AN ALGORITHM FOR THE VERIFICATION OF TIMING DIAGRAMS REALIZABILITY

Abdelhalim El-Aboudi, El-Mostapha Aboulhamid

Laboratoire LASSO, Département IRO,
Université de Montréal, Montréal (Qc), CANADA

Abstract

In this paper, we present a new method for verifying the realizability of a timing diagram with linear timing constraints, thus ensuring that the implementation of the underlying interface is feasible. The method is based on the consistency of the timing constraints derived from the timing diagram and accepts unknown occurrence times for events produced by the environment.

Keywords
Timing diagrams, timing constraints, consistency, realizability.

1. Introduction

Many applications in microelectronic design systems require the adequate elaboration of interfaces between communicating parts especially in an environments of hardware software codesign or distributed systems. Among these applications, real time systems represent an important case, since interfacing is subject to hard timing constraints. I.e., Communicating protocols between components are mainly characterized by temporal constraints, and necessitate a proper timing within the interface controllers. Generally, these protocols are specified using timing diagrams (TDs), because they are expressive, simple and familiar to the designers. The problem to be resolved in this paper is the realizability of a given timing diagram specification. Among the relevant works which have considered TDs with quantitative timing constraints, we can find important concepts and properties. Such properties constitute the background for the temporal analysis of underlying interfaces, such as the maximum separation time for determining temporal distances between events in the TD [3,4,5,6,9]. Consistency ensures that the given system of constraints has at least one solution [7]. Causality is defined in [1] as the existence of a partition over events that verifies specific properties. Such properties constitute sufficient conditions of realizability defined in [10]. Satisfiability [7] as well as compatibility [1] verify whether devices built according to their TD specifications can correctly interact when connected together. It should be noted that, unlike causality and compatibility, neither consistency nor maximum separation time computations take into account the nature (input or output) of events involved in the interface. Distinguishing between input and output events is a relevant information for the verification and synthesis tasks, since only output events can be controlled by the designer. In this work we present a new method for determining whether the interface specified by a TD is realizable by establishing a necessary and sufficient condition on the assumed behavior of the environment. The paper is organized as follows: In Section 2, the necessary background and useful terminology and concepts are defined. In Section 3, the notion of realizability of timing diagrams is defined. In Section 4, our approach to realizability verification is explained. In Section 5, experimental results are discussed and conclusions are drawn.

2. Definitions

An *interface* can be defined by three components: a set of resources called *ports* serving to exchange information between the system and its environment, a set of rules defining a *protocol of communication*, and a set of timing relationships between *events* occurring on the ports. The interface behavior can be specified using timing diagrams (TDs). An *event graph EG* can be associated with each TD: $EG = (E,C)$ where each vertex in E represents an event in TD, and each directed edge in C corresponds to a timing constraint between a pair of events in TD. We denote by $C = \{c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}]) \mid e_i, e_j \in E\}$ the set of constraints, and by $t(e_i)$ the *occurrence time* of the event e_i such that $l_{ij} \le t(e_j) - t(e_i) \le u_{ij}$ *for all* $c_{ij} \in C$. For a constraint $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$, e_i is called the *parent* of e_j and the *source* event of c_{ij}. If an event (node) has more than one parent, then it is said to be a *convergence event* (*node*). Constraints relating events are said to be *linear* if each of them must be satisfied separately, i.e., for each e_i parent of e_j we have:

$l_{ij} \le t(e_j) - t(e_i) \le u_{ij}$ or:

$$max_{e_i \in parents(e_j)}(t(e_i) + l_{ij}) \le t(e_j) \text{ and}$$

$$t(e_j) \le min_{e_i \in parents(e_j)}(t(e_i) + u_{ij}) .$$

A direction is associated with each event: input or output. Denote by I the set of all input events, and O the set of all output events. We have $E = I \cup O$ and $I \cap O = \emptyset$. A timing constraint $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$ is a *commit constraint* if $e_j \in O$, otherwise it is an *assume constraint*. We denote by A (respectively K) the set of assume (commit) constraints over E, $A \cup K = C$. A commit constraint is under the control of the designer, since it concerns an output event to be produced by the system under construction. An assume constraint is guaranteed by the environment, and we cannot force any specific separation time between events e_i and e_j. We denote by *CS-G*, *CS-A*, and *CS-K* the system of equations generated by all timing constraints in C, the assume constraints in A, and the commit constraints in K, respectively.

2.1 Maximum Separation Time

A *separation time* is the difference between the occurrence times of a pair of events: $(s_{ij} = t(e_j) - t(e_i))$. The computation of the maximum separation times between events in a timing diagram does not take into account the nature of events (input or output). Several algorithms have been developed for computing the maximum separation. The complexity of these algorithms depends on the type of the timing constraints allowed[3][4][7][8][9].

2.2 Tightness of Event Graphs

A timing constraint $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$ is said to be *tight* if its bounds correspond exactly to the maximum separation time computed on the whole event graph. i.e. $u_{ij} = s_{ij}$ and $l_{ij} = -s_{ji}$. An event graph is said to be *tight* if all timing constraints are tight.

2.3 Consistency of Event Graphs

An event graph *(E, C)* is *consistent* if and only if the set of n-tuples $(t(e_1), \ldots, t(e_n))$ satisfying *CS-G* is not empty[7].

Note that consistency ensures the existence of an assignment of occurrence times to all events in the event graph when their direction is not taken into account. This however may not guarantee that a given specification is implementable because the exact occurrence times of input events may not be known.

3. Realizability of Event Graphs

Let $EG = (E, C)$ be an event graph, $E = I \cup O$, $C = A \cup K$. $|I| = m$. Let $\boldsymbol{e} = (e_1, \ldots, e_k)$ be a tuple of events in E, and denote by $\boldsymbol{t(e)}$ the vector of occurrence times $(t(e_1), \ldots, t(e_k))$. Let O_s be a set of all output events which constitute the source events of constraints in A, $O_s = \{e_i \in O \mid c_{ij} \in A\}$, $|Os| = q$. For each constraint $c_{ij} \in A$ (respectively K), we write $\delta_{ij} = t(e_j) - t(e_i)$ (respectively $\gamma_{ij} = t(e_j) - t(e_i)$). The interval $[l_{ij}, u_{ij}]$ is denoted by I_{ij}, hence $\delta_{ij} \in I_{ij}$ for linear constraints. We denote by δ the vector of δ_{ij} corresponding to all c_{ij} in A.

Definition 1 A function f from $(R^+)^n$ to R^+ is a causal function if and only if it is a constant function or for each vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in (R^+)^n$ there exist a variable x_i in $\boldsymbol{x}$ such that $f(\boldsymbol{x}) \geq x_i$.

Examples of causal functions are the *min* and *max*.

Definition 2 A function h from O_s to R^+ is causal if there exists q causal functions f_k from $(R^+)^m$ to R^+such that for each event $o_k \in O_s$, (k = 1, …, q), we have $h(o_k) = f_k(\boldsymbol{t(i)})$ where $\boldsymbol{i} = (i_1, \ldots, i_m)$ the vector of all inputs events in I.

The space of the occurrence times of the input events which respect the assume constraints may depend on the occurrence times chosen for the output events in O_s. Hence the possible values of δ_{ij} depend on these choices. Given that $\boldsymbol{t(o)} = \boldsymbol{h(o)}$, where $\boldsymbol{h(o)} = (h(o_1), \ldots, h(o_q))$, we denote by S_h the space of all possible values of the vector δ. $S_h = \{\delta \in D_\delta$ for $c_{ij} \in A \mid$ *CS-A is consistent}* where $D_\delta = \prod_{ij} I_{ij}$. Note that the system of equations *CS_A* involves inevitably $t(o_1), \ldots, t(o_q)$ as parameters.

Definition 3 An event graph $EG = (E, C)$ is said to be *realizable* if and only if there exists a causal function h from O_s to R^+ such that $\boldsymbol{t(o)} = \boldsymbol{h(o)}$, $\boldsymbol{o} = (o_1, \ldots, o_q)$ the vector of events in O_s with $S_h \neq \emptyset$, and $\forall\ \delta \in S_h$, the system (*CS-K*) is consistent.

more details relatively to the realizability characteristic can be found in [10].

4. Realizability Verification Method

For timing diagrams with linear constraints, an event graph *EG = (E, C)* can be represented by a classical directed weighed graph called linear event graph *LEG = (E, C')*. Such representation takes the advantage of using the known algorithms. The relation $l_{ij} \leq t(e_j) - t(e_i) \leq u_{ij}$. can be split into two inequalities $t(e_j) - t(e_i) \leq u_{ij}$ and $t(e_i) - t(e_j) \leq -l_{ij}$. The nodes of *LEG* correspond to the events in E, and the edges correspond the two inequalities as shown in Figure 1. It has been shown that the maximum separation time can be computed by using the shortest paths algorithm [2], and the event graph is consistent if and only if it contains no negative cycles [7]

0-7803-5474-5/99/$10.00(C)1999 IEEE

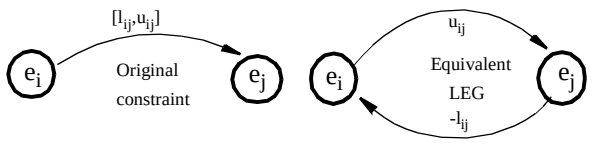

Figure 1 Edges in LEG

Because the occurrence times of input events are under the control of the environment of the system, the consistency property cannot be sufficient to guarantee that the system is realizable. In the following, we suppose that the event graph is consistent and all constraints are tight.

4.1 Extreme Configurations

Definition 4 : Consider an input event z and $P(z)$ the set of its parents. Suppose that $P(z)$ is a singleton and let e_1 be the single parent of z and $(e_1, z, [l_1, u_1])$ the related timing constraint. The *time occurrence* of z can be expressed as : $t(z) = t(e_1) + \delta_1$ with $\delta_1 \in [l_1, u_1]$. The *time occurrence* of z is called *extreme* when it corresponds to one of the bounds of the assume constraint, i.e., $t(z) = t(e_1) + l_1$ or $t(z) = t(e_1) + u_1$. In the case that $P(z)$ contains more than one parent, the lower and the upper bound considered are respectively:

$$max_{e_i \in parents(z)}(t(e_i) + l_i)\,,$$

$$min_{e_i \in parents(z)}(t(e_i) + u_i)\,.$$

In this case, the original constraints can be replaced by: $(e_1, z, [l_1, l_1])$ (respectively $(e_1, z, [u_1, u_1])$) in case of $P(z) = \{e_1\}$ and $(e_i, z, [l_i, \infty])$ (respectively $(e_i, z, [-\infty, u_i])$) for each constraint relating e_i to z in case of $|P(z)| > 1$.

In general case, $t(z)$ depends on the values of δ_i, $\delta_i \in [l_i, u_i]$ such that: $t(e_1) + \delta_1 = t(e_2) + \delta_2 = ... = t(e_n) + \delta_n$. (1)

If $P(z)$ is a singleton, $P(z) = \{e_1\}$, then:

$$t(z) = t(e_1) + \delta_1 \text{ with } \delta_1 \in [l_1, u_1]. \quad (2)$$

Definition 5 : A *configuration* of an event graph is an event graph where the weight of each edge k that corresponds to an assume constraint is replaced by a constant delay δ_k such that equations (1) and (2) are satisfied. A configuration is said to be *extreme* when all the constant delays coincide with the bounds of the corresponding assume constraints.

Proposition 1: An event graph (E,C) is realizable if and only if all configurations relative to input events are consistent.

Outline of the Proof: Let take all configurations relative to input events. using the Definition 5, this is equivalent to the set $Q = \{\delta \in D_\delta$ for $c_{ij} \in A \mid$ equations (1) and (2) are satisfied$\}$. This is equivalent to the set $S = \{\delta \in D_\delta$ for $c_{ij} \in A \mid$ *CS-A is consistent*$\}$. All configurations are consistent i.e., $\forall\ \delta \in Q$, *CS_G* is consistent. Which is the same as the realizability definition.

Theorem: An event graph (E,C) is realizable if and only if all *extreme configurations* relative to input events are consistent.

Outline of the Proof: consider a realizable event graph (E,C), then we have: $\exists\ \boldsymbol{h}$ and $\{\forall\ \delta \in S_h$, *CS-G* is consistent$\}$. Where $S_h = \{\delta \in D_\delta \mid$ *CS-A* is consistent$\}$.

Let E_{config} be the set of vectors $\delta \in D_\delta$ such that $\delta_i \in \{l_i, u_i\}$ and the equations (1) and (2) are satisfied.

We have $E_{config} \subseteq S_h$ ==> $\{\forall\ \delta \in E_{config}$, *CS-G* is consistent$\}$. So all extreme configurations are consistent.

Suppose now that all extreme configurations are consistent. We can prove, for *EGs* with linear constraints that all other configurations are consistent.

For the example of Figure 2 there are four configurations to verify: $\{(o_3, i_2, [10, 10]), (o_4, i_3, [10, 10])\}$;
$\{(o_3, i_2, [10, 10]), (o_4, i_3, [30, 30])\}$;
$\{(o_3, i_2, [20, 20]), (o_4, i_3, [10, 10])\}$; and
$\{(o_3, i_2, [20, 20]), (o_4, i_3, [30, 30])\}$. The second configuration is illustrated in Figure 3. No negative cycles appear in these configurations, so the event graph is realizable.

4.2 Algorithm

Algorithm for testing realizability of event graphs:

Step1: tighten the event graph
/* warnings are generated on each eventual modification of assume constraints*/
Step2: for each input event find the extreme values of corresponding δ_{ij}
Step3:
repeat until all extreme configurations are examined or a inconsistency is determined
{take an extreme configuration relative to input events
check consistency (no negative cycles)}
if all extreme configurations are consistent
then the event graph is realizable
else it is not realizable

In the worst case, the given algorithm has the time complexity of $2^{|I|}$.

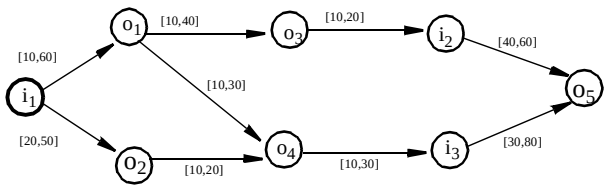

Figure 2 Example1

0-7803-5474-5/99/$10.00(C)1999 IEEE

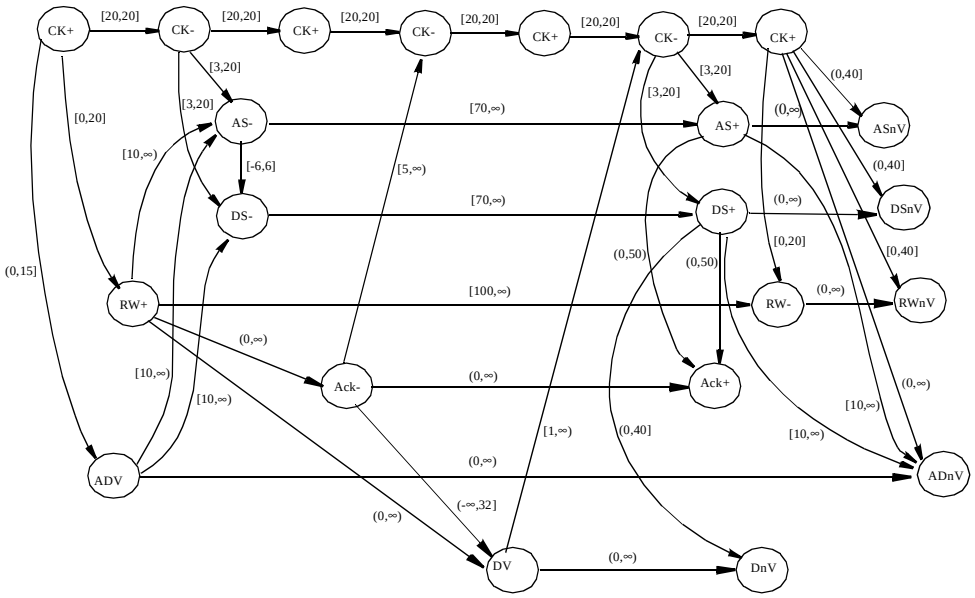

Figure 4 Event graph for MC68360 Read Cycle

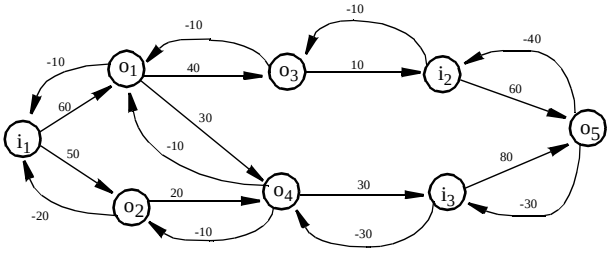

Figure 3 Example of configuration

5. Conclusions and Experimental Results

The algorithm has been tested on realistic examples, such as the read cycle TD of the processor MC68360 (Figure 4). The event graph contains 4 input events. And the resulting extreme configurations are consistent. In summary, we have developed a verification method of the realizability properties of a timing diagram, this method is based on the consistency of the timing diagram for each configuration of occurrence times generated by the environment. The method takes advantage of known techniques developed for weighted graphs.

6. References

[1] E. Cerny, K. Khordoc, "*Semantics and Verification of Action Diagrams with Linear Timing Constraints*", Transactions on Design Automation of Electronic Systems, Vol.3, No.1, Jan.1998.

[2] G. Borriello, "*A New Interface Specification Methodology and its Application to Transducer Synthesis*", Ph.D. Thesis, EECS, UC, Berkeley, 1988.

[3] K.McMillan, D.L.Dill, "*Algorithms for Interface Timing Verification*", IEEE International Conference on Computer Design, 1992, pp.48-51.

[4] T.-Y.Yen, A.Ishii, A.Casavant, W.Wolf, "*Efficient Algorithms for Interface Timing Verification*", the European Design Automation Conference, 1994.

[5] H.Hulgaard, S.M.Burns, T.Amon, G.Borriello, "*An Algorithm for Exact Bounds on the Time Separation of Events in Concurrent Systems*", IEEE Transactions on Computers. Vol.44, No.11, Nov. 1995, pp.1306-1317.

[6] T.Amon, H.Hulgaard, G.Borriello, S.Burns, "*Timing Analysis of Concurrent Systems: An Algorithm for Determining Time Separation of Events*", IEEE International Conference on Computer Design, 1993.

[7] J.A.Brzozowski, T.Gahlinger, F.Mavaddat, "*Consistency and Satisfiability of Waveform Timing Specifications*", Networks, Vol.21, 1991, pp.91-107.

[8] P.Vanbekbergen, G.Goossens, H. De Man, *"Specification and Analysis of Timing Constraints in Signal Transition Graphs"*, the European Conference on Design Automation, 1992, pp.302-306.

[9] P. Girodias, E. Cerny and W.J. Older, "*Solving Linear, Min and Max Constraint Systems Using CLP based on Relational Interval Arithmetic*", Theoretical Computer Science, CP'95, Special Issue, Volume 173, Feb. 1997.

[10] A.El-Aboudi, E-M.Aboulhamid, E.Cerny, "*Synthesis of Interface Controllers from Timing Diagram Specifications*", Custom Integrated Circuits Conference, 1998, pp.89-92.

Acknowledgments: The work was partially supported by an NSERC and Micronet Grant.

0-7803-5474-5/99/$10.00(C)1999 IEEE