Timing analysis model for network based multiprocessor systems

Arno Moonen[1,2], Marco Bekooij[2] and Jef van Meerbergen[1,2]

[1]Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

[2]Philips Research Laboratories Eindhoven
WDC31, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands

{Arno.Moonen,Marco.Bekooij,Jef.van.Meerbergen}@philips.com

Abstract—**In this paper an embedded multiprocessor system on top of a network on chip is proposed which is amenable for timing analysis. This multiprocessor system is intended for multimedia application that process data streams. The temporal behavior of applications executed on this multiprocessor system is derived with a Synchronous Data Flow (SDF) graph in which computation, communication, buffer sizes as well as arbitration is modeled. This graph can be transformed in an event graph which is a special case of a Petri net from which properties like the minimal throughput can be derived with results of MaxPlus Linear System Theory [1]. Our main contribution in this paper is an SDF model of the network in which an arbiter is applied which allows the transfer of a possibly varying but bounded number of words per period.**

Keywords— **Embedded Systems, Multi-processor, Predictable temporal behavior, Network-on-chip**

I. Introduction

Consumers have high expectations about the quality and robustness of multimedia systems like DVD players, set-top boxes, and television sets. The applications executed by these multimedia systems require a computational performance in the order of 10-100 GOps. Such a performance can be delivered by embedded programmable multiprocessor systems. In order to guarantee a certain quality level it is necessary that these multiprocessor system exhibit a predictable temporal behavior. The communication between processors can be made predictable by making use of a loss less packet switched network that supports connections with a guaranteed throughput service. Connections with a guaranteed throughput service provide a guaranteed bandwidth and a bounded delay and jitter. Given these guaranteed throughput connections it is shown in this paper that the worst-case temporal behavior of the system can be derived by making use of an SDF model in which the computation, communication, arbitration and storage is modeled. A comparison with related network performance analysis models is presented in the next section before our SDF model is presented in subsequent sections.

II. Related work

An overview of service disciplines for packet-switched computer networks with a guaranteed performance is given in [2]. The applied service discipline determines when packets are transferred from one specific input of a router in the network to one specific output of the router. The maximal delay and jitter of a connection is derived given the applied service disciplines in the network and a characterization of the traffic at the inputs of the network. Derivation of the maximal delay requires for so-called work-conserving service disciplines that the traffic on the boundary as well as inside the network can be characterized. Characterization of traffic in the network is difficult and may not always be possible if there are so-called traffic loops.

In this paper it is shown that the need to characterize traffic can be eliminated by making use of flow control between every producer and consumer of data in the system. Examples of producers and consumers in the system are processors and routers. It will also be shown that due to the flow control the network and the system cannot become unstable even if there are traffic loops. The worst-case arrival time of data at any point in the system is derived from an SDF model of the system. This SDF model includes an SDF model of a network connection. The SDF model of a network connection captures accurately the worst-case temporal behavior despite that potentially a variable number of words is transferred per period.

In this paper the term arbitration policy is used instead of service discipline because all local arbitration decisions for links in the network as well as processors and memory ports are handled in a uniform way.

III. Outline

For sake of clarity, we step-by-step introduce new concepts and elements in this paper. Therefore the organization of this paper is as follows. First the SDF graph and its properties are described in section IV. Then, in section V, an SDF graph is used to derive the temporal behavior of a multiprocessor system which executes two software tasks which communicate via a point to point connection. In section VI this point to point connection is replaced by a network with shared interconnect. It is assumed that TDMA arbitration is applied inside the network and that each period it is guaranteed that up to N words of data can be transferred. For this type of arbitration an SDF model is introduced and a proof of its correctness is presented in section VII. That an SDF model of such an arbiter can be derived is a surprising result because the amount of data transfered per period does not need to be constant. Given this SDF model of the arbiter, the temporal behavior of a system with a network is derived. In section VIII, a network is introduced in which flow control via a guaranteed throughput connection in the opposite direction is applied. In the same section, an SDF model is presented which captures this type of flow control. Finally in section IX a multiprocessor system is described which allows the execution of multiple tasks per processor. Each of these tasks might produce a substantial amount of data. The data produced by a task is stored in its local memory after which it is copied by a Communication Assist (CA) in a small but fast FIFO at the input of the network. The worst-case temporal behavior of the system is derived with an SDF model which includes a model of a network connection. This network model is applied in section X in the SDF model of a system in which multiple software tasks are executed on a processor.

IV. Synchronous Data Flow graph properties

Some useful properties of SDF graphs are presented in this section. These SDF graphs are used in this paper to derive the worst-case temporal behavior of a multiprocessor system which includes a packet-switched network.

First of all, an SDF graph is defined as follows:

Definition 1 (Synchronous Data Flow Graph.) The tuple (V, E, d, T_v) defines a Synchronous Data Flow (SDF) graph, where

- V is the set of nodes (actors),
- $E \subseteq V \times V$ is the set of edges,
- $d_{uv} : E \rightarrow \mathbb{N}$ is a function describing the number of initial tokens on an edge $(u, v) \in E$,
- $T_v : V \rightarrow \mathbb{R}^+$ is a function describing the worst-case execution time of actor $v \in V$.
- $O_{uv} : E \rightarrow \mathbb{N}$ is a function describing the number of tokens produced on edge $(u, v) \in E$ by actor u.
- $I_{uv} : E \rightarrow \mathbb{N}$ is a function describing the number of tokens consumed from edge $(u, v) \in E$ by actor v.

An arbitrary SDF graph is depicted in figure 1. The nodes in an SDF graph are called actors. Actors have a well defined input/output behavior and a worst-case execution time. Actors produce and consume tokens. A token is a container in which a fixed amount of data can be stored and is depicted in figure 1 as a black dot. If more than one token is present on an edge then the number of tokens (d_{uv})is specified next to the dot. An actor has a worst-case execution time which is denoted by T_{Ax} in figure 1. An actor can fire (starts its execution) after at least the number of tokens is available as is specified at the head of the data edge of every incoming edge of the actor. The specified number of tokens is consumed from the input edges of the actor before the execution of an actor finishes, that is within the worst-case execution time of the actor. The number at the tail of an edge denotes the number of tokens an actor produces before the execution of the actor finishes. Actors with internal state are modeled in an SDF with a self edge, like the self edge of actor A4 in figure 1. This self edge is given one initial token such that the next execution cannot start before the previous execution of the actor is finished.

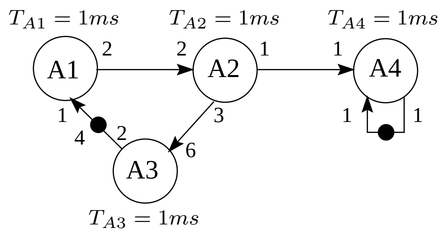

Fig. 1. A Synchronous Data Flow (SDF) graph example.

An SDF graph can be transformed into a Homogeneous Synchronous Data Flow (HSDF) graph on which analysis is performed. An algorithm which transforms any SDF graph into an HSDF graph is described in [3]. An HSDF graph is a special case of an SDF graph in which the execution of an actor results in the consumption of one token from every incoming edge of the actor and the production of one token on every outgoing edge of the actor.

An HSDF graph can be executed in a self-timed manner, which is defined as a sequence of firings of HSDF actors in which the actors start immediately when there is at least one token on each input of the actor. In the case that the HSDF graph is a strongly connected graph and a FIFO ordering is maintained for the tokens, then the self-timed execution of the HSDF graph has some important properties. A FIFO ordering is maintained if the completion events of firings of a specific actor occurs in the same order as the corresponding start-events. This is the case if an actor has a constant execution time or belongs to a cycle in the HSDF graph with only one token. In [1] are the properties of the self-timed execution of such HSDF graphs derived with MaxPlus algebra.

First of all, the most important property of the self-timed execution of an HSDF graph is, that it is deadlock-free if there is on every cycle in the HSDF graph at least one initial token. Secondly, the execution of the HSDF graph is monotonic, i.e. decreasing actor execution times result in non-increasing actor start times. Third, an HSDF graph will always enter a periodic regime. More precisely, there exist a $K \in \mathbb{N}$, an $N \in \mathbb{N}$ and a $\lambda \in \mathbb{R}$, such that for all $v \in V$, $k > K$ the start time $s(v, k+N)$ of actor v in iteration $k+N$ is described by:

$$s(v, k+N) = s(v, k) + \lambda \cdot N \tag{1}$$

Equation 1 states that the execution enters a periodic regime after K executions of an actor in the HSDF graph. The time one period spans is $\lambda \cdot N$. The number of firings of an actor v in one period is denoted by N. Thus, λ is equal to the inverse of the average throughput measured over period.

The Maximum Cycle Mean (MCM) [3] of an HSDF, which is equal to λ, is given by equation 2. The MCM of an HSDF graph is also called in literature the maximal cost to time ratio [4]. The Cycle Mean (CM) of a simple cycle c in the HSDF graph G is given by equation 3. In this equation denotes $d(c)$ the number of initial tokens on the edges in a cycle c. The Worst Case Execution Time (WCET) of actor v is denoted by WCET(v). The MCM of an HSDF graph can be derived with a pseudopolynomial algorithm [5] [4].

$$\text{MCM}(G) = \max_{c \in C_G} \text{CM}(c) \tag{2}$$

$$\text{CM}(c) = \sum_{v \ on \ c} \text{WCET}(v)/d(c) \tag{3}$$

The worst-case start-times of the actors during the transition state as well as the steady state can be derived by simulation. During this simulation, all actors must have an execution time equal to their worst-case execution time. The start-times observed during this simulation are equal to the worst-case start times of the actor due to the monotonicity of the HSDF. From equation 1 it follows that a periodic regime will be entered and therefore simulation can be stopped after the first period of the periodic regime.

V. Communication via a FIFO

In this section the throughput of a simplified multiprocessor system is derived by applying the analysis techniques that were presented in the previous section. For ease of understanding, a multiprocessor systems is considered in this section with only two processors which communicate via a FIFO. The application on this multiprocessor system consists of two communicating HSDF actors. Similar techniques will be applied in section IX to derive the throughput of a multiprocessor system which communicate via a network and execute an application described by an SDF graph.

In figure 2 a multiprocessor system is shown which consists of two processors. These two processors communicate via a FIFO. The application executed on this system is represented by an HSDF graph which is shown in figure 3. It is assumed that the actor P is executed on processor *proc1* and the actor C is executed on processor *proc2*. The tokens communicated between actor P and actor C have a size of one word which is produced (or consumed) by a processor in one clock cycle. Processor *proc1* is stalled (for example by stopping its clock) if the FIFO is full and processor *proc2* is stalled if the FIFO is empty.

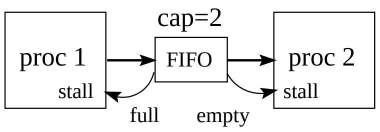

Fig. 2. A multi processor system which consist of two processors which communicate via a FIFO. The data producing/consuming processor is stalled if the FIFO is full/empty. The capacity of the FIFO is assumed to be 2 tokens.

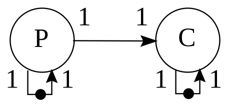

Fig. 3. An implementation unaware HSDF graph. Succesive executions of the actors result in a stream of tokens between the actors.

The throughput of this multiprocessor system can be derived with the implementation aware HSDF graph shown in figure 4. Each actor in this HSDF graph is annotated with its Worst-Case Execution Time (WCET). The WCET of an actor must include the time the processor is stalled during the execution of the actor. The processors are not stalled in this example during the execution of an actor because the actors fire as soon as all space or data is available to finish their execution. This is the case because the next firing of actor P takes place as soon as the previous execution of this actor finishes and there is space for one token in the FIFO. That there is space is indicated by the deactivation of the full signal. After the full signal is deactivated the processor can start executing the actor and it will finish its execution because it is assumed that the full signal is ignored by the processor till the execution of the actor is finished. The actor C will fire as soon as there is one token in the FIFO and the empty signal is deactivated. The processor will not stall during the execution of actor C because the empty signal is ignored till the processor finishes the execution of actor C.

In this example it is assumed that the capacity of the FIFO between the processors is two tokens. This FIFO capacity is modeled in the implementation aware HSDF in figure 4 by the 2 initial tokens on the edge from actor C to actor P.

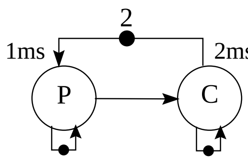

Fig. 4. An implementation aware HSDF graph in which the edge from C to P with 2 initial tokens models a FIFO with a capacity of 2 tokens.

The MCM of the HSDF graph in figure 4 equals the inverse of the guaranteed minimal throughput of the system. This throughput will be obtained if there is no interaction with the environment. For systems that do interact with the environment it should be guaranteed that no data is lost due to overflow or underflow of the buffers between the system and the environment. A typical example of such a system is a system in which the input data is provided by a strict period external source like an A/D converter and consumed by a strict period external sink like a D/A converter. In order to verify whether the FIFOs at the input and output of the system do not overflow or underflow it is needed that the strict period source and sink are modeled as actors in the HSDF graph as is shown in figure 5. The source and sink actors are given a WCET equal to the length of period of the A/D's and D/A's strict periodic clock. The self edge with one initial token ensures that the next execution of the source and sink actor cannot start before the previous execution is finished. Thus, the self-edge in combination with the WCET of the source and sink actors enforces a maximal execution frequency of these actors during the self-timed execution of the HSDF graph in the simulator. During selftimed execution of the HSDF in a simulator it should be verified that the time between successive executions of the source as well as the sink actor is equal to the WCET of these actors. This simulation can be stopped after the first period of the periodic regime. If the source and sink actor execute strict periodic in the simulator then it is guaranteed that, in an implementation of the system, the FIFO between source and the system never overflows and the FIFO between the system and the sink never underflows. The reason is that, due to monotonicity, tokens will not arrive and depart later in the implementation than during a simulation run in which all actors have an execution time equal to their worst-case execution time. If tokens do not depart later than during the simulation run, then this results in the same or less tokens in the FIFO between the source and the system. If tokens do not arrive later than during the simulation run, then a greater or equal number of tokens is in the FIFO between the system and the sink in the implementation.

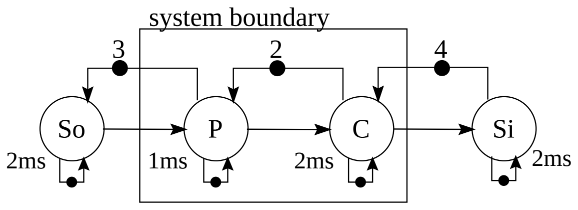

Fig. 5. An HSDF graph which is used to prove that, given a strict periodic source and sink, the FIFOs at the input and at the output of the system have sufficient capacity.

In the next section a network instead of FIFO between two communicating processors will be introduced and the worst-case arrival times of tokens will be derived.

VI. Communication via a network

In this section a network for the communication between processors is introduced in the multiprocessor system. It is assumed that this network supports the guaranteed throughput service which guarantees that a predefined amount of data can be transferred per period. Given such a network it is possible to derive the worst-case temporal behavior of the system.

The packet switched network that is considered in this paper contains shared interconnect between two routers which is called a link between these routers. The routers in this network are schematically depicted in figure 6 as switches which are during a predefined time interval in one position and then switched to the next position. During this time interval a predefined amount of data can be transferred from an input FIFO into an output FIFO. Data does not need to be buffered inside the network because it is assumed that the switches in this network are synchronized.

The controller which moves the switch into the next position can be seen as a Time Division Multiple Access (TDMA) arbiter which grants the link during a time-interval for transfer of data from an input FIFO into an output FIFO. The TDMA arbitration performed by this arbiter is modeled in an HSDF model which is used for the derivation of the worst-case temporal behavior of the system.

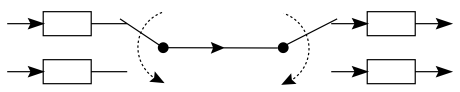
Fig. 6. Model of a time shared unidirectional connection in the network.

The switch in figure 6 is represented as a router in the multiprocessor system in figure 7. This router can be seen as a special purpose processor which transfers data during an interval from one of its inputs to one of its outputs. No data is transferred if the input FIFO of the router is empty or if the output FIFO is full.

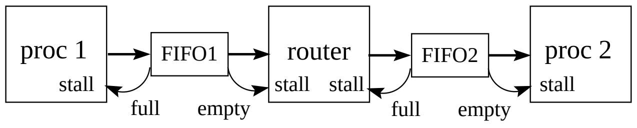

Fig. 7. Two processors which communicate via a packet switched network. The data producing/consuming processor is stalled if the *local* FIFO is full/empty.

An abstract model of the TDMA arbitration performed in this router is depicted in figure 8. The arrival time of the j-th token in FIFO1 and FIFO2 is respectively denoted by $a(j)$ and $b(j)$. The TDMA arbitration is represented by a time wheel which rotates every period T_A. During the interval T_{A1} up to N tokens are transferred from FIFO1 into FIFO2. It is not exactly known when these tokens are transferred during the interval T_{A1} but it is sure that these tokens have arrived in FIFO2 at the end of interval T_{A1}. The end of the interval T_{A1} is denoted by $f(j)$. At this time the j-th token is transferred, therefore $a(j) \leq f(j)$ and $b(j) \leq f(j)$.

In figure 9 an HSDF model is shown of the arbitration performed inside the router. In the next section it will be proven that if the j-th token does not arrive later at observation point a in the implementation than at observation point $\hat{a}$ in the HSDF model that then it will also not arrive later at observation point b in the implementation than at point $\hat{b}$ in the HSDF model. In other words it will be proven that if equation 4 holds that this implies that equation 5 holds. In [6] it is proven that in this case tokens in the HSDF model of a system, which includes the HSDF

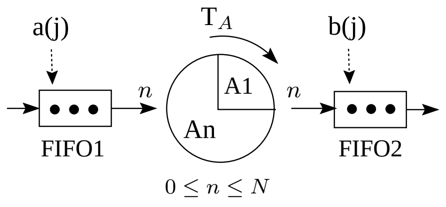

Fig. 8. Abstract model of a TDMA arbiter which allows the transfer of maximally N tokens during the interval T_{A1} per period T_A.

model of an arbiter, do not arrive later at observation points than tokens in the implementation.

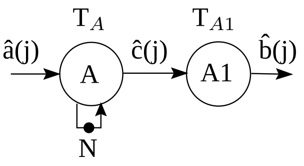

Fig. 9. HSDF model of the TDMA arbiter which allows the transfer of maximally N tokens during the interval T_{A1} per period T_A.

$$a(j) \leq \hat{a}(j), \quad j \geq 0 \tag{4}$$

$$b(j) \leq \hat{b}(j), \quad j \geq 0 \tag{5}$$

The implementation aware HSDF model of the multiprocessor system in figure 10 includes the HSDF model of the arbitration performed inside the router. The N1 initial tokens on the edge from actor A1 to actor P model that FIFO1 has a capacity of N1 tokens and that processor *proc1* will be stalled if FIFO1 is full. The N2 initial tokens on the edge from actor C to actor A model that FIFO2 has a capacity of N2 tokens and that the router does not transfer data if FIFO2 is full. The minimum throughput of the system in figure 7 is equal to the inverse of the MCM of the HSDF graph in figure 10.

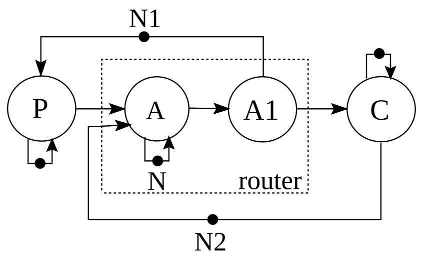

Fig. 10. Implementation aware HSDF model which includes an HSDF model of the TDMA arbitration performed inside the router.

VII. HSDF model of a TDMA arbiter

In this section it is proven that if equation 4 holds that this implies that equation 5 holds for the TDMA arbiter in figure 8 and the HSDF graph in figure 9.

It is given that FIFO1 in figure 8 is initially empty. Therefore, the arrival of the first N tokens ($0 \leq j \leq N-1$) in FIFO2 satisfies inequality 6.

$$f(j) \leq a(j) + T_A + T_{A1} \tag{6}$$

Given that initially N tokens are on the selfedge of actor A in figure 9, it will be the case that the arrival time $\hat{b}(j)$ of first N tokens is according to equation 7.

$$\hat{b}(j) = \hat{a}(j) + T_A + T_{A1} \tag{7}$$

From equation 4, equation 6 and equation 7 it follows that equation 8 holds.

$$f(j) \leq \hat{b}(j), \quad 0 \leq j \leq N-1 \tag{8}$$

Now we want to establish our inductive step by showing how the truth of our induction hypothesis in equation 9 forces us to accept the truth of $f(j+N) \leq \hat{b}(j+N)$ for $j \geq 0$.

$$f(j) \leq \hat{b}(j) \tag{9}$$

For the implementation and $j \geq 0$ the following equations hold in which the intermediate variables tx and ty are defined:

$$tx = a(j+N) + T_A + T_{A1} \tag{10}$$

$$ty = f(j) + T_{An} + T_{A1} = f(j) + T_A \tag{11}$$

$$f(j+N) \leq max(tx, ty) \tag{12}$$

Equation 12 holds because two situations can occur. If $a(j+N) > f(j)$ then token $j+N$ has arrived after the execution of actor A1 is finished (see figure 11). In this case there are less than N tokens in FIFO1 at the moment that token $j+N$ arrives in FIFO1. After arrival of token $j+N$ it will take maximally $T_A + T_{A1}$ before this token departs from FIFO1 because up to N tokens can be transferred per rotation of the time-wheel.

If $a(j+N) \leq f(j)$ then token $j+N$ has arrived in FIFO1 before the execution of actor A1 is finished (see figure 12). After the execution of actor A1, where the j-th token has transferred, it takes maximally $T_{An}+T_{A1}$ before the execution of actor A1, where the $(j+N)$-th token has transferred, is finished.

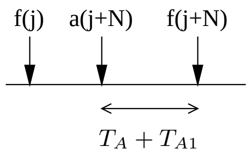

Fig. 11. Arrival of token j+N in FIFO1 after the execution of actor A1, where the j-th token has transferred, is finished.

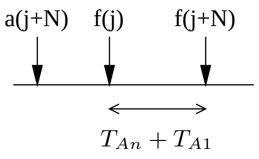

Fig. 12. Arrival of token j+N in FIFO1 before the the execution of actor A1, where the j-th token has transferred, is finished.

For the SDF model in figure 9 the following equations hold for $j \geq 0$.

$$\hat{c}(j+N) = max(\hat{a}(j+N), \hat{b}(j) - T_{A1}) + T_A \tag{13}$$

$$\hat{b}(j+N) = \hat{c}(j+N) + T_{A1} \tag{14}$$

Equation 13 substituted in equation 14 results in:

$$\hat{b}(j+N) = max(\hat{a}(j+N), \hat{b}(j) - T_{A1}) + T_A + T_{A1} \tag{15}$$

Therefore, for $j \geq 0$ the following equations hold in which the intermediate variables tp and tq are defined:

$$tp = \hat{a}(j+N) + T_A + T_{A1} \tag{16}$$

$$tq = \hat{b}(j) + T_A \tag{17}$$

$$\hat{b}(j+N) = max(tp, tq) \tag{18}$$

It follows from equation 4 that $tx \leq tp$ and from our induction hypothesis in equation 9 it follows that $ty \leq tq$. Because equation 19 holds we have proven that equation 20 holds for $j \geq 0$:

$$tx \leq tp \wedge ty \leq tq \Rightarrow max(tx, ty) \leq max(tp, tq) \tag{19}$$

$$f(j+N) \leq \hat{b}(j+N) \tag{20}$$

And given that equation 8 holds we have therefore proven that if equation 4 holds that this implies that equation 5 holds because $b(j) \leq f(j)$.

VIII. MULTIPLE ROUTERS

In section VI a network was considered with only one router. However large scale multiprocessor systems will contain networks with a number of routers. Buffering of data inside the network is undesirable and can be prevented by synchronization of the switches inside the routers and by taking care that data is only transferred through the network if it can be stored at the output of the network. In this case it is necessary to inform the sending router how much space there is in the output buffer. This information is send in the opposite direction via a second guaranteed throughput connection. In order to derive the throughput of the system this second connection must be taken into account in the HSDF model of the system. In this section such a HSDF model is derived for the multiprocessor system in figure 13 on which the application in figure 3 is executed.

The amount of space that is available in FIFO2 in figure 13 depends on when processor 2 reads data from this FIFO. As soon as data has been read it is necessary to inform router R1 that space has become available in FIFO2 such that router R1 can start to transfer data from FIFO1 to FIFO2. A straightforward solution would be the transfer of a credit token to router R1 for each word read from FIFO2. However, in order to save bandwidth in the network the amount of space that is available in FIFO2 is sampled by the router R2 with a fixed period and send as one data word to router R1.

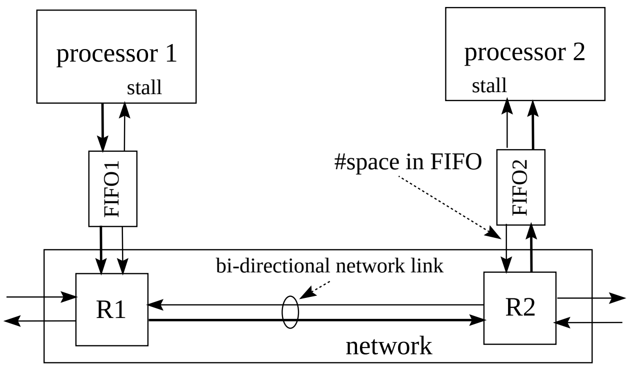

Fig. 13. Network with multiple routers.

In figure 14 an implementation aware HSDF model is shown which includes a model of the network. Actor C send in this model a credit token to actor R1 to indicate that a data token has been consumed from FIFO2 by this actor. This credit token arrives at actor R after T_{Lc} which reflects that router R1 in the implementation receives periodically a credit token in which a value is stored which indicates how much space has become available since the last credit token has been sent.

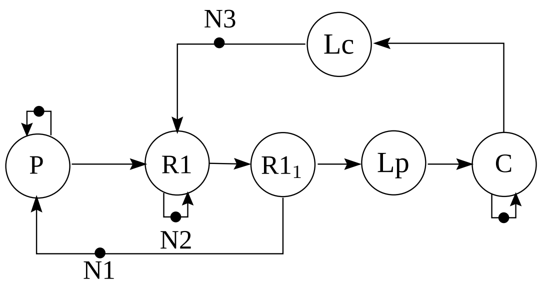

Fig. 14. HSDF model of a connection in the network with multiple routers.

IX. MULTIPROCESSOR TEMPLATE

An assumption made in the previous section was that the application was described as an HSDF and that the tokens had a size of one word. In this section, the multiprocessor system is extended such that it can be used to execute an application which is described by arbitrary SDF graph in which there are no restriction on the size of the tokens. The throughput of this system is derived with an SDF model. The type of applications for which this multiprocessor system is intended, is described in more detail in [7].

The multiprocessor template in figure 15 includes besides a network also a local data memory (DMEM) for each processor and a Communication Assist (CA) which copies data between the FIFOs in the Network Interface (NI) and the data memory. The input and output data of the actors that are executed on a processor are stored in logical FIFO buffers in the local memory of the processor. These logical FIFO buffers can be implemented in software with a Cheap [8] like protocol. The data can be randomly written and read by the processor within the space reserved for a token in these logical FIFOs.

In order to derive the WCET of an actor it is necessary to know in advance the number of cycles that the processor will be stalled during the execution of the actor. Stalling of the processor due to absence of data is prevented by checking that all input tokens for the actor are present in the local memory as well as sufficient output space is present in the local memory before an actor fires. The number of stall cycles caused by each load and store operation that is executed on the processor will be maximally one cycle by taking care that the bus is granted to the processor at least once every other clock cycle. Given maximally one stall cycle for each memory access it is possible to derive the WCET of an actor with static program analysis techniques [9].

Actors executed on a processor access the logical FIFOs in the memory instead of the FIFOs in the network interface in order to allow the production and consumption of tokens that are larger than the capacity of the FIFOs in the

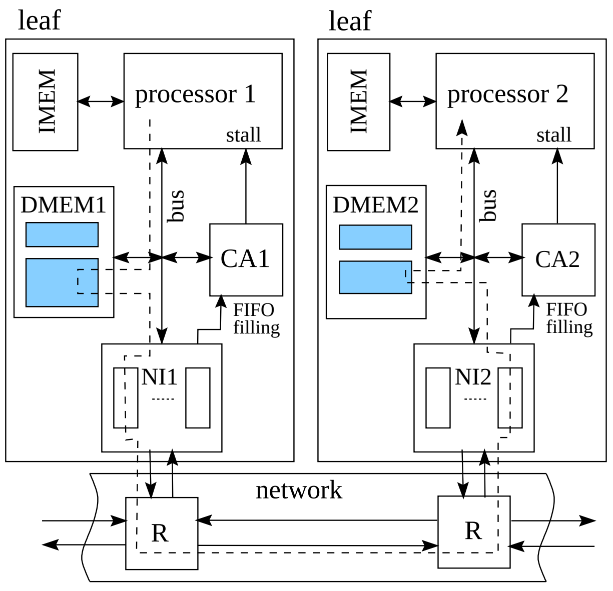

Fig. 15. Multiprocessor template suitable for the execution SDF graphs. The dashed line indicate the path the data flows from an actor that produces the data to an actor that consumes this data.

network interface. These tokens are typically larger than the capacity of these FIFOs (32 words) because these FIFOs must have a small physical size in order to handle the high accesses frequency of the routers in the network.

The communication assists are responsible for the transfer of data between the FIFOs in the network interface and the data memory. A communication assist transfers up to N tokens during a predefined interval just like a router in the network does. It can therefore be represented by the HSDF model that is shown in figure 9.

The SDF model from which the throughput of the multiprocessor system can be derived is obtained by replacing each edge in the implementation unaware SDF graph by the model of a guaranteed throughput connection. Such an SDF model is shown in figure 16. The model of a guaranteed throughput connection is surrounded by a dashed box in this figure. This model includes the arbitration performed by the CAs. In this model it is assumed that credit tokens in the network are handled by the network interfaces instead of by the routers. Each actor inside the dashed box consumes and produces one token per execution. The number of tokens produced or consumed by actor P and C is respectively equal to the values of the parameters n8 and n9. The values of the parameters N1, N3, N5, and N7 are respectively equal to the capacities of the FIFOs in DMEM1, NI1, NI2, and DMEM2. The values of the parameters N2, N4 and N6 represent the maximum number tokens transferred per period by respectively CA1, NI1 and CA2.

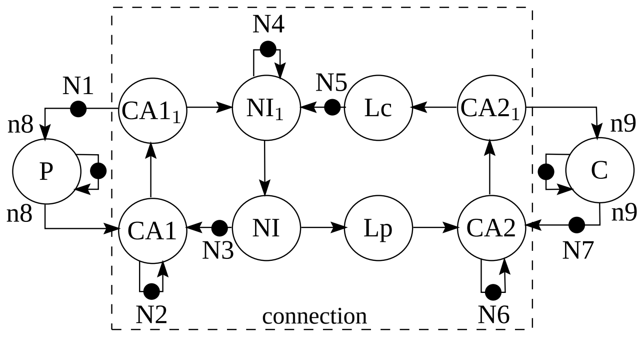

Fig. 16. SDF model of a guaranteed throughput connection.

X. Processors sharing

In the previous section an example was presented in which only one actor was executed on each processor. In this section the execution of multiple actors on a processor is considered. The execution of multiple actors on one processor requires a local scheduler. This local scheduler can be seen as an arbiter whose task is to grant one actor out of a set of actors to start its execution on the processor. Before an actor executes, it first checks whether there are sufficient tokens on each of its inputs and there is sufficient amount of space available on each of its outputs. The actor returns immediately if it detects that there are an insufficient number of tokens or there is an insufficient amount of space available.

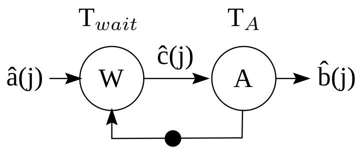

Fig. 17. SDF model in case round-robin arbitration is applied.

In [6] it has been shown that the arbitration of a processor can be modeled in an SDF graph. This is possible if so-called predictable arbitration policies are applied for which a waiting time T_{wait} and a use time T_{use} can be defined. Examples of predictable arbitration policies are Rate Monotonic, Earliest Deadline First, TDMA and round-robin. If for example round-robin arbitration is applied then the effects of the arbitration can be modeled by replacing each actor in the implementation unaware SDF graph with the SDF graph that is shown in figure 17.

In figure 18 an implementation aware SDF graph is shown for which it is assumed that actor A1 and A3 are executed on processor 1. The implementation aware SDF graph for this application is shown in figure 19. Each box in this figure represents an HSDF model of a guaranteed throughput connection. The WCET of actor W1 in this

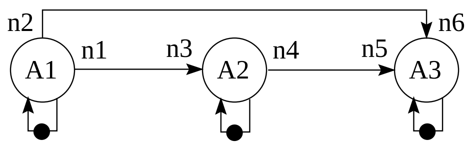

Fig. 18. Implementation unaware SDF model.

model is equal to the WCET of actor A3 and the WCET of actor W3 is equal to WCET of actor A1 because it was assumed that actor A1 and A3 share processor 1. Actor A2 in figure 18 is not replaced in figure 19 by the SDF graph shown in figure 17 because this actor is the only actor that is executed on processor 2.

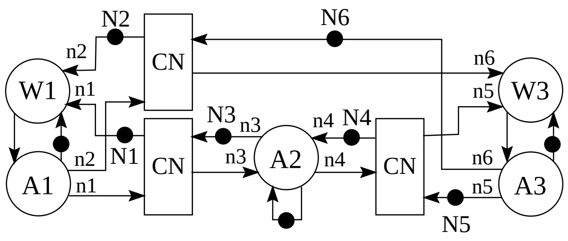

Fig. 19. Implementation aware SDF model. Each box in this figure represents an HSDF model of a connection.

The worst-case arrival times of tokens in the system can be found by simulating the SDF graph in figure 19 in a self-timed manner. The MCM of the system can be determined after the SDF graph in figure 19 is transformed into an HSDF graph. This MCM is equal to the inverse of the minimal throughput of the system.

XI. Conclusion

In this paper an embedded multiprocessor system is proposed which includes a packet switched network that supports communication between processors with a predefined guaranteed bandwidth and a maximum latency. A synchronous data flow model of such a network is presented which is used to derived the worst-case temporal behavior of an application that is executed on this multiprocessor system. It is proven that this synchronous data flow model captures the worst-case behavior of the TDMA arbiter in the network despite that this arbiter allows a variable number of tokens to be transferred in one time slice. Also the local scheduling performed on the processors can be captured in the same SDF graph. The minimal throughput and the worst-case arrival times of tokens in the system can be derived given this SDF graph.

References

[1] F. Bacelli, G. Cohen, G.J. Olsder, and J-P. Quadrat, *Synchronization and Linearity*, John Wiley & Sons, Inc., 1992.

[2] H. Zhang, "Service disciplines for guaranteed performance services in packet-switching networks", *Proceedings of the IEEE*, October 1995.

[3] S. Sriram and S.S. Bhattacharyya, *Embedded Multiprocessors: Scheduling and Synchronization*, Marcel Dekker, Inc, 2000.

[4] E.L. Lawler, *Combinatorial optimization: Networks and Matroids*, Holt, Reinhart, and Winston, New York, NY, USA, 1976.

[5] J. Cochet-Terrasson, G. Cohen, S. Gaubert, M. McGettrick, and J.-P. Quadrat, "Numerical computation of spectral elements in max-plus algebra", in *Proc. IFAC Conf. on Syst. Structure and Control*, 1998.

[6] Marco Bekooij and Jef van Meerbergen, "Timing analysis of data driven hard-RT multiprocessor systems", Not published yet.

[7] M. Bekooij, O. Moreira, P. Poplavko, B. Mesman, M. Pastrnak, and J. van Meerbergen, "Predictable embedded multiprocessor system design", Accepted for: Proceeding of the SCOPES workshop, September 2004.

[8] O.P. Gangwal, A. Nieuwland, and P. Lippens, "A scalable and flexible data synchronization scheme for embedded hw-sw shared-memory systems", International Symposium on System Synthesis, 2001, pp. 1–6, ACM.

[9] Y-T. S. Li and S. Malik, *Performance analysis of real-time embedded software*, ISBN 0-7923-8382-6, Kluwer academic publishers, 1999.