Logic Synthesis of Handshake Components Using Structural Clustering Techniques

Francisco Fernández-Nogueira and Josep Carmona*
Universitat Politècnica de Catalunya, Barcelona, Spain

Abstract. A methodology to optimize handshake circuits is presented. The approach selects clusters of a handshake network for which signals representing internal channels within a cluster are hidden. To guarantee asynchronous implementability on the resulting cluster, state encoding is applied using modern structural techniques. The theory of Petri nets is used to identify clusters for which the structural techniques perform successfully. Finally logic synthesis is employed for each reencoded cluster. The approach is integrated into the Balsa synthesis flow and represent a significant improvement with respect to the local optimizations typically applied. Experimental results in area and performance have been obtained to measure the optimization on typical Balsa examples.

1 Introduction

Asynchronous circuits represent a robust alternative for overcoming the problems of current and future technologies [1]. The nightmares of the synchronous paradigm like power dissipation, clock distribution, EMI, worst case performance among others are naturally avoided when one gets rid of the clock [2].

However, asynchronous circuits appear seldom in current technologies. The reason for this is simple: a circuit that lacks a global coordinator is difficult to design and verify. In the last decades, theories, methodologies and tools for the design and verification of asynchronous circuits have appeared, but their scope have been mostly academic. These asynchronous paradigms traditionally use as specification language formal models like automata or Petri Nets [3],[4],[5], which are not well-suited as front-end for the design of large and complex systems.

Hardware Description Languages (HDL) offer a simple way to design circuits. Many nuisances of the design process are hidden or automated, and allow the designer to have a system-level view of the circuit. The complexities of asynchronous circuit design can also be hidden by using an HDL as a front-end. With this idea in mind, the asynchronous community has provided some HDLs for the asynchronous design [6],[7]. Typically those programming environments transform the program, using a syntax-directed translation of each primitive, into a netlist of handshake components. Latterly each handshake component can be synthesized into an asynchronous circuit. Hence the size of the resulting

* Research funded by project FORMALISM (TIN2007-66523).
circuit is linear with respect to the size of the HDL program. This can limit the use of asynchronous HDLs when area and/or performance is a key factor.

Logic synthesis achieves global optimizations that can improve in orders of magnitude the local (peephole) optimizations applied in asynchronous HDLs [8,9]. In [10], a back-end to incorporate logic synthesis into the Balsa system was presented. The work showed the tangible improvements that can be obtained by optimizing the netlists of handshake circuits.

In this paper we provide a Petri net-based back-end to the Balsa system, offering resynthesis capabilities that include state encoding and logic synthesis of selected clusters of handshake components. The approach can be considered a follow-up of previous work [10,11,12,13], with the differences listed below:

1. State-based methods are used in [10,12,13], thus suffering from the state space explosion problem. Hence their application is limited to small specifications. In the work presented in this paper, modern structural methods for state encoding and synthesis [9,15] are employed, allowing large specifications to be handled.

2. Petri nets are used as intermediate language, whereas the underlying formalism in [10] for synthesis are burst-mode machines, that impose limitations on modeling the inherent concurrency of asynchronous systems.

3. A structural clustering approach guides the composition of handshake components, which are described by labeled Petri nets, into clusters. Those clusters grow as far as the induced Petri net composition of the selected components belongs to a class for which structural methods perform well. A blind clustering is used in the Petri net-based approaches [11,12,13], often deriving unrestricted clusters that synthesis methods can not handle.

4. No change in the specification language is required: the designer might benefit from the optimizations provided in this paper without even knowing that they are applied. This differs from the approach in [14], where a data-oriented Balsa language is presented to improve the performance of Balsa.

2 Signal Transition Graphs and Handshake Circuits

A Petri Net (PN) [16] is a 4-tuple, $N = (P, T, F, m_0)$, where $P$ is a finite set of places, $T$ is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation and $m_0 \in \mathbb{N}^{|P|}$ is the initial marking. Given a node $x \in P \cup T$, the set $\cdot x = \{y | (y, x) \in F\}$ is the preset of $x$ and the set $x \cdot = \{y | (x, y) \in F\}$ is the postset of $x$. A place such that $|p\cdot| > 1$ is called choice place.

Four special PN classes [16] are of interest in this paper. A PN $N$ is a: Marked graph (MG) if $\forall p \in P : |\cdot p| = |p\cdot| = 1$, State machine (SM) if $\forall t \in T : |\cdot t| = |t\cdot| = 1$, Free-choice (FC) if $\forall p_1, p_2 \in P : p_1 \cap p_2 \neq \emptyset \Rightarrow p_1 \subseteq p_2$ and Asymmetric Choice (AC) if $\forall p_1, p_2 \in P : p_1 \cap p_2 \neq \emptyset \Rightarrow p_1 \subseteq p_2$ or $p_1 \supseteq p_2$. Considering set inclusion as class inclusion, the following holds: MG, SM $\subset$ FC $\subset$ AC

To model digital circuits, the events of a PN can be interpreted as signal changes. A Signal Transition Graph (STG) is a triple $G = (N, \Sigma, \Lambda)$, where