Implication of Assertion Graphs in GSTE

Guowu Yang, Jin Yang[1], William N. N. Hung[1], and Xiaoyu Song

Department of Electrical & Computer Engineering, Portland State University, OR, USA

[1]Intel Corporation, Hillsboro, OR, USA

Abstract - We address the problem of implication of assertion graphs that occur in generalized symbolic trajectory evaluation (GSTE). GSTE has demonstrated its powerful capacity in formal verification of digital systems. Assertion graphs are used for property and model specifications. We present a novel implication technique for assertion graphs. It relies on direct Boolean reasoning on each edge (and vertex) of an assertion graph, thus avoiding the reachability computation in GSTE. We have successfully applied both model-based and language-based implications on real industrial circuits. Experimental results demonstrate the promising performance of our approach.

I. Introduction

Symbolic Trajectory Evaluation (STE) [1] is the main alternative to symbolic model checking (SMC). Compared with SMC, which usually requires abstraction [7], STE has the advantage of being applied to very large circuits directly. The STE theory consists of a simple specification language, a simulation-based model checking algorithm, and a mapping of the algorithm to a coarse abstract domain. The specification language of STE has the limited expressiveness where only properties over finite time intervals are allowed. In the generalized STE (GSTE) [2], all Omega-regular properties can be expressed and verified with the same space efficiency and comparable time efficiency. Assertion graphs are introduced in GSTE as an extension of STE's specification language.

GSTE specifications are expressed in the form of assertion graphs. It is common knowledge that many RTL designs are rather complicated, primarily because they model complex functional behavior while accommodating tight performance constraints. If we have already proved an assertion graph G_1 against the RTL, a desirable usage is to use G_1 to prove (imply) another assertion graph G_2. Having such an implication mechanism would enable us to achieve higher level abstractions and pursue assume-guarantee prove strategies. Previously, people have tried to convert the assumed assertion graph G_1 into a circuit [3], and then re-run GSTE on the compiled circuit [4] to prove G_2. The problem with such an approach is that when G_1 is a complicated assertion graph, the GSTE run on its compiled circuit (against G_2) may easily blow up.

In this paper, we present a novel way to prove that G_1 implies G_2 without running GSTE. Our method relies on direct Boolean reasoning on each edge (and vertex) of the assertion graphs, thus avoiding the reachability computation in GSTE and side step the biggest potential cause for the BDD explosion problem. We have successfully applied both model-based implication and language-based implication on real industrial circuits.

II. Preliminaries

We introduce some basic definitions in this section. Some of them were given in [1, 2]. We assume a non-empty set of finite states, denoted by S. A relation $T \subseteq S \times S$ is a transition relation if $\forall s \in S$, $\exists s' \in S$, $(s, s') \in T$, where S is a non-empty set of finite states. The model M induced by the transition relation T is the pair (pre, $post$) where: (1) the pre-image transformer pre: $2^S \to 2^S$ is defined as: $pre(Q)=\{s|\exists s' \in Q, (s, s') \in T\}$ for all $Q \in 2^S$; and (2) the $post$-image transformer $post$: $2^S \to 2^S$ is defined as: $post(Q)=\{s'|\exists s \in Q, (s, s') \in T\}$ for all $Q \in 2^S$. Let $M=(pre, post)$ be a directed graph $M=(S, T)$. We use pre and $post$ to represent two functions based on M. Note that $pre(s)=pre(\{s\})$, $post(s)=post(\{s\})$, for all $s \in S$. If for all $s \in S$, $post(s)$ is defined and nonempty, then M is well-defined. Namely, if we first define $post$: $S \to 2^S-\{\varnothing\}$, where $\varnothing$ is an empty set, then a transition relation T can be defined as $T=\{(s, s') | s \in S, s' \in \text{post(s)}\}$. A trace in $M=(pre, post)$ is a state sequence σ such that $\sigma[i+1] \in post(\sigma[i])$, for all $1 \le i < |\sigma|$, i.e., $(\sigma[i], \sigma[i+1]) \in T$.

An assertion graph is a quintuple $G=(V, v_0, E, ant, cons)$ where V is a finite set of vertices, v_0 is the initial vertex, $E \subseteq V \times V$ is a set of edges, satisfying $\forall u \in V$, $\exists v \in V$, such that $(u, v) \in E$, ant is a mapping $E \to 2^S$, cons is a mapping $E \to 2^S$. Let $ban(e)=ant(e)-cons(e)$. We use $start(e)$ and $end(e)$ to denote the start and end vertices of a directed edge e, respectively. We define $out(e)$ as the set of successor edges of e, i.e., $start(e')=end(e)$ for all $e' \in out(e)$.

Let $G = (V, v_0, E, ant, cons)$ be an assertion graph, and let $M=(pre, post)$ be a model. We define an edge labeling γ as γ: $E \to 2^S$ where γ is either ant or $cons$. A trace σ in M satisfies a path ρ of the same length under γ, denoted by $(M, \sigma) \models \gamma (G, \rho)$, iff $\sigma[i] \in \gamma(\rho[i])$, $1 \le i \le |\sigma|$. A trace satisfies a path, denoted by $(M, \sigma) \models (G, \rho)$, iff $[(M, \sigma) \models_{ant} (G, \rho)] \Rightarrow [(M, \sigma) \models_{cons} (G, \rho)]$. A model M strongly satisfies an assertion graph G, denoted by $M \models G$ iff $(M, \sigma) \models (G, \rho)$ for all finite initial path ρ in G and all finite trace σ in M of the same length. Given two assertion graphs $G_1 = (V, v_0, E_1, ant_1, cons_1)$ and $G_2 = (U, u_0, E_2, ant_2, cons_2)$, G_1

model-based implies G_2, denoted by $G_1 \Rightarrow_{model} G_2$, iff $\forall M$, $M \models G_1 \Rightarrow M \models G_2$. A *word* w on S is a *concatenation* (string) of m states, $w = w[1]w[2]\ldots w[m]$, where $w[i] \in S$, $1 \le i \le m$. We use $w[i]$ to denote the i^{th} *letter* of w. The length of w is $m = |w|$. We use ε to denote the *null word*, and $|\varepsilon| = 0$. We use S^* to denote the set of all words on S. Given a word w and an assertion graph $G = (V, v_0, E, ant, cons)$, w satisfies a path ρ of the same length under γ (ant or $cons$), denoted by $w \models_\gamma \rho$, iff $w[i] \in (\rho[i])$ for every $1 \le i \le |w|$. A word satisfies a path, denoted by $w \models \rho$, iff $w \models_{ant} \rho \Rightarrow w \models_{cons} \rho$. A word satisfies the assertion graph, denoted by $w \models G$, iff $w \models \rho$ for all path ρ with the same length $|w|$ in the assertion graph. Given an assertion graph $G = (V, v_0, E, ant, cons)$, the language $L(G)$ of the assertion graph G is $L(G) = \{w \in S^* \mid w \models G\}$. Given a model $M=(S, T)$, the language $L(M)$ of the model M is $L(M)=\{w \in S^* | (w[i], w[i+1]) \in T, 1 \le i \le |w|-1\}$. Given two assertion graphs $G_1 = (V, v_0, E_1, ant_1, cons_1)$ and $G_2 = (U, u_0, E_2, ant_2, cons_2)$, G_1 *language-based implies* G_2, denoted by $G_1 \Rightarrow_{lang} G_2$, iff $L(G_1) \subseteq L(G_2)$.

III. Assertion Graph Implication

In this section, we prove some conditions sufficient to establish language-based and model-based implications for assertion graphs. At the end of the section, we will show that language-based implication is sufficient to establish model-based implication; however, the reverse is not always true.

3.1 Language-based implication

Theorem 3.1 Given $G_1 = (V, v_0, E, ant_1, cons_1)$ and $G_2 = (V, v_0, E, ant_2, cons_2)$, if $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \subseteq cons_2(e))$, for all $e \in E$, then $G_1 \Rightarrow_{lang} G_2$.

Proof: $\forall\ w \in L(G_1)$, For any finite initial path ρ with the same length $|w|$ such that $(M, w) \models_{ant2} (G_2, \rho)$, we have $w[i] \in ant_2(\rho[i])$, for every $1 \le i \le |w|$. Then $w[i] \in ant_2(\rho[i]) \subseteq ant_1(\rho[i])$. $w \in L(G_1) \Rightarrow w[i] \in cons_1(\rho[i]) \subseteq cons_2(\rho[i])$, therefore $(M, w) \models_{cons2} (G_2, \rho)$, which means $w \in L(G_2)$. Thus $L(G_1) \subseteq L(G_2)$, $G_1 \Rightarrow_{lang} G_2$. □

Notice the condition $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \subseteq cons_2(e))$ is natural and direct. But it is too strong. Theorem 3.2 gives a weaker condition which guarantees $G_1 \Rightarrow_{lang} G_2$.

Theorem 3.2 Given $G_1 = (V, v_0, E, ant_1, cons_1)$ and $G_2 = (V, v_0, E, ant_2, cons_2)$, if $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \cap ant_2(e) \subseteq cons_2(e))$ for all $e \in E$, then $G_1 \Rightarrow_{lang} G_2$.

Proof: $\forall\ w \in L(G_1)$, For any finite initial path ρ with the same length $|w|$ such that $(M, w) \models_{ant2} (G_2, \rho)$, we have $w[i] \in ant_2(\rho[i])$, for every $1 \le i \le |w|$. Then $w[i] \in ant_2(\rho[i]) \subseteq ant_1(\rho[i])$. $w \in L(G_1) \Rightarrow w[i] \in cons_1(\rho[i])$, so $w[i] \in cons_1(\rho[i]) \cap ant_2(\rho[i]) \subseteq cons_2(\rho[i])$, therefore $(M, w) \models_{cons2} (G_2, \rho)$, which means $w \in L(G_2)$. Thus $L(G_1) \subseteq L(G_2)$, $G_1 \Rightarrow_{lang} G_2$. □

Lemma 3.1 $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \cap ant_2(e) \subseteq cons_2(e)) \Leftrightarrow (ant_1(e) \supseteq ant_2(e)) \wedge (ban_1(e) \supseteq ban_2(e))$, for all $e \in E$.

Proof: $\Rightarrow$: $ban_1(e) = ant_1(e) - cons_1(e) \supseteq ant_2(e) - cons_1(e)$
$= ant_2(e) - cons_1(e) \cap ant_2(e) \supseteq ant_2(e) - cons_2(e) = ban_2(e)$.
$\Leftarrow$: $cons_1(e) \cap ant_2(e) = cons_1(e) \cap (ant_2(e) \cap cons_2(e) \cup ban_2(e))$
$\subseteq (cons_1(e) \cap ant_2(e) \cap cons_2(e)) \cup (cons_1(e) \cap ban_1(e))$
$= cons_1(e) \cap ant_2(e) \cap cons_2(e) \subseteq cons_2(e)$. □

Theorem 3.3 If G_1 and G_2 have the same graph structure and $ant_1(e) \supseteq ant_2(e)$ and $ban_1(e) \supseteq ban_2(e)$, for all $e \in E$, then, $G_1 \Rightarrow_{lang} G_2$.

Proof: Directly from Theorem 3.2 and Lemma 3.1. □

Set ban(e) is a useful concept when we analyses the unsatisfying word of an assertion graph. In the following, we give soundness and completeness condition of $G_1 \Rightarrow_{lang} G_2$ when G_1 ang G_2 are linear assertion graph and have the same graph structure.

Definition 3.1 *Linear assertion graph* $G = (V, v_0, E, ant, cons)$: Every edge has one and only one successor edge. In the following, without special announcement, if an assertion graph G is a linear assertion graph, we always assume that $|E| = m$, $e[m] = (v_{m-1}, v_t)$, $0 \le t \le m-1$, $e[m+1] = e[t+1]$, namely, after m, edges have a periodicity.

Definition 3.2 $L_k(G) = \{w \mid w \in L(G), |w| = k\}$, i.e., $L_k(G)$ is all the words satisfying the assertion graph G with length k. $UNL_k(G) = \{w \mid w \notin L(G), |w| = k\}$, i.e., $UNL_k(G)$ is all the words unsatisfying the assertion graph G with length k.

Lemma 3.2 $G_1 \Rightarrow_{lang} G_2 \Leftrightarrow L(G_1) \subseteq L(G_2) \Leftrightarrow L_k(G_1) \subseteq L_k(G_2)$ for all $k>0 \Leftrightarrow UNL_k(G) \supseteq UNL_k(G)$ for all $k>0$.

Proof: Directly from the definitions. □

Lemma 3.3 Assuming $G = (V, v_0, E, ant, cons)$ is a linear assertion graph with $|E| = m$, $e[m] = (v_{m-1}, v_t)$, $0 \le t \le m-1$, $e[m+1] = e[t+1]$, then
$UNL_k(G) = ban(e[1]) \times ant(e[2]) \times \ldots \times ant(e[k]) \cup$
$ant(e[1]) \times ban(e[2]) \times \ldots \times ant(e[k]) \cup \ldots$
$\cup ant(e[1]) \times ant(e[2]) \times \ldots \times ban(e[k])$ for all $k>0$.

Proof: $\forall\ w \in UNL_k(G)$, Since the definition of a word satisfying an assertion graph, $\exists$ path ρ, $|\rho| = k$, such that $w \models_{ant} \rho$ but $w \models_{cons} \rho$ is invalid, i.e., $\exists 1 < h \le k$ such that $w[h] \notin cons(\rho[h])$, which implies $w[h] \in ban(\rho[h])$. Therefore $w \in ant(e[1]) \times \ldots \times ban(e[h]) \times \ldots \times ant(e[k])$. And obviously, for any w, if $w \in ban(e[1]) \times ant(e[2]) \times \ldots \times ant(e[k]) \cup ant(e[1]) \times ban(e[2]) \times \ldots \times ant(e[k]) \cup \ldots \cup ant(e[1]) \times ant(e[2]) \times \ldots \times ban(e[k])$, then $w \in UNL_k(G)$. Thus $UNL_k(G) = ban(e[1]) \times ant(e[2]) \times \ldots \times ant(e[k]) \cup ant(e[1]) \times ban(e[2]) \times \ldots \times ant(e[k]) \cup \ldots \cup ant(e[1]) \times ant(e[2]) \times \ldots \times ban(e[k])$ for all $k>0$. □

Theorem 3.4 If G_1 and G_2 have the same graph structure

and are linear assertion graphs, then $G_1 \Rightarrow_{lang} G_2$ iff

1. If there exists r with $t \le r \le m$, such that $ban_2(e[r]) \neq \varnothing$, then $ant_1(e) \supseteq ant_2(e)$, and
   - If there is a minimum integer c such that $ban_1(e[c]) \supseteq ant_2(e[c])$, then $ban_1(e[i]) \supseteq ban_2(e[i])$ for $1 \le i \le c$;
   - Else $ban_1(e) \supseteq ban_2(e)$;

   Or,
2. Else assume j is the maximum integer number such that $ban_2(e[j]) \neq \varnothing$, then $ant_1(e[i]) \supseteq ant_2([i])$ for any $1 \le i \le j$, and
   - If there is a minimum integer $c \le j$ such that $ban_1(e[c]) \supseteq ant_2(e[c])$, then $ban_1(e[i]) \supseteq ban_2(e[i])$ for $1 \le i \le c$;
   - Else $ban_1(e[i]) \supseteq ban_2(e[i])$ for $1 \le i \le j$.

Proof: See technical report [6].

3.2 Model-based implication

There are some differences between model-based implication and language-based implication. A trace in a model satisfying an assertion graph is a word satisfying the assertion graph, while a word satisfying an assertion graph may not be a trace in a model satisfying the assertion graph. There are two essential distinctions. First, if a trace in a model satisfies an assertion graph, then any part of this trace is still a trace satisfying the assertion graph; therefore any part of this trace is a word satisfying the assertion graph. However, if a word satisfies an assertion graph, a part of it may not be a word satisfying the assertion graph. Second, any finite trace in a model must be a prefix of a trace in the model; but a word satisfies the assertion graph may not be a prefix of another satisfying word. We will see these differences in the following example, which demonstrate the relationship of the two implications.

Example 3.1: Consider two assertion graphs G_1 and G_2:

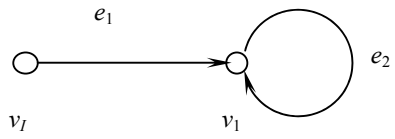


Figure 1: structure of assertion graph G_1 and G_2.

G_1 : $ant_1(e_1) = cons_1(e_1) = \{2\}$, $ant_1(e_2) = \{3, 4, 5, 6, 7\}$,
$cons_1(e_2) = \{5, 6\}$, $ban_1(e_2) = \{3, 4, 7\}$,
G_2 : $ant_2(e_1) = cons_2(e_1) = \{2\}$, $ant_2(e_2) = \{2, 3, 4, 5\}$,
$cons_2(e_2) = \{2, 4, 5\}$, $ban_2(e_2) = \{3\}$.

We use numbers to denote states. From [6], we have $G_1 \Rightarrow_{model} G_2$. But obviously a word (2,2,3) satisfies G_1, which does not satisfy G_2, so $G_1 \Rightarrow_{lang} G_2$ is invalid. First, notice that the word (2,2,3) cannot be a trace in a model satisfying G_1(otherwise (2,3) also can be a trace, which is a contradiction). This fact shows that the union of the languages of all the models satisfying an assertion graph may be less than the language of the assertion graph. Second, a part (2,3) of the word (2,2,3) does not satisfy G_1. This is consistent with the differences between a word and a trace. The fact that $ant_1(e_2)$ does not contain $ant_2(e_2)$ shows that $ant_2(e) \subseteq ant_1(e)$ is not necessary for $G_1 \Rightarrow_{model} G_2$.

Lemma 3.4 Given a model M and an assertion graph $G = (V, v_0, E, ant, cons)$, $M \models G$ iff $L(M) \subseteq L(G)$.

Proof: According to the definitions of trace and the language of a model, a word in $L(M)$ is a trace in M, vice versa. For any finite initial path ρ and any finite trace σ of the same length such that $(M, \sigma) \models_{ant} (G, \rho)$, from $M \models G$, we have $\sigma[i] \in cons(\rho[i])$, for every $1 \le i \le |\sigma|$, which means the word σ in $L(G)$. Therefore $L(M) \subseteq L(G)$. For any finite initial path ρ and any finite trace σ of the same length such that $(M, \sigma) \models_{ant} (G, \rho)$, from $L(M) \subseteq L(G)$, we have $\sigma[i] \in cons(\rho[i])$, for every $1 \le i \le |\sigma|$, which means $(M, \sigma) \models_{cons} (G, \rho)$. Therefore $M \models G$. □

Using Lemma 3.1 and Example 3.1, we have:

Theorem 3.5

1) $(G_1 \Rightarrow_{lang} G_2) \Rightarrow (G_1 \Rightarrow_{model} G_2)$.
2) $(G_1 \Rightarrow_{model} G_2) \Rightarrow (G_1 \Rightarrow_{lang} G_2)$ is not valid.

Using Theorems 3.1, 3.2, 3.3, and 3.5, we have:

Theorem 3.6 If two aspersion graphs G_1 and G_2 have the same graph structure and one of the following conditions is satisfied, then $G_1 \Rightarrow_{model} G_2$.

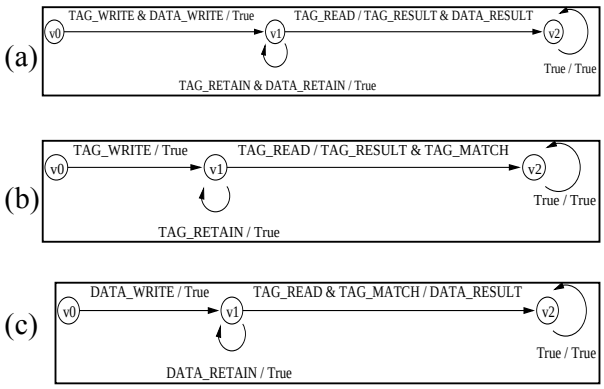
(1) $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \subseteq cons_2(e))$, for all $e \in E$,
(2) $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \cap ant_2(e) \subseteq cons_2(e))$, for all $e \in E$, namely, $(ant_1(e) \supseteq ant_2(e)) \wedge (ban_1(e) \supseteq ban_2(e))$, for all $e \in E$.

IV. Experiments

In this section we show two industrial verification examples using our GSTE assertion graph implication approach: a content-addressable memory and a memory unit with complicated output alignment/mask operations.

We have successfully applied both model-based implication and language-based implication on a content-addressable memory (CAM) previously shown in Figure 5 of [4] for assertion graph implication. The top level property specification is shown in Figure 2(a). We first prove that the tag portion of the CAM functions correctly, and on a tag read it produces the correct tag match, shown in Figure 2(b). We also prove that the data portion of the CAM functions correctly, and given a tag match, outputs the corresponding data, shown in Figure 2(c). We prove these two assertion graphs (Figure 2(b) and 2(c)) using GSTE. We then create a product assertion graph G, which is the cross product of (b) and (c). Since both (b) and (c) have already been proven to be true, their product G is also true. Also, if a graph G is true, a portion of the graph G' should also be true. We extract a portion (subgraph G') of the product graph that is structurally identical to the top level property (Figure 2(a)). The only difference between G' and the top level specification are the antecedents and consequents on each

edge. At this point, we apply Theorem 3.1 and Theorem 3.6 to show that G' implies the top level specification. Both model-based implication and language-based implication were successfully established.



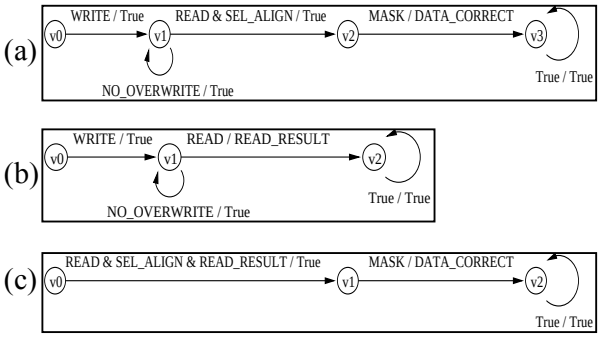
TAG_WRITE := (twrite = 1) ∧ (taddr = A) ∧ (tagin = T)
DATA_WRITE := (dwrite = 1) ∧ (daddr = A) ∧ (din = D)
TAG_RETAIN := (twrite = 0) ∨ (taddr ≠ A)
DATA_RETAIN := (dwrite = 0) ∨ (daddr ≠ A)
TAG_READ := (aread = 1) ∧ (tagin = T)
TAG_RESULT := (hit = 1)
TAG_MATCH := ∀i [(i = A) ⇒ match[i]]
DATA_RESULT := ∀i [(i = A) ⇒ (matchout[i] = D)]

Figure 2. Assertion graphs for CAM.

For runtime comparison, it took 8 CPU seconds to verify the original property through GSTE (without decomposition). Each of the decomposed properties (Figure 2(b) and (c)) took 1.5 seconds to verify through GSTE. The implication based Theorem 3.1 took 0.01 second and the implication based Theorem 3.4 took 0.1 second. So the runtime for the decomposed properties plus the implication total around 3 seconds, which is much faster than the original verification.

We have also verified a memory unit previously discussed in [2, 4]. The top level specification is shown in Figure 3(a). We decompose the proof into two parts. First we prove that the core part of the memory unit send out the correct data upon a read (Figure 3(b)). Given the read results from the core memory, we prove that the select, align and mask operations are carried out correctly for the data output (Figure 3(c)). We use GSTE to prove the assertion graphs in Figure 3(b) and 3(c) against the memory unit, create a product assertion graph G from these two graphs, extract a part of the product graph (partial assertion graph), G', that is structurally the same as the top level specification (Figure 3(a)). We then use theorem 3.1 and theorem 3.6 to show that G' implies the top level specification (Figure 3(a)).

Verification of the original property (Figure 3(a)) through GSTE took 182 CPU seconds (without decomposition). Verification of each of the decomposed properties (Figure 3(b) and (c)) took 2 seconds each. Runtime for the implication based on Theorem 3.1 and 3.6 took 0.01 and 0.3 seconds respectively. So the overall runtime using a decomposition and assertion graph implication strategy is much faster than proving the original property through GSTE.



WRITE := (we = 1) ∧ (addr = A) ∧ (datawr = D)
NO_OVERWRITE := (we = 0) ∨ (addr ≠ A)
READ := (ck = 0) ∧ (we = 0) ∧ (addr = A)
SEL_ALIGN := (sel = S) ∧ (align = R)
MASK := (ck = 1) ∧ (maskbegin = B) ∧ (maskend = E)
DATA_CORRECT := (dataout=mask(align(sel(D,S),R),B,E))
READ_RESULT := (memout = D)

Figure 3. Assertion graphs for the memory unit

V. Summary and Conclusions

We investigated the problem of implication of assertion graphs that occur in generalized symbolic trajectory evaluation (GSTE). We presented a novel implication technique. It relies on direct Boolean reasoning on each edge (and vertex) of the assertion graphs, thus avoiding the reachability computation in GSTE. We successfully applied both model-based implication and language-based implication on industrial circuits. Experimental results demonstrate the promising performance of our approach.

References

[1] J. Yang and C. Seger, "Generalized Symbolic Trajectory Evaluation," *Technical Report*, 2002, Intel Corporation.

[2] J. Yang and C. Seger, "Introduction to Generalized Symbolic Trajectory Evaluation," *IEEE Trans on VLSI Systems*, 11(3), pp. 345-353, 2003.

[3] A. J. Hu, J. Casas, and J. Yang, "Efficient Generation of Monitor Circuits for GSTE Assertion Graphs," *IEEE/ACM International Conference on Computer-Aided Design, 2003.*

[4] A. J. Hu, J. Casas, and J. Yang, ``Reasoning about GSTE Assertion Graphs," *12th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME)*, 2003.

[5] C. Segar and R.E. Bryant, "Formal verification by symbolic evaluation of partially-ordered trajectories," *Formal Methods in System Design*, 6(2): 147-190, March 1995.

[6] G. Yang and X. Song, "Formal Verification by Generalized Symbolic Trajectory Evaluation", Technical Report, Portland State University, 2004. http://www.ece.pdx.edu/~song/

[7] W. N. N. Hung and N. Narasimhan, "Reference Model Based RTL Verification: An Integrated Approach", *IEEE International High Level Design Verification and Test Workshop (HLDVT)*, Sonoma, California, November 2004.