

# STG Timing Extensions and Simulation.

Michael V. Goncharov<sup>†</sup>  
Alexander B. Smirnov<sup>†</sup>

Ilya V. Klotchkov<sup>‡</sup>  
Nikolai A. Starodoubtsev<sup>†‡</sup>

<sup>†</sup> Institute for Analytical Instrumentation of RAS  
26, Rizhsky Prospect, Saint-Petersburg 198103 Russia

<sup>‡</sup> St.Petersburg State University  
7-9, Universitetskaya nab., Saint-Petersburg 190004 Russia

## Abstract

*In this paper the Signal Transition Graph (STG) model – a specification form that is widely used for asynchronous circuits’ behavior specification is extended to allow the delay information specifying. Timed STG models and their firing semantics are defined.*

*Then we present an automatic technique for timed STG behavior specifying in VHDL. The resulting VHDL specification may be used to simulate STG behavior with the specified delays. The requirements imposed on the STG to allow the joint environment – circuit behavior simulation are defined. Joint simulation of a circuit synthesized from STG with its environment is also presented.*

## 1 Introduction.

*Signal Transition Graph (STG) model was introduced in 1985 [1]. On one hand STG can be viewed as an interpreted Petri net (PN), where transitions are labeled with signal up and down transitions. On the other hand it is an alternative representation of a signal diagram. Arcs between STG transitions represent causality relations. There are also STG transitions (so-called *dummy* transitions) that are not labeled with signal transitions. They do not represent any signal changes and are used mostly to specify the events’ synchronization. Thus, classical STG model has no notion of time while causality relations are crucial. STG graphical examples may be found in Figures 1, 3 and 4.*

STG model is widely used to specify circuits’ behavior with concurrency and choice. Since delays cannot be specified in STG its usage is primarily situated in the asynchronous self-timed circuits’ design area,

where only transitions’ ordering is important. Autonomous STG usually defines the behavior of both circuit and environment, what makes it possible to design a circuit for the particular environment it is assumed to operate in. Extensions of this model that facilitate the behavior specifying by introducing various events’ types in addition to signal and dummy transitions are known [5].

Here we consider loading STG model with delay information. At the moment the primary purpose of specifying delays with STG is to take into account environment delays and enable the circuit optimization during synthesis. However the delay information specified with STG may be also used to constrain delays of the circuit to be synthesized and for a preliminary behavior timing analysis, what is shown in [4]. Considerations on the delay information usage during the logic synthesis are out of the scope of this paper. Here we concentrate on STG timing extensions and simulating of STG behavior in VHDL environment.

The paper is further organized as follows. In the section 2 the STG model timing extensions are considered and the timed STG models are defined. The section 3 presents the VHDL specification of STG behavior and the automatic technique for its generation.

## 2 STG delay extensions.

In this section we discuss the ways of incorporating time in STG and give a motivation to our choice. Since STG is a PN, we define STG delay models to conform to existing timed PN (TPN) models where possible. This allows applying of the techniques developed for TPNs to *timed STG (TSTG)* model. The reader not familiar with timed PN model may refer to [2] or [6] for introduction.

This section is further organized as follows. After

referring to the previous works on timed STG in subsection 2.1 we briefly consider the timing semantics and their use in STG (2.2). Later on the timed STG models are informally defined.

The following are formal definitions of a Petri net, classical STG and a few more definitions and notions that are supposed to facilitate the subject understanding.

- *Petri net* (PN) is a 4-tuple  $N = (P, T, F, m_0)$ , where  $P$  is a finite set of places,  $T$  is a finite set of transitions,  $F \subseteq (P \times T) \cup (T \times P)$  is the flow relation, and  $m_0$  is the initial marking;  $V = P \cup T$  stands for a set of all vertices of net  $N$ ;
- *STG* is a 4-tuple  $\mathcal{G} = (N, X, Z, \Delta)$  where  $N$  is a PN,  $X$  and  $Z$  are disjoint sets of input and output signals respectively and mapping  $\Delta : T \rightarrow (X \cup Z) \times \{+, -\}$  labels transitions of  $N$  with signal transitions; transitions unlabeled with signal transitions are called *dummy*;  $Y = X \cup Z$  denotes a set of all signals in STG;
- $t\bullet$  and  $\bullet t$  denote the sets of output and input places for transition  $t$ , correspondingly;
- $p\bullet$  and  $\bullet p$  denote the sets of output and input transitions for place  $p$ , correspondingly;
- for every arc  $f_{ij} \in F$  (arc from  $v_i$  to  $v_j$ ) denote  $\bullet f_{ij} = v_i$  and  $f_{ij}\bullet = v_j$  ( $v_i, v_j \in V$ );
- STG  $\mathcal{G}$  is called *autonomous* if for  $\forall v_i \in V : |\bullet v_i| > 0, V = P \cup T$  i.e. if there are no vertices without predecessors in the STG;
- A place in an STG is called an *implicit place* if: 1) it has one input and one output arc; 2) it is not shown in the STG drawings, but is represented by the corresponding arc between two transitions.
- An STG  $\mathcal{G} = (P, T, F, m_0)$  is called *free-choice* if  $\forall p_i \in P$  such that  $|p_i\bullet| > 1$  for  $\forall t_j \in p_i\bullet : |\bullet t_j| = 1$ , i.e. for every place with multiple outputs all directly succeeding transitions have a single input.
- place (arc) *capacity* is the maximal reachable number of tokens the place (arc) can be marked with;
- STG is called *n-bounded* if the number of tokens at any place at any time is less then or equal to  $n$ ;

- STG is called *consistent* if: 1) every rising transition of a signal is followed (not necessarily immediately) by a falling transition of this signal and vice-versa; 2) signal transition cannot be concurrent to another transition of the same signal;

## 2.1 Previous work on timed STG.

Timed STG models were previously investigated in [3], [4] and in [10]. In the first two works STG timing semantics were discussed along with some TSTG analysis algorithms. In [4] deterministic acyclic classical STG is extended with interval delays assigned to arcs (implicit places). Furthermore the set of flow relations (arcs) is subdivided to *specification* and *delay* relations that specify the circuit timing constraints and assumptions correspondingly. The issues on synthesis of circuits with timing constraints from TSTG specification can be also found there. In [10] TSTG models with constant delays are defined and their usage is introduced. The correspondence of casual STG paths to physical wires is shown and VHDL simulation issues are presented.

## 2.2 Delay semantics.

There are many ways time can be specified with STG. They are investigated for timed PNs.

1. constant delay - used in timed Petri nets;
2. interval delay - used in *interval* TPNs;
3. delay probability density - used in *stochastic* TPNs;

The first way provides specifications, which are clear and easy analyzable though impractical for circuit behavior specification. Stochastic TPNs are general, but inadequate for the circuits design problems. Thus, we fix on the interval delay specification model.

In the TSTG models that we use the firing semantics are defined as follows.

If some delay interval ( $d_{min}^p, d_{max}^p$ ) is assigned to an STG arc (place) any token that entered the arc (place) is assumed to become available only after its delay  $d^p$  such that  $d_{min}^p < d^p < d_{max}^p$  elapses since the absolute time the token entered the arc (place). On the other hand, if a delay interval ( $d_{min}^t, d_{max}^t$ ) is assigned to an up (down) transition of a signal  $a$  one token is removed from each arc (place) entering the transition immediately, but the signal value change and incrementing the tokens' counters for the arcs leaving the transition occurs as soon as the transition delay  $d^t$  such that  $d_{min}^t < d^t < d_{max}^t$  elapses.

As long as time is specified for an STG element a delay value may be used to calculate absolute firing

times and, thus, establish the signal transitions' order. This is inapplicable since most of STG analysis and synthesis methods are based on the causality expressed with arcs. For the cases where only timing information is available such a specification must be analyzed in order to derive and specify the dependencies between signal transitions for the behavior. The resulting STG must be consistent and live (the latter ensures that every event fires at least once).

Thus, in the TSTG definitions we assume that 1) all arcs are causal; 2) delay (wherever the term is used) may vary in the range of the time intervals specified for the STG element.

### 2.3 Timed STG models.

*Place timed* and *transition timed STG* models are defined in accordance with place timed and transition timed Petri net models. The *arc timed STG* and combined models are introduced to address the ease of behavior specification. For more details on these models please refer to [10].

- *Place timed STG model (PTSTG)*. In PTSTG delays are assigned to places while transitions are said to fire and arcs – to transfer tokens immediately.
- *Transition timed STG model (TTSTG)*. In this model delay is assigned to up (down) transitions of a signal. Dummy transitions as well as other STG elements are assigned zero delays. TTSTG is reducible to the PTSTG.
- *Arc timed STG model (ATSTG)*. For ATSTG model delays are assigned to arcs, while places and transitions are assigned zero delays.

ATSTG model is the most general among those defined above. However, it is worth to note that firing semantics defined for timed Petri nets where arcs are labeled with delay intervals in works on timed PNs may differ from the firing semantics we use. Thus, we define the firing semantics we use for ATSTG as follows.

- For every particular marking such that places  $p_0, \dots, p_m$  are marked correspondingly at absolute times  $C_{p_0}^R, \dots, C_{p_m}^R$  transition  $t_i$  becomes enabled at the time  $C_{t_i}^E = \max(C_{p_0}^R + D(f_{qi}), \dots, C_{p_m}^R + D(f_{ri}))$  where  $\{p_0, \dots, p_m\} = \bullet t_i$  and  $\forall f_{xi} \bullet = t_i$ .
- The delays' difference between the output arcs of a place does not affect the choice specified by the place. Therefore, a *nondeterministic choice*

*behavior is preserved regardless of the delays assigned to the arcs leaving a place.*

Despite the convenience of timed behavior specifying provided by this model the lack (as far as we know) of techniques worked out for the semantics we use for ATSTG may limit its (the model's) usage.

Other models: *transition-place timed STG (TPTSTG)* and *transition-arc timed STG (TATSTG)* are obtained by the primary models combining. These are redundant - can be reduced to PTSTG and ATSTG correspondingly, but provide the clearest way of specifying known delays.

## 3 STG VHDL specification.

The technique presented in this section is developed to bring VHDL based tools in the asynchronous circuits' design environment.

This section briefly describes the idea behind the TSTG behavior simulation in VHDL environment, while the simulation of unsafe TSTGs and the joint simulation of the environment behavior with the synthesized circuit are presented in more details.

### 3.1 Previous work.

Previous applications of VHDL environment to PN's and STG's behavior simulation are known. A technique for PN composition and simulation was proposed in [7]. Peter Vanbekbergen et al [8] used VHDL in their design environment for the STG behavior simulation and the synthesized circuits' validation. In this work the use of VHDL assertions was shown to check certain STG properties. VHDL environment has been also used for interactive synthesis of asynchronous circuits in [9]. In the latter work the signal graph semantically different from STG has been used.

### 3.2 Specifying STG behavior in VHDL.

For the VHDL specification generated from an STG the following features must be provided.

- 1) The specification modularity, i.e. circuit specification must be independent from the environment specification; This is achieved by generating distinct entities for circuit and environment while third entity is created to specify their interconnection.
- 2) The ability to simulate the signals' timed behavior; It is provided by the VHDL signal notation.
- 3) The ability to specify multiple up (down) transitions of a signal; It is provided by associating an integer (the last fired transition index) to VHDL signal as follows.

```
type Indexed is
    record value : std_logic;
```

```

    index : integer;
end record;

```

4) The ability to simulate the time behavior of places and/or arcs with the capacity greater than one; This is considered in section 3.3.

5) The ability to simulate a nondeterministic choice; To simulate a choice the *preselection policy* (the function that determines the event(s) to fire for every STG reachable marking with choice) is necessary. We use either the function that iterates through the allowed events' sets (so-called *transitions patterns*) or a pseudo-random numbers generator each time the choice is encountered. The choice simulation also imposes the problem of guaranteeing that every conditional path is activated. It can be handled either with the use of a global preselection policy or by developing special local policies.

Environment and circuit entities are generated for every STG to satisfy the modularity requirement. Such an entity comprises a single process with three groups of assignment rules and a `wait on` statement with a complete list of entity signals in the process end. The latter makes assignment rules to be checked every time a transition occurs. Assignment rules are organized as follows (the order is important).

1. Increment available tokens counters for places (arcs) for which the associated delay has elapsed since a token had entered the place (arc);
2. Start timers for places (arcs) that are output to STG fired transitions.
3. Schedule STG transitions that has become enabled (all predecessor places (arcs) are marked i.e. their number of available tokens is greater than zero), decrement the tokens counters for every input arc (place).

Assignment rules examples are given in section 3.3 for the TPTSTG model.

Since the signals output for the entity are not observed inside the entity the signals that duplicate output are defined in every entity. See [10] for more details on STG VHDL specification.

The simulation output is a well-known signals' timing diagram. All timed STG models presented in section 2.3 can be specified.

### 3.3 Simulating unsafe STGs with interval delays.

Consistency and boundedness are necessary conditions for the behavior of STG to be implementable. Many techniques are known for asynchronous circuits'

synthesis from such a specification. Thus, the simulation techniques must handle that class of STGs. Generally, since a delay is assigned to a place (arc) a VHDL signal is necessary to simulate it (transitions scheduling is only supported for signals in VHDL). On the other hand a number of timers greater than one may be active for a place to simulate multiple markers entered the place (arc). That involves the number of signals equal to the place (arc) capacity. Every signal implementing a place (arc) timer is associated a Boolean variable to indicate if the timer is currently active. The signals relative to a given place may be organized in a vector. If interval delays are used a variable of type `time` may be necessary for every such a vector to be used for calculating of the next delay value. With such a data structure declared for every place as soon as a marker enters a place a spare timer is activated (a positive transition is scheduled for the first unused signal in the corresponding vector) and the delay value to be used the next time is calculated. Thus, any up transition in the vector means that the markers counter for the place must be incremented and the corresponding signal – reset.

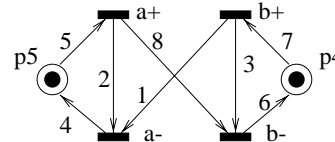


Figure 1: Simple 2-bounded STG example. (Numbers on arcs represent names.)

On the other hand, under the constant delay assumption it is guaranteed that a place (arc) behaves as a FIFO i.e. markers cannot overtake each other. That makes it possible to implement a place with a single signal as follows. Markers counter and a Boolean to keep the track of the last signal assignment are also necessary. For the signal assigned in this case it is crucial to use `transport` VHDL delays to preserve events previously scheduled for the signal.

The latter technique is illustrated with a simple example. The code examples below are extracted from the VHDL code generated for the STG shown in Figure 1.

Examples correspond to the assignment rules generated for the arc labeled with 8 in the figure. The rules order is changed here comparing to the real code.

- 1) Start counters for all arcs leaving the up transition of signal `a` if it has fired.

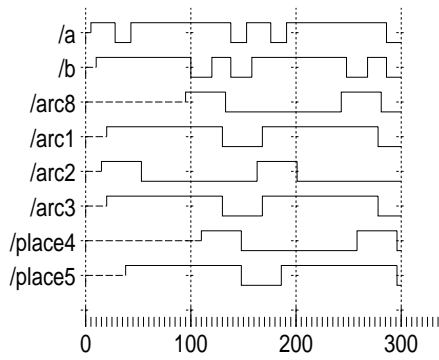


Figure 2: Timing diagram for the example of 2-bounded STG.

```

if ((a'event) and (a = '1')) then
  if ( arc8drive ) then
    arc8 <= transport '0' after 90 ns;
  else
    arc8 <= transport '1' after 90 ns;
  end if;
  if (arc2drive) then
    arc2 <= transport '0' after 10 ns;
  else
    arc2 <= transport '1' after 10 ns;
  end if;
  arc8drive := not arc8drive;
  arc2drive := not arc2drive;
end if;

2) Increment the markers counter (arc8ready) for
implicit place arc8 as soon as its delay has elapsed.

if ( arc8'event ) then
  arc8ready := arc8ready + 1;
end if;

3) Fire signal b transition in 5 nanoseconds after it
has become enabled (i.e. all input places are marked).

if((arc8ready > 0) and (arc3ready > 0)) then
  arc8ready := arc8ready - 1;
  arc3ready := arc3ready - 1;
  b <= '0' after 5 ns;
end if;

```

The timing diagram illustrating the behavior of this STG is shown in Figure 2. The successive transitions of the signal `arc8` one after another show, where two timers are active simultaneously for the implicit place 8. (`arc8` is associated the delay of 90 nanoseconds as it can be seen from the VHDL examples).

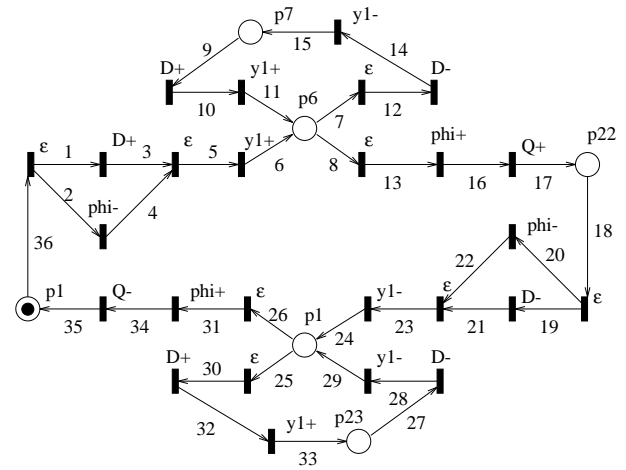


Figure 3: Original STG specifying the flip-flop behavior.

### 3.4 Joint behavior-circuit simulation.

Despite the fact that most of the design methods and CAD tools are proved to provide correct solutions under some particular assumptions, simulation is still widely used to check the design behavior. This convinces a designer in the correctness of the synthesized circuit as well as lets a designer check if the assumptions important for the design match those of the synthesis system.

The joint simulation possibility is provided by the behavior VHDL specification modularity. A few more requirements are, however, imposed on the STG behavior specification to make the circuit-environment simulation possible.

These are outlined as follows.

1) For the behavior simulation initial values of STG signals are not important. Only initial marking of the underlying Petri net is necessary for STG behavior simulation. However, for the circuit signals initial state is crucial to simulate the circuit's behavior.

2) The type `indexed` may not be used for circuit output signals. Indeed, assume that the circuit's behavior is specified with a set of equations that implement circuit signals.

3) Behavior and circuit interfaces must match.

These restrictions can be easily met.

The first restriction makes the signals' initial state specification necessary. It is calculated from the STG using its initial marking.

To satisfy the second requirement it is necessary to specify explicitly the environment state transitions'

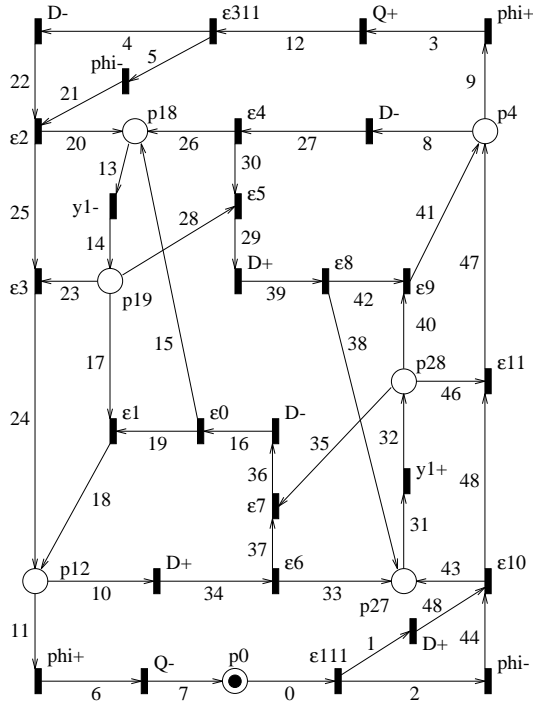


Figure 4: STG specifying the flip-flop behavior with multiple transitions of the same signal merged. (Numbers on arcs represent names, p0 corresponds to place0,  $\epsilon_0$  corresponds to dummy0 etc. in VHDL code.)

conditions with no internal circuit signals observed. We consider two general ways of implementing this ability below.

The first approach is to analyze the STG behavior prior to the VHDL code generation and to extend the environment VHDL entity with the historical data sufficient for explicit specification of the environment events' conditions. Such data may include the state of environment and/or circuit output signals, events' counters and succession information.

The second approach is based on merging together multiple up (down) transitions of the same signal prior to the generation of VHDL code. Such a merge would make the environment to keep independently track of the circuit events. The merging itself may be completed in a number of ways known in the Petri nets theory. We use the following simple algorithm that preserves the specified TSTG behavior for instance.

Let  $Drn(t_i) \in \{+, -\}$  for  $\forall t_i : (\Delta t_i \in Y)$  be a direction function for a signal transition  $t_i$ ;

**function** *MergeTransitions* ( $y \in Y, d \in \{+, -\}$ )

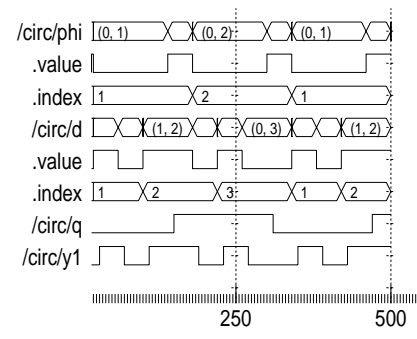


Figure 5: Timing diagram of the DFF behavior.

```

insert places p1, p2
insert transition t :  $\Delta(t) = y, Drn(t) = d$ 
insert arc f :  $\bullet f = p1, f \bullet = t$ 
insert arc f :  $\bullet f = p2, f \bullet = t$ 
for  $\forall t_i \in T : \Delta(t_i) = y, Drn(t_i) = d$ 
  insert dummy transitions  $E1_i, E2_i$  in the STG
  for  $\forall p_j \in P : p_j \in \bullet t_i$ 
    insert arc  $f^{orgn} : f^{orgn} \bullet = E1_i, f^o = p_j$ 
    delete arc  $f^{new} \in F : \bullet f^{new} = t_i, f^{new} \bullet = p_j$ 
     $D(f^{new}) := D(f^{orgn}) /*delay assignment*/$ 
  end for
  for  $\forall p_j \in P : p_j \in t_i \bullet$ 
    insert arc  $f^{orgn} : \bullet f^{orgn} = E2_i, f^{orgn} \bullet = p_j$ 
    delete arc  $f^{new} \in F : \bullet f^{new} = t_i, f^{new} \bullet = p_j$ 
     $D(f^{new}) := D(f^{orgn}) /*delay assignment*/$ 
  end for
  delete transition  $t_i$ 
  insert arc f :  $\bullet f = E1_i, f \bullet = p1$ 
  insert arc f :  $\bullet f = p2, f \bullet = E2_i$ 
end for

```

The design of D-flip-flop is illustrated in Figures 3 (the original STG), and 4 (STG obtained after the transitions of y1 are merged). Finally, Figures 5 and 6 are the circuit and environment signal diagrams that show the simulation results relative to the circuit output signals' transitions (it can be seen from the latter diagram how environment keeps track of the y1 transitions prehistory).

In this example the TPTSTG model is used. I.e. delays are associated with places (those called like *arc1*, *arc2* are implicit) and transitions. On signal diagrams transitions of signals implementing STG places correspond to the instants when the counter of ready (available) tokens for the corresponding place is incremented.

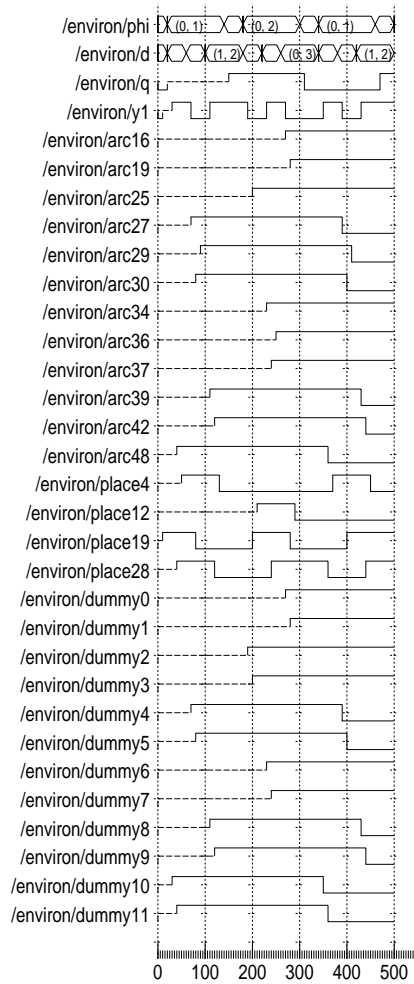


Figure 6: Timing diagram of the DFF environment behavior (part).

### 3.5 The use of VHDL simulation in the asynchronous circuits' design framework.

The behavior simulation in the design framework is primarily intended to convince the designer in the correctness of the specification, the circuit designed and the layout generated. The STG behavior simulation is used the same way. The simulation of the STG behavior makes it possible for the designers unfamiliar with STGs to check the specification. Joint behavior-circuit simulation in turn allows checking the designed circuit(s) in its interaction with environment(s).

## 4 Conclusion.

In this paper extensions of the STG model that make it possible to specify time delays on STG on one

hand and preserve the causality crucial for existing asynchronous circuits' design techniques using STG on the other hand are considered. Timed STG models and their semantics are defined.

A technique is presented for automatic translation of STG specification to VHDL. Simulation of timed STG behavior is presented and illustrated with examples.

All algorithms and techniques presented in the paper are implemented in the TaxoSynthesis CAD tool.

## References

- [1] T.A.Chu, C.K.C.Leung and T.S.Wanuga A design methodology for concurrent VLSI Systems. In *Proceedings of ICCD* pages 407-410, 1985.
- [2] T. Murata "Petri Nets: Properties, Analysis and Applications." In *Proceedings of the IEEE. Vol.77, No 4*, p.541-580, 1989
- [3] P. Vanbekbergen, G. Goossens, H. De Man "Specification and Analysis of Timing Constraints in Signal Transition Graphs." IMEC Laboratory, Kapeldreef 75, B-3001 Leuven, BELGIUM
- [4] P. Vanbekbergen, G. Goossens, B. Lin "Relational and Timing Semantics for a Timed Signal Transition Graph Model." IMEC Laboratory, Kapeldreef 75, B-3001 Leuven, BELGIUM
- [5] P. Vanbekbergen, C. Ykman-Couvereur, B. Lin, and H. De Man. Generalizing signal transition graphs for modelling mixed asynchronous/synchronous and arbitration behavior. In *Proceedings of International Workshop of Logic Synthesis*, May 1993
- [6] F. Bowden "Modeling time in Petri net", In proceedings of "The second workshop on stochastic models", Gold Coast, July 1996
- [7] S. Mohanty. "An Integrated Design Environment for Rapid System Prototyping, Performance Modeling and Analysis using VHDL". (Thesis.) September, 1994
- [8] P. Vanbekbergen, A. Wang, K. Keutzer. "A Design and Validation System for Asynchronous Circuits", In *Proceedings of DAC*, 1995.
- [9] N.A.Starodoubtsev, A.V.Yakovlev and S.Yu.Petrov. Use of VHDL - based environment for interactive synthesis of asynchronous circuits, *VHDL User Forum Europe: Proceedings SIG-VHDL Spring'96 Working Conference*. Dresden, Germany, Shaker Verlag, Aachen, May 1996, pp. 21-33.
- [10] M.V.Goncharov, I.V.Klotchkov, A.B.Smirnov, N.A.Starodoubtsev. "Timing extensions of STG model and a method to simulate timed STG behavior in VHDL environment." In *Proceedings of International Conference on Application of Concurrency to System Design.*, Aizu, March 1998