An Analysis of Reshuffled Handshaking Expansions

Rajit Manohar
Computer Systems Laboratory
Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853, U.S.A.

Abstract

We present a method for reasoning about the synchronization behavior of reshuffled handshaking expansions. The technique introduced converts the handshaking expansion into communicating hardware processes. We identify and discuss some of the limitations of the method. We show how the approach can be applied to analyze both the performance and the correctness of handshaking expansions.

1. Introduction

Martin's synthesis method for asynchronous circuit design begins with a sequential specification of a computation in the Communicating Hardware Process (CHP) notation. This specification is then transformed into a number of CHP programs whose actions are synchronized by actions on communication channels. The next step in the synthesis procedure replaces all channel communication actions with the appropriate handshake protocols. This handshaking expansion is then reshuffled before being transformed into a circuit. This reshuffling phase is one of the challenging phases of the synthesis method, and it can significantly impact the efficiency of the final implementation.

In this paper, we present a method for analyzing reshuffled handshaking expansions by converting them back to CHP programs. This permits us to analyze the correctness of reshuffled handshaking expansions at the CHP level, thus simplifying the analysis. We introduce a new composition operator among synchronization actions as a result of this analysis that differs from existing synchronization constructs used in CHP. However, we show that one cannot represent all handshaking expansions (HSE) in CHP, and we discuss the limitations of our technique.

We apply our technique to various commonly used reshuffled handshaking expansions, and show how the technique can be used for deadlock detection and timing analysis. We show how this method can analyze the performance of different reshuffled handshaking expansions in a simpler manner, because the analysis is performed at the CHP level.

van der Goot created a formal semantics for CHP programs, handshaking expansions, and production rules [3]. In his semantics, CHP processes, handshaking expansions, and production rules are represented using trees of traces. Smith and Zwarico present a semantics for determinstic asynchronous computations [11]. Several researchers have presented approaches that attempt to directly determine both the performance (cf. [2], [12]) and correctness (cf. [1]) of asynchronous circuit implementations. While these semantics can be used to analyze the circuit implementations (and in some cases the handshaking expansions) directly, we believe our approach is more intuitive and simpler to use when examining handshaking expansions because it describes the effect of reshuffling at the CHP level rather than in terms of computation traces.

2. The Problem

CHP programs are translated into handshaking expansions by replacing communication actions with handshake protocols. (A description of the CHP notation can be found in the appendix.) For simplicity, we consider dataless communication channels. (An extension of this analysis to communication actions with data can be found in [6].) Each communication channel C is implemented using two wires $\langle ci, co \rangle$ (An extension of this analysis to single wire handshake protocols can be found in [6]), and a communication action is translated into one of the following two protocols:

$co\uparrow$; $[ci]$; $co\downarrow$; $[\neg ci]$
$[ci]$; $co\uparrow$; $[\neg ci]$; $co\downarrow$

The first protocol is called an *active* four-phase protocol, whereas the second is referred to as a *passive* four-phase protocol. So, for instance, a simple buffer process $*[L; R]$ would be translated into the following HSE if L were passive and R were active:

$$*[\;[li]; lo\uparrow; [\neg li]; lo\downarrow; ro\uparrow; [ri]; ro\downarrow; [\neg ri]\;]$$

At this point the relationship between the CHP program $*[L; R]$ and the HSE is clear since the HSE is obtained from the CHP by syntactically replacing communication actions with handshake protocols. However, this handshaking expansion need not lead to the most efficient circuit implementation (where the definition of efficient depends on the design goals). Therefore, one might *reshuffle* the actions in the handshaking expansion to overlap various parts of the two handshake protocols in the interest of efficiency. The reshuffling transformation changes the relative order of actions in the handshaking expansion while preserving the individual handshake protocols. For instance, consider the following reshuffling:

$$PCHB \equiv *[\;[li]; ro\uparrow; lo\uparrow; [\neg li]; lo\downarrow; [ri]; ro\downarrow; [\neg ri]\;]$$

Since the components of the handshake protocol for the L and R actions have been mixed, the relationship between the CHP program $*[L; R]$ and this handshaking expansion is no longer easy to determine. In the extreme case, we could use the reshuffling

$$*[\;ro\uparrow; [ri]; ro\downarrow; [\neg ri]; [li]; lo\uparrow; [\neg li]; lo\downarrow\;]$$

which would clearly not be a valid implementation of $*[L; R]$—it implements $*[R; L]$. The presence of multiple actions as an implementation of a single synchronization action makes the relation between reshuffled handshaking expansions and CHP programs non-obvious.

Notation. In the rest of this paper, we use L and R to denote synchronization actions. We use $\langle li, lo\rangle$ and $\langle ri, ro\rangle$ to denote the variables used to implement the two synchronization actions respectively. Unless otherwise stated, L will be implemented using a passive four-phase protocol, and R will use an active four-phase protocol.

3. Two Phase CHP

One of the difficulties in comparing the CHP to its reshuffled handshaking expansion is that a four-phase handshaking expansion comprises *two* synchronization actions. Therefore, the first step in analyzing the relation between CHP programs and reshuffled handshaking expansions is to represent a four-phase communication action as two two-phase communication actions. With this translation, we get

$$\begin{array}{l} *[\;L; R\;] \\ \triangleright\ \{\ \textit{two phase actions}\ \} \\ *[\;L^\uparrow; L^\downarrow; R^\uparrow; R^\downarrow\;] \end{array}$$

where $L^\uparrow$ and $L^\downarrow$ represent the up-going and down-going parts of the four-phase handshake respectively. For an active four-phase protocol, we have:

$$L^\uparrow \;\equiv\; lo\uparrow; [li] \qquad L^\downarrow \;\equiv\; lo\downarrow; [\neg li]$$

and for a passive protocol we have:

$$L^\uparrow \;\equiv\; [li]; lo\uparrow \qquad L^\downarrow \;\equiv\; [\neg li]; lo\downarrow$$

Since $L^\uparrow$ and $L^\downarrow$ are both two-phase synchronization actions, we refer to the resulting CHP as *two-phase CHP*. Using this notation, the $PCHB$ process with L passive and R active can be partially written in CHP:

$$\begin{array}{l} PCHB \equiv *[[li]; ro\uparrow; lo\uparrow; \underbrace{[\neg li]; lo\downarrow}_{L^\downarrow}; [ri]; \underbrace{ro\downarrow; [\neg ri]}_{R^\downarrow}] \\ \qquad \triangleright\ *[[li]; ro\uparrow; lo\uparrow; L^\downarrow; [ri]; R^\downarrow] \end{array}$$

We have replaced the fragments of the original computation that correspond to the up-going or down-going phase of a four-phase synchronization action with the action itself. The remaining actions are those that have been reshuffled through other synchronization actions.

Consider the sequence $ro\uparrow; lo\uparrow$. Since $lo\uparrow$ cannot be blocked by the environment in any way, it would be semantically equivalent to replace this fragment with $lo\uparrow; ro\uparrow$ because the two fragments have the same synchronization behavior. With this observation, we can rewrite $PCHB$ as:

$$\begin{array}{l} PCHB \equiv *[\underbrace{[li]; lo\uparrow}_{L^\uparrow}; ro\uparrow; L^\downarrow; [ri]; R^\downarrow] \\ \qquad \triangleright\ *[\;L^\uparrow; ro\uparrow; L^\downarrow; [ri]; R^\downarrow\;] \end{array}$$

To analyze the rest of the handshaking expansion, we make the following observation: we can determine the net synchronization behavior of two reshuffled two-phase communication actions by composing them with different *environments* having known synchronization behavior. This can be summarized as follows. Assume the two synchronization actions are $L^\uparrow$ and $R^\uparrow$, and we are examining reshuffled versions of their handshaking expansions. Then the reshuffling corresponds to:

- $L^\uparrow; R^\uparrow$ if it deadlocks with environment $R^\uparrow; L^\uparrow$ but not with $L^\uparrow; R^\uparrow$
- $R^\uparrow; L^\uparrow$ if it deadlocks with environment $L^\uparrow; R^\uparrow$ but not with $R^\uparrow; L^\uparrow$
- $L^\uparrow \| R^\uparrow$ if it does not deadlock with environments $R^\uparrow; L^\uparrow$ or $L^\uparrow; R^\uparrow$
- $L^\uparrow \star R^\uparrow$ if it deadlocks with environments $R^\uparrow; L^\uparrow$ and $L^\uparrow; R^\uparrow$

We use "$\star$" to denote the last operator; it should not be confused with the bullet operator introduced by Martin [8]. (The bullet operator composes two synchronization actions so that their relative slack is zero.) Since the environment corresponds to either $L^\uparrow; R^\uparrow$ or $R^\uparrow; L^\uparrow$, the handshaking expansion for the environment is always unreshuffled. Using

L Passive, R Passive			L Active, R Active			L Active, R Passive		
$\texttt{[}li\texttt{]}; lo\uparrow; \texttt{[}ri\texttt{]}; ro\uparrow$	$\triangleright$	$L^\uparrow; R^\uparrow$	$lo\uparrow; \texttt{[}li\texttt{]}; ro\uparrow; \texttt{[}ri\texttt{]}$	$\triangleright$	$L^\uparrow; R^\uparrow$	$lo\uparrow; \texttt{[}li \wedge ri\texttt{]}; ro\uparrow$	$\triangleright$	$L^\uparrow; R^\uparrow$
$\texttt{[}li \wedge ri\texttt{]}; lo\uparrow, ro\uparrow$	$\triangleright$	$L^\uparrow \star R^\uparrow$	$lo\uparrow, ro\uparrow; \texttt{[}li \wedge ri\texttt{]}$	$\triangleright$	$L^\uparrow \Vert R^\uparrow$	$lo\uparrow; \texttt{[}ri\texttt{]}; ro\uparrow; \texttt{[}li\texttt{]}$	$\triangleright$	$L^\uparrow \Vert R^\uparrow$
$\texttt{[}ri\texttt{]}; ro\uparrow; \texttt{[}li\texttt{]}; lo\uparrow$	$\triangleright$	$R^\uparrow; L^\uparrow$	$ro\uparrow; \texttt{[}ri\texttt{]}; lo\uparrow; \texttt{[}li\texttt{]}$	$\triangleright$	$R^\uparrow; L^\uparrow$	$\texttt{[}ri\texttt{]}; lo\uparrow; \texttt{[}li\texttt{]}; ro\uparrow$	$\triangleright$	$L^\uparrow \star R^\uparrow$
						$\texttt{[}ri\texttt{]}; ro\uparrow, lo\uparrow; \texttt{[}li\texttt{]}$	$\triangleright$	$R^\uparrow; L^\uparrow$

Table 1. Summary of all reshuffled two-phase communications.

this observation, we can determine the CHP for any two adjacent synchronization actions. Table 1 contains all possible reshufflings of two two-phase dataless communication actions with the corresponding CHP program fragments they represent. Using this table, we can finally write the CHP corresponding to $PCHB$ as follows:

$$PCHB \equiv *\texttt{[}L^\uparrow; \underbrace{ro\uparrow; L^\downarrow; \texttt{[}ri\texttt{]}}_{L^\downarrow \Vert R^\uparrow}; R^\downarrow\texttt{]}$$
$$\triangleright\ *\texttt{[}\ L^\uparrow; (L^\downarrow \parallel R^\uparrow); R^\downarrow\ \texttt{]}$$

The only unfamiliar operator in Table 1 is "$\star$". This composition method is possible between an active and a passive protocol, or between two passive protocols. When both $L^\uparrow$ and $R^\uparrow$ are passive, the environment of $L^\uparrow \star R^\uparrow$ must use active protocols on both L and R. Since there is no reshuffling of an active $L^\uparrow$ and active $R^\uparrow$ action that leads to $L^\uparrow \star R^\uparrow$, the environment must use $L^\uparrow \Vert R^\uparrow$ to ensure absence of deadlock.

Consider the case when reshuffling $L^\uparrow \star R^\uparrow$ is used with L active and R passive. Observe that the reshuffling is asymmetric. If the environment uses the same reshuffling for R active and L passive, the computation deadlocks. Therefore, we conclude that "$\star$" is not the bullet operator introduced by Martin [8]. The environment *must* permit the concurrent execution of $L^\uparrow$ and $R^\uparrow$ to avoid deadlock.

An alternative to treating "$\star$" as a new operator is to rewrite $L \star R$ using probes [7]. Using the probe to describe the "$\star$" construct also exposes the asymmetry introduced when composing an active and passive handshake using this operator. For $L^\uparrow$ passive, $R^\uparrow$ passive, we get:

$$L^\uparrow \star R^\uparrow \quad \equiv \quad \texttt{[}\overline{L^\uparrow} \wedge \overline{R^\uparrow}\texttt{]}; L^\uparrow \Vert R^\uparrow$$

For $L^\uparrow$ active and $R^\uparrow$ passive, we get:

$$L^\uparrow \star R^\uparrow \quad \equiv \quad \texttt{[}\overline{R^\uparrow}\texttt{]}; L^\uparrow; R^\uparrow$$

The normal convention in choosing active versus passive protocols for communication channels is that a probed channel must use a passive protocol. Note that the above construction preserves this convention—all probed channels are passive.

If the handshaking expansions contain internal concurrency, then additional reshufflings are possible. We can analyze these by using the following simple rules. Let us write the two synchronization actions being composed as $L_1^\uparrow; L_2^\uparrow$ and $R_1^\uparrow; R_2^\uparrow$, where the two parts correspond to the two actions (a wait and an assignment) that implement the handshake. If the reshuffling corresponds to $L_1^\uparrow; L_2^\uparrow \Vert R_1^\uparrow; R_2^\uparrow$, then the composition operator is clearly parallel composition. In all other cases, we must have a single action from $L^\uparrow$ (or $R^\uparrow$) in parallel with one or more actions of $R^\uparrow$ (or $L^\uparrow$). In that case, we sequentialize the reshuffling as follows: if the single $L^\uparrow$ (or $R^\uparrow$) action is a wait, we postpone it to the end of the $R^\uparrow$ (or $L^\uparrow$) actions it overlaps with; if the action is an assignment, we prepone it to the beginning of the $R^\uparrow$ (or $L^\uparrow$) actions it overlaps with. Now we can use Table 1 to determine the synchronization behavior.

The following Theorem on $\star$ composition follows directly from Table 1.

Theorem 1 (Active star-composition) *A sequence of n two-phase communication actions that are composed using $\star$ can be implemented if and only if at most one of the n actions uses an active protocol.*

Proof: Follows from the fact that one cannot compose two two-phase active protocols with a $\star$ operator. ∎

3.1. Uniform Characterization of Synchronization Behavior

An examination of Table 1 shows that the different synchronization operators ";", "$\Vert$", and "$\star$" can be characterized in a simple way that does not depend on whether the participating synchronization actions are active or passive.

Let X and Y be two synchronization actions. Let $\mathbf{a}X$ denote the assignment statement and $\mathbf{w}X$ denote the wait statement in the two-phase synchronization action X. Furthermore, let $P \prec Q$ mean that P occurs before Q in the handshaking expansion. The following assertions follow from Table 1:

- X and Y are composed as $X; Y$ if and only if: $(\mathbf{w}X \prec \mathbf{a}Y) \vee (\mathbf{a}X \prec \mathbf{w}Y)$
- X and Y are composed as $X \parallel Y$ if and only if: $(\mathbf{a}X \prec \mathbf{w}Y) \vee (\mathbf{a}Y \prec \mathbf{w}X)$

- X and Y are composed as $X \star Y$ if and only if: $(\mathbf{w}X \prec \mathbf{a}Y) \vee (\mathbf{w}Y \prec \mathbf{a}X)$

Note that this characterization is correct regardless of whether X and Y are implemented with active or passive protocols. The protocol type determines how the two components of action X (and Y) are ordered. If $\mathbf{w}X \prec \mathbf{a}X$, then X uses a passive protocol; if $\mathbf{a}X \prec \mathbf{w}X$, then X uses an active protocol.

3.2. Bullets

If C_1, C_2, ..., C_n are n communication actions, then the bullet $C_1 \bullet C_2 \bullet \cdots \bullet C_n$ denotes the *simultaneous composition* of the n actions. Intuitively, this means that none of the communication actions can complete until all others have begun. We note that the "$\star$" operator can only be used to compose two-phase communication actions, whereas the bullet can be used to compose two-phase as well as four-phase communication actions. Also, if $L \star R$ is connected to a process that executes $L \star R$ as well, the computation will deadlock. It is not clear if this property is true for $L \bullet R$ connected to $L \bullet R$ at the CHP level.

In an attempt to clearly define the bullet in terms of two-phase CHP, we start from the the definition of the bullet. Martin defines the bullet operator as follows: "For $S1$ and $S2$ non-interfering communication commands, if the executions of both $S1$ and $S2$ in a certain state of the computation terminate, then the execution of $S1 \bullet S2$ in that state terminates. Furthermore, the completion of $S1$ coincides with the completion of $S2$." (from [9]). The notion of completion of a non-atomic action can also be defined following Martin: "A non-atomic action X is said to be completed when it is initiated and is guaranteed to terminate, i.e., when all possible continuations of the computation contain the complete sequence of atomic actions of X." Therefore, a single non-atomic action can have multiple completion points, thereby allowing the existence of a single point in the computation that is a valid completion point for multiple actions. For the remainder of the discussion, we interpret "continuations of the computation" to include computations with arbitrary environments when examining the bulleted actions *in isolation*. Otherwise any deadlock-free computation by definition could have its initial state be the completion point for all actions.

If A_1 and A_2 are two communication actions composed using the bullet, we conclude that $\mathbf{c}A_1 = \mathbf{c}A_2$, where $\mathbf{c}X$ denotes the number of completed X-actions for any action X. Consider the following collection of processes:

$$*[A_1 \bullet A_2] \;\|\; *[A_2 \bullet A_3] \;\| \cdots \|\; *[A_{n-1} \bullet A_n]$$

By definition of the bullet, we conclude that $\mathbf{c}A_i = \mathbf{c}A_j$ for all $1 \leq i, j \leq n$. We assume that these communications are implemented with a four-phase protocol, as we can consider the two-phase case as a four-phase protocol where the first and second halves of the protocol are symmetric. We focus our attention on the constraint $\mathbf{c}A_1 = \mathbf{c}A_n$.

Let C and D be two four-phase communication actions that are bulleted together. We first consider the case when the various two-phase components of C and D are only composed using sequential composition. We are left with the following possible two-phase CHP implementations of $C \bullet D$ (together with the same implementations with C and D interchanged):

$$C^{\uparrow}; C^{\downarrow}; D^{\uparrow}; D^{\downarrow}$$
$$C^{\uparrow}; D^{\uparrow}; C^{\downarrow}; D^{\downarrow}$$
$$C^{\uparrow}; D^{\uparrow}; D^{\downarrow}; C^{\downarrow}$$

It should be immediately clear that the first possibility can be discarded since it is an implementation of $C; D$. The second can be discarded for the same reason; it cannot preserve the constraint that $\mathbf{c}A_1 = \mathbf{c}A_n$ in the example above, since $\mathbf{c}A_n$ could be as large as $\mathbf{c}A_1 + \lfloor (n-1)/2 \rfloor$. The third possibility does indeed prevent the difference $\mathbf{c}A_1 - \mathbf{c}A_n$ from growing as a function of n. The reader can verify that this reshuffling can be used to implement the bullet (details can be found in [6]).

Normally when implementing $C \bullet D$ we know that the environment will attempt to execute C and D concurrently. In this case, we can use the $\star$ operator to compose the two-phase components of C and D to implement the bullet.

Two-phase bullets. When implementing two-phase communication actions, we are forced to use the $\star$-operator to compose them in order to implement synchronization between the two operations. However, in the strictest definition of operator $\bullet$, this is illegal because it does not permit the bulleted action to be composed with itself. Therefore, if X is an active two-phase communication and Y is a passive two-phase communication, then there is no valid implementation of $X \bullet Y$ under the interpretation that $X \bullet Y$ connected to $X \bullet Y$ does not deadlock without the introduction of additional synchronization operations. However, if we know that X and Y will be executed concurrently by the environment, then we can synchronize the two communication actions by implementing the bullet using $\star$. Therefore, we can think of $\star$ as a two-phase bullet operator.

4. Reshuffling Multiple Synchronization Actions

Consider the following fragment of a handshaking expansion with L passive and R active:

$$\ldots; [li]; ro\uparrow; [ri]; ro\downarrow; [\neg ri]; lo\uparrow; \ldots$$

We cannot use Table 1 directly to determine the corresponding CHP because action $L^{\uparrow}$ has been reshuffled across multiple synchronization actions ($R^{\uparrow}$ and $R^{\downarrow}$). Instead, we examine each *pair* of synchronization actions independently. The three pairs of synchronization actions are:

$$\ldots;\, [li];\, ro\uparrow;\, [ri];\, \_;\, \_;\, lo\uparrow;\, \ldots$$
$$\ldots;\, [li];\, \_;\, \_;\, ro\downarrow;\, [\neg ri];\, lo\uparrow;\, \ldots$$
$$\ldots;\, \_;\, ro\uparrow;\, [ri];\, ro\downarrow;\, [\neg ri];\, \_\_;\, \ldots$$

These correspond to the following CHP fragments:

$$\ldots;\, L^{\uparrow} \star R^{\uparrow};\, \ldots$$
$$\ldots;\, L^{\uparrow} \star R^{\downarrow};\, \ldots$$
$$\ldots;\, R^{\uparrow};\, R^{\downarrow};\, \ldots$$

The CHP that is *consistent* with these three fragments is

$$\ldots;\, L^{\uparrow} \star (R^{\uparrow};\, R^{\downarrow});\, \ldots$$

The reason this is program is consistent with the others is that if we project the CHP onto the pairs of synchronization actions, we get the CHP fragments shown earlier. Therefore, this CHP corresponds to the original handshaking expansion.

Example. Consider the following process in which L is passive and R is active.

$$*[[\overline{L} \longrightarrow L;\, R]]$$

We provide some of the reshufflings of the handshaking expansions for this process derived using the method outlined above. (The interested reader is referred to [6] for an exhaustive list.)

$$*[[li];\, lo\uparrow;\, [\neg li];\, lo\downarrow, ro\uparrow;\, [ri];\, ro\downarrow;\, [\neg ri]]$$
$$\equiv \quad *[L^{\uparrow};\, L^{\downarrow};\, R^{\uparrow};\, R^{\downarrow}]$$
$$*[[li];\, lo\uparrow, ro\uparrow;\, [\neg li];\, lo\downarrow;\, [ri];\, ro\downarrow;\, [\neg ri]]$$
$$\equiv \quad *[L^{\uparrow};\, (L^{\downarrow} \| R^{\uparrow});\, R^{\downarrow}]$$
$$*[[li];\, lo\uparrow, ro\uparrow;\, [ri];\, ro\downarrow;\, [\neg li];\, lo\downarrow;\, [\neg ri]]$$
$$\equiv \quad *[L^{\uparrow};\, R^{\uparrow};\, (L^{\downarrow} \| R^{\downarrow})]$$

An important point illustrated by these examples is that although it may look like a handshaking expansion is sequential, it may not actually order synchronization actions because of the particular reshuffling used.

In the remainder of this section, we formalize the intuition above into a technique for constructing the CHP that is consistent with a general handshaking expansion. In the process, we identify and discuss some of the limitations of the technique.

4.1. The Construction

In this section we will formalize the observations we made above by providing a method to construct the CHP corresponding to a reshuffled handshaking expansion. Observe that the CHP notation is a concise, textual way to specify how different operations in a computation are ordered. In this section, we use a graph-based approach for depicting the ordering among actions, and then reconstruct the CHP program from the graph for the handshaking expansion.

In order to gain some insight into the procedure and its limitations, consider the following handshaking expansion with L active, X active, and R passive:

$$\ldots;\, lo\uparrow;\, [ri];\, ro\uparrow;\, xo\uparrow;\, [li];\, [\neg ri];\, ro\downarrow;\, [xi];\, \ldots$$

Applying the technique outlined above, we get the following operators between pairs of synchronization actions:

$$L^{\uparrow} \| R^{\uparrow} \qquad L^{\uparrow};\, R^{\downarrow} \qquad L^{\uparrow} \| X^{\uparrow}$$
$$R^{\uparrow};\, X^{\uparrow} \qquad R^{\uparrow};\, R^{\downarrow} \qquad X^{\uparrow} \| R^{\downarrow}$$

After some thought, it is evident that there is no CHP program that can simultaneously represent all six synchronization constraints. The problem here is that we have a reshuffling that contains improperly nested parallelism—the $X^{\uparrow}$ action begins before $L^{\uparrow}$ completes, but $L^{\uparrow}$ completes before $X^{\uparrow}$ completes. Consider the following handshaking expansion:

$$..;\, [li];\, ao\uparrow;\, [ai];\, [ri];\, bo\uparrow;\, [bi];\, lo\uparrow;\, co\uparrow;\, [ci];\, ro\uparrow;\, ..$$

Applying the technique outlined above, we get the following operators between pairs of synchronization actions:

$$L^{\uparrow} \star A^{\uparrow} \quad L^{\uparrow} \star R^{\uparrow} \quad L^{\uparrow} \star B^{\uparrow} \quad L^{\uparrow};\, C^{\uparrow} \quad A^{\uparrow};\, R^{\uparrow}$$
$$A^{\uparrow};\, B^{\uparrow} \quad A^{\uparrow};\, C^{\uparrow} \quad R^{\uparrow} \star B^{\uparrow} \quad R^{\uparrow} \star C^{\uparrow} \quad B^{\uparrow};\, C^{\uparrow}$$

Examining a subset of the synchronization actions, it becomes apparent that the CHP must contain fragments of the form $L^{\uparrow} \star (A^{\uparrow};\, B^{\uparrow});\, C^{\uparrow}$ and $A^{\uparrow};\, R^{\uparrow} \star (B^{\uparrow};\, C^{\uparrow})$, which is not possible. Once again, the problem is that we have a reshuffling that contains improperly nested star-operators.

We can show that in the absence of improperly nested synchronization actions, we can represent the handshaking expansion at the CHP level. In what follows, we only consider straight-line handshaking expansions. To simplify the treatment, we consider multiple occurrences of the same synchronization action to be labelled as distinct.

Definition 1 (Ordering Relation) *Given a handshaking expansion $*[S]$, we define an ordering relation $\prec$ on two-phase synchronization actions in the body S as follows: if A and B are two synchronization actions, then $A \prec B$ just when S projected on the actions A and B is equivalent to $A;B$ according to Table 1.*

For example, if the handshaking expansion of interest was:

$$*[lo\uparrow;\, [ri];\, ro\uparrow;\, xo := \neg xo;\, [li];\, [\neg ri];\, ro\downarrow;\, [xi = xo]]$$

then we would have the following relations:

$$L^{\uparrow} \prec R^{\downarrow} \qquad R^{\uparrow} \prec X \qquad R^{\uparrow} \prec R^{\downarrow}$$

where X denotes the two-phase synchronization on (xi, xo).

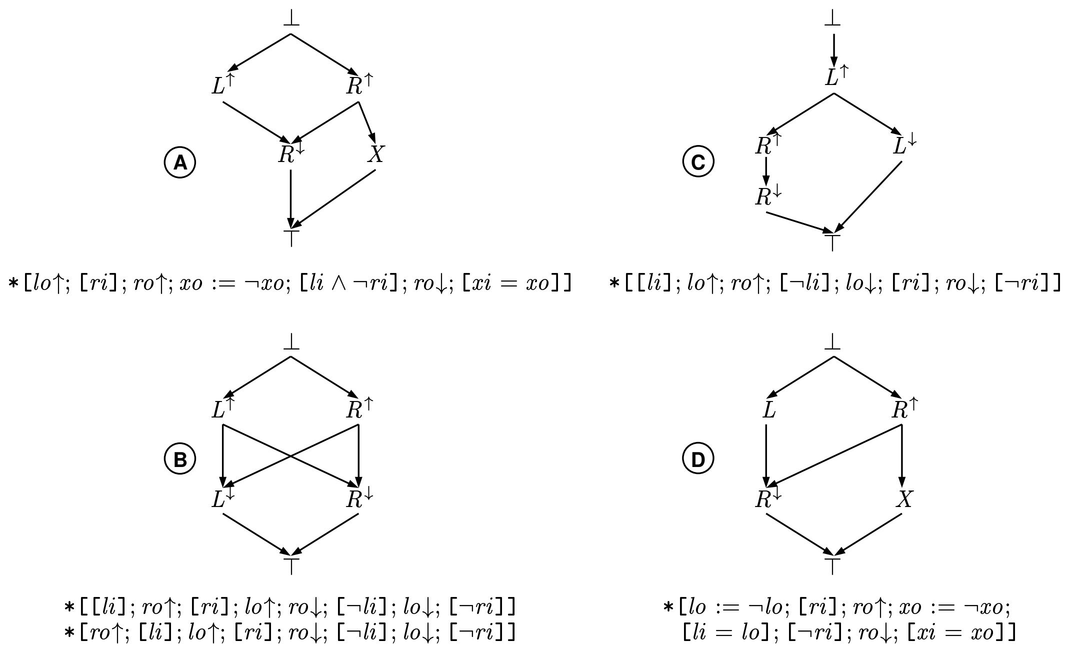


Figure 1. Flow graphs for various handshaking expansions.

Theorem 2 (Partial Order) *The relation $\prec$ is a strict partial order, and $\preceq$ defined by $a \preceq b \equiv (a = b) \vee (a \prec b)$ is a partial order.*

Proof: Follows from the definitions given above. ∎

From the partial order $\preceq$, we can draw a graph that represents the ordering of synchronization actions in the handshaking expansion.

Definition 2 (Flow Graph) *The flow graph (V, E) is obtained from the partial order $\preceq$ as follows. The set V is the synchronization actions in S, along with two special labels $\perp$ (the start symbol) and $\top$ (the end symbol). Let ρ be the transitive reduction of the relation $\prec$. Given two synchronization actions $a, b \in V$, there is an edge from a to b just when $a\rho b$. Finally, we introduce edges from $\perp$ to a for all a with no incoming edges from synchronization actions, and edges from b to $\top$ for all b with no outgoing edges to synchronization actions.*

Figure 1 shows a number of handshaking expansions along with their flow graphs. Note that the flow graph does not distinguish between $X \star Y$ and $X \| Y$; in both cases, the actions X and Y are not totally ordered. Therefore, multiple handshaking expansions could have identical flow graphs.

A node in a flow graph is said to be a *fork node* if it has out-degree greater than one; a node is said to be a *join node* if it has in-degree greater than one. For each vertex v we define the set $pred(v)$ to be the set of vertices u such that $(u, v) \in E$.

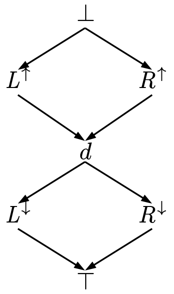
Definition 3 (Expanded Flow Graph) *The expanded flow graph (V, E) of flow graph (V', E') is defined as follows. For each maximal set of vertices $\{v_1, \ldots, v_n\} \subseteq V'$ such that $pred(v_1) = pred(v_2) = \cdots = pred(v_n)$ and $|pred(v_1)| > 1$, we apply the following transformation: (a) A new dummy vertex d is introduced; (b) The set of edges is modified by replacing each edge (u, v_1) with edge (u, d); (c) The edges (d, v_1), (d, v_2), …, (d, v_n) are introduced. This transformation is repeatedly applied until no such vertex set is possible.*

In the examples shown in Figure 1, the only graph whose expanded flow graph differs from its flow graph is graph B. The expanded flow graph for B is shown in Figure 2. In what follows, we use flow graph to mean the expanded flow graph of a handshaking expansion.

If a CHP process is entirely sequential, there are no fork or join nodes in its flow graph. Fork nodes correspond to points in the CHP process where actions that are composed with $\|$ or $\star$ are initiated; join nodes correspond to points where these actions terminate. We can use this intuition to determine when an flow graph corresponds to a CHP process.

Definition 4 (Properly Nested Flow Graphs) *A flow graph is said to be properly nested if it can be reduced to graph $G_0 = (\{\top, \perp\}, \{(\perp, \top)\})$ by repeatedly applying the following transformation:*

- (EDGE-ELIM). *If vertex v has both out-degree and in-degree 1 with corresponding edges (u, v), (v, w), delete v from the graph and introduce edge (u, w).*



$*$[[li]; $ro{\uparrow}$; [ri]; $lo{\uparrow}$; $ro{\downarrow}$; [$\neg li$]; $lo{\downarrow}$; [$\neg ri$]]
$*$[$ro{\uparrow}$; [li]; $lo{\uparrow}$; [ri]; $ro{\downarrow}$; [$\neg li$]; $lo{\downarrow}$; [$\neg ri$]]

Figure 2. An expanded flow graph.

If we examine the flow graphs given in Figure 2 and Figure 1, graphs labelled B and C are properly nested, while A and D are not. Properly nested flow graphs correspond to CHP programs. We capture this fact by the following theorem, whose proof is constructive. Before proceeding, we restrict our attention to programs where pairs of synchronization actions are not composed with $\star$; we extend the construction to this class of programs at the end of this section.

Theorem 3 (Graph-CHP) *If a flow graph is properly nested, then there exists a CHP program that corresponds to the ordering relation specified by the flow graph.*

Proof: Let (V, E) be a properly nested flow graph. We construct a *labelled* flow multigraph by attaching program fragments to edges of the flow graph. Therefore, there could be more than one edge between two pairs of vertices, but the edges would have different labels. Initially, each edge is labelled with **skip**. Since the flow multigraph is properly nested, repeated application of EDGE-ELIM results in graph G_0. When applying transformation EDGE-ELIM, edge (u, v) with label S and (v, w) with label T are replaced with edge (u, w) with label $S; v; T$ when v is a synchronization action, and with label $S; T$ when v is a dummy vertex. If EDGE-ELIM can no longer be applied and vertex u and v have multiple edges between them with labels S_1, S_2, ..., S_n, then the edges are replaced with a single edge with label $(S_1 \parallel S_2 \parallel \cdots \parallel S_n)$. The reader can verify that this CHP program contains the ordering relations specified by the flow graph (details can be found in [6]). Figure 3 shows this process for the flow graph in Figure 2. ∎

Theorem 4 (CHP-graph) *Every straightline CHP program has a properly nested flow graph.*

Proof: By structural induction on CHP programs [6]. ∎

Theorem 3 shows how a CHP program can be constructed from a flow graph when the only composition operators among pairs of synchronization actions are semicolon and parallel composition. The same argument can be used when the only composition operators permitted are semicolon and star.

If we examine the proof of Theorem 3, we observe that the only point of difficulty in extending it to arbitrary composition operators occurs when we reduce multiple edges between two vertices into a single edge. As an example of this problem, consider the handshaking expansion:

$$*[\ ao{\uparrow};\ [ai];\ yo{\uparrow};\ [zi];\ xo{\uparrow};\ [xi];\ [yi];\ zo{\uparrow}; \ldots\ ]$$

where "..." denotes the reset part of the handshaking expansion where the second half of the two-phase protocols are completed. Figure 4 shows the flow graph for this handshaking expansion. The problem is that while $X^{\uparrow}$ and $Y^{\uparrow}$ can execute in parallel, $X^{\uparrow}$ and $Z^{\uparrow}$ are $\star$-composed, and there is no CHP program that corresponds to this particular flow graph. Therefore, the only cases when we can construct a CHP program are those when, given a flow graph of the form shown above, the composition operator among elementary synchronization actions in parallel branches are unique. If this is the case, then we say that the corresponding HSE satisfies the *unique-operator* constraint.

Theorem 5 (Reshuffling Theorem) *A handshaking expansion corresponds to a CHP program if the flow graph generated from its partial order relation is properly nested and satisfies the unique-operator constraint.*

Proof: A constructive proof of this theorem is provided in [6]. The proof is similar to Theorem 3. ∎

5. Applications

The techniques we have presented for the analysis of handshaking expansions can be used to check the correctness of reshuffled handshaking expansions, and to estimate the performance of the circuit before its final CMOS implementation is synthesized.

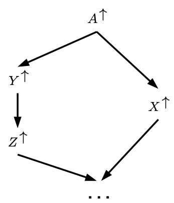


Figure 4. A problematic case for HSE to CHP conversion.

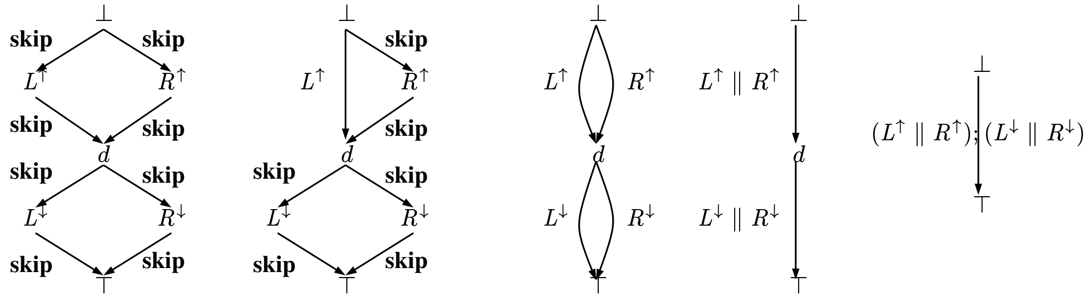


Figure 3. Transforming a flow graph into its CHP program.

5.1. Analysis of Deadlock

Consider a simple ring with three processes, as shown in Figure 5. Process I is the initiator, and it begins by sending a value on channel R. Processes A and B are buffers, and they simply copy values received on their input to their output. The three processes can be described as follows:

$$
\begin{array}{lcl}
I & \equiv & R; *[\ F; R\ ] \\
B & \equiv & *[\ R; S\ ] \\
A & \equiv & *[\ S; F\ ]
\end{array}
$$

This collection of processes is clearly deadlock-free. We annotate the channels with their protocol types (subscript a indicating active, and subscript p indicating passive) to obtain:

$$
\begin{array}{lcl}
I & \equiv & R_a; *[\ F_p; R_a\ ] \\
B & \equiv & *[\ R_p; S_p\ ] \\
A & \equiv & *[\ S_a; F_a\ ]
\end{array}
$$

Suppose we implement processes I and A using the following reshufflings:

$$
\begin{array}{l}
I \equiv ro\uparrow; [ri]; ro\downarrow; [\neg ri]; \\
\qquad *[[fi]; fo\uparrow, ro\uparrow; [ri \wedge \neg fi]; fo\downarrow, ro\downarrow; [\neg ri]] \\
A \equiv *[so\uparrow; [si]; fo\uparrow; [fi]; so\downarrow; [\neg si]; fo\downarrow; [\neg fi]]
\end{array}
$$

In both cases, the body of the handshaking expansion uses standard buffer reshufflings that are commonly used. Suppose buffer B was implemented using the following reshuffling:

$$B \ \equiv\ *[[ri \wedge si]; ro\uparrow, so\uparrow; [\neg ri \wedge \neg si]; ro\downarrow, so\downarrow]$$

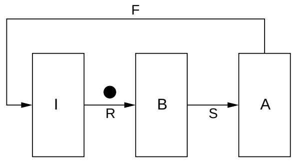


Figure 5. A ring of processes containing a single token.

which corresponds to a buffer that is implemented with a C-element. At first glance, these seem to be a reasonable reshuffling choices. However, this choice leads to deadlock. Using the techniques previously outlined, we can write the two-phase CHP for the three processes as:

$$
\begin{array}{lcl}
I & \equiv & R^\uparrow; R^\downarrow;\ \ *[F^\uparrow; R^\uparrow; F^\downarrow; R^\downarrow] \\
A & \equiv & *[\ S^\uparrow; F^\uparrow; S^\downarrow; F^\downarrow\ ] \\
B & \equiv & *[\ R^\uparrow \star S^\uparrow; R^\downarrow \star S^\downarrow\ ]
\end{array}
$$

A quick glance shows that the system deadlocks after executing $R^\uparrow \star S^\uparrow$. Observe that even if we introduce additional buffer processes that use the same reshuffling as B, we do not eliminate the deadlock.

This technique was put to use in the analysis of an early version of the divider used in the MiniMIPS processor [10]. A subtle deadlock problem was eliminated by changing the reshuffling of a single process in the iterative datapath for the divider. Other solutions such as the introduction of buffers would have degraded the performance of the divider by increasing the loop latency.

Checking handshaking expansions for deadlock by examining the corresponding two-phase CHP has some limitations because, as we have seen, not all handshaking expansions correspond to CHP programs. However, we can use the flow graph for the handshaking expansion to directly determine whether the reshuffling is deadlock-free. Intuitively, the reshuffling will be deadlock-free when the flow graph for the complete system is cycle-free. We also have to introduce additional undirected edges in the flow graph that correspond to the case when two synchronization actions are composed using $\star$. Details of this transformation can be found in [6].

5.2. The Lazy-Active Protocol

A standard reshuffling of the four-phase active protocol is what is referred to as the lazy-active protocol. In this protocol, the final wait of the four-phase communication is postponed to the next occurrence of the handshake. The protocol on wires (xi, xo) is typically written:

$[\neg xi]; xo\uparrow; [xi]; xo\downarrow$

What happens when we replace an active protocol with a lazy-active one? Intuitively, it is clear that the circuit will have better performance characteristics because we have postponed the final wait until the last possible moment. If we examine this transformation at the CHP level, the net effect of introducing the lazy-active protocol is to replace what was a semicolon at the CHP level with parallel composition. To illustrate this, consider a simple one-place active-passive buffer with the active part implemented using a lazy-active protocol. The reshuffling is:

$$*[\ [\neg li]; lo\uparrow; [li]; lo\downarrow; [ri]; ro\uparrow; [\neg ri]; ro\downarrow\ ]$$

Since we are using a lazy-active protocol, the wait $[\neg li]$ is the last part of the communication action on L in the previous loop iteration. Simply put, this handshaking expansion is equivalent to

$$*[\ lo\uparrow; [li]; lo\downarrow; [ri]; ro\uparrow; [\neg ri]; ro\downarrow; [\neg li]\ ]$$

and the resulting two-phase CHP is:

$$*[\ L^{\uparrow}; (L^{\downarrow} \parallel R^{\uparrow}; R^{\downarrow})\ ]$$

If L were implemented with an active protocol, the resulting two-phase CHP would be:

$$*[\ L^{\uparrow}; L^{\downarrow}; R^{\uparrow}; R^{\downarrow}\ ]$$

Given a deadlock free CHP computation, the unreshuffled handshaking expansion derived from it will also be deadlock free. It should now be immediately obvious that replacing any active protocol with a lazy-active protocol will keep the computation deadlock-free, because the net effect (at the CHP level) of the transformation is to permit the second half of the synchronization protocol to execute concurrently with whatever follows it. Also, since all probes only depend on the first half of the communication, this transformation does not affect the values of probes.

5.3. CRT Reshufflings

A collection of CHP processes is said to exhibit *constant response time* (CRT) if the time between successive communications on their inputs or outputs is bounded by a constant, independent of the number of processes. If the collection of processes under consideration is a linear array of N identical processes, we can use Lee's result to determine if the CHP computation exhibits CRT [5]. To determine if CHP computations exhibit CRT, let the CHP for the repeated process be given by:

$$*[\ \alpha_0; \alpha_1; \ldots; \alpha_{N-1}\ ]$$

where each α_i is a synchronization action. Let the set $C \subseteq \{\alpha_0, \ldots, \alpha_{N-1}\}^2$ describe the set of *matching* synchronization actions. (Two actions match if they are connected in adjacent processes.) Lee's CRT criterion states that a CHP computation is CRT if and only if the following two conditions hold [5]:

1. Deadlock freedom:
$$(\forall e, f, e', f' : (\alpha_e, \alpha_f), (\alpha_{e'}, \alpha_{f'}) \in C : e < e' \Leftrightarrow f < f')$$

2. CRT:
$$(\forall e, f : (\alpha_e, \alpha_f) \in C : e > f) \vee (\forall e, f : (\alpha_e, \alpha_f) \in C : e < f)$$

(The reader is referred to [5] for a discussion of these conditions as well as the proof.) Therefore, given a straightline handshaking expansion, we can determine if the reshuffling exhibits CRT by constructing the two-phase CHP for the HSE and using Lee's criterion. We can extend the CRT and deadlock-freedom criteria to allow for concurrent execution of synchronization actions by noting that all that is necessary is that there should exist an interleaving of the concurrent operations that satisfies Lee's criterion.

5.4. Control-data Decomposition

Control-data decomposition is a transformation that eliminates operations on data items with dataless communication actions coupled with standard datapath processes. The transformation replaces communication action $D!x$ with a dataless communication C, along with the process $*[C \bullet D!x]$. Communication action $D?x$ is replaced by a dataless communication C, along with the process $*[C \bullet D?x]$. The two-phase CHP for the implementation of datapath process $*[C \bullet D!x]$ is given by $*[C^{\uparrow} \star D^{\uparrow}?x; C^{\uparrow} \star D^{\downarrow}]$, and the CHP for datapath process $*[C \bullet D?x]$ is given by $*[C^{\uparrow} \star D^{\uparrow}!x; C^{\uparrow} \star D^{\downarrow}]$. Note that control-data decomposition *guarantees* that the control action C and data action D are attempted in parallel, since the C-action is introduced to replace the original matching synchronization action on channel D.

6. Summary

We presented a new technique for reasoning about the correctness and performance of handshaking expansions. The technique introduces the notion of "decompiling" a handshaking expansion into a CHP program that has the same synchronization behavior. We showed several applications of the analysis technique and also discussed its limitations.

Acknowledgments

The author did initial work on two-phase CHP when he was part of Alain Martin's group at Caltech (specifically, the first part of section 3), and would like to thank members of the group for various discussions.

The research described in this report was supported in part by the Defence and Advanced Research Projects Agency and monitored by the Office of Army Research. The work was also supported in part by the Multidisciplinary University Research Initiative (MURI) under the Office of Naval Research Contract N00014-00-1-0564, and in part by a National Science Foundation CAREER award under contract CCR 9984299.

A. Summary of CHP Notation

The notation we use is based on Hoare's CSP [4]. A full description of the notation and its semantics can be found in [9]. What follows is a short and informal description of the notation we use.

- Assignment: $a := b$. This statement means "assign the value of b to a." We also write $a\uparrow$ for $a := true$, and $a\downarrow$ for $a := false$.
- Selection: $[G1 \rightarrow S1 \; [] \; ... \; [] \; Gn \rightarrow Sn]$, where Gi's are boolean expressions (guards) and Si's are program parts. The execution of this command corresponds to waiting until one of the guards is $true$, and then executing one of the statements with a $true$ guard. The notation $[G]$ is short-hand for $[G \rightarrow skip]$, and denotes waiting for the predicate G to become true. If the guards are not mutually exclusive, we use the vertical bar "|" instead of "[]."
- Repetition: $*[G1 \rightarrow S1 \; [] \; ... \; [] \; Gn \rightarrow Sn]$. The execution of this command corresponds to choosing one of the $true$ guards and executing the corresponding statement, repeating this until all guards evaluate to $false$. The notation $*[S]$ is short-hand for $*[true \rightarrow S]$.
- Send: $X!e$ means send the value of e over channel X.
- Receive: $Y?v$ means receive a value over channel Y and store it in variable v.
- Probe: The boolean expression $\overline{X}$ is $true$ iff a communication over channel X can complete without suspending.
- Sequential Composition: $S; T$
- Parallel Composition: $S \parallel T$ or S, T.
- Simultaneous Composition: $S \bullet T$ both S and T are communication actions and they complete simultaneously.

References

[1] J. Brzozowski and C.-J. H. Seger. Asynchronous Circuits. Springer-Verlag, 1995.

[2] Steven M. Burns and Alain J. Martin. Performance analysis and optimization of asynchronous circuits. In Carlo H. Séquin, editor, *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, pp. 71–86, 1991.

[3] Marcel van der Goot. The Semantics of VLSI Synthesis. Ph.D. thesis CS-TR-95-08, California Institute of Technology, 1996.

[4] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, **21**(8):666–677, August 1978.

[5] Tak-Kwan Lee. Communication Behavior of Linear Arrays of Processes. M.S. thesis, Caltech technical report CS-TR-89-13, June 1988.

[6] Rajit Manohar. Two-phase CHP: Analyzing Handshaking Expansions by Decompilation. Cornell Computer Systems Laboratory technical report CSL-TR-2000-1006, September 2000.

[7] Alain J. Martin. The Probe: An addition to communication primitives. *Information Processing Letters*, **20**:125–130, 1985.

[8] Alain J. Martin. Programming in VLSI: From communicating processes to delay-insensitive circuits. In C.A.R. Hoare, editor, *Developments in Concurrency and Communication*, UT Year of Programming Series, pp. 1–64. Addison-Wesley, 1990.

[9] Alain J. Martin. Synthesis of Asynchronous VLSI Circuits. Caltech Computer Science technical report CS-TR-93-28.

[10] Alain J. Martin, Andrew Lines, Rajit Manohar, Mika Nyström, Paul Penzes, Robert Southworth, Uri V. Cummings, and Tak-Kwan Lee. The Design of an Asynchronous MIPS R3000. *Proceedings of the 17th Conference on Advanced Research in VLSI*, September 1997.

[11] S.F. Smith and A.E. Zwarico. Correct Compilation of Specifications to Deterministic Asynchronous Circuits. *Formal Methods in System Design*, **7**:155–226, 1995.

[12] Ted Eugene Williams. Self-Timed Rings and their Application to Division. Ph.D. thesis, Stanford University, May 1991.