Speed without Fear: Composable Self-Timed GaAs Circuits

Erik Brunvand and Nick Michell

UUCS-93-028

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

Dec. 1993

Speed without Fear: Composable Self-Timed GaAs Circuits

E. Brunvand* N. Michell

Dept. of Computer Science, University of Utah
Salt Lake City, UT, 84112

Phone: (801) 581-4345
FAX: (801) 581-5843
Email: brunvand@cs.utah.edu

Area: Custom and Application-Specific Circuits

Abstract

We have designed a set of self-timed gallium arsenide building blocks that are suitable for composing into larger systems. Systems designed using these modules can be easier to design and modify than their globally clocked counterparts, while maintaining high performance. The modules are suitable for use in a translation systems that automatically translates an OCCAM program into a circuit netlist.

*This work is supported in part by DARPA award J.FBI.89.102, and in part by NSF award MIP-9115372

1 Introduction

High performance is of great importance in many circuit designs. Unfortunately, design effort seems to go up dramatically as the system speed increases. A great deal of effort goes into such tasks as clock distribution, timing analysis, and the placement of latches to balance pipeline delays. A mistake or an overlooked critical path may have effects worse than simply losing performance – the resulting circuit may not work at all. For complex systems the total effort may be measured in man-years. While such an expenditure of resourses is certainly worthwhile for high-volume circuits such as commercial microprocessors, for lower volume applications and research projects it is much harder to justify. In addition, such efforts must usually be repeated in whole or in part with each change to the design or to the process used to build it.

One way around this problem is to use *self-timed* circuit techniques that separate functionality from timing. In a self-timed system the individual system parts are designed to obey a simple communication protocol which is insensitive to the delays in the circuits or the wires that connect them. Protocols with this property are called *delay-insensitive* [7] and allow the designer to compose parts based only on their functionality, not on their timing properties.

2 Circuit Modules

Although self-timed circuits can be designed in a variety of ways, the circuits considered here predominantly use two-phase transition signalling for control and a bundled protocol for data paths. Two-phase transition signalling uses transitions on signal wires to communicate. Only the transitions are meaningful; a transition from low to high has the same effect as a transition from high to low and the particular voltage on each wire is not important. The communication protocol uses two wires: A wire called *Req* to request service, and a wire called *Ack* to acknowledge completion of that service.

A bundled data path is a compromise to complete self-timing that uses transition control wires and a set, or *bundle* of conventional data wires. The *bundling constraint* requires that the data bundle and the control wires be constructed such that the value on the data bundle is stable at the receiver before a signal appears on the control wire. This condition is similar to, but weaker than, the equipotential constraint [5]. Bundled datapaths allow the designer to use familiar circuit elements and decreases the area penalty of self-timed designs, but it adds timing constaints. However, the places where timing must be checked are few in number and limited in scope, so we believe the tradeoff is worthwhile.

The circuit modules used here are based on those described in more detail elsewhere [1, 6]. They include the following circuits:

Merge The "OR" function for transitions, implemented by an XOR gate.

Join The "AND" function for transitions, implemented by a C-element.

Call A module that acts as a hardware subroutine call allowing multiple access to a shared resource. The Call module routes the *Req* signal from a client to the subroutine, and after the subroutine acknowledges, routes the *Ack* back to the appropriate client. The requests must be mutually exclusive.

Select A module that steers an input transition to one of two outputs based on the value of a Boolean *select* signal. The *select* signal is a bundled signal with respect to the input transition.

Arbiter A module that prevents two signals from being asserted simultaneously. A transition on one request input is propagated to the grant output but a subsequent transition on the other request line will be delayed until after a done transition is received.

Latch A module that latches bundled data signals upon receipt of transition control signals.

Carry Completion Adder A form of adder that reports when the addition is complete by sensing when the carry chain is complete.

The control circuits that are built from these modules can be thought of as *distributed control state machines*. To understand how control is passed around the system, it may be helpful to imagine that the signal transitions are physical tokens passed along the wires of the circuit. The current state can be thought of as a set of tokens corresponding to request and acknowledge signals in the control circuit. This is similar to viewing the control circuit as a Petri net.

3 GaAs Module Library

The module library has been implemented in GaAs as a set of cells for use with the PPL tools developed at the University of Utah [3]. PPL cells are similar to standard cells, with the difference that cells are placed on a grid and may connect on all four sides. A physically-based design tool, called ACME, is used to capture the schematic and create the physical layout at the same time. The modules are designed using direct-coupled FET logic (DCFL) circuits and fabricated using the Vitesse HGaAsIII process through the MOSIS chip brokerage service.

The DCFL style of GaAs circuits encourages the use of NOR gates as circuit building blocks. For example, the DCFL version of a C-element is built from a two-level network of NOR gates which implement the function $A \cdot B + A \cdot Out + B \cdot Out$. Other self-timed control and data modules are similarly implemented using predominantly NOR-style circuits.

The modules vary in size from a few gates to several dozen gates. Most of them are hierarchical in nature. For example, the Call module described above, is implemented from Merge and C-element cells, together with some "glue logic". This is shown in Figure 1. The two cells at the bottom of the figure are C-elements, and the other complex cells are Merge cells. One special circuit is the mutual exclusion element, shown in Figure 2, which is the key component of the arbiter [4]. A two-phase arbiter using the mutual exclusion element is shown in Figure 3.

We have fabricated and tested an unoptimized version of the library that did not vary the transistor sizes. Delay values for the Merge, C, Select, and Call modules are shown in Table 1. The test chip also contained a four-deep FIFO which has not yet been evaluated. Based on the results of this test chip we are in the process of redesigning the library cells to optimize transister sizes and reduce stray capacitance.

In addition to designing circuits by hand, we have also generated netlists of these circuit modules automatically. A routing circuit was designed by writing a program in a version of OCCAM, a concurrent programming language useful for describing collections of small concurrent processes that cooperate through communication. This program description, was translated automatically to a circuit description using a technique described elsewhere [1]. This translation process results in a netlist of circuit modules from the set of modules described earlier. This netlist was implemented in several technologies: CMOS standard cells, Actel FPGA macros, and an earlier GaAs process from Vitesse [2]. We plan to use this router circuit as a test of the new optimized HGaAsIII cell set. Currently, the netlist for GaAs must be entered by into our design capture system.

4 Conclusions

Many of the advantages of self-timed circuits over more traditional synchronous circuits stem from the separation of timing from functionality found in self-timed designs. This separation does not mean that timing is ignored; on the contrary, the designer if free to perform timing optimizations without the fear that the circuit will fail to perform correctly.

We have implemented compatible module libraries in CMOS standard cells and Actel FPGA macros, as well as in GaAs, and have been able to automatically generate netlists for all three technologies from a high-level description. The self-timed nature of the circuits can allow incremental replacement and mixing of parts that operate at different speeds with no change to the circuit or retiming of the system.

References

[1] Erik Brunvand. *Translating Concurrent Communicating Programs into Asynchronous Circuits.* PhD thesis, Carnegie Mellon University, 1991.

[2] N. Michell E. Brunvand and K. Smith. A Comparison of Self-Timed Design using FPGA, CMOS, and GaAs Technologies. In *ICCD*, Oct. 1992.

[3] Jun Gu and Kent F. Smith. A structured approach for VLSI circuit design. *Computer*, November 1989.

[4] E. Brunvand N. Michell and K. Smith. A Gallium Arsenide Mutual Exclusion Element. Submitted to JSSC.

[5] C. L. Seitz. System timing. In *Mead and Conway, Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980.

[6] Ivan Sutherland. Micropipelines. *CACM*, 32(6), 1989.

[7] Jan Tijmen Udding. A formal model for defining and classifying delay-insensitive circuits and systems. *Distributed Computing*, 1:197–204, 1986.

Module	Measured Delay	Est. Optimized Delay
Merge	336pS	250pS
C	344pS	240pS
Select	1.19nS	0.9nS
Call	1.77nS	1.2nS

Table 1: Module Timing Information

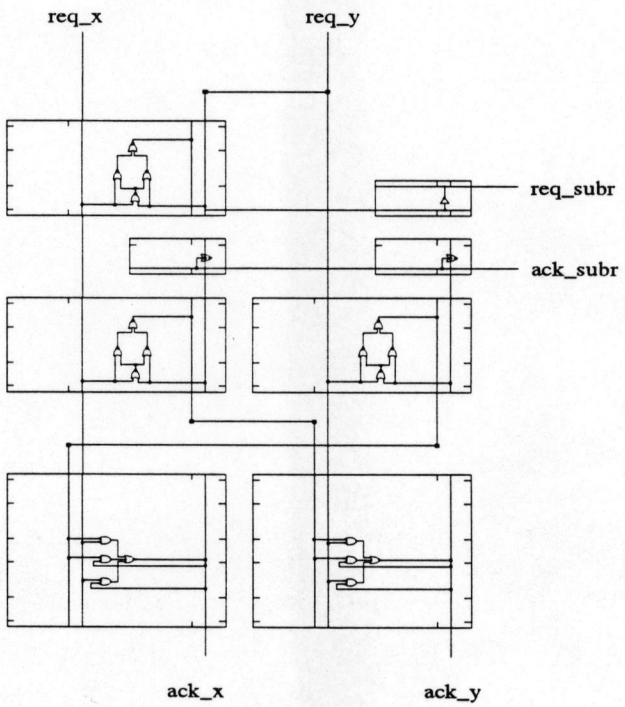

Figure 1: Call Element

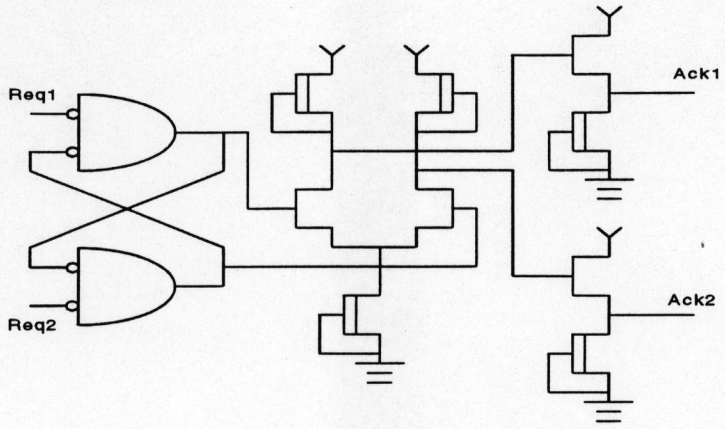

Figure 2: GaAs Mutual Exclusion Element

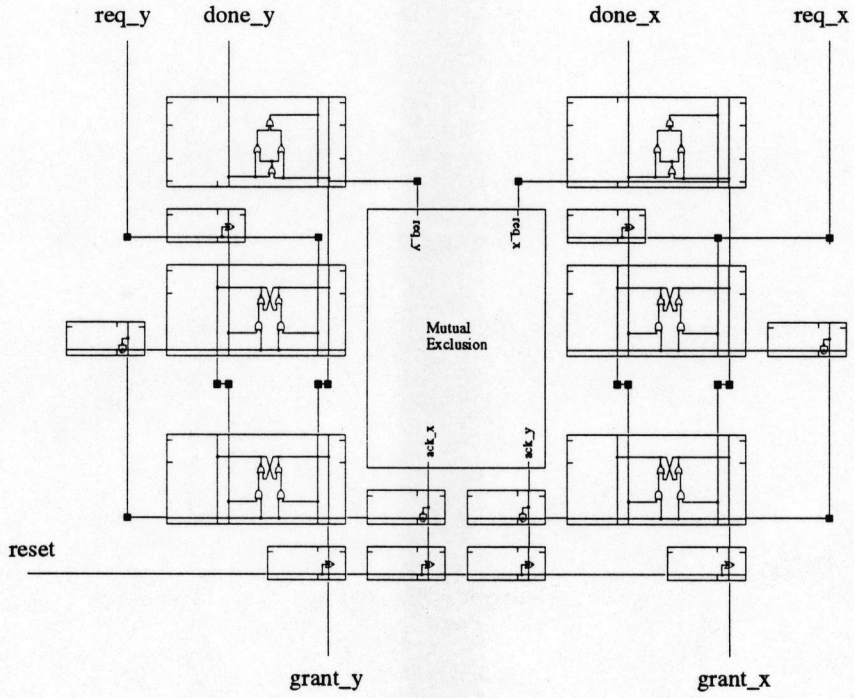

Figure 3: Two-Phase Arbiter Circuit