Formal Verification of Specification Partitioning

Samar Abdi and Daniel Gajski

Technical Report CECS-03-06
April 23, 2003

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{sabdi,gajski}@cecs.uci.edu

Formal Verification of Specification Partitioning

Samar Abdi and Daniel Gajski

Technical Report CECS-03-06
April 23, 2003

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

{sabdi,gajski}@cecs.uci.edu

Abstract

This report presents a formal approach to verify models in a system level design environment. It is a first in series of reports that demonstrate how we use this formal approach to refine a given specification down to its cycle-accurate implementation. We formally define models and develop theorems and proofs to show that our well defined refinement algorithms produce functionally equivalent models. In this report, we specifically look at generation of an architecture level model by refinement of a specification model. The refinement process follows a well defined system level partitioning algorithm. We prove that executing the individual steps of the refinement algorithm, in the predefined order, leads to an equivalent model.

Contents

List of Figures

Formal Verification of Specification Partitioning

Samar Abdi and Daniel Gajski
Center for Embedded Computer Systems
University of California, Irvine

Abstract

This report presents a formal approach to verify models in a system level design environment. It is a first in series of reports that demonstrate how we use this formal approach to refine a given specification down to its cycle-accurate implementation. We formally define models and develop theorems and proofs to show that our well defined refinement algorithms produce functionally equivalent models. In this report, we specifically look at generation of an architecture level model by refinement of a specification model. The refinement process follows a well defined system level partitioning algorithm. We prove that executing the individual steps of the refinement algorithm, in the predefined order, leads to an equivalent model.

1 Introduction

The continuous increase in behavioral and structural complexity of SoC designs has raised the abstraction level of system specification. The common approach in system design is to write models at different levels of abstraction. However,with the size of these designs, traditional verification and simulation based approaches for validation are no longer practical. Besides, verification by comparing two separately written models is not tractable at the system level.

The only solution is to correctly generate one model from another using a well defined sequence of refinements [2]. The output model is the product of gradual refinements to the input model. Each gradual refinement produces an output model that is functionally equivalent to the input model. Figure 1 shows how a model refinement is broken into a sequence of gradual refinement steps. We must ensure that model at level i is equivalent to the one at level $i-1$. By transitivity this ensures that the final output model is equivalent to the input model. To achieve this, we develop formalisms to describe models at different abstraction levels and perform refinements on them. This report presents a limited set of formalisms useful in the context of system level design partitioning.

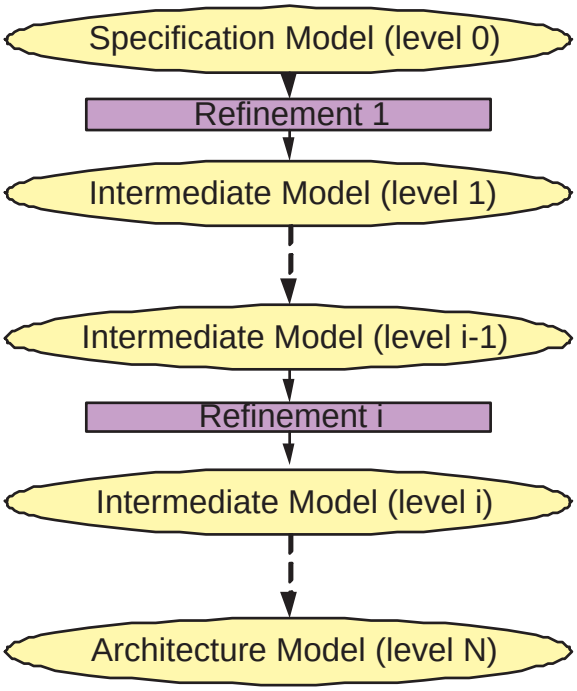

Figure 1. The gradual refinement process.

Figure 2 shows the universal set of system models. This set is divided into classes of equivalent models. As shown, a specification model m_s and its corresponding implementation m_i belong to the same equivalence class. There may be several implementations for the same specification and they would all belong to the same equivalence class. We must ensure that when a model at abstraction level j is refined to the one at level $j+1$, then model m_{j+1} must belong to the same equivalence class as model m_j. In other words, the refinement must be **contained** in the equivalence class.

This report is a first in series of reports on formal verification of system level model refinements. Here, we focus on the behavioral partitioning of a specification model to derive an architecture level model. The report is divided as follows. We begin by introducing the model algebra in the next section. This includes a formal definition of a model in

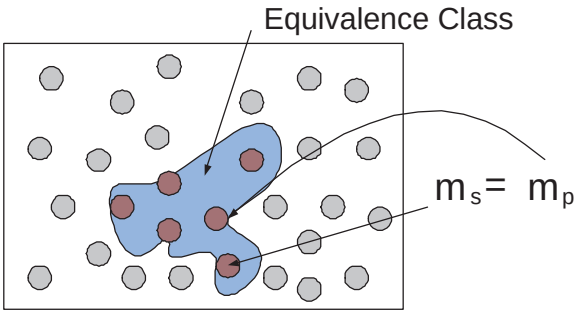

Figure 2. The universal set of system models.

terms of the model algebra in Section 2. We also present the basic axioms associated with this algebra. We then present the partitioning refinement algorithm in Section 3. Next, we prove some useful theorems. The final theorem uses the axioms of Model Algebra and these theorems to prove that the partitioned architecture model and specification model are equivalent. The model in this report has been simplified to demonstrate the concept. The methods used are completely scalable and can be used for large models as well. Finally, we wind up with a summary and conclusion.

2 Model Algebra

For proving correctness of model refinements, we need to define a model algebra that can be used to formally represent system models. Generally speaking, a system is a set of tasks that are executed in a predefined partial order. These tasks also talk to each other by exchanging data. In order to develop an algebra for system models, we must introduce primitives to represent tasks and the data transactions amongst them.

The first primitive is the unit of computation in a model, referred to as a *behavior*. Behaviors that can either be *leaf* or *composite*. A *leaf behavior* is a sequence of operations being executed in a serial order. A *composite behavior* on the other hand is formed by combination of *leaf behaviors* using operations of the model algebra. A model is constructed out of these leaf behaviors by using the basic concept of hierarchy and composition operations. Two or more leaf behaviors are put together to compose a composite behavior. The composite behaviors may also be combined with other leaf or composite behaviors to generate larger composite behaviors. In the scope of this report, the composition may be sequential or parallel. Moreover, we need synchronization between behaviors to ensure the correct temporal order of execution.

The other primitive is the unit of computation referred to as a channel. It is used for communication between behaviors. A channel encapsulates the data item to be transferred and events to ensure the transfer semantics. The channels we consider for the purpose of this report are two way blocking channels as shown in figure 3. As we can see, the behaviors b_1 and b_2 use the channel c to exchange *DATA*. The transfer semantics ensure that the receiving behavior will wait until the sender has written the data, and the sender behavior waits until the data item has been read by the receiver.

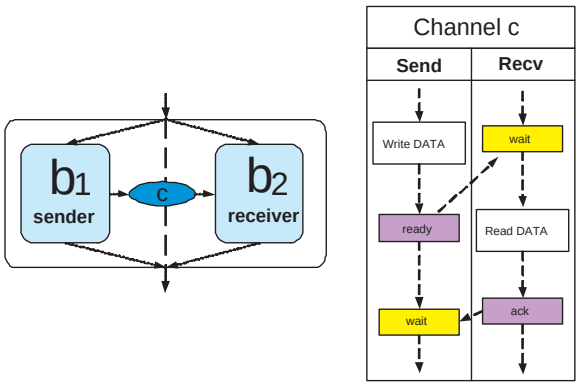

Figure 3. A two way blocking channel

The Model Algebra is defined as:

$$A =< B, C, O, R >$$

B is the set of behaviors,
C is the set of channels,
$O = \{seq, par\}$ (Set of Operations)
$R = \{\rightsquigarrow, \overset{v}{\rightarrow}\}$(Set of Relations)

Operations

The operations mentioned in the above algebra are defined on elements in B. The set B is closed with respect to both *seq* and *par*.

$$\forall b_1, b_2, b_3, \ldots \in B$$

1. $seq(b_1, b_2, b_3, \ldots) \in B$
2. $par(b_1, b_2, b_3, \ldots) \in B$

The *seq* operator implies that behaviors execute sequentially in time. Hence if $b = seq(b_1, b_2, \ldots, b_n)$, then b_2 starts after b_1 has completed, b_3 starts after b_2 has completed and so on. Behavior b is said to start when b_1 starts and it completes when b_n completes. Note that the composite behavior b can be used to create more composite behaviors.

The *par* operator creates a parallel composition of behaviors. If $b = par(b_1, b_2, \ldots, b_n)$, then there is no predefined order of execution between the behaviors $b_1, b_2, \ldots, b_n$. Behavior b is said to start when any behavior in b_1 through b_n

starts. Behavior b is said to complete when all behaviors from b_1 through b_n complete.

Composite behaviors are essentially functions formed using operators *seq* and *par* on behaviors. We will use the notation $f(b_1, b_2, ..., b_n)$ to represent a composite behavior formed using behaviors b_1 through b_n.

Relations

We define the *synchronization* relation on B in the following way. $\forall b_1, b_2 \in B$, if $b_1 \rightsquigarrow b_2$ then irrespective of the hierarchical composition, behavior b_2 cannot start executing until behavior b_1 completes.

Data transfers in a system can take place either through variables or channels. Sequentially composed behaviors communicate through variables, while those composed in parallel use channels. In the latter case, data from the sender behavior is written to the channel. Subsequently, the receiver behavior reads the data from the channel. We define three kinds of *data transfer* relations as follows.

1. Data variable v sent from behavior b_1 to behavior b_1

$$b_1 \xrightarrow{v} b_2, \text{ where } (b_1, b_2) \in B \times B$$

2. Data variable v sent from behavior b_1 to channel c_1

$$b_1 \xrightarrow{v} c_1, \text{ where } (b_1, c_1) \in B \times C$$

3. Data variable v received by behavior b_1 from channel c_1

$$c_1 \xrightarrow{v} b_1, \text{ where } (c_1, b_1) \in C \times B$$

Class of Identity Behaviors

We define the class of *Identity Behaviors* I to be a subset of B such that all behaviors belonging to I output the input data. The following behaviors belong to the class I :

- Behavior(in var v_1, out var v_2)
  { $v_2 = v_1$ }
- Behavior(in var v, out channel c)
  { c.write(v) }
- Behavior(in channel c, out var v)
  { $v = c$.read() }
- Behavior(in channel c_1, out channel c_2)
  { $c_2.write(c_1.read())$ }

Note that an identity behavior does not perform any other operation except reading and writing a data item.

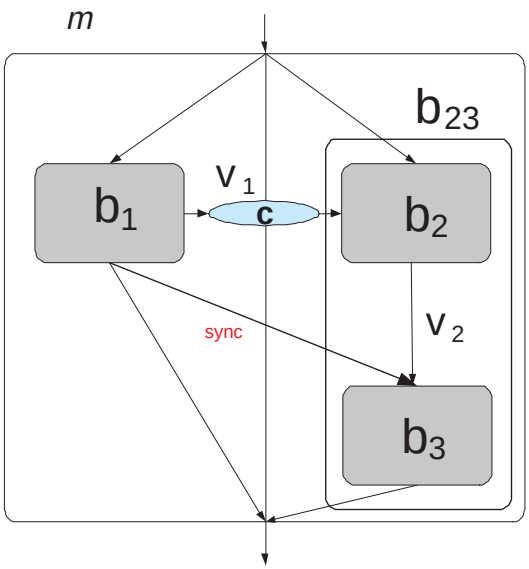

Figure 4. A simple model

2.1 Model Definition

Based on the above algebra, a system model m can be defined as a tuple

$$m = B(m), R(m), \text{where}$$

B(m) : the hierarchical composition of behaviors representing the model m, and

R(m) : set of relations on the behaviors used to compose B(m)

Figure 4 shows a simple model comprising of three *leaf* behaviors b_1, b_2 and b_3. Behaviors b_2 and b_3 are sequentially composed to create a *composite* behavior b_23, which in turn is composed in parallel with behavior b_1. Channel c is used to send data v_1 from b_1 to b_2. Variable v_2 is sent directly from behavior b_2 to b_3 since they are in sequential composition. Also note the synchronization edge from b_1 to b_3 which guarantees that b_3 cannot start executing before b_1 completes, despite their parallel composition. The model m can be written as follows:

$$\begin{aligned} B(m) &= par(b_1, seq(b_2, b_3)), \\ R(m) &= \{b_1 \rightsquigarrow b_3, b_1 \xrightarrow{v_1} c, c \xrightarrow{v_1} b_2, b_2 \xrightarrow{v_2} b_3\} \end{aligned}$$

2.1.1 Terms and definitions

For the purpose of explaining and formally proving correctness of model refinements, we need to introduce some notations.

Sub-behavior The relation *sub-behavior* is defined on B as follows

$$\forall b_1, b_2 \in B, \text{ if } b_1 \triangleleft b_2, \text{ then}$$

b_1 is a sub-expression in the hierarchical expression of b_2.
Example from Figure 4:
$b_{23} = seq(b_2, b_3)$, $B(m) = par(b_1, seq(b_2, b_3))$,
therefore $b_{23} \triangleleft B(m)$.

Leafs This is the set of all leaf level sub-behaviors of a behavior.

$$Leafs(b) = \{x | x \triangleleft b, \nexists y \triangleleft b \text{ s.t. } y \triangleleft x\}$$

Example from Figure 4:

1. $Leafs(b_23) = \{b_2, b_3\}$
2. $Leafs(B(m)) = \{b_1, b_2, b_3\}$

Predecessor A behavior b_1 is said to be a *predecessor* of behavior b_2 in model m (denoted by $b_1 \overset{m}{<} b_2$), if in the temporal order of execution b_1 must complete before b_2 begins. Formally, the predecessor relation can be defined as:
$\forall b_1, b_2, b_3 \triangleleft B(m)$

1. $seq(.., b_1, b_2, ..) \triangleleft B(m) \Rightarrow b_1 \overset{m}{<} b_2$
2. $b_1 \rightsquigarrow b_2 \in R(m) \Rightarrow b_1 \overset{m}{<} b_2$
3. $b_1 \overset{m}{<} b_2 \wedge b_3 \triangleleft b_2 \Rightarrow b_1 \overset{m}{<} b_3$
4. $b_1 \overset{m}{<} b_2 \wedge b_3 \triangleleft b_1 \Rightarrow b_3 \overset{m}{<} b_2$
5. $b_1 \overset{m}{<} b_2 \wedge b_2 \overset{m}{<} b_3 \Rightarrow b_1 \overset{m}{<} b_3$

Example from Figure 4:

1. $b_1 \overset{m}{<} b_3$
2. $b_2 \overset{m}{<} b_3$

Immediate Predecessor A behavior b_1 is said to be an *immediate predecessor* of behavior b_2, in a model m (denoted by $b_1 \overset{m}{\ll} b_2$) if

$$\nexists b_3 \triangleleft B(m), b_3 \in B, \text{ such that } b_1 \overset{m}{<} b_3 \overset{m}{<} b_2$$

Example from Figure 4:

1. $b_1 \overset{m}{\ll} b_3$
2. $b_2 \overset{m}{\ll} b_3$

2.1.2 Axioms

Now that we have established the basic building blocks of the system model, we need to define a set of axioms that are associated with the model algebra. These axioms will be used to construct theorems that will validate model refinements and transformations.

Axiom 1 (Synchronization) *A sequential composition of behaviors b_1, b_2 in a model M may be replaced by a parallel composition of b_1, b_2 by adding a synchronization relation $b_1 \rightsquigarrow b_2$ in R(m).*

$$f(seq(b_1, b_2), b3, b4...), R(m) =$$
$$f(par(b_1, b_2), b3, b4...), R(m) \cup b_1 \rightsquigarrow b_2$$

Axiom 2 (Flattening) *If a behavior x, in model M, is composite and parent of x is the same composite type as x, and x does not have any synchronization constraints, then x may be removed through flattening.*

Sub-Axiom 2.1 *For the* seq *behavior:*

$$if\ x = seq(b_{i+1}, b_{i+2}, ...b_j)$$
$$b = seq(b_1, b_2...b_i, x, b_{j+1}, b_{j+2}...b_k) \triangleleft B(m), \text{ and}$$
$$\nexists a \triangleleft B(m), \text{ such that}$$
$$a \rightsquigarrow x \in R(m) \text{ or } x \rightsquigarrow a \in R(m), \text{ then}$$
$$b = seq(b_1, b_2, ..., b_i, b_{i+1}, b_{i+2}, ..., b_j, b_{j+1}, b_{j+2}, ..., b_k)$$

Sub-Axiom 2.2 *The dual for* par *behavior is as follows:*

$$if\ x = par(b_{i+1}, b_{i+2}, ...b_j)$$
$$b = par(b_1, b_2...b_i, x, b_{j+1}, b_{j+2}...b_k) \triangleleft B(m), \text{ and}$$
$$\nexists a \triangleleft B(m), \text{ such that}$$
$$a \rightsquigarrow x \in R(m) \text{ or } x \rightsquigarrow a \in R(m), \text{ then}$$
$$b = par(b_1, b_2, ..., b_i, b_{i+1}, b_{i+2}, ..., b_j, b_{j+1}, b_{j+2}, ..., b_k)$$

Axiom 3 (Forward Substitution) *Synchronization relation for composite behaviors may be replaced by synchronization relation(s) on their child behaviors.*

Sub-Axiom 3.1 *If a behavior b is a sequential composition such that $b = seq(b_1, b_2..., b_n)$ and if there exists a synchronization constraint from a behavior $a \rightsquigarrow b$, then the constraint may be replaced by a synchronization constraint $a \rightsquigarrow b_1$. Similarly, a constraint $b \rightsquigarrow n$ may be replaced by $b_n \rightsquigarrow n$*

$$if\ b = seq(b_1, b_2, ..., b_n) \triangleleft B(m), \text{ then}$$
$$\forall a \triangleleft B(m), \text{ if } a \rightsquigarrow b \in R(m)$$
$$R(m) = (R(m) - a \rightsquigarrow x) \cup a \rightsquigarrow b_1$$

$$\forall a \triangleleft B(m), \text{ if } b \rightsquigarrow a \in R(m)$$
$$R(m) = (R(m) - b \rightsquigarrow a) \cup b_n \rightsquigarrow a$$

Sub-Axiom 3.2 *If a behavior x is a parallel composition such that* $b = par(b_1, b_2 \ldots, b_n)$ *and if there exists a synchronization constraint from a behavior* $a \rightsquigarrow b$*, then the constraint may be replaced by synchronization constraints* $a \rightsquigarrow b_1, a \rightsquigarrow b2, \ldots, a \rightsquigarrow b_n$*. Similarly, a constraint* $b \rightsquigarrow a$ *may be replaced by constraints* $b_1 \rightsquigarrow a, b_2 \rightsquigarrow a, \ldots b_n \rightsquigarrow a$

$$if\ b = par(b_1, b_2, \ldots, b_n) \triangleleft B(m),\ then$$
$$\forall a \triangleleft B(m),\ if\ a \rightsquigarrow b \in R(m),$$
$$R(m) = (R(m) - a \rightsquigarrow b) \cup a \rightsquigarrow b_1, a \rightsquigarrow b_2, \ldots, a \rightsquigarrow b_n$$
$$\forall a \triangleleft B(m),\ if\ b \rightsquigarrow a \in R(m),$$
$$R(m) = (R(m) - b \rightsquigarrow a) \cup b_1 \rightsquigarrow a, b_2 \rightsquigarrow a, \ldots b_n \rightsquigarrow a.$$

Axiom 4 (Commutativity) *A parallel composition of the type* $par(b_1, b_2)$ *is equivalent to* $par(b_2, b_1)$.

$$par(b_1, b_2) = par(b_2, b_1)$$

.

Axiom 5 (Identity) *Given a model m and behavior* $e \in I$ *such that e does* not *have any relations in* $R(m)$*, the following hold true* $\forall b \triangleleft B(m)$

1. $seq(b, e) = b$
2. $seq(e, b) = b$
3. $par(b, e) = b$

Axiom 6 (Transitivity) *Given a model m and* $b_1, b_2, b3 \triangleleft B(m)$

$$b_1 \xrightarrow{v} b_2 = \{b_1 \xrightarrow{v} b_3, b_3 \xrightarrow{v} b_2\}\ iff\ b_1 \overset{m}{\leq} b_3 \wedge b_3 \overset{m}{\leq} b_2$$

Axiom 7 (Channel creation) *Given identity behaviors* $e_1, e_2 \in I$ *and channel* $c \in C$

Sub-Axiom 7.1

$$\{e_1 \xrightarrow{v} e_2, e_1 \rightsquigarrow e_2\} = \{e_1 \xrightarrow{v} c, c \xrightarrow{v} e_2\}$$

Sub-Axiom 7.2

$$\{e_1 \rightsquigarrow e_2\} = \{e_1 \xrightarrow{0} c, c \xrightarrow{0} e_2\}$$

3 System Level Partitioning

This is the first step in deriving the architecture model from a given specification [1]. Once we have determined the components in the proposed system architecture, we need to divide the system tasks into suitable groups. Each of these groups is assigned to a unique component in the architecture. After this assignment, we need to evaluate the correctness and suitability of our partition. For this purpose, we need to generate an executable partitioned model. The model may be generated automatically from the specification, once we have the partitioning decisions. This process is known as partitioning refinement [3]. In this section, we present the partitioning refinement algorithm using our model algebra. Subsequently, we develop a theorem and prove it to show that this algorithm indeed works correctly.

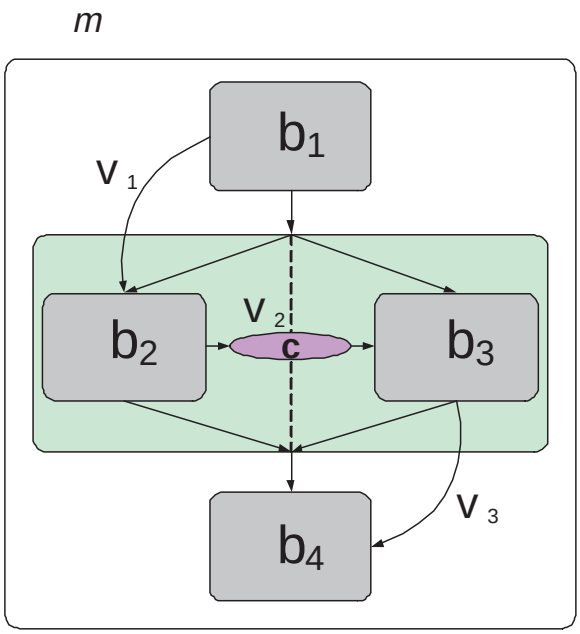

Figure 5. A simple specification model.

3.1 Partitioning refinement algorithm

Given a model $m = B(m), R(m)$, a set of n components $PE_1, PE_2, \ldots, PE_n$ and n partitions $partition_1, partition_2, \ldots, partition_n$. Each partition is a set of leaf behaviors in m. The partitions follow the following rules:

1. $\bigcup_{i=1}^{n} partition_i = Leafs(B(m)),$ and
2. $partition_i \cap partition_j = \phi, 1 \leq i, j \leq n$
3. $partition_i$ is assigned to PE_i

The partitioned model m_p is generated as follows.

1. Initialize B(m_p) as a parallel composition of n behaviors PE_1 through PE_n such that $PE_i = f(b_{i1}, b_i i2, \ldots, b_{im})$, where

$$b_{ij} = \begin{cases} b_j & : \quad b_j \in partition_i \\ \phi & : \quad otherwise \end{cases}$$

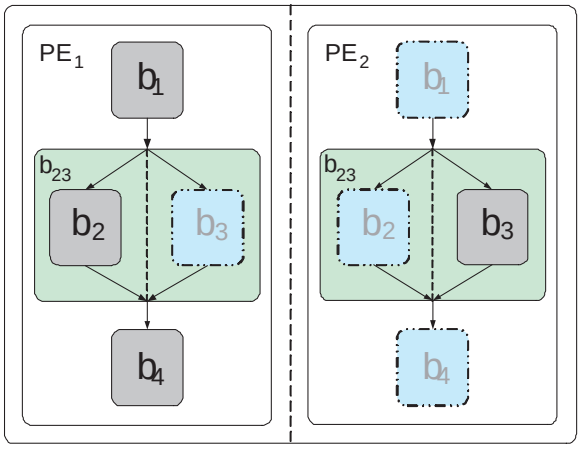

Figure 6. Intermediate model after step 1 of partitioning algorithm.

2. Add following synchronization relations
$\bigcup_{i=1}^{n}\{x \rightsquigarrow y | y \in partition_i \wedge x \overset{m}{\ll} y \wedge$
$x \in (Leafs(B(m)) - partition_i)\}$

3. Introduce identity behaviors for each synchronization relation as follows
$\forall b_1, b_2 \triangleleft B(m)$, such that $b_1 \rightsquigarrow b_2 \in R(m), e_1, e_2 \in I$
replace behavior b_1 with $seq(b_1, e_1)$ and b_2 with $seq(e_2, b_2)$ and modify R(m) as follows

   (a) if $\exists b_1 \overset{v}{\rightarrow} b_2 \in R(m)$, then $R(m) = (R(m) - \{b_1 \overset{v}{\rightarrow} b_2, b_1 \rightsquigarrow b_2\}) \cup \{b_1 \overset{v}{\rightarrow} e_1, e_1 \overset{v}{\rightarrow} c_1, c_1 \overset{v}{\rightarrow} e_2, e_2 \overset{v}{\rightarrow} b_2\}$

   (b) else $R(m) = (R(m) - \{b_1 \rightsquigarrow b_2\}) \cup \{e_1 \overset{0}{\rightarrow} c_1, c_1 \overset{0}{\rightarrow} e_2\}$

4. Flatten redundant hierarchy.

An example of the refinement process is demonstrated on a simple specification model. The specification, comprising of four leaf behaviors viz. b_1, b_2, b_3 and b_4 is shown in figure 5. Leaf behaviors b_2 and b_3 are composed in parallel to form a composite behavior. This composite behavior follows b_1 and precedes b_4 in a larger sequential composition. Also note the data transactions between the behaviors. Data item v_1 is sent from behaviors b_1 to b_2. Similarly v_3 is sent from b_3 to b_4. Since b_2 and b_3 are composed in parallel, they talk using the channel c, which sends data item v_2 from b_2 to b_3. The model m may be expressed in our algebra as follows:

$$\begin{aligned} B(m) &= seq(b_1, par(b_2, b_3), b_4), \\ R(m) &= \{b_1 \overset{v_1}{\rightarrow} b_2, b_2 \overset{v_2}{\rightarrow} c, c \overset{v_2}{\rightarrow} b_3, b_3 \overset{v_3}{\rightarrow} b_4\} \end{aligned}$$

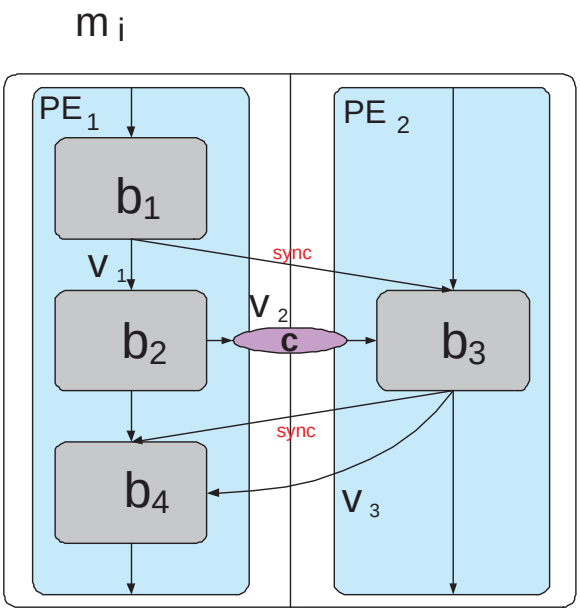

Figure 7. Intermediate model after step 2 of partitioning algorithm.

The partitioning decision is as follows.
$partition_1 = b_1, b_2, b_4$ to PE_1
$partition_2 = b_3$ to PE_2

After partitioning, the model is refined to a parallel composition of PE_1 and PE_2. After step 1 the intermediate model is shown in figure 6. The relevant leaf behaviors are copied into each of the PEs in the original hierarchy. After step 2 of the partitioning algorithm, we derive an intermediate model m_i. Synchronization constraints are added across the partitions to maintain the original partial order of execution. $b_1 \overset{m}{\ll} b_3$ and $b_3 \overset{m}{\ll} b_4$, are the only immediate predecessor relations across partitions. Therefore, we add corresponding synchronization constraints i.e. $b_1 \rightsquigarrow b_3$ and $b_3 \rightsquigarrow b_4$ to derive intermediate model m_i as shown in figure 7.

$$\begin{aligned} B(m_i) &= par(seq(b_1, b_2, b_4), seq(b_3)), \\ R(m_i) &= \{b_1 \overset{v_1}{\rightarrow} b_2, b_2 \overset{v_2}{\rightarrow} c, c \overset{v_2}{\rightarrow} b_3, b_3 \overset{v_3}{\rightarrow} b_4, \\ &\quad b_1 \rightsquigarrow b_3, b_3 \rightsquigarrow b_4\} \end{aligned}$$

The final model m_p is shown in Figure 8. This model is derived by executing step 3 and 4 of our algorithm on model m_i. Identity behaviors e_1, e_2, e_3 and e_4 and channels c_1 and c_2 are inserted corresponding to synchronization relations $b_1 \rightsquigarrow b_3$ and $b_3 \rightsquigarrow b_4$. The pair of relations $\{b_3 \rightsquigarrow b_4, b_3 \overset{v_3}{\rightarrow} b_4\}$ in m_i is replaced by $\{b_3 \overset{v_3}{\rightarrow} e_3, e_3 \overset{v_3}{\rightarrow} c_2, c_2 \overset{v_3}{\rightarrow} e_4, e_4 \overset{v_3}{\rightarrow} b_4\}$ in m_p. Similarly, channel c_1 is introduced to implement

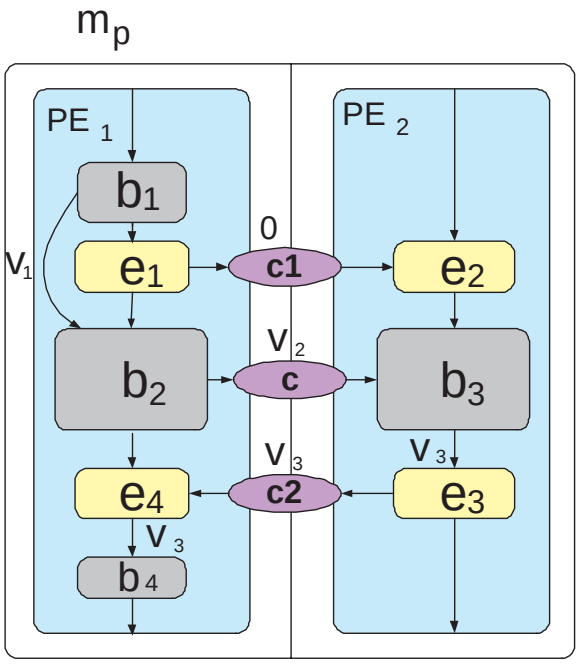

Figure 8. Final model after partitioning.

the synchronization relation between b_1 and b_3. The final partitioned model m_p can be expressed as:

$$\begin{aligned} B(m_i) &= par(seq(b_1,e_1,b_2,e_4,b_4), \\ &\quad seq(e_2,b_3,e_3)), \\ R(m_i) &= \{b_1 \xrightarrow{v_1} b_2, b_2 \xrightarrow{v_2} c, c \xrightarrow{v_2} b_3, b_3 \xrightarrow{v_3} b_4, \\ &\quad e_1 \xrightarrow{0} c_1, c_1 \xrightarrow{0} e_2, b_3 \xrightarrow{v_3} e_3, e_3 \xrightarrow{v_3} c_2, \\ &\quad c_2 \xrightarrow{v_3} e_4, e_4 \xrightarrow{v_3} b_4\} \end{aligned}$$

3.2 Theorems

The axioms in Section 2 establish the basic properties of a model. We now focus on developing some useful theorems from these axioms. The theorems in turn will be employed to prove the correctness of model refinement algorithms. In particular, we are trying to prove that our refinement algorithm for partitioning a specification model is correct. The theorems in this sub-section will help in proving that the model obtained after partitioning is *equivalent* to the specification.

Theorem 1 (Expression Exchange) *A sequential composition of the type $seq(b_1,b_2,...b_i)$ in a given model m, may be replaced by a parallel composition of the type $par(b_1,b_2,...b_i)$ by adding the synchronization constraints $b_1 \rightsquigarrow b_2, b_2 \rightsquigarrow b_3, ...b_{i-1} \rightsquigarrow b_i$ to R(m).*
$f(seq(b_1,b_2,...b_i),b_{i+1},b_{i+2}..),R(m) =$
$f(par(b_1,b_2,...b_i),b_{i+1},b_{i+2}..),$
$R(m) \cup \{b_1 \rightsquigarrow b_2, b_2 \rightsquigarrow b_3,...,b_{i-1} \rightsquigarrow b_i\}$

Proof:
We prove this theorem by mathematical induction.
$m = f(seq(b_1,b_2,...b_i),b_{i+1},b_{i+2}..),R(m)$
for i = 3, we have,

$$\begin{aligned} m &= f(seq(b_1,b_2,b_3),b_4,...),R(m) \\ &= f(seq(seq(b_1,b_2),b_3),b_4,...),R(m) \\ &\quad \text{using axiom 2} \\ &= f(par(seq(b_1,b_2),b_3),b_4,...), \\ &\quad R(m) \cup seq(b_1,b_2) \rightsquigarrow b_3 \\ &\quad \text{using axiom 1} \\ &= f(par(seq(b_1,b_2),b_3),b_4,...), \\ &\quad R(m) \cup b_2 \rightsquigarrow b_3 \\ &\quad \text{using axiom 3} \\ &= f(par(par(b_1,b_2),b_3),b_4,...), \\ &\quad R(m) \cup b_2 \rightsquigarrow b_3 \cup b_1 \rightsquigarrow b_2 \\ &\quad \text{using axiom 1} \\ &= f(par(b_1,b_2,b_3),b_4,...), \\ &\quad R(m) \cup b_1 \rightsquigarrow b_2, b_2 \rightsquigarrow b_3 \\ &\quad \text{using axiom 2} \end{aligned}$$

So, the theorem is proved for $i = 3$. By principle of induction, let us assume that the theorem is true for integer $i = N, N > 3$. We have to prove that the theorem holds true for $i = N+1$ also.

for i = N+1, we have

$$\begin{aligned} m &= f(seq(b_1,b_2,...,b_N,b_{N+1}),b_{N+2},...),R(m) \\ &= f(seq(seq(b_1,b_2,...,b_N),b_{N+1}),b_{N+2},...),R(m) \\ &\quad \text{using axiom 2} \\ &= f(par(seq(b_1,b_2,...,b_N),b_{N+1}),b_{N+2},...), \\ &\quad R(m) \cup \{seq(b_1,b_2,...,b_N) \rightsquigarrow b_{N+1}\} \\ &\quad \text{using axiom 1} \\ &= f(par(seq(b_1,b_2,...,b_N),b_{N+1}),b_{N+2},...), \\ &\quad R(m) \cup \{b_N \rightsquigarrow b_{N+1}\} \\ &\quad \text{using axiom 3} \\ &= f(par(par(b_1,b_2,...,b_N),b_{N+1}),b_{N+2},...), \\ &\quad R(m) \cup \{b_N \rightsquigarrow b_{N+1}\} \\ &\quad \cup \{b_1 \rightsquigarrow b_2, b_2 \rightsquigarrow b_3, \cdots, b_{N-1} \rightsquigarrow b_N\} \\ &\quad \text{using assumption for i = N} \\ &= f(par(b_1,b_2,...,b_N,b_{N+1}),b_{N+2},...), \\ &\quad R(m) \cup \{b_1 \rightsquigarrow b_2, b_2 \rightsquigarrow b_3, \cdots \\ &\quad \cdots, b_{N-1} \rightsquigarrow b_N, b_N \rightsquigarrow b_{N+1}\} \\ &\quad \text{using axiom 2} \end{aligned}$$

Hence proved.

Theorem 2 (Permutation of parallel behaviors) *A parallel composition of the type $par(b_1, b_2, ...b_i)$ in a given model m, may be replaced by a parallel composition of behaviors b_1 through b_i in any permutation.*

This amounts to proving that

$$par(b_1, b_2, .., b_x, b_{x+1}, ..., b_y, b_{y+1}, ...b_i) = par(b_1, b_2, ..., b_y, b_{x+1}, ..., b_x, b_{y+1}, ...b_i), \text{ where } x \neq y, 1 \leq x, y \leq i$$

Proof:
Without loss of generality, let us assume $x < y$ Let $y = x + n, n \geq 1$. We have

$$\begin{aligned} b &= par(b_1, b_2, ..., b_x, b_{x+1}, ..., b_{x+n-1}, b_y, ..., b_i) \\ &= par(b_1, b_2, ..., b_x, b_{x+1}, ..., par(b_{x+n-1}, b_y), ..., b_i) \\ &= \text{using axiom 2} \\ &= par(b_1, b_2, ..., b_x, b_{x+1}, ..., par(b_y, b_{x+n-1}), ..., b_i) \\ &\quad \text{using axiom 4} \\ &= par(b_1, b_2, ..., b_x, b_{x+1}, ..., b_y, b_{x+n-1}, ..., b_i) \\ &\quad \text{using axiom 2} \end{aligned}$$

Using n iterations of the above three steps and moving b_y to the left, we get

$$\begin{aligned} b &= par(b_1, b_2, ..., b_y, b_x, b_{x+1}, ..., b_{x+n-1}, ..., b_i) \\ &= par(b_1, b_2, ..., b_y, par(b_x, b_{x+1}), ..., b_{x+n-1}, ..., b_i) \\ &\quad \text{using axiom 2} \\ &= par(b_1, b_2, ..., b_y, par(b_{x+1}, b_x), ..., b_{x+n-1}, ..., b_i) \\ &\quad \text{using axiom 4} \\ &= par(b_1, b_2, ..., b_y, b_{x+1}, b_x, ..., b_{x+n-1}, ..., b_i) \\ &\quad \text{using axiom 2} \end{aligned}$$

Performing n iterations of the above three steps and moving b_x to the right, we get
$b = par(b_1, b_2, ..., b_y, b_{x+1}, ..., b_x, b_{y+1}, ...b_i)$
Hence proved.

Theorem 3 (Redundant Synchronization) *Given model m, if behavior b_1 is a* predecessor, *but* **not** *an* immediate predecessor, *of behavior b_2, then the synchronization relation $b_1 \rightsquigarrow b_2$ is redundant in m.*

$$b_1 \overset{m}{<} b_2 \wedge b_1 \overset{m}{\not\ll} b_2 \Rightarrow R(m) = R(m) - \{b_1 \rightsquigarrow b2\}$$

Proof:
By definition of immediate predecessor,
$b_1 \overset{m}{<} b_2 \wedge b_1 \overset{m}{\not\ll} b_2$
$\Rightarrow \exists a \triangleleft B(m)$, such that $b_1 \overset{m}{<} a \wedge a \overset{m}{<} b_2$

The condition $b_1 \overset{m}{<} b_2$ is satisfied by clause 5 of the predecessor definition, which renders the relation $b_1 \rightsquigarrow b_2$ redundant. Therefore $R(m) = R(m) - \{b_1 \rightsquigarrow b2\}$.

Theorem 4 (Canonical form) *Any system model m is equivalent to a canonical model m', which is a parallel composition of all leaf behaviors in m, and each leaf behavior in m' has a synchronization constraint from all its immediate predecessors in m.*

$$\begin{aligned} B(m') &= par(Leafs(B(m))), \\ R(m') &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\ &\quad \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x, y \in Leafs(B(m))\} \end{aligned}$$

Proof:
We start with a model m of the system. In order to convert m to its canonical form m',we perform the following steps.

1. Convert all *seq* behaviors to *par* and add synchronization relations using theorem 1.
2. Recursively substitute synchronization relations between composite behaviors with synchronization relations between their child behaviors using axiom 3 to derive model m_1.
3. Remove all redundant synchronization relations in m_1 using theorem 3.
4. Flatten the hierarchy in m_1 using axiom 2 to get the canonical model m'

It is easy to see that $m = m_1 = m'$ since each of the above steps follows directly from an established theorem or axiom.

We start with showing that

$$\begin{aligned} R(m_1) &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\ &\quad \{x \rightsquigarrow y | x \overset{m}{<} y \wedge x, y \in Leafs(B(m))\} \end{aligned}$$

This means that we have to prove that

$$\forall b_1, b_2 \in Leafs(B(m)), b_1 \overset{m}{<} b_2 \Leftrightarrow b_1 \rightsquigarrow b_2 \in R(m_1)$$

The proof is divided in two parts.
Assertion 1:
$\forall b_1, b_2 \in Leafs(B(m)), b_1 \overset{m}{<} b_2 \Rightarrow b_1 \rightsquigarrow b_2 \in R(m_1)$

Consider the clauses of the predecessor definition in Section 2. The definition is *inductive* with two terminal cases. If $b_1 \overset{m}{<} b_2$, then one of the following is true:

1. $seq(...b_1, b_2...) \triangleleft B(m)$. In this case step 1 of the above process would convert $seq(...b_1, b_2...)$ to

$par(...b_1, b_2...)$ and add the synchronization relation $b_1 \rightsquigarrow b_2$. If $b_1, b_2 \in Leafs(B(m))$, then this relation would not be modified. Thus **Assertion1** is true in this case.

2. $b_1 \rightsquigarrow b_2 \in R(m)$. Existing relations between *leaf behaviors* are not modified in the conversion process, thus $b_1 \rightsquigarrow b_2 \in R(m_1)$

3. $b_1 \overset{m}{<} b_3 \wedge b_2 \triangleleft b_3$. This is an inductive definition, so if we need to show that if $b_1 \rightsquigarrow b_3$ was added to $R(m)$ during the conversion process and $b_1, b_2 \in Leafs(B(m))$, then $b_1 \rightsquigarrow b_2 \in R(m_1)$ is true. Since all composite behaviors in the model m are converted to parallel before step 2, the substitution of synchronization relations in step 2 replaces $b_1 \overset{m}{<} b_3$ with $b_1 \overset{m}{<} b_2$ if $b_2 \triangleleft b_3$. Since synchronization relations between *leaf behaviors* cannot be substituted, we have $b_1 \rightsquigarrow b_2 \in R(m_1)$. Thus **Assertion 1** is true in this case by inductive reasoning.

4. $b_3 \overset{m}{<} b_2 \wedge b_1 \triangleleft b_3$. We use the same inductive reasoning as above to show that the relation $b_3 \rightsquigarrow b_2$ is replaced with $b_1 \rightsquigarrow b_2$ during the conversion process.

5. $b_1 \overset{m}{<} b_3 \wedge b_3 \overset{m}{<} b_2$. As above, we assume that if $b_1 \rightsquigarrow b_3$ and $b_3 \rightsquigarrow b_2$ are added to $R(m)$ during the conversion process, then the redundant synchronization relation $b_1 \rightsquigarrow b_2$ may be added using theorem 3. Thus $R(m_2) = R(m_2) \cup b_1 \rightsquigarrow b_2$ without changing the model.

We have shown that in each of the above cases, **Assertion 1** holds true.

Assertion 2:

$\forall b_1, b_2 \in Leafs(B(m)), b_1 \rightsquigarrow b_2 \in R(m_1) \Rightarrow b_1 \overset{m}{<} b_2$

A synchronization relation $b_1 \rightsquigarrow b_2$ is added to the model m under the following cases.

1. $seq(...b_1, b_2...) \triangleleft B(m)$. is converted to $par(...b_1, b_2...)$. Clearly $b_1 \overset{m}{<} b_2$. Thus **Assertion1** is true in this case.

2. $b_1 \rightsquigarrow b_3 \wedge b_2 \triangleleft b_3$. Using inductive reasoning, if $b_1 \rightsquigarrow b_3 \rightarrow b_1 \overset{m}{<} b_3$, then $b_1 \overset{m}{<} b_2$ by clause 3 of the predecessor definition.

3. $b_3 \rightsquigarrow b_2 \wedge b_1 \triangleleft b_3$. As above, if $b_3 \rightsquigarrow b_2 \rightarrow b_3 \overset{m}{<} b_2$, then $b_1 \overset{m}{<} b_2$ by clause 4 of the predecessor definition.

We have shown that in each of the above cases, **Assertion 2** holds true. Let,

$$
\begin{aligned}
pred(m_1) &= \{x \rightsquigarrow y | x \overset{m}{<} y \wedge x, y \in Leafs(B(m))\}, \\
impred(m_1) &= \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x, y \in Leafs(B(m))\}. \\
redund(m_1) &= \{x \rightsquigarrow y | x \overset{m}{<} y \wedge x \overset{m}{\not\ll} y \wedge x, y \in Leafs(B(m))\} \\
&= pred(m) - impred(m), \\
&\quad \text{since } impred(m) \subset pred(m).
\end{aligned}
$$

During the conversion process, all synchronization relations between *composite behaviors* are substituted with those between leaf behaviors. The synchronization relations between leaf behaviors of model m are still retained in $R(m_1)$. Using **Assertion 1** and **Assertion 2**, we have

$$
\begin{aligned}
R(m_1) &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m) \wedge \\
&\quad x, y \notin Leafs(B(m))\}) \cup pred(m_1) \\
&= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup pred(m_1), \\
&\quad \text{since } \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m) \wedge \\
&\quad x, y \in Leafs(B(m))\} \subset pred(m_1) \\
R(m') &= R(m_1) - redund(m_1) \\
&\quad \text{by step 4 of the conversion process} \\
&= ((R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup pred(m_1)) \\
&\quad - redund(m_1), \text{using theorem 3} \\
&= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\
&\quad (pred(m_1) - (pred(m_1) - impred(m_1))) \\
&= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup impred(m_1) \\
&= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\
&\quad \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x, y \in Leafs(B(m))\}
\end{aligned}
$$

Finally, executing step 5 of the conversion process, we flatten the hierarchy of *par* behaviors to one composite *par* behavior consisting of all the leaf behaviors in $B(m)$. Therefore,

$$
\begin{aligned}
B(m') &= par(Leafs(B(m))), \\
R(m') &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\
&\quad \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x, y \in Leafs(B(m))\} \\
\Rightarrow \quad m &= m'
\end{aligned}
$$

Hence proved.

3.3 Formal Verification of Specification Partitioning

As mentioned in the beginning of this section, we are interested in proving the correctness of our partition refinement algorithm. Towards that end, we established and proved the above theorems. Using these theorems and the basic axioms of the Model Algebra, we prove the correctness of the partitioning refinement algorithm.

Theorem 5 (Partitioning refinement) *Model m_p generated by partitioning refinement of specification model m, is equivalent to m.*

Proof:
The proof for this theorem is divided into two parts. First we prove that the intermediate model m_i, produced after step 2 of the partitioning algorithm, is equivalent to m. In the second part of the proof we show that the final partitioned model, m_p, is equivalent to the intermediate model m_i.

Part 1

$$\begin{aligned} \text{Let } m' &= par(Leafs(B(m))), \\ &\quad (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\ &\quad \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x,y \in Leafs(B(m))\} \end{aligned}$$

We have

$$m' = m, \text{using theorem 4}$$

Similarly, for model m_i, we have

$$\begin{aligned} m_i' &= par(Leafs(B(m_i))), \\ &\quad (R(m_i) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m_i)\}) \cup \\ &\quad \{x \rightsquigarrow y | x \overset{m_i}{\ll} y \wedge x,y \in Leafs(B(m_i))\} \end{aligned}$$

$$m_i' = m_i, \text{ using theorem 4}$$

$$B(m_i) = par(PE_1, PE_2 \ldots, PE_n)$$

Therefore,

$$\begin{aligned} Leafs(B(m_i)) &= \bigcup_{i=1}^{n} Leafs(PE_i) \\ &= \bigcup_{i=1}^{n} partition_i \\ &= Leafs(B(m)) \text{ by partitioning rules.} \end{aligned}$$

This implies,

$$\begin{aligned} B(m_i') &= par(Leafs(B(m_i))), \\ &= par(Leafs(B(m))) \\ &= B(m') \text{ using theorem 4} \end{aligned}$$

Now, let us try to prove that the immediate predecessor relations are unchanged as we transform m to m_i.
$\forall b_1 \in partition_i, b_2 \in partition_j$,
if $i = j$, then $b_1 \overset{m}{\ll} b_2 \Leftrightarrow x \overset{m_i}{\ll} y$,
since each PE is a copy of B(m)
if $i \neq j$, then
$b_1 \overset{m}{\ll} b_2 \Leftrightarrow b_1 \rightsquigarrow b_2 \in R(m_i)$,
since m has no synchronization relations
Also $b_1 \rightsquigarrow b_2 \in R(m_i) \Leftrightarrow b_1 \overset{m_i}{\ll} b_2$
since $b_1 \triangleleft PE_i \wedge b_2 \triangleleft PE_j$ and
PE_i and PE_j are composed in parallel
Therefore $b_1 \overset{m}{\ll} b_2 \Leftrightarrow b_1 \overset{m_i}{\ll} b_2$

For the relations in m_i', we have

$$\begin{aligned} R(m_i') &= (R(m_i) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m_i)\}) \cup \\ &\quad \cup \{x \rightsquigarrow y | x \overset{m_i}{\ll} y \wedge x,y \in Leafs(B(m_i))\} \\ &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\ &\quad \cup \{x \rightsquigarrow y | x \overset{m_i}{\ll} y \wedge x,y \in Leafs(B(m_i))\}, \\ &\quad \text{since data-transfer relations are preserved in } m_i \\ &= (R(m) - \{x \rightsquigarrow y | x \rightsquigarrow y \in R(m)\}) \cup \\ &\quad \cup \{x \rightsquigarrow y | x \overset{m}{\ll} y \wedge x,y \in Leafs(B(m))\}, \\ &\quad \text{since immediate predecessors are unchanged} \\ &= R(m') \end{aligned}$$

Therefore,

$$m_i' = m'$$

$$\Longrightarrow m_i = m$$

Part 2

We now prove that executing steps 3 and 4 of our partitioning algorithm on the intermediate model m_i produces an equivalent model m_p.

Let $b_1, b_2 \triangleleft B(m_i)$ and $b_1 \overset{v}{\rightarrow} b_2, b_1 \rightsquigarrow b_2 \in R(m)$. By definition of predecessor $b_1 \overset{m_i}{<} b_2$. Using axiom 5, we have

$$\begin{aligned} b_1 &= seq(b_1, e_1), e_1 \in I \text{ and} \\ b_2 &= seq(e_2, b_2), e_2 \in I \end{aligned}$$

hence $B(m_i) = B(m_p)$

We now prove equivalence of relations in the two models,

$$\begin{aligned} b_1 \rightsquigarrow b_2 &= seq(b_1, e_1) \rightsquigarrow b_2, \text{ using axiom 5} \\ &= e_1 \rightsquigarrow b_2, \text{ using axiom 3} \\ &= e_1 \rightsquigarrow seq(e_2, b_2,) \text{ using axiom 5} \\ &= e_1 \rightsquigarrow e_2, \text{ using axiom 3} \end{aligned}$$

Also,

$$\begin{aligned} b_1 \xrightarrow{v} b_2 &= b_1 \xrightarrow{v} e_1, e_1 \xrightarrow{v} b_2 \text{ using axiom 6} \\ &= b_1 \xrightarrow{v} e_1, e_1 \xrightarrow{v} e_2, e_2 \xrightarrow{v} b_2 \text{ using axiom 6} \end{aligned}$$

Using the above results, we have

$$\begin{aligned} R(m_i) &= (R(m_i) - \{b_1 \xrightarrow{v} b_2, b_1 \rightsquigarrow b_2\}) \cup \\ &\quad \{b_1 \xrightarrow{v} e_1, e_1 \xrightarrow{v} e_2, e_2 \xrightarrow{v} b_2, e_1 \rightsquigarrow e_2\} \\ &= (R(m_i) - \{b_1 \xrightarrow{v} b_2, b_1 \rightsquigarrow b_2\}) \cup \\ &\quad \{b_1 \xrightarrow{v} e_1, e_1 \xrightarrow{v} c, c \xrightarrow{v} e_2, e_2 \xrightarrow{v} b_2\}, \\ &\quad c \in C \text{ using axiom 7} \\ &= R(m_p) \end{aligned}$$

If there is no data transfer relation between b_1 and b_2, but $b_1 \rightsquigarrow b_2 \in R(m_i)$, we have

$$\begin{aligned} R(m_i) &= (R(m_i) - \{b_1 \rightsquigarrow b_2\}) \cup \{b_1 \rightsquigarrow b_2\} \\ &= (R(m_i) - \{b_1 \rightsquigarrow b_2\}) \cup \{e_1 \rightsquigarrow e_2\} \\ &= (R(m_i) - \{b_1 \rightsquigarrow b_2\}) \cup \{e_1 \xrightarrow{0} c, c \xrightarrow{0} e_2\}, \\ &\quad c \in C \text{ using axiom 7} \\ &= R(m_p) \end{aligned}$$

Hence under all cases

$$B(m_i) = B(m_p) \wedge R(m_i) = R(m_p) \Longrightarrow m_i = m_p$$

Using results from Part 1 and Part 2 of the proof, we get $m = m_p$
Hence proved.

4 Conclusion and Future Work

In this report, we presented an algebra and a formal verification scheme based on model refinement. This is a new concept wherein we perform formal verification in a system design process by deriving one model from another, rather than the traditional way of comparing two independently written models. We showed how models at different abstraction levels may be expressed in the proposed model algebra and how their transformations can be proved to be correct. The system design partitioning produced a new model that was derived from the specification model through a series of well defined refinement steps. A theorem was established and proved to show that this refinement produced a model that is functionally equivalent to the specification model.

This approach to system level design validation shows a lot of promise. In the future, we will try to expand the algebra to incorporate powerful modeling capabilities like FSM and pipeline compositions. This will enable us to develop theorems that will be used to verify more general and complex refinement algorithms.

References

[1] D. Gajski, R. Domer, A. Gerstlauer, and J. Peng. *System Design with SpecC*. Kluwer Academic Publishers, January 2002.

[2] D. Gajski, J. Zhu, R. Domer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Methodology*. Kluwer Academic Publishers, January 2000.

[3] J. Peng, S. Abdi, and D. Gajski. Automatic model refinement for fast architecture exploration. In *Proceedings of the Asia-Pacific Design Automation Conference*, pages 332–337, January 2002.