A Low-Control-Overhead Asynchronous Differential Equation Solver*

KennethY.Yun† PeterA.Beerel‡ VidaVakilotojar‡ AyoobE.Dooply† JulioArceo†
† Dept. of ECE, University of California, San Diego
‡ EE-Systems Dept., University of Southern California

Abstract

This paper describes the design of a high-performance asynchronous differential equation solver, a common DSP application. The high performance is achieved by two means. First, efficient self-timed datapath elements were designed, including a self-timed carry-bypass adder with low-overhead domino completion-detection and a staggered-evaluation precharged multiplier using carry-save-addition. Second, asynchronous control circuit overhead was minimized to 12% by using an efficient 3D control circuit design style and timing assumptions to effectively hide control circuit delay. The design has been simulated in a 1.2μm two-metal HP SCMOS process and compared to comparable synchronous designs. Its average-case performance (simulated at 25°C and 5V) is estimated to be 37% faster than the synchronous worst-case performance (simulated at 100°C and 4.5V).

1 Introduction

Because of varying data and environmental conditions, asynchronous circuits have the potential for an average-case delay that is smaller than the worst-case delay of synchronous counterparts. To achieve this potential, however, the asynchronous completion detection and control circuitry must have minimal performance overhead [5]. Williams demonstrated essentially zero overhead in a 54-bit self-timed divider that has relatively simple control circuitry [8]. This paper demonstrates that low overhead and superior average-case delay can be achieved for a differential equation solver which requires more complex control. Moreover, although low-power was not the principal goal, a side benefit of the design's asynchronous nature is that it works with reduced supply voltages and performs optimal gated clocking.

The paper describes the high-level algorithmic specification and chip architecture (Section 2), presents the adder and multiplier designs contained in the datapath (Section 3), describes the design of the control circuitry (Section 4), and concludes with simulation results and comparisons (Section 5).

2 Algorithm and Architecture

A high-level description of the differential equation solver algorithm is presented below [2].

```
diffeq {
    read ( x, y, u, dx, a );
    while ( x < a ) {
       xl := x + dx;  ul := u - 3 * dx * (u * x + y);  yl := y + u * dx;
       x := xl;  u := ul;  y := yl;
    }
    write ( y );
}
```

*Supported in part by a gift from Intel Corporation.

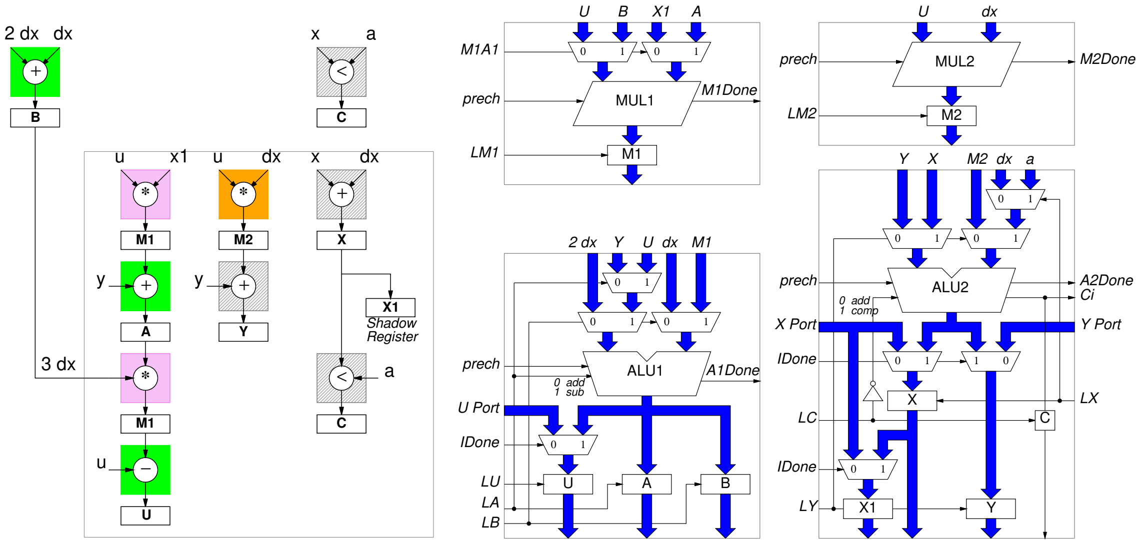

Figure 1: (a) Dataflow graph and (b) Datapath.

The algorithm implements the *forward Euler method* to numerically obtain the values of y that satisfy the differential equation $y'' + 3xy' + 3y = 0$ as x ranges from $x(0)$ to a with a step size of dx and with initial values of y and y' equal to $y(0)$ and u, respectively.

Using *operator reduction* [2] to convert multiplication by 3 to a shift-and-add operation and *code motion* [2] to pull the shift-and-add operation out of the loop, the data-flow graph depicted in figure 1a is obtained. We use two ALUs and two multipliers to perform the four subtraction/additions, one comparison, and three multiplications needed for each loop iteration. The shading of operations in figure 1a depicts the *binding* of operations to the two ALUs and two multipliers. The resulting datapath is shown in figure 1b.

3 Datapath Elements and Control Circuits

This section describes the ALU and multiplier designs implemented in a 1.2μm HP SCMOS two-metal process.

A strong justification for self-timed addition is found in Von Neumann's early paper collected in [1], which proved that the average longest carry chain length is $\log_2 n$ for n-bit addition for random input statistics. However, in many applications [4, 5], input statistics are not as favorable for self-timed addition. Thus our adder design uses both a worst-case delay-reduction mechanism (carry bypass) and an average-case delay reduction feature (completion detection). Although the completion detection overhead is as much as 1.6ns when there is no carry propagation, it is only 0.2ns when the computation takes the worst-case delay; i.e., when the carry propagates through the entire carry chain. As a result, the actual time from the beginning of evaluation to the completion detection is never longer than 4ns (0.2ns higher than the worst-case delay) and usually is much shorter that the worst-case delay. Our preliminary results indicate that the average-case delay is 2ns for random inputs. This appears to outperform other proposed self-timed adders [3, 4].

The 16-bit adder is divided into four sections. Each section consists of precharged carry-generate, carry-kill and carry-propagate circuit bit slices, a Manchester carry chain, and static sum circuit bit slices (XOR of carry-propagate and carry in). In order to speed up the worst-case addition, carry bypass circuits are inserted between successive four-bit sections. Furthermore, the Manchester carry chain consists of two rails as shown in figure 2a; one chain propagates the carry and the other the complement of the carry. Completion detection is accomplished by a domino logic implementation of the OR-AND circuit as shown in figure 2b. In order to minimize the completion detection overhead, late arriving signals are placed closer to the output.

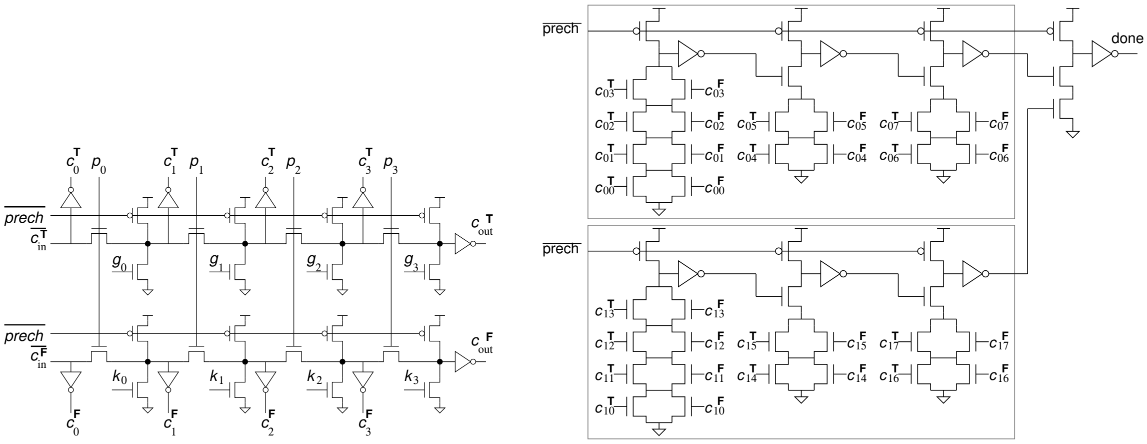

Figure 2: (a) Dual-rail Manchester carry chain and (b) Completion detection domino logic.

The 16-bit multiplier consists of 16 stages of carry save adders. Because only the lower 16 bits of the 32 bit products are used, there is no need for a carry propagate adder to generate the upper 16-bits and no need for the lower half of the CSA array (as depicted in figure 3a).

In many applications, multipliers are implemented in a pipelined fashion [7] in order to optimize the throughput. In this application, however, the latency must be optimized, not the throughput. We use a novel implementation technique called *staggered evaluation* (see figure 3b) to speed up the multiplication latency. In this scheme, the evaluation of each stage is delayed by just enough time for the predecessor stage to complete.

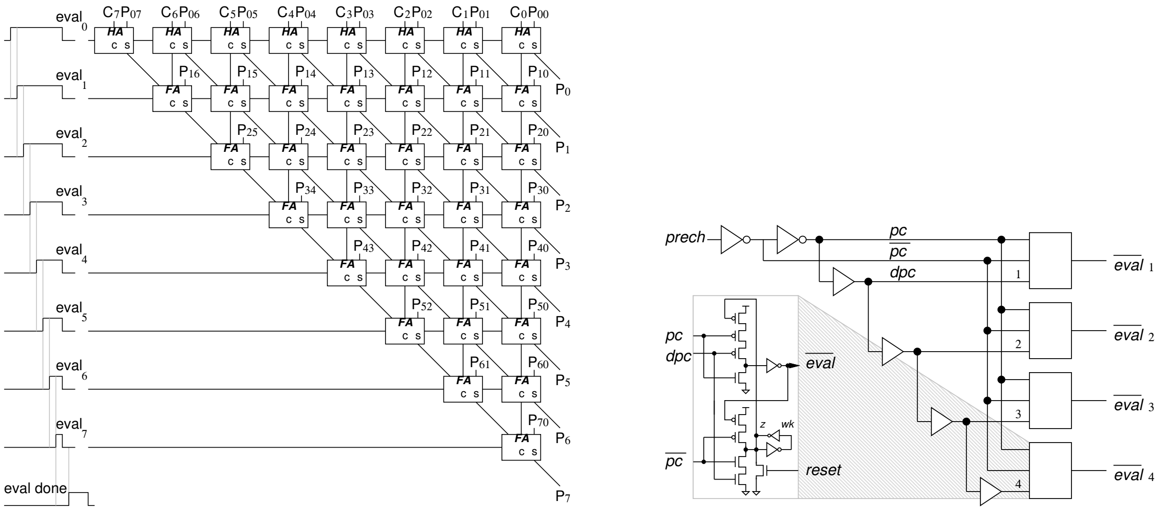

Figure 3: (a) Staggered-evaluation precharged array multiplier built with CSAs (only 8 stages drawn) and (b) Evaluation signal generation circuit.

Four control units have been designed, one for each datapath unit. The control units are responsible for controlling the sequencing between operations, the datapath precharge signals, the ALUs' function-select signals, the MUX selects, and the flip-flops' "clock input". The controller specifications have been automatically translated into the language of the formal verification tool SMV [6] and their communication has been verified to be deadlock-free and to adhere to the specified asynchronous protocol. The control circuits have been designed using a *3D design style*, and implemented with generalized C-elements [9] that have low latency. The control circuitry overhead was further minimized by using timing assumptions that effectively hide it in the delay of the datapath. These timing assumptions were validated using Mentor Graphics Accusim (SPICE).

4 Results

We have completed the design and simulation of the differential equation solver. For a proof of concept, a key component of the differential equation solver, the 16-bit self-timed adder, was implemented on a separate die in 1.2μm HP SCMOS two-metal process and taped out for fabrication. At the time of paper submission, all the cell designs, including adders and multipliers, are complete; it is anticipated that the complete design will be ready for tapeout shortly.

The simulations were performed for two operating conditions and for various data-dependent delays (see table 1). We used 25°C and 5V for typical case and 100°C and 4.5V for worst case. The total delays through the MUXes and registers under the typical operating condition are shown in the column with the heading *Glue*; two multiplier delays of the critical path (MUL1 $\rightarrow$ ALU1 $\rightarrow$ MUL1 $\rightarrow$ ALU1) in the column *Mul*; two adder delays of the critical path in the column *Add*. The average delay per iteration of the algorithm under the typical operating condition is 26.5ns. The control overhead is 3.2ns, which is 12%. On the other hand, the average delay per iteration under the worst-case condition is 40.4ns; the worst-case over all the data-dependent delays we simulated is 41.2ns; the worst-case delay minus the asynchronous control overhead is 36.5ns. This means that synchronous versions of this differential equation solver cannot be made to operate faster than 36.5ns per iteration (assuming no control overhead or clock skew) — 37% slower than the average-case delay of our asynchronous version.

	typical case						**worst case**	
#	Glue	Mul	Add	Sum	Actual	Overhead	Actual	Overhead
1	6.4	13.4	3.5	23.2	26.2	3.0	41.2	6.0
2			3.3	23.0	26.7	3.8	40.9	5.7
3			4.2	23.9	26.9	3.0	40.4	3.9
4			3.6	23.3	26.4	3.1	40.0	4.4
5			3.5	23.2	26.2	3.0	39.5	4.0
Avg	6.4	13.4	3.8	23.5	26.5	3.2	40.4	4.8

Table 1: Experimental Results.

References

[1] B. E. Briley. Some new results on average worst carry. *IEEE Transactions on Computers*, C-22(5):459–463, 1973.

[2] G. De Micheli. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, New York, New York, 1994.

[3] Hema Dhanesha and Alexander Albicki. Self-timed adder with pipelined output. In *Proceedings of the Midwest Symposium on Circuits and Systems*, pages 855–858, 1993.

[4] Jim D. Garside. A CMOS VLSI implementation of an asynchronous ALU. In S. Furber and M. Edwards, editors, *Asynchronous Design Methodologies*, volume A-28 of *IFIP Transactions*, pages 181–207. Elsevier Science Publishers, 1993.

[5] D. J. Kinnement. An evaluation of asynchronous addition. *IEEE Transactions on VLSI Systems*, 4(1):137–140, March 1996.

[6] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[7] J. Sparsø, C. D. Nielsen, L. S. Nielsen, and J. Staunstrup. Design of self-timed multipliers: A comparison. In S. Furber and M. Edwards, editors, *Asynchronous Design Methodologies*, volume A-28 of *IFIP Transactions*, pages 165–179. Elsevier Science Publishers, 1993.

[8] Ted E. Williams and Mark A. Horowitz. A zero-overhead self-timed 160ns 54b CMOS divider. *IEEE Journal of Solid-State Circuits*, 26(11):1651–1661, November 1991.

[9] Kenneth Y. Yun. Automatic synthesis of extended burst-mode circuits using generalized C-elements. In *EURO-DAC'96*, 1996.