On the Capabilities of Systolic Systems

Shimon Even and Ami Litman

Computer Science Department, Technion, Haifa, Israel 32000,
and
Bellcore, 445 South St., Morristown, NJ 07960-1910.

ABSTRACT

This paper studies transformations of systems into systolic systems with related functionality. It distinguishes two antithetical transformation methods: one syntactic, the other semantic.

The syntactic method considers the topology of the system, but ignores its behavior and the behavior of its combinational units. We introduce two basic syntactic techniques: tiling and bypassing. Based on them, we present syntactic transformations that perform the following: conversion of a semisystolic system to a systolic one; elimination of either broadcast or instant-accumulation from a system that is otherwise systolic; and speeding up a systolic system by any constant factor. Leiserson and Saxe [9] have developed transformations to accomplish the first two tasks, but failed to preserve the behavior of the system. Our transformations leave the behavior of the system intact.

The semantic method considers the functionality of the system as a whole, but ignores its internal structure. A system is called $\Psi$-homogeneous if all its combinational units are identical and equal the given unit $\Psi$. We show that every semisystolic system can be transformed into a $\Psi$-homogeneous systolic system, where $\Psi$ depends only on the alphabet used by the system to communicate with the external world. As a special case, any regular language $L \subseteq \Sigma^*$ is defined by some $\Psi$-homogeneous systolic system, where $\Psi$ depends only on $\Sigma$.

For binary systems, this technique produces a systolic system with a feasible clock period of $O(i+\log(o))$, where $i$ and $o$ are the numbers of input and output ports of the system. This clock period is independent of the size and complexity of the given system.

1. Introduction

Consider a large digital system built of combinational units and registers. All the registers are triggered by the same central clock and all the connections between the components are point-to-point (no buses). The combinational-units (comb-units in short) have the property that, at any given time, their outputs are determined by their inputs; that is, they have no memory. Although we usually ignore the propagation delays of the
comb-units, these delays determine the feasible period of the central clock. The registers are clocked memory elements. Assume the clock period is one, then for an integer \( \tau \) the output of a register during the time interval \( [\tau, \tau+1] \) is equal to its input at time \( \tau \).

Following Leiserson et. al., [9,10,11], the structure of the system is modeled by a directed weighted multigraph \( G = (V, E, H, w) \), called the communication graph (c-graph in short) which is defined as follows:

1. \( V \) is a set of vertices. A vertex represents a comb-unit.
2. \( E \) is a set of directed edges. An edge \( u \rightarrow v \) represents a connection from an output port of the comb-unit in \( u \) to an input port of the comb-unit in \( v \).
3. \( H \in V \) is the host. It represents the user of the system, and the edges incident to \( H \) are the only connections of the system to the outside world.
4. \( w \) is a function from \( E \) to the nonnegative integers. \( w(e) \) represents the number of registers on the edge \( e \).

The c-graph extracts only the communication aspect of the system. It models neither the functionality of the system nor the functionality of the comb-units; e.g., two systems of unrelated functionality may have identical c-graphs.

A system is called semisystolic if every cycle (directed circuit) of \( G \) has, at least, one register. This condition is essential for a proper operation of the system; it prevents possible instability as well as inconsistency. Henceforth, we will require that c-graphs have no register-less cycle. A semisystolic system may yet have long directed paths with no registers. In such a case, signals may ripple through long paths before they stabilize. This forces the slowing of the central clock to allow for such transitory effects to die down, rendering the system inefficient.

A semisystolic system is called systolic if every edge, except those emanating from the host, has at least one register. Systolic systems are attractive for several reasons. If we consider the registers to be an integral part of the vertex preceding them, then each vertex becomes a finite automaton (Moore’s finite state machine). Networks of finite automata are conceptually simpler than semisystolic systems, and have been popular models of parallel computation since the ’fifties [15]. A practical advantage of a systolic system is that if each comb-unit can operate at a certain rate of the central clock, then the system as a whole can operate at this rate, whatever its structure or size.

This paper presents transformations of systems (semisystolic, systolic and others) into systolic systems with related functionality. It distinguishes two antithetical transformation methods, one syntactic the other semantic.

1.1. Syntactic techniques

The syntactic techniques consider the topology of the system, but ignore its behavior and the behavior of its comb-units. The topology of the resulting system is strongly related to the original topology, and the new comb-units are built of copies of the old ones. Clearly, there is no interest in a transformation which glues the whole system into a single finite state machine; one prefers transformations which keep the comb-units small.

We extend the work of Leiserson et al. [9,10]. They introduced two basic syntactic techniques, namely retiming and slowdown, which modify only the number of registers.

March 1, 1994
on the system edges. We add two basic techniques, namely tiling and bypassing, which modify the topology of the system and its comb-units; bypassing even increases the total combinational logic of the system. All the syntactic transformations presented in this paper are based on these two techniques and retiming.

We will show that any semisystolic system can be syntactically transformed into an equivalent† systolic system. A similar result appears in [9], however, there is a significant functional difference between the end products. Our system has the same functionality as the original one, while theirs does not. The systolic system they produce is a \( k \)-\textit{slow} variant of the original one, for some integer \( k \). That is, it simulates a computation of the original system \( k \)-times slower, but it can simulate \( k \) such computations simultaneously. In one cycle the host communicates values of the first computation, the next cycle it communicates values of the second computation, etc. In many applications, the host has only a single task to compute, and it needs to provide immaterial values in the irrelevant cycles. In any case, the host must be modified to accommodate the idiosyncrasy of the new system. Although our transformation leaves the behavior of the system intact, our system does not necessarily work \( k \) times faster (in real time) than that of [9]; our comb-units may have longer propagation delay, thus a slower clock rate may be required.

There are two powerful computational mechanisms, broadcast and instant-accumulation, which seem at odds with the framework of systolic systems. Broadcast is a mechanism that makes signals, sent by the host over a single edge, immediately available to all other vertices. Instant-accumulation is a mechanism that instantaneously presents to the host, via a single edge, a value which is an “accumulation” of values produced, for this purpose, by all comb-units. The accumulation can be any operation that combines values by a commutative and associative operator, like AND.

Consider a systolic system augmented by an auxiliary mechanism of either broadcast or instant-accumulation. We will show that this auxiliary mechanism may be eliminated via a syntactic transformation. Sieferas [14] has proved that broadcast can be eliminated in the framework of iterative arrays, a special case of infinite systolic systems; and Leiserson and Saxe [9] have shown that any one of these two mechanisms can be eliminated in the framework of systolic systems, with a serious side effect. The resulting system is a 2-slow variant of the original.

A \( k \)-speedup of a given system is another system which computes in a single tick what the original system computes in \( k \) clock ticks. Cole [1] has proved a speedup theorem for iterative arrays. We will present a syntactic speedup transformation for systolic systems in general.

Leiserson and Saxe [9] have advocated the use of transformations to facilitate the design of systolic systems. They recommended starting from a simple semisystolic system of related functionality, and reaching the final systolic system via a sequence of transformations. All the syntactic transformations presented here are applicable in this context. These transformations expand the comb-units and sometimes expand the total combinational logic of the system. However, in many applications the expansion is only by a constant factor — it does not depend on the size of the system in question.

† The meaning of “equivalent” will be described later.