Asynchronous system design flow based on Petri nets

Microelectronics System Design Research Group
University of Newcastle upon Tyne, UK
Web-page: *http://async.org.uk/besst/*

The BESST (BEhavioural Synthesis of Self-Timed Systems) tool kit is used for asynchronous system synthesis based on Petri Nets (PNs). It incorporates software tools for high-level partitioning, scheduling, direct mapping and logic synthesis. These can be used to generate efficient speed-independent circuits with optional security features. The conventional behavioural Verilog is used as an initial specification of the system. A diagram of the design flow is shown in Figure 1.

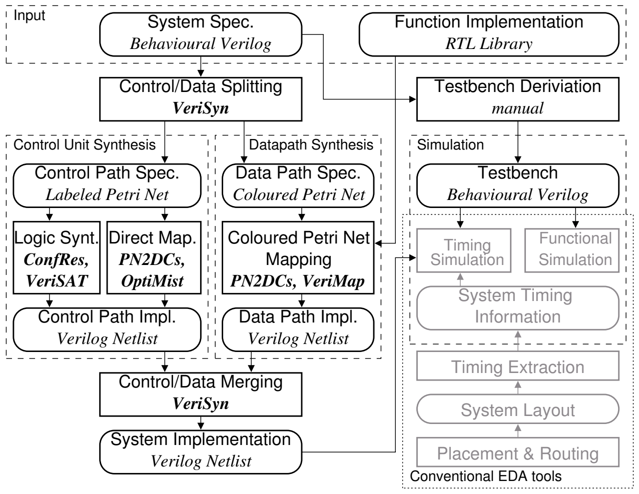

Figure 1. Design flow

The initial system specification is first partitioned into subsystems and then divided into control and data paths. Both paths are synthesised separately and merged into the system implementation netlist. Conventional EDA tools are used for place-and-route and simulation of the system.

The variations in the design flow appear in (a) extracting control and data paths; (b) synthesis of the control unit by either direct mapping or logic synthesis; (c) synthesis of the data path.

The step (a), control/data separation, is done by VERISYN, an asynchronous high-level synthesis tool, which operates as a front-end. The input for the tool is the behavioural Verilog specification. The VERISYN tool performs scheduling to produce a Labeled PN (LPN) for the control unit and a Coloured PN (CPN) for the data path. These nets are passed to the synthesis tools, which perform steps (b) and (c).

The synthesis of the control unit from an LPN can be done by either *direct mapping* or *logic synthesis*.

In the direct mapping approach the LPN is mapped into a David Cell (DC) netlist by either the PN2DCS or OPTIMIST tool. The DCs are used as state holding elements. PN2DCS uses an LPN, whereas OPTIMIST uses a Signal Transition Graph (STG), which is a special kind of LPN with transitions associated with rising and falling edges of signals. OPTIMIST aims at latency optimisation by using a *tracker-and-bouncer* architecture. The tracker computes the state of the system (context) concurrently to the environment operation. The bouncer produces the outputs based on the tracker's state as soon as the inputs are received from the environment.

In the logic synthesis approach boolean equations for the control path are derived using next-state functions, which are obtained from its STG. PETRIFY, a state-based synthesis tool, derives equations by exploring all possible orders of STG events, i.e. building its state space. Alternatively, VERISAT, an event-based synthesis and verification tool, can be used to derive equations from STG unfolding, thus avoiding the enumeration of states. This tool uses structural information from the STG unfolding and is based upon Incremental Boolean Satisfiability (SAT).

The important part of the logic synthesis is the *Complete State Coding* (CSC) problem. A CSC conflict arises when semantically different states of an STG have the same binary encoding. The CONFRES tool offers an interactive resolution of CSC conflicts based on core visualisation. This tool can be used with both PETRIFY and VERISAT to optimise solutions.

The data path is mapped directly from the CPN into a netlist of hardware components using the PN2DCS tool. These components are synthesised in the standard RTL-based design flow. Security features can be added by applying VERIMAP to the design. This is done by introducing dual-rail encoding of signals.