IGR Report GR/M94366/01
Model Visualisation for Asynchronous Circuit Design (MOVIE)

1 Background/Context

The state of the art in asynchronous design tools is presented in the 2nd edition of ACiD-WG report "Design, Automation and Test for Asynchronous Circuits and Systems" by D.A.Edwards and W.B.Toms, which is available http://www.scism.sbu.ac.uk/ccsv/ACiD-WG/. This site also contains a report on the status of asynchronous design in industry. It clearly reflects an increased industrial interest in asynchronous design both in the UK and abroad, particularly in response to the design and test challenges of the deep submicron era (cf. ITRS-2001). The role of and demand for a flexible design flow is rapidly increasing as was emphasized at the last three (EU-funded) ACiD-WG workshops. One of the key stages in a successful design flow, with a good balance between productivity and optimality, would be automatic synthesis from behavioural descriptions.

The MOVIE project has been part of Newcastle's steady involvement in the research in asynchronous synthesis over the last ten years. This research found its support from EPSRC through the ASAP (GR/J52327) grant, in collaboration with the University of Bristol (Prof. E. Dagless), and visiting fellowships (GR/J72486, J78334, L24038, M94359) of Professors M. Kishinevsky, L. Lavagno, A. Kondratyev and N. Starodoubtsev, which was accompanied by travel grants from the British Council and Spanish Ministry of Science and Culture to support collaboration with the Group of Prof. J. Cortadella from UPC (Barcelona). The main result of this work was a set of synthesis methods and a software tool Petrify described in the recent monograph [1].

The research carried out under the MOVIE grant was a natural progression from those developments as well as the long-standing Newcastle's research in the area of concurrent system modelling and analysis. This project tackled a number of limitations of the state-of-the-art of Petrify (and more indirectly, other relevant tools such as e.g. MINIMALIST) and was aimed at extending the power of modelling and visualisation techniques and the applicability of synthesis tools for a broader range of users – a critical issue for the process of promulgation of asynchronous design techniques into industrial domains. Particular emphasis has been on improving visualisation support within the logic synthesis environment based on Petri nets and Signal Transition Graphs (STGs), where designer had little or no control over the circuit level solution. This problem had a particularly negative effect on the design of relatively small-size, long-life, re-usable circuit components, such as generic parts of control logic, e.g. arbiters, interrupt handlers, interface controllers [1]. Some of these circuits, though small in size, could be a bottleneck in systems in terms of timing and thus serriously affect systems performance (cf. pipeline latch controllers).

Petrify [1] is probably the only synthesis tool that attempts to work with highly concurrent circuits at the logic level. In order to synthesise an asynchronous controller using Petrify, the designer must enter its behavioural specification in the form of an STG (conceptually similar to Timing Diagrams), which is an interpreted Petri net. Petrify produces a gate netlist for the circuit implementation. The process of synthesis involves model analysis and transformation, including new state signal insertion or concurrency reduction/expansion. Lacking crucial interaction with the designer, the tool may produce inadequate circuit solutions (e.g. irregular and asymmetric) or sometimes fail in solving problems with state coding, decomposition and technology mapping, etc. Manual design often yields more compact and elegant solutions: e.g., using certain model refinements at the behavioural level one can build a circuit out of negative gates. To cope with the above problem, the MOVIE project set out to investigate the development of techniques and tools which could expose the highly concurrent behavioural patterns to the designer but in such a way that he would be able to easily grasp the characteristic patterns of circuit behaviour and manipulate the model more interactively. This required better ways of visualisation of STGs, Petri nets, Transition Systems, State Graphs, STG Unfoldings, and for efficient graphical transformations to be performed on these models.

This research was closely related with other projects being conducted at Newcastle, such as the ones on parallel model checking (PMC, GR/M99293), on asynchronous communication mechanisms (COMFORT, GR/L93775), and more recently the BESST project (GR/R16754) focused on behavioural synthesis of systems with heterogeneous timing.

2 Key Advances and Supporting Methodology

The **original aim** of the project was the development of theory and an associated set of tools for graphical representation of asynchronous circuit behaviour, to support the process of synthesis of such circuits by both skilled and inexperienced designers. The tools would enable the former to achieve higher quality and productivity, and greater confidence in their designs. For the latter, the tools would provide help, through adequate visual association, in understanding the specific nature of concurrent processes in asynchronous circuits, and in capturing characteristic, and often local, features of their behaviour. Behaviour that appears very complex in terms of the size of its reachable state space (and this is the primary obstacle for the traditional "sequential designer") could be made intuitively simple if represented adequately. Such a step in further development of asynchronous synthesis tools was deemed essential for their greater exploitation by practical designers and their promulgation into commercial CAD packages. Such tools would also play a key role in training asynchronous designers.

2.1 Relationship between concurrency models in asynchronous design. State-transition systems

The work on the project started from a wide ranging investigation aimed at identifying a key set of models, *based on graphs*, to represent the concurrent behaviour in circuits, both at the level of their initial specification and their logical implementation (**first objective**). The types of models involved two main categories, *state-based* and *partial order based*. While state-based models play an important role in logic synthesis of circuits in tools like Petrify, their ability to visualise asynchronous circuit behaviour is limited, mainly due to the state explosion for highly concurrent models. Nevertheless, sometimes designers using STGs and Petri nets as initial specifications, require access to state-based information when certain properties, crucial for analysis of circuit implementability are derived from state transition systems. For example, properties such as non-persistence, effects of timing constraints and the manifestation of state encoding conflicts were seen to be appropriate candidates for visualisation in the state-based domain. This work was presented in a number of publications [1, 3, 21, 36].

The **second objective** of the project was to develop a method and a tool for visualisation of state-transition systems in their different representation forms. We have investigated two major types of state-based models, and how they could be visualised as hypercubes in their orthographic projection from N-space to 2-space. The first form is an uninterpreted state transition system, with parallel edges based on events, applicable for transition signalling and two-phase asynchronous control protocols. The second form was a binary-encoded state graph, with parallel edges corresponding to signals (or signal transitions). The use of this projection and the property of preserving edge parallelism between N-space and 2-space, is valuable in understanding the relationships between states, regions of states and events in the system. From this projection, used for the unencoded form, it is easy to see which events are concurrent, by examining diamond structures with confluence, and which events disable each other because they are in conflict. In the latter case, depending on whether such events are inputs or outputs, one could conclude whether the conflict stands for the environment's choice or internal arbitration. These investigations have resulted in the development of the GraphPad tool (see below) and subsequent interfacing with the tools for verification of arbitration conditions in circuits, such as Versify from UPC Barcelona. This work was closely linked with the investigation of behaviour of arbiters, synchronisers and multistable devices using analytical methods and Matlab software [10]. In the future we seek for more research in the area of mixed (discrete and continuuuous) mode representations, combining transition systems and phase portraits. Some initial work in this direction was began within an MSc project [27].

Within MOVIE we continued our collaboration with the Petrify consortium, involving Intel Research Labs, on the development of models of asynchronous circuits with timing constraints. The latter play crucial role in supporting logic synthesis in Petrify, both through solving various implementability problems (such as CSC) and logic circuit optimization [2]. An important part of this work was the idea of visualising the timing constraints at the STG level. This provided crucial methodological links with the tools developed for the **third objective** of the project, which appeared to be central to this project. The next section describes this work in detail.

2.2 Use of unfoldings in visualisation and interactive synthesis of asynchronous circuits

In our original plans we expected that algorithms for analysis of concurrent behaviour based on Petri net unfoldings (originated from the work by K.McMillan of 1992 and further advanced by J.Esparza and others) could not only be efficient for verification but also for visualisation support in asynchronous design. The crucial factor was the compactness of the representation of concurrent processes in unfoldings and the possibility of using finite prefixes as opposed to infinite branching processes. When compared with other analysis techniques, by explicitly representing causality, concurrency and choice, unfoldings support verification based on local states, and yield a direct visual representation of complex concurrent behaviours. This is particularly useful for the debugging and development process, especially at its early stages. Increasing research efforts in this direction was quite natural for Newcastle, where this research had been in progress since 1993 (PhD work of Alex Semenov "Verification and Synthesis of Asynchronous Control Circuits Using Petri Net Unfoldings" and a number of conference and journal publications).

Our continued research on the use of unfoldings for visualisation was closely linked with that of model-checking [6]. Both directions required further advancement of the basic algorithms for constructing the unfolding prefix which could capture the characteristic part of the partial order semantics sufficient for demonstrating critical safety and liveness properties. Overall, the following advancements were made in this area.

Separating concurrency from choice. One of the problems with the direct use of unfoldings is the fact that, if the behaviour of the modelled system is a mix of non-deterministic choice and concurrency, the structure of the unfolding may be quite complex. This particularly affects the visualisation aspect. We investigated the branching process model underlying the net unfolding, and developed algorithms for the visualisation of a finite complete prefix of a Petri net or an STG. This required defining a key transformation to convert such a prefix into a hierarchical (two-level) semantic model. At the top level, it had a finite state machine, describing the modes of operation and transitions between them. At the low level, there were acyclic marked graphs, which could be drawn as waveforms, embedded into the top level nodes. The models of both levels confirmed closely to abstractions traditionally used by electronics engineers. The resultant model was equivalent to the original prefix in terms of its completed trace semantics. Moreover, the branching structure of the prefix maximally preserved. This work was presented in [9, 32] and implemented in tool `unfPad` (see below).

Generation of possible extensions of a prefix. The problem of generating a complete prefix of the unfolding sufficient for detecting or visualising required properties is highly important. The core of this problem is an efficient way of extending a prefix. Prior to our efforts, this problem was addressed by Esparza and Römer, who considerably improved on the original McMillan's technique by maintaining the concurrency relation. This approach is fast for simple systems, but soon deteriorates as the amount of memory needed to store the concurrency relation is quadratic in the size of the already built part of the prefix. In [29, 7], we proposed another method, which is fully compatible with the concurrency relation approach but can also be used independently. The essence of the method is to add new transition instances to the prefix being built not by trying all the transitions one at a time, but several at once, merging the common parts of the work. Experimental results demonstrated that one can achieve significant speedups if the transitions of the Petri net being unfolded have overlapping parts.

Canonical prefixes. While proving the correctness of the parallel unfolding algorithm, it has been discovered that generated prefixes enjoy a property akin to canonicity. We expounded this in [31, 11, 5], where a general framework for truncating Petri net unfoldings has been developed. It provides a powerful tool for dealing with different variants of the unfolding technique, in a flexible and uniform way. In particular, by finely tuning the so-called *cutting contexts,* one can build prefixes which better suit a particular model checking or visualisation problem. A fundamental result is that, for an arbitrary Petri net and a cutting context, there exists a 'special' *canonical* prefix of its unfolding, which can be *defined without resorting to any algorithmic argument.* This should be contrasted with former approaches, where a prefix was 'defined' as any of the results produced by a non-deterministic unfolding algorithm, which resulted in a very 'inconvenient' theory. We introduced a new, stronger notion of completeness of prefixes, which was implicitly assumed by many existing model checking algorithms employing unfoldings. We have shown that the canonical prefix is complete w.r.t. this notion. As a result, relevant verification and visualisation tools can now make stronger assumptions about the properties of the prefixes they use. In particular, they can safely assume that for each configuration containing no cut-off events, all firings are preserved.

Prefixes of high level nets and processes of inhibitor nets. In [33, 15], we defined branching processes and unfoldings of high-level Petri nets and proposed an algorithm which builds finite and complete prefixes of such nets. We established an important relation between the branching processes of a high-level net and those of its low-level *expansion*, viz. that the sets of their branching processes are the same, allowing us to import results proven for low-level nets. Among such results are the canonicity of the prefix for different cutting contexts. The experiments demonstrated that it is, on one hand, superior to the traditional approach for data-intensive application, and, on the other hand, has the same performance for control-intensive ones.

Asynchronous circuits are often efficiently modelled by Petri nets with inhibitor arcs. Although such nets in the bounded case can be easily simulated by ordinary Petri nets at the level of interleaving (state-based) semantics, their representation in terms of true causality semantics, such a using an unfolding technique, is a non-trivial problem. In [4, 13] we laid a formal foundation for further developments in the area of unfolding nets with inhibitor arcs by constructing theory of branching processes for them.

Detection of state coding conflicts. One of the stages of circuit synthesis based on STGs involves the detection of state coding (CSC [1]) conflicts, and their elimination. In [30, 8], we proposed a solution for the CSC checking problem, characterized in terms of a system of integer constraints, and developed an efficient technique for solving such a system. The integer programming based algorithm in many cases achieved significant speedups, but on some of the benchmarks its performance was still not entirely satisfactory: on several large instances the test did not terminate within the time limit. In [34, 16] we characterized the CSC problem in terms of boolean satisfiability (SAT). Though being more complicated than the integer programming translation, the SAT one allows more dependencies between the variables to be exploited. State-of-the-art SAT solvers are quite efficient, and have been for long employed in the model checking community. In our experiments, we achieved significant speedups using this method. We also showed how the proposed translation can be extended to the USC and normalcy problems.

Visualisation and interactive resolution of CSC conflicts. The above results in theory of unfolding prefixes and algorithms for their efficient construction and detection of CSC conflicts have created a solid foundation for defining and tackling the central problem of this project, which was visualisation support and and interactive synthesis of asynchronous circuits from STGs. Having realised that the most problematic stage in circuit synthesis is that of CSC conflict detection and resolution, we concentrated on finding effective ways of visualising CSC conflicts. The major developments here were the compact representation of CSC conflicts in the unfolding prefix using complementary sets and conflict cores, and use of the "height map" technique for identifying the areas in the prefix with the largest concentration of conflicts. The latter provided an important heuristic guiding manual and semi-automatic resolution of conflicts with the additional signals. The method proved highly successful on a number practical design examples, such as control logic for an asynchronous A/D converter. It was described in [14] and implemented in the ConfRes tool [17].

2.3 Prototype tools

The **fourth and final objective** of MOVIE was the development of software tools and integration with the related existing synthesis and verification software such as Petrify and Versify. The following tools were maintained and developed within the project, and were extensively used to obtain experimental results:

- PUNT – Petri net and STG unfolder and verifier (Newcastle's first generation unfolding tool origianlly developed by Alex Semenov in his PhD). This tool uses Petrify's .g format and generates unfolding prefixes in the .g and .dot formats for subsequent visualisation using `dot`, a graph drawing software from Graphviz.
- PUNF – Petri net unfolder. It builds a finite and complete prefix of a safe Petri net or a safe STG (Petrify's format .g can be used as input. The prefixes generated by PUNF can be passed as an input to the CLP tool for model-checking and to the ConfRes tool for visualisation.
- CLP – Linear programming model checker. It uses a finite complete prefix of a Petri net or STG and can check deadlock freeness, reachability or coverability of a marking, and detect state coding conflicts in an STG. The state coding conflict information can be supplied to the ConfRes tool for visualisation.
- ConfRes – a tool for visualising cores of state coding conflicts in STG unfolding prefix and interactive resolution of conflicts using height map heuristics as described in [14, 17]. For actual graph displaying ConfRes calls `dot`, a graph drawing software from Graphviz.
- GraphPad – a tool for visualisation of state spaces of concurrent systems. It uses as input a state graph format, similar to the one generated by Petrify's utility write_sg or by Versify, and displays the state graph as a hypercube in its orthographic projection from N-space to 2-space. The tool exports the state graph in postscript.
- unfPad – a tool for visualisation of unfoldings with choice seprated from concurrency. It takes an unfolding produced by PUNT in .dot format and converts it into another .dot file for subsequent visualisation, using utility tree, in 2-level models described in [9].

These tools can be downloaded from:
`http://www.cs.ncl.ac.uk/people/victor.khomenko/home.formal/tools/tools.html` (PUNF and CLP)
and `http://async.org.uk/movie/` (other tools).

2.4 Research output

Overall, the project has resulted in contributions to one monograph [1], four journal papers [2, 3, 4, 5], twelve peer reviewed conference papers [6]–[17], two PhD Theses [21, 22] (plus one PhD thesis, by A. Madalinski, is in write-up), five MSc dissertations [23]–[27] seven technical reports [28]–[34], two submitted journal articles [35], and six tools.

3 Research Impact and Benefits to Society

Our research group at Newcastle considers this research to have been very successful. All of the tasks have been completed and in many cases the results exceeded the orginal expectation considerably. For example, the major impact has been in developing the partial order techniques, based on Petri net unfoldings. Publications, numerous presentations of academic and visiting staff, RAs, PG students in our weekly seminar of the Asynchronous Systems

Laboratory (see `http://www.cs.ncl.ac.uk/events/ASL`), have played an important role in promoting this results into a number of follow-up projects and external contacts. Our major deliverable on this project, paper [14], has been selected for a special issue of the IEE Proc CDT journal, a collection of the best papers at DATE'03, to be published in October 2003. The main author of this paper, Agnes Madalinski, is the PhD student associated with MOVIE. She is currently writing up her thesis. With this research Newcastle has been part of the Petrify team shortlisted for the Descartes 2002 Prize (see ftp://ftp.cordis.lu/pub/descartes/docs/2002descartes_prize_catalogue-en.pdf)

The results of the project are relevant to potential users at three different levels:

- The use of theoretical results, especially in the unfolding-based analysis, will benefit researchers in area of designing complex asynchronous and concurrent systems (verification and synthesis) in a wide range of industrial applications.
- The use of unfolding-based algorithms and tools in asynchronous design flow will benefit CAD developers and vendors working on promoting asynchronous design techniques in microelectronics.
- The use of the unfolding based tools as a way to represent concurrent behaviours, will benefit both industry and academia in offering courses on concurrency and asynchronous systems and providing efficient ways of semantic model structuring and visualization.

For example, these results will be useful to the UK research in asynchronous design, supported by EPSRC at Manchester and Cambridge (projects on Balsa synthesis environment and GALS). We are also planning to use these techniques in promoting asynchronous design at Atmel-UK.

4 Explanation of Expenditure

The expenditure plans in the original proposal have been followed without changes.

5 Further Research or Dissemination Activities

The research outcome from this grant is directly applicable for exploitation in our three EPSRC grants, on synthesis and testing of low latency asynchronous logic (STELLA, GR/R12036), on behavioural synthesis of systems with heterogeneous timing asynchronous communication (BESST, GR/R16754) and on mechanisms for SOCs held jointly between Newcastle and Kingston Universities (COHERENT, GR/R32666/R32895). The ideas of interactive synthesis from STGs and analysis of related properties is now part of PhD research of D. Sokolov and D. Koppad.

Besides publications, this research has had its impact on preparing two advanced tutorials at the international conferences: Adv. Tutorial on Hardware Design and Petri Nets at ICATPN2000 in Aarhus, June 2000, and Tutorial on Asynchronous Circuit Design in ASP-DAC-VLSI, Bangalore, Jan. 2002 [36]. It also affected preparing for ACiD-WG Summer School in Grenoble in July 2002. It will also be put forward in our presentations in the forthcoming conferences on Applications of Concurrency in Systems Design (ACSD'03), ASYNC'03 and our invited lectures at the fourth advanced course on Petri nets in Eichstätt in September 2003 (`http://www.acpn.de`).

References

[1] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev, Logic Synthesis of Asynchronous Controllers and Interfaces, Springer Series in Advanced Microelectronics, vol. 8, Springer (2002)

[2] J. Cortadella, M.Kishinevsky, S.M. Burns, K.S. Stevens, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev: Lazy Transition Systems and Asynhronous Circuit Synthesis with Relative Timing Assumptions. IEEE Trans. of CAD, Vol. 21, No. 2, pages 109-130 (2002)

[3] M.Pietkiewicz-Koutny: Synthesising Elementary Net Systems with Inhibitor Arcs from Step Transition Systems. Fundamenta Informaticae 50 (2002)

[4] H.C.M.Kleijn and M.Koutny: Process Semantics of General Inhibitor Nets. To appear in Information and Computation.

[5] V.Khomenko, M.Koutny and W.Vogler: Canonical Prefixes of Petri Net Unfoldings. To appear in Acta Informatica.

[6] V.Khomenko and M.Koutny: LP Deadlock Checking Using Partial Order Dependencies. Proc. of CONCUR'2000, Palamidessi C. (Ed.). Springer-Verlag, Lecture Notes in Computer Science 1877 (2000)

[7] V.Khomenko and M.Koutny: Towards An Efficient Algorithm for Unfolding Petri Nets. Proc. of CONCUR'2001, Larsen K.G., Nielsen M. (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2154 (2001)

[8] V.Khomenko, M.Koutny, and A.Yakovlev: Detecting State Coding Conflicts in STGs Using Integer Programming. Proc. of DATE'2002, Kloos C.D., Franca J. (Eds.). IEEE Computing Society (2002)

[9] A. Bystrov, M. Koutny and A. Yakovlev: Visualization of partial order models in VLSI design flow, Proc. DATE'02, Kloos C.D., Franca J. (Eds.). IEEE Computing Society (2002)

[10] O. Maevsky, D.J. Kinniment, A.Yakovlev, and A. Bystrov: Analysis of the oscillation problem in tri-flops, Proc. ISCAS'02, Scottsdale, Arizona, IEEE, volume I (2002)

[11] V. Khomenko, M. Koutny, and W. Vogler: Canonical Prefixes of Petri Net Unfoldings. Proc. of CAV'2002, Brinksma E., Larsen K.G. (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2404 (2002)

[12] A.Madalinski, A.Bystrov, and A.Yakovlev, Visualisation of Coding Conflicts in Asynchronous Circuit Design, IEEE/ACM 11th International Workshop on Logic and Synthesis, New Orleans, June 2002.

[13] H.C.M.Kleijn and M.Koutny: Causality Semantics of Petri Nets with Weighted Inhibitor Arcs. Proc. of CONCUR'02, L. Brim, P. Jancar, M.Ketinsky, A. Kucera (Eds.), Lecture Notes in Computer Science 2421 (2002)

[14] A. Madalinski, A. Bystrov, V. Khomenko, and A. Yakovlev: Visualization and Resolution of Coding Conflicts in Asynchronous Circuit Design. Proc. of DATE'2003, Verkest, D., Wehn, N. (Eds.). IEEE Computing Society (2003) (invited for a special issue of IEE Proc. with the best contributions from the technical program DATE03).

[15] V. Khomenko and M. Koutny: Branching Processes of High-Level Petri Nets. Proc. of TACAS'2003, Garavel, H., Hatcliff, J. (Eds.). Springer-Verlag, Lecture Notes in Computer Science 2619 (2003)

[16] V. Khomenko, M. Koutny, and A. Yakovlev: Detecting State Coding Conflicts in STG Unfoldings SAT. Proc. of ICACSD'2003, Balarin, F., Lilius, J. (Eds.). IEEE Computing Society (2003)

[17] A.Madalinski: ConfRes: Interactive Coding Conflict Resolver based on Core Visualisation. Proc. ICACSD'2003, Balarin, F., Lilius, J. (Eds.). IEEE Computing Society (2003)

[18] A. Bystrov, A. Yakovlev and M. Koutny, Visualisation of Partial Order Models in VLSI Design Flow, Proc. 10th UK Asynchronous Forum, University of Edinburgh (2001)

[19] A. Madalinski, A. Bystrov and A. Yakovlev: Visualisation of Coding Conflicts in Asynchronous Circuit Design Proc. 12th UK Asynchronous Forum, South Bank University, London (2002)

[20] A. Madalinski, A. Bystrov and A. Yakovlev: A software tool for State Coding Conflict Detection by Partial Order Techniques. Proc. 1st UK ACM SIGDA Workshop on Design Automation, Imperial College, London (2001)

[21] M.Pietkiewicz-Koutny: Relating Formal Models of Concurrency for the Modelling of Asynchronous Digital Hardware. PhD Thesis, Department of Computing Science, University of Newcastle upon Tyne (2000).

[22] V.Khomenko: Model Checking Based on Prefixes of Petri Net Unfoldings. PhD Thesis. School of Computing Science, University of Newcastle upon Tyne (2003)

[23] I.Williamson: The Visualization of Asynchronous System Behaviour in Timing Diagrams. MSc Dissertation. School of Computing Science, University of Newcastle upon Tyne (1999)

[24] L.Guo: Construction of Petri net models for Asynchronous Logic Circuits. MSc Dissertation. School of Computing Science, University of Newcastle upon Tyne (2000)

[25] A.A.Walker: Visualization and Animation of Asynchronous Circuit Behaviour. MSc Dissertation. School of Computing Science, University of Newcastle upon Tyne (2000)

[26] Z.Dong: Web-driven Interface for Asynchronous Circuit Visualisation. MSc Dissertation. School of Computing Science, University of Newcastle upon Tyne (2002)

[27] Ch.Lalawattanachai: The Visualisation of Metastability Behaviour of the Logical Circuit. MSc Dissertation. School of Computing Science, University of Newcastle upon Tyne (2002)

[28] S.Grey: Visualisation of State Spaces of Concurrent Systems, Project report, Dept. of CS, University of Newcastle upon Tyne (2001)

[29] V.Khomenko and M.Koutny: An Efficient Algorithm for Unfolding Petri Nets. Technical Report CS-TR-726, Dept. of Computing Science, University of Newcastle (2001)

[30] V.Khomenko, M.Koutny, and A.Yakovlev: Detecting State Coding Conflicts in STGs Using Integer Programming. Technical Report CS-TR-736, Dept. of Computing Science, University of Newcastle (2001)

[31] V.Khomenko, M.Koutny, and W.Vogler: Canonical Prefixes of Petri Net Unfoldings. Technical Report CS-TR-741, Dept. of Computing Science, University of Newcastle (2001)

[32] A.Bystrov, A.Yakovlev, and M.Koutny: Visualisation of Partial Order Models in VLSI Design Flow, CS-TR-744, Dept. of Computing Science, University of Newcastle (2001).

[33] V.Khomenko and M.Koutny: Branching Processes of High-Level Petri Nets. Technical Report CS-TR-763, Dept of Computing Science, University of Newcastle (2002)

[34] V.Khomenko, M.Koutny, and A.Yakovlev: Detecting State Coding Conflicts in STG Unfoldings Using SAT. Technical Report CS-TR-778, Department of Computing Science, University of Newcastle upon Tyne (2002)

[35] V.Khomenko and M.Koutny: Verification of Bounded Petri Nets Using Integer Programming. Formal Methods in System Design (submitted paper)

[36] J. Cortadella, J. Garside and A. Yakovlev: Logic Design of Asynchronous Circuits, ASPDAC/VLSI Design 2002, Bangalore, Jan. 2002.