Discrete-Time Scheduling under Real-Time Constraints

Eduard Cerny[1], Yuke Wang[2], Mostapha Aboulhamid[1]

[1]Laboratoire LASSO, Dép. d'IRO, Université de Montréal
[2]Dept. ECE, Concordia University
Montréal (Québec) Canada

Abstract

We introduce a new method for scheduling under real-time constraints that is suitable for synchronous system implementations. The input specification is in the form of timing diagrams in which the occurrence times of signal transitions or actions are related by linear constraints, expressing the assumptions on the input actions (the environment) and the commitments on the output actions. Provided that the specification is causal, we give an algorithm for deriving ASAP and ALAP relative schedules for the output actions. We then present a new algorithm for determining whether a given clock period is correct. Based on a schedule and a valid clock period, we transform the specification into a discrete-time relative schedule. Such a schedule serves as the basis for implementing a synchronous state-machine controller.

Keywords: Timing diagrams, relative scheduling, real-time constraints, synchronous state machines.

1 Introduction

High-level synthesis of digital hardware for real-time applications [10, 22], as well as the synthesis of interface transducers and controllers [2, 4, 11, 12], require to perform scheduling of actions and operations under explicit real-time constraints. In [22], operations with unknown duration were modeled using unbounded delays from the start of such an operation to the start of any successor operation that depended on it. Linear timing constraints were used to restrict the occurrence times of operations relative to each other. Since linear constraints may impose restrictions on the time separation of the anchor operations, thus making the specification unrealizable for all possible durations of the anchor operation, the authors of [22] defined the so-called well-posedness condition of the timing constraints of the specification. In interface transducer synthesis, the situation is to some extent similar, in that events are constrained by timing onstraints (linear, latest, and earliest) [4, 7]. Here, events usually represent signal transitions or enabling situations for data to appear on busses. The synthesis has, however, much in common with the problem of scheduling under real-time constraints used in high-level synthesis.

In this paper, we describe a relative scheduling method for specifications inspired by timing diagrams as formalized in [14, 15, 19, 21]. In summary, the main distinguishing charactersitics of our approach are as follows:

- Instead of dealing with operations (that have duration) and signal transitions (events), we consider only instantaneous actions. These then can be used in pairs to delimit the start and the end of an operation, or to model individually some specific events or signal transitions in the system. This approach has been quite effective in modeling systems using (real-time) process algebras.

- We explicitly distinguish output (or internal) actions whose occurrence time is under the control of the synthesized device, and input actions whose occurrence time is controlled by the environment. In both cases, the occurrence time of an action may be restricted by the occurrence time of some preceding actions (input or output). We allow linear, latest and earliest type of timing constraints, however, in this paper we consider only linear constraints, since these are the source of noncausality in TD specifications and require special techniques for deriving schedules.

- To provide means for describing timing assumptions that can be made on the occurrence time of input actions, and the limits on the reaction time of the output actions, we distinguish two kinds of timing constraints, assume and commit, respectively. This allows to state what assumptions are made on the environment and then take this information into account during synthesis (assumption-based reasoning); this is more realistic than assuming that any duration of, say, an input operation is possible [22].

- Allowing finite assumption constraints requires a generalization of the well-possedness condition of the system of constraints. We have recognized this in the context of interface specifications and their compatibility verification [15] as a problem of causality. Causality conditions allow us to use a more complex structure of a specification than in [22] such that the input and output actions can be intermixed as in timing diagrams.

The contributions of the present paper are:

- A method is presented for deriving a relative schedule for output and internal actions, relative to the so-called trigger actions [15]. We can guarantee the existence of the schedule only if the specification is causal.

- The timing constraints can be given in dense time (the reals), however, hardware implementation is often carried out using synchronous design techniques where all operations are synchronized to a common clock of period C. We present a method for verifying that a period C is valid, and then produce a discrete-time relative schedule for the output actions in which the time unit is the clock tick. Such a schedule can be used in existing synthesis methods, e.g., [10, 22].

This methodology is centered around the so called Hierarchical Annotated Action Diagrams (HAAD) [17, 18]. Leaf diagrams correspond to the timing diagrams used as the basis for synthesis in this paper. The hierarchical diagrams form more complex behaviors using composition operators over leaf and other hierarchical diagrams. The operators are inspired by process algebras (parallel with causal rendez-vous communication, concatenation, loop, delayed choice, and exception handling). All diagrams can also be annotated by VHDL procedures, variables, and predicates. For a useful subset of this specification language we have defined the formal semantics and provided axiomatization [19], including conversion to a normal form that can be used to transform the specification to a network of Timed Automata for verification using existing tools. Furthermore, the verification of causality and compatibility on leaf-level diagrams can be efficiently carried out using Constraint Logic Programming based on Relational Interval Arithmetic [20, 21].

The paper is organized as follows: Section 2 introduces basic concepts about timing diagrams, causality, and scheduling. Section 3 describes the determination of a valid clock period and the derivation of a discrete-time relative schedule; it also contains a complete example. Section 4 concludes the presentation.

2 Background Information

2.1 Timing diagrams and the causality property

In this section, we introduce some background concepts about timing diagrams, event graphs of timing diagrams, and the causality property [15, 17].

Definition 1: There is a global time variable T that increases monotonically. The *current time* is the value of the variable T. Initially, the global time variable is reset to some real value τ. Let $E = \{e_1, e_1, \ldots, e_n\}$be a set of events. A time-stamp variable t_i is associated with each event e_i; $t_i = x$ means that e_i occurs when the value of the time variable T becomes x. Current time and time stamps take on finite, possibly unbounded, real values.

Definition 2: Let $E = \{e_1, e_1, \ldots, e_n\}$ be a set of events. $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$ represents the constraint $l_{ij} \le t_j - t_i \le u_{ij}$ on the separation between the occurrence times of $e_i, e_j \in E$. We call e_i the source and e_j the sink of the constraint. A constraint $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$ is a *precedence* constraint if $u_{ij} \ge l_{ij} > 0$, and a *concurrency* constraint if $u_{ij} \ge 0 \ge l_{ij}$.

Let $E_{in} \subseteq E$ the set of all input events, and $E_{out} \subseteq E$ the set of all output events, $E_{in} \cap E_{out} = \varnothing$, $E = E_{in} \cup E_{out}$

Definition 3: A timing diagram TD = (E, C) is determined by its set of events E and the set CS of constraints over E. A constraint $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}]) \in C$ such that e_i and e_j are of different directions must be a precedence constraint.

The constraints in CS have one of two possible **intents**. It is an *assume* constraint if the sink is an *input* event; otherwise it is a *commit* constraint. Assume constraints delimit the expected or assumed behavior of the environment, while the commit constraints define the limits on the occurrence times of the output events. In other words, the device implementation must satisfy the commit constraints, provided that the environment satisfies the assumptions.

Definition 4: Consider a timing diagram TD = (E, C) with $c_{ij} = (e_i, e_j, [l_{ij}, u_{ij}])$. The **event graph** $EG = (V, E_g)$ of TD is a directed weighted graph where the vertex set V is E, and for each constraint $c = (e_i, e_j, [l_{ij}, u_{ij}]) \in C$ in TD, there are two edges in E_g, (e_i, e_j) labeled by u_{ij}, and (e_j, e_i) labeled by $-l_{ij}$; the label is called the weight of the edge.

An edge (e_i, e_j) in EG labeled by weight w_{ij} represents the constraint $t_j - t_i \le w_{ij}$. That is, a two-sided constraint in TD is represented by two one-sided constraints in EG.

Example 1: Figure 1(a) shows a TD where $E_{out} = \{e_1, e_3\}$, $E_{in} = \{e_2, e_4\}$. The constraints $(e_1, e_2, [l_{12}, u_{12}])$ and $(e_1, e_3, [l_{13}, u_{13}])$ are assume constraints, the other two are commit constraints. Its corresponding EG is given by Figure 1(b).

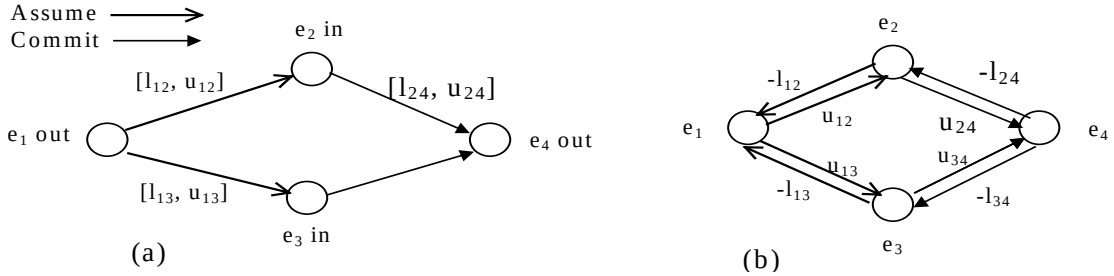

Figure 1: (a) Sample Timing Diagram TD **(b)** its corresponding Event graph.

A path $p(e_i, e_j)$ in EG from e_i to e_j is a sequence of edges $(e_i, e_{j1})(e_{j1}, e_{j2}) \ldots (e_{jk-1}, e_{jk})(e_{jk}, e_j)$. A cycle is a path $p(e_i, e_i)$. The weight of a path is the sum of the weights of all edges along the path. The shortest path from e_i to e_j is the path whose weight is the smallest from among all the paths from e_i to e_j. The constraint system *CS* is consistent if there is no cycle with negative weight in EG, otherwise *CS* is inconsistent [3].

An event graph can be checked alone for consistency which is a minimal form of realizability [3, 14, 15]. It assures that the constraint system *CS* has a solution and thus an occurrence time can be assigned to every event. As pointed out in [14, 15, 17], the notion of consistency of the event graph of a TD is insufficient for constructing correct implementations. In fact, inconsistency is a special case of non-causality. We now introduce causal TDs [15, 17]:

Definition 5: In an EG, the maximum separation time from event e_j to event e_i is defined as max $(t_i - t_j)$, where t_i and t_j satisfy the TD constraints [9]. If max $(t_i - t_j) < 0$, then event e_i *strictly precedes* event e_j. It is well known that the maximum separation from e_j to e_i is the shortest distance from e_j to e_i in EG [3].

We describe next the execution semantics of the model underlying a TD specification. They are based on the notion of a block of events.

Definition 6: Consider the event graph EG of a timing diagram TD = (*S*, *E*, *C*). Let $\{EB_i\}$ be a partition over the event set $E = \cup EB_i$, $EB_i \cap EB_j = \emptyset$, $\forall i$, $EB_i \subseteq E_{in}$ or $EB_i \subseteq E_{out}$. Each EB_i is called an *event block*.

Let $E_{ij} = \{e_k \in EB_j \mid \exists\, e_l \in EB_i$ such that there is an edge from e_k to e_l or from e_l to $e_k\}$, i.e., E_{ij} contains all events from EB_j related by a constraint to some event in EB_i. The block EB_j is the *predecessor* of the block EB_i, denoted by $EB_j = \text{pred}(EB_i)$, if events $e_j \in E_{ij}$ strictly precede all events $e_i \in EB_i$, i.e., max $(t_j - t_i) < 0$. In this case, the events in E_{ij} are called the *triggers* of EB_i in EB_j. The *local constraints* of EB_i are those constraints of CS that (1) relate pairs of events in EB_i or (2) relate events in EB_i to its triggers. The partition $\{EB_i\}$ must also satisfy the following property:

Property 1: For all pairs of blocks $EB_i, EB_j \in \{EB_i\}$, if $E_{ij} \neq \emptyset$, then either $EB_i = \text{pred}(EB_j)$ or $EB_j = \text{pred}(EB_i)$. In a consistent *CS*, the relation *pred* induces a partial order (<) over blocks.

An event block EB_i is *enabled* when all its trigger events have occurred. EB_i becomes enabled at time t if the last trigger(s) occurred at t. An enabled event block EB_i is *fixed* when the occurrence times of all its events are assigned a value such that the local constraints of the block are satisfied, given the occurrence times of the triggers. If no such assignment exists then the block cannot be fixed.

Definition 7: A partition $\{EB_i\}$ of EG is causal iff it satisfies Property 1 and every event block can be enabled and fixed. A TD is causal if its EG has a causal partition.

Theorem 1 ([15, 17]): A TD is causal iff for each pair of triggers of each block the maximum separation between the triggers as computed using the local constraints of the block is strictly greater than the maximum separation of that pair computed over the entire EG.

In the rest of this paper, we assume that all TDs are causal with a partition $\{EB_i\}$.

2.2 Scheduling of events under TD constraints

In the preceding section, we introduced two kinds of constraints in TDs: assume and commit. We can schedule only the output events controlled by the commit constraints, since the input events related by the assume constraints are controlled by the environment. In this section, we present scheduling algorithms for output events of a causal TD. Some of the concepts introduced here are similar as in [10, 22].

Definition 8: A schedule of a TD is a function that assigns an occurrence time to each output event such that all commit constraints in the timing diagram are satisfied, given any occurrence times of the input events satisfying the assume constraints and the occurrence times of preceding output events. Such an assignment of occurrence times is called a *valid assignment*.

We first describe a method to fix an output block, assuming that all its triggers have occurred. The complete schedule for a causal TD can then be obtained block by block, following any total order derived from the partial order between blocks (Section 2.1).

Consider a block $EB = \{e_i\}$ with its trigger set $Tr = \{Tr_j\}$, and let the occurrence time of trigger Tr_j be T_j. The occurrence times t_i of events $e_i \in EB$ is a function of T_j given by the local constraints of *EB*: $-w_{ki} \leq t_k - t_i \leq w_{ik}$ and $-w_{ij} \leq t_i - T_j \leq w_{ji}$ Let σ_{jk} be the shortest distance from Tr_j to e_k, and σ_{kj} the shortest distance from e_k to Tr_j.

Lemma 1:

(1) For any event $e_1 \in EB$ and a trigger Tr_j of *EB* the following relations hold: $-w_{1j} \leq \sigma_{j1} \leq w_{j1}$ and $-w_{j1} \leq \sigma_{1j} \leq w_{1j}$, where w_{j1} and w_{1j} are the weights between e_1 and Tr_j.

(2) For any two events e_1 and e_2 in EB, and an edge from e_1 to e_2 with weight w_{12}, the relations $\sigma_{j2} \leq w_{12} + \sigma_{j1}$ and $\sigma_{2j} \leq w_{21} + \sigma_{1j}$ hold.

Proof:

(1) This follows directly from the definition of the shortest path $\sigma_{j1} \le w_{j1}$ since w_{j1} is a path weight, and in a consistent constraint graph we must have $\sigma_{j1} + w_{1j} \ge 0$ (no negative cycle), similarly for σ_{1j}.

(2) Suppose that $\sigma_{j1} + w_{12} < \sigma_{j2}$. Therefore, the shortest distance from Tr_j to e_2 is not σ_{j2} because the path underlying σ_{j1} and the edge from e_1 to e_2 form a shorter path - contradiction. Q.E.D.

Lemma 2: In a causal TD, for all events $e_i \in EB$, its triggers and the local constraints of EB, the following holds: $\max_{Tr_j \in Tr} \{T_j - \sigma_{ij}\} \le \min_{Tr_j \in Tr} \{T_j + \sigma_{ji}\}$, where T_j is the occurrence time of trigger Tr_j.

Proof: Let Tr_1 and Tr_2 be any two triggers of EB and $e_i \in EB$. $\sigma_{1i} + \sigma_{i2}$ is the distance of the shortest path from Tr_1 to Tr_2 using local constraints of EB and passing through e_i. Since the system is causal, by Theorem 1, we have $T_2 - T_1 < \sigma_{1i} + \sigma_{i2}$, which induces the condition $T_2 - \sigma_{i2} < \sigma_{1i} + T_1$. This holds for any pair Tr_1, Tr_2; therefore, $\max_{Tr_j \in Tr} \{T_j - \sigma_{ij}\} \le \min_{Tr_j \in Tr} \{T_j + \sigma_{ji}\}$. Q.E.D.

Proposition 1: For all events $e_i \in EB$ and all triggers $Tr_j \in Tr$ of EB, the following holds:

$$\max_{Tr_j \in Tr} \{T_j - \sigma_{ij}\} \le t_i \le \min_{Tr_j \in Tr} \{T_j + \sigma_{ji}\}.$$

Proof: For any trigger Tr_j and event e_i, $t_i - T_j \le \sigma_{ji}$, i.e., $\forall Tr_j \in Tr$, $\sigma_{ji} + T_j \ge t_i$, hence $t_i \le \min_{Tr_j \in Tr} \{T_j + \sigma_{ji}\}$. We can prove the other half of the inequality in a similar fashion. Q. E. D.

Corollary 1: For all $e_i \in EB$, $t_i = \min_{Tr_j \in Tr} \{T_j + \sigma_{ji}\}$ is a valid occurrence time assignment. So is $t_i = \max_{Tr_j \in Tr} \{T_j - \sigma_{ij}\}$, *i.e.*, either min or max is used for all e_i, but not mixed within one block, in general.

Proof: We need to prove that for any trigger Tr_j, and any pair of events e_1 and e_2, the following conditions are satisfied (1) $t_2 - t_1 \le w_{12}$, and (2) $t_1 - T_j \le w_{j1}$ and $T_j - t_1 \le w_{1j}$.

We first prove (1). Since $t_1 = \min_{Tr_j \in Tr} \{T_j + \sigma_{j1}\}$, there exist triggers Tr_a and TR_b such that $t_1 = T_a + \sigma_{a1} = \min_{Tr_j \in Tr} \{T_j + \sigma_{j1}\}$ and $t_2 = T_b + \sigma_{b2} = \min_{Tr_j \in Tr} \{T_j + \sigma_{j2}\}$. Based on the definition of t_1 and t_2, we have

$T_a + \sigma_{a1} \le T_b + \sigma_{b1}$
$T_b + \sigma_{b2} \le T_a + \sigma_{a2}$, and then
$t_2 - t_1 \le [T_a + \sigma_{a2}] - [T_a + \sigma_{a1}] = \sigma_{a2} - \sigma_{a1} \le w_{12}$, and
$t_1 - t_2 \le [T_b + \sigma_{b1}] - [T_b + \sigma_{b2}] = \sigma_{b1} - \sigma_{b2} \le w_{21}$, both by Lemma 1.

Hence, (1) is proven. Next we prove (2). Let Tr_j be an arbitrary trigger of EB. Since $t_1 = T_a + \sigma_{a1} = \min_{Tr_j \in Tr} \{T_j + \sigma_{j1}\}$, we have $\forall Tr_j$, $T_j + \sigma_{j1} \ge t_1$. By Lemma 2, $t_1 \ge \max_{Tr_j \in Tr} \{T_j - \sigma_{1j}\} \ge T_j - \sigma_{1j}$. Hence, $T_j - \sigma_{1j} \le t_1 \le T_j + \sigma_{j1}$). It follows that

$t_1 - T_j = [T_j + \sigma_{j1}] - T_j = \sigma_{j1} \le w_{j1}$, and $T_j - t_1 \le \sigma_{1j} \le w_{1j}$ Q. E. D.

Definition 9: Denote the shortest distance from e_k to Tr_j as $\sigma^S(e_k, Tr_j) = \sigma_{kj}$ and be the shortest distance from Tr_j to e_k as $\sigma^L(e_k, Tr_j) = \sigma_{jk}$. The time interval $[\max_j \{T_j + \sigma^S(e_i, Tr_j)\}, \min_j \{T_j + \sigma^L(e_i, Tr_j)\}]$ is called the *feasible interval of* e_i, denoted by $[S_{e_i}, L_{e_i}]$. Moreover, S_{e_i} and L_{e_i} are called the as-soon-as-possible (ASAP) and the as-late-as-possible (ALAP) types of relative schedules of the output events, respectively.

3 Discrete-Time Schedules

We discuss here the conversion of the dense-time TD specification into a set of discrete-time relative schedules that can be implemented using sampled input synchronous finite state machines synchronized by a clock of period C. Such a machine can be used as the interface controller between the environment and a synchronous device that runs from the same clock as the controller, or as a controller in high-level synthesis under real-time constraints if the events of a TD represent the activation and deactivation of some high-level operations. Figure 2 illustrates a sampled input synchronous FSM. We will give an algorithm to determine whether a clock period of the FSM is valid for implementing the controller. Then, we shall present an algorithm for translating the timing diagram specification into its scheduled version in discrete time where the time unit is a clock tick.

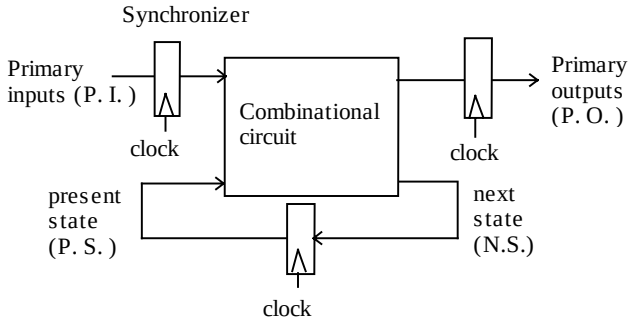

Figure 2: A sampled input Moore FSM.

Since in general the inputs are not synchronous with the FSM clock, they must be first synchronized. We assume that the simplest synchronizer is used, consisting of a synchronous sampling register that introduces a one-cycle delay. The proposed solution can be adapted to the case where a more complex synchronizer that introduces a delay of k > 1 cycles is used. In order not to miss any input signal transitions, we must make sure that the clock has a sufficiently short period to sample the input signals between any two consecutive changes as defined by the assume constraints of the TD. For controlling the output signals in time as specified by the TD, we also place a register at the outputs so as to have better control over the combinational output delays. Due to this register, any output change must be scheduled at least one clock cycle after detecting the last trigger event.

The inputs to the FSM are the sampled input values. It means that input events must be determined by examining the difference between consecutive sampled input values. Even though we can thus detect the occurrence of input events, we cannot determine their exact occurrence times; it is within some time interval determined by the TD and the clock period. In the next section we show how to find this interval and how to determine whether a given clock period is valid.

3.1 Clock Period Determination

In the method introduced in [11], the clock period is specified by the designer, and the tool is to verify that the period is consistent with the constraints. However, there is no algorithm given to do that. A similar problem exists in [13] where the synthesis of timed VHDL processes is discussed. In [18], to determine if C is valid, the state machine of the controller must be constructed first. This complex synthesis task thus must be carried out to find out that the solution is infeasible. In addition, the method cannot handle linear assume and commit timing constraints. In our approach, the validity of C is determined by analyzing the timing constraints only.

Recall that an event block $EB = \{e_1, \cdots, e_n\}$ has a trigger set $Tr = \{Tr_1, Tr_2, \ldots, Tr_m\}$. An occurrence time assignment to events of an event block is a function that determines the value of each t_i such that all the local constraints of the timing diagram are satisfied given the occurrence times of the triggers of the block. To determine that a number *C* is a valid clock period, we have to check whether there is an occurrence time assignment that satisfies all the constraints with respect to *C*. We thus consider the following two questions: (1) Determine if *C* is a valid clock period, and (2) find a discrete-time schedule in which the unit of time is *C*. Based on the solution to (1), we can use a binary search to find the largest valid C.

Every true trigger occurrence time has an associated sampling time at which it is detected. Let $\tilde{T}_j$ be the sampling time of Tr_j whose real occurrence time is T_j. The true trigger time and the associated sampling time must satisfy the following set of constraints, where Tr_i and Tr_j are arbitrary triggers.

$$\tilde{T}_j = m_j C, \qquad m_j > 0 \text{ is an integer} \tag{1}$$

$$0 \le \tilde{T}_j - T_j < C \tag{2}$$

$$-w_{ij} \le T_i - T_j \le w_{ji} \tag{3}$$

Relation (1) means that the sampling time of triggers can only happen at multiples of the clock period, Relation (2) states that the difference between the sampling time and the real time of a trigger is in the interval [0, C), and Relation (3) constrains the difference between two different trigger occurrence times to be in the intervals as given in the event graph.

Definition 10: The set of possible true trigger times associated with each sampling time is $S_{(\tilde{T}_1,\cdots,\tilde{T}_m)} = \{[T_1, \cdots, T_m] \mid 0 \le \tilde{T}_j - T_j < C; -w_{ij} \le T_j - T_i \le w_{ji}\}$. Let $\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j$ be the least value such that for any

$[T_1, \cdots, T_m] \in S_{(\tilde{T}_1, \cdots, \tilde{T}_m)}$, $T_j \le \max_{(\tilde{T}_1, \cdots, \tilde{T}_m)} T_j$. Similarly let $\min_{(\tilde{T}_1, \cdots, \tilde{T}_m)} T_j$ be the greatest value satisfying $T_j \ge \min_{(\tilde{T}_1, \cdots, \tilde{T}_m)} T_j$.

Example 2: Consider the event graph shown in Figure 3 and assume that each event is on a different port. Let Tr_1 and Tr_2 be input events and O an output event. Without loss of generality, we can assume that the sampling time of the input Tr_1 is $\tilde{T}_1 = 0$. Let C= 3. The sampling times of Tr_2 can be $\tilde{T}_2$ =3, 6 or 9, relative to $\tilde{T}_1$ =0.

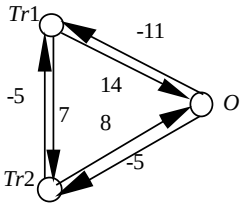

Figure 3: Event Graph of Example 2

For $\tilde{T}_1 = 0$ and $\tilde{T}_2 = 3$, the true trigger times T_1 and T_2 satisfy $5 \le T_2 - T_1 \le 7$ (from (3)), and $-3 < T_1 \le 0$ and $0 < T_2 \le 3$ (from (2)). The associated true trigger time set is $S_{0,3} = \{(T_1, T_2) \mid 5 \le T_2 - T_1 \le 7, \ -3 < T_1 \le 0, \ 0 < T_2 \le 3\}$ with $\max_{(0,3)} T_2 = 3$, $\min_{(0,3)} T_2$= 2, $\max_{(0,3)} T_1$ = -2, $\min_{(0,3)} T_1$ = -3.

Similarly, for $\tilde{T}_1 = 0$ and $\tilde{T}_2 = 6$, the true trigger time set is $S_{0,6} = \{(T_1, T_2) \mid 5 \le T_2 - T_1 \le 7, \ -3 < T_1 \le 0, \ 3 < T_2 \le 6\}$. Therefore, $\max_{(0,6)} T_2 = 6$ and $\min_{(0,6)} T_2 = 3$. $\max_{(0,3)} T_1$ = 0, $\min_{(0,3)} T_1$ = -3.

Finally, for $\tilde{T}_1 = 0$ and $\tilde{T}_2 = 9$, the true trigger time set is $S_{0,9} = \{(T_1, T_2) \mid 5 \le T_2 - T_1 \le 7, -3 < T_1 \le 0, 6 < T_2 \le 9\}$, yielding $\max_{(0,9)} T_2 = 7$ and $\min_{(0,9)} T_2 = 6$, as shown by the triangle GFE, $\max_{(0,3)} T_1$ = 0, $\min_{(0,3)} T_2$ = -1

Since scheduling the occurrence time of any output event $e_i \in EB = \{e_1, \cdots, e_n\}$ can be done only in multiples of C relative to the sampling trigger times, we use $\tilde{t}_i$ to represent the synchronized occurrence time of the outputs. The following must hold for any trigger $Tr_j \in \{Tr_1, Tr_2, \cdots, Tr_m\}$:

$$\tilde{t}_i = \tilde{T}_j + k_{ji} C \qquad (4)$$

This states that all the possible trigger times in $S_{(\tilde{T}_1, \cdots, \tilde{T}_m)}$ share the same schedule for the output events; moreover, the time difference between the output event and the sampling time of the triggers is a multiple of the clock period. It follows from (4) that for any two output events e_1 and e_2, $\tilde{t}_2 - \tilde{t}_1$ is divisible by C. The constraints $\tilde{t}_2 - \tilde{t}_1 \le w_{12}$ can thus be modified as $\tilde{t}_2 - \tilde{t}_1 \le \lfloor w_{12} / C \rfloor C$, i.e., the weight w_{12} can be changed to $\lfloor w_{12} / C \rfloor C$.

Finally, the local constraints of the block must be satisfied. For any output events e_1 and e_2, and any trigger Tr_j, the following relations must be satisfied:

$$\tilde{t}_2 - \tilde{t}_1 \le \lfloor w_{12} / C \rfloor C \qquad (5)$$

$$-w_{1j} \le \tilde{t}_1 - T_j \le w_{j1} \qquad (6)$$

Based on the results of output scheduling (Proposition 1), for each trigger time T_j such that $[T_1, \cdots, T_m] \in S_{(\tilde{T}_1, \cdots, \tilde{T}_m)}$, the possible occurrence time assignments to an output event e_1 must thus be in the interval

$$\tilde{t}_1 \in [\max_j \{T_j + \sigma^s(e_1, Tr_j)\}, \min_j \{T_j + \sigma^L(e_1, Tr_j)\}].$$

Therefore if C is to be a valid clock period, then for each output event e_1, we have

$$\tilde{t}_1 \in \bigcap_{[T_1, \cdots, T_m] \in S_{(\tilde{T}_1, \cdots, \tilde{T}_m)}} [\max_j \{T_j + \sigma^s(e_1, Tr_j)\}, \min_j \{T_j + \sigma^L(e_1, Tr_j)\}],$$

where $\cap$ is the intersection of intervals defined as [a,b] $\cap$ [c,d] = [max(a,c), min(b,d)]. If max(a,c) > min(b,d) then [a,b] $\cap$ [c,d] =$\varnothing$. It follows that

$$\tilde{t}_1 \in \bigcap_{[T_1, \cdots, T_m] \in S_{(\tilde{T}_1, \cdots, \tilde{T}_m)}} [\max_j \{T_j + \sigma^s(e_1, Tr_j)\}, \min_j \{T_j + \sigma^L(e_1, Tr_j)\}] \stackrel{def}{=} [S_{(\tilde{T}_1, \cdots, \tilde{T}_n)}(e_1), L_{(\tilde{T}_1, \cdots, \tilde{T}_n)}(e_1)]$$

where

$$S_{(\tilde{T}_1,\cdots,\tilde{T}_n)}(e_1) = \max_{[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}} \{\max_j \{T_j + \sigma^S(e_1, Tr_j)\}\} = \max_j \{\max_{[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}} \{T_j\} + \sigma^S(e_1, Tr_j)\}$$

$= \max_j \{\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^S(e_1, Tr_j)\}$, for any $[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}$, $T_j \le \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j$, and

$$L_{(\tilde{T}_1,\cdots,\tilde{T}_n)}(e_1) = \min_{[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}} \{\min_j \{T_j + \sigma^L(e_1, Tr_j)\}\} = \min_j \{\min_{[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}} \{T_j\} + \sigma^L(e_1, Tr_j)\}.$$

$= \min_j \{\min_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^L(e_1, Tr_j)\}$, for any $[T_1,\cdots,T_m]\in S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}$, $T_j \ge \min_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j$.

Proposition 2: Given a set of sampling times $\{\tilde{T}_j = m_j C \mid Tr_j \in Tr = \{Tr_1,\cdots,Tr_m\}\}$, if for all output events e_i, the relation

$$\left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)/C \right\rceil C \le L_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i) \qquad (7)$$

is satisfied, then the set $\{\left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)/C \right\rceil C \mid e_i \in EB\}$ is a valid ASAP occurrence time assignment for the events in the output event block. If (7) does not hold, then there is no occurrence time that satisfies all the constraints.

Proof: If $\left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)/C \right\rceil C > L_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)$, then no value in the interval $[S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i), L_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)]$ is divisible by *C*. Therefore, in this case, there is no valid occurrence time assignment for e_i as a multiple of C relative to the sampled occurrence times of the triggers.

Note that the ASAP schedule expressed in clock cycles must be greater or equal to one, to take into account the delay introduced by the output register.

Suppose now that (7) holds. We need to prove that the set $\{\left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)/C \right\rceil C \mid e_i \in EB = \{e_1,\cdots,e_n\}\}$ is a valid occurrence time assignment for the events in EB, i.e., for any trigger Tr_j and any pair of events e_1 and e_2, the following conditions must be satisfied: (1) $t_2 - t_1 \le \lfloor w_{12}/C \rfloor C$; and (2) $-w_{j1} \le t_1 - T_j \le w_{j1}$. However, $t_1 = \left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_1)/C \right\rceil C = \left\lceil [\max_j \{\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^S(e_1, Tr_j)\}]/C \right\rceil C$, and thus there exists triggers Tr_1 such that $t_1 = \{\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_1 + \sigma^S(e_1, Tr_j)\} + \Delta_1$ and Tr_2 such that $t_2 = \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_2 + \sigma^S(e_2, Tr_2) + \Delta_2$, where Δ_1 and Δ_2 are two non-negative numbers less than *C* such that t_1 and t_2 can be divided by *C*. It follows that :

$$\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_1 + \sigma^S(e_1, Tr_1) + \Delta_1 \ge \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_2 + \sigma^S(e_1, Tr_2) \qquad \text{(i)}$$

$$\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_2 + \sigma^S(e_2, Tr_2) + \Delta_2 \ge \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_1 + \sigma^S(e_2, Tr_1) \qquad \text{(ii)}$$

Therefore, for an arbitrary trigger Tr_j,

$$\begin{aligned} \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^S(e_1, Tr_j) \le t_1 &= \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_1 + \sigma^S(e_1, Tr_j) + \Delta_1 \\ &\le L_{(\tilde{T}_1,\cdots,\tilde{T}_n)}(e_1) \le \min_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^L(e_1, Tr_j) \end{aligned} \qquad \text{(iii)}$$

Based on (i) and (ii), the following deduction is easy to follow.

$$t_2 - t_1 \le [\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_2 + \sigma^S(e_2, Tr_2) + \Delta_2] - [\max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_2 + \sigma^S(e_1, Tr_2)]$$

$$= \sigma^S(e_2, Tr_2) - \sigma^S(e_1, Tr_2) + \Delta_2 \le \lfloor w_{12}/C \rfloor C + \Delta_2$$

On the other hand, $t_2 - t_1$ and $\lfloor w_{12}/C \rfloor C$ are divisible by *C*, $\Delta_2 < C$, thus $t_2 - t_1 \le \lfloor w_{12}/C \rfloor C$.

We now prove (2). Based on (iii), the following inequalities hold.

$$t_1 - T_j \ge \max_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^S(e_1, Tr_j) - T_j \ge \sigma^S(e_1, Tr_j) \ge -w_{1j}$$

$$t_1 - T_j \le \min_{(\tilde{T}_1,\cdots,\tilde{T}_m)} T_j + \sigma^L(e_1, Tr_j) - T_j \le \sigma^L(e_1, Tr_j) \le w_{j1}. \qquad \text{Q.E.D.}$$

Corollary 2: For *C* to be a valid clock period, condition (7) must be satisfied for all possible sampling times of the triggers.

Example 3: Consider again the event graph in Figure 3. We wish to determine whether C = 3 is valid. The possible trigger sampling times are $\tilde{T}_1$ = 0; $\tilde{T}_2$ = 3, 6, and 9. For each possible $(\tilde{T}_1, \tilde{T}_2)$, we verify (7) using Proposition 2:

$S_{0,3}(o) = \max\{\max_{(0,3)} T_1 + 11,\ \max_{(0,3)} T_2 + 5\} = \max\{-2+11, 3+5\} = 9, \lceil S_{0,3}(o)/3 \rceil = 3$

$L_{(0,3)}(o) = \min\{\min_{(0,3)} T_1 + 14, \min_{(0,3)} T_2 + 8\} = \min(-3+14, 2+8) = 10$

$S_{0,6}(o) = \max\{\max_{(0,6)} T_1 + 11,\ \max_{(0,6)} T_2 + 5\} = \max\{0+11, 6+5\} = 11, \lceil S_{0,6}(o)/3 \rceil = 4$

$L_{(0,6)}(o) = \min\{\min_{(0,6)} T_1 + 14, \min_{(0,6)} T_2 + 8\} = \min(-3+14, 3+8) = 11.$

Since $3 * \lceil S_{0,6}(o)/3 \rceil = 12 > 11 = L_{0,6}(o)/3$, there is no feasible time assignment for *o*: when the sampling trigger times are $(\tilde{T}_1, \tilde{T}_2)$ = (0, 6), the true trigger times could be (0, 6); therefore, to satisfy the two constraints, the output time has to be greater than 11 and divisible by 3, which is 12. Furthermore, the true trigger times could also be (-2.5, 3.1). If the output time is 12, the difference between Tr_1 and *o* is 14.5 >14, violating the maximum bound of 14 on the separation between *o* and Tr_1.

If we change the constraints by replacing $(o, Tr_1) = -11$ by $(o, Tr_1) = -12$; $(Tr_2, o) = 8$ by $(Tr_2, o) = 10$, $(Tr_1, o) = 14$ by $(Tr_1, o) = 15$, we can then verify that C = 3 becomes a valid clock period.

$S_{0,3}(o) = \max\{\max_{(0,3)} T_1 + 12,\ \max_{(0,3)} T_2 + 5\} = \max\{-2+12, 3+5\} = 10, \lceil S_{0,3}(o)/3 \rceil = 4$

$L_{(0,3)}(o) = \min\{\min_{(0,3)} T_1 + 15, \min_{(0,3)} T_2 + 10\} = \min(-3+15, 2+10) = 12$

$S_{0,6}(o) = \max\{\max_{(0,6)} T_1 + 12,\ \max_{(0,6)} T_2 + 5\} = \max\{0+12, 6+5\} = 12, \lceil S_{0,6}(o)/3 \rceil = 4$

$L_{(0,6)}(o) = \min\{\min_{(0,6)} T_1 + 15, \min_{(0,6)} T_2 + 10\} = \min(-3+15, 3+10) = 12.$

$S_{(0,9)}(o) = \max\{\max_{(0,9)} T_1 + 12, \max_{(0,9)} T_2 + 5\} = \max\{0+12, 7+5\} = 12, \lceil S_{0,9}(o)/3 \rceil = 4$

$L_{(0,9)}(o) = \min\{\min_{(0,9)} T_1 + 15, \min_{(0,9)} T_2 + 10\} = \max\{-1+15, 6+10\} = 14.$

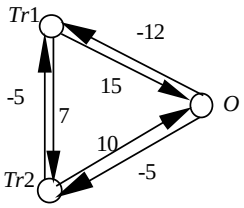

Figure 4: Modified event graph

Therefore the time assignment $\tilde{t}_o = \tilde{T}_1 + 12$ satisfies all the possible input sampling times.

To determine the occurrence time of the events in an output block given a valid C, we only need to know the maximum separations $\sigma^s(e_1, Tr_j)$ and $\sigma^L(e_1, Tr_j)$ between the output events and the triggers, where the maximum separations are computed over the commit constraints with weights adjusted to multiples of the clock period, i.e., as $\lfloor w_{12} / C \rfloor C$. Once these maximum and minimum separations are computed, there is no need to keep the constraints between the output events, because the occurrence time assignment based on the maximum and minimum separations satisfies all the original timing constraints between the output events, provided that the inputs satisfy all the assumptions and the TD is causal.

It follows that we can modify the event graph so that the constraints between the triggers and the output events are of the form $\sigma^s(e_1, Tr_j)$ and $\sigma^L(e_1, Tr_j)$. The resulting TD thus has no output to output constraints in the same event block, but all these events must be scheduled as ASAP or as ALAP.

The above process is summarized in the following algorithm that computes the occurrence time schedule for output events in a causal TD, given a valid clock period C.

Algorithm

(1) Given a valid *C*, modify the constraints between output events e_1 and e_2 from w_{12} to $\lfloor w_{12} / C \rfloor C$.

(2) For each output block, compute the maximum separations $\sigma^s(e_1, Tr_j)$ and $\sigma^L(e_1, Tr_j)$ of each event with respect to the triggers as defined by the local constraints of the block. Replace the constraints between events and their triggers by the maximum separation relative to the triggers. Remove all constraints between output events in the same block.

(3) For each solution of equations (1), (2), and (3), verify that condition (7) holds. If (7) does hold for all cases, we can schedule (ASAP) the occurrence time as $\{\left\lceil S_{(\tilde{T}_1,\cdots,\tilde{T}_m)}(e_i)/C \right\rceil C \mid e_i \in EB\}$.

Example 4: Next we show a more complex example. The TD and its block structure is shown in figure 5. We apply the above algorithm to determine if C = 10 is a valid clock period and if yes then we compute the discrete-time relative ASAP schedule for the output events.

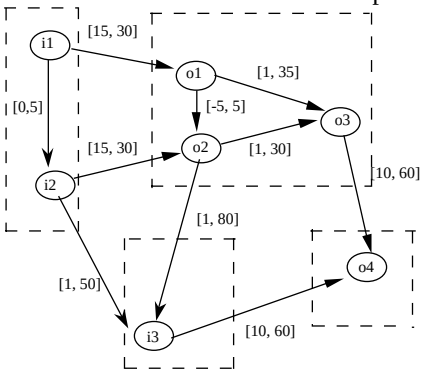

Figure 5: Block structure of example 4

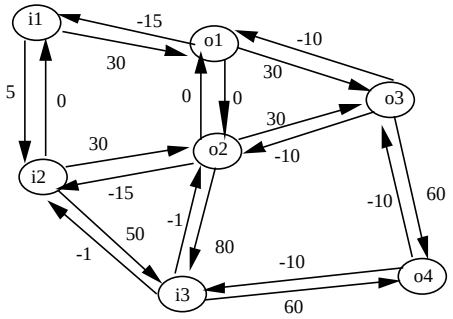

Figure 6: Modified graph (outputs separated by multiples of clock period)

According to Step (1), we modify the constraints between output events to the multiple of 10 and obtain the event graph of Figure 6. In the next step we calculate the shortest distance between the output events and the triggers and then modify the constraints to the output events. Figure 7 gives the new modified TD,. The block structure is not shown in the TD, however, it is the same as in Figure 5.

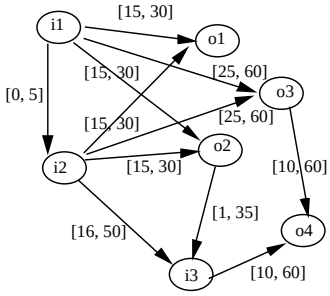

Figure 7: Modified TD after Step (2) of the algorithm

When C = 10, we have two possible sampling times for the input events i_1 and i_2, which are $\tilde{i}_1 = 0, \tilde{i}_2 = 0$ and $\tilde{i}_1 = 0, \tilde{i}_2 = 1$. Therefore, we have $\max_{0,0} T_1 = 0$, $\min_{0,0} T_1 = -10$, $\max_{0,0} T_2 = 0$, $\min_{0,0} T_2 = -10$, $\max_{0,10} T_1 = 0$, $\min_{0,10} T_1 = -5$, $\max_{0,10} T_2 = 5$, and $\min_{0,10} T_2 = 0$.

The schedule of the output events o_1, o_2, and o_3 are determined as follows. Note that the constraints for o_1 and o_2 are exactly the same, therefore we only need to calculate one of its schedules.

$S_{0,0}(o_1) = \max\{\max_{0,0} i_1 + \sigma^S(o_1, i_1), \max_{0,0} i_2 + \sigma^S(o_1, i_2)\} = \max\{0+15, 0+15\} = 15$,

$L_{0,0}(o_1) = \min\{\min_{0,0} T_1 + \sigma^L(o_1, i_1), \min_{0,0} T_2 + \sigma^L(o_1, i_2)\} = \min\{-10+30, -10+30\} = 20$

We can find the other values in a similar way: $S_{0,0}(o_3) = 25$, $L_{0,0}(o_3) = 50$, $S_{0,10}(o_1) = 20$, $L_{0,10}(o_1) = 25$, $S_{0,10}(o_3) = 30$, $L_{0,10}(o_3) = 55$.

Therefore the ASAP schedule is: $\tilde{T}_1 = 0$, $\tilde{T}_2 = 0 \text{ or } C$, $\tilde{t}_1 = \tilde{t}_2 = \tilde{T}_1 + 2C$ $\tilde{t}_3 = \tilde{T}_1 + 3C$. We next consider the block with only one event i_3. Given $\tilde{T}_1 = 0$, $\tilde{T}_2 = 0$ or C, $\tilde{t}_2 = \tilde{T}_1 + 2C$, the sampling time of i_3 is $\tilde{T}_3 = 3C$, $4C$, $5C$, or $6C$, Using ASAP scheduling, we have $\tilde{t}_4 = \tilde{T}_3 + C$.

4 Conclusions

In this paper, a new way for scheduling events under real-time constraints and for the synthesis of interface controllers based on timing diagram specifications was described. The method allows to determine a valid clock period and is suitable for synchronous system implementations. An algorithm for deriving ASAP and ALAP relative schedules

for the output actions was presented for causal specifications. Expected applications of this method range from high-level synthesis to the synthesis of sampled-input synchronous interface controllers.

References

[1] M. McFarland, A. Parker, R. Camposano, "The high-level synthesis of digital systems", Proc. of the IEEE, No. 2, February 1990.

[2] G. Borriello, R. H. Katz, "Synthesizing transducers from interface specifications", VLSI'87, North Holland, 403-418, 1988.

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

[4] G. Borriello, "Formalized Timing Diagrams", Proc. Euro-DAC'92, pp. 372-377, 1992

[5] S. Lenk, "Extended Timing Diagrams as a specification language", Proc. Euro-DAC'94, pp.28-33, 1994.

[6] R. Schlor, "A prover for VHDL-based hardware design", Proc. IFIP CHDL'95, 1995.

[7] K. McMillan and D. Dill, "Algorithms for interface timing verification", Proc. IEEE ICCD, 1992.

[8] E. Walkup, G. Borriello, "Interface Timing Verification with Application to Synthesis", Proc. DAC'94, 1994.

[9] T. Yen, A Ishii, A. Casavant, W. Wolf, "Efficient Algorithms for interface timing verification", Euro-DAC, 1994.

[10] G. De Micheli, *Synthesis and Optimization of Digital Circuit*, McGraw-Hill Inc. New York, 1994.

[11] W. Grass, C. Grobe, S. Lenk, and W. Tiedemann, "Timing diagrams as a specification language for interface circuits and their transformation into synchronous FSMs", BENEFIT-DMM 95. pp.280-35, Sept. 1995.

[12] W. Tiedemann, "An approach to multi-paradigm controller synthesis from timing diagram specifications", Euro-DAC'92, 1992.

[13] P. Gutberlet, W. Rosenstiel, "Interface Specification and Synthesis for VHDL Processes", Euro-DAC, 1993.

[14] K. Khordoc, E. Cerny, "Modeling cell processing hardware with action diagrams", Proc. ISCAS'94, 1994.

[15] K. Khordoc, E. Cerny, "Semantics and Verification of Timing Diagrams with Linear Timing Constraints," accepted to ACM Transactions on Design Automation of Electronic Systems (TODAES), May 1997, 25 p.

[16] P. Moeschler, H. Amann, F. Pellandini, "High-Level Modeling using Extended Timing Diagrams", Proc. Euro-VHDL '93, Hamburg, FRG, Sept. 1993, pp. 494-499.

[17] K. Khordoc, "Action Diagrams: A Methodology for the Specification and Verification of Real-Time Systems", Ph.D. thesis, Dept. of Electrical and Computer Engineering, McGill University, March 1996.

[18] W-D. Tiedmann, "Introducing Clock Cycles", Report COPRODES/UPA/1995/2, University of Passau, Nov.1995.

[19] B. Berkane, S. Gandrabur, E. Cerny, "Algebra of Communicating Timing Charts for Describing and Verifying Hardware Interfaces," Proc. IFIP Conf. on Computer Hardware Descr. Languages (CHDL'97), 1997.

[20] P. Girodias, E. Cerny, W.J. Older, "Solving Linear, Min and Max Constraint Systems Using CLP based on Relational Interval Arithmetic," J. on Theor. Comp. Science, 173(2), Feb.97.

[21] P. Girodias, E. Cerny, "Interface Timing Verification with Delay Correlation Using Constraint Logic Programming," ED&TC'97

[22] D.C. Ku, G. De Micheli, "Relative Scheduling under Timing Constraints: Algorithm for High-Level Synthesis of Digital Circuits," IEEE Trans. CAD ICS, 11(6), June 1992, pp. 696-718.

Acknowledgments: The work was partially supported by an Micronet Grant No. S4.MC1.