Universal Journal of Electrical and Electronic Engineering 7(4): 242-261, 2020
DOI: 10.13189/ujeee.2020.070403
http://www.hrpub.org

Translating Hierarchical Simulink Applications to Real-time multi-core Execution

Asma Rebaya[1], **Karol Desnos**[2,*], **Salem Hasnaoui**[3]

[1]SYSCOM Research Laboratory National School of Engineering of Tunis University of Tunis El Manar, Tunisia
[2]IETR, INSA Rennes CNRS UMR 6164, UEB Rennes, France
[3]SYSCOM Research Laboratory, National School of Engineering of Bizerte Carthage University, Tunisia

Received May 26, 2020 ; Revised July 6, 2020; Accepted July 29, 2020

Abstract Matlab & Simulink i s i s widely used as a defacto standard t o design i ndustrial applications, video coding & decoding, and signal processing applica-tions. However, with t h e spectacular i ncrease i n t h e num-ber of the cores available i n hardware platforms over t h e s e l a s t years, passing f r o m Simulink t o multi-core execution becomes more and more complex. I n t h i s context, several researches are done t o t a k e benefit f r o m t h e high degree of parallelism and t o perform multi-core programming of Simulink applications.

In this paper, we present an automated method for transforming hierarchical Simulink applications to embedded parallel software implementation. Our method consists of using IBSDF (Interfaced based Synchronous Dataflow) as an intermediate representation to extract parallelism. Moreover, our approach permits preserving synchronous semantics and hierarchical behavior of the Simulink model. The model-based approach makes it possible to verify the key properties of the system at compile-time, such as deadlock freeness and memory boundedness. The method has been implemented as an extension of the rapid prototyping tool named Preesm. Experiments show that our proposal gives, as a transformation result, a schedulable IBSDF graph equivalent in size to the Simulink model and allows better multi-core implementation performance than Matlab&Simulink sequential execution.

Keywords Matlab & Simulink models, Hierarchy, parallelism, multi-core architecture, rapid prototyping, Code generation.

1 Introduction

In recent years, the number of cores available in hardware platforms has increased dramatically, from tens of cores to hundreds of heterogeneous processing elements. Concurrently, the development in digital communications, telecommunications, digital signal processing and coding/decoding videos becomes increasingly complex, thus requiring more computational power. Then, there is a need for modeling software applications with a high-level and implementation independent model that can be translated efficiently into high performance implementation on modern parallel architectures. These reasons motivate the transition from imperative programming languages, which are intrinsically sequential, to parallel models of computation. This transition offers two main advantages for the software/hardware applications programming: parallelism and performance improvement.

To take benefit of this approach and exploit the data parallelism, SDF (Synchronous Dataflow)[14] models has proven to be a well suited representation for programming multi-core architectures. Indeed, thanks to its semantics, the SDF MoC (model of computation) provides a practical mean to decompose applications into coarse grain computational entities: the actors. Mapping and scheduling these actors on available processing elements makes it possible to optimize diverse performance criteria of the application, such as throughput, speedup and latency. Moreover, each SDF actor is described by a host program. In our work, host programs are written in C language which will contribute to the generation of compatible C codes targeting multi-core architectures.

Currently, Simulink, a software package from Mathworks, is widely used as defacto standard to design, simulate and validate industrial applications in many domains, such as digital communication, digital signal processing, and image processing. However, Simulink models are difficult to be implemented into multi-core platforms. This is due to the fact that passing from a Simulink application to multi-core platform must preserve model semantic and consistency, besides to the determinism in data exchanged between different Simulink blocks. Additionally, the implementation must meet parallel hardware constraints and ensure that the behavior of the generated code is conformed to the Simulink model behavior. Further, up to now, most references in this context do not gain advantages from the hierarchical behavior of Simulink model to extract parallelism and optimize code generation. These references focus only on the transformation of atomic blocks and flattened systems.

The question left is how to overcome lacks cited above, facilitate and perform multi-core programming of hierarchical Simulink applications. This question leads to an idea of extending rapid prototyping tool with code genera-

tion capabilities to support Simulink applications. In fact, rapid prototyping tools allow to model hardware/software applications and give early decisions that contribute to improve and perform applications deployment into multi-core platforms. We choose to extend an open source tool called Preesm (the Parallel and Real-time Embedded Executives Scheduling Method)[21] thanks to its ability to prototype efficient multi-core applications starting from SDF representation. Therefore, we allow designers to go from a Simulink model to an efficient hardware implementation. To concretize the idea above, we put forward an efficient solution to automate Simulink models transformation and deployment over multi-core architecture.

In this paper, we propose a translation approach and a rapid prototyping tool extension to support multi-core programming of Simulink applications. The method consists of automatically translating an application designed with Simulink into an extended version of the SDF MoC called IBSDF (Interfaced based Synchronous Dataflow) [15] while keeping models hierarchical behavior. The choice of IBSDF MoC as an intermediate representation is based on the fact that it is a specific class of SDF MoC characterized by a hierarchy mechanism. This mechanism enables the description of the internal behavior of nodes with SDF subgraphs and ensures deadlock freeness between levels thanks to interfaces mechanism. Moreover, using IBSDF MoC allows us to benefit from recent researches such as [16] which exploit hierarchical behavior of IBSDF MoC to optimize code generation process and enhance performance on multi-core architectures.

The aim of this work is to provide an effcient solution to automatize Simulink programs deployment over multi-core architectures. The two main contributions of this work are as follows:First, we translate hierarchical Simulink models into IBSDF graphs in such way we preserve hierarchy and synchronous semantics of both models. During the translation we respect deadlock-freeness and consistency constraints in order to obtain a schedulable IBSDF graph. Secondly, we implement our proposal as a plugin extension into Preesm work-flow to generate optimal C codes compatible for multi-cores platform.

To achieve this, we perform, as first step, a mathematical study to demonstrate that hierarchical Simulink systems can be translated into schedulable IBSDF graphs. To do t this, we define, first, multi-levels precedence, concistency and deadlock-freeness constraints which must be taken into account in the translation process. Then, we differntiate three block communication cases: direct communication, delayed communication and hybrid communication. Considering these three cases, we propose our translation theorems. Each proposed theorem was followed with a proof. For each communication case, we illustrate the translation process with simple Simulink models. The objective of the mathematical study is to demonstrate that during the translation process, consistency and deadlock freeness were conserved and the resulting hierarchical graph is a schedulable graph. To the best of our knowledge, our paper is the first that gives a detailed study about conserving hierarchy during the translation process of hierarchical Simulink models. All previous works focus only on the synchronous behavior of Simulink models and do not consider the internal hierarchical behavior which is very important to extrat parallelism and allows better multi-core implementation performance.

Our proposed translation process was implemented into three main sub-tasks: Simulink parser, translator and IBSDF generator integrated in the extended Preesm work-flow. To the best of our knowledge, this work is the first that proposes a whole work-flow starting from Hierarchical Simulink models to parallel C code generation for multi-cores architecture. Indeed, Previous works, such as [1], [2] [3] and [4], focused only on the translation process and did not integrate their results into code generation tools to show the effictiveness of their approaches.

In this work, we consider only multi-rate and discrete-time Simulink models with multiple sampling times blocks. Blocks can be hierarchical or atomic. This means that blocks with dynamic behavior, such as enabled-subsystem and triggered-subsystem, and continuous-time part of Simulink models are not considered in this work. Throughout the translation to IBSDF, we ensure that the resulted graph has the same Simulink model behavior and the same size. This means that semantics of both models were conserved during the execution of the whole extended work-flow starting from Simulink model to multi-core deployment.

The remainder of this paper is structured as follows: In Section 2, we give a background of models of computation used in the presented work. Section 3 gives a study of deadlock freeness and consistency constraints and details our solution to translate hierarchical Simulink models. In section 4, we present Preesm tool support and the overall extended workf-low. Experimental results which are based on the state-of-art telecommunication application "LTE QPSK transmitter" are presented in Section 5. Finally, the last section concludes the paper and underlines the future work.

2 Background

2.1 Synchronous Data-flow and Interface-Based Synchronous Data-flow

Synchronous data-flow SDF, introduced by Lee and Messerschmitt [9, 10], are MoC providing high level design and implementation of embedded programs, which are notably popular for specifying digital signal processing applications.

An SDF graph $G = (A, F, P)$ consists of a set of actors A interconnected by a set of FiFo F carrying data tokens and a set of port P used as anchors for FiFo connection such that:

- $A = (a_1, a_2, a_3, ...)$ is the set of actors that consume and/or produce data tokens.
- $F = (F_1, F_2, F_3, ...)$ is the set of channels carrying data streams.
- $P = (inP, outP)$ is the port set of an actor a_i.
- $inP(a_i) = (inP_1(a_i), inP_2(a_i), inP_3(a_i), ...)$ is the set of input ports of an actor a_i.
- $outP(a_i) = (outP_1(a_i), outP_2(a_i), outP_3(a_i), ...)$ is the set of output ports of an actor (a_i).

- $IN(a_i) = (in_1(a_i), in_2(a_i), in_3(a_i), , ...)$ is the data consumed by the different a_i input ports firing.
- $OUT(a_i) = (out_1(a_i), out_2(a_i), out_3(a_i), ...)$ is of data produced by the different a_i output p
- $d(a_i, a_j)$ is the amount of initial tokens on a FIFO connecting the actors a_i and a_j , we refer to it as a delay. The delay allows to represent data dependencies between successive graph iteration [23] and perform analysis. A graph iteration is the minimal fixed sequence of actor firings that can be repeated indefinitely to execute the graph.

Further, to ease graph development and to allow optimization for the scheduling process, SDF graph may support hierarchical behavior. An extended version of SDF MoC, called IBSDF, adds to the SDF semantics a hierarchy mechanism. This mechanism allows more expressiveness in SDF graph by enabling the specification of the internal behavior of an actor with an encapsulated subgraph. This hierarchy mechanism offers more flexibility to optimize application for the scheduling and generation code process. IBSDF semantics, in addition to SDF semantics, is based on interfaces mechanism allowing a hierarchical graph design. Interfaces mechanism obeys the "compositionality" principle. In fact, interfaces mechanism was proven in [15] to be able to ensure that if a graph is instantiated, its behavior might not be modified by its parent graph. Further, its behavior might not also introduce deadlock in its parent graph.

In addition to the SDF semantics, an IBSDF graph $G = (A, F, P, D, I)$ consists of a set of hierarchical actors HA and a set of interfaces I separating the hierarchical levels where:

- $HA = (Ha_1, Ha_2, Ha_3, ...)$ is the set of hierarchical actors that consume and/or produce data tokens.
- $I = (srcI, snkI)$ is the set of interfaces.
- $srcI$ is the vertex transmitting to the subgraph the amount of tokens received by its corresponding parent actor.
- $snkI$ is the vertex transmitting to the parent actor the amount of tokens produced by its corresponding subgraph.
- $srcI^{data}$ is the amount of data available on its corresponding source interface.
- $snkI^{data}$ is the amount of data available on its corresponding sink interface.

In the following sections, a hierarchical actor will refer to an actor which encapsulates a hierarchy level, an actor will refer to an atomic actor and a sub-graph will refer to the graph embedded into a hierarchical actor. Figure 1 and Figure 2 illustrates SDF and IBSDF graphs, respectively.

2.2 Description of Simulink

Simulink, developed by Math-works, is a wide-spread commercial tool for embedded system simulation and

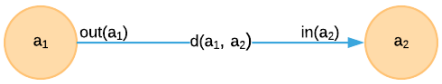

Figure 1. SDF graph representation.

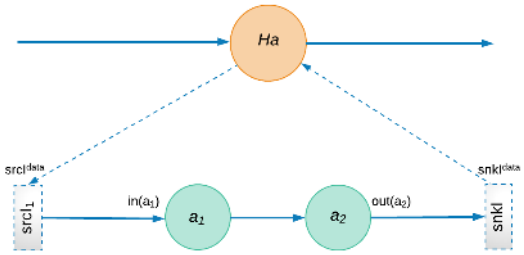

Figure 2. IBSDF graph representattion.

model based design. Thanks to its graphical user interface (GUI) and its rich library of blocks, Simulink users are able to build, simulate and verify a variety of embedded systems such as telecommunication, signal processing, image and video processing application.

Simulink modeling consists of creating a block network connected by lines, representing signals. Blocks ports (inports and outports) represent connection endpoints for signals. There are two types of block parts, data ports, which gives the dataflow of the Simulink model and control ports which produce conditional (enabled, triggered) events for the execution of subsystems. In this work, we consider only the data ports type.

Further, Simulink is a synchronous model with synchronous language based on hierarchical behavior. Indeed, Simulink enables users to group basic blocks in a recursive manner to design more complex diagrams. We refer to the composite diagram as subsystem and the non-composite as atomic block. Each port on the subsystem corresponds to an InPort-Block or an OutPort-Block. A subsystem may contain blocks sampled at different rates; we called as multi-rate subsystem. We distinct two classifications of subsystems: virtual subsystems and non-virtual (functional) subsystems. A virtual subsystem is helpful to graphically organize the Simulink model and increases the design readability, but it does not influence the internal behavior of the hierarchy. While the non-virtual subsystem is provided to model functionalities and control the internal hierarchy.

Simulink supports simulation for discrete (sampled data) and continuous systems. For the discrete-time blocks, the sample time is defined as the time between two consecutive instants when a block executes. Networks can include discrete-time blocks operating at different rates (so-called multi-rate systems) where each block is associated with a different sample time period. Regarding to the continuous time blocks, the sample time is obtained by simulating ordinary differential equations.

We differentiate two simulation options: multi-tasking which assign priority to each block and single-tasking which does not assign priority to blocks and respect

the simulation time execution semantics. Both multi-tasking and single-tasking support different communication mechanisms: direct communication mechanism and delayed communication mechanism. In direct communication mechanism, a block uses the data produced by the block connected to its input at the same time step. Otherwise, in the delayed communication, a block does not use the data produced by the block connected to its input at the same time step. A combination of both mechanisms, so-called hybrid communication mechanism, is used when the data is transferred from low priority block to high priority block and block periods are different. Indeed, when blocks composing multi-rate system are not executed at the same time, direct data exchanging mechanism is adopted because the lower priority block driving the input is fired after the higher priority block. As well, when blocks composing multi-rate system are executed at the same time, delayed data exchanging mechanism is adopted because the lower priority block driving the input is fired after the higher priority block. These communication rules are applied for blocks from the same level of hierarchy and for blocks from different level of hierarchy. Then, additionally to the hierarchical behavior, relation precedence exists between levels and obeys the same communication rules.

Through the next sections we use the following terminology:

- $B = (b_1, b_2, b_3, ...)$ is the set of atomic blocks or/and atomic subsystems.
- $S = (S_1, S_2, S_3, ...)$ is the set of composed subsystems.
- $InB = (inb_1, inb_2, inb_3, ...)$ is the set of the input ports of a composed subsystem S_i.
- $OutB = (outb_1, outb_2, outb_3, ...)$ is the set of the output ports of a composed subsytem S_i.
- T_{S_i} and T_{b_i} denote the sample time period of a composed subsystem S_i and the sample time period of an atomic block b_i, respectively.

3 Translation principle

In this section, we present our transformation technique of discrete-time Simulink model to its corresponding IBSDF representation. The main aim of this transformation methodology is to obtain a schedulable IBSDF where structural properties of the Simulink model are preserved. To reach this goal, the translation must be performed in a way that ensures the consistency and the deadlock freeness of the obtained IBSDF. For this, we have to determine data tokens consumed/produced by each actor and the delay available on each FiFo while respecting precedence, consistency and deadlock freeness constraints. In the following subsections we give an overview about these constraints before detailing the translation process.

3.1 Deadlock freeness and consistency constraints

We consider a hierarchical actor Ha_1 encapsulating n atomic actors $a_1, a_2, ..., a_n$ where a_1 is the consumer actor (consumes tokens produced by the hierarchical actor) and a_n is the producer actor (produces tikens to the hierarchical actor). We note $srcI$ the source interface of Ha_1 that transfers tokens to its sub-graph and $inP(a_1)$ is the target port of the first consumer sub-actor a_1. We also note $snkI$ the sink interface of Ha_1 which consumes tokens produced by the last producer sub-actor a_n and $outP(a_n)$ is the source port of a_n as showed in figure 3.

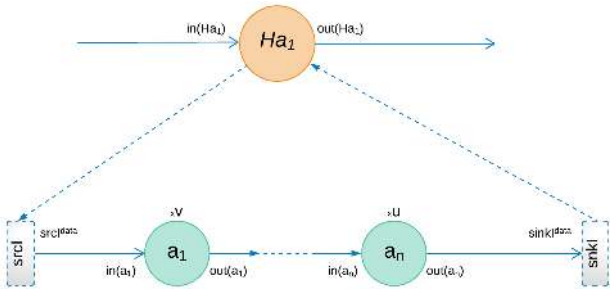

Figure 3. IBSDF example with a hierarchical actor and n atomic actors where a_1 is the consumer actor and a_n is the producer actor.

As mentioned in section **??**, IBSDF MoC introduces interface elements to model the hierarchy behavior of an SDF graph and insulate its level of hierarchy. This hierarchy semantic must obey some constraints, imposed by the IBSDF model, in order to ensure deadlock freeness and consistency at every level of graph hierarchy:

- Deadlock freeness constraints: during a subgraph iteration, source and sink interfaces must stay write-locked and read-locked, respectively. Meaning that the internal behavior is independent of any external actor during an iteration. Further, if a consumer sub-actor needs an amount of tokens greater than the amount available in the source interface, this latest will behave like a ring buffer. Regarding the sink interface, if the producer sub-actor produces an amount of tokens greater than required, it will behave like a circular buffer. Hence, it will forward only the number of tokens produced during the subgraph execution and required by the hierarchical actor.
- Consistency constraints: to ensure the consistency of the internal SDF subgraph, the subgraph must consume all the tokens made available by the source interface during an iteration. Symmetrically, all tokens required by the sink interface must be produced by the sub-graph during an iteration.

Lemma 1 highlights the conditions required to ensure deadlock freeness and consistency in every level of the hierarchy.

Lemma 3.1 *Let us consider a hierarchical actor Ha with source interface $srcI$, sink interface $sinkI$ and encapsulating n atomic actors $a_1, a_2, ..., a_n$ where a_1 is the consumer actor and a_n is the producer actor. The internal SDF of the hierarchical actor Ha is consistent and deadlock free if source and sink interfaces obey these conditions at each iteration:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases} \tag{1}$$

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases} \tag{2}$$

Where:

- $x \in> 1$ *and* $\alpha \in [0..1]$ *are positive constants representing duplication numbers of the rate of tokens available in the source interface within two different cases.*
- $\gamma > 1$ *and* $\beta \in [0..1]$ *representing duplication numbers of the rate of tokens available in the sink interface within two different cases.*
- $v \in N^*$ *is the execution repetition number of* a_1.
- $u \in N^*$ *is the execution repetition number of* a_n.

Proof: Deadlock freeness constraints can be written as: If $srcI^{data} \leq v \cdot in(a_1)$ then

$$v \cdot in(a_1) = x \cdot srcI^{data}. \tag{3}$$

If $sinkI^{data} \leq u \cdot out(a_n)$ then

$$\gamma \cdot sinkI^{data} = u \cdot out(a_n). \tag{4}$$

Consistency constraints can be written as:
If $srcI^{data} > v \cdot in(a_1)$then

$$\alpha \cdot srcI^{data} = v \cdot in(a_1). \tag{5}$$

If $sinkI^{data} > u \cdot out(a_n)$ then

$$\beta \cdot sinkI^{data} = u \cdot out(a_n). \tag{6}$$

Equations (3) and (5) give the source interface conditions. Similarly, equations (4) and (6) give the sink interface conditions. Figure 4 helps to illustrate Lemma 1 by an example. In this case, $in(Ha_1) = 2$ is the amount of

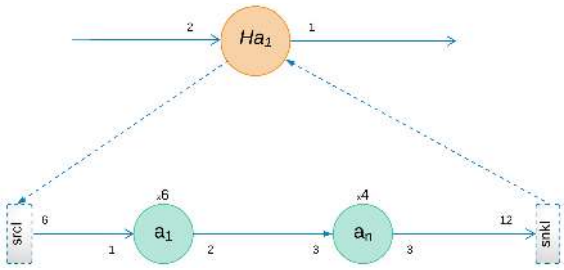

Figure 4. Example of IBSDF graph with a hierarchical actor Ha_1 containing two atomic actors a_1 and a_2.

tokens produced to the source interface $srcI$, $in(a_1) = 1$ and its execution repetition number v is equal to 6. We have, consequently, $srcI^{data} = 2 \leq V \cdot in(a_1) = 6$. According to Lemma 13.1, the number of data tokens available in the source interface must be duplicated x time. The duplication number is determined by applying equation (1). Then, x is equal to 3. Likewise, $out(Ha_1) = 1$ is the amount of data tokens required to be produced by the sink interface $sinkI$. Although, $out(a_n) = 3$ and its execution repetition number u is equal to 4. We have, consequently, $sinkI^{data} = 1 \leq u \cdot out(a_n) = 12$. Based in Lemma 1 3.1, the amount of tokens available in the sink interface must be duplicated γ time in such way that $sinkI^{data} = \frac{u}{\gamma} out(a_n)$. Then, we have γ equal to 12.

3.2 Precedence constraints

3.2.1 One-level precedence constraints

In [18], Marchetti gives a detailed study about couples firings in SDF graphs, considering only flattened graphs (no hierarchy). Based on the fact that a schedule is feasible if the number of tokens is positive in every FiFo of the graph, author demonstrates the existence of precedence constraint between two executions of connected actors. Let us consider a couple of actors a_i and a_j linked by a FiFo F, $d(a_i, a_j)$ is the initial amount of tokens available in F, $[v]$ is the v^{th} execution of a_i, $[u]$ is the u^{th} execution of a_j, $out(a_i)$ is the amount produced by a_i and $in(a_j)$ is the amount consumed by a_j.

A precedence relationship exists between a_i and a_j if firings obey the following two conditions:

- Condition (1): $[u]$ can start after $[v]$.
- Condition (2): $[u-1]$ can start before $[v]$ while $[u]$ cannot.

Lemma 2 characterizes the precedence constraint between the v^{th} execution of a_i and u^{th} execution of a_j.

Lemma 3.2 *A precedence constraint exists between the* v^{th} *execution of* a_i *and* u^{th} *execution of* a_j *iff:*

$$\begin{aligned} &out(a_i) > d(a_i, a_j) + out(a_i) \cdot v - in(a_j) \cdot u \\ &\geq max(out(a_i) - in(a_j), 0). \end{aligned} \tag{7}$$

Proof: A precedence relation is modeled between the v^{th} execution of a_i and u^{th} execution of a_j if condition (1) and condition (2) are fulfilled:
Condition (1) $\iff \mathrm{d}(\mathrm{a_i}, a_j) + out(a_i) \cdot v - in(a_j) \cdot u \geq 0$.
Condition (2) $\iff \mathrm{out}(\mathrm{a_i}) > d(a_i, a_j) + out(a_i) \cdot (v-1) - in(a_j) \cdot (u-1) \geq 0$.
Combining these resulting inequalities, we obtain inequality (7).

3.2.2 Multi-levels precedence constraints

We consider a hierarchical actor Ha containing n atomic actors $a_1, a_2, \ldots, a_n$. We denote the actor a_1 as the consumer sub-actor which consumes the amount of tokens $in(a_1)$, the actor a_n as the sub-actor which produced the amount of tokens $out(a_n)$, $srcI^{data}$ is the amount of data available on the source interface linking the hierarchical graph Ha and the consumer sub-actor a_1 and $d(Ha, a_1)$ is the initial amount of tokens in the FiFo linking the source interface and the consumer sub-actor. We also denote $snkI^{data}$ as the amount of data available on the sink interface linking the hierarchical actor Ha and the producer sub-actor $d(a_n, Ha)$ as the initial amount of tokens in the FiFo linking the sink interface and the producer sub-actor.
Building on the precedence constraints between two firings of the same level and the hierarchy dependency, we define the precedence constraints between an hierarchical actor and its sub-graph.

A precedence relationship exists between an hierarchical actor and its sub-graph if firings obey the following conditions:

- Condition (1): $[w \cdot \#v + v]$ can start after $[w]$.

- Condition (2): $[w \cdot \#v + v - 1]$ can start before $[w]$.
- Condition (3): $[w]$ can start after $[w][w \cdot \#v + v]$.
- Condition (4): $[w - 1]$can start before $[w \cdot \#u + u]$.

Where:

- $[w]$ is the w^{th} execution of Ha.
- $[v]$ is the v^{th} execution of a_1.
- #v is the execution number of a_1 up to $[v]$.
- $[u]$ is the u^{th} execution of a_n.
- #u is the execution number of a_n up to $[u]$.

We can now determine initial amounts of tokens modeling the precedence relation between firings of an hierarchical actor and its subgraph.

Lemma 3.3 *A precedence relation between an hierarchical actor and its sub-graph if:*

$$srcI^{data} > d(Ha, a_1) + srcI^{data} \cdot w - in(a_1) \cdot (w \cdot \#v + v) \geq max(srcI^{data} - in(a_1), 0). \quad (8)$$

$$out(a_n) > d(a_n, Ha) + out(a_n) \cdot (w \cdot \#u + u) - snkI^{data} \cdot w \geq max(out(a_n) - snkI^{data}, 0). \quad (9)$$

Proof: A precedence relation is modeled between the w^{th} execution of Ha and v^{th} execution of a_1 if condition (1)and condition (2) presented in Defintion 1 are fulfilled:
Condition (1) $\iff d(Ha, a_1) + srcI^{data} \cdot w - in(a_1) \cdot (w \cdot \#v + v) \geq 0$.
Condition (2) $\iff srcI^{data} > d(Ha, a_1) + srcI^{data} \cdot (w - 1) - in(a_1) \cdot (w \cdot \#v + v - 1) \geq 0$.

Then, when combining these two inequalities, we obtain inequality (8).
A precedence relation is modeled between the w^{th} execution of Ha and u^{th} execution of a_n if condition (3)and condition (4) presented in Defintion 1 are fulfilled:
Condition (3) $\iff d(a_n, Ha) + out(a_n) \cdot (w \cdot \#u + u) - snkI^{data} \cdot w \geq 0$.
Condition (4) $\iff out(a_n) > d(a_n, Ha) + out(a_n) \cdot (w \cdot \#u + u - 1) - snkI^{data} \cdot (w - 1) \geq 0$.

Then, when combining these two inequalities, we obtain inequality (9).

3.3 Translation Process

3.3.1 One level blocks translation

To illustrate one level blocks translation, we consider a Simulink system S containing three atomic Blocks B_{i-1}, B_i and B_{i+1}.

Atomic subsystems and basic blocks (such sum blocks, constant blocks,...) are translated into atomic actors in the IBSDF graph. Each resulting atomic actor a_i is named with the corresponding name of the Simulink block B_i. Simulink Blocks sample times T_{B_i} are recuperated when simulating the Simulink model.

Unit delays block is not taken into account during the translation process. It is only used to mark the delayed behavior of the communication between blocks. Further, blocks belonging to one level can be atomic or composed. During the translation of one level of an hierarchical Simulink model , composed blocks behave as atomic blocks.

The input and output blocks connecting blocks of the same level are respectively converted into input and output ports transferring data in the IBSDF graph. We refer to the rules proved in [1] to determine the amount of tokens available in these ports (consumed data rates and produced data rates). We differentiate three communication cases; direct communication case, delayed communication and hybrid communication. In the three cases, the consumed data available in the import of an actor a_i, $in(a_i)$, and the produced data available in a_i out-port, $out(a_i)$, are similarly determined:

- $in(a_i) = \frac{T_{B_i}}{g_{(B_{i-1}, B_i)}}$.
- $out(a_i) = \frac{T_{B_i}}{g_{(B_{i+1}, B_i)}}$.

Input and output blocks are also used to transfer signals between levels of an hierarchical Simulink model. Input and output blocks respectively correspond to source interface $srcI$ and sink interface $sinkI$ in the IBSDF graph. To translate rates available in these interfaces and ensure deadlock freeness and consistency between levels, we based on theorems detailed and proved in the following section.

Lines transferring signals in Simunlink model are converted into FiFo channels connecting actors in the IBSDF graph. Each resulting FiFo is characterized with an initial amount of tokens obtained depending on the communication type:

- Direct communication : $d(a_i, a_{i+1}) = out(a_i) - 1$.
- Delayed communication : $d(a_i, a_{i+1}) = in(a_{i+1}) + out(a_i) - 1$.
- Hybrid communication : $d(a_i, a_{i+1}) = out(a_i)$.

3.3.2 Hierarchical subsystems translation

We consider S_1 a composed subsystem with sample time T_{S_1} containing a set of atomic blocks $B_1, B_2, \cdots, B_n$. A sample time T_{B_i} is associated to each block B_i. Two subsystems S_2 and S_3 are connected to S_1 with sample times T_{S_2} and T_{S_3}.The block B_1 is the sub-consumer block and B_n is the sub-producer block as depicted in figure 5. We note that virtual subsystems are not taken into account during the translation process; they are only used to group blocks.
We pose:

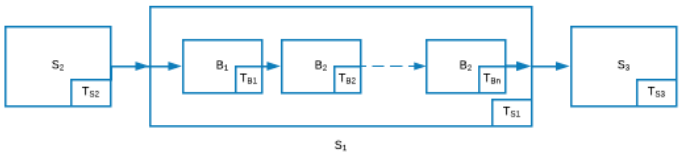

Figure 5. Multi-rate Simulink system in direct communication case.

$g_{(S_1, B_1)}$ the greatest common divisor of T_{S_1} and T_{B_1}.

$g_{(S_1,B_n)}$ the greatest common divisor of T_{S_1} and T_{B_n}.
$g_{(S_1,S_2)}$ the greatest common divisor of T_{S_1} and T_{S_2}.
$g_{(S_3,S_1)}$ the greatest common divisor of T_{S_3} and T_{S_1}.

Modeling levels direct communication

To model levels direct communication, we have to model source/sub-consumer actor communication and sink/sub-producer actor communication.

Source interface/sub-consumer actor direct communication: a direct communication between the source interface and the consumer sub-block is defined through the following hierarchical dependency conditions:

- $[w{\cdot}\#v{+}v]$ fires at the same time or after the beginning time of [w].
- $[w \cdot \#v + v - 1]$ fires strictly before the beginning time of [w].
- $[w \cdot \#v + v]$ fires strictly before the beginning time of [w+1].

Based on these conditions we deduce Lemma 4.

Lemma 3.4 *Let S_1 be a composed subsystem with sample period T_{S_1} containing a set of atomic blocks $B_1, B_2, ..., B_n$ firing in direct communication mode. B_1 represents the sub-consumer atomic block with sample period T_{B_1}. A hierarchical dependency exists between the w^{th} execution of S_1 and v^{th} execution of B_1 if:*

$$T_{S_1} = b \cdot T_{B_1}. \tag{10}$$

Where b is a coefficient superior or equal to 1.

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{S_1} \cdot (w-1) \leq T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{11}$$

$$T_{S_1} \cdot (w-1) > T_{B_1} \cdot (w \cdot \#v + v - 2). \tag{12}$$

$$T_{S_1} \cdot w > T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{13}$$

When we added inequality (12) and inequality (13) we obtain:

$$2T_{S_1} \cdot w - T_{S_1} > 2T_{B_1} \cdot (w \cdot \#v + v - 1) - T_{B_1}.$$

We multiply inequality (11) by -1 and add it with the resulted inequality. We obtain:

$$T_{S_1} > T_{B_1} \frac{w \cdot \#v + v - 2}{w}.$$

Since $\frac{w\cdot\#v+v-2}{w} > 1$, it exists a coefficient $b \geq 1$ such that:

$$T_{S_1} = b \cdot T_{B_1}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-consumer actor in direct communication case, we refer to Theorem 1:

Theorem 3.5 *To ensure deadlock freeness between an hierarchical actor and its sub-consmer actor, IBSDF introduces the source interface concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

where $srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}}$; $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v \geq 1$ *and* $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v < 1$.

Proof: We multiply equality (10) of Lemma 4 by $\frac{v}{g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}}$.
$We obtain: \frac{v}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{v}{g_{(S_1,S_2)}} \cdot \frac{T_{B_1}}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{T_{B_1}}$.

Equality (1) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_2)}}$ by $srcI^{data}$, $\frac{T_{B_1}}{g_{(S_1,B_1)}}$ by $in(a_1)$, x and α by $\frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v$. Where x and α represent duplication numbers of the rate of tokens available in the source interface within two different cases. Hence, the source interface $srcI$ and the sub-consumer actor a_1 obey the deadlock freeness and consistency condition already proved in Lemma 1.

Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-consumer actor.

Theorem 3.6 *In the direct communication case, the initial amount of tokens $d(Ha, a_1)$ of FiFo connecting the hierarchical actor and the sub-consumer actor is given by $in(a_1) - 1$.*

Proof: Inequality (11) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) \leq T_{S_1} - T_{B_1}.$$

Inequality (12) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) > T_{S_1} - 2T_{B_1}.$$

Inequality (13) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \ v + v) > -T_{B_1}.$$

We combine the three inequalities, add T_{B_1} and subtract $g_{(S_1,B_1)}$ from the middle, which results in:

$$T_{S_1} \geq T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) + T_{B_1} - g_{(S_1,B_1)} > max(T_{S_1} - T_{B_1}, 0). \tag{14}$$

We pose $Y = g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}$. When dividing (14) by Y, we obtain:

$$\frac{T_{S_1}}{Y} \geq \frac{T_{S_1}}{Y} w - \frac{T_{B_1}}{Y} \cdot (w \cdot \#v + v) + \frac{T_{B_1}}{Y} - \frac{1}{g_{(S_1,S_2)}} > max(\frac{T_{S_1}}{Y} - \frac{T_{B_1}}{Y}, 0).$$

We multiply the resulting equation by $g_{(S_1,S_2)}$ we obtain:

$$Z \cdot srcI^{data} \geq Z \cdot srcI^{data} \cdot w - in(a_1) \cdot (w \cdot \#v + v) + in(a_1) - 1 > max(Z \cdot srcI^{data} - in(a_1), 0).$$

Where:

- $srcI^{data}$ and $in(a_1)$ are deduced from Theorem 1.
- $Z = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}}$.

Since $Z \cdot srcI$ and $srcI$ are both strictly superior than 0, then, even if we replace $Z \cdot srcI^{data}$ by $srcI^{data}$ this inequality remains true. Hence, referring to inequality (8) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $in(a_1)-1$ by $d(Ha, a_1)$.

Sink interface/sub-producer actor direct communication: a direct communication between the sub-producer actor and the sink interface is defined through the following hierarchical dependency conditions:

- $[w]$ fires at the same time or after the beginning time of $[w \cdot \#u + u]$.
- $[w-1]$ fires strictly before the beginning time of $[w \cdot \#u + u]$.
- $[w]$ fires strictly before the beginning time of $[w \cdot \#u + u + 1]$.

Based on these conditions we deduce the Lemma 5.

Lemma 3.7 *Let S_1 be a composed subsystem with sample period T_{S_1} containing a set of atomic blocks $B_1, B_2, ..., B_n$ firing in direct communication mode. B_n represents the sub-producer atomic block with sample period T_{B_n}. A hierarchical dependency exists between the w^{th} execution of S_1 and u^{th} execution of B_n if:*

$$T_{S_1} = c \cdot T_{B_n}. \tag{15}$$

Where c is in $[0..1[$

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{B_n} \cdot (w \cdot \#u + u - 1) \leq T_{S_1} \cdot (w-1). \tag{16}$$

$$T_{B_n} \cdot (w \cdot \#u + u - 1) > T_{S_1} \cdot (w-2). \tag{17}$$

$$T_{B_n} \cdot (w \cdot \#v + v) > T_{S_1} \cdot (w-1). \tag{18}$$

When combining inequalities (16), (17) and (18) we obtain:

$$min(\frac{w-1}{w \cdot \#u + u - 1}, \frac{w-2}{w \cdot \#u + u}) \cdot T_{S_1} < T_{B_n} \leq \frac{w-1}{w \cdot \#u + u - 1} \cdot T_{S_1}.$$

Since $0 \leq w - 1 < w \cdot \#u + u - 1$, it exists a coefficient $c \in [0..1[$ such that:

$$T_{S_1} = c \cdot T_{B_n}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-producer actor in direct communication case, we refer to Theorem 3.

Theorem 3.8 *To ensure deadlock freeness between an hierarchical actor and its sub-producer actor, IBSDF introduces the sink interface concept such that:*

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq \\ & u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

where $snkI^{data} = \frac{T_{S_1}}{gcd(T_{S_1}, T_{S_3})}$; $out(a_n) = \frac{T_{B_n}}{gcd(T_{S_1}, T_{B_n})}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v \geq 1$ *and* $\beta = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v < 1$.

Proof: We multiply equality (15)of Lemma 5 by $\frac{v}{g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}}$.

We obtain:

$$\frac{v}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{v}{g_{(S_1,S_3)}} \cdot \frac{T_{B_n}}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{T_{B_n}}.$$

Equality (2) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_3)}}$ by $sinkI^{data}$, $\frac{T_{B_n}}{g_{(S_1,B_n)}}$ by $out(a_n)$, γ and β by $\frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} \cdot v$. Where γ and β represent duplication numbers of the rate of tokens available in the sink interface within two different cases. Hence, the sink interface $sinkI$ and the sub-producer actor a_n obey the deadlock freeness and consistency condition already proved in Lemma 1. Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-producer actor.

Theorem 3.9 *In the direct communication case, the initial amount of tokens $d(a_n, Ha)$ of FiFo connecting the hierarchical actor and the sub-producer actor is given by $snkI^{data} - 1$.*

Proof: Inequality (16) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w \leq T_{B_n} - T_{S_1}.$$

Inequality (17) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w > T_{B_n} - 2T_{S_1}.$$

Inequality (18) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w > -T_{S_1}.$$

We combine the three inequalities, add T_{S_1} and we subtract $g_{(S_1,S_3)}$ from the middle. This yields to:

$$T_{B_n} \geq T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w + T_{S_1} - g_{(S_1,S_3)} > max(T_{B_n} - T_{S_1}, 0). \tag{19}$$

We pose $Y' = g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}$. When dividing (19) by Y', we obtain:

$$\frac{T_{B_n}}{Y'} \geq \frac{T_{B_n}}{Y'}(w \cdot \#u + u) - \frac{T_{S_1}}{Y'} w + \frac{T_{S_1}}{Y'} - \frac{1}{g_{(S_1,B_n)}} > max(\frac{T_{B_n}}{Y'} - \frac{T_{S_1}}{Y'}, 0).$$

The multiplication of the obtained equation by $g_{(S_1,B_n)}$ yields to:

$$Z' \cdot out(a_n) \geq Z' \cdot out(a_n) \cdot (w \cdot \#u + u) - sinkI^{data} \cdot w + \text{sinkI}^{data} - 1 > max(Z' \cdot sinkI^{data} - out(a_n), 0).$$

Where:

- $sinkI^{data}$ and $out(a_n)$ are deduced from Theorem 3.
- $Z' = \frac{g_{(S_1,B_n)}}{g_{(S_1,S_3)}}$.

Since $Z'.out(a_n)$ and $out(a_n)$ are both strictly superior than 0, then, even if we replace $Z'.out(a_n)$ by $out(a_n)$,

this inequality remains true. Hence, referring to inequality (9) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $snkI^{data} - 1$ by $d(a_n, Ha)$.

To transform a composed Simulink subsystem S_1, with direct communication between levels, into a deadlock free and consistent hierarchical actor, we rely on the two following Corollary 1 and Corollary 2.

Corollary 3.9.1 *To model direct communication between two levels of the hierarchy and ensure deadlock freeness and consistency, IBSDF introduces the source and sink interfaces concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

where $srcI^{data} = \frac{T_{S_1}}{gcd(T_{S_1}, T_{S_2})}$; $in(a_1) = \frac{T_{B_1}}{gcd(T_{S_1}, T_{B_1})}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v \geq 1$ *and* $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v < 1$.

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

where $snkI^{data} = \frac{T_{S_1}}{gcd(T_{S_1}, T_{S_3})}$; $out(a_n) = \frac{T_{B_n}}{gcd(T_{S_1}, T_{B_n})}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v \geq 1$ *and* $\beta = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v < 1$.

Corollary 3.9.2 *In the direct communication case, the initial amount of tokens $d(Ha, a_1)$ of FiFo connecting the hierarchical actor and the sub-consumer actor is given by $in(a_1) - 1$ and the initial amount of tokens $d(a_n, Ha)$ of FiFo connecting the hierarchical actor and the sub-producer actor is given by $snkI^{data} - 1$.*

To illustrate Simulink to IBSDF transformation in the direct communication case, we consider a multi-rate Simulink system S shown in figure 6 containing five blocks S_1, S_2, S_3, B_1 and B_2. S_1, S_2 and S_3 are the blocks of the top level with sample times $T_{S_1} = 100ms$, $T_{S_2} = 50ms$ and $T_{S_3} = 80ms$, respectively. S_1 is composed subsystem containing two atomic blocks B_1 and B_2 with sample times $T_{B_1} = 20ms$ and $T_{S_3} = 30ms$, respectively. (S_2 and S_3 can be atomic or composed blocks, in this example S_2 and S_3 are composed subsystems but we only focus on S_1 transformation to illustrate our results.)

Subsystems S_1, S_2 and S_3 are transformed into hierarchical actors Ha_1, Ha_2 and Ha_3, respectively. Atomic blocks B_1 and B_2 are transformed into atomic actors a_1 and a_2, respectively. Communications between S_1 and S_2, Communications between S_1 and S_3, Communications between B_1 and B_2 are obtained according to the rule of modeling one-level direct communication mentioned in section 3.3.1. We obtain as results:

- $in(Ha_1) = srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{100}{gcd(100,50)} = 2.$
- $out(Ha_1) = snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{100}{gcd(100,80)} = 5.$
- $out(a_1) = \frac{T_{B_1}}{g_{(B_1,B_2)}} = \frac{20}{gcd(20,30)} = 2.$
- $in(a_2) = \frac{T_{B_2}}{g_{(B_1,B_2)}} = \frac{30}{gcd(20,30)} = 3.$
- $d(a_1, a_2) = out(a_1) - 1 = 2.$

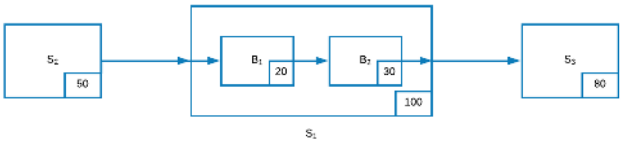

Figure 6. Multi-rate Simulink system in direct communication case.

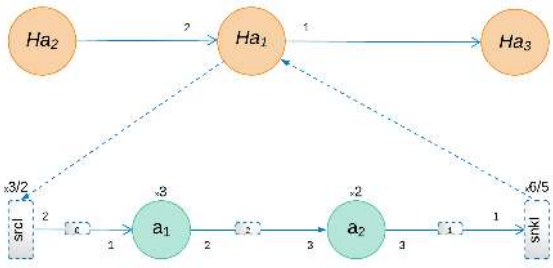

Figure 7. Resulting IBSDF in direct communication case.

Communication between S_1 and B_1 and Communication between S_1 and B_2 are both direct multi-levels communications. To model these communications and ensure deadlock freeness and consistency during the transformation process , we apply Corollary 1. Note that the execution repetition numbers v and u are obtained basing on the "Compute Repetition Algorithm".

- $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}} = \frac{20}{gcd(20,100)} = 1.$
  Since $v = 3$, $srcI^{data} \leq v \cdot in(a_1)$ then x is equal to $\frac{g_{(s_1,S_2)}}{g_{(s_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v = \frac{50}{20} \cdot \frac{20}{100} \cdot 3 = \frac{3}{2} \geq 1.$
- $out(a_2) = \frac{T_{B_2}}{g_{(S_1,B_2)}} = \frac{30}{gcd(30,100)} = 3.$
  Since $u = 2$, $snkI^{data} = 2 \leq u \cdot out(a_2) = 2 \times 3$ then γ is equal to $\frac{g_{(s_1,S_3)}}{g_{(s_1,B_2)}} \cdot \frac{T_{B_2}}{T_{S_1}} \cdot u = \frac{20}{10} \cdot \frac{30}{100} \cdot 2 = \frac{6}{5} \geq 1.$

Delays between levels are obtained according to Corollary 2:

- $d(Ha_1, a1) = in(a_1) - 1 = 0.$
- $d(Ha_1, a2) = snkI^{data} - 1 = 1.$

Figure 7 illustrates the resulting IBSDF graph.

Modeling levels delayed communication

To model levels delayed communication, we have to model source/sub-consumer actor delayed communication and sink/sub-producer actor delayed communication.

Source/sub-consumer actor delayed communication: a delayed communication between the source interface and the sub-consumer actor is defined through the following hierarchical dependency conditions:

- $[w \cdot \#v + v]$ fires at the same time or after the end time of [w].
- $[w \cdot \#v + v - 1]$ fires strictly before the end time of [w].
- $[w \cdot \#v + v]$ fires strictly before the end time of [w+1].

Based on these conditions we deduce the Lemma 6.

Lemma 3.10 *Let S_1 be a composed subsystem with sample period T_{S_1} containing a set of atomic blocks $B_1, B_2, ..., B_n$ firing in delayed communication mode. B_1 represents the sub-consumer atomic block with sample period T_{B_1}. A hierarchical dependency exists between the w^th execution of S_1 and v^th execution of B_1 if:*

$$T_{S_1} = b \cdot T_{B_1}. \tag{20}$$

Where b is a coefficient superior or equal to 1.

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{S_1} \cdot w \leq T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{21}$$

$$T_{S_1} \cdot w > T_{B_1} \cdot (w \cdot \#v + v - 2). \tag{22}$$

$$T_{S_1} \cdot (w+1) > T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{23}$$

Combining the three inequalities (21), (22) and (23) we obtain:

$$min(\frac{(w \cdot \#v + v - 2)}{x}, \frac{(w \cdot \#v + v - 1)}{w+1}) \cdot T_{B_1} < T_{S_1} \leq \frac{w \cdot \#v + v - 1}{w} \cdot T_{B_1}$$

. Since $w \cdot \#v + v - 1 \geq w$, it exists a coefficient $b \geq 1$ such that:

$$T_{S_1} = b \cdot T_{B_1}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-consumer actor in direct communication case, we refer to Theorem 5.

Theorem 3.11 *To ensure deadlock freeness between an hierarchical actor and its sub-consumer actor, IBSDF MoC introduces the source interface concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

where $srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}}$; $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v \geq 1$ and $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v < 1$.

Proof: We multiply equality (20) of Lemma 6 by $\frac{v}{g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}}$.
We obtain:

$$\frac{v}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{v}{g_{(S_1,S_2)}} \cdot \frac{T_{B_1}}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{T_{B_1}}.$$

Equality (1) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_2)}}$ by $srcI^{data}$, $\frac{T_{B_1}}{g_{(S_1,B_1)}}$ by $in(a_1)$, x and α by $\frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v$. Where x and α represent duplication numbers of the rate of tokens available in the source interface within two different cases. Hence, the source interface $srcI$ and the sub-consumer actor a_1 obey the deadlock freeness and consistency condition already proved in Lemma 1.
Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-consumer actor.

Theorem 3.12 *In the delayed communication case, the initial amount of tokens $d(Ha, a_1)$ of FiFo connecting the hierarchical actor and the sub-consumer actor is given by $in(a_1) + srcI^{data} - 1$.*

Proof: Inequality (21) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) \leq -T_{B_1}.$$

Inequality (22) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) > -2T_{B_1}.$$

Inequality (23) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) > -T_{S_1} \cdot w - T_{B_1}.$$

We combine the three inequalities, add $T_{B_1} + T_{S_1}$ and subtract $g_{(S_1,B_1)}$ from the middle, which results in:

$$T_{S_1} \geq T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) + T_{S_1} + T_{B_1} - g_{(S_1,B_1)} > max((T_{S_1} - T_{B_1}), 0). \tag{24}$$

We pose $Y = g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}$. When dividing (24) by Y, we obtain:

$$\frac{T_{S_1}}{Y} \geq \frac{T_{S_1}}{Y} w - \frac{T_{B_1}}{Y} \cdot (w \cdot \#v + v) + \frac{T_{B_1}}{Y} + \frac{T_{S_1}}{Y} - \frac{1}{g_{(S_1,S_2)}} > max(\frac{T_{S_1}}{Y} - \frac{T_{B_1}}{Y}, 0).$$

We multiply the resulting equation by $g_{(S_1,S_2)}$ we obtain:

$$Z \cdot srcI^{data} \geq Z \cdot srcI^{data} \cdot w - in(a_1) \cdot (w \cdot \#v + v) + in(a_1) + Z \cdot srcI^{data} - 1 > max(Z \cdot srcI^{data} - in(a_1), 0).$$

Where:

- $srcI^{data}$ and $in(a_1)$ are deduced from Theorem 5.
- $Z = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}}$.

Since $Z \cdot srcI^{data}$ and $srcI^{data}$ are both strictly superior than 0. Then, even if we replace $Z \cdot srcI^{data}$ by $srcI^{data}$ this inequality remains true. Hence, referring to inequality (8) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $in(a_1) + srcI^{data} - 1$ by $d(Ha, a_1)$.

Sink interface/sub-producer actor delayed communication: a delayed communication between the sub-producer actor and the sink interface is defined through the following hierarchical dependency conditions:

- $[w]$ fires at the same time or after the end time of $[w \cdot \#u + u]$.
- $[w-1]$ fires strictly before the end time of $[w \cdot \#u + u]$.
- $[w]$ fires strictly before the end time of $[w \cdot \#u + u + 1]$.

Based on these conditions we deduce the Lemma 7.

Lemma 3.13 *Let S_1 be a composed subsystem with sample period T_{S_1} containing a set of atomic blocks $B_1, B_2, ..., B_n$ firing in delayed communication mode. B_n represents the sub-producer atomic block with sample period T_{B_n}. A hierarchical dependency exists between the w^{th} execution of S_1 and u^{th} execution of B_1 if:*

$$T_{S_1} = c \cdot T_{B_n}. \tag{25}$$

Where c is in [0..1[

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{B_n} \cdot (w \cdot \#u + u) \leq T_{S_1}(w-1). \quad (26)$$

$$T_{B_n} \cdot (w \cdot \#u + u) > T_{S_1}(w-2). \quad (27)$$

$$T_{B_n} \cdot (w \cdot \#v + v + 1) > T_{S_1}(w-1). \quad (28)$$

When combining inequalities (26), (27) and (28) we obtain:

$$min(\frac{w-1}{w \cdot \#u + u + 1}, \frac{w-2}{w \cdot \#u + u}) \cdot T_{S_1} < T_{B_n} \leq \frac{w-1}{w \cdot \#u + u} \cdot T_{S_1}.$$

Since $0 \leq w-1 < w.\#u+u$, it exists a coefficient $c \in [0..1[$ such that:

$$T_{S_1} = c \cdot T_{B_n}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-producer actor in direct communication case, we refer to Theorem 7.

Theorem 3.14 *To ensure deadlock freeness between an hierarchical actor and its sub-producer actor, IBSDF introduces the sink interface concept such that:*

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

where $sinkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}}$; $out(a_n) = \frac{T_{B_n}}{g_{(S_1,B_n)}}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} \cdot u \geq 1$ *and* $\beta = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} \cdot u < 1$.

Proof: We multiply equality (25) of Lemma 7 by $\frac{v}{g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}}$. We obtain:

$$\frac{v}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{v}{g_{(S_1,S_3)}} \cdot \frac{T_{B_n}}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{T_{B_n}}.$$

Equality (2) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_3)}}$ by $sinkI^{data}$, $\frac{T_{B_n}}{g_{(S_1,B_n)}}$ by $out(a_n)$, γ and β by $\frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} \cdot v$. Where γ and β represent duplication numbers of the rate of tokens available in the sink interface within two different cases. Hence, the sink interface $sinkI$ and the sub-producer actor a_n obey the deadlock freeness and consistency condition already proved in Lemma 1. Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-producer actor.

Theorem 3.15 *In the delayed communication case, the initial amount of tokens $d(a_n, Ha)$ of FiFo connecting the hierarchical actor and the sub-producer actor is given by $snkI^{data} + out(a_n) - 1$.*

Proof: Inequality (26) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w \leq -T_{S_1}.$$

Inequality (27) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w > -2T_{S_1}.$$

Inequality (28) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w > -T_{B_n} - T_{S_1}.$$

We combine the three inequalities, add $T_{S_1} + T_{B_n}$ and subtract $g_{(S_1,S_3)}$ from the middle, which results in:

$$T_{B_n} \geq T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w + T_{S_1} + T_{B_n} - g_{(S_1,S_3)} > max(T_{B_n} - T_{S_1}, 0). \quad (29)$$

We pose $Y' = g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}$. When dividing (29) by Y', we obtain:

$$\frac{T_{B_n}}{Y'} \geq \frac{T_{B_n}}{Y'}(w \cdot \#u + u) - \frac{T_{S_1}}{Y'}w + \frac{T_{S_1}}{Y'} + \frac{T_{B_n}}{Y'} - \frac{1}{g_{(S_1,B_n)}} > max((\frac{T_{B_n}}{Y'} - \frac{T_{S_1}}{Y'}), 0).$$

The multiplication of the obtained equation by $g_{(S_1,B_n)}$ yields to:

$$Z' \cdot out(a_n) \geq Z' \cdot out(a_n) \cdot (w \cdot \#u + u) - sinkI^{data} \cdot w + sinkI^{data} + Z' \cdot out(a_n) - 1 > max(sinkI^{data} - Z' \cdot out(a_n), 0).$$

Where:

- $sinkI^{data}$ and $out(a_n)$ are deduced from Theorem 7.
- $Z' = \frac{g_{(S_1,B_n)}}{g_{(S_1,S_3)}}$.

Since $Z' > 0$, then, even if we replace $Z' \cdot out(a_n)$ by $out(a_n)$, this inequality remains true. By consequence, referring to inequality (9) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $snkI^{data} + out(a_n) - 1$ by $d(a_n, Ha)$.

To transform a composed Simulink subsystem S, with delayed communication between levels, into a deadlock free and consistent hierarchical actor, we rely on the two following corollaries.

Corollary 3.15.1 *To model delayed communication between two levels of the hierarchy and ensure deadlock freeness and consistency, IBSDF MoC introduces the source and sink interfaces concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

where $srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}}$; $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v \geq 1$ *and* $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v < 1$.

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq \\ & u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

where $snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}}$; $out(a_n) = \frac{T_{B_n}}{g_{(S_1,B_n)}}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v \geq 1$ *and* $\beta = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v < 1$.

Corollary 3.15.2 *In the delayed communication case, the initial amount of tokens $d(Ha, a_1)$ of FiFo connecting the hierarchical actor and the sub-consumer actor is given by $in(a_1) + srcI^{data} - 1$ and the initial amount of tokens $d(a_n, Ha)$ of FiFo connecting the hierarchical actor and the sub-producer actor is given by $snkI^{data} + out(a_n) - 1$.*

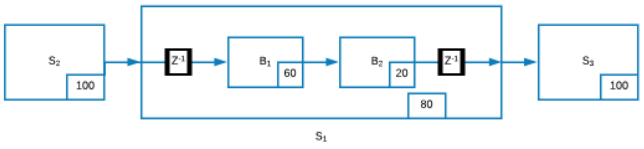

Figure 8. Multi-rate Simulink system in delayed communication case.

To illustrate Simulink to IBSDF transformation in the delayed communication case, we consider a multi-rate Simulink system S shown in figure 8 containing five blocks S_1, S_2, S_3, B_1 and B_2. S_1, S_2 and S_3 are the blocks of the top level with sample times $T_{S_1} = 80ms$, $T_{S_2} = 100ms$ and $T_{S_3} = 100ms$, respectively. S_1 is composed subsystem containing two atomic blocks B_1 and B_2 with sample times $T_{B_1} = 60ms$ and $T_{B_2} = 20ms$, respectively. (S_2 and S_3 can be atomic or composed blocks, in this example S_2 and S_3 are composed subsystems but we only focus on S_1 transformation to illustrate our results.)

Subsystems S_1, S_2 and S_3 are transformed into hierarchical actors Ha_1, Ha_2 and Ha_3, respectively. Atomic blocks B_1 and B_2 are transformed into atomic actors a_1 and a_2, respectively. Communication between S_1 and S_2, Communication between S_1 and S_3, Communications between B_1 and B_2 are obtained according to the rule of modeling one-level delayed communication mentioned in section 3.3.1. We obtain as results:

- $in(Ha_1) = srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{80}{gcd(100,80)} = 4.$
- $out(Ha_1) = snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{100}{gcd(100,80)} = 4.$
- $out(a_1) = \frac{T_{B_1}}{g_{(B_2,B_1)}} = \frac{60}{gcd(60,20)} = 3.$
- $in(a_2) = \frac{T_{B_2}}{g_{(B_1,B_2)}} = \frac{20}{gcd(60,20)} = 1.$
- $d(a_1, a_2) = out(a_1) - 1 = 0.$

Communication between S_1 and B_1 and Communication between S_1 and B_2 are both delayed multi-levels communication. To model these communications and ensure deadlock freeness and consistency during the transformation process , we apply Corollary 3. Note that the execution repetition numbers v and u are obtained based on the "Compute Repetition Algorithm":

- $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}} = \frac{60}{gcd(60,100)} = 3.$ Since $v = 1$, $srcI^{data} = 4 \geq v.in(a_1) = 3$ then α is equal to $\frac{g_{(S_1,S_2)}}{g_{(s_1,B_1)}}.\frac{T_{B_1}}{T_{S_1}}.v = \frac{20}{20}.\frac{60}{80}.3 = \frac{3}{4} \leq 1.$
- $out(a_2) = \frac{T_{B_2}}{g_{(S_1,B_2)}} = \frac{20}{gcd(20,80)} = 1.$ Since $u = 3$, $snkI^{data} = 4 \geq u.out(a_2) = 3 * 1$ then β is equal to $\frac{g_{(S_1,S_3)}}{g_{(S_1,B_2)}}.\frac{T_{B_2}}{T_{S_1}}.u = \frac{20}{20}.\frac{20}{80}.3 = \frac{3}{4} \leq 1.$

Delays between levels are obtained according to Corollary 4:

- $d(Ha_1, a1) = in(a_1) + srcI^{data} - 1 = 6.$
- $d(Ha_1, a2) = out(a_2) + snkI^{data} - 1 = 4.$

Figure 9 illustrates the resulting IBSDF graph.

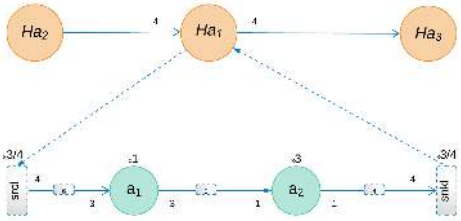

Figure 9. Resulting IBSDF in delayed communication case.

Modeling levels hybrid communication

To model levels direct communication, we have to model source/sub-consumer actor communication and sink/sub-producer actor communication.

Source/sub-consumer actor hybrid communication: a hybrid communication between the source interface and the consumer sub-actor is defined through the following hierarchical dependency conditions:

- $[w \cdot \#v + v]$ fires strictly after the beginning time of [w].
- $[w \cdot \#v + v - 1]$ fires before or at the same beginning time of [w].
- $[w \cdot \#v + v]$ fires before or at the same beginning time of [w+1].

Based on these conditions we deduce the Lemma 8.

Lemma 3.16 *Let S_1 be a composed subsystem with sample period T_{S_1} containing a set of atomic blocks $B_1, B_2, ..., B_n$ firing in hybrid communication mode. B_1 represents the sub-consumer atomic block with sample period T_{B_1}. A hierarchical dependency exists between the w^{th} execution of S_1 and v^{th} execution of B_1 if:*

$$T_{S_1} = b \cdot T_{B_1}. \tag{30}$$

Where b is a coefficient superior or equal to 1.

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{S_1} \cdot (w-1) < T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{31}$$

$$T_{S_1} \cdot (w-1) \geq T_{B_1} \cdot (w \cdot \#v + v - 2). \tag{32}$$

$$T_{S_1} \cdot w \geq T_{B_1} \cdot (w \cdot \#v + v - 1). \tag{33}$$

When we added inequality (32) and inequality (33) we obtain:

$$2T_{S_1} \cdot w - T_{S_1} \geq 2T_{B_1} \cdot (w \cdot \#v + v) - 3T_{B_1}.$$

We multiply inequality (31) by -1 and added it with the resulted inequality. We obtain:

$$T_{S_1} > T_{B_1} \frac{w \cdot \#v + v - 2}{w}.$$

Since $\frac{w \cdot \#v + v - 2}{w} > 1$, it exists a coefficient $b > 1$ such that:

$$T_{S_1} = b \cdot T_{B_1}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-consumer actor in direct communication case, we refer to Theorem 9.

Theorem 3.17 *To ensure deadlock freeness between an hierarchical actor and its sub-consumer actor, IBSDF introduces the source interface concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v \cdot in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

where $srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}}$; $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v \geq 1$ *and* $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v < 1$.

Proof: We multiply equality (30) of Lemma 8 by $\frac{v}{g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}}$.
We obtain:

$$\frac{v}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{v}{g_{(S_1,S_2)}} \cdot \frac{T_{B_1}}{g_{(S_1,B_1)}} \cdot \frac{T_{S_1}}{T_{B_1}}.$$

Equality (1) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_2)}}$ by $srcI^{data}$, $\frac{T_{B_1}}{g_{(S_1,B_1)}}$ by $in(a_1)$, x and α by $\frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} \cdot v$. Where x and α represent duplication numbers of the rate of tokens available in the source interface within two different cases. Hence, the source interface $srcI$ and the sub-consumer actor a_1 obey the deadlock freeness and consistency condition already proved in Lemma 1.

Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-consumer actor.

Theorem 3.18 *In the hybrid communication case, the initial amount of tokens* $d(Ha, a_1)$ *of FiFo connecting the hierarchical actor and the sub-consumer actor is given by* $in(a_1)$.

Proof: Inequality (31) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) < T_{S_1} - T_{B_1}.$$

Inequality (32) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) \geq T_{S_1} - 2T_{B_1}.$$

Inequality (33) is equivalent to:

$$T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) \geq -T_{B_1}.$$

We combine the three inequalities and add T_{B_1}, which results in the following quation:

$$T_{S_1} > T_{S_1} \cdot w - T_{B_1} \cdot (w \cdot \#v + v) + T_{B_1} \geq max(T_{S_1} - T_{B_1}, 0). \quad (34)$$

We pose $Y = g_{(S_1,B_1)} \cdot g_{(S_1,S_2)}$. When dividing (34) by Y, we obtain:

$$\frac{T_{S_1}}{Y} > \frac{T_{S_1}}{Y} w - \frac{T_{B_1}}{Y} \cdot (w \cdot \#v + v) + \frac{T_{B_1}}{Y} \geq max(\frac{T_{S_1}}{Y} - \frac{T_{B_1}}{Y}, 0).$$

We multiply the resulting equation by $g_{(S_1,S_2)}$ we obtain:

$$Z \cdot srcI^{data} > Z \cdot srcI^{data} \cdot w - in(a_1) \cdot (w \cdot \#v + v) + in(a_1)$$
$$\geq max(Z \cdot srcI^{data} - in(a_1), 0).$$

Where:

- $srcI^{data}$ and $in(a_1)$ are deduced from Theorem 9.
- $Z = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}}$.

Since $Z > 0$, then, even if we replace $Z \cdot srcI^{data}$ by $srcI^{data}$ this inequality remains true. Hence, referring to inequality (8) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $in(a_1)$ by $d(Ha, a_1)$.

Sink interface/sub-producer actor hybrid communication: a hybrid communication between the sub-producer actor and the sink interface is defined through the following hierarchical dependency conditions:

- $[w]$ fires strictely after the beginning time of $[w \cdot \#u + u]$.
- $[w-1]$ fires before or at the same beginning time of $[w \cdot \#u + u]$.
- $[w]$ fires before or at the same beginning time of $[w \cdot \#u + u + 1]$.

Based on these conditions we deduce the Lemma 9.

Lemma 3.19 *Let* S_1 *be a composed subsystem with sample period* T_{S_1} *containing a set of atomic blocks* $B_1, B_2, ..., B_n$ *firing in hybrid communication mode.* B_n *represents the sub-producer atomic block with sample period* T_{B_n}. *A hierarchical dependency exists between the* w^{th} *execution of* S *and* u^{th} *execution of* B_n *if:*

$$T_{S_1} = c \cdot T_{B_n}. \quad (35)$$

Where c *is in* $[0..1[$

Proof: Hierarchical dependency conditions are translated into the following in-equations:

$$T_{B_n} \cdot (w \cdot \#u + u - 1) < T_{S_1} \cdot (w - 1). \quad (36)$$

$$T_{B_n} \cdot (w \cdot \#u + u - 1) \geq T_{S_1} \cdot (w - 2). \quad (37)$$

$$T_{B_n} \cdot (w \cdot \#v + v) \geq T_{S_1} \cdot (w - 1). \quad (38)$$

When combining inequalities (36), (37) and (38), we obtain:

$$min(\frac{w-1}{w \cdot \#u + u - 1}, \frac{w-2}{w \cdot \#u + u}) \cdot T_{S_1} \leq T_{B_n}$$
$$< \frac{w-1}{w \cdot \#u + u - 1} \cdot T_{S_1}.$$

Since $0 \leq w - 1 < w \cdot \#u + u - 1$, it exists a coefficient $c \in [0..1[$ such that:

$$T_{S_1} = c \cdot T_{B_n}.$$

To ensure deadlock freeness between the hierarchical actor and the sub-producer actor in hybrid communication case, we refer to Theorem 11.

Theorem 3.20 *To ensure deadlock freeness between an hierarchical actor and its sub-producer actor, in hybrid communication case, IBSDF MoC introduces the sink*

interface concept such that:

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq \\ & u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

where $snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}}$; $out(a_n) = \frac{T_{B_n}}{g_{(S_1,B_n)}}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v \geq 1$ *and* $\beta = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} . \frac{T_{B_n}}{T_{S_1}} v < 1$.

Proof: We multiply equality (35) of Lemma 9 by $\frac{v}{g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}}$. We obtain:

$$\frac{v}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{v}{g_{(S_1,S_3)}} \cdot \frac{T_{B_n}}{g_{(S_1,B_n)}} \cdot \frac{T_{S_1}}{T_{B_n}}.$$

Equality (2) of Lemma 1 is obtained by replacing, in the resulting equation, $\frac{T_{S_1}}{g_{(S_1,S_3)}}$ by $sinkI^{data}$, $\frac{T_{B_n}}{g_{(S_1,B_n)}}$ by $out(a_n)$, γ and β by $\frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} \cdot v$. Where γ and β represent duplication numbers of the rate of tokens available in the sink interface within two different cases. Hence, the sink interface $sinkI$ and the sub-producer actor a_n obey the deadlock freeness and consistency condition already proved in Lemma 1. Based on the precedence constraints between two levels we determine the initial token amount in the FiFo connecting the hierarchical actor and the sub-producer actor.

Theorem 3.21 *In the hybrid communication case, the initial amount of tokens* $d(a_n, Ha)$ *of FiFo connecting the hierarchical actor and the sub-producer actor is given by* $srcI^{data}$.

Proof: Inequality (36) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w < T_{B_n} - T_{S_1}.$$

Inequality (37) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w \geq T_{B_n} - 2T_{S_1}.$$

Inequality (38) is equivalent to:

$$T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w \geq -T_{S_1}.$$

We add T_{S_1} and we combine the three inequalities, which results in the following equation:

$$T_{B_n} > T_{B_n} \cdot (w \cdot \#u + u) - T_{S_1} \cdot w + T_{S_1} \leq max(T_{B_n} - T_{S_1}, 0). \quad (39)$$

We pose $Y' = g_{(S_1,B_n)} \cdot g_{(S_1,S_3)}$. When dividing (39) by Y', we obtain:

$$\frac{T_{B_n}}{Y'} \geq \frac{T_{B_n}}{Y'}(w \cdot \#u + u) - \frac{T_{S_1}}{Y'} w + \frac{T_{S_1}}{Y'} > max((\frac{T_{B_n}}{Y'} - \frac{T_{S_1}}{Y'}), 0).$$

The multiplication of the obtained equation by $g_{(S_1,B_n)}$ yields to:

$$Z' \cdot out(a_n) \geq Z' \cdot out(a_n) \cdot (w \cdot \#u + u) - sinkI^{data} \cdot w + sinkI^{data} > max(sinkI^{data} - Z' \cdot out(a_n), 0).$$

Where:

- $sinkI^{data}$ and $out(a_n)$ are deduced from Theorem 11.
- $Z' = \frac{g_{(S_1,B_n)}}{g_{(S_1,S_3)}}$.

Since $Z' > 0$, then, even if we replace $Z' \cdot out(a_n)$ by $out(a_n)$, this inequality remains true. Hence, referring to inequality (9) , we obtain a precedence relation between an hierarchical actor and its sub-consumer actor when replacing $sinkI^{data}$ by $d(a_n, Ha)$.

To transform a composed Simulink subsystem S, with hybrid communication between levels, into a deadlock free and consistent hierarchical actor, we rely on the two following Corollary 5 and Corollary 6.

Corollary 3.21.1 *To model direct communication between two levels of the hierarchy and ensure deadlock freeness and consistency, IBSDF MoC introduces the source and sink interfaces concept such that:*

$$srcI^{data} = \begin{cases} \frac{v}{x} in(a_1) & if \quad srcI^{data} \leq v.in(a_1). \\ \frac{v}{\alpha} in(a_1) & otherwise. \end{cases}$$

Where $srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}}$; $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}}$; $x = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} v \geq 1$ *and* $\alpha = \frac{g_{(S_1,S_2)}}{g_{(S_1,B_1)}} . \frac{T_{B_1}}{T_{S_1}} v < 1$.

$$sinkI^{data} = \begin{cases} \frac{u}{\gamma} out(a_n) & if \quad sinkI^{data} \leq \\ & u \cdot out(a_n). \\ \frac{u}{\beta} out(a_n) & otherwise. \end{cases}$$

Where $snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}}$; $out(a_n) = \frac{T_{B_n}}{g_{(S_1,B_n)}}$; $\gamma = \frac{g_{(S_1,S_3)}}{g_{(S_1,B_n)}} \cdot \frac{T_{B_n}}{T_{S_1}} v \geq 1$ *and* $\beta = \frac{T_{B_n}}{T_{S_1}} v < 1$.

Corollary 3.21.2 *In the hybrid communication case, the initial amount of tokens* $d(Ha, a_1)$ *of FiFo connecting the hierarchical actor and the sub-consumer actor is given by* $in(a_1)$ *and the initial amount of tokens* $d(a_n, Ha)$ *of FiFo connecting the hierarchical actor and the sub-producer actor is given by* $sinkI^{data}$.

To illustrate Simulink to IBSDF transformation in the hybrid communication case, we consider a multi-rate Simulink system S shown in figure 10 containing five blocks S_1, S_2, S_3, B_1 and B_2. S_1, S_2 and S_3 are the blocks of the top level with sample times $T_{S_1} = 100ms$, $T_{S_2} = 10ms$ and $T_{S_3} = 100ms$, respectively. S_1 is a composed subsystem containing two atomic blocks B_1 and B_2 with sample times $T_{B_1} = 50ms$ and $T_{S_3} = 30ms$, respectively. (S_2 and S_3 can be atomic or composed blocks, in this example S_2 and S_3 are composed subsystems but we only focus on S_1 transformation to illustrate our results.)

Subsystems S_1, S_2 and S_3 are transformed into hierarchical actors Ha_1, Ha_2 and Ha_3, respectively. Atomic blocks B_1 and B_2 are transformed into atomic actors a_1 and a_2 respectively. Communications between S_1 and S_2, Communications between S_1 and S_3, Communications between B_1 and B_2 are obtained according to the rules of modeling one-level hybrid communication mentioned in section 3.3.1. we obtain as results:

- $in(Ha_1) = srcI^{data} = \frac{T_{S_1}}{g_{(S_1,S_2)}} = \frac{100}{gcd(100,10)} = 10.$
- $out(Ha_1) = snkI^{data} = \frac{T_{S_1}}{g_{(S_1,S_3)}} = \frac{100}{gcd(100,100)} = 1.$

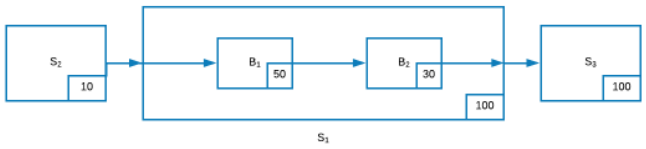

Figure 10. Multi-rate Simulink system in hybrid communication case.

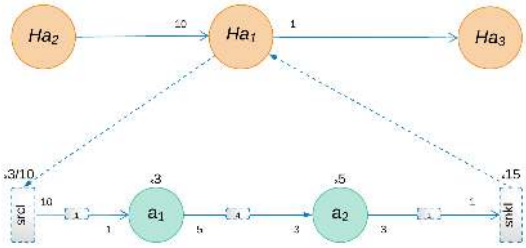

Figure 11. Resulting IBSDF in hybrid communication case.

- $out(a_1) = \frac{T_{B_1}}{g_{(B_2,B_1)}} = \frac{50}{gcd(50,30)} = 5.$
- $in(a_2) = \frac{T_{B_2}}{g_{(B_1,B_2)}} = \frac{30}{gcd(50,30)} = 3.$
- $d(a_1, a_2) = out(a_1) - 1 = 4.$

Communication between S_1 and B_1 and Communication between S_1 and B_2 are both hybrid multi-levels communications. To model these communications and ensure deadlock freeness and consistency during the transformation process , we apply Corollary 5. Note that the execution repetition numbers v and u are obtained based on the "Compute Repetition Algorithm":

- $in(a_1) = \frac{T_{B_1}}{g_{(S_1,B_1)}} = \frac{50}{gcd(50,100)} = 1.$ Since $v = 3$, $srcI^{data} = 10 \geq v.in(a_1) = 3 * 1$ then α is equal to $\frac{g_{(s_1,S_2)}}{g_{(s_1,B_1)}} \cdot \frac{T_{B_1}}{T_{S_1}} . v = \frac{10}{50} \cdot \frac{50}{100} . 3 = \frac{3}{10} \leq 1.$
- $out(a_2) = \frac{T_{B_2}}{g_{(S_1,B_2)}} = \frac{30}{gcd(30,100)} = 3.$ Since $u = 10$, $snkI^{data} = 1 \leq u.out(a_2) = 5 * 3$ then γ is equal to $\frac{g_{(s_1,S_3)}}{g_{(s_1,B_2)}} \cdot \frac{T_{B_2}}{T_{S_1}} . u = \frac{100}{10} \cdot \frac{30}{100} . 5 = 15 \geq 1.$

Delays between levels are obtained according to Corollary 6:

- $d(Ha_1, a1) = in(a_1) = 1.$
- $d(Ha_1, a2) = snkI^{data} = 1.$

Figure 11 illustrates the resulting IBSDF graph.

4 Implementation

The overall extended work-flow (figure 12) of our proposed approach is based on a specification of the application behavior with Simulink model, multi-core described with IPXACT Language, performance metrics estimation and automatic C code generation.

The first task is to transform a given Simulink model into IBSDF graph. During this task, three main functionalities are executed: As first step, Simulink model elements are gathered and converted into software objects using a Simulink Parser. Secondly, these software objects are translated into IBSDF objects as detailed in section 3 . Then an IBSDF graph Generator reconstructs the obtained objects into the IBSDF graph elements and generates the IBSDF graph format. The resulted graph undergoes some transformations [25] until obtaining a DAG graph to expose parallelism in an intuitive manner to the mapping.

The next step is to map each actor of the DAG into the multi-core platform in a specific manner using the simple ordering heuristic algorithm [21] which is a modified version of list scheduling algorithm [26]. The scheduling solution performance is evaluated using ABC modules [27]. The performance metrics estimation serves to evaluate the parallel system and helps designer to take the suitable decisions.

Once the mapping decision is made, the last task of the work-flow is to automatically generate a compatible C code for the target hardware platform. In order to achieve this, a host C code library is required. This library is resulted from the code generation of each Simulink block composing the model using Simulink coder tool. This workf-low was implemented into S-Preesm tool.

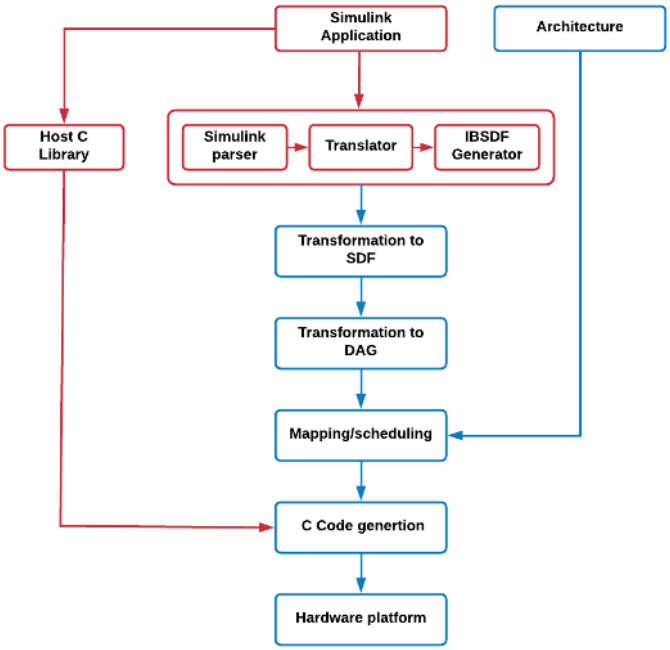

Figure 12. The extended rapid prototyping work-flow.

5 Results and Discussion

In this section, an embedded signal processing application is used to illustrate the efficiency of our approach in a realistic setting. Such LTE QPSK is a complex model adopting multi-core architecture on the transmitter and receiver sides, it is a well suitable example to demonstrate our approach capabilities. We have used S-Preesm tool to translate the Simulink model provided by [22] and generate a compatible C code to the parallel hardware platform.

5.1 Case study overview: LTE QPSK

The Long-Term Evolution LTE QPSK is a wireless communication of high speed data for mobile. The LTE system design, based on the MIMO OFDM technology and

Turbo coding, is required to optimize mobile speeds ranging from 15 to 120km/h.

The LTE QPSK is a multi-rate Simulink model which has three levels of hierarchy. The top level contains three adjacent subsystems: the transmitter, the receiver and the channel. The channel is required only for simulation, consequently, we do not take it into account. Further the transmitter and receiver channels are alike and treated in the same way. Then, in the rest of our work we only illustrate the LTE QPSK transmitter side. The Simulink model of the transmitter is presented in Figure 13. The top level includes 8 subsystems and atomic blocks:

- The Bernoulli Binary Generator: it creates a Bernoulli random binary number. It generates 20 samples.
- The CRC encoder: it produces cyclic redundancy code bits for each input data frame.
- The Turbo encoder: it encodes continuous stream of data using a concatenated encoding structure and an iterative algorithm to decode the sequence. The turbo encoder was implemented as a composed subsystem. More details are found in [22].
- Modulation QPSK, 16QAM, 64QAM : the modulation is performed with a gray mapping.
- OFDM block: the orthogonal frequency division multiplexing (OFDM) is based on the fast reverse fourier transform (IFFT) of each data symbol corresponding to each transmitting antenna. OFDM is known as the best kind of modulation which is able to overcome multipath problems. OFDM block is implemented as a composed subsystem. OFDM block is implemented as a composed subsystem such as depicted in figure 14.
- The serial-to-parallel P/S block: it consists of converting multiple data stream, received simultaneously, from serial format to parallel format.

5.2 Transformation

The Simulink model of the LTE QPSK Transmitter chain had a hierarchy depth of two levels. We count 24 atomic blocks and 4 subsystems (we did not count output and input blocks).

As first step, we simulated the Simulink model to obtain sample times of each block (atomic and composed) composing the given model. We have, then, executed the transformation task of the work-flow. The transformation of the LTE QPSK transmitter Simulink model is successfully done by applying algorithms detailed and proved in section 3.

The resulting IBSDF graph is a consistent and deadlock free graph with the same hierarchy depth, the same numbers of actors (atomic and hierarchical) and the same number of FiFo channels as the input Simulink model. The output graph can be seen in figure 15.

To illustrate how our proposed approach is applied to this case study, we focused on the OFDM subsystem and detailed its translation. The OFDM subsystem belongs to the top level of the LTE QPSK transmitter model. This subsystem is connected to Training subsystem Training insertion subsystem and QPSK modulator atomic block with sample times respectively $8.10-1ms$, $8.10-1ms$ and $200.10-1ms$ OFDM contains seven atomic blocks as depicted in figure 14. Communications between blocks and levels are direct. According to corollaries 1 and 2 we obtained:

- the amounts of tokens available in the source interfaces and sink interface are respectively $srcI_1^{data} = 25$, $srcI_2^{data} = 1$ and $sinkI_1^{data} = 25$.
- The consumed data by the sub-consumer concatenate2, the sub-consumer Select rows and the produced data by the sub-producer Add cyclic prefix are respectively equal to $in(concatenate2) = 1$, $in(Selectrows) = 1$ and $out(Addcyclicprefix) = 1$.
- Source and the sink interfaces must be respectively duplicated $\alpha = 1/25$ $x = 1$ and $\beta = 1/25$ times to ensure deadlock freeness between levels and the output sub-graph consistency.
- The initial amount of tokens available in the FiFo connecting the hierarchical actor OFDM and the sub-consumer concatenate2 is equal to $d(OFDM, concatenate2) = 0$. The initial amount of tokens available in the FiFo connecting the hierarchical actor OFDM and the sub-consumer Select rows is equal to $d(OFDM, Selectrows) = 0$. Similarly, the initial amount of tokens available in the FiFo connecting the hierarchical actor OFDM and the sub-producer Add cyclic prefix is equal to $d(Addcyclicprefix, OFDM) = 0$.

The resulted graph is illustrated in figure 16.

Since the transformation task is realized, the resulting graph is converted into a DAG containing 203 actors and 305 FiFo channels. The information about generated LTE QPSK transmitter graph is shown in Tab.1. Starting from this result, we can provide solutions for scheduling and code generation. The execution of the whole work-flow using S-Preesm tool is achieved over the shortest feasible time intervals. It takes only few Milli-seconds.

Table 1. Translation statistic of LTE QPSK transmitter application.

Simulink model	Subsystems number	4
	atomic blocks number	24
	Lines number	35
IBSDF graph	Subgraphs number	4
	atomic actors number	24
	FiFo number	35
DAG graph	atomic actors number	233
	FiFo number	305

5.3 Code generation results and performance evaluation

After the transformation of the LTE QPSK transmitter side into a schedulable IBSDF graph through the pro-

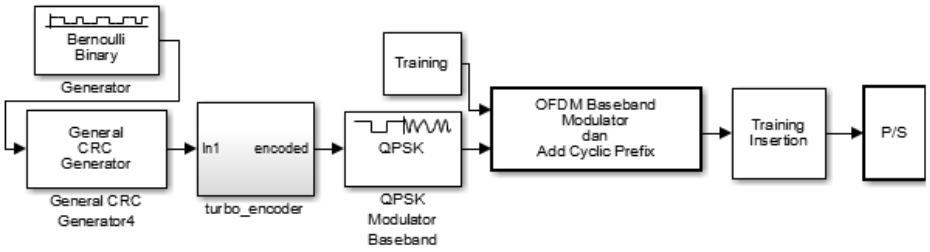

Figure 13. The Simulink model representing the LTE QPSK transmitter side.

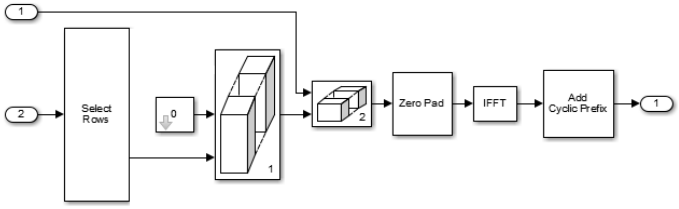

Figure 14. OFDM Simulink subsystem

posed model transformation framework, The IBSDF undergoes several operations to be ready for the generation code process, as described in section 4. The LTE transmitter Simulink application is translated into C parallel code utilizing the AAM (Algorithm Architecture Matching) [28] method. This method is based on generating self-timed coordination code from data-flow graph schedule. The host code and communication libraries are obtained by generating the code of each Simulink block composing the model by means of Simulink coder tool. In the one-core architecture case, the generated code using S-Preesm counts 1084 lines in which the host code library is not considered.

The output result from the code generation using Simulink coder is a vector of size (1000*1). This output corresponds to the data stream transferred to the LTE QPSK receiver side via the AWGN channel. To prove the correctness and efficiency of our approach we generate the code using S-Preesm tool. The result file was fairly close to the Simulink coder result.

To show the positive impact of our approach on the application performance, we deploy the generated codes into the Raspberry pi3 architecture. The Raspberry pi3 had a Broadcom BCM2837 processor 64Bit with Quad cores ARM Cortex-A53 and a clock speed with 1.2 GHZ. The Raspberry pi 3 represents a good hardware platform to introduce multi-core programming. Furthermore, we choose to evaluate performance using Raspberry pi3 because most of smart phone devices are using similar multi-core ARM processors as Raspberry. Metrics taken into account during the overall process are: execution time, speedup and efficiency. The speedup measure allows programmer to detect how much an application executed on multiple processors is faster than its execution on a single processor. Efficiency, the second performance metric, is deduced from the speed up metric. In fact, efficiency is the average utilization of n processors. It is obtained from the ratio of speed up and the number of processors allocated.

5.3.1 Performance results using Simulink coder tool

In this section we present the result using Simulink coder tool. Since passing from Simulink applications to multi-core implementations is not trivial as detailed in previous sections, we generated, using Simulink coder, a C code compatible only for single-core architecture. Then, we deployed the generated code into the Raspberry pi3 platform. The resulted execution time is of $0.288s$. The achieved speedup and efficiency are equal to 1. This result is due to the fact that we use only a single core.

5.3.2 Performance results using S-Preesm tool

In this section, we generated C compatible codes of the LTE transmitter side application for several multi-core architecture using S-Preesm tool. First, generated code was deployed onto single-core. The resulted execution time is of 0.273 s. Since the target architecture is constructed with one core, speedup and efficiency are consequently equal to 1.

To analyze the impact of multi-core architecture on the "LTE transmitter side" execution time, speedup and efficiency, we generated C codes compatible for dual-cores, 3-cores and 4-cores using S-Preesm and starting from the application Simulink model.

Table 2. Performance results of the code generated using the S-Preesm tool.

Cores number	Execution time	Speedup	Efficiency
1	0.273 s	1	1
2	0.183 s	1.49	0.745
3	0.153 s	1.78	0.59
4	0.110 s	2.48	0.62

When dealing with dual core, the application execution time is of $0.185s$. Speedup and efficiency values reach 1.49 and 0.745, respectively. We can observe in figure 17 that deploying into dual-cores noticeably improves the Simulink application performance compared to deploying into single-core.

Better performance results in terms of execution time and speedup when deploying into 3-cores are realized. Indeed, executing the Simulink application 3-cores architecture return an implementation with an execution time of $0.153s$ and improvement of 178% in speedup compared to the reached speedup when using single-core as depicted in figure 17.

The same application was deployed into 4-cores. Figure 17 showed That the minimum execution time of the

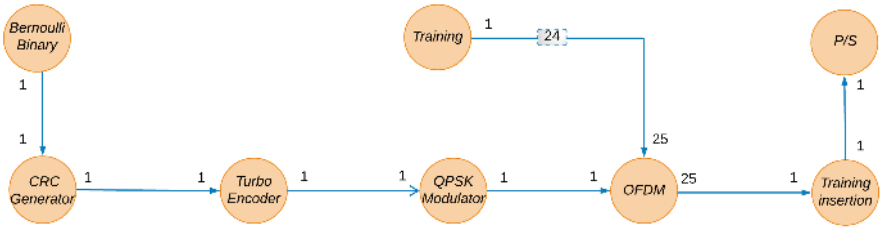

Figure 15. The top level of the generated IBSDF graph after the translation of the LTE transmitter Simulink model using S-Preesm.

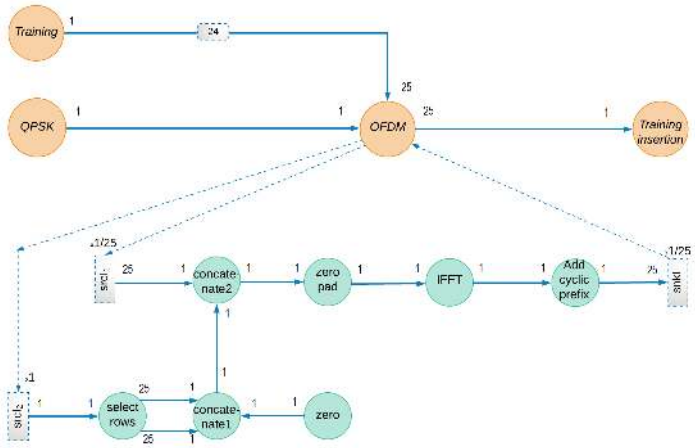

Figure 16. The obtained IBSDF sub-graph after the translation of the OFDM Simulink sub-system using S-Preesm.

case study application is achieved on 4-cores compared to the single-core execution. Likewise, the best speedup value is reached with 2.48 on 4-cores architecture meaning that, compared to single-core execution, speedup is approved with 248%. Table 2 summarizes performance measurements for each target hardware implementation when using S-Preesm tool.

5.3.3 Results analysis

The obtained results in previous sections showed the efficiency of our proposal to improve Simulink applications performance. In fact, even deploying generated codes on single-core platform, the execution time of the code generated using our proposal is lower than the one generated using Simulink coder. This is due to the fact that Simulink Coder tool enforces the addition of memory buffers and latencies whenever there is a rate transition among non-virtual blocks. Hence, the time performance of the application is negatively influenced. However, these additions are not requiered when using our approach. Further, S-Preesm implements a scheduling module which splits the scheduling/mapping functionnality and the evaluation cost of the generated solutions functionnality into two sub-modules. This division produces an advanced scalability in terms of schedule quality and execution time.

In order to demonstrate the effectiveness of our approach in improving Hierarchical Simulink application performance, we investigate the execution time-efficiency profile. This profile represents an important cost-benefit trade-off in evaluating multi-core application performance. Efficiency indicates benefit and execution time indicates cost. Figure 18 illustrates the profile for "LTE Transmitter side" Simulink application when using S-Preesm. In the first instance, we compare only ratios of

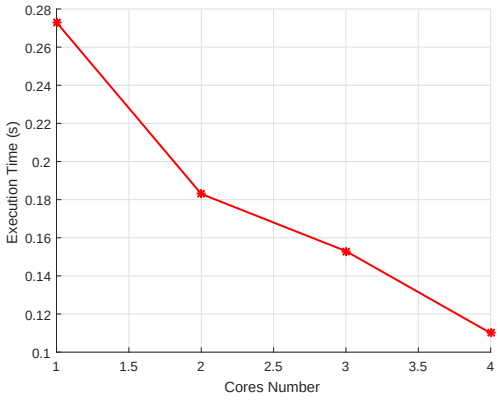

Figure 17. Execution time evaluation in function of number of cores.

efficiency to execution time resulting from Simulink coder and S-Preesm when deploying generated codes on single-core platform. The ratio of efficiency to execution time resulting from Simulink coder is equal to 3.47 and the ratio of efficiency to execution time resulting from S-Preesm is equal to 3.66. We find that the use of S-Preesm yields better result. Furthermore, as depicted in Figure 18 the ratio of efficiency to execution time reaches the maximum when the execution is achieved onto 4-cores architecture. Hence, compared to single-core execution, our Simulink application archives the most efficiency utilization of each core when executing onto 4-cores with a ratio of efficiency to execution time equal to 5.63. Thus, surveying results above, we reveal the impact of transforming hierarchical Simulink models into multi-core execution using our proposal in improving performance in terms of execution time, speedup and execution time-efficiency.

Further, transforming the Simulink model of LTE

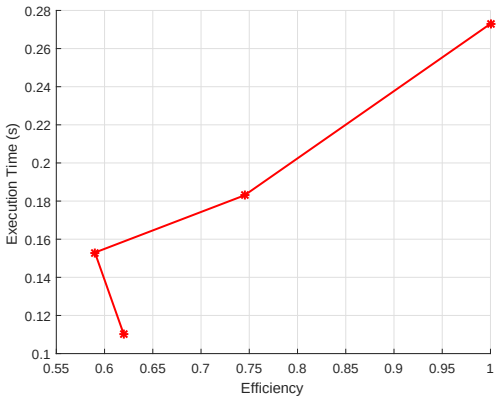

Figure 18. Execution time-Efficiency profile for "LTE Transmitter side.

QPSK Transmitter chain into multi-core execution using S-Preesm ease its parallelizing and allows us to take advantages of this high degree of parallelism. Moreover, OFDM subsystem can be reused for other similar systems. The use of our open source proposal allows to eliminate many constraints and configurations imposed by the commercial toolbox "real-Time Workshop Embedded Coder" required before code generation. S-Preesm allows also a cost-free parallel C code generation.

6 Conclusion

In this article, we have described an efficient approach to automatically optimize and transform hierarchical Simulink to multi-core execution. The proposed methodology consists of converting hierarchical Simulink models into an intermediate model before generating parallel codes. In this work, we proposed IBSDF as an intermediate representation. Our translation approach is the first to preserve and exploit hierarchy behavior of Simulink applications.

To achieve this, we extended the existing tool Preesm to support Simulink applications which we named S-Preesm. S-Preesm has been successfully applied to the hierarchical Simulink application "LTE transmitter side". Thanks to our translation strategy, we succeed to transform the complex Simulink application into a deadlock free and consistent IBSDF graph; where we can determine initial amount of tokens for each FiFo channel, consumed and produced data according to communication type between blocks and level of the Simulink model. After transforming the Simulink application, the obtained graph is subject to scheduling/mapping algorithm to perform parallel code generation. In addition, a host C code library corresponding to each graph actor, is created to contribute to the code generation.

Based on the complex Signal processing application "LTE transmitter side", experiments show the effectiveness and the potential of our approach in embedded systems developments. The comparison of our approach results and Simulink coder results demonstrates the efficiency of our technique to perform and facilitate the transformation of hierarchical Simulink applications into multi-core execution.

For further developments, we may extend this work to support other block types such as conditional execution block which is characterized with variable periods. As well as future work , we may also adopt the hierarchical approach proposed in [16] to perform hierarchical Simulink applications mapping into multi-core architecture. We also aim to extend S-Preesm work-flow to support the optimal scheduler proposed by Rebaya et al. [24].

REFERENCES

[1] P. Boström and J. Wiik, Contract-based verification of discrete-time multi-rate Simulink models, Software and Systems Modeling, vol. 15, 1141–1161 (2016).

[2] E. C. Klikpo, J. Khatib, and A. M. Kordon. Modeling multi-periodic simulink systems by synchronous dataflow graphs, in IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS),2016.

[3] Rebaya, Asma, et al. "Workflow for NoC/DSP multicores-based platform: From Matlab/Simulink models to hardware mapping and scheduling." Control, Automation and Diagnosis (ICCAD), 2017 International Conference on. IEEE, 2017.

[4] S. Tripakis, C. Pinello, A. Benveniste, A. Sangiovanni-Vincent, P. Caspi, and M. D. Natale, "Implementing synchronous models on loosely time triggered architectures," IEEE Transactions on Computers, vol. 57, no. 10, 2008.

[5] A. Benveniste and G. Berry, "The synchronous approach to reactive and real-time systems," in Readings in Hardware/Software Co-design. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[6] E. Lee and D. Messerschmitt. Pipeline interleaved programmable dsp's: Synchronous data flow programming. Proceedings of the IEEE, 35(9):1334–1345, Sept. 1987.

[7] Mathworks Inc., Simulink. http://www.mathworks.com (2015)

[8] Markus Herrmannsdoerfer and Stefano Merenda. Result of the Tool Questionnaire. Technical report, Technische Universitat Munchen, Nov 2009

[9] E. A. Lee and D. G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processings. IEEE Transaction on Computers, 36(1):24–35, 1987.

[10] E. A. Lee and D. G. Messerschmitt. Synchronous data flow. Proceedings of the IEEE, 75(9):1235–1245, 1987.

[11] E. Lee and T. Parks, "Dataflow process networks," Proceedings of the IEEE, 1995.

[12] J. Piat, M. Raulet, M. Pelcat, P. Mu, and O. Deforges. An extensible framework for fast prototyping of multiprocessor dataflow applications. In IDT08: Proceedings of the 3rd International Design and Test Workshop, december 2008.

[13] W.Sung,M.Oh,C.Im,andS.Ha. DemonstrationOfCodesign Workflow In PeaCE. In in Proc. of International Conference of VLSI Circuit, 1997.

[14] E.A. Lee and D.G. Messerschmitt. Synchronous data flow. Proceedings of the IEEE, 75(9):1235–1245, 1987. 50, 51

[15] J. Piat, S. Bhattacharyya, and M. Raulet, "Interface-based hierarchy for synchronous data-flow graphs," in SiPS Proceedings, 2009.

[16] J. Hascoët, K. Desnos, J-F. Nezan and B. Dupont de Binechin , "Hierarchical Dataflow Model for Efficient Programming of Clustered Manycore Processors" in IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2017.

[17] H. Deroui, K. Desnos, J-F. Nezan, A. Munier-Kordon, "Throughput Evaluation of DSP Applications based on Hierarchical Dataflow Models " in International Symposium on Circuits and Systems (ISCAS), May 2017.

[18] O. Marchetti and A. M. Kordon, "Cyclic scheduling for the synthesis of embedded systems," Introduction to scheduling, 2009.

[19] HA, S., LEE, C., YI, Y., KWON, S., AND JOO, Y.-P. 2006. Hardwaresoftware codesign of multimedia embedded systems: the PeaCE approach. In Proceedings of the Conference on Embedded and Real Time Computing Systems and Applications (RTCSA). 207–214.

[20] Brunei, S.C., Mattavelli, M., Janneck, J.W.: TURNUS: a design exploration framework for dataflow system design. In: ISCAS 2013

[21] Pelcat, M., Aridhi, S., Piat, J., Nezan, J.F.: Physical Layer Multi-Core Prototyping: A Dataflow-Based Approach for LTE eNodeB. Springer (2012)

[22] https://behindthesciences.com/useful-links/ltesimulinkmodel

[23] Lee, E.A., Parks, T.M.: Dataflow process networks. Proceedings of the IEEE 83(5), 773801 (1995)

[24] A. Rebaya, I. Amari, K. Gasmi and S. Hasnaoui. Core Number Optimization Based Scheduler to Order/Map Hardware/Software Applications. In the 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2017

[25] J. Piat, S. S. Bhattacharyya, and M. Raulet. Interface-based hierarchy for synchronous data-flow graphs. submitted SAMOS conference IX, july 2009. 57, 59

[26] Y.-K. Kwok, High-performance algorithms for compile-time scheduling of parallel processors, PhD. Thesis, 1997

[27] M. Pelcat, P. Menuet, S. Aridhi and J-F. Nezan, Scalable compile-time scheduler for multi-core architectures, DATE 2009, Nice.

[28] S. Moreau, S. Aridhi, M. Raulet, and J.-F. Nezan. On modeling the RapidIO communication link using the AAA methodology. In DASIP, 2007. 140.