Advanced Optimistic Approaches in Logic Simulation

St. Schmerler, Y. Tanurhan, K.D. Müller-Glaser

F Z I - Research Center for Information Technology Karlsruhe
Dept. of Electronic Systems and Microsystems (ESM)
Haid-und-Neu-Str. 10-14, D-76131 Karlsruhe, Germany
http://www.fzi.de/esm/esm.html, email: esm@fzi.de

Abstract

This paper presents the optimistic synchronization mechanism Predictive Time Warp (PTW) based on the implementation Time Warp of the Virtual Time paradigma for use in the simulation of electronic systems and high level system simulation. In comparison to most existing approaches extending and improving classical Time Warp, the aim of this development was to reduce the rollback frequency of optimistic logical processes without imposing waiting periods. Part of PTW is the introduction of forecast events predicting a certain period in the future and thus reduce the rollback probability. On the example of a distributed logic simulation the benefit of the PTW synchronization approach is shown.

1 Introduction

The Time Warp (TW) mechanism as defined by Jefferson and Sowizral [11], [12] uses the sending of messages for synchronization. To keep the local causality constraint *lcc* between all participating logical processes fulfilled, a synchronization mechanism is used employing rollbacks in the local simulation time. The receipt of external events with timestamps in the local past (straggler events) cause a logical process to roll back to the most recently saved state consistent with the timestamp of the straggler event and then to restart the simulation from this point with the aim to correct the *lcc* violation with this procedure. However, rollbacks require a periodical state saving with respect to internal and external events of the logical process. In addition, an input queue IQ for received messages and output queue OQ for sent messages have to be administrated. The registration of the communication history of a logical process is performed in chronological order. Since the arrival of event messages in increasing time stamp order cannot be guaranteed, TW introduces two different kinds of messages for the implementation of the synchronization protocol:

- The usual external events *ee* exchanged between logical processes are marked with a *positive sign* (m_+=<*ee*@*t*, +>). The time *t* is a copy the local virtual time of the sender at sending time.
- Besides positive messages, *antimessages* with negative sign m_-=<*ee*@*t*, -> have been introduced. Antimessages have the ability to annihilate formerly and prematurely sent positive messages for the same event *ee* after being received by the same logical process.

Annihilation has to be performed after the receipt of a straggler message. It has to be applied to all events in the local event queue with timestamps greater than the straggler´s timestamp because all of those events have (possibly) been based on an erroneous computation. Of course, new external events may be the result of this erroneous system state which have already been sent to other logical processes. In order to be able to cancel all of those messages, each logical process keeps records of all sent external events (positive messages) up to as certain time stamp and then resends them for a second time with a negative sign to achieve annihilation. An important issue on the area of memory consumption in Time Warp is to apply fossil collection as a technique to reclaim memory consumed by history recording that will definitely not be used any more due to an assured lower bound on the timestamp of any possible future rollback. This lower bound is referred to as the Global Virtual Time GVT of the whole system. External events from a logical process LP_k are received from the communication system by communication interfaces. Any logical process LP_k is connected to the regarded LP_1 by the two directed data channels $ch_{1,k}$ and $ch_{k,1}$. In Time Warp, messages are not required to arrive in the same order as they were sent. It is not necessary to separate the incoming data channels, furthermore they all lead to one input queue IQ or are coming from one output queue OQ, where the communication history of any logical process is stored. The state history is kept in the state stack SS. All those data structures together build the Communication Interface CI of the logical process, the event-driven Simulation Engine is the central location for the execution of

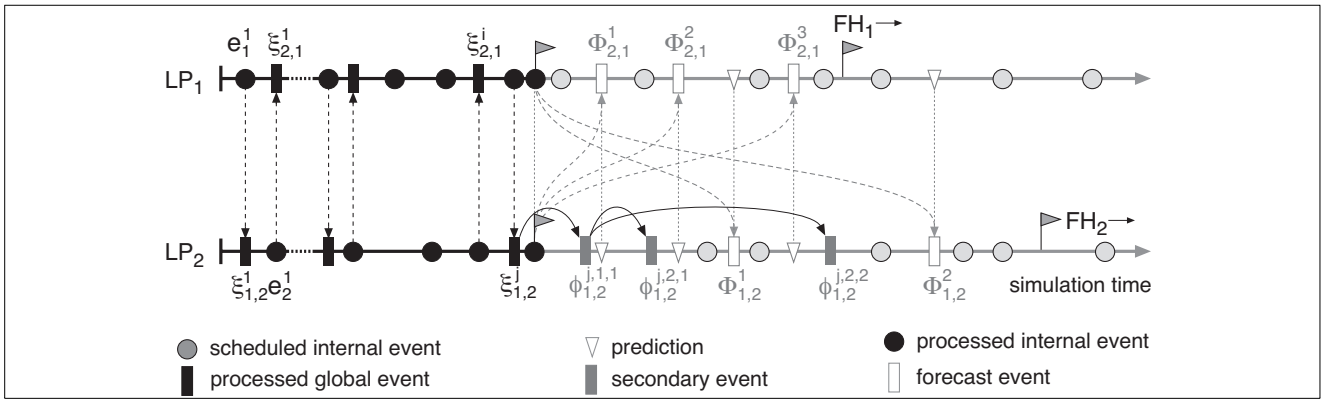

Figure 1: Forecasting Process

the simulation process. Recent developments in the optimization of the classical TW approach have been concerned with the reduction of the rollback frequency by limiting the optimism of TW. These efforts address the fact, that optimistic synchronization protocols like TW tend towards *overoptimism*, i.e. optimism lacking rational justification. Optimistic logical processes process their local event queue regardless to future external events of other processes (potential stragglers). This is on the one hand a main advantage compared to conservative synchronization approaches using waiting cycles as synchronization means and having to struggle with deadlock avoidance [5], [6]. On the other side, optimistic LPs act totally independent and thus may exploit inherent model parallelism. However, future external events will very probably cause a rollback in the receiving process when the LVT of sender and receiver differ to a certain amount ($LVT_{send} < LVT_{rec}$). The rollback probability is a rising function of this time increment and - if both receiver and sender make differently good progress in the simulation time - also a rising function of time.

In the past years this problem has been addressed by limiting the optimism of TW processes for instance in the approach of Time Windows [15], the Breathing Time Bucket Protocol BTB [8], Breathing Time Warp BTW [16], probabilistic, distributed DES protocols [9] or analytical methods [1].

2 Predictive Time Warp

2.1 Motivation

Common to all described approaches is the mean to block running logical processes (or delay certain classes of events which results in the same) in order to reduce the risk of a future causality violation and subsequent rollback chains. Predictive Time Warp (PTW), which is presented in this paper, addresses the same problem in another way. PTW uses predictions for the estimation of the timestamps of external events. Those predictions, however, are not performed to decide whether or not to process an event currently in execution [9]. Instead, the predicted behavior of a communication channel is used to create forecast events, i.e. estimations about arriving external events in the future. In its optimistic simulation, the LP will consider such *forecast events* in the same way as if they were real events. Forecast events are also maintained in the EQ, IQ and OQ of the LP's communication interface. The main idea behind this approach can be described as follows: When a Time Warp process at a certain instant *t=LVT* progresses in time, it executes its events currently available in the EQ and cannot consider external events arriving in the future. Consequently, a TW process *assumes that in the future (later than t) no external event of any process will ever arrive* (in any other case the computed state would be wrong). This assumption, of course, is not correct in most cases and leads to causality violations and rollbacks. However, by integrating forecast events in the EQ and considering them in the progressing simulation, the process does not assume that from now on no external events will be sent by the foreign LPs but it makes the assumption *that the foreign processes will in the future behave as they did in the past*, which is a much more reasonable hypothesis.

2.2 Extensions to Time Warp

As already described, the classical TW uses two kinds of messages: those with a positive sign m_+=<*ee@t*, +> and antimessages with a negative sign m_-=<*ee@t*, ->. PTW extends this protocol by one further event type: *forecast events* Φ=(*ee@t*). Forecast events $\Phi_{i,k}$ are treated as if they were sent by a logical process LP_i to LP_k if a communication channel $ch_{i,k}$ exists between LP_i and LP_k. However, forecast events are generated by the (virtual) receiver itself for each input communication channel $ch_{i,k}$. Fig. 1 shows the forecasting process for two LPs 1 and 2. $\Phi^1_{1,2}$ in

Fig. 1 is the first (upper index) forecast event for the future behavior of LP_1 - it is treated like an external event sent from LP_1 to LP_2, however, is the result of a prediction of LP_2 for communication channel $ch_{1,2}$. Forecast events always are positive, external (global) events for the predicting process and in the local future of it. Once executed, they remain in the event queue in the same way as normal events do.

Based on several constraints which will be discussed later in this paper, a certain number of predictions will always be active in the local future of any LP_i at any instant LVT_i and for each channel $ch_{i,k}$. Together with the logical processes on the x-axis, the *forecast horizons* $FH_{i,k}$ of LP_i, i=1..N of all processes form a *prediction space*, where $FH_{i,k}$ are the respective upper time limits for predictions in the local future of each virtual sender LP_i. for all input communication channels $ch_{i,k}$.The calculation of the prediction space will be dealt with later on in this document. Besides the introduction of new message types, PTW provides for an extension of the TW protocol. Forecast events are treated as external events in classical Time Warp, i.e. they have influence on the state variables S_k of an LP_k. and the execution of forecast events $\Phi^n_{i,k}$ may also result in the generation of secondary events $\phi^{n,d,\nu}_{i,k}$ of degree d (second upper index) on their target processes. A secondary event of first degree (d=1) is defined as an event directly generated by the simulation engine when executing the primary event, i.e. secondary events of degree 1 are causal to their primary events and a secondary event of degree d to the secondary event with degree d-1 (see Fig. 1). All secondary events with degree d of a forecast event $\Phi^n_{i,k}$ are defined as its secondary event set $\Sigma^{n,d}_{i,k} = \bigcup_\nu \phi^{n,d,\nu}_{i,k}$. This set is important for handling prediction errors, which are dealt with later in this paper.

An extension to TW is the use of an integrated *prediction engine* PE_i of any LP_i to generate forecast events $\Phi_{i,k}$ for each input communication channel $ch_{i,k}$. Based on the current confidence of a prediction for the maintained channels, the PE calculates the respective current forecast horizon $FH_{i,k}(ch_{i,k})$ for each channel and integrates the resulting forecast events in the event queue EQ_k.

2.3 Prediction Machine

The Prediction Machine (PM) is the functional unit in the PTW system responsible for the determination of forecast horizon and performing the predictions. Together with the CI, forecast events are generated. The rest of this chapter gives an introduction to the prediction algorithm used in PTW and other related algorithms.

Common to all prediction algorithms is the use of a current value x_t and previous values x_{t-1}, x_{t-2}, ... (memory depth) of a variable $x = (x_0, x_1, ..., x_t)$ to predict a certain number of variable values up to a *prediction horizon* or *lead time* l in the future. Thus, predictions are performed up to $\hat{x}_{t+l} = \hat{x}_t(l)$. Constraints to the prediction process are accuracy, computation time and memory consumption.

ARMA Predictors: Predictors based on stochastic processes like ARMA (autoregressive-moving-average), although being subject to recent research [9], have not been considered in this work for two reasons. First, an ARMA predictor requires a high amount of computation time and thus had to be executed on a processor of its own in real implementations [9]. The benefit of the predictor, i.e. the proven decrease of the rollback frequency in a TW system, and thus the gain of performance, has to be seen in the light that the resources allocated by the predictor could also have been used by a simulation process increasing the performance in this way. The second reason for not choosing autoregressive approaches was the nature of most signals in technical systems (only those are regarded in this approach) which is *not* stochastic but often regular and pattern-oriented. The chosen prediction mechanism consequently is pattern-oriented and requires significantly less computation resources than ARMA predictors.

Pattern-Oriented Predictors: *Pattern-Oriented Predictors* (POP) recognize already "learned" value sequences and try to predict the future behavior based on a pattern dictionary [3], [4].

Lempel-Ziv (LZ78)-based algorithms are a sub group within POP based on Lempel-Ziv algorithms [17], [18], [2] which had been developed mainly for the application of data compression. LZ78 creates a dictionary with different phrases. A read sequence will be included in the dictionary as a new phrase when it is the *smallest*, *unknown* sequence, which is not yet part of the dictionary. Thus, a tree of patterns is built and based on the traversal frequency of each leaf, the predictions are performed. LZ78 algorithms possess a constant computation time but the memory consumption is a linear function of the number of already learned patterns and thus rises logarithmically with the length of the input sequence.

Prediction by Partial Matching (PPM) algorithms register the appearance frequency for each token of the input token stream with respect to the fact, how often this token had followed certain patterns of different lengths in the past (the *contexts*). Probability density functions are used to predict a new token for a given context [7].

Greedy Parsing Predictors (GPP), based on LZ78, use a heuristic algorithm computing sub-optimal solutions in a short time. The predictions are sub-optimal because it cannot be guaranteed that in every case the (based on his-

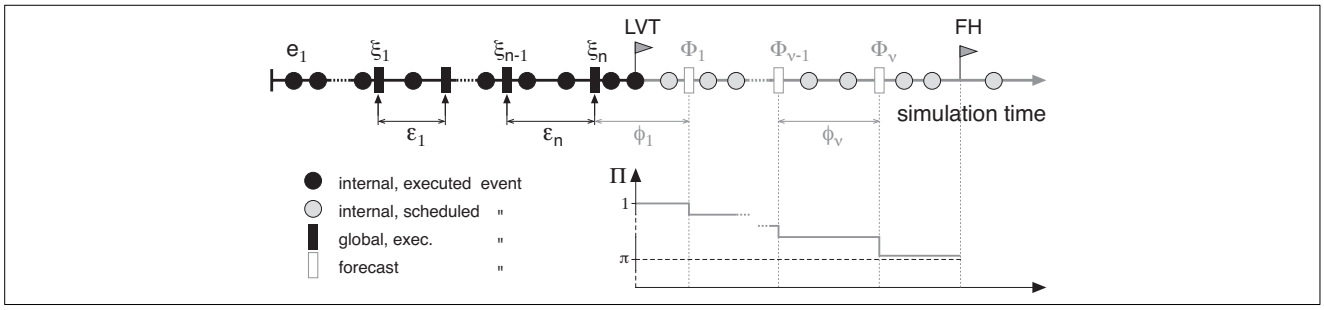

Figure 2: Prediction Confidence Function Π

tory) most probable token will be chosen, i.e. the choice will fulfill the Maximum-Likelihood-criterion [10].

Compared to the already presented predictors, GPP achieves a high prediction reliability after a longer learning phase. In GPP, the input token stream is partitioned into subsequent token patterns which are enrolled in a dictionary. In this process, beginning with the current (latest) token, the longest pattern being not yet part of the dictionary pattern is determined and registered as new entry in the dictionary. The dictionary usually is implemented as a search tree encoding token patterns as the sequences of "visited" nodes in the tree traversal.

The *Partial Matching Predictor* (PMP) algorithm is very similar to GPP, however, works with a fixed context length and computes optimal predictions with respect to the Maximum-Likelihood-criterion.

In PTW, a PMP-based approach has been chosen and extended to support the prediction process by introducing *Backtracking Search Trees*, special transformations of the computed dictionary tree. The PTW predictor also uses an exponential smoothing effect to give recent changes in the pattern flow more influence on the prediction process.

Experimental results on the area of distributed logic simulation using ISCAS circuits with up to 100k gates show an excellent prediction confidence (success rate) of at least 90% in the running PTW-based simulation.

2.4 Forecast Horizons

Forecast horizons $FH_{i,k}$ represent an upper limit for the timestamps of forecast events in a certain input communication channel $ch_{i,k}$ of a logical process LP_i. Per definition, a forecast horizon is the highest timestamp of all currently pending (index n) forecast events for an input channel $ch_{i,k}$ of an LP_i:

$$FH_{i,k} = FH(ch_{i,k}) = \max\left(\mathrm{ts}\left(\Phi^n_{i,k}\right)\right),\ n = 1 \ldots \nu_{i,k}$$

The forecast number $\nu_{i,k}$ of a channel is defined as the number of allowed predictions (pending forecast events) at an instant and therefore, also a function of time. It strongly depends on the accuracy of the past prediction processes. Clearly, the accuracy of a prediction will decrease with a rising number of forecasts or an increasing distance between *LVT* and the timestamp of the predicted event. This prediction confidence for a forecast event will be called Π in the following. As shown in Fig. 2, Π can be described as a step function over the simulation time:

$$\Pi_{i,k}(t_{sim}) = \pi_n,\ \mathrm{ts}\left(\Phi^n_{i,k}\right) \le t_{sim} < \mathrm{ts}\left(\Phi^{n+1}_{i,k}\right),\ 1 \le n < \nu$$

It can be derived by the introduction of local relative error ψ_n for each forecast event $\Phi^n_{i,k}$. This error is defined as

$$\psi_n = \frac{\left|\mathrm{ts}(\Phi_n) - \mathrm{ts}(\mathrm{ee}(\Phi_n))\right|}{\mathrm{ts}(\Phi_n) - \mathrm{ts}(\Phi_{n-1})} \quad \forall\, ch_{i,k}$$

where $\mathrm{ee}(\Phi^n_{i,k})$ denotes the (matched) partner event to $\Phi^n_{i,k}$, which is the external event with the nearest timestamp to the regarded forecast event arrived via $ch_{i,k}$. The local error is a function of time according to

$$\psi(t) = \psi_n \text{ for } \mathrm{ts}(\Phi_{n-1}) \le t < \mathrm{ts}(\Phi_n) \quad \forall\, ch_{i,k}$$

The prediction confidence function can be derived by this local error function with the relation $\Pi(t) = 1 - \psi(t)$.

The forecast number ν of a communication channel $ch_{i,k}$ can now be calculated by the local error function. The following equation assumes $LVT = \mathrm{ts}(\Phi^0_{i,k})$:

$$\nu_n = \left\lceil \nu_0 \int_{\varepsilon}^{\mathrm{ts}(\Phi_n)} \exp\frac{t - \mathrm{ts}(\Phi_n)}{\tau}\left\{1 - \psi(t)\right\}\mathrm{d}t \right\rceil \quad \forall\, ch_{i,k}$$

This function uses exponential smoothing for considering recent prediction results to a higher degree than those in the further past. The time ε denotes the depth of the memory for the forecasting history.

2.5 Forecast Event Shifts

A *prediction fault* occurs when the timestamp of a forecast event $\Phi^n_{i,k}$ gets no correspondence $\mathrm{ee}(\Phi^n_{i,k})$ in the set of the via input communication channel $\mathrm{ch}_{i,k}$ arriving external events. Such a pair $[\Phi, \mathrm{ee}(\Phi)]$ has been found when the timestamps of both events are equal. The frequency of prediction faults highly depends on the used algorithm for finding the forecast timestamps and on the

nature of the simulated model. Most technical systems such as VHDL-, logic- or high level systems - even coupled differential equations[1] - show a periodical behavior in the event generation or can easily be partitioned to subsystems with a high prediction confidence for the communication channels between them (for instance when using flip flop states as interface signals). Nevertheless, finding no partner event ee($\Phi_{i,k}$) with identical timestamp does not necessarily result in an uncorrectable error for an event prediction. An error would mean that the correct simulation behavior in means of the system state time function differs from the correct time behavior. When the system state S is causal and time-invariant, it is a function of the last system state and of the next event e in the EVL with ts(e) ≥ *LVT*, but *not* a function of time. This means, that the state function at simulation step n will always change in the same way for a given event e_{n+1}, independently from the *time* of the event execution: $S^{n+1} = \sigma(S^n, e_{n+1})$. Consequently, it is allowed (with respect to correct further simulation) to change the execution time of a given event as long as it is not moved on the simulation time axis over the timestamp of another event *affecting the same state variables*. Events fulfilling this condition will be called *causally independent* from each other in the following. Of course, in the time span between those event shifts, the system state change (if any) appears either too early or too late. This fact will be considered separately in a later chapter. The important fact in this context is, that after a legal event shift, the system state function will be identical with the original function. This fact offers a

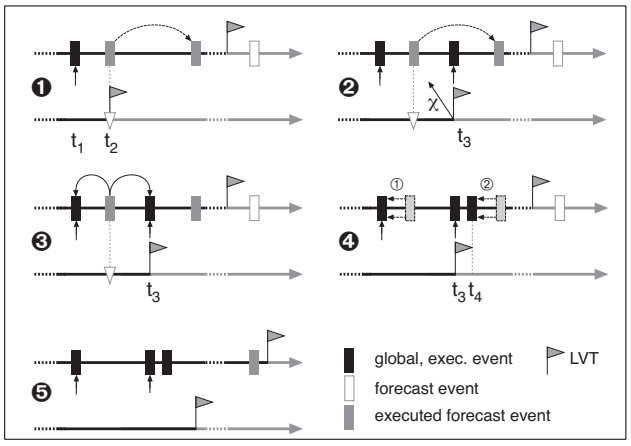

Figure 3: Event Shift

degree of freedom which is exploited in the PTW algorithm to eliminate correctable prediction faults. Fig. 3 shows a legal shift operation. The first picture shows an already executed primary and its secondary forecast event (solid gray). Picture two shows the receipt of an external event with timestamp t_3 causing a shift operation. As no valid partner event can be found, both neighbor events to the forecast event are considered as shift targets (picture 3). Chosen is the earlier event at t_1. Both primary and secondary event are shifted by exactly that virtual time span which is necessary to move the primary forecast event (at t_2) to t_1(picture 4). The reason, why the set of secondary events Σ has to be shifted with their primary event is, that we assume a deterministic, time-invariant system. This means: if an event in reality is executed to (for example) a later instant, its secondary events will have to be executed to a later time point because the delay between both event executions is assumed constant and the evaluation of the state function is time-independent. The shifting of secondary events, however, requires a consideration of the legacy for the respective secondary events to be shifted. The same rules have to be applied on secondary events as on primary forecast events, which means, that on the one hand the causality constraint *lcc* has to be fulfilled for shifted secondary forecast events and on the other hand, no event shift may be performed over another event in the past or the future affecting the *same* state variables (i.e. a causally dependent event) because the order of event evaluation might be relevant to the resulting state:

$$S^{n+1} = \sigma(\sigma(S^n, e_1), e_2) \neq \sigma(\sigma(S^n, e_2), e_1) = S^{n+1*}$$

Hence, each primary and secondary forecast event has a certain time range it can legally be moved by in the future (δ_f) or in the past (δ_b). To keep the system deterministic and time-invariant, *the whole set* of secondary events Σ(Φ) has to be shifted by the *same* virtual time distance as their primary forecast event Φ. The actual shifting distance δ is influenced by both primary and secondary events and bounded by δ_b and δ_f according to $\delta_b \leq \delta \leq \delta_f$ ($\delta_b < 0$, $\delta_f > 0$). To find the legal shifting span for a primary or secondary event, the term of *event classes* is introduced. All events contained in the same event class affect the same state variables (and may consequently not be moved in a way which would change the execution order). An example for the definition of event classes in a real application will be given later in this paper. An event will additionally be called *dynamic*, if it belongs to the current set of shift events and *static* in the other case. Fig. 4 shows the determination process of the legal shifting distances δ_b and δ_f. The forecast (primary) event and its secondary events belong to the dynamic event classes ❶, ❷ and ❸ and are subject to the current shift operation. All three events are now moved towards future and past until an event of a static class with the same class number (①, ② or ③) is reached. As shown in the upper picture, the event of class ❶ determines the past boundary (finding δ_b) and the event of class ❷ defines the future-side boundary.

1. Using threshold interfaces for event conversion

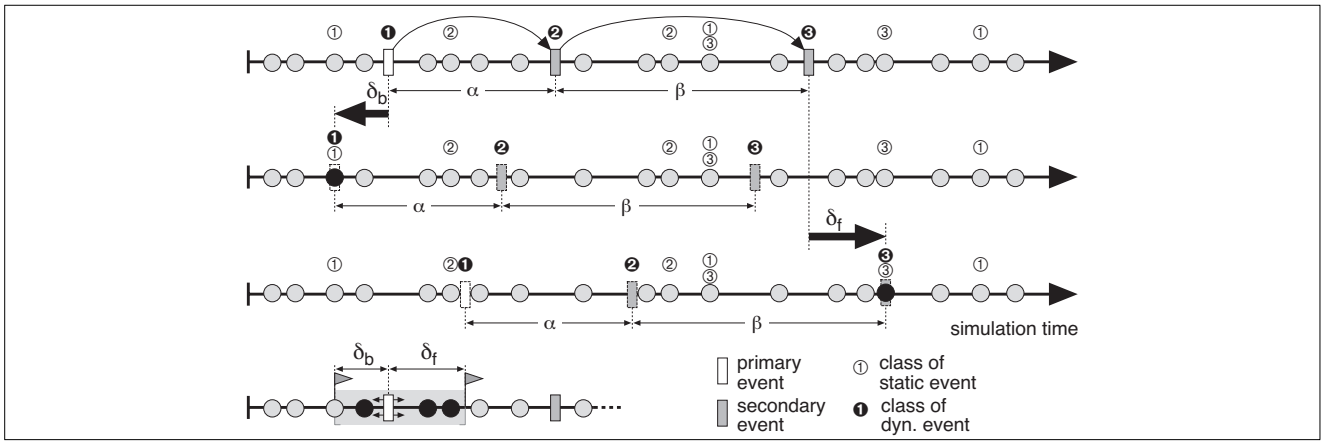

Figure 4: Shifting Process of Primary and Secondary Events

The remaining secondary event does not affect the solution as the static events of corresponding classes are too distant. The legal shifting area of the regarded event set $\{\Phi, \Sigma(\Phi)\}$ is shown in the lower picture of Fig. 4.

Event shift operations also result in a state function which may be delayed or advanced in the time span of the shifting area, however, afterwards is correct again. In case of event shift operations the state function bears some intervals within which the system state might be wrong (changes appear either too early or too late compared to the unshifted case). Fig. 5 shows the time behavior of the state function for three logical processes. Process ■ got a forecast event from process ● which has to be shifted together with the secondary event (②). This results in temporarily incorrect state function (data status). A forecast event becomes a confirmed event (solid color) when the LP, the prediction has been performed for, has sent at least one external event which could successfully be matched to the regarded forecast event. Thus, for each LP a local *forecast time LFT* can be defined behind (i.e. before) which no pending unmatched forecast event exists. With this definition, no more shift operations are possible for $t <$ LFT and the system state is *safe*. A lower limit for each LFT of course is the global virtual time GVT of the system. During simulation, each LP registers time spans of incorrect state resulting from shift operations. As PTW does not shift secondary external events, the behavior of any LP is identical to the behavior it had without shift operations.

The difference between causality violations and event shifts is that shift operations do not change the *order* of event occurrences in relevant cases and accordingly, any LP continues processing its EVL without having to roll back. In shifted intervals, the state function shows a delayed or premature behavior but the respective time spans are well-known and always are located *before GVT*, so they are correctable (see below). In no case, the delayed or premature behavior can cause erroneous simulation results for the future simulation, which supports the optimistic character of PTW. However, as mentioned before, the wrong timing of the system function - even if it is well-known - has to be considered. As described above, this is not relevant for the ongoing system state calculation or for the event traffic on interfaces between LPs but tools directly accessing the *internal* state of the LP, such as graphical user front ends also would show the wrong timing of the system function (in the simulation of a logic circuit this might be a *monitor* on a certain pin). To solve this problem, the state calculation function and the output function (writing simulation results in output buffers) are separated. Each LP_i writes its output information at $t = LFT_i$ *- after the timing has been corrected*, i.e. after a reshift has been performed (see intervals *valid data* and *corrected data* in Fig. 5).

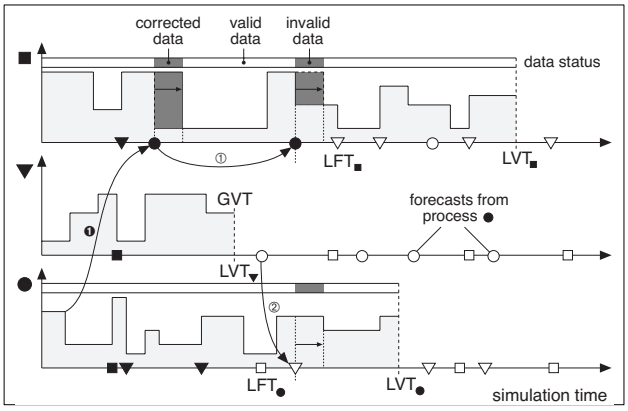

Figure 5: Event Shifting and State Function

3 Implementation

PTW is as synchronization mechanism in a cosimulation environment based on a Simulation Backplane concept as part of a design and specification environment for heterogeneous electronic systems and microsystems [13], [14].

PTW has been implemented on a Sun SPARC 10 workstation cluster using up to 6 machines for the distributed simulation via ethernet running one process of a logic simulator on each host. Among others, ISCAS circuits had been simulated with a varying size of 10k gates up to 100k gates. The partitions were chosen with respect to a similar number of gates. The number of interpartition connections did not have any obvious influence on the comparison between TW and PTW. For a distributed simulation with partition size of 100k gates, a typical channel size of 50-100 interface signals had been assumed, however, was not restricted to this. The partitioning rules preferred flip flop states as boundary signals on the output side of a design.

The first result to be mentioned here is the relation between the time resources spent for the computation process and for message handing (sending, receiving via TCP/IP and general message management) which was bounded by few percents, further decreasing with a rising partition size.

In the comparison of Time Warp (using fossil collection and lazy cancellation) and Predictive Time Warp under similar conditions, an average prediction confidence between 0.8 and 0.95 on each data flow could be observed which mandates for pattern-oriented prediction algorithms. The needed computation resources used by the predictor allowed for the integration of the PE in the communication interface of any LP without obviously affecting the overall simulation performance. The reduced rollback frequency, however, turned out to be model-dependent. It varied between ca. 20% and 60% compared to the classical TW approach.

4 Conclusions and Future Work

PTW as a special optimistic synchronization approach has proven to be well-suited for cases of distributed discrete event simulation where binary events can be used for the communication between logical processes. In logic simulation it reduced the rollback frequency in the optimistic protocol up to 60% compared to classical Time Warp.

Future work will examine, how the requirement of binary events can be weakened to multi-valued events by also predicting the value space. Especially in the area of mixed-mode simulation with threshold event triggers on interface signals between analog and digital circuit partitions, PTW seems to be successfully applicable.

5 Literature

[1] BALL, D., AND HOYT, S.: The Adaptive Time Warp Concurrency Control Algorithm, *Proceedings of the SCS Multiconference on Distributed Simulation*, 22(1): 174-177, 1990.

[2] BELL, T.C., CLEARY, J.G., AND WITTEN, I.H.: *Text Compression*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1990.

[3] BOX, G.E.P., AND JENKINS, G.M.: *Time Series Analysis: Forecasting and Control*, Holden Day, San Francisco, CA, USA, 1970.

[4] BOX, G.E.P., JENKINS, G.M., AND REINSEL, G.C.: *Time Series Analysis: Forecasting and Control*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1994.

[5] BRYANT, R.E.: Simulation on a Distributed System, *Proceedings of the Conference on Distributed Computing Systems*,: 544-552, 1979.

[6] CHANDY, K., AND MISRA, J.: Distributed Simulation: Asynchronous Distributed Simulation via a Sequence of Parallel Computations, *Communications of the ACM*, 24(11): 198-206, 1981.

[7] CLEARY, J.G., AND WITTEN, I.H.: Data Compression Using Adaptive Coding and Partial String Matching, *IEEE Transactions on Communications*, 32(4), 396-402, 1984.

[8] DICKENS, P.M., AND REYNOLDS, P.F.: SRADS with Local Rollback, *Proceedings of the SCS Multiconference on Distributed Simulation*, 22(1): 161-164, 1990.

[9] FERSCHA, A.: Probabilistic Adaptive Direct Optimism Control in Time Warp, *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*: 120-129, 1995.

[10] HAMILTON, J.D.: *Time Series Analysis*, Princeton University Press, Princeton, NJ, USA, 1994.

[11] JEFFERSON, D.R.: Virtual Time, *ACM Transactions on Programming Languages and Systems*, 7(3): 404-425, 1985.

[12] JEFFERSON, D.R., AND SOWIZRAL, H.: Fast Concurrent Simulation Using the Time Warp Mechanism, *Proceedings of the Conference on Distributed Simulation*, 63-69, 1985.

[13] SCHMERLER, S., TANURHAN, Y., AND MÜLLER-GLASER, K.D.: A Backplane Approach for Cosimulation in High-Level System Specification Environments, *Proceedings of the European Design Automation Conference EURO-DAC '95*: 262-267, 1995.

[14] SCHMERLER, S., TANURHAN, Y., AND MÜLLER-GLASER, K.D.: Predictive Time Warp, In: *Proceedings of the 11th European Simulation Multiconference ESM'97,* SCS, p.40-47, 1997.

[15] Sokol, L.M., Briskoe, D.P., and Wieland, A.P., MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution, *Proceedings of the SCS Multiconference on Distributed Simulation*, 34-42, 1988.

[16] STEINMANN, J.: Breathing Time Warp, *Proceedings of the 9th Workshop on Parallell and Distributed Simulation*, 109-118, 1993.

[17] ZIV, J., AND LEMPEL, A.: A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory*, 23(3): 337-343, 1977.

[18] ZIV, J., AND LEMPEL, A.: Compression of Individual Sequences via Variable-Rate Coding, *IEEE Transactions on Information Theory*, 24(5): 530-536, 1978.