Partial order approach to design, verification and synthesis of asynchronous circuits

Alex Semenov and Alex Yakovlev[1]

1 Introduction

Petri nets (PNs) and their related formalisms have been proven to be useful in modelling of concurrent systems. Asynchronous circuits and their specifications, with their inherent concurrency, have therefore been the subject of modelling by PNs among a substantial part of the asynchronous research community. One of the most popular formalisms is a *Signal Transition Graphs* (STGs) which is a type of Labelled PN (LPN) for representing low-level circuit behaviour. Each transition in an STG represents a transition of some signal in the future circuit. Several tools exist today for the verification and synthesis of asynchronous circuits from their STG specifications, e.g. SIS from Berkeley, Assassin from IMEC, Petrify and Versify from UPC.

Most of the existing tools build the Reachability Graph (RG) of the underlying PN. The RG fully represents the behaviour of the system expressed in the form of a PN. Thus the properties describing the behaviour can be derived from the RG. However, it is known that the RG approach suffers from the states explosion problem. The number of reachable states in the system grows exponentially in the number of transitions. Petrify and Versify attempt to avoid this problem by using Binary Decision Diagrams (BDDs) to represent implicitly the RG. This makes these tools more powerful than their counterparts using explicit RG generation. Nevertheless, these tools also choke on moderate-size examples.

Partial order methods have also been tried in analysis and synthesis of asynchronous circuits. Change Diagrams were used to analyse speed-independent circuits. Unfortunately, this formalism is restricted to choice free specifications. Recent research into the PN unfolding technique demonstrated that this method can be applied to verification and synthesis of speed-independent circuits as well as to verification of already existing circuits. The PN unfolding represents the RG implicitly, in the form of an acyclic PN, preserving concurrency between events which is the main cause of the state space blow-up. Therefore, it is often possible to construct a fragment of the PN unfolding representing the state space for those examples where RG construction fails. The STG unfolding, which takes into account binary values of the signals, represents the behaviour described by an STG.

Exponential explosion is also a problem for the conventional methods for the synthesis of speed-independent circuits. Since these methods and tools rely on deriving truth tables from the RG. However, if we can verify the STG specification using the unfolding technique, it would be natural to perform the circuit synthesis from the STG unfolding as well. Indeed, the STG unfolding segment already exists. Using STG unfolding segment it is possible to capture the behaviour of a particular signal change in a fragment of the segment. Thus it is possible to use structural information of the STG unfolding segment to derive the logic functions for non-input signals.

The purpose of this talk is to outline the use of PN/STG unfoldings in the design, synthesis and verification of asynchronous circuits. The application of the unfoldings in the design process

[1]The authors are with the Department of Computing Science, University of Newcastle

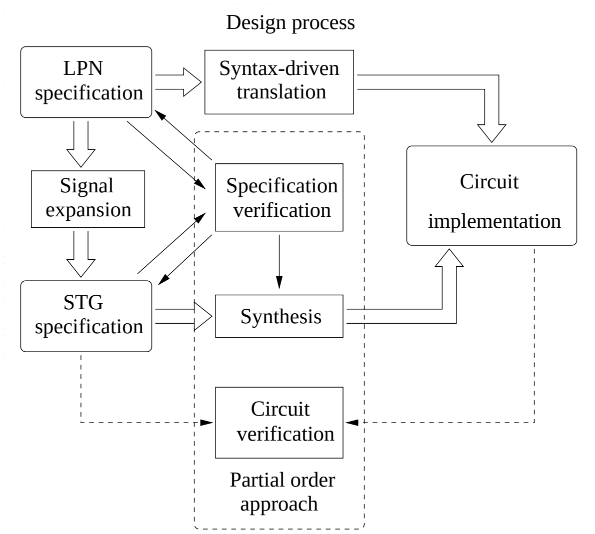

Figure 1: Partial order methods in design process.

is shown in Figure 1. It has been applied throughout the design process from the verification of specifications to synthesis and, in addition, to the verification of circuits designed in an "ad hoc" or semi-automated manner. The following sections review each of the stages separately.

2 Specification verification

The purpose of the specification verification is to determine if the PN specification can be implemented as an asynchronous circuit. The main properties that are checked are:

- *Boundedness* (safeness) and finiteness, which indicates that the behaviour can be implemented using a finite number of elements (gates);
- *Persistency*, which indicates that no signal can be non-deterministically disabled and thus manifest a hazard (spike);
- *Deadlock freedom*, which indicates that the system does not enter states from which no advancement can be made.

The first two properties are necessary conditions for a behaviour specified by a PN to be implementable as an asynchronous logic circuit. An initial specification can be made at a higher level than the circuit level used in STGs. From the high level the designer may decide to go directly towards the implementation using, for example, syntax directed translation. Examples of syntax-driven translation are CFPP and AMP processors designed at Newcastle as an exercise.

At the same time, the specification can be made at the signal level using STG. In addition, the designer may wish to reach the lower level by means of refinement. Such a refinement, introducing circuit level signal transitions, is called *signal expansion*. Thus the STG specification, whether initial or obtained through signal expansion, needs to be verified. The signal interpretation of transitions adds another necessary condition to the specification verification:

- *STG validity*, which indicates that the behaviour can be implemented as a circuit with two signal levels.

PNs do not assume any timing restrictions on their transition firings. However, sometimes it is possible to specify the future circuit together with timing constraints imposed on events. This can be done by in Time PNs where each transition is assigned with a delay range indicating the time in which it must fire from the moment of its enabling. Thus the verification of specifications can be divided into verification of timed and untimed specifications.

The verification method based on PN unfolding (timed or untimed) is exploits the fact that the finite fragment of the unfolding represents all reachable states of the specification along with all transitions enabled at each state. However, only a small subset of all states is visited during the construction of this finite segment, thus rendering this construction lower cost in time and space. The segment represents a partial run of the system built until the point where further advancement does not add any new information about the system's behaviour. In addition to the implicit state representation, the segment also provides temporal relations between occurrences of the events, called *instances*. Thus it is possible to check the correctness properties by examining the relations between instances.

Relations between elements of the unfolding prove useful in other applications. Methods based on BDDs are known to suffer from a variable ordering problem. Since BDD variables represent PN places, the temporal relations between places are used to create clusters of variables which, in turn help reduce the size of the BDD. The relations between transition instances help estimate the degree of concurrency in the design. The latter shows how many parts of the system may work concurrently on different parts of data. Therefore, the throughput of the system rises with the increase of the degree of concurrency[2].

3 Synthesis

Although acknowledged to be important, the use of methods alternative to RG construction has not gained much popularity among circuit designers. One of the main reasons is that to synthesise the implementation the full state space must be known. Thus even if the specification can be shown to be correct and implementable, the existing synthesis tools may still be unable to synthesise it. The synthesis process must produce a boolean function for each signal, called *cover*, implementing the signal. The covers are required to satisfy implementation correctness criteria according to the chosen architecture, in order to guarantee hazard-freedom at the gate level.

A recent research in structural synthesis methods has shown that it is possible to derive an approximated covers using the structural properties of the STG. This method is however restricted to free-choice specifications.

STG unfolding segment proves to be efficient in the verification of specifications. Furthermore, after the verification is complete, the segment represents all reachable states. Each instance in the STG unfolding segment represents a signal transition excited in a set of states. This set of states can be uniquely identified by the instances and captured in a fragment of the segment called *slice*. From the relations between the elements of the STG unfolding which belong to the slice it is possible to obtain an initial approximation of the boolean function which covers all the states captured in the slice. If the current implementation does not satisfy the correctness criteria, then it is refined using temporal relations extracted from the slices. The cover refinement is performed until the correct implementation is reached.

[2] This is, of course, a qualitative estimate; the numerical results should be sought through proper performance evaluation, e.g. using timed stochastic PNs.

The STG unfolding segment synthesis method has been shown to be comparable with the existing methods on moderate size examples; it becomes significantly better for the specifications of larger size (over 25 signals).

4 Circuit verification

There exists a number of examples where the asynchronous circuits were designed in an "ad hoc" or semi-automated way. Unfortunately, simulation cannot guarantee that all possible states of the circuit will be examined. Thus these circuits need to be verified for hazard free operation in a particular environment.

A PN based approach to circuit verification assumes that the circuit gates are represented as PN fragments. According to the protocol chosen, the PN representations are divided into an LPN representation, which is used in the modelling of a two-phase protocol, and a Circuit PN (CPN) representation, which is used to model four-phase circuits. In two-phased circuits, the up- and down-changes are symmetrical and one transition can be used to model the change (an event) of the signal. In four-phase circuits, on the other hand, the up- and down-change of a signal may have different excitation conditions and must be represented by different transitions in the PN model.

An LPN or CPN model of a circuit is then composed with the model of its environment. The composed model is then verified for absence of hazards using the LPN/STG unfolding. Hazards are detected either as unsafeness or nonpersistency of transitions representing the changes of the gate outputs. Similar to specification verification, it is sometimes convenient to assume that some transition have bounded time to fire. In this case, the transitions are assigned with a delay range and the whole system is verified using Time PN unfolding. Examples of verified circuits include AMULET micropipeline control circuitry.

When a two-phase circuit is modelled using LPNs, the places represent the wires in the circuit. Since transitions in LPN represents events, the tokens propagate in the model denoting the changes in voltage levels on the wires. Once an event has happened, the new conditions for this event can only be established when another token arrives into its input places. However, when a four-phase circuit is modelled, places represent the current states of the signals. Thus transitions can only move token in the places representing the output signal of a gate. The enabling conditions are established by connecting the input places and transitions with two-way arcs. The transition thus only "observes" the value of the gate's inputs and fires when the excitation conditions are reached.

The dependency between places and transitions connected by two-way arcs can be interpreted as the context in which a transition is allowed to fire. This was captured in the notion of *contextual nets*. If an ordinary net is used for modelling contextual dependencies, then the power of STG unfolding will make a distinction between an input place marked before and after a transition fired. This, in turn, may cause an exponential explosion in the size of the unfolding which would simply represent all possible firing of contextually dependent transitions. To avoid this, a *Contextual Net unfolding* was introduced. This unfolding uses the notion of *context independency* to construct new instances. Contextual Net unfolding was used to verify a real life examples such as a four slot communication mechanism, due to H. Simpson, where an ordinary unfolding failed due to its size. The experimental results have shown that the Contextual Net unfolding is very efficient in the verification of PN models with many contextual dependencies and high degree of concurrency.