Journal of VLSI Signal Processing, 1, 345–365 (1990)
© 1990 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.
Vol. 1, #4, 4/90

A Clock-Free Chip Set for High-Sampling Rate Adaptive Filters

TERESA H.-Y. MENG*
Department of Electrical Engineering, Stanford University, Stanford, California 94305.

ROBERT W. BRODERSEN AND DAVID G. MESSERSCHMITT
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720.

Received May 8, 1989. Revised October 24, 1989.

Abstract. As digital signal processing systems become larger and clock rates increase, the typical design approach using global clock synchronization will become increasingly difficult. The application of asynchronous clock-free designs to high-performance digital signal processing systems is one promising approach to alleviating this problem. To demonstrate this approach for a typical signal processing task, the system architecture and circuit design of a chip set for implementing high-rate adaptive lattice filters using the asynchronous design techniques is presented.

1. Introduction

The issues in designing computing machines, both in hardware and in software, have always shifted in response to the evolution in technology. VLSI promises great processing power at low-cost, but there are also new constraints that potentially prevent us from taking advantage of technology advances. The increase in processing power available is a direct consequence of scaling the digital IC process. As the scaling of the IC process continues, it is doubtful that the benefits of the faster devices can be fully exploited due to other fundamental limitations. System clock speeds are starting to lag behind logic speeds in recent chip designs. While gate delays are well below 1 *ns* in advanced CMOS technologies, clock rates of more than 50 MHz are difficult to obtain and where they have been attained require extensive design and simulation effort.

One important design consideration that limits clock speeds is *clock skew* [1], which is the difference in phase of a global synchronization signal observed at different locations in the system. Clock skew can be reduced to a minimum with proper clock distribution [2], [3] but because of this global constraint high-performance circuitry is confined to a small chip area.

The alternative fully asynchronous design synthesis approach eliminates the need for a global clock and circumvents problems associated with clock skew. At the chip level, since there are no global timing considerations, the design time spent on layout and circuit timing simulation for the worst-case design are greatly reduced. At the board level, since the asynchronous approach provides design modularity, systems built using this approach can be easily extended without problems in global synchronization [4], [5].

Several recent dedicated hardware designs have achieved high-speed performance by eliminating clock skew problems through asynchronous designs [6], [7] and globally-asynchronous locally-synchronous systems [8]–[10] have been considered. Our work is highly motivated by feasibility and implementation. We are particularly concerned with performance, since we believe that the usefulness of a system will be eventually judged by its cost-effectiveness. Hence, we put special emphasis on exploiting maximum concurrency at both the architecture and the circuit levels, so that high-performance implementation can be obtained without dependence on advanced technology.

Motivations for adopting an asynchronous design methodology can be summarized as follows.

Scaling and Technological Reasons. The main motivation for going to asynchronous circuitry is to eliminate the requirement for a global clock. Clock skew is a direct result of the RC time constants of the wires and the capacitive loading on the clock lines. While scaling the IC process reduces the area requirements for a given circuit, more circuitry is usually placed on the chip to take advantage of the extra space and to further reduce system costs [11]. Hence, for a global signal such as a clock, the capacitive load tends to stay constant with scaling. Therefore, the capacitance associated with the interconnect layers cannot scale below a lower limit.

*This research was sponsored in part by the Semiconductor Research Corporation and by DARPA.

As the devices get smaller, the clock load represents a larger proportional load to the scaling factor and more buffering is needed to drive it, complicating the clock skew problem. Calculations show that it would take roughly 1 *nsec* for a very large transistor ($1/gm = 50\Omega$) to charge a capacitance of 10 *pf* in advanced CMOS technology [12]. This delay constitutes a constant speed limitation if a global signal is to be transmitted through the whole chip, not to mention the whole board.

CAD Tools and Layout Factors. Increased circuit complexity has been addressed by the development of CAD tools to aid the designer. These tools often allow the designer to specify the functionality of a system at a structural level and have a computer program generate the actual chip layout according to the interconnection specified. The routing of clock wires as well as the load on the clock signal are global considerations which are difficult to extract from local connectivity. From a system design point of view, asynchrony means design modularity. A modular design approach greatly simplifies the design efforts of complicated systems and fits well with the macro-cell design methodology often adopted by ASIC CAD tools.

Board Level Design. One of the primary advantages of using fully asynchronous design in implementing high-performance DSP systems is the ease of design at the board level, where clock distribution is not an issue. Inter-board communications have long used asynchronous links (for example, the VME bus protocol) and interchip intra-board communications are becoming asynchronous too (for example, the MC68000 series). As DSP systems become complex, it is advantageous to simplify the design task to only local considerations by using asynchronous components. This is particularly important for signal processing applications using pipelined architectures, where computation can be extended by pipelining without any degradation in overall system throughput.

One goal of this work is to demonstrate the ease with which asynchronous systems can be designed with a minimum amount of design efforts. We have thus designed a chip set that realizes a vectorized adaptive lattice filter, with arbitrary vector size, using an asynchronous design methodology. Our adaptive filter architecture has localized forward-only interconnections, full pipelining in which the communication paths between chips constitute pipeline stages, and asynchronous interconnection eliminating the global constraint of clock distribution. Hence our architecture can achieve arbitrary throughput consistent with input/output rate limitations (such as the speed of A/D converters). The experimental implementation showed that asynchronous design simplifies the design process since no effort was devoted to clock distribution, nor were any timing simulations to verify clock skewing necessary.

Section 2 gives a short description of the vectorized adaptive lattice filter structure and the chip partition of the adaptation algorithm. Section 3 reviews the asynchronous design methodology used in the chip implementation. Section 4 illustrates the design of a fast self-timed array multiplier as an example of designing computation blocks with completion signal generation. Section 5 describes the design of the chip set for an adaptive lattice filter along with chip performance evaluations, and Section 6 gives the conclusions.

2. Vectorized Adaptive Filters

The realization of adaptive filters at high sampling rates is important in many applications but made inherently difficult by the recursive nature of the algorithms. The block adaptive filter scheme was proposed in which filters adapt coefficients at time T using the coefficients at time $T-L$, where L is the block size. In applications that require both fast convergence and fast tracking, introducing delays in the adaptation algorithm will degrade the filter tracking capability and destabilize the filter dynamics, such as in some radar signal processing systems.

The vectorized adaptive filter scheme introduced in [13] achieves two objectives simultaneously. The first objective is to allow arbitrarily high-sampling rates for a given speed of hardware, at the expense of parallel hardware. The second objective is to not modify the input-output characteristics of the algorithm, and hence not affect the convergence and tracking capability. We chose to use a lattice filter [14] in our design because the adaptation feedback in a lattice filter can be limited to a single stage. This results in much less computation within the feedback path and higher inherent speed in a given technology. The successive stages of a lattice filter are independent and therefore, can be pipelined.

2.1. The Adaptation Algorithm

Linear adaptive filtering is a special case of a first-order linear dynamical system. The adaptation of a single stage of a lattice filter structure can be described by two equations, the state-update equation

$$k(T) = a(T)k(T-1) + b(T), \qquad (1)$$

and the output computation

$$y(T) = f(k(T), T), \qquad (2)$$

where $a(T)$ and $b(T)$ are read-out (memoryless) functions of the input signals at time T, and $y(T)$ is a read-out function of the state $k(T)$ and the input signals. Since the computation of $y(T)$ is memoryless, the calculation can be pipeline-interleaved [15] and the computation throughput can be increased without theoretical limit, if input signals and state information can be provided at a sufficiently high speed. However, the state-update represents a recursive calculation which results in an iteration bound on the system throughput [16]. In order to relax the iteration bound without changing the system's dynamical properties, Equation (1) can be written as

$$\begin{aligned} k(T+L-1) &= a(T+L-1)a(T+L-2)\cdots a(T+1)a(T)k(T-1) \\ &\quad + a(T+L-1)a(T+L-2)\cdots a(T+1)b(T) + \cdots \\ &\quad + a(T+L-1)b(T+L-2) + b(T+L-1) \\ &= c(T,L)k(T-1) + d(T,L) \qquad (3) \end{aligned}$$

where $c(T, L)$ and $d(T, L)$ are memoryless functions of input signals, but independent of the state $k(T)$. $c(T, L)$ and $d(T, L)$ can be calculated with high-throughput using pipelining and parallelism, and the recursion $k(T + L - 1) = c(T, L)k(T - 1) + d(T, L)$ need only be computed every L samples. Therefore, the iteration bound is increased by a factor of L. L is called the vector size, since a vector of L samples will be processed simultaneously.

2.2. *The Chip Partition for the Normalized LMS Lattice Filter*

A normalized least mean squares (LMS) or stochastic gradient adaptation algorithm is chosen for our example because of its simplicity and the absence of numerical overflow. A vectorized LMS lattice filter stage with $L = 3$ is shown in figure 1, and the operations that each processor needs to perform are shown in figure 2. The derivation of the algorithm for normalized LMS lattice filters can be found in [17]. Processors A_1 and A_2 jointly calculate the $c(T, L)$ and $d(T, L)$ in Equation (3), processor B_1 and B_2 calculate the state-update for state k, and processor C_1 performs the output computation. For every processing cycle, a

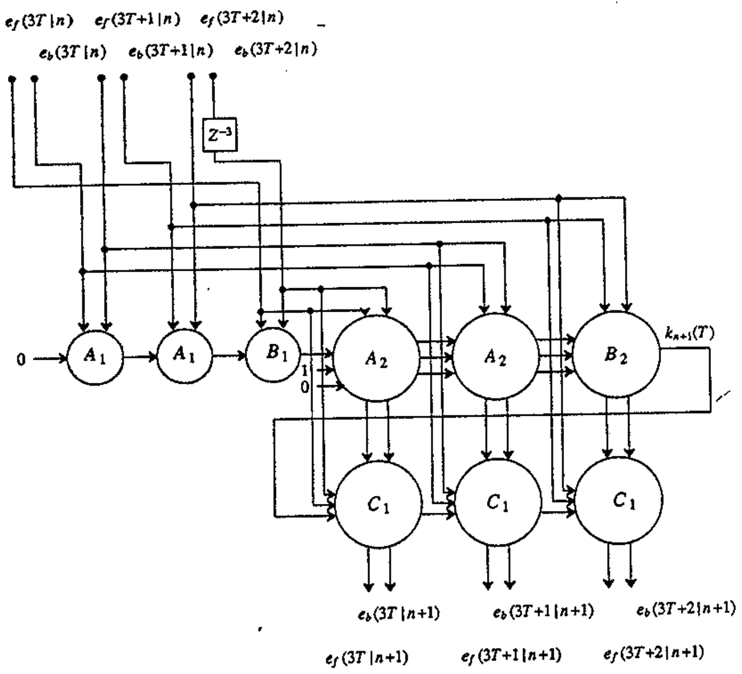

Fig. 1. An LMS adaptive lattice filter stage with a vector size of three.

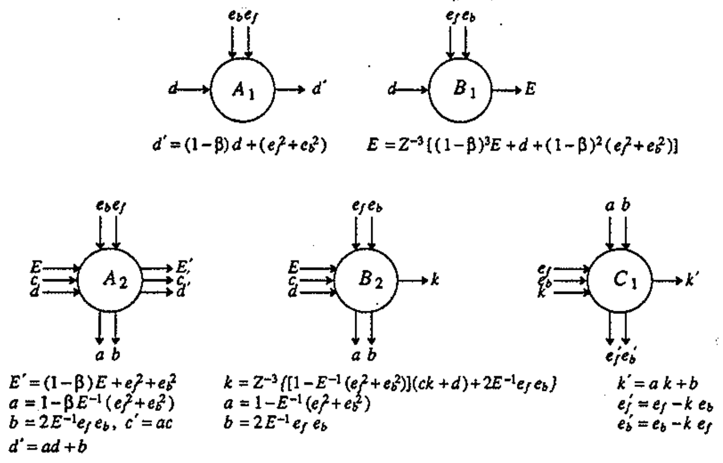
$e_b e_f$
d
A_1
d'
$e_f e_b$
d
B_1
E
$d' = (1-\beta)d + (e_f^2 + e_b^2)$
$E = Z^{-3}\{(1-\beta)^3 E + d + (1-\beta)^2(e_f^2 + e_b^2)\}$
$e_b e_f$
E c d
A_2
E' c' d'
a b
$e_f e_b$
E c d
B_2
k
a b
a b
e_f e_b k
C_1
k'
$e_f' e_b'$
$E' = (1-\beta)E + e_f^2 + e_b^2$
$a = 1 - \beta E^{-1}(e_f^2 + e_b^2)$
$b = 2E^{-1} e_f e_b, \; c' = ac$
$d' = ad + b$
$k = Z^{-3}\{[1 - E^{-1}(e_f^2 + e_b^2)](ck + d) + 2E^{-1} e_f e_b\}$
$a = 1 - E^{-1}(e_f^2 + e_b^2)$
$b = 2E^{-1} e_f e_b$
$k' = a k + b$
$e_f' = e_f - k e_b$
$e_b' = e_b - k e_f$

Fig. 2. Operations required of each processor shown in figure 1.

vector of L input samples are processed and a vector of L output samples are calculated. The processing speed is L times slower than the filter sampling rate. Since there is no theoretical limit to the number of samples allowed in a vector, the filter sampling rate is not constrained by the processing speed, but rather it is limited by only the I/O bandwidth (such as the speed of data converters). However, the higher sampling rate is achieved at the expenses of additional hardware complexity and system latency, and very high-sampling rates will require multiple-chip realizations. For example, at the sampling rate of 100 MHz the computational requirement is approximately 3 billion operations per second per stage. Since there is no global clock routing at the board level with asynchronous design, pipelined computation hardware can be easily extended without any degradation in overall throughput as the vector size is increased and the number of lattice filter stages is increased.

Our goal is to design a set of chips that can be connected at the board level to achieve any sampling rate consistent with I/O limitations. This requires the partitioning of a single lattice filter stage into a set of chips which can be replicated to achieve any vector size. In this partitioning we attempted to minimize the number of different chips that had to be designed, allow flexibility in the vector size (that is, amount of parallelism), and minimize the data bandwidth (number of pins) on each chip.

We found a partitioning that uses five different chips and meets these requirements. Block diagrams of four of them are shown in figure 3, and a fifth chip simply implements variable-length pipeline stages. Two of the four chips (PE1 and PE4) have built-in hardware multiplexers so that the chip function can be controlled by external signals to form a reconfigurable data path; otherwise eight different chip designs would have been required. The same chip set can also be used to construct a lattice LMS joint process estimator. These chips will be replicated as required and interconnected at the board level to achieve any specified system sampling rate.

3. Asynchronous System Design

An asynchronous processor is composed of two types of basic blocks: computation blocks and interconnection blocks. Computation blocks perform processor operations, which include any combinational logic such as multipliers and ALUs, and memory elements. A computation block must be designed such that the computation is started by an external request signal and generates an output signal to indicate that the computation has completed. The completion signal can be readily generated by using a class of logic family called Differential Cascode Voltage Switch Logic (DCVSL). A comprehensive discussion of DCVSL is given in [12]. The completion signal can be viewed as a locally generated clock to indicate to the succeeding blocks when the output data is ready to be fetched and to request a data transfer.

When computation blocks are to be interconnected, asynchronous interconnection blocks must be inserted among them to ensure correct data transfers under all temporal variations of the various completion signals.

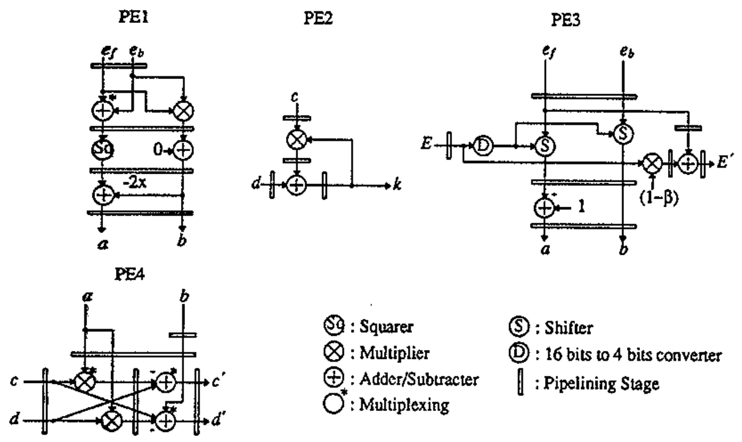

Fig. 3. A chip set designed to implement the LMS adaptive lattice filter shown in figure 1. The partition was done in such a way that any required filter sampling rate can be achieved by replicating these chips at the board level without redesign. PE1 and PE4 have built-in hardware multiplexers so that the chip function can be controlled by external signals to form a reconfigurable data path.

These interconnection blocks take the completion signals generated by computation blocks and issue request signals to start the computation in other computation blocks. It can be shown that the two binary signals, the request signal for start of computation and the completion signal for finish of computation, are necessary and sufficient for realizing general asynchronous computation with unbounded delays [18]. Interconnection blocks are composed of hazard-free asynchronous logic, which can be synthesized from a behavioral description specifying the sequence of operation that each interconnection block needs to perform [19]. We will now discuss the properties of asynchronous computation from a system point of view.

3.1. Properties of Asynchronous Computation Blocks

DCVSL computation blocks use the so-called *dual-rail* [20] coded signals inside the logic, which compute both the data signal and its complement. Contrary to past expectations, if there is no carry-chain in the design, asynchronous computation using DCVSL blocks always gives the same worst-case computation delay, independent of input data patterns, since both rising and falling transitions have to be completed before the completion signal can go high. Hence, the computational latency of each computation block for different input data patterns is approximately a constant. This lack of data dependency to the first order is actually an advantage for real time signal processing, since the worst-case computation delay can be easily determined.

Once an asynchronous processor has been designed, there is no way to slow down the internal computation. The throughput can be controlled by issuing the request signal at a certain rate, but the computation within the processor is performed at the full speed. This property has an impact on testing: unlike synchronous circuits, we cannot slow down the circuitry to make it work with a slower *clock*.

In asynchronous design it is tempting to introduce shortcuts that improve performance at the expense of speed-independence. However, these shortcuts necessitate extensive timing simulations to verify correctness under varying conditions. To minimize design complexity we therefore generally stayed true to the speed independence design, with the result that timing simulations were required only to predict performance and not to verify correctness.

3.2. Properties of Asynchronous Interconnection Blocks

When computation blocks communicate between one another, an interconnection block is designed to control the data transfer mechanism. We require that an interconnection circuit be delay-insensitive; that is, the circuit's behavior is independent of the relative gate delays among all the physical elements in the circuit. Delay-insensitive circuits demand that the Boolean functions of these circuits must not allow any hazard conditions. An automated synthesis procedure for designing hazard-free asynchronous interconnection circuits from a

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.