Visualisation of Partial Order Models in VLSI Design Flow[*]

A. Bystrov, A. Yakovlev, M. Koutny

Dept. of Computing Science, University of Newcastle upon Tyne

E-mail: `{a.bystrov,alex.yakovlev,maciej.koutny}@ncl.ac.uk`

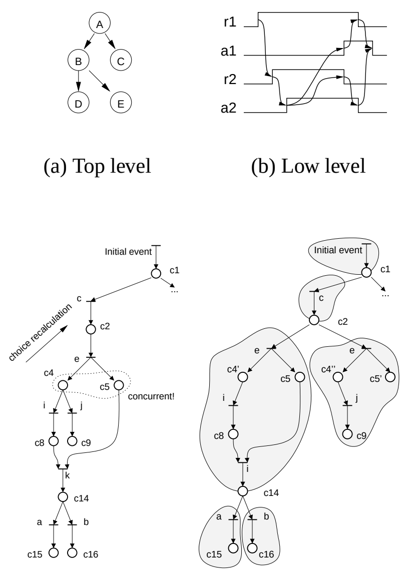
Summary. A new method, algorithms and tool for the visualisation of a finite complete prefix (FCP) of a Petri net (PN) or a signal transition graph are presented. A transformation is defined that converts such a prefix into a two-level model. At the top level, it has a finite state machine (FSM), describing modes of operation and transitions between them. At the low level, there are marked graphs, which can be drawn as waveforms, embedded into the top level nodes. The models of both levels are abstractions traditionally used by electronics engineers. The resultant model is completed trace equivalent to the original prefix. Moreover, the branching structure of the latter is preserved as much as possible.

Overview. The issues of power consumption, electromagnetic compatibility, clock skew, robustness and scalability draw attention to asynchronous designs. Unfortunately, the design effort required to implement an asynchronous circuit is high, partially due to the lack of efficient visualisation methods and tools facilitating model understanding. These are the main focus of this paper.

The model we use as system specification is an unfolding of 1-safe PN. An FCP [1], being a representative part of an unfolding, encapsulates three types of relations between nodes, viz. causality, concurrency and conflict. Whilst the models based on causality and concurrency, such as timing diagrams, or on causality and conflict, such as FSM, are well known to electronics designers, the raw FCP model might be too complex to manipulate directly. Usually, the designers would create a top-level model of the system, as an FSM defining the operational modes and interaction between them (see Figure 1(a)), and the low-level model would be a timing diagram, such as shown in Figure 1(b), against which the waveforms of signals are then compared.

The algorithm used in the tool 'unfPad' [2] converts an FCP into the desired form. For every choice condition concurrent to some node (e.g. e in Figure 1(c)) it: (a) finds the nearest non-concurrent event (NNE); (b) makes copies of the subgraph succeding the NNE; (c) restricts the choice in every subgraph to a single option, so that all options were covered. This algorithm supports free, controlled, arbitrated choice and confusion. The resultant model is completed trace equivalent to the original prefix. The branching structure of the latter is preserved as much as possible (cf. 'decorated trace semantics' in [3]).

[*]Project on model visualisation MOVIE, EPSRC grant GR/M94366



(a) Top level (b) Low level

(c) FCP fragment (d) Choice recalculation

Figure 1. Desired model representation

References

[1] J. Esparza, S. Romer, W. Vogler: 'An Improvement of McMillan's Unfolding Algorithm', LNCS 1055, Springer, 87-106.

[2] A.Bystrov, A.Yakovlev, M.Koutny: 'Visualisation of Partial Order Models in VLSI Design Flow', CS-TR-744, 2001, Univ. of Newcastle upon Tyne.

[3] van Glabeek, R.J.: 'The Linear Time – Branching Time Spectrum I; The Semantics of Concrete, Sequential Processes', Handbook of Process Algebra, Elsevier, 3–99.