storage infrastructure. Moreover, platforms are supposed to be application domain specific.

Background and fundamentals:

- The system infrastructure is a network of platforms. A platform is described with a topological design pattern that is a network relationship in between components that are connected via ports.
- The system behavior is a network of processes. Processes communicate and synchronize according to a computation model ².
- Starting from high level requirements, the process network further refined as networks of techniques is mapped onto components; i.e. the behavior is mapped onto the infrastructure. After the network assignment, a set of constraints are derived on the components of the platforms according to the network relationships.
- As a result of the mapping, a detailed description of the architecture (i.e. number of resources and performances) is obtained. Then, for every component, the performance is evaluated given the resources available. This information is taken bottom up in the model for validation.

²for systems implementing parallel, signal processing applications, a Kahn Process Network is a convenient model. For modeling control communication other models such as Finite State Machines or Petri Nets are more convenient

A. Semantics

The characterisation of a platform is incomplete when it comes to the issue of constructing platforms.

Processing, communicating/synchronizing and storing are distinct functions that match behaviors of computation, communication and storage, respectively that are the constituents of a platform. A processing (communication, storage) unit can be any of a partially ordered set of processing (communication, storage) component types. Types in the set may be further subdivided into subtypes down to specific components.

Platforms p and component types c have ports. A port ρ is a triple (n, d, t) where n is a unique name, d is the port type: Input, Output, or Input-Output, and t is a compatibility type. Let ρ_k and ρ_l be two ports, a relation r is defined over ρ_k, ρ_l iff there is a wire from ρ_k to ρ_l in the platform. We denote $r(\rho_k, \rho_l)$ as $\rho_k \longrightarrow \rho_l$.

The platform is composed with ports, components and relations that are fixed in a topology. Let pt be a platform topology $pt = (C, M, R, P)$ where C is a set of component types, M the multiplicity, R is a set of relations, P a port set.

A special type of component is the connection (line, bus). $Line \subseteq Connection$, $Bus \subseteq Connection$, that is a medium for communication in between components of different types. Given the platform topology: ports, component types and relations; communication channels and communications paths are established.

Communication channels : A communication channel Cc in between two ports ρ_k and ρ_l of components $k, l \in C \cap Connection$ on a platform is a triple $(r1, r2, m)$. With $m \in Connection$ a component, $r1(\rho_k, \rho_m) | \rho_m \in P_m$ and $r2(\rho_l, \rho'_m) | \rho'_m \in P_m$ respectively the relations to m .

Communication path : A communication path Cp in between two ports ρ_k and ρ_l of components $k, l \in C \cap Communication$ on a platform is a pair (CC, X) . CC a set of communication channels and X a set of components $X \in Communication$.

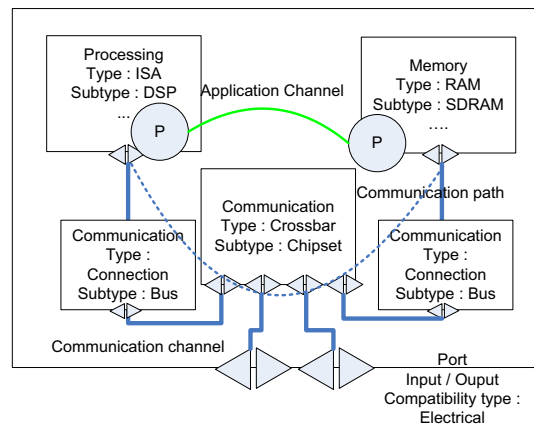


Fig. 1. Example of a platform topology. The components and the ports are indicated with their types, and subtypes. An application network with two processes (P) communicate via the communication path (dashed curved line).

To illustrate, a platform in Figure 1 is composed of a DSP, a SDRAM, two busses, a crossbar and two ports. The communication channels are in between the crossbar and the processor, the memory and the ports of the platform. There is a communication path in between the memory and the processor on the application for two processes communicating.

B. The generic Platform

The generic platform comprises a unique set of rules and methods for composing, constructing and evaluating. In short, the composition of a platform and then the instantiation is forced by the compatibility types (components, ports). The performance on the platform is validated with an hypothesis test on every metric. These metrics are evaluated as a function of the components' metrics that are evaluated separately given a set of constraints. Versions of the generic platform can have different functions for aggregating the metrics. The communication mechanism is made also unique as presented next.

IV. COMMUNICATION MECHANISM

We established a separation in between the computation model chosen for the behavior and the architecture. The computation model determines the communication mechanism in between processes P in a network in the application domain, however it should match the architecture. The *application channels* Ac are mapped $M(Ac)$ onto the communication paths such that $Cp_i = M(Ac_i)$, (Cp_i, Ac_i) is a matching pair. However, many application channels after mapping can share the same communication channels. Therefore, the platform must coordinate the communication between the components to allow the inter-process communication in the application considering the channel sharing.

The platform communication mechanism consists in allocating a communication bandwidth on the connection components where processes are performed repeatedly indefinitely. At this coarse level, the bandwidth is shared in between application channels on the same component. The mechanism involves an arbitration of the communication channels for exchanging data within and across platforms. The communication is arbitrated/scheduled globally on the platform such that the conflicts in the sharing of communication paths is avoided.

This section is organised as follows: first we establish the communication system after the mapping of an application onto a platform. This communication is driven by rules and priorities that are given next. Third, we present the communication arbitration and last we discuss briefly the performance evaluation of the communication on the platform.

A. The communication system

In the application domain, the computation model can exhibit the notion of precedence and parallelism. With P_1 and P_2 two processes that communicate and Ac the application channel, the precedence denoted \prec (or the edge between P_1 and P_2) is $P_1 \prec P_2$. P_2 is only executed after a receive in P_2 , and a send from P_1 . The parallelism is denoted with \parallel :

$P_1 \parallel P_2$. P_1 and P_2 have no precedence and can be executed in parallel.

An application $A = (P, AC, E)$ is a directed multi-graph with :

- a set of processes P ,
- a set of communication channels AC ,
- and a set of edges E . E represents the precedence constraints in between communication channels:
 $\forall Ac_i, Ac_j \in AC, e = (Ac_i, Ac_j) \in E$ iff $Ac_i \prec Ac_j$

Parallelism and precedence are seen in the architecture domain together with the notion of time. The notion of time in the communication system is illustrated in Figure 2.

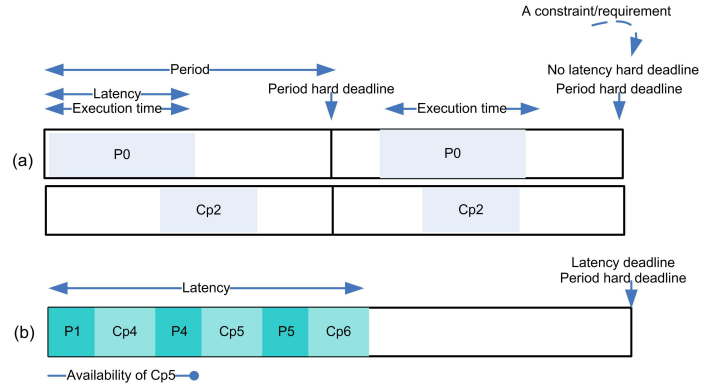


Fig. 2. The time vocabulary.

The processes in the network run indefinitely however the time has been subdivided into cyclic periods where the processes and the communications are scheduled. The time t is continuous, T is the scheduled period duration (makespan of A) that repeats indefinitely, $n = 1..N$ are indexes of time intervals (i.e. slices) in a sequence. The duration d in a communication path is a measure of the time for communication and includes the communication startup. It corresponds to a number of tokens transmitted within a time constraint (i.e. related to the throughput), the transmission overhead and to some mapping objectives for the communication. The parallelism in a slice is denoted \parallel_t .

A communication system on a platform is a directed multi-graph $CG = (CP, E, O, d, r, a)$ where :

- CP is a set of communication path,
- E is a set of edges between communication paths: $e = (Cp_i, Cp_j) \in E$ iff $Ac_i \prec Ac_j$,
- O is a set of transaction ordering edges for strongly connected paths within an iteration period (directed multi-graphs with the same period) : $e = (Cp_i, Cp_j) \in O$ iff $Cp_i \prec Cp_j$,
- d is a function returning a duration for each communication path with $\forall Cp_i \in CP, \exists d_i$,
- $r(R,L)$ is a set of hard deadline interval requirements, R is the period bound, L is the relative deadline (i.e. the latency bound) $\forall Cp_i \in CP, \exists r_i \in r$,
- a is a function returning a delay (starting availability time) for a communication path in a period. $\forall Cp_i \in$

$CP, \exists a_i \in a. a_i = \max(a(\rho_k), a(\rho_l))$ is a delay as a function of the blocking effects that are also delays on the ports of Cp_i .

The edges E correspond to the communication edges in the application within an iteration period. The requirement set r is known for a system prior to mapping the application onto an architecture. The duration d , and the availability time a are estimated after the mapping on the individual components of the platform and their independent performance evaluation.

B. Communication rules

By construction, a pair (Cp_i, Ac_i) matches if at any given time when Ac_i is active, every communication channels in Cp_i can be shared or is not used. A communication system matches an application, if the precedence relations are respected. If Ac_i and Ac_j $j \neq i$ are application channels mapped on a communication path respectively Cp_i, Cp_j then the following rules apply:

If $Cp_i \cap Cp_j \neq \emptyset$ and $Ac_i \parallel Ac_j \Rightarrow M(Ac_i) \parallel_t M(Ac_j)$.

If $Cp_i \cap Cp_j = \emptyset$ and $Ac_i \parallel Ac_j \Rightarrow M(Ac_i) \parallel_t M(Ac_j)$.

If $Ac_i \prec Ac_j \vee M(Ac_i) \prec M(Ac_j) \Rightarrow M(Ac_i) \parallel_t M(Ac_j)$.

Given communication paths $Cp = (CC, X)$. For every component $m \in X$ there is a number of communication paths Nm a component shares in parallel $Nm \leq MaxNm$, $MaxNm$ the maximum number of paths that do not intersect in the component. Communication components such as cross bars and switches allow simultaneous communications. If $\forall Nm, Nm > MaxNm \Rightarrow Cp_i \cap Cp_j \neq \emptyset$.

If two communication paths have no precedence but share a resource, then the communication is ordered according to priorities. If the priorities change or a new task with higher priority competes for a resource, the communications that are started can be interrupted in between individual tokens and the higher priority task will access the resource. Some resources cannot be preempted, consequently if one of the components in Cp is not preemptable, Cp cannot be preempted. If the communication path is preemptable, the communication can be interrupted in between tokens. If there are no specified priorities in the application prior to mapping, the communications can have static priorities. For instance in the RM scheduling [14], the inter-component communication is arbitrated at the level of platforms such that the priority of each communication path is a monotonically decreasing function of their period (RM is a preemptive scheduling policy with fixed priorities). The inter-component communication is arbitrated at the level of platforms such that the priority of each communication path is respected.

C. Communication arbitration

The communication arbitration on a platform given a communication system CG , and a level of confidence α is a function $CA(CG, \alpha)$ returning :

- a sequence of groups of paths s for a period of CG ,
- time intervals g for each group,
- a set of metrics l, la that are the slacks l , and latency la for the directed multi-graphs with the same period

(strongly connected paths) $\forall r_i \in r, SCP_i = r_i \cap CP$ and $\forall SCP_i, \exists l_i \in l$,

- a metric MLM the maximum latency mean,
- a metric MCM the maximum cycle mean,
- and an error if no sequence exist.

The function orders the communication in the time intervals following the matching rules and the priorities. Given a communication system on a platform, groups of matching communication paths are established. These groups are ordered and given a duration. Then performances are measured and validated.

The application channels mapped on a platform are grouped in sets G , where $\forall M(Ac_i), M(Ac_j) \in G \Leftrightarrow M(Ac_i) \parallel_t M(Ac_j)$. G corresponds to non intersecting communication paths on the platform at any given time. Moreover the groups are ordered by the dependance relations to match the application dependencies and the priorities. The outcome of the arbitration is a sequence of groups $s = G_1, \dots, G_n, \dots, G_N$ in the time interval T (the period of CG) with G_n in the time interval $t_n \leq t < t_{n+1}$, $g_n = t_{n+1} - t_n$.

Even with unambiguous priorities and dependencies, there are many possible combinations of orders and groups with each different effects on the communication system performance. They cannot be all tested in general, therefore it is proposed to take a Gantt chart as an input for the arbitration of GC . It specifies an ordered transaction schedule s and optionally a fully-static schedule (s, g) for the communication arbitration. The schedule is admissible if it respects all the precedences and requirements in CG and verifies the matching rules determined above. As an alternative, if no schedule is given, a default procedure is used. To establish a schedule of the communication according to these rules, we assume that once started the communications are not interrupted unless a higher priority communication is preemptive. Note that according to the definition of a group, the same application channel in a group can be present in adjacent time intervals. This sequence matches the application model and the resource sharing on the architecture, however there can be many possible sequences.

The communication arbitration on the platform can be derived with a default procedure. Once the ordering is established, error bounds for the arbitration are also evaluated following principles given in [15] considering uncertainties in the estimates of the durations.

D. Performance evaluation

In [3] it is proposed to calibrate an analytical model of the performance and to extrapolate / interpolate with a limited set of measures. The communication can be specified as an instance of a parameterized technique (e.g. synchronous transmission, asynchronous handshaking) and the analytical model returns performance measures such as the latency, the execution time or the throughput but also the power consumption. Prototyping for a communication type, interpolation and extrapolation are methods associated to an error estimate. In the worst case, when no measures are available, the procedure

proposed in [3] gives a coarse estimate with a large error bound.

The values of the slacks l are evaluated for every strongly connected paths $SCP_i \in SCP$. The makespan is $T_i = \sum_{n=entry}^{exit} g_n$ where the entry and exits of the graph are in G_{entry}, G_{exit} . For j cycles in the sequence, the slack is estimated as $\langle l_i \rangle = R_i - \max(\langle T_{i,j} \rangle)$ with $\sigma_l = \sigma_T$ and $l \geq 0$. The latency is estimated as $la_i = \sum_{n=entry}^{exit} (k_n) + T_i$.

We identify the critical iteration in the sequence with the maximum cycle mean. MCM is estimated as $MCM = \max(T_{i,j}/R_i), \forall SCP_i \in SCP$ that is an indication of the maximum throughput that can be obtained, with $1 \geq MCM \geq 0$.

We identify the critical path with the maximum latency mean. L_i is the constraint latency for SCP_i . MLM is estimated as $MLM = \max(la_i/L_i) \forall SCP_i \in SCP$ with $1 \geq MLM \geq 0$.

If the communication errors are independent and their probability distribution is normal then the mean value μ_T and the standard deviation σ_T of the makespan is a function of the number L of communication paths in the sequence:

$$\mu_T = \sum_{i=1}^L \mu_i \text{ and } \sigma_T = \sqrt{\sum_{i=1}^L \sigma_i^2}.$$

However, estimation errors contribute in the grouping procedure that can have an unpredicted behaviour (a non normal distribution of the durations for different sequence pairs). Therefore the makespan estimator does not have the same properties as the independent communications. Nevertheless, the arbitration procedure can be tested with a level of confidence α , returning a makespan $[T]_\alpha$ such that $Pr(T > [T]_\alpha) \leq \alpha$. The probability of α in the normal distribution corresponds to $B\sigma$ the level of the test. If the duration of a communication path is evaluated in the procedure as $[d_i]_\alpha = \mu_i + B\sigma_i$, the algorithm will be conservative. We adopt instead a method proposed in [16] for job scheduling (static schedule) and similar to [17] for scheduling on a single processor. The probability durations in the algorithm are weighted by $q = B/\sqrt{L}$ such that the duration of Cp_i estimated in the procedure is $[d_i]_\alpha = \mu_i + q\sigma_i$.

V. EXAMPLE

We illustrate the communication mechanism for a simple example taken from a sub-system developed at Astron. The application is an adaptive signal processing algorithm shown in Figure 3. Two processes DBF and AWE are in a network. The processes are refined into subnetworks that are strongly connected paths. Two constraints/requirements for the processes are the period TA and TB hard deadlines and the latency for TB . The subnetwork communications in the AWE are ordered to meet the latency constraint such that with $A = (P, AC, E)$ and $P = \{P[0, \dots, 7]\}$, $AC = \{Ac[1, \dots, 7]\}$, $E = \{(Ac1, Ac5), (Ac5, Ac6)\}$.

The application network is mapped onto two platforms called "A" and "B" in Fig. 4. The platform "A" is similar to the platform shown in Fig. 1. In platform "A" a communication network (CN) connects 2 memory and 2 processing

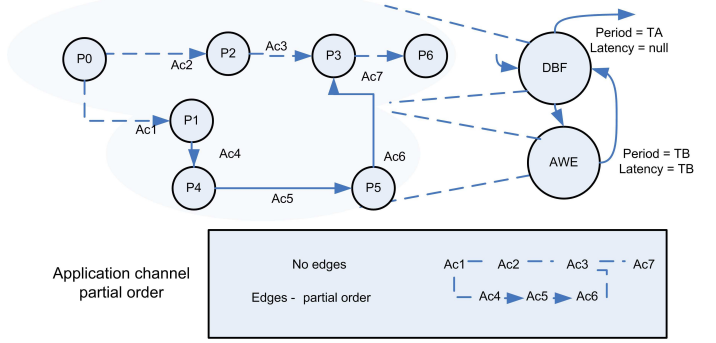


Fig. 3. The application network. Two communicating processes DBF and AWE are refined into subnetworks. The two refined networks are strongly connected (they share the same period) and are exposed to the architecture in the communication system as graphs SCP_A and SCP_B for the periods T_A and T_B .

components with a crossbar. The communication paths corresponding are shown with curved lines passing through the communication network.

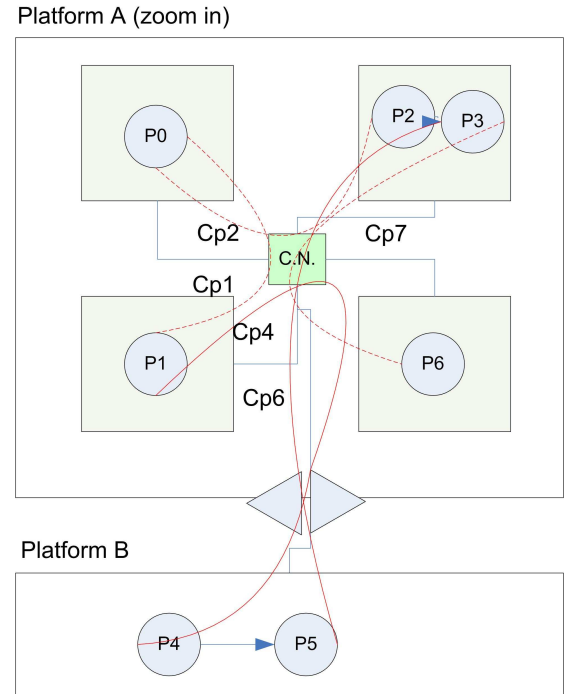


Fig. 4. The communicating system on platform "A" after the partitioning of the application network on platforms "A" and "B".

We obtain a communication system on the platform "A" that is $CG = (CP, E, d, r, a)$ with $CP = \{Cp[1, \dots, 4], Cp6, Cp7\}$, $E = \{(Cp4, Cp6)\}$ and d, r, a as indicated in Table I.

A possible arbitration outcome for the communication system following the default procedure is shown in Figure 5. The same Gantt chart can be given for the first period TA as $s = \{(Cp1, Cp7), (Cp1), (Cp2, Cp4), (Cp4)\}$ and $e =$

TABLE I
REQUIREMENTS, DURATIONS, AVAILABILITY OF CG

	Duration $[d]_\alpha$	σ	Requirement $r(R,L)$	Availability (a)
Cp1	2	0.2	5	
Cp2	2	0.2	5	
Cp4	4	0.4	20,20	
Cp6	2	0.2	20,20	5
Cp7	1	0.1	5	

$\{1, 1, 2, 1\}$. The slacks in the schedule are for $l(DBF)$: $\mu_l = 1$, $\sigma_l = 0.3$ and for $l(AWE)$: $\mu_l = 5$, $\sigma_l = 0.44$. The maximum latency mean is $MLM = 0.8$ that corresponds to the latency of Cp6. The maximum cycle mean is $MCM = 0.9$ with $MCM(DBF) = 0.8$ and $MCM(AWE) = 0.9$. $MLM < 1$ and $MCM < 1$ thus the communication system on the platform is valid.

In this example and considering the communication on the platform, the adaptive weight estimation is the limiting subnetwork. Without taking the communication mechanism into account, the predictions for $MCM(AWE)$ would have been underestimated to 0.7.

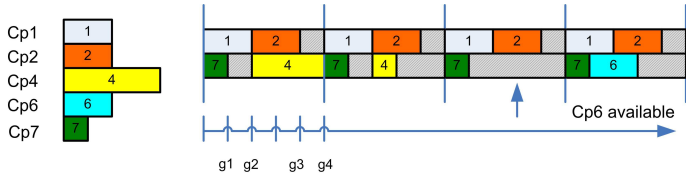


Fig. 5. Gantt chart. The period deadlines on the time line are indicated with long bars. The time slices are g_1, g_2, \dots and the durations in the schedule are given different colours. The idle periods, delays due to the sharing of resources and the slack are indicated filled with diagonal bars.

CONCLUSION

We present a communication mechanism for a Metamodel of a platform whose system architecture is static and the performance evaluated analytically (it is assumed that the application behaviours are predictable and execute periodically). Large systems can be specified and evaluated/verified rapidly with hierarchies of such platforms without a full simulation. As for dynamic analytical performance models, this model accounts for shared resources in the network infrastructure and topology variations.

ACKNOWLEDGMENT

The authors would like to thank Laurentiu Nicolae for his review comments and his contribution to the models developed in the STW-MASSIVE project.

REFERENCES

- [1] S. Alliot, E. Deprettere, "Architecture exploration of a large scale system," *Proc. Fifteenth IEEE International Workshop on Rapid System Prototyping*, pp. 217–225, June 2004.
- [2] G. Kahn, "The semantics of a simple language for parallel programming," in *Proceedings of the IFIP Congress 74*. North-Holland Publishing Co., 1974.
- [3] S. Alliot, "A performance/cost estimation model for the large distributed array signal processing system and specification," *SAMOS 03 proceedings*, pp. 154–160, July 2003.

- [4] L. Nicolae, E. Deprettere, "Derivation of constraints," *Computer Systems: Architectures, Modeling, and Simulation, LNCS 3133*, pp. 550–559, July 2004.
- [5] D.C.C. Peixoto and C.da Silva Junior, "A framework for architectural description of embedded system," in *Proc. Software and Compilers for Embedded Systems, SCOPEs*, ser. LNCS 3199. Springer, Sept. 2004, pp. 2–16.
- [6] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, A. L. Sangiovanni-Vincentelli, "Metropolis: An integrated electronic system design environment," *IEEE Computer*, vol. 36, no. 1, pp. 45–52, 2003.
- [7] V. Zivkovic, E. de Kock, E. Deprettere, and P. van der Wolf, "Fast and accurate multiprocessor architecture exploration with symbolic programs," in *Proceedings of Design Automation and Test in Europe (DATE'03)*, March 2003.
- [8] L. Thiele, S. Chakraborty, M. Gries, and S. Kunzli, "Design space exploration of network processor architectures," vol. 1, 2003.
- [9] A. S.-V. J. Burch, R. Passerone, "Overcoming heterophobia: Modeling concurrency in heterogeneous systems," *Proceedings of the second International Conference on Application of Concurrency to System Design*, 2001.
- [10] E. Wandeler, L. Thiele, M. Verhoef, P. Liverser, "System architecture evaluation using modular performance analysis - a case study," 2004.
- [11] S. Chakraborty, S. Kunzli, L. Thiele, "A general framework for analysing system properties in platform-based embedded system design," *Proc. 6th Design, Automation and Test in Europe*, pp. 190–195, 2003.
- [12] A. Sangiovanni-Vincentelli, "Defining platform-based design," *EEDesign of EETimes*, February 2002.
- [13] S. Alliot, "Architecture exploration for large scale array signal processing systems," Ph.D. dissertation, Leiden University, December 2003.
- [14] J. W. L. "C. L. Liu, "Scheduling algorithms for multi-programming in a hard real time environment," *Journal of the Association for Computing Machinery (JACM)*, pp. 40–61, 1974.
- [15] R. H. Mohring, "Scheduling under uncertainty: bounding the makespan distribution," pp. 79–97, 2001.
- [16] J. Christopher Beck and Nic Wilson, "Job scheduling with probabilistic durations," in *16th Conference on Artificial Intelligence*, 2004.
- [17] R. L. Danels, J. E. Carrillo, "beta-robust scheduling for single-machine systems with uncertain processing times," *IIE Transactions*, vol. 29, pp. 977 – 985, 1997.