A Short Introduction to the TRANSYT Verification Tool

Enric Pastor, Marco Peña and Marc Solé

UPC/DAC Report No. RR-2004/14 April 2004

This work has been supported in part by the Ministry of Education of Spain (CICYT) under contract TIC 2001-2476-C03-02 and grant AP2001-2819.

A Short Introduction to the TRANSYT Verification Tool

Enric Pastor, Marco Peña and Marc Solé

Department of Computer Architecture
Technical University of Catalonia
08860 Castelldefels (Barcelona), Spain
{enric, marcoa, msole}@ac.upc.es

April 2, 2004

Abstract

This document provides a brief introduction to TRANSYT, a tool designed for the verification of both untimed and timed asynchronous concurrent systems. The tool functionalities as well as its architecture is described. References to published research related to TRANSYT are also provided.

1 Introduction

TRANSYT is a BDD-based tool specifically designed for the verification of timed and untimed asynchronous concurrent systems. TRANSYT system architecture is designed to be modular, open and flexible, such that additional capabilities can be easily integrated. An state of the art BDD package [1] is integrated into the system, and a middleware extension [2] provides support complex BDD manipulation strategies.

TSIF (Transition System Interchange Format) is the main input language of TRANSYT. TSIF is a low level languaje designed to describe asynchronous event-based systems, although synchronous systems can be also described. Many formalisms can be mapped onto it, digital circuits, Petri nets, etc. TRANSYT integrates specialized algorithms for untimed reachability analysis based on disjunctive TR partitioning, and relative-time verification for timed systems. Invariant verification for both timed and untimed systems is fully supported, while CTL model checking is currently supported for untimed systems. TRANSYT provides orders of magnitude improvement over general untimed verification tools like NuSMV [3] and VIS [4], and expands the horizon of timed verification to middle-size real examples.

TRANSYT is designed with a precise application field in mind: (1) asynchronous systems, including digital circuits and any type of event-based description formalism; and (2) mixed synchronous-asynchronous systems or synchronous systems with multiple clock domains, like Globally Asynchronous Locally Synchronous (GALS) systems.

2 System Functionalities

We provide here a high-level overview of the the most relevant features of TRANSYT as offered to the user. Details of the architecture and algorithms will be provided in Section 3.

User Interaction

TRANSYT works with an interactive shell, processing systems according to command-line options. The user can activate all phases of the verification process (file parsing, system construction, reachability analysis, model checking, simulation, counter-example generation, etc.) with full control of all available options. On top of the interactive shell, a limited but under expansion GUI front-end offers access to all interactive commands, as well as an improved visualization of the systems and the properties under analysis. Visualization and manipulation of the state-space and the counter-example traces is also offered via the Graphviz toolkit [5]. Additionally, batch files can also be processed.

System Description

TRANSYT can process hierarchical systems formalized (using the TSIF format) as a set of variables to encode the state and *events* to describe the "actions" that the system can execute, i.e. the state changes that can be executed. Systems can be simultaneously coordinated by variable interchange or event synchronization. Other formalisms can be encoded and easily mapped onto TSIF. Currently we offer a front-end for BLIF [4] and Petri nets [6], and we are working toward a new SMV front-end.

Reachability Analysis

TRANSYT implements specialized reachability schemes based on disjunctive TRs and uses a mixed BFS/DFS traversal that schedules the application of the TR parts in order to maximize the state generation ratio and minimize BDD peaks [7]. These algorithms have demonstrated orders of magnitude improvement over existing BFS / conjunctive TR traversal schemes (e.g. [3, 4]) when applied to asynchronous concurrent systems. State-of-the art conjunctive TR traversal schemes are also available to efficiently manipulate mixed synchronous/asynchronous systems.

Model Checking

TRANSYT implements fair CTL model checking as defined in [8], and also offers specialized on-the-fly invariant verification. The tool can be configured to detect a minimum number of failures in a single traversal. Failing states are stored to allow selective counter-example generation for all of them. Failing states are stored associated to the verified properties in order to allow selective counter-example generation for all of them.

Semi-formal Verification

In addition to classical reachability analysis, TRANSYT offers an automated two-phase simulation-verification hybrid scheme [9]. Simulation follows a branching scheme that generates traces as divergent as possible (interleaved traces will be rejected). All generated traces are stored for further analysis. The second phase will select a number of simulation traces as seed of a guided-traversal algorithm. Guided-traversal exploits the behavioral information in the traces to efficiently identify additional states. On-the-fly invariant verification can be carried out during both phases.

First, simulation follows a low-CPU-cost branching scheme oriented toward generating execution traces as divergent as possible (interleaved traces will be detected and rejected). On-the-fly invariant verification can be carried out, and the user can specify the minimum number of failures to be detected. All generated traces are stored for further analysis. The second phase will select a number of simulation traces as seed of a

guided-traversal algorithm. Guided-traversal exploits the behavioral information in the traces to efficiently identify additional states. On-the-fly invariant verification can be also carried out until the selected number of failures has been identified.

Relative-time Verification

TRANSYT offers invariant verification of timed systems based on the *relative-timing* paradigm [10, 11]. TSIF events can be annotated with mix-max delays. If it exists, the tool provides a timed counter-example. Otherwise, TRANSYT provides a set of graphic structures that inform the user about how the execution of events in the system is ordered due to timing. Note that not all existing orderings are provided, but just those that are relevant to prove the invariants under verification.

3 Tool Architecture and Algorithms

This section describes the main functional modules in TRANSYT (see Figure 1), the peculiarities of the algorithms implemented in them and their interrelations. Currently TRANSYTcontains up to 28 modules, many of them providing low level services that are not accessed directly by the user, but available to the programmer.

The System Instantiation and Boolean Model Construction provides support for creating the internal representation of the TSIF format. The system is mapped onto a Boolean model after an encoding process, in which TRs, properties, etc. are constructed. Currently these process is supported by BDDs, although provisions exists for other data structures like MDDs. Once the TRs are build, their causal interrelations can be analyzed in order to determine if the application of a TR part may trigger the execution of other TRs. The granularity of the TR partition is decided here depending on TR size, number of parts and detected causality. Different partition schemes can be used according to these parameters.

The BDD-based verification module is the core of TRANSYT. It implements highly efficient traversal schemes based on mixed BFS/DFS algorithms that schedule the application of the disjunctive TR parts. New states generated by one part are immediately reused as source for following parts. If causality is taken into account, these simple scheme provides orders-of-magnitude improvement over BFS traversal. Based on BFS/DFS traversal, on-the-fly invariant verification and fair CTL model checking can be performed on the selected design. Currently, CTL only partially exploits BFS/DFS traveresal. Once failures are detected, counter-example traces can be generated and stored —associated to each failing property— for both further manipulation or visualization.

An alternative invariant verification approach, combining simulation and guided-traversal, is provided. Simulation can be executed following a branching strategy that resembles partial-order verification. At each visited state the causality between enabled events is analyzed. In case of detecting concurrent events, only one of the execution traces is followed. In case of conflict (i.e. a choice in the execution path) both execution traces are explored. Active execution traces are stored in a prioritized list. Exploration continues until no additional states are available or until a certain number of failures have been identified. Explored traces are stored for future analysis. Once traces are available, causality information can be extracted from traces. Given a trace, its associated causality can be extracted and directly applied to a guided-traversal process. Guided-traversal executes a BFS/DFS reachability analysis applying events in the best order as indicated by causality.

TRANSYT extends symbolic invariant verification to timed systems. The use of *relative timing* [12] allows to avoid the computation of the exact timed state space of the system. Instead, the timed behavior of events is captured by means of partial orders that represent relative temporal relations. Timed systems provide delays for all the events in the system, however many of the constraints imposed by such delays

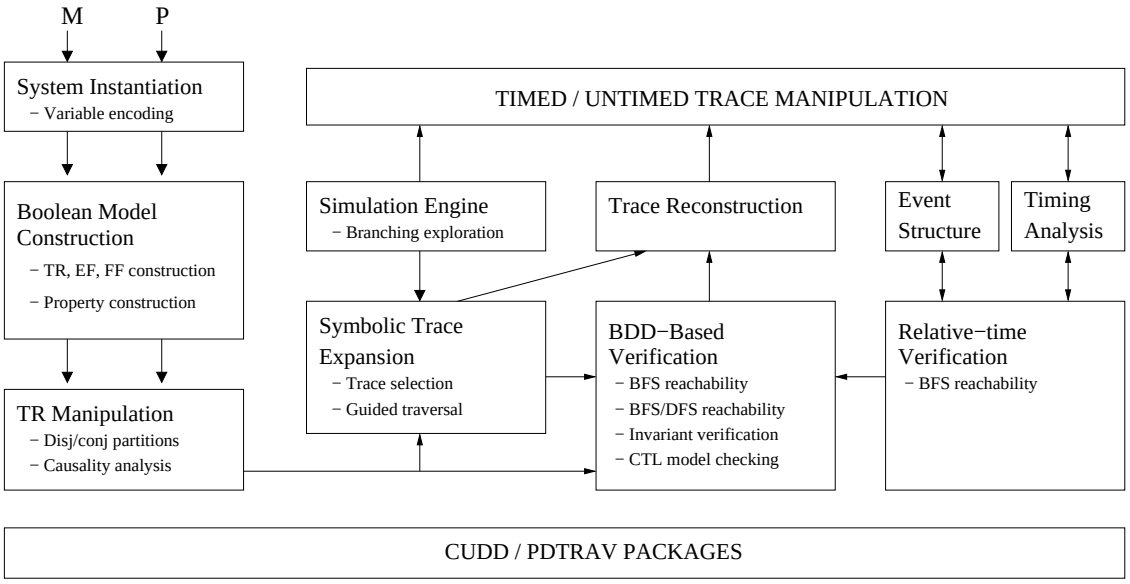

Figure 1: TRANSYT modular architecture.

are not actually required. TRANSYT only considers timing information in an *on-demand* basis, as long as it is required to prove a given property. Moreover, the timing analysis is performed over the subset of events involved in such proof by any external timing analysis algorithm. As a result, the untimed state space of the system is refined incrementally as long as new timing information is incorporated. This incremental technique allows to obtain at least partial results even on systems for which complete solutions could be too complex to compute. The tool not only proves or disproves the correctness of the system with respect to a set of invariants, but also provides a set of sufficient relative-time relations that guarantee such correctness or a counterexample trace.

4 Results and future directions

TRANSYT [13] is a well-structured and flexible platform designed to be applicable to the verification of asynchronous concurrent systems. The TSIF front-end provides a flexible entry point for most specification languages relevant to the area. The precise separation between different modules allows the implementation of additional functionalities and algorithms for each phase of the verification flow. The performance of the tool is extremely satisfactory due to the BDD package and its specific algorithms.

TRANSYT has been successfully used to analyze a number systems, both in the timed and untimed domain. Extensive comparisons have been carried out with the state generation engines in NuSMV [3] and VIS [4]. In both cases, orders of magnitude improvements have been obtained when analyzing real-live examples [14]. Complex timed systems have been also analyzed using TRANSYT. In particular, several interface FIFO implementations (IPCMOS by S. Schuster and STARI by M.R. Greenstreet) connecting different clock domains have been successfully verified.

Currently, several new functionalities are under development. A mixed conjunctive/disjunctive TR construction and scheduling scheme is being implemented for complex Globally Asynchronous Locally Synchronous (GALS) systems. The CTL verification algorithm is being upgraded to exploit the same causality information used during the reachability process. A restricted version of timed-CTL is being developed to be integrated with the relative-time verification engine. On the user's side, better feedback visualization is

being implemented through more powerful visual libraries.

References

[1] F. Somenzi, "CUDD: CU decision diagram package release," 1998.

[2] G. Cabodi and S. Quer, "PdTRAV package politecnico di torino reachability analysis for verification (release 1.2)," 1999.

[3] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, 2000.

[4] T. V. Group, "VIS: A system for verification and synthesis," in *Proc. International Workshop on Computer Aided Verification*, vol. 1102 of *LNCS*, pp. 428–432, Springer-Verlag, 1996.

[5] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software – Practice & Experience*, vol. 30, no. 11, pp. 1203–1233, 2000.

[6] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, vol. E80-D, no. 3, pp. 315–325, 1997.

[7] M. Solé and E. Pastor, "Traversal techniques for concurrent systems," in *Proc. Formal Methods in Computed-Aided Design (FMCAD 2002)*, vol. 2517 of *LNCS*, (Portland, Oregon, USA), pp. 220–238, Springer-Verlag, November 2002.

[8] E. M. Clarke, O. Grumberg, K. L. McMillan, and X. Zhao, "Efficient generation of counterexamples and witnesses in symbolic model checking," in *Proceedings of the 32nd ACM/IEEE conference on Design automation conference*, pp. 427–432, 1995.

[9] E. Pastor and M. A. Peña, "Efficient hybrid reachability analysis for asynchronous concurrent systems," in *12th Advanced Research Working Conference on Correct Hardware Design and Verification Methods*, vol. 2860 of *LNCS*, (L'Aquila, (Italy)), pp. 378–393, Springer-Verlag, October 2003.

[10] M. Peña, J. Cortadella, A. Kondratyev, and E. Pastor, "Formal verification of safety properties in timed circuits," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, (Eilat, Israel), pp. 2–11, Apr. 2000.

[11] M. Peña, J. Cortadella, A. Smirnov, and E. Pastor, "A case study for the verification of complex timed circuits: IPCMOS," in *Proc. Design, Automation and Test in Europe*, (Paris, France), pp. 44–51, Mar. 2002.

[12] K. Stevens, R. Ginosar, and S. Rotem, "Relative timing," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 208–218, Apr. 1999.

[13] "`http://research.ac.upc.es/VLSI/transyt/transyt.html`."

[14] M. Sole and E. Pastor, "Evaluating symbolic traversal algorithms applied to asynchronous concurrent systems," Tech. Rep. RR-2004/07, UPC/DAC, Jan. 2004.