Formal Verification of Specification Partitioning

Samar Abdi and Daniel Gajski

Technical Report CECS-03-06
March 6, 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
March 6, 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 dramatic increase of behavioral and structural complexity of SoC designs has raised the abstraction level of system specification. The common approach is 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.

The only solution is to 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 modifies the model such that the model's functionality is retained. 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.

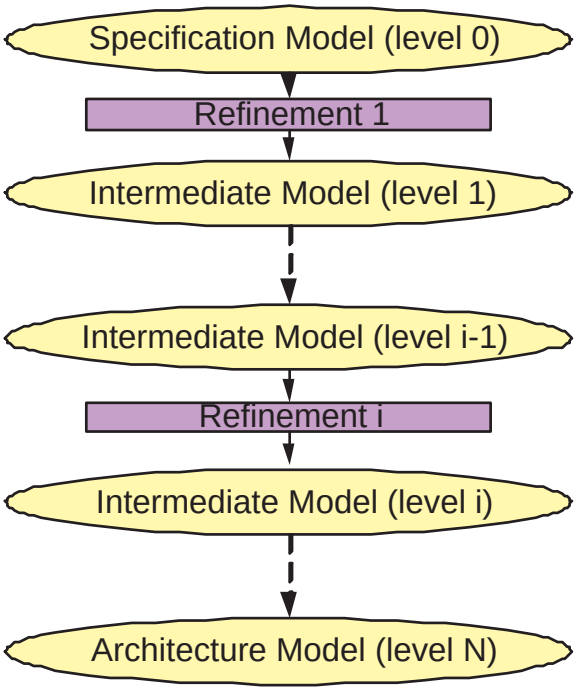

Figure 1. The gradual refinement process.

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 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. Therefore, intuitively, the system model should be a composition of these tasks. Hence, the basic unit of computation in the model is a task called *leaf behavior*. A leaf behavior is formally defined as a sequence of operations being executed in a serial order. 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 Model Algebra is defined as:

$$A =< B, \theta, \gamma >$$

Here,

B = Set of Behaviors,
θ = $\{seq, par\}$(Set of Operations)
γ = $\{\rightarrow, \triangleleft\}$(Set of Relations)

seq = Sequential composition operation

$$\text{if } b_1, b_2, b_3, ... \in B,$$
$$\text{then } seq(b_1, b_2, b_3, ...) \in B$$

par = Parallel composition operation

$$\text{if } b_1, b_2, b_3, ... \in B,$$
$$\text{then } par(b_1, b_2, b_3, ...) \in B$$

In this algebra, we have a set of objects (behaviors) and a set of operations (seq, par). The *seq* operator implies that behaviors are composed sequentially. Hence if

$$b = seq(b_1, b_2, ..., b_i), b_1, b_2, ..., b_i \in B$$

b_2 starts after b_1 has completed, b_3 starts after b_2 has completed and so on. In this case, b is a composite behavior formed from behaviors b_1 through b_i. Behavior b is said to start when b_1 starts and it ends when b_i ends. Note that $b \in B$, and can be used to create more composite behaviors.

The *par* operator combines behaviors into a parallel composition. Hence, if

$$b = par(b_1, b_2, ..., b_i), b_1, b_2, ..., b_i \in B$$

then, there is no predefined order of execution between the behaviors $b_1, b_2, ..., b_i$. In this case, b is a composite behavior formed from behaviors b_1 through b_i. Behavior b is said to start when any behavior in b_1 through b_i starts. Behavior b is said to complete when all behaviors from b_1 through b_i complete.

We define the relation *synchronization constraint* on B as follows

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

irrespective of the hierarchical composition, behavior b_2 cannot start executing until behavior b_1 completes.

We also define the relation *sub-behavior* 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.

2.1 Model Definition

Based on the above algebra, a system model can be defined as an expression in A along with a set of *synchronization constraints*.

$$M = H(M), C(M), \text{where}$$

H(M) = the hierarchical composition of behaviors representing the model M, and
C(M) = synchronization constraint on the set of behaviors in H(M)

2.1.1 Terms and definitions

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

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

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

Predecessor A behavior x is said to be a *predecessor* of behavior y in model M (denoted by $x \overset{M}{<} y$), if in the temporal order of execution x must complete before y begins. Formally,

1. $\forall x, y \triangleleft H(M)$, if $seq(.., x, y, ..) \triangleleft H(M)$ then $x \overset{M}{<} y$
2. $\forall x, y \triangleleft H(M)$, if $x \rightarrow y \in C(M)$ then $x \overset{M}{<} y$
3. $\forall x, y, z \triangleleft H(M)$, if $z \triangleleft y$ and $x \overset{M}{<} y$ then $x \overset{M}{<} z$
4. $\forall x, y, z \triangleleft H(M)$, if $z \triangleleft x$ and $x \overset{M}{<} y$ then $z \overset{M}{<} y$

5. $\forall x,y,z \triangleleft H(M),$ if $x \overset{M}{<} y$ and $y \overset{M}{<} z$ then $x \overset{M}{<} z$

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

$$\nexists z \triangleleft H(M), z \in B, \text{ such that } x \overset{M}{<} z \overset{M}{<} y$$

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 proofs 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 constraint $b_1 \rightarrow b_2$ in C(M).*

$$f(seq(b_1,b_2),b3,b4...),C(M) =$$

$$f(par(b_1,b_2),b3,b4...),C(M) \cup b_1 \rightarrow b_2$$

Axiom 2 (Flattening) *If a behavior x 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.*

$$x = seq(b_1, b_2, ...b_j)$$

$$y = seq(a_1, a_2...a_i, x, c_1, c_2...c_k) \triangleleft H(M), \text{ and}$$

$$\nexists n \triangleleft H(M), \text{ such that}$$

$$n \rightarrow x \in C(M) \text{ or } x \rightarrow n \in C(M), \text{ then}$$

$$y = seq(a_1, a_2, ..., a_i, b_1, b_2, ..., b_j, c_1, c_2, ..., c_k)$$

The dual for par *behavior is as follows:*

$$x = par(b_1, b_2, ...b_j)$$

$$y = par(a_1, a_2...a_i, x, c_1, c_2...c_k) \triangleleft H(M), \text{ and}$$

$$\nexists n \triangleleft H(M), \text{ such that}$$

$$n \rightarrow x \in C(M) \text{ or } x \rightarrow n \in C(M), \text{ then}$$

$$y = par(a_1, a_2, ..., a_i, b_1, b_2, ..., b_j, c_1, c_2, ..., c_k)$$

Axiom 3 (Forward Substitution) *If a behavior x is a sequential composition such that $x = seq(b_1, b_2...,b_i)$ and if there exists a synchronization constraint from a behavior $n \rightarrow x$, then the constraint may be replaced by a synchronization constraint $n \rightarrow b_1$. Similarly, a constraint $x \rightarrow n$ may be replaced by $b_i \rightarrow n$*

$$if\ x = seq(b_1, b_2, ..., b_i) \triangleleft H(M), \text{ then}$$

$$\forall n \triangleleft H(M), \text{ if } n \rightarrow x \in C(M)$$

$$C(M) = (C(M) - n \rightarrow x) \cup n \rightarrow b_1$$

$$\forall n \triangleleft H(M), \text{ if } x \rightarrow n \in C(M)$$

$$C(M) = (C(M) - x \rightarrow n) \cup b_i \rightarrow n$$

If a behavior x is a parallel composition such that $x = par(b_1, b_2...,b_i)$ and if there exists a synchronization constraint from a behavior $n \rightarrow x$, then the constraint may be replaced by synchronization constraints $n \rightarrow b_1, n \rightarrow b2, ..., n \rightarrow b_i$. Similarly, a constraint $x \rightarrow n$ may be replaced by constraints $b_1 \rightarrow n, b_2 \rightarrow n, ...b_i \rightarrow n$.

$$if\ x = par(b_1, b_2, ..., b_i) \triangleleft H(M), \text{ then}$$

$$\forall n \triangleleft H(M), \text{ if } n \rightarrow x \in C(M),$$

$$C(M) = (C(M) - n \rightarrow x) \cup n \rightarrow b_1, n \rightarrow b2, ..., n \rightarrow b_i$$

$$\forall n \triangleleft H(M), \text{ if } x \rightarrow n \in C(M),$$

$$C(M) = (C(M) - x \rightarrow n) \cup b_i \rightarrow n, b_2 \rightarrow n, ...b_i \rightarrow n.$$

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

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

.

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 model of the 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 sub-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. We wind up with a conclusion and proposed future work in this direction.

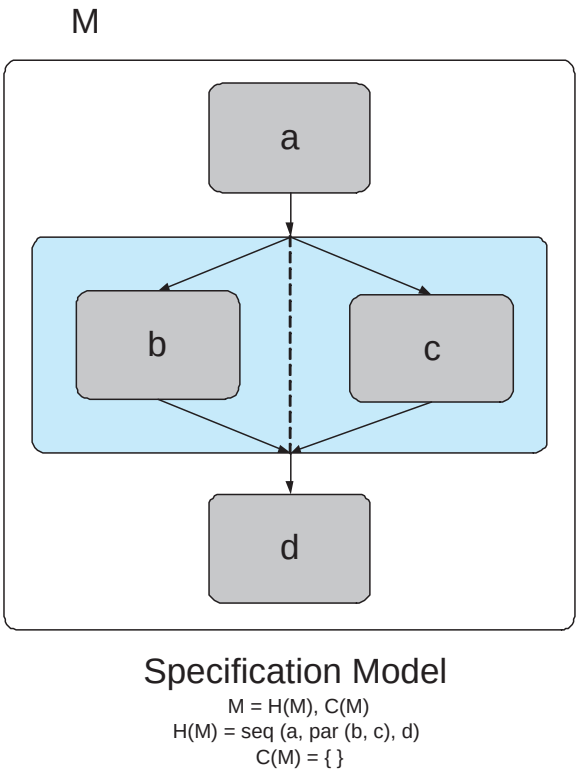

Figure 2. A simple specification model.

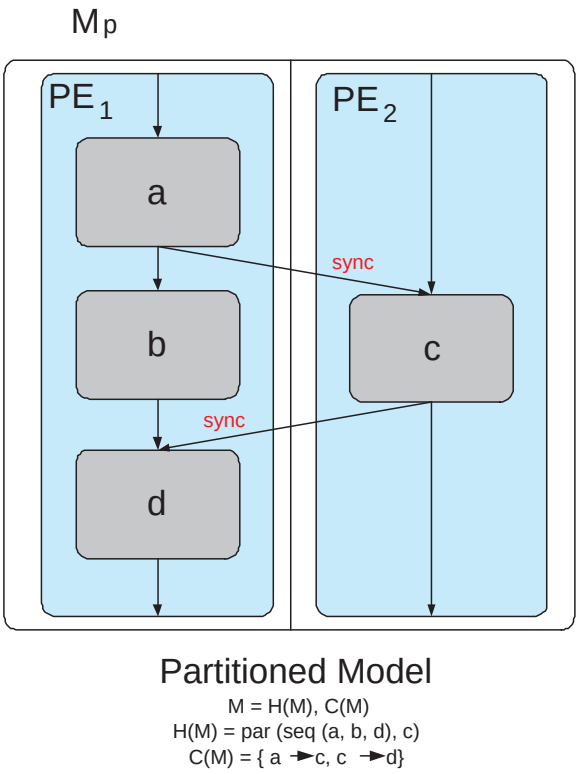

Figure 3. Model after partitioning.

3.1 Partitioning refinement algorithm

Given a model M = H(M), C(M), a set of n components $PE_1, PE_2, ..., PE_n$ and n partitions $partition_1, partition_2, ..., partition_n$. Each partition is a set of leaf behaviors in M. The partitions follow the property that

$$\bigcup_{i=1}^{n} partition_i = Leafs(H(M)), \text{ and}$$

$$partition_i \cap partition_j = \phi, 1 \leq i, j \leq n$$

$partition_i$ is assigned to PE_i

The partitioned model M_p is generated as follows.

1. We initialize H(M_p) as a parallel composition of n behaviors PE_1 through PE_n.
2. H(M) is copied into each $PE_i, 1 \leq i \leq n$
3. From each behavior $PE_i 1 \leq i \leq n$, remove all leaf behaviors b_l such that $b_l \in (Leafs(H(M)) - partition_i)$
4. Add following synchronization constraints

$$\bigcup_{i=1}^{n} \{x \rightarrow y | y \in partition_i, x \overset{M}{\ll} y,$$

$$x \in (Leafs(H(M)) - partition_i)\}$$

An example of the refinement process is demonstrated on a simple specification model. The specification, comprising of four leaf behaviors viz. *a, b, c and d* is shown in figure 2. Leaf behaviors *b* and *c* are composed in parallel to form a composite behavior. This composite behavior follows *a* and precedes *d* in a larger sequential composition. The partitioning decision is as follows.

$$\begin{aligned} partition_1 &= a, b, d, \text{ to } PE_1 \\ partition_2 &= c, \text{ to } PE_2 \end{aligned}$$

After partitioning, the model is refined to a parallel composition of PE_1 and PE_2. Synchronization constraints are added across the partitions to maintain the original partial order of execution. $a \overset{M}{\ll} c$ and $c \overset{M}{\ll} d$, are the only immediate predecessor relations across partitions. Therefore, we add corresponding synchronization constraints i.e. $a \rightarrow c$ and $c \rightarrow d$ to derive model M_p as shown in figure 3.

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 $b_1 \rightarrow b_2, b_2 \rightarrow b_3, ...b_{i-1} \rightarrow b_i$ to C(M).*

$$\begin{aligned} & f(seq(b_1, b_2, ...b_i), b_{i+1}, b_{i+2}..), C(M) \\ = \; & f(par(b_1, b_2, ...b_i), b_{i+1}, b_{i+2}..), \\ & C(M) \cup \{b_1 \rightarrow b_2, b_2 \rightarrow b_3, ..., b_{i-1} \rightarrow b_i\} \end{aligned}$$

The proof is as follows:

$$M = f(seq(b_1, b_2, ...b_i), b_{i+1}, b_{i+2}..), C(M)$$

We prove this theorem by mathematical induction. for i = 3, we have,

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

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

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

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

$$\begin{aligned} & par(b_1, b_2, .., b_x, b_{x+1}, ..., b_y, b_{y+1}, ...b_i) \\ = \; & par(b_1, b_2, ..., b_y, b_{x+1}, ..., b_x, b_{y+1}, ...b_i), \\ & x \neq y, 1 \leq x, y \leq i \end{aligned}$$

The proof is as follows: Without loss of generality, let us assume $x < y$ Let $y = x + n, n \geq 1$ So, 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

$$b = par(b_1, b_2, ..., b_y, b_x, b_{x+1}, ..., b_{x+n-1}, ..., b_i)$$

$$= par(b_1, b_2, ..., b_y, par(b_x, b_{x+1}), ..., b_{x+n-1}, ..., b_i)$$
using axiom 2

$$= par(b_1, b_2, ..., b_y, par(b_{x+1}, b_x), ..., b_{x+n-1}, ..., b_i)$$
using axiom 4

$$= par(b_1, b_2, ..., b_y, b_{x+1}, b_x, ..., b_{x+n-1}, ..., b_i)$$
using axiom 2

Again, using 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 (Immediate Predecessors) *In order for y to be an immediate predecessor of behavior x in a model M,one of the following conditions must be fulfilled.*

1. $seq(..., y, x, ...) \triangleleft H(M)$
2. $y \overset{M}{\ll} seq(x, ...)$
3. $y \overset{M}{\ll} par(..., x, ...)$
4. $seq(..., y) \overset{M}{\ll} x$
5. $par(..., y, ...) \overset{M}{\ll} x$

The proof for this theorem is derived from the definitions of *predecessors* and *immediate predecessors* as described in section 2.1.1. We prove this theorem by contradiction. Let there be a behavior $z \triangleleft H(M)$, such that $z \overset{M}{\ll} x$ and z does not follow any of the conditions 1 through 5.

Let $y_1 \triangleleft H(M)$ and $y_1 \overset{M}{\ll} x$ by condition 1. Therefore, we have $seq(..., y_1, x, ...) \triangleleft H(M)$. By definition of *predecessor*, $z \overset{M}{<} x$, so it must be true that either $z \triangleleft y_1$ or $z \overset{M}{<} seq(..., y_1, x, ...)$.

if $z \overset{M}{<} seq(..., y_1, x, ...)$, then by definition $z \overset{M}{<} y_1$. Hence, $z \overset{M}{\not\ll} x$, since $y_1 \overset{M}{<} x$.

if $z \triangleleft y_1$ and $y = seq(..., y_2)$, then $z \neq y_2$ since z does not follow condition 4. if $z \not\triangleleft y_2$, we have by definition $z \overset{M}{<} y_2$. Hence, $z \overset{M}{\not\ll} x$, since $y_2 \overset{M}{<} x$ by definition of *predecessor*. if $z \triangleleft y_2$, we revisit this proof recursively.

if $z \triangleleft y_1$ and $y = par(b_1, b_2...)$, then by condition 5, $z \neq b_1, z \neq b_2....$ If $z \triangleleft b_1, or z \triangleleft b_2...$, we revisit this proof recursively.

if $seq(x, ...) \triangleleft H(M)$,then by definition of *predecessor*, if $z \overset{M}{<} x$, we have $z \overset{M}{<} seq(x, ...)$. By Condition 2, $z \overset{M}{\not\ll} seq(x, ...)$. Therefore, there exists behavior y_3, such that $y_3 \overset{M}{<} seq(x, ...)$ and $z \overset{M}{<} y_3$. Thus $y_3 \overset{M}{<} x$, by definition. Hence $z \overset{M}{\not\ll} x$.

if $par(..., x, ...) \triangleleft H(M)$,then by definition of *predecessor*, if $z \overset{M}{<} x$, we have $z \overset{M}{<} par(..., x, ...)$. By Condition 3, $z \overset{M}{\not\ll} par(..., x, ...)$. Therefore, there exists behavior y_4, such that $y_4 \overset{M}{<} par(..., x, ...)$ and $z \overset{M}{<} y_4$. Thus $y_4 \overset{M}{<} x$, by definition. Hence $z \overset{M}{\not\ll} x$.

We have shown that z cannot be an *immediate predecessor* of x if it does not follow any of the conditions in the theorem. Therefore, an *immediate predecessor* of x must follow at least one of the conditions.

Hence proved.

Theorem 4 (Canonical form) *Any system model M = H(M), C(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.*

$$H(M') = par(Leafs(H(M))),$$

$$C(M') = \{x \rightarrow y | x \overset{M}{\ll} y, x, y \in Leafs(H(M))\}$$

The proof is as follows. We start with a specification model M of the system. We can assume that at the specification level, the designer is free to model the system behavior as any arbitrary hierarchical composition of behaviors. Thus, we can assume that there are no synchronization constraints between behaviors in M.

$$M = H(M), C(M), \text{ where } C(M) = \{\}$$

In order to reach the canonical model ′,we perform the following steps. Starting at the top of hierarchy H(M),

1. Convert *seq* nodes to *par* using theorem 1.
2. Re-arrange synchronization constraints using axioms 3 and 3.
3. Repeat from 1 till all composite nodes are parallel and synchronization constraints are between leaf behaviors.

Let there be behavior $b \in Leafs(H(M))$. if $\exists x \triangleleft H(M)$ such that $seq(...x, b, ...) \triangleleft H(M)$, then $par(...x, b...) \triangleleft H(M')$. If $x \in Leafs(H(M))$, then $x \rightarrow b \in C(M')$. By theorem 3, $x \overset{M}{\ll} b$.

if $x \notin Leafs(H(M))$, then we have two cases

1. $x = seq(x_1, x_2, \ldots, x_n)$. In this case, when we convert $seq(\ldots x, b, \ldots)$ to $par(\ldots x, b, \ldots)$, we add the synchronization constraint $x \rightarrow b$ to $C(M)$. By axiom 3, this constraint is replaced by $x_n \rightarrow b$. By theorem 3, $x_n \overset{M}{\ll} b$.

2. $x = par(x_1, x_2, \ldots, x_n)$. In this case, when we convert $seq(\ldots x, b, \ldots)$ to $par(\ldots x, b, \ldots)$, we add the synchronization constraint $x \rightarrow b$ to $C(M)$. By axiom 3, this constraint is replaced by $x_1 \rightarrow b, x_2 \rightarrow b, \ldots, x_n \rightarrow b$. By theorem 3, $x_1 \overset{M}{\ll} b, x_2 \overset{M}{\ll} b, \ldots, x_n \overset{M}{\ll} b$.

Therefore, for the case $seq(\ldots x, b, \ldots) \triangleleft H(M)$, the only synchronization constraints added to $C(M)$ for behavior b are $\{x \rightarrow b | x \overset{M}{\ll} b, x, b \in Leafs(H(M))\}$

Next, we consider the case when $seq(b, \ldots) \triangleleft H(M)$. By theorem 3, any *immediate predecessor* of b is an *immediate predecessor* of $seq(b, \ldots)$. Let us assume that a synchronization constraint $x \rightarrow seq(b, \ldots)$ is added to $C(M)$ such that $x \overset{M}{\ll} seq(b, \ldots)$. Using theorem 3, we have $x \overset{M}{\ll} b$. Also, using axiom 3, the synchronization constraint $x \rightarrow seq(b, \ldots)$ added to $C(M)$ is replaced by $x \overset{M}{\ll} b$. Therefore, by inductive reasoning, for the case $seq(b, \ldots) \triangleleft H(M)$, the only synchronization constraints added to $C(M)$ for behavior b are $\{x \rightarrow b | x \overset{M}{\ll} b, x, b \in Leafs(H(M))\}$

Finally, we have the case when $par(\ldots, b, \ldots) \triangleleft H(M)$. As in the previous scenario, by theorem
refipred, any *immediate predecessor* of b is an *immediate predecessor* of $par(\ldots, b, \ldots)$. Let us assume that a synchronization constraint $x \rightarrow par(\ldots, b, \ldots)$ is added to $C(M)$ such that $x \overset{M}{\ll} par(\ldots, b, \ldots)$. Using theorem 3, we have $x \overset{M}{\ll} b$. Also, using axiom **??**, the synchronization constraint $x \rightarrow par(\ldots, b, \ldots)$ added to $C(M)$ is replaced by $x \overset{M}{\ll} b$, amongst other constraints. Therefore, by inductive reasoning, for the case $par(\ldots, b, \ldots) \triangleleft H(M)$, the only synchronization constraints added to $C(M)$ for behavior b are $\{x \rightarrow b | x \overset{M}{\ll} b, x, b \in Leafs(H(M))\}$

Using the above three cases, we have shown that in general

$$C(M') = \{x \rightarrow y | x \overset{M}{\ll} y, x, y \in Leafs(H(M))\}$$

If we look at the steps in reaching the canonical form, note that theorem 1 is used to convert all *seq* expressions to *par* expressions. So, finally the hierarchy of the model M will have only *par* expressions. Now, using axiom 2 we can remove all composite *par* behaviors, since none of the composite behaviors have any synchronization constraints (as shown above). Therefore, after flattening, we have

$$H(M') = par(Leafs(H(M)))$$

Hence proved.

3.3 Validation of Partitioning refinement

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 functionally equivalent to M.*

The proof is as follows:

$$M = H(M), C(M)$$
$$M_p = H(M_p), C(M_p)$$

$$\begin{aligned} \text{Let } M' &= par(Leafs(M)), \\ & \quad x \rightarrow y | x \overset{M}{\ll} y, x, y \in Leafs(M) \end{aligned}$$

We have

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

For model M_p, we have

$$H(M_p) = par(PE_1, PE_2 \ldots, PE_n)$$

$$\forall i, 1 \leq i \leq n$$

Flattening PE_i using theorem 4 we get

$$\begin{aligned} M_p &= par(PE_1, \ldots par(partition_i), \ldots PE_n), \\ &= C(M_p) \cup x \rightarrow y | x \overset{M_p}{\ll} y, x, y \in partition_i \end{aligned}$$

This implies,

$$\begin{aligned} H(M'_p) &= par(par(partition_1), par(partition_2), \\ &\quad \cdots, par(partition_n)) \\ &= par(partition_1, partition_2, \ldots, partition_n) \\ &\quad \text{using axiom 2} \\ &= par(\bigcup_{i=1}^{n} partition_i), \\ &\quad \text{since partitions are disjoint} \\ &= par(Leafs(M)) \\ &= H(M') \text{ using theorem 2} \end{aligned}$$

For the synchronization constraints in the refined model, we have

$$
\begin{aligned}
C(M'_p) &= C(M_p) \cup \{\bigcup_{i=1}^{n} \{x \to y | x \overset{M_p}{\ll} y, \\
&\quad x,y \in partition_i\}\}, \\
&\quad \text{using theorem 4} \\
&= C(M) \cup \{\bigcup_{i=1}^{n} \{x \to y | y \in partition_i, x \overset{M}{\ll} y, \\
&\quad x \in (Leafs(H(M)) - partition_i)\}\} \cup \\
&\quad \{\bigcup_{i=1}^{n} \{x \to y | x \overset{M_p}{\ll} y, x,y \in partition_i\}\} \\
&= C(M) \cup \{\bigcup_{i=1}^{n} \{x \to y | y \in partition_i, x \overset{M}{\ll} y, \\
&\quad x \in (Leafs(H(M)) - partition_i)\}\} \cup \\
&\quad \{\bigcup_{i=1}^{n} \{x \to y | x \overset{M}{\ll} y, x,y \in partition_i\}\}, \\
&\quad \text{since } PE_i \text{ is a copy of H(M)} \\
&= C(M) \cup \{\bigcup_{i=1}^{n} \{x \to y | y \in partition_i, x \overset{M}{\ll} y, \\
&\quad x \in Leafs(H(M))\}\} \\
&= C(M) \cup \{x \to y | x \overset{M}{\ll} y, x,y \in Leafs(H(M))\} \\
&= C(M')
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
& M'_p = M' \\
\Longrightarrow\ & M_p = M
\end{aligned}
$$

Hence proved.

4 Conclusion

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 modeling capabilities like memory, data transactions etc. 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.